Dokumentverifikation mit Temporaler Beschreibungslogik

Franz Weitl

Fakultät für Informatik und Mathematik
Universität Passau
Franz.Weitl@uni-passau.de

Abstract: Es wird ein neues formales Framework für die automatische Prüfung inhaltlich-struktureller Vorgaben an Dokumente vorgestellt. Aus der Hard-/Softwareverifikation bekannte Model-Checking Verfahren werden mit Methoden zur Repräsentation von Ontologien kombiniert, um sowohl die Struktur des Dokuments als auch inhaltliche Zusammenhänge bei der Prüfung von Konsistenzkriterien berücksichtigen zu können. Als Spezifikationssprache für Konsistenzkriterien wird die neue temporale Beschreibungslogik $\mathcal{ALC}$CTL vorgeschlagen. Grundlegende Eigenschaften wie Entscheidbarkeit, Ausdruckskraft und Komplexität werden untersucht. Die Ergebnisse übertreffen bekannte Ansätze wie symbolisches Model-Checking in Performanz und Ausdruckskraft hinsichtlich der prüfbaren Kriterien.

1 Einführung

Wohl jeder hat sich schon einmal über Handbücher geärgert, in denen die benötigte Information nicht enthalten, nicht auffindbar, oder nicht verständlich dargestellt ist. Was bei Handbüchern für Consumer-Elektronik oder Haushaltsgeräte ärgerlich ist, kann bei der technischen Dokumentation von medizinischen Geräten oder Verkehrsflugzeugen ernste Folgen haben.

Elektronische Dokumente wie technische Dokumentationen oder Handbücher inhaltlich konsistent, aktuell und verständlich zu halten, ist insbesondere dann schwierig, wenn der Inhalt häufig aktualisiert und an verschiedene Produktvarianten oder Zielgruppen mit unterschiedlichen Informationsbedürfnissen angepasst werden muss. Erschwerend für die Qualitätssicherung von technischen Dokumenten kommt hinzu, dass diese nicht linear vom Anfang bis zum Ende gelesen werden, sondern fragmentarisch und abhängig vom aktuellen Informationsbedarf. Entlang der individuell gewählten Lesepfade soll das Dokument stimmig, konsistent und verständlich sein, oder allgemein, bestimmte Vorgaben an Inhalt und Struktur einhalten.

Die Anzahl individueller Lesepfade in nicht linear aufgebauten Dokumenten wächst exponentiell in der Größe des Dokuments, so dass die manuelle Überprüfung inhaltlicher Vorgaben entlang von Lesepfaden schon bei kleinen Dokumenten nicht mehr durchführbar ist und durch automatisierte Verifikation unterstützt werden muss. Die bislang existierenden Verfahren zur automatischen Dokumentprüfung [HH06, ISO06, NCEF02, Sch04, SFC98] können aber die für die Lesbarkeit von Dokumenten wichtigen inhaltlichen Zusammen-

hänge nicht ausreichend berücksichtigen oder scheitern an der kombinatorischen Explosion der möglichen Lesepfade durch ein Dokument.

Die neue temporale Beschreibungslogik $\mathcal{ALC}$CTL, eine Kombination der Beschreibungslogik $\mathcal{ALC}$ [BCM+03] und der temporalen Logik CTL [Eme90], bietet die für die Repräsentation von inhaltlichen Vorgaben nötige Ausdruckkraft, bleibt aber im Gegensatz zu ähnlich ausdrucksstarken Formalismen [AF01] entscheidbar und effizient prüfbar. Ein neuer Model-Checking Algorithmus, der erste für eine temporale Beschreibungslogik, ermöglicht die Überprüfung von inhaltlichen Zusammenhängen entlang von Lesepfaden durch Dokumente von mehr als 5000 Seiten in wenigen Sekunden und übertrifft die Geschwindigkeit des State-of-the-Art Model-Checkers NuSMV [CCG+02] um bis zu drei Größenordnungen.

Die hier zusammengefasste Dissertation ist im Rahmen des DFG-geförderten Projekts *Verdikt* entstanden, das sich mit der Verifikation semi-strukturierter Dokumente im Kontext befasst [SJWF09, WJF09]. In der Folge werden die behandelte Problemstellung präzisiert, der Ansatz und seine wesentlichen formalen Grundlagen skizziert und die wichtigsten Ergebnisse der Arbeit zusammengefasst.

2 Problemstellung

Als Szenario für die Illustration der Aufgabenstellung und -lösung dient ein Web-basiertes Training im XML-Format, das in Zusammenarbeit mit Industriepartnern entwickelt wurde (vgl. [WJF09]). Das Dokument vermittelt grundlegende Eigenschaften verschiedener Typen industrieller Roboter. Es wird in unterschiedlichen Varianten zur Schulung von Auszubildenden, als Referenz für Dozierende und als Informationsbasis für den technischen Support verwendet.

Abbildung 1 zeigt die grundsätzliche Struktur des Dokuments. Auf der Startseite (Home) werden die Lernziele und behandelten Themen des Dokuments aufgeführt. Der Benutzer kann hier verschiedene Lektionen wählen, beispielsweise eine allgemeine Einführung in Lektion 1 oder der Vorstellung unterschiedlicher Robotertypen in Lektion 2 usw. Die einzelnen Lektionen des Dokuments beinhalten neben Informationsseiten wie L2.1 und L2.2 in Abbildung 1) auch Aufgaben zum Selbsttest (L2.3) und zur Übung (L2.4) mit den zugehörigen Musterlösungen (L2.5), die aber nur in der Variante für Dozierende verfügbar sein sollen. Weitergehende Informationen werden in Exkursen (L2.1a und L2.1b) zur Verfügung gestellt.

Das WBT soll u.a. folgende Vorgaben erfüllen:

1. Alle erwähnten Hauptthemen sollen in der Folge behandelt werden.
2. Zu allen Aufgaben gibt es in der Folge Musterlösungen, die aber nur für Dozierende einsehbar sind.
3. Alle Informationen, die benötigt werden, um eine Aufgabe zu lösen, sind vorher vermittelt worden.

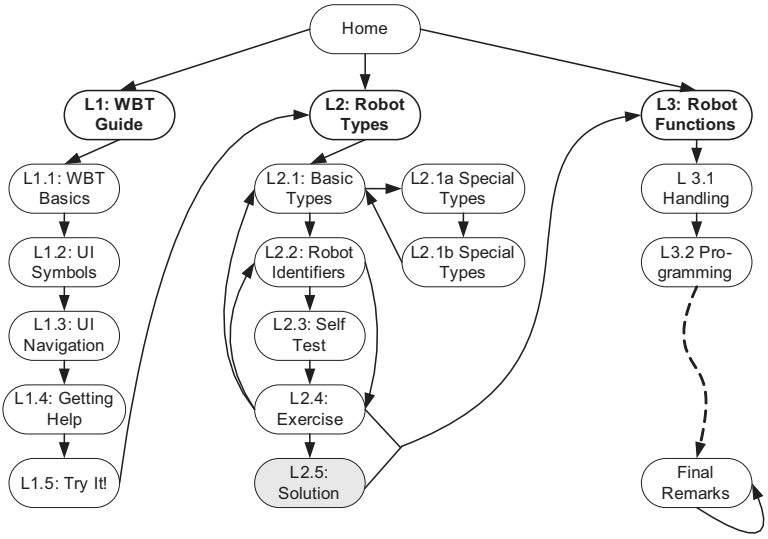

Abbildung 1: Web-basiertes Training über Roboter

Für die automatisierte Prüfung inhaltlich-struktureller Vorgaben müssen folgende Fragestellungen gelöst werden.

(**P1**) Wie kann Wissen über ein Dokument mit allgemeinem *Hintergrundwissen* über die Diskursdomäne kombiniert werden, so dass die für die Prüfung inhaltlicher Vorgaben nötigen Informationen (z.B. "Hauptthemen" in Vorgabe 1) hergeleitet werden können?

(**P2**) Was ist die adäquate formale Basis für die Definition einer Reihenfolge von Abschnitten in Dokumenten, die nicht linear strukturiert sind?

(**P3**) Welcher Formalismus bietet die passende Ausdruckskraft, um inhaltliche Vorgaben an Dokumente kompakt zu repräsentieren?

(**P4**) Welche Methode und welcher Algorithmus ist effizient genug, um auch große Dokumente und Dokumentkorpora gegen komplexe Vorgaben effizient prüfen zu können?

3 Ansatz

Abbildung 2 gibt einen Überblick über die Komponenten des Ansatzes.

Der Benutzer erstellt und aktualisiert ein *Dokument*, das aus ein oder mehreren Teilen zusammengesetzt und mit Metadaten annotiert ist (ⓐ in Abbildung 2 unten). Die Softwarekomponente für die *Wissensextraktion* (① in Abbildung 2) analysiert das Markup und die Metadatenannotationen des Dokuments und erstellt daraus ein *semantisches Modell*, das relevante Dokumenteigenschaften in Form von beschreibungslogischen Zusicherungen abbildet. Der Benutzer formalisiert *Hintergrundwissen* über die Diskursdomäne in

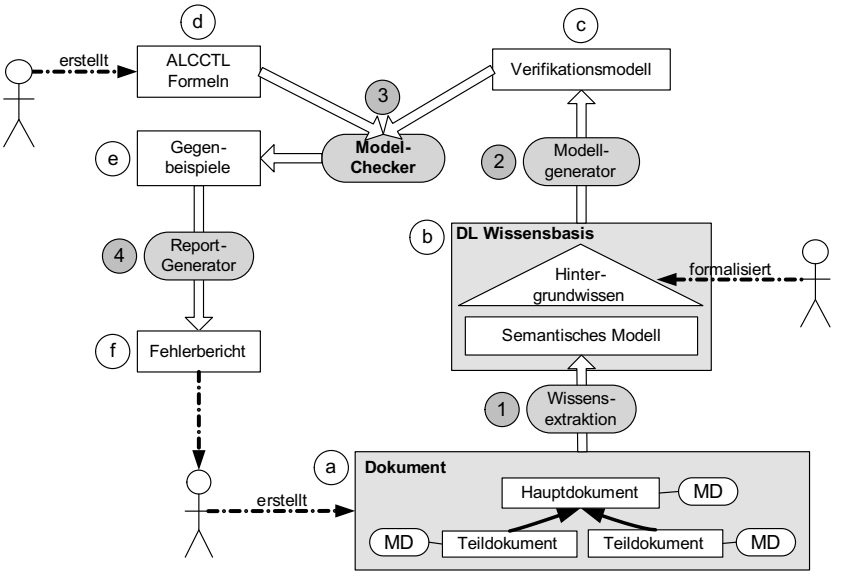

Abbildung 2: Komponenten des Systems zur Verifikation von Dokumenten

Beschreibungslogik (DL). Das semantische Modell und das Hintergrundwissen ergeben eine *DL Wissensbasis* (ⓑ in Abbildung 2), aus der die für die Prüfung von Vorgaben relevanten Informationen über das Dokument hergeleitet werden (→ **P1**).

Ein *Modellgenerator* (② in Abbildung 2) leitet aus der DL Wissensbasis ein *Verifikationsmodell* ab, das die Struktur und den Inhalt des Dokuments in Form eines annotierten Graphen repräsentiert (→ **P2**). Dieses Verifikationsmodell wird durch einen *Model-Checker* (→ **P4**) gegen eine Spezifikation in Form von *Formeln* der temporalen Beschreibungslogik $\mathcal{ALC}$CTL verifiziert (③ in Abbildung 2). $\mathcal{ALC}$CTL ermöglicht im Gegensatz zu existierenden Spezifikationsformalismen die kompakte Repräsentation von Vorgaben, die sich auf inhaltliche Zusammenhänge entlang von Lesepfaden beziehen (→ **P3**). Der Model-Checker generiert für jede Spezifikationsverletzung ein *Gegenbeispiel*. Auf der Basis dieser Gegenbeispiele erstellt der *Reportgenerator* (④ in Abbildung 2) einen detaillierten *Fehlerbericht*, der die Fehlerstellen im Ausgangsdokument lokalisiert.

4 Die temporale Beschreibungslogik $\mathcal{ALC}$CTL und Model-Checking

Die temporale Beschreibungslogik $\mathcal{ALC}$CTL und Model-Checking bilden die formale Basis für die Repräsentation und automatische Überprüfung von Vorgaben an Dokumente. Tabelle 1 definiert die Syntax von $\mathcal{ALC}$CTL. $\mathcal{ALC}$CTL Formeln p, q werden durch aussagenlogische Operatoren wie $\neg$ (Negation), $\wedge$ (Konjunktion) u.a., sowie durch die temporalen Operatoren von CTL wie EX (”some path next”), AX (”all paths next”) usw. aufgebaut.

$$
\begin{array}{rcl}
p, q & \rightarrow & C \sqsubseteq D | \neg p | p \wedge q | p \vee q | p \rightarrow q | \\
 & & \mathsf{EX}\, p | \mathsf{AX}\, p | \mathsf{EF}\, p | \mathsf{AF}\, p | \mathsf{EG}\, p | \mathsf{AG}\, p | \mathsf{E}(p\ \mathsf{U}\ q) | \mathsf{A}(p\ \mathsf{U}\ q) \\
C, D & \rightarrow & A | \top | \bot | \neg C | C \sqcap D | C \sqcup D | \exists R.C | \forall R.C | \\
 & & \mathsf{EX}\, C | \mathsf{AX}\, C | \mathsf{EF}\, C | \mathsf{AF}\, C | \mathsf{EG}\, C | \mathsf{AG}\, C | \mathsf{E}(C\ \mathsf{U}\ D) | \mathsf{A}(C\ \mathsf{U}\ D)
\end{array}
$$

Tabelle 1: $\mathcal{ALC}$CTL Syntax

Über CTL hinausgehend können $\mathcal{ALC}$CTL Formeln auch wie bei Beschreibungslogiken mit Hilfe des Subsumptionsoperators $\sqsubseteq$ und Konzeptausdrücken C, D konstruiert werden. Dafür stehen die $\mathcal{ALC}$ Operatoren $\top$ (universelles Konzept), $\bot$ (unerfüllbares Konzept), $\neg$ (Negation), $\sqcap$ (Konjunktion), $\sqcup$ (Disjunktion), $\forall$ und $\exists$ (universelle / existentielle Quantifikation über Rollen) zur Verfügung. Ferner können auch in Konzeptausdrücken temporale Operatoren verwendet werden (Tabelle 1 letzte Zeile). Die so entstehenden "temporalen Konzepte" ermöglichen erst die Repräsentation von Vorgaben an inhaltliche Zusammenhänge entlang von Pfaden und können weder in CTL noch in $\mathcal{ALC}$ ausgedrückt werden.

Bespiel 1 ($\mathcal{ALC}$CTL Syntax)

Die in Abschnitt 2 aufgeführten Vorgaben können durch $\mathcal{ALC}$CTL in der folgenden Weise repräsentiert werden:

1. Alle erwähnten Hauptthemen sollen in der Folge behandelt werden.

$$MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.\top \tag{1}$$

   "Jedes Hauptthema ($MajorTopic \sqsubseteq$) wird auf allen Pfaden (A) schließlich (F) von irgendeiner Dokumenteinheit behandelt ($\exists addressedBy.\top$).

2. Zu allen Aufgaben gibt es in der Folge Musterlösungen, die aber nur für Dozierende einsehbar sind.

$$\mathsf{AG}(Exercise \sqsubseteq \forall hasTask.\mathsf{EF}\ \exists solvedIn.\forall accessibleTo.Teacher) \tag{2}$$

   "Es gilt auf allen Pfaden (A) an jeder Stelle (G), dass Übungen ($Exercise$) nur solche Aufgaben haben ($\sqsubseteq\ \forall hasTask.$), die auf einem Pfad (E) schließlich (F) gelöst werden ($\exists solvedIn.$), wobei nur Dozierende auf die Lösung zugreifen können ($\forall accessibleTo.Teacher$).

3. Alle Informationen, die benötigt werden, um eine Aufgabe zu lösen, sind vorher vermittelt worden.

$$Information \sqsubseteq \neg\mathsf{E}(\neg Presented\ \mathsf{U}\ \exists requiredFor.Task) \tag{3}$$

   "Für jede Information gilt ($Information \sqsubseteq$): es gibt keinen Pfad ($\neg$E), auf dem diese Information niemals präsentiert wird ($\neg Presented$) bis (U) sie von einer Übungsaufgabe benötigt wird ($\exists requiredFor.Task$).

$MajorTopic$, $Exercise$, $Teacher$, $Information$, $Presented$, $Task$ sind ***Konzepte***, die Dokumentteile, Themen, Nutzergruppen und andere Objektklassen der Interpretationsdomäne repräsentieren. $addressedBy$, $hasTask$, $topicOf$, $solvedIn$, $accessibleTo$, $requiredFor$ sind ***Rollen***, die binäre Beziehungen zwischen Objekten der Interpretationsdomäne repräsentieren. $\sqsubseteq, \forall, \exists$ sind Operatoren der Beschreibungslogik $\mathcal{ALC}$ und quantifizieren Objekte der Interpretationsdomäne. A ("all paths") und E ("some paths") sind Pfadquantoren und F ("future"), G ("globally"), U ("until") sind temporale Operatoren der temporalen Logik CTL. Die Vorgaben (1) bis (3) können weder in Beschreibungslogiken wie $\mathcal{ALC}$ noch in propositionalen temporalen Logiken wie CTL ausgedrückt werden, da in keiner der beiden Logikklassen temporale Konzepte wie AF $\exists addressedBy.\top$ repräsentiert werden können. □

Der Wahrheitswert von $\mathcal{ALC}$CTL Formeln wird in Verifikationsmodellen der in Abbildung 3 skizzierten Form ermittelt. Verifikationsmodelle sind annotierte Graphen, deren

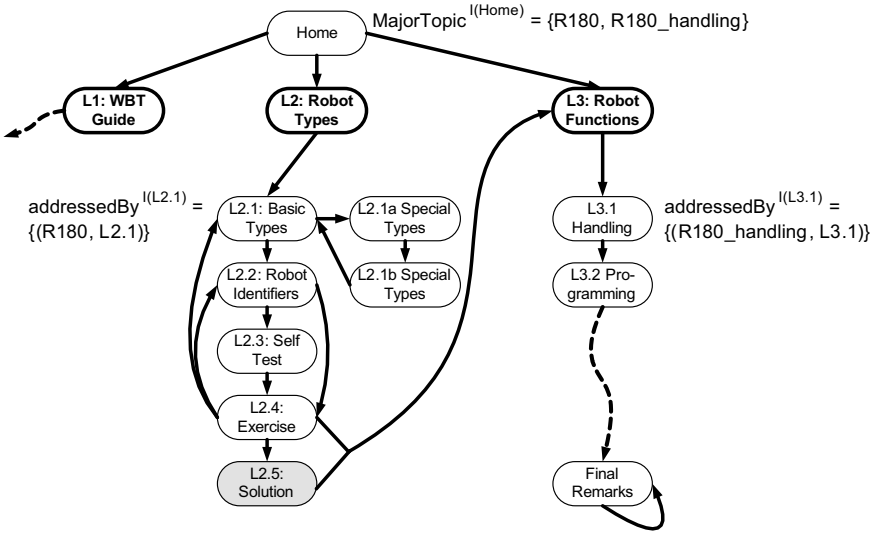

Abbildung 3: Ausschnitt aus dem Verifikationsmodell des Dokuments von Abbildung 1

Knoten in der betrachteten Anwendung inhaltlich geschlossene Dokumenteinheiten (z.B. Kapitel, Lektionen, Abschnitte, Absätze) repräsentieren. Die Kanten repräsentieren eine Nachfolgebeziehung zwischen Dokumenteinheiten. Beispielsweise drückt die Kante zwischen Knoten $Home$ und $L2$ in Abbildung 3 aus, dass nach der Einheit $Home$ Lektion $L2$ gelesen werden kann. Durch Verfolgen von Kanten entstehen Lesepfade durch das Dokument. Die Knoten des Graphen sind annotiert mit Interpretationen von atomaren Konzepten der Logik $\mathcal{ALC}$CTL wie $MajorTopic$ und atomaren Rollen von $\mathcal{ALC}$CTL wie $addressedBy$, welche lokale Eigenschaften von Dokumenteinheiten abbilden. Exemplarisch sind in Abbildung 3 Annotationen für die Einheiten $Home$, $L2.1$ und $L3.1$ enthalten. $MajorTopic^{I(Home)} = \{R180, R180\_handling\}$ in Abbildung 3 oben bedeutet,

dass der Robotertyp $R180$ und dessen Handhabung ($R180\_handling$) Hauptthemen der Einheit $Home$ sind. $addressedBy^{I(L2.1)} = \{(R180, L2.1)\}$ in Abbildung 3 links repräsentiert, dass Thema $R180$ von Lektion $L2.1$ behandelt wird. In realen Szenarien ist jede Dokumenteinheit mit ca. 10 bis 100 solcher Interpretationen annotiert. Die Annotationen werden aus der Analyse des Dokumentmarkups und extern vorliegender Metadaten gewonnen.

Die Semantik von $\mathcal{ALC}$CTL [Wei08] definiert die Gültigkeit $M, s \models f$ einer $\mathcal{ALC}$CTL Formel f in einem Verifikationsmodell M an einer Stelle (d.h. einer Dokumenteinheit) s. In dem in Abbildung 3 skizzierten Modell M gilt beispielsweise

$$\begin{aligned} M, Home &\models MajorTopic \sqsubseteq \mathsf{EF}\ \exists addressedBy.\top \\ M, Home &\not\models MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.\top \end{aligned}$$

M erfüllt an der Stelle $Home$ Formel $MajorTopic \sqsubseteq \mathsf{EF}\ \exists addressedBy.\top$, da es für jedes der beiden Hauptthemen $R180$ und $R180\_handling$ von $Home$ einen Pfad gibt, auf dem es schließlich behandelt wird, nämlich den Pfad $(Home, L2, L2.1, ...)$ für Thema $R180$ und den Pfad $(Home, L3, L3.1, ...)$ für Thema $R180\_handling$.

M erfüllt an der Stelle $Home$ nicht Formel $MajorTopic \sqsubseteq \mathsf{AF}\ \exists addressedBy.\top$, da nicht jedes Hauptthema von $Home$ auf allen Pfaden irgendwann behandelt wird. Beispielsweise wird das Hauptthema $R180$ auf dem Pfad $(Home, L3, L3.1, L3.2, ...)$ nicht behandelt.

Definition 2 ($\mathcal{ALC}$CTL Model-Checking)

Für eine gegebene $\mathcal{ALC}$CTL Formel f und ein gegebenes Verifikationsmodell M mit einer endlichen Menge von Stellen S die Menge der Stellen $\{s \in S \mid M, s \models f\}$ zu bestimmen, an denen f gilt, bezeichnet man als $\mathcal{ALC}$CTL *Model-Checking*. □

5 Formale Eigenschaften und zentrale Ergebnisse

Die wichtigsten formalen Eigenschaften von $\mathcal{ALC}$CTL sind [Wei08]:

Satz 3 (Entscheidbarkeit von $\mathcal{ALC}$CTL Model-Checking)

$\mathcal{ALC}$CTL Model-Checking ist entscheidbar.

Beweis: Reduktion von $\mathcal{ALC}$CTL Model-Checking auf das entscheidbare CTL Model-Checking Problem. □

Die Reduktion auf das CTL Model-Checking Problem liefert zugleich einen ersten vollständigen und korrekten $\mathcal{ALC}$CTL Model-Checking Algorithmus, der aber eine exponentielle Laufzeit besitzt und deswegen nicht praktikabel ist.

Satz 4 (Laufzeitkomplexität von $\mathcal{ALC}$CTL Model-Checking)

Das $\mathcal{ALC}$CTL Model-Checking Problem ist in polynomieller Zeit bezüglich der Größe des Modells und der Formel lösbar.

Beweis: Geschichtete CTL-Reduktion von $\mathcal{ALC}$CTL-Formeln und "bottom-up" Aggregation der Zwischenergebnisse. □

Der Beweis der polynomiellen Laufzeitkomplexität definiert die Grundstruktur eines vollständigen, korrekten und effizienten Model-Checking Algorithmus für $\mathcal{ALC}$CTL.

Satz 5 ($\mathcal{ALC}$CTL Model-Checking Algorithmus)

Es gibt einen vollständigen und korrekten Algorithmus, der das $\mathcal{ALC}$CTL Model-Checking Problem in $\mathcal{O}(|f|\cdot|M|)$, also in linearer Zeit bezüglich der Formelgröße $|f|$ und der Modellgröße $|M|$ löst.

Beweis: Algorithmische Umsetzung der geschichteten CTL-Reduktion des $\mathcal{ALC}$CTL Model-Checking Problems und Laufzeitanalyse. □

Da CTL in $\mathcal{ALC}$CTL enthalten ist und CTL Model-Checking nicht schneller als in $\mathcal{O}(|f|\cdot|M|)$ lösbar ist [Sch03], ist der entwickelte $\mathcal{ALC}$CTL Model-Checking Algorithmus optimal. Obwohl die asymptotischen Laufzeitkomplexitäten identisch sind, unterscheiden sich die Laufzeiten von CTL Model-Checking und $\mathcal{ALC}$CTL Model-Checking in der Anwendung, denn die jeweiligen Modelle wachsen unterschiedlich schnell in der Größe des modellierten Systems.

Satz 6 (Zusammenhang zwischen Dokument- und Modellgröße)

Zwischen der Größe eines Dokuments $|d|$ und seines $\mathcal{ALC}$CTL Verifikationsmodells $|M|$ besteht der Zusammenhang $|M| \in \mathcal{O}(|d|^3)$.

Beweis: Abschätzung des Anwachsens der verschiedenen Bestandteile des Verifikationsmodells in der Größe des Dokuments. □

Im Fall von CTL Model-Checking wachsen Modelle typischerweise exponentiell in der Größe des modellierten Systems an [CGP02].

Folgerung 7 (Komplexität von $\mathcal{ALC}$CTL Model-Checking für Dokumente)

Die Laufzeit für die Verifikation von Dokumenten durch $\mathcal{ALC}$CTL Model-Checking ist in $\mathcal{O}(|f|\cdot|d|^3)$ bzgl. der Formelgröße $|f|$ und der Dokumentgröße $|d|$. □

Experimente mit Testreihen realitätsnaher Dokumente belegen, dass in typischen Szenarien die Verifikationszeit sogar nur linear oder quadratisch in der Größe des Dokuments wächst [Wei08]. Die reichhaltigeren Modellierungskonstrukte von $\mathcal{ALC}$CTL erlauben, Verifikationsmodelle kompakter zu repräsentieren als dies mit CTL-Methoden der Fall ist, was zu einem besseren Skalierungsverhalten in der untersuchten Anwendung führt. So benötigt der in Java implementierte $\mathcal{ALC}$CTL Model-Checking Algorithmus 0,24 Sekunden für die Prüfung von 10 Vorgaben an ein Web-Dokument mit 512 HTML-Seiten,

während der State-of-the-Art Model-Checker NuSMV [CCG+02] für die Verifikation der CTL-Repräsentation der Vorgaben 25,6 Sekunden benötigt. Die Verifikation pfadbasierter Vorgaben durch XML-Methoden wie XSLT und XPath scheitert auf Grund der kombinatorischen Explosion der zu verfolgenden Lesepfade bereits bei Dokumenten mit 250 Seiten, während mit $\mathcal{ALC}$CTL Model-Checking selbst Dokumente mit 5120 Seiten in weniger als 2 Sekunden verifiziert werden können. Mit den vorgeschlagenen Methoden wurden bisher unbekannte und z.T. kritische Fehler in bereits veröffentlichten technischen Dokumentationen lokalisiert [SJWF09, Wei08, WJF09].

6 Zusammenfassung

Es wurde ein neuer Ansatz zur automatischen Prüfung von inhaltlich-strukturellen Vorgaben an nicht linear aufgebaute Dokumente vorgestellt. Die automatisierte Überprüfung inhaltlich-struktureller Vorgaben reduziert den Aufwand in der Qualitätssicherung und hilft, Fehler zu vermeiden. Der Ansatz basiert auf der semantischen Modellierung von Dokumenten mit Beschreibungslogik, der Repräsentation von Vorgaben durch die neue temporale Beschreibungslogik $\mathcal{ALC}$CTL und der Verifikation der Vorgaben durch ein neues Model-Checking Verfahren. $\mathcal{ALC}$CTL erweitert bekannte Formalismen so, dass inhaltliche Zusammenhänge auch dann überprüft werden können, wenn wie bei Web-Dokumenten oder technischen Dokumentationen davon auszugehen ist, dass einzelne Abschnitte des Dokuments nicht in einer fest vorgegebenen Reihenfolge gelesen werden. Werden solche individuellen Lesepfade durch ein Dokument zugelassen, entstehen in der Regel exponentiell viele zu prüfende Konstellationen, so dass der Einsatz bisheriger maschineller Prüfverfahren bereits bei Dokumenten mit wenigen hundert Seiten zu inakzeptablen Laufzeiten führt. Mit $\mathcal{ALC}$CTL Model-Checking lassen sich selbst Dokumente von 5000 Seiten in wenigen Sekunden vollständig und exakt prüfen.

Die hier zusammengefasste Arbeit definiert den ersten Model-Checking Algorithmus für eine temporale Beschreibungslogik und stellt die erste Anwendung einer temporaler Beschreibungslogik für Verifikationsaufgaben dar. Die Ergebnisse werden derzeit im DFG-Projekt *Verdikt* [WJF09] vertieft und hinsichtlich Bediendbarkeit und Anwendbarkeit für neue Dokumentklassen und -formate weiterentwickelt. Die nachgewiesenen positiven Komplexitätseigenschaften und die etablierten Formalismen überlegene Ausdruckskraft von $\mathcal{ALC}$CTL bilden eine vielversprechende Grundlage für weitere Anwendungen im Bereich der Verifikation von Geschäftsprozessmodellen und von Modellen objektorientierter Software – Strukturen, in denen wie bei Dokumenten sowohl semantische Zusammenhänge als auch zeitliche Abläufe berücksichtigt werden müssen.

Literatur

[AF01] A. Artale und E. Franconi. A Survey of Temporal Extensions of Description Logics. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 30(1-4):171–210, 2001.

[BCM+03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi und P. Patel-Schneider, Hrsg. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003.

[CCG+02] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani und A. Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In *Proc. of Computer Aided Verification (CAV 02)*, Jgg. 2404 of *LNCS*. Springer, 2002.

[CGP02] E. M. Clarke, O. Grumberg und D. A. Peled. *Model Checking*, Kapitel 6: Symbolic Model Checking. MIT Press, 4.. Auflage, 2002.

[Eme90] E.A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, Hrsg., *Handbook of Theoretical Computer Science: Formal Models and Semantics*, Seiten 996–1072. Elsevier, 1990.

[HH06] M. Han und C. Hofmeister. Modeling and verification of adaptive navigation in web applications. In *Proc. of 6th International Conference on Web Engineering*, Seiten 329 – 336, Palo Alto, CA, USA, 2006. ACM Press.

[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.

[NCEF02] C. Nentwich, L. Capra, W. Emmerich und A. Finkelstein. xlinkit: a consistency checking and smart link generation service. *ACM Transactions on Internet Technology (TOIT)*, 2(2):151–185, 2002.

[Sch03] P. Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic*, Jgg. 4, Seiten 393–436. King's College Publications, London, 2003.

[Sch04] J. Scheffczyk. *Consistent Document Engineering*. Dissertation, University of Armed Forces Munich, 2004.

[SFC98] P. D. Stotts, R. Furuta und C. R. Cabarrus. Hyperdocuments as Automata: Verification of Trace-Based Browsing Properties by Model Checking. *Information Systems*, 16(1):1–30, 1998.

[SJWF09] C. Schönberg, M. Jakšić, F. Weitl und B. Freitag. Verification of Web-Content: A Case Study on Technical Documentation. In *Proc. of 5th International Workshop on Automated Specification and Verification of Web Systems*, Linz, Austria, 2009.

[Wei08] F. Weitl. *Document Verification with Temporal Description Logics*. Dissertation, University of Passau, 2008.

[WJF09] F. Weitl, M. Jakšić und B. Freitag. Towards the Automated Verification of Semi-structured Documents. *J. of Data & Knowledge Engineering*, 68(3):292–317, 2009.

Franz Weitl wurde am 23.12.1972 in Altötting geboren. Nach einer Ausbildung und Tätigkeit als mathematisch-technischer Assistent am Forschungszentrum Jülich studierte er Informatik an der Universität Passau. Dort arbeitete er von 2001 bis 2004 als wissenschaftlicher Mitarbeiter am Institut für Informationssysteme und Softwaretechnik. Seit 2004 ist er wissenschaftlicher Mitarbeiter am Lehrstuhl für Informationsmanagement der Universität Passau. Als Postdoc beschäftigt er sich mit der Weiterentwicklung formaler, logik-basierter Methoden für praktische Anwendungen in Informationssystemen. Der DAAD hat ihm ein Postdoc-Stipendium vom Sep. 2009 bis Aug. 2010 am National Institute of Informatics in Tokio zugesprochen.