An Approach to the Example-based Consistency Checking of Web Documents

Mirjana Jakšić

Universität Passau, Fakultät für Mathematik und Informatik
D-94030 Passau, Germany
Mirjana.Jaksic@uni-passau.de

Abstract

In this paper we present a method for the end-user specification of consistency rules for Web documents. Temporal description logic is used as the internal formalism. It is difficult for an author of a Web document to understand and use a formal logic. Therefore the goal is to develop an intuitive front-end allowing the authors of a document to specify their requirements and constraints at an application-oriented level. To this end we propose a specification-by-example approach.

1 Introduction

Consider the process of authoring and revising *Web documents*, i.e. a collection of Web pages correlated to each other regarding content and structure. We consider a Web document *consistent* if its argumentation structure and content conform to a set of target properties. For example, in a formal document every *term* has to be *defined before* it is *used*.

In this paper we address the question how consistency rules for Web documents can be expressed in a way that is both suitable for a formal verification system and intuitive enough to support an author in this process. One possible approach is to use XML-based techniques. However, the XML presentation level is too technical for users and it is not suitable for the specification of high-level properties like the one mentioned above.

The argumentation structure and interrelationships of topics in the discussion domain can be represented by knowledge representation formalisms like description logics [1]. Some formal logics have already been shown to be suitable for expressing consistency rules for Web content: description logic (DL) is used in SCHEMA ST4 system [2], temporal logic (TL) [3] in [4] and temporal description logic (TDL) in [5].

Obviously, it is difficult for an author to understand and use formal logics. Therefore a high-level specification method for expressing consistency rules is required.

2 Consistency Checking of Web Documents

2.1 Consistency Checking System

Assume there are several text components that build a Web document. Individual components are indexed with various metadata. The first step in the authoring process of the composite document is the metadata extraction from text components and its integration into a knowledge base. We consider description logic as a suitable formalism for this purpose.

Frequently, the authors of a Web document recycle fragments of existing documents to compose a new document. In this case, the document's components are most probably based on different information models. The relations between different information models are represented in the form of an ontology, which serves as a framework for formulation of consistency rules. An ontology for documents describes, for example, typical *structure units* (chapters, sections, paragraphs, definitions, examples, comments) and *topics* of the domain of discourse. It also enables the specification of document properties on the different abstraction levels.

Step two is the construction of the verification model, which reflects the narrative structure of the Web document. The consistency rules are expressed in the temporal description logic $\mathcal{ALC}$CTL [6] and the verification model is checked against consistency rules by model checking.

2.2 Consistency Rules

In our system, consistency rules constrain the content and structure of a Web document. They address certain more or less abstract document properties like content and function of a text unit. Content comprises, for example, *themes* and *concepts*, and a function can be *exercise*, *solution*, *description of functionality*, etc. Examples of consistency rules are:

- "Every learning unit in a Web-based training begins with an *introduction* and concludes with an *exercise*, *test* and *conclusion*."
- "An *exercise* should come after the concepts needed for its solution are presented."
- "In every variant of a manual, every *function* listed in the overview must be followed by the appropriate *description*."

3 Example-based High-level Specification

3.1 Solution to High-level Specification

To create a method for building the consistency rules in a way that is on the one hand adequate for the verification system, and on the other hand "user friendly" enough for an author, we have decided to combine several basic ideas - example-based and diagram-based approaches with already existing specification patterns for reactive systems [7].

We are aware of the fact that the proposed approach cannot provide the full expressive power of the formal specification. However, to obtain an intuitive and "user friendly" system, we have to sacrifice some expressiveness.

3.2 Specification Patterns for Web Documents

A pattern-based approach to the presentation, codification and reuse of property specifications in reactive systems has been presented in [7]. Until now, seven specification formalisms are supported, among them CTL [8]. For each of the supported formalisms, a set of the possible constraints has been defined and patterns have been created for them. The patterns together with descriptions of their meaning are provided to the users who can identify similar requirements in their systems and select patterns that address those requirements.

Altogether eight patterns have been defined, which are divided in two groups - Occurrence patterns and Order patterns. Occurrence patterns are: *Universality*, *Absence*, *Existence* and *Bounded Existence*. Order patterns are: *Response*, *Response Chain*, *Precedence* and *Precedence Chain*. Following scopes

are defined for every pattern: *Global*, *Before X*, *After X*, *Between X and Y* and *After X until Y*. The scope describes the extend of the program execution over which a pattern must hold.

According to performed research, we take all of the existing patterns into account for Web documents. We add two scopes for all patterns from the Occurrence patterns group: *Immediately Before X* and *Immediately After X*. Our adaptation of patterns has already been tested on a small set of use cases.

3.3 Specification Process

The basic idea of our approach is to support the process of specifying the consistency rules, rather than to build a new specification language. The entire process can be described as a stepwise refinement. The specification process runs in seven steps alternately between the author and the system. At the beginning of the process it is supposed that the author has specified the document he/she works with. Further, this document will be referenced as the *current document*.

Step 1: The author's goal in this step is to choose a general example of a possible path through a Web document that is most adequate for his/her needs. It comprehends the basic patterns and scopes. To make it easier to the author, this decision is supported by a simple example that should be clear to most of the users.

Step 2: The goal of this step is the visual presentation of the knowledge base content. To help the author to build the $\mathcal{ALC}$ part of an $\mathcal{ALC}$CTL formula, i.e. the ontological part, the system shows only concepts relevant to the *current document*. For every relevant concept its instances from the *current document* are also presented.

Step 3: The author's goal in this step is to extend the general path (chosen in the first step) to an example path through the *current document*. The example path represents the temporal component of an $\mathcal{ALC}$CTL formula.

First, the author chooses the concept instances (presented in the previous step) and places them in the general path. From now on, the *current document* with its concepts and their instances is considered. Further, this step is divided into smaller sub-steps, so that the author refines the example path stepwise.

The author's second goal in this step is to determine the domain of objects - the class of concepts for which is the rule valid.

Step 4: The goal of this step is the generalization of the DL part of the example path - from the instances to the concept level, and the transformation of the entire rule from the high-level specification into $\mathcal{ALC}$CTL. The second goal is building of several correct example documents, in accordance to the *current document*. These example documents are graphically presented to the author.

Step 5: The goal is to consider the example documents created in the previous step. If not satisfied with the examples he/she can go back to the previous steps.

Step 6: The goal of this step is verification, i.e. a model-checking process follows.

Step 7: At the end the author gets the verification results. He/she can go back to one of the previous steps if not satisfied.

3.4 An Example

Let us now consider the example of a Web-based training (WBT) unit about robots. Some of the concepts in this system are: *robot*, *topic*, *text fragment*, *load distribution*, *maintenance*, *description*, *example*, and *test*. Different properties, like *load distribution* or *maintenance*, can be presented for every robot. Every property is considered as a separate *topic*, i.e. the concept *topic* is a super-concept of all property concepts. Every *topic* can be subject of different *text fragments*, like for example of a *description* or *example*.

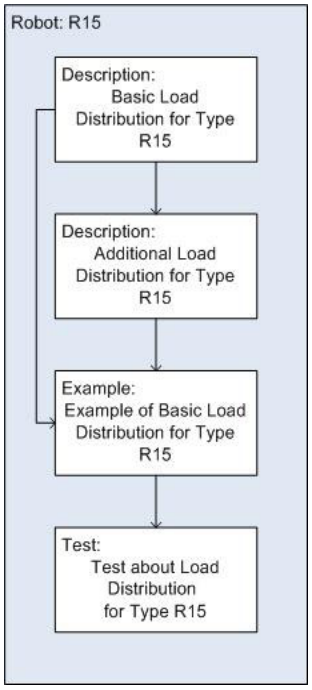

Figure 1: A part of an example WBT document

Figure 1 shows part of an abstract representation of a sample Web document. This document contains information about robot *R15*. First *Basic load distribution* is described, followed by *Additional load distributions*, which in turn is followed by the *Example of basic load distribution*. At the end comes *Test*. There are two possible paths through the document.

As an example, we want to express the following rule: "For every path holds: A *Description of a load distribution* comes always before an *Example* of it, if there is an *Example of a load distribution* at all. If there is no *Example*, the *Description* is optional."

The following describes steps 1-4 of the specification process for the above named specification rule.

Step 1: Concerning the offered examples, which representthe basic patterns and scopes, the author chooses the pattern "A *term definition* comes always before a *term usage*" in the context of the existence of a *term definition*.

Step 2: Relevant concepts and their instances in the *current document* are: *robot* (instance: *R15*), *load distribution* (instances: *Basic load distribution for R15* and *Additional load distribution for R15*), *description* (instances: *Description of a basic load distribution for R15* and *Description of an additional load distribution for R15*), *example* (instance: *Example of a basic load distribution for R15*), and *test* (instance: *Test about load distribution for R15*).

Step 3: For our example, an author chooses the following instances: *Description of a basic load distribution for R15* and *Example of a basic load distribution for R15*. Then he/she substitutes them for *term definition* and *term usage*, respectively, in the example chosen in the Step 1.

Now, we have a rule: "A *Description of a basic load distribution for R15* comes always before an *Example of a basic load distribution for R15*". From the formal point of view the previous sentence is incomplete in many respects. It is not clear whether a *Description* comes only before the first *Example* or before some *Example*. Or is there actually one *Description* before every *Example*? It is also not specified whether a *Description* and/or an *Example* are optional or mandatory. Furthermore, does this rule hold for all or only for some paths through the document? Each of these features is determined in separate sub-steps. After all sub-steps are performed, we have the following pattern: A $[\neg Q \ \mathsf{W} \ (P \sqcap \neg Q)]$ (Q

stands for *Example of a basic load distribution for R15* and *P* for *Description of a basic load distribution for R15*). The domain of objects is the concept *load distribution*.

Step 4: The following is the result of the generalization and transformation from the high-level specification into $\mathcal{ALC}$CTL:

$loadDistribution \sqsubseteq \mathsf{A}\,[\neg\exists topicOf.Example\ \mathsf{W}\ (\exists topicOf.Description \sqcap \neg\exists topicOf.Example)]$

The system also generates examples of valid component documents (see Figure 1) like the following:

- "*Basic load distribution for R15, Example of basic load distribution for R15, Test about basic and additional load distribution for R15.*"
- "*Basic load distribution for R15, Additional load distribution for R15, Example of basic load distribution for R15, Test about basic and additional load distribution for R15.*"

4 Conclusion

To make the expression of consistency rules simple and understandable to an author of WBT materials, we are developing an intuitive supporting system. The consistency rules are built as example paths through the *current document* in the process of the stepwise refinement.

After completing the implementation of our system, the development of an error explanation component is planned. The high-level specification formalism and the error explanation component together build the complete authoring environment for WBT materials.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] U. Egly, B. Schiemann, and J. Schneeberger. Technical documentation authoring based on Semantic Web methods. *Künstliche Inteligenz*, (2/2005):56–59, 2005.

[3] 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.

[4] D. Stotts and J. Navon. Model checking cobweb protocols for verification of html frames behaviour. In *Proc. of the 11th Int. Conf. on WWW*, pages 182–190. ACM Press, 2002.

[5] F. Weitl and B. Freitag. Checking semantic integrity constraints on integrated web documents. In *Proc. of the CoMWIM 2004*, volume 3289, pages 198–209. Springer-Verlag, 2004.

[6] F. Weitl and B. Freitag. Model checking semantic properties of web documents based on temporal description logics. Technical report, Teaching Chair of Information Management, University of Passau, 2006. fortcoming.

[7] 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.

[8] Property Pattern Mappings for CTL. http://www.patterns.projects.cis.ksu.edu/documentation/patterns/ctl.shtml, February 2006. Last visited Feb. 2006.