Institut für Informatik,
Lehr- und Forschungseinheit
für Programmierung und Softwaretechnik

der Ludwig-Maximilians-Universität München

Diplomarbeit

Modellprüfung von UML-Zustandsmaschinen und UML-Kollaborationen in SAL

Alexander Harhurin

Aufgabensteller:	Prof. Dr. Martin Wirsing
Betreuer:	Dr. Alexander Knapp
Abgabetermin:	21. Juni 2005

Zusammenfassung

Wir beschreiben eine Komponente des Prototyp-Werkzeugs HUGO/RT, das entworfen wurde, um den Einsatz von UML-Zustandsmaschinen in Software-Projekten zu erleichtern. HUGO/RT ist nun in der Lage Zustandsmaschinen mit Hilfe des Modellüberprüfers SAL zu verifizieren. Die Modellüberprüfung wurde in erster Linie entworfen, um zu verifizieren ob Interaktionen, die in bestimmten Kollaborationen ausgedrückt werden, auch tatsächlich durch eine Menge von Zustandsmaschinen realisiert werden können. Eine Prüfung auf mögliche Verklemmungen kann durchgeführt werden. Die vom SAL produzierten Gegenbeispiele werden in einer vereinfachten UML-Form ausgegeben.

Inhaltsverzeichnis

Kapitel 1

Einführung

Die Software-Entwicklung befasst sich mit technischen Aspekten der Herstellung und Wartung von Software-Systemen. Dabei bedient sich die Software-Entwicklung heute einer Vielzahl von Methoden und Techniken. Neben natürlichsprachlicher Dokumentation finden vor allem die graphische Modellierungssprache UML sowie die formale Sprache OCL Verwendung. Da die Semantik dieser Sprachen formal definiert ist, schaffen sie eine Basis zur Verifikation der Korrektheit verschiedener Eigenschaften von Modellen.

Die Modellüberprüfung findet in der Software-Entwicklung eine breite Anwendung. Es ist durchaus möglich, Programme direkt zu überprüfen (zu testen). Allerdings scheint es sinnvoller, die Überprüfung auf einem hohen Abstraktionsniveau des Modells durchzuführen. Die Modellüberprüfung hat mehrere Vorteile gegenüber anderen Verifikationstechniken. Erstens, sie ist komplett automatisch, erfordert nur die Eingabe eines Modells und einer Beweisverpflichtung. Zweitens, sie führt eine erschöpfende Suche durch und kann beweisen, dass ein Modell korrekt ist. Drittens, wenn ein Fehler gefunden wird, kann sie ein Gegenbeispiel produzieren, mit dessen Hilfe der Fehler leicht zu finden ist.

Ein Modellüberprüfer verifiziert die Korrektheit der Semantik eines Modells. Er überprüft, ob eine der semantisch erlaubten Konfigurationen die Beweisverpflichtung verletzt. Die größten Probleme eines Modellüberprüfers sind die Antwortzeit und der Speicherverbrauch.

Es gibt mehrere Arten von Modellüberprüfern. Die expliziten Modellüberprüfer überprüfen alle möglichen Pfade auf der Suche nach der Verletzung einer Beweisverpflichtung. SPIN ist heute der bekannteste Vertreter dieser Klasse. Die symbolischen Modellüberprüfer verwenden BDDs zur Zustandsrepräsentation. Bei einer kleinen Anzahl von Variablen erzielen sie gute Ergebnisse. SAL ist ein Vertreter dieser Klasse. Eine andere symbolische Technik ist die begrenzte Modellüberprüfung, in der aussagelogische Formeln zum Fehlerentdecken verwendet werden. Die Suche nach einem Fehler wird auf das SAT-Problem zurückgeführt, dabei können hocheffiziente SAT-Beweiser verwendet werden. Ein begrenz-

ter Modellüberprüfer durchsucht nicht alle möglichen Pfade, sondern nur diejenigen, die kürzer als eine gegebene Länge sind. Damit kann er einen relativ kurzen Pfad schnell verifizieren, scheitert aber meistens bei größeren Modellen.

Diese Arbeit stellt eine Komponente des Prototyp-Werkzeugs HUGO/RT vor, die automatisch UML-Zustandsmaschinen und UML-Kollaborationen anhand des Modellüberprüfers SAL verifiziert. Es wird überprüft, ob die Modelleigenschaften, spezifiziert in UML-Interaktionen und OCL-Constraints korrekt sind, das heißt, ob das beschriebene Modellverhalten von den UML-Zustandsautomaten realisiert werden kann.

SAL erweitert das Werkzeug HUGO/RT um den symbolischen und den begrenzten Modellüberprüfer. Der unendliche begrenzte Modellüberprüfer ist in der Lage, mit unendlichen Typen zu arbeiten, was bis jetzt die anderen Modellüberprüfer nicht konnten. Das wichtigste Ziel der Anwendung der SAL-Modellüberprüfer ist, unendliche UML-Modelle (mit unendlichen Typen) verifizieren zu können.

Kapitel 2

Zusammenfassung und Ausblick

2.1 Zusammenfassung

Die UML-Sprache ist ein sehr umfangreicher Formalismus zur Beschreibung von Zustandsautomaten. Bei der Verifikation dieser Zustandsautomaten ist es sinnvoll Werkzeuge einzusetzen. HUGO/RT verwaltet verschiedene Modellüberprüfer zur Verifikation von objektorientierten Modellen.

Es ist allgemein bekannt, dass Fehler in den Entwurfs- und Modellierungsphasen eines Software-Projekts die teuersten sind. Das Einsetzen des HUGO/RT auf dem abstrakten Niveau eines Modells hält den Aufwand für die Fehlersuche gering und vermeidet einen exponentiellen Kostenanstieg in den späteren Projektphasen.

Die Übersetzung von UML-Modellen in SAL-Kontexte wurde im Rahmen des Prototyp-Werkzeugs HUGO/RT implementiert. Das Werkzeug kann nun UML-Zustandsmaschinen mit Hilfe eines der SAL-Modellüberprüfer automatisch verifizieren, d. h., überprüfen, ob Interaktionen, die in bestimmten Kollaborationen ausgedrückt werden, auch tatsächlich durch eine Menge von Zustandsmaschinen realisiert werden können. Ergänzend kann eine Prüfung auf mögliche Verklemmungen durchgeführt werden. Die Ergebnisse werden in einer vereinfachten UML-Syntax an den Benutzer zurückgegeben.

Die Menge von UML-Zustandsautomaten und eine UML-Kollaboration werden in einen SAL-Kontext mit einer Beweisverpflichtung übersetzt, anschließend wird einer der SAL-Modellüberprüfer aufgerufen, um die Verifikation der Beweisverpflichtung durchzuführen. Bei der Übersetzung wird die HUGO-interne Sprache SMILE verwendet. Sie beschreibt die Semantik der UML-Zustandsautomaten mit einfachen Konstrukten. Die SAL-Abbildung unterstützt alle Elemente der SMILE-Sprache bis auf die Stoppuhr-Elemente. Aus diesem Grund können keine zeitgesteuerten Zustandsautomaten in SAL verifiziert werden.

SAL erweitert das Werkzeug HUGO/RT um die Familie der symbolischen Modellüberprü-

fer. Das wichtigste Ziel der Arbeit war, den unendlichen SAL-Modellüberprüfer einsetzen zu können. Er ist in der Lage, mit unendlichen Typen zu arbeiten, was bis jetzt die anderen Modellüberprüfer nicht konnten. Mit den unbegrenzten Integer- und Realtypen kann man unendliche Zustandssysteme, solche wie hybride oder zeitgesteuerte Automaten und andere Formalismen des kontinuierlichen oder echtzeitlichen Verhaltens, darstellen. Eine weitere Verbesserung brachten die SAL-Datentypen. Mit ihrer Hilfe können UML-Ereignisse semantisch sauberer implementiert werden. Ein Ereignis und seine Parameter werden auf eine einzige SAL-Konstante abgebildet.

Allerdings wurde während der Implementierung festgestellt, dass der unendliche SAL-Modellüberprüfer mit den Datentypen noch nicht kompatibel ist. Der Verzicht auf die Datentypen löst das Problem nicht. Der unendliche begrenzte Modellüberprüfer ist nicht in der Lage, Systeme mit über 40 Zustandsübergängen zu behandeln. Deswegen muss man auf die nächste Version der SAL-Modellüberprüfer warten, um in den Genuss der Unendlichkeit zu kommen.

Mehrere Fallstudien bestätigten die Behauptung, dass die symbolische Modellüberprüfung eher für Hardware als für Software geeignet ist. Schon bei einer relativ kleinen Anzahl von logischen Variablen waren die Antwortzeiten der symbolischen Modellüberprüfung nicht zufriedenstellend. Der schnellere begrenzte Modellüberprüfer scheiterte an Zustandsautomaten, die länger als 40 Zustände waren. Und SMILE-Maschinen stellen in der Regel deutlich längere Zustandsautomaten dar.

2.2 Ausblick

Die wichtigste Verbesserung der SAL-Modellüberprüfung hängt leider nicht von den Entwicklern des HOGO/RT ab. Man muss abwarten, bis die angekündigten unendlichen Modellüberprüfer tatsächlich von Computer Science Laboratory realisiert werden.

Das umfangreiche SAL-Typensystem macht es möglich die UML-Typen `real` und `String` zu unterstützen. Diese Erweiterung hängt in erster Linie von der Evolution der SMILE-Sprache ab. Auch die Beschränkungen, die HUGO/RT zur Zeit noch hat (*Change*-Ereignisse, „*deep history*“ und synchrone Zustände, Signalhierarchie und interne Zustandsübergänge werden nicht unterstützt), müssen bei der Übersetzung von UML in SMILE beseitigt werden und betreffen die Übersetzung von SMILE in SAL nicht. Dasselbe gilt für die Umstellung auf UML 2.0. Das zeigt den Vorteil der Verwendung der Sprache SMILE deutlich.

Die SAL-Sprache zur Deklaration von Beweisverpflichtungen ermöglicht eine Verifikation von deutlich komplizierteren OCL-Constraints. Es ist z. B. sinnvoll die partielle Korrektheit einer Operation verifizieren zu können, d. h., man muss zeigen, dass die Vorbedingung im Zustand vor der Ausführung und die Nachbedingung im Zustand nach der Ausführung

gelten. Das Problem dabei ist, dass ein UML-Zustand auf eine Folge von internen SAL-Zuständen abgebildet wird, was die Festlegung der Vor- und Nachzustände einer Operation erschwert. Der LTL-Operator `UNTIL` kann bei der Lösung dieses Problems hilfreich sein. Weiterhin kann man in SAL Eigenschaften einer Vererbungs- oder Abstraktionsbeziehung verifizieren, was vom HUGO/RT noch nicht unterstützt wird.

Es ist von Computer Science Laboratory geplant, SAL mit PVS in eine gemeinsame Umgebung zu integrieren, sodass es möglich wird, eine SAL- in eine PVS-Spezifikation und umgekehrt zu übersetzen. Es kann aber nicht mit einer schnellen Realisierung dieses Vorhabens gerechnet werden. Deswegen ist es durchaus sinnvoll, im Rahmen des HUGO/RT Projekts eine Übersetzung von SAL-Modellen in PVS-Spezifikationen zu implementieren. Eine PVS-Spezifikation besteht aus parametrisierten Theorien, die Annahmen, Definitionen, Axiome und Theoreme enthalten können. Auf den ersten Blick scheint es, dass alle in der UML2SAL-Übersetzung verwendeten SAL-Typen, -Ausdrücke und -Anweisungen direkte Abbildungen in der PVS-Sprache haben. SAL-Module werden nach PVS-Theorien abgebildet. So kann man UML-Modelle anhand einer einfachen Transformation mit einem mächtigen und hocheffizienten Theorembeweiser verifizieren.