Dissertation

Document Verification with Temporal Description Logics

by
Franz Weitl

submitted to

Fakultät für Informatik und Mathematik
Universität Passau

19 December 2007

Supervisor:
Prof. Dr. Burkhard Freitag

Co-Supervisor:
Prof. Dr.-Ing. Günther Görz

To Misaki, Yuto, and Riku.

Abstract

The subject of this thesis is checking the content consistency of web documents along reading paths.

Web documents are typically not read linearly but by following different alternative paths through the content. Manually ensuring the consistency of content along the possible reading paths is difficult, time-consuming, and error-prone. Tool support is highly desirable.

This thesis presents a new formal framework for the representation and verification of consistency criteria concerning the reading structure and content of web documents. The content of documents is modelled by description logic knowledge bases. The narrative structure of documents is represented by a state transition system. The temporal description logic $\mathcal{ALC}$CTL is defined for the representation of semantic criteria on documents. The combination of description logics and temporal logics allows for addressing structural properties as well as semantic interrelationships within the content of the document. The proposed formalism exceeds existing approaches in expressiveness while preserving decidability, low computational complexity, and high compatibility with the knowledge representation standards of the semantic web effort.

For the verification of properties, existing methods of reasoning in description logics and model checking temporal logics are combined and enhanced. Formal properties such as the decidability and complexity of the verification problem, as well as the soundness, completeness, and runtime complexity of the proposed verification algorithms are shown.

Case studies on eLearning documents demonstrate the performance and adequacy of the proposed methods. The experimental results confirm that the developed methods scale to application relevant problem sizes and exceed existing approaches in performance, expressive power, and flexibility regarding the type of criteria and documents being handled.

Contents

List of Figures

Part I.

Foundations of Document Consistency Checking

1. Introduction

1.1. Motivation

This thesis aims at automatically checking the consistency of the structure and content of digital documents. We restrict ourselves to documents composed of different identifiable components that contain sufficient metadata about their content and structure. Such documents are frequently used in the domains of eLearning and technical documentation.

As an example, consider a web-based training (WBT) about the operation and maintenance of industrial robots (Figure 1.1). This document aims at teaching the necessary foundations and skills for operating and programming different types of industrial robots.

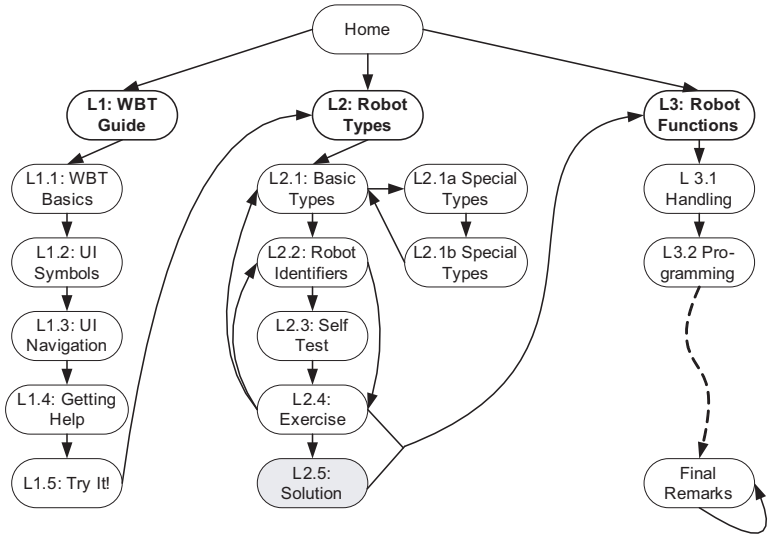

Figure 1.1.: a web-based training about robots

The document is composed of several web pages, for instance ”Home”, ”L1: WBT Guide”, ”L2: Robot Types”. The user can navigate through the document by following links between web pages, for instance from page ”Home” to page ”L1: WBT Guide”.

Web documents usually have a highly branching structure: they are not meant to be read linearly from the beginning to the end but offer many alternative reading paths depending on the information demands, pre-knowledge, and individual preferences of the reader. The document needs to make sense to reader on any of the many possible reading paths.

For instance, the following criteria should be met by the WBT depicted in Figure 1.1:

1. Each major topic of the document has to be addressed by some lesson within the document.
2. At the end of each lesson, an exercise must be solved.
3. In the sequel of each exercise task, a sample solution is provided.
4. Sample solutions are accessible to teachers only.
5. Concepts required to solve exercise tasks have been trained before.

Ensuring the consistency of the document along any of the possible reading paths by manual inspection is infeasible even for moderately large publications because the number of reading paths usually grows exponentially with the document size. For non-linearly structured documents, tool support in maintaining the consistency of the content along reading paths becomes vital.

Existing techniques for maintaining the consistency of structure and content focus on technical aspects such as absence of dangling hyperlinks or validity w.r.t. the document format. However, a proper mechanism for checking content-related consistency criteria needs to take the argumentation structure and relationships of concepts in the document's domain of discourse into account.

Content-related consistency criteria address the following aspects.

1. *Structural dependencies*: requirements on the composition structure and order of functional units of the document, e.g. as suggested by a standard for technical documentation [DIN89]. Structural dependencies follow the pattern:

   A content unit of type A occurs at location x relative to the location y of a content unit of type B.

   Example:

   Each chapter starts with an introduction stating the objectives of the chapter. After the introduction, new concepts are explained and trained. Each chapter ends with a conclusion containing a summary of major concepts.

2. *Content dependencies*: requirements on the way and order of discussing topics within the document. Content dependencies follow the pattern:

   Concepts, which are discussed in the way A at location x, are discussed in the way B at a location y related to x.

Example:

Each question within a self test refers to concepts only that have been explained previously.

3. *Domain dependencies*: compliance with an expert model of the domain of discourse. Domain dependencies follow the pattern:

   Concepts of type A are discussed in a certain way B at a location x in the document.

   Example:

   In the overview section, every important function of each robot type must be listed. The basic knowledge necessary to activate the function must be covered within the quick start section of the manual.

Structural and content dependencies are *internal dependencies* while domain dependencies are instances of *external dependencies*. Internal dependencies require a certain type of relationship between different parts of the document. In contrast, external dependencies relate parts of the document to external structures such as the document's domain of discourse. Both types of requirements are supported by the verification methods presented in chapter 6.

1.2. Summary of Goals and Requirements

The core targets of this work are 1) the design of a formal specification language adequate for representing criteria on internal and external dependencies as stated above, and 2) the development of a method for checking the specification on documents. The following general requirements should be met.

1. Format independency and general applicability: the specification and verification methods need to work independently from the kind of criteria and document being checked. Even documents being assembled from heterogeneous sources using different document formats and media types should be accounted for. The methods need to work with existing documents without needing to change them. Changes within the document format should not necessarily lead to changes of the specification or verification framework and vice versa.
2. Adequacy and focus: the specification formalism should be focussed on semantic criteria on the level of content structure and topics of the document. Specifications should be free from technical details about how these structures are represented within the document or metadata format. This should lead to compact specifications that are easy to create, change, and adapt to different application scenarios.

3. High expressiveness: the specification formalism needs to be expressive enough to allow for the representation of a wide range of relevant semantic consistency criteria. Internal and external dependencies, as stated above, should be accounted for. The specification language needs to be capable of addressing types of content, topics, and ways of discussing them. In addition, means are to be provided for defining locations relative to other locations on reading paths through the document. Further, it needs to be possible to relate document structures to external structures such as domain models.
4. Scalable precision and avoidance of over-specification: the specification formalism should enable loose specifications that allow for a wide range of possible solutions. At the same time, highly specific properties should also be expressible.
5. Compatibility to existing knowledge representation standards: as a prerequisite for verifying content-related criteria, some machine-processible information about the document's content needs to be available. Manual semantic annotation of documents is costly and prone to errors. Therefore, it should be possible to make use of existing information sources about the document's content and its domain of discourse. Metadata annotations in RDF (Resource description framework [W3C04b]), XML markup or other kind of tagging, and available ontologies in OWL (Web Ontology Language) [W3C04a] should be exploited for checking document properties.
6. Soundness and completeness: The applied verification methods should be provably sound and complete. The system should find all and only the violations of the given specification within the given document. A prerequisite of the existence of sound and complete algorithms is the decidability of the according verification problem.
7. Efficiency: since the proposed methods are targeted at large and complex publications, the performance of the implementation needs to scale up to large problem sizes. A prerequisite for efficient algorithms is the polynomial complexity of the according verification problem.

The complete assessment of the correctness of a document cannot be the ultimate goal in most application cases. This would require 1) the complete and correct formalization of the document's content and 2) a complete and correct formalization of an widely accepted expert model of the domain of discourse as a reference for assessing the correctness of the document's content. Both tasks are, except for very formal domains such as mathematics, not feasible in general. In addition, any formalization is costly.

Consequently, not the most powerful techniques for checking semantic criteria on documents are the ultimate goal but methods that offer the best compromise between power and cost of applying them in existing documentation scenarios and technical infrastructures.

1.3. Assumptions and Approach

Crucial for the applicability of any verification method is the amount and precision of information available about the content and the structure of the document. For the presented approach to be applicable, the following minimal requirements need to be fulfilled.

- The content of the document is coherent in the sense that it is possible to identify some preferable ways of how to read the document. The presented verification methods can be applied to completely incoherent data but a large part of the expressive power would remain unused.
- Some distinct units larger than sentences can be identified and referred to unambiguously within the document. Examples of suitable units are chapters, sections, paragraphs, learning units, test questions, etc.
- There is some discrete information about each document unit available or extractable by some text analysis tool [BCRS06, KT03, UCI+06]. The information could concern the type or function of the unit (e.g. definition, example, remark), the topic of the unit (as provided by keywords, index terms, etc) or media type (e.g. text, image, animation).

Structured documents formats such as SCORM [Adv04b], DocBook [WMS05], and DITA [TC05] satisfy these requirements. Further information sources are external metadata e.g. on the basis of RDF [W3C04c] and metadata extracted by intelligent information extraction tools [KT03, UCI+06].

Figure 1.2 shows the basic components of the presented framework.

We assume a document to be composed of several fragments and tagged by some metadata (MD, fig. 1.2 rhs bottom), e.g. on the basis of standards for structured documents such as SCORM [Adv04b] or DITA [TC05].

The *knowledge extraction* component (fig. 1.2 rhs bottom) collects the information required for verifying content-related criteria from several information sources and integrates it into a consolidated *semantic model* (fig. 1.2 rhs center). The semantic model serves as an abstraction from implementation details irrelevant for the verification task and provides a unified access to information from different sources.

The semantic model is related to *background knowledge* about the document represented by a *description logic (DL) knowledge base* (fig. 1.2 rhs center). Ontological background knowledge is adopted for modelling and verifying external dependencies (cf. section 1.1). In addition, ontologies help to align the vocabulary used in specifications with the vocabulary used in the representation of the document's content. This leads to simpler specifications and higher robustness against specification errors and inconsistent metadata.

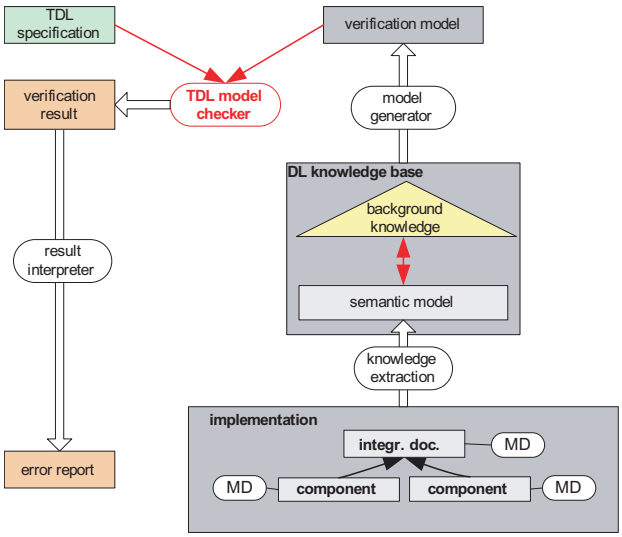

Figure 1.2.: knowledge-based verification of integrated documents

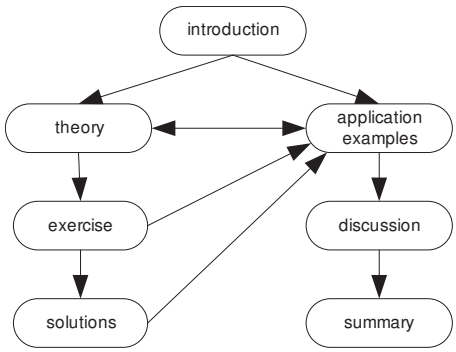

Figure 1.3.: part of the narrative structure of a web-based training

Web documents typically have some sort of coherent, possibly branching *narrative structure* that guides the user in reading the document (fig. 1.3). The narrative structure defines, which content units are sensibly read *ahead* or *after* other content units. In many cases, consistency criteria refer to sensible ”reception sequences” of content units. Hence, we consider, in accordance to [dA01, SFC98], a temporal formalism as adequate for the representation of consistency criteria.

More precisely, we propose the new temporal description logics (TDL) $\mathcal{ALC}$CTL (chapter 6) as a suitable formal basis. Temporal description logics are expressive for representing content- and structure-related criteria [WF04, WF06] while remaining decidable under certain conditions [AF01, HWZ02].

The modular structure of the specification formalism $\mathcal{ALC}$CTL allows for separating different concerns of criteria: the structure dimension and the content dimension of criteria can be treated separately, which reduces the complexity of specifications.

Finally, a *result interpreter* component (fig. 1.2 lhs center) takes the verification results (counterexamples) from the model checker and builds an error report pinpointing error locations within the document.

In summary, the major components of the proposed framework are:

- ontology- and graph-based semantic document model;
- temporal description logics as a formal basis for representing criteria;
- verification by model checking;

1.4. Contribution and Relevance of the Results

We propose a technical framework for representing and verifying semantic criteria on structured documents.

From the application-oriented perspective, the major contribution is the realization of a flexible, efficient, and precise tool to represent and verify a new type of criteria on the content and argumentation structure of documents. The presented approach exceeds existing methods for document verification in terms of expressiveness and efficiency. More complex criteria can be checked in less time than using state-of-the-art verification systems for temporal logics or validation techniques for XML documents.

From the theoretical perspective, the major contribution is the definition and analysis of the temporal description logic $\mathcal{ALC}$CTL, which is proposed as a formal basis for the representation and verification of semantic criteria. Important properties such as the decidability and complexity of the $\mathcal{ALC}$CTL model checking problem are proven. The first model checking algorithm for a temporal description logic is defined, examined, and evaluated. The superior expressive power and efficiency of $\mathcal{ALC}$CTL model checking as compared to existing model checking techniques is demonstrated in the domain of document management. The developed formalism and algorithms are motivated by but not limited to an application in document management.

The innovative aspects of the proposed framework are in detail.

- Document properties are checked based on a new semantic model of the narrative structure and covered topics of documents. A graph-based model is combined with description logics to represent both structural aspects and semantic interrelationships within the content of the document. The logic-based approach allows for

  - the abstraction from irrelevant implementation details and, consequently, for compact specifications.
  - the integration of ontological background knowledge and, consequently, high expressive power and flexibility regarding the document and metadata format.

- Description logics and temporal logics are combined to a new specification language $\mathcal{ALC}$CTL that maintains a good balance between relevant expressive power and low computational complexity. Document verification is modelled as an $\mathcal{ALC}$CTL model checking problem that is shown to be decidable and in polynomial time.
- Reasoning methods for description logics and model checking methods for temporal logics are combined to enable the verification of content-related properties along reading paths in documents. It is shown how non-temporal and temporal reasoning can be combined to achieve high flexibility, efficiency, and accuracy. The proposed algorithms are shown to be sound, complete, and optimal. In addition, the high performance of their prototypical implementation is demonstrated in a number of case studies.

1.5. Overview

This thesis consists of three parts of three chapters each. In part I, the foundations of document consistency checking are covered. Part II presents the formal aspects of the proposed approach. The document model, the specification formalism, and the verification algorithms are defined and relevant properties such as decidability, computational complexity, soundness, and correctness are shown. Part III discusses the results obtained in practical experiments.

In the sequel of this chapter, a short introduction to the application domain of this thesis – checking the consistency of documents – is given. The state of the art in document consistency checking is summarized and the focus, goals, an approach of this thesis are justified.

Chapter 3 shortly introduces the reader to description logics and temporal logics, which are the fundamental methods of the presented approach.

In chapter 4, a brief overview of the proposed framework is given.

Chapter 5 formally defines the semantic document model that represents relevant knowledge about the content and structure of the document to be verified.

Chapter 6 defines the specification formalism $\mathcal{ALC}$CTL. The model checking problem of $\mathcal{ALC}$CTL and a model checking algorithm are defined and analyzed. Based on $\mathcal{ALC}$CTL model checking and DL reasoning, an algorithm for verifying documents is

presented and studied. The results are related to existing work in the field of temporal logics.

Chapter 7 sketches the implementation of the formal framework and reports on evaluation results obtained in case studies on real and synthetic document bases. The performance, expressive power, and adequacy of the approach is demonstrated and compared with CTL model checking.

Chapter 8 compares the presented approach with existing methods for XML documents. The chapter discusses the major results of a comparative study with XSLT [W3C99b] and XPath [W3C99a].

Finally, chapter 9 summarizes the core contributions of this thesis.

2. Introduction to Checking Document Consistency

2.1. General Goals of Consistency Checking and Management

In this work, we regard *documents* as structured data objects composed of several *components* (Figure 2.1 bottom), for instance, chapters, sections, web pages, and paragraphs. Both, documents and their components, are *resources*, that are identified by a unique *ID* and are described by some *metadata* (Figure 2.1 top). In addition, components can refer to other components of the document by some referencing mechanism such as hyperlinks in HTML documents or XLink [XLi01] in XML documents (Figure 2.1 lhs bottom). Typical examples of such documents are web documents (Figure 2.1 rhs bottom), i.e. documents that are published on the world wide web.

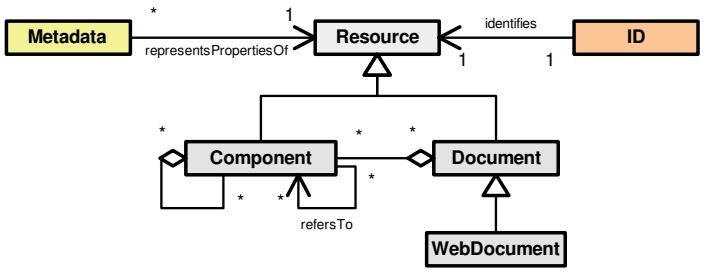

Figure 2.1.: basic document model

In document management, the consistency of a document or a collection of documents is defined as the conformance to a set of rules or constraints [NCEF02, Sch04] that are expected to be met at any or at certain times within the document's life cycle. In contrast to knowledge bases or databases, the goal is not the exclusion of inconsistencies but the detection, report, precise location, and appropriate treatment of inconsistent parts of documents in interaction with the user [Fin00]. Often, inconsistencies do not need to be corrected immediately but can be tolerated for a certain time [Sch04]. Appropriate responds to detected inconsistencies include the analysis of its root cause, the correction of the document, the correction of the consistency rules, and documenting

but (temporarily) tolerating the error. The set of measures for detection, analysis, and reaction on inconsistent structures is called *consistency management* [SZ01].

The general goals of consistency management are increasing the quality of artefacts within the development process and reducing the cost of quality assurance. Whenever documents are developed in larger teams, assembled from different sources, updated frequently, and adapted to different subjects, target groups, and contexts of use, tool support for consistency management becomes vital [KBHL+03].

2.2. Aspects of Document Consistency

The following different aspects of document consistency can be distinguished.

Completeness and Correctness

Consistency plays a major role in software development [NER00]. One aspect of software consistency is the correctness and completeness of the implementation. In [KBHL+03, TL97], the terms *correctness* and *completeness* are applied to collections of documents. A collection of documents is considered as *correct* iff its content conforms to a formal (expert) model of its domain of discourse, i.e. no document contains a wrong statement about some entity of its domain of discourse. A collection of documents is considered as *complete* iff it contains appropriate content for each of the intended contexts of use (cf. [KBHL+03]).

Automatically checking correctness and completeness of collections of documents is expensive to realize because this requires a correct and complete expert model of the domain of discourse and contexts of use.

Static and Dynamic Consistency

In databases, there is often a distinction between static consistency and dynamic consistency or integrity [CER01, ER98, KE06].

Static consistency rules express constraints on a collection of documents w.r.t. single points in time while dynamic consistency rules express constraints w.r.t. the allowed changes to a collection of documents.

An example of a static consistency rule is: all targets of references within the document are existing (referential integrity) [Sch04].

An example of a dynamic consistency rule is: the name and type of major documents should never be changed [Sch04].

Data-level and Content-level Consistency

From a system's perspective, documents are structured data objects. From a reader's perspective, documents are structured presentations of content. The consistency of documents is desirable at both the data- and content-level. Consistency at the data-level is the conformance to some document format or schema definition and aims at ensuring the correct processing of the document by excluding unexpected structures. Examples of data-level consistency are the absence of dangling references, the correct nesting of elements in XML documents and the conformance to data type definitions.

Content-level consistency is the conformance of the information represented by document data with a set of requirements. The major goal is to ensure that the content of the document meets the intentions of the authors and makes sense to the readers.

[Sch04] distinguishes the following aspects of content-level consistency:

- *referential*: the targets of (semantic) references within the document's content exist.
- *unique*: some content is unique within the document or within a collection of documents.
- *naming*: certain terms used within the document are conforming to some naming conventions.
- *calculation*: numeric constraints on properties of the document content are satisfied.
- *linguistic*: requirements regarding readability and semantic similarity of content are met.
- *logic*: these are requirements on the logical consistency of statements within the document.

We identify further important aspects of content-level consistency:

- requirements on the *topics and interrelationships of topics* that are covered by the content of the document.

  Example: "the manual needs to address all functions and related operation instructions of the documented technical system."

- requirements on the *mode of presentation* of certain topics:

  Example: "Safety critical functions need to be marked as such within the overview and covered by detailed handling instructions within the safety instructions of the manual."

- requirements on the *structure and order of presentation*:

  Examples:

  ” Safety critical functions always need to be presented ahead of other functions.”

  ”The learning document starts with an introduction presenting the objectives of the document, continues with a presentation of concepts, provides study tasks to solve towards the end of each lesson, and ends with a summary and a final test.”

- requirements regarding the *coherence of content* of the document. Coherence requirements refer to semantic interrelationships of content along reading paths through the document (cf. Definition 2.3.1).

  Examples:

  ”All functions listed in the overview of the manual need to be explained in detail later on.”

  ”New concepts need to be defined before they are used.”

  ”Each defined concept should be used somewhere later.”

  ”Information required to solve training tasks should be given before.”

  ”For each training task there should be a sample solution available but not before having tried to solve the training task”.

2.3. Focus and Goals of this Work

2.3.1. Focus

This work aims at checking the static content-level consistency of documents. Within this class of consistency criteria, we focus on *coherence criteria* on documents along standard reading paths.

Definition 2.3.1 (Coherence Criterion)

Coherence criteria are the set of properties regarding the reading structure of a document, which contain a combination of the following type of requirements:

- there is a (semantic) relation R of objects within a component of the document c_1 to objects within another component of the document $c_2 \neq c_1$ (cf. Figure 2.1).
- component c_1 should be read either before or after component c_2 on some or all suggested reading paths through the document.

□

Example 2.3.2 (Coherence Criteria)

Typical examples of coherence criteria are:

1. "Every theorem must be proven immediately."
2. "A solution to every task to solve is presented eventually."
3. "The information required to solve a task has previously been presented."
4. "Defined concepts need to be used later on and used new concepts need to be defined before."
5. "For every robot function presented in the overview of the manual, a detailed handling instruction must be provided later on."
6. "Every objective mentioned in an introduction needs to be addressed by some related content later on but before its major aspects are summarized in a conclusion."
7. "The user cannot access protected content until she/he has read and accepted an according end user license agreement."

The listed criteria have in common that they relate some property of a currently visited part of the document to properties of a previously visited or subsequent part of the document.

□

Checking the coherence of content along reading paths has not yet been addressed within the field of document checking (cf. section 2.4), although content coherence is an important prerequisite for the readability of the text structures [GG95, MT87, SSS00]. In addition, web documents are typically not structured linearly but offer many alternative reading tracks. Hence, manually checking the coherence of content along reading paths in web documents is particularly difficult, error-prone, and time-consuming.

The presented work is a valuable contribution to reducing the cost of quality assurance and increasing the quality of documents because the checking of an important class of criteria is automated, which are very hard to check manually or using existing methods (cf. chapters 7 and 8).

The presented approach is meant to complement but not to substitute existing methods. There are various important aspects of document consistency such as data-level consistency or dynamic consistency that are better handled by dedicated existing methods [ISO06, Sch04, W3X04]. These methods do not require a semantic document model and hence can be more efficient for checking simple data-level properties.

2.3.2. Goals and Requirements

For checking the coherence of documents, the document's content and reading structure need to be represented. Further, an expressive language for specifying content-related criteria as well as efficient algorithms for checking the document model against the specification need to be provided. The following sub-goals arise:

- requirements for the semantic document model:
  - the document's content structure needs to be modelled from the perspective of the reader. Not the composition structure of document data (e.g. the XML document tree) but the document's high-level "narrative structure" is relevant for checking the coherence of content. The model of the document's narrative structure should represent *sensible* paths of reading the document in standard situations. It needs to be ensured that the content along such paths is coherent and makes sense to the reader (cf. Definition 5.2.19).
  - content-related consistency criteria refer to *what* is presented and *how* it is presented. Hence, checking content-related criteria requires the representation of topics and interrelationships of topics of documents as well as the representation of content types and modes of presentation.
  - content-related criteria refer to background knowledge about the domain of discourse. For instance, checking a criterion such as "all major topics need to be addressed" requires some definition of what "major topics" are. Hence, it should be possible to represent general background knowledge and apply it to the verification of criteria.
  - for reducing application cost, the document model should be automatically generated. Manual modelling effort should only be required for necessary background knowledge not being represented within the document.
  - the document model should abstract from the implementation structure of the document and irrelevant details w.r.t. the criteria to check. This should result in simpler specifications and higher performance of the system.
  - the document model should be compatible with existing knowledge representation and metadata standards such as RDF [W3C04b] and OWL [W3C04a]. This should allow for making use of existing information sources and extraction tools [KT03, UCI+06] and thus for reducing the cost of modelling and knowledge extraction.
  - the document model should be independent of the type and data format of the document and should be applicable to any kind of document satisfying the following conditions:

    - some information about the content of the document is available, i.e. the document's content is not a black box.
    - the content of the document can be divided into discrete, uniquely identifiable units at some level of granularity, i.e. the document's content is not completely unstructured.
    - sensible orders of reading the units of the document's content can be determined, i.e. the document's content is not a completely incoherent collection of information.

  Most structured documents or collections of documents meet the conditions above.

- requirements for the specification formalism:
  - the specification language should be adequate and focussed on coherence criteria. It should allow for concise specifications that are as simply structured as possible.
  - properties of *structured* objects and relationships between objects of the document's content and its domain of discourse need to be expressible.
  - constructs for representing loose criteria on the order of properties along reading paths through the document need to be provided.
  - to avoid over-specification, different levels of abstraction from implementation and content details should be supported (scalable precision of specifications).
  - for reducing the complexity of specifications, it should be possible to refer to externally represented background knowledge.
  - a prerequisite for sound and complete verification procedures is a precise unambiguous semantics and the decidability of the specification language.
  - to ensure that the checking of specifications scales to application relevant problem sizes, the verification problem of the specification formalism should have a polynomial runtime complexity.
- requirements for the verification procedure:
  - the verification algorithms should be sound and complete w.r.t. the semantics of specifications.
  - the verification procedure should precisely locate errors within the document in the case of specification violations.
  - since automatic consistency checks are of most use in complex scenarios the verification procedure needs to be sufficiently efficient for checking large documents.

- for supporting interactive use, the overall verification procedure should have a stable and predictable runtime for different documents and specifications.
- incremental checking of documents should be supported such that a significant portion of runtime can be saved when parts of the document and parts of the specification remain unchanged.
- the overall verification procedure should be modular such that parts of the system can independently be modified and tuned towards the specific setting and requirements of an application scenario.
- the verification procedure should be robust in the sense that potential errors in the specification or in the available document metadata and background knowledge can be detected.

2.4. Existing Methods for Checking Document Consistency

In the sequel, we summarize existing work and methods for checking document consistency and outline the contribution of this work to the field of document verification.

2.4.1. Validation of XML Documents

For the process of finding errors in XML documents the term *validation* has been coined. XML validation in the original sense amounts to checking the conformance of an XML document with some document type definition (DTD) [W3C06] or schema definitions [W3X04]. Validation mainly aims at excluding unexpected data structures for ensuring that the document can be correctly processed by different systems.

Some approaches [Jel02, NCEF02] enhance the notion of validity towards content-level consistency criteria. They enforce business rules that can address complex semantic interrelationship within and between XML documents.

Validation techniques can be roughly divided into grammar and rule-based approaches [DSD04]. [Abs06] gives an overview of XML validation methods and tools.

2.4.1.1. Grammar-based Document Validation

Grammar-based schema languages use some form of tree grammar [DSD04] for defining an XML document format. The most relevant grammar-based schema languages are DTD [W3C06], XSchema [W3X04], and RelaxNG [OAS01].

Grammar-based schema languages typically define the nesting of XML elements and attributes, referential integrity constraints, and constraints regarding the data types of elements and attributes [Vli01]. Some schema languages support the definition of default values that can be seen as a basic mechanism of automatic repairs in the case of incomplete information (cf. [NEF03, Sch04]).

Grammar-based schema languages usually describe the allowed XML structures completely in a constructive manner. They adopt a *closed content model*, i.e. everything, which is not explicitly allowed by the schema definition, is considered as an error. In addition, grammar-based schema definitions are usually interpreted strictly such that invalid documents are considered technically corrupt and not acceptable to the processing system.

As a result, the constraints of grammar-based schema languages are rather general and abstract from the content of the document. A document, which is valid w.r.t. a grammar-based schema definition, does not necessarily make sense to a human reader.

"Business rules" [Vli01], which address complex semantic interrelationships within the data of the XML document, are beyond the scope of standard schema languages and require a more flexible approach.

2.4.1.2. Rule-based Document Validation

Rule-based schema languages have been proposed to complement grammar-based schema languages for complex content-related business rules [Jel02]. They typically adopt an *open content model*, i.e. every structure, which is not excluded by the formulated rules, is considered as correct.

Content-related consistency rules are not always enforced strictly but violations can often be tolerated without jeopardizing the reliability of the processing system. As a result, some rule-based approaches do not aim at excluding inconsistent structures in documents but rather at discovering and reporting them to the user [NCEF02].

Our approach compares best to rule-based approaches applying an open content model. In the sequel, we briefly introduce the most relevant and powerful methods for rule-based validation of XML documents: Schematron [ISO06] and CLiX [MN04]. A detailed comparison of the presented work with Schematron and CLiX is given in chapter 8.

Schematron

Schematron has been developed in 1999 by Rick Jelliffe [Jel02] and has just recently become an ISO standard [ISO06] as a result of the Document Schema Definition Languages (DSDL) initiative [DSD04]. Schematron is free and supported by a range of

free as well as commercial tools. Its technical simplicity and low system requirements support the easy integration of Schematron in XML processing environments.

Schematron uses XPath [W3C99a, W3C07b] and selected XSLT-constructs [W3C99b, W3C07d] for the representation of consistency rules [ISO06]. Consistency criteria can be formulated positively (*assert*) or negatively (*report*), grouped to rules and patterns, and assigned to specific stages of the document's life cycle. The separation of consistency condition and evaluation context – each represented by an XPath expression – allows for a precise localization of errors within the document.

A Schematron schema is translated into an XSLT stylesheet that is applied to the document to check using a standard XSLT processor.

CLiX

CLiX (Constraint Language in XML) has been developed in 2000 at the University College London as a rule language for the consistency checking and smart link generation service xlinkit [NCEF02]. Later, systemwire, a new spin-off of the University College London, has further developed the xlinkit system to an industrial product. In 2006, systemwire has been acquired by messageAUTOMATION. The xlinkit system and related development tools have been integrated into messageAUTOMATION's product *validator* [mes06]. The xlinkit system and its descendent validator are protected by copyrights and not available for free use.

CLiX uses a combination of predicate logic and XPath for the specification of consistency rules. The structure and expressiveness of CLiX rules do not differ much from Schematron rules. Similar to Schematron, CLiX rules specify both a consistency condition and a context for evaluating the condition. The validation environments for CLiX focus on checking the consistency of XML documents with external data, for instance, in relational databases. The validation system xlinkit [W3C01] can generate examples of a specification violation as well as examples for XML data satisfying the specification.

Schematron and CLiX are very powerful tools for expressing constraints on the XML representation of a document. In contrast, the presented approach focuses on validating criteria related to the content and "high-level" narrative structure of the document independent as perceived by the reader. The narrative structure of documents differs from the XML structure in the following aspects:

- the narrative structure of web documents is typically not linear, i.e. there may be many sensible ways of reading the document and the content along each reading track should make sense to the reader. In contrast, the XML data model assumes a linear "document" order of elements.

- the narrative structure of web documents is typically not acyclic. It may make sense to return to previously visited parts of the document, for instance, for looking up some information or for getting an overview of the document's content. In contrast, documents are represented as trees in the XML data model.

In addition, content-related criteria often refer to some general background knowledge about document structures and concepts of the domain of discourse. In the case of XML-based document validation, this background knowledge needs to be encoded into the specification, which violates the principle of separation of concerns and leads to complex, hard to maintain specifications.

As a result, criteria related to the "high-level" content and narrative structure of the document are hard to check directly on the XML-representation of the document (cf. evaluation results in chapter 8).

We propose a model-based approach to document verification that exceeds XML-based validation techniques in the following aspects:

- the proposed methods are not limited to XML documents but can be applied to any document satisfying some basic assumptions (section 2.3).
- a DL-based representation of the document's content and background knowledge makes it possible to detect logically inconsistent information and deriving implicit knowledge by ontological reasoning.
- a specification formalism based on temporal logics allows for a compact representation and efficient verification of properties related to reading paths within the narrative structure of the document.

2.4.2. Verification of Hypertext and Hypermedia

There is a body of work for the verification of hypertext and hypermedia.

A hypertext verification method based on temporal logic has been developed by Stotts, Furuta et al. [SFC98, SFR92]. A hypertext is modelled as a finite state machine (*browsing automata*). Its states represent hypertext nodes, its transitions hypertext links. Boolean state variables represent local properties of nodes such as the availability of certain buttons, menu options, and content fragments. The temporal logic HTL* (hypertext temporal logic), a syntactical extension of CTL* (computation tree logic) [Eme90], is used as a specification language for global properties of browsing histories such as reachability of certain pages from other pages and availability of certain functions on pages. In [SN02], the method has been enhanced for the verification of quasi-parallel browsing activities in framesets.

[SDM+05] extends the work of Stotts and Furuta towards the verification of dynamically generated web content. Web-applications are modelled by a specific UML profile. UML models are then automatically translated into the modelling language of the

CTL model checker SMV [McM93]. Criteria on web sites are represented in CTL and verified by SMV. Missing components of the web application are determined by model checking and visualized by a *web application graph*.

[HH06] extends the approach of Stotts and Furuta towards the verification of web applications with adaptive navigation. In such a web application, the set of reachable pages depends on the current page, previously visited pages, and the current state of the user model. Navigation models are represented by Statecharts [Har87] that are translated into the modelling language of the CTL model checker SMV [McM93]. Consistency criteria on the adaptive navigation structure of a web site are expressed as CTL formulae. The categories of verified properties include reachability of web pages and states of the user model, constraints on browsing sequences and sequences of states of the user model, and restrictions on the state of the user model for certain types of visited pages.

In the McWeb project [dA01], a restricted version of the μ-calculus [Koz83] is used to express and verify properties of frame-based web pages. The frameset and link structure of web sites is modelled as a propositional Kripke structure [CGP02b]. Properties of web pages are modelled by atomic propositions. A variant of the μ-calculus called *constructive* μ-calculus is proposed to enable the verification of web sites that are not completely known a-priori. Criteria related to the reachability of pages, maximal length of paths, and the composition structure of framesets can be verified.

Propositional temporal logics such as CTL and the related μ-calculus enable the specification of complex properties along browsing paths in hypermedia structures. However, the applied propositional formalisms cannot express semantic relationships across different objects within the modelling domain and hence are not sufficient for expressing coherence criteria (Corollary 6.5.8). We extend the approach of Stotts and Furuta in the following aspects:

- the content and narrative structure rather than its hypertext or frameset structure is modelled. This simplifies the checking of content coherence along standard reading paths through the document.
- the combination of the document model with terminological background knowledge increases the expressive power of specifications and reliability of verification results.
- temporal description logics as a specification formalism provide higher expressiveness regarding content-related and coherence criteria than propositional temporal logics.

Dong [Don00] applies the μ-calculus [Koz83] for the specification and verification of designs of hypermedia application that are composed of different design patterns of object oriented programming. Knowledge about design patterns is represented in

Prolog [Llo87]. Desired properties such as "whenever a link is followed the *showpage*-method is eventually invoked" are specified using the μ-calculus and verified against the model of the design using the model checker XMC [RRS+00].

Santos et al. propose an approach to verification of hypermedia *presentations* [SCSD98, SSC99, SSdC98]. The goal is to discover possible inconsistencies in interactive, synchronized presentations of multimedia content such as multiple access on exclusive resources (audio) and deadlock situations. The nested context model is used to model relevant aspects of an interactive multimedia presentation. The specification language RT-LOTOS (Real Time Language of Temporal Ordering Specification) [CdO95] is used to specify desired properties of the interactive presentation process.

The approaches of Dong and Santos aim at the *technical correctness* of the design or the implementation of a hypermedia application. Technical correctness, however, does not imply that the presented content makes sense to a *human* recipient.

[FFLS99] suggests Datalog extended with path expressions for modelling and verifying the structure and content of web sites. Integrity constraints such as the existence of navigation paths between certain types of pages can be expressed and checked for. Both the web site structure and integrity constraints are represented as a set of Datalog rules complemented with regular path expressions. A sound and complete verification algorithm based on the transformation into a Datalog query containment problem is presented. The given verification problem is NP-complete and hence scales poorly in the problem size. In addition, the need for extensive manual modelling increases the application cost and reduces the reliability of the approach.

In the presented approach, we minimize manual modelling effort by automatically constructing the semantic document model from document markup and external metadata. The proposed verification algorithms have a polynomial runtime complexity and scale up to very large documents.

2.4.3. Checking Content-related Consistency of Documents

"Semantically" structured document formats based on XML [Koh00, LTV03, SF02, WMS05] or LaTeX [KBHL+03, LWR01], metadata standards [Lea02], and intelligent annotation tools [HS03, UCI+06] set the ground for taking "semantic" properties such as types of content units or discussed topics into account when processing documents.

An early attempt to guarantee the soundness and completeness of user manuals for technical systems has been described in [TA96] and [TL97]. The consistency of manuals is defined as the close correspondence of the manual's content to a formal specification of the system's behaviour. In [TA96], an annotated finite state machine, which models the behaviour of a technical system, is used as a basis to automatically construct a manual consistent with the specification. In [TL97] properties of the system

and the corresponding manual are expressed in a *temporal logic of actions* and verified using Prolog.

The MMiSS-project [KBHL+03] aims at providing a web-based adaptive educational system in the domain of Safe Systems. Consistency and completeness of dynamically assembled or frequently changed learning documents are a major concern of the project. Therefore, the semantic interrelationships of learning content are represented on a fine grained level based on ontologies of structural units and concepts used in the documents [KBLL+04]. A rule-based approach for the specification of constraints on semantic interrelationships is sketched but not defined in detail. Learning content, metadata that describes the content, and ontologies that describe the logical structure of the content are represented using a proprietarily extended LaTeX format. As a result, the integration of external web resources requires a considerable re-engineering effort.

The mathematical knowledge management system *MBase* [KF01] represents mathematical knowledge in both a human-readable and a machine-processible way. Mathematical knowledge objects (symbols, definitions, assertions, proofs, and examples) are represented by the XML language OMDoc [Koh00]. The formal semantics of definitions, assertions, and proofs is defined based on an extended sorted λ-calculus. *Logic morphisms* [KF01] are introduced to enable the integration of different mathematical tools such as theorem prover and computer algebra systems. *Development graphs* are used to manage the change of formal knowledge representations and maintain their logical consistency [AHMS02, Hut00, HS01b]. MBase is specialized on maintaining the consistency of mathematical *theories* rather than the consistency of mathematical (or other) *documents* for which further aspects such as didactic considerations are a major concern [KA03].

The HCT help-checker-tool [SK04] is specialized on checking online help systems for Siemens medical systems such as magnetic resonance tomographs. The customization of medical devices results in many variants of the product. The corresponding manual must describe all and only the features of the system as delivered to the customer. To ensure this, documentations are modularized to small help packages, structured in terms of a product taxonomy, and tagged with Boolean constraints using XML. The system compiles a product online help automatically in correspondence to the specific features of a customized system and verifies by satisfiability checking if a compiled documentation is complete and without redundancy.

The approaches listed above are limited to a specific application domain. In contrast, proposed methods are applicable to structured documents of any domain as long as discrete units and coherent paths within the document can be identified and a minimum of content-related information is available.

A formal consistency management component [ESS05] based on description logics is proposed as an extension to the content management system for technical documentation Schema ST4 [Gru06]. Models of the domain of discourse and the structure of documentation projects are represented by DL TBoxes (Definition 3.1.12). These

models scaffold the development of technical documentations in a way that a close relationship between the content of the documentation and the documented artefact can be ensured within the document's life cycle. Description logics, however, are not sufficiently expressive for representing coherence criteria along reading paths through the document (cf. section 6.1).

A powerful and flexible framework for checking the consistency of collections of interrelated documents is proposed by [Sch04]. The system focuses on controlling the evolution of a document repository over time and checking the dynamic consistency of a document repository (cf. section 2.2). For instance, consistency rules can express that certain documents must not be deleted or that certain parts of document must not be modified. In contrast, our approach is restricted to checking the static consistency at a certain time and cannot compare different versions of a document.

The formal basis of the system in [Sch04] is full first order logic interpreted on a language defined in terms of the functional programming language Haskell. For preserving decidability, all variables in formulae are quantified over finite sets (cf. CLiX in section 8.2). Path-related criteria, which are relevant for ensuring the consistency of content along reading paths through the document, are hard to represent and expensive to verify using first order logics [Pil06]. Moreover, a general restriction to finite, completely known domains can be inadequate when representing the content of a document. This is because the document's domain of discourse may contain infinite or partially known structures.

The formalisms suggested in [Sch04] are complex both in terms of computation and application cost. The presented proprietary specification formalism is not trivial. For each application case, a dedicated specification language has to be implemented in terms of Haskell functions.

The proposed verification framework offers a better compromise between expressiveness, application cost, and performance for large documents. The proposed specification language combines the standard formalisms description logic and temporal logic, which are both well supported by powerful tools and a large base of pre-defined solutions to common specification problems [BBF+01, FMPR04] (so called specification patterns [DAC99, KC05]). Moreover, the comparably simple structure of the proposed specification language sets the ground for advanced methods for user guidance and support [Jak06].

3. Fundamental Methods

The methods and algorithms developed in this work base on two fundamental formalisms: description logics (DL) [BCM+03] and temporal logics (TL) [Eme90].

A decidable description logic is used for the representation of (background) knowledge about the document's content. Description logics are powerful for the representation of terminological knowledge about partially known and possibly infinite domains such as the domain of discourse of a document. The decidability of logical implication allows for deriving new implicit knowledge from the available knowledge about the document by combining schema-level background information and instance-level information about data objects of the document and objects of the document's domain of discourse.

A temporal logic is used for the representation of consistency criteria on documents. Temporal logics such as computation tree logic (CTL) [Eme90] are expressive for representing properties of possibly infinitely running processes. Propositional temporal logics have been applied to modelling, reasoning, and verifying the behaviour of systems and protocols [MP92, PM95]. They are expressive for representing loose criteria on the sequence of events in processes [BBF+01, DAC99]. Propositional temporal logics have also been applied to verifying browsing processes along the link structure of hypertext documents [dA01, HH06, SDM+05, SFC98, SFR92, SN02]. However, the expressiveness of propositional temporal logics for representing content-related criteria is limited because propositional logics cannot capture *structured* properties of document objects at a fixed state within the browsing process. Modelling semantic interrelationships between different parts of the document or its domain of discourse require some means for representing relations between (certain types of) objects.

Various combinations of description logics and temporal logics have been suggested in literature [AF01, BKW03, HWZ01, WZ00]. The DL part of temporal description logics (TDL) contributes high expressiveness for schema-level knowledge and relationships of objects at given points in time. The temporal part of a TDL contributes expressiveness for the change of properties of objects in time [AF01]. As a result, TDL are a powerful tool for modelling and reasoning about time-varying structured domains, for instance actions and plans [AF98, AF01] or temporal databases [AFM03, AFM+01, AFWZ02].

We define a new temporal description logic tailored to the representation of criteria on the structure and content of documents (chapter 6). The reception of a document is modelled as a (reading) process on structured and interrelated objects: the document

parts and their content. Temporal description logics are a powerful tool for checking properties of such processes.

In the subsequent sections, we introduce description logics and temporal logics as the basic formalisms fundamental to our approach.

3.1. Description Logics

3.1.1. Introduction

Description logics (DL) are a family of knowledge representation formalisms for defining the terminology used for specifying properties of objects of an application domain [BN03].

Applications of description logics include conceptual modelling, software engineering, configuration management, medical systems, digital libraries, web-based information systems, natural language processing, and reasoning on databases [BCM$^+$03].

DL have gained some importance as the formal basis for representing ontologies within the semantic web effort [BLHL01]. Ontologies are formal, explicit specifications of a shared conceptualization [Gru98].

DL typically share two important properties:

1. they have a precise and unambiguous logic-based semantics.
2. logical implication and satisfiability is decidable.

These properties set the ground for application-independent, sound and complete inference services for knowledge represented in a DL.

The represented knowledge is organized in a DL *knowledge base*. DL knowledge bases consist of two components: a *TBox*, which is a definition of terminology, and an *ABox* consisting of assertions on the individuals of a domain of discourse [BN03]. The terminology defined in a DL TBox comprises *concepts*, which represent classes of objects, and *roles* which represent binary relations on objects of the domain. Individuals, properties of which are defined in the ABox, represent single objects of the domain.

Example 3.1.1 (TBox, Concept, Role)

The following terminological axiom defines the concept of "web document" as a "document" that is "delivered" by some "web server".

$$WebDocument \doteq Document \sqcap \exists deliveredBy.WebServer$$

$WebDocument$ is a concept representing the class of objects that are "web documents". $Document$ is a concept representing the class of "documents". Concept

$WebServer$ represents the class of "web servers". $deliveredBy$ is a role representing the binary relation between documents and delivering systems, for instance, web servers.

$\doteq$ is an equivalence definition on two concepts. The axiom above defines the set of instances of concept $WebDocument$ as equal to the set of instances of concept $Document \sqcap \exists deliveredBy.WebServer$.

$\sqcap$ expresses the conjunction of concepts. $\exists deliveredBy.WebServer$ represents the concept the instances of which are related to some instance of concept $WebServer$ via the role $deliveredBy$.

In section 3.1.2, we will define the precise semantics of terminological axioms. □

Example 3.1.2 (ABox, Individual, Assertion)

The following ABox assertions define some instances of concepts $Document$ and $WebServer$ as well as fillers of the role $deliveredBy$.

$Document(d)$	d is an instance of concept $WebDocument$
$WebServer(s)$	s is an instance of concept $WebServer$
$deliveredBy(d, s)$	d is in the $deliveredBy$ relation with s

In section 3.1.2, we define the precise semantics of ABox assertions. □

DL allow to define an important reasoning service: classification of concepts along their *subsumption* hierarchy. A more general concept D subsumes a more specific concept C, denoted as $C \sqsubseteq D$, iff all instances of concept C are also instances of concept D. Further reasoning services such as *concept equivalence* (two or more concepts have the same set of instances), *disjoint concepts* (two or more concepts do not have common instances), and *concept satisfiability* (some concept may have instances) can be reduced to concept subsumption.

Example 3.1.3 (Subsumption)

Let KB be a DL knowledge base containing the axiom of Example 3.1.1:

$$WebDocument \doteq Document \sqcap \exists deliveredBy.WebServer$$

Then the knowledge base KB implies that concept $WebDocument$ is subsumed by concept $Document$, i.e. each instance of $WebDocument$ is an instance of $Document$. □

Further reasoning services available for DL knowledge bases are classification of individuals (i.e. determining the set of concepts which an individual is an instance-of), instance retrieval (i.e. determining the set of individuals of a concept) and (ABox) satisfiability (see section 3.1.3).

3.1.2. Syntax and Semantics

The DL family of knowledge representation formalism comprises many languages which differ considerably in expressiveness and computational complexity. Description logics differ in the available constructors for the definition of concepts and roles for terminological axioms and ABox assertions.

The minimal description logic of practical relevance is $\mathcal{ALC}$ (attribute language with complement). $\mathcal{ALC}$ offers negation, conjunction, and disjunction of concepts as well as universal and existential quantification of roles. The terminological axiom of Example 3.1.1 is in $\mathcal{ALC}$.

3.1.2.1. $\mathcal{ALC}$ Syntax

Description logics define a knowledge representation language based on *application dependent* and *application independent symbols*. The application dependent symbols consist of a set of atomic concepts representing classes of objects, a set of atomic roles representing binary relations on objects, and a set of individuals representing single objects of the application domain. Application independent symbols are the set of connectives for building formulae. While application dependent symbols can be arbitrarily chosen, the set of connectives is fixed for a given DL. The expressiveness of a DL is determined by the available constructors for specifying concepts and roles.

Definition 3.1.4 (Application Dependent Symbols)

The countable set AS denotes the set of *application dependent symbols* of a description logics.

AS is partitioned into three pairwise disjoint subsets:

- $IV := AS \backslash (AC \cup AR)$, the set of *individuals*,
- $AC := AS \backslash (IV \cup AR)$, the set of *atomic concepts*,
- $AR := AS \backslash (IV \cup AC)$, the set of *atomic roles*.

□

Example 3.1.5 (Application (In)dependent Symbols)

The application dependent symbols used in Examples 3.1.1 and 3.1.2 are:

$$\begin{aligned} AC &= \{WebDocument, Document, WebServer\} \\ AR &= \{deliveredBy\} \\ IV &= \{d, s\} \\ AS &= AC \cup AR \cup IV \\ &= \{WebDocument, Document, WebServer, deliveredBy, d, s\} \end{aligned}$$

The application independent symbols of Example 3.1.1 are

$\doteq$	(concept equivalence)
$\sqcap$	(concept conjunction or intersection)
$\exists \cdot . \cdot$	(existential quantification)

□

Definition 3.1.6 ($\mathcal{ALC}$ Concept Description)

Let $A \in AC$ be an atomic concept and $R \in AR$ an atomic role. Then the set of $\mathcal{ALC}$ concept descriptions (or simply concepts) $\mathcal{C}_{\mathcal{ALC}}$ is the minimal set of expressions that are generated by the following syntax rule:

C, D	$\longrightarrow$	A	|	(atomic concept)
		$\top$	|	(top or universal concept)
		$\bot$	|	(bottom or empty concept)
		$\neg C$	|	(negation or complement)
		$C \sqcap D$	|	(conjunction or intersection)
		$C \sqcup D$	|	(disjunction or union)
		$\forall R.C$	|	(universal quantification or value restriction)
		$\exists R.C$		(existential quantification)

□

Remark 3.1.7 ($\mathcal{ALC}$ Concept Description)

A concept description specifies a class of objects *intensionally* by describing the common properties of its instances. The semantics of $\mathcal{ALC}$ concept descriptions are defined in Definition 3.1.22.

□

Example 3.1.8 ($\mathcal{ALC}$ Concept Description)

Let the set of atomic concepts AC and the set of atomic roles AR be as follows:

$$\begin{aligned} AC &= \{Document, WebDocument, WebServer\} \\ AR &= \{delivers, deliveredBy\} \end{aligned}$$

Then the following are $\mathcal{ALC}$ concept descriptions:

$Document$	(atomic concept)
$\exists deliveredBy.WebServer$	(existential quantification)
$Document \sqcap \exists deliveredBy.WebServer$	(conjunction, ...)
$\forall delivers.(\neg Document \sqcup WebDocument)$	(universal quantification, ...)
$\neg\exists deliveres.\forall deliveredBy.WebServer$	(negation, ...)

The following are not $\mathcal{ALC}$ concepts:

$deliveredBy$	($deliveredBy$ is not an atomic concept)
$deliveredBy.WebServer$	(missing quantification)
$\exists\neg deliveredBy.WebServer$	(negation of roles not allowed in $\mathcal{ALC}$)
$\exists(deliveredBy \sqcup delivers).\top$	(union of roles not available in $\mathcal{ALC}$)
$\forall Document.WebDocument$	(quantification of concepts not allowed)

□

Remark 3.1.9 (Role Constructors)

There are description logics allowing for complex role expressions such as negation and union of rules as shown in Example 3.1.8 [CG03].

Since complex role expressions are not supported by most available reasoning systems (see section 3.1.5), we refrain from using such constructs in the course of this work. □

Concept descriptions are the basic building blocks of terminological axioms and ABox assertions.

Definition 3.1.10 (Terminological Axioms)

Let $\mathcal{C}_{\mathcal{ALC}}$ be the set of $\mathcal{ALC}$ concepts for atomic concepts AC and atomic roles AR. Then f is a *terminological axiom* iff $f = C \sqsubseteq D$ or $f = C \doteq D$ for some concepts $C, D \in \mathcal{C}_{\mathcal{ALC}}$.

$$\begin{aligned} \mathcal{TA}_{\mathcal{ALC}} \quad := \quad & \{C \sqsubseteq D \mid C, D \in \mathcal{C}_{\mathcal{ALC}}\} \cup \\ & \{C \doteq D \mid C, D \in \mathcal{C}_{\mathcal{ALC}}\} \end{aligned}$$

denotes the set of $\mathcal{ALC}$ terminological axioms. □

Remark 3.1.11 (Terminological Axioms)

In $\mathcal{ALC}$, there are just two possible types of terminological axioms: *concept implications or inclusions* ($C \sqsubseteq D$) and *concept equivalences or equalities* ($C \doteq D$).

$C \sqsubseteq D$ expresses that all instances of concept description C are also instances of concept description D.

$C \doteq D$ expresses that the set of instances of concept description C is equal to the set of instances of concept description D. The precise semantics is given in Definition 3.1.24.

Example 3.1.1 shows an instance of a terminological axiom. □

Definition 3.1.12 ($\mathcal{ALC}$ TBox)

A *TBox* is a finite, possibly empty set of terminological axioms.

Let $\mathcal{TA}_{\mathcal{ALC}}$ be the set of all $\mathcal{ALC}$ terminological axioms. Then T is an $\mathcal{ALC}$ *TBox* iff T is a finite subset of $\mathcal{TA}_{\mathcal{ALC}}$. □

Remark 3.1.13 ($\mathcal{ALC}$ TBox)

TBoxes represent schema-level knowledge about the domain of discourse. A TBox defines general relationships between concepts (or classes), which hold at any time for all possible objects of knowledge domain. Hence, a TBox can be considered as a logic-based representation of an entity-relationship diagram or an UML class diagram. It represents the static structure of the domain of discourse.

Other description logics support, in addition, the definition of role implications and equivalences and further types of axioms. The set of axioms on roles is sometimes called *RBox*. □

Definition 3.1.14 (ABox Assertion)

Let $\mathcal{C}_{\mathcal{ALC}}$ be the set of $\mathcal{ALC}$ concept descriptions, AR the set of atomic roles, and IV the set of individuals. Then for $C \in \mathcal{C}_{\mathcal{ALC}}$, $R \in AR$, and $a, b \in IV$

- $C(a)$ is a *concept assertion*.
- $R(a, b)$ is a *role assertion*.
- nothing else is a concept or role assertion.

An $\mathcal{ALC}$ *ABox assertion* is either a concept assertion or a role assertion. $\mathcal{AA}_{\mathcal{ALC}}$ denotes the set of $\mathcal{ALC}$ ABox assertions. □

Definition 3.1.15 ($\mathcal{ALC}$ ABox)

An *ABox* is a finite, possibly empty set of ABox assertions.

Let $\mathcal{AA}_{\mathcal{ALC}}$ be the set of $\mathcal{ALC}$ ABox assertions. Then A is an $\mathcal{ALC}$ ABox iff A is a finite subset of $\mathcal{AA}_{\mathcal{ALC}}$. □

Definition 3.1.16 ($\mathcal{ALC}$ Knowledge Base)

KB is an $\mathcal{ALC}$ knowledge base iff there are an $\mathcal{ALC}$ TBox T and an $\mathcal{ALC}$ ABox A such that $KB := T \cup A$. □

Example 3.1.17 ($\mathcal{ALC}$ TBox, ABox, Knowledge Base)

The following is an $\mathcal{ALC}$ knowledge base about the domain of documents and servers:

$$
\begin{aligned}
KB \quad = \{ & \\
& \top \sqsubseteq \forall deliveredBy.Server, && 1)\\
& \exists deliveredBy.\top \sqsubseteq Document, && 2)\\
& Server \sqcap Document \sqsubseteq \bot, && 3)\\
& WebDocument \doteq Document \sqcap \exists deliveredBy.WebServer && 4)\\
& WebServer(s1) && 5)\\
& deliveredBy(d1, s1) && 6)\\
& deliveredBy(d1, s2) && 7)\\
& deliveredBy(d2, s2) && 8)\\
\}&
\end{aligned}
$$

Expressions 1) to 4) are terminological axioms comprising the TBox of KB. Expressions 5) to 8) are assertions comprising the ABox of KB.

Axiom 1) is a *range* definition of role $deliveredBy$. It specifies that each object ($\top$) is only "delivered by" an instance of concept $Server$, i.e. all individuals, which appear in the range of role $deliveredBy$, are instances of $Server$. This is the case for individuals $s1$ and $s2$ in assertions 6), 7), and 8), respectively.

Axiom 2) is a *domain* definition of role $deliveredBy$. It specifies that all objects, which are delivered by something ($\exists deliveredBy.\top$), are instances of $Document$, i.e. all individuals, which appear in the domain of role $deliveredBy$, are instances of $Document$. This is the case for individuals $d1$ and $d2$ in assertions 6), 7), and 8), respectively.

Axiom 3) is a *disjointness* axiom. It specifies, that no object is both an instance of $Server$ and an instance of $Document$.

Axiom 4) is a terminological definition. It defines the concept $WebDocument$ as being equal to documents that are delivered by at least one web server. As a result, $d1$ is an instance of $WebDocument$ since $d1$ is an instance of $Document$ because of Axiom 2) and, in addition, $d1$ is delivered by some web server as a consequence of assertions 5) and 6). □

Remark 3.1.18 ($\mathcal{ALC}$ TBox, ABox, Knowledge Base)

Most DL reasoning systems support a simplified syntax for the definition of domain, range, and disjointness axioms.

$\mathcal{ALC}$ is not expressive enough for representing the function property of roles (e.g. each document is delivered by *exactly one* server) or cardinality restrictions (e.g. each document is delivered by *at least* two and *at most* three different servers). There are extensions to $\mathcal{ALC}$ that enable the representation of the function property and cardinality restrictions (see section 3.1.2.3). □

3.1.2.2. $\mathcal{ALC}$ Semantics

DL concept descriptions, terminological axioms, and ABox assertions are interpreted w.r.t. a possibly infinite set of objects called *interpretation domain* and an association of atomic concepts, atomic roles, and individuals to objects of the interpretation domain. The semantics definition of a DL assigns to each terminological axiom and ABox assertion a truth-value under a given interpretation. Hence, terminological axioms and ABox assertions can be seen as closed formulae that evaluate to *true* or *false* within a given interpretation.

Definition 3.1.19 (DL Interpretation)

Let AC be the set of atomic concepts, AR the set of atomic roles, and IV the set of individuals. A *DL interpretation* I is a pair $(\Delta^I, \cdot^I)$ where

- Δ^I is a countable set of objects called *interpretation domain*.
- $\cdot^I$ is a function assigning each atomic concept $A \in AC$ a set of objects $A^I \subseteq \Delta^I$, each atomic role $R \in AR$ a binary relation $R^I \subseteq \Delta^I \times \Delta^I$, and each individual $a \in IV$ an object $a^I \in \Delta^I$.

□

Example 3.1.20 (DL Interpretation)

Let $AC = \{Drama, Theater\}$, $AR = \{shownAt\}$, and $IV = \{ham, mac, shake, kings\}$.

Let $I = (\Delta^I, \cdot^I)$ where

$$\begin{aligned}
\Delta^I &= \{Shakespeare, Hamlet, Othello, Macbeth, \\
&\quad KingsTheater, NationalPlayhouse\} \\
Drama^I &= \{Hamlet, Othello, Macbeth\} \\
Theater^I &= \{KingsTheater, NationalPlayhouse\} \\
shownAt^I &= \{(Hamlet, KingsTheater), (Macbeth, NationalPlayhouse)\} \\
ham^I &= Hamlet \\
mac^I &= MacBeth \\
shake^I &= Shakespeare \\
kings^I &= KingsTheater
\end{aligned}$$

Then I is a (possible) DL interpretation. □

Remark 3.1.21 (Unique Name Assumption)

Many reasoning systems for description logics apply the *unique name assumption* (UNA) for the interpretation of individuals [BN03, HM01]. Under the unique name assumption, the following holds for each interpretation I:

$$\forall a, b \in IV : a^I = b^I \rightarrow a = b$$

i.e. two different individuals are always interpreted as different objects of the interpretation domain.

The interpretation of Example 3.1.23 meets the unique name assumption. If the interpretation mac^I of individual mac were changed to $mac^I = Hamlet$, the UNA would be violated. In the sequel, we always apply the UNA. □

The semantics of concept descriptions is defined by extending the interpretation function $\cdot^I$ to concept descriptions by inductively defining the extended interpretation for all concept constructors of the respective DL.

Definition 3.1.22 (Semantics of $\mathcal{ALC}$ Concept Descriptions)

Let $I = (\Delta^I, \cdot^I)$ be a DL interpretation for atomic concepts AC and atomic roles AR. Let $C, D \in \mathcal{C}_{\mathcal{ALC}}$ be $\mathcal{ALC}$ concepts. Then the extension of I to $\mathcal{ALC}$ concepts descriptions is inductively defined as:

$$\begin{aligned}
(\top)^I &:= \Delta^I \\
(\bot)^I &:= \emptyset \\
(\neg C)^I &:= \Delta^I \backslash C^I \\
(C \sqcup D)^I &:= C^I \cup D^I \\
(C \sqcap D)^I &:= C^I \cap D^I \\
(\exists R.C)^I &:= \{a \in \Delta^I \mid \exists b \in \Delta^I : (a, b) \in R^I \wedge b \in C^I\} \\
(\forall R.C)^I &:= \{a \in \Delta^I \mid \forall b \in \Delta^I : (a, b) \in R^I \rightarrow b \in C^I\}
\end{aligned}$$

□

Example 3.1.23 (Semantics of $\mathcal{ALC}$ Concept Descriptions)

Under the interpretation I of Example 3.1.20 we get:

$$\begin{aligned}
\top^I &= \Delta^I = \{Shakespeare, Hamlet, Othello, Macbeth, \\
&\quad KingsTheater, NationalPlayhouse\} \\
\bot^I &= \emptyset \\
(Theater \sqcap Drama)^I &= Theater^I \cap Drama^I = \emptyset \\
\neg(Theater \sqcup Drama)^I &= \Delta^I \backslash (Theater^I \cup Drama^I) = \{Shakespeare\} \\
(\exists shownAt.\top)^I &= \{a \in \Delta^I \mid \exists b \in \Delta^I : (a,b) \in shownAt^I \wedge b \in \top^I\} \\
&= \{Hamlet, Macbeth\} \\
(\forall shownAt.Theater)^I &= \{a \in \Delta^I \mid \forall b \in \Delta^I : (a,b) \in shownAt^I \rightarrow \\
&\quad b \in Theater^I\} = \Delta^I \\
(\forall shownAt.\neg Theater)^I &= \{a \in \Delta^I \mid \forall b \in \Delta^I : (a,b) \in shownAt^I \rightarrow \\
&\quad b \in \Delta^I \backslash Theater^I\} \\
&= \{Shakespeare, Othello, KingsTheater, \\
&\quad NationalPlayhouse\}
\end{aligned}$$

Note that $\forall shownAt.\neg Theater$ is equivalent to $\neg\exists shownAt.Theater$ and hence is interpreted as the set of objects of the interpretation domain Δ^I that are not shown at any theater. Since the set of objects, which are shown at some theater, is $\{Hamlet, Macbeth\}$ (see Example 3.1.20), we get

$$\begin{aligned}
&(\forall shownAt.\neg Theater)^I \\
= \; &(\neg\exists shownAt.Theater)^I = \Delta^I \backslash \{Hamlet, Macbeth\} \\
= \; &\{Shakespeare, Othello, KingsTheater, NationalPlayhouse\}
\end{aligned}$$

□

Based on the extended interpretation, it is possible to define when a terminological axiom or ABox assertion holds w.r.t. a given interpretation.

Definition 3.1.24 (Semantics of Axioms and Assertions)

Let $C, D \in \mathcal{C}_{\mathcal{ALC}}$ be $\mathcal{ALC}$ concept descriptions, $R \in AR$ an atomic role, and $a, b \in IV$ individuals. Let I be an interpretation extended to concept descriptions.

Then

$$\begin{array}{lcll}
I & \models & C \sqsubseteq D & \text{iff} \quad C^I \subseteq D^I \\
I & \models & C \doteq D & \text{iff} \quad C^I = D^I \\
I & \models & C(a) & \text{iff} \quad a^I \in C^I \\
I & \models & R(a,b) & \text{iff} \quad (a^I, b^I) \in R^I
\end{array}$$

For an axiom or assertion f, $I \models f$ is read as "f is true in I" or "f holds in I" or "f is valid in I" or "I satisfies f" or "I is a model of f" . □

Example 3.1.25 (Semantics of Axioms and Assertions)

Let I be the interpretation as of Example 3.1.20. Then the following holds:

$I \models Drama(ham)$	because $ham^I \in Drama^I$
$I \not\models shownAt(mac, kings)$	because $(mac^I, kings^I) \notin shownAt^I$
$I \models (\exists shownAt.\top)(mac)$	because $mac^I \in (\exists shownAt.\top)^I$ (cf. Example 3.1.23)
$I \models Drama \sqcap Theater \sqsubseteq \bot$	because $(Drama \sqcap Theater)^I \subseteq \bot^I$ (cf. Example 3.1.23)
$I \models \top \doteq \forall shownAt.Theater$	because $\top^I = (\forall shownAt.Theater)^I$ (cf. Example 3.1.23)
$I \not\models Drama \sqsubseteq \exists shownAt.\top$	because $Drama^I \not\subseteq (\exists shownAt.\top)^I$ (cf. Examples 3.1.20 and 3.1.23)

□

The $\models$ relationship between interpretations and axioms/assertions can be extended to knowledge bases in the following way.

Definition 3.1.26 (Models of DL Knowledge Bases)

Let KB be an $\mathcal{ALC}$ knowledge base and I an interpretation.

Then $I \models KB$ (read I is a model of KB or I satisfies KB) iff $I \models f$ for each axiom/assertion $f \in KB$. □

Example 3.1.27 (Models of DL Knowledge Bases)

Consider the following TBox and ABoxes:

$$\begin{aligned} T &= \{Drama \sqcap Theater \sqsubseteq \bot, \\ &\quad \top \doteq \forall shownAt.Theater\} \\ A_1 &= \{shownAt(ham, kings)\} \\ A_2 &= \{Drama(kings)\} \end{aligned}$$

Let I be the interpretation of Example 3.1.20. Then

$I \models T$	(cf. Example 3.1.25)
$I \models T \cup A_1$	(cf. Examples 3.1.25 and 3.1.20)
$I \not\models A_2$	(cf. Example 3.1.20)
$I \not\models T \cup A_1 \cup A_2$	because $I \not\models Drama(kings)$

□

Not all knowledge bases actually have a model. If a knowledge base contains logical contradictions there is no interpretation that satisfies *every* axiom and assertion of the knowledge base.

Definition 3.1.28 (Consistent Knowledge Base)

Let KB be an $\mathcal{ALC}$ knowledge base. Then KB is *consistent* (or *satisfiable*) iff it has a model $I \models KB$.

□

Example 3.1.29 (Consistent Knowledge Base)

Consider the TBox T and ABoxes A_1 and A_2 of Example 3.1.27. Then $T \cup A_1$ is consistent because the interpretation of Example 3.1.20 is a model of $T \cup A_1$ (cf. Example 3.1.27).

Also, $T \cup A_2$ is consistent because $I = (\Delta^I, \cdot^I)$ where $\Delta^I = \{KingOfBeggars\}$, $Drama^I = \{KingOfBeggars\}$, $Theater^I = \emptyset$, $shownAt^I = \emptyset$, $kings^I = KingOfBeggars$ is a model of $T \cup A_2$.

In contrast, $T \cup A_1 \cup A_2$ is not consistent. Assume, there were an interpretation $I \models T \cup A_1 \cup A_2$.

Then A_2 implies $kings^I \in Drama^I$. As a consequence of ABox A_1 and axiom $\top \doteq \forall shownAt.Theater$ in T, it holds: $kings^I \in Theater^I$.

The fact, that $kings^I \in Drama^I$ and $kings^I \in Theater^I$, violates axiom $Drama \sqcap Theater \sqsubseteq \bot$, i.e. $I \not\models Drama \sqcap Theater \sqsubseteq \bot$.

This is a contradiction to the assumption $I \models T \cup A_1 \cup A_2$ and thus $T \cup A_1 \cup A_2$ is shown to be inconsistent.

□

Remark 3.1.30 (Consistency of Knowledge Bases)

There are sound and complete algorithms for checking the consistency of knowledge bases in $\mathcal{ALC}$ and most relevant description logics (see section 3.1.4).

□

Definition 3.1.31 (Logical Implication)

Let KB be an $\mathcal{ALC}$ knowledge base and f be an $\mathcal{ALC}$ axiom or assertion. Then $KB \models f$ (read KB *logically implies* f or KB *entails* f) iff all models of KB are models of f:

$$\forall I : I \models KB \rightarrow I \models f$$

□

Example 3.1.32 (Logical Implication)

Let T be the TBox and A_1, A_2 be the ABoxes of Example 3.1.27. Then

$$\begin{aligned}
T &\models Theater \sqsubseteq \neg Drama \\
T &\not\models Drama \sqsubseteq \exists shownAt.Theater \\
T \cup A_1 &\models Theater(kings) \\
T \cup A_1 &\models \neg Drama(kings) \\
T \cup A_1 &\models (\exists shownAt.Theater)(ham) \\
T \cup A_1 &\not\models Drama(ham) \\
T \cup A_1 &\not\models \neg Drama(ham) \\
T \cup A_1 \cup A_2 &\models Drama(kings) \\
T \cup A_1 \cup A_2 &\models \neg Drama(kings) \\
T \cup A_1 \cup A_2 &\models \top \doteq \bot
\end{aligned}$$

Since knowledge base $T \cup A_1 \cup A_2$ does not have a model, it trivially implies any axiom or assertion.

□

Remark 3.1.33 (Logical Implication)

Axioms and assertions not contained in a given knowledge base KB but logically implied by KB are called *implicit knowledge* represented by KB.

Example 3.1.32 shows that inconsistent knowledge bases are not suitable for deriving implicit knowledge because they logically imply any axiom or assertion.

□

It is shown that satisfiability and logical implication are decidable for $\mathcal{ALC}$ and also most other relevant description logics [BN03, CG03, Don03]. Sound and complete algorithms for deciding logical implication exist [BN03].

Satisfiability and logical implication form the basis for standard inference services for description logics (section 3.1.3). These are powerful concepts since they are abstract from a given interpretation I of a DL knowledge base but are defined w.r.t. all possible interpretations. As a result, logical implications also hold for interpretation domains that are infinite or only partially known.

3.1.2.3. Expressive Description Logics

Expressive description logics extend $\mathcal{ALC}$ by adding various concept constructors and introducing role constructors. Expressive description logics such as $\mathcal{SHIQ}$ [HST00] and $\mathcal{SHOQ}(D)$ [HS01a, PH02] have gained relevance in the context of the semantic web because they build the formal foundations of the W3C standards for web ontology languages (OWL) [W3C04a]. Possible additional constructors for concept descriptions include [BCM+03, CG03]:

- *qualified number restrictions* on roles:

  $\exists^{\leq n}R.C$ represents the set of objects that have at most n R role fillers being instances of C:

  $(\exists^{\leq n}R.C)^I := \{a \in \Delta^I \mid |\{b \in \Delta^I \mid (a,b) \in R^I \wedge b \in C^I\}| \leq n\}$

  $\exists^{\geq n}R.C$ represents the set of objects that have at least n R role fillers being instances of C:

  $(\exists^{\geq n}R.C)^I := \{a \in \Delta^I \mid |\{b \in \Delta^I \mid (a,b) \in R^I \wedge b \in C^I\}| \geq n\}$

- *transitive roles*: let R be a role. Then R^+ is interpreted as the transitive closure of the interpretation of R, i.e. $(R^+)^I := \{(a,b) \in \Delta^I \times \Delta^I \mid \exists a_0, ..., a_n \in \Delta^I : a_0 = a \wedge a_n = b \wedge \forall i \in \{1..n\} : (a_{i-1}, a_i) \in R^I\}$.

- *inverse roles*: R^- denotes the inverse of role $R \in AR$ such that

  $R^{-I} := \{(b,a) \in \Delta^I \times \Delta^I \mid (a,b) \in R^I\}$.

- *role hierarchies*: For roles $S, R \in AR$, the axiom $S \sqsubseteq R$ expresses

  $I \models S \sqsubseteq R$ iff $S^I \subseteq R^I$ for some interpretation I.

- *nominals*: nominals enable constructing concepts by enumerating finite sets of individuals.

  $\{i_1, i_2, ..., i_n\}$, where $i_1, i_2, ..., i_n \in IV$, denotes a concept that is interpreted as $\{i_1, i_2, ..., i_n\}^I := \{i_1^I, i_2^I, ..., i_n^I\}$.

- and others (cf. [BN03, CG03]).

Some of the most relevant expressive description logics are $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ [Tob01]. $\mathcal{SHIQ}$ is $\mathcal{ALC}$ extended by transitive closure of roles, role hierarchies, inverse roles and qualified number restrictions. $\mathcal{SHIQ}$ is supported by many DL reasoning systems [HM01, Hor98, SPG+07]. $\mathcal{SHOIQ}$ adds nominals to $\mathcal{SHIQ}$ and closely resembles the expressiveness of the knowledge representation standard OWL-DL [W3C04a].

Note that there are subtle restrictions to the use of qualified number restrictions in $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$. These are put in place to preserve satisfiability [Don03].

3.1.3. Inference Services

Inference or reasoning services for description logics can be divided into reasoning services for knowledge bases with an empty ABox on the one hand and a nonempty ABox on the other. In the sequel, $\mathcal{DL}$ denotes an arbitrary decidable description logics such as $\mathcal{ALC}$, $\mathcal{SHIQ}$, or $\mathcal{SHOIQ}$.

3.1.3.1. Inference Services for TBoxes

Standard inference services for a $\mathcal{DL}$ TBox T include [BN03]:

- *satisfiability check*: a concept description $C \in \mathcal{C}_{\mathcal{DL}}$ is satisfiable w.r.t. T iff there is a model $I \models T$ such that $C^I \neq \emptyset$. Unsatisfiable concepts within a TBox usually indicate a severe modelling error that can easily lead to inconsistent knowledge bases when adding ABox assertions.

- *subsumption check* or *classification of concepts*: subsumption check is deciding if $T \models C \sqsubseteq D$ for $\mathcal{DL}$ concept descriptions $C, D \in \mathcal{C}_{\mathcal{DL}}$. If T entails $C \sqsubseteq D$ then D *subsumes* C w.r.t. T, i.e. all instances of C are also instances of D. D can be seen as a generalization of C or, conversely, C as a specialization of D.

  It can easily be shown that the subsumption relation on concepts w.r.t. a TBox is a partial order and hence defines a directed acyclic graph on concepts of a TBox. Arranging concepts of a TBox along their subsumption order is called classification of concepts. The generalization/specialization hierarchy of concepts of a TBox provides useful information on the structure of the represented domain and can be used for speeding up queries.

- *equivalence check*: equivalence check is deciding if $T \models C \doteq D$ for $\mathcal{DL}$ concepts $C, D \in \mathcal{C}_{\mathcal{DL}}$. Concepts, which describe the same class of objects, can be seen as synonyms. Synonyms may indicate unintended redundancies within the knowledge base. Equivalences are useful for simplifying concept descriptions and speeding up queries.

- *checking for disjoint concepts*: this is deciding if $T \models C \sqcap D \doteq \bot$ for $\mathcal{DL}$ concepts $C, D \in \mathcal{C}_{\mathcal{DL}}$. It can be important to prove that two different concepts cannot share any instances.

It is shown that all inference services can be reduced to either satisfiability or subsumption [BN03]. Hence, if satisfiability is decidable then all standard inference tasks are decidable.

3.1.3.2. Inference Services for TBoxes and ABoxes

When combining a TBox T with an ABox A, further reasoning services may be useful. Let $KB := T \cup A$ be a knowledge base containing both a TBox and an ABox. Then the following standard inference services for KB can be distinguished:

- *consistency check*: KB is consistent, if there is a model $I \models KB$ (cf. Definition 3.1.28). It is important to discover inconsistent knowledge bases since they logically imply any statement (cf. Example 3.1.32) and hence cannot be used for reasoning services based on logical implications as listed in the previous section 3.1.3.1. Note that $\mathcal{ALC}$ TBoxes cannot be inconsistent because the interpretation I with empty domain Δ^I is a model of any TBox. However, TBoxes can contain unsatisfiable concepts, which may lead to an inconsistent knowledge base when adding an ABox. Hence, it is important to discover unsatisfiable concepts in TBoxes (section 3.1.3.1).
- *instance check*: this is deciding whether $KB \models C(a)$ or $KB \models R(a, b)$ for a $\mathcal{DL}$ concept description $C \in \mathcal{C}_{\mathcal{DL}}$, a role $R \in \mathcal{R}_{\mathcal{DL}}$, and individuals $a, b \in IV$. $\mathcal{R}_{\mathcal{DL}}$ denotes the set of role expressions of description logic $\mathcal{DL}$. $\mathcal{R}_{\mathcal{ALC}}$ is equal to the set of atomic roles AR because $\mathcal{ALC}$ does not provide any role constructors.
- *instance retrieval*: let IV_{KB} denote the set of individuals that appear in some axiom or assertion of knowledge base KB. Then instance retrieval determines the set $\{a \in IV_{KB} \mid KB \models C(a)\}$ for a $\mathcal{DL}$ concept description $C \in \mathcal{C}_{\mathcal{DL}}$.
- *classification of individuals*: this is determining the set of concepts an individual is an instance of. Let AC_{KB} be the set of atomic concepts that appear in some axiom or assertion of knowledge base KB. Then classification of individuals is determining the set $\{A \in AC_{KB} \mid KB \models A(a)\}$ for an individual $a \in IV$. A variant of classification is *realization* [BN03, Don03]. Realization is determining the set of most specific concepts an individual $a \in IV$ is instance of. This is the set

  $\{A \in AC_{KB} \mid KB \models A(a) \wedge \forall A' \in AC_{KB} \backslash \{A\} : KB \models A' \sqsubseteq A \rightarrow KB \not\models A'(a)\}$.

  Realization is used for determining the most specific terms for characterizing properties of a certain object.
- *role filler retrieval*: let IV_{KB} denote the set of individuals that appear in knowledge base KB. Then role filler retrieval is determining the set

  $\{b \in IV_{KB} \mid KB \models R(a, b)\}$ for a $\mathcal{DL}$ role $R \in \mathcal{R}_{\mathcal{DL}}$ and an instance $a \in IV$.

The inference services *instance retrieval* and *role filler retrieval* are often used as standard query mechanisms to DL knowledge bases. All of the inference services above can be reduced to consistency checking [BCM+03].

3.1.4. Properties of Relevant Description Logics

An important property of most relevant description logics is the decidability of satisfiability of concept descriptions and the decidability of knowledge base consistency. Any other standard inference service as listed in sections 3.1.3.1 and 3.1.3.2 can be reduced to concept satisfiability or knowledge base consistency [BN03].

Decidable description logics include:

- $\mathcal{ALCN}$ which is $\mathcal{ALC}$ with unqualified number restrictions [BN03].
- $\mathcal{SHIQ}$ [Tob01]
- $\mathcal{SHOIQ}$ [Tob01]
- and many more [BCM$^+$03]

There are a number of sound and complete algorithms for checking knowledge base consistency. Most current reasoning systems implement some form of *tableau algorithm* [BN03]. Tableau calculi consist of a sound and complete set of transformation rules for exhaustively expanding the ABox of a given knowledge base to equivalent ABoxes in which logical contradictions become obvious. A new approach claimed to be more efficient for knowledge bases with large ABoxes reduces DL knowledge bases to disjunctive Datalog [HMS04a, HMS04b].

Besides the decidability, the computational complexity of DL inference services is most relevant for practical applications. Unfortunately, checking the satisfiability of quite simple description logics is already exponential in the size of the knowledge base. On the other hand, adding more expressiveness does not add much to the computational complexity of a description logic.

Some complexity results for relevant description logics are:

- checking satisfiability of $\mathcal{AL}$ concept descriptions w.r.t. a nonempty TBox is EXPTIME-hard [Don03]. $\mathcal{AL}$ is $\mathcal{ALC}$ without complement and with a restricted form of existential quantification [BN03].
- checking consistency of $\mathcal{SHIQ}$ knowledge bases (including a TBox) is EXPTIME-complete [Tob01].
- checking consistency of $\mathcal{ALCFIO}$ [Lut04] knowledge bases is NEXPTIME-hard [CG03, Tob00]. $\mathcal{ALCFIO}$ is $\mathcal{ALC}$ extended with functional roles, inverse roles, and nominals (see section 3.1.2.3).
- reasoning in $\mathcal{SHOIQ}$ knowledge bases is NEXPTIME-complete [Tob01].
- reasoning in OWL-Lite [W3C04a] is EXPTIME-complete. This is because OWL-Lite contains $\mathcal{AL}$ and is contained in $\mathcal{SHIQ}$.

- reasoning in OWL-DL [W3C04a] – a widely supported knowledge representation standard – is NEXPTIME-complete. This is because OWL-DL contains $\mathcal{ALCFIO}$ and is contained in $\mathcal{SHOIQ}$.

In summary, reasoning in most relevant description logics is at least EXPTIME-hard. Fortunately, in practical applications an exponential blow up can often be avoided. However, much care must be taken in structuring the knowledge base for avoiding an exponential blow up.

3.1.5. Reasoning Systems

There are a number of free and commercial systems offering standard inference services for expressive description logics. Most of them apply tableau calculi. In the sequel, we briefly introduce state-of-the-art reasoning systems adopted for the implementation and evaluation of this work.

- **Racer**, a DL reasoning server implemented in LISP, has been developed in 1997 and since then continuously improved at the Concordia University Montreal and Hamburg University of Technology [HM01]. An extended commercial version of Racer, called RacerPro, has become available recently [Rac07].

  Both systems support knowledge bases in $\mathcal{SHIQ}$ extended with concrete domains and constraint reasoning. The standard inference services of Racer and RacerPro base on tableau calculi with many optimizations. Racer and especially RacerPro provide a number of non-standard inference services such as rules, constraint reasoning, and conjunctive queries with negation as failure and aggregation.

  Racer and RacerPro support the DIG interface [BMC03] and other interfaces for the communication with client applications. DIG is a standard HTTP- and XML-based protocol for submitting knowledge bases and queries to a reasoning system and receiving reasoning results. RacerPro supports, in addition, the RDF-query language SPARQL [W3C07a] and different completeness levels that enable reducing the completeness of reasoning results in favour of performance.

- **Pellet** [SPG+07] is an open source reasoning system developed at the University of Maryland's Mindswap Lab. Pellet is implemented in Java and applies different tableau algorithms and optimizations. Pellet supports, at current, $\mathcal{SHOIQ}(D)$, which is $\mathcal{SHOIQ}$ extended with concrete domains (such as string, integer). In addition, Pellet handles full OWL-DL [W3C04a] and many features of OWL 1.1 [PSH07]. SPARQL [W3C07a] is supported as query language to OWL and RDF data. Clients can communicate with Pellet by DIG and other standard interfaces.

3.2. Temporal Logics

3.2.1. Introduction

Temporal logics are a special type of modal logic [Eme90] for representing time-dependent properties of domains. Standard logics express statements that hold (or do not hold) independently from time such as "a web document is a document that is delivered by a web server" (cf. Example 3.1.1).

In contrast, temporal logics provide a set of temporal operators for describing the change of truth of statements over time. An example of such a statement is "whenever a web server is off-line it will be on-line again by the next day." This statement can be represented in propositional linear temporal logic (LTL) as

$$\mathsf{G}\,(\neg WebServerOnline \rightarrow \mathsf{X}\,WebServerOnline)$$

The informal interpretation of this formula is: at any time (G) holds: if statement $\neg WebServerOnline$ holds then statement $WebServerOnline$ holds at the next (X) point in time.

G (globally) and X (next) are temporal connectives that specify the points in time at which a proposition or formula holds.

In 1977, temporal logics have been discovered as a particular useful formalism for representing properties of infinitely running concurrent processes of operating systems and network communications [Pnu77]. After the advent of efficient verification methods [BCMH92, McM92], temporal logics are routinely applied for verifying properties of hardware, software, and business processes [BBF+01, BDSV05]. Further applications include the specification of reactive and concurrent systems [MP92], the verification of hardware/software protocols [McM92], the support of manual and automatic program composition [FSMZ95, SFC+94], and, finally yet importantly, the verification of documents and hypermedia applications [HH06, SDM+05, SFC98, SFR92, SN02, TA96, TL97].

3.2.2. Syntax and Semantics of CTL

There is a variety of approaches to defining a temporal logics. [Eme90] gives a characterization of different types of temporal logics. Most relevant in the context of this work is computation tree logic (CTL).

CTL is a propositional, branching time, future tense temporal logic evaluated w.r.t. discrete points of time. In CTL, time is modelled as a structure with deterministic past and non-deterministic future, i.e. at each moment the flow of time may take one of several possible directions. CTL allows to distinguish properties, which should hold at *all* paths, from properties, which should hold at *some* path within a given structure. This

is particularly useful for representing criteria for non-linearly structured documents. As a result, CTL is frequently adopted in document verification (section 2.4.2).

CTL defines a language over a set of application dependent symbols and a set of application independent symbols. In the case of CTL, there is only one kind of application dependent symbols: the countable set of *atomic propositions* denoted as AP. Atomic propositions represent atomic statements, which can be either true or false at a given point in time. Application independent symbols comprise a fixed set of connectives as defined by the syntax of CTL [HR04].

Definition 3.2.1 (CTL **Syntax)**

The set of CTL formulae is the minimal set of expressions, which are generated by the following syntax rule for atomic propositions $a \in AP$:

p, q	$\longrightarrow$	a	(atomic formula)
		$\top$	(true)
		$\bot$	(false)
		$\neg p$	(negation)
		$p \wedge q$	(conjunction)
		$p \vee q$	(disjunction)
		$p \rightarrow q$	(implication)
		$\mathsf{AX}\, p \mid \mathsf{EX}\, p$	(all paths / some path next p)
		$\mathsf{AF}\, p \mid \mathsf{EF}\, p$	(all paths / some path future p)
		$\mathsf{AG}\, p \mid \mathsf{EG}\, p$	(all paths / some path globally p)
		$\mathsf{A}(p\ \mathsf{U}\ q) \mid \mathsf{E}(p\ \mathsf{U}\ q)$	(all paths / some path p until q)

□

Remark 3.2.2 (Extended CTL **Syntax)**

The derived connectives W (weak until) and B (before) are commonly used. They are defined as follows:

$\mathsf{A}(p\ \mathsf{W}\ q)$	:=	$\neg\mathsf{E}(\neg q\ \mathsf{U}\ (\neg q \wedge \neg p))$	(all paths p weak until q)
$\mathsf{E}(p\ \mathsf{W}\ q)$	:=	$\mathsf{E}(p\ \mathsf{U}\ q) \vee \mathsf{EG}\, p$	(some path p weak until q)
$\mathsf{A}(p\ \mathsf{B}\ q)$	:=	$\neg\mathsf{E}(\neg p\ \mathsf{U}\ q)$	(all paths p before q)
$\mathsf{E}(p\ \mathsf{B}\ q)$	:=	$\neg\mathsf{A}(\neg p\ \mathsf{U}\ q)$	(some path p before q)

□

CTL can be used for representing properties of web sites (cf. [SFC98], for instance). A web site can be modelled as a hypertext graph (P, L) where the set of nodes P represents the pages of the web site and the set of edges $L \subseteq P \times P$ represents the hyperlinks between the pages of the web site. Interesting properties of browsing paths within such a hypertext graph (P, L) can be represented by CTL as demonstrated by the subsequent example of CTL formulae.

Example 3.2.3 (CTL **Syntax)**

For modelling properties of pages of a web site, let for the set AP of atomic propositions hold

$$AP \supseteq \{home, impressum, help, public, submit, cancel, acknowledge, login, logout\}$$

Then the following are CTL formulae:

$\top$	true
$home$	atomic formula: $home$ holds (at current)
EF $impressum$	on some path $impressum$ holds eventually, i.e. the "impressum" of a web site is reachable from the current page.
EG $public$	there is some path such that in every state $public$ is true, i.e. the user can remain within the public part of a web site forever.
AG EF $home$	on all paths holds globally that there is some path on which $home$ is true eventually, i.e. at any point within a web site $home$ is reachable.
AG EX $help$	on all paths holds globally that at some next state $help$ is true, i.e. at any point $help$ is reachable within one step.
AG $(login \rightarrow$ AF $logout)$	whenever the user logs in, he/she must eventually log out.
AG $(login \rightarrow$ AG EF $logout)$	whenever the user logs in, he/she may remain logged in forever but has always the possibility to eventually log out.
AG$(submit \rightarrow$ EX $cancel$ $\wedge$ EX $acknowledge)$	whenever the user submits some data, it can be cancelled in some next step and acknowledged in some (other) step.
$\neg$E$(\neg login$ U $\neg public)$	there is no path on which the user has never logged in until a non-public part of the web site is reached.

The following are not CTL formulae because in CTL the temporal connectives X, F, G, U always need to paired with a path quantifier E or A.

E$\neg$G $public$
AFG $public$
AG$(submit \rightarrow$ X $cancel \vee$ X $acknolwedge)$

These latter formulae are expressible in an extension of CTL called CTL* [Eme90]. □

CTL formulae are interpreted on labelled state transition systems of the following form.

Definition 3.2.4 (CTL **Temporal Structure)**

A CTL *temporal structure* is a triple $M = (S, R, L)$ where

- S is a nonempty set of *states*.
- $R \subseteq S \times S$ is a *left-total* binary *transition* relation on S.
- $L : S \to \mathcal{P}(AP)$ is a *labelling* of states such that $L(s)$ is the set of atomic propositions that hold at state $s \in S$.

□

Remark 3.2.5 (CTL **Temporal Structure)**

Labelled finite state transition systems (S, R, L) are also referred to as *Kripke structures* [BBF$^+$01, CGP02c].

□

The semantics of temporal connectives is defined w.r.t. full paths within a temporal structure (S, R, L).

Definition 3.2.6 (Full Paths in Temporal Structures)

Let S be a set of states and $R \subseteq S \times S$ a left-total state transition relation.

Then an infinite sequence of states $(s_0, s_1, s_2, ...)$ is a *full path* in (S, R) iff $(s_i, s_{i+1}) \in R$ for $i \in \mathbb{N}$.

$FP_s := \{(s_0, s_1, s_2, ...) \in S^\infty \mid s_0 = s \wedge \forall i \in \mathbb{N} : (s_i, s_{i+1}) \in R\}$ denotes the set of *full paths* starting from a state $s \in S$.

□

CTL temporal structures model processes in terms of states and state transitions. Properties of states (for instance values of variables) are represented by sets of atomic propositions or Boolean variables, which are true at a given state. This way, also browsing processes in hypertext documents can be represented [dA01, HH06, SDM$^+$05, SFC98, SFR92, SN02] as shown by the following example.

Example 3.2.7 (CTL **Temporal Structure)**

Let $M = (S, R, L)$ where

$$\begin{aligned} S &= \{s_0, s_1, s_2, s_3\} \\ R &= \{(s_0, s_1), (s_1, s_2), (s_1, s_0), (s_2, s_3), (s_3, s_0)\} \\ L &= \{s_0 \mapsto \{public, home\}, s_1 \mapsto \{public, login\}, \\ &\quad\ s_2 \mapsto \{welcome, help\}, s_3 \mapsto \{logout\}\} \end{aligned}$$

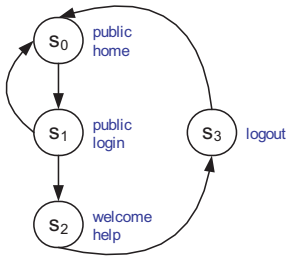

Figure 3.1.: a simple temporal structure

Figure 3.1 depicts the temporal structure M as defined above.

The temporal structure models part of a web site consisting of a public home page s_0, a login window s_1, a welcome page s_2 with help on what can be done next, and a logout window s_3.

The truth of CTL formulae such as AG EF $home$ and AG($login \rightarrow$ AF $logout$) (cf. Example 3.2.3) can be evaluated w.r.t. M. Whether the CTL structure M satisfies a CTL formula p is determined by the formal semantics of CTL connectives (Definition 3.2.8).

□

The *semantics* of CTL defines when a CTL formula p is $true$ in a structure $M = (S, R, L)$ at a state $s \in S$, in symbols: $M, s \models p$.

Definition 3.2.8 (Semantics of CTL)

Let $M = (S, R, L)$ be a temporal structure (Definition 3.2.4), and $s_0 \in S$ a state. Let

$a \in AP$ be an atomic proposition and p, q CTL formulae. $\models$ is inductively defined as

$$
\begin{array}{lcll}
M, s_0 & \models & \top & \\
M, s_0 & \not\models & \bot & \\
M, s_0 & \models & a & \text{iff} \quad a \in L(s_0) \\
M, s_0 & \models & p \wedge q & \text{iff} \quad M, s_0 \models p \text{ and } M, s_0 \models q \\
M, s_0 & \models & \neg p & \text{iff} \quad M, s_0 \not\models p \\
M, s_0 & \models & p \vee q & \text{iff} \quad M, s_0 \models \neg(\neg p \wedge \neg q) \\
M, s_0 & \models & p \rightarrow q & \text{iff} \quad M, s_0 \models \neg p \vee q \\
M, s_0 & \models & \mathsf{EX}\, p & \text{iff} \quad \exists (s_0, s_1, ...) \in FP_{s_0} : M, s_1 \models p \\
M, s_0 & \models & \mathsf{AX}\, p & \text{iff} \quad \forall (s_0, s_1, ...) \in FP_{s_0} : M, s_1 \models p \\
M, s_0 & \models & \mathsf{E}(p\ \mathsf{U}\ q) & \text{iff} \quad \exists (s_0, s_1, ...) \in FP_{s_0}\ \exists i \in \mathbb{N} : (M, s_i \models q \\
 & & & \quad \text{and } \forall j \in \{0, ..., i-1\} : M, s_j \models p) \\
M, s_0 & \models & \mathsf{A}(p\ \mathsf{U}\ q) & \text{iff} \quad \forall (s_0, s_1, ...) \in FP_{s_0}\ \exists i \in \mathbb{N} : (M, s_i \models q \\
 & & & \quad \text{and } \forall j \in \{0, ..., i-1\} : M, s_j \models p) \\
M, s_0 & \models & \mathsf{EF}\, p & \text{iff} \quad M, s_0 \models \mathsf{E}(\top\ \mathsf{U}\ p) \\
M, s_0 & \models & \mathsf{AF}\, p & \text{iff} \quad M, s_0 \models \mathsf{A}(\top\ \mathsf{U}\ p) \\
M, s_0 & \models & \mathsf{EG}\, p & \text{iff} \quad M, s_0 \models \neg\mathsf{AF}\, \neg p \\
M, s_0 & \models & \mathsf{AG}\, p & \text{iff} \quad M, s_0 \models \neg\mathsf{EF}\, \neg p
\end{array}
$$

A temporal structure $M = (S, R, L)$ together with a state $s \in S$ is a *(temporal) model* of a CTL formula p iff $M, s \models p$. □

Example 3.2.9 (Semantics of CTL)

Let M be the temporal structure of Example 3.2.7. Then

$$
\begin{array}{lcll}
M, s_0 & \models & \mathit{home} & (3.1) \\
M, s_1 & \not\models & \mathit{home} & (3.2) \\
M, s_2 & \models & \neg \mathit{public} \wedge \mathit{help} & (3.3) \\
M, s_0 & \models & \mathsf{EG}\, \mathit{public} & (3.4) \\
M, s_0 & \models & \mathsf{AG}\, \mathsf{EF}\, \mathit{home} & (3.5) \\
M, s_0 & \models & \mathsf{AG}\, \mathsf{AF}\, \mathit{home} & (3.6) \\
M, s_0 & \models & \mathsf{AG}(\mathit{login} \rightarrow \mathsf{EF}\, \mathit{logout}) & (3.7) \\
M, s_0 & \not\models & \mathsf{AG}(\mathit{login} \rightarrow \mathsf{AF}\, \mathit{logout}) & (3.8) \\
M, s_0 & \models & \mathsf{AG}(\mathit{login} \rightarrow \mathsf{AG}\, \mathsf{EF}\, \mathit{logout}) & (3.9) \\
M, s_0 & \models & \neg\mathsf{E}(\neg \mathit{login}\ \mathsf{U}\ \neg \mathit{public}) & (3.10) \\
M, s_2 & \not\models & \neg\mathsf{E}(\neg \mathit{login}\ \mathsf{U}\ \neg \mathit{public}) & (3.11)
\end{array}
$$

The first three of the equations above are a direct result of the labelling L of states.

The formula of Equation (3.4) holds in s_0 because $p = (s_0, s_1, s_0, s_1, ...)$ is a full path in (S, R) such that for each state s of p holds $public \in L(s)$.

Equation (3.5) holds because s_0 (the state where atomic proposition $home$ holds) is reachable from any of the other states s_1, s_2, and s_3.

Equation (3.6) holds because s_0 is eventually reached on any path from any state $s \in S$.

Equation (3.7) holds because state s_3, in which $logout$ holds, is reachable from state s_1, in which $login$ holds.

Equation (3.8) does not hold because an infinite path $(s_1, s_0, s_1, s_0, s_1, ...)$ starts from s_1 (in which $login$ holds) such that s_3 is never reached and hence atomic proposition $logout$ never holds. This is a counterexample to the formula $\mathsf{AG}(login \rightarrow \mathsf{AF}\ logout)$.

Equation (3.9) holds because s_3 ($logout$) is reachable from every state that is reachable from s_1 ($login$).

Equation (3.10) holds because there is no path starting from s_0, which reaches state s_2 or s_3 (in which $\neg public$ holds) without passing through state s_1 (in which $login$ holds).

Equation (3.11) does not hold because $\neg public$ holds in state s_2 and thus also $M, s_2 \models \mathsf{E}(\neg login\ \mathsf{U}\ \neg public)$ which contradicts $M, s_2 \models \neg\mathsf{E}(\neg login\ \mathsf{U}\ \neg public)$. □

3.2.3. Other Temporal Logics

If path quantifiers E and A are omitted in the syntax definition of CTL (Definition 3.2.1), we get LTL (propositional linear temporal logic). LTL is usually interpreted w.r.t. single time lines isomorphic to $(\mathbb{N}, <)$ [Eme90]. However, LTL formulae can also be interpreted w.r.t. Kripke structures (S, R, L). A LTL formula p is satisfied by a Kripke structure (S, R, L) at state $s \in S$ iff p holds for all full paths $f \in FP_s$ within (S, R) originating from s [CGP02d, HR04].

Surprisingly, the complexity of LTL model checking is higher than the complexity of CTL model checking [Eme90, Sch03] and neither CTL contains LTL nor vice versa [CGP02d, HR04]. A relevant class of criteria, which are expressible in LTL but not in CTL, are fairness properties [BBF+01].

A branching temporal logic that contains both LTL and CTL is CTL* [CGP02d]. As opposed to CTL, CTL* allows temporal connectives in formulae also without pairing them with a path quantifier E or A (see Example 3.2.3). For instance, AFG $terminated$ ("on all paths eventually a state is reached from which on the process will be $terminated$ forever") is in CTL* but not in CTL (cf. [CGP02d]).

Unfortunately, the increased expressiveness of CTL* also results in a higher computational complexity and more complex model checking algorithms as compared to

CTL [Eme90, Sch03]. As a result, CTL* is not well-supported by verification systems [Sch03].

There are also various temporal extensions to description logics [AF01]. First point-based temporal extensions of the description logics have been suggested by Schild [Sch93]. Schild's logic $\mathcal{ALCT}$ extends $\mathcal{ALC}$ by temporal connectives $\Box$ (always in the future), $\Diamond$ (eventually in the future), $\bigcirc$ (next moment), $\mathcal{U}$ (until), and U (reflexive until) in concept descriptions and terminological axioms [AF01]. A much cited example of Schild's temporal description logic is [AF01, BKW03]:

$$Mortal \doteq LivingBeing \sqcap (LivingBeing\, \mathcal{U}\, \Box\neg LivingBeing)$$

"Mortals are living beings and they are living begins until they are dead forever."

Point-based linear temporal description logics are interpreted on structures $\mathfrak{M} = \langle \mathfrak{T}, I \rangle$ where $\mathfrak{T}$ is a countable linearly ordered set representing a single flow of time and the temporal interpretation I is a function associating every point $t \in \mathfrak{T}$ with an interpretation $I(t) = \langle \Delta^{I,t}, \cdot^{I,t} \rangle$ [BKW03].

The semantics of temporal concept descriptions is then defined w.r.t. structures $\mathfrak{M}$ at a point $t \in \mathfrak{T}$. For instance, the semantics of the until operator $\mathcal{U}$ applied to concept descriptions C and D is [BKW03]:

$x \in (C\, \mathcal{U}\, D)^{I,t}$ iff there is $t' \in \mathfrak{T}$, $t' > t$ such that $x \in D^{I,t'}$ and for each $t'' \in \mathfrak{T}$, $t < t'' < t'$ holds: $x \in C^{I,t''}$.

$C^{I,t}$ denotes the (temporal) interpretation of concept C at time $t \in \mathfrak{T}$ in structure $\mathfrak{M} = \langle \mathfrak{T}, I \rangle$.

The validity of terminological axioms can be "temporalized", too. For instance, for concept descriptions C, D, E, F, the temporal axiom $(C \sqsubseteq D)\, \mathcal{U}\, (E \sqsubseteq F)$ holds in $\mathfrak{M}$ at time t, in symbols $\mathfrak{M}, t \models (C \sqsubseteq D)\, \mathcal{U}\, (E \sqsubseteq F)$, iff $E^{I,t'} \subseteq F^{I,t'}$ for some $t' \in \mathfrak{T}$, $t' > t$ and $C^{I,t''} \subseteq D^{I,t''}$ for each $t'' \in \mathfrak{T}$, $t < t'' < t'$.

Branching time extensions to description logics are described in [HWZ01, HWZ02]. A detailed comparison of existing temporal description logics with the temporal description logic $\mathcal{ALC}$CTL, which is proposed for the specification of semantic document properties, is given in section 6.5.

3.2.4. Model Checking CTL

3.2.4.1. The Model Checking Problem

Determining whether a given finite temporal structure is a model of a given CTL formula as demonstrated in Example 3.2.9 is called *model checking*.

CTL model checking is feasible for temporal structures with more than 10^{20} states [BCMH92] by using a compact symbolic representation of the temporal structure M

[CGP02c, McM92]. CTL model checking has become a widely adopted method in hardware/software verification as soon as efficient algorithms for model checking CTL have been available. In contrast, standard applications of description logics are based on satisfiability or logical entailment (cf. section 3.1.3).

Definition 3.2.10 (CTL **Model Checking Problem**)

Let $M = (S, R, L)$ be a CTL temporal structure such that S is nonempty and finite. Let $p \in$ CTL be a finite CTL formula. Then the CTL *model checking problem* is to determine the set $\{s \in S \mid M, s \models p\}$ [CGP02e]. □

Remark 3.2.11 (CTL **Model Checking Problem**)

Sometimes, the CTL model checking problem is restricted to a chosen nonempty set of *initial* states $S_0 \subseteq S$ (cf. [HR04]). Then a temporal structure (S, R, L) satisfies a CTL formula p iff $S_0 \subseteq \{s \in S \mid (S, R, L), s \models p\}$. □

Theorem 3.2.12 (CTL **Model Checking Complexity**)

The CTL model checking problem for a CTL formula p and a finite CTL structure (S, R, L) can be solved in $\mathcal{O}(|p| \cdot (|S| + |R|))$ time [CGP02e, HR04] where $|p|$ denotes the number of sub-expressions in formula p. □

Remark 3.2.13 (CTL **Model Checking Complexity**)

CTL model checking scales linearly in the sizes of the temporal structure M and formula p and hence can be efficiently solved even for large formulae and structures.

However, in many hard-/software verification problems the size of the temporal structure grows exponentially in the size of the "high level description" of a process/algorithm. This is referred to as *state explosion problem* in literature [BBF$^+$01]. When representing programs in terms of a CTL temporal structure, a state represents a possible assignment of variables during the execution of a program and the set of states represents the possible combinations of variable assignments that can occur during the program's execution. As a result, every new Boolean variable may double the number of states of the program in execution.

The problem becomes even worse in the case of loosely coupled concurrent processes [McM92]. Suppose, a system runs n instances of a process P each of which can be in one of $|S|$ different states. If the state transitions of each process are independent of the state transitions of the other processes, the total number of states of the system is $|S|^n$ (cf. [BBF$^+$01]). In such a scenario, the explicit representation of the temporal structure as a graph of states and transitions becomes quickly intractable. □

An important feature of CTL model checking is the availability of finite counterexamples if a formula $p \in \mathsf{CTL}$ is violated in a verified temporal structure M at state s [CGP02c].

Counterexamples are finite violating paths within the state transition system, also called *execution trace* [CCJ+07]. An execution trace is a finite prefix of a full path within the model (S, R, L) from a starting state $s \in S$.

Definition 3.2.14 (Execution Trace)

Let $S \neq \emptyset$ be a finite set of states, $R \subseteq S \times S$ a left-total state transition relation, and $s \in S$ a state. Let FP_s be the set of full paths in (S, R) starting from s (Definition 3.2.6).

The set of *execution traces* in (S, R) starting from s, denoted as $ET_{S,R,s}$, is the set of finite prefixes of some full path $(s_i)_{i\in\mathbb{N}} \in FP_s$:

$$ET_{S,R,s} := \{(s'_0, ..., s'_n) \in S^{n+1} \mid \exists (s_i)_{i\in\mathbb{N}} \in FP_s \; \forall i \in \{0..n\} : s_i = s'_i\}$$

□

Remark 3.2.15 (Execution Trace)

It is shown that a violation of a CTL formula p in a finite temporal structure M can always be demonstrated by a *finite* prefix of a violating full path [CGP02c].

□

Example 3.2.16 (Counterexample)

Let $M = (S, R, L)$ be the temporal structure of Example 3.2.7. Consider the formula $\mathsf{AG}(login \rightarrow \mathsf{AF}\ logout)$ that does not hold in M at state s_0 (cf. Equation (3.8) of Example 3.2.9):

$$M, s_0 \not\models \mathsf{AG}(login \rightarrow \mathsf{AF}\ logout)$$

Then $(s_0, s_1, s_0) \in ET_{S,R,s_0}$ is a counterexample to

$$M, s_0 \models \mathsf{AG}(login \rightarrow \mathsf{AF}\ logout)$$

This is because (s_0, s_1, s_0) demonstrates that there must be a full path $(s'_i)_{i\in\mathbb{N}} = (s_0, s_1, s_0, s_1, s_0, s_1, ...) \in FP_{s_0}$ such that $login$ holds at state $s'_1 = s_1$ in $(s'_i)_{i\in\mathbb{N}}$ but $logout$ does not hold at any state in $(s'_i)_{i\in\mathbb{N}}$ (cf. Example 3.2.9).

□

3.2.4.2. Model Checking Algorithms

Early model checking algorithms for CTL [CE81, CES86, Eme81] are based on an explicit representation of the temporal structure (S, R, L) being checked against a set of CTL formulae [CGP02e]. These algorithms are of theoretical relevance because they provide an upper bound of runtime complexity of CTL model checking (Theorem 3.2.12). The state explosion problem 3.2.13 excludes an explicit representation of the state transition system for many relevant use cases and problem sizes.

There are various strategies for avoiding an exponential explosion of the representation of a temporal structure (S, R, L).

The most prominent class of algorithms saves space by constructing a compact symbolic representation of the temporal structure using binary decision diagrams (BDD) [BBF$^+$01, CGP02c, McM92]. Symbolic model checking has been applied successfully to models of more than 10^{20} states [BCMH92]. However, also a BDD-based representation of temporal structure may grow exponentially in size of the system description under certain conditions [BCC$^+$03] and thus the state explosion problem is not completely resolved.

A recently proposed alternative to symbolic model checking is *bounded model checking* [BCC$^+$03] (also called *SAT-based model checking* [BCCZ99]). Bounded model checking transforms the CTL model checking problem into a satisfiability problem for propositional logic (SAT) [BCC$^+$03] and applies advanced methods for checking satisfiability of propositional formulae. The term *bounded* stems from the general strategy of searching for counterexamples the length of which does not exceed some chosen constant $k \in \mathbb{N}$. If no such counterexample can be found, k is iteratively increased until a pre-known upper bound for the length of counterexamples is reached [BCC$^+$03].

A recent variant of bounded model checking bases on a transformation of the CTL model checking problem into the consistency problem of a knowledge base in the decidable description logics $\mathcal{ALCI}$ [BDTW06, BDTW07]. This allows for solving the CTL model checking problem by checking concept satisfiability w.r.t. a TBox using an off-the-shelf DL reasoner. Although DL reasoning is exponential and thus more expensive than CTL model checking, a higher performance can be achieved in certain cases because the state explosion problem is avoided [BDTW07].

3.2.4.3. Model Checking Systems

The first system that implements symbolic CTL model checking, is SMV (symbolic model verifier) [McM93]. SMV has been developed in 1992 at the Carnegie Mellon University and maintained until 1998. SMV supports CTL and additionally allows the specification of certain fairness constraints that are not in CTL.

Since 1998, SMV has been continued under the name NuSMV (new symbolic model verifier) as a joint project of the Center for Scientific and Technological Research (Trento, Italy), the Carnegie Mellon University, the University of Genova, and the University of Trento [CCGR00]. NuSMV is implemented in C and is open source.

Version 2 of NuSMV [CCG+02] supports SAT-based model checking in addition to symbolic model checking algorithms [CGP+02a]. Also, NuSMV2 adds support of LTL (linear temporal logics) and other temporal formalism [CCJ+07].

In SMV and NuSMV, Kripke structures are defined indirectly in terms of a basic programming language-like modelling language allowing for the definition of modules (encapsulated program units with interface), variables of various types (Boolean, integer, enumerations, sets, arrays), and expressions for a (non-)deterministic change of variable values. In the case of specification violations, NuSMV constructs a counterexample in the form of an execution trace that demonstrates the violation of some formula of the specification (Example 3.2.16).

Part II.

Formal Framework

4. General Overview on the Proposed Solution

In this chapter, we give a brief informal outline and motivation of our approach to consistency checking.

4.1. Basic Principles and Goals

The general approach is characterized as model-based as illustrated by Figure 4.1.

The implementation of a document is not checked directly against a specification but an abstract model of the document's content is constructed and verified.

Information about the document is available either within the implementation of the document (e.g. an XML-file) or in the form of external metadata (e.g. RDF-Annotations). Relevant information is automatically extracted and structured as a semantic document model.

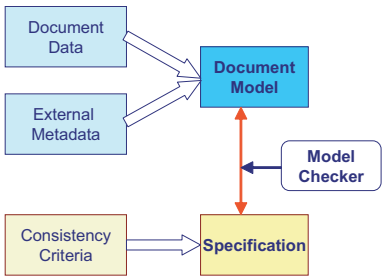

Figure 4.1.: model-based verification

Consistency criteria are often part of the tacit knowledge communicated among authors and reviewers. First, consistency criteria need to be made explicit as part of the authoring and reviewing guidelines, for instance. Consistency criteria are usually expressed in natural language. For automated checking they need to be formalized using a specification language with a precise unambiguous semantics. This is done by some quality assurance expert.

Finally, the document model is checked against the formal specification by some model checking technique.

The advantages of a model-based approach are:

- decoupling of the specification and verification of content-related criteria from implementation-related aspects (separation of concerns): The document model abstracts from irrelevant implementation details and models the content and structure of the document from the user's perspective. Criteria are formalized w.r.t. the document model and are independent of the way the content of the document is implemented. This leads 1) to compact specifications, 2) to applicability of the proposed methods to a wide range of document types and formats, and 3) to less maintenance overhead when either the implementation of the document or the criteria are changed.
- performance: since the document model can be restricted to relevant information the verification of the criteria can be sped up.
- support of evolving documents: it is desirable to discover possible inconsistencies as early as possible within the development process of a document. A useful document model can be constructed from a skeleton or first draft of the structure and content of a document. As a result, inconsistencies can be discovered and corrected prior to the full implementation of a document (cf. [FFLS99]). This results in lower development cost and higher quality of the final product.
- integration of various information sources. The document data is just one possible source of information for constructing a document model. In addition, external sources such as RDF annotations or metadata returned by information extraction methods can be exploited for constructing a rich semantic model about the document's content. As a result, a higher flexibility and precision in checking criteria can be achieved.
- integration of background knowledge: the document model can be combined with ontological background knowledge as shown below. Ontologies help to consolidate the terminology which is particularly relevant in the case of integrating information from heterogeneous sources. Furthermore, the integration of background knowledge increases the expressiveness of the specification formalism.

4.2. An Introductory Example

In the sequel, we illustrate the overall approach using a simple scenario. Consider a manual about the operation and maintenance of industrial robots published as a

web-based training (WBT). This interactive web document aims at teaching the necessary foundations and skills for operating and programming different types of industrial robots. Section 7.4 covers technical details about the document.

Figure 4.2 depicts a simplified part of the basic *narrative structure* of the sample WBT.

The narrative structure of a document determines recommended but not necessarily enforced ways of reading the document in standard situations. It guides the user through the content along coherent *narrative paths*. Complex web documents such as manuals, text books, or web-based trainings usually offer alternative narrative paths for different target groups or information demands. Hence, the narrative structure of a document is represented by a directed graph of *content units* (vertices) and *narrative relations* (edges). Since it has to be guaranteed that the document makes sense along narrative paths, the narrative structure is an appropriate basis for determining the content-related consistency of documents.

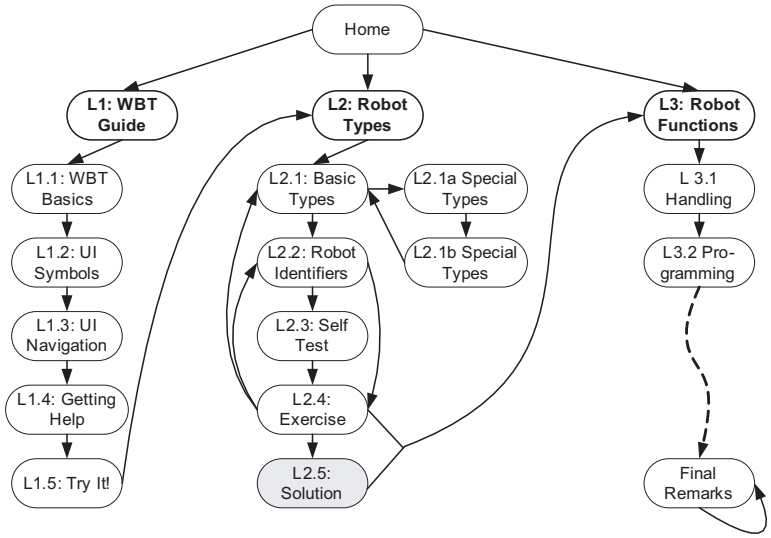

Figure 4.2.: sample narrative structure of a document

The narrative structure depicted in Figure 4.2 contains the following content units: The home page (”Home”) of the WBT lists the learning objectives and covered concepts. After ”Home”, the user can proceed with a general introduction to web-based trainings (lesson L1) or moving on immediately to lesson L2 about ”robot types” or lesson L3 about ”robot functions” (Figure 4.2 top). Each lesson contains information units introducing and explaining new concepts (e.g. L2.1 and L2.2 in Figure 4.2), self test units (L2.3 in Figure 4.2), and exercises (L2.4 in Figure 4.2) which are followed by solutions (L2.5 in Figure 4.2) in the trainers variant of the WBT. For increased convenience, exercises provide references to units containing information related to

the exercise tasks. A lesson may contain side tracks (L2.1a and L2.1b in Figure 4.2) covering topics beyond the immediate learning objectives of the WBT.

Let us assume that in our scenario the narrative structure is extracted automatically by analyzing the XML markup of the document (see section 7.4).

To make sure that the document has the desired consistent structure, the following criteria should be met.

1. Each major topic of the document has to be addressed by some lesson within the document.
2. At the end of each lesson, an exercise must be solved.
3. In the sequel of each exercise task, a sample solution is provided.
4. Sample solutions are accessible to teachers only.
5. Concepts required to solve exercise tasks have been trained before.

These properties can be formalized in the proposed specification formalism $\mathcal{ALC}\mathsf{CTL}$ that is a combination of the description logic $\mathcal{ALC}$ (section 3.1.2) and the temporal logic CTL (section 3.2.2).

Recall that $\mathcal{ALC}$ formulae are axioms $C \sqsubseteq D$ where C and D are concepts composed by using the constructors: $\neg$ (complement), $\sqcap$ (intersection), $\sqcup$ (union), $\exists R.C$ (existential quantification) and $\forall R.C$ (universal quantification) (cf. Definition 3.1.6).

In $\mathcal{ALC}\mathsf{CTL}$, the concept constructors of $\mathcal{ALC}$ are complemented by the temporal connectives of CTL that are E (on some path), A (on all paths), X (next moment in time), F (eventually), G (globally), U (until), W (weak until), and B (before) (cf. Definition 3.2.1 and Remark 3.2.2).

The syntax and semantics of $\mathcal{ALC}\mathsf{CTL}$ is defined in section 6.2. For now, we informally illustrate the expressiveness of $\mathcal{ALC}\mathsf{CTL}$ for documents using the sample criteria 1. to 5. given above. These criteria can be represented in $\mathcal{ALC}\mathsf{CTL}$ as follows:

1. a) $MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.Lesson$

   For each major topic holds ($majorTopic \sqsubseteq$): on all paths eventually (AF) a content unit is reached such that the topic is addressed by a lesson ($\exists addressedBy.Lesson$). In short: every major topic is addressed by a lesson on *all* paths through the document. An alternative weaker formalization is:

   b) $MajorTopic \sqsubseteq \mathsf{EF}\ \exists addressedBy.Lesson$

   every major topic is addressed by a lesson on *some* path through the document.

2. a) $\mathsf{AG}(Lesson \sqsubseteq \mathsf{A}(\exists contains.Exercise \sqcap \mathsf{EX}\ LessonEnd\ \mathsf{B}\ LessonEnd))$

   Anywhere within the document holds (AG): a lesson is an object ($Lesson \sqsubseteq$) such that on all paths ($\mathsf{A}(...)$) holds: before the end of the lesson is reached (...B $LessonEnd$) there must be a content unit which contains an exercise ($\exists contains.Exerise$) and in some next step the end of the lesson is reached ($\sqcap$ EX $LessonEnd$). Based on this formalization an exercise *must* be taken immediately before the end of the lesson. A weaker formalization is:

   b) $\mathsf{AG}(Lesson \sqsubseteq \mathsf{E}(\exists contains.Exercise \sqcap \mathsf{EX}\ LessonEnd\ \mathsf{B}\ LessonEnd))$

   There is some path such that an exercise occurs immediately before the end of the lesson, i.e. an exercise *can* be taken in each lesson but the exercise may be skipped on some other path.

3. $\mathsf{AG}(Exercise \sqsubseteq \forall hasTask.\mathsf{EX}\ \exists addressedBy.Solution)$

   Anywhere within the document holds (AG): an exercise is an object ($Exercise \sqsubseteq$) such that each exercise task ($\forall hasTask.$) is – in some next content unit (EX) – addressed by some solution ($\exists addressedBy.Solution$). The solution is optional, i.e. the reader may skip the solution to an exercise task.

4. $\mathsf{AG}(Solution \sqsubseteq \forall accessibleTo.Teacher)$

   Anywhere within the document holds (AG): every solution ($Solution \sqsubseteq$) is accessible to teachers only ($\forall accessibleTo.Teacher$).

5. $Concept \sqsubseteq \neg\mathsf{E}(\neg\exists topicOf.Training\ \mathsf{U}\ \exists topicOf.Exercise)$

   For any concept holds ($Concept \sqsubseteq$): there is no such path ($\neg\mathsf{E}(...)$) that the concept is never topic of a training object ($\neg\exists topicOf.Training$) until (U) it is topic of some exercise ($\exists topicOf.Exercise$).

$\mathcal{ALC}$CTL offers temporal connectives of CTL in combination with $\mathcal{ALC}$ concept descriptions and terminological axioms. This enables the expression of semantic interrelationships of parts of the document along different reading paths as demonstrated by specifications 3 and 5.

For checking content-related specifications, knowledge about the document's content needs to be represented. This is done by means of DL knowledge bases:

As an example let us assume that lesson "L2: Robot Types" in Figure 4.2 addresses the robots of type $R180$. This can be represented by the following ABox assertions

$$\begin{aligned} AB_{L2} \;=\; \{&Lesson(L2),\\ &RobotType(R180),\\ &addresses(L2, R180)\} \end{aligned}$$

As part of the background knowledge about the document's domain of discourse it is known that robot types are major topics. Moreover, role $addressedBy$ may be defined as inverse of role $addresses$. This can be represented by the terminological axioms

$$\begin{aligned} TB &= \{RobotType \sqsubseteq MajorTopic \\ &\quad addresses^{-} \doteq addressedBy\} \end{aligned}$$

By combining knowledge bases AB_{L2} and TB, new implicit knowledge can be derived. For instance, $AB_{L2} \cup TB$ imply that $R180$ is a major topic which is addressed by a lesson:

$$\begin{aligned} AB_{L2} \cup TB &\models MajorTopic(R180) \\ AB_{L2} \cup TB &\models (\exists addressedBy.Lesson)(R180) \\ AB_{L2} \cup TB &\models (MajorTopic \sqcap \exists addressedBy.Lesson)(R180) \end{aligned}$$

Let us assume that $R180$ is the only major topic of the document and $R180$ is just addressed in unit L2: Robot Types (Figure 4.2).

Then specification 1b) $MajorTopic \sqsubseteq \mathsf{EF}\ \exists addressedBy.Lesson$ is satisfied within the narrative structure of Figure 4.2 because there is a path from starting unit "Home" to unit L2: Robot Types in which $R180$ is addressed. In contrast, specification 1a) $MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.Lesson$ is not satisfied since there is a path from "Home" immediately proceeding with unit L3: Robot Functions and never reaching unit L2: Robot Types. As a consequence, $R180$ is not addressed on *all* paths starting from "Home".

Such proofs of the validity of specifications w.r.t. a given document are automated by transforming the narrative graph and the DL-based knowledge representation of the document's content into a finite $\mathcal{ALC}$CTL temporal structure and applying $\mathcal{ALC}$CTL model checking.

4.3. Basic Components of the Framework

Figure 4.3 depicts the basic components and their interaction with user roles in our framework.

We distinguish three different *user roles*: a) the author, b) the quality assurance expert, and c) the knowledge engineer.

The framework consists of four basic **components**: 1) the knowledge extractor, 2) the model generator, 3) the TDL model checker, and 4) the result interpreter.

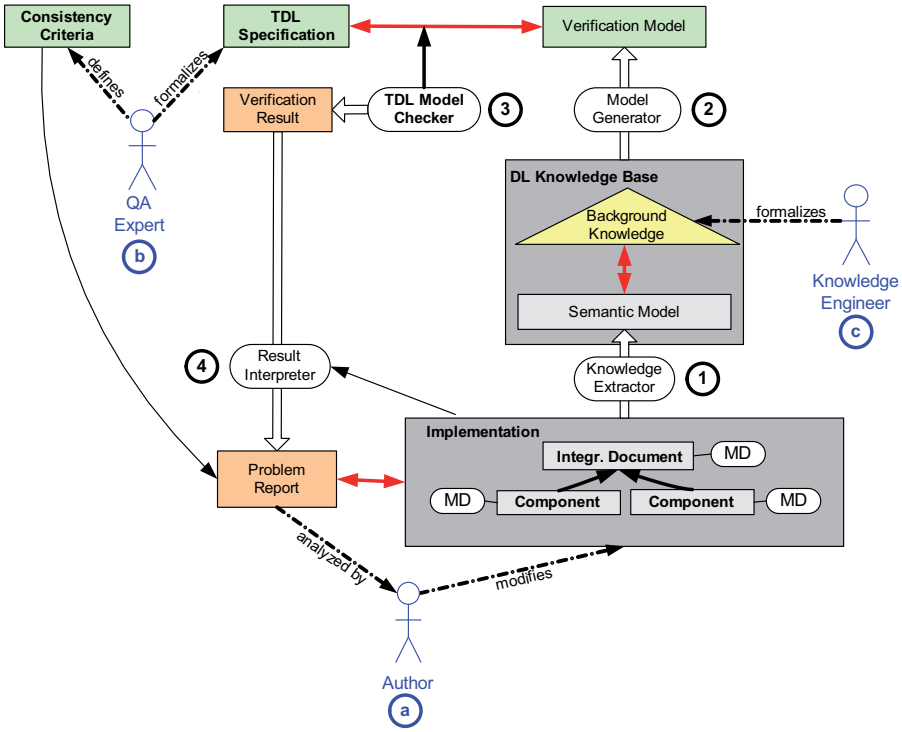

Figure 4.3.: basic components of the framework

The users interact with the **system components** in the following *roles*.

One or more *authors (a)* implement a document using some authoring tool. The document may consist of one or more distributed components, each of them possibly implemented in a different document format. In addition, external metadata (MD) about the document may be provided by the author directly or by applying some text analysis tool.

Prior to the implementation of the document, a *quality assurance team (b)* defines, in accordance with the *authoring team (a)*, some authoring guidelines part of which are a set of consistency criteria being met by the result. The *quality assurance experts (b)* formalize the consistency criteria to a specification in the temporal description logic (TDL) which is proposed as the formal specification language of the system.

In a pre-processing step, the **knowledge extraction component (1)** extracts relevant information from the document and external information sources. After the information has been assembled, filtered, and aggregated, an integrated semantic model about the narrative structure and content of the document is constructed.

Within the general framework, the "knowledge extractor" is an abstract component that needs to be instantiated for each application scenario. In the course of this work, the issue of knowledge extraction is not discussed. The feasibility of suitable knowledge extraction components is demonstrated in various case studies in chapters 6 and 8. The technical details of these components are described in [Rad05b] and [Lü06] but have been omitted in this thesis for brevity.

The semantic model represents the narrative structure of the document as a graph and the content of the document as a set of description logic ABoxes.

The semantic model may be backed up by some ontologies representing general background knowledge about document structures and domain of discourse. The background knowledge is formalized by a *knowledge engineer (c)* using some ontology editing tool for OWL (Web Ontology Language [W3C04a]).

The temporal formulae of a specification cannot be checked against the DL knowledge base directly but a suitable temporal verification model, which bridges the gap between non-temporal DL-based knowledge representation and temporal specifications, needs to be constructed. This is done by the **model generator component (2)**. The **model generator (2)** constructs a temporal verification model from the DL knowledge representation by using inference services of a DL reasoning system.

In the next step the verification model is checked against the specification: the **TDL model checker component (3)** verifies if the verification model is a model of a set of TDL formulae.

The **TDL model checker** constructs detailed verification results for every violated formula. The **result interpreter component (4)** filters and structures the verification results and outputs a comprehensive problem report.

Finally, the *author (a)* analyzes the problem report. If the problem report indicates relevant errors, the author modifies the implementation of the document.

The central methods of the presented approach are:

- representation of information about the document and it's background using description logics.
- representation of specifications using temporal description logics.
- verification of TDL specifications by model checking.

5. Document Model

5.1. Introduction

This chapter presents a formal meta-model for the representation information about a document's content and structure by means of a *semantic document model*. This meta-model bases on the notion of documents as described in section 2.1 and depicted in the UML model of Figure 2.1.

The document model is responsible for a large part of the flexibility and power of the approach. It decouples verification methods from the document format in a double sense: it is an abstraction of document structures on the documents side and an abstraction from different verification models and methods on the specification side. Furthermore, it sets the ground for a precise and scalable verification of semantic properties by relating document metadata/specifications to ontologies.

Overview

Figure 5.1 gives an overview of the representation of knowledge about a document.

Assume that there is a hypermedia document d that refers to the external resources "fragment e" and "fragment f" (Figure 5.1 bottom). Although these external resources may be presented to the reader as part of document d, they still may adhere to different document and metadata formats as well as a different "style" of metadata tagging.

Relevant information is extracted by knowledge extractor components tailored to a specific document source (Figure 5.1 center). As already sketched in section 4.3, the knowledge extractors build up a semantic model of the document consisting of two parts: 1) a graph of content units (nodes) and narrative relations (edges) representing the narrative structure of the document (Figure 4.2), and 2) a set of DL ABoxes each representing extracted facts such as discussed topics or type and function of the content (Figure 5.1 center).

The facts represented as ABoxes correspond to general *background knowledge* consisting of one or more ontologies represented by terminological axioms (TBox) (Figure 5.1 top). The *structure* of the document is described in terms of a *structure ontology* which represents a general content model for a certain type of documents (e.g. teachware [SFB99]). The topics of the document are described in terms of a *domain ontology* representing entities and relationships across entities in the domain of discourse

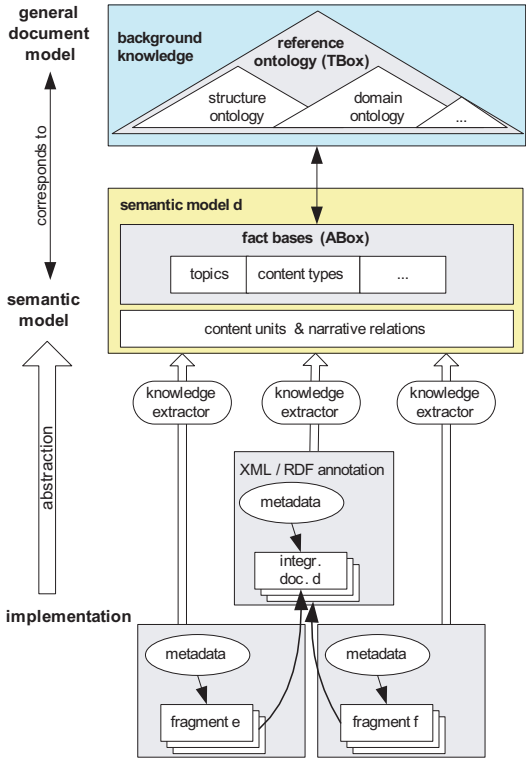

Figure 5.1.: document model overview

[KBHL+03, NP01]. Depending on the type of document other ontologies may be used to represent models of further aspects which might be relevant for determining the document's consistency such as learning objectives, prerequisites, and processes [AM03, IM04, KYNM04].

In the following example, we illustrate how ontologies help to integrate information from different sources and to derive implicit knowledge.

Example 5.1.1 (Knowledge Representation)

Assume, a document d in context C_d contains a paragraph $defBTree$ defining the term "BayerTree" and imports a fragment e from context C_e containing an example of "BayerTree" $exaBTree$. From the XML syntax or external metadata, certain facts about the document are derived and represented in the *semantic model* using DL as-

sertions:

$$Paragraph(defBTree) \tag{5.1}$$
$$defines(defBTree, bayertree) \tag{5.2}$$
$$Example(exaBTree) \tag{5.3}$$
$$hasTopic(exaBTree, b{-}tree) \tag{5.4}$$

The assertions (5.1) and (5.3) describe the fragments $defBTree$ and $exaBTree$ in terms of the *structure ontology*. The assertions (5.2) and (5.4) describe the fragments in terms of the *domain ontology*.

We assume that the *domain ontology* for the document contains the assertions $BTree(bayertree)$ and $BTree(b{-}tree)$ representing that both terms $bayertree$ and $b{-}tree$ are instances of concept $BTree$.

In the *structure ontology*, further knowledge about the concepts $Paragraph$ and $Example$, and the roles $defines$ and $hasTopic$ of Equations (5.1) to (5.4) is represented.

Definitions are paragraphs that define some topic:

$$Definition \doteq Paragraph \sqcap \exists defines.Topic$$

Examples are paragraphs that exemplify some topic:

$$Example \doteq Paragraph \sqcap \exists exemplifies.Topic$$

$defines$ and $exemplifies$ are sub-roles of $hasTopic$:

$$\begin{aligned} defines &\sqsubseteq hasTopic \\ exemplifies &\sqsubseteq hasTopic \end{aligned}$$

The *domain ontology* represents further information about topics and relationships of topics:

Every B-tree is a tree and an index structure:

$$BTree \sqsubseteq Tree \sqcap IndexStructure$$

Index structures are used in databases:

$$IndexStructure \sqsubseteq \exists usedIn.DataBases$$

Let KB be a knowledge base containing the facts of the semantic model, the structure ontology, and domain ontology as sketched above. We now can reason in a uniform

way about the two fragments and infer, for instance, that fragment $defBTree$ is a $Definition$ and that both fragments deal with something used in databases:

$$\begin{aligned} KB &\models (Definition \sqcap \exists hasTopic.BTree)(defBTree) \\ KB &\models (\exists hasTopic.\ \exists usedIn.DataBases)(defBTree) \\ KB &\models (\exists hasTopic.\ \exists usedIn.DataBases)(exaBTree) \end{aligned}$$

Knowledge derived by use of logical implications from the knowledge base about a document is used for verifying semantic criteria.

□

5.2. Modelling the Narrative Structure of Documents

The document model as introduced in the course of this section is suitable for documents that have a formal internal structure such that a) distinct relatively self-contained parts can be identified, b) the parts of the document are in a narrative relationship in the sense that there are certain preferred sequences of reading them, and c) discrete machine-processable information about the type and topics of document parts is available. Such kind of documents appear frequently in eLearning and technical documentation and are typically implemented in XML.

Documents such as novels or other kind of literature, which do not have a formal structure or formal properties, are not in the focus of this work.

5.2.1. Content Units

We assume that the content of a document can be represented in a relational way, i.e. content objects are in certain relations with each other or with other structures. In this chapter, we will introduce the concept of *content units*, which represent coherent chunks of the documents content, and the concept of the *narrative relation*, which enables to determine recommended reading paths through the document (cf. section 4.2). Content units and narrative paths will play a major role in expressing and verifying semantic properties of the document's content.

Definition 5.2.1 (Content Unit)

The nonempty, finite set CU_d of *content units* denotes the set of cohesive, self-contained parts of the content of a document d.

□

Remark 5.2.2 (Content Unit)

Definition 5.2.1 of content units is rather informal. In fact, we do not pose formal requirements on the choice of content units.

However, not an arbitrary choice of content units is sensible in a specific application. An appropriate selection of CU_d is an important modelling task when applying the proposed framework. When modelling content units the following non-technical "soft" criteria should be considered.

- Content units should be aligned with the content structure of the document. The content of most documents is structured in terms of larger chunks such as parts, modules or chapters, which in turn are recursively divided into sub-chunks such as sections, lessons, learning units, pages, paragraphs, tables, etc. Content units should correspond to one of the "natural" modularization levels of the document such as sections, pages, or paragraphs.
- Content units should be chosen such that narrative relations to other content units exist (see also Definition 5.2.4 and subsequent remarks). The narrative relation for a content unit U defines the set of content units which can be sensibly read immediately after reading U. Hence, it is probably not sensible to choose content units to be overlapping, i.e. (part of) the content of one content unit should not be already covered by the content of another content unit.
- The set of content units CU_d should cover the content that is meant to be read in standard situations by some target group. A document may contain some sort of support content, which can be accessed on demand but reading is not required by default. Examples of support content are indices, glossaries, references, help pages, and additional explanations and materials.

  Support content is not represented by content units but by means of the content knowledge base (section 5.3.1). Hence, CU_d typically does not comprise the entire content of the document.

□

Example 5.2.3 (Content Unit)

Consider a technical manual d about a video recorder. The technical manual comprises the following parts:

U_1 preface

U_2 important safety instructions

U_3 first steps

U_4 table of contents

U_5 front and rear panel

U_6 remote control

U_7 maintenance

U_8 glossary of technical terms

U_9 specifications

Then a sensible set of content units is

$$CU_d = \{U_1, U_2, U_3, U_5, U_6, U_7\}$$

U_4 (table of contents), U_8 (glossary of technical terms), and U_9 (specifications) contain support content that is not meant to be read in standard situations but serves as a reference for looking up specific information. Hence U_4, U_8, and U_9 are not included in CU_d in this example. However, other situations are imaginable that require to include support content. □

5.2.2. Narrative Relationships

As mentioned in Remark 5.2.2, we assume that content units are in a *narrative relationship* to each other.

Definition 5.2.4 (Narrative Relation)

A pair of content units $(U_1, U_2) \in CU_d \times CU_d$ are in *narrative relationship* iff it is sensible in standard situations to proceed with content unit U_2 immediately after having read U_1.

$proceed \subseteq CU_d \times CU_d$ denotes the *narrative relation* on content units of document d.

$proceed(U) = \{U' \in CU_d \mid (U, U') \in proceed\}$ denotes the *successors* of content unit $U \in CU_d$.

$proceed^{-1}(U) = \{U' \in CU_d \mid (U', U) \in proceed\}$ denotes the *predecessors* of content unit $U \in CU_d$. □

Remark 5.2.5 (Narrative Relation)

Within the scope of this thesis we consider the narrative relation $proceed$ as given. The way, the narrative relation is generated by the knowledge extractor component of the system, depends on the document format, the available metadata, and the target criteria to be verified,

In many cases, the narrative relation is closely related to the structure of the document's implementation. In the case of XML documents, the narrative relation among

content units is probably closely related to the document order of elements within the XML data model. In addition, the presence of hyperlinks (e.g. on the basis of XLink [W3C01]) or other kinds of references can give rise to a narrative relation among the respective content units.

In document standards for eLearning there are several ways of how narrative relations among content units can be represented (see section 7.4). By interpreting these structures it is possible to derive the $proceed$ relation among content units without manual modelling effort.

Although the narrative relation $proceed$ introduces a notion of "before" and "after" among content units it is neither a strict nor partial order in general. More precisely, $proceed$ is in general neither reflexive nor irreflexive nor antisymmetric nor asymmetric nor transitive.

$proceed$ is typically not reflexive because it is not necessarily sensible to read every content unit twice. Also, $proceed$ might not be irreflexive because it can be sensible to read a content unit twice. As an example consider a test unit. Here it might be sensible to repeat the test immediately after a failing attempt.

As a counterexample for transitivity consider three content units U_1, U_2, U_3 such that after U_1 is sensible to proceed with U_2 and after U_2 it is sensible to proceed with U_3. Moreover, U_3 relies on a concept introduced in U_2 such that it is not sensible to proceed with U_3 immediately after having read U_1. Hence, $(U_1, U_2) \in proceed$ and $(U_2, U_3) \in proceed$ but $(U_1, U_3) \notin proceed$.

Note further that $(U_1, U_2) \in proceed$ does not necessarily mean that having read U_1 is *required* for reading U_2, i.e. that the content of U_2 *depends on* the content of U_1. As an example consider two different units U_1 and U_2 which should both be read but which can be read in arbitrary order, i.e. U_2 does neither depend on having read U_2 nor vice versa. This can be modelled by $(U_1, U_2) \in proceed$ **and** $(U_2, U_1) \in proceed$. Thus $proceed$ is neither asymmetric nor antisymmetric in general.

As a consequence, $proceed$ does not represent an order on parts of the document as a whole such as, for instance, the document-order in the case of XML. "Before-after"-relationships between content units are always interpreted relative to a given path through the document, i.e. it depends on the choice of reading the document which unit comes before or after another unit. We consider this notion of order - although a bit more complex and not intuitive at first - more flexible and appropriate especially in the case of web documents, where a global order among resources would not be an adequate representation of sensible ways of browsing the content.

□

The set of content units CU_d and the binary relation $proceed$ define a graph on the content of a document.

Definition 5.2.6 (Narrative Graph)

Let CU_d be the set of content units of document d and $proceed \subseteq CU_d \times CU_d$ the respective narrative relation.

Then $NG_d := (CU_d, proceed)$ denotes the *narrative graph* of document d. □

Remark 5.2.7 (Narrative Graph)

The narrative graph represents sensible flows of reading the document. Note that especially web documents are neither read nor meant to be read in a linear way. In addition, it may be sensible to visit parts of the document repeatedly for example after failing to answer a test question in learning documents or looking up some previously read concept. Hence, the narrative graph of a document is not assumed to be acyclic in general. □

Within the narrative graph, we distinguish a unit which is a sensible starting point for reading the document.

Definition 5.2.8 (Beginning of Document)

We assume that within the set of content units there is a distinct uniquely defined content unit $BOD_d \in CU_d$ representing the beginning of the document such that it is sensible to start reading the document at unit BOD_d when accessing the document for the first time. □

Definition 5.2.9 (End Unit)

Let $(CU_d, proceed)$ be the narrative graph of document d. Then

$EU_d := \{U \in CU_d \mid \forall U' \in proceed(U) : U' = U\}$

denotes the set of *end units* of document d. □

Remark 5.2.10 (End Unit)

An end unit is a content unit without a successor other than itself. As a consequence, when reaching unit U it is not sensible to read any other content unit of document d and hence the reading process can be considered as terminated.

Modelling end units as reflexive nodes w.r.t. the $proceed$ relation allows to assume $proceed$ to be left-total, i.e. any content unit is assumed to have some successor unit. The left-totality of $proceed$ simplifies the application of temporal logics as a specification formalism. □

5.2.3. Narrative Paths

Of special interest for expressing and verifying semantic criteria are paths in the narrative graph, which represent sensible flows of reading the document in standard situations.

Definition 5.2.11 (Narrative Path)

Let CU_d be the set of content units, $proceed$ the narrative relation on content units, and BOD_d the beginning of document d.

A finite sequence $(U_0, U_1, ..., U_n) \in CU_d^{n+1}$ for $n \in \mathbb{N}$ is a *finite narrative path* iff $U_0 = BOD_d$ and $(U_{i-1}, U_i) \in proceed$ for each $i \in \{1..n\}$.

NP_d denotes the *set of finite narrative paths* of document d.

$NP_{U,d} := \{(U_0, U_1, ..., U_n) \in NP_d \mid U_n = U\}$ denotes the set of finite narrative paths from BOD_d to content unit $U \in CU_d$.

An infinite sequence $(U_0, U_1, ...) \in CU_d^{\infty}$ is an *infinite narrative path* iff $U_0 = BOD_d$ and $(U_{i-1}, U_i) \in proceed$ for each $i \in \mathbb{N}_1$.

A finite narrative path $fp = (U_0, U_1, ..., U_n)$ is *maximal* iff fp cannot be continued, i.e. $\neg\exists U \in CU_d : (U_n, U) \in proceed$.

A narrative path is a *full (narrative) path* iff it is maximal or infinite.

FP_d denotes the *set of full narrative paths* of document d.

FP_d^{∞} denotes the *set of infinite narrative paths* of document d. □

Remark 5.2.12 (Narrative Path)

Since the narrative graph of a document can be cyclic, narrative paths may be infinite. By allowing infinite paths, the number of times a certain narrative unit can be visited is not limited in principal by the document model.

Of special interest for the specification and verification of semantic criteria are full narrative paths within the document, i.e. paths which either cannot be continued or return to a previously visited unit. The set of full paths represent all sensible ways of reading the document. Any other narrative path is contained as a sub-path in some full path.

By definition, full paths can be both finite or infinite. As a simplification of the technical framework we assume *all* full paths to be infinite in the sequel. This is not a limitation because any finite full path $(U_0, ..., U_n) \in FP_d$ can be extended to an infinite full path by one of the following constructions.

Let $(CU_d, proceed)$ be such that $FP_d \backslash FP_d^{\infty} \neq \emptyset$ and let $FEU_d := \{U \in CU_d \mid \exists (U_1, ..., U_n) \in FP_d : U = U_n\}$ be the set of end units of *finite* full narrative paths.

1. We define $proceed' := proceed \cup \{(U, U) \in FEU_d \times FEU_d\}$ and $CU'_d := CU_d$.
2. We introduce a new "virtual" content unit $EOD_d \notin CU_d$, which represents the end of the document and define $CU'_d := CU_d \cup \{EOD_d\}$ and $proceed' := proceed \cup FEU_d \times \{EOD_d\} \cup \{(EOD_d, EOD_d)\}$.

In both cases, all full narrative paths are infinite within the modified narrative graph $(CU'_d, proceed')$. It depends on the document and criteria to check, which of the two options above is preferable. In most documents, natural end units can be identified. In these cases, the first option is preferable because the introduction of an artificial content unit can be avoided. The second option, i.e. introducing a distinct end-of-document unit EOD_d, has the advantage that it may be easier to address criteria to narrative paths only which actually come to the end of the document such as:

"On all narrative paths of the document any central topic must be covered before the end of the document is reached."

In the criterion above, the property of "covering all central topics" does *not* need to hold on narrative paths which do *not* reach the end of the document because they are "caught" in a local cycle. □

These considerations justify the following simplifying assumption.

Assumption 5.2.13 (Infinite Full Narrative Paths)

We assume the narrative graph $(CU_d, proceed)$ of a document d to be such that $FP_d = FP_d^\infty$, i.e. any full path in $(CU_d, proceed)$ is infinite. □

As a second simplifying assumption we postulate for the narrative graph of a document $(CU_d, proceed)$ that any content unit CU_d can be reached on some narrative path in document d, i.e. there are no content units that are unreachable from a starting unit of a document. We call this property of the narrative graph *coherence*:

Assumption 5.2.14 (Narrative Graph is Coherent)

We assume the narrative graph $(CU_d, proceed)$ of a document d to be *coherent* in the sense that $\forall U \in CU_d : NP_{U,d} \neq \emptyset$. □

Remark 5.2.15 (Narrative Graph is Coherent)

The assumption of *coherence* (5.2.14) is not a limitation in practical applications and can easily be met.

A narrative graph $(CU_d, proceed)$ being not coherent can be considered as a modelling flaw because in such narrative graphs there are content units that cannot be reached on

any flow of reading from a starting point of the document. These content fragments are then irrelevant w.r.t. standard flows of reading the document and therefore should not be represented within the set of content units CU_d (cf. Remark 5.2.2). If such content fragments are relevant for checking criteria they can still be represented within the knowledge bases of the semantic model (see section 5.3.1).

An alternative way of handling a document without a coherent narrative graph is splitting the document up into sub-documents having a coherent narrative graph. Then each of the sub-documents is represented and checked independently. □

As a consequence of Assumptions 5.2.13 and 5.2.14, $proceed$ is left-total:

Proposition 5.2.16 ($proceed$ is Left-Total)

Let $(CU_d, proceed)$ be a coherent narrative graph. Then any full narrative path is infinite iff $proceed$ is left-total.

Proof:

"⇒":

Let $(CU_d, proceed)$ be a coherent narrative graph such that any full narrative path is infinite.

We show that $proceed$ is left-total: $\forall U \in CU_d \exists U' \in CU_d : (U, U') \in proceed$.

Let $U \in CU_d$. Then, because $(CU_d, proceed)$ is coherent, there is $(U_0, U_1, ..., U_n) \in NP_{U,d}$ such that $U = U_n$. Assume, there is no $U' \in CU_d$ such that $(U_n, U') \in proceed$. Then $(U_0, U_1, ..., U_n)$ is maximal and hence a finite full path (Definition 5.2.11) which is a contradiction. Thus the assumption fails and there is $U' \in CU_d$ such that $(U_n, U') \in proceed$ and, since $U = U_n$, also $(U, U') \in proceed$.

"⇐":

Let $(CU_d, proceed)$ be a coherent narrative graph such that $proceed$ is left-total.

We show any full narrative path to be infinite: Let $fp \in FP_d$ be a full narrative path. Assume that fp were finite, i.e. $fp = (U_0, ..., U_n)$ and $\neg \exists U \in CU_d$ such that $(U_n, U) \in proceed$. Then this is a contradiction to $proceed$ being left-total. Hence, our assumption must have been wrong and fp is infinite. □

Proposition 5.2.17 (The Narrative Graph is Cyclic)

Let $(CU_d, proceed)$ be a coherent narrative graph such that $proceed$ is left-total. Then $(CU_d, proceed)$ contains a cycle.

Proof: Assume, $(CU_d, proceed)$ is acyclic. Let $(U_i)_{i \in \mathbb{N}} \in FP_d$ be an infinite full narrative path. Such a path exists since $proceed$ is left-total (Proposition 5.2.16). Then,

because $(CU_d, proceed)$ is acyclic, it holds for $j, i \in \mathbb{N} : i \neq j \Rightarrow U_i \neq U_j$. Hence, there is an injection $f : \mathbb{N} \rightarrow CU_d : f(i) = U_i$. This is a contradiction to CU_d being finite (Definition 5.2.1) and thus the assumption of $(CU_d, proceed)$ being acyclic must have been wrong. □

Remark 5.2.18 ($proceed$ is not a Strict Order)

Proposition 5.2.17 implies that $proceed$ is not a strict order. Further, it is unlikely but possible for $proceed$ to be an order relation. This requires $(U, U) \in proceed$ for all $U \in CU_d$ and the narrative graph not to contain cycles other than (U, U) (antisymmetry). □

5.2.4. Narrative Structure

We call a coherent narrative graph $(CU_d, proceed)$ with a distinct starting unit BOD_d and a left-total relation $proceed$ the *narrative structure* of a document. Since the narrative structure guides the reader through the document's content, it is a major reference structure for expressing and verifying content-related criteria.

Definition 5.2.19 (Narrative Structure)

Let $(CU_d, proceed)$ be the coherent narrative graph of document d such that $proceed$ is left-total. Let $BOD_d \in CU_d$ be the beginning of the document d.

Then $NS_d := (CU_d, BOD_d, proceed)$ denotes the *narrative structure* of document d. □

Corollary 5.2.20 (Full-Paths in Narrative Structures)

Let $(CU_d, BOD_d, proceed)$ be a narrative structure. Then, because $BOD_d \in CU_d$ and $proceed$ is left-total, it holds for the set of full paths that $FP_d \neq \emptyset$.

Since $(CU_d, proceed)$ is coherent and $proceed$ is left-total it holds by Proposition 5.2.16 that every full path $f \in FP_d$ is infinite. □

Example 5.2.21 (Narrative Structure)

The narrative structure $NS_d = (CU_d, BOD_d, proceed)$ depicted in Figure 5.2 is:

$$\begin{aligned} CU_d &= \{startOfStory, introduction, basicDefinitions, theorem, \\ &\quad proof, applicationExamples, conclusion, endOfStory\} \end{aligned}$$

$$BOD_d = startOfStory$$

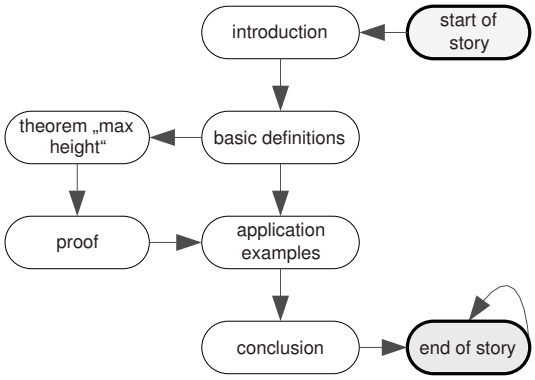

Figure 5.2.: a simple narrative structure

$$
\begin{aligned}
proceed \;=\; & \{(startOfStory, introduction),\\
& (introduction, basicDefinitions),\\
& (basicDefinitions, theorem),\\
& (basicDefinitions, applicationExamples),\\
& (theorem, proof), (proof, applicationExamples),\\
& (applicationExamples, conclusion),\\
& (conclusion, endOfStory), (endOfStory, endOfStory)\}
\end{aligned}
$$

The set of full narrative paths is

$$
\begin{aligned}
FP_d \;=\; & \{(startOfStory, introduction, basicDefinitions,\\
& applicationExamples, conclusion, endOfStory, endOfStory, ...),\\
& (startOfStory, introduction, basicDefinitions, theorem, proof,\\
& applicationExamples, conclusion, endOfStory, endOfStory, ...)\}
\end{aligned}
$$

□

In summary, we have modelled the narrative structure of content in terms of a narrative graph based on content units CU_d, starting unit BOD_d, and the relation $proceed$ which represents sensible ways of proceeding through the content of the document. In addition, it is necessary for expressing and verifying content-related criteria to represent properties of content units and fragments such as topics, intended target groups, objectives, or difficulty level.

Therefore, we introduce a relational model for representing content properties based on description logics.

5.3. Representing Content Properties

5.3.1. Representation of Instance-level Knowledge

We distinguish between *global* properties of the document as a whole and *local* properties of distinct content units $U \in CU_d$. Examples of global properties are the size and type of the document, the authors, target groups, intent of the document, and the major topics and keywords. Examples of local properties of a content unit $U \in CU_d$ are the parts of U and their content types and topics. Separating global from local properties enables compact and focussed knowledge representations, which increase tractability of errors and efficiency of reasoning.

Definition 5.3.1 (Global / Local Fact Base)

The *global fact base* of document d, denoted as gAB_d, is a possibly empty ABox representing global information about a document d.

The *set of local fact bases* of document d, denoted as $\mathbf{lAB}_d$, is a set of ABoxes such that there is a surjective function

$$content : CU_d \rightarrow \mathbf{lAB}_d : content(U) = AB_U$$

where AB_U is an ABox representing information about content unit U.

$$cAB_d := gAB_d \cup \bigcup_{AB \in \mathbf{lAB}_d} AB$$

denotes the *combined fact base* of document d.

□

Remark 5.3.2 (Combined Fact Base)

Note that the term *combined fact base* is an entirely technical term that is used as an abbreviation of the union of the global fact base and all local fact bases. The term ”combined” does not imply or require that the fact bases of a document are integrated on the semantic level.

□

Example 5.3.3 (Global and Local Fact Base)

Consider a global fact base gAB_d representing the major topics of the document, e.g. in the form:

$$\begin{aligned} gAB_d \quad \supseteq \quad & \{KeyWord(TDL) \\ & KeyWord(CTL) \\ & KeyWord(ModelChecking)\} \end{aligned}$$

The local fact base of a certain content unit $u1$ contains a representation of the topics covered in paragraph $p1$ of $u1$:

$$\begin{aligned} AB_{u1} \quad \supseteq \quad & \{hasTopic(p1, LTL), \\ & hasTopic(p1, CTL)\} \end{aligned}$$

By combining the global fact base gAB_d and the local fact base AB_{u1}, we can infer that $p1$ covers a major topic of document d, i.e.

$$gAB_d \cup AB_{u1} \models (\exists hasTopic.KeyWord)(p1)$$

Generally speaking, all instance-level knowledge relevant to inferring properties of a certain content unit $U \in CU_d$ are considered to be represented in $gAB_d \cup AB_U$. □

Remark 5.3.4 (Scalable Modularity of Fact Base)

The mapping $content$ of content units onto ABoxes is surjective. Hence, all ABoxes of the local fact base $\mathbf{lAB}_d$ are related to at least one content unit, i.e. $content^{-1}(\{AB\}) \neq \emptyset$ for any $AB \in \mathbf{lAB}_d$.

However, $content$ is not necessarily injective. As a result, an ABox $AB \in \mathbf{lAB}_d$ can represent information about several content units. Moreover, ABoxes within the fact base may or may not be pairwise disjoint, i.e. they may share some assertions. This way, information about content units that share some or all properties can be compactly represented. □

Corollary 5.3.5 (Fact Base is Nonempty and Finite)

The cardinality of the local fact base ranges from 1 to the cardinality of content units, i.e. the following holds:

$$1 \leq |\mathbf{lAB}_d| \leq |CU_d| \tag{5.5}$$

This is a consequence of the existence of a surjective mapping between CU_d and $\mathbf{lAB}_d$ and the non-emptyness of CU_d (Def. 5.2.1).

Since, by definition, an ABox is a finite set of assertions and CU_d is finite (Definition 5.2.1) the combined fact base cAB_d is finite. □

5.3.2. Representation of Background Knowledge

So far, we have investigated how instance level knowledge about the document's content is represented in terms of a fact base - a set of ABoxes.

A major reason for choosing a DL-based knowledge representation is the possibility to combine instance- and schema-level knowledge. Examples of schema-level knowledge are models of the domain of discourse or document structures that are relatively independent of a single document but valid for a certain class of documents.

Schema-level knowledge enables checking for abstract criteria such as "any major topic must be handled at a central location of the document". For checking such criteria, the system needs to be able to determine, which "topics" are "major topics" and which "locations" of the document are "central". This can be done based on an ontology about document structures and topics represented in terms of a DL knowledge base.

Abstraction allows for formulating simple, general criteria that can be applied in a wide range of use cases because they do not address specific implementation-related properties of a single document. In this way, abstraction allows for decoupling criteria from the implementation of a document.

Even if abstraction is not the major concern within a specific use case, schema-level knowledge can be beneficial. Ontologies of topics and structures of the document serve as a definition of the vocabulary used for both representing knowledge about a document and formalizing criteria. As such, they help to ensure that the checking of criteria does not fail because of inconsistent vocabulary of specification and verification model. Further, contradictive metadata can be discovered prior to model checking. In summary, ontologies help to increase the robustness of the system against specification errors and faulty metadata.

Finally yet importantly, ontologies can help to integrate metadata from several sources and derive implicit knowledge about the document's content, which simplifies knowledge extraction methods and enables compact knowledge representations.

Definition 5.3.6 (Reference Ontology)

The *reference ontology*, denoted as RO, is a DL knowledge base representing a commonly agreed upon reference model for content structures, domain of discourse, learning objects, and other aspects necessary to describe the content and structure of a certain class of documents. Concepts and roles of the reference ontology form the base vocabulary for specifications. □

Remark 5.3.7 (Reference Ontology)

Note that RO can be empty in simple cases (no heterogeneous or distributed metadata, no abstraction of criteria required).

RO represents schema-level knowledge independent of instances of objects found within the current document d. Consequently, RO should mostly consist of a TBox containing axioms about concepts and roles used for representing properties of the content.

□

Example 5.3.8 (Reference Ontology)

A possible reference ontology may contain the following axioms:

$topicOf \doteq hasTopic^-$	role $topicOf$ is the inverse of role $hasTopic$.
$keyWord \sqsubseteq majorTopic$	every key word of the document is a major topic of the document.
$\exists^{\geq 3} topicOf.\top \sqsubseteq majorTopic$	every topic that is topic of more than 2 different fragments is a major topic.
$\exists hasTopic.majorTopic \doteq centralContent$	a fragment covering a major Topic is a central content of the document.

...

□

Remark 5.3.9 (Structure of Reference Ontology)

The reference ontology may be structured into sub-ontologies for different aspects of the content of a document. I.e. there may be a "structure ontology" $SO \subseteq RO$ which represents general content types and structural units of a document. Also, there may be a "domain ontology" $DO \subseteq RO$ representing topics and relationships of topics within the domain of discourse of the document. Further, specific "mediator ontologies" may be provided for aligning the vocabulary used in the fact base about the document to the vocabulary used in specifications.

We assume that these "special-purpose-ontologies" are aligned to each other such that RO is consistent (Definition 3.1.28).

□

For reasoning about the content of a document d, instance-level knowledge represented by the fact bases and schema-level knowledge represented by the reference ontology are combined as follows.

Definition 5.3.10 (Local Knowledge Base and Knowledge Representation)

Let RO be a reference ontology, gAB_d a global fact base of a document d, $U \in CU_d$ a content unit of document d, and $AB_U \in \mathbf{lAB}_d$ the local fact base of U. Let further cAB_d denote the combined fact base of d. Then

- $KB_U := (RO \cup gAB_d \cup AB_U)$ denotes the local knowledge base of content unit U.
- $cKB_d := RO \cup cAB_d$ denotes the *combined knowledge base* of document d.
- $\mathbf{KB}_d = \{KB_U \mid U \in CU_d\}$ denotes the DL *knowledge representation* of the content of document d.
- $kmap : CU_d \rightarrow \mathbf{KB}_d : kmap(U) = KB_U$ denotes the *knowledge mapping* of content units onto their respective local knowledge base.

□

Remark 5.3.11 (Combined Knowledge Base)

Note that similar to the term *combined fact base* (Definition 5.3.1) , *combined knowledge base* is an entirely technical term that does not imply or require that the knowledge bases of a document are integrated on the semantic level (cf. Remark 5.3.2). □

Corollary 5.3.12 (Combined Knowledge Base)

$$cKB_d = \bigcup_{KB \in \mathbf{KB}_d} KB$$

as a consequence of Definition 5.3.1 (cAB_d).

□

Corollary 5.3.13 (Finite Knowledge Representation)

1. $|\mathbf{KB}_d| \leq |CU_d|$ by Definition 5.3.10. Since CU_d is finite (Definition 5.2.1) also $\mathbf{KB}_d$ is finite.
2. Each local knowledge base $KB_U \in \mathbf{KB}_d$ of a document knowledge representation is finite as a consequence of RO being finite (Definition 5.3.6) and gAB_d as well as AB_U being finite (Definition 5.3.1).
3. As a consequence of 1.) and 2.) and Corollary 5.3.12 also the combined knowledge base cKB_d is finite.

□

Remark 5.3.14 (Local Knowledge Base and Knowledge Representation)

$\mathbf{KB}_d$ represents the entire symbolic knowledge about the document. $\mathbf{KB}_d$, in combination with the narrative structure NS_d (Definition 5.2.19) and the knowledge mapping $kmap$, is used to construct a *temporal verification model* about the document's content. The temporal verification model is verified against an $\mathcal{ALC}$CTL specification by model checking. □

The set of individuals, which appear in some local knowledge base, is called *knowledge domain* (see Definition 5.3.15). The knowledge domain is the set of *objects* representing the content of the document. It is an important structure for the definition of the verification procedure in section 6.4.

Definition 5.3.15 (Knowledge Domain)

Let KB be a DL knowledge base, AS the set of application dependent symbols and $IV \subseteq AS$ the set of individuals (Definition 3.1.4). Then

$KB_{|\alpha} := \{\phi \in KB \mid \alpha \text{ appears in } \phi\}$ denotes the set of statements in KB about an application dependent symbol $\alpha \in AS$.

Let KB_U be a local knowledge base of content unit $U \in CU_d$ and cKB_d be the combined knowledge base of document d (Definition 5.3.10).

Then

$IV_{KB_U} := \{i \in IV \mid KB_{U|i} \neq \emptyset\}$ denotes the *local knowledge domain* of content unit U and

$IV_{cKB_d} := \{i \in IV \mid cKB_{d|i} \neq \emptyset\}$ denotes the *knowledge domain* of document d. □

Corollary 5.3.16 (Knowledge Domain is Finite)

The knowledge domain IV_{cKB_d} of a document d is finite.

This is a direct consequence of cKB_d being finite (Corollary 5.3.13) and the fact that each statement of cKB_d is finite and thus contains finitely many individuals. □

Corollary 5.3.17 (Knowledge Domain)

$$IV_{cKB_d} = \bigcup_{U \in CU_d} IV_{KB_U}$$

This is a direct consequence of Corollary 5.3.12 and Definition 5.3.15. □

Corollary 5.3.18 (Local Knowledge Domain is Finite)

Each local knowledge domain IV_{KB_U} for $U \in CU_d$ is finite.

This is the consequence of IV_{cKB_d} being finite (Corollary 5.3.16) and Corollary 5.3.17. □

Remark 5.3.19 (Knowledge Domain)

The local knowledge domain IV_{KB_U} of content unit $U \in CU_d$ represents the set of objects that we have knowledge about in the context of the content unit U. The assertions in KB_U represent properties of these objects.

The knowledge domain IV_{cKB_d} of document d represents the set of objects that we have knowledge about in the context of the entire document. The assertions in cKB_d represent properties of these objects.

The elements of the document's knowledge domain can be constrained by specifications. Hence, the appropriate selection of the knowledge domain is a major concern for representing the content of the document within the semantic model.

Elements of the knowledge domains are representatives of all sort of concrete or abstract objects that can be addressed by specifications. They may represent content fragments such as media objects, paragraphs, sections, but also resources related in some sense to the document's content such as topics, level of difficulty, target groups etc. □

5.3.3. Knowledge Representation - Some Remarks

We use assertions and axioms of a description logic for the representation of the document's content. The choice of the DL is determined by the complexity of background knowledge and respective terminological axioms. Any description logic can be used that has a decidable satisfiability problem. For being of practical use, the chosen description logic should be supported by some reasoning system. Most reasoning systems support very expressive DL such as $\mathcal{SHIQ}$ [HST00] or $\mathcal{SHOQ}(\mathcal{D})$ [HS01a, PH02]. As our framework does not make any assumptions on the DL language besides decidability, future extensions to descriptions logics can be adopted whenever it is beneficial for the application scenario.

The representation of the entire instance- and schema-level knowledge about a document within a single knowledge base KB would cause several problems:

- the likelihood of logical inconsistency of the knowledge base grows in the size of the knowledge base. Since inconsistent knowledge bases imply any logical statement they cannot be used for the verification of semantic criteria (cf. Remark 3.1.33).

- reasoning is exponential in the size of the knowledge base for most relevant description logics (section 3.1.4). As a result, the reasoning performance would not scale well to larger documents when representing the entire knowledge about the document within a single knowledge base.
- finding errors in large knowledge bases is a difficult task [BFH00, LH05, PSK05, WHR+05]. The consistency and soundness of smaller knowledge bases is easier to establish and maintain.

Therefore, we partition the knowledge about a document into several knowledge bases. We distinguish:

- the *fact base*: a set of ABoxes representing instance-level knowledge about the document.
- the *reference ontology*: the reference ontology represents schema-level knowledge about document structures and content and defines the vocabulary used in specifications.

The modular structure of the knowledge representation leads to high scalability, adaptability to different verification scenarios and criteria, and low maintenance cost.

5.3.4. Consistency of Content Knowledge Representations

When combining knowledge from several sources and about different aspects of a document d, inconsistencies may arise from contradictive metadata or interpretation of metadata. For example, consider a definition which is attributed as difficult by one set of metadata and as easy by another. Obviously, not both can be the case at the same time. Contradictions may also arise indirectly. For instance, consider that a definition is attributed as easy and formal. The reference ontology contains the axiom that any formal content is difficult. As a result, content that is attributed as easy *and* formal has contradictive properties.

Obviously, contradictive metadata indicates an error in the implementation of a document and should be discovered and corrected.

Contradictive metadata leads to inconsistent knowledge bases. Recall, a knowledge base is inconsistent iff it is not satisfiable (Definition 3.1.28). An inconsistent knowledge base implies any logical statement. Hence, reasoning results based on inconsistent knowledge bases are meaningless. As a result, there is also a technical necessity for maintaining a certain degree of consistency of the knowledge representation.

Regarding the knowledge representation of a document we distinguish local and global (in-)consistency:

Definition 5.3.20 (Global and Local Consistency)

A knowledge representation $\mathbf{KB}_d$ of document d is *locally consistent* iff KB is satisfiable for each $KB \in \mathbf{KB}_d$. Otherwise $\mathbf{KB}_d$ is *locally inconsistent*.

KB_d is *globally consistent* iff the corresponding combined knowledge base cKB_d is satisfiable. Otherwise $\mathbf{KB}_d$ is *globally inconsistent*. □

Remark 5.3.21 (Global and Local Consistency)

When a document is assembled from different sources, the global consistency of the entire knowledge about the document is usually quite hard to maintain [BGvH+03]. Consequently, we do not require global consistency but just local consistency of the knowledge representation $\mathbf{KB}_d$ of document d. Since the knowledge about the document is partitioned along content units, the maintenance of local inconsistency is significantly simplified. Recall, assertions of KB_U represent only knowledge relevant to a content unit $U \in CU_d$, which is usually a rather small portion of the document's content.

The differences between of local vs. global consistency are illustrated by Example 5.3.22 and discussed in the subsequent remark. □

Example 5.3.22 (Global and Local Consistency)

Let $CU_d = \{U_1, U_2, U_3\}$ be the set of content units of document d and RO be a reference ontology as follows:

$$Plant \sqcap Datastructure \doteq \bot \tag{5.6}$$
$$NatTree \sqsubseteq Plant \sqcap Tree \tag{5.7}$$
$$DSTree \sqsubseteq Datastructure \sqcap Tree \tag{5.8}$$
$$BinTree \sqsubseteq DSTree \tag{5.9}$$
$$\ldots$$

Equation 5.6 expresses that $Plant$ and $Datastructures$ are disjoint concepts, i.e. no individual can be both an instance of $Plant$ and an instance of $Datastructure$. Equation 5.7 defines the class of "natural trees" as "trees" which are "plants". Equation 5.8 defines the class of "data structure trees" as "trees" which are "data structures". Finally, Equation 5.9 defines $BinTree$ as specialization of $DSTree$.

Further, let $gAB_d = \emptyset$ and $\mathbf{lAB}_d = \{AB_{U_1}, AB_{U_2}, AB_{U_3}\}$ with

$$AB_{U_1} = \{DSTree(tree)\} \tag{5.10}$$
$$AB_{U_2} = \{NatTree(tree)\} \tag{5.11}$$
$$AB_{U_3} = \{BinTree(tree)\} \tag{5.12}$$

This expresses that the term $tree$ represents a "data structure" in content unit U_1, a "natural tree" in content unit U_2 and a "binary tree" in content unit U_3, i.e. the meaning of the term $tree$ changes within the document.

Then the combined fact base of document d is

$$cAB_d = \{DSTree(tree), NatTree(tree), BinTree(tree)\}$$

and the combined knowledge base is $cKB_d = RO \cup cAB_d$.

cKB_d is not satisfiable because

$$\{DSTree(tree), DSTree \sqsubseteq Datastructure \sqcap Tree\} \models Datastructure(tree)$$

and

$$\{NatTree(tree), NatTree \sqsubseteq Plant \sqcap Tree\} \models Plant(tree)$$

but $Plant$ and $Datatstructure$ are defined as disjoint concepts in RO Equation 5.6. Hence, the inconsistent use of term $tree$ leads to a globally inconsistent fact base.

However, $\mathbf{KB}_d$ is locally consistent because $RO \cup gAB_d \cup AB_U$ is satisfiable for each $AB_U \in \mathbf{lAB}_d$.

□

Remark 5.3.23 (Global and Local Consistency)

Global consistency is not required for the verification of the document and hence not enforced by our method by default. Still, a globally consistent knowledge representation can be desirable. This is because a globally inconsistent knowledge representation indicates some inconsistent use of terminology within the metadata descriptions of the document as illustrated by the term $tree$ in Example 5.3.22.

For documents composed of heterogeneous sources, inconsistent use of terminology may be tolerable. This is the case, for instance, if different parts of the document apply different metadata standards. For documents that apply a single standard for metadata descriptions, a globally inconsistent knowledge representation may indicate an error in the content of the document that should be accounted for.

If desirable, global consistency can be enforced within the given framework by adding appropriate axioms or assertions to the reference ontology. For instance, adding the assertion $DSTree(tree)$ to RO ensures that the term $tree$ is used consistently as a data structure within the scope of the document. This expresses that in any content unit of the document the term $tree$ represents a data structure. Then $RO \cup gAB_d \cup AB_{U_2}$ is not satisfiable for the same reason as $RO \cup cAB_d$ and hence $\mathbf{KB}_d$ is not locally consistent.

□

Another technical requirement for the knowledge representation $\mathbf{KB}_d$ of document d is that the knowledge domain IV_{cKB_d} is not empty. Recall that the knowledge domain represents the set of objects that we have knowledge about in the context of the document d (Definition 5.3.15 and subsequent Remark 5.3.19). $IV_{cKB_d} = \emptyset$ means that no individuals occur in any of the statements of the knowledge bases $KB_U \in \mathbf{KB}_d$. This in turn implies that none of the local knowledge bases contains an ABox and, consequently, all fact bases about the document are empty.

If no facts about the document are available properties of the document cannot be verified. Therefore, we require for technical reasons that $IV_{cKB_d} \neq \emptyset$ for any knowledge representation $\mathbf{KB}_d$ of document d (see Definition 5.3.24). Note that the condition $IV_{cKB_d} \neq \emptyset$ can always be satisfied by adding the assertion $\top(d)$ to the global fact base gAB_d of the document d. $\top(d)$ represents the fact that (document) d is some known object.

Definition 5.3.24 (Semantic Model)

Let $NS_d = (CU_d, BOD_d, proceed)$ be the narrative structure of a document d (Definition 5.2.19).

Let $\mathbf{KB}_d$ be a locally consistent knowledge representation of document d such that its knowledge domain IV_{cKB_d} is not empty. Let $kmap : CU_d \rightarrow \mathbf{KB}_d$ be the knowledge mapping of content units onto the knowledge representation of the content of document d (Definition 5.3.10).

Then $SM_d := (NS_d, \mathbf{KB}_d, kmap)$ denotes the *semantic model* of a document d. □

Example 5.3.25 (Semantic Model)

Consider a document d consisting of three content units $CU_d = \{sec1, sec2, sec3\}$ where $BOD_d = sec1$ is the beginning of the document. Further there is a narrative relation $proceed = \{(sec1, sec2), (sec1, sec3), (sec2, sec3), (sec3, sec3)\}$, i.e. starting from $sec1$ the user may go to $sec2$ or go immediately to $sec3$ that is the end unit of the document.

Then, by Definition 5.2.19, $NS_d := (CU_d, BOD_d, proceed)$ is the narrative structure of document d.

Assume that knowledge about each content unit is represented in a set of local fact bases $\mathbf{lAB}_d = \{AB_{sec1}, AB_{sec2}, AB_{sec3}\}$ where

$$\begin{aligned} AB_{sec1} &= \{hasObjective(intro, R180), hasObjective(intro, R180\_Handling)\} \\ AB_{sec2} &= \{teaches(L1, R180)\} \\ AB_{sec3} &= \{teaches(L2, R180\_Handling)\} \end{aligned}$$

I.e. content unit $sec1$ contains an introduction, represented by the individual $intro$, which states two objectives: "robot type R180" represented by the individual $R180$ and "Handling of robots of type R180" represented by the individual $R180\_Handling$. Unit $sec2$ contains Lesson 1 of the document represented by individual $L1$. $L1$ teaches topic $R180$. Unit $sec3$ contains Lesson 2 represented by individual $L2$. $L2$ teaches topic $R180\_Handling$.

Assume further that no global facts about the document are available, i.e. $gAB_d = \emptyset$.

As a reference ontology consider

$RO = \{$	$objectiveOf \doteq hasObjective^-$,	$objectiveOf$ is the inverse of role $hasObjective$.
	$addressedBy \doteq addresses^-$,	$addressedBy$ is the inverse of role $addresses$.
	$\top \sqsubseteq \forall addresses.Topic$,	$Topic$ is the range of role $addresses$.
	$teaches \sqsubseteq addresses$,	$teaches$ is a sub-role of $addresses$.
	$\exists teaches.Topic \doteq Lesson$,	$Lesson$ is the class of objects that teach some topic.
	$\exists objectiveOf.\top \sqsubseteq MajorTopic$,	$MajorTopic$ is the domain of role $objectiveOf$.
	$MajorTopic \sqsubseteq Topic$	Every major topic is a topic.
$\}$		

By Definition 5.3.10, we get for the knowledge representation $\mathbf{KB}_d$ of document d:

$$\mathbf{KB}_d = \{KB_{sec1}, KB_{sec2}, KB_{sec3}\}$$

where

$$\begin{aligned}
KB_{sec1} &= RO \cup gAB_d \cup AB_{sec1} \\
&= RO \cup \{hasObjective(intro, R180), \\
&\qquad hasObjective(intro, R180_Handling)\} \\
KB_{sec2} &= RO \cup gAB_d \cup AB_{sec2} \\
&= RO \cup \{teaches(L1, R180)\} \\
KB_{sec3} &= RO \cup gAB_d \cup AB_{sec3} \\
&= RO \cup \{teaches(L2, R180_Handling)\}
\end{aligned}$$

For the knowledge mapping $kmap : CU_d \rightarrow \mathbf{KB}_d$, we get by Definition 5.3.10:

$$\begin{aligned} kmap(sec1) &= KB_{sec1} \\ kmap(sec2) &= KB_{sec2} \\ kmap(sec3) &= KB_{sec3} \end{aligned}$$

The knowledge domain (Definition 5.3.15) of document d is the set of individuals mentioned in any of the local knowledge bases of $\mathbf{KB}_d$ and hence

$$IV_{cKB_d} = \{intro, L1, L2, R180, R180\_Handling\} \neq \emptyset$$

In addition, each of the local knowledge bases KB_{sec1}, KB_{sec2}, KB_{sec3} is satisfiable and hence $\mathbf{KB}_d$ is locally consistent (Definition 5.3.20).

Hence $SM_d := (NS_d, \mathbf{KB}_d, kmap)$ is a semantic model of document d by Definition 5.3.24. □

The semantic model is verified against an $\mathcal{ALC}$CTL specification as shown in section 6.4.

5.4. Semantic Modelling - Related Work

We apply DL to the representation of knowledge about web documents. The semantic model can be implemented using the semantic web standard OWL [BvHH+04] which has a DL semantics. Various tools for editing [BHGS01, NSD+01] and reasoning with OWL [HM01, HMS04b, SPG+07] exist and can be used as component of the proposed system.

We adopt a graph- and logic-based approach to model the discourse domain and content structure of documents. Similar approaches can be found in [BFGHS04], [DHN03], [ESS05], [KF01], and [KBLL+04].

The Living Book system [BFGHS04] aims at intelligent management of personalized and scenario-based teaching material. Parts (so called slices) of documents are associated with learning objectives. Further, semantic dependencies between slices, scenario-specific composition rules, and models of the individual knowledge and goals of each user are represented based on a first-order logic with default negation. A model generating theorem prover is applied for automatically assembling personalized learning documents.

[DHN03] describes a logic-based approach to the automated generation of personalized hypertext structures from distributed metadata. Information provided by markup

within documents, by external metadata files, and by ontologies is used to derive personalized relations between information units. As a unifying framework for different knowledge representation formats the rule language TRIPLE [SD02] is suggested.

[ESS05] suggests DL knowledge bases for aiding the compilation of technical documentation from a base of documentation modules. Modules are annotated by the covered concepts. Based on these annotations, the systems offers a concept-based search of the document base and provides recommendations of which text modules may fit to the current documentation context. Moreover, an ontology-based model of the discourse domain is used to give hints on how to structure larger documentations.

Within the MMiSS project [KBHL+03, KBLL+04] LaTeXdocuments are annotated by ontologies modelling the entities and interrelationships of the discourse domain of the document. This way the terminology used within the documents is standardized and related to the represented general concept(s). This enables "semantic" referencing and automated compilation of documents.

Further approaches to automated hypertext generation based on semantic document models are described in [Boc03, CW01, Dah01, GBvOH03, Hüb00, OBOC04, See03, SSS01]. In contrast to these approaches, we do not aim at automated construction of hypertext but on the verification of given documents. This simplifies the requirements on knowledge representation and reasoning. As opposed to [ESS05, KBLL+04, KF01, See03, SSS01] we do not use ontologies as *complete specifications* of the document's structure and content but for representing *necessary background knowledge* that otherwise would have to be encoded into specifications (cf. section 7.4.1.4). This helps to keep both the ontologies and the specifications small and limits the manual effort for semantic modelling. In addition, ontologies are partitioned into small units. This prevents an exponential explosion of reasoning time and supports the debugging and maintenance of ontologies.

Moreover, in contrast to [BFGHS04, KF01, KBLL+04, See03, SSS01], W3C standards are adopted for the knowledge representation of documents which enable the easy integration of existing metadata sources as well as the application of existing tools for semantic annotation [HS03, KPT+04, UCI+06], information extraction [BCRS06, CHS04, PKO+03], creation and management of ontologies [BHGS01, NSD+01], and ontological reasoning [HM01, SPG+07].

Many approaches, for instance [BFGHS04, Hüb00, OBOC04, See03, SSS01], model semantic dependencies between document parts or covered concepts of the form "part/concept X is a prerequisite for part/concept Y". Representing semantic dependencies on the level of document parts or concepts is expensive in general because the required manual modelling effort may scale quadraticly in the document size and/or concept space. Moreover, in many domains a fixed, generally valid prerequisite relationship between topics is hard to define because the choice of an appropriate order of topics highly depends on individual didactic considerations of the author, the background and focus of the reader, and other properties of the reading context.

In the presented approach, prerequisite relationships are not modelled explicitly within the content model of a document. Instead, such kind of dependencies are represented as general coherence criteria using the specification language which we will introduce in the next chapter. The decoupling of the representation of prerequisite relationships from the document model leads to simpler content models and less manual tagging effort.

5.5. The Semantic Model - Summary of Structures

The semantic model $SM_d = (NS_d, \mathbf{KB}_d, kmap)$ consists of a model of the narrative structure of the document NS_d, a set of DL knowledge bases $\mathbf{KB}_d$, and a mapping $kmap$ between content units of the document and their local knowledge representation.

The narrative structure $NS_d = (CU_d, BOD_d, proceed)$ is a directed graph which models the content structure of the document in terms of a set of content units CU_d with a distinct first unit BOD_d and the narrative relation $proceed$.

The knowledge representation $\mathbf{KB}_d = \{KB_U \mid U \in CU_d\}$ is a set of knowledge bases such that the content of each content unit $U \in CU_d$ is represented by a local knowledge base $KB_U = kmap(U)$. A local knowledge base $KB_U = RO \cup gAB_d \cup AB_U$ contains the reference ontology RO representing general background knowledge about the document, the global fact base gAB_d representing information about the document as a whole, and AB_U representing information about content unit U.

The semantic model will be transformed into an $\mathcal{ALC}$CTL temporal structure that will be verified against a set of $\mathcal{ALC}$CTL formulae by model checking.

6. Document Verification with $\mathcal{ALC}$CTL

6.1. Introduction

In the sequel, we define $\mathcal{ALC}$CTL – a temporal description logic tailored to represent criteria on the semantic document model as introduced in chapter 5. $\mathcal{ALC}$CTL is targeted at achieving the best compromise between high expressiveness for criteria on documents and low computational complexity.

Problem Description (Motivating Example)

Consider the following criteria:

1. "In the overview section, every important function of each robot type must be listed."
2. "For each concept defined in a definition there should be a *matching* example *in some next* content unit."

The first criterion expresses an *external dependency* of the document's content with its domain of discourse while the second criterion expresses an *internal dependency* between parts of the document (cf. section 1.1). Both criteria require to represent sets of objects and relations between objects, namely the concepts of definitions and examples. Such relations among sets of objects can be expressed well in the description logic $\mathcal{ALC}$:

A proper $\mathcal{ALC}$ formalization of the first criterion is:

$$ImportantFunction \sqcap \exists functionOf.RobotType \sqsubseteq \exists listedIn.OverviewSection$$

Note that this formalization relies on some background knowledge about the instances of $ImportantFunction$ of some $RobotType$. This background knowledge is represented in the reference ontology (Definition 5.3.6) and accessed within the verification process.

While $\mathcal{ALC}$, in combination with ontologies, is obviously sufficiently expressive to represent (simple) external dependencies, we run into problems in the case of internal

dependencies along reading paths such as criterion 2) above. Using $\mathcal{ALC}$, we can just approximate criterion 2) as follows:

$$Definition \sqsubseteq \forall defines.\exists illustratedBy.Example$$

"For any definition holds: everything, which is defined by the definition, is illustrated by at least one example."

While description logics allow for the specification of sets of objects (such as definitions) and relationships between objects such as the "defines" relation between definitions and concepts, they cannot express temporal properties of objects. As for criterion 2), it is not possible to express that the required example of each defined concept should appear *in some next* content unit.

Temporal logics are expressive for such temporal properties. For instance, the widely adopted temporal logic CTL (section 3.2.2) can represent the required temporal aspect of criterion 2) as:

$$\mathsf{AG}(definition \rightarrow \mathsf{EX}\ example)$$

"Whenever (AG) a definition occurs there is an example in some of the next content units (EX)."

Due to the lack of variables, propositional temporal logics such as CTL cannot represent relations among sets of objects. Consequently, it is not possible to represent the relation between definitions and defined concepts and between examples and illustrated concepts.

$\mathcal{ALC}$CTL combines CTL and $\mathcal{ALC}$ in a way such that combinations of semantic and temporal relations between objects of the modelling domain can be expressed:

$$\mathsf{AG}(Definition \sqsubseteq \forall defines.\mathsf{EX}\ \exists illustratedBy.Example)$$

"Whenever (AG) a definition occurs, each defined concept needs to be illustrated by an example in some of the next content units (EX)."

The combination of $\mathcal{ALC}$ and CTL allows to express semantic relationships of content objects along paths within the narrative structure, which allows to check the coherence of the content along standard reading paths.

For the verification of temporal specifications there are two basic approaches: theorem proving and model checking. We have opted for modelling the document verification problem as an $\mathcal{ALC}$CTL model checking problem. It is adequate to model document verification as a model checking problem because the narrative structure of documents can be considered finite and completely known. Moreover, theorem proving of temporal description logics is at least EXPTIME-hard [AF01, AFM03, AFWZ02] and thus intractable for large structures (cf. [Sch94]). In the course of this chapter we will show that the $\mathcal{ALC}$CTL model checking problem is in polynomial time and thus can be solved efficiently.

Overview of the Chapter

In the sequel we define the syntax and semantics of $\mathcal{ALC}$CTL. We then define the $\mathcal{ALC}$CTL model checking problem and show basic properties such as decidability and computational complexity. Since model checking temporal description logics has not yet been considered in literature, a new model checking algorithm for $\mathcal{ALC}$CTL had to be developed. This new algorithm is presented and its soundness, completeness, and computational complexity are proven.

We then show how the verification of documents can be modelled as an $\mathcal{ALC}$CTL model checking problem. We define a new verification algorithm for documents, which combines DL reasoning and $\mathcal{ALC}$CTL model checking, and prove its soundness and completeness as well as its computational complexity in different scenarios.

6.2. The Specification Language $\mathcal{ALC}$CTL

$\mathcal{ALC}$CTL is a combination of the description logic $\mathcal{ALC}$ (section 3.1.2) and the branching time temporal logic CTL (section 3.2.2).

6.2.1. Syntax

For the definition of syntax and semantics of $\mathcal{ALC}$CTL, we adopt the notations and style of [Eme90].

Definition 6.2.1 (Syntax of $\mathcal{ALC}$CTL)

Let AC be a countably infinite set of atomic concepts and AR be a countably infinite set of atomic roles. The set of $\mathcal{ALC}$CTL *concepts* on AC and AR, denoted as $\mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$, is the minimal set of concepts generated by the following rules:

(C1) each atomic concept $\mathcal{A} \in AC$ is a concept;
(C2) if C, D are concepts then so are $\neg C$ and $C \sqcap D$;
(C3) if C is a concept and $\mathcal{R} \in AR$ is an atomic role then $\exists \mathcal{R}.C$ is a concept;
(C4) if C is a concept then $\mathsf{AX}\ C$ and $\mathsf{EX}\ C$ are concepts.
(C5) if C and D are concepts then $\mathsf{A}(C\ \mathsf{U}\ D)$ and $\mathsf{E}(C\ \mathsf{U}\ D)$ are concepts.

The set of $\mathcal{ALC}$CTL *formulae*, denoted as $\mathcal{ALC}$CTL, is the minimal set of formulae generated by the following rules:

(F1) if C, D are $\mathcal{ALC}$CTL concepts then $C \sqsubseteq D$ is a formula;
(F2) if p and q are formulae then $\neg p$ and $p \wedge q$ are formulae;
(F3) if p is a formula then $\mathsf{AX}\ p$ and $\mathsf{EX}\ p$ are formulae;
(F4) if p and q are formulae then $\mathsf{A}(p\ \mathsf{U}\ q)$ and $\mathsf{E}(p\ \mathsf{U}\ q)$ are formulae. □

Remark 6.2.2 (Syntax of $\mathcal{ALC}$CTL)

In the $\mathcal{ALC}$CTL language, two similarly structured levels can be identified – the *concept* level and the *formula* level. Concept terms are – much the same as in non-temporal DL – interpreted as sets whereas formulae are interpreted as sentences which may or may not hold in an interpretation.

Similar to CTL, temporal connectives such as X and U always need to be paired with path quantifiers E or A.

It is known that a language over a countable alphabet is countable (cf. [EFT96], for instance). Hence, the set of $\mathcal{ALC}$CTL concepts $\mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ and the set of $\mathcal{ALC}$CTL formulae are countable. □

The syntax definition of $\mathcal{ALC}$CTL is minimal in the sense that further connectives can be expressed in terms of the base connectives of Definition 6.2.1. The distinction between base and extended syntax simplifies subsequent definitions, algorithms, and proofs since only base connectives need to be considered.

Definition 6.2.3 (Extended Syntax of $\mathcal{ALC}$CTL)

Let C, D be $\mathcal{ALC}$CTL concepts, $\mathcal{R} \in AR$ be an atomic role, p, q be $\mathcal{ALC}$CTL formulae, and α, β be $\mathcal{ALC}$CTL concepts or formulae. Then

$\bot$	:=	$C \sqcap \neg C$	"bottom" or "empty" concept
$\top$	:=	$\neg\bot$	"top" or "universal" concept
$C \sqcup D$	:=	$\neg(\neg C \sqcap \neg D)$	C "or" D
$\forall\mathcal{R}.C$	:=	$\neg\exists\mathcal{R}.\neg C$	"universal quantification" on role R
$C \doteq D$	:=	$C \sqsubseteq D \wedge D \sqsubseteq C$	C "equal" D
$false$	:=	$p \wedge \neg p$	constant
$true$	:=	$\neg false$	constant
$p \vee q$	:=	$\neg(\neg p \wedge \neg q)$	p "or" q
$p \rightarrow q$	:=	$\neg p \vee q$	p "implies" q
$\mathsf{AF}\, C$	:=	$\mathsf{A}(\top\, \mathsf{U}\, C)$	"all paths future" C
$\mathsf{AF}\, p$	:=	$\mathsf{A}(true\, \mathsf{U}\, p)$	"all paths future" p
$\mathsf{EF}\, C$	:=	$\mathsf{E}(\top\, \mathsf{U}\, C)$	"some path future" or "reachable" C
$\mathsf{EF}\, p$	:=	$\mathsf{E}(true\, \mathsf{U}\, p)$	"some path future" or "reachable" p
$\mathsf{AG}\, \alpha$	:=	$\neg\mathsf{EF}\, \neg\alpha$	"all paths globally" or "always" α
$\mathsf{EG}\, \alpha$	:=	$\neg\mathsf{AF}\, \neg\alpha$	"some path globally" α
$\mathsf{A}(C\, \mathsf{W}\, D)$	:=	$\neg\mathsf{E}(\neg D\, \mathsf{U}\, (\neg D \sqcap \neg C))$	"all path" C "weak until" D
$\mathsf{A}(p\, \mathsf{W}\, q)$	:=	$\neg\mathsf{E}(\neg q\, \mathsf{U}\, (\neg q \wedge \neg p))$	"all path" p "weak until" q
$\mathsf{E}(C\, \mathsf{W}\, D)$	:=	$\mathsf{E}(C\, \mathsf{U}\, D) \sqcup \mathsf{EG}\, C$	"some path" C "weak until" D
$\mathsf{E}(p\, \mathsf{W}\, q)$	:=	$\mathsf{E}(p\, \mathsf{U}\, q) \vee \mathsf{EG}\, p$	"some path" p "weak until" q
$\mathsf{A}(\alpha\, \mathsf{B}\, \beta)$	:=	$\neg\mathsf{E}(\neg\alpha\, \mathsf{U}\, \beta)$	"all path" α "before" β
$\mathsf{E}(\alpha\, \mathsf{B}\, \beta)$	:=	$\neg\mathsf{A}(\neg\alpha\, \mathsf{U}\, \beta)$	"some path" α "before" β

The binding precedence of connectives introduced above is as follows: all connectives used as concept constructors (rules C1 through C5 in Definition 6.2.1) have higher binding power than the connectives used to build formulae (rules F1 through F4 in Definition 6.2.1). The precedence of concept constructors from highest to lowest binding power is as follows: path quantifiers A, E, temporal connectives $\mathsf{X}, \mathsf{F}, \mathsf{G}, \mathsf{U}, \mathsf{W}, \mathsf{B}$, non-temporal concept constructors $\forall, \exists, \neg, \sqcap, \sqcup$. The binding precedence of connectives in formulae is: path quantifiers A, E, temporal operators $\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{B}$, non-temporal formula connectives $\sqsubseteq, \doteq, \neg, \wedge, \vee, \rightarrow$. □

Example 6.2.4 (Syntax of $\mathcal{ALC}$CTL)

The following are valid $\mathcal{ALC}$CTL formulae:

$Introduction \sqsubseteq ContentUnit$	Every introduction is a content unit.
$Overview \sqsubseteq \exists hasTitle.$ $(ShortTitle \sqcup FullTitle)$	Every overview has a title that is a short or full title.
$\neg(Definition \sqsubseteq Formal)$	Not all definitions are formal.
$\neg Definition \sqsubseteq Formal$	Everything, which is not a definition, is formal.
$Theorem \sqsubseteq Formal \rightarrow$ $Proof \sqsubseteq Formal$	If every theorem is formal then every proof must be formal, too.
$Theorem \sqcap Formal \sqsubseteq$ $\exists shownBy.(Proof \sqcap Formal)$	Every formal theorem must be shown by a formal proof.
$\mathsf{AG}(Theorem \sqcap Formal \sqsubseteq$ $\mathsf{EX}\ \exists shownBy.(Proof \sqcap Formal))$	On all paths holds globally that every formal theorem must be shown by a formal proof in one of the next states.
$\mathsf{EG}(Difficult \sqsubseteq \bot)$	There is a path such that in every state there is no difficult object.
$\mathsf{EF}\ Difficult \sqsubseteq \bot$	There is no object that is on some path at some state classified as difficult.
$\mathsf{EF}\ Defined \sqsubseteq$ $\mathsf{A}(\neg Applied\ \mathsf{W}\ Defined)$	Everything, which is defined eventually on some path, is on all paths not applied (weak) until it is defined.
$\mathsf{AG}(Definition \sqsubseteq$ $\forall defines.\mathsf{EF}\ \exists appliedIn.\top)$	On all paths in all states, it holds for any definition that every concept defined is on some path eventually applied in something.

The following expressions are not valid:

$\top \sqsubseteq Fragment \vee Topic$	$\vee$ can only be applied to formulae but not to concepts.
AF $Conclusion$	This is an $\mathcal{ALC}$CTL concept but not an $\mathcal{ALC}$CTL formula.
$\exists$EF $hasTopic.Tested$	Modal operators such as EF are not allowed for roles.
$\top \sqsubseteq$ A(X $Help$ U $Test$)	Temporal connectives X, U, F, G, B, U, W need to be paired with a path quantifier E or A.
A($Test \sqsubseteq$ EX $Solution$)	$Test \sqsubseteq$ EX $Solution$ is a formula. However, path quantifier E or A can only be applied to path formulae.

□

Remark 6.2.5 (Syntax of $\mathcal{ALC}$CTL)

Example 6.2.4 demonstrates that the non-temporal part of $\mathcal{ALC}$CTL alone already exceeds $\mathcal{ALC}$. This is because Boolean connectives such as $\wedge, \vee, \rightarrow$ and negation of formulae are available in $\mathcal{ALC}$CTL but not in $\mathcal{ALC}$.

In contrast, the syntax definition of $\mathcal{ALC}$CTL implies that $\mathcal{ALC}$CTL contains $\mathcal{ALC}$, i.e. every $\mathcal{ALC}$ formula is also a syntactically correct $\mathcal{ALC}$CTL formula. □

6.2.2. Semantics

$\mathcal{ALC}$CTL formulae are interpreted over temporal structures which are closely related to the semantic model of a document. The semantics of $\mathcal{ALC}$CTL defines when an $\mathcal{ALC}$CTL formula holds in a given temporal structure.

There are several different approaches to defining the semantics of a first order branching time logic. Our definition is based on state transition graphs and thus is closely related to [Eme90] and [vB95].

Definition 6.2.6 (Temporal Structure)

$\mathcal{ALC}$CTL formulae are interpreted on *temporal structures* $M = (S, R, \Delta, I)$ such that

- S is a nonempty set of states.
- $R \subseteq S \times S$ is a *left-total* binary relation on S assigning to each state in S at least one successor in S.
- Δ is a set of objects – the *interpretation domain*.

- I is a function $S \rightarrow LI$ associating each state $s \in S$ with a *local interpretation* $I(s) \in LI$, with LI being the set of local $\mathcal{ALC}$ interpretations. Each local interpretation $I(s) \in LI$ is a function associating each atomic concept $\mathcal{A} \in AC$ with a set of domain objects $\mathcal{A}^{I(s)} \subseteq \Delta$ and each atomic Role $\mathcal{R} \in AR$ with a set of pairs $\mathcal{R}^{I(s)} \subseteq \Delta \times \Delta$.

$\mathbf{M}_{\mathcal{ALC}\mathsf{CTL}}$ denotes the set of $\mathcal{ALC}$CTL temporal structures.

$\mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}} := \{(S, R, \Delta, I) \in \mathbf{M}_{\mathcal{ALC}\mathsf{CTL}} \mid |S| \in \mathbb{N}_1 \wedge |\Delta| \in \mathbb{N}_1\}$ denotes the set of *finite* and *nonempty* $\mathcal{ALC}$CTL temporal structures

□

Example 6.2.7 (Temporal Structure)

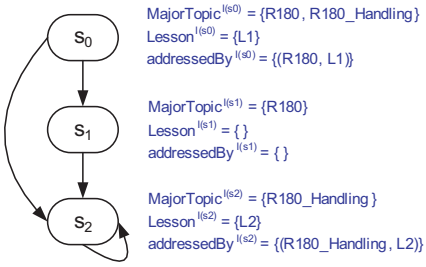

Figure 6.1.: sample temporal structure

Consider the set of atomic concepts $AC = \{MajorTopic, Lesson\}$ and the set of atomic roles $AR = \{addressedBy\}$. $MajorTopic$ is interpreted as the set of major topics of a document. $Lesson$ is interpreted as the set of lessons of a learning document. $addressedBy$ is interpreted as $\{(a, b) \mid$ topic a is addressed by fragment $b\}$.

Consider the temporal structure $M = (S, R, \Delta, I)$ depicted in Figure 6.1 with

- the set of states $S := \{s_0, s_1, s_2\}$;
- the left-total relation $R := \{(s_0, s_1), (s_0, s_2), (s_1, s_2), (s_2, s_2)\} \subseteq S \times S$;
- the domain of objects $\Delta := \{$*L1, L2, R180, R180_Handling*$\}$;

- the interpretation of concepts in AC:

MajorTopic$^{I(s_0)}$	$=$	$\{$*R180, R180_Handling*$\}$	*R180* and *R180_Handling* are the major topics in state s_0.
Lesson$^{I(s_0)}$	$=$	$\{$*L1*$\}$	*L1* is the (only) lesson in state s_0.
MajorTopic$^{I(s_1)}$	$=$	$\{$*R180*$\}$	*R180* is the (only) major topic in state s_1.
Lesson$^{I(s_1)}$	$=$	$\emptyset$	there is no lesson in state s_1.
MajorTopic$^{I(s_2)}$	$=$	$\{$*R180_Handling*$\}$	*R180_Handling* is the (only) major topic in state s_2.
Lesson$^{I(s_2)}$	$=$	$\{$*L2*$\}$	*L2* is the (only) lesson in state s_2.

- and the interpretation of roles in AR:

addressedBy$^{I(s_0)}$	$=$	$\{$*(R180,L1)*$\}$	*R180* is addressed by *L1* in state s_0.
addressedBy$^{I(s_1)}$	$=$	$\emptyset$	there is no topic addressed in state s_1.
addressedBy$^{I(s_2)}$	$=$	$\{$*(R180_Handling,L2)*$\}$	*R180_Handling* is addressed by *L2* in state s_2.

The truth of an $\mathcal{ALC}$CTL formula such as

$$MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.Lesson$$

– every major topic is eventually addressed by a lesson on all paths – can be evaluated in the structure (S, R, Δ, I) (see Example 6.2.11). □

Similar to CTL, the semantics of temporal connectives of $\mathcal{ALC}$CTL are defined w.r.t. full paths in the state transition system (S, R) (Definition 3.2.6). Recall that FP_s denotes the set of full paths $\{(s_0, s_1, ...) \mid s_0 = s \wedge \forall i \in \mathbb{N} : (s_i, s_{i+1}) \in R\}$ within (S, R) starting from state $s \in S$.

Definition 6.2.8 (Semantics of $\mathcal{ALC}$CTL)

Let $M = (S, R, \Delta, I)$ be a temporal structure (Definition 6.2.6), AC be a set of atomic concepts, AR be a set of atomic roles, $s \in S$ a state, $R(s) := \{s' \in S \mid (s, s') \in R\}$ be the set of successor states of s in R, and FP_s be the set of full paths in (S, R) starting from s.

Let C, D be $\mathcal{ALC}$CTL concepts, $\mathcal{A} \in AC$ an atomic concept, and $\mathcal{R} \in AR$ an atomic role.

Then the semantics of $\mathcal{ALCC}$CTL *concepts* w.r.t. M, s, in symbols $(M, s)(C)$, is defined as follows:

$$
\begin{array}{lrcl}
\text{(C1)} & (M,s)(\mathcal{A}) & := & \mathcal{A}^{I(s)} \\
\text{(C2)} & (M,s)(C \sqcap D) & := & (M,s)(C) \cap (M,s)(D) \\
 & (M,s)(\neg C) & := & \Delta \backslash (M,s)(C) \\
\text{(C3)} & (M,s)(\exists \mathcal{R}.C) & := & \{a \in \Delta \mid \exists b \in \Delta : (a,b) \in \mathcal{R}^{I(s)} \wedge b \in (M,s)(C)\} \\
\text{(C4)} & (M,s)(\mathsf{AX}\ C) & := & \bigcap_{s' \in R(s)} (M,s')(C) \\
 & (M,s)(\mathsf{EX}\ C) & := & \bigcup_{s' \in R(s)} (M,s')(C) \\
\text{(C5)} & (M,s)(\mathsf{A}(C\ \mathsf{U}\ D)) & := & \bigcap_{(s_0,s_1,\ldots) \in FP_s} \{a \in \Delta \mid \exists i \in \mathbb{N} : a \in (M,s_i)(D) \wedge \\
 & & & \forall j \in \{0,\ldots,i-1\} : a \in (M,s_j)(C)\} \\
 & (M,s)(\mathsf{E}(C\ \mathsf{U}\ D)) & := & \bigcup_{(s_0,s_1,\ldots) \in FP_s} \{a \in \Delta \mid \exists i \in \mathbb{N} : a \in (M,s_i)(D) \wedge \\
 & & & \forall j \in \{0,\ldots,i-1\} : a \in (M,s_j)(C)\}
\end{array}
$$

$(M, s)(C)$ is abbreviated as $C^{I(s)}$ if M is understood from the context.

The following rules determine when an $\mathcal{ALCC}$CTL formula p is true in M, s, in symbols $M, s \models p$. Let p, q be $\mathcal{ALCC}$CTL formulae. Then

$$
\begin{array}{lrcll}
\text{(F1)} & M,s & \models & C \sqsubseteq D & \text{iff } C^{I(s)} \subseteq D^{I(s_0)} \\
\text{(F2)} & M,s & \models & \neg p & \text{iff } M,s \not\models p \\
 & M,s & \models & p \wedge q & \text{iff } M,s \models p \text{ and } M,s \models q \\
\text{(F3)} & M,s & \models & \mathsf{AX}\ p & \text{iff } M,s' \models p \text{ for each } s' \in R(s) \\
 & M,s & \models & \mathsf{EX}\ p & \text{iff } M,s' \models p \text{ for some } s' \in R(s) \\
\text{(F4)} & M,s & \models & \mathsf{A}(p\ \mathsf{U}\ q) & \text{iff } \forall (s_0,s_1,\ldots) \in FP_s\ \exists i \in \mathbb{N} : M,s_i \models q \text{ and} \\
 & & & & \quad \forall j \in \{0,\ldots,i-1\} : M,s_j \models p \\
 & M,s & \models & \mathsf{E}(p\ \mathsf{U}\ q) & \text{iff } \exists (s_0,s_1,\ldots) \in FP_s\ \exists i \in \mathbb{N} : M,s_i \models q \text{ and} \\
 & & & & \quad \forall j \in \{0,\ldots,i-1\} : M,s_j \models p
\end{array}
$$

□

Remark 6.2.9 (Semantics of $\mathcal{ALCC}$CTL)

The semantics of $\mathcal{ALCC}$CTL is defined on two layers: the concept layer and the formula layer. Formulae $p \in \mathcal{ALCC}$CTL hold or do not hold in a temporal structure M at a state s which is denoted as $M, s \models p$ (or $M, s \not\models p$, respectively).

In contrast to formulae, the interpretation of $\mathcal{ALCC}$CTL concepts yields subsets of the interpretation domain Δ. For instance, the interpretation of the temporal concept $\mathsf{EX}\ Defined$ at a state $s \in S$ yields the subset of domain objects in Δ, which are instances of concept $Defined$ at some next state s', in symbols $(\mathsf{EX}\ Defined)^{I(s)} = \bigcup_{s' \in R(s)} Defined^{I(s')}$.

□

Definition 6.2.10 (Validity and Temporal Models)

Let $M = (S, R, \Delta, I)$ be an $\mathcal{ALCC}$CTL temporal structure, $s \in S$ a state, and p an $\mathcal{ALCC}$CTL formula.

Then M, s is a *(temporal) model* of p (or alternatively, M *satisfies* p at s or p *holds in* M at s) iff $M, s \models p$. □

Example 6.2.11 (Interpretation and Validity of $\mathcal{ALCC}$CTL Formulae)

Consider the $\mathcal{ALCC}$CTL formula $f :=$ *MajorTopic* $\sqsubseteq$ AF $\exists$*addressedBy.Lesson* and the temporal structure $M = (S, R, \Delta, I)$ of Example 6.2.7 as depicted in Figure 6.1.

The validity of the formula f in M can be determined as follows.

By rule (C3) of the $\mathcal{ALCC}$CTL semantics (Definition 6.2.8) it holds:

$$\begin{aligned}(\exists \textit{addressedBy.Lesson})^{I(s_0)} &= \{\textit{R180}\}\\ (\exists \textit{addressedBy.Lesson})^{I(s_1)} &= \emptyset\\ (\exists \textit{addressedBy.Lesson})^{I(s_2)} &= \{\textit{R180\_Handling}\}\end{aligned}$$

i.e. the temporal interpretation of the $\mathcal{ALCC}$CTL concept $\exists$*addressedBy.Lesson* yields {*R180*} in state s_0, $\emptyset$ in state s_1, and {*R180_Handling*} in state s_2.

By application of Definition 6.2.3, the expression AF $\exists$*addressedBy.Lesson* in f expands to A($\top$ U $\exists$*addressedBy.Lesson*).

Using rule (C5) of Definition 6.2.8 we get:

$$(\mathsf{AF}\ \exists \textit{addressedBy.Lesson})^{I(s_0)} = \{\textit{R180, R180\_Handling}\} \quad (6.1)$$

$$(\mathsf{AF}\ \exists \textit{addressedBy.Lesson})^{I(s_1)} = \{\textit{R180\_Handling}\} \quad (6.2)$$

$$(\mathsf{AF}\ \exists \textit{addressedBy.Lesson})^{I(s_2)} = \{\textit{R180\_Handling}\} \quad (6.3)$$

Equation (6.1) can be seen as follows:

$$\begin{aligned}&(\mathsf{AF}\ \exists \textit{addressedBy.Lesson})^{I(s_0)}\\ =\ &(M, s_0)(\mathsf{A}(\top\ \mathsf{U}\ \exists \textit{addressedBy.Lesson}))\\ =\ &\bigcap_{(s_0,s_1,\ldots)\in FP_{s_0}} \{a \in \Delta \mid \exists i \in \mathbb{N} : a \in (M, s_i)(\exists \textit{addressedBy.Lesson}) \wedge\\ &\forall j \in \{0, ..., i-1\} : a \in (M, s_j)(\top)\}\\ =\ &\bigcap_{(s_0,s_1,\ldots)\in FP_{s_0}} \{a \in \Delta \mid \exists i \in \mathbb{N} : a \in (M, s_i)(\exists \textit{addressedBy.Lesson})\}\end{aligned}$$

Hence $(\mathsf{AF}\ \exists \mathit{addressedBy.Lesson})^{I(s_0)}$ yields the set of objects which are addressed by a lesson at some state on all paths from state s_0.

In (S, R) there are exactly two paths $p = (p_0, p_1, ...) := (s_0, s_1, s_2, s_2, ...)$ and $p' = (p'_0, p'_1, ...) := (s_0, s_2, s_2, ...)$ starting from s_0 and hence $FP_{s_0} = \{p, p'\}$. Thus we get

$$
\begin{aligned}
& \bigcap_{(s_0,s_1,...)\in FP_{s_0}} \{a \in \Delta \mid \exists i \in \mathbb{N} : a \in (M, s_i)(\exists \mathit{addressedBy.Lesson})\} \\
= \ & \{a \in \Delta \mid \exists i \in \mathbb{N} : a \in (M, p_i)(\exists \mathit{addressedBy.Lesson})\} \cap \\
& \{a \in \Delta \mid \exists i \in \mathbb{N} : a \in (M, p'_i)(\exists \mathit{addressedBy.Lesson})\} \\
= \ & (\bigcup_{i\in\mathbb{N}} (M, p_i)(\exists \mathit{addressedBy.Lesson})) \cap (\bigcup_{i\in\mathbb{N}} (M, p'_i)(\exists \mathit{addressedBy.Lesson})) \\
= \ & ((\exists \mathit{addressedBy.Lesson})^{I(s_0)} \cup (\exists \mathit{addressedBy.Lesson})^{I(s_1)} \cup \\
& (\exists \mathit{addressedBy.Lesson})^{I(s_2)}) \\
& \cap ((\exists \mathit{addressedBy.Lesson})^{I(s_0)} \cup (\exists \mathit{addressedBy.Lesson})^{I(s_2)}) \\
= \ & (\{\mathit{R180}\} \cup \emptyset \cup \{\mathit{R180\_Handling}\}) \cap (\{\mathit{R180}\} \cup \{\mathit{R180\_Handling}\}) \\
= \ & \{\mathit{R180}, \mathit{R180\_Handling}\}
\end{aligned}
$$

Equations (6.2) and (6.3) can be shown analogously.

Since $\mathit{MajorTopic}^{I(s_0)} = \{\mathit{R180, R180\_Handling}\}$ and hence $\mathit{MajorTopic}^{I(s_0)} \subseteq (\mathsf{AF}\ \exists \mathit{addressedBy.Lesson})^{I(s_0)}$ we get by rule (F1) in Definition 6.2.8:

$$M, s_0 \models \mathit{MajorTopic} \sqsubseteq \mathsf{AF}\ \exists \mathit{addressedBy.Lesson}$$

Analogously we get $M, s_1 \not\models f$ and $M, s_2 \models f$. □

6.3. Model Checking $\mathcal{ALC}$CTL

6.3.1. Definition of the $\mathcal{ALC}$CTL Model Checking Problem

Verification of document structures is based on the model checking problem of $\mathcal{ALC}$CTL.

Definition 6.3.1 ($\mathcal{ALC}$CTL Label Set)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite temporal structure, i.e. S and Δ are finite, nonempty sets. Let $f \in \mathcal{ALC}$CTL be a formula.

Then

$$LS_{M,f} := \{s \in S \mid M, s \models f\}$$

denotes the *label set* of f in M. □

Definition 6.3.2 ($\mathcal{ALCC}$CTL Model Checking Problem)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALCC}\mathsf{CTL}}$ be a finite temporal structure and $f \in \mathcal{ALCC}\mathsf{CTL}$ be a formula.

Then the $\mathcal{ALCC}$CTL *model checking problem*, denoted as $MC_{\mathcal{ALCC}\mathsf{CTL}}(M, f)$, is determining the label set $LS_{M,f}$ of f in M. □

6.3.2. Properties of the $\mathcal{ALCC}$CTL Model Checking Problem

Decidability

The $\mathcal{ALCC}$CTL model checking problem can be reduced to the model checking problem of propositional CTL. The reduction to CTL shows the decidability and an upper bound of computational complexity of the $\mathcal{ALCC}$CTL model checking problem. Also, the CTL reduction of $\mathcal{ALCC}$CTL sets the ground for a sound and complete $\mathcal{ALCC}$CTL model checking algorithm.

Proposition 6.3.3 (CTL Reducibility of $\mathcal{ALCC}$CTL)

The $\mathcal{ALCC}$CTL model checking problem is reducible to the CTL model checking problem in the following way.

Let $\mathbf{fM}_{\mathcal{ALCC}\mathsf{CTL}}$ be the set of finite $\mathcal{ALCC}$CTL temporal structures (Definition 6.2.6) and $\mathbf{fM}_{\mathsf{CTL}}$ be the set of finite CTL temporal structures (Definition 3.2.4).

Then there is a structure mapping $sm : \mathbf{fM}_{\mathcal{ALCC}\mathsf{CTL}} \times \mathcal{ALCC}\mathsf{CTL} \rightarrow \mathbf{fM}_{\mathsf{CTL}}$ and a formula mapping $fm : \mathbf{fM}_{\mathcal{ALCC}\mathsf{CTL}} \times \mathcal{ALCC}\mathsf{CTL} \rightarrow \mathsf{CTL}$ such that for each finite $\mathcal{ALCC}$CTL temporal structure $(S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALCC}\mathsf{CTL}}$, state $s \in S$, and formula $f \in \mathcal{ALCC}\mathsf{CTL}$ holds:

$$M, s \models_{\mathcal{ALCC}\mathsf{CTL}} f \Leftrightarrow sm(M, f), s \models_{\mathsf{CTL}} fm(M, f) \tag{6.4}$$

□

The general idea for proving Proposition 6.3.3 is as follows: since the interpretation domain Δ of M is finite, the temporal interpretation I can be encoded in terms of a propositional labelling function L. As a consequence, $\mathcal{ALCC}$CTL temporal structures can be encoded by means of CTL temporal structures. Moreover, in the case of finite domains, the explicit and implicit first order quantifications in $\mathcal{ALCC}$CTL formulae can be unfolded to finite Boolean expressions. For $\Delta = \{a_1, ..., a_n\}$ the formula $C \sqsubseteq D$ can be expanded to $(C(a_1) \rightarrow D(a_1)) \wedge (C(a_2) \rightarrow D(a_2)) \wedge ... \wedge (C(a_n) \rightarrow D(a_n))$ $= \bigwedge_{a \in \Delta}(C(a) \rightarrow D(a))$.

To prove Proposition 6.3.3 we define suitable structure and formula mappings sm and fm and then show that sm and fm satisfy the assertion of Proposition 6.3.3.

As for the definition of the structure mapping sm, consider a finite $\mathcal{ALC}$CTL temporal structure $(S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$. Since the interpretation domain Δ is finite, the interpretation function I can be encoded in terms of a propositional labelling function L assigning each state in S a set of atomic propositions satisfied in S in the following way.

For each state $s \in S$, objects $a, b \in \Delta$, atomic concept $\mathcal{A} \in AC$, and atomic role $\mathcal{R} \in AR$: if $a \in \mathcal{A}^{I(s)}$ then add the atomic proposition $\mathcal{A}(a)$ to $L(s)$; if $(a, b) \in \mathcal{R}^{I(s)}$ then add the atomic proposition $\mathcal{R}(a, b)$ to $L(s)$. More formally:

Definition 6.3.4 ($\mathcal{ALC}$CTL to CTL Structure Mapping)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure, f an $\mathcal{ALC}$CTL formula, AC the set of atomic concepts, AR the set of atomic roles, $AC_{|f} \subseteq AC$ be the set of atomic concepts occurring in f, and $AR_{|f} \subseteq AR$ the set of atomic roles occurring in f.

Let AP be a set of *atomic propositions* such that

$$AP \supseteq \{\mathcal{A}(a) \mid \mathcal{A} \in AC_{|f} \wedge a \in \Delta\} \cup \{\mathcal{R}(a, b) \mid \mathcal{R} \in AR_{|f} \wedge a, b \in \Delta\}$$

The relevant part of the interpretation function I w.r.t. f is transformed into a propositional labelling $L_{I,f}$ of states S in the following way:

$$\begin{aligned} L_{I,f} := S \to \mathcal{P}(AP) : L_{I,f}(s) \quad = \quad & \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f} \wedge a \in \mathcal{A}^{I(s)}\} \cup \\ & \{\mathcal{R}(a, b) \in AP \mid \mathcal{R} \in AR_{|f} \wedge (a, b) \in \mathcal{R}^{I(s)}\} \end{aligned}$$

Then $(S, R, L_{I,f})$ is a finite CTL temporal structure and

$$sm : \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}} \times \mathcal{ALC}\mathsf{CTL} \to \mathbf{fM}_{\mathsf{CTL}} : sm((S, R, \Delta, I), f) := (S, R, L_{I,f})$$

denotes a CTL *structure mapping* of $\mathcal{ALC}$CTL temporal structures w.r.t. $\mathcal{ALC}$CTL formulae.

□

Remark 6.3.5 ($\mathcal{ALC}$CTL to CTL Structure Mapping)

The structure mapping sm in Definition 6.3.4 is well defined because it is restricted to *finite* $\mathcal{ALC}$CTL temporal structures $\mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$.

□

Example 6.3.6 ($\mathcal{ALCC}$CTL to CTL Structure Mapping)

Consider the $\mathcal{ALCC}$CTL temporal structure (S, R, Δ, I) as of Example 6.2.7 and the $\mathcal{ALCC}$CTL formula $f := MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.\top$.

Then the result of the structure mapping $sm((S, R, \Delta, I), f)$ is $(S, R, L_{I,f})$ with

$$\begin{aligned} L_{I,f}(s_1) &= \{MajorTopic(R180), MajorTopic(R180\_Handling), \\ &\quad addressedBy(R180, L1)\} \\ L_{I,f}(s_2) &= \{MajorTopic(R180)\} \\ L_{I,f}(s_3) &= \{MajorTopic(R180\_Handling), \\ &\quad addressedBy(R180\_Handling, L2)\} \end{aligned}$$

□

For the definition of a mapping of $\mathcal{ALCC}$CTL formulae onto CTL formulae consider an $\mathcal{ALCC}$CTL formula f interpreted w.r.t. finite $\mathcal{ALCC}$CTL temporal structures. Again, since the interpretation domain Δ is finite, f can be reduced to an equivalent CTL formula f' by a finite Boolean encoding of each quantified expression appearing in f and recursively replacing the DL connectives with Boolean expressions as follows:

Definition 6.3.7 ($\mathcal{ALCC}$CTL to CTL Formula Mapping)

$fm : \mathbf{fM}_{\mathcal{ALCC}\mathsf{CTL}} \times \mathcal{ALCC}\mathsf{CTL} \rightarrow \mathsf{CTL}$ is a mapping of $\mathcal{ALCC}$CTL temporal structures and formulae onto CTL formulae inductively defined as follows:

Let $M = (S, R, \Delta, I)$ be an $\mathcal{ALCC}$CTL temporal structure, C, D $\mathcal{ALCC}$CTL concepts, and p, q $\mathcal{ALCC}$CTL formulae. Then

$$\begin{aligned} fm(M, C \sqsubseteq D) &:= \bigwedge_{a \in \Delta} (C[a]_\Delta \rightarrow D[a]_\Delta) \quad (Def.\ C[a]_\Delta \text{ see below}) \\ fm(M, \neg p) &:= \neg fm(M, p) \\ fm(M, p \wedge q) &:= fm(M, p) \wedge fm(M, q) \\ fm(M, \mathsf{EX}\ p) &:= \mathsf{EX}\ fm(M, p) \\ fm(M, \mathsf{AX}\ p) &:= \mathsf{AX}\ fm(M, p) \\ fm(M, \mathsf{E}(p\ \mathsf{U}\ q)) &:= \mathsf{E}(fm(M, p)\ \mathsf{U}\ fm(M, q)) \\ fm(M, \mathsf{A}(p\ \mathsf{U}\ q)) &:= \mathsf{A}(fm(M, p)\ \mathsf{U}\ fm(M, q)) \end{aligned}$$

For the definition of $fm(M, C \sqsubseteq D)$ we inductively define a concept mapping $C[a]_\Delta$ as follows.

Let $\mathcal{A} \in AC$ be an atomic concept, $\mathcal{R} \in AR$ an atomic role, C, D $\mathcal{ALC}$CTL concepts, and $a \in \Delta$ a domain object. Then

$$
\begin{aligned}
\top[a]_\Delta &:= true \\
\bot[a]_\Delta &:= false \\
\mathcal{A}[a]_\Delta &:= \mathcal{A}(a) \\
(C \sqcap D)[a]_\Delta &:= C[a]_\Delta \wedge D[a]_\Delta \\
(\neg C)[a]_\Delta &:= \neg(C[a]_\Delta) \\
(\exists \mathcal{R}.C)[a]_\Delta &:= \bigvee_{b \in \Delta} (\mathcal{R}(a,b) \wedge C[b]_\Delta) \\
(\mathsf{EX}\ C)[a]_\Delta &:= \mathsf{EX}\ C[a]_\Delta \\
(\mathsf{AX}\ C)[a]_\Delta &:= \mathsf{AX}\ C[a]_\Delta \\
(\mathsf{E}(C\ \mathsf{U}\ D))[a]_\Delta &:= \mathsf{E}(C[a]_\Delta\ \mathsf{U}\ D[a]_\Delta) \\
(\mathsf{A}(C\ \mathsf{U}\ D))[a]_\Delta &:= \mathsf{A}(C[a]_\Delta\ \mathsf{U}\ D[a]_\Delta)
\end{aligned}
$$

□

Remark 6.3.8 ($\mathcal{ALC}$CTL to CTL Formula Mapping is Well-Defined)

The formula mapping fm as defined in Definition 6.3.7 replaces every $\mathcal{ALC}$CTL connective not available in CTL by a finite expression over Boolean connectives. Connectives coinciding in $\mathcal{ALC}$CTL and CTL remain unchanged by fm.

$\mathcal{ALC}$CTL connectives not available in CTL are $\sqsubseteq, \top, \bot, \sqcap, \exists \mathcal{R}.C$. fm maps "$\sqsubseteq$" onto an expression consisting of "$\wedge$" and "$\rightarrow$" connectives, "$\top$" is mapped onto the Boolean constant "$true$", "$\bot$" onto the Boolean constant "$false$", "$\sqcap$" onto "$\wedge$", and the existential role quantification "$\exists \mathcal{R}.C$" onto an expression consisting of "$\vee$" and "$\wedge$" connectives.

Since Δ is finite and nonempty (Definition 6.2.6), the mappings of "$\sqsubseteq$" and "$\exists \mathcal{R}.C$" result in finite and nonempty expressions. Also, all other mappings are finite. With $\mathcal{A}(a) \in AP$ and $\mathcal{R}(a,b) \in AP$ for each atomic concept $\mathcal{A} \in AC$, atomic role $\mathcal{R} \in AR$, domain objects $a, b \in \Delta$ and AP being the set of atomic propositions, we get by induction on the structure of f that $fm(M, f)$ results in a finite CTL formula over the set of atomic propositions AP. As a result $fm(M, f) \in$ CTL for $M \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ and $f \in \mathcal{ALC}$CTL and thus fm is well-defined. □

Example 6.3.9 ($\mathcal{ALC}$CTL to CTL Formula Mapping)

Consider the $\mathcal{ALC}$CTL temporal structure $M = (S, R, \Delta, I)$ as of Example 6.2.7 and the $\mathcal{ALC}$CTL formula $f := MajorTopic \sqsubseteq \exists addressedBy.\top$.

Recall that in Example 6.2.7 $\Delta = \{L1, L2, R180, R180\_Handling\}$, which is abbreviated as $\Delta = \{l_1, l_2, r_1, r_2\}$ for convenience within the scope of this example.

The formula mapping $fm(M, f)$ can be constructed in a bottom-up manner as follows:

$$\top[l_1]_\Delta = \top[l_2]_\Delta = \top[r_1]_\Delta = \top[r_2]_\Delta = true$$

$$\begin{aligned}
(\exists addressedBy.\top)[l_1]_\Delta &= \bigvee_{b\in\Delta} (addressedBy(l_1, b) \wedge \top[b]_\Delta) \\
&= \bigvee_{b\in\Delta} (addressedBy(l_1, b) \wedge true) \\
&= \bigvee_{b\in\Delta} addressedBy(l_1, b) \\
&= addressedBy(l_1, l_1) \vee addressedBy(l_1, l_2) \\
&\quad \vee addressedBy(l_1, r_1) \vee addressedBy(l_1, r_2) \\
(\exists addressedBy.\top)[l_2]_\Delta &= addressedBy(l_2, l_1) \vee addressedBy(l_2, l_2) \\
&\quad \vee addressedBy(l_2, r_1) \vee addressedBy(l_2, r_2) \\
(\exists addressedBy.\top)[r_1]_\Delta &= addressedBy(r_1, l_1) \vee addressedBy(r_1, l_2) \\
&\quad \vee addressedBy(r_1, r_1) \vee addressedBy(r_1, r_2) \\
(\exists addressedBy.\top)[r_2]_\Delta &= addressedBy(r_2, l_1) \vee addressedBy(r_2, l_2) \\
&\quad \vee addressedBy(r_2, r_1) \vee addressedBy(r_2, r_2)
\end{aligned}$$

Finally, $fm(M, MajorTopic \sqsubseteq \exists addressedBy.\top)$

$$\begin{aligned}
&= \bigwedge_{a\in\Delta} (MajorTopic[a]_\Delta \rightarrow (\exists addressedBy.\top)[a]_\Delta) \\
&= (MajorTopic(l_1) \rightarrow (\exists addressedBy.\top)[l_1]_\Delta) \wedge \\
&\quad (MajorTopic(l_2) \rightarrow (\exists addressedBy.\top)[l_2]_\Delta) \wedge \\
&\quad (MajorTopic(r_1) \rightarrow (\exists addressedBy.\top)[r_1]_\Delta) \wedge \\
&\quad (MajorTopic(r_2) \rightarrow (\exists addressedBy.\top)[r_2]_\Delta) \\
&= (MajorTopic(l_1) \rightarrow (addressedBy(l_1, l_1) \vee addressedBy(l_1, l_2) \\
&\qquad \vee addressedBy(l_1, r_1) \vee addressedBy(l_1, r_2))) \wedge \\
&\quad (MajorTopic(l_2) \rightarrow (addressedBy(l_2, l_1) \vee addressedBy(l_2, l_2) \\
&\qquad \vee addressedBy(l_2, r_1) \vee addressedBy(l_2, r_2))) \wedge \\
&\quad (MajorTopic(r_1) \rightarrow (addressedBy(r_1, l_1) \vee addressedBy(r_1, l_2) \\
&\qquad \vee addressedBy(r_1, r_1) \vee addressedBy(r_1, r_2))) \wedge \\
&\quad (MajorTopic(r_2) \rightarrow (addressedBy(r_2, l_1) \vee addressedBy(r_2, l_2) \\
&\qquad \vee addressedBy(r_2, r_1) \vee addressedBy(r_2, r_2)))
\end{aligned}$$

□

Remark 6.3.10 (Example $\mathcal{ALCC}$CTL to CTL Formula Mapping)

Example 6.3.9 shows that the CTL mapping of an $\mathcal{ALCC}$CTL formula w.r.t. a temporal structure M can become large. As a worst case for the size of the CTL mapping, consider a finite temporal structure $M = (S, R, \Delta, I)$ and an $\mathcal{ALCC}$CTL formula of the following shape:

$$f_{wc} = \top \sqsubseteq \exists R_1.\exists R_2....\exists R_n.C$$

The size $|fm(M, f)|$ of the CTL mapping $fm(M, f)$ can be approximated as follows: let $a \in \Delta$ be an atomic object. Then

$|(\exists R_n.C)[a]_\Delta| = |\bigvee_{b\in\Delta}(R_n(a, b) \wedge C[b]_\Delta)| \geq |\Delta|$.

$|(\exists R_{n-1}.\exists R_n.C)[a]_\Delta| = |\bigvee_{b\in\Delta}(R_{n-1}(a, b) \wedge (\exists R_n.C)[b]_\Delta)| \geq |\Delta|\cdot|\Delta| = |\Delta|^2$.

$|(\exists R_{n-2}.\exists R_{n-1}.\exists R_n.C)[a]_\Delta| = |\bigvee_{b\in\Delta}(R_{n-2}(a, b) \wedge (\exists R_{n-1}.\exists R_n.C)[b]_\Delta)| \geq |\Delta|\cdot|\Delta|^2 = |\Delta|^3$.

...

$|fm(M, f_{wc})| \geq |(\exists R_1.\exists R_2....\exists R_n.C)[a]_\Delta| \geq |\Delta|^n$

Hence, in the worst case, the size of the CTL mapping $fm(M, f)$ is exponential in the size of f and linear in the size of Δ, i.e. $|fm(M, f)| \in \Omega(|\Delta|^{p(|f|)})$ with $M \in \mathbf{fM}_{\mathcal{ALCC}\mathsf{CTL}}$ and $f \in \mathcal{ALCC}$CTL, and p a polynomial $\mathbb{N} \to \mathbb{R}^+$ of degree higher than 0. This suggests that the complexity of model checking $\mathcal{ALCC}$CTL formulae is exponential in the size of the formula $|f|$ to check. Fortunately, this is not the case as shown in Proposition 6.3.14. □

Next, it is shown that the mappings sm and fm as defined in Definitions 6.3.4 and 6.3.7 are sound and complete, i.e.

$$M, s \models_{\mathcal{ALCC}\mathsf{CTL}} f \Leftrightarrow sm(M, f), s \models_{\mathsf{CTL}} fm(M, f)$$

for any finite temporal structure $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALCC}\mathsf{CTL}}$, state $s \in S$, and formula $f \in \mathcal{ALCC}$CTL.

As a first step we show that the concept mapping $\cdot\,[a]_\Delta$ is sound and complete in the following sense:

Lemma 6.3.11 (Correctness of Concept Mapping)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALCC}\mathsf{CTL}}$ be a finite $\mathcal{ALCC}$CTL temporal structure, $s \in S$ a state, f an $\mathcal{ALCC}$CTL formula, C' an $\mathcal{ALCC}$CTL concept occurring in f, and $M' := (S, R, L_{I,f}) = sm(M, f)$ the CTL structure mapping of M w.r.t. f. Then

$$\forall a \in \Delta : a \in (M, s)(C') \Leftrightarrow M', s \models_{\mathsf{CTL}} C'[a]_\Delta \tag{6.5}$$

Proof:

Assume $C' = \top$. Let $a \in \Delta$. Then it holds by Definitions 6.2.3 and 6.2.8, that $a \in (M, s)(\top)$. Since $C'[a]_\Delta = true$, also $M', s \models_{\mathsf{CTL}} C'[a]_\Delta$. Since the case $a \notin (M, s)(\top)$ does not occur for any $a \in \Delta$, the equivalence (6.5) is shown in the case of $C' = \top$. For $C' = \bot$, the equivalence (6.5) holds analogously.

Assume $C' = \mathcal{A}$ with $\mathcal{A} \in AC$ being an atomic concept. Then

$$
\begin{array}{lll}
 & a \in (M, s)(C') & \\
\Leftrightarrow & a \in (M, s)(\mathcal{A}) & \\
\Leftrightarrow & a \in \mathcal{A}^{I(s)} & (\mathcal{ALC}\mathsf{CTL}\text{ Semantics}) \\
\Leftrightarrow & \mathcal{A}(a) \in L_{I,f}(s) & (\text{Def. of } L_{I,f} \text{ in Definition 6.3.4}) \\
\Leftrightarrow & M', s \models \mathcal{A}(a) & (\text{Def. of } M', \mathsf{CTL}\text{ Semantics}) \\
\Leftrightarrow & M', s \models \mathcal{A}[a]_\Delta & (\text{Def. of } \mathcal{A}[a]_\Delta \text{ in Definition 6.3.7}) \\
\Leftrightarrow & M', s \models C'[a]_\Delta &
\end{array}
$$

The remaining cases follow by induction on the structure of C':

Assume $C' = C \sqcap D$. Then

$$
\begin{array}{lll}
 & a \in (M, s)(C') & \\
\Leftrightarrow & a \in (M, s)(C \sqcap D) & \\
\Leftrightarrow & a \in (M, s)(C) \text{ and } a \in (M, s)(D) & (\mathcal{ALC}\mathsf{CTL}\text{ Sem. '}\sqcap\text{'}) \\
\Leftrightarrow & M', s \models C[a]_\Delta \text{ and } M', s \models D[a]_\Delta & (\text{Induction Hypothesis}) \\
\Leftrightarrow & M', s \models C[a]_\Delta \wedge D[a]_\Delta & (\mathsf{CTL}\text{ Semantics '}\wedge\text{'}) \\
\Leftrightarrow & M', s \models (C \sqcap D)[a]_\Delta & (\text{Definition of } \cdot\,[a]_\Delta) \\
\Leftrightarrow & M', s \models C'[a]_\Delta &
\end{array}
$$

For $C' = \neg C$ we get analogously:

$$
\begin{array}{lll}
 & a \in (M, s)(C') & \\
\Leftrightarrow & a \in (M, s)(\neg C) & \\
\Leftrightarrow & a \notin (M, s)(C) & (\mathcal{ALC}\mathsf{CTL}\text{ Sem. '}\neg\text{'}) \\
\Leftrightarrow & M', s \not\models C[a]_\Delta & (\text{Induction Hypothesis}) \\
\Leftrightarrow & M', s \models \neg(C[a]_\Delta) & (\mathsf{CTL}\text{ Semantics '}\neg\text{'}) \\
\Leftrightarrow & M', s \models (\neg C)[a]_\Delta & (\text{Definition of } \cdot\,[a]_\Delta) \\
\Leftrightarrow & M', s \models C'[a]_\Delta &
\end{array}
$$

Furthermore, we get for $C' = \exists\mathcal{R}.C$

$$
\begin{array}{lll}
& a \in (M,s)(C') & \\
\Leftrightarrow & a \in (M,s)(\exists\mathcal{R}.C) & \\
\Leftrightarrow & \exists b \in \Delta : (a,b) \in \mathcal{R}^{I(s)} \wedge b \in (M,s)(C) & (\mathcal{ALC}\mathsf{CTL}\text{ Semantics}) \\
\Leftrightarrow & \bigvee_{b\in\Delta}((a,b) \in \mathcal{R}^{I(s)} \wedge b \in (M,s)(C)) & (\text{because } \Delta \text{ is finite}) \\
\Leftrightarrow & \bigvee_{b\in\Delta}(\mathcal{R}(a,b) \in L_{I,f}(s) \wedge M',s \models C[b]_\Delta) & (\text{Def. } L_{I,f}\text{, Ind. Hypothesis}) \\
\Leftrightarrow & \bigvee_{b\in\Delta}(M',s \models \mathcal{R}(a,b) \wedge M',s \models C[b]_\Delta) & (\mathsf{CTL}\text{ Semantics}) \\
\Leftrightarrow & \bigvee_{b\in\Delta} M',s \models \mathcal{R}(a,b) \wedge C[b]_\Delta & (\mathsf{CTL}\text{ Semantics of '}\wedge\text{'}) \\
\Leftrightarrow & M',s \models \bigvee_{b\in\Delta}(\mathcal{R}(a,b) \wedge C[b]_\Delta) & (\mathsf{CTL}\text{ Semantics of '}\vee\text{'}) \\
\Leftrightarrow & M',s \models (\exists\mathcal{R}.C)[a]_\Delta & (\text{Definition of } \cdot\,[a]_\Delta) \\
\Leftrightarrow & M',s \models C'[a]_\Delta &
\end{array}
$$

The remaining cases are concepts formed by the temporal connectives $\mathsf{EX}, \mathsf{AX}, \mathsf{EU}, \mathsf{AU}$. Since the structure mapping sm does not affect the transition system S, R, the concept mapping $\cdot\,[a]_\Delta$ does not affect the temporal connectives, and the semantics of temporal connectives is analogous in $\mathcal{ALC}$CTL and CTL, Equation (6.5) holds also in the cases with the temporal connectives as shown in detail for the case of $C' = \mathsf{E}(C\ \mathsf{U}\ D)$:

$$
\begin{array}{lll}
& a \in (M,s)(C') & \\
\Leftrightarrow & a \in (M,s)(\mathsf{E}(C\ \mathsf{U}\ D)) & \\
\Leftrightarrow & a \in \bigcup_{(s_0,s_1,\ldots)\in FP_s}\{a' \in \Delta \mid \exists i \in \mathbb{N} : a' \in (M,s_i)(D) \wedge & \\
& \forall j \in \{0,\ldots,i-1\} : a' \in (M,s_j)(C)\} & (\mathcal{ALC}\mathsf{CTL}\text{ Semantics}) \\
\Leftrightarrow & \exists(s_0,s_1,\ldots) \in FP_s \exists i \in \mathbb{N} : (a \in (M,s_i)(D) \wedge & \\
& \forall j \in \{0,\ldots,i-1\} : a \in (M,s_j)(C)) & \\
\Leftrightarrow & \exists(s_0,s_1,\ldots) \in FP_s \exists i \in \mathbb{N} : (M',s_i \models D[a]_\Delta \wedge & \\
& \forall j \in \{0,\ldots,i-1\} : M',s_j \models C[a]_\Delta) & (\text{Ind. Hypothesis}) \\
\Leftrightarrow & M',s \models \mathsf{E}(C[a]_\Delta\ \mathsf{U}\ D[a]_\Delta) & (\mathsf{CTL}\text{ Semantics,} \\
& & S,R\text{ same in } M,\ M') \\
\Leftrightarrow & M',s \models (\mathsf{E}(C\ \mathsf{U}\ D))[a]_\Delta & (\text{Definition of } \cdot\,[a]_\Delta) \\
\Leftrightarrow & M',s \models C'[a]_\Delta &
\end{array}
$$

For $C' = \mathsf{A}(C\ \mathsf{U}\ D)$ we can prove analogously

$$
\begin{array}{lll}
& a \in (M,s)(C') & \\
\Leftrightarrow & a \in (M,s)(\mathsf{A}(C\ \mathsf{U}\ D)) & \\
\Leftrightarrow & M',s \models \mathsf{A}(C[a]_\Delta\ \mathsf{U}\ D[a]_\Delta) & (\mathcal{ALC}\mathsf{CTL}\text{ Sem. + Ind. Hyp. + }\mathsf{CTL}\text{ Sem.}) \\
\Leftrightarrow & M',s \models (\mathsf{A}(C\ \mathsf{U}\ D))[a]_\Delta & \\
\Leftrightarrow & M',s \models C'[a]_\Delta &
\end{array}
$$

and for $C' = \mathsf{EX}\ C$

$$
\begin{array}{ll}
 & a \in (M,s)(C') \\
\Leftrightarrow & a \in (M,s)(\mathsf{EX}\ C) \\
\Leftrightarrow & M', s \models \mathsf{EX}(C[a]_\Delta) \quad (\mathcal{ALC}\text{CTL Sem. + Ind. Hyp. + CTL Sem.}) \\
\Leftrightarrow & M', s \models (\mathsf{EX}\ C)[a]_\Delta \\
\Leftrightarrow & M', s \models C'[a]_\Delta
\end{array}
$$

The remaining case $C' = \mathsf{AX}\ C$ is along the same lines. Thus, the equivalence (6.5) has been shown for all concepts $C' \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$. □

We now show Proposition 6.3.12 by induction on the structure of an $\mathcal{ALC}$CTL formula f. Proposition 6.3.3 follows from Proposition 6.3.12.

Proposition 6.3.12 (Correctness of Structure and Formula Mapping)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure, $s \in S$ a state, f an $\mathcal{ALC}$CTL formula, $M' := (S, R, L_{I,f}) = sm(M, f)$ be the CTL mapping of the $\mathcal{ALC}$CTL structure M w.r.t. formula f, and $f' := fm(M, f)$ be the CTL mapping of the $\mathcal{ALC}$CTL formula f w.r.t. structure M. Then

$$M, s \models_{\mathcal{ALC}\mathsf{CTL}} f \Leftrightarrow M', s \models_{\mathsf{CTL}} f' \tag{6.6}$$

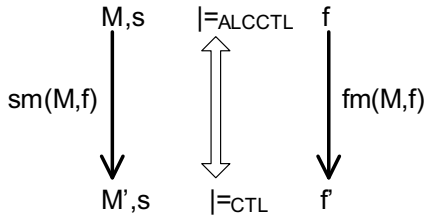

Figure 6.2.: validity-equivalent mappings from $\mathcal{ALC}$CTL to CTL

Figure 6.2 depicts the relationship between $\mathcal{ALC}$CTL structures M, f and their CTL mappings M', f' as expressed by equivalence (6.6).

Proof:

Let $f = C \sqsubseteq D$ with C, D $\mathcal{ALC}$CTL being concepts. Then

$$
\begin{array}{lll}
& M, s \models f & \\
\Leftrightarrow & M, s \models C \sqsubseteq D & \\
\Leftrightarrow & (M,s)(C) \subseteq (M,s)(D) & (\mathcal{ALC}\text{CTL Sem. of '}\sqsubseteq\text{'}) \\
\Leftrightarrow & \forall a \in \Delta : a \in (M,s)(C) \rightarrow a \in (M,s)(D) & ((M,s)(C) \subseteq \Delta \text{ and} \\
 & & (M,s)(D) \subseteq \Delta) \\
\Leftrightarrow & \bigwedge_{a \in \Delta} (a \in (M,s)(C) \rightarrow a \in (M,s)(D)) & \text{(because } \Delta \text{ is finite)} \\
\Leftrightarrow & \bigwedge_{a \in \Delta} ((M', s \models C[a]_\Delta) \rightarrow (M', s \models D[a]_\Delta)) & \text{(Lemma 6.3.11)} \\
\Leftrightarrow & \bigwedge_{a \in \Delta} M', s \models C[a]_\Delta \rightarrow D[a]_\Delta & (\text{CTL Sem. '}\rightarrow\text{'}) \\
\Leftrightarrow & M', s \models \bigwedge_{a \in \Delta} (C[a]_\Delta \rightarrow D[a]_\Delta) & (\text{CTL Sem. '}\wedge\text{'}) \\
\Leftrightarrow & M', s \models fm(M, C \sqsubseteq D) & \text{(Definition 6.3.7)} \\
\Leftrightarrow & M', s \models fm(M, f) &
\end{array}
$$

The remaining cases follow by induction on the structure of f. Assume $f = \neg p$ with $p \in \mathcal{ALC}$CTL. Then

$$
\begin{array}{lll}
& M, s \models f & \\
\Leftrightarrow & M, s \models \neg p & \\
\Leftrightarrow & M, s \not\models p & (\mathcal{ALC}\text{CTL Sem. '}\neg\text{'}) \\
\Leftrightarrow & M', s \not\models fm(M, p) & \text{(Ind. Hypothesis)} \\
\Leftrightarrow & M', s \models \neg fm(M, p) & (\text{CTL Sem. '}\neg\text{'}) \\
\Leftrightarrow & M', s \models fm(M, \neg p) & \text{(Definition 6.3.7)} \\
\Leftrightarrow & M', s \models fm(M, f) &
\end{array}
$$

In the case of $f = p \wedge q$ we get analogously:

$$
\begin{array}{lll}
& M, s \models f & \\
\Leftrightarrow & M, s \models p \wedge q & \\
\Leftrightarrow & M, s \models p \text{ and } M, s \models q & (\mathcal{ALC}\text{CTL Sem. '}\wedge\text{'}) \\
\Leftrightarrow & M', s \models fm(M, p) \text{ and } M', s \models fm(M, q) & \text{(Ind. Hypothesis)} \\
\Leftrightarrow & M', s \models fm(M, p) \wedge fm(M, q) & (\text{CTL Sem. '}\wedge\text{'}) \\
\Leftrightarrow & M', s \models fm(M, p \wedge q) & \text{(Definition 6.3.7)} \\
\Leftrightarrow & M', s \models fm(M, f) &
\end{array}
$$

Also the cases of temporal connectives are straight forward as demonstrated in detail

for the case of $f = \mathsf{A}(p \,\mathsf{U}\, q)$:

$$
\begin{array}{lll}
& M, s \models f & \\
\Leftrightarrow & M, s \models \mathsf{A}(p \,\mathsf{U}\, q) & \\
\Leftrightarrow & \forall (s_0, s_1, ...) \in FP_s \exists i \in \mathbb{N} : (M, s_i \models q & \\
& \wedge \forall j \in \{0, ..., i-1\} : M, s_j \models p) & (\mathcal{ALC}\mathsf{CTL} \text{ Sem. '}\mathsf{A}(\cdot \,\mathsf{U}\, \cdot)\text{'}) \\
\Leftrightarrow & \forall (s_0, s_1, ...) \in FP_s \exists i \in \mathbb{N} : (M', s_i \models fm(M, q) & \\
& \wedge \forall j \in \{0, ..., i-1\} : M', s_j \models fm(M, p)) & \text{(Ind. Hypothesis)} \\
\Leftrightarrow & M', s \models \mathsf{A}(fm(M, p) \,\mathsf{U}\, fm(M, q)) & (\mathsf{CTL} \text{ Sem. } \mathsf{A}(\cdot \,\mathsf{U}\, \cdot), \\
& & S, R \text{ same in } M, M') \\
\Leftrightarrow & M', s \models fm(M, \mathsf{A}(p \,\mathsf{U}\, q)) & \text{(Definition 6.3.7)} \\
\Leftrightarrow & M', s \models fm(M, f) &
\end{array}
$$

The remaining cases $f = \mathsf{E}(p \,\mathsf{U}\, q)$, $f = \mathsf{EX}\, p$, and $f = \mathsf{AX}\, p$ can be proven analogously:

$$
\begin{array}{lll}
& M, s \models \mathsf{E}(p \,\mathsf{U}\, q) & \\
\Leftrightarrow & M', s \models \mathsf{E}(fm(M, p) \,\mathsf{U}\, fm(M, q)) & (\mathcal{ALC}\mathsf{CTL} \text{ Sem. EU + Ind. Hyp. +} \\
& & \mathsf{CTL} \text{ Sem. EU}) \\
\Leftrightarrow & M', s \models fm(M, \mathsf{E}(p \,\mathsf{U}\, q)) & \text{(Definition 6.3.7)}
\end{array}
$$

Finally,

$$
\begin{array}{lll}
& M, s \models \mathsf{EX/AX}\, p & \\
\Leftrightarrow & M', s \models \mathsf{EX/AX}\, fm(M, p) & (\mathcal{ALC}\mathsf{CTL} \text{ Sem. EX/AX + Ind. Hyp. +} \\
& & \mathsf{CTL} \text{ Sem. EX/AX}) \\
\Leftrightarrow & M', s \models fm(M, \mathsf{EX/AX}\, p) & \text{(Definition 6.3.7)}
\end{array}
$$

In total, we have proven the equivalence (6.6) and, consequently, Propositions 6.3.12 and 6.3.3.

□

Corollary 6.3.13 (Decidability of $MC_{\mathcal{ALC}\mathsf{CTL}}$)

The $\mathcal{ALC}$CTL model checking problem is decidable, i.e. it is decidable for a given finite temporal structure $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$, state $s \in S$, and formula $f \in \mathcal{ALC}\mathsf{CTL}$ if $M, s \models f$.

This is a direct consequence of Proposition 6.3.3 and the decidability of the CTL model checking problem [Eme90]. □

Computational Complexity

Besides decidability, the computational complexity of the model checking problem is most relevant for practical applications because it gives a bound for the optimal runtime complexity of a sound and complete model checking algorithm.

Proposition 6.3.14 (Complexity of $MC_{\mathcal{ALC}\mathsf{CTL}}$)

The runtime complexity of the $\mathcal{ALC}$CTL model checking problem $MC_{\mathcal{ALC}\mathsf{CTL}}(M, f)$ (Definition 6.3.2) is in polynomial time w.r.t. the size of the model M and the size of the formula f.

Proof (Sketch):

The polynomial time complexity of the model checking problem is somewhat surprising since the CTL formula mapping fm between $\mathcal{ALC}$CTL and CTL (Definition 6.3.7) generates CTL formulae of exponential size (Remark 6.3.10).

The general idea of showing the polynomial complexity of the $\mathcal{ALC}$CTL model checking problem is as follows.

Let $f \in \mathcal{ALC}\mathsf{CTL}$ be an $\mathcal{ALC}$CTL formula and $M = (S, R, \Delta, I)$ a finite $\mathcal{ALC}$CTL temporal structure.

According to Proposition 6.3.12, $M, s \models_{\mathcal{ALC}\mathsf{CTL}} f$ can be decided by solving the CTL model checking problem $sm(M, f), s \models_{\mathsf{CTL}} fm(M, f)$.

Hence, the complexity of the $\mathcal{ALC}$CTL model checking problem is not higher than the complexity of the structure mapping $sm(M, f)$ plus the complexity of the formula mapping $fm(M, f)$ plus the complexity of deciding the CTL model checking problem $sm(M, f), s \models_{\mathsf{CTL}} fm(M, f)$.

Assume that all concepts in f are atomic. Then any subsumption expression $A \sqsubseteq B$ in f can be mapped onto an equivalent Boolean expression $\bigwedge_{a \in \Delta}(A(a) \rightarrow B(a))$ of length $\mathcal{O}(|\Delta|)$. The CTL mappings of the remaining connectives do not change the size of the expression and thus each are in $\mathcal{O}(1)$. f contains at most $|f|$ subsumptions and $|f|$ other connectives and hence, the size of the formula mapping $fm(M, f)$ results in a CTL expression of size $|fm(M, f)| \in \mathcal{O}(|f| \cdot |\Delta|)$ which is also the complexity of the formula mapping fm. Recall, the size $|f|$ of a formula f is the number of sub-expressions in f (cf. Theorem 3.2.12).

It can be shown that the structure mapping $sm(M, f)$ is in $\mathcal{O}(|f| \cdot |S| \cdot |\Delta|)$ time if f contains atomic concepts only.

Deciding $sm(M, f), s \models_{\mathsf{CTL}} fm(M, f)$ for each $s \in S$ is known to be in $\mathcal{O}((|S| + |R|) \cdot |fm(M, f)|)$ (Theorem 3.2.12) and hence the time complexity of deciding $sm(M, f), s \models_{\mathsf{CTL}} fm(M, f)$ for each $s \in S$ is in $\mathcal{O}((|S| + |R|) \cdot |f| \cdot |\Delta|)$.

As a result, in the case of f containing atomic concepts only, we get the overall complexity of

$\mathcal{O}(|f|\cdot|S|\cdot|\Delta|) + \mathcal{O}(|f|\cdot|\Delta|) + \mathcal{O}((|S|+|R|)\cdot|f|\cdot|\Delta|) \subseteq \mathcal{O}(|f|\cdot(|S|+|R|)\cdot|\Delta|)$

If f contains complex concepts we can reduce the problem of checking $M, s \models_{\mathcal{ALC}\text{CTL}} f$ to $M', s \models_{\mathcal{ALC}\text{CTL}} f'$ with f' containing atomic concepts only. This can be done by extending the interpretation I in M successively to complex concepts in a bottom up manner which takes at most $|f|$ steps each of which is in $\mathcal{O}((|S|+|R|)\cdot|\Delta|^2)$.

In total, the elimination of complex concepts in f is in $\mathcal{O}(|f|\cdot(|S|+|R|)\cdot|\Delta|^2)$.

The overall complexity $\mathcal{O}(|f|\cdot(|S|+|R|)\cdot|\Delta|^2)$ of the $\mathcal{ALC}$CTL model checking problem for M and f sums up from

- the time for calculating the complex concept reduction M', f' from M, f, which is in $\mathcal{O}(|f|\cdot(|S|+|R|)\cdot|\Delta|^2)$,
- the complexity of deciding $M', s \models_{\mathcal{ALC}\text{CTL}} f'$ which is in $\mathcal{O}(|f'|\cdot(|S|+|R|)\cdot|\Delta|)$.

□

In the sequel, we introduce the necessary formal constructs for proving the complexity of reducing an $\mathcal{ALC}$CTL structure M and formula f to a structure M' and a formula f' without any complex concepts.

Definition 6.3.15 (Sub-Concept, Simple / Complex Concept)

$\lfloor C \rfloor$ denotes the set of *direct sub-concepts* of an $\mathcal{ALC}$CTL concept $C \in \mathcal{C}_{\mathcal{ALC}\text{CTL}}$, i.e.

$$\lfloor C \rfloor := \begin{cases} \emptyset & \text{iff } C \in AC \cup \{\top, \bot\} \\ \{C_1\} & \text{iff } C \in \{\neg C_1, \exists R.C_1, \mathsf{EX}\ C_1, \mathsf{AX}\ C_1\} \\ \{C_1, C_2\} & \text{iff } C \in \{C_1 \sqcap C_2, \mathsf{E}(C_1\ \mathsf{U}\ C_2), \mathsf{A}(C_1\ \mathsf{U}\ C_2)\} \end{cases}$$

where C_1, C_2 are $\mathcal{ALC}$CTL concepts.

If $\lfloor C \rfloor = \emptyset$ then C is a *simple* $\mathcal{ALC}$CTL concept otherwise C is called *complex*. □

Example 6.3.16 (Sub-Concept)

Consider the $\mathcal{ALC}$CTL concepts $C = \mathsf{EX}\ \exists R.\top \sqcap \neg D$ and $C' = \mathsf{EX}\ (\exists R.\top \sqcap \neg D)$.

Then $\lfloor C \rfloor = \{\mathsf{EX}\ \exists R.\top, \neg D\}$ and $\lfloor C' \rfloor = \{\exists R.\top \sqcap \neg D\}$ □

Definition 6.3.17 (Mapping onto Atomic Concepts)

Let $atom : \mathcal{C}_{\mathcal{ALC}\text{CTL}} \to AC$ be a fixed injective mapping of $\mathcal{ALC}$CTL concepts onto atomic concepts. Such a mapping exists since the set of $\mathcal{ALC}$CTL concepts is countable (cf. Remark 6.2.2) and the set of atomic concepts is countably infinite (Definition 6.2.1).

Then for $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$

$$\langle C \rangle := atom(C)$$

denotes the *atomic substitute* for $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$.

$C^{\langle\rangle} \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ denotes the *semi-atomic* substitute of C, which is C with all its direct sub-concepts being replaced by their atomic substitute: for a concept $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$

$$C^{\langle\rangle} := \begin{cases} C[C_1/\langle C_1 \rangle] & \text{if } \lfloor C \rfloor = \{C_1\} \\ C[C_1/\langle C_1 \rangle][C_2/\langle C_2 \rangle] & \text{if } \lfloor C \rfloor = \{C_1, C_2\} \\ C & \text{otherwise} \end{cases}$$

where $C[x/y]$ denotes the substitution of term x by term y in expression C. □

Example 6.3.18 (Mapping onto Atomic Concepts)

Consider the $\mathcal{ALC}$CTL concepts $C = \mathsf{EX}\ \exists R.\top \sqcap \neg D$ and $C' = \mathsf{EX}\ (\exists R.\top \sqcap \neg D)$.

Then $\langle C \rangle = \langle \mathsf{EX}\ \exists R.\top \sqcap \neg D \rangle$ and $\langle C' \rangle = \langle \mathsf{EX}\ (\exists R.\top \sqcap \neg D) \rangle$ are atomic concepts such that $\langle C \rangle \neq \langle C' \rangle$.

Further,

$C^{\langle\rangle} = \langle \mathsf{EX}\ \exists R.\top \rangle \sqcap \langle \neg D \rangle$ and

$C'^{\langle\rangle} = \mathsf{EX}\ \langle \exists R.\top \sqcap \neg D \rangle$

with $\langle \mathsf{EX}\ \exists R.\top \rangle$, $\langle \neg D \rangle$, and $\langle \exists R.\top \sqcap \neg D \rangle$ being pairwise different atomic concepts. □

Regarding the mapping $\langle \cdot \rangle$ of complex onto atomic concepts we want to ensure that the interpretation of an atomic concepts $\langle C \rangle$ in a structure M remains equal to the interpretation of the original complex concept C in M as defined by the semantics of the connectives in C.

Definition 6.3.19 (Extended Interpretation)

Let $M = (S, R, \Delta, I) \in \mathbf{M}_{\mathcal{ALC}\mathsf{CTL}}$ be an $\mathcal{ALC}$CTL temporal structure and $\mathbf{C} \subseteq \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite, nonempty set of $\mathcal{ALC}$CTL concepts.

Then $I^{\mathbf{C},M}$ denotes the *extension* of I to $\mathbf{C}$ w.r.t. M iff for each $s \in S$ holds:

$\langle C \rangle^{I^{\mathbf{C},M}(s)} = (M, s)(C)$ for each $C \in \mathbf{C}$ and $\mathcal{R}^{I^{\mathbf{C},M}(s)} = \mathcal{R}^{I(s)}$ for each atomic role $\mathcal{R} \in AR$.

$M^{\mathbf{C}} := (S, R, \Delta, I^{\mathbf{C},M})$ denotes the temporal structure M with an interpretation extended to all concepts in $\mathbf{C}$. □

If an interpretation is extended to all sub-concepts $\lfloor C \rfloor$ of C, the semi-atomic mapping $C^{\langle\rangle}$ evaluates under the extended interpretation $I^{\lfloor C \rfloor, M}$ to the same set as C under the original interpretation I:

Lemma 6.3.20 (Extended Interpretation of $\mathcal{ALC}$CTL Concepts)

Let $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ be a complex $\mathcal{ALC}$CTL concept, $M = (S, R, \Delta, I) \in \mathbf{M}_{\mathcal{ALC}\mathsf{CTL}}$ an $\mathcal{ALC}$CTL temporal structure, and $s \in S$ a state. Let $M^{\lfloor C \rfloor} = (S, R, \Delta, I^{\lfloor C \rfloor, M})$ be the temporal structure M with an interpretation extended to all sub-concepts of C (Definition 6.3.19).

Then $(M^{\lfloor C \rfloor}, s)(C^{\langle\rangle}) = (M, s)(C)$.

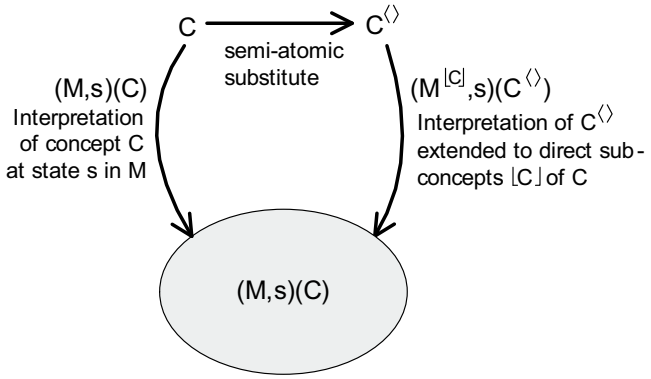

Figure 6.3.: equivalence between interpretations $(M, s)(C)$ and $(M^{\lfloor C \rfloor}, s)(C^{\langle\rangle})$

Proof:

By Definition 6.3.17, the top level connective of $C^{\langle\rangle}$ is equal to the top level connective of C but each direct sub-concept $C' \in \lfloor C \rfloor$ is replaced by $\langle C' \rangle$ in $C^{\langle\rangle}$. S, R, and Δ are equal in M and $M^{\lfloor C \rfloor}$ (Definition 6.3.19). If S, R, and Δ are fixed, the interpretation of a complex concept only depends on the interpretation of its direct sub-concepts and atomic roles (Definition 6.2.8) at all states $s \in S$. As a consequence,

$(M^{\lfloor C \rfloor}, s)(C^{\langle\rangle}) = (M, s)(C)$ for each $s \in S$ if for each state $s \in S$, concept $C' \in \lfloor C \rfloor$, and atomic role $\mathcal{R} \in AR$ holds:

$\langle C' \rangle^{I^{\lfloor C \rfloor, M}(s)} = (M, s)(C'^{I(s)})$ and $\mathcal{R}^{I^{\lfloor C \rfloor, M}(s)} = \mathcal{R}^{I(s)}$.

Exactly this is the case by Definition 6.3.19 and thus Lemma 6.3.20 is shown. □

Lemma 6.3.20 is the basis of an incremental "bottom-up" reduction of complex concepts in an $\mathcal{ALC}$CTL formula f to atomic concepts.

A support structure for showing the complexity of the atomic concept reduction of a complex concept is the *label set* of a concept.

Definition 6.3.21 (Label Set of a Concept)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure, $a \in \Delta$ a domain object, and $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ an $\mathcal{ALC}$CTL concept.

Then $LS_{a,C,M} := \{s \in S \mid a \in (M, s)(C)\}$ denotes the *label set* of C for a w.r.t. M.

$ls_{C,M} : \Delta \to \mathcal{P}(S) : ls_{C,M}(a) = LS_{a,C,M}$ denotes the mapping of domain objects onto their label set w.r.t. C and M. □

The mapping $ls_{C,M}$ is essentially an alternative representation of the interpretation of C in a structure M.

Corollary 6.3.22 (Equivalence of Label Set and Interpretation of a Concept)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure, $s \in S$ a state, and $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ an $\mathcal{ALC}$CTL concept.

Then $(M, s)(C) = \{a \in \Delta \mid s \in LS_{a,C,M}\} = \{a \in \Delta \mid s \in ls_{C,M}(a)\} =: ls^{-1}_{C,M}(s)$.

This is a direct consequence of Definitions 6.3.21 and 6.2.8. □

Hence, computing the label set $LS_{a,C,M}$ for each $a \in \Delta$ determines the interpretation of a complex concept C. In contrast to the case of $\mathcal{ALC}$CTL , the complexity of calculating such a label set for CTL formulae is known (Theorem 3.2.12). Using the structure mapping sm (Definition 6.3.4) and concept mapping $\cdot\,[a]_\Delta$ (Definition 6.3.7) an $\mathcal{ALC}$CTL label set can be mapped onto a CTL label set.

Corollary 6.3.23 (CTL Representation of a Label Set)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure, $a \in \Delta$ a domain object, $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ an $\mathcal{ALC}$CTL concept, and f an $\mathcal{ALC}$CTL formula containing C.

Let $LS_{a,C,M}$ be the label set of C for a w.r.t. M (Definition 6.3.21).

Then $LS_{a,C,M} = \{s \in S \mid sm(M, f), s \models_{\mathsf{CTL}} C[a]_\Delta\}$.

This is a direct consequence of Definition 6.3.21 and Lemma 6.3.11. □

Determining the $\mathcal{ALC}$CTL label set $LS_{a,C,M}$ by using its CTL representation as defined in Corollary 6.3.23 is inefficient because the concept mapping $C[a]_\Delta$ can result in a CTL expression of exponential size as compared to the size of C (Remark 6.3.10).

More efficiently, the $\mathcal{ALC}$CTL label set is calculated incrementally from the semi-atomic substitute $C^{\langle\rangle}$ of complex concepts since the CTL mapping $C^{\langle\rangle}[a]_\Delta$ of the semi-atomic substitute $C^{\langle\rangle}$ does not grow exponentially in the size of C.

Lemma 6.3.20 provides the ground for representing label sets based on the semi-atomic mapping of concepts:

Corollary 6.3.24 (Label Set of the Extended Interpretation)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure, $a \in \Delta$ a domain object, $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ a complex $\mathcal{ALC}$CTL concept, $C^{\langle\rangle}$ its semi-atomic substitute (Definition 6.3.17), and $M^{\lfloor C \rfloor} = (S, R, \Delta, I^{\lfloor C \rfloor, M})$ the temporal structure M with an interpretation extended to all sub-concepts of C (Definition 6.3.19).

Then as a direct consequence of Definition 6.3.21 and Lemma 6.3.20 holds:

$$LS_{a,C,M} = \{s \in S \mid a \in (M, s)C\} = \{s \in S \mid a \in (M^{\lfloor C \rfloor}, s)C^{\langle\rangle}\} = LS_{a,C^{\langle\rangle},M^{\lfloor C \rfloor}}$$

□

Joining the results of Corollaries 6.3.23 and 6.3.24, we get:

Corollary 6.3.25 (Extended CTL Representation of a Label Set)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure, $a \in \Delta$ a domain object, $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ a complex $\mathcal{ALC}$CTL concept, $C^{\langle\rangle}$ its semi-atomic substitute (Definition 6.3.17), f an $\mathcal{ALC}$CTL formula containing $C^{\langle\rangle}$, and $M^{\lfloor C \rfloor} = (S, R, \Delta, I^{\lfloor C \rfloor, M})$ the temporal structure M with an interpretation extended to all sub-concepts of C (Definition 6.3.19).

Then, as a consequence of Corollaries 6.3.24 and 6.3.23 holds:

$$LS_{a,C,M} = LS_{a,C^{\langle\rangle},M^{\lfloor C \rfloor}} = \{s \in S \mid sm(M^{\lfloor C \rfloor}, f), s \models_{\mathsf{CTL}} C^{\langle\rangle}[a]_{\Delta}\}$$

Note that this formula holds for arbitrary $\mathcal{ALC}$CTL formulae f containing $C^{\langle\rangle}$. □

This means that a label set $LS_{a,C,M}$ can be calculated from an efficient (i.e. non-exponentially large) CTL representation if the interpretation $I^{\lfloor C \rfloor, M}$ extended to all direct sub-concepts of concept C is given.

The following lemma shows an upper bound for the size of the CTL mapping of the semi-atomic substitute of C. This will be important for showing the complexity of the atomic-concept reduction of a complex concept.

Lemma 6.3.26 (Size of CTL Mapping of Semi-atomic Concepts)

Let Δ be a finite, nonempty $\mathcal{ALC}$CTL domain, $a \in \Delta$ a domain object, and $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ a complex $\mathcal{ALC}$CTL concept.

Then for the size $|C^{\langle\rangle}[a]_{\Delta}|$ of the CTL mapping of the semi-atomic substitute $C^{\langle\rangle}$ holds:

$$|C^{\langle\rangle}[a]_{\Delta}| \in \mathcal{O}(|\Delta|)$$

Proof:

Assume $C = \exists\mathcal{R}.C'$ for some atomic role $\mathcal{R} \in AR$ and concept $C' \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$.

Then $C^{\langle\rangle} = \exists\mathcal{R}.\langle C'\rangle$ with $\langle C'\rangle \in AC$ being an atomic concept and, by Definition 6.3.7,

$$\begin{aligned} C^{\langle\rangle}[a]_\Delta &= \bigvee_{b\in\Delta} (\mathcal{R}(a,b) \wedge \langle C'\rangle[b]_\Delta) \\ &= \bigvee_{b\in\Delta} (\mathcal{R}(a,b) \wedge \langle C'\rangle(b)) \end{aligned}$$

$C^{\langle\rangle}[a]_\Delta$ consists of $|\Delta| - 1$ disjunctions, $|\Delta|$ conjunctions and $2|\Delta|$ atomic propositions which sum up to $|C^{\langle\rangle}[a]_\Delta| \in \mathcal{O}(|\Delta|)$ sub-expressions.

In all other cases (i.e. $C \neq \exists\mathcal{R}.C'$)) holds by Definitions 6.3.7 and 6.3.17:

$|C^{\langle\rangle}[a]_\Delta| = |C^{\langle\rangle}| \leq 3$ and hence in all cases $|C^{\langle\rangle}[a]_\Delta| \in \mathcal{O}(|\Delta|)$. □

Using the results of Corollary 6.3.25 and Lemma 6.3.26 we can show that having knowledge about the interpretation $I^{\lfloor C\rfloor,M}$ extended to all direct sub-concepts of C w.r.t. M we can compute the interpretation of C w.r.t. M in $\mathcal{O}((|S|+|R|)\cdot|\Delta|^2)$ time.

Lemma 6.3.27 (Complexity of Extended Interpretation of $\mathcal{ALC}$CTL Concepts)

Let C be a complex $\mathcal{ALC}$CTL concept, $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ a finite $\mathcal{ALC}$CTL temporal structure, and $I^{\lfloor C\rfloor,M}$ the interpretation I extended to all direct sub-concepts $\lfloor C\rfloor$ of C. Assume that $I^{\lfloor C\rfloor,M}$ is known.

Then the interpretation $I^{\{C\},M}$ extended to C w.r.t. M can be obtained in $\mathcal{O}((|S| + |R|)\cdot|\Delta|^2)$ time.

Proof:

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite temporal structure and C a complex $\mathcal{ALC}$CTL concept.

By Definition 6.3.19, $I^{\{C\},M}$ is an interpretation such that $\langle C\rangle^{I^{\{C\},M}(s)} = (M, s)(C)$ for each $s \in S$ and $\mathcal{R}^{I^{\{C\},M}(s)} = \mathcal{R}^{I(s)}$ for each atomic role $\mathcal{R} \in AR$ and state $s \in S$. Since the interpretations of atomic roles are fixed, $I^{\{C\},M}$ is determined by the mapping $i_{C,M} : S \to \mathcal{P}(\Delta) : i_{C,M}(s) = (M, s)(C)$ of states $s \in S$ onto the respective interpretation $(M, s)(C)$ of C in M at s.

Let $ls_{C,M} : \Delta \to \mathcal{P}(S) : ls_{C,M}(a) = LS_{a,C,M}$ be the mapping of domain objects $a \in \Delta$ onto their respective label set $LS_{a,C,M} \subseteq S$ w.r.t. C in M (Definition 6.3.21). Then, by Corollary 6.3.22, it holds:

$i_{C,M}(s) = (M, s)(C) = \{a \in \Delta \mid s \in ls_{C,M}(a)\}$ for each $s \in S$.

I.e. given the mapping $ls_{C,M}$ of domain objects $a \in \Delta$ onto their respective label sets $LS_{a,C,M}$ and given that the element containment check $s \in ls_{C,M}(a)$ can be done in constant time (by using hash tables, for instance), $i_{C,M}(s)$ can be calculated in time $\mathcal{O}(|\Delta|)$ for each $s \in S$ and hence the entire mapping $i_{C,M}$ can be calculated in $\mathcal{O}(|S|\cdot|\Delta|)$ time.

As a result, $I^{\{C\},M}$ can also be obtained from $ls_{C,M}$ in $\mathcal{O}(|S|\cdot|\Delta|) \subseteq \mathcal{O}((|S| + |R|)\cdot|\Delta|^2)$ time and, as a consequence, it is sufficient to show that $ls_{C,M}$ can be constructed in $\mathcal{O}((|S| + |R|)\cdot|\Delta|^2)$ using the knowledge about $I^{\lfloor C \rfloor,M}$.

By Corollary 6.3.25, $ls_{C,M}(a) = LS_{a,C,M} = \{s \in S \mid sm(M^{\lfloor C \rfloor}, f), s \models_{\mathsf{CTL}} C^{\langle\rangle}[a]_\Delta\}$ for each $a \in \Delta$ with f being an (arbitrary) $\mathcal{ALC}$CTL formula containing $C^{\langle\rangle}$ (w.l.o.g. we can assume $f = \bot \sqsubseteq C^{\langle\rangle}$).

Hence, when $sm(M^{\lfloor C \rfloor}, f)$ is given, $LS_{a,C,M}$ can be computed in time

$$T(LS_{a,C,M}) = T(C^{\langle\rangle}[a]_\Delta) + T(\models_{\mathsf{CTL}})$$

for each $a \in \Delta$ where $T(C^{\langle\rangle}[a]_\Delta)$ is the time for computing the concept mapping $C^{\langle\rangle}[a]_\Delta$ and $T(\models_{\mathsf{CTL}})$ the total time of deciding $sm(M^{\lfloor C \rfloor}, f), s \models_{\mathsf{CTL}} C^{\langle\rangle}[a]_\Delta$ for each $s \in S$.

To compute the mapping $ls_{C,M}$ we need to determine $sm(M^{\lfloor C \rfloor}, f)$ just once and $LS_{a,C,M}$ for each $a \in \Delta$ which takes

$$\begin{aligned} T(ls_{C,M}) &= T(sm(M^{\lfloor C \rfloor}, f)) + |\Delta|\cdot T(LS_{a,C,M}) \\ &= T(sm(M^{\lfloor C \rfloor}, f)) + |\Delta|\cdot (T(C^{\langle\rangle}[a]_\Delta) + T(\models_{\mathsf{CTL}})) \end{aligned} \tag{6.7}$$

time.

According to Definition 6.3.4, $sm(M^{\lfloor C \rfloor}, f)$ maps the interpretations of every atomic concept and role in f onto a propositional labelling $L_{I^{\lfloor C \rfloor,M},f} : S \to \mathcal{P}(AP)$ such that

$$\begin{aligned} L_{I^{\lfloor C \rfloor,M},f}(s) = \; & \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f} \wedge a \in \mathcal{A}^{I^{\lfloor C \rfloor,M}(s)}\} \cup \\ & \{\mathcal{R}(a,b) \in AP \mid \mathcal{R} \in AR_{|f} \wedge (a,b) \in \mathcal{R}^{I^{\lfloor C \rfloor,M}(s)}\} \end{aligned} \tag{6.8}$$

Hence,

$$T(sm(M^{\lfloor C \rfloor}, f)) = T(L_{I^{\lfloor C \rfloor,M},f}) \tag{6.9}$$

with $T(L_{I^{\lfloor C \rfloor,M},f})$ being the time for calculating the propositional labelling function $L_{I^{\lfloor C \rfloor,M},f}$.

Note that $L_{I^{\lfloor C \rfloor,M},f}(s)$ just depends on $AC_{|f}$, $AR_{|f}$, and $\cdot^{I^{\lfloor C \rfloor,M}(s)}$. Since $\cdot^{I^{\lfloor C \rfloor,M}(s)}$ is given by assumption, the time of calculating $L_{I^{\lfloor C \rfloor,M},f}(s)$ depends on the time for determining $AC_{|f}$ and $AR_{|f}$ and the size of $L_{I^{\lfloor C \rfloor,M},f}(s)$.

For determining $AC_{|f}$ and $AR_{|f}$ we assume w.l.o.g. that $f = \bot \sqsubseteq C^{\langle\rangle}$ such that $AC_{|f} = \lfloor C^{\langle\rangle} \rfloor$, i.e. the set of atomic concepts in f is the set of direct sub-concepts of $C^{\langle\rangle}$. Since all sub-concepts of $C^{\langle\rangle}$ are atomic by Definition 6.3.17, $C^{\langle\rangle}$ contains at most two atomic concepts $\langle C_1\rangle, \langle C_2\rangle \in AC_{|f}$ and at most one atomic role $\mathcal{R} \in AR_{|f}$ by $\mathcal{ALC}$CTL syntax (Definition 6.2.1). Thus both $AC_{|f}$ and $AR_{|f}$ can be determined in constant time.

Moreover, $\mathcal{A}^{I^{\lfloor C \rfloor, M}(s)} \subseteq \Delta$ for $\mathcal{A} \in AC$ and $\mathcal{R}^{I^{\lfloor C \rfloor, M}(s)} \subseteq \Delta \times \Delta$ for $\mathcal{R} \in AR$ by Definition 6.2.8.

Thus we can approximate Equation (6.8) by

$$\begin{aligned} L_{I^{\lfloor C \rfloor, M}, f}(s) \quad \subseteq \quad & \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in \{\langle C_1\rangle, \langle C_2\rangle\} \wedge a \in \Delta\} \cup \\ & \{\mathcal{R}(a,b) \in AP \mid \mathcal{R} \in \{\mathcal{R}\} \wedge (a,b) \in \Delta \times \Delta\} \end{aligned}$$

with $\{\langle C_1\rangle, \langle C_2\rangle\}$ being the maximal set of atomic concepts in f and $\{\mathcal{R}\}$ being the maximal set of atomic roles in f. For the size of the propositional labelling $L_{I^{\lfloor C \rfloor, M}, f}(s)$ we get

$$\begin{aligned} |L_{I^{\lfloor C \rfloor, M}, f}(s)| \quad \leq \quad & |\{\mathcal{A}(a) \in AP \mid \mathcal{A} \in \{\langle C_1\rangle, \langle C_2\rangle\} \wedge a \in \Delta\}| \\ & + |\{\mathcal{R}(a,b) \in AP \mid \mathcal{R} \in \{\mathcal{R}\} \wedge (a,b) \in \Delta \times \Delta\}| \\ = \quad & 2|\Delta| + |\Delta^2| \\ \in \quad & \mathcal{O}(|\Delta|^2) \end{aligned}$$

for each $s \in S$ and, in total, $|L_{I^{\lfloor C \rfloor, M}, f}| \in \mathcal{O}(|S| \cdot |\Delta|^2)$.

Since $AC_{|f}$ and $AR_{|f}$ can be determined in constant time (see above), the time for determining the labelling $L_{I^{\lfloor C \rfloor, M}, f}$ is approximately equal to its size $|L_{I^{\lfloor C \rfloor, M}, f}|$. Hence, $T(L_{I^{\lfloor C \rfloor, M}, f})$ is in $\mathcal{O}(|L_{I^{\lfloor C \rfloor, M}, f}|) = \mathcal{O}(|S| \cdot |\Delta|^2)$ and, because of Equation 6.9,

$$T(sm(M^{\lfloor C \rfloor}, f)) \leq k \cdot |S| \cdot |\Delta|^2 \tag{6.10}$$

for some $k \in \mathbb{R}^+$ and sufficiently large sets S and Δ.

Similar to $sm(M^{\lfloor C \rfloor}, f)$, the time required for calculating the CTL mapping $C^{\langle\rangle}[a]_\Delta$ grows linearly in its size $|C^{\langle\rangle}[a]_\Delta|$ because the mapping $\cdot[a]_\Delta$ involves elementary operations only.

By Lemma 6.3.26, $|C^{\langle\rangle}[a]_\Delta| \in \mathcal{O}(|\Delta|)$ and thus

$$T(C^{\langle\rangle}[a]_\Delta) \leq k' \cdot |\Delta| \tag{6.11}$$

for some $k' \in \mathbb{R}^+$ and a sufficiently large set Δ.

If $sm(M^{\lfloor C \rfloor}, f)$ and $C^{\langle\rangle}[a]_\Delta$ are given, the label set

$$LS_{a,C,M} = \{s \in S \mid sm(M^{\lfloor C \rfloor}, f), s \models_{\mathsf{CTL}} C^{\langle\rangle}[a]_\Delta\}$$

can be calculated in

$$T(\models_{\mathsf{CTL}}) \in \mathcal{O}((|S| + |R|)\cdot |\ C^{\langle\rangle}[a]_\Delta|)$$

time (Theorem 3.2.12).

By Lemma 6.3.26, $|C^{\langle\rangle}[a]_\Delta| \in \mathcal{O}(|\Delta|)$.

As a result, we get for $T(\models_{\mathsf{CTL}})$:

$$T(\models_{\mathsf{CTL}}) \leq k''\cdot (|S| + |R|)\cdot |\Delta| \tag{6.12}$$

for some $k'' \in \mathbb{R}^+$ and sufficiently large sets S, R, and Δ.

Applying Equations (6.10), (6.11), (6.12) to Equation (6.7) we get

$$\begin{aligned} T(ls_{C,M}) &= T(sm(M^{\lfloor C\rfloor}, f)) + |\Delta|\cdot (T(C^{\langle\rangle}[a]_\Delta) + T(\models_{\mathsf{CTL}})) \\ &\leq k\cdot |S|\cdot |\Delta|^2 + |\Delta|\cdot (k'\cdot |\Delta| + k''\cdot (|S| + |R|)\cdot |\Delta|) \\ &\leq (k + k' + k'')((|S| + |R|)\cdot |\Delta|^2) \\ &\in \mathcal{O}((|S| + |R|)\cdot |\Delta|^2) \end{aligned}$$

which proves Lemma 6.3.27. □

Lemma 6.3.27 shows that calculating the interpretation of a complex concept C w.r.t. M from a given interpretation $I^{\lfloor C\rfloor,M}$ of all direct sub-concepts of C takes $\mathcal{O}((|S| + |R|)\cdot |\Delta|^2)$ time. By repeating this "lifting step" starting from the atomic concepts in a complex concept C, the interpretation $(M, s)(C)$ can be calculated in less than $|C|$ steps each of them taking $\mathcal{O}((|S| + |R|)\cdot |\Delta|^2)$. This results in the overall time complexity of $\mathcal{O}(|C|\cdot (|S| + |R|)\cdot |\Delta|^2)$ for calculating the interpretation of C from the interpretation of its atomic concepts as shown in the subsequent lemma.

Lemma 6.3.28 (Cumulative Complexity of $\mathcal{ALC}$CTL Concept Interpretation)

Let $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ be a (simple or complex) $\mathcal{ALC}$CTL concept and $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ a finite $\mathcal{ALC}$CTL temporal structure.

Then the interpretation $I^{\{C\},M}$ extended to C can be obtained from I in $\mathcal{O}(|C|\cdot (|S| + |R|)\cdot |\Delta|^2)$ time.

Proof:

Let $T(I^{\{C\},M})$ be the time complexity of determining $I^{\{C\},M}$.

We show by induction on the structure of C that there is $k \in \mathbb{R}^+$ such that

$$T(I^{\{C'\},M}) \leq k\cdot |C'|\cdot (|S| + |R|)\cdot |\Delta|^2$$

for any concept C' occurring in C.

By Lemma 6.3.27, $I^{\{C'\},M}$ can be obtained from $I^{\lfloor C' \rfloor,M}$ in $\mathcal{O}((|S| + |R|)\cdot |\Delta|^2)$.

For a complex concept C' occurring in C let $T(I^{\lfloor C' \rfloor,M} \rightarrow I^{\{C'\},M})$ denote the time for determining $I^{\{C'\},M}$ from $I^{\lfloor C' \rfloor,M}$.

Since C contains finitely many complex concepts C' and $T(I^{\lfloor C' \rfloor,M} \rightarrow I^{\{C'\},M}) \in \mathcal{O}((|S| + |R|)\cdot |\Delta|^2)$ for each complex concept C' in C there is $k \in \mathbb{R}^+$ such that $T(I^{\lfloor C' \rfloor,M} \rightarrow I^{\{C'\},M}) \leq k\cdot (|S| + |R|)\cdot |\Delta|^2$ for any complex concept C' in C. W.l.o.g. we can assume $k > 1$.

Let C' be a (simple or complex) concept in C.

- Let $|C'| = 1$. Then $C' \in AC \cup \{\top, \bot\}$.

  $I^{\{C'\},M}$ is determined by the set $\langle C' \rangle^{I^{\{C'\},M}(s)}$ for each $s \in S$. For $C' \in AC \cup \{\top, \bot\}$ we have by Definitions 6.3.19 and 6.2.8

  $$\langle C' \rangle^{I^{\{C'\},M}(s)} = (M, s)(C') = C'^{I(s)}$$

  $C'^{I(s)}$ is given by definition of M. Hence, $I^{\{C'\},M}$ can, for $|C'| = 1$, be determined without any calculation and thus:

  $$T(I^{\{C'\},M}) < k\cdot (|S| + |R|)\cdot |\Delta|^2$$

- Let $|C'| > 1$. Then C' is a complex concept and, by Lemma 6.3.27, $I^{\{C'\},M}$ can be obtained from $I^{\lfloor C' \rfloor,M}$ in $\mathcal{O}((|S| + |R|)\cdot |\Delta|^2)$. By choice of k it holds:

  $$T(I^{\lfloor C' \rfloor,M} \rightarrow I^{\{C'\},M}) \leq k\cdot (|S| + |R|)\cdot |\Delta|^2$$

  By $\mathcal{ALC}$CTL syntax (Definition 6.2.1), $\lfloor C' \rfloor$ contains one or two concepts.

  Assume $\lfloor C' \rfloor = \{C_1\}$. Then $|C'| = |C_1| + n$ with $n = 2$ if $C' = \exists R.C_1$ and $n = 1$ in the remaining cases $C' = \neg C_1, C' = \mathsf{EX}\, C_1$, or $C' = \mathsf{AX}\, C_1$.

  Since C_1 is a concept occurring in C' and thus also in C, and, in addition, $|C_1| < |C'|$, we can apply the induction hypothesis and get

  $$\begin{aligned} T(I^{\lfloor C' \rfloor,M}) = T(I^{\{C_1\},M}) &\leq k\cdot |C_1|\cdot (|S| + |R|)\cdot |\Delta|^2) \\ &= k\cdot (|C'| - n)\cdot (|S| + |R|)\cdot |\Delta|^2 \end{aligned}$$

  because $|C_1| = |C'| - n$.

  By choice of k, $I^{\{C'\},M}$ can be obtained from $I^{\lfloor C' \rfloor,M}$ in time $T(I^{\lfloor C' \rfloor,M} \rightarrow I^{\{C'\},M}) \leq k\cdot (|S| + |R|)\cdot |\Delta|^2$ and hence we get for $T(I^{\{C'\},M})$ in total:

  $$\begin{aligned} T(I^{\{C'\},M}) &= T(I^{\lfloor C' \rfloor,M}) + T(I^{\lfloor C' \rfloor,M} \rightarrow I^{\{C'\},M}) \\ &\leq k\cdot (|C'| - n)\cdot (|S| + |R|)\cdot |\Delta|^2 + k\cdot (|S| + |R|)\cdot |\Delta|^2 \\ &= k\cdot (|C'| - n + 1)\cdot (|S| + |R|)\cdot |\Delta|^2 \\ &\leq k\cdot |C'|\cdot (|S| + |R|)\cdot |\Delta|^2 \end{aligned}$$

Assume $\lfloor C' \rfloor = \{C_1, C_2\}$. Then $I^{\lfloor C' \rfloor, M} = I^{\{C_1, C_2\}, M}$ and $|C'| = |C_1| + |C_2| + 1$.

By Definition 6.3.19, $I^{\{C_1, C_2\}, M}$ is determined by $I^{\{C_1\}, M}$ and $I^{\{C_2\}, M}$. Hence, $T(I^{\{C_1, C_2\}, M}) = T(I^{\{C_1\}, M}) + T(I^{\{C_2\}, M})$.

Since $|C_1| < |C'|$ and $|C_2| < |C'|$ and C_1, C_2 are both concepts occurring in C we can assume by the induction hypothesis that

$$\begin{aligned} T(I^{\{C_1\}, M}) &\leq k \cdot |C_1| \cdot (|S| + |R|) \cdot |\Delta|^2 \\ T(I^{\{C_2\}, M}) &\leq k \cdot |C_2| \cdot (|S| + |R|) \cdot |\Delta|^2 \end{aligned}$$

and, as a consequence,

$$\begin{aligned} T(I^{\lfloor C' \rfloor, M}) &= T(I^{\{C_1, C_2\}, M}) \\ &= T(I^{\{C_1\}, M}) + T(I^{\{C_2\}, M}) \\ &\leq k \cdot (|C_1| + |C_2|) \cdot (|S| + |R|) \cdot |\Delta|^2 \end{aligned}$$

In total we get for $T(I^{\{C'\}, M})$:

$$\begin{aligned} T(I^{\{C'\}, M}) &= T(I^{\lfloor C' \rfloor, M}) + T(I^{\lfloor C' \rfloor, M} \to I^{\{C'\}, M}) \\ &\leq k \cdot (|C_1| + |C_2|) \cdot (|S| + |R|) \cdot |\Delta|^2 + k \cdot (|S| + |R|) \cdot |\Delta|^2 \\ &= k \cdot (|C_1| + |C_2| + 1) \cdot (|S| + |R|) \cdot |\Delta|^2 \\ &= k \cdot |C'| \cdot (|S| + |R|) \cdot |\Delta|^2 \end{aligned}$$

We have shown by induction on the structure of the $\mathcal{ALC}$CTL concept C that for any concept C' occurring in C and thus also for $C' = C$ holds:

$T(I^{\{C'\}, M}) \leq k \cdot |C'| \cdot (|S| + |R|) \cdot |\Delta|^2$.

Consequently, it holds in general that $T(I^{\{C\}, M}) \in \mathcal{O}(|C| \cdot (|S| + |R|) \cdot |\Delta|^2)$. □

Lemma 6.3.28 shows that the interpretation extended to complex concepts can be calculated from the interpretation of atomic concepts in $\mathcal{O}(|C| \cdot (|S| + |R|) \cdot |\Delta|^2)$ time. f can be treated as containing atomic concepts only if the interpretation of all complex concepts in an $\mathcal{ALC}$CTL formula f w.r.t. an $\mathcal{ALC}$CTL structure M is given.

More formally, the formula f and the structure M can easily be mapped onto f' and M' such that f' contains atomic concepts only and f' is satisfied in the same set of states w.r.t. M' as f does w.r.t. M, i.e. the label sets $LS_{f,M}$ and $LS_{M',f'}$ (Definition 6.3.2) are identical.

In the sequel, we assume that all concepts in an $\mathcal{ALC}$CTL formula f are atomic and examine the time complexity of the respective model checking problem.

Proposition 6.3.29 (Model Checking $\mathcal{ALC}$CTL Restricted to Atomic Concepts)

Let $f \in \mathcal{ALC}\mathsf{CTL}$ be an $\mathcal{ALC}\mathsf{CTL}$ formula such that all concepts in f are atomic. Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}\mathsf{CTL}$ temporal structure.

Then the label set $LS_{M,f} = \{s \in S \mid M, s \models f\}$ of f for M (Definition 6.3.2) can be computed in $\mathcal{O}(|f|\cdot(|S| + |R|)\cdot|\Delta|)$ time.

Proof:

As a consequence of Proposition 6.3.12 it holds that

$$\begin{aligned} LS_{M,f} &= \{s \in S \mid M, s \models f\} \\ &= \{s \in S \mid sm(M,f), s \models_{\mathsf{CTL}} fm(M,f)\} \end{aligned}$$

Let $T(sm(M,f))$ be the time for determining $sm(M,f)$, let $T(fm(M,f))$ be the time for calculating $fm(M,f)$, and let $T(\models_{\mathsf{CTL}})$ be the time for calculating $\{s \in S \mid sm(M,f), s \models_{\mathsf{CTL}} fm(M,f)\}$ from given mappings $sm(M,f)$ and $fm(M,f)$.

Then $LS_{M,f}$ can be calculated in time

$$T(LS_{M,f}) = T(sm(M,f)) + T(fm(M,f)) + T(\models_{\mathsf{CTL}}) \tag{6.13}$$

By Definition 6.3.4, $sm(M,f) = (S, R, L_{I,f})$ with

$$\begin{aligned} L_{I,f}(s) &= \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f} \wedge a \in \mathcal{A}^{I(s)}\} \cup \\ &\quad \{\mathcal{R}(a,b) \in AP \mid \mathcal{R} \in AR_{|f} \wedge (a,b) \in \mathcal{R}^{I(s)}\} \end{aligned}$$

Since all concepts in f are atomic and roles occur in complex concepts only, no roles occur in f and hence $AR_{|f} = \emptyset$. As a result,

$$L_{I,f}(s) = \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f} \wedge a \in \mathcal{A}^{I(s)}\}$$

Since $|\mathcal{A}^{I(s)}| \leq |\Delta|$ as a consequence of Definition 6.2.8 and $|AC_{|f}| \leq |f|$ we get $|L_{I,f}(s)| \leq |f|\cdot|\Delta|$ for each $s \in S$ and, in total, $|L_{I,f}| \leq |f|\cdot|S|\cdot|\Delta|$.

Since $L_{I,f}$ can directly be obtained from I and f, the time complexity $T(L_{I,f})$ of calculating $L_{I,f}$ is approximately equal to the size of $L_{I,f}$ and hence

$$T(sm(M,f)) = T(L_{I,f}) \in \mathcal{O}(|L_{I,f}|) = \mathcal{O}(|f|\cdot|S|\cdot|\Delta|) \subseteq \mathcal{O}(|f|\cdot(|S|+|R|)\cdot|\Delta|) \tag{6.14}$$

By Definition 6.3.7, for any sub-formula f' in f holds that $fm(M,f')$ does not change the top level connective in f' in all cases but $f' = C \sqsubseteq D$ with C, D being atomic concepts in f. In the latter case

$$fm(M,f') = \bigwedge_{a \in \Delta} (C[a]_\Delta \rightarrow D[a]_\Delta) = \bigwedge_{a \in \Delta} (C(a) \rightarrow D(a))$$

$fm(M, f')$ contains $|\Delta - 1|$ conjunctions, $|\Delta|$ implications and $2|\Delta|$ atomic propositions and hence $|fm(M, f')| \in \mathcal{O}(|\Delta|)$.

f contains less than $|f|$ sub-formulae of type $C \sqsubseteq D$, each of them being mapped by $fm(M, f)$ onto Boolean expressions the length of which each is in $\mathcal{O}(|\Delta|)$. Hence, the total length of the mappings of all sub-formulae of kind $C \sqsubseteq D$ in f is in $\mathcal{O}(|f| \cdot |\Delta|)$.

Since all connectives other than $\sqsubseteq$ remain unaffected by the formula mapping $fm(M, f)$ of f and there are less than $|f|$ sub-formulae built on connectives other than $\sqsubseteq$ we get

$$|fm(M, f)| \in \mathcal{O}(|f| \cdot |\Delta| + |f|) = \mathcal{O}(|f| \cdot |\Delta|)$$

Calculating $fm(M, f)$ requires a single traversal of the expression tree of f. Further, $fm(M, f)$ results in a CTL formula that is larger than the corresponding $\mathcal{ALC}$CTL formula f. Hence the time for calculating $fm(M, f)$ grows linearly with the output size $|fm(M, f)|$ and we get:

$$T(fm(M, f)) \in \mathcal{O}(|f| \cdot |\Delta|) \subseteq \mathcal{O}(|f| \cdot (|S| + |R|) \cdot |\Delta|) \tag{6.15}$$

Given $sm(M, f)$ and $fm(M, f)$, the time $T(\models_{\mathsf{CTL}})$ required for calculating the set $LS_{M,f} = \{s \in S \mid sm(M, f), s \models_{\mathsf{CTL}} fm(M, f)\}$ is in $\mathcal{O}(|fm(M, f)| \cdot (|S| + |R|))$ (Theorem 3.2.12). Since $|fm(M, f)| \in \mathcal{O}(|f| \cdot |\Delta|)$ we get

$$T(\models_{\mathsf{CTL}}) \in \mathcal{O}(|f| \cdot |\Delta| \cdot (|S| + |R|)) \tag{6.16}$$

Combining Equations (6.13), (6.14), (6.15), and (6.16) we have proven that

$$T(LS_{M,f}) \in \mathcal{O}(|f| \cdot (|S| + |R|) \cdot |\Delta|)$$

□

In the sequel, we introduce the required formal concepts for reducing an $\mathcal{ALC}$CTL formula f to an equivalent formula f' without any complex concepts. Using Lemma 6.3.28 and Proposition 6.3.29, we then show that any $\mathcal{ALC}$CTL formula can be model checked in $\mathcal{O}(|f| \cdot (|S| + |R|) \cdot |\Delta|^2)$ time.

Definition 6.3.30 (Elimination of Complex Concepts in $\mathcal{ALC}$CTL Formulae)

Let $f \in \mathcal{ALC}$CTL be an $\mathcal{ALC}$CTL formula.

Then $\mathcal{C}_f := \{C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}} \mid C \sqsubseteq D$ occurs in f or $D \sqsubseteq C$ occurs in f for some concept $D \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}\}$ denotes the set of *top level concepts occurring in* f. Since f is finite, $\mathcal{C}_f$ is finite as well.

$f^{\langle\rangle} \in \mathcal{ALC}$CTL denotes the *atomic concept mapping* of f which is defined as follows. Let $\{C_1, ..., C_n\} := \mathcal{C}_f$. Then

$$f^{\langle\rangle} := f[C_1/\langle C_1\rangle][C_2/\langle C_2\rangle]...[C_n/\langle C_n\rangle]$$

where $f[x/y]$ denotes f with each occurrence of term x being substituted by term y.

□

Remark 6.3.31 (Elimination of Complex Concepts)

By Definitions 6.3.17 and 6.2.1, $f^{\langle\rangle}$ is a formula containing atomic concepts only. □

Example 6.3.32 (Elimination of Complex Concepts)

Consider the $\mathcal{ALCC}$CTL formulae

$f_1 = \mathsf{AG}\ (\neg C \sqsubseteq \mathsf{EF}\ \exists R.D)$ and
$f_2 = \mathsf{A}((C \sqsubseteq \bot)\ \mathsf{U}\ \mathsf{AG}(\neg C \sqsubseteq \mathsf{EF}\ \exists R.D))$.

Then

$$\begin{aligned} \mathcal{C}_{f_1} &= \{\neg C, \mathsf{EF}\ \exists R.D\} \\ \mathcal{C}_{f_2} &= \{C, \bot, \neg C, \mathsf{EF}\ \exists R.D\} \end{aligned}$$

and

$$\begin{aligned} f_1^{\langle\rangle} &= \mathsf{AG}(\langle\neg C\rangle \sqsubseteq \langle\mathsf{EF}\ \exists R.D\rangle) \\ f_2^{\langle\rangle} &= \mathsf{A}((\langle C\rangle \sqsubseteq \langle\bot\rangle)\ \mathsf{U}\ \mathsf{AG}(\langle\neg C\rangle \sqsubseteq \langle\mathsf{EF}\ \exists R.D\rangle)) \end{aligned}$$

□

For the atomic concept mapping $f^{\langle\rangle}$ of an $\mathcal{ALCC}$CTL formula f we require that $f^{\langle\rangle}$ holds in a respective structure $M^{\mathcal{C}_f}$ at the same states as f does in M. This is the case as shown in the following lemma:

Lemma 6.3.33 (Elimination of Complex Concepts)

Let f be an $\mathcal{ALCC}$CTL formula, $M = (S, R, \Delta, I) \in \mathbf{M}_{\mathcal{ALCC}\mathsf{CTL}}$ an $\mathcal{ALCC}$CTL temporal structure, and $s \in S$ a state. Let $M^{\mathcal{C}_f} = (S, R, \Delta, I^{\mathcal{C}_f, M})$ be the temporal structure M with I being extended to all top level concepts $\mathcal{C}_f$ of f (Definition 6.3.19).

Then $M, s \models f \Leftrightarrow M^{\mathcal{C}_f}, s \models\ f^{\langle\rangle}$.

Proof:

f and $f^{\langle\rangle}$ are equal except for each expression $C \sqsubseteq D$ in f being replaced by $\langle C\rangle \sqsubseteq \langle D\rangle$ in $f^{\langle\rangle}$.

Let $C, D \in \mathcal{C}_f$ be top level concepts in f and $s \in S$ a state.

Then, by Definition 6.3.19, it holds that

$$\begin{aligned} &(M^{\mathcal{C}_f}, s)(\langle C\rangle) = \langle C\rangle^{I^{\mathcal{C}_f, M}(s)} = (M, s)(C) \quad \text{and} \\ &(M^{\mathcal{C}_f}, s)(\langle D\rangle) = \langle D\rangle^{I^{\mathcal{C}_f, M}(s)} = (M, s)(D) \end{aligned}$$

By the semantics of "$\sqsubseteq$" (Definition 6.2.8), we have

$$M^{\mathcal{C}_f}, s \models \langle C \rangle \sqsubseteq \langle D \rangle \Leftrightarrow M, s \models C \sqsubseteq D$$

for each formula $C \sqsubseteq D$ in f.

Since $M^{\mathcal{C}_f}$ and M share the same state transition system (S, R) and the sub-formula structures of $f^{\langle\rangle}$ and f coincide, we get by induction on the structure of $f^{\langle\rangle}$:

$$M^{\mathcal{C}_f}, s \models f^{\langle\rangle} \Leftrightarrow M, s \models f$$

□

Lemma 6.3.33 shows that $M^{\mathcal{C}_f}$ and $f^{\langle\rangle}$ are a validity-equivalent representation of M, f with $f^{\langle\rangle}$ containing atomic concepts only.

Corollary 6.3.34 (Equivalence with CTL Mapping)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure and f an $\mathcal{ALC}$CTL formula.

Then

$$M, s \models_{\mathcal{ALC}\mathsf{CTL}} f \Leftrightarrow sm(M^{\mathcal{C}_f},\ f^{\langle\rangle}), s \models_{\mathsf{CTL}} fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$$

for each $s \in S$.

This a direct consequence of the combination of Lemma 6.3.33 which says that

$$M^{\mathcal{C}_f}, s \models_{\mathcal{ALC}\mathsf{CTL}} f^{\langle\rangle} \Leftrightarrow M, s \models_{\mathcal{ALC}\mathsf{CTL}} f \text{ for each } s \in S$$

and Proposition 6.3.12 stating that

$$M^{\mathcal{C}_f}, s \models_{\mathcal{ALC}\mathsf{CTL}} f^{\langle\rangle} \Leftrightarrow sm(M^{\mathcal{C}_f},\ f^{\langle\rangle}), s \models_{\mathsf{CTL}} fm(M^{\mathcal{C}_f},\ f^{\langle\rangle}) \text{ for each } s \in S$$

□

Using the results of Lemma 6.3.33, Lemma 6.3.28, and Proposition 6.3.29 we can show:

Proposition 6.3.35 (Complexity of Model Checking $\mathcal{ALC}$CTL Formulae)

Let f be an $\mathcal{ALC}$CTL formula and $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ a finite $\mathcal{ALC}$CTL temporal structure.

Then the label set $LS_{M,f} = \{s \in S \mid M, s \models f\}$ of f for M can be computed in $\mathcal{O}(|f| \cdot (|S| + |R|) \cdot |\Delta|^2)$ time.

Proof:

As a consequence of Lemma 6.3.33 we have

$$LS_{M,f} = \{s \in S \mid M, s \models f\} = \{s \in S \mid M^{\mathcal{C}_f}, s \models\ f^{\langle\rangle}\} = LS_{M^{\mathcal{C}_f},\, f^{\langle\rangle}}$$

Hence, $LS_{M,f}$ can be computed by computing $M^{\mathcal{C}_f}$ from M and f, $f^{\langle\rangle}$ from f, and $LS_{M^{\mathcal{C}_f},\, f^{\langle\rangle}}$ from $M^{\mathcal{C}_f}$ and $f^{\langle\rangle}$. The respective computation cost sum up to $T(LS_{M,f})$, $T(M^{\mathcal{C}_f})$, $T(f^{\langle\rangle})$, and $T(LS_{M^{\mathcal{C}_f},\, f^{\langle\rangle}})$ and it holds:

$$T(LS_{M,f}) = T(M^{\mathcal{C}_f}) + T(f^{\langle\rangle}) + T(LS_{M^{\mathcal{C}_f},\, f^{\langle\rangle}}) \tag{6.17}$$

Let $\{C_1, ..., C_n\} = \mathcal{C}_f$ for some $n \in \{0, ..., |f|\}$.

By Lemma 6.3.28, $I^{\{C'\},M}$ can be obtained from I in $T(I^{\{C'\},M}) \in \mathcal{O}(|C'|\cdot(|S| + |R|)\cdot|\Delta|^2)$ time for each $C' \in \mathcal{C}_f$. Since $\mathcal{C}_f$ is finite there is $k \in \mathbb{R}^+$ such that $T(I^{\{C'\},M}) \leq k\cdot|C'|\cdot(|S| + |R|)\cdot|\Delta|^2$ for each $C' \in \mathcal{C}_f$.

By Definition 6.3.19, $I^{\mathcal{C}_f,M}$ is directly determined by $\{I^{\{C'\},M} \mid C' \in \mathcal{C}_f\}$ and thus:

$$\begin{aligned} T(I^{\mathcal{C}_f,M}) &= \sum_{C' \in \mathcal{C}_f} T(I^{\{C'\},M}) \\ &\leq \sum_{C' \in \mathcal{C}_f} k\cdot|C'|\cdot(|S| + |R|)\cdot|\Delta|^2 \\ &= k\cdot(|S| + |R|)\cdot|\Delta|^2\cdot \sum_{C' \in \mathcal{C}_f} |C'| \\ &\leq k\cdot(|S| + |R|)\cdot|\Delta|^2\cdot|f| \end{aligned}$$

because $\sum_{C' \in \mathcal{C}_f} |C'| \leq |f|$ since each $C' \in \mathcal{C}_f$ is a concept in f by Definition 6.3.30.

$M^{\mathcal{C}_f}$ can be obtained from M in $T(M^{\mathcal{C}_f}) = T(I^{\mathcal{C}_f,M})$ because M and $M^{\mathcal{C}_f}$ coincide in S, R, and Δ. As a result,

$$T(M^{\mathcal{C}_f}) \leq k\cdot|f|\cdot(|S| + |R|)\cdot|\Delta|^2 \tag{6.18}$$

$f^{\langle\rangle}$ can be obtained from f in

$$T(f^{\langle\rangle}) = k'\cdot|f| \tag{6.19}$$

time with $k' \in \mathbb{R}^+$ since $f^{\langle\rangle}$ is a simple term substitution (Definition 6.3.30).

By Proposition 6.3.29 and Remark 6.3.31, $LS_{M^{\mathcal{C}_f},\, f^{\langle\rangle}}$ can be obtained in $T(LS_{M^{\mathcal{C}_f},\, f^{\langle\rangle}}) \in \mathcal{O}(|f^{\langle\rangle}|\cdot(|S| + |R|)\cdot|\Delta|)$ time and hence there is $k'' \in \mathbb{R}^+$ such that

$$\begin{aligned} T(LS_{M^{\mathcal{C}_f},\, f^{\langle\rangle}}) &\leq k''\cdot|f^{\langle\rangle}|\cdot(|S| + |R|)\cdot|\Delta| \\ &\leq k''\cdot|f|\cdot(|S| + |R|)\cdot|\Delta| \end{aligned} \tag{6.20}$$

because $|f^{\langle\rangle}| \leq |f|$.

Applying Equations (6.18), (6.19), and (6.19) in Equation (6.17) we get

$$\begin{aligned} T(LS_{M,f}) &= T(M^{\mathcal{C}_f}) + T(f^{\langle\rangle}) + T(LS_{M^{\mathcal{C}_f},\, f^{\langle\rangle}}) \\ &\leq k\cdot|f|\cdot(|S|+|R|)\cdot|\Delta|^2 + k'\cdot|f| + k''\cdot|f|\cdot(|S|+|R|)\cdot|\Delta| \\ &\leq (k+k'+k'')\cdot|f|\cdot(|S|+|R|)\cdot|\Delta|^2 \end{aligned}$$

since $|S| \geq 1$ and $|\Delta| \geq 1$ by Definition 6.2.6 and, as a result,

$$T(LS_{M,f}) \in \mathcal{O}(|f|\cdot(|S|+|R|)\cdot|\Delta|^2)$$

□

Remark 6.3.36 (Complexity of Model Checking $\mathcal{ALC}$CTL Formulae)
The proof of Proposition 6.3.35 also shows the validity of Proposition 6.3.14.
The moderate polynomial complexity of the $\mathcal{ALC}$CTL model checking problem gives rise to the assumption that verification based on $\mathcal{ALC}$CTL model checking scales to application relevant problem sizes. As compared to model checking CTL, which is in $\mathcal{O}(|f|\cdot(|S|+|R|))$ (Theorem 3.2.12), model checking $\mathcal{ALC}$CTL is more expensive by the factor $|\Delta|^2$. This is not surprising since $\mathcal{ALC}$CTL contains roles that are interpreted as subsets of $\Delta \times \Delta$ in every state $s \in S$, i.e. the input size of the model checking problem is already in $\mathcal{O}(|S|\cdot|\Delta|^2)$.
For determining the runtime complexity $\mathcal{O}(|f|\cdot(|S|+|R|)\cdot|\Delta|^2)$ of $\mathcal{ALC}$CTL model checking, the worst case assumption is applied. In many application scenarios, a better scaling of runtime in the problem size can be achieved (see evaluation results in section 7.5). □

6.3.3. Model Checking Algorithm

The analysis of the runtime complexity of the $\mathcal{ALC}$CTL model checking problem in the previous section provides the foundations for a sound, complete, and efficient model checking algorithm.

Recall that the $\mathcal{ALC}$CTL model checking problem is defined in Definition 6.3.2 as: given a finite $\mathcal{ALC}$CTL model $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ and a formula $f \in \mathcal{ALC}$CTL, calculate the label set $LS_{f,M} = \{s \in S \mid M, s \models f\}$.

The general structure of the $\mathcal{ALC}$CTL model checking algorithm closely resembles the proof structure for the runtime complexity of computing $LS_{f,M}$ in the previous section:

1. Successively extend the interpretation I of atomic concepts in f to an interpretation $I^{\mathcal{C}_f,M}$ of the top-level concepts $\mathcal{C}_f$ w.r.t. M by a mapping onto an equivalent CTL model checking problem.

2. Reduce all top-level concepts in f to atomic concepts in $f^{\langle\rangle}$ and generate a CTL representation $f_{\mathsf{CTL}} := fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ of $f^{\langle\rangle}$ w.r.t. the structure $M^{\mathcal{C}_f} = (S, R, \Delta, I^{\mathcal{C}_f,M})$.

3. Reduce the extended temporal structure $M^{\mathcal{C}_f}$ to a CTL structure $M_{\mathsf{CTL}} := sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$.

4. Determine $LS_{M,f} = \{s \in S \mid M_{\mathsf{CTL}}, s \vdash_{\mathsf{CTL}} f_{\mathsf{CTL}}\}$ using a CTL model checking algorithm $\vdash_{\mathsf{CTL}}$.

The algorithm consists of two major functions:

- **verify**: takes a finite $\mathcal{ALC}$CTL structure $M = (S, R, \Delta, I)$ and an $\mathcal{ALC}$CTL formula f as input and returns the set $LS_{M,f} = \{s \in S \mid M, s \models f\}$ of states at which formula f holds in M.

- **getInterpretation**: takes a finite $\mathcal{ALC}$CTL structure $M = (S, R, \Delta, I)$ and an $\mathcal{ALC}$CTL concept C as input and returns a representation of the interpretation $I^{\{C\},M}$ extended to concept C w.r.t. M.

6.3.3.1. Algorithm for Checking Formulae

Algorithm 6.3.37 ($\mathcal{ALC}$CTL Model Checking)

The following pseudo code sketches the function **verify** that is the main function of the $\mathcal{ALC}$CTL model checking algorithm.

function verify(S, R, Δ, I, f) {	
$M := (S, R, \Delta, I)$;	For a more compact access to structure M.
$\mathcal{C}_f :=$ f.getTopLevelConcepts();	Initialize the set $\mathcal{C}_f$ of top level concepts in f.
$I^{\mathcal{C}_f,M} := \emptyset$;	Initialize the extended interpretation $I^{\mathcal{C}_f,M}$ of concepts $\mathcal{C}_f$ w.r.t. M.
for each $C' \in \mathcal{C}_f$ {	Construct the extended interpretation $I^{\mathcal{C}_f,M}$.
$i_{C',M} :=$ getInterpretation(M, C');	Get the interpretation $i_{C',M} : S \to \mathcal{P}(\Delta) : i_{C',M}(s) = (M, s)(C')$ of concept C' at all states in M.
$I^{\mathcal{C}_f,M} := I^{\mathcal{C}_f,M} \cup \{\langle C' \rangle \mapsto i_{C',M}\}$;	Add a mapping of the atomic substitute $\langle C' \rangle$ of C' onto its interpretation $i_{C',M}$ in M to $I^{\mathcal{C}_f,M}$.
}	
$M^{\mathcal{C}_f} := (S, R, \Delta, I^{\mathcal{C}_f,M})$;	Build a temporal structure $M^{\mathcal{C}_f}$ with an interpretation $I^{\mathcal{C}_f,M}$ extended to all top level concepts $\mathcal{C}_f$ of formula f.
$f' := fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$;	Translate f to its CTL representation f' according to the definition of $fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$.
$M' := sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$;	Generate a CTL structure M' according to the definition of $sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$.
return $\{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\}$	Model check f' for M' using $\vdash_{\mathsf{CTL}}$.
}	

□

verify uses the following sub-routines:

- f.getTopLevelConcepts();

  returns the set of top-level concepts $\mathcal{C}_f$ in the $\mathcal{ALC}$CTL formula f as defined in Definition 6.3.30.

- getInterpretation(M, C');

  returns a mapping $i_{C',M} : S \to \mathcal{P}(\Delta) : i_{C',M}(s) = (M, s)(C')$ of states $s \in S$ onto the interpretation of $(M, s)(C')$ of concept C' in M at s. By Definition 6.3.19, $(M, s)(C')$ is equal to $\langle C' \rangle^{I^{\{C'\},M}(s)}$.

  $i_{C',M}$ is a sufficient representation of the interpretation $I^{\{C'\},M}$ in the context of **verify**.

  Recall, $I^{\{C'\},M}$ is defined as an interpretation such that for each $s \in S$: $\langle C' \rangle^{I^{\{C'\},M}(s)} = (M, s)(C')$ and $\mathcal{R}^{I^{\{C'\},M}(s)} = \mathcal{R}^{I(s)}$ for each atomic role $\mathcal{R} \in AR$ (Definition 6.3.19).

  Since the interpretation of roles is not referred to within **verify** it is sufficient for the representation of $I^{\{C'\},M}$ to calculate the mapping $i_{C',M} : S \to \mathcal{P}(\Delta) : i_{C',M}(s) = (M, s)(C')$.

- $I^{\mathcal{C}_f,M} := I^{\mathcal{C}_f,M} \cup \{\langle C' \rangle \mapsto i_{C',M}\}$;

  this adds a new mapping of the atomic substitute $\langle C' \rangle$ of concept C' onto its temporal interpretation function $i_{C',M} : S \to \mathcal{P}(\Delta) : i_{C',M}(s) = (M,s)(C')$ to $I^{\mathcal{C}_f,M}$.

  Let $\langle \mathcal{C}_f \rangle := \{\langle C' \rangle \mid C' \in \mathcal{C}_f\}$.

  Then note that within **verify** $I^{\mathcal{C}_f,M}$ is represented as a mapping of $\langle \mathcal{C}_f \rangle \to \mathcal{P}(S \to \mathcal{P}(\Delta)) : (I^{\mathcal{C}_f,M}(\langle C' \rangle))(s) = (M,s)(C')$ whereas formally $I^{\mathcal{C}_f,M}$ defines a mapping $S \to \mathcal{P}(\langle \mathcal{C}_f \rangle \to \mathcal{P}(\Delta)) : \langle C' \rangle^{I^{\mathcal{C}_f,M}(s)} = (M,s)(C')$.

  Both representations are equivalent since both associate the atomic substitute $\langle C \rangle$ of every concept $C' \in \mathcal{C}_f$ and every state $s \in S$ with the interpretation $(M,s)(C')$ of C' in M at state s.

  The former representation of $I^{\mathcal{C}_f,M}$ has been chosen because it better suits the structure of the **verify** algorithm.

- $fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$;

  returns the CTL representation of $f^{\langle\rangle}$ w.r.t. $M^{\mathcal{C}_f}$ according to Definitions 6.3.7, 6.3.19, and 6.3.30. Recall, $f^{\langle\rangle}$ is an $\mathcal{ALC}$CTL formula f the top-level concepts of which are treated as atomic (Definition 6.3.30) and $M^{\mathcal{C}_f}$ is the temporal structure M the interpretation of which is extended to all top-level concepts $\mathcal{C}_f$ of formula f (Definition 6.3.19).

- $sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$;

  returns the CTL representation of the part of the temporal structure $M^{\mathcal{C}_f}$ that is relevant for determining the validity of $f^{\langle\rangle}$ according to Definition 6.3.4. Recall, $sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ is equal to $M^{\mathcal{C}_f}$ except that the interpretation of atomic concepts in $f^{\langle\rangle}$ (i.e. the interpretation $I^{\mathcal{C}_f,M}$ of the top-level concepts $\mathcal{C}_f$ of f) are encoded in terms of a labelling $L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}} : S \to \mathcal{P}(AP)$ of states $s \in S$ with sets of atomic propositions $L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}}(s) \subseteq AP$ which hold at s.

- $M', s \vdash_{\mathsf{CTL}} f'$;

  $\vdash_{\mathsf{CTL}}$ denotes a sound and complete algorithm for deciding $M', s \models_{\mathsf{CTL}} f'$. It checks whether the CTL formula f' holds in the finite CTL temporal structure M' at state s.

Example 6.3.38 (Algorithm verify)

Consider a slightly simplified version of the $\mathcal{ALC}$CTL temporal structure of Example 6.2.7: $M = (S, R, \Delta, I)$ where

- $S := \{s_0, s_1, s_2\}$;
- $R := \{(s_0, s_1), (s_0, s_2), (s_1, s_2), (s_2, s_2)\} \subseteq S \times S$;
- $\Delta :=$ {*L1, L2, R1, R2*}.
- I is an interpretation of atomic concepts AC as follows

MajorTopic$^{I(s_0)}$	$=$	{*R1, R2*}	Topic *R180*, abbreviated as $R1$, and topic *R180_Handling*, abbreviated as $R2$ are the major topics in state s.
MajorTopic$^{I(s_1)}$	$=$	{*R1*}	*R180* is the (only) major topic in state s_1.
MajorTopic$^{I(s_2)}$	$=$	{*R2*}	*R180_Handling* is the (only) major topic in state s_2.

and I is an interpretation of atomic roles AR as follows

addressedBy$^{I(s_0)}$	$=$	{*(R1,L1)*}	*R180* is addressed by lesson *L1* in state s.
addressedBy$^{I(s_1)}$	$=$	$\emptyset$	There is no topic addressed in state s_1.
addressedBy$^{I(s_2)}$	$=$	{*(R2,L2)*}	*R180_Handling* is addressed by lesson *L2* in state s_2.

Further, consider the $\mathcal{ALC}$CTL formula $MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.\top$ - every major topic is eventually addressed on all paths (cf. Example 6.2.11).

Then a call of verify(S, R, Δ, I, f) runs as follows:

function verify(S, R, Δ, I, f) {	
$M := (S, R, \Delta, I)$;	
$\mathcal{C}_f :=$ f.getTopLevelConcepts();	$\mathcal{C}_f = \{MajorTopic, \mathsf{AF}\ \exists addressedBy.\top\}$
$I^{\mathcal{C}_f, M} := \emptyset$;	
for each $C' \in \mathcal{C}_f$ {	
$i_{C',M} :=$ getInterpretation(M, C');	$i_{MajorTopic,M} = \{s_0 \mapsto \{R1, R2\}, s_1 \mapsto \{R1\}, s_2 \mapsto \{R2\}\}$ $i_{\mathsf{AF}\ \exists addressedBy.\top,M} = \{s_0 \mapsto \{R1, R2\}, s_1 \mapsto \{R2\}, s_2 \mapsto \{R2\}\}$
$I^{\mathcal{C}_f,M} := I^{\mathcal{C}_f,M} \cup \{\langle C' \rangle \mapsto i_{C',M}\}$;	$I^{\mathcal{C}_f,M} = \{\langle MajorTopic \rangle \mapsto i_{MajorTopic,M}, \langle \mathsf{AF}\ \exists addressedBy.\top \rangle \mapsto i_{\mathsf{AF}\ \exists addressedBy.\top,M}\}$
}	
$M^{\mathcal{C}_f} := (S, R, \Delta, I^{\mathcal{C}_f,M})$;	
$f' := fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$;	$f' = (\langle MajorTopic \rangle(L1) \rightarrow \langle \mathsf{AF}\ \exists addressedBy.\top \rangle(L1)) \wedge (\langle MajorTopic \rangle(L2) \rightarrow \langle \mathsf{AF}\ \exists addressedBy.\top \rangle(L2)) \wedge (\langle MajorTopic \rangle(R1) \rightarrow \langle \mathsf{AF}\ \exists addressedBy.\top \rangle(R1)) \wedge (\langle MajorTopic \rangle(R2) \rightarrow \langle \mathsf{AF}\ \exists addressedBy.\top \rangle(R2))$
$M' := sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$;	$M' = (S, R, L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}})$ where $L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}} = \{ s_0 \mapsto \{\langle MajorTopic \rangle(R1), \langle MajorTopic \rangle(R2), \langle \mathsf{AF}\ \exists addressedBy.\top \rangle(R1), \langle \mathsf{AF}\ \exists addressedBy.\top \rangle(R2)\}, s_1 \mapsto \{\langle MajorTopic \rangle(R1), \langle \mathsf{AF}\ \exists addressedBy.\top \rangle(R2)\}, s_2 \mapsto \{\langle MajorTopic \rangle(R2), \langle \mathsf{AF}\ \exists addressedBy.\top \rangle(R2)\} \}$
return $\{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\}$	$\{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\} = \{s_0, s_2\}$
}	

□

Proposition 6.3.39 (Soundness and Completeness of verify)

Assume that all sub-routines called in **verify** are sound and complete, i.e. for a finite $\mathcal{ALC}$CTL temporal structure $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ and $\mathcal{ALC}$CTL formula f holds:

- f.getTopLevelConcepts() = $\mathcal{C}_f$ according to Definition 6.3.30;

- getInterpretation(M, C') = $i_{C',M} : S \to \mathcal{P}(\Delta) : i_{C',M}(s) = (M, s)(C')$;
- the sub-routine implementing $fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ adheres to Definitions 6.3.7, 6.3.19, and 6.3.30;
- the sub-routine implementing $sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ adheres to Definitions 6.3.4, 6.3.19, and 6.3.30;
- and $\vdash_{\mathsf{CTL}}$ is a sound and complete model checking procedure for finite CTL temporal structures and CTL formulae.

Then

verify(S, R, Δ, I, f) = $LS_{M,f} = \{s \in S \mid M, s \models_{\mathcal{ALC}\mathsf{CTL}} f\}$.

Proof:

In the **for**-loop of **verify**, a representation of $I^{\mathcal{C}_f,M}$ is calculated, which consists of a mapping $\langle C'\rangle \mapsto i_{C',M}$ of the atomic substitute $\langle C'\rangle$ of concepts $C' \in \mathcal{C}_f$ onto their interpretation $i_{C',M}$ that is a function $S \to \mathcal{P}(\Delta) : i_{C',M}(s) = (M, s)(C')$. Hence, the representation of $I^{\mathcal{C}_f,M}$ as calculated in the **for**-loop defines a mapping $f : \langle\mathcal{C}_f\rangle \times S \to \mathcal{P}(\Delta) : f(\langle C'\rangle, s) = (I^{\mathcal{C}_f,M}(\langle C'\rangle))(s) = (M, s)(C')$ where $\langle\mathcal{C}_f\rangle := \{\langle C'\rangle \mid C' \in \mathcal{C}_f\}$.

Recall, $I^{\mathcal{C}_f,M}$ is defined as an interpretation such that the following holds:

for each $s \in S$ and $C' \in \mathcal{C}_f$: $\langle C'\rangle^{I^{\mathcal{C}_f,M}(s)} = (M, s)(C')$ and for each state $s \in S$ and atomic role $\mathcal{R} \in AR$: $\mathcal{R}^{I^{\mathcal{C}_f,M}(s)} = \mathcal{R}^{I(s)}$ (Definition 6.3.19).

Hence, regarding the interpretation of concepts, $I^{\mathcal{C}_f,M}$ is determined by a mapping of pairs $(\langle C'\rangle, s) \in \langle\mathcal{C}_f\rangle \times S$ onto the interpretation $(M, s)(C')$ of concept C' in structure M at state s. As a result, the mapping $I^{\mathcal{C}_f,M}$ as calculated by **verify** is complete and sound w.r.t. the interpretation of concepts in $\mathcal{C}_f$.

Regarding the interpretation of roles we observe the following:

f is not accessed directly in the sequel of the **for**-loop in **verify** since the mappings fm and sm are based on $f^{\langle\rangle}$ instead of f. Recall, $f^{\langle\rangle}$ is the formula f except that the top-level concepts are substituted by atomic concepts. Since atomic roles occur in complex concepts only, $f^{\langle\rangle}$ does not contain any atomic roles and thus the formula and structure mappings fm and sm are independent of the interpretation of roles in f. As a result, within the context of **verify** it is sufficient to address the interpretation of concepts $C' \in \mathcal{C}_f$ in the representation of $I^{\mathcal{C}_f,M}$ and hence the representation of $I^{\mathcal{C}_f,M}$ as calculated in the **for**-loop of **verify** is sound and complete.

By Corollary 6.3.34, it holds:

$M, s \models_{\mathcal{ALC}\mathsf{CTL}} f \Leftrightarrow sm(M^{\mathcal{C}_f},\ f^{\langle\rangle}), s \models_{\mathsf{CTL}} fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ for each $s \in S$.

Hence,

$$
\begin{array}{lll}
& LS_{M,f} & \\
= & \{s \in S \mid M, s \models_{\mathcal{ALC}\mathsf{CTL}} f\} & \text{Def. } LS_{M,f} \\
= & \{s \in S \mid sm(M^{\mathcal{C}_f},\ f^{\langle\rangle}), s \models_{\mathsf{CTL}} fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})\} & \text{Corollary 6.3.34} \\
= & \{s \in S \mid M', s \models_{\mathsf{CTL}} f'\} & \text{Assignment of } M' \text{ and } f' \text{ in verify} \\
= & \{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\} & \text{because } \vdash_{\mathsf{CTL}} \text{ is sound and complete w.r.t. } \models_{\mathsf{CTL}} \\
= & \text{verify}(S, R, \Delta, I, f) &
\end{array}
$$

□

Remark 6.3.40 (Soundness and Completeness of verify)

Proposition 6.3.39 shows the soundness and completeness of **verify** under the assumption that all sub-routines used in verify are sound and complete.

This assumption is justified in the case of f.getTopLevelConcepts(), $fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$, and $sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ because they can be implemented directly based on their formal definitions which involve finite structures only.

The assumption is also justified in the case of $\vdash_{\mathsf{CTL}}$ since sound and complete model checking algorithms for finite CTL temporal structures and CTL formulae exist [CES86] and efficient implementations are available [CCG+02, CCGR00].

As of getInterpretation(M, C'), the existence of a sound and complete algorithm is still to be shown. This will be done in section 6.3.3.2. □

Proposition 6.3.41 (Runtime Complexity of verify)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure and f an $\mathcal{ALC}$CTL formula.

Let $T(S, R, \Delta, I, f)$ denote the runtime required for a call to verify(S, R, Δ, I, f).

If all sub-routines used in **verify** are optimal then $T(S, R, \Delta, I, f) \in \mathcal{O}(|f| \cdot (|S| + |R|) \cdot |\Delta|^2)$.

Proof:

Let

- $T(f$.getTopLevelConcepts()$)$ be the time required for getting the top-level concepts of f. This requires a recursive traversal of the term structure of f which can be done in

  $T(f$.getTopLevelConcepts()$) \in \mathcal{O}(|f|)$.

- $T(\text{getInterpretation}(M, C'))$ be the time required for getting the interpretation $i_{C,M} : S \rightarrow \mathcal{P}(\Delta) : i_{C,M}(s) = (M, s)(C')$ by a call to getInterpretation(M, C'). This is equivalent to getting the interpretation $I^{\{C\},M}$ extended to C w.r.t. M which is in

  $T(\text{getInterpretation}(M, C')) \in \mathcal{O}(|C'| \cdot (|S| + |R|) \cdot |\Delta|^2)$ (Lemma 6.3.28).
- $T(fm(M^{\mathcal{C}_f},\ f^{\langle\rangle}))$ be the time required for calculating $fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$. This can be done in $\mathcal{O}(|f^{\langle\rangle}| \cdot |\Delta|)$ as shown in the proof of Proposition 6.3.29, Equation (6.15). Since $|f^{\langle\rangle}| \leq |f|$, we get:

  $T(fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})) \in \mathcal{O}(|f| \cdot |\Delta|)$.
- $T(sm(M^{\mathcal{C}_f},\ f^{\langle\rangle}))$ be the time required for calculating $sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$. This can be done in $\mathcal{O}(|f^{\langle\rangle}| \cdot |S| \cdot |\Delta|)$ as shown in the proof of Proposition 6.3.29, Equation (6.14). Since $|f^{\langle\rangle}| \leq |f|$, we get:

  $T(sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})) \in \mathcal{O}(|f| \cdot |S| \cdot |\Delta|)$.
- $T(\vdash_{\mathsf{CTL}})$ be the time required for calculating the set $\{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\}$. This is in $\mathcal{O}(|f'| \cdot (|S| + |R|))$ (Theorem 3.2.12). Since $|f'| = |fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})| \in \mathcal{O}(|f^{\langle\rangle}| \cdot |\Delta|)$ (proof of Proposition 6.3.29) and $|f^{\langle\rangle}| \leq |f|$ we get:

  $T(\vdash_{\mathsf{CTL}}) \in \mathcal{O}(|f| \cdot (|S| + |R|) \cdot |\Delta|)$.
- $k \in \mathbb{R}^+$ be the cumulative runtime of all other statements in **verify** (all of them can be done in constant time).

For the overall runtime of verify(S, R, Δ, I, f) holds:

$$
\begin{aligned}
T(S, R, \Delta, I, f) \;=\;& T(f.\text{getTopLevelConcepts}()) \\
&+ \sum_{C' \in \mathcal{C}_f} T(\text{getInterpretation}(M, C')) \\
&+ T(fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})) + T(sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})) \\
&+ T(\vdash_{\mathsf{CTL}}) + k \\
\leq\;& k_1 \cdot |f| \\
&+ \sum_{C' \in \mathcal{C}_f} k_2 \cdot |C'| \cdot (|S| + |R|) \cdot |\Delta|^2 \\
&+ k_3 \cdot |f| \cdot |\Delta| + |k_4| \cdot |f| \cdot |S| \cdot |\Delta| \\
&+ k_5 \cdot |f| \cdot (|S| + |R|) \cdot |\Delta| + k
\end{aligned}
$$

It holds for the term $\sum_{C' \in \mathcal{C}_f} k_2 \cdot |C'| \cdot (|S| + |R|) \cdot |\Delta|^2$:

$$
\begin{aligned}
\sum_{C' \in \mathcal{C}_f} k_2 \cdot |C'| \cdot (|S| + |R|) \cdot |\Delta|^2 \;&=\; k_2 \cdot (|S| + |R|) \cdot |\Delta|^2 \cdot \sum_{C' \in \mathcal{C}_f} |C'| \\
&\leq\; k_2 \cdot (|S| + |R|) \cdot |\Delta|^2 \cdot |f|
\end{aligned}
$$

since all $C' \in \mathcal{C}_f$ are sub-expressions in f. Hence, we get

$$\begin{aligned} T(S,R,\Delta,I,f) &\leq k_1 \cdot |f| \\ &\quad + k_2 \cdot (|S| + |R|) \cdot |\Delta|^2 \cdot |f| \\ &\quad + k_3 \cdot |f| \cdot |\Delta| + |k_4| \cdot |f| \cdot |S| \cdot |\Delta| \\ &\quad + k_5 \cdot |f| \cdot (|S| + |R|) \cdot |\Delta| + k \\ &\leq (k_1 + k_2 + k_3 + k_4 + k_5 + k) \cdot |f| \cdot (|S| + |R|) \cdot |\Delta|^2 \\ &\in \mathcal{O}(|f| \cdot (|S| + |R|) \cdot |\Delta|^2) \end{aligned}$$

□

Remark 6.3.42 (Runtime Complexity of verify)

Proposition 6.3.41 implies the termination of **verify** for any finite input S, R, Δ and f if all sub-routines are sound and complete and thus terminate.

Proposition 6.3.41 shows that **verify** is optimal w.r.t. the complexity of the $\mathcal{ALC}$CTL model checking problem shown in Proposition 6.3.35.

In contrast to the CTL model checking problem, which is inherently sequential and thus cannot be efficiently solved using parallel computation [Sch03], the **verify** algorithm for model checking $\mathcal{ALC}$CTL can easily be parallelized. The instances of the **for** loop of **verify**, which contribute most to the overall runtime of **verify**, can be executed in parallel because the interpretations $i_{C',M}$ for $C' \in \mathcal{C}_f$ can be computed independently. Hence, the runtime can be sped up by factor n on a machine with as many processors as there are top-level concepts $n = |\mathcal{C}_f|$ in formula f.

Since typically $|\mathcal{C}_f| < 10$, this is not a large but still relevant source of parallelism.

□

6.3.3.2. Algorithm for Computing Concept Interpretations

The algorithm **verify** delegates a large part of the model checking work to the sub-routine **getInterpretation**(M, C') that actually belongs to the core of model checking $\mathcal{ALC}$CTL. In the sequel, we sketch an abstract implementation of **getInterpretation**.

Algorithm 6.3.43 (Interpretation of $\mathcal{ALC}$CTL Concepts)

function getInterpretation(S, R, Δ, I, C) {	
if C.isAtomic() **then**	If $C \in AC$
return $\{s \mapsto C^{I(s)} \mid s \in S\}$;	return a mapping of s onto the interpretation $C^{I(s)}$ of C in state s for each $s \in S$.
if $C = \top$ **then**	If $C = \top$
return $\{s \mapsto \Delta \mid s \in S\}$;	return a mapping of states S onto Δ since $\top^{I(s)} := \Delta$ for each $s \in S$.
if $C = \bot$ **then**	If $C = \bot$
return $\{s \mapsto \emptyset \mid s \in S\}$;	return a mapping of states S onto $\emptyset$ since $\bot^{I(s)} := \emptyset$ for each $s \in S$.
$M := (S, R, \Delta, I)$;	For a more compact access to structure M.
$\lfloor C \rfloor := C$.getDirectSubConcepts();	Initialize the set $\lfloor C \rfloor$ of direct sub-concepts of C.
$I^{\lfloor C \rfloor, M} := \emptyset$;	Initialize the interpretation $I^{\lfloor C \rfloor, M}$ extended to direct sub-concepts $\lfloor C \rfloor$ of C w.r.t. M.
for each $C' \in \lfloor C \rfloor$ {	Construct the extended interpretation $I^{\lfloor C \rfloor, M}$.
$i_{C',M} :=$ getInterpretation(M, C');	Get the interpretation $i_{C',M} : S \to \mathcal{P}(\Delta) : i_{C',M}(s) = (M, s)(C')$ of concept C' at each state of M.
$I^{\lfloor C \rfloor, M} := I^{\lfloor C \rfloor, M} \cup \{\langle C' \rangle \mapsto i_{C',M}\}$;	Add a mapping of the atomic substitute $\langle C' \rangle$ of C' onto its interpretation $i_{C',M}$ in M to $I^{\lfloor C \rfloor, M}$.
}	
$I^{\lfloor C \rfloor, M} := I^{\lfloor C \rfloor, M} \cup$ I.getRoleInterpretations();	Copy the interpretations of atomic roles in I to $I^{\lfloor C \rfloor, M}$, i.e. $\mathcal{R}^{I^{\lfloor C \rfloor, M}(s)} = \mathcal{R}^{I(s)}$ for each state $s \in S$ and atomic role $\mathcal{R} \in AR$.
$M^{\lfloor C \rfloor} := (S, R, \Delta, I^{\lfloor C \rfloor, M})$;	Assign to the temporal structure $M^{\lfloor C \rfloor}$ an interpretation $I^{\lfloor C \rfloor, M}$ extended to all direct sub-concepts $\lfloor C \rfloor$ of C.
$M' := sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq C^{\langle\rangle})$;	Generate a CTL structure M' according to the definition of $sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq C^{\langle\rangle})$.
$ls_{C,M} := \emptyset$;	Initialize the mapping $ls_{C,M} : \Delta \to \mathcal{P}(S) : ls_{C,M} = LS_{a,C,M}$ where $LS_{a,C,M}$ is the label set of C for a w.r.t. M. $ls_{C,M}$ is iteratively computed in the subsequent **for** loop.
for each $a \in \Delta$ {	Construct the label set $LS_{a,C,M}$ by reduction to CTL model checking.
$f' := C^{\langle\rangle}[a]_\Delta$;	Generate a CTL formula f' according to the definition of $C^{\langle\rangle}[a]_\Delta$.
$LS_{a,C,M} := \{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\}$;	Model check f' for M' using $\vdash_{\mathsf{CTL}}$ which results in $LS_{a,C,M} = \{s \in S \mid a \in (M, s)(C)\}$.
$ls_{C,M} := ls_{C,M} \cup \{a \mapsto LS_{a,C,M}\}$;	Add a mapping of a onto its label set $LS_{a,C,M}$ to $ls_{C,M}$.
}	
$i_{C,M} := \{s \mapsto \{a \in \Delta \mid$ $s \in ls_{C,M}(a)\} \mid s \in S\}$;	Convert the mapping $ls_{C,M}$ to a mapping $i_{C,M} : S \to \mathcal{P}(\Delta) : i_{C,M}(s) = \{a \in \Delta \mid s \in ls_{C,M}(a)\} = (M, s)(C)$ of states $s \in S$ onto their intepretation $(M, s)(C)$ of C in M at s.
return $i_{C,M}$;	Return the interpretation function $i_{C,M}$.
}	□

getInterpretation uses the following sub-routines:

- C.isAtomic();

  returns $true$ if $C \in AC$.

- C.getDirectSubConcepts();

  returns the set $\lfloor C \rfloor$ of direct sub-concepts of a complex concept C (Definition 6.3.15).

- I.getRoleInterpretations();

  returns the mapping $AR \rightarrow \mathcal{P}(S \rightarrow \mathcal{P}(\Delta \times \Delta))$ of atomic roles $\mathcal{R} \in AR$ onto their temporal interpretation $s \mapsto \mathcal{R}^{I(s)}$ in each state $s \in S$.

- $sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle})$;

  returns the CTL representation of the temporal structure $M^{\lfloor C \rfloor}$ w.r.t. the formula $\bot \sqsubseteq\ C^{\langle\rangle}$ according to Definitions 6.3.4, 6.3.15, 6.3.19, and 6.3.17.

  Recall, $sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle})$ is equal to $M^{\lfloor C \rfloor}$ except that the interpretation of atomic concepts in $\bot \sqsubseteq\ C^{\langle\rangle}$ (which are $\{\langle C' \rangle | C' \in \lfloor C \rfloor\}$) and the interpretation of atomic roles in $\bot \sqsubseteq\ C^{\langle\rangle}$ are encoded in terms of a labelling $L_{I^{\lfloor C \rfloor}, M, \bot \sqsubseteq C^{\langle\rangle}} : S \rightarrow \mathcal{P}(AP)$ of states $s \in S$ with sets of atomic propositions $L_{I^{\lfloor C \rfloor}, M, \bot \sqsubseteq C^{\langle\rangle}}(s) \subseteq AP$ satisfied at s.

- $C^{\langle\rangle}[a]_{\Delta}$;

  returns the CTL mapping of concept $C^{\langle\rangle}$ for a domain object $a \in \Delta$ according to Definitions 6.3.7 and 6.3.17. Recall, $C^{\langle\rangle}$ is concept C except that the direct sub-concepts of C are treated as atomic (Definition 6.3.17).

- $M', s \vdash_{\mathsf{CTL}} f'$;

  $\vdash_{\mathsf{CTL}}$ denotes a sound and complete algorithm for deciding $M', s \models_{\mathsf{CTL}} f'$. It checks whether the CTL formula f' holds in the finite CTL temporal structure M' at state s.

Example 6.3.44 (Algorithm getInterpretation)

We illustrate a call to **getInterpretation** on the scenario of Example 6.3.38 which is

- $S := \{s_0, s_1, s_2\}$;
- $R := \{(s_0, s_1), (s_0, s_2), (s_1, s_2), (s_2, s_2)\} \subseteq S \times S$;
- $\Delta :=$ {*L1, L2, R1, R2*}.
- an interpretation of atomic concepts AC:

MajorTopic$^{I(s_0)}$	$=$	{*R1, R2*}	Topic *R180*, abbreviated as $R1$, and topic *R180_Handling*, abbreviated as $R2$, are the major topics in state s_0.
MajorTopic$^{I(s_1)}$	$=$	{*R1*}	*R180* is the (only) major topic in state s_1.
MajorTopic$^{I(s_2)}$	$=$	{*R2*}	*R180_Handling* is the (only) major topic in state s_2.

and an interpretation of atomic roles AR:

addressedBy$^{I(s_0)}$	$=$	{*(R1,L1)*}	*R180* is addressed by lesson *L1* in state s_0.
addressedBy$^{I(s_1)}$	$=$	$\emptyset$	There are no topics addressed in state s_1.
addressedBy$^{I(s_2)}$	$=$	{*(R2,L2)*}	*R180_Handling* is addressed by lesson *L2* in state s_2.

Let $C = \mathsf{AF}\ \exists addressedBy.\top$ - the set of topics that are eventually addressed on all paths.

Then a call of getInterpretation(S, R, Δ, I, C) yields the following results:

function getInterpretation(S, R, Δ, I, C) {	
if C.isAtomic() **then**	C.isAtomic() is $false$
return $\{s \mapsto C^{I(s)} \mid s \in S\}$;	
if $C = \top$ **then**	$C = \top$ is $false$
return $\{s \mapsto \Delta \mid s \in S\}$;	
if $C = \bot$ **then**	$C = \bot$ is $false$
return $\{s \mapsto \emptyset \mid s \in S\}$;	
$M := (S, R, \Delta, I)$;	
$\lfloor C \rfloor :=$ C.getDirectSubConcepts();	$\lfloor C \rfloor = \{\exists addressedBy.\top\}$
$I^{\lfloor C \rfloor, M} := \emptyset$;	
for each $C' \in \lfloor C \rfloor$ {	
$i_{C',M} :=$ getInterpretation(M, C');	$i_{\exists addressedBy.\top, M} = \{s_0 \mapsto \{R1\}, s_1 \mapsto \emptyset, s_2 \mapsto \{R2\}\}$
$I^{\lfloor C \rfloor, M} := I^{\lfloor C \rfloor, M} \cup \{\langle C' \rangle \mapsto i_{C',M}\}$;	$I^{\lfloor C \rfloor, M} = \{ \langle \exists addressedBy.\top \rangle \mapsto i_{\exists addressedBy.\top, M}\}$.
}	
$I^{\lfloor C \rfloor, M} := I^{\lfloor C \rfloor, M} \cup$ I.getRoleInterpretations();	$I^{\lfloor C \rfloor, M} = \{ \langle \exists addressedBy.\top \rangle \mapsto i_{\exists addressedBy.\top, M}, addressedBy \mapsto \{s_0 \mapsto \{(R1, L1)\}, s_1 \mapsto \emptyset, s_2 \mapsto \{(R2, L2)\}\}$ }.
$M^{\lfloor C \rfloor} := (S, R, \Delta, I^{\lfloor C \rfloor, M})$;	
$M' := sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq C^{\langle\rangle})$;	Let $f = \bot \sqsubseteq C^{\langle\rangle}$ $= \bot \sqsubseteq \mathsf{AF}\, \langle \exists addressedBy.\top \rangle$. Then $M' = (S, R, L_{I^{\lfloor C \rfloor, M}, f})$ where $L_{I^{\lfloor C \rfloor, M}, f} = \{ s_0 \mapsto \{\langle \exists addressedBy.\top \rangle (R1)\}, s_1 \mapsto \emptyset, s_2 \mapsto \{\langle \exists addressedBy.\top \rangle (R2)\}$ }
$ls_{C,M} := \emptyset$;	
for each $a \in \Delta$ {	$\Delta = \{L1, L2, R1, R2\}$
$f' := C^{\langle\rangle}[a]_\Delta$;	$C^{\langle\rangle}[L1]_\Delta = \mathsf{AF}\, (\langle \exists addressedBy.\top \rangle (L1))$ $C^{\langle\rangle}[L2]_\Delta = \mathsf{AF}\, (\langle \exists addressedBy.\top \rangle (L2))$ $C^{\langle\rangle}[R1]_\Delta = \mathsf{AF}\, (\langle \exists addressedBy.\top \rangle (R1))$ $C^{\langle\rangle}[R2]_\Delta = \mathsf{AF}\, (\langle \exists addressedBy.\top \rangle (R2))$
$LS_{a,C,M} := \{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\}$;	$LS_{L1,C,M} = \emptyset$ $LS_{L2,C,M} = \emptyset$ $LS_{R1,C,M} = \{s_0\}$ $LS_{R2,C,M} = \{s_0, s_1, s_2\}$
$ls_{C,M} := ls_{C,M} \cup \{a \mapsto LS_{a,C,M}\}$;	
}	$ls_{C,M} = \{L1 \mapsto \emptyset, L2 \mapsto \emptyset, R1 \mapsto \{s_0\}, R2 \mapsto \{s_0, s_1, s_2\}\}$
$i_{C,M} := \{s \mapsto \{a \in \Delta \mid s \in ls_{C,M}(a)\} \mid s \in S\}$;	$i_{C,M} = \{s_0 \mapsto \{R1, R2\}, s_1 \mapsto \{R2\}, s_2 \mapsto \{R2\}\}$
return $i_{C,M}$;	
}	

The mapping $i_{C,M} = \{s_0 \mapsto \{R1, R2\}, s_1 \mapsto \{R2\}, s_2 \mapsto \{R2\}\}$ as returned by getInterpretation(S, R, Δ, I, C) represents the interpretation of concept $C = \mathsf{AF}\ \exists addressedBy.\top$ in the states $s \in S$ of $M = (S, R, \Delta, I)$ which is

$$\begin{aligned}(M, s_0)(\mathsf{AF}\ \exists addressedBy.\top) &= \{R1, R2\}\\ (M, s_1)(\mathsf{AF}\ \exists addressedBy.\top) &= \{R2\}\\ (M, s_2)(\mathsf{AF}\ \exists addressedBy.\top) &= \{R2\}\end{aligned}$$

(cf. Examples 6.3.38 and 6.2.11).

□

In the sequel, we prove the termination, soundness, completeness, and runtime complexity of **getInterpretation**. We start with an approximation of the total number of recursive calls to **getInterpretation** for a concept $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$.

Lemma 6.3.45 (Total Number of Recursive Calls to getInterpretation)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure and $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ an $\mathcal{ALC}$CTL concept.

Then a call to **getInterpretation**(S, R, Δ, I, C) results in at most $|C|$ total calls to **getInterpretation**.

Proof:

The prove is by induction on the term structure of C.

- If $C \in AC \cup \{\top, \bot\}$ then getInterpretation(S, R, Δ, I, C) terminates without recursive calls, i.e. **getInterpretation** is called once in total.
- If $C \notin AC \cup \{\top, \bot\}$ then $\lfloor C \rfloor \neq \emptyset$ and hence **getInterpretation** calls itself for every direct sub-concept $C' \in \lfloor C \rfloor$.

  By induction hypothesis we can assume that each call of **getInterpretation** for $C' \in \lfloor C \rfloor$ results in at most $|C'|$ recursive calls to **getInterpretation** in total.

  Hence, the total number of calls to **getInterpretation** is at most

  $$1 + \sum_{C' \in \lfloor C \rfloor} |C'| \leq |C|$$

  since the size $|C|$ of concept C is the sum of the sizes $|C'|$ of its direct sub-concepts $C' \in \lfloor C \rfloor$ plus one additional connective plus potentially an additional role.

□

Corollary 6.3.46 (Termination of getInterpretation)

getInterpretation terminates when called for a finite temporal structure $(S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ and a finite concept $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$.

This is a direct consequence of Lemma 6.3.45 and the fact that every loop in **getInterpretation** ranges over finite sets according to Definitions 6.3.15 and 6.2.6. □

Proposition 6.3.47 (Soundness and Completeness of getInterpretation)

Assume, all sub-routines called in **getInterpretation** are sound and complete, i.e. for a finite $\mathcal{ALC}\mathsf{CTL}$ temporal structure $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ and $\mathcal{ALC}\mathsf{CTL}$ formula f holds:

- C.isAtomic() is $true \Leftrightarrow C \in AC$.
- C.getDirectSubConcepts() $= \lfloor C \rfloor$ according to Definition 6.3.15.
- I.getRoleInterpretations() $= \{\mathcal{R} \mapsto \{s \mapsto \mathcal{R}^{I(s)} \mid s \in S\} \mid \mathcal{R} \in AR\}$.
- the sub-routine implementing $sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle})$ adheres Definitions 6.3.4, 6.3.15, 6.3.19, and 6.3.17.
- the sub-routine implementing $C^{\langle\rangle}[a]_\Delta$ adheres Definitions 6.3.7 and 6.3.17.
- and $\vdash_{\mathsf{CTL}}$ is a sound and complete model checking procedure for finite CTL temporal structures and CTL formulae.

Then getInterpretation$(S, R, \Delta, I, C) = i_{C,M} : S \to \mathcal{P}(\Delta) : i_{C,M}(s) = (M, s)(C)$.

Proof:

The proof is by induction on the structure of C.

- Assume $C \in AC$.

  Then getInterpretation$(S, R, \Delta, I, C) = \{s \mapsto C^{I(s)} \mid s \in S\} = \{s \mapsto (M, s)(C) \mid s \in S\} = i_{C,M}$ by Definition 6.2.8.

- Assume $C = \top$.

  Then getInterpretation$(S, R, \Delta, I, C) = \{s \mapsto \Delta \mid s \in S\} = \{s \mapsto (M, s)(\top) \mid s \in S\} = i_{C,M}$ by Definitions 6.2.3 and 6.2.8.

- Assume $C = \bot$.

  Then getInterpretation$(S, R, \Delta, I, C) = \{s \mapsto \emptyset \mid s \in S\} = \{s \mapsto (M, s)(\bot) \mid s \in S\} = i_{C,M}$ by Definitions 6.2.3 and 6.2.8.

- Assume $C \notin AC \cup \{\top, \bot\}$. Then C is a complex concept and $\lfloor C \rfloor \neq \emptyset$. By induction hypothesis, we can assume that getInterpretation(M, C') = $i_{C',M} = \{s \mapsto (M,s)(C') \mid s \in S\}$ for each $C' \in \lfloor C \rfloor$.

  In the first **for**-loop of **getInterpretation** a mapping
  $f := \{\langle C' \rangle \mapsto i_{C',M} \mid C' \in \lfloor C \rfloor\}$ of the atomic substitutes $\langle C' \rangle$ of direct subconcepts $C' \in \lfloor C \rfloor$ onto their interpretation function $i_{C',M}$ is computed where $i_{C',M}$ = getInterpretation(S, R, Δ, I, C').

  f determines a mapping
  $g : \langle \lfloor C \rfloor \rangle \times S \to \mathcal{P}(\Delta) : g(\langle C' \rangle, s) = (f(\langle C' \rangle))(s) = i_{C',M}(s)$ where $\langle \lfloor C \rfloor \rangle := \{\langle C' \rangle \mid C' \in \lfloor C \rfloor\}$.

  By induction hypothesis $i_{C',M}(s) = (M,s)(C')$ for each $s \in S$.

  Hence, f determines a function $g : \langle \lfloor C \rfloor \rangle \times S \to \mathcal{P}(\Delta)$ such that $g(\langle C' \rangle, s) = (M,s)(C')$ for each $C' \in \lfloor C \rfloor$ and $s \in S$.

  Recall, $I^{\lfloor C \rfloor, M}$ is defined in Definition 6.3.19 as an interpretation such that

  $i)$ for each $C' \in \lfloor C \rfloor$ and $s \in S$ holds: $\langle \mathcal{C}' \rangle^{I^{\lfloor C \rfloor, M}(s)} = (M,s)(C)$

  $ii)$ for each $\mathcal{R} \in AR$ and $s \in S$ holds: $\mathcal{R}^{I^{\lfloor C \rfloor, M}(s)} = \mathcal{R}^{I(s)}$.

  Regarding $i)$, it holds for each $C' \in \lfloor C \rfloor$ and $s \in S$:

  $$(f(\langle C' \rangle))(s) = g(\langle C' \rangle, s) = (M,s)(C) = \langle \mathcal{C}' \rangle^{I^{\lfloor C \rfloor, M}(s)}$$

  Hence, the mapping f as computed by **getInterpretation** is a sound and complete representation of $I^{\lfloor C \rfloor, M}$ regarding the interpretation of concepts.

  Also, regarding the interpretation of roles $ii)$, $I^{\lfloor C \rfloor, M}$, as computed in **getInterpretation**, is sound and complete since all interpretations of roles are copied without modification from I to $I^{\lfloor C \rfloor, M}$.

  By Corollary 6.3.25, it holds for each $a \in \Delta$:

  $$LS_{a,C,M} = LS_{a,C^{\langle\rangle},M^{\lfloor C \rfloor}} = \{s \in S \mid sm(M^{\lfloor C \rfloor}, f), s \models_{\mathsf{CTL}} C^{\langle\rangle}[a]_\Delta\}$$

  where f is an $\mathcal{ALC}$CTL formula containing $C^{\langle\rangle}$ and $M^{\lfloor C \rfloor} = (S, R, \Delta, I^{\lfloor C \rfloor, M})$.

  Since $\bot \sqsubseteq C^{\langle\rangle}$ is an $\mathcal{ALC}$CTL formula containing $C^{\langle\rangle}$, M' is assigned $sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq C^{\langle\rangle})$, f' is assigned $C^{\langle\rangle}[a]_\Delta$, and $\vdash_{\mathsf{CTL}}$ is sound and complete w.r.t. $\models_{\mathsf{CTL}}$ in **getInterpretation**, it holds for each $a \in \Delta$:

  $$\begin{array}{lcll} LS_{a,C,M} & = & \{s \in S \mid a \in (M,s)(C)\} & \text{Definition 6.3.21} \\ & = & \{s \in S \mid sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq C^{\langle\rangle}), s & \\ & & \quad \models_{\mathsf{CTL}} C^{\langle\rangle}[a]_\Delta\} & \text{Corollary 6.3.25} \\ & = & \{s \in S \mid M', s \models_{\mathsf{CTL}} f'\} & \text{Assignment of } M' \text{ and } f' \text{ in } \textbf{getInterpretation} \\ & = & \{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\} & \text{because } \vdash_{\mathsf{CTL}} \text{ is sound and complete w.r.t. } \models_{\mathsf{CTL}} \end{array}$$

Hence, $LS_{a,C,M}$ as calculated in the second **for**-loop of **getInterpretation** is sound and complete w.r.t. its Definition 6.3.21 for each $a \in \Delta$ and, consequently, the label mapping $ls_{C,M}$ as calculated in **getInterpretation** is a sound and complete mapping $\Delta \to \mathcal{P}(S) : ls_{C,M}(a) = LS_{a,C,M} = \{s \in S \mid a \in (M,s)(C)\}$ of domain objects $a \in \Delta$ onto their label set $LS_{a,C,M}$.

By Corollary 6.3.22, $(M,s)(C) = \{a \in \Delta \mid s \in LS_{a,C,M}\} = \{a \in \Delta \mid s \in ls_{C,M}(a)\}$ for each $s \in S$.

As result, getInterpretation$(S, R, \Delta, I, C) = \{s \mapsto \{a \in \Delta \mid s \in ls_{C,M}(a)\}\} = \{s \mapsto (M,s)(C) \mid s \in S\} = i_{C,M}$ which proves **getInterpretation** to be sound and complete.

□

Proposition 6.3.48 (Runtime Complexity of getInterpretation)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure and $C \in \mathcal{C}_{\mathcal{ALC}\mathsf{CTL}}$ an $\mathcal{ALC}$CTL concept.

Let $T(S, R, \Delta, I, C)$ denote the runtime required for a call to getInterpretation(S, R, Δ, I, C).

If all sub-routines (other than **getInterpretation** itself), which are called by **getInterpretation**, are optimal then $T(S, R, \Delta, I, C) \in \mathcal{O}(|C| \cdot (|S| + |R|) \cdot |\Delta|^2)$.

Proof:

Let $T(getInterpretation)$ be the time required for a single call to **getInterpretation** without the time required for recursive sub-calls.

Assume that $T(getInterpretation) \in \mathcal{O}((|S| + |R|) \cdot |\Delta|^2)$.

Then this implies $T(S, R, \Delta, I, C) \in \mathcal{O}(|C| \cdot (|S| + |R|) \cdot |\Delta|^2)$ since a call of getInterpretation(S, R, Δ, I, C) results at most $|C|$ calls to **getInterpretation** (Lemma 6.3.45) each of them taking $T(getInterpretation) \in \mathcal{O}((|S| + |R|) \cdot |\Delta|^2)$ time.

Hence, it is sufficient to show:

$$T(getInterpretation) \in \mathcal{O}((|S| + |R|) \cdot |\Delta|^2) \tag{6.21}$$

Assume $|C| = 1$.

Then $C \in AC \cup \{\top, \bot\}$ and the runtime $T(getInterpretation)$ is in $\mathcal{O}(|\text{getInterpretation}(S, R, \Delta, I, C)|) = \mathcal{O}(|i_{C,M}|) = \mathcal{O}(|S| \cdot |\Delta|) \subseteq \mathcal{O}((|S| + |R|) \cdot |\Delta|^2)$ which shows Equation (6.21).

Assume $|C| > 1$.

Then C is a complex concept and hence $C \notin AC \cup \{\top, \bot\}$.

By $\mathcal{ALC}$CTL syntax $||\lfloor C \rfloor|| \leq 2$ and hence the first **for** loop in **getInterpretation** runs at most twice.

If we neglect the time required for the recursive call of **getInterpretation**, all statements except
$M' := sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle})$,
$f' := C^{\langle\rangle}[a]_\Delta$,
$LS_{a,C,M} := \{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\}$, and
$i_{C,M} := \{s \to \{a \in \Delta \mid s \in ls_{C,M}(a) \mid s \in S\}$
can be done in constant time because all of them can be implemented based on a constant number of simple pointer accesses and assignments. Also, the unions of sets "$\cup$" can be done in constant time by concatenation since the unified sets are disjoint and hence a duplicate check and elimination does not need to be performed.

Let

- $T(sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle}))$ be the time required for calculating $sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle})$,
- $T(C^{\langle\rangle}[a]_\Delta)$ be the time required for calculating $C^{\langle\rangle}[a]_\Delta$,
- $T(\vdash_{\mathsf{CTL}})$ be the time required for calculating $\{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\}$, and
- $T(i_{C,M})$ be the time required for calculating $\{s \to \{a \in \Delta \mid s \in ls_{C,M}(a)\} \mid s \in S\}$.

Then $C^{\langle\rangle}[a]_\Delta$ and $\{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\}$ are calculated $|\Delta|$ times within the second **for**-loop of **getInterpretation** while $sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle})$ and $\{s \to \{a \in \Delta \mid s \in ls_{C,M}(a)\} \mid s \in S\}$ are calculated just once.

In total we get for the runtime of a single run of **getInterpretation**:

$T(getInterpretation) =$
$k + T(sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle})) + |\Delta|(k' + T(C^{\langle\rangle}[a]_\Delta) + T(\vdash_{\mathsf{CTL}})) + T(i_{C,M})$

where $k' \in \mathbb{R}^+$ is the total time required for all "constant time operations" within the second **for**-loop of **getInterpretation** and $k \in \mathbb{R}^+$ the total time required for all other "constant time operations" in **getInterpretation**.

It holds:

- $sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle})$ can be calculated in

  $$T(sm(M^{\lfloor C \rfloor}, \bot \sqsubseteq\ C^{\langle\rangle})) \leq k_1 \cdot |S| \cdot |\Delta|^2$$

  with $k_1 \in \mathbb{R}^+$ as shown in the proof of Lemma 6.3.27, Equation 6.10.
- $C^{\langle\rangle}[a]_\Delta$ can be calculated in

  $$T(C^{\langle\rangle}[a]_\Delta) \leq k_2 \cdot |\Delta|$$

  with $k_2 \in \mathbb{R}^+$ as shown in the proof of Lemma 6.3.27, Equation 6.11.

- For calculating $\{s \in S \mid M', s \vdash_{\mathsf{CTL}} f'\}$, a sound and complete model checking algorithm $\vdash_{\mathsf{CTL}}$ exists [CES86, CGP02d], the runtime of which is $T(\vdash_{\mathsf{CTL}}) \in \mathcal{O}(|f'|\cdot(|S|+|R|))$.

  Since $|f'| = |C^{\langle\rangle}[a]_\Delta| \in \mathcal{O}(|\Delta|)$ (Lemma 6.3.26) we get $T(\vdash_{\mathsf{CTL}}) \in \mathcal{O}(|\Delta|\cdot(|S|+|R|))$.

  Hence, there is $k_3 \in \mathbb{R}^+$ such that for sufficiently large sets S, R, and Δ it holds:

  $$T(\vdash_{\mathsf{CTL}}) \leq k_3\cdot(|S|+|R|)\cdot|\Delta|$$

- Finally, $i_{C,M} = \{s \to \{a \in \Delta \mid s \in ls_{C,M}(a)\} \mid s \in S\}$ can be calculated simply by two nested loops: the outer one ranges over the set S and the inner one ranges over the set Δ for each interation of the outer loop. This results in $|S|\cdot|\Delta|$ iterations each of which performs a table lookup $ls_{C,M}(a)$ and an element check $s \in ls_{C,M}(a)$, which both can be done in constant time using hash tables.

  As a result, $T(i_{C,M}) = k_4\cdot|S|\cdot|\Delta|$ with some constant $k_4 \in \mathbb{R}^+$.

In total, we get for the runtime of a single call to **getInterpretation**:

$$\begin{aligned}
T(getInterpretation) &= k + T(sm(M^{\lfloor C\rfloor}, \bot \sqsubseteq C^{\langle\rangle})) + T(i_{C,M}) + \\
&\quad |\Delta|(k' + T(C^{\langle\rangle}[a]_\Delta) + T(\vdash_{\mathsf{CTL}})) \\
&\leq k + k_1\cdot|S|\cdot|\Delta|^2 + k_4\cdot|S|\cdot|\Delta| + \\
&\quad |\Delta|(k' + k_2\cdot|\Delta| + k_3\cdot(|S|+|R|)\cdot|\Delta|) \\
&= k + k_1\cdot|S|\cdot|\Delta|^2 + k_4\cdot|S|\cdot|\Delta| + \\
&\quad k'\cdot|\Delta| + k_2\cdot|\Delta|^2 + k_3\cdot(|S|+|R|)\cdot|\Delta|^2 \\
&\leq (k + k' + k_1 + k_2 + k_3 + k_4)\cdot(|S|+|R|)\cdot|\Delta|^2 \\
&\in \mathcal{O}((|S|+|R|)\cdot|\Delta|^2)
\end{aligned}$$

Hence, Equation (6.21) is shown for each $|C| \in \mathbb{N}_1$, which proves Proposition 6.3.48. □

Remark 6.3.49 (Runtime Complexity of getInterpretation)

Proposition 6.3.41 shows that **getInterpretation** is optimal w.r.t. the complexity of calculating the interpretation of $\mathcal{ALC}$CTL concepts as shown in Lemma 6.3.28.

The algorithm for **getInterpretation** is well suited for parallel computation. This is because every iteration of both **for**-loops in **getInterpretation** can be executed independently. In addition, the calculation of the structure mapping $sm(M^{\lfloor C\rfloor}, \bot \sqsubseteq C^{\langle\rangle})$ and of the interpretation function $i_{C,M}$ can be done in at least $|\Delta|$ independent threads. This leads to a speed up by factor $|\Delta|$ on a machine with $|\Delta|$ processors. Since $|\Delta|$ is between 100 and 100000 in application relevant scenarios, this is a large source of parallelism. □

6.3.3.3. Optimizations

Some optimizations can further speed up the performance of **verify** and **getInterpretation** in practical applications.

Avoiding Recalculation of Common Sub-Expressions

verify calls **getInterpretation** for every top-level concept C in the formula f to check. **getInterpretation** recursively calls itself for every sub-concept in C. If a formula f contains the same concept C several times, **getInterpretation** is called for each occurrence of C in f and hence the interpretation of C is calculated several times. Since the interpretation of C does not depend on the context, in which C occurs in f, it is sufficient to calculate the interpretation of every concept C in f once and then reuse the result for each further occurrence of C in f by applying *dynamic programming* [CLRS01].

Looking at a single formula f, recurrent complex concepts in f may appear to be rare. This is certainly not the case when checking a larger set of formulae which is typically done in application scenarios. Hence, a dynamic programming technique is particularly efficient when enhancing **verify / getInterpretation** towards checking a *set* of $\mathcal{ALC}$CTL formulae within a single call.

Avoiding Inefficient CTL Mappings

To make correctness and runtime analysis easier all validity checks in **verify** and **getInterpretation** are delegated to a given CTL model checking method $\vdash_{\mathsf{CTL}}$ using sound and complete mappings of $\mathcal{ALC}$CTL structures onto CTL structures and $\mathcal{ALC}$CTL expressions onto CTL formulae.

Most of the validity checks are actually quite easy. For instance, consider the formula $C \sqsubseteq D$. If the interpretation of concepts C and D is known at all states $s \in S$, the set of states, at which $C \sqsubseteq D$ holds in a model $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$, can easily be determined. By $\mathcal{ALC}$CTL semantics (Definition 6.2.8), it holds:

$$LS_{M,C\sqsubseteq D} = \{s \in S \mid M, s \models C \sqsubseteq D\} = \{s \in S \mid C^{I(s)} \subseteq D^{I(s)}\}$$

I.e. computing the label set $LS_{M,C\sqsubseteq D}$ can be done by a subset check $C^{I(s)} \subseteq D^{I(s)}$ for all states $s \in S$. Considering such a simple task, it appears a tremendous overhead to translate the formula $C \sqsubseteq D$ to a large CTL formula $fm(M, C \sqsubseteq D) = \bigwedge_{a\in\Delta}(C(a) \rightarrow D(a))$ as well as the $\mathcal{ALC}$CTL structure M to a CTL structure $sm(M, C \sqsubseteq D)$ and then to calculate the set $LS_{M,\, C\sqsubseteq D} = \{s \in S \mid sm(M, C \sqsubseteq D), s \vdash_{\mathsf{CTL}} fm(M, C \sqsubseteq D)\}$ via CTL model checking $\vdash_{\mathsf{CTL}}$.

In an implementation of **verify** and **getInterpretation**, dedicated validity checking methods can be combined with checks via a CTL reduction. It is possible to do the easy cases, e.g. checks of non-temporal connectives, internally and delegate the checks of complex temporal connectives to an external CTL model checker.

6.3.4. Counterexamples

If the verification of a formula p for a temporal structure M fails, a counterexample is provided that indicates the error location within the document.

Recall that the $\mathcal{ALC}$CTL model checking problem is reduced to a CTL model checking problem in the $\mathcal{ALC}$CTL model checking algorithm **verify** (Algorithm 6.3.37). Recall further that for CTL model checking always a finite counterexample $ce \in ET_{S,R,s}$ (Definition 3.2.14) can be provided if a CTL formula p is violated at some state $s \in S$ of a finite CTL temporal structure (S, R, L) (Remark 3.2.15). Since the CTL reduction of the $\mathcal{ALC}$CTL model checking problem is sound and complete (Proposition 6.3.39), a counterexample for the CTL reduction is also a valid counterexample for the corresponding $\mathcal{ALC}$CTL model checking problem.

The subsequent propositions formalize the considerations above.

Proposition 6.3.50 ($\mathcal{ALC}$CTL Counterexample Soundness)

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure, $f \in \mathcal{ALC}\mathsf{CTL}$ a formula, and $s \in S$ a state.

Let $f' = fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ and $M' = sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ be the CTL reductions of formula f and temporal structure M as applied in the $\mathcal{ALC}$CTL model checking algorithm **verify** (Algorithm 6.3.37).

Then every counterexample $ce \in ET_{S,R,s}$ for $M', s \models_{\mathsf{CTL}} f'$ (Definition 3.2.14) is also a counterexample for $M, s \models_{\mathcal{ALC}\mathsf{CTL}} f$.

Proof:

Let $ce = (s_0, ..., s_n) \in S^{n+1}$ be a counterexample for $M', s \models_{\mathsf{CTL}} f'$. Since S, R are identical in M' and M (Definitions 6.3.4 and 6.3.19) the path $(s_0, ..., s_n)$ is also an execution trace in M starting from state $s_0 = s$. Since $M', s \models_{\mathsf{CTL}} f'$ is equivalent to $M, s \models_{\mathcal{ALC}\mathsf{CTL}} f$ (Corollary 6.3.34) and the syntax and semantics of $\mathcal{ALC}$CTL and CTL coincide at the level of formulae when interpreting subsumption expressions $C \sqsubseteq D$ as atomic propositions, ce is a counterexample for $M, s \models_{\mathcal{ALC}\mathsf{CTL}} f$. □

Proposition 6.3.51 ($\mathcal{ALC}$CTL Counterexample Completeness)

There is a finite counterexample for any finite $\mathcal{ALC}$CTL temporal structure M and violated $\mathcal{ALC}$CTL formula p.

Let $M = (S, R, \Delta, I) \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$ be a finite $\mathcal{ALC}$CTL temporal structure, $s \in S$ a state, and f an $\mathcal{ALC}$CTL formula such that $M, s \not\models f$. Then there is an execution trace $ce \in ET_{S,R,s}$ (Definition 3.2.14) such that ce is a counterexample for $M, s \models p$.

Proof:

$M, s \not\models_{\mathcal{ALC}\mathsf{CTL}} f$ implies by Corollary 6.3.34 that $M', s \not\models_{\mathsf{CTL}} f'$ with $M' = sm(M^{\mathcal{C}_f}, f^{\langle\rangle})$ being a finite CTL temporal structure and $f' = fm(M^{\mathcal{C}_f}, f^{\langle\rangle})$ being a CTL formula.

Then there is a finite execution trace $ce \in ET_{S,R,s}$ such that ce is a counterexample for $M', s \models_{\mathsf{CTL}} f'$ [CGP02c]. By Proposition 6.3.50, ce is also a counterexample for $M, s \models_{\mathcal{ALC}\mathsf{CTL}} f$ which shows the proposition. □

In the sequel, the major components of the $\mathcal{ALC}$CTL model checking procedure, including the generation of counterexamples, are demonstrated in a sample scenario.

Example 6.3.52 ($\mathcal{ALC}$CTL Model Checking and Counterexamples)

Consider the $\mathcal{ALC}$CTL temporal structure $M = (S, R, \Delta, I)$ where

- $S = \{intro, defTree, exaTree, concl\}$
- $R = \{(intro, defTree), (intro, exaTree),$
  $(defTree, concl), (exaTree, concl), (concl, concl)\}$
- $\Delta = \{Tree, BinTree\}$
- I is a temporal interpretation of atomic concepts $definedTopic$, $exemplifiedTopic \in AC$ at states $s \in S$ as follows:

$$\begin{aligned} definedTopic^{I(intro)} &= exemplifiedTopic^{I(intro)} = \emptyset \\ definedTopic^{I(defTree)} &= exemplifiedTopic^{I(exaTree)} = \{Tree, BinTree\} \\ definedTopic^{I(exaTree)} &= exemplifiedTopic^{I(defTree)} = \emptyset \\ definedTopic^{I(concl)} &= exemplifiedTopic^{I(concl)} = \emptyset \end{aligned}$$

Figure 6.4 depicts the sample temporal structure M.

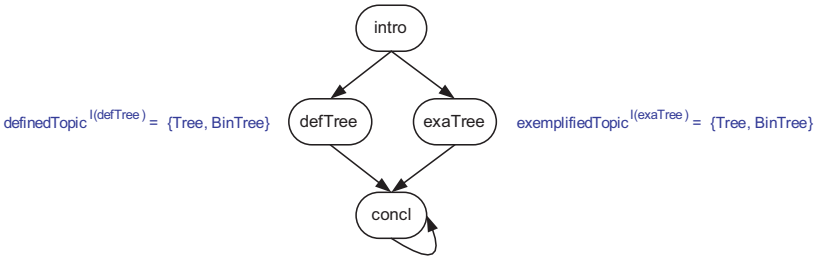

Figure 6.4.: temporal structure for illustrating counterexamples

Assume, M is checked against the $\mathcal{ALC}$CTL formula

$$f = \mathsf{AG}\ (definedTopic \sqsubseteq \mathsf{EF}\ exemplifiedTopic)$$

"Everywhere within the document: each defined topic is eventually exemplified at some path."

Then $M, intro \not\models f$ because from state $intro$ state $defTree$ is reachable, $definedTopic^{I(defTree)} = \{Tree, BinTree\}$, and from state $defTree$ no state other than $concl$ is reachable.

As a result, $(\mathsf{EF}\ exemplifiedTopic)^{I(defTree)} = \emptyset$ and thus

$$(\mathsf{EF}\ exemplifiedTopic)^{I(defTree)} \not\supseteq definedtopic^{I(defTree)}$$

which is equivalent to

$$M, defTree \not\models definedTopic \sqsubseteq \mathsf{EF}\ exemplifiedTopic$$

Since state $defTree$ can be reached on some path from state $intro$, we get

$$M, intro \not\models \mathsf{AG}(definedTopic \sqsubseteq \mathsf{EF}\ exemplifiedTopic)$$

verify checks $M, intro \models f$ via the CTL reduction $M', intro \vdash_{\mathsf{CTL}} f'$ with $M' = sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ and $f' = fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$.

$M^{\mathcal{C}_f} = (S, R, \Delta, I^{\mathcal{C}_f, M})$ is the temporal structure M with an interpretation $I^{\mathcal{C}_f, M}$ extended to the top-level concepts $\mathcal{C}_f$ of formula f.

For the given scenario, we get:

- $\mathcal{C}_f = \{definedTopic, \mathsf{EF}\ exemplifiedTopic\}$
- $I^{\mathcal{C}_f, M}$ is a temporal interpretation of atomic concepts $\langle definedTopic\rangle$, $\langle \mathsf{EF}\ exemplifiedTopic\rangle \in AC$ at states $s \in S$ as follows:

$$\begin{aligned}
\langle definedTopic\rangle^{I^{\mathcal{C}_f, M}(s)} &= definedTopic^{I(s)} \text{ for each } s \in S \\
\langle \mathsf{EF}\ exemplifiedTopic\rangle^{I^{\mathcal{C}_f, M}(intro)} &= (\mathsf{EF}\ exemplifiedTopic)^{I(intro)} \\
&= \{Tree, BinTree\} \\
\langle \mathsf{EF}\ exemplifiedTopic\rangle^{I^{\mathcal{C}_f, M}(defTree)} &= (\mathsf{EF}\ exemplifiedTopic)^{I(defTree)} \\
&= \emptyset \\
\langle \mathsf{EF}\ exemplifiedTopic\rangle^{I^{\mathcal{C}_f, M}(exaTree)} &= (\mathsf{EF}\ exemplifiedTopic)^{I(exaTree)} \\
&= \{Tree, BinTree\} \\
\langle \mathsf{EF}\ exemplifiedTopic\rangle^{I^{\mathcal{C}_f, M}(concl)} &= (\mathsf{EF}\ exemplifiedTopic)^{I(concl)} \\
&= \emptyset
\end{aligned}$$

$f^{\langle\rangle}$ is formula f with all top-level concepts being interpreted as atomic concepts:

$$f^{\langle\rangle} = \mathsf{AG}\ (\langle definedTopic\rangle \sqsubseteq \langle \mathsf{EF}\ exemplifiedTopic\rangle)$$

The structure mapping $sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ translates the $\mathcal{ALC}$CTL temporal structure $M^{\mathcal{C}_f}$ to an equivalent CTL temporal structure $M' = (S, R, L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}})$ w.r.t. formula $f^{\langle\rangle}$ where $L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}}$ is a labelling $S \rightarrow \mathcal{P}(AP)$ of states with sets of atomic propositions such that

$$\begin{aligned} L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}}(s) \quad = \quad & \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f^{\langle\rangle}} \wedge a \in \mathcal{A}^{I^{\mathcal{C}_f,M}(s)}\} \cup \\ & \{\mathcal{R}(a,b) \in AP \mid \mathcal{R} \in AR_{|f^{\langle\rangle}} \wedge (a,b) \in \mathcal{R}^{I^{\mathcal{C}_f,M}(s)}\} \end{aligned}$$

(Definition 6.3.4).

As for our sample case the set of atomic concepts $AC_{|f^{\langle\rangle}}$ of formula $f^{\langle\rangle}$ is

$$AC_{|f^{\langle\rangle}} = \{\langle definedTopic\rangle, \langle \mathsf{EF}\ exemplifiedTopic\rangle\}$$

The set of atomic roles in $f^{\langle\rangle}$ is $AR_{|f^{\langle\rangle}} = \emptyset$. Hence,

$$L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}}(s) \quad = \quad \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f^{\langle\rangle}} \wedge a \in \mathcal{A}^{I^{\mathcal{C}_f,M}(s)}\}$$

for $s \in S$.

The relevant set of atomic propositions AP are

{	$\langle definedTopic\rangle(Tree)$,	representing "$Tree$ is a defined topic at the current state".
	$\langle definedTopic\rangle(BinTree)$	representing "$BinTree$ is a defined topic at the current state".
	$\langle \mathsf{EF}\ exemplifiedTopic\rangle(Tree)$	representing "on some path starting from the current state, $Tree$ is eventually an exemplified topic".
	$\langle \mathsf{EF}\ exemplifiedTopic\rangle(BinTree)$	representing "on some path starting from the current state, $BinTree$ is eventually an exemplified topic".
}		

For the propositional labelling of states, we get (cf. Figure 6.5):

$$
\begin{aligned}
L_{I^{\mathcal{C}_f,M},\, f^{\langle\rangle}}(intro) &= \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f^{\langle\rangle}} \wedge a \in \mathcal{A}^{I^{\mathcal{C}_f,M}(intro)}\} \\
&= \{\langle \mathsf{EF}\ exemplifiedTopic\rangle(a) \in AP \mid \\
&\quad a \in \langle \mathsf{EF}\ exemplifiedTopic\rangle^{I^{\mathcal{C}_f,M}(intro)}\} \\
&= \{\langle \mathsf{EF}\ exemplifiedTopic\rangle(Tree), \\
&\quad \langle \mathsf{EF}\ exemplifiedTopic\rangle(BinTree)\} \\
L_{I^{\mathcal{C}_f,M},\, f^{\langle\rangle}}(defTree) &= \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f^{\langle\rangle}} \wedge a \in \mathcal{A}^{I^{\mathcal{C}_f,M}(defTree)}\} \\
&= \{\langle definedTopic\rangle(a) \in AP \mid \\
&\quad a \in \langle definedTopic\rangle^{I^{\mathcal{C}_f,M}(defTree)}\} \\
&= \{\langle definedTopic\rangle(Tree), \langle definedTopic\rangle(BinTree)\} \\
L_{I^{\mathcal{C}_f,M},\, f^{\langle\rangle}}(exaTree) &= \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f^{\langle\rangle}} \wedge a \in \mathcal{A}^{I^{\mathcal{C}_f,M}(exaTree)}\} \\
&= L_{I^{\mathcal{C}_f,M},\, f^{\langle\rangle}}(intro) \\
&= \{\langle \mathsf{EF}\ exemplifiedTopic\rangle(Tree), \\
&\quad \langle \mathsf{EF}\ exemplifiedTopic\rangle(BinTree)\} \\
L_{I^{\mathcal{C}_f,M},\, f^{\langle\rangle}}(concl) &= \{\mathcal{A}(a) \in AP \mid \mathcal{A} \in AC_{|f^{\langle\rangle}} \wedge a \in \mathcal{A}^{I^{\mathcal{C}_f,M}(concl)}\} \\
&= \emptyset
\end{aligned}
$$

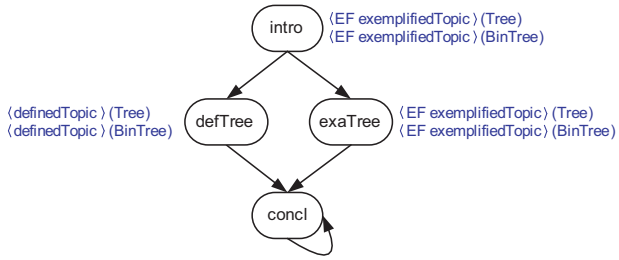

Figure 6.5.: propositional labelling $L_{I^{\mathcal{C}_f,M},\, f^{\langle\rangle}}$ of temporal structure $sm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$

The formula mapping $fm(M^{\mathcal{C}_f},\ f^{\langle\rangle})$ translates the $\mathcal{ALC}$ connectives in $f^{\langle\rangle}$ to equivalent Boolean expressions for finite domains Δ. The only $\mathcal{ALC}$ connective in $f^{\langle\rangle}$ is $\sqsubseteq$ since all concepts in $f^{\langle\rangle}$ are atomic by definition.

Hence,

$$\begin{aligned} f' &= fm(M^{\mathcal{C}_f},\ f^{\langle\rangle}) \\ &= \mathsf{AG}\,(\textstyle\bigwedge_{a\in\Delta}(\langle definedTopic\rangle[a]_\Delta \rightarrow \langle \mathsf{EF}\ exemplifiedTopic\rangle[a]_\Delta)) \\ &= \mathsf{AG}\,(\textstyle\bigwedge_{a\in\Delta}(\langle definedTopic\rangle(a) \rightarrow \langle \mathsf{EF}\ exemplifiedTopic\rangle(a))) \\ &= \mathsf{AG}\,((\langle definedTopic\rangle(Tree) \rightarrow \langle \mathsf{EF}\ exemplifiedTopic\rangle(Tree))\wedge \\ &\quad (\langle definedTopic\rangle(BinTree) \rightarrow \langle \mathsf{EF}\ exemplifiedTopic\rangle(BinTree))) \end{aligned}$$

Then $M', intro \not\models_{\mathsf{CTL}} f'$, which is demonstrated by the counterexample $ce = (intro, defTree)$ with $ce \in ET_{S,R,intro}$.

ce is a path from state $intro$ to state $defTree$, at which

$$\begin{aligned} &M', defTree \not\models_{\mathsf{CTL}} \\ &\quad (\langle definedTopic\rangle(Tree) \rightarrow \langle \mathsf{EF}\ exemplifiedTopic\rangle(Tree))\wedge \\ &\quad (\langle definedTopic\rangle(BinTree) \rightarrow \langle \mathsf{EF}\ exemplifiedTopic\rangle(BinTree)) \end{aligned}$$

This because $\langle definedTopic\rangle(Tree) \in L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}}(defTree)$

i.e. $\langle definedTopic\rangle(Tree)$ holds in M' at state $defTree$, but

$\langle \mathsf{EF}\ exemplifiedTopic\rangle(Tree) \notin L_{I^{\mathcal{C}_f,M},\ f^{\langle\rangle}}(defTree)$

i.e. $\langle \mathsf{EF}\ exemplifiedTopic\rangle(Tree)$ does not hold in M' at state $defTree$ (cf. Figure 6.5).

By Proposition 6.3.50 ce, is also a counterexample to $M, intro \models_{\mathcal{ALC}\mathsf{CTL}} f$: ce is path from state $intro$ to the state $defTree \in S$, at which

$$definedTopic^{I(defTree)} \not\subseteq (\mathsf{EF}\ exemplifiedTopic)^{I(defTree)}$$

Hence, ce demonstrates that

$$M, intro \not\models \mathsf{AG}(definedTopic \sqsubseteq \mathsf{EF}\ exemplifiedTopic)$$

□

Remark 6.3.53 ($\mathcal{ALC}$CTL Counterexamples)

Example 6.3.52 demonstrates that counterexamples returned by CTL model checking of the transformed structure M' and formula f' are counterexamples w.r.t. the original $\mathcal{ALC}$CTL structure M and formula f. Hence, at the level of counterexamples we do not need to worry about the bulky transformations on the bases of which $\mathcal{ALC}$CTL formulae are verified. Although the CTL transformations and the overall model checking algorithm may appear complex, the counterexamples show a quite simple structure: they are paths from a selected starting state to a state at which the verified formula is violated.

As for the simple scenario of Example 6.3.52, the information delivered by a counterexample appears small. In application scenarios, however, there are usually hundreds or thousands of states and an exponential or infinite number of paths within the state transition system. Considering this, an isolated finite path that demonstrates a specification violation is a valuable information.

Obviously, isolating a violating path within the verified structure is just a first step in identifying the root cause of an error. In Example 6.3.52, the reason for the specification violation is that state $exaTree$ is not reachable from the definition $defTree$ of $Tree$ and $BinTree$. Although the structure contains an example of each defined concept, $\mathsf{AG}\,(definedTopic \sqsubseteq \mathsf{EF}\; examplifiedTopic)$ is violated because the examples of concepts cannot be reached *in the sequel* of their definition. In another scenario, the specification could have been violated because the document does not contain any example of either $Tree$ or $BinTree$ or the definition defines the wrong terms.

Supporting the error identification and correction process in interaction with the user remains a topic of future research. □

6.4. Applying $\mathcal{ALC}$CTL to Checking Documents

So far, we have introduced $\mathcal{ALC}$CTL and its model checking problem in an application independent way. In principal, $\mathcal{ALC}$CTL can be used to represent properties of any domain that can be modelled in terms of an $\mathcal{ALC}$CTL temporal structure. We apply $\mathcal{ALC}$CTL for the verification of documents based on the semantic model introduced in chapter 5.

Recall, the semantic model of a document d has been introduced as a structure $SM_d := (NS_d, \mathbf{KB}_d, kmap)$ (Definition 5.3.24). The narrative structure $NS_d = (CU_d, BOD_d, proceed)$ consists of a set of content units CU_d of the document, a starting unit BOD_d, and the narrative relation $proceed$ between content units.

Knowledge about the content of the document is represented by the set of DL knowledge bases $\mathbf{KB}_d = \{KB_U \mid U \in CU_d\}$ where KB_U denotes the knowledge base representing information about content unit $U \in CU_d$.

We observe that there is a structural mismatch between semantic models of documents and $\mathcal{ALC}$CTL temporal structures (S, R, Δ, I) which the semantics of $\mathcal{ALC}$CTL formulae is defined upon. Hence, target criteria for the content and structure represented by $\mathcal{ALC}$CTL formulae cannot be evaluated directly on a semantic model of a document. For using $\mathcal{ALC}$CTL as a specification formalism for document properties, we have to clarify first when a (set of) $\mathcal{ALC}$CTL formula(e) holds for a semantic model SM_d of a document.

In the sequel, we define the document verification problem and show how semantic models of documents can be verified using $\mathcal{ALC}$CTL model checking.

6.4.1. Basic Definitions

For the verification of the semantic model SM_d of a document d against a set of $\mathcal{ALC}$CTL formulae F, we define a mapping of SM_d onto an $\mathcal{ALC}$CTL temporal structure M and then verify whether F holds in M by model checking.

Definition 6.4.1 ($\mathcal{ALC}$CTL Temporal Structure of a Document)

Let $NS_d = (CU_d, BOD_d, proceed)$ be the narrative structure of document d and $\mathbf{KB}_d = \{KB_U \mid U \in CU_d\}$ be a locally consistent knowledge representation of document d.

- $S_d := CU_d$ denotes the set of *states* of document d, i.e. the content units of document d (Definition 5.2.1) are treated as states of the temporal structure of d.
- $R_d := proceed$ denotes the *transition relation* of document d, i.e. the narrative relation $proceed$ (Definition 5.2.4) is treated as the transition relation on states.
- $\Delta_d := IV_{cKB_d}$ denotes the *domain* of document d. Recall, IV_{cKB_d} is the set of individuals which occur in some assertion of some local knowledge base $KB_U \in \mathbf{KB}_d$ (Definition 5.3.15 *knowledge domain*).
- I_d denotes the *temporal interpretation* of document d iff

  $$\mathcal{A}^{I_d(s)} = \{a \in IV_{KB_s} \mid KB_s \models \mathcal{A}(a)\} \text{ and}$$
  $$\mathcal{R}^{I_d(s)} = \{(a,b) \in IV_{KB_s} \times IV_{KB_s} \mid KB_s \models \mathcal{R}(a,b)\}$$

  for each $\mathcal{A} \in AC$, $\mathcal{R} \in AR$, $s \in S_d$, and IV_{KB_s} being the set of individuals occurring in some statement of the local knowledge base KB_s (Definition 5.3.15 *local knowledge domain*).
- $M_d := (S_d, R_d, \Delta_d, I_d)$ denotes the *temporal structure* of document d.

□

Proposition 6.4.2 ($\mathcal{ALC}$CTL Temporal Structure of a Document)

The temporal structure M_d of a document d is a finite $\mathcal{ALC}$CTL temporal structure, in symbols, $M_d \in \mathbf{fM}_{\mathcal{ALC}\mathsf{CTL}}$.

Proof:

M_d is a finite $\mathcal{ALC}$CTL temporal structure because

- S_d is nonempty and finite. This is because $S_d = CU_d$ and CU_d is nonempty and finite (Definition 5.2.1).
- R_d is a left-total relation on S_d because $R_d = proceed$, $S_d = CU_d$, and proceed is a left-total relation on CU_d (Definition 5.2.19).
- $\Delta_d = IV_{cKB_d}$ is nonempty by Definition 5.3.24 and finite by Corollary 5.3.16.

- I_d is a temporal interpretation if $IV_{KB_s} \subseteq \Delta_d$ for each $s \in S_d$. This is the case because $IV_{KB_s} \subseteq IV_{cKB_d}$ (Corollary 5.3.17) and $IV_{cKB_d} = \Delta_d$ (Definition 6.4.1).

□

Example 6.4.3 ($\mathcal{ALC}$CTL Temporal Structure of a Document)

Consider a document d consisting of three content units $CU_d = \{sec1, sec2, sec3\}$ where $BOD_d = sec1$ is the beginning of the document. Further there are narrative relations $proceed = \{(sec1, sec2), (sec1, sec3), (sec2, sec3), (sec3, sec3)\}$, i.e. starting from $sec1$ the user may go to $sec2$ or go immediately to $sec3$ that is the end unit of the document.

Knowledge about each content unit is represented in a set of local fact bases $\mathbf{lAB}_d = \{AB_{sec1}, AB_{sec2}, AB_{sec3}\}$ where

$$\begin{aligned} AB_{sec1} &= \{hasObjective(intro, R180), hasObjective(intro, R180\_Handling)\} \\ AB_{sec2} &= \{teaches(L1, R180)\} \\ AB_{sec3} &= \{teaches(L2, R180\_Handling)\} \end{aligned}$$

I.e. content unit $sec1$ contains an introduction, represented by the individual $intro$, which states two objectives: "robot type R180" represented by the individual $R180$ and "Handling of robots of type R180" represented by the individual $R180\_Handling$. Unit $sec2$ contains Lesson 1 represented by individual $L1$. $L1$ teaches topic $R180$. Unit $sec3$ contains Lesson 2 represented by individual $L2$. $L2$ teaches topic $R180\_Handling$.

Further, no global facts about the document are available, i.e. $gAB_d = \emptyset$.

As a reference ontology consider

$RO = \{$	$objectiveOf \doteq hasObjective^-$,	$objectiveOf$ is the inverse of role $hasObjective$.
	$addressedBy \doteq addresses^-$,	$addressedBy$ is the inverse of role $addresses$.
	$\top \sqsubseteq \forall addresses.Topic$,	$Topic$ is the range of role $addresses$.
	$teaches \sqsubseteq addresses$,	$teaches$ is a sub-role of $addresses$.
	$\exists teaches.Topic \doteq Lesson$,	$Lesson$ is the class of objects that teach some topic.
	$\exists objectiveOf.\top \sqsubseteq MajorTopic$,	$MajorTopic$ is the domain of role $objectiveOf$.
	$MajorTopic \sqsubseteq Topic$	Every major topic is a topic.
$\}$		

By Definition 5.3.10, we get:

$$\mathbf{KB}_d = \{KB_{sec1}, KB_{sec2}, KB_{sec3}\}$$

where

$$\begin{aligned}
KB_{sec1} &= RO \cup gAB_d \cup AB_{sec1} \\
&= RO \cup \{hasObjective(intro, R180), \\
&\qquad hasObjective(intro, R180_Handling)\} \\
KB_{sec2} &= RO \cup gAB_d \cup AB_{sec2} \\
&= RO \cup \{teaches(L1, R180)\} \\
KB_{sec3} &= RO \cup gAB_d \cup AB_{sec3} \\
&= RO \cup \{teaches(L2, R180_Handling)\}
\end{aligned}$$

and

$$\begin{aligned}
cKB_d &= RO \cup AB_{U_1} \cup AB_{U_2} \cup AB_{U_3} \\
&= \{objectiveOf \doteq hasObjective^{-}, \\
&\quad addressedBy \doteq addresses^{-}, \\
&\quad \top \sqsubseteq \forall addresses.Topic, \\
&\quad teaches \sqsubseteq addresses, \\
&\quad \exists teaches.Topic \doteq Lesson, \\
&\quad \exists objectiveOf.\top \sqsubseteq MajorTopic, \\
&\quad MajorTopic \sqsubseteq Topic, \\
&\quad hasObjective(intro, R180), hasObjective(intro, R180_Handling), \\
&\quad teaches(L1, R180), teaches(L2, R180_Handling)\}
\end{aligned}$$

The knowledge domain (Definition 5.3.15) is

$$IV_{cKB_d} = \{intro, L1, L2, R180, R180\_Handling\}$$

The temporal structure $M_d = (S_d, R_d, \Delta_d, I_d)$ of document d is such that

- $S_d = CU_d = \{sec1, sec2, sec3\}$.
- $R_d = proceed = \{(sec1, sec2), (sec1, sec3), (sec2, sec3), (sec3, sec3)\}$.
- $\Delta_d = IV_{cKB_d} = \{intro, L1, L2, R180, R180\_Handling\}$.

$$\begin{aligned}
IV_{KB_{sec1}} &= \{intro, R180, R180_Handling\} \\
IV_{KB_{sec2}} &= \{L1, R180\} \\
IV_{KB_{sec3}} &= \{L2, R180_Handling\}
\end{aligned}$$

- I_d is such that

$$
\begin{aligned}
hasObjective^{I_d(sec1)} &= \{(a,b) \in IV_{KB_{sec1}} \times IV_{KB_{sec1}} \mid \\
&\quad KB_{sec1} \models hasObjective(a,b)\} \\
&= \{(intro, R180), (intro, R180\_Handling)\} \\
hasObjective^{I_d(sec2)} &= hasObjective^{I_d(sec3)} = \emptyset \\
MajorTopic^{I_d(sec1)} &= \{a \in IV_{KB_{sec1}} \mid KB_{sec1} \models MajorTopic(a)\} \\
&= \{R180, R180\_Handling\} \\
MajorTopic^{I_d(sec2)} &= MajorTopic^{I_d(sec3)} = \emptyset \\
addressedBy^{I_d(sec1)} &= \{(a,b) \in IV_{KB_{sec1}} \times IV_{KB_{sec1}} \mid \\
&\quad KB_{sec1} \models addressedBy(a,b)\} = \emptyset \\
addressedBy^{I_d(sec2)} &= \{(R180, L1)\} \\
addressedBy^{I_d(sec3)} &= \{(R180\_Handling, L2)\} \\
Lesson^{I_d(sec1)} &= \{a \in IV_{KB_{sec1}} \mid \\
&\quad KB_{sec1} \models Lesson(a)\} = \emptyset \\
Lesson^{I_d(sec2)} &= \{L1\} \\
Lesson^{I_d(sec3)} &= \{L2\} \\
&\ldots
\end{aligned}
$$

Figure 6.6 depicts the temporal structure of document d with selected interpretations of atomic concepts and roles.

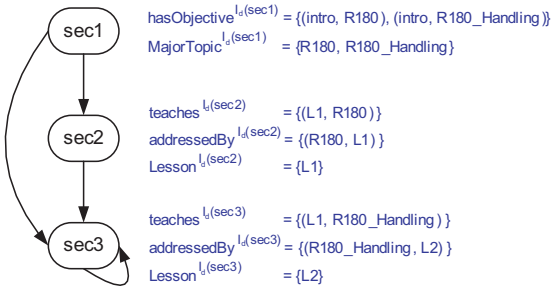

Figure 6.6.: a sample temporal document structure

The temporal interpretation of each atomic concept and role in a state $s \in S_d$ is calculated as the set of assertions that are logically implied by the local knowledge base KB_s of the content unit $s \in CU_d$. For instance, $Lesson^{I_d(sec2)} = \{L1\}$ can be seen as follows.

By definition of I_d (Definition 6.4.1), it holds:

$$\begin{aligned} Lesson^{I_d(sec2)} &= \{a \in IV_{KB_{sec2}} \mid KB_{sec2} \models Lesson(a)\} \\ &= \{a \in \{L1, R180\} \mid KB_{sec2} \models Lesson(a)\} \end{aligned}$$

$\{a \in \{L1, R180\} \mid KB_{sec2} \models Lesson(a)\}$ can be obtained as follows:

- $KB_{sec2} \models Lesson(L1)$ because

$$\begin{aligned} \{teaches(L1, R180), & \\ teaches \sqsubseteq addresses\} &\models addresses(L1, R180), \\ \{addresses(L1, R180), & \\ \top \sqsubseteq \forall addresses.Topic\} &\models Topic(R180), \\ \{teaches(L1, R180), & \\ Topic(R180)\} &\models (\exists teaches.Topic)(L1), \\ \{(\exists teaches.Topic)(L1), & \\ \exists teaches.Topic \doteq Lesson\} &\models Lesson(L1) \end{aligned}$$

- $KB_{sec2} \not\models Lesson(R180)$ because let $I = (\Delta^I, \cdot^I)$ be an interpretation such that $\Delta^I = \{L1^I, R180^I\}$ and

$$\begin{aligned} hasObjective^I &= objectiveOf^I = \emptyset \\ teaches^I &= \{(L1^I, R180^I)\} \\ addresses^I &= \{(L1^I, R180^I)\} \\ addressedBy^I &= \{(R180^I, L1^I)\} \\ Topic^I &= \{R180^I\} \\ Lesson^I &= \{L1^I\} \\ MajorTopic^I &= \emptyset \end{aligned}$$

  Then $I \models KB_{sec2}$ but $R180^I \notin Lesson^I$ and hence $I \not\models Lesson(R180)$ which contradicts $KB_{sec2} \models Lesson(R180)$ (Definition 3.1.31).

In summary, the state transition system (S_d, R_d) represents the narrative structure of the document. The domain Δ_d represents the set of objects the properties of which describe the content of the document. The temporal interpretation I_d represents the properties of objects at a content unit $U \in CU_d$, which are logically implied by the respective local knowledge base KB_U. KB_U contains local facts AB_U about the content unit U, global facts gAB_d about the document as a whole, and general background knowledge represented by the reference ontology RO.

Semantic criteria represented in $\mathcal{ALC}$CTL are evaluated w.r.t M_d (see Definition 6.4.4). □

Definition 6.4.4 ($\mathcal{ALC}$CTL Document Verification Problem)

Let M_d be the $\mathcal{ALC}$CTL temporal structure and BOD_d the beginning of document d. Let f be an $\mathcal{ALC}$CTL formula.

Then document d satisfies f (or f holds in d) iff $M_d, BOD_d \models f$. □

Example 6.4.5 ($\mathcal{ALC}$CTL Document Verification Problem)

Consider the $\mathcal{ALC}$CTL temporal structure M_d of document d as of Example 6.4.3 with beginning $BOD_d = sec1$ (Figure 6.6).

Consider the following $\mathcal{ALC}$CTL formulae:

$$\begin{aligned} f_1 &= MajorTopic \sqsubseteq \mathsf{EF}\ \exists addressedBy.Lesson \\ f_2 &= MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.Lesson \end{aligned}$$

Then f_1 holds in document d but document d does not satisfy f_2:

$$M_d, sec1 \models f_1$$
$$M_d, sec1 \not\models f_2$$

$M_d, sec1 \models f_1$ can be seen as follows:

In state $sec1$, it holds: $MajorTopic^{I_d(sec1)} = \{R180, R180\_Handling\}$ (cf. Example 6.4.3, Figure 6.6).

Hence, $M_d, sec1 \models f_1$ iff $R180 \in (\mathsf{EF}\ \exists addressedBy.Lesson)^{I_d(sec1)}$ and $R180\_Handling \in (\mathsf{EF}\ \exists addressedBy.Lesson)^{I_d(sec1)}$, i.e. iff there is some path $(s_0, s_1, ...) \in FP_{sec1}$ such that eventually a state s_i is reached at which holds $R180 \in (\exists addressedBy.Lesson)^{I_d(s_i)}$ and, in addition, there is a (not necessarily different) path $(s'_0, s'_1, ...) \in FP_{sec1}$ such that eventually a state s'_j is reached at which $R180\_Handling \in (\exists addressedBy.Lesson)^{I_d(s'_j)}$ holds.

In state $sec2$, it holds: $addressedBy^{I_d(sec2)} = \{(R180, L1)\}$ and $Lesson^{I_d(sec2)} = \{L1\}$ (Figure 6.6). This implies by $\mathcal{ALC}$CTL semantics (Definition 6.2.8): $R180 \in (\exists addressedBy.Lesson)^{I_d(sec2)}$.

In state $sec3$, it holds: $addressedBy^{I_d(sec3)} = \{(R180\_Handling, L2)\}$ and $Lesson^{I_d(sec3)} = \{L2\}$ (Figure 6.6). This implies by $\mathcal{ALC}$CTL semantics (Definition 6.2.8): $R180\_Handling \in (\exists addressedBy.Lesson)^{I_d(sec3)}$.

Since there is a path $(sec1, sec2, ...) \in FP_{sec1}$ such that $sec2$ is eventually reached from $sec1$ and there is a path $(sec1, sec2, sec3, ...) \in FP_{sec1}$ such that $sec3$ is eventually reached from $sec1$, we get:

$$MajorTopic^{I_d(sec1)} \subseteq (\mathsf{EF}\ \exists addressedBy.Lesson)^{I_d(sec1)}$$

which is equivalent to

$$M_d, sec1 \models MajorTopic \sqsubseteq \mathsf{EF}\ \exists addressedBy.Lesson$$

In contrast, $f_2 = MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.Lesson$ is violated by path $(sec1, sec3, sec3, ...) \in FP_{sec1}$ because for topic $R180 \in MajorTopic^{I_d(sec1)}$ there is no state s_i in $(sec1, sec3, sec3, ...)$ such that $R180 \in (\exists addressedBy.Lesson)^{I_d(s_i)}$. Hence, major topic $R180$ is not eventually addressed by a learning unit in *all* paths starting from $sec1$ as postulated by f_2 and thus

$$MajorTopic^{I_d(sec1)} \not\subseteq (\mathsf{AF}\ \exists addressedBy.Lesson)^{I_d(sec1)}$$

which is equivalent to

$$M_d, sec1 \not\models MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.Lesson$$

□

Remark 6.4.6 ($\mathcal{ALC}$CTL Document Verification Problem)

The document verification problem is modelled as an $\mathcal{ALC}$CTL model checking problem restricted to a single state BOD_d.

As a consequence the document verification problem is decidable and in polynomial time if the $\mathcal{ALC}$CTL temporal structure of the document is given. □

6.4.2. Verification Algorithm for Documents

Given the semantic model SM_d of a document d we can derive a document checking algorithm based on Definition 6.4.1 of the temporal document structure and the $\mathcal{ALC}$CTL model checking algorithm **verify** (Algorithm 6.3.37).

Algorithm 6.4.7 shows the structure of the document verification algorithm **docverify**.

The function **docverify** takes the following parameters as an input:

S_d = CU_d: the set of states of the $\mathcal{ALC}$CTL temporal structure of document d, which is equal to the set of content units CU_d (Definition 6.4.1).

s_d = BOD_d: the starting state of document d that represents the beginning of document d (Definition 6.4.1).

R_d = $proceed$: the narrative relation $proceed$ on content units (Definition 6.4.1).

$kmap$: a mapping $CU_d \rightarrow \mathbf{KB}_d : kmap(U) = KB_U$ of content units $U \in CU_d$ onto their local knowledge base $KB_U \in \mathbf{KB}_d$ of the knowledge representation $\mathbf{KB}_d$ of document d (Definition 5.3.10).

f : an $\mathcal{ALC}$CTL formula to check.

Algorithm 6.4.7 (Verification of Documents using $\mathcal{ALC}$CTL)

The following pseudo code sketches the verification algorithm **docverify** for documents.

function docverify($S_d, s_d, R_d, kmap, f$) {	
$AC_{\|f}$:= f.getAtomicConcepts();	Get the set of atomic concepts in f.
$AR_{\|f}$:= f.getAtomicRoles();	Get the set of atomic roles in f.
$\Delta_d := \emptyset$;	Initialize the interpretation domain Δ_d and
$I_d := \emptyset$;	the interpretation I_d of the temporal document structure M_d.
for each $s \in S_d$ {	Calculate the interpretation domain Δ_d and the interpretation I_d for the atomic concepts and roles in f.
$KB_s := kmap(s)$;	Get the local knowledge base KB_s of state s.
if not KB_s.isConsistent() **then**	Check KB_s for consistency (= satisfiability).
return errorInconsistentKB(s);	If inconsistent return an error.
$\Delta_d := \Delta_d \cup KB_s$.getIndividuals();	Add the individuals IV_{KB_s} of KB_s to the interpretation domain Δ_d.
for each $\mathcal{A} \in AC_{\|f}$ {	Calculate the interpretation $\cdot^{I_d(s)}$ of atomic concepts $AC_{\|f}$ in f.
$\mathcal{A}^{I_d(s)} := KB_s$.getInstances($\mathcal{A}$);	Get the instances of atomic concept $\mathcal{A}$ as implied by knowledge base KB_s.
$I_d := I_d \cup \{(\mathcal{A}, s) \mapsto \mathcal{A}^{I_d(s)}\}$;	Add the interpretation of $\mathcal{A}$ at state s to I_d.
}	
for each $\mathcal{R} \in AR_{\|f}$ {	Calculate the interpretation $\cdot^{I_d(s)}$ of atomic roles $AR_{\|f}$ in f.
$\mathcal{R}^{I_d(s)} := KB_s$.getFillers($\mathcal{R}$);	Get the fillers of atomic role $\mathcal{R}$ as implied by knowledge base KB_s.
$I_d := I_d \cup \{(\mathcal{R}, s) \mapsto \mathcal{R}^{I_d(s)}\}$;	Add the interpretation of $\mathcal{R}$ at state s to I_d.
}	
}	
if $\Delta_d = \emptyset$ **then**	Check Δ_d for emptyness.
return errorEmptyDomain();	If empty then return an error
return $s_d \in$ verify($S_d, R_d, \Delta_d, I_d, f$);	otherwise verify the temporal model $M_d = (S_d, R_d, \Delta_d, I_d)$ of document d against f
}	using the $\mathcal{ALC}$CTL model checking algorithm **verify**.

□

docverify returns an error "errorInconsistentKB(s)" if there is an unsatisfiable knowledge base $KB_s = kmap(s) \in \mathbf{KB}_d$, i.e. if KB_d is not a locally consistent knowledge representation of document d (Definition 5.3.20).

docverify returns an error "errorEmptyDomain()" if the knowledge domain $\Delta_d = IV_{cKB_d}$ is empty.

Otherwise **docverify** returns $true$ if the formula f holds in the $\mathcal{ALC}$CTL temporal structure $M_d = (S_d, R_d, \Delta_d, I_d)$ of document d at state s_d, in symbols $M_d, s_d \models f$, and $false$ if document d does not satisfy f, in symbols $M_d, s_d \not\models f$.

docverify uses the following sub-routines:

- f.getAtomicConcepts();

  returns the set $AC_{|f} = \{\mathcal{A} \in AC \mid \mathcal{A} \in f\}$ of atomic concepts occurring in formula f.

- f.getAtomicRoles();

  returns the set $AR_{|f} = \{\mathcal{R} \in AR \mid \mathcal{R} \in f\}$ of atomic roles occurring in formula f.

- KB_s.isConsistent();

  returns $true$ if the DL knowledge base KB_s is satisfiable (Definition 3.1.28). Otherwise KB_s.isConsistent() returns $false$.

  KB_s.isConsistent() is implemented by a call to an external DL reasoner such as Racer (section 3.1.5).

- KB_s.getIndividuals();

  returns the set of individuals IV_{KB_s} (Definition 5.3.15) which appear in some statement of knowledge base KB_s .

  KB_s.getIndividuals() is implemented by a call to an external DL reasoner such as Racer (section 3.1.5).

- KB_s.getInstances($\mathcal{A}$);

  returns the set $\{a \in IV_{KB_s} \mid KB_s \models \mathcal{A}(a)\}$ of instances of the atomic concept $\mathcal{A} \in AC$ as implied by knowledge base KB_s.

  A realization of KB_s.getInstances($\mathcal{A}$) is available as *instance retrieval service* that is offered by most DL reasoning systems (section 3.1.5).

- KB_s.getFillers($\mathcal{R}$);

  returns the set $\{(a, b) \in IV_{KB_s} \times IV_{KB_s} \mid KB_s \models \mathcal{R}(a, b)\}$ of role fillers of the atomic role $\mathcal{R} \in AR$ as implied by knowledge base KB_s.

  A realization of KB_s.getFillers($\mathcal{R}$) is available as *role filler retrieval service* which is offered by most DL reasoning systems (section 3.1.5).

- $kmap(s)$ stands for a sub-routine returning the local knowledge base KB_s for $s \in CU_d$ according to Definition 5.3.10.

- verify($S_d, R_d, \Delta_d, I_d, f$);

  is a call to the $\mathcal{ALC}$CTL model checking algorithm **verify** (Algorithm 6.3.37). It returns the set $\{s \in S_d \mid (S_d, R_d, \Delta_d, I_d), s \models f\}$ for a finite $\mathcal{ALC}$CTL temporal structure $(S_d, R_d, \Delta_d, I_d)$ and an $\mathcal{ALC}$CTL formula f.

Proposition 6.4.8 (Soundness and Completeness of docverify)

docverify is sound and complete:

Let

- $NS_d = (CU_d, BOD_d, proceed)$ be the narrative structure of a document d,
- $S_d = CU_d$ the set of content units,
- $s_d = BOD_d$ the beginning of document d,
- $R_d = proceed$ the narrative relation $proceed$,
- $kmap : CU_d \rightarrow \mathbf{KB}_d : kmap(U) = KB_U$ a knowledge mapping of content units $U \in CU_d$ onto their local knowledge bases $KB_U \in \mathbf{KB}_d$ where $\mathbf{KB}_d$ is a locally consistent knowledge representation of document d with a nonempty knowledge domain IV_{cKB_d}, and
- f an $\mathcal{ALC}$CTL formula.

Then docverify$(S_d, s_d, R_d, kmap, f) = true \Leftrightarrow M_d, s_d \models f$.

Proof:

Since $\mathbf{KB}_d$ is locally consistent and $KB_s = kmap(s) \in \mathbf{KB}_d$ for each $s \in S_d$, each KB_s is consistent and hence **docverify** does not abort because of an inconsistent knowledge base.

The set $S_d = CU_d$ is finite as a consequence of Definition 5.2.1. Also, the sets $AC_{|f}$ and $AR_{|f}$ are finite because f is finite and hence f contains finitely many different atomic concepts $AC_{|f}$ and roles $AR_{|f}$.

As a result, **docverify** reaches statement

if $\Delta_d = \emptyset$ **then** ...

in finite time.

Since the knowledge domain $IV_{cKB_d} \neq \emptyset$ and $IV_{cKB_d} = \bigcup_{U \in CU_d} IV_{KB_U}$ (Corollary 5.3.17), there is some $U \in CU_d$ such that $IV_{KB_U} \neq \emptyset$ and, because $S_d = CU_d$, there is some $s \in S_d$ such that $IV_{KB_s} \neq \emptyset$. Hence, there is some assignment of KB_s in **docverify** such that KB_s.getIndividuals() $= IV_{KB_s} \neq \emptyset$.

Since KB_s.getIndividuals() is added to Δ_d and no other assignments are made to Δ_d in the **for**-loops of **docverify**, the expression $\Delta_d = \emptyset$ evaluates to $false$ at the final **if** statement of **docverify** and hence verify$(S_d, R_d, \Delta_d, I_d, f)$ is called.

We show next: the parameter Δ_d of **verify** is equal to the knowledge domain IV_{cKB_d} of document d:

$$\Delta_d = IV_{cKB_d} \tag{6.22}$$

This is because **docverify** calculates

Δ_d	$=$	$\bigcup_{s \in S_d} KB_s$.getIndividuals(s)	Assignment of Δ_d in **docverify**.
	$=$	$\bigcup_{s \in S_d} IV_{KB_s}$	Def. of KB_s, KB_s.getIndividuals(s) and $kmap(s)$ in **docverify**.
	$=$	$\bigcup_{U \in CU_d} IV_{KB_U}$	$S_d = CU_d$.
	$=$	IV_{cKB_d}	Corollary 5.3.17.

Further, for each $\mathcal{A} \in AC_{|f}$, $\mathcal{R} \in AR_{|f}$, and $s \in S_d$, **docverify** calculates the sets $\mathcal{A}^{I_d(s)}$, $\mathcal{R}^{I_d(s)}$, and a representation I_d of the temporal interpretation function such that

$$\begin{aligned} \mathcal{A}^{I_d(s)} &= I_d(\mathcal{A}, s) = \{a \in IV_{KB_s} \mid KB_s \models \mathcal{A}(a)\} \text{ and} \\ \mathcal{R}^{I_d(s)} &= I_d(\mathcal{R}, s) = \{(a,b) \in IV_{KB_s} \times IV_{KB_s} \mid KB_s \models \mathcal{R}(a,b)\} \end{aligned} \tag{6.23}$$

where IV_{KB_s} is the set of individuals occurring in some statement of the local knowledge base KB_s.

This is because in **docverify**, f.getAtomicConcepts() $= AC_{|f}$ and

$\mathcal{A}^{I_d(s)}$	$=$	KB_s.getInstances($\mathcal{A}$)	Assignment of $\mathcal{A}^{I_d(s)}$.
	$=$	$\{a \in IV_{KB_s} \mid KB_s \models \mathcal{A}(a)\}$	Def. KB_s.getInstances($\mathcal{A}$).

for each $\mathcal{A} \in AC_{|f}$.

The analogous arguments hold for the calculation of the interpretation of atomic roles $\mathcal{R} \in AR_{|f}$ in **docverify**.

Hence, I_d is a sound representation of the interpretation of the $\mathcal{ALC}$CTL temporal structure M_d of document d according to Definition 6.4.1 and I_d is complete w.r.t. the interpretation of atomic concepts $AC_{|f}$ and roles $AR_{|f}$ occurring in formula f.

The truth of $M_d, s \models f$ for $s \in S_d$ depends on the interpretation of atomic concepts and roles of f only (Definition 6.2.8). The interpretations of atomic concepts and roles, which do not occur in f, are irrelevant.

Hence, I_d, as calculated in **docverify**, is a *complete* representation of the interpretation of the temporal model M_d as for model checking formula f.

By Equations (6.22), (6.23), and Definition 6.4.1, **docverify** calls **verify** with a sound and complete representation of the finite $\mathcal{ALC}$CTL temporal structure $M_d = (S_d, R_d, \Delta_d, I_d)$ of document d for model checking f. Since **verify** is sound and complete (Proposition 6.3.39), we get

verify($S_d, R_d, \Delta_d, I_d, f$) = $\{s \in S_d \mid M_d, s \models f\}$

As a result:

docverify($S_d, s_d, R_d, kmap, f$) $= true \Leftrightarrow s_d \in$ verify($S_d, R_d, \Delta_d, I_d, f$) $\Leftrightarrow s_d \in \{s \in S_d \mid M_d, s \models f\} \Leftrightarrow M_d, s_d \models f$. □

6.4.3. Analysis of Runtime Complexity

Proposition 6.4.9 (Runtime Complexity of docverify)

The runtime complexity of **docverify** is in EXPTIME.

- Let $NS_d = (CU_d, BOD_d, proceed)$ be the narrative structure of a document d.
- Let $S_d = CU_d$ be the set of content units.
- Let $s_d = BOD_d$ be the beginning of document d.
- Let $R_d = proceed$ be the narrative relation on content units.
- Let $kmap : CU_d \rightarrow \mathbf{KB}_d : kmap(U) = KB_U$ be a knowledge mapping of content units $U \in CU_d$ onto their local knowledge bases $KB_U \in \mathbf{KB}_d$ where $\mathbf{KB}_d$ is a locally consistent knowledge representation of document d with a nonempty knowledge domain $\Delta_d = IV_{cKB_d}$, $\mathbf{KB}_d$ is a set of knowledge bases in the description logic $\mathcal{AL}$, and cKB_d is the combined knowledge base of knowledge representation $\mathbf{KB}_d$.
- Let $max(kmap) := max(\{|kmap(U)| \mid U \in CU_d\})$ be the size of the largest local knoweldge base in $\mathbf{KB}_d$.
- Let f be an $\mathcal{ALC}$CTL formula.
- Let $T(S_d, R_d, kmap, f)$ denote the runtime of a call to docverify($S_d, s_d, R_d, kmap, f$).

Then $T(S_d, R_d, kmap, f) \in \mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(max(kmap))} + |f| \cdot (|S_d| + |R_d|) \cdot |\Delta_d|^2)$ with p being a polynomial $\mathbb{N} \rightarrow \mathbb{R}^+$ of a degree higher than 0.

Proof:

For the proof of Proposition 6.4.9, regard the subsequent annotated version of Algorithm 6.4.7. In the sequel, p is a polynomial $\mathbb{N} \rightarrow \mathbb{R}^+$ of a degree higher than 0. Further, T_i denotes the runtime of instruction i in the subsequent annotated version of Algorithm 6.4.7.

function docverify($S_d, s_d, R_d, kmap, f$) {	
$AC_{\|f}$:= f.getAtomicConcepts();	$T_1 \in \mathcal{O}(\|f\|)$: single traversal of the expression tree of f.
$AR_{\|f}$:= f.getAtomicRoles();	$T_2 \in \mathcal{O}(\|f\|)$: single traversal of the expression tree of f.
$\Delta_d := \emptyset$;	$T_3 \in \mathcal{O}(1)$.
$I_d := \emptyset$;	$T_4 \in \mathcal{O}(1)$.
for each $s \in S_d$ {	$T_5 \in \mathcal{O}(1)$.
$KB_s := kmap(s)$;	$T_6 \in \mathcal{O}(1)$: can be done by hashtable lookups.
if not KB_s.isConsistent() **then**	$T_7 \in \mathcal{O}(2^{p(\|KB_s\|)})$: deciding satisfiability of $\mathcal{AL}$ knowledge bases is exponential in the size of the knowledge base (section 3.1.4).
return errorInconsistentKB(s);	$T_8 \in \mathcal{O}(1)$.
$\Delta_d := \Delta_d \cup KB_s$.getIndividuals();	$T_9 \in \mathcal{O}(\|KB_s\|)$: getIndividuals() requires a traversal of the knowledge base KB_s which is in $\mathcal{O}(\|KB_s\|)$. Also, the union $\cup$ can be done in $\mathcal{O}(\|KB_s\|)$ by using hashtables.
for each $\mathcal{A} \in AC_{\|f}$ {	$T_{10} \in \mathcal{O}(1)$.
$\mathcal{A}^{I_d(s)} := KB_s$.getInstances($\mathcal{A}$);	$T_{11} \in \mathcal{O}(2^{p(\|KB_s\|)})$: instance retrieval is exponential in the size of the knowledge base for the description logic $\mathcal{AL}$ (section 3.1.4).
$I_d := I_d \cup \{(\mathcal{A}, s) \mapsto \mathcal{A}^{I_d(s)}\}$;	$T_{12} \in \mathcal{O}(1)$: by using a hashtable.
}	
for each $\mathcal{R} \in AR_{\|f}$ {	$T_{13} \in \mathcal{O}(1)$.
$\mathcal{R}^{I_d(s)} := KB_s$.getFillers($\mathcal{R}$);	$T_{14} \in \mathcal{O}(2^{p(\|KB_s\|)})$: role filler retrieval is at most in exponential time for $\mathcal{AL}$.
$I_d := I_d \cup \{(\mathcal{R}, s) \mapsto \mathcal{R}^{I_d(s)}\}$;	$T_{15} \in \mathcal{O}(1)$: by using a hashtable.
}	
}	
if $\Delta_d = \emptyset$ **then**	$T_{16} \in \mathcal{O}(1)$.
return errorEmptyDomain();	$T_{17} \in \mathcal{O}(1)$.
return $s_d \in$ verify($S_d, R_d, \Delta_d, I_d, f$);	$T_{18} \in \mathcal{O}(\|f\| \cdot (\|S_d\| + \|R_d\|) \cdot \|\Delta_d\|^2)$ (Proposition 6.3.41)
}	

Let $T(M_d) = T(S_d, R_d, kmap, f) - T_{18}$ be the runtime that **docverify** requires for constructing the temporal structure $M_d = (S_d, R_d, \Delta_d, I_d)$ of document d.

$T(M_d)$ is dominated by the runtime T_{11} of the instance retrieval KB_s.getInstances($\mathcal{A}$) for each $s \in S_d$ and $\mathcal{A}$ which is executed $|S_d| \cdot |AC_{|f}|$ times. Since $|AC_{|f}| \leq |f|$ and $|KB_s| \leq max(kmap)$, we get for the cumulative runtime T'_{11} of all instance retrieval queries KB_s.getInstances($\mathcal{A}$) executed in **docverify**:

$$T'_{11} \leq |S_d| \cdot |f| \cdot T_{11} \in \mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(max(kmap))})$$

Since all other statements within the **for**-loops of **docverify** are in $\mathcal{O}(2^{p(max(kmap))})$ and no statement is executed more than $|f| \cdot |S_d|$ times, we get for the cumulative runtime $T'_{5..15}$ of the "**for each** $s \in S_d$ { ... }" loop in **docverify**:

$$T'_{5..15} \in \mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(max(kmap))})$$

$T_1, T_2, T_3, T_4, T_{16}, T_{17} \in \mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(max(kmap))})$ because $|f| \geq 1$ and $|S_d| \geq 1$ and $p(max(kmap)) \geq 0$.

Since $T(M_d) = T_1 + T_2 + T_3 + T_4 + T'_{5..15} + T_{16} + T_{17}$, we get:

$$T(M_d) \in \mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(max(kmap))})$$

$T_{18} \in \mathcal{O}(|f| \cdot (|S_d| + |R_d|) \cdot |\Delta_d|^2)$ and thus

$$\begin{aligned} T(S_d, R_d, kmap, f) &= T(M_d) + T_{18} \\ &\in \mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(max(kmap))} + |f| \cdot (|S_d| + |R_d|) \cdot |\Delta_d|^2) \end{aligned}$$

□

Remark 6.4.10 (Runtime Complexity of docverify)

If we assume $|R_d| \cdot |\Delta|^2 \leq 2^{p(max(kmap))}$ the runtime of **docverify** is dominated by the cost $T(M_d)$ of constructing the $\mathcal{ALC}$CTL temporal structure M_d and we get

$$T(S_d, R_d, kmap, f) \in \mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(max(kmap))})$$

In such a scenario, the runtime of **docverify** is dominated by the complexity of DL reasoning which is EXPTIME-hard already for simple description logics such as $\mathcal{AL}$ (section 3.1.4).

An exponential blow-up of reasoning time can be avoided by limiting the size of each local knowledge base $KB_U \in \mathbf{KB}_d$. If the size of each content unit $U \in CU_d$ of some document d is limited we can assume that there is $k \in \mathbb{N}$ such that $|KB_U| \leq k$ for each $KB_U \in \mathbf{KB}_d$ and knowledge representation $\mathbf{KB}_d$ of some document d and thus $max(kmap) \leq k$.

Under these conditions, we get for the runtime $T(S_d, s_d, R_d, kmap, f)$ of **docverify**:

$$\begin{aligned} T(S_d, R_d, kmap, f) &\in \mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(k)} + |f| \cdot (|S_d| + |R_d|) \cdot |\Delta_d|^2) \\ &= \mathcal{O}(|f| \cdot |S_d| + |f| \cdot (|S_d| + |R_d|) \cdot |\Delta_d|^2) \\ &= \mathcal{O}(|f| \cdot (|S_d| + |R_d|) \cdot |\Delta_d|^2) \end{aligned}$$

Hence, when restricting the size of each local knowledge base, the runtime scaling of **docverify** is determined by the cost of $\mathcal{ALC}$CTL model checking.

□

In the sequel, we define a worst, best, and average case scenario for the verification of documents using $\mathcal{ALC}$CTL. Based on these scenarios, we will determine the runtime complexity of **docverify** w.r.t. the formula and document size.

A typical application scenario - the average case scenario - can be characterized as follows:

Let $|d|$ denote the size of the document d, for instance, measured in number of bytes or lines of text. If we fix the size of each content unit $U \in CU_d$ we can assume that the number of content units grows linearly in the size of the document d and hence

$$|S_d| = |CU_d| = \alpha|d|$$

for some $\alpha \in \mathbb{R}^+$. As for the number of narrative relations R_d, we can assume a fixed upper limit $n \in \mathbb{N}_1$ for the number of suitable successor units of each content unit $U \in CU_d$ such that n does not grow with the document size. Consequently, the narrative graph NG_d of a document d can be assumed to be sparsely connected, which results in

$$|R_d| = \alpha'|S_d| = \beta|d|$$

for some $\alpha' \in \mathbb{R}^+$ and $\beta = \alpha \cdot \alpha'$. The knowledge domain Δ_d representing the set of objects we have knowledge about in the context of document d can be assumed to grow linearly in the size of d:

$$|\Delta_d| = \gamma|d|$$

for some $\gamma \in \mathbb{R}^+$. In contrast, the size $|KB_U|$ of each local knowledge base $KB_U \in \mathbf{KB}_d$ does not depend on the size of the document d but on the chosen granularity for representing content units only. Hence,

$$|KB_U| \leq k$$

for each $KB_U \in \mathbf{KB}_d$ and some $k \in \mathbb{N}_1$ and thus also $max(kmap) \leq k$.

Under these assumptions, we get for the runtime complexity of **docverify**:

$$\begin{aligned} T(S_d, R_d, kmap, f) &\in \mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(k)} + |f| \cdot (|S_d| + |R_d|) \cdot |\Delta_d|^2) \\ &= \mathcal{O}(|f| \cdot \alpha|d| + |f| \cdot (\alpha|d| + \beta|d|) \cdot \gamma|d|^2) \\ &= \mathcal{O}(|f| \cdot |d|^3) \end{aligned}$$

As a result, in an average case scenario the verification of documents using $\mathcal{ALC}$CTL model checking can be expected to be linear in the size of the formula and cubic in the size of the document.

In the sequel, we formally define the properties of a worst, an average, and a best case, and prove the runtime of **docverify** in each case.

Definition 6.4.11 (Best / Average / Worst Case Scenario for docverify)

Let $M_d = (S_d, R_d, \Delta_d, I_d)$ be the $\mathcal{ALC}$CTL temporal structure of document d and $|d|$ denote the size of document d.

Then we assume for the best, average, and worst case scenario that the number of content units CU_d and thus the number of states S_d grows at most linearly in the size of the document and thus there is some $\alpha \in \mathbb{R}^+$ such that

$$|S_d| = |CU_d| \leq \alpha|d|$$

- In the *worst case* scenario we assume:

$$\begin{aligned} |R_d| = |S_d|^2 &\leq \alpha^2|d|^2 \\ max(kmap) = |cKB_d| &\leq \beta|d| \\ |\Delta_d| &\leq \gamma|d| \end{aligned}$$

  with $\beta, \gamma \in \mathbb{R}^+$.

  In the worst case, $R_d = S_d \times S_d$. Further, there is a local knowledge base $KB_U \in \mathbf{KB}_d$ containing the entire knowledge about the document d and thus $KB_U = cKB_d$ for some $U \in CU_d$ with cKB_d being the combined knowledge base of document d (Definition 5.3.10). This results in $max(kmap) = |cKB_d|$. The combined knowledge base cKB_d, which represents the entire knowledge about the document, is assumed to grow linearly in the size $|d|$ of the document. Also, the knowledge domain $\Delta_d = IV_{cKB_d}$, which represents the set of content objects of the document (Definition 5.3.15), can be assumed to grow linearly in the size of the document.

- In an *average case* scenario we assume:

$$\begin{aligned} |R_d| = \alpha'|S_d| &= \beta|d| \\ max(km_d) &\leq k \\ |\Delta_d| &\leq \gamma|d| \end{aligned}$$

  for some fixed $k \in \mathbb{N}_1$, $\alpha', \gamma \in \mathbb{R}^+$, and $\beta = \alpha \cdot \alpha'$ (see explanations ahead of Definition 6.4.11).

- As for the *best case* we assume:

$$\begin{aligned} |R_d| = |S_d| &= \alpha|d| \\ max(kmap) &\leq k \\ |\Delta_d| &\leq k' \end{aligned}$$

  for some fixed $k, k' \in \mathbb{N}_1$ and $\alpha \in \mathbb{R}^+$.

  In the best case, each state $s \in S_d$ has just one R_d successor. This is the minimal because R_d is left-total. In addition, we assume for the best case that the

knowledge domain $\Delta_d = IV_{cKB_d}$ does not grow in the size of the document, but remains below a constant upper bound for growing documents, i.e. larger documents do not cover more topics but cover each topic in greater detail.

□

Based on the definitions of worst, average, and best case scenarios we can determine the respective runtime complexity of **docverify** w.r.t. the document size $|d|$:

Proposition 6.4.12 (Best/Average/Worst Case Runtime Complexity of docverify)

For a document d and an $\mathcal{ALC}$CTL formula f, a call to docverify($S_d, s_d, R_d, kmap, f$) takes the following time in the worst, average, and best case, respectively.

- In the *worst case* (Definition 6.4.11), it holds:

  $$T^{wc}(S_d, R_d, kmap, f) \in \mathcal{O}(|f| \cdot 2^{p'(|d|)})$$

  with p' being a polynomial $\mathbb{N} \rightarrow \mathbb{R}^+$ of degree higher than 0.

- In the *average case* (Definition 6.4.11), it holds:

  $$T^{ac}(S_d, R_d, kmap, f) \quad \in \quad \mathcal{O}(|f| \cdot |d|^3)$$

- In the *best case* (Definition 6.4.11), it holds:

  $$T^{bc}(S_d, R_d, kmap, f) \quad \in \quad \mathcal{O}(|f| \cdot |d|)$$

Proof:

By Proposition 6.4.9, it holds: the runtime of **docverify** is in $\mathcal{O}(|f| \cdot |S_d| \cdot 2^{p(max(kmap))} + |f| \cdot (|S_d| + |R_d|) \cdot |\Delta_d|^2)$ with p being a polynomial $\mathbb{N} \rightarrow \mathbb{R}^+$ of degree higher than 0.

In the worst case (Definition 6.4.11) holds:

$$\begin{aligned} |S_d| &\leq \alpha|d| \\ |R_d| &\leq \alpha^2|d|^2 \\ max(kmap) &\leq \beta|d| \\ |\Delta_d| &\leq \gamma|d| \end{aligned}$$

with $\alpha, \beta, \gamma \in \mathbb{R}^+$.

Hence, we get for the worst case runtime $T^{wc}(S_d, R_d, kmap, f)$ of **docverify**:

$$\begin{aligned} T^{wc}(S_d, R_d, kmap, f) &\in \mathcal{O}(|f|\cdot \alpha|d|\cdot 2^{p(\lceil \beta|d| \rceil)} + |f|\cdot (\alpha|d| + \alpha^2|d|^2)\cdot \gamma^2|d|^2) \\ &= \mathcal{O}(|f|\cdot |d|\cdot 2^{p'(|d|)} + |f|\cdot |d|^4) \\ &\subseteq \mathcal{O}(|f|\cdot |d|^4\cdot 2^{p'(|d|)}) \\ &\subseteq \mathcal{O}(|f|\cdot 2^{|d|^4}\cdot 2^{p'(|d|)}) \\ &= \mathcal{O}(|f|\cdot 2^{p'(|d|)+|d|^4}) \\ &= \mathcal{O}(|f|\cdot 2^{p''(|d|)}) \end{aligned}$$

with $p' : |d| \mapsto p(\lceil \beta|d| \rceil)$ and $p'' : |d| \mapsto p'(|d|) + |d|^4$ being polynomials $\mathbb{N} \to \mathbb{R}^+$.

In the average case (Definition 6.4.11) holds:

$$\begin{aligned} |S_d| &\leq \alpha|d| \\ |R_d| &\leq \beta|d| \\ max(kmap) &\leq k \\ |\Delta_d| &\leq \gamma|d| \end{aligned}$$

with $\alpha, \beta, \gamma \in \mathbb{R}^+$ and $k \in \mathbb{N}$.

Using this in the runtime complexity $\mathcal{O}(|f|\cdot |S_d|\cdot 2^{p(max(kmap))} + |f|\cdot (|S_d| + |R_d|)\cdot |\Delta_d|^2)$ of **docverify**, we get for the average case:

$$\begin{aligned} T^{ac}(S_d, R_d, kmap, f) &\in \mathcal{O}(|f|\cdot \alpha|d|\cdot 2^{p(k)} + |f|\cdot (\alpha|d| + \beta|d|)\cdot \gamma^2|d|^2) \\ &= \mathcal{O}(|f|\cdot |d| + |f|\cdot |d|^3) \\ &= \mathcal{O}(|f|\cdot |d|^3) \end{aligned}$$

As for the best case (Definition 6.4.11), we have:

$$\begin{aligned} |S_d| &\leq \alpha|d| \\ |R_d| &\leq \alpha|d| \\ max(kmap) &\leq k \\ |\Delta_d| &\leq k' \end{aligned}$$

with $\alpha \in \mathbb{R}^+$ and $k, k' \in \mathbb{N}$.

Using the best case equations for the document size $|d|$, we get:

$$\begin{aligned} T^{bc}(S_d, R_d, kmap, f) &\in \mathcal{O}(|f|\cdot \alpha|d|\cdot 2^{p(k)} + |f|\cdot (\alpha|d| + \alpha|d|)\cdot k'^2) \\ &= \mathcal{O}(|f|\cdot |d| + |f|\cdot |d|) \\ &= \mathcal{O}(|f|\cdot |d|) \end{aligned}$$

□

Remark 6.4.13 (Best/Average/Worst Case Runtime Complexity of docverify)

The runtime scaling of **docverify** highly depends on the maximal size of the local knowledge bases of the document's knowledge representation $\mathbf{KB}_d$: if it can be kept constant, **docverify** is guaranteed to scale polynomially in the size of the document; if it grows linearly in the document size, the runtime of **docverify** may grow exponentially in the size of the document.

In the *average case*, a constant maximal size of each local knowledge base and a sparsely connected narrative graph are assumed. Further, a worst case assumption for the model checking cost of the document's temporal structure M_d w.r.t. formula f is applied. Under these assumptions, the scaling of **docverify** in the document size is cubic. If the narrative graph of a document is not sparsely connected, we get a runtime of **docverify** in $\mathcal{O}(|f|\cdot|d|^4)$. This is the worst case complexity for constant bounded local knowledge bases.

The best case results for **docverify** are not unrealistic. In fact, in many practical experiments a linear growth of runtime w.r.t. the document size is observed (see sections 7.5 and 8.3.4).

□

6.4.4. Possible Optimizations

The following optimizations can reduce the runtime of **docverify**.

Parallelization

Constructing the $\mathcal{ALC}$CTL temporal structure M_d of document d in **docverify** can be well parallelized.

This is because the calculation of the temporal interpretation $\cdot^{I_d(s)}$ of every atomic concept $\mathcal{A} \in AC_{|f}$ and atomic role $\mathcal{R} \in AR_{|f}$ at every state $s \in S_d$ can be done independently. As a result, the calculation of I_d can be done in parallel within one "step" if $|S_d|\cdot(|AC_{|f}| + |AR_{|f}|)$ processors are available. This is a significant speed up of the overall algorithm because calculating I_d is exponential and thus is the most expensive part of the algorithm.

Incremental Verification

Calculating the temporal structure M_d of document d can be done incrementally. I.e. M_d as calculated by **docverify** is stored in a persistent result cache and (partially) reused in subsequent calls of **docverify** for document d.

In subsequent checks of the same document d, usually the document has been modified just locally. Hence, the set $CU_{mod} \subseteq CU_d$ of content units, which have been modified as compared to the previous run of **docverify** on document d, is typically small.

Let $S_{mod} := CU_{mod}$ be the set of modified content units. Then the local interpretation $\cdot^{I_d(s)}$ of an atomic concept $\mathcal{A} \in AC_{|f}$ or atomic role $\mathcal{R} \in AR_{|f}$ at state s needs to be calculated for states $s \in S_{mod}$ only. For every other state, the previously calculated interpretation can be reused from the result cache. This can save a significant amount of runtime if just few content units have been modified.

Checking Sets of Formulae instead of Single Formulae

Enhancing **docverify** from checking a single formula $f \in \mathcal{ALC}$CTL to a set of formulae $F \subseteq \mathcal{ALC}$CTL may significantly increase the overall performance of **docverify** in application scenarios. This is, because a document d is rarely checked against a single criterion but d usually needs to satisfy several criteria. It is likely that many formulae of a larger specification F share some atomic concepts and roles.

When calling **docverify** for each $f \in F$, the temporal interpretation of shared atomic concepts and roles is calculated several times. Recalculation of previously calculated results can easily be avoided when constructing the temporal interpretation I_d for the atomic concepts and roles of the entire set F of $\mathcal{ALC}$CTL formulae. In addition, **verify** can work more efficiently when called for a set of $\mathcal{ALC}$CTL formula to check on the same temporal structure as in the case of a single formula at a time (section 6.3.3.3).

Accelerated Methods for Simple Cases

Calculating the temporal interpretation I_d in **docverify** involves a large number of instance/role-filler retrieval queries to the local DL knowledge bases $KB_s \in \mathbf{KB}_d$. In many application scenarios, there are a significant number of atomic concepts/roles the instances/role-fillers of which can be determined without complex ontological reasoning. We call these atomic concepts/roles *primitive*.

An atomic concept $\mathcal{A}$ / role $\mathcal{R}$ is *primitive* iff it does not appear in any of the terminological axioms of the reference ontology RO. If $\mathcal{A}$ is a primitive concept the set $\mathcal{A}^{I_d(s)} = \{a \in \Delta_s \mid KB_s \models \mathcal{A}(a)\}$ does not depend on any terminological axiom in KB_s because the only terminological axioms in KB_s are the terminological axioms of RO (Definition 5.3.10). In such a case, the set $\{a \in \Delta_s \mid KB_s \models \mathcal{A}(a)\}$ can be retrieved directly from the ABox of KB_s. The same holds for the fillers of primitive atomic roles.

State of the art model checkers such as Racer (section 3.1.5) check for such and other "simple cases" and apply accelerated reasoning strategies automatically. However, by building a lookup table of primitive concepts and roles and storing the instances of primitive atomic concepts/roles internally, the expensive communication cost with an external reasoning tool can be saved in addition.

6.5. Comparison with Other Temporal Logics

First temporal extensions of description logics have been proposed by Schmiedel in 1990 [Sch90]. Schmiedel's logic is based on time intervals as opposed to $\mathcal{ALC}$CTL which is evaluated on discrete points in time. The first point-based temporal description logics has been suggested by Schild in 1993 (see section 3.2.3). Since then, many approaches to temporally extending description logics have been suggested [AF00, AF01, HWZ01, WZ00]. We do not attempt to give a complete overview of approaches and results but refer the reader to the surveys in [AF01], [WZ00], [HWZ01], and [HWZ02].

Many of the proposed logics are undecidable and hence not well suited for automated reasoning [AF01]. Decidable fragments of point-based branching time temporal description logics have been discovered and studied recently [BHWZ04, HWZ02]. Most of them are fragments of the branching time logic $\mathcal{DPCTL}^*$ [HWZ01].

6.5.1. Comparison with $\mathcal{DPCTL}^*$

$\mathcal{ALC}$CTL is a syntactic fragment of $\mathcal{DPCTL}^*$. $\mathcal{DPCTL}^*$ extends the standard description logic $\mathcal{ALC}$ w.r.t. temporal specifications by introducing following operators:

- past tense operators S (since), $\diamond_P$ (some time in the past), $\Box_P$ (always in the past);
- future tense operator U (until), $\bigcirc$ (next time), $\diamond_F$ (some time in the future), $\Box_F$ (always in the future);
- path quantifiers A (all paths) and E (some path).

$\mathcal{DPCTL}^*$ is interpreted on bundled ω-trees, i.e. trees the full branches of which are order-isomorphic to $\langle \mathbb{N}_0, < \rangle$ [HWZ02]. A $\mathcal{DPCTL}^*$ model $\mathfrak{M} = \langle \mathfrak{F}, H, D, I \rangle$ is composed of

- an ω-tree $\mathfrak{F}$ representing states (or moments in time) and a "proceed" relation between them,
- a set of full branches H in $\mathfrak{F}$ representing linearly ordered flows of time in $\mathfrak{F}$,
- a domain of objects D,
- an interpretation function I assigning each moment in time s an $\mathcal{ALC}$-interpretation $I(s)$ of $\mathcal{DPCTL}^*$ atomic concepts, roles, and constants.

For the complete definition of the syntax and semantics of $\mathcal{DPCTL}^*$ we refer the reader to [HWZ01]. Since there are few restrictions on how path quantifiers, temporal, and non-temporal connectives can be combined, the expressiveness of $\mathcal{DPCTL}^*$

is high and $\mathcal{DPCTL}^*$ can be considered as an "upper bound" of point-based temporal description logics [HWZ01]. Unfortunately, the high expressiveness of $\mathcal{DPCTL}^*$ leads to undecidability of satisfiability and logical implications (cf. [HWZ02]).

In contrast to $\mathcal{DPCTL}^*$, $\mathcal{ALC}$CTL offers future tense quantifiers only and restricts the use of temporal connectives such that they always need to be paired with path quantifiers.

The semantics of $\mathcal{ALC}$CTL differs from $\mathcal{DPCTL}^*$ because $\mathcal{ALC}$CTL formulae are evaluated w.r.t. states of a state transition system (S, R) while $\mathcal{DPCTL}^*$ formulae are evaluated on ω-trees w.r.t. full branches of the tree [HWZ01]. The semantics of $\mathcal{ALC}$CTL formulae is closer to the semantic document model (Definition 5.3.24) representing the structure of the document as a graph of content units (states) and narrative relations (transitions).

The differences in the definition of the semantics of either logic render a formal comparison of their expressiveness difficult. We demonstrate the differences of $\mathcal{ALC}$CTL and $\mathcal{DPCTL}^*$ by giving application-related examples.

Example 6.5.1 (LTL Expressions)

The syntactic restriction of $\mathcal{ALC}$CTL, that each temporal connective needs to be paired with a path quantifier, limits the expressiveness of $\mathcal{ALC}$CTL such that, as opposed to $\mathcal{DPCTL}^*$, the linear fragment of temporal description logics is not contained in $\mathcal{ALC}$CTL.

For instance, linear temporal formulae of the type $\Diamond\Box p$ with p being a formula are not expressible in CTL [CD88]. The formula holds on a path iff eventually a state is reached from which on p holds in every state.

Since the LTL formula $\Diamond\Box p$ is not expressible in CTL and $\mathcal{ALC}$CTL coincides with CTL on the level of formulae, there is no formula equivalent to $\Diamond\Box p$ in $\mathcal{ALC}$CTL.

For instance, there is no equivalent expression in $\mathcal{ALC}$CTL for the $\mathcal{DPCTL}^*$ formula

$$f = \Diamond\Box(Difficult \sqsubseteq \bot) \tag{6.24}$$

"Eventually no more difficult content is presented." (cf. [HR04])

The following $\mathcal{ALC}$CTL approximation, for instance, is not semantically equivalent to f:

$$f' = \mathsf{AF}\ \mathsf{AG}(Difficult \sqsubseteq \bot) \tag{6.25}$$

The formula holds iff on every path eventually a state s is reached from which no state s' is reachable at which the interpretation of $Difficult$ is nonempty.

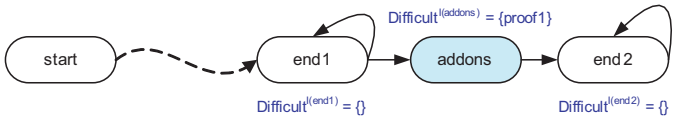

Figure 6.7.: temporal structure illustrating the difference between LTL and CTL

Figure 6.7 illustrates the difference between $\mathcal{DPCTL}^*$ formula f of Equation (6.24) and the $\mathcal{ALCC}$CTL formula f' of Equation (6.25). Figure 6.7 sketches a narrative structure of a document as follows. From the starting state $start$, on all paths eventually state $end1$ is reached, which represents the "first" end of the document. The user can either remain in state $end1$ forever (i.e. stop reading in $end1$) or proceed with state $addons$ containing additional material such as $proof1$. $proof1$ is difficult, which is represented by $proof1 \in Difficult^{I(addons)}$ (Figure 6.7). After state $addon$, the "second" end of the document $end2$ is reached.

The $\mathcal{DPCTL}^*$ formula f holds in the temporal structure depicted in Figure 6.7 for all infinite paths starting from state $start$. This is because each infinite path from $start$ eventually reaches state $end1$ and then either stays in state $end1$ forever or eventually reaches state $end2$ and then stays in state $end2$ forever. In both cases eventually a state endN is reached such that $(Difficult \sqsubseteq \bot)$ holds forever.

In contrast f' does not hold in the temporal structure of Figure 6.7 at state $start$. This is because there is a full path $fp = (start, ..., end1, end1, end1, ...)$ from state $start$ to state $end1$ which remains in $end1$ for ever. In fp, it holds at every state that state $addons$ is reachable at which $(Difficult \sqsubseteq \bot)$ does not hold. Hence, $M, start \models \mathsf{EG}\,\mathsf{EF}\neg(Difficult \sqsubseteq \bot)$ and, since $\mathsf{EG}\,\mathsf{EF}\neg(Difficult \sqsubseteq \bot) \equiv \neg\mathsf{AF}\,\mathsf{AG}(Difficult \sqsubseteq \bot) = \neg f'$, we get

$$M, s \not\models f'$$

with M denoting the temporal structure sketched in Figure 6.7.

For a detailed discussion on the expressiveness of CTL as compared to LTL we refer the reader to [CD88, HR04]. □

Remark 6.5.2 (LTL Expressions)

The limitations of CTL as compared to LTL have high practical relevance in the area of verifying processes because many fairness constraints expressible in LTL are not expressible in CTL and fairness is an important property of protocols for concurrent processes [BBF+01]. However, the concept of "fairness" is of little relevance in the presented use case of documents because the reader does not compete with other readers on getting access to some limited resource. Up to now, it is unclear if there is an

important class of criteria for documents, which are expressible in linear time logics but not in $\mathcal{ALC}$CTL. In contrast, relevant criteria being expressible in $\mathcal{ALC}$CTL but not in linear time logics are easy to find.

The simplest such kind of criteria are reachability requirements of the shape AG EF p, for instance, AG EF $\neg(Home \sqsubseteq \bot)$ "The reader can always get back to "home". Such a property is not expressible in linear time logics [HR04]. As a consequence, we opted for using a branching time logic for the representation of criteria on documents.

Note that linear and branching time can be combined: $\mathcal{DPCTL}^*$ includes the linear time fragment as well as the CTL fragment of description logics. However, the resulting CTL* type logics are computationally complex: model checking *propositional* CTL* is exponential in the size of the formula [Sch03]).

Even state-of-the-art model checking systems such as SMV and NuSMV do not support full CTL* (section 3.2.4.3). Instead, they enhance CTL model checking algorithms to the class of fairness constraints that can be expressed in LTL only [BBF$^+$01, CGP02c]. We can probably leverage these enhancements for $\mathcal{ALC}$CTL model checking because $\mathcal{ALC}$CTL model checking is reduced to CTL model checking problems and thus most of the model checking work can be delegated to an external CTL model checking system. When "fairness-style" properties turn out to be important for documents, an according enhancement of the verification framework may become an issue of future research. □

Example 6.5.3 (Past Tense Quantifiers)

The following $\mathcal{DPCTL}^*$ formula is not in $\mathcal{ALC}$CTL:

$$usedConcept \sqsubseteq \Diamond_P definedConcept \tag{6.26}$$

$\Diamond_P$ is a past tense temporal connective expressing "some time in the past". Equation (6.26) represents the property "every used concept has been defined some time in the past".

Formula (6.26) is not in $\mathcal{ALC}$CTL because 1) $\mathcal{ALC}$CTL does not offer past tense connectives such as $\Diamond_P$ and 2) in $\mathcal{ALC}$CTL temporal connectives such as $\Diamond_P$ cannot be used in isolation but need to be paired with path quantifiers.

However, a re-formulation of the criterion can be expressed in $\mathcal{ALC}$CTL:

$$\top \sqsubseteq \mathsf{A}(definedConcept\ \mathsf{B}\ usedConcept) \tag{6.27}$$

"everything must be, on all paths, a defined concept before it is a used concept".

In many cases, properties expressed by past tense quantifiers can also be represented by expressions using future tense quantifiers [LS00, LS95]. However, not in all cases a equivalent transformation exists and, even if one exists, it is not always straight forward [BBF$^+$01, LS95]. □

Remark 6.5.4 (Past Tense Quantifiers)

As for LTL and CTL*, past tense connectives can always be transformed into equivalent future-tense expressions when formulae are interpreted on finite state transition systems [LS95]. In the case of CTL, however, past tense connectives add additional expressiveness but also additional computational complexity [LS95]. As a consequence, most available model checking systems support future tense expressions only [BBF$^+$01].

Limiting the specification language $\mathcal{ALC}$CTL to future tense has been a pragmatic choice which ensures the feasibility and efficiency of the framework. □

Results for temporal description logics address the decidability of satisfiability and the computational complexity of logical implication [AF00, AF01, AFM$^+$01, AFM03, AFW$^+$01, AFWZ02, BHWZ02, BHWZ04, HWZ01, HWZ02, WZ00]. All of the suggested temporal description logics are either undecidable or computationally intractable. Consequently, existing applications of temporal description logic do not scale well to application relevant problem sizes (cf. section 6.5.3).

DL-based specification languages that address temporal aspects and are applied in practical systems such as T-REX [WL92, WL94], CLASP [DL91, DL96], and RAT [HKNP92], are special-purpose languages limited to a specific application domain, for instance, representation and recognition of plans. There is little known about the computational complexity of these languages [AF01].

The model checking problem of temporal description logics has not yet been studied in detail. Results on the decidability and computational complexity of model checking temporal description logics have not been available. This work presents the first model checking algorithm for a temporal description logic and proves its complexity, soundness, and correctness. It is shown that model checking $\mathcal{ALC}$CTL is in polynomial time and that TDL-based model checking enables to verify properties that are inefficient to verify using existing methods (cf. sections 6.5.2, 7.5, and 8.3).

6.5.2. Comparison with CTL

CTL is a widely adopted formalism for the verification of state transition systems (chapter 3). It is also applied in hypermedia verification (section 2.4.2). We argued that CTL is not sufficient for expressing coherence criteria on documents, which motivated the choice for a temporal description logic as a specification formalism. In the sequel, we will demonstrate the advantages of $\mathcal{ALC}$CTL as compared to CTL for the specification and verification of content-related criteria on documents and compare the computational complexity of $\mathcal{ALC}$CTL- and CTL-based document verification.

6.5.2.1. Expressiveness and Adequacy

Coherence criteria (Definition 2.3.1) are an important class of criteria that are expressible in $\mathcal{ALC}$CTL but not in CTL. An example of a coherence criterion has been given in the introduction of this chapter (section 6.1). With respect to the semantic document model introduced in chapter 5, we define coherence criteria as follows.

Definition 6.5.5 (Coherence Criteria in Narrative Structures)

Coherence criteria are the set of properties regarding the narrative structure of a document (Definition 5.2.19), which contain a combination of the following type of requirements:

- there is a (semantic) relation R of objects within a content unit U_1 to objects within another content unit $U_2 \neq U_1$.
- content U_1 has to be passed either before or after unit U_2 on some or all narrative paths through the document.

□

Example 6.5.6 (Coherence Criteria)

Typical examples of coherence criteria have been given in Example 2.3.2. For convenience, we repeat these sample criteria here and then give a possible formalization of them in $\mathcal{ALC}$CTL.

1. "Every theorem must be proven immediately."
2. "Eventually a solution to every task to solve is presented."
3. "The information required to solve a task has previously been presented."
4. "Defined concepts need to be used later on and used new concepts need to be defined before."
5. "For every robot function presented in the overview of the manual, a detailed handling instruction must be provided later on."
6. "Every objective mentioned in an introduction needs to be addressed by some related content later on but before its major aspects are summarized in a conclusion."
7. "The user cannot access protected content until she/he has read and accepted an according end user license agreement."

Possible $\mathcal{ALC}$CTL formalizations of the properties above are:

1. $\mathsf{AG}(Theorem \sqsubseteq \forall asserts.\mathsf{EX}\, \exists shownIn.Proof)$
2. $\mathsf{AG}(Exercise \sqsubseteq \forall hasTask.\mathsf{AF}\, \exists topicOf.Solution)$
3. $Information \sqsubseteq \mathsf{A}(\exists topicOf.Lesson\ \mathsf{B}\ \exists requiredFor.\exists taskOf.Exercise)$
4. $\mathsf{AG}(Definition \sqsubseteq \forall defines.\mathsf{EF}\, \exists usedIn.\top) \wedge$
   $newConcept \sqsubseteq \mathsf{A}(\neg\exists usedIn.\top\ \mathsf{U}\ definedIn.Definition)$
5. $\mathsf{AG}(Overview \sqsubseteq \forall presents.(\neg Function \sqcup \mathsf{EF}\, \exists describedIn.$
   $(HandlingInstruction \sqcap \exists hasDetailLevel.High)))$
6. $\mathsf{AG}(Introduction \sqsubseteq \forall hasObjective.\forall relatesTo.$
   $\mathsf{A}(\exists topicOf.\top\ \mathsf{B}\ \exists topicOf.(Summary \sqcap \exists partOf.Conclusion)) \sqcap$
   $(\neg MajorAspect \sqcup \mathsf{EF}\, \exists topicOf.(Summary \sqcap \exists partOf.Conclusion)))$
7. $ProtectedContent \sqsubseteq \mathsf{A}(\neg Accessible\ \mathsf{U}\ \forall protectedBy.$
   $(\neg License \sqcup (Presented \sqcap Accepted)))$

□

Remark 6.5.7 (Coherence Criteria)

Criterion 7) in Example 6.5.6 illustrates that formal verification of documents is particularly relevant if restricted access to content needs to be enforced. Only sound and complete algorithms can provide a proof of the presence or absence of some property. Note, however, that the value of the proof depends on the correctness of the model that has been extracted from the document and is checked against the specification. For determining the states in which a "license" is "presented" and "accepted", the semantic document model also needs to represent relevant aspects of the document *presentation system*. This is possible in principle but is beyond the current prototypical implementations of the knowledge extraction components.

Most of the $\mathcal{ALC}$CTL formalizations shown in Example 6.5.6 can be simplified using a reference ontology. For instance, given that the reference ontology RO contains the axiom

$$summarizedTopic \doteq \exists topicOf.(Summary \sqcap \exists partOf.Conclusion)$$

expression 6) in Example 6.5.6 can be simplified to

$$\begin{aligned}\mathsf{AG}(Introduction \ \sqsubseteq\ & \forall hasObjective.\forall relatesTo.\\ & \mathsf{A}(\exists topicOf.\top\ \mathsf{B}\ summarizedTopic)\\ & \sqcap(\neg MajorAspect \sqcup \mathsf{EF}\ summarizedTopic))\end{aligned}$$

Still, Example 6.5.6 demonstrates that the formalization of criteria is not trivial although their natural language formulation may appear quite simple. This is a general problem in the field of formal methods [DAC99, FMPR04, KC05]. Therefore, user support will be a major concern in future research [Jak06]. □

Corollary 6.5.8 (Coherence Criteria cannot be Expressed in CTL)

Coherence criteria on arbitrary domains are not expressible in CTL. This is because CTL provides no means for expressing relationships between objects (Definition 3.2.8).
□

Note, however, that coherence criteria can be approximated by CTL for a *given finite* interpretation domain as shown in the following example.

Example 6.5.9 (CTL Approximations of Coherence Criteria)

A possible CTL approximation of criterion 1) of Example 6.5.6 is:

$$\begin{aligned}&\mathsf{AG}(\\ &\quad (Theorem1 \rightarrow \mathsf{EX}\ ProofOfTheorem1) \wedge\\ &\quad (Theorem2 \rightarrow \mathsf{EX}\ ProofOfTheorem2) \wedge\\ &\quad ... \wedge\\ &\quad (TheoremN \rightarrow \mathsf{EX}\ ProofOfTheoremN))\end{aligned}$$

where $Theorem1, ..., TheoremN$ are atomic propositions representing the set of theorems of the document and $ProofOfTheorem1, ..., ProofOfTheoremN$ are atomic propositions representing the set of respective proves.

Note that the CTL formula given above is just an approximation of the $\mathcal{ALC}$CTL formula $\mathsf{AG}(Theorem \sqsubseteq \forall asserts.\mathsf{EX}\ \exists shownIn.Proof$. The $\mathcal{ALC}$CTL formalization accounts for theorems and proofs, which contain/show more than one assertion, while the CTL formalization abstracts from single assertions of a theorem or proof.

In *finite* domains, relations can always be encoded in terms of atomic propositions (Proposition 6.3.3). From a practical perspective, however, a propositional encoding of relations becomes infeasible already for moderately sized documents because the resulting formulae can grow exponentially (Remark 6.3.10). From an engineering perspective, the tight coupling of the CTL specification to the checked documents is problematic because a change of the document is likely to require an update of the specification, which is a source of overhead and errors.
□

6.5.2.2. Computational Complexity

Recall that model checking CTL is in $\mathcal{O}(|f| \cdot (|S| + |R|))$ (Theorem 3.2.12). If CTL is applied to document structures we can assume that the set of states S grows linearly in the document size $|d|$ and the set of transitions R grows linearly in the document size in the average case and quadraticly in the size of $|d|$ in the worst case (cf. Definition 6.4.11).

The complexity results for CTL and $\mathcal{ALC}$CTL based verification of formula f on document d summarize as (cf. Proposition 6.4.12) :

	CTL	$\mathcal{ALC}$CTL
best case	$\mathcal{O}(\|f\| \cdot \|d\|)$	$\mathcal{O}(\|f\| \cdot \|d\|)$
average case	$\mathcal{O}(\|f\| \cdot \|d\|)$	$\mathcal{O}(\|f\| \cdot \|d\|^3)$
worst case	$\mathcal{O}(\|f\| \cdot \|d\|^2)$	$\mathcal{O}(\|f\| \cdot 2^{p(\|d\|)})$

$|f|$ is the size of the checked CTL or $\mathcal{ALC}$CTL formula, $|d|$ is the size of the verified document, and p is a polynomial $\mathbb{N} \rightarrow \mathbb{R}^+$ of degree higher than 0. The runtime results and underlying best/worst/average case assumptions are presented in section 6.4.3.

In the best case scenario, $\mathcal{ALC}$CTL- and CTL-based document verification have the same approximate complexity. Under optimal conditions, $\mathcal{ALC}$CTL model checking is expected to achieve the same performance as CTL model checking. In the average case, however, $\mathcal{ALC}$CTL-based document verification scales worse than CTL. Clearly, the additional expressiveness of $\mathcal{ALC}$CTL results in a higher computational complexity.

The exponential worst case runtime of $\mathcal{ALC}$CTL-based document verification stems from the adoption of DL reasoning for the construction of temporal document structures. Since DL reasoning is not applied by standard CTL model checking methods, the complexity of CTL-based document verification remains polynomial in all cases.

For a fair comparison of the runtime of $\mathcal{ALC}$CTL and CTL, we have to consider that in the case of CTL the size or number of formulae $|f|$ often grows with the document size (cf. Example 6.5.9) while the size and number of $\mathcal{ALC}$CTL formulae usually remain constant for growing documents. In practical application, $\mathcal{ALC}$CTL model checking can be more efficient than CTL model checking (see experimental results in section 7.5).

6.5.3. Existing Applications of Temporal Description Logics

As opposed to propositional temporal logics such as CTL, all existing temporal extensions of description logics lack a mature infrastructure for automated reasoning based on theorem proving or model checking techniques [LSWZ01]. This severely limits their applicability in practice. As a result, existing applications of temporal description logics are at a rather conceptual level.

[AF98] describes an approach to representing actions and plans by the temporal description logic TL-$\mathcal{ALCF}$ that is a restricted version of the interval-based temporal description logic of Schmiedel [Sch90]. In contrast to Schmiedel's logic, the subsumption relationship between concepts is decidable in TL-$\mathcal{ALCF}$. Subsumption reasoning is used for plan recognition and plan retrieval based on an action taxonomy.

In [AFM+01], [AFW+01], and [AFWZ02], the temporal logic $\mathcal{DLR}_{US}$ is suggested for representing conceptual schemata and queries to temporal databases. The decidability of $\mathcal{DLR}_{US}$ enables various inference services for temporal databases, for instance, proving query containment or checking the satisfiability of queries and schemata w.r.t. all possible states of the data model. It is shown, however, that deciding containment of non-recursive Datalog queries in $\mathcal{DLR}_{US}$ is in 2EXPTIME, and schema satisfiability and implication is EXPSPACE-complete.

A similar approach is described in [AFM03]. The temporal description logic $\mathcal{ALCQI}_{US}$ is suggested for conceptual modelling of dynamic information. Two different scenarios are distinguished: (1) modelling of dynamic (time-varying) aspects of data and (2) representing the evolution of schemata in time. It is shown how temporal ER diagrams can be mapped onto $\mathcal{ALCQI}_{US}$. This mapping is used for automated checking of the satisfiability and logical implication of temporal schemata.

All applications above are based on theorem-proving, which is already EXPTIME-hard (section 3.1.4) for relevant *non-temporal* description logics. This and the lacking availability of reasoning systems for temporal description logics [LSWZ01] restrict the described applications to theoretical concepts.

In contrast to existing approaches, we apply temporal description logics for *model checking* finite document structures. To the best of our knowledge, this work presents the first application of temporal description logics that bases on model checking. It is shown how reasoning in non-temporal DL and TDL model checking can be combined to achieve higher expressiveness, flexibility, and performance in the domain of document verification than existing approaches (see section 2.4 and chapters 7 and 8). The proposed verification algorithms are sound and complete, and have a moderate polynomial runtime in the average case. Evaluation results on real and realistic document bases confirm the high efficiency, adequacy, and expressive power of the approach (chapters 7 and 8).

6.6. $\mathcal{ALC}$CTL-based Document Verification – Summary

We have defined $\mathcal{ALC}$CTL, a branching time temporal description logic that is expressive for document properties. We have shown that the $\mathcal{ALC}$CTL model checking problem can be solved in polynomial time for finite structures and provided a sound, complete, and optimal model checking algorithm. Further, it has been shown how document verification can be transformed into an $\mathcal{ALC}$CTL model checking problem by applying semantic modelling based on DL knowledge bases and DL reasoning. A sound and complete document verification algorithm has been defined, which runs in polynomial time in the average case despite of the exponential complexity of involved DL reasoning. The polynomial runtime is achieved by partitioning the information about the document into local DL knowledge bases of constant size.

We have demonstrated that relevant criteria on documents can be expressed by $\mathcal{ALC}$CTL and that $\mathcal{ALC}$CTL exceeds the expressiveness of commonly applied propositional temporal logics regarding the important class of coherence criteria.

As compared to existing approaches to document verification, the combination of DL reasoning and TDL model checking offers higher expressive power regarding both content- and path-related criteria, a higher flexibility regarding the type of documents and criteria being checked, and a better decoupling of the specification from the document format and background knowledge, which is adopted to derive implicit information about of the content of a document.

Part III.

Evaluation and Discussion

7. Implementation and Evaluation

7.1. Introduction

Overview of Case Studies and Benchmarks

We have implemented the core components of the $\mathcal{ALC}$CTL verification framework described in chapters 5 and 6. The implementation has been evaluated on XML-based eLearning documents as well as on synthetic benchmarks.

The aim of the implementation is to demonstrate the feasibility, adequacy, and efficiency of the approach for real problems and realistic problem sizes.

The implementations are prototypical in the following aspects:

- not all components are optimized in terms of efficiency and memory usage. For a productive system more can be done to increase the efficiency of the current implementation.
- the knowledge extraction components are restricted to the test cases shown in the course of the chapter. In a productive system the knowledge extraction component can easily be enhanced to extract more information about the document's content.
- the user interfaces are rudimentary. Hence, no conclusions about the "usability" of the approach at the end-user level can be drawn at current.

However, the prototypes are sufficiently complete to demonstrate the expressive power and performance of the presented approach.

Goals and Requirements

The prototypical application and case studies are designed to find answers to the following questions:

- Is $\mathcal{ALC}$CTL adequate for representing relevant semantic criteria on documents?
- Do real document bases contain sufficiently detailed and structured information (metadata) which is required to verify relevant criteria? Can it be extracted at reasonable cost?
- Is the system effective for finding (previously unknown) errors in documents?

- Do the analytically derived runtime results transfer to realistic application scenarios? Which is the maximal problem size being processable on an average office computer in less than 30 seconds?

7.2. Some Comments on the Implementation

The $\mathcal{ALC}$CTL prototype system has been implemented in Java 1.5. As a subcomponent for the generation of verification models, the DL reasoning systems Racer [HM01] and Pellet [SPG+07] have been integrated using the DIG [BMC03] interface standard. $\mathcal{ALC}$CTL formulae are reduced by a framework component to CTL formulae that are verified by a re-implementation of the explicit state model checking algorithms described in [HR04].

Knowledge extraction components for different XML-based formats for eLearning content such as SCORM [Adv04a], LMML [SF02], and $<$ML$>^3$ [LTV03] have been implemented.

The experimental results reported in the sequel have been obtained on a web-based training (WBT) about industrial robots and a number of synthetic benchmarks that closely resemble a manual of an eLearning system. The "Robot WBT" is implemented in SCORM [Adv04a] and proprietary XML formats. In total, the WBT consists of 1102 files, 90 of them being relevant for the checking of consistency criteria. The WBT delivers content tailored to three different target groups: trainees, trainers, and support. The synthetic benchmarks simulate SCORM documents of different sizes and structure. The evaluation results on synthetic and real SCORM documents are presented in section 7.3.

In addition, the $\mathcal{ALC}$CTL system has been evaluated on a selection of LMML- and $<$ML$>^3$ online learning documents developed within the national joint project WWR (knowledge factory of computer engineering) [LT02]. These documents are in use at undergraduate and graduate courses in computer engineering at the University of Passau and other institutions. For results of the additional case studies, we refer the reader to [Sch06] and [Lü06].

Figure 7.1 gives an overview of the components and major data structures of the $\mathcal{ALC}$CTL verification system. For a detailed description of the system's architecture and implementation is given in [Sch06] and [Lü06].

The core components of the system as depicted in Figure 7.1 are:

- *knowledge extractor* (Figure 7.1 rhs bottom): this component parses the XML sources of the document implemented in SCORM [Adv04a], for instance. It extracts the relevant metadata about the document and transforms it into the structures of the semantic model: the narrative structure (Definition 5.2.19) and the DL knowledge representation (Definition 5.3.10).

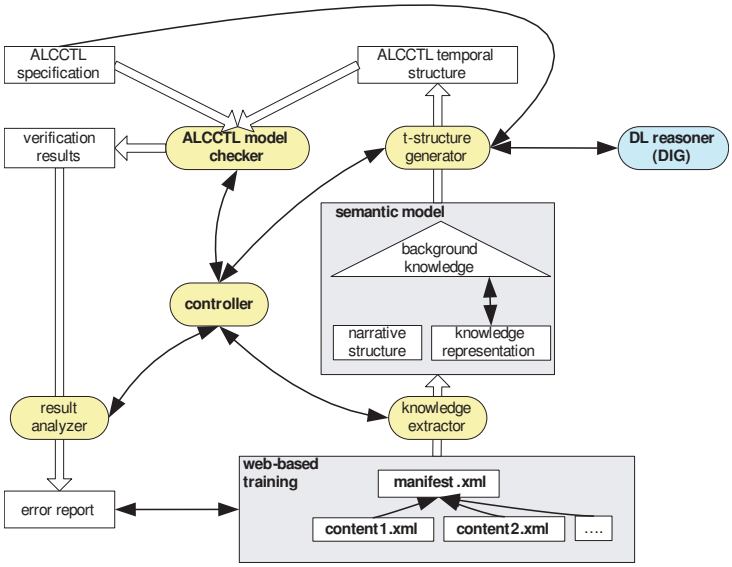

Figure 7.1.: components of the $\mathcal{ALC}$CTL verification system

- *temporal structure generator* (Figure 7.1 rhs top): this component computes the $\mathcal{ALC}$CTL temporal structure from the semantic model of the document w.r.t. a set of $\mathcal{ALC}$CTL formulae (Definition 6.4.1). The component sends the necessary instance retrieval queries to an external DIG-compatible reasoner (cf. Algorithm 6.4.7). The most appropriate reasoners for the given application scenario have been determined in a preparatory study. The evaluated candidates were KAON2 [HMS04b], Pellet [SPG+07], Fact++ [TH06], Racer [HM01], and RacerPro [Rac07]. Among them Racer, RacerPro, and Pellet delivered the best performance and the highest stability. The description logics supported by Racer(Pro) and Pellet are sufficiently expressive for the requirements of the evaluation scenario. As a consequence, Racer, RacerPro, and Pellet have been chosen as reference systems in later experiments. The DL systems are used in the standard configuration with all optimization options switched off to keep the achieved runtime results realistic, generally valid, and easily reproducible.

- $\mathcal{ALC}$CTL *model checker* (Figure 7.1 lhs top): this component implements $\mathcal{ALC}$CTL model checking along the lines of Algorithm 6.3.37. Recall that Algorithm 6.3.37 reduces the $\mathcal{ALC}$CTL model checking problem to a CTL model checking problem that can be solved using existing methods and tools. For saving communication overhead with an external model checking tool such as NuSMV [CCG+02], we opted for internally checking CTL formulae by re-implementing the comparably simple CTL model checking algorithms described in [HR04]. These algorithms are based on an explicit representation of the state

space. State of the art model checkers such as NuSMV usually adopt a symbolic representation of the state transition system (cf. section 3.2.4.2), which enables the verification of very large temporal structures. In the case of documents, an explicit representation of the state space is possible because the number of states is comparably small. Recall that the number of states directly corresponds to the number of content units of a document, which closely corresponds with the number of pages of the document in the given case.

Besides saving communication overhead with an external model checking tool, the internal implementation of CTL model checking provides access to intermediate verification results and thus enables a more detailed error report if a formula fails to hold.

- *result analyzer* (Figure 7.1 lhs bottom): the result analyzer extracts relevant information from the detailed verification results as returned by the $\mathcal{ALC}$CTL model checker and displays them to the user. For instance, details about violated subsumption ($\sqsubseteq$) and equals ($\doteq$) connectives within unsatisfied formulae are displayed. These are particularly helpful for tracking the root cause of an error.

- *controller* (Figure 7.1 lhs center): this component controls the overall verification process and manages the dataflow between the other components of the framework.

The following optimizations as described in section 6.3.3.3 and 6.4.4 have been implemented:

- *checking sets of formulae*: the temporal structure of the document is checked against a set of $\mathcal{ALC}$CTL formulae in one go instead of a single formula at a time. Shared expressions within and across different formulae of the specification are detected and re-calculation of common sub-expressions is avoided (section 6.3.3.3).

- *avoiding inefficient* CTL *Mappings*: non-temporal connectives are checked by dedicated methods and not mapped onto an equivalent CTL problem (section 6.3.3.3).

- *accelerated methods for simple cases*: in the process of generating the $\mathcal{ALC}$CTL temporal document structure, not all instance-retrieval queries are actually sent to the DL reasoner. Cases that do not involve TBox reasoning are handled internally. This significantly speeds up the generation of the $\mathcal{ALC}$CTL temporal document structure when the reference ontology is small or even empty (section 6.4.4).

7.3. Evaluation Environment

The runtime results have been obtained in the following runtime environment:

- desktop personal computer with 32 Bit single core Intel Pentium IV processor at 2.4 Ghz, 1 GB DDR Ram, and 80 GB ATA hard disk;
- Microsoft Windows XP SP2;
- Java runtime environment 1.6.0_02-b06, using 256 MByte of heap space;
- DL reasoning systems Racer version 1.7.23, RacerPro version 1.9.0, and Pellet version 1.5.0;
- CTL model checking system NuSMV version 2.4.3 for the comparison the $\mathcal{ALC}$CTL runtime results with the performance of a state-of-the-art CTL model checker.

The runtime results obtained within this environment are almost identical to the runtimes on an average portable computer (Pentium M at 1.7 Mhz, 1 Gbyte DDR Ram, 40 GB hard disk).

The prototype runs without modification on Windows Vista. Its performance on Vista is about 5% lower than on Windows XP.

Finally, some experiments were conducted on a relatively up-to-date desktop computer with Intel Core 2 Duo Processor E6400 at 2.13 Ghz with 2 GB DDR2-667 memory and 250 GB SATA hard disk. This machine required about one third of the time required by the standard environment in all experiments. As a result, on recent hardware the runtime values can be expected to be at a fraction of the values presented in the subsequent sections.

7.4. Case Study

The $\mathcal{ALC}$CTL framework has been evaluated on a web-based training about robots based on SCORM, which is a commonly adopted standard for web-based eLearning documents [Adv04a]. In the sequel, we briefly summarize the most important characteristics of the case. Further details about the content, structure, and knowledge representation of the web-based training can be found in [Rad05a].

7.4.1. Description of the Case

7.4.1.1. Content and Structure of the Document Base

The verified web-based training teaches the basics for classifying and using different kinds of industrial robots to the trainees and customers of a major robot company. Figure 7.2 sketches the general structure of the document. It consists of 82 interactive web-pages structured into 6 larger lessons. Each lesson starts with an introduction presenting its objectives and major concepts, followed by several exposition pages presenting new concepts to learn (Figure 7.2). Exposition pages are frequently interleaved with pages containing interactive exercises for applying or repeating the currently studied concepts. In the end of each lesson there is a test to assess the progress of the learner.

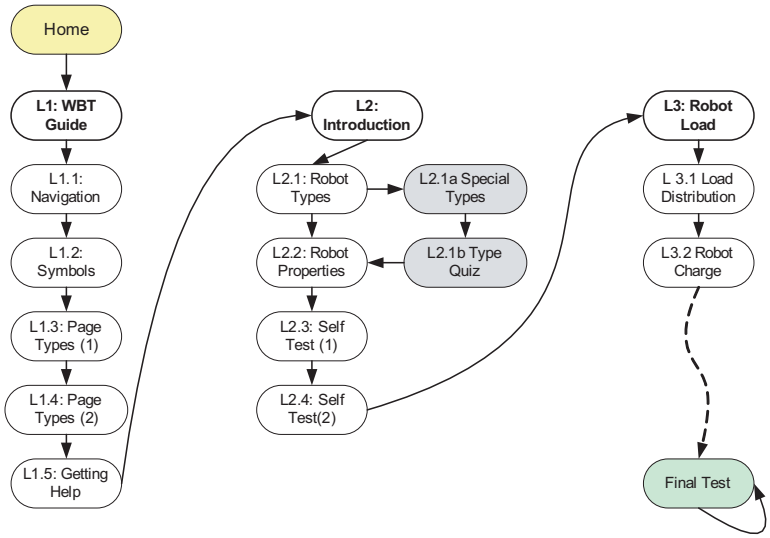

Figure 7.2.: narrative structure of the robot WBT

The narrative structure of the WBT is basically linear, i.e. it is meant to be learned lesson by lesson. However, there are few optional side tracks (e.g. L2.1a and L2.1b in Figure 7.2) that present additional information about the current topic to the interested reader.

The WBT is published in different variants tailored to each of the three major target groups: trainees, trainers, and support staff. The variants of the document vary in size and complexity of content. For comparing the performance for different problem sizes, a separate semantic model has been constructed for each variant and verified against a specific set of criteria.

Table 7.1 characterizes the size of each variant and its semantic model.

	trainees	trainers	support
# lessons	6	2	3
# web pages	82	32	48
# content units / states	79	29	45
# narrative relations / transitions	81	29	47
# DL assertions in combined knowledge base	339	124	195
# assertions per content unit	4.3	4.3	4.3
# criteria / formulae ”long spec”	25	15	9
# criteria / formulae ”short spec”	11	11	9
# violated criteria / formulae	5	5	3

Table 7.1.: size of different variants of the robot WBT

7.4.1.2. Document Format

The WBT is implemented using the SCORM standard version 1.2 [Adv04a]. The structure of the WBT is determined by the file `imsmanifest.xml`, which contains different *organizations* and a list of *resources* according to the IMS Content Packing Standard [IMS03]. An *organization* is basically a table of contents of the document annotated by metadata describing the content of each item.

Listing 7.1 shows a part of the organization of `imsmanifest.xml`. Each `item` element represents a page (or ”screen”) of the WBT. Its attribute `identifierref` refers to an external resource (an XML-file, for instance) that contains the content of the item. An item can contain subitems. For instance, `item45` and `item46` are subitems of `item44` in Listing 7.1.

The narrative structure and parts of the local fact bases of the document are extracted from `imsmanifest.xml`. For instance, the attribute `pagetype` indicates the type of content being presented by a specific resource. `item46` being attributed as `pagetype="testquestion"` (line 14 in Listing 7.1) indicates that the respective web-page presents a test question to the learner. Other ”page types” are ”testresult” and ”additional material”. If the attribute `pagetype` is left empty (line 3 and 7 in Listing 7.1), the default page type ”information” is implied.

```
0 <items>
    ...
    <item identifier="item44" identifierref="resource3_02"
      isvisible="1" pagetype="" parameters="">
      <title>Robot Charge</title>
5
      <item identifier="item45" identifierref="resource3_02-1"
        isvisible="1" pagetype="" parameters="">
        <title>Introduction to Robot Charge</title>
      </item>
```

```
10
    ...

    <item identifier="item46" identifierref="resource3_02-17"
      isvisible="0" pagetype="testquestion" parameters="">
15      <title>Distance to Center of Gravity</title>
    </item>
  </item>
</items>
```

Listing 7.1: organization of items within `imsmanifest.xml`

The major part of the resources - in the presented case - consists of XML files in a proprietary XML-format which embeds standards such as Learning Object Metadata (LOM) [Lea02]. Listing 7.2 shows a part of a test page of the WBT. A major part of the local fact bases are extracted from the metadata within resource files as sketched in Listing 7.2. For instance, the entry

```
<relation>
    <resource>lesson03_02-1l.xml</resource>
</relation>
```

represents the fact that the content of the current page is related to the content of resource `lesson03_02-1l.xml`.

```
0 <document>
   <metadata>
     <general>
       <title>Distance to Center of Gravity</title>
       <identifier>resource3_02-17</identifier>
5    </general>
     <educational>
       <learningresourcetype>test</learningresourcetype>
       <learningresourcesubtype>PP</learningresourcesubtype>
       <difficulty>high</difficulty>
10   </educational>
     <relation>
       <resource>lesson03_02-1l.xml</resource>
     </relation>
   </metadata>
15

(... Content of the Document ...)

</document>
```

Listing 7.2: the resource file for item `item46` of the WBT

7.4.1.3. Description of Knowledge Representation

Items of the manifest file (Listing 7.1) represent single web-pages containing content about a distinct topic of the document. Hence, items are good candidates for content units within the semantic model of the document. Narrative relations can be derived from the organization and metadata of items within the manifest file. The document order of the manifest file (Listing 7.1) defines a standard successor for each item / content unit. Some items are attributed as pages of type "additional material". These items are bypassed on the standard path through the document which leads to a narrative structure as depicted in Figure 7.2. For further details, see [Rad05a].

The numbers in table 7.1 indicate that the narrative structure of the document is rather linear, i.e. each content unit has exactly one successor unit most of the time. The performance of the system for "highly branching" documents is evaluated in a dedicated experiment (section 7.5.3). In each of the three variants the local fact base of each content unit contains 4 to 5 assertions on average.

The local knowledge base for the content unit `item46` sketched in Listings 7.1 and 7.2 is

$Assertion$	extracted from	meaning
$Testquestion(item46)$	Lst. 7.1, line 14, attribute `pagetype`	$item46$ is a test question
$hasType(item46, test)$	Lst. 7.2, line 7	$item46$ has content type $test$
$hasSubType(item46, PP)$	Lst. 7.2, line 8	the subtype of $item46$ is "PP"
hasDifficulty$(item46, high)$	Lst. 7.2, line 9	$item46$ is difficult
$relatedToFile(item46,$ *lesson03_02-1l*$)$	Lst. 7.2, line 11–13	$item46$ is related to *lesson03_02-1l*

These assertions have been extracted from the XML markup of the manifest file (Listing 7.1) and the resource file of the content unit (Listing 7.2).

The following atomtic concepts are used in subsequent sample specifications:

$MajorLesson$	represents the set of major lessons of the WBT.
$Home$	represents the beginning of the WBT (cf. Figure 7.2).
$EndOfDocument$	represents a distinct set of final content units of the document (cf. Definition 5.2.9).
$Excursion$	content units on side tracks that contain additional information for the interested user.
$Testbegin$	the introduction to a series of test questions.
$Testquestion$	a content unit that contains of one or more test questions.
$Testresult$	a unit that presents the result of a test series.
$Information$	content type of units that present concepts to learn.
$Test$	content type of units that test previously studied concepts.
$Low, Medium, High$	concepts that represent low, medium, and high difficulty levels of content units.

(7.1)

The following atomic roles are used in specifications:

$hasType$	associates content units with their content type $Information$ or $Test$.
hasDifficulty	associates content units with their difficulty level Low, $Medium$, or $High$
$isInFile$	associates content units with their source (an XML file).
$contains$	inverse of $isInFile$: $contains \doteq isInFile^{-}$.
$relatedToFile$	associates content units with files that contain related content.
$referredBy$	inverse of $relatedToFile$: $referredBy \doteq relatedToFile^{-}$.

(7.2)

7.4.1.4. Checked Criteria

Depending on the variant of the document (Table 7.1), a selection of 9 to 25 relevant criteria have been formalized in $\mathcal{ALC}$CTL and verified. The criteria have been chosen in such a way that most of $\mathcal{ALC}$CTL connectives are covered and, at the same time, the criteria can be approximated well in CTL. This allows for comparing the performance of $\mathcal{ALC}$CTL-based model checking with existing state-of-the-art CTL model checkers such as NuSMV. For the same reason, we refrained from using a reference ontology.

The benefits and cost of ontological reasoning for the verification of content-related criteria are examined in a separate experiment in section 7.5.4.

Introductory Remarks on the Formalization of Checked Criteria

Many of the checked criteria include a statement of the type

"... there should be some content of type T ...".

Such criteria require to express the existence of some object satisfying some specific predicate. In first order logic, a proper formalization has the form

$$\exists x\, T(x) \tag{7.3}$$

where T is a unary predicate that evaluates to $true$ for some object x iff x is of type T.

Recall that $\mathcal{ALCC}$CTL formulae are built on top of the basic expression $C \sqsubseteq D$ with C and D being concepts (Definition 6.2.1). This basic expression is equivalent to the first order formula $\forall x(C(x) \rightarrow D(x))$. Hence, $\mathcal{ALCC}$CTL formulae are implicitly universally quantified. As a consequence, statements about the *existence* of some specific object as represented by formula (7.3) cannot be expressed in $\mathcal{ALCC}$CTL directly.

However, since $\mathcal{ALCC}$CTL includes negation, this is not a general limitation as demonstrated in the sequel. Formula (7.3) is equivalent to

$$\neg\forall x\, \neg T(x)$$

which in turn is equivalent to

$$\neg\forall x(\neg T(x) \vee \bot(x)) \tag{7.4}$$

where $\bot$ is a unary predicate that evaluates to $false$ for all elements of the interpretation domain. Further, formula (7.4) is equivalent to

$$\neg\forall x(T(x) \rightarrow \bot(x))$$

which directly translates to the $\mathcal{ALCC}$CTL formula

$$\neg(T \sqsubseteq \bot) \tag{7.5}$$

with T being a concept that represents the set of objects of type T. In summary, formula (7.5) is equivalent to formula (7.3) and expresses the existence of some object of type T.

Formulae of type (7.5) are frequently used in the $\mathcal{ALCC}$CTL-based specifications of criteria within the following case studies.

List of Criteria and Formalizations

The complete list of criteria, which apply to each of the three variants of the document, is:

1. "At least one content unit is reachable, which contains some additional material."

   $$\begin{array}{ll} \mathcal{ALC}\mathsf{CTL}: & f_1 := \mathsf{EF}\ \neg(Excursion \sqsubseteq \bot) \\ \mathsf{CTL}: & p_1 := \mathsf{EF}\ excursion \end{array}$$

   $Excursion$ is a concept representing the set of content fragments at the current state / content unit, which contain additional information (cf. Equation (7.1)). $\neg(Excursion \sqsubseteq \bot)$ expresses the existence of some object of type $Excursion$ within the current content unit (cf. Equation (7.5)).

   $excursion$ is an atomic proposition that is $true$ at a given state iff the corresponding content unit contains some additional information.

2. "There is at least one standard path without any additional material through the document."

   $$\begin{array}{ll} \mathcal{ALC}\mathsf{CTL}: & f_2 := \mathsf{EG}(Excursion \sqsubseteq \bot) \\ \mathsf{CTL}: & p_2 := \mathsf{EG}\neg excursion \end{array}$$

3. "There is no path through the document without a test." I.e. the user has to take a test somewhere within the document:

   $$\begin{array}{ll} \mathcal{ALC}\mathsf{CTL}: & f_3 := \neg\mathsf{EG}(\exists hasType.Test \sqsubseteq \bot) \\ \mathsf{CTL}: & p_3 := \neg\mathsf{EG}\ \neg hasTypeTest \end{array}$$

   The atomic proposition $hasTypeTest$ is chosen in a way that it is $true$ at a state s iff $(\exists hasType.Test)^{I_d(s)} \neq \emptyset$, i.e. iff content unit s contains some test. Thus, p_3 is an equivalent abstraction of f_3.

   f_3 requires that there is no cycle within the narrative graph of the document before a test is reached. Otherwise, there were an infinite path never reaching a test. If cycles are allowed but it still needs to be ensured that the user encounters a test at some point of reading the document, the criterion can be changed to

   $$f_3' := \mathsf{A}(\neg(\exists hasType.Test \sqsubseteq \bot)\ \mathsf{B}\ \neg(EndOfDocument \sqsubseteq \bot))$$

   Here, the requirement of encountering a test is restricted to those paths that eventually reach an *end of the document* (represented by concept $EndOfDocument$). Paths that remain in a cycle before reaching an end unit are not required to contain a test.

4. "After information has been provided, a test must always be taken."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_4 := & \mathsf{AG}\,((\exists hasType.Information \sqsubseteq \bot) \vee \\ & & \mathsf{AF}\neg(\exists hasType.Test \sqsubseteq \bot)) \\ \mathsf{CTL}: & p_4 := & \mathsf{AG}\,(hasTypeInformation \rightarrow \mathsf{AF}\; hasTypeTest) \end{array}$$

The formalizations f_4 and p_4 just require that *some* test is presented after an information unit. It is sensible to postulate, in addition, that the topics of each information unit are actually covered by some test in the sequel. This can be achieved by modifying f_4 to

$$\begin{array}{lll} f_4' := & \mathsf{AG}(\exists contains.\exists hasType.Information \sqsubseteq \\ & \quad \mathsf{AF}\; \exists referredBy.Testquestion) \end{array}$$

with $contains$ being the inverse of role $isInFile$ and $referredBy$ being the inverse of role $relatedToFile$ (cf. Equation (7.2)). Since f_4' represents a coherence criterion (Definition 6.5.5) it is not expressible in CTL (Corollary 6.5.8).

5. "Immediately after a series of test questions, the test result is presented". This criterion can be reformulated to "immediately after each test question either another test question or the test result is presented". This can be expressed as

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_5 := & \mathsf{AG}((Testquestion \sqsubseteq \bot) \vee \\ & & (\mathsf{AX}\,\neg(Testquestion \sqcup Testresult \sqsubseteq \bot) \\ & & \wedge\neg\mathsf{EG}\,\neg(Testquestion \sqsubseteq \bot))) \\ \mathsf{CTL}: & p_5 := & \mathsf{AG}(testquestion \rightarrow \mathsf{AX}(testquestion \vee testresult) \\ & & \wedge\neg\mathsf{EG}\; testquestion) \end{array}$$

The term $\neg\mathsf{EG}\,\neg(Testquestion \sqsubseteq \bot)$ is added to exclude cycles within a series of test questions such that the test result is never reached on some narrative path.

6. "After the beginning of a test, a series of test questions is provided until the test result is presented".

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_6 := & \mathsf{AG}((Testbegin \sqsubseteq \bot) \vee ((Testresult \sqsubseteq \bot) \wedge \\ & & \mathsf{AX}((Testresult \sqsubseteq \bot) \wedge \\ & & \mathsf{A}(\neg(Testquestion \sqsubseteq \bot)\; \mathsf{U}\; \neg(Testresult \sqsubseteq \bot))))) \\ \mathsf{CTL}: & p_6 := & \mathsf{AG}(testbegin \rightarrow \neg testresult \wedge \\ & & \mathsf{AX}(\neg testresult \wedge \mathsf{A}(testquestion\; \mathsf{U}\; testresult))) \end{array}$$

The term $(Testresult \sqsubseteq \bot)$ is added twice to exclude that the test result is presented already at the beginning or the first content unit after the beginning of the test, and to ensure that at least one test question must be answered between the beginning of the test and the presentation of the test result.

7. "Each test is nothing else but a test begin, test question, or test result."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_7 := & \mathsf{AG}\,(\exists hasType.Test \doteq \\ & & Testbegin \sqcup Testquestion \sqcup Testresult) \\ \mathsf{CTL}: & p_7 := & \mathsf{AG}\,((hasTypeTest \rightarrow testbegin \vee testquestion \vee \\ & & testresult) \wedge (testbegin \vee testquestion \vee \\ & & testresult \rightarrow hasTypeTest)) \end{array}$$

   This checks the consistency of metadata about tests within the manifest and content files of the document.

8. "There is a path through the document without any difficult content".

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_8 := & \mathsf{EG}\,\neg(\exists \textit{hasDifficulty}.(Low \sqcup Medium) \sqsubseteq \bot) \\ \mathsf{CTL}: & p_8 := & \mathsf{EG}(\textit{hasDifficultyLow} \vee \textit{hasDifficultyMedium}) \end{array}$$

9. "On any path through the document the user will hit something difficult".

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_9 := & \mathsf{AF}\neg(\exists \textit{hasDifficulty}.High \sqsubseteq \bot) \\ \mathsf{CTL}: & p_9 := & \mathsf{AF}\ \textit{hasDifficultyHigh} \end{array}$$

   f_9 is a control property. It should be violated iff f_8 is satisfied because each content unit should have a unique difficulty.

f_1 through f_9 are general structural requirements applying to each of the three variants of the document. The remaining criteria refer to document topics and hence differ according to the topics covered by each of the variants.

10. "The content of each major lesson is tested on all paths through the document." In absence of a reference ontology defining the set of "major lessons" of each variant of the document, the "major lesson"-property needs to be encoded into the specification. This can be done by providing a separate formula for each major lesson as follows:

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{[10]} := & \mathsf{AF}\ \neg(\exists hasType.Test \sqcap \\ & & \exists relatedToFile.[MajorLesson] \sqsubseteq \bot) \\ \mathsf{CTL}: & p_{[10]} := & \mathsf{AF}\ hasTypeTest \wedge [relatedToMajorLesson] \end{array}$$

   $f_{[10]}$ and $p_{[10]}$ are not formulae but templates for a set of n_v formulae with n_v being the number of major lessons of variant $v \in \{trainees, trainers, support\}$ (cf. Table 7.1). $[MajorLesson]$ is instantiated by concepts representing a single major lesson such as *Lesson03_02-1l*. Analogously, $[relatedToMajorLesson]$ is instantiated by an atomic proposition for each major lesson, which is true in a state, iff the state represents a content unit that is related to the respective major lesson. For instance,

$\mathsf{AF}\ \neg(\exists$*hasType.Test* $\sqcap\ \exists$*relatedToFile.Lesson03_02-1l* $\sqsubseteq \bot)$

and

AF *hasTypeTest* $\wedge$ *relatedToLesson03_02-1l*

are instances of formulae $f_{[10]}$ and $p_{[10]}$, respectively, for major lesson *Lesson03_02-1l*.

11. "Whenever a test of a major lesson is reached, the tested lesson has been visited before."

    This is a coherence criterion since it expresses a combination of a semantic relationship between two objects (the test and the tested lesson) and a temporal relationship (test after the lesson) (Definition 6.5.5). In the case of CTL, coherence criteria require the explicit encoding of the semantic relationship into the specification (cf. Example 6.5.9). Since a reference ontology is missing, the set of "major lessons" also needs to be encoded into $\mathcal{ALC}$CTL formalization (cf. formula $f_{[10]}$). This results in sets of formulae of the following shape:

$$
\begin{array}{lll}
\mathcal{ALC}\mathsf{CTL}: & f_{[11]} := & \mathsf{A}(\neg(\exists hasType.Information \sqcap \\
 & & \exists isInFile.[MajorLesson] \sqsubseteq \bot) \\
 & & \mathsf{B}\ \neg(\exists hasType.Test \sqcap \\
 & & \exists relatedToFile.[MajorLesson] \sqsubseteq \bot)) \\
\mathsf{CTL}: & p_{[11]} := & \neg\mathsf{E}(\neg(hasTypeInformation \wedge [isMajorLesson]) \\
 & & \mathsf{U}\ (hasTypeTest \wedge [relatedToMajorLesson]))
\end{array}
$$

    Recall, $\mathsf{A}(p\ \mathsf{B}\ q)$ is defined as $\neg\mathsf{E}(\neg p\ \mathsf{U}\ q)$ (Definition 6.2.3). Since the CTL model checker NuSMV does not support the "before" operator B, we have chosen the "until"-based formalization in the case of CTL.

    Similar to $f_{[10]}$ and $p_{[10]}$, $f_{[11]}$ and $p_{[11]}$ are formula templates. $[MajorLesson]$ is replaced by concepts representing a single major lesson. $[isMajorLesson]$ and $[relatedToMajorLesson]$ are replaced by atomic propositions that are true at a state if the according content unit is the given major lesson or is related to the given major lesson, respectively.

    For instance,

    $\mathsf{A}(\neg(\exists$*hasType.Information* $\sqcap\ \exists$*isInFile.Lesson03_02-1l* $\sqsubseteq \bot)$
    $\mathsf{B}\ \neg(\exists$*hasType.Test* $\sqcap\ \exists$*relatedToFile.Lesson03_02-1l* $\sqsubseteq \bot))$

    and

    $\neg\mathsf{E}(\neg(hasTypeInformation \wedge$ *isLesson03_02-1l*$)$
    $\mathsf{U}\ (hasTypeTest \wedge$ *relatedToLesson03_02-1l*$))$

    are instances of formulae $f_{[11]}$ and $p_{[11]}$, respectively, for major lesson *Lesson03_02-1l*.

There are 8 major lessons in the trainees variant, 3 in the trainers variant, and none the variant for the support staff. Hence, for the trainees variant, 8 instances of each of the formula templates $f_{[10]}$, $p_{[10]}$, $f_{[11]}$, and $p_{[11]}$ are required while the trainers variant requires just 3 instances of each of these formulae templates. The criteria specified by the formula templates $f_{[10]}$, $p_{[10]}$, $f_{[11]}$, and $p_{[11]}$ are irrelevant for the variant of the support staff and hence instances of them are not generated in this case.

Together with the variant-independent formulae f_1 through f_9, we get 9+8+8=25 formulae for the trainees' variant, 9+3+3=15 formulae for the trainers' variant, and 9+0+0=9 formulae for the variant of the support staff (cf. Table 7.1 "long spec").

Suppose, in the reference ontology or global fact base of the document the set of major lessons of each variant is specified. This can be done explicitly by a set of ABox assertions as follows

$$MajorLesson(\textit{Lesson03\_02-1l})$$
$$MajorLesson(\textit{Lesson03\_03-1l})$$
$$MajorLesson(\textit{Lesson04\_02-1l})$$
$$\ldots$$

Alternatively, an intensional definition can be given by a terminological axiom such as

$$\exists^{\geq 4} contains.Units \sqcap \exists referredBy.Home \sqsubseteq MajorLesson$$

"every object, which contains at least 4 units and is referred by the home unit, is a major lesson"

Then the criteria 10) and 11) can be represented in $\mathcal{ALC}$CTL more compactly as follows:

$$\begin{aligned} f'_{10} &:= MajorLesson \sqsubseteq \mathsf{AF}\, \exists referredBy.\exists hasType.Test \\ f'_{11} &:= MajorLesson \sqsubseteq \mathsf{A}(\exists contains.\exists hasType.Information \\ &\qquad\qquad \mathsf{B}\, \exists referredBy.\exists hasType.Test) \end{aligned}$$

Recall, $referredBy$ is the inverse of role $relatedToFile$ and $contains$ the inverse of role $isInFile$ (Equation (7.2)).

These alternative formalizations are independent of the content of each document variant and thus do not need to be instantiated for each major lesson of a variant. This diminishes the number of formulae from 25 / 15 / 9 (Table 7.1 "long spec.") to 11 / 11 / 9 for the variants trainees / trainers / support (Table 7.1 "short spec."). In the subsequent experiment, the runtime results of both formalizations are determined.

Remark 7.4.1 (Formalization of Criteria)

Many criteria are intentionally chosen in a way that the CTL formalization is more compact than the respective $\mathcal{ALC}$CTL formalization (see, for instance formulae f_3

and p_3). This is to compare the performance of the $\mathcal{ALC}$CTL model checking system with CTL model checking for criteria that still can be represented well in CTL.

Criteria 4), 10), and 11) illustrate the superior expressiveness of $\mathcal{ALC}$CTL, which allows for more precise specifications (f'_4) and a more compact representations of complex criteria (f'_{10} and f'_{11}). Criteria 10) and 11) illustrate, in addition, the benefit of background knowledge for the specification and verification of criteria. Reference ontologies can help to decouple specifications from variations of the document's content and to reduce the complexity of specifications. Since the $\mathcal{ALC}$CTL formalizations f'_{10} and f'_{11} are independent of the content of a specific document variant, f'_{10} and f'_{11} can remain unchanged, if the content of the document is modified. In constrast, formula templates $p_{[10]}$ and $p_{[11]}$ need to be instantiated after each major update of the document for each of the variants of the document.

Note further that the $\mathcal{ALC}$CTL formalizations are intentionally not optimized to the shortest possible expression or most efficient to check representation. The formalizations have been chosen in such a way that they cover a wide range of the expressiveness of $\mathcal{ALC}$CTL. Moreover, we assume that also in practice not always the most efficient formalization of a consistency criterion is chosen. For getting most realistic runtime results, we refrained from manually tuning $\mathcal{ALC}$CTL formalizations. □

7.4.2. Qualitative Results of the Case Study

Table 7.2 shows the result of checking each variant of the web-based training.

	trainees	trainers	support
# web pages	82	32	48
# instances of $f_{[10]}$	8	3	0
# instances of $f_{[11]}$	8	3	0
# formulae	25	15	9
# violated formulae	5	5	3
violated formulae	$f_4, f_6, f_7, f_8, +$ one instance of $f_{[10]}$	f_1, f_4, f_6, f_7, f_8	f_3, f_4, f_8

Table 7.2.: summary of verification results

The verification results revealed the following previously unknown errors within the document:

- $f_{[10]}$ failed for the trainees variant and the lesson about "Robot Types". This means that for "Robot Types" there is no test reachable on any path within the document. A closer examination of this error revealed that actually a test unit exists for that topic but it is not referred to by the organization of the manifest

(Listing 7.1) and hence cannot be reached by the user. This type of error indicates that in some cases it is not sufficient to check for the "availability" of certain content within a web document since content can be available but still not *reachable* on some path through the content.

- f_7 (consistency of test metadata) failed for two variants of the document. This was because some test result has been marked up incorrectly as information unit, which leads to presentation errors.
- The same error caused f_4 (test after information) to fail in the trainees and trainers variant. f_4 was also violated in the support variant but for a different reason. In this variant there was actually no test available.
- f_6 (test begin $\rightarrow$ question $\rightarrow$ result) was violated in the trainees and trainers variant because some test question has not been marked as such within the manifest file of the document. Again, this may cause a misleading presentation of content.
- f_8 (some easy path) was violated in each of the three variants because all of them covered difficult subjects on every path through the document.
- f_1 (some additional material) was violated by the trainers variant because this variant simply does not contain any additional material.

The violation of $f_{[10]}$, f_7, and f_6 indicate severe errors. They came as a surprise because the web-based training has been released and has already been in use within a professional environment.

7.4.3. Quantitative Results of the Case Study

	trainees	trainers	support
# content units	79	29	45
# assertions in cKB_d	339	124	195
# formulae "long spec."	25	15	9
# formulae "short spec."	11	11	9
# violated formulae (long / short)	5 / 5	5 / 5	3 / 3
total runtime ($\mathcal{ALC}$CTL long spec)	0.99 s	0.62 s	0.67 s
DL reasoning	–	–	–
$\mathcal{ALC}$CTL model checker (long spec)	0.18 s	0.08 s	0.07 s
$\mathcal{ALC}$CTL model checker (short spec)	0.13 s	0.09 s	0.07 s
framework	0.81 s	0.54 s	0.60 s
CTL model checker NuSMV (long spec)	0.93 s	0.08 s	0.10 s

Table 7.3.: runtime results of the WBT case study

Table 7.3 lists the time required for checking each of the variants.

The total runtime is the sum of the following components:

- the time required for constructing the $\mathcal{ALC}$CTL temporal document structure, which generally involves instance retrieval queries to the external DL reasoning system. As for the given case, however, no reference ontology is used. Hence, DL reasoning is not required for the verification of the specifications.
- the time required to do the $\mathcal{ALC}$CTL model checking and to analyze the model checking results of violated formulae. Table 7.3 contains the runtime results of model checking the "long spec", which can be translated directly to CTL, and the "short spec", which reduces the number of formulae for the trainees and trainers variant by making use of the full expressiveness of $\mathcal{ALC}$CTL (section 7.4.1.4). The runtime values for $\mathcal{ALC}$CTL model checking include the time for extraction and output of a detailed verification report, i.e. they are the sum of the runtime of the components $\mathcal{ALC}$CTL *model checker* and *result analyzer* in Figure 7.1.
- the time required by the remaining components of the system for extracting the required metadata from the XML sources, and building the narrative structure and DL knowledge representation (penultimate row of Table 7.3).

For comparison, the temporal structure of the document has been converted into an input file of the CTL model checker NuSMV and checked against the CTL formalization of the criteria 1) to 11) (section 7.4.1.4). This is possible because the given scenario is chosen such that pure CTL model checking can be applied (Remark 7.4.1). The runtimes of NuSMV are included in the last row of Table 7.3. These values include the construction and output of a counterexample, but do not include the time for constructing the input file for NuSMV from the semantic model of the document. Hence, they are directly comparable with the values for $\mathcal{ALC}$CTL model checking listed in Table 7.3.

The total runtime of $\mathcal{ALC}$CTL-based document verification ranges between 0.62 and 0.99 seconds. More than 80% of the total time is taken by the framework. The $\mathcal{ALC}$CTL model checker requires just 0.07 to 0.18 seconds. In the case of the largest variant (trainees), the model checking time for the short specification is slightly less than for the long specification (Table 7.3). This indicates that the high expressiveness of $\mathcal{ALC}$CTL can help to increase the efficiency of verifying larger documents.

Surprisingly, the performance of $\mathcal{ALC}$CTL model checking can compete with NuSMV within the given application scenario, although the chosen CTL formalizations of criteria tends to be simpler than the corresponding $\mathcal{ALC}$CTL formulae (Remark 7.4.1). While in the smaller variants the runtime difference between CTL and $\mathcal{ALC}$CTL model checking is small, $\mathcal{ALC}$CTL model checking clearly outperforms NuSMV for the largest variant.

7.4.4. Case Study – Summary

In total, $\mathcal{ALC}$CTL has proven both expressive in representing criteria and efficient in verifying them. The collection of 25 formulae for the trainees' variant can be reduced to 11 formulae when making use of the full expressiveness of $\mathcal{ALC}$CTL. The smaller specification can be verified in less time and is less bound to the specific contents of a document variant. Compared to CTL model checking, a performance gain of up to a factor of 7 is achieved. The total runtimes remain below one second, which allows for smooth interactive use.

The documents of the case study range between 32 and 82 web pages and thus are rather small publications. In subsequent benchmarks, we examine if the encouraging runtime results also transfer to more complex scenarios.

7.5. Benchmarks

7.5.1. General Description and Research Questions

The subsequent experiments are targeted at the following research questions:

- How does the runtime of $\mathcal{ALC}$CTL prototype scale with the document size?
- How large is the runtime difference between satisfied and violated formulae?
- How does the system perform on documents offering many alternative reading trails?
- How does the system perform on reference ontologies of different complexity?
- Finally, how can the system cope with an increasing number of formulae?

To approach these questions, a series of synthetic benchmarks have been designed.

The benchmark bases on a section of 32 web-pages / content units, the structure and content of which are close to the web-based training of the case study.

Figure 7.3 depicts the narrative structure of the base section for building benchmark documents of different sizes. The document fragment contains 32 content units, 25 of them being "information" pages and 4 of them containing additional material in two "excursions". In the end of the section, 3 "test units" are presented.

Documents of different sizes have been constructed by appending the base section several times at the end of the document and linking the final state (32) of each section with the first state (1) of the subsequent section. A selection of 10 representative formulae have been chosen as a test specification.

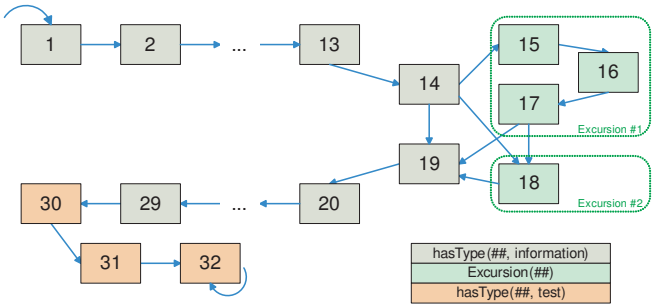

Figure 7.3.: basic building block of benchmark documents (cf. [Rad05a])

7.5.2. Experiment 1 - Scaling in Document Size

Description of the Experiment

The first benchmark aims at determining how the runtime correlates with the document size. For creating documents of different sizes the basic building block of Figure 7.3 has been copied 1 to 16 times which simulates 16 documents (D1), (D2), ..., (D16) consisting of 1 to 16 chapters of 32 pages each (Table 7.4).

The size of the narrative structure ranges from $1 \cdot 32$ to $16 \cdot 32 = 512$ content units and from $1 \cdot 35$ to $16 \cdot 35 = 560$ narrative relations (Table 7.4). The maximal length of an acyclic part of a narrative path increases from 32 for the smallest document (D1) to 512 for the largest document (D16). The number of narrative paths within the document ranges between 4^1 and $4^{16} = 4.294.967.296$.

The runtime of $\mathcal{ALCC}$CTL-based document verification further depends on size of the interpretation domain Δ_d of the $\mathcal{ALCC}$CTL temporal document structure (Definition 6.4.1). The interpretation domain Δ_d is the set of objects that represent the content of document d. The size $|\Delta_d|$ of the interpretation domain grows from $46 + 1 \cdot 32 = 78$ for the smallest document (D1) to $46 + 16 \cdot 32 = 558$ for the largest document (D16) (Table 7.4).

A good indicator of the amount of knowledge about the document is the sum of the sizes of the local knowledge bases

$$sizeOf(\mathbf{KB}_d) := \sum_{KB \in \mathbf{KB}_d} |KB|$$

with $\mathbf{KB}_d = \{KB_U \mid U \in CU_d\}$ being the knowledge representation of document d (Definition 5.3.10). The local knowledge bases consist of ABox assertions only, because a reference ontology is not applied within the scope of this Experiment. The impact of reference ontologies of different complexity on the performance of the system is examined in Experiment 3 (section 7.5.4).

Table 7.4 summarizes the relevant numbers of each of the test cases (D1) through (D16).

	D1	D2	D3	...	D16
# content units	32	64	96	...	512
# narrative relations	35	70	105	...	560
$sizeOf(\mathbf{KB}_d)$	332	664	996	...	5312
$\|\Delta_d\|$	78	110	142	...	558
# formulae	10	10	10	...	10
# violated formulae	3	3	3	...	3

Table 7.4.: benchmark test cases (standard documents)

To determine the upper limit of the size of documents, which can be handled by the $\mathcal{ALC}$CTL system, a second test series of very large documents (VLD1 - 10) is constructed in the same manner as the first series (D1 - D16). The relevant numbers for the second test series are listed in Table 7.5

	VLD1	VLD2	VLD3	...	VLD10
# content units	512	1024	1536	...	5120
# narrative relations	560	1120	1680	...	5600
$sizeOf(\mathbf{KB}_d)$	5312	10624	15936	...	53120
$\|\Delta_d\|$	558	1070	1582	...	5166
# formulae	10	10	10	...	10
# violated formulae	3	3	3	...	3

Table 7.5.: benchmark test cases (very large documents)

Each document is checked against 10 criteria of different complexity, three of them being violated. As in the case study in section 7.4, the criteria have been chosen in such a way that they can be represented equally well in $\mathcal{ALC}$CTL and in CTL. This enables a direct and fair comparison of the performance of $\mathcal{ALC}$CTL model checking with existing state-of-the-art CTL model checkers.

1. "Some additional material is reachable."

   $\mathcal{ALC}$CTL : $f_{1.1} :=$ EF $\neg(Excursion \sqsubseteq \bot)$
   CTL : $p_{1.1} :=$ EF $excursion$

   This criterion is satisfied.

2. "There is a path without any additional material."

   $\mathcal{ALC}$CTL : $f_{1.2} :=$ EG$(Excursion \sqsubseteq \bot)$
   CTL : $p_{1.2} :=$ EG$\neg excursion$

This criterion is satisfied.

3. "On all paths eventually additional material is presented."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{1.3} := & \mathsf{AF}\ \neg(Excursion \sqsubseteq \bot) \\ \mathsf{CTL}: & p_{1.3} := & \mathsf{AF}\ excursion \end{array}$$

This criterion is violated.

4. "There is a path without any tests."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{1.4} := & \mathsf{EG}(\exists hasType.Test \sqsubseteq \bot) \\ \mathsf{CTL}: & p_{1.4} := & \mathsf{EG}\neg hasTypeTest \end{array}$$

This criterion is violated.

5. "There is no path without any information units."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{1.5} := & \neg\mathsf{EG}(\exists hasType.Information \sqsubseteq \bot) \\ \mathsf{CTL}: & p_{1.5} := & \neg\mathsf{EG}\neg hasTypeInformation \end{array}$$

This criterion is met.

6. "Whenever some information is presented, eventually a test is reached."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{1.6} := & \mathsf{AG}((\exists hasType.Information \sqsubseteq \bot) \vee \\ & & \mathsf{AF}\ \neg(\exists hasType.Test \sqsubseteq \bot)) \\ \mathsf{CTL}: & p_{1.6} := & \mathsf{AG}(hasTypeInformation \rightarrow \mathsf{AF}\ hasTypeTest) \end{array}$$

This criterion is met.

7. "Whenever a test is reached, information has been presented before."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{1.7} := & \mathsf{A}(\neg(\exists hasType.Information \sqsubseteq \bot) \\ & & \mathsf{B}\ \neg(\exists hasType.Test \sqsubseteq \bot)) \\ \mathsf{CTL}: & p_{1.7} := & \neg\mathsf{E}(\neg hasTypeInformation\ \mathsf{U}\ hasTypeTest) \end{array}$$

This criterion is met.

8. "At the end of a series of test questions, the test result is presented."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{1.8} := & \mathsf{AG}((Testquestion \sqsubseteq \bot) \vee \\ & & (\mathsf{AX}\ \neg(Testquestion \sqcup Testresult \sqsubseteq \bot) \\ & & \wedge\neg\mathsf{EG}\ \neg(Testquestion \sqsubseteq \bot))) \\ \mathsf{CTL}: & p_{1.8} := & \mathsf{AG}(testquestion \rightarrow \mathsf{AX}(testquestion \vee testresult) \\ & & \wedge\neg\mathsf{EG}\ testquestion) \end{array}$$

This criterion is satisfied.

9. "After the beginning of a test, a series of test questions is provided until the test results are presented".

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{1.9} := & \mathsf{AG}((Testbegin \sqsubseteq \bot) \vee ((Testresult \sqsubseteq \bot) \wedge \\ & & \mathsf{AX}((Testresult \sqsubseteq \bot) \wedge \mathsf{A}(\neg(Testquestion \sqsubseteq \bot) \\ & & \mathsf{U}\ \neg(Testresult \sqsubseteq \bot))))) \\ \mathsf{CTL}: & p_{1.9} := & \mathsf{AG}(testbegin \rightarrow \neg testresult \wedge \\ & & \mathsf{AX}(\neg testresult \wedge \mathsf{A}(testquestion\ \mathsf{U}\ testresult))) \end{array}$$

This criterion is satisfied.

10. "Each test is nothing else but a test begin, test question, or test result."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{1.10} := & \mathsf{AG}\ (\exists hasType.Test \doteq Testbegin \sqcup \\ & & Testquestion \sqcup Testresult) \\ \mathsf{CTL}: & p_{1.10} := & \mathsf{AG}\ ((hasTypeTest \rightarrow testbegin \vee testquestion \vee \\ & & testresult) \wedge (testbegin \vee testquestion \vee \\ & & testresult \rightarrow hasTypeTest)) \end{array}$$

This criterion is violated.

Since f_1 through f_{10} are chosen such that they can be translated directly to CTL, they do not make use of the full expressiveness of $\mathcal{ALC}$CTL. For instance, path quantifiers and temporal connectives appear at the level of formulae only but not on the level of concepts. For getting more typical runtime results for the $\mathcal{ALC}$CTL case, we build a second specification by replacing formulae $f_{1.1}$ through $f_{1.5}$ with formulae $f'_{1.1}$ through $f'_{1.5}$ (cf. Table 7.6).

Remark 7.5.1 ($f'_{1.1}$ through $f'_{1.5}$ are not Equivalent to $f_{1.1}$ through $f_{1.5}$)

Although the formulae $f'_{1.1}$ through $f'_{1.5}$ are syntactically similar to their counterparts $f_{1.1}$ through $f_{1.5}$, they are not indented to be equivalent to their original version. Rather, $f'_{1.1}$ through $f'_{1.5}$ are targeted at determining the performance of the system for formulae making use of temporal concepts which are not available in CTL. Equivalent re-formalizations of formulae $f_{1.1}$ through $f_{1.5}$ would still remain within the class of properties, which can be represented well in CTL, and hence would not cause a significantly different scaling of runtime.

Still, $f'_{1.1}$ through $f'_{1.5}$ are chosen as close as possible to their original version: they have similar sizes and yield, in the given case, verification results that are identical to those using $f_{1.1}$ through $f_{1.5}$. This allows for assessing the extra-cost of using the full expressiveness of $\mathcal{ALC}$CTL as compared to CTL-like specifications. □

Formulae $f'_{1.1}$ through $f'_{1.5}$ are derived from $f_{1.1}$ through $f_{1.5}$ by "pushing down" the temporal connectives to the concept level of the formula. This way, harder-to-verify

original "CTL-like"criterion	**new "$\mathcal{ALC}$CTL-like" criterion**
"Some additional material is reachable."	"There is some object which is on some path eventually an excursion."
$f_{1.1} := \mathsf{EF}\ \neg(Excursion \sqsubseteq \bot)$	$f'_{1.1} := \neg(\mathsf{EF}\ Excursion \sqsubseteq \bot)$
This criterion is satisfied.	This criterion is satisfied.
"There is a path without any additional material."	"There no such object that is on all paths eventually an excursion."
$f_{1.2} := \mathsf{EG}(Excursion \sqsubseteq \bot)$	$f'_{1.2} := \mathsf{AF}\ Excursion \sqsubseteq \bot$
This criterion is satisfied.	This criterion is satisfied.
"On all paths eventually additional material is presented."	"There is some object that is on all paths eventually an excursion."
$f_{1.3} := \mathsf{AF}\ \neg(Excursion \sqsubseteq \bot)$	$f'_{1.3} := \neg(\mathsf{AF}\ Excursion \sqsubseteq \bot)$
This criterion is violated.	This criterion is violated.
"There is a path without any tests."	"There no such object that has on all paths eventually the type $Test$."
$f_{1.4} := \mathsf{EG}(\exists hasType.Test \sqsubseteq \bot)$	$f'_{1.4} := \mathsf{AF}\ \exists hasType.Test \sqsubseteq \bot$
This criterion is violated.	This criterion is violated.
"There is no path without any information units."	"There is some object that has on all paths eventually the type $Information$."
$f_{1.5} := \neg\mathsf{EG}(\exists hasType.Information \sqsubseteq \bot)$	$f'_{1.5} := \neg(\mathsf{AF}\ \exists hasType.Information \sqsubseteq \bot)$
This criterion is met.	This criterion is met.

Table 7.6.: original and new $\mathcal{ALC}$CTL specifications

"temporalized" concepts are generated but, in general, the equivalence to the original version is not preserved (cf. Remark 7.5.1).

In the sequel, $S := \{f_{1.1}, f_{1.2}, ..., f_{1.10}\}$ denotes the standard specification while the alternative specification is denoted as $S' := \{f'_{1.1}, ..., f'_{1.5}, f_{1.6}, ..., f_{1.10}\}$.

To assess, if the system performs differently on satisfied and violated formulae, we built a specification S_{sat} with all formulae being satisfied and a specification S_{vio} with all formulae being violated. S_{sat} is derived from S by negating the violated formulae $f_{1.3}$, $f_{1.4}$, and $f_{1.10}$ of S. S_{vio} is derived from S by negating all satisfied formulae of S.

To examine, how the runtime of $\mathcal{ALC}$CTL document verification scales with the document size, the specifications S and S' are checked against both test series (D1 - D16) and (VLD1 - VLD10). Since an ontology is missing within this experiment, DL reasoning is not required and just the runtimes of the $\mathcal{ALC}$CTL model checker (including error report) and of the framework (document parsing, knowledge extraction, generating the semantic model and temporal document model) are determined. For comparing the performance of $\mathcal{ALC}$CTL model checking with state-of-the-art methods for CTL

model checking, the temporal document model is converted to a NuSMV input file and checked against the CTL formalization $p_{1.1}, ..., p_{1.10}$ of the test criteria.

Expectations and Hypotheses

We expect the following results for each of the involved components.

- $\mathcal{ALC}$CTL model checking is in $\mathcal{O}(|f| \cdot |d|^3)$ in the average case and in $\mathcal{O}(|f| \cdot |d|)$ in the best case (cf. Proposition 6.4.12). Since the size $|f|$ of the formulae being checked remains constant, we expect that the model checking time grows linearly to cubicly from (D1) to (D16) and from (VLD1) to (VLD10). Temporal connectives are more complex to check at the level of concepts than at the level of formulae. Hence, checking S is expected to be faster than checking S'. Since the runtime complexity of $\mathcal{ALC}$CTL model checking does not depend on whether a formula is violated or satisfied, we expect little difference between the runtimes for checking S, S_{sat}, and S_{vio}.
- CTL model checking is in $\mathcal{O}(|f| \cdot |d|)$ in the best and average case (cf. section 6.5.2.2). Hence, we expect a linear growth of runtime of the CTL model checker NuSMV for the test series (D1 - 16) and (VLD1 - 10).
- The framework has to do a lot of "bookkeeping work" such as extracting knowledge from the document and constructing the semantic model. In the presented case, all of these tasks can be done in a straightforward manner and do not involve complex calculations. Thus we expect a linear growth of framework time w.r.t. the document size.

Outcome of the Experiment

Figures 7.4 and 7.5 show the experimental results for checking specifications S and S' on the series of standard documents (D1 - 16).

The total runtime for verifying specification S grows approximately linearly from 0.6 seconds for the smallest document (32 content units) to 2.7 seconds for the largest (512 content units).

The major part of the runtime is consumed by the framework. In case of S, the framework time takes between 88.3% (224 content units) and 91.3% (64 content units) of the total runtime with an average of 89.9%. In case of S', the contribution of the framework ranges between 87.3% (224 content units) and 90.3% (32 content units) with an average of 88.5%. The absolute runtime of the framework is identical for specifications S and S'.

The model checking times seem to grow linearly for both specifications S and S' (Figure 7.5). The relatively large variations of values indicate, however, that the test

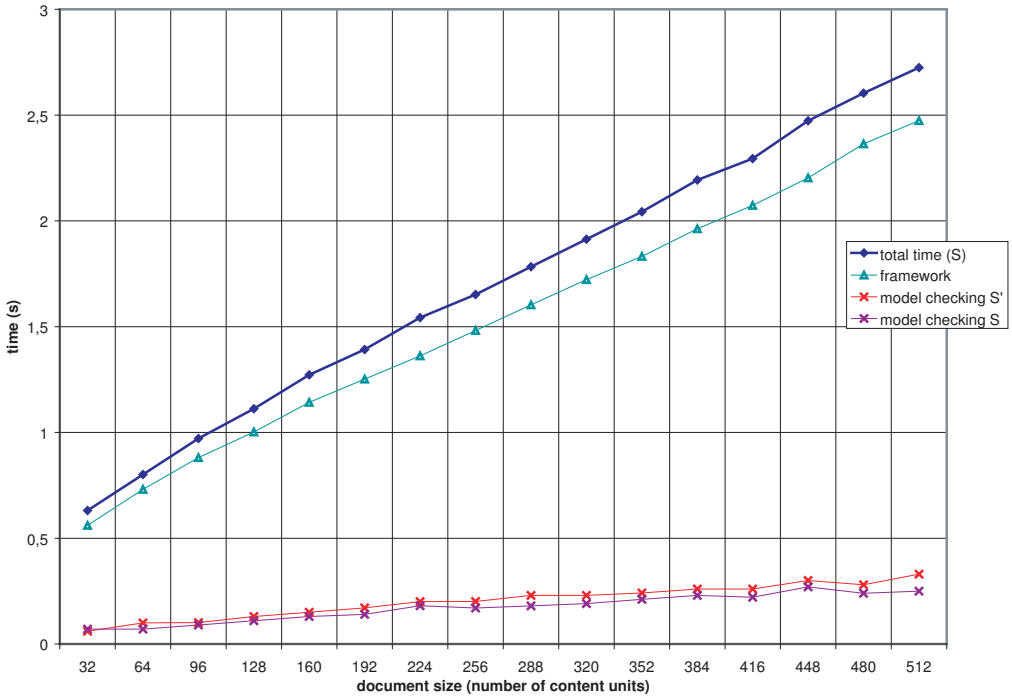

Figure 7.4.: scaling of runtime in the document size

cases are too small for getting runtime results stable enough for predictions on larger documents.

The runtime of the model checking component can, within the limits of this experiment, be approximated by the linear function

$$p(|d|) = (0.48 \cdot |d| + 50)\ ms$$

where $|d| = |CU_d|$ is the number of content units of the document (thick gray line in Figure 7.5).

As expected, the values of model checking specification S' are higher than for model checking S. However, the increase in runtime for S' is rather moderate (17.4% on average).

For comparison, Figure 7.5 includes the results of NuSMV for checking the CTL formulae $p_{1.1}, ..., p_{1.10}$. The runtimes of NuSMV include the construction of a counterexample but do not include the time for constructing the input file from the temporal document structure. Hence, the runtimes of NuSMV are directly comparable with the runtimes of $\mathcal{ALC}$CTL model checking.

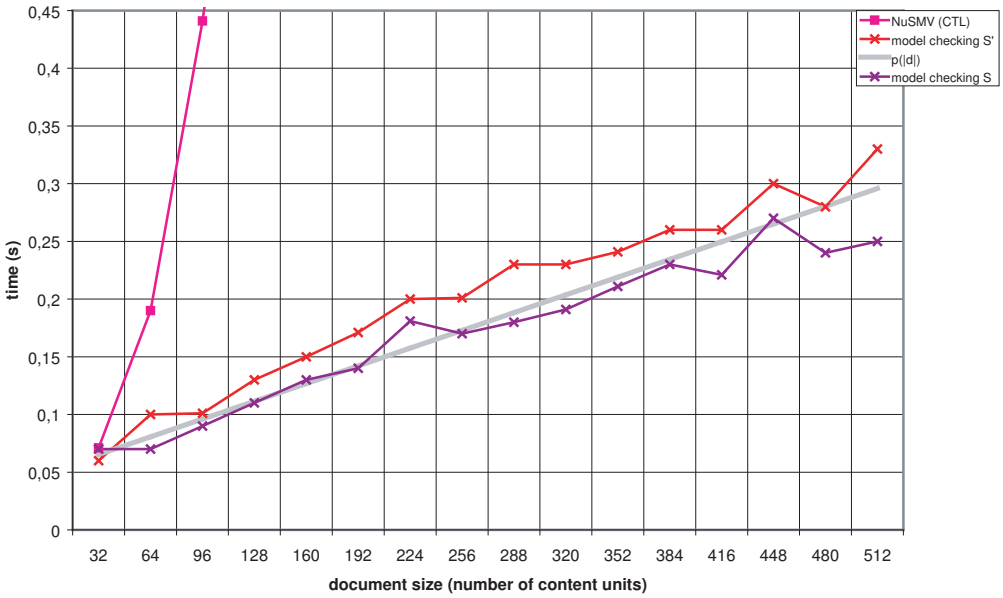

Figure 7.5.: runtime results for specifications S and S'

NuSMV and the $\mathcal{ALC}$CTL model checker deliver exactly the same runtime for the smallest document of 32 content units. However, the runtime of NuSMV increases significantly faster for larger documents (cf. Figure 7.5). This is surprising because the runtime complexity of CTL-based document checking is lower than the complexity of $\mathcal{ALC}$CTL-based document checking in the average case (cf. section 6.5.2.2). Obviously, within the given application scenario, the CTL model checking algorithm implemented in NuSMV is not optimal w.r.t. its runtime complexity.

The runtime of NuSMV already exceeds 0.4 seconds for a document of 96 content units and reaches 25.6 seconds for 512 content units. In contrast, the $\mathcal{ALC}$CTL model checking time for 512 content units is as low as 0.25 seconds for S and 0.33 seconds for S', which is just 1.0% / 1.33% of NuSMV's runtime.

Table 7.7 shows the differences in runtime for checking the specifications S_{sat} (all formulae satisfied), S (3 of 10 formulae violated), and S_{vio} (all formulae violated) on document (D16).

As expected, the runtimes for checking S, S_{sat}, and S_{vio} do not differ much. The total runtime for 512 content units ranges between 2.64 seconds (S_{sat} - all formulae satisfied) and 2.88 seconds (S_{vio} - all formulae violated). The slightly higher runtime of checking S_{vio} results from the longer error report. However, the time for constructing the error report is small compared to the total time. Hence, the impact of the number of violated formulae on the overall performance of the system is negligible.

test case	S_{sat}	S	S_{vio}
# content units	512	512	512
# narrative relations	560	560	560
# formulae	10	10	10
# violated formulae	0	3	10
total runtime (s)	2.64	2.73	2.88
$\mathcal{ALC}$CTL model checker (s)	0.19	0.25	0.41
framework (s)	2.45	2.48	2.47

Table 7.7.: dependency of runtime on the validity of formulae

Figure 7.6 shows how the system copes with very large documents (VLD1 - 10).

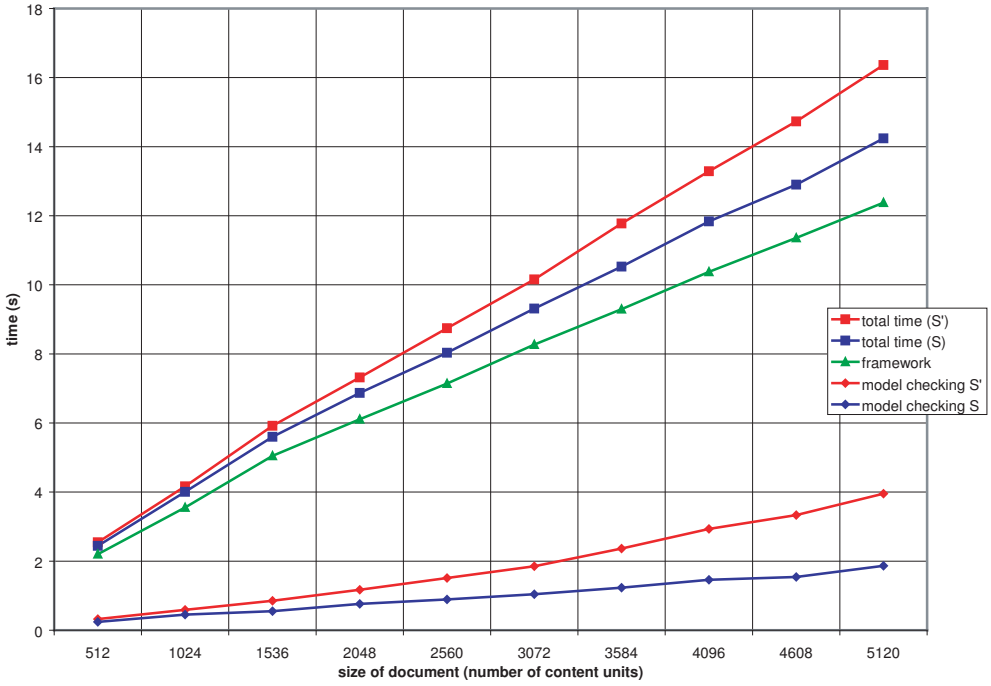

Figure 7.6.: scaling of runtime for very large documents

The scaling of runtime of the single components of the system is identical to series (D1 - D16) of smaller documents except for the model checking time of S' for which a super-linear scaling becomes apparent (Figure 7.6).

The total runtime is still dominated by the framework time although the influence of the model checking time increases for larger documents in the case of specification S'

(red graphs in Figure 7.6). The total runtime grows linearly in the document size with a mininum of 2.44 seconds at 512 content units and a maximum of 14.24 seconds at 5120 content units in the case of checking specification S. For specification S', the total time grows from 2.54 seconds (512 content units) to 16.36 seconds (5120 content units). On a more recent hardware (Intel Core2Duo E6400 processor at 2,13 GHz, 2 GB DDR2 RAM, 250 GB SATA Hard-Disk) the runtimes remain below 6 seconds in all cases.

The framework time of the $\mathcal{ALC}$CTL system is identical for specifications S and S' and grows linearly in the document size from 2.2 seconds (512 content units) to 12.4 seconds (5120 content units).

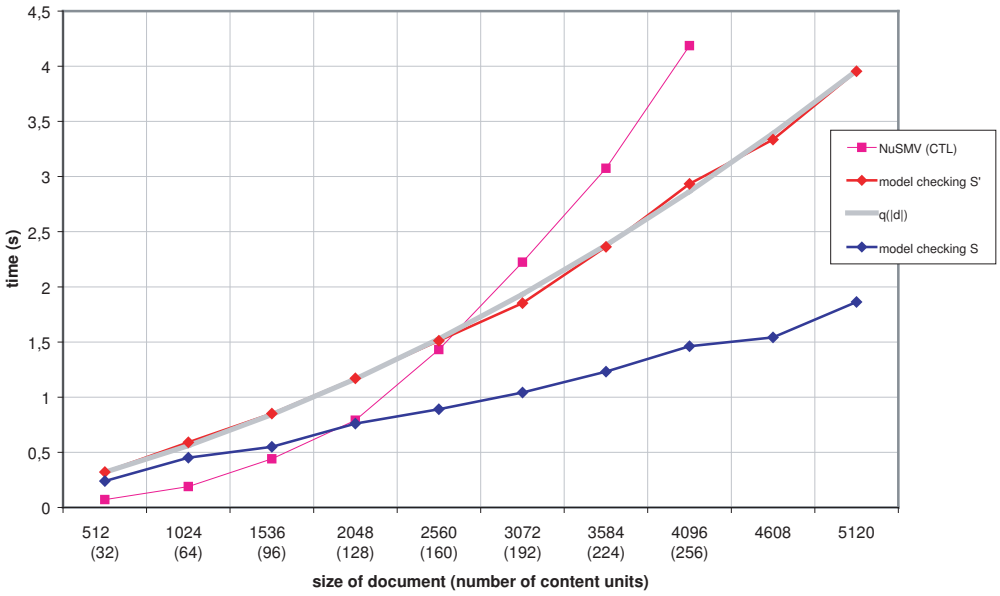

Figure 7.7.: runtime of the $\mathcal{ALC}$CTL model checker for very large documents

More interesting is the scaling of model checking time for very large documents as depicted in Figure 7.7. For better visibility of runtime results, the document sizes for the runtimes of NuSMV are divided by 16 (values in brackets on the horizontal axis of Figure 7.7).

For the simpler specification S, the $\mathcal{ALC}$CTL model checking time grows linearly in the size of the document with the minimum of 0.24 seconds for 512 content units and the maximum of 1.86 seconds for 5120 content units (Figure 7.7). The relative portion of model checking time as compared to the total time increases slightly from 9.8% (512 and 1536 content units) to 13.1% (5120 content units) with an average of 11.3%. A linear scaling can be achieved within the given setting because the formulae

in S do not make full use of the expressiveness of $\mathcal{ALC}$CTL. For instance, temporal connectives appear only at the level of formula terms but not at the level of concept terms.

In contrast to S, the first five formulae of S' contain temporal connectives at the level of concepts. This results in a super-linear growth of model checking time with a minimum of 0.32 seconds for 512 content units and a maximum of 3.96 seconds for 5120 content units (Figure 7.7). In the case of S', the relative amount of model checking time as compared to the total time grows from 12.6% (512 content units) to 24.2% (5120 content units).

The model checking time for S' can be approximated well by the polynomial

$$q(|d|) = (\frac{|d|^2}{12800} + 0.35 \cdot |d| + 120)ms$$

(gray line in Figure 7.7) where $|d| := |CU_d|$ is the number of content units.

The runtime of CTL model checking by NuSMV scales significantly worse in the document size. NuSMV requires more time (4.19 seconds) for checking a document of 256 content units than the $\mathcal{ALC}$CTL model checker requires for 5120 content units (S: 1.86 seconds, S': 3.96 seconds). Checking a document of 512 content units with NuSMV already takes 25.6 seconds such that we refrained from evaluating NuSMV on the series of very large documents (VLD1 - 10).

We suspect that symbolic model checking as implemented in NuSMV does not perform optimally for documents. NuSMV is highly optimized for parallel processes with a huge number of states [BCMH92]. Our application scenario is rather different. The number of states is comparably small but each state has rather complex properties representing the content of a distinct part of the document. Moreover, there is usually little redundancy between the content units of a document, while the states of a system of parallel processes often share many of their properties. This may lead to the unexpected bad performance of NuSMV within the given application scenario.

Experiment 1 - Summary and Conclusions

The results of Experiment 1 show that $\mathcal{ALC}$CTL-based document verification performs well for growing documents. Within the given application scenario, $\mathcal{ALC}$CTL model checking clearly outperforms a state-of-the-art CTL model checker. Within the same time span, $\mathcal{ALC}$CTL model checking can verify documents up to 25 times as large. Obviously, the symbolic model checking techniques applied by NuSMV do not perform optimally for document structures. In contrast, the implemented $\mathcal{ALC}$CTL model checking algorithm uses an explicit representation of states. This turned out to be more efficient for documents the temporal models of which comprise relatively few states.

The $\mathcal{ALC}$CTL model checking time (including error report) for the largest document of this experiment (5120 content units) is as low as 1.9 seconds for a simple specification and 4.0 s seconds for a more complex one. NuSMV requires 4.2 seconds for 256 content units and 25.6 seconds for 512 content units.

The total verification time (including knowledge extraction and model generation) is less than 15 seconds for a document of 5120 content units and a specification of 10 formulae. Hence, even extremely large publications can be verified in a reasonable amount of time.

The total runtime of the system is dominated by the framework time. Consequently, a significant improvement of performance is not possible by tuning the model checking algorithms alone. In future developments the focus of performance optimizations should be shifted towards the collection of required metadata and construction of the semantic and temporal document models.

A promising direction is an incremental approach to knowledge extraction and model generation. In constrast to model checking, which usually requires complete information about the entire document, the knowledge extraction and model generation can be done separately for each component of a document. A document as large as several thousand units is not built within a single step and then verified. Instead, it grows and changes over time. Whenever a component of the document is completed or changed and committed to a document repository, the document model can be updated for the modified component.

In such a scenario, the semantic model and large parts of the document's temporal structure are pre-computed offline before verifying the document. This nearly eliminates the framework time and thus allows for interactive use of the verification system with response times of less than 2 seconds for documents as large as 5000 content units.

7.5.3. Experiment 2 - Influence of the Number of Relations

Description of the Experiment

The documents used in Experiment 1 (Figure 7.3) have a rather linear structure and thus the number of relations per content unit is small ($\frac{35}{32} \approx 1.1$).

In the subsequent experiment, we examine how the verification and particularly the model checking runtime responds to an increasing number of relations per content unit. Therefore, more and more excursions (optional side tracks) are added to an entirely linear document of 256 content units similar to (D8) of Experiment 1. Table 7.8 lists the relevant numbers of each of the 8 test cases.

The verified documents (B0) through (B7) each have $|S_d|$ = 256 content units but the number of relations increases from $|R_d|$ = 312 (B0) to $|R_d|$ = 680 (B7). The size of the interpretation domain $|\Delta_d|$ remains constant at 302.

test case	B0	B1	B2	B3	B4	B5	B6	B7
# content units	256	256	...				...	256
# excursions	0	32	64	96	128	160	192	216
# narrative relations	256	312	376	440	504	568	632	680
# relations / # units	1.0	1.22	1.47	1.72	1.97	2.22	2.47	2.66
$\|\Delta_d\|$	302	302	...				...	302
# formulae	10	10	...				...	10
# violated formulae	4	3	...				...	3

Table 7.8.: test cases for experiment with varying branching factor

The test documents (B0 - 7) are checked against specification S of Experiment 1. In case (B0) one additional formula is violated (last row of Table 7.8) because formula $f_{1.1}$ "there is an excursion reachable" is, as opposed to cases (B1-7), not satisfied.

Expectations and Hypotheses

We expect the runtime of the framework to grow moderately because the amount of extracted metadata and the sizes of the constructed semantic and temporal document models increase slightly from document (B0) to (B7).

The complexity of the $\mathcal{ALC}$CTL model checking algorithm is in $\mathcal{O}(|f|\cdot(|S| + |R|)\cdot|\Delta|^2)$ (Proposition 6.3.41). Since the number $|S|$ of states, the size $|\Delta|$ of the interpretation domain, and the size $|f|$ of the verified formulae remain constant and the number $|R|$ of relations increases approximately linearly from (B0) to (B7), we expect a moderate linear increase in runtime of $\mathcal{ALC}$CTL model checking.

Outcome of the Experiment

Figure 7.8 shows the actual experimental results. The number of narrative relations has little impact on the runtime of any of the evaluated components.

The total time reaches its minium of 1.64 seconds for the cases (B1),(B2), and (B3) and its maximum of 1.69 seconds for case (B7). The relative difference between minimal and maximal total time is just 3%.

As expected, the framework time increases slightly from (B0) to (B7). The minimal framework time is 1.47 seconds (B0, B2, B3). The maximum of 1.51 seconds is reached for case (B7).

Also, the runtime of the $\mathcal{ALC}$CTL model checker is hardly affected by the number of narrative relations. The variation of values remains within the measurement tolerance. The runtimes are between 0.16 and 0.18 seconds for all cases, which is between 9.7% and 10.9% of the total runtime.

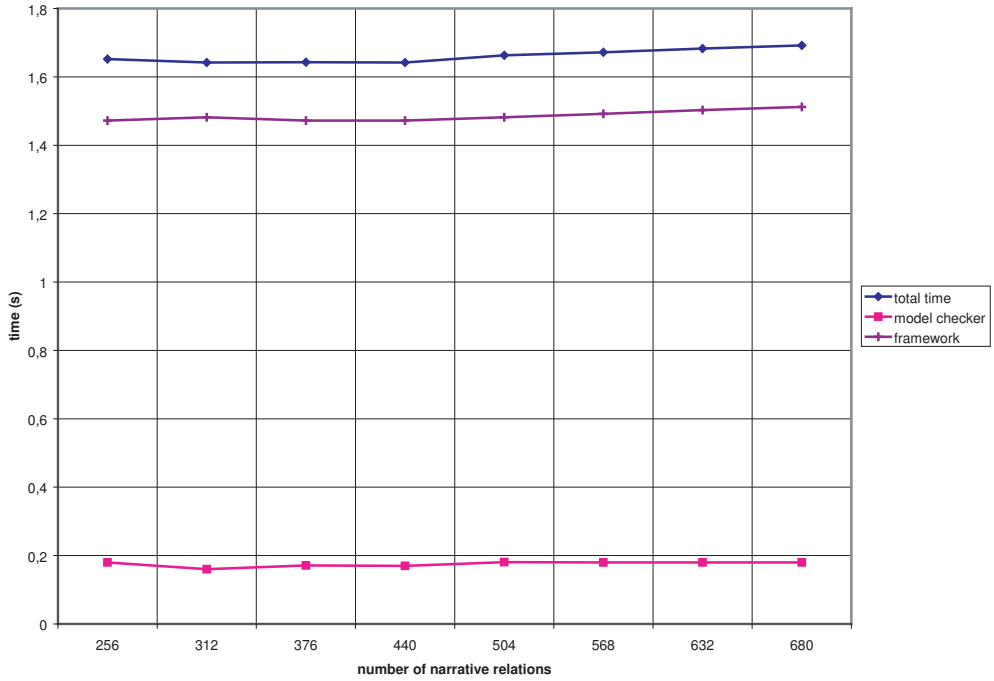

Figure 7.8.: scaling of runtime in the number of narrative relations

Experiment 2 - Summary and Conclusions

The results of Experiment 2 indicate that the performance of the system is hardly affected by the number of narrative relations within a document. The runtime of the system can be expected to be stable and predictable for different document structures.

7.5.4. Experiment 3 - Impact of the Reference Ontology

Description of the Experiment

The previous experiments do not apply a reference ontology. In absence of a reference ontology the system performs significantly faster because the $\mathcal{ALC}$CTL temporal document structure, which is checked against the set of $\mathcal{ALC}$CTL formulae of a specification, can be constructed without expensive DL reasoning.

When a reference ontology is given, the instance retrieval service of a DL reasoner such as RacerPro is used for computing the temporal interpretation I_d of the temporal document structure $(S_d, R_d, \Delta_d, I_d)$ (Algorithm 6.4.7). Recall that the temporal interpretation $\mathcal{A}^{I_d(s)}$ of an atomic concept $\mathcal{A} \in AC$ is computed as

$$\mathcal{A}^{I_d(s)} := \{a \in IV_{KB_s} \mid KB_s \models \mathcal{A}(a)\}$$

where KB_s is the local knowledge base of the content unit $s \in S_d$ and IV_{KB_s} is the set of individuals occuring in KB_s (Definition 6.4.1).

Recall further that the local knowledge base KB_s of a content unit $s \in S_d$ comprises the reference ontology RO, the global fact base gAB_d, and the local fact base AB_s representing information about the content of unit s (Definition 5.3.10). The number of local knowledge bases $|\mathbf{KB}_d|$ of a document grows linearly in the document size while the size $|KB_s|$ of a single local knowledge base $KB_s \in \mathbf{KB}_d$ stays constant.

In this experiment, we examine how the verification and particularly the DL reasoning time responds to increasingly complex reference ontologies. The documents and formulae used in this experiment are identical to those of Experiment 1 (section 7.5.2). However, reference ontologies RO of different complexity are added.

Apart from performance issues, we are interested in how different reference ontologies influence the quality of verification results: can additional errors be detected when using appropriate reference ontologies?

To approaching these questions, four ontologies have been built:

R1 : this reference ontology has a minimal complexity. It basically consists of domain and range definition of seven roles $hasType$, *hasDifficulty*, $forUserRole$, $isInFile$, $teaches$, $asks$, $relatedToFile$ used in the fact bases of the document's knowledge representation. Furthermore, (R1) contains the following axioms to classify different kinds of content units:

$$\begin{aligned} ContentUnit &\doteq TestUnit \sqcup ExpositionUnit \\ TestUnit &\doteq Testbegin \sqcup Testquestion \sqcup Testresult \\ Excursion &\sqsubseteq ExpositionUnit \end{aligned}$$

R2 : this ontology is an extension of (R1) in the following aspects:

- two additional equivalences between concepts and one additional implication between concepts are defined as follows:

$$\begin{aligned} ContentProperty &\doteq ContentType \sqcup Difficulty \sqcup UserRole \sqcup \\ &\quad Source \sqcup Topic \\ \top &\doteq ContentUnit \sqcup ContentProperty \\ File &\sqsubseteq Source \end{aligned}$$

- three disjointness axioms such as $ContentUnit \sqcap ContentProperty \doteq \bot$ are added.
- three additional roles $hasSource, hasTopic, relatedTo$ are defined as super-roles of respective roles $isInFile, teaches/asks, relatedToFile$ in (R1).

- the inverse of every role is defined in ten additional axioms.
- the roles $hasType$, *hasDifficulty*, $isInFile$, and $hasSource$ are defined as right-unique.

(R2) can be considered as a realistic reference ontology with moderate complexity.

R3 : this ontology is just a slight extension of (R2). The following two equivalence axioms between concepts are added:

$$\begin{aligned} TestUnit &\doteq \exists hasType.(Test \sqcup Exercise) \\ ExpositionUnit &\doteq \exists hasType.Information \end{aligned}$$

R4 : represents a realistic but rather complex ontology. Compared to (R3), the additional roles $hasPart$ and $partOf$ as well as four further concept equivalence and seven further concept implication axioms are added. They are of a rather complex nature using negation, universal quantification, existential quantification, "at-most" and "at-least" qualified number restrictions. Nominals and concrete domains are not used.

Table 7.9 lists the relevant numbers of each case. (R0) is the setting of Experiment 1 without a reference ontology.

In this experiment, the documents (D1-16) and specification S of Experiment 1 (section 7.5.2) are used. Although the presence of a reference ontology would allow for simplifying some of the formulae within the specification S, we refrained from such modifications to keep the verification results comparable.

The runtime results for DL reasoning are obtained using the reasoning systems Racer version 1.7.23, RacerPro version 1.9.0, and Pellet 1.5.0 (section 3.1.5). These systems have shown the best performance in preliminary tests.

Expectations and Hypotheses

Since the size of each local knowledge base does not grow with the document size, we expect a linear increase in runtime in the document size for each of the reference ontologies (R1 - 4) (cf. Proposition 6.4.12).

We expect that the system performs worse for the more complex ontologies than for the simpler ones. The impact of the complexity of each of the ontologies on the runtime is hard to predict because ontological reasoning can be already very hard for simple knowledge bases (cf. section 3.1.4).

Regarding the different reasoning systems, we expect a better performance of the commercial system RacerPro than of the non-commercial systems Racer and Pellet.

The remaining components of the framework are hardly affected by the size and complexity of the reference ontology.

test case	R0, R1, R2, R3, R4				
#content units	256				
#narrative relations	280				
#assertions in cKB_d	1232				
#formulae	10				
#violated formulae	3				
reference ontology	**R0**	**R1**	**R2**	**R3**	**R4**
# concepts	0	12	14	17	29
# roles	0	7	20	20	22
# concepts and roles	0	19	34	37	51
# concept equiv. axioms	0	2	4	6	10
# concept impl. axioms	0	1	2	2	9
# disjointness axioms	0	0	3	3	3
# role domain defs	0	7	8	8	9
# role range defs	0	7	7	7	8
# inverse role defs	0	0	10	10	11
# right-unique roles	0	0	4	4	4
# role impl. axioms	0	0	4	4	4
# axioms	0	17	42	44	58

Table 7.9.: test case with reference ontologies of different complexity

- The reference ontology has influence on the temporal interpretation I_d of the $\mathcal{ALC}$CTL temporal document structure $(S_d, R_d, \Delta_d, I_d)$. This can lead to a reduced but also to an increased model checking time depending on the criteria to check. As for the average case, we expect that the reference ontology has no effect on the $\mathcal{ALC}$CTL model checking time.
- The framework time is little affected by the size of the reference ontology. The framework loads the reference ontology into the reasoner, which may take a noticeable time for very large reference ontologies. As for the reference ontologies used in this experiment, the upload time should be negligible and thus we expect no relevant changes in the framework time.

No changes are expected for the verification results (set of satisfied and violated formulae).

Outcome of the Experiment

Qualitative Results

Surprisingly, the use of reference ontologies revealed further errors.

- Already the simplest ontology (R1) turned out to be useful as it helped to reveal a typo in an early version of the specification. This is because the reasoners accept concepts and roles only that are defined within the reference ontology. Of course, the final results of all experiments as presented in this chapter have been acquired based on the corrected specification without any flaws.
- A preliminary version of (R2) contained 4 instead of 3 disjointness axioms. The additional disjointness axiom was

  $$ExpositionUnit \sqcap TestUnit \doteq \bot \tag{7.6}$$

  "No thing (content unit) is both an exposition unit and a test unit".

  This disjointness axiom, however, led to an inconsistent knowledge base because some content unit of the document was actually classified as an exposition unit and test unit because of contradictive metadata about the content unit. This indicated an error in the metadata attributes of the document that was not previously known.

  For keeping the documents of experiment 3 identical to the documents used in other experiments, we opted for not correcting the error in the document but for discarding the disjointness axiom of Equation (7.6) from the reference ontologies (R2), (R3), and (R4).
- The additional axioms of (R3) as compared to (R2) revealed further problems within the document regarding the metadata about test units (formula $f_{1.10}$ of Experiment 1). As a consequence of the additional axiom $TestUnit \doteq \exists hasType.(Test \sqcup Exercise)$, more content units of the document are classified as "test units", which do not satisfy formula $f_{1.10}$. Formula $f_{1.10}$ is also violated in cases (R0), (R1), and (R2), but with a lower number of counterexamples.

The results of Racer, RacerPro, and Pellet do not differ exept for the most complex ontology (R4). Using (R4), Pellet does not deliver any reasoning results but returns a severe internal server error ("null pointer exception"). The error could be eliminated by removing the axiom

$$\begin{aligned} ContentUnit \quad \sqsubseteq \quad & \exists hasType.\top \sqcap \exists \textit{hasDifficulty}.\top \sqcap \\ & \exists forUserRole.\top \sqcap \exists hasSource.\top \end{aligned}$$

from (R4). This is surprising since this axiom is not the most complex axiom by far and is also not structurally different from other axioms of (R4). The runtime of Pellet for (R4) of Table 7.10 is obtained by using the modified slightly smaller version (R4*) of (R4) and thus is not directly comparable with the runtimes of Racer and RacerPro, which are obtained by using the original full version of (R4).

Quantitative Results

Table 7.10 lists the runtime results for document (D8) of 256 content units being checked against specification S of Experiment 1 containing 10 formulae.

test case	R0	R1	R2	R3	R4 / R4*
total time using RacerPro	1.57	434.4	1750	2768	8302
total time using Pellet	1.57	11.8	14.1	15.7	17.1
RacerPro	–	432.9	1749	2766	8301
Racer	–	26.9	58.4	59.2	98.8
Pellet	–	10.3	12.5	14.2	15.6*
$\mathcal{ALC}$CTL model checking	0.18	0.13	0.13	0.13	0.14
framework	1.39	1.41	1.42	1.41	1.41

Table 7.10.: runtime results for 256 content units using different reference ontologies

As predicted, the reference ontology only has little effect on the runtime of the $\mathcal{ALC}$CTL model checker and of the framework.

The slightly lower runtime of the $\mathcal{ALC}$CTL model checker in cases (R1 - 4) results from internal optimizations of the representation of the temporal document structure when a reference ontology is available. Compared to the total time, however, the differences in model checking time are negligible.

In contrast to the model checking time, the framework time in case (R0) is slightly lower than in the other cases (Table 7.10 last row). The higher framework time in cases (R1 - R4) stems from sending the reference ontology to the reasoning system and is irrelevant for the total runtime of the system.

The total time is clearly dominated by the runtime consumed by the DL reasoning component. Even in the case of the best performing reasoner Pellet and the simplest ontology (R1), the total runtime increases by a factor of 7.5 from 1.57 to 11.8 seconds for a document of 256 content units (Table 7.10).

The runtimes of the reasoners RacerPro, Racer, and Pellet differ extremely. Surprisingly, the commercial system RacerPro performs extremely poor – much worse than its free predecessor system Racer. Already for the smallest ontology (R1), the runtime of RacerPro exceeds 7 minutes for a document of 256 content units, which is beyond any acceptable limit. For more complex ontologies (R2), (R3), and (R4), the runtime

of RacerPro degrades even more. In the most complex case (R4), RacerPro takes more than 2 hours, which is about 19.2 times as much time as in the simplest case (R1).

The precedessor of RacerPro, Racer, actually performs much better in all cases. Obviously, the optimizations of RacerPro as compared to Racer are counterproductive within the setting of Experiment 3. The runtime of Racer ranges between between 27 seconds for (R1) and 99 seconds for (R4). The runtime of Racer for the most complex ontology (R4) is just 3.7 times as high as for the simplest one (R1).

Pellet clearly outperforms both RacerPro and Racer . The runtimes of Pellet lie between 10.3 seconds for (R1) and 15.6 for (R4). In addition, Pellet shows the lowest relative increase in runtime for the most complex ontology (R4) as compared to the most simple one (R1): Pellet takes about 1.5 times as long for (R4) as for (R1).

The absolute values for Pellet for a medium sized document of 256 content units are acceptable for a non-interactive application scenario but too high for smooth interactive use. More important than the absolute values for a medium sized document is the scaling of runtime for growing documents.

We assumed a linear scaling of reasoning time in the document size for all ontologies and reasoning systems.

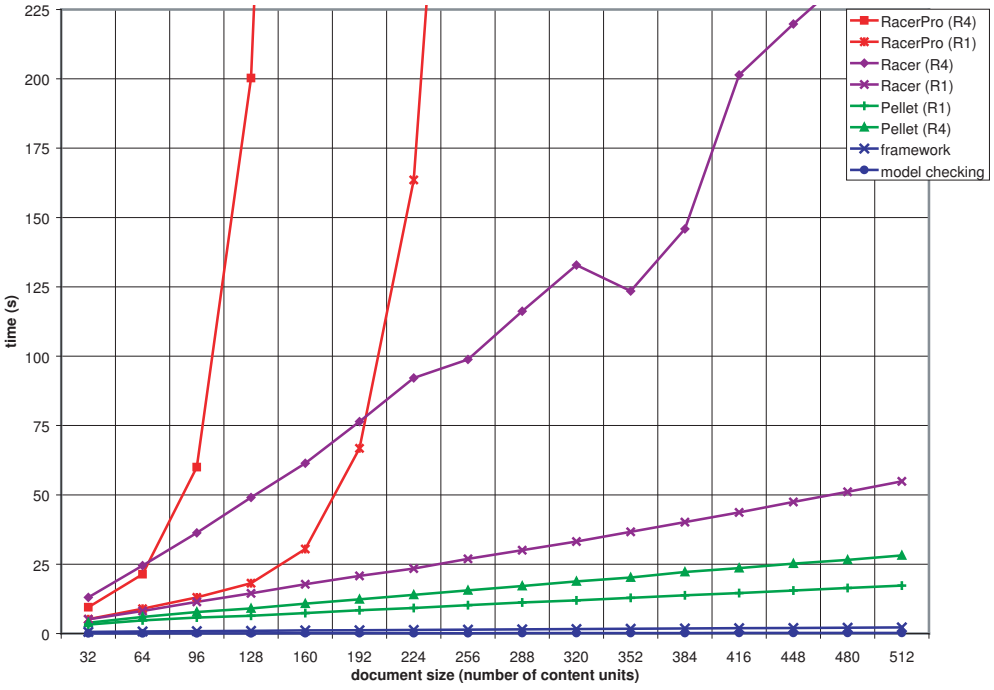

Figure 7.9.: scaling of reasoning time for different reference ontologies

Figure 7.9 shows the scaling of runtime of RacerPro, Racer, and Pellet for ontologies (R1) and (R4). Contrary to expectations, RacerPro does not scale linear in the document size neither in case of (R1) nor in case of (R4) (red graphs in Figure 7.9). Using the very simple ontology (R1), the runtime of RacerPro grows linearly for documents with less than 160 content units. From 160 content units on the runtime suddenly increases. Using the most complex ontology (R4), the runtime already explodes for documents larger than 64 content units. This runtime behaviour cannot be explained from the test data because the knowledge bases for reasoning do not become larger or more complex for larger documents. Obviously, there is some problem in the internal memory management of RacerPro, which leads to a drastic degeneration of runtime after a certain amount of submitted and released knowledge bases.

Racer behaves more stable within our evaluation scenario (Figure 7.9). At least in the case of (R1) the runtime shows the expected linear scaling in the document size. In the case of (R4), however, the runtime of Racer displays some anormalities: the document of 352 content units can be handled more quickly than the document of 320 content units. Moreover, the runtime increase from 384 to 416 is significantly larger than in the other cases. Obviously, Racer also suffers from internal memory reallocation problems, which results in a rather unstable and unpredictable performance in the case of complex ontologies and larger documents.

In contrast to Racer and RacerPro, the runtime results for Pellet meet our expectations exactly. The runtime grows strictly linearly in the size of the document for both cases (R1) and (R4). Pellet performs quicker for (R4) than Racer and RacerPro for (R1). Also, the relative difference between (R4) and (R1) remains small for all document sizes: on average Pellet requires 50% more time for ontology (R4) than for ontology (R1). The absolute runtime of Pellet increases from 3.2 seconds for 32 content units to 17.3 seconds for 512 content units in the case of (R1), and from 3.9 seconds for 32 content units to 28.2 seconds for 512 content units in the case of (R4). Hence, Pellet is sufficiently quick to handle even large documents and complex ontologies in acceptable time. In any case, however, the runtimes are too large for smooth interactive use.

Even when applying the simple ontology (R1), the runtime of Pellet clearly dominates the runtime of the system and consumes between 83% and 88% of the total runtime for different document sizes. In case of (R4), the relative amount of runtime consumed by Pellet ranges between 86% and 92% of the total time.

Experiment 3 - Summary and Conclusions

Experiment 3 revealed that the performance of the $\mathcal{ALC}$CTL verification system is highly depending on the performance of the integrated reasoning system if a reference ontology is applied.

The performance of different reasoning system varies extremely within our application scenario. RacerPro, one of the best performing reasoners on the market according to standard benchmarks [GPH05, MYQ+06, WLLB06], performed extremely poorly within the $\mathcal{ALC}$CTL system. In contrast, the comparably simple open source reasoner Pellet shows acceptable runtime results that vary little for reference ontologies of different complexity and remain below 30 seconds even in case of complex ontologies and large document (512 content units). Hence, the general feasibilty and the performance of the overall approach for application relevant problem sizes is demonstrated.

Even in the case of using Pellet and a simple reference ontology, however, the total runtime increases by a factor of at least 7.5 as compared to using no ontology. In addition, Pellet shows functional weaknesses within our evaluation scenario. In the case of the most complex ontology, Pellet returned an internal error for unclear reasons and refused to deliver any reasoning results. Further evaluation of reasoning systems are necessary to determine the best performing and most reliable system.

The poor performance of current DL reasoning systems allow interactive use for small documents (up to 100 content units) only. For larger documents, the $\mathcal{ALC}$CTL temporal document structure should be calculated offline, for instance, whenever a new version of a document is committed to the document repository. This is possible because the verification of the specification by $\mathcal{ALC}$CTL model checking is separated from the construction of the $\mathcal{ALC}$CTL temporal document structure in Algorithm 6.4.7.

In addition, the modularity of knowledge representation and the monotonic behaviour of DL reasoning ideally support an incremental approach because previously calculated results remain valid if new metadata or new components are added to the document. If just a few parts of the documents are changed or added, the $\mathcal{ALC}$CTL temporal document structure only needs to be updated in parts and the major portion of the necessary DL reasoning can be saved.

7.5.5. Experiment 4 - Scaling in the Size of the Specification

Description of the Experiment

In the last experiment, we examine how the systems copes with increasingly large specifications.

This experiment is based on document (D16) of Experiment 1. (D16) consists of 16 sections each containing 32 pages / content units (Figure 7.3). (D16) is checked against 10 specifications the size of which varies between 3 and 30 formulae (Table 7.11). All specifications are built using a base set of the following three formulae:

1. "Additional material is reachable."

   $\mathcal{ALC}$CTL : $f_{4.1} :=$ EF $\neg(Excursion \sqsubseteq \bot)$
   CTL : $p_{4.1} :=$ EF $excursion$

   This criterion is satisfied.

2. "Additional material is found eventually on every path."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{4.2} := & \mathsf{AF}\ \neg(Excursion \sqsubseteq \bot) \\ \mathsf{CTL}: & p_{4.2} := & \mathsf{AF}\ excursion \end{array}$$

   This criterion is violated.

3. "Whenever some information is presented, a test is reachable."

$$\begin{array}{lll} \mathcal{ALC}\mathsf{CTL}: & f_{4.3} := & \mathsf{AG}((\exists hasType.Information \sqsubseteq \bot) \vee \\ & & \mathsf{EF}\ \neg(\exists hasType.Test \sqsubseteq \bot)) \\ \mathsf{CTL}: & p_{4.3} := & \mathsf{AG}(hasTypeInformation \rightarrow \mathsf{EF}\ hasTypeTest) \end{array}$$

   This criterion is met.

As in Experiment 1, the criteria have been chosen in such a way that they can be equally well represented in $\mathcal{ALC}$CTL and CTL . This enables the direct comparability of the runtime results for $\mathcal{ALC}$CTL and CTL model checking.

For growing specifications, we can distinguish two scenarios of different complexity:

- in the *best case* scenario, the total number of sub-expressions occurring within the specification does not grow with the size of the specification but remains constant. In this scenario, there are a growing number of formulae but each formula shares its sub-expressions with other formulae of the specification.

  We simulate the best case scenario by simply repeating the base set of three formulae within a specification. This results in specifications of 3, 6, 9, ..., 30 formulae such that each specification contains 3 *different* formulae only (Table 7.11).

- in the *worst case* scenario, none of the formulae within a specification shares any sub-expression with another formula. In this scenario, the number of sub-expressions used within the specification grows linear with the size of the specification.

  We simulate the worst case scenario by introducing new concepts and relations for every formula of the specification. This results in specifications of the following shape:

  $\mathsf{EF}\ \neg(Excursion_1 \sqsubseteq \bot)$
  $\mathsf{AF}\ \neg(Excursion_2 \sqsubseteq \bot)$
  $\mathsf{AG}((\exists hasType_1.Information_1 \sqsubseteq \bot) \vee \mathsf{EF}\ \neg(\exists hasType_1.Test_1 \sqsubseteq \bot))$
  $\mathsf{EF}\ \neg(Excursion_3 \sqsubseteq \bot)$
  $\mathsf{AF}\ \neg(Excursion_4 \sqsubseteq \bot)$
  $\mathsf{AG}((\exists hasType_2.Information_2 \sqsubseteq \bot) \vee \mathsf{EF}\ \neg(\exists hasType_2.Test_2 \sqsubseteq \bot))$
  $\mathsf{EF}\ \neg(Excursion_5 \sqsubseteq \bot)$
  ...

The concepts $Excursion_1, Excursion_2, ...$ have the same set of instances but are treated as different concepts. This way. the smallest specification (F1) comprising three formulae uses four different concepts and one role, (F2) containing six formulae has eight different concepts and two different roles, and (F10) with 30 formulae uses 40 different concepts and 10 different roles in total (cf. Table 7.11).

	F0	F1	...	F10
# content units	512	512	...	512
# narrative relations	560	560	...	560
# formulae	3	6	...	30
# different formulae (best / worst case)	3 / 3	3 / 6	...	3 / 30
# concepts (best / worst case)	3 / 4	3 / 8	...	3 / 40
# roles (best / worst case)	1 / 1	1 / 2	...	1 / 10
# violated formulae	1	2	...	10

Table 7.11.: test cases for the experiment with a varying number of formulae

Expectations and Hypotheses

As for the components of the system we expect the following runtime behaviour.

- $\mathcal{ALC}$CTL model checking is in $\mathcal{O}(|f|\cdot(|S|+|R|)\cdot|\Delta|^2)$ (Proposition 6.3.41). The number of states S and relations R stays constant within the test series (F1 - 10). The size $|f|$ of the formulae to check can be identified with the number of formulae within the specification because a specification containing several formulae can be simulated by a single formulae, which is the conjunction of all formulae of the specification. Consequently, we expect a linear growth of runtime of the $\mathcal{ALC}$CTL model checker for both the best and the worst case scenario. Since the $\mathcal{ALC}$CTL model checker checks for redundant sub-expressions within a specification, the model checking time should be noticably lower in the best case scenario than in the worst case scenario.

- Also, CTL model checking is linear in the size of the formula (Theorem 3.2.12). Hence, we expect a linear growth of runtime of model checking the CTL version of the specification using NuSMV.

- The framework requires some time to parse the specification. Hence, we except a moderate linear growth of runtime for the framework in the best as well as worst case scenario.

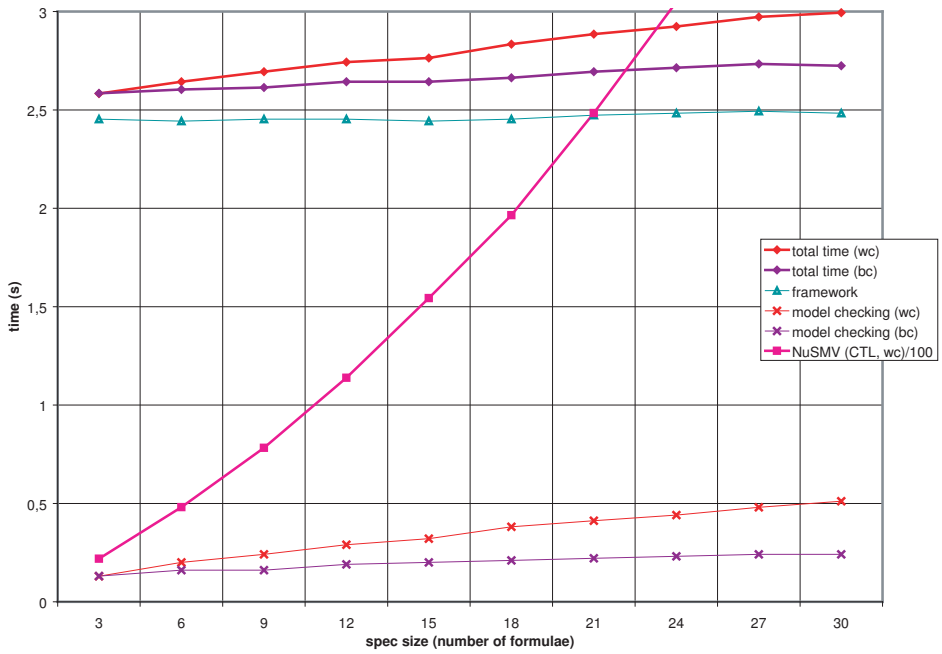

Figure 7.10.: scaling of runtime for a growing number of formulae

Outcome of the Experiment

Figure 7.10 shows the benchmark results for the best case and the worst case scenario.

The runtime results of the $\mathcal{ALC}$CTL system meet our expectations. The framework time increases slightly from 2.44 seconds (F2, F5) to 2.49 seconds (F9). The framework time within the best case scenario does not differ from the framework time within the worst case scenario.

The model checking time apparently grows linearly in the size of the specification in both the best and the worst case scenario. However, the runtimes of the worst case scenario grow about three times as fast as the runtimes of the best case scenario. The absolute values increase from 0.13 seconds for 3 formulae to 0.24 seconds for 30 formulae in the best case scenario and from 0.13 seconds for 3 formulae to 0.51 seconds for 30 formulae in the worst case scenario.

The total runtime increases from 2.58 seconds for three formulae to 2.72 seconds for 30 formulae in the best case scenario and from 2.58 seconds for 3 formulae to 2.99 seconds for 30 formulae in the worst case scenario. The relative difference of total time between the specification of 3 formulae and the specification of 30 formulae is 5.4% in the best case and 15.9% in the worst case scenario.

Again, the $\mathcal{ALC}$CTL model checker performs significantly better than NuSMV (Figure 7.10). Note that in Figure 7.10 the runtimes of NuSMV for checking the CTL versions of the worst case specifications are divided by 100. NuSMV takes 21.9 seconds for checking 3 formulae and more than 7 minutes for checking 30 formulae. Compared to $\mathcal{ALC}$CTL model checking, NuSMV takes 168 times as much time for 3 formulae and 836 times as much time for 30 formulae.

Surprisingly, NuSMV shows a super-linear growth of runtime in the size of the specification (Figure 7.10), which indicates that the symbolic model techniques applied by NuSMV do not perform optimally within the given setting.

Experiment 4 - Summary and Conclusions

The $\mathcal{ALC}$CTL-based document verification scales well to large specifications, i.e. the size of the specification has little effect on the total runtime of the system. This is because the total runtime is dominated by the framework time that is little affected by the size of the specification. The model checking time scales, in accordance with analytical results (Propositions 6.3.41 and 6.4.9), linearly in the size of the specifications. The increase in model checking time is moderate even in the worst case scenario (no shared sub-expressions) and the model checking time does not exceed 0.51 seconds for specifications of 30 formulae verified on documents of 512 content units.

The performance of $\mathcal{ALC}$CTL model checking exceeds that of CTL model checking by up to three magnitudes even for criteria that can be expressed equally well in CTL and in $\mathcal{ALC}$CTL. Since, in many cases, a CTL-based formalization of criteria on documents results in much more formulae than an equivalent $\mathcal{ALC}$CTL-based formalization (cf. Example 6.5.9 and section 7.4.1.4), $\mathcal{ALC}$CTL model checking outperforms CTL model checking rather more than demonstrated by the results of this experiment.

7.6. Summary of Findings

The major findings of the presented experiments are as follows.

- $\mathcal{ALC}$CTL is sufficiently expressive for representing useful content-related properties. As compared to CTL, $\mathcal{ALC}$CTL offers useful additional expressiveness as demonstrated by the WBT case study in section 7.4.
- Reference ontologies are a useful but expensive tool for increasing the robustness and precision of the verification system. When applying an ontology, the performance of the system is dominated by the efficiency of the adopted DL reasoning system. Using an efficient reasoner, the runtime of the system scales linearly in the document size even when applying complex ontologies. The evaluated reasoning systems, however, are too slow for a smooth interactive use of the system such that DL reasoning should be done offline.

- The $\mathcal{ALC}$CTL system performs extremely well if no reference ontology is used. Documents of 5000 content units ($\approx$ pages) can be verified in a few seconds on an up-to-date personal computer. The runtime of the system increases linearly in the document size in many scenarios. Just for very large documents and complex specifications a quadratic scaling of runtime in the document size is observed.
- Within our scenario, the relatively simple explicit state algorithms applied for $\mathcal{ALC}$CTL model checking clearly beat the performance of the symbolic model checking algorithms of NuSMV. $\mathcal{ALC}$CTL model checking offers a significantly better scaling in both the document and specification size. This contrasts with the analytical runtime results for CTL and $\mathcal{ALC}$CTL model checking (section 6.5.2.2).
- Model checking the specification consumes a rather small portion of the overall runtime. Knowledge extraction and model generation take the major part of the system's runtime.
- The runtime of the system is hardly influenced by the number of narrative relations, the size of the specification, and the number of violated or satisfied formulae. The runtime of the system remains stable and predictable in different settings.

The major implications drawn from the experimental results are the following.

- The feasibility, usefullness, and scaling of the approach to application relevant problem sizes is demonstrated.
- Pre-processing tasks such as knowledge extraction and model generation should be performed offline and separated from core verification tasks. This allows for online verification of documents as large as 5000 content units in less than 2 seconds.
- Reasoning systems should be selected with care and evaluated thoroughly before using them in a productive environment. Any of the tested reasoning systems displayed severe functional or performance-related problems in the given setting.

8. Comparison with XML Validation

In the course of this chapter, we compare $\mathcal{ALC}$CTL-based document verification with existing methods for XML documents. XML documents are not the only but certainly one of the most relevant use case of the presented approach. There are a variety of methods for ensuring the validity and consistency of XML documents (section 2.4.1). Among them, Schematron and CLiX are most relevant.

8.1. Comparing $\mathcal{ALC}$CTL with Schematron

A Short Introduction to Schematron

Schematron is an ISO standard [ISO06] for specifying business rules on XML documents. It is well-supported by a range of commercial and non-commercial tools (see section 2.4.1.2).

The basic approach of Schematron is simple. A consistency *rule* is represented by two XPath expressions: the first one selects a *context* (set of nodes) in which a certain condition is evaluated. The second XPath expression represents a condition that must or must not be met by all nodes of the selected context. In addition, simple means for generating error messages in the case of violated rules are provided.

Example 8.1.1 (Schematron Rule)

The criterion ”every defined term must be used in the sequel of its definition” can be represented by the following Schematron rule:

```
<sch:rule context="//definition/definedTerm">
  <sch:assert test= "text() = ../following::usedTerm/text()"
        diagnostics="d1">
     Every defined term is used in the sequel of
     its definition.
  </sch:assert>
</sch:rule>

<sch:diagnostics id="d1">
    Term "<value-of select='text()'/>" defined in definition
    <value-of select='../@title'/> is not used in the sequel.
</sch:diagnostics>
```

The XPath expression `//definition/definedTerm` of the element `sch:rule` selects every `definedTerm` element within a `definition` element of the XML document. Defined terms are the context for evaluating the subsequent assertion.

For each context element, the assertion
`text() = ../following::usedTerm/text()` is evaluated. The assertion is satisfied iff the text of each `definedTerm` element is equal to the text contained in some `usedTerm` element following the parent of the `definedTerm` element w.r.t. the document order of elements.

The element `sch:assert` can optionally refer to a `sch:diagnostics` element that defines an error message for each violating context element. □

A Schematron specification or *schema* is basically a structured set of rule and diagnostic definitions (cf. [Vli07] for further examples).

The evaluation of such a schema on a set of documents runs in two steps. In the first step, the Schematron schema is translated into an XSL stylesheet. This can be done independently from the documents to be validated. In the second step, the translated schema is applied to documents by using a standard XSLT processor.

The approach of Schematron is simple yet powerful. Even complex criteria can be represented by XPath expressions. In addition, Schematron inherits many advanced XSLT features such as parameters, variables, `key` definitions, and various pre-defined XSLT functions [ISO06] that further extend the expressive power of Schematron rules.

In addition, Schematron can be bound to query languages different to XPath as long as they support the basic mechanisms of Schematron such as context and constraint definitions [ISO06]. Suitable alternative query languages include XQuery [W3C07c] and XSLT [W3C99b, W3C07d].

These query languages are Turing-complete [Kep04] and thus theoretically expressive enough to evaluate any decidable constraint on XML documents (assuming an unrestricted amount of available memory and time).

Comparison with $\mathcal{ALC}$CTL

In contrast to Schematron, $\mathcal{ALC}$CTL does not have universal expressiveness. However, the kind of criteria being expressible in $\mathcal{ALC}$CTL are very hard to realize and inefficient to check using XSLT processing and XPath, which are fundamental to Schematron. This will be demonstrated in detail by a number of experiments presented in section 8.3.4. Further, consider that the $\mathcal{ALC}$CTL system applies XSLT for extracting knowledge from XML documents. Hence, within the presented framework the expressive power of XSLT and XPath can be utilized for pre-computing properties not expressible in $\mathcal{ALC}$CTL (see section 8.3.3).

8.2. Comparing $\mathcal{ALC}$CTL with CLiX

A Short Introduction to CLiX

CLiX (Constraint Language in XML) is a proprietary format for consistency rules on XML documents. As opposed to Schematron, CLiX is supported by commercial tools only (section 2.4.1.2).

CLiX is a combination of XPath 1.0 and first order logic: similar to XPath 2.0, it introduces quantified variables in XPath expressions. To preserve decidability, all variables must be constrained by predicates that are true for a finite set of elements only. It remains unclear if CLiX rules actually exceed the expressive power of XPath 1.0 from a theoretical perspective. However, the availability of quantified variables can help users acquainted with first order logic to express complex constraints in a more natural way than using a variable-free language such as XPath 1.0. Since XPath 2.0 also allows for the quantification of variables in XPath expressions, it is possible to represent CLiX rules quite similarly in XPath 2.0.

Example 8.2.1 (CLiX Rule)

The criterion ”every defined term must be used in the sequel of its definition” can be represented in CLiX as follows:

```
<clix:rule id="rule-1">
  <clix:forall var="dTerm" in="//definition/definedTerm">
    <clix:exists var="uTerm" in="$dTerm/../following::usedTerm">
      <clix:equal op1="$dTerm" op2="$uTerm"/>
    </clix:exists>
  </clix:forall>
</forall:rule>
```

A CLiX rule starts with the element `clix:forall` that defines the context of the rule. Each node of the context is bound to a variable (named `dTerm` in the sample rule above). The context of the sample rule is defined by the XPath expression `//definition/definedTerm`, which returns every `definedTerm` element within a `definition` element of the XML document.

Each node bound to `dTerm` is checked against the subsequent condition: there is some node `uTerm` in the node set returned by the XPath expression `$dTerm/../following::usedTerm`, such that the concatenated text nodes of `uTerm` are identical to the concatenated text nodes of `dTerm`.

For users acquainted with first order logic, the CLiX representation of the consistency criterion may be more readable than the corresponding Schematron rule in Example 8.1.1. For users experienced in XSLT programming, the Schematron style of consistency rules may be easier to get along with. □

Examples 8.1.1 and 8.2.1 illustrate that there is little difference in the general concept of and expressiveness of CLiX and Schematron. Both constraint languages support XPath with embedded XSLT functions, variables bound to finite sets of elements, and further XSLT constructs such as `key` definitions. However, since the definition and evaluation of Schematron schemas entirely base on open standards, Schematron is more open to future enhancements than CLiX that is bound to patented proprietary evaluation algorithms.

Comparison with $\mathcal{ALC}$CTL

It is shown in [Pil06] that $\mathcal{ALC}$CTL restricted to finite models is contained in first order logics interpreted over finite domains. Consequently, every criterion expressible in $\mathcal{ALC}$CTL can be represented by a CLiX rule. $\mathcal{ALC}$, the non-temporal part of $\mathcal{ALC}$CTL, is a two variable fragment of predicate logic, while CLiX expressions can make use of more than two variables. Hence, the expressiveness of CLiX exceeds the expressiveness of $\mathcal{ALC}$CTL for non-temporal conditions.

However, [Pil06] also demonstrates that the simulation of temporal conditions using first order logic leads to very complex expressions that are, in general, expensive to evaluate. As a result, $\mathcal{ALC}$CTL allows for the compact representation and efficient checking of criteria that are hard to represent and verify by methods based on first order logics such as CLiX.

Note further that $\mathcal{ALC}$CTL formulae are interpreted on vocabulary described by DL knowledge bases. DL reasoning is adopted for constructing the temporal verification model of a document. In contrast to CLiX rules, DL reasoning is not restricted to finite domains and yields also sound and complete results when reasoning about infinite domains. Hence, when adding ontologies, $\mathcal{ALC}$CTL-based verification is more powerful than CLiX for criteria referring to schema level knowledge about the concepts and relations of the document's domain of discourse.

It has to be added that CLiX offers a basic extension mechanism for introducing new self-defined operators. The evaluation of self-defined operators has to be implemented by dedicated algorithms within the runtime environment of CLiX. With such extensions, CLiX can be considered as Turing-complete, i.e. every decidable condition for XML documents can be expressed and evaluated.

An analogous instrument is also available within the $\mathcal{ALC}$CTL system. Recall that relevant knowledge about the document is extracted from the document and other information sources and represented as DL assertions within the semantic document model. Hence, properties that are not expressible in $\mathcal{ALC}$CTL can be computed by dedicated algorithms within the knowledge extraction process and represented within the semantic model. We call such properties *computed properties* (cf. Table 8.1).

As an example consider the following criterion: "eLearning documents should not contain any unfair tests." Whether a test is fair or unfair may depend on complex

conditions, the representation and evaluation of which are out of scope of $\mathcal{ALC}$CTL. Still, on an abstract level, the criterion can be represented by $\mathcal{ALC}$CTL as $\mathsf{AG}(Test \sqsubseteq \forall classifiedAs.Fair)$.

The tough work – the assessment of the fairness of tests as represented by role $classifiedAs$ – is delegated to the knowledge extraction process and the reference ontology, respectively.

8.3. Comparative Evaluation Based on XSLT and XPath

Within the subsequent section, the expressiveness and performance of $\mathcal{ALC}$CTL as compared to XSLT and XPath are evaluated.

8.3.1. Why XSLT and XPath?

The most relevant rule-based validation techniques for XML documents – Schematron and CLiX – highly rely on XPath and XSLT for the representation and evaluation of criteria. Schematron schemas are converted to XSL stylesheets and evaluated by standard XSLT processors. CLiX rules are very similar to Schematron rules but are validated by proprietary tools such as xlinkit. In preliminary case studies [Abs06], the performance of CLiX/xlinkit turned out to be rather lower than the performance of XSLT-based evaluation of Schematron.

Thus, the combination of XSLT and XPath can be considered as an upper bound regarding efficiency, expressiveness, and flexibility of XML-based document validation. As such, XSLT and XPath are a good benchmark for assessing the performance of $\mathcal{ALC}$CTL-based document verification.

Further arguments for choosing XSLT + XPath as a tough benchmark for $\mathcal{ALC}$CTL are:

- XPath is adequate for expressing complex path-related conditions on XML documents.
- XSLT is Turing-complete [Kep04]. Consequently, every condition that cannot directly be expressed in XPath can be checked by using advanced XSLT features such as recursive templates. The experiments in section 8.3.6 show that recursive templates are actually very helpful for checking criteria represented in $\mathcal{ALC}$CTL.
- highly optimized and mature processing engines are available for both XPath and XSLT. XSLT and XPath are routinely applied in many system environments.

Recently, the XSLT and XPath specifications version 2.0 have been released [W3C07d, W3C07b]. Version 2.0 adds a lot of comfort and functionality [Kay04]. It turned out, however, that within the scope of the subsequent case studies and experiments the differences between versions 2.0 and 1.0 of XSLT / XPath 1.0 are of little relevance. The implementation of the selected test cases in XSLT/XPath 2.0 does not differ from XSLT/XPath 1.0 apart from irrelevant syntactic variations. Version 1.0 of XSLT and XPath is still more widely adopted and supported by more processing tools. For instance, the Microsoft XSLT processor version 4.0, one of the most efficient XSLT processors, does not support version 2.0 of XSLT and XPath. As a consequence of the sufficient expressiveness, better tool support, and higher performance, we opted for using XSLT/XPath 1.0 in subsequent experiments.

8.3.2. Evaluation Goals

Within this section, we compare the developed verification techniques with standard XML validation techniques in the following aspects:

- *features*: which kind of features useful for representing consistency criteria are supported?
- *performance*: how do the runtimes of $\mathcal{ALC}$CTL-based document verification compare to XML-based techniques?

8.3.3. Comparing Features

The most important features of $\mathcal{ALC}$CTL -based document verification are:

- access to schema-level and background knowledge.

  The $\mathcal{ALC}$CTL framework provide means for adopting general background knowledge about the entities and relationships of the domain of discourse. This simplifies the specifications because background knowledge does not need to be encoded into the specifications. Background knowledge can be represented at the instance *and* schema level. General document and discourse models can be applied for checking complex content-related criteria. This increases the expressiveness of the overall approach.

- controlled specification terminology.

  Reference ontologies can be used to define the vocabulary for the representation of the document's content and for formalizing criteria by means of $\mathcal{ALC}$CTL. A defined, agreed-upon, common vocabulary greatly enhances the robustness against specification or representation errors (e.g. due to typos) and thus increases the reliability of verification results. Moreover, the vocabulary of specifications can be aligned by ontologies to different document and content models.

This allows for decoupling specifications from possible changing or heterogeneous document formats and supports re-use and adaption of specifications to different use cases.

- consistency of metadata.

  The decidability of DL allows for checking the *logical* consistency of metadata about the document. Faulty and contradictive metadata, which could reduce the reliability of verification results, can be discovered before evaluating specifications.

- implicit knowledge.

  By combining instance-level knowledge about the document's content and structure with schema-level knowledge about its domain of discourse, implicit knowledge can be derived by means of ontological reasoning. Implicit knowledge helps to reduce the complexity of specifications and of the knowledge extraction process. Further, additional errors may be detected based on derived document properties (cf. section 7.5.4).

- open world semantics.

  For deriving implicit knowledge from the DL representation of the document's content, the open world assumption is applied. Open world reasoning distinguishes if a property is not known to hold or if a property is known not to hold. As a consequence, a property is not assumed not to hold if it cannot be proven based on the available information.

  The open world assumption is adequate for reasoning about (possibly infinite) domains, the knowledge about which is incomplete. This is assumed, for instance, for the document's domain of discourse because a complete formal representation of the domain of discourse is, in general, impossible or too expensive.

  The decidability of the DL, which is used for the representation of knowledge about the document, enables sound and complete conclusions based on the partial knowledge about potentially infinite discourse domains. Drawn conclusions remain valid even if additional information becomes available (monotonic reasoning).

- closed world semantics.

  The closed world assumption is applied for the verification of narrative paths through the document by model checking techniques. I.e. if a property cannot be verified it is assumed not to hold. Hence, the $\mathcal{ALC}$CTL framework combines open and closed world reasoning within the verification process. This property distinguishes the presented approach not only from standard XML techniques but, to the best of our knowledge, also from all other approaches to document verification.

A closed world semantics is adequate if a complete representation of relevant structures can be assumed. Regarding the narrative structure of the document, such an assumption is valid because documents are finite, fully accessible data objects.

Closed world semantics is more intuitive for most users and allows - in the case of finite structures - for adopting highly efficient model checking techniques instead of expensive theorem proving as required for open world reasoning.

- negation as failure.

  This is negation under a closed world assumption: if a property can not be verified it is assumed not hold. Negation as failure is widely adopted and therefor also known as *default negation* [Jan01]. The connective "$\neg$" of $\mathcal{ALC}$CTL specifications and the `not` operator in XPath expressions have a "negation as failure" semantics.

- standard negation.

  The standard negation expresses that a property is asserted not to hold (in contrast to negation as failure that expresses the absence of a proof a property). The DL used for the knowledge representation of the document's content offers standard negation. Standard negation is not available in $\mathcal{ALC}$CTL specifications and in XPath. However, the reference ontology can define properties based on standard negation as atomic concepts that are used within $\mathcal{ALC}$CTL specifications.

- properties of narrative paths.

  Computation tree logics such as $\mathcal{ALC}$CTL are expressive for a wide range of path-related properties [BBF$^+$01]. Causal as well as temporal dependencies between properties of content units can both be addressed [DAC99, Jak06]. Causal dependencies are of the form: "when(ever) property p holds also property q must hold". Temporal interdependencies are of the form: "property p holds once/always before/after/until q". Finally, property p can be required to hold for some path or for all paths (satisfying a certain property q).

- structured local properties.

  In $\mathcal{ALC}$CTL, local properties are represented by the description logics $\mathcal{ALC}$ that is less expressive than standard description logics such as $\mathcal{SHIQ}$ or $\mathcal{SHOQ}(D)$. However, complex properties not expressible in $\mathcal{ALC}$ can be defined within the reference ontology of the $\mathcal{ALC}$CTL system and used as atomic concepts/roles within $\mathcal{ALC}$CTL specifications (cf. standard negation). Thus, in $\mathcal{ALC}$CTL, very expressive DL can indirectly be used for expressing local properties.

- coherence properties.

  $\mathcal{ALC}$CTL is expressive for coherence criteria requiring a combination of semantic and temporal relation between content objects (Definition 6.5.5).

The most important features of XSLT and XPath are:

- Turing-completeness.

  XSLT is Turing-complete [Kep04]. Any decidable criterion can be checked by XSLT. In contrast to $\mathcal{ALC}$CTL, the number of variables used within rules is not limited. The limitation of variables in $\mathcal{ALC}$CTL formulae is necessary to preserve decidability [BCM+03, HWZ01].

- traversing paths within the composition tree.

  XPath allows for traversing the document tree of an XML document along various axis such as descendant, ancestor, preceding, and following. This way, elements relative to a given context can be selected very flexibly.

- properties of narrative paths.

  Narrative paths, which do not follow any axis of the document tree, can be tracked by recursively resolving references within an XML document. This is supported in XSLT by recursive templates (see also 8.3.6). However, as shown in later case studies (section 8.3.4), the expressiveness of a computation tree logic is very expensive to realize by means of XSLT.

- structured local properties.

  XPath offers a rich language for representing structured local properties of content units, for instance, by constraining the values of certain attributes or subelements.

- coherence properties.

  Coherence properties can be represented by means of XPath as long as the temporal condition is relatively simple and evaluated w.r.t. the linear document order of elements (section 8.3.5). XPath and XSLT, however, do not match the flexibility and performance of $\mathcal{ALC}$CTL for coherence criteria on documents with a branching narrative structure.

- computed properties.

  Computed properties are not directly represented by XML data but require some sort of calculation based on the XML data. A simple computed property, for instance, is the total number of content units of the document. Being Turing-complete, XSLT can compute every decidable property. Also, XSLT offers a rich collection of built-in functions for efficient calculations.

- numeric constraints.

  $\mathcal{ALC}$CTL is not very expressive for numeric constraints. $\mathcal{ALC}$CTL cannot express directly, for instance, that a document should contain a certain percentage of easy, medium, and hard content. Numeric constraints can, in principal, be represented and checked by means of XSLT. For complex numeric constraints,

however, an XML-based approach reaches its limitations [Sch08] and cannot compete with dedicated constraint solving techniques such as linear programming [Dar91], constraint logic programming [MS98], or answer set programming [Bar03].

Aspect	**XPath + XSLT**	CTL	$\mathcal{ALC}$CTL
access to background and schema knowledge	limited (data types in 2.0)	–	+
controlled specification terminology	–	–	+
consistency of metadata	–	–	+
implicit knowledge	–	–	+
open world semantics	–	–	+
closed world semantics	+	+	+
negation as failure	+	+	+
standard negation	–	–	indirect (ontology)
Turing-completeness	+	-	-
maximal number of variables per condition	unlimited	0	2 (implicit)
traversing paths within the composition tree	++	-	limited (via $partOf$ role)
properties of narrative paths	limited	++	++
structured local properties	++	–	+
coherence properties	+	–	+
computed properties	+	–	indirect (knowledge extraction)
numeric constraints	+	–	indirect (ontology)

Table 8.1.: features of different methods for document verification

Tables 8.1 gives an overview of the relevant features of $\mathcal{ALC}$CTL-based document verification as compared to XSLT/XPath. For completeness, a column for CTL has been added because CTL is frequently applied to checking the consistency of hypertext documents (section 2.4.2).

In summary, XSLT + XPath is the more powerful and flexible tool for checking the XML representation of a document, whereas $\mathcal{ALC}$CTL is targeted at the verification of the document's content and narrative structure independent of its implementation.

8.3.4. Comparing Performance - General Setting

In the subsequent experiments, we examine the performance of XSLT-based consistency checking as compared to $\mathcal{ALC}$CTL model checking.

Since XSLT is Turing-complete, all criteria expressible in $\mathcal{ALC}$CTL can also be checked based on XSLT. However, we demonstrate that from a certain level of complexity onwards the XSLT-based solution is not efficient neither in realization cost nor in performance.

We start the comparison by assuming the best case scenario for XSLT processing: linearly structured documents and criteria of limited complexity. Subsequently, we evaluate how XSLT processing performs on documents with a branching narrative structure.

The following XSLT processors are adopted:

- Microsoft XSLT processor MSXML 4.0 SP2 [Liv02]
- Xalan-Java version 2.7.0 [Xal07]

In preliminary evaluations, MSXML turned out to be the best performing XSLT processor within the given setting. However, MSXML is only available for Windows environments. The Java version of Xalan has been chosen as a state-of-the-art XSLT processor that is platform independent and used in many software projects dealing with XML documents. In addition, a Java-based implementation of XSLT processing enables a fair comparison of the runtime results because the $\mathcal{ALC}$CTL model checker is also implemented in Java.

Test Cases

We choose the SCORM-based XML documents and criteria of the $\mathcal{ALC}$CTL benchmarks of Experiment 1 (section 7.5.2) as test cases for the subsequent experiments.

Using XSLT, the criteria could be checked directly on the XML sources of the test documents. This, however, would be a very difficult task because each test document is not a single XML file but is implemented in terms of a large set of interrelated XML files of different types and contents (cf. section 7.4). Consider, in addition, that $\mathcal{ALC}$CTL formulae are evaluated w.r.t. a comparably small semantic model about the document that integrates all relevant information from various sources and abstracts from many irrelevant details.

For a fair comparison, also XSLT-based checking of criteria is conducted on preprocessed XML documents: relevant metadata and structural information is extracted by the same knowledge extraction methods as used in $\mathcal{ALC}$CTL system and represented in a single, integrated XML file that has a much smaller size and simpler structure than the original XML documents of the test case.

Criteria expressed in temporal logics are evaluated w.r.t. paths within the narrative graph of a document, which is mapped onto a state transition system. The representation of the narrative structure of a document by means of XML is not straight forward. This is because the XML data model bases on linearly ordered trees while the narrative graph of a document is neither linearly ordered nor is it a tree (cf. section 5.2).

There are two basic approaches for representing the narrative structure of a document by means of XML:

- *flat structure*: the nodes (i.e. content units) of the narrative graph are represented by a flat list of XML elements each carrying a unique ID. The vertices (i.e. narrative relations) between content units are represented as references between the elements representing nodes.
- *deep structure*: each direct successor of a content unit $U \in CU_d$ is represented as a child of the element representing U. The unique starting unit of the document becomes the root element of the XML representation and all direct and indirect successors are ancestors of the root element. The set of narrative paths of the document is represented by the set of paths from the root element along the child axis of the XML structure.

Example 8.3.1 (Flat vs. Deep XML representation of Narrative Structure)

Consider the narrative structure $NS_d = (CU_d, BOD_d, proceed)$ where

$$\begin{aligned} CU_d &= \{s_0, s_1, s_2\} \\ BOD_d &= s_0 \\ proceed &= \{(s_0, s_1), (s_0, s_2), (s_1, s_2), (s_2, s_2)\} \end{aligned}$$

The following XML fragment illustrates a flat representation of NS_d:

```
<contentUnit name="s0">
   <successor name="s1"/>
   <successor name="s2"/>
</contentUnit>

<contentUnit name="s1">
   <successor name="s2"/>
</contentUnit>

<contentUnit name="s2">
   <successor name="s2"/>
</contentUnit>
```

The following XML fragment illustrates a deep representation of NS_d:

```
<contentUnit name="s0">
   <contentUnit name="s1">
       <contentUnit name="s2"/>
   </contentUnit>
   <contentUnit name="s2"/>
</contentUnit>
```

□

The flat representation requires the iterative resolution of references for tracking paths within the document. The deep representation translates the narrative graph into a tree that can efficiently be represented and processed by XML methods without resolving references.

At a first glance, the deep structured representation seems to be the more adequate approach. However, an unfolding of the narrative graph can lead to trees of infinite depth and breadth because, in general, there are infinitely many narrative paths each of them having an infinite length. Even when restricting the narrative graph to directed acyclic graphs, a tree unfolding can grow exponentially in the number of content units. Hence, the sizes of deep structured XML representations may explode for larger documents.

Table 8.2 shows the document size of deep vs. flat structured representations of the narrative document structure for the test cases (D1 - 6) (section 7.5.2).

	D1	D2	D3	D4	D5	D6
# content units	32	64	96	128	160	192
# narrative relations	35	70	105	140	175	210
# narrative paths	4	16	64	256	1024	4096
flat repr. size in kByte	37	74	110	148	185	223
deep repr. size in kByte	75	374	1580	6444	26066	105202

Table 8.2.: size of flat vs. deep XML representation of narrative document graphs

The size of the deep structured XML representation grows exponentially in the number of content units. This renders the checking of criteria inefficient or even impossible for larger documents.

As a consequence, we opted for the flat representation of the narrative graph. The narrative structure and the extracted facts about each of the content units are represented as a single, compact XML document (Listing 8.1).

```
0 <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

  <document source="manifest_32.xml">

    <content>
```

```
    ...
    <contentUnit name="ITEM_a14">
      <successor name="ITEM_a19" />
      <successor name="ITEM_a15" />
      <successor name="ITEM_a18" />
   ...
      <related name="hasDifficulty">
        <pair leftObject="ITEM_a14" rightObject="difficult" />
      </related>
   ...
      <related name="hasType">
        <pair leftObject="ITEM_a14" rightObject="information" />
      </related>
   ...
    </contentUnit>

    <contentUnit name="ITEM_a15">
      <successor name="ITEM_a16" />
   ...
      <property name="Excursion">
        <object name="ITEM_a15" />
      </property>
      <related name="hasDifficulty">
        <pair leftObject="ITEM_a15" rightObject="easy" />
      </related>
   ...
    </contentUnit>

    <contentUnit name="ITEM_a16">
      <successor name="ITEM_a17" />
   ...
    </contentUnit>
   ...
  </content>


</document>
```

Listing 8.1: fragment of the flat XML representation of the semantic document model

Listing 8.1 shows a fragment of the flat XML representation of the semantic document model. It consists of a sequence of `contentUnit` elements representing the set of content units CU_d of the document. `contentUnit` contains one or more `successor` elements representing the successor units according to the *proceed* relation of the narrative structure of the document. In addition, `contentUnit` may contain `property` elements and `related` elements. `property` elements represent concepts while `related` elements represent roles of the local fact bases of a content unit. `property` elements contain one or more `object` elements that represent the instances of the respective concept. Analogously, `pair` elements represent the role fillers of the respective role.

Listing 8.1 illustrates the XML-representation of 3 content units with identifiers `ITEM_a14`, `ITEM_a15`, and `ITEM_a16`.

Let $U_{15} \in CU_d$ be the content unit corresponding with `ITEM_a15` and $U_{16} \in CU_d$ be the content unit corresponding with `ITEM_a16`. Let, as usual, $proceed \subseteq CU_d \times CU_d$ denote the set of narrative relations across content units and $AB_{U_{15}} \in \mathbf{lAB}_d$ be the local fact base of content unit U_{15}.

Then the XML fragment of Listing 8.1 represents the following information about content unit U_{15}:

$$\begin{aligned} (U_{15}, U_{16}) &\in proceed \\ Excursion(U_{15}) &\in AB_{U_{15}} \\ \textit{hasDifficulty}(U_{15}, easy) &\in AB_{U_{15}} \end{aligned}$$

The document order of `contentUnit` elements within the XML representation of the semantic model is equal to the document order of the respective items within the original XML files of the document. This enables efficient checking of criteria w.r.t. the document order of content units because no successor references need to be resolved.

8.3.5. Experiment 1 - Performance on Linear Documents

In Experiment 1, the performance of XSLT-based verification for entirely linear documents is determined and compared to $\mathcal{ALC}$CTL model checking. In the linear case, the narrative structure of content units coincides with the (linear) document order of XML elements. This simplifies the formalization of criteria using XPath because XPath allows for directly evaluating conditions along the document order of elements.

Description of the Test Case

As a test case consider the following coherence criterion:

"Every lesson must be addressed by some test question in the sequel."

This property is formalized in $\mathcal{ALC}$CTL as

$$\mathsf{AG}(\exists contains.\exists hasType.Information \sqsubseteq \mathsf{AF}\ \exists referredBy.Testquestion) \tag{8.1}$$

(cf. formula f_4' in section 7.4).

As for the XSLT-based formalization, we disregard alternative narrative paths within the test documents, i.e. the "all paths" conditions of the modal operators AG and AF in Equation (8.1) are dropped. Instead, the XSLT-based version evaluates the criterion

w.r.t. the linear document order of content units only, which significantly simplifies the verification task.

An XPath-based representation of the simplified criterion can be constructed along the following lines:

- First, the set of files containing some information unit is determined, which are represented by the $\mathcal{ALC}$CTL concept $\exists contains.\exists hasType.Information$. We call these objects *initial elements* of the coherence condition. The names of initial elements can be determined by the following XPath query:

  ```
  //contentUnit/related[@name='isInFile']/pair/@rightObject
  [../@leftObject = ../../../related[@name='hasType']/pair
  [@rightObject='information']/@leftObject]
  ```

  Recall that $contains$ is the inverse of role $isInFile$ that is represented by `<related name='isInFile'>` in the XML representation of the semantic model. The sub-expression

  ```
  related[@name='isInFile']/pair/@rightObject
  ```

  returns the "right-hand side" fillers of role $isInFile$, i.e. the file names of lessons. Let n be such a file name. Then in the context of n, `../@leftObject` returns the "left-hand side" filler of n in role $isInFile$, which is the name l_n of the lesson contained in file n. In the context of a file name n, the filter expression

  ```
  [../@leftObject = ../../../related[@name='hasType']/pair
  [@rightObject='information']/@leftObject]
  ```

  evaluates to true iff the respective lesson name l_n (`../@leftObject`) has a "right-hand side" role filler in role $hasType$ (`= ../../../related [@name='hasType']/pair[...]/@leftObject`) that is an information (`[@rightObject='information']`).

- For each initial element, the "coherence condition" represented by the $\mathcal{ALC}$CTL concept AF $\exists referredBy.Testquestion$ is checked. Recall that $referredBy$ is the inverse of role $isRelatedToFile$ (Equation (7.2)). Let i be an initial element. Then there must be some object o that is a test question and is related to i via relation `isRelatedToFile` within the same or a following content unit.

  This condition can be represented by the following XPath expression evaluated in the context of an initial element i.

  ```
  . = (../../../related[@name='relatedToFile']
      /pair[@leftObject = ../../property
      [@name='Testquestion']/object/@name] |
      following::contentUnit/related[@name='relatedToFile']
      /pair[@leftObject = ../../property
      [@name='Testquestion']/object/@name])
      /@rightObject
  ```

The expression above evaluates to true in the context of a file name n, iff n appears in the right-hand side of role $relatedToFile$ in the current or some following content unit
(`. = (../../../related[@name='relatedToFile']/pair[...] | following::contentUnit/related[@name='relatedToFile']/pair[...])/@rightObject`)
and the respective left-hand side partner of n w.r.t. role $relatedToFile$ has the property "test question" (`[@leftObject = ../../property [@name='Testquestion']/object/@name]`).

- In total, each initial element selected by the first XPath expression must satisfy the "coherence" condition as represented by the second XPath expression. In other words, there must not be some initial element that does not satisfy the coherence condition.

  This can be checked by the following XPath expression:

```
not(
//contentUnit/related[@name='isInFile']/pair/@rightObject
[../@leftObject = ../../../related[@name='hasType']/pair
[@rightObject='information']/@leftObject]
[not(. = (../../../related[@name='relatedToFile']
    /pair[@leftObject = ../../property
    [@name='Testquestion']/object/@name] |
    following::contentUnit/related[@name='relatedToFile']
    /pair [@leftObject = ../../property
    [@name='Testquestion']/object/@name])
    /@rightObject
)])
```

For constructing a meaningful error report, the list of initial elements that violate the "coherence condition" are determined using the `for-each` construct of XSLT:

```
<xsl:for-each select="//contentUnit/related
        [@name='isInFile']/pair/@rightObject
        [../@leftObject = ../../../related[@name='hasType']
        /pair[@rightObject='information']/@leftObject]
        [not(. = (../../../
        related[@name='relatedToFile']/pair
        [@leftObject = ../../property[@name='Testquestion']
        /object/@name] | following::contentUnit/related
        [@name='relatedToFile']/pair[@leftObject =
        ../../property[@name='Testquestion']
        /object/@name])/@rightObject)]">

        Untested lesson <xsl:value-of select="."/>
        in content unit
        <xsl:value-of select="../../../@name"/>;
</xsl:for-each>
```

This outputs the names and locations of all initial elements that violate the coherence condition.

Remark 8.3.2 (XPath Representation of Coherence Criteria)

The relatively simple sample criterion of this experiment is already sufficient to demonstrate that criteria expressible in $\mathcal{ALC}$CTL are difficult to represent by XPath even when restricting the XPath-based formalization to entirely linear structures. In the case of branching structures, the emulation of $\mathcal{ALC}$CTL formulae by means of XSLT and XPath becomes even more cumbersome (cf. subsequent experiments). The much lower complexity of $\mathcal{ALC}$CTL-based formalizations as compared to XML methods results in lower formalization and maintenance cost, higher robustness against specification errors, easier adaptability towards different specification problems and document formats, and, last but not least, in a higher performance as shown subsequently. □

Experiment 1 - Evaluation Hypothesis

The specification of this experiment is represented by a single variable-free XPath expression. In preliminary experiments, MSXML has proven extremely efficient for evaluating variable-free XPath expressions. Hence, we expect a good performance of MSXML for the given task.

The evaluated XPath expression is quite complex. The number of initial elements to evaluate increases linearly with the document size. For each initial element i, the number of elements to consider for checking of the "coherence condition" grows linear in the document size. Hence, we expect a quadratic scaling of runtime in the document size.

In the average case, $\mathcal{ALC}$CTL model checking has a cubic scaling of runtime in the document size (Proposition 6.4.12). In practical experiments of section 7.5, $\mathcal{ALC}$CTL model checking displayed a linear to quadratic scaling of runtime. Hence, it is hard to predict if $\mathcal{ALC}$CTL model checking performs better or worse than the XSLT-based solution.

Experiment 1 - Runtime Results

Figure 8.1 displays the runtimes of the XSLT processors Xalan and MSXML as compared to $\mathcal{ALC}$CTL model checking for the series of very large documents (VLD1 - 10) of Experiment 1 (section 7.5.2). The runtimes of XSLT processing and $\mathcal{ALC}$CTL model checking do not include the pre-processing time for constructing the verification model but include the time for generating an error report. For better visibility, the values of the $\mathcal{ALC}$CTL model checker are multiplied by 10 in Figure 8.1.

The scaling of XSLT processing in the document size is - as expected - super linear in the document size. The Microsoft processor performs 7 to 8 times as fast as the

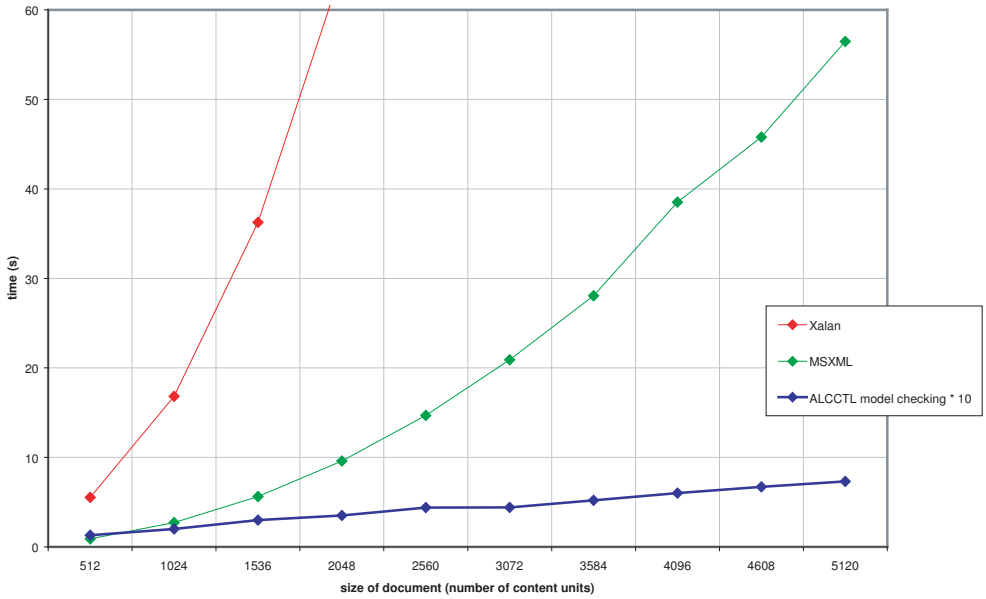

Figure 8.1.: runtime results of XML Experiment 1

Xalan processor. The absolute runtime of MSXML ranges between 0.9 seconds (512 content units) and 56.5 seconds (5120 content units). The largest document being checked within 10 seconds is 2048 content units. The runtimes of Xalan range between 5.5 seconds (512 content units) and 439.6 seconds (5120 content units). The largest document being checked within 10 seconds is 512 content units.

The runtimes of $\mathcal{ALC}$CTL model checking (including error report) range between 0.13 seconds (512 content units) and 0.73 seconds (5120 content units) and scale approximately linearly in the document size.

$\mathcal{ALC}$CTL model checking is between 7 (512 content units) and 77 (5120 content units) times as fast as MSXML for the given verification task although the $\mathcal{ALC}$CTL model checker verifies the sample criterion for *all* paths through the document while the XSLT-based solution is restricted to the linear document order of content units. More surprising than the absolute runtime values is the better scaling of $\mathcal{ALC}$CTL model checking in the document size as compared to XSLT processing.

Experiment 1 - Interpretation of Results and Conclusions

Criteria expressible in $\mathcal{ALC}$CTL can be represented in XPath when restricting them to the linear document order of elements. However, the resulting XPath expressions

are complex, difficult to built, and expensive to evaluate. XSLT processing is sufficiently efficient for checking even large documents. Still, $\mathcal{ALC}$CTL model checking outperforms XSLT processing clearly and scales better in the size of the document.

8.3.6. Experiment 2 - Performance on Non-Linear Documents

In the subsequent experiments, we evaluate the performance of XSLT processing for conditions evaluated along narrative paths. Evaluating criteria along paths in a state transition system is a complex task, in general. To simplify the complexity of the scenario, the following assumptions are applied:

- the narrative structure is an acyclic directed graph.
- all narrative paths start from the first content unit w.r.t. the document order of content units within the XML document.
- all narrative paths end at the last content unit w.r.t. the document order content units within the XML document.

These conditions ensure that there are finitely many narrative paths through the document each of them having a finite length. In general, narrative structures contain infinitely many different narrative paths each of them having infinite lengths. As a consequence, the XSLT-based algorithms presented in the subsequent experiments are not generally applicable but restricted to documents of limited structural complexity.

8.3.6.1. Experiment 2a - Tracking a Single Path in Acyclic Narrative Graphs

Description of the Test Case

In this experiment, we evaluate the performance of XSLT processing as compared to $\mathcal{ALC}$CTL model checking for criteria, the checking of which requires to track just a single narrative path through the document. This is the best case scenario for path-based conditions. In general, more than one or even all paths through the document need to be considered for determining the validity of an $\mathcal{ALC}$CTL formula.

As a test case, consider the following simple criterion:

"There should be path through the document without hitting any additional material."

This criterion can be expressed by $\mathcal{ALC}$CTL as follows:

$$\mathsf{EG}\ (Excursion \sqsubseteq \bot)$$

Recall, $Excursion$ is a concept representing the set of content fragments that contain additional information.

The test criterion is true within documents (VLD1 - 10) iff the path condition "no Excursion" – represented by the $\mathcal{ALC}$CTL sub-formula $Excursion \sqsubseteq \bot$ – holds on each content unit of some path from the first content unit to the final content unit. I.e. the criterion is satisfied iff the last unit can be reached from the first unit by iteratively pursuing narrative relations that lead to content units without any excursion.

Thus, verifying the criterion $\mathsf{EG}\ (Excursion \sqsubseteq \bot)$ can – within the scope of this experiment – be modelled as a reachability problem under a certain condition applied to the edges to follow. Checking "conditional reachability" within the narrative graph is not straight forward in XSLT for two reasons:

- XPath expressions can select elements just along a number of given axis. Narrative paths, however, do not necessarily follow any of the given axis of XPath such as `child`, `parent`, `ancestor`, `descendent`, `preceding`, `following`, ... Instead, tracking narrative paths requires the iterative resolution of references within the XML document.
- XPath can express predicates (so called node tests) for selecting nodes from a given sequence of elements but does not provide means for applying arbitrary conditions to the paths to be followed. All axis expressions are unconditional [W3C99a, W3C07b].

As a result, XPath alone is not sufficient for representing path-related criteria such as the test criterion. Instead, the path criterion must be checked iteratively. Iterations in XSLT can be realized by recursive templates. A recursive formulation of the given conditional reachability problem can be obtained by using the following equivalences (cf. [HR04]):

$$\begin{aligned} \mathsf{EG}(Excursion \sqsubseteq \bot) &\equiv (Excursion \sqsubseteq \bot) \wedge \mathsf{EX}\ \mathsf{EG}(Excursion \sqsubseteq \bot) \\ \mathsf{EX}\ \mathsf{EG}(Excursion \sqsubseteq \bot) &\equiv \mathsf{EX}((Excursion \sqsubseteq \bot) \wedge \mathsf{EX}\ \mathsf{EG}(Excursion \sqsubseteq \bot)) \end{aligned}$$

These equivalences are the basis of the following inductive definitions:

- if the first content unit contains an excursion, $\mathsf{EG}\ (Excursion \sqsubseteq \bot)$ is not satisfied. The absence of an excursion within the first content unit can be checked by the following XPath expression:

  ```
  //contentUnit[1][not(property[@name='Excursion']/object)]
  ```

- if the first content unit does not contain an excursion, then it satisfies $\mathsf{EG}\ (Excursion \sqsubseteq \bot)$ iff it satisfies $\mathsf{EX}\ \mathsf{EG}\ (Excursion \sqsubseteq \bot)$.
- the last content unit satisfies $\mathsf{EX}\ \mathsf{EG}\ (Excursion \sqsubseteq \bot)$ by assumption.

  The XPath expression `not(following-sibling::contentUnit)` checks if the current content unit is the last content unit of the document.

- any other content unit satisfies $\mathsf{EX}\,\mathsf{EG}\ (Excursion \sqsubseteq \bot)$ iff there is some successor unit that does not contain an excursion and satisfies $\mathsf{EX}\,\mathsf{EG}\ (Excursion \sqsubseteq \bot)$.

  In the context of a `contentUnit` element, the names of successor units can be determined by the XPath expression `current()/successor/@name`. The following XPath expression evaluated in the context of a `contentUnit` element U returns the sequence of successor units of U, which do not contain an excursion:

  ```
  //contentUnit[@name=current()/successor/@name]
                [not(property[@name='Excursion']/object)]
  ```

  Note that in the Xpath expression above the filter expression `@name=current()/successor/@name` evaluates to true if there is *some* node n in `current()/successor/@name` such that attribute `@name` is equal to n.

  The inductive definition of $\mathsf{EX}\,\mathsf{EG}\ (Excursion \sqsubseteq \bot)$ is checked by a recursive XSL template `checkEXEG` as shown in the code fragment below.

The definitions above lead to following XSL code fragment for checking $\mathsf{EG}(Exursion \sqsubseteq \bot)$:

```
<xsl:for-each select="//contentUnit[1][not(property
        [@name='Excursion']/object)]">
    <xsl:call-template name="checkEXEG"/>
</xsl:for-each>

<xsl:template name="checkEXEG">
  <xsl:choose>
    <xsl:when test="following-sibling::contentUnit">
      <xsl:for-each select="
            //contentUnit[@name=current()/successor/@name]
            [not(property[@name='Excursion']/object)]">
        <xsl:call-template name="checkEXEG"/>
      </xsl:for-each>
    </xsl:when>
    <xsl:otherwise>
        true
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

The XSL fragment outputs `true` at least once iff $\mathsf{EG}(Excursion \sqsubseteq \bot)$ is satisfied by the first content unit (`//contentUnit[1]`) of the document.

The recursive template `checkEXEG` successively selects `contentUnit` elements by their unique name attribute. This can be sped up by using the `xsl:key` construct.

```
<xsl:key name="CUIndex" match="contentUnit" use="@name"/>
```

This instructs the XSLT processor to create an access index named `CUIndex` for element `contentUnit` using key `@name`.

Now `contentUnit` elements can be accessed efficiently by name using the XSLT function `key`. The indexed access to `contentUnit` is realized by modifying the for-loop of template `checkEXEG` to:

```
<xsl:for-each select="key('CUIndex',successor/@name)
                [not(property[@name='Excursion']/object)]">
    <xsl:call-template name="checkEXEG"/>
</xsl:for-each>
```

In subsequent runtime evaluations, we tested both the indexed and non-indexed access to `contentUnit` elements.

Note that the given XSLT formalization is sufficient within the scope of the experiment but not generally valid. For instance, the stylesheet does not terminate in the case of cycles within the narrative graph because it does not check for already visited content units.

Moreover, the code fragment outputs `true` several times if there are more than one path satisfying the property $\mathsf{G}(Excursion \sqsubseteq \bot)$. This is because the template `checkEXEG` performs an exhaustive search for all paths on which $Excursion \sqsubseteq \bot$ globally holds. An exhaustive search of paths is not tractable in general because the number of paths in acyclic directed graphs can grow exponentially in the number of vertices. Within the scope of this experiment, an exponential blow up of tracked paths does not arise because each of the test documents (VLD1 - VLD10) contains just a single path satisfying the condition $\mathsf{G}(Excursion \sqsubseteq \bot)$.

In total, the test case and the given XSLT implementation are designed to represent a best case scenario for XSLT-based checking of path-related criteria.

Experiment 2a - Evaluation Hypothesis

For the non-indexed version, we suspect a quadratic growth of runtime in the document size because the number of content units, which are "visited" by the recursive template `checkEXEG`, grow linearly in the document size and a linear growth of the time required for selecting the successor units of each content unit within the `checkEXEG` template can be assumed.

An access index to `contentUnit` elements should result in a linear scaling of runtime because each content unit is visited at most once and the selection of successor units should be possible in constant time using an efficient index structure for `contentUnit` elements.

For $\mathcal{ALC}$CTL, we expect a linear scaling of runtime as in Experiment 1 of $\mathcal{ALC}$CTL benchmarking (section 7.5.2).

Experiment 2a - Test Results

Figure 8.2 shows the runtime results of the XSL stylesheet without using an index on `contentUnit` elements.

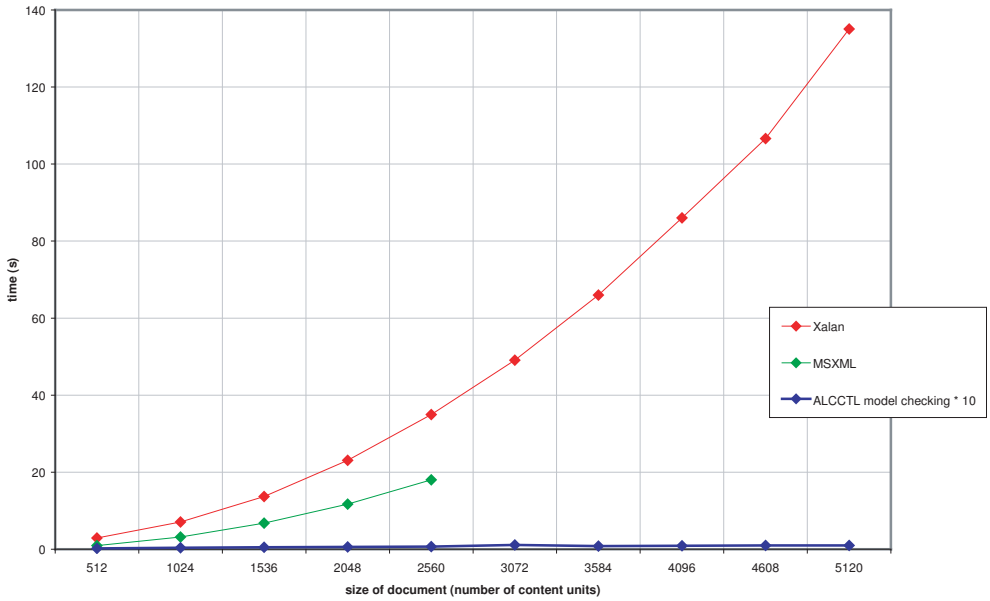

Figure 8.2.: runtime results of Experiment 2a - no access index

As expected, the runtimes of MSXML and Xalan scale super-linearly within the given scenario. The Microsoft processors fails to check documents larger than 2560 content units because of a ”stack overflow”. The Xalan processor can handle all documents increasing the stack size of the Java Virtual Machine to 4 MByte (standard is 256 kByte).

MSXML performes about twice as fast as Xalan on each of the test cases (VLD1 - 5). Interestingly, the same test scenario executed on a laptop computer led to a clearly better performance of Xalan as compared to MSXML. Obviously, the code of MSXML is highly optimized for desktop environments.

The absolute values of the Microsoft processor range between 1.0 seconds (512 content units) and 18.0 seconds (2560 content units) while Xalan takes 2.9 seconds for 512 content units, 35.0 seconds for 2560 content units, and 135.1 seconds for 5120 content units.

The runtime values of $\mathcal{ALC}$CTL model checking, although magnified by factor 10, are hardly visible in Figure 8.2. The model checking times do not exceed 0.11 seconds for any of the given test cases.

Figure 8.3 shows the runtime results when using an access index to `contentUnit` elements.

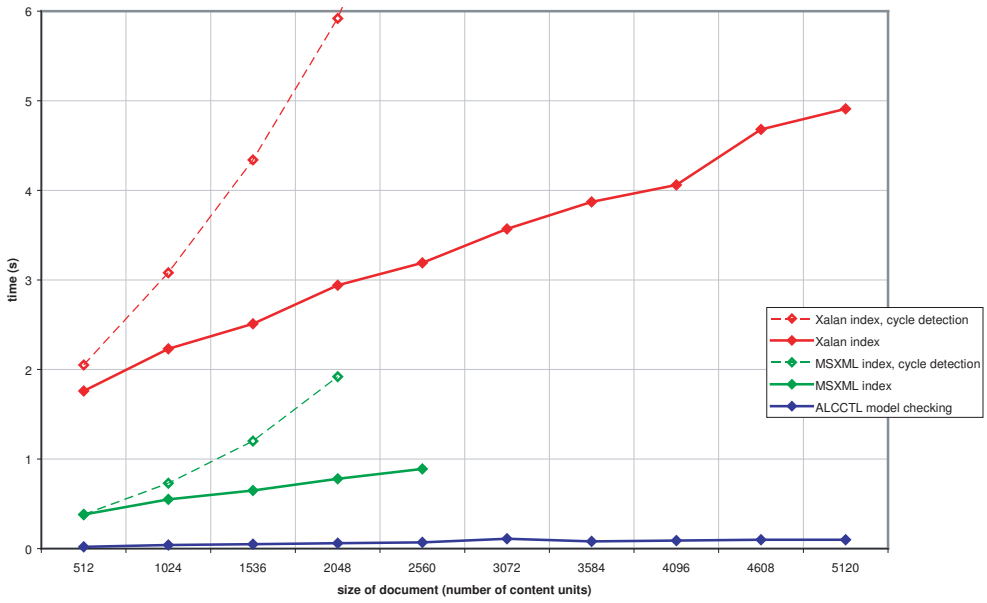

Figure 8.3.: runtime results of Experiment 2a - applying an access index

The index significantly speeds up the execution of the stylesheet. Both processors achieve the optimal linear scaling of runtime. The runtime of Xalan exceeds the runtime of MSXML by a factor of 3.5 to 4. In contrast to Xalan, MSXML again fails to check documents larger than 2560 content units because of a stack overflow. The runtimes of MSXML range from 0.4 seconds for 512 content units to 0.9 seconds for 2560 content units. Xalan requires 1.8 seconds for 512 content units, 3.2 seconds for 2560 content units, and 4.9 seconds for 5120 content units.

Note that the good performance of XSLT processing can only be achieved under the specific simplifying conditions of this experiment: just one narrative path has to be tracked and the narrative structure is free of cycles. Figure 8.3 includes the runtime results of XSLT processing when cycle detection is added to the stylesheet (dotted graphs in Figure 8.3). Checking for cycles significantly slows down the runtime of the stylesheet and leads to a super-linear scaling of runtime. In addition, the Microsoft processor runs into a stack overflow already for documents larger than 2048 content units.

$\mathcal{ALC}$CTL model checking outperforms both the non-indexed and indexed XSLT-based solutions clearly. A document of 2560 content units can be checked about 13 times as

fast as MSXML and about 46 times as fast as Xalan on the stylesheet that applies an indexed access to content units and does not check for cycles.

Experiment 2a - Interpretation of Results and Conclusions

An XSLT-based formalization of criteria related to narrative paths is feasible if many simplifying conditions (simple conditions, no cycles, few alternative paths) are met. The XSL feature of access indices enables efficient tracking of single narrative paths within the document. Still, XSLT cannot match the performance of $\mathcal{ALC}$CTL model checking. For the general case, (cyclic narrative structure, many alternative paths, complex path-related criteria) checking properties of narrative paths in XML documents by means of XSLT can be expected to result in very complex and inefficient code.

In contrast, model checking is not only efficient in easy scenarios but also scales to complex cases containing many alternative paths and complex path-related conditions (see benchmarks in section 7.5).

8.3.6.2. Experiment 2b - Tracking All Paths in Acyclic Narrative Graphs

Description of the Test Case

In this experiment, we examine the performance of XSLT processing when all narrative paths need to be considered for the evaluation of a criterion.

As a test case consider the following criterion:

"On all paths eventually the final test results will be presented."

This criterion can be expressed in $\mathcal{ALC}$CTL as

$$\mathsf{AF}\ \mathsf{AG}\ \neg(Testresult \sqsubseteq \bot)$$

"On all paths eventually a state is reached, from which onwards the set of test results will never be empty."

The sub-formula $\mathsf{AG}\ \neg(Testresult \sqsubseteq \bot)$ characterizes the set of content units containing *final* test results (end unit property, cf. Definition 5.2.9).

The XSL representation of the criterion is simplified by making use of background knowledge about test cases (VLD1 - 10). Here, the final test result is always presented in the last content unit. Then the given criterion reduces to checking the reachability of the last content unit on all narrative paths starting from the first content unit.

We can check this condition by a slight modification of the XSL code of Experiment 2a. If the path condition $\mathsf{G}(Excursion \sqsubseteq \bot)$ within the path tracking template `checkEXEG` is dropped, all paths from the first to the last content unit are tracked. The XSLT realization of $\mathsf{AF}\ \mathsf{AG}\ \neg(Testresult \sqsubseteq \bot)$ is:

```
<xsl:key name="CUIndex" match="contentUnit" use="@name"/>

<xsl:for-each select="//contentUnit[1]">
    <xsl:call-template name="checkAF"/>
</xsl:for-each>

<xsl:template name="checkAF">
  <xsl:choose>
    <xsl:when test="following-sibling::contentUnit">
      <xsl:for-each select="key('CUIndex',successor/@name)">
        <xsl:call-template name="checkAF"/>
      </xsl:for-each>
    </xsl:when>
    <xsl:otherwise>
        true
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

This stylesheet fragment outputs true for each path within the narrative graph that starts from the first content unit and reaches the last content unit of the XML document.

The validity of AF AG $\neg(Testresult \sqsubseteq \bot)$ can be evaluated for the given series of documents by comparing the output size with the total number of narrative paths from the first to the last content unit. The number of such narrative paths within the test cases (D1 - 11) calculates as follows:

document	# content units	# narrative paths
D1	32	$4^1 = 4$
D2	64	$4^2 = 16$
D3	96	$4^3 = 64$
D4	128	$4^4 = 256$
D5	160	$4^5 = 1024$
D6	192	$4^6 = 4096$
D7	224	$4^7 = 16384$
D8	256	$4^8 = 65536$
D9	288	$4^9 = 262144$
D10	320	$4^{10} = 1048576$
D11	352	$4^{11} = 4194304$

Remark 8.3.3 (Checking All Path Conditions)

In principle, it is possible to check ”all paths” conditions more efficiently than by simply tracking all paths from a given start node. The marking of already visited nodes and dynamic programming [CLRS01] can prevent an exponential blow-up of runtime. These methods, however, are clearly beyond the scope of XSLT programming because of the missing support of complex data types and updates to the values of variables.

Note that the given criterion can be reformulated to a criterion of the same type as in Experiment 2a by using the equivalence $\mathsf{AF}\ p \equiv \neg\mathsf{EG}\ \neg p$ [HR04]. The evaluation results of Experiment 2a suggest that EG type of criteria are easier to check by means of XSLT than AF type of criteria. This is not the case because also EG type of criteria require to consider all paths, in general. For the criterion of this experiment, the mapping onto an EG type of expression is not helpful for finding a more efficient XSLT realization. □

Experiment 2b - Evaluation Hypothesis

The stylesheet for checking $\mathsf{AF}\,\mathsf{AG}\,\neg(Testresult \sqsubseteq \bot)$ conducts an exhaustive search of all narrative paths from the first to the last content unit. Since the number of such narrative paths grows exponentially in the document size, the runtime of the stylesheet is expected to grow exponentially.

In contrast, $\mathcal{ALC}$CTL model checking is polynomial for all types of expressions (Proposition 6.3.14).

Experiment 2b - Test Results

Figure 8.4 shows the evaluation results.

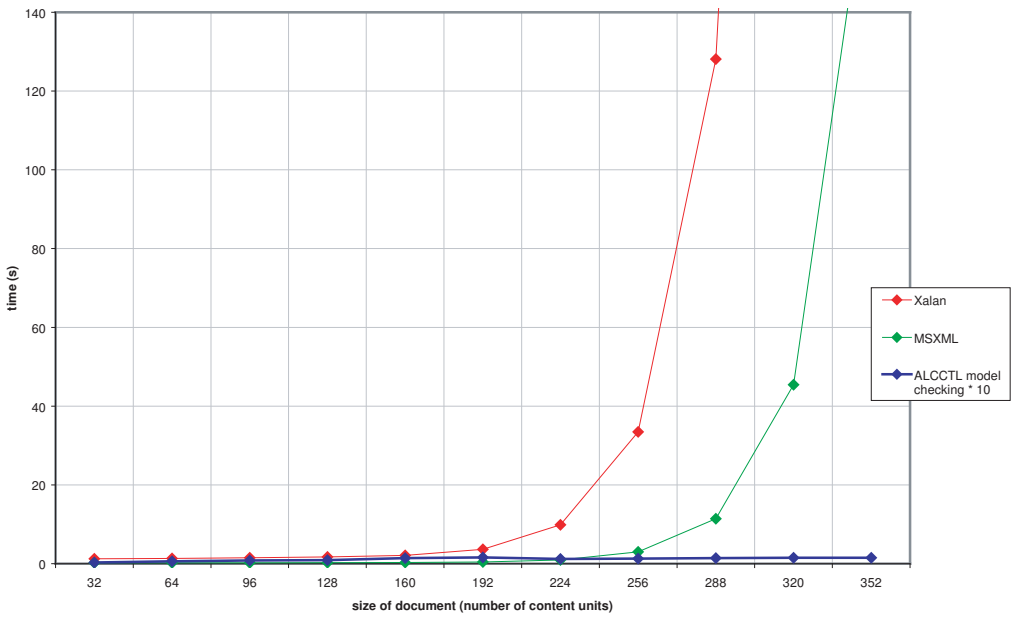

Figure 8.4.: runtime results of Experiment 2b

As expected, the runtime of tracking all paths within the narrative structures of the test documents explodes from a certain document size on. The largest document being processed within 10 seconds is 256 content units in case of MSXML and 224 content units in case of Xalan. A document of 320 content units requires already 45.4 seconds on the Microsoft processor and more than 8 minutes on Xalan.

In contrast, the runtime of $\mathcal{ALC}$CTL model checking does not exceed 0.15 seconds even for documents as large as 5120 content units.

Experiment 2b - Interpretation of Results and Conclusions

A naive approach to checking properties of narrative paths through the document is only sufficiently efficient for small documents. Advanced data structures and methods are required for efficiently checking temporal conditions within the narrative graph of a document. For checking paths not following any of the given structural axis of XML documents, XSLT-based verification of criteria is not adequate and efficient.

8.4. Comparative Evaluation - Summary and Conclusion

XSLT + XPath is a very flexible and powerful tool for checking consistency conditions related to the XML implementation of documents. However, when criteria are not directly related to the structure of the XML representation of a document but to its content and narrative structure, XSLT is sufficiently efficient in simple cases only.

Already in the case of linear documents, XML-based checking of coherence criteria results in complex specifications and does not match the performance of model checking. For checking properties of narrative paths, which do not follow one of the structural axis of XML documents, model checking is clearly more powerful and efficient than XSLT processing. In general, XSLT is not an adequate and efficient platform for checking narrative paths in documents.

In addition, the performance of XSLT processing highly depends on the XSLT processor and the implementation of criteria in terms of XPath expressions and XSL code. The user has to deal with many low level decisions, for instance, if and how index structures can be applied to increase the performance of the implementation. Consequently, much experience and many test cycles are required to achieve good results when adopting XSLT for the verification of the document's content and structure.

$\mathcal{ALC}$CTL enables the compact representation and generic verification of an important class of structural conditions and content-related criteria that are difficult to check using standard XML techniques. $\mathcal{ALC}$CTL model checking exceeds the performance of XSLT processing in simple scenarios and offers a much higher and more predictable performance for complex criteria and large documents.

In contrast, XML validation techniques are powerful and flexible for ensuring the correctness of the XML representation of content, which is an important aspect of the document's consistency, indeed.

Obviously, the combination of XML-based document validation and logic-based verification offers the highest level of flexibility, expressiveness, and performance for different types of criteria on XML documents. Such a "hybrid" approach to document checking may be a promising direction of future research.

9. Conclusion

We have presented a new approach to combining semantic modelling, process modelling, ontological reasoning, and model checking, to enable the verification of content-related consistency criteria on documents with a branching narrative structure.

A semantic document model based on DL knowledge bases serves as an abstraction from irrelevant implementation details and allows for combining instance-level knowledge about the document with background knowledge about its domain of discourse. Ontological reasoning is adopted for checking the local and global consistency of the knowledge representation and for deriving implicit knowledge about the document. This increases the robustness and expressiveness of the framework and simplifies specifications and the knowledge extraction process.

The temporal description logics $\mathcal{ALC}$CTL is defined as a new specification formalism for structured content- and path-related properties. $\mathcal{ALC}$CTL is expressive for constraining properties of processes in structured domains. By modelling the reception of documents as a (reading) process on structured content objects, $\mathcal{ALC}$CTL is a powerful tool for checking the consistency and coherence of content along individual reading paths through the document. This is very difficult to achieve using existing verification methods for propositional temporal logics, or using ”low level” methods, such as XML validation, that work directly on the document data.

Document verification is modelled as an $\mathcal{ALC}$CTL model checking problem. For the first time, the model checking problem of a temporal description logics is examined in detail. Assuming finite domains and a finite set of states, the $\mathcal{ALC}$CTL model checking problem is shown to be decidable and computable in polynomial time. The first model checking algorithm for a temporal description logics is proposed and its soundness, completeness, and polynomial runtime complexity are shown.

Furthermore, it has been shown how knowledge about a document represented in *non-temporal* DL can be adopted for checking specifications in a *temporal* DL. Open world reasoning based on non-temporal DL and closed world reasoning (model checking) based on a temporal DL are combined in such a way that high expressiveness for content-related and structural criteria is achieved while a moderate polynomial runtime complexity can be guaranteed. In addition, the flexible combination of open world and closed world reasoning methods allows for tuning the framework to the characteristics of the given application scenario.

The formal framework has been implemented and evaluated in a number of case studies on real and synthetic documents. Already the first prototypical implementation of $\mathcal{ALC}$CTL model checking exceeds the performance of state-of-the-art CTL model checkers by magnitudes, although CTL model checking is computationally less complex than $\mathcal{ALC}$CTL model checking. Also, efficient "low level" methods such as XSLT processing cannot match the performance of $\mathcal{ALC}$CTL model checking. This gives evidence to the high potential of the developed methods not only for verifying documents. Further promising applications include the modelling and verification of business processes as well as the specification and verification of service-oriented architectures and web services.

Bibliography

[Abs06] Matthias Absmeier. Eine Validierungsumgebung für adaptierbare XML-basierte Dokumente. Diplomarbeit, Universität Passau, 2006.

[Adv04a] Advanced Distributed Learning (ADL). *Sharable Content Object Reference Model (SCORM) 2004 2nd Edition, Overview*, 2004.

[Adv04b] Advanced Distributed Learning (ADL). *Sharable Content Object Reference Model (SCORM), Content Aggregation Model (CAM), Version 1.3.1*, 2004.

[AF98] Alessandro Artale and Enrico Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research*, 9:463–506, 1998.

[AF00] Alessandro Artale and Enrico Franconi. Temporal description logics. In L. Vila, P. van Beek, M. Boddy, M. Fisher, D.M. Gabbay, A. Galton, and R. Morris, editors, *Handbook of Time and Temporal Reasoning in Artificial Intelligence*. MIT Press, 2000.

[AF01] Alessandro Artale and Enrico Franconi. A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 30(1-4):171–210, 2001.

[AFM+01] A. Artale, E. Franconi, M. Mosurovic, F. Wolter, and M. Zakharyaschev. Reasoning over conceptual schemas and queries in temporal databases. Technical report, Dept. of Computer Science, Univ. of Manchester, UK, 2001.

[AFM03] Alessandro Artale, Enrico Franconi, and Federica Mandreoli. Description logics for modelling dynamic information. In Jan Chomicki, Ron van der Meyden, and Günter Saake, editors, *Logics for Emerging Applications of Databases*. Springer, 2003.

[AFW+01] Alessandro Artale, Enrico Franconi, Frank Wolter, Milenko Mosurovic, and Michael Zakharyaschev. The DLR_{US} temporal description logic. In *Workshop Notes of the International Workshop on Description Logics (DL-01)*, Stanford University, CA, 2001.

[AFWZ02] Alessandro Artale, Enrico Franconi, Frank Wolter, and Michael Zakharyaschev. A temporal description logic for reasoning over conceptual schemas and queries. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA-02)*, volume 2424 of *LNAI*, pages 98–110, Cosenza, Italy, 2002. Springer.

[AHMS02] S. Autexier, D. Hutter, T. Mossakowski, and A. Schairer. The development graph manager maya (system description). In *Proceedings of the AMAST 2002*, volume 2422 of *LNCS*, pages 495–502. Springer, 2002.

[AM03] L. Aroyo and R. Mizoguchi. Process-aware authoring of web-based educational systems. In *Proceedings of the First International Workshop of Semantic Web for Web-based Learning (SW-WL03)*, pages 212–221, Velden, Austria, 2003.

[Bar03] C. Baral. *Knowledge representation, reasoning and declarative problem solving with Answer Sets*. Cambridge University Press, 2003.

[BBF+01] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, and P. McKenzie. *Systems and software verification, model-checking techniques and tools*. Springer, Berlin, 2001.

[BCC+03] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. In Marvin Zelkowitz, editor, *Highly Dependable Software*, volume 58 of *Advances in Computers*, pages 118–149. Academic Press, 2003.

[BCCZ99] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. *LNCS*, 1579:193–207, 1999.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003.

[BCMH92] J.R. Burch, E.M. Clarke, K.L. McMillan, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98:142–170, 1992.

[BCRS06] Paul Buitelaar, Philipp Cimiano, Stefania Racioppa, and Melanie Siegel. Ontology-based information extraction with SOBA. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Genoa, Italy, 2006.

[BDSV05] Marco Brambilla, Alin Deutsch, Liying Sui, and Victor Vianu. The role of visual tools in a web application design and verification framework: A visual notation for LTL formulae. In D. Lowe and M. Gaedke, editors, *Proceedings of the 5th International Conference of Web Engineering, ICWE 2005*, volume 3579 of *LNCS*, pages 557–568. Springer, 2005.

[BDTW06] S. Ben-David, R. Trefler, and G. Weddell. Model checking the basic modalities of CTL with description logic. In *Proc. International Workshop on Description Logics*, pages 223–230, 2006.

[BDTW07] Shoham Ben-David, Richard Trefler, and Grant Weddell. Bounded model checking with description logic reasoning. In *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 4548 of *LNCS*, pages 60–72. Berlin / Heidelberg, 2007.

[BFGHS04] P. Baumgartner, U. Furbach, M. Gross-Hardt, and A. Sinner. Living book – deduction, slicing, and interaction. *Journal of Automated Reasoning*, 32(3):259 – 286, 2004.

[BFH00] Alexander Borgida, Enrico Franconi, and Ian Horrocks. Explaining alc subsumption. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI)*, pages 209–213, Berlin, Germany, 2000. IOS Press.

[BGvH+03] Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Proceedings of the Second International Semantic Web Conference*, volume 2870 of *LNCS*, pages 164–179, Florida, USA, 2003. Springer.

[BHGS01] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reasonable ontology editor for the semantic web. In *Proceedings of the 2001 Description Logic Workshop(DL 2001)*, pages 1–9, 2001.

[BHWZ02] S. Bauer, I. Hodkinson, F. Wolter, and M. Zakharyaschev. On Non-Local Propositional and Local One-Variable Quantified CTL*. In *Proceedings of TIME'02*, pages 2–9, Manchester, UK, 2002. IEEE Computer Science Press.

[BHWZ04] S. Bauer, I. Hodkinson, F. Wolter, and M. Zakharyaschev. On non-local propositional and weak monodic quantified CTL*. *Journal of Logic and Computation*, 14:3–22, 2004.

[BKW03] F. Baader, R. Küsters, and F. Wolter. Extensions to description logics. In *[BCM+03]*, chapter 6, pages 226 – 268. 2003.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 2001.

[BMC03] Sean Bechhofer, Ralf Möller, and Peter Crowther. The DIG description logic interface. In *Proceedings of International Workshop on Description Logics (DL2003)*, Rome, Italy, 2003.

[BN03] Franz Baader and Werner Nutt. Basic description logics. In *[BCM+03]*, chapter 2, pages 47 – 100. 2003.

[Boc03] Stefano Bocconi. Automatic presentation generation for scholarly hypermedia. In *Proceedings of the 1st International Workshop on Scholarly Hypertext at the fourteenth conference on Hypertext and Hypermedia (HyperText 2003)*, Nottingham, UK, 2003. ACM.

[BvHH+04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-ref/, 2004. last visit Nov. 2007.

[CCG+02] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Proceedings of Computer Aided Verification (CAV 02)*, volume 2404 of *LNCS*. Springer, 2002.

[CCGR00] Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, and Marco Roveri. NuSMV: A new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[CCJ+07] Roberto Cavada, Alessandro Cimatti, Charles Arthur Jochim, Gavin Keighren, Emanuele Olivetti, Marco Pistore, Marco Roveri, and Andrei Tchaltsev. *NuSMV 2.4 User Manual*. ITC-irst, Trento, Italy, 2007.

[CD88] Edmund M. Clarke and I. A. Draghicescu. Expressibility results for linear-time and branching-time logics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop*, volume 354 of *LNCS*, pages 428 – 437, London, UK, 1988. Springer.

[CdO95] J.-P. Courtiat and R.C. de Oliveira. On RT-LOTOS and its application to the formal design of multimedia protocols. *Annals of Telecommunications*, 50(11-12):888–906, 1995.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logic of Programs: Workshop*, volume 131 of *LNCS*, Yorktown Heights, NY, 1981. Springer.

[CER01] Frans Coenen, Barry Eaglestone, and Mick Ridley. Verification, validation, and integrity issues in expert and database systems: Two perspectives. *International Journal of Intelligent Systems*, 16(3):425 – 447, 2001.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244 – 263, 1986.

[CG03] Diego Calvanese and Giuseppe De Giacomo. Expressive description logics. In *[BCM+03]*, chapter 5, pages 184 – 225. 2003.

[CGP+02a] Alessandro Cimatti, Enrico Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Integrating BDD-based and SAT-based symbolic model checking. In *Frontiers of Combining Systems*, pages 49–56, 2002.

[CGP02b] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 4. edition, 2002.

[CGP02c] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*, chapter 6: Symbolic Model Checking. MIT Press, 4. edition, 2002.

[CGP02d] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*, chapter 3: Temporal Logics. MIT Press, 4. edition, 2002.

[CGP02e] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*, chapter 4: Model Checking. MIT Press, 4. edition, 2002.

[CHS04] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the self-annotating web. In *Proceedings of the 13th WWW Conference*, New York, USA, 2004.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, chapter 15: Dynamic Programming, pages 323 – 369. MIT Press, Cambridge, MA, USA, 2. edition, 2001.

[CW01] J. Caumanns and A. Wendt. On demand generation of instructional resources for on demand vocational training. In *Proceedings of the Workshop on Global Transdisciplinary Education, Research and Training*, Pasadena, CA, USA, 2001.

[dA01] L. de Alfaro. Model checking the world wide web. In *CAV 01: 13th Conference on Computer Aided Verification*, volume 2102 of *LNCS*. Springer, 2001.

[DAC99] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st International Conference on Software Engineering*, pages 411–420. IEEE Computer Society Press, 1999.

[Dah01] I. Dahn. Automatic textbook construction and web delivery in the 21st century. *Journal of Structural Learning and Intelligent Systems*, 14(4):401–413, 2001.

[Dar91] Richard B. Darst. *Introduction to linear programming, applications and extensions*. Dekker, New York, 1991.

[DHN03] Peter Dolog, Nicola Henze, and Wolfgang Nejdl. Logic-based open hypermedia for the semantic web. In *Proceedings of the International Workshop on Hypermedia and the Semantic Web, Hypertext 2003 Conference*, Nottinngham, UK, August 2003.

[DIN89] *DIN - Taschenbuch 166: Software Entwicklung - Programmierung - Dokumentation*. Beuth Verlag, 1989.

[DL91] P.T. Devanbu and D.J. Litman. Plan-based terminological reasoning. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 128–138, Cambridge, MA, 1991. Morgan Kaufmann.

[DL96] P.T. Devanbu and D.J. Litman. Taxonomic plan reasoning. *Artificial Intelligence*, 84:1–35, 1996.

[Don00] Jing Dong. Model checking the composition of hypermedia design components. In *Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research*, Mississauga, Ontario, Canada, 2000. IBM Press.

[Don03] Francesco M. Donini. Complexity of reasoning. In *[BCM+03]*, chapter 3, pages 101 – 141. 2003.

[DSD04] DSDL Overview, Preparatory Draft of 14 November 2004. http://dsdl.org, 2004. last visit Nov. 2007.

[EFT96] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Einführung in die mathematische Logik*, chapter 2: Syntax der Sprachen erster Stufe. Spektrum Akad. Verlag, Heidelberg, 4. edition, 1996.

[Eme81] E.A. Emerson. *Branching Time Temporal Logic and the Design of Correct Concurrent Programs*. PhD thesis, Harvard University, 1981.

[Eme90] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Models and Semantics*, pages 996–1072. Elsevier, 1990.

[ER98] B. Eaglestone and M. Ridley. Verification, validation and integrity issues in expert and database systems: The database perspective. In R. R. Wagner, editor, *Proceedings DEXA 98 Ninth International Workshop on Database and Expert Systems Applications*, pages 22 – 29. IEEE Computer Society, 1998.

[ESS05] U. Egly, B. Schiemann, and J. Schneeberger. Technical documentation authoring based on semantic web methods. *Künstliche Intelligenz*, 2:56–59, 2005.

[FFLS99] M. Fernandez, D. Florescu, A. Y. Levy, and D. Suciu. Verifying integrity constraints on web sites. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 614–619. Morgan Kaufmann Publishers Inc., 1999.

[Fin00] A. Finkelstein. A foolish consistency: Technical challenges in consistency management. In M. Ibrahim, J. Küng, and N. Revell, editors, *Database and Expert Systems Application - 11th International Conference DEXA 2000, Proceedings*, volume 1873 of *LNCS*. Springer, 2000.

[FMPR04] S. Flake, W. Mueller, U. Pape, and J. Ruf. Specification and formal verification of temporal properties of production automation systems. In H. Ehrig, W. Damm, J. Desel, M. Große-Rhode, W. Reif, E. Schnieder, and E. Westkämper, editors, *Integration of Software Specification Techniques for Applications in Engineering, Priority Program SoftSpez of the German Research Foundation (DFG), Final Report*, volume 3147 of *LNCS*, pages 206–226. Springer, Heidelberg, Germany, 2004.

[FSMZ95] Burkhard Freitag, Bernhard Steffen, Tiziana Margaria, and Ulrich Zukowski. An approach to intelligent software library management. In *Proceedings of the 4th International Conference on Database Systems for Advanced Applications (DASFAA '95), National University of Singapore*, pages 71–78, 1995.

[GBvOH03] Joost Geurts, Stefano Bocconi, Jacco van Ossenbruggen, and Lynda Hardman. Towards ontology-driven discourse: From semantic graphs to multimedia presentations. In *Proceedings of the Second International Semantic Web Conference (ISWC2003)*, volume 2870 of *LNCS*, pages 597–612, Sanibel Island, Florida, USA, 2003. Springer.

[GG95] M.A. Gernsbacher and T. Givón, editors. *Coherence in spontaneous text*. John Benjamins, Philadelphia, 1995.

[GPH05] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2):158 – 182, 2005.

[Gru98] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):21–66, 1998.

[Gru06] Schema Gruppe. Schema ST4 Leistungsbeschreibung. SCHEMA Electronic Documentation Solutions GmbH, 2006.

[Har87] D. Harel. Statecharts: a visual formalism for complex systems. *The Science of Computer Programming*, 8:231–274, 1987.

[HH06] Minmin Han and Christine Hofmeister. Modeling and verification of adaptive navigation in web applications. In *Proceedings of the 6th international conference on Web engineering*, pages 329 – 336, Palo Alto, California, USA, 2006. ACM Press.

[HKNP92] J. Heinsohn, D. Kudenko, B. Nebel, and H.J. Profitlich. RAT: representation of actions using terminological logics. Technical report, Saarbrücken, Germany, 1992.

[HM01] V. Haarslev and R. Möller. RACER system description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR-01)*, volume 2083 of *LNAI*. Springer, 2001.

[HMS04a] U. Hustadt, B. Motik, and U. Sattler. Reasoning for description logics around $\mathcal{SHIQ}$ in a resolution framework. Technical Report 3-8-04/04, FZI, 2004.

[HMS04b] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}$ description logic to disjunctive datalog programs. In *Proceedings of the 9th International Conference on Knowledge Representation and Reasoning (KR2004)*, pages 152–162, Whistler, Canada, 2004.

[Hor98] Ian Horrocks. The FaCT system. In Harrie de Swart, editor, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods(TABLEAUX-98)*, volume 1397 of *LNAI*, pages 307–312. Springer, 1998.

[HR04] Michael Huth and Mark Ryan. *Logic in Computer Science, modelling and reasoning about systems*, chapter 3: Verification by model checking. Cambridge University Press, 2. edition, 2004.

[HS01a] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$ description logic. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 199 – 204. Morgan Kaufmann, 2001.

[HS01b] D. Hutter and A. Schairer. Towards an evolutionary formal software-development. In *Proceedings of the 16th International Conference on Automated Software Engineering (ASE'01)*, San Diego, USA, 2001.

[HS03] Siegfried Handschuh and Steffen Staab. *Annotation for the Semantic Web*, volume 96 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2003.

[HST00] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with individuals for the description logic SHIQ. In *Proceedings of the 17th International Conference on Automated Deduction*, volume 1831 of *LNCS*, pages 482–496, Pittsburgh, USA, 2000. Springer.

[Hüb00] R. Hübscher. Logically optimal curriculum sequences for adaptive hypermedia systems. In P. Brusilovsky, O. Stock, and C. Strapparava, editors, *International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, volume 1892 of *LNCS*, pages 121–132. Springer, 2000.

[Hut00] D. Hutter. Management of change in verification systems. In *Proc. of the 15th IEEE International Conference on Automated Software Engineering*, pages 23–34. IEEE Computer Society, 2000.

[HWZ01] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Monodic fragments of first-order temporal logics: 2000–2001 a.d. *Logic for Programming, Artificial Intelligence and Reasoning, LNAI*, 2250:1–23, 2001.

[HWZ02] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable and undecidable fragments of first-order branching temporal logics. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science (LICS 2002)*, pages 393–402, Copenhagen, Denmark, 2002.

[IM04] A. Inaba and R. Mizoguchi. Learners' roles and predictable educational benefits in collaborative learning - an ontological approach to support design and analysis of CSCL. In *Proc. of the seventh International Conference on Intelligent Tutoring Systems (ITS2004)*, Alagoas, Brazil, 2004.

[IMS03] IMS Global Learning Consortium. IMS Content Packaging Information Model, Version 1.1.3 Final Specification, 2003.

[ISO06] Information technology – Document Schema Definition Languages (DSDL) - Part 3: Rule-based validation - Schematron, International Standard, ISO/IEC 19757-3, First edition 01 June 2006, 2006.

[Jak06] Mirjana Jakšić. An approach to the example-based consistency checking of web documents. In *Proceedings of the 18th Workshop on Foundation of Databases*, pages 75–79, Wittenberg, Germany, 2006.

[Jan01] Tomi Janhunen. On the effect of default negation on the expressiveness of disjunctive rules. *LNCS*, 2173:93+, 2001.

[Jel02] Rick Jelliffe. The schematron assertion language 1.6. http://xml.ascc.net /resource/schematron/Schematron2000.html, 2002. last visit Nov. 2007.

[KA03] Michael Kohlhase and Romeo Anghelache. Mathematical knowledge management. In *Proceedings of the Second International Conference, MKM 2003*, Bertinoro, Italy, 2003. Springer LCNS.

[Kay04] Michael Kay. *XSLT 2.0 Programmer's Reference. (Programmer to Programmer)*. Wiley & Sons, 3. edition, 2004.

[KBHL+03] B. Krieg-Brückner, Dieter Hutter, Arne Lindow, Christoph Lüth, Achim Mahnke, Erica Mehlis, Philipp Meier, Arnd Poetzsch-Heffter, Markus Roggenbach, George Russell, Jan-Georg Smaus, and Martin Wirsing. Multimedia instruction in safe and secure systems. *Recent Trends in Algebraic Development Techniques,LNCS*, 2755:82–117, 2003.

[KBLL+04] Bernd Krieg-Brückner, Arne Lindow, Christoph Lüth, Achim Mahnke, and George Russell. Semantic interrelation of documents via an ontology. In G. Engels and S. Seehusen, editors, *Proceedings of the 2nd e-Learning Workshop Computer Science*, pages 271–282, Paderborn, Germany, 2004. Springer.

[KC05] Sascha Konrad and Betty H. C. Cheng. Real-time specification patterns. In *Proceedings of the 27th International Conference on Software Engineering*, pages 372 – 381, St. Louis, MO, USA, 2005. ACM Press.

[KE06] A. Kemper and A. Eickler. *Datenbanksysteme - Eine Einführung*. Kapitel 5: Datenintegrität. Oldenbourg Verlag, 6. Auflage, 2006.

[Kep04] Stephan Kepser. A simple proof of the turing-completeness of XSLT and XQuery. In Tommie Usdin, editor, *Proceedings of the Extreme Markup Languages Conference*, Montreal, Canada, 2004. IDEAlliance.

[KF01] Michael Kohlhase and Andreas Franke. Mbase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation*, 23(4):365–402, 2001.

[Koh00] Michael Kohlhase. OMDoc: Towards an Internet Standard for the Administration, Distribution and Teaching of Mathematical Knowledge. In *Proceedings of Artificial Intelligence and Symbolic Computation*, LNAI, Madrid, Spain, 2000. Springer.

[Koz83] D. Kozen. Results on the propositional mu-calculus. In *Theoretical Computer Science*, volume 27, pages 333–355. 1983.

[KPT+04] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic annotation, indexing, and retrieval. *Elsevier's Journal of Web Semantics, ISWC2003 special issue*, 1(2), 2004.

[KT03] S. Kuhlins and R. Tredwell. Toolkits for generating wrappers - a survey of software toolkits for automated data extraction from websites. In Mehmet Aksit, Mira Mezini, and Rainer Unland, editors, *Objects, Components, Architectures, Services, and Applications for a Networked World, International Conference NetObjectDays (NODe 2002)*, volume 2591 of *LNCS*, pages 184–198. Springer, 2003.

[KYNM04] T. Kasai, H. Yamaguchi, K. Nagano, and R. Mizoguchi. Development of a system that provides teachers with useful resources from various viewpoints based on ontology. In *Proc. of the sixteenth World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA 2004)*, pages 3349–3356, Lugano, Switzerland, 2004.

[Lea02] Learning Technology Standards Committee. Draft Standard for Learning Object Metadata, IEEE 1484.12.1/2002, July 2002.

[LH05] Thorsten Liebig and Michael Halfmann. Explaining subsumption in $\mathcal{ALEHF}_{R^+}$ TBoxes. In Ian Horrocks, Ulricke Sattler, and Frank Wolter, editors, *Proceedings of the 2005 International Workshop on Description Logics - DL2005*, pages 144–151, Edinburgh, Scotland, 2005.

[Liv02] Steven Livingstone. An Overview of MSXML 4.0. *XML.com*, 2002.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 1987.

[LS95] F. Laroussine and Ph. Schnoebelen. A hierarchy of temporal logics with past. *Theoretical Computer Science*, 148(2):303–324, 1995.

[LS00] F. Laroussine and Ph. Schnoebelen. Specification in CTL+Past for Verification in CTL. *Information and Computation*, 156(1–2):236–263, 2000.

[LSWZ01] C. Lutz, H. Sturm, F. Wolter, and M. Zakharyaschev. A tableau calculus for temporal description logic: The constant domain case. Technical Report LTCS-01-01, Germany, 2001.

[LT02] U. Lucke and D. Tavangarian. Turning a current trend into a valuable instrument: Multidimensional educational multimedia based on XML. In *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA 2002)*, Denver/Colorado/USA, 2002.

[LTV03] U. Lucke, D. Tavangarian, and D. Voigt. Multidimensional educational multimedia with <ML>^{3}. In *Proc. of the E-Learn Conference*, Phoenix, Arizona, USA, 2003.

[Lü06] BeiQi Lü. Evaluation von DL-Reasoning-Systemen für die Konsistenzprüfung von Dokumenten. Diplomarbeit, Universität Passau, 2006.

[Lut04] Carsten Lutz. An improved NEXPTIME-hardness result for description logic $\mathcal{ALC}$ extended with inverse roles, nominals, and counting. Ltcs-report 04-07, Technical University Dresden, 2004.

[LWR01] W. Lenski and E. Wette-Roch. The trial-solution approach to document re-use. In *Electronic Media in Mathematics*, Coimbra, 2001.

[McM92] K. L. McMillan. *Symbolic model checking - an approach to the state explosion problem*. PhD thesis, SCS, Carnegie Mellon University, 1992.

[McM93] K. L. McMillan. *Symbolic model checking*. Kluwer, Boston, 1993.

[mes06] validator product home page. http://www.messageautomation.com / products / validator.html, 2006. last visit Nov. 2007.

[MN04] Michael Marconi and Christian Nentwich. CLiX Language Specification Version 1.0, Specification, 31 January 2004. http://www.clixml.org, 2004. last visit Nov. 2007.

[MP92] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems - Specification*. Springer, 1992.

[MS98] Kim Marriott and Peter J. Stuckey. *Programming with constraints, an introduction*. MIT Press, Cambridge, 1998.

[MT87] W. C. Mann and S. A. Thomson. Rhetorical structure theory: A theory of text organization. Technical Report RS-87-190, Information Science Institute, USC ISI, USA, 1987.

[MYQ+06] L. Ma, Y. Yang, Z. Qiu, G. Xie, Y. Pan, and S. Liu. Towards a complete OWL ontology benchmark. In Y. Sure and J. Domingue, editors, *Proceedings of the 3rd European Semantic Web Conference (ESWC'06)*, volume 4011 of *LNCS*, pages 125 – 139, Budva, Montenegro, 2006. Springer.

[NCEF02] C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein. xlinkit: a consistency checking and smart link generation service. *ACM Transactions on Internet Technology (TOIT)*, 2(2):151–185, 2002.

[NEF03] Christian Nentwich, Wolfgang Emmerich, and Anthony Finkelstein. Consistency management with repair actions. In *Proceedings of the 25th international conference on software engineering*, pages 455–464, Portland, Oregon, 2003.

[NER00] B. Nuseibeh, S. Easterbrook, and A. Russo. Leveraging inconsistency in software development. *Computer*, 33(4):24 – 29, 2000.

[NP01] I. Niles and A. Pease. Towards a standard upper ontology. In C. Welty and B. Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, Ogunquit, Maine, 2001.

[NSD+01] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. Creating semantic web contents with Protege-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[OAS01] RELAX NG Specification, Committee Specification 3 December 2001. http://www.w3.org/TR/xmlschema-0/, 2001. last visit Nov. 2007.

[OBOC04] G. Ozsoyoglu, N.H. Balkir, Z.M. Ozsoyoglu, and G. Cormode. On automated lesson construction from electronic textbooks. *IEEE Transactions on knowledge and data engineering*, 16(3):130–140, 2004.

[PH02] Jeff Z. Pan and Ian Horrocks. Semantic web ontology reasoning in the $\mathcal{SHOQ}(\mathbf{D_n})$ description logic. In *Proc. of the 2002 Description Logic Workshop (DL 2002)*, volume 63 of *CEUR*, pages 53–62, 2002.

[Pil06] Andreas Pilger. Model-Checking von Temporalen Beschreibungslogiken durch Transformation in Prädikatenlogik erster Stufe. Diplomarbeit, Universität Passau, 2006.

[PKO+03] Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, and Miroslav Goranov. Towards semantic web information extraction. In *Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC2003)*, Florida, USA, 2003.

[PM95] A. Pnueli and Z. Manna. *Temporal Verification of Reactive Systems - Safety*. Springer, 1995.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[PSH07] Peter F. Patel-Schneider and Ian Horrocks. OWL 1.1 Web Ontology Language, Overview, Editor's Draft of 23 May 2007. http://www.webont.org/owl/1.1/overview.html, 2007. last visit Nov. 2007.

[PSK05] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL ontologies. In *Proceedings of the 14th international conference on World Wide Web*, pages 633 – 640. ACM Press, 2005.

[Rac07] RacerPro product web site. http://www.racer-systems.com, 2007. last visit Nov. 2007.

[Rad05a] Sven Radde. Ein System zur Verifikation von Webdokumenten mit temporaler Beschreibungslogik. Master's thesis, Lehrstuhl für Informationsmanagement, Universität Passau, 2005.

[Rad05b] Sven Radde. Ein System zur Verifikation von Webdokumenten mit temporaler Beschreibungslogik. Diplomarbeit, Universität Passau, 2005.

[RRS+00] C. R. Ramakrishnan, I. V. Ramakrishnan, S. A. Smolka, Y. Dong, X. Du, A. Roychoudhury, and V. N. Venkatakrishnan. XMC: A logic-programming-based verification toolset. In *Proceedings of CAV'00*, volume 1855, pages 576–580, Chicago, USA, 2000. Springer.

[Sch90] A. Schmiedel. A temporal terminological logic. In *Proceedings of the 8th National Conferene of the American Association for Artificial Intelligence*, pages 640–645, Boston, 1990.

[Sch93] K. Schild. Combining terminological logics with tense logic. In *Proceedings of the 6th Portuguese Conference on Artificial Intelligence*, pages 105–120, Porto, 1993.

[Sch94] Klaus Schild. Tractable reasoning in a universal description logic. In Franz Baader, Martin Buchheit, Manfred A. Jeusfeld, and Werner Nutt, editors, *Reasoning about Structured Objects: Knowledge Representation Meets Databases, Proceedings of 1st Workshop KRDB'94*, volume 1 of *CEUR Workshop Proceedings*, pages 20–22, Saarbrücken, Germany, September, 1994. CEUR-WS.org.

[Sch03] P. Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic*, volume 4, pages 393–436. King's College Publications, London, 2003.

[Sch04] Jan Scheffczyk. *Consistent Document Engineering*. PhD thesis, University of Armed Forces Munich, 2004.

[Sch06] Christian Schönberg. Model checking temporal description logics. Master's thesis, University of Passau, 2006.

[Sch08] Petra Schwaiger. *Ein Bedingungsmodell für Planungsprobleme in strukturierten Domänen*. Dissertation, Universität Passau, 2008. to be published.

[SCSD98] C. A. S. Santos, Jean-Pierre Courtiat, Luiz Fernando G. Soares, and Guido L. De Souza. Formal specification and verification of hypermedia documents based on the nested context model. In *Proceedings of the Conference on Multimedia Modeling*, 1998.

[SD02] Michael Sintek and Stefan Decker. TRIPLE – an query, inference, and transformation language for the semantic web. In *Proceedings of the 1st International Semantic Web Conference*, June 2002.

[SDM+05] E. Di Sciascio, F. M. Donini, M. Mongiello, R. Totaro, and D. Castelluccia. Design verification of web applications using symbolic model checking. In D. Lowe and M. Gaedke, editors, *Proceedings of the 5th International Conference of Web Engineering ICWE 2005*, volume 3579 of *LNCS*, pages 69–74. Springer, 2005.

[See03] Cornelia Seeberg. *Life long Learning - Modulare Wissensbasen für elektronische Lernumgebungen*. Springer, Berlin, Heidelberg, New York, 2003.

[SF02] Christian Süß and Burkhard Freitag. LMML - the learning material markup language framework. In *International Workshop ICL*, Villach, Austria, September 2002.

[SFB99] Christian Süß, Burkhard Freitag, and Peter Brössler. Metamodeling for web-based teachware managment. In P. P. Chen, D. W. Embley, J. Kouloumdijan, S. W. Liddle, and J. F. Roddick, editors, *Advances in Conceptual Modeling. ER'99 Workshop on the World-Wide Web and Conceptual Modeling*, volume 1727 of *LNCS*, pages 360–373, Paris, France, 1999. Springer.

[SFC+94] Bernhard Steffen, Burkhard Freitag, Andreas Claßen, Tiziana Margaria, and Ulrich Zukowski. Intelligent software synthesis in the DaCapo environment. In *Proceedings of the 6th Nordic Workshop on Programming Theory, available as BRICS Report 94-6, 1994*, University of Aarhus, 1994.

[SFC98] P. David Stotts, Richard Furuta, and Cyrano Ruiz Cabarrus. Hyperdocuments as automata: Verification of trace-based browsing properties by model checking. *Information Systems*, 16(1):1–30, 1998.

[SFR92] P.D. Stotts, R. Furuta, and J.C. Ruiz. Hyperdocuments as automata: Trace-based browsing property verification. In D. Lucarella, J. Nanard, M. Nanard, and P. Paolini, editors, *Proceedings of the ACM Conference on Hypertext (ECHT'92)*, pages 272–281. ACM Press, 1992.

[SK04] Carsten Sinz and Wolfgang Küchlin. Verifying the on-line help system of SIEMENS magnetic resonance tomographs. In *Proc. of the 6th Intl. Conf. on Formal Engineering Methods (ICFEM'2004)*, pages 391–402, Seattle, WA, USA, 2004.

[SN02] David Stotts and Jaime Navon. Model checking cobweb protocols for verification of HTML frames behaviour. In *Proceedings of the Eleventh International Conference on World Wide Web*, pages 182–190, Honolulu, Hawaii, USA, 2002. ACM Press.

[SPG+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2), 2007.

[SSC99] C. A. S. Santos, P. N. M. Sampaio, and J. P. Courtiat. Revisiting the concept of hypermedia document consistency. In *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, pages 183–186, Orlando, Florida, United States, 1999. ACM Press.

[SSdC98] C. A. S. Santos, L. F. G. Soares, G. L. de Souza, and J.-P. Courtiat. Design methodology and formal validation of hypermedia documents. In *Proceedings of the sixth ACM international conference on Multimedia*, pages 39–48, Bristol, United Kingdom, 1998. ACM Press.

[SSS00] Cornelia Seeberg, Achim Steinacker, and Ralf Steinmetz. Coherence in modularly composed adaptive learning documents. In *Proceedings of the AH2000*, 2000.

[SSS01] Ralf Steinmetz, Cornelia Seeberg, and Achim Steinacker. Coherence in the Learning System k-Med. In *Proceedings der GLDV-Frühjahrstagung 2001*, pages 17–27, March 2001.

[SZ01] G. Spanoudakis and A. Zisman. Inconsistency management in software engineering: Survey and open research issues. In S. K. Chang, editor, *Handbook of Software Engineering and Knowledge Engineering volume I*, pages 329 – 380. World Scientific Publishing Co., 2001.

[TA96] H. Thimbleby and M. A. Addison. Intelligent adaptive assistance and its automatic generation. *Interacting with Computers*, 8(1):51–68, 1996.

[TC05] OASIS Darwin Information Typing Architecture (DITA) TC. OASIS Darwin Information Typing Architecture (DITA) Language Specification v1.0, OASIS Standard, May 2005.

[TH06] Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNAI*, pages 292 – 297. Springer, 2006.

[TL97] H. Thimbleby and P. B. Ladkin. From logic to manuals again. *Software Engineering Journal*, 11(6):347–354, 1997.

[Tob00] Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research (JAIR)*, 12:199–217, 2000.

[Tob01] Stephan Tobies. *Complexity results and practical algorithms for logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[UCI+06] Victoria Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 4(1):14–28, 2006.

[vB95] J. van Benthem. Temporal logic. In Dov M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 4: Epistemic and Temporal Reasoning*. Oxford Science Publications, 1995.

[Vli01] Eric van der Vlist. Comparing XML Schema Languages. *XML.com*, 2001.

[Vli07] Eric van der Vlist. *Schematron*. Short Cut. O'Reilly, 2007.

[W3C99a] XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999. http://www.w3.org/TR/xpath, 1999. last visit Nov. 2007.

[W3C99b] XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999. http://www.w3.org/TR/xslt, 1999. last visit Nov. 2007.

[W3C01] XML Linking Language (XLink) Version 1.0, W3C Recommendation 27 June 2001. http://www.w3.org/TR/xlink/, 2001. last visit Nov. 2007.

[W3C04a] OWL Web Ontology Language, Overview, W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-features/, 2004. last visit Nov. 2007.

[W3C04b] RDF Primer, W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-primer/, 2004. last visit Nov. 2007.

[W3C04c] Resource Description Framework (RDF) Concepts and Abstract Syntax, W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-concepts/, 2004. last visit Nov. 2007.

[W3C06] Extensible Markup Language (XML) 1.1 (Second Edition), W3C Recommendation 16 August 2006, edited in place 29 September 2006. http://www.w3.org/TR/xml11/, 2006. last visit Nov. 2007.

[W3C07a] SPARQL Query Language for RDF, W3C Candidate Recommendation 14 June 2007. http://www.w3.org/TR/rdf-sparql-query/, 2007. last visit Nov. 2007.

[W3C07b] XML Path Language (XPath) 2.0, W3C Recommendation 23 January 2007. http://www.w3.org/TR/xpath20, 2007. last visit Nov. 2007.

[W3C07c] XQuery 1.0: An XML Query Language, W3C Recommendation 23 January 2007. http://www.w3.org/TR/xquery, 2007. last visit Nov. 2007.

[W3C07d] XSL Transformations (XSLT) Version 2.0, W3C Recommendation 23 January 2007. http://www.w3.org/TR/xslt20, 2007. last visit Nov. 2007.

[W3X04] XML Schema Part 0: Primer Second Edition, W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-0/, 2004. last visit Nov. 2007.

[WF04] Franz Weitl and Burkhard Freitag. Checking semantic integrity constraints on integrated web documents. In *Proceedings of the ER'04 International Workshop on Conceptual Model-directed Web Information Integration and Mining*, volume 3289 of *LNCS*, pages 198–209, Shanghai, China, 2004. Springer.

[WF06] Franz Weitl and Burkhard Freitag. Checking content consistency of integrated web documents. *Journal of Computer Science & Technology*, 21(3):418–429, 2006.

[WHR+05] Hai Wang, Matthew Horridge, Alan L. Rector, Nick Drummond, and Julian Seidenberg. Debugging OWL-DL ontologies: A heuristic approach. In *International Semantic Web Conference*, pages 745–757, 2005.

[WL92] R. Weida and D. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 282–293, Cambridge, MA, 1992. Morgan Kaufmann.

[WL94] R. Weida and D. Litman. Subsumption and recognition of heterogeneous constraint networks. In *Proceedings of CAIA-94*, 1994.

[WLLB06] Timo Weithoener, Thorsten Liebig, Marko Luther, and Sebastian Boehm. What's wrong with OWL benchmarks? In *Proceedings of the Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006)*, pages 101–114, Athens, GA, USA, 2006.

[WMS05] Norman Walsh, Leonard Muellner, and Bob Stayton. *DocBook: The Definitive Guide, Version 2.0.12*. O'Reilly, Apr 2005. Online Version.

[WZ00] F. Wolter and M. Zakharyaschev. Temporalizing description logic. *Frontier of Combining Systems*, 2:379–402, 2000.

[Xal07] Xalan-java version 2.7.0., project home page. http://xml.apache.org /xalan-j, 2007. last visit Nov. 2007.

[XLi01] XML Linking Language (XLink) Version 1.0 W3C Recommendation 27 June 2001. http://www.w3.org/TR/xlink/, 2001. zuletzt besucht Apr. 2004.

Index