Model Checking im Automotivbereich

Hans-Werner Wiesbrock, Heiko Dörr

Daimler Chrysler AG
Alt-Moabit 96 A
10559 Berlin
{ hans-werner.wiesbrock | heiko.doerr@
DaimlerChrysler.com }

Hans Jürgen Holberg

OSC-ES AG
Industriestraße 11
26121 Oldenburg
holberg@osc-es.de

Abstract: Um die hohen Qualitätsansprüche an Software im Fahrzeug zu sichern, wird der Einsatz eines Model Checkers empfohlen. Es werden Erfahrungen aus der Evaluation heute verfügbarer Werkzeuge präsentiert. Der Einsatz dieser Technologie wird durch die inhärente Schwierigkeit, die geforderten Eigenschaften an das Modell mathematisch zu präzisieren, erschwert. Um den Benutzer darin zu unterstützen wird eine Klassifikation geeigneter Mustereigenschaften zusammen mit einem Konzept eines Pattern Wizard vorgestellt, so dass der Anwender nach konkreten Handlungsanweisungen zur geeigneten Formulierung seiner Abfragen kommt.

1 Einleitung

Höherer Fahr- und Bedien-Komfort und innovative Lösungen zu neuen Kundenwünschen führen zu einem verstärkten Einsatz elektronischer und mechatronischer Bauteile in Automobilen. Seit Ende der 90er Jahre erfolgt die Entwicklung der softwarebasierten, eingebetteten Regelsysteme im Automotivbereich zunehmend Modell-basiert, (vgl. [Be00]). Diese Modelle sind in der Regel ausführbar, so dass bereits in einem frühen Entwicklungsstadium die Funktionalität des Zielsystems evaluiert und konzeptionelle Schwachstellen ohne große Kosten behoben werden können. Aus den entstehenden Modellen wird anschließend der C-Code für die Steuergeräte generiert.

Da die Funktionalität der eingebetteten Systemen vermehrt in den Bereich funktionssicherer Anwendungen geht, müssen verstärkt Maßnahmen zur Absicherung der erforderlichen Qualität durchgeführt werden. Wichtige Methoden sind unter anderem Reviews und Tests von Modellen (z.B. [Co99]). Dieses Techniken leisten einen wichtigen Beitrag zur Aufdeckung von Entwicklungsfehlern. Die Abwesenheit von Fehlern können sie jedoch nicht nachweisen. Gerade bei komplexen Modellen, deren Funktionssicherheit untersucht werden soll, besteht die Gefahr, dass kritische Situationen nicht entsprechend erkannt werden.

Zur vollständigen Absicherung von Modellen muss daher eine andere Technik, das Model Checking, herangezogen werden [CGP99]. Dieses Verfahren ist seit einigen Jahren bereits in der Verifikation von Hardware-Komponenten erfolgreich im Einsatz. Die verifizierten Komponenten sind hochkomplex, aber aus einfachen Bausteinen zusammengesetzt. So können mathematische Methoden zum Nachweis von Eigenschaften angewendet werden.

Schlechte Handhabung und Skalierbarkeit haben in der Vergangenheit häufig die Verifikation durch Model Checking erschwert. Doch wurden in den letzten Jahren neue leis-

tungsstarke Werkzeuge entwickelt, die einfach und bequem zu bedienen sind. So gibt es beispielsweise seit Mitte 2001 die integrierte Verifikationsumgebung *ModelChecker und ModelCertifier* für *Statemate* (Magnum 3.x) auf dem Markt, mit deren Hilfe komplexe Statecharts und auch ganze Statemate-Modelle auf ihre korrekte Umsetzung von Anforderungen hin verifiziert werden können.
Für die Verifikation von StateFlow-Modellen wurde ein Prototyp für Matlab/Simulink R12, Stateflow V4.0 von der Firma OSC-ES AG, Oldenburg, fertig gestellt, der zur Zeit von der DaimlerChrysler AG evaluiert wird.

2 Model Checking

Ausgehend von dem zu verifizierenden Modell und einer spezifizierten Beweisaufgabe wird durch den Model Checker das Modell vollständig auf diese zu beweisende Eigenschaft hin untersucht. Die zu beweisende Eigenschaft wird über alle möglichen Zustände des Systems geprüft. Dieses Ausschöpfen aller Möglichkeiten unterscheidet diese Technologie von Testen.
Das Model Check Verfahren überprüft, welche erreichbaren Zustände des Modells die temporallogische Spezifikation erfüllen. Falls alle Zustände die Spezifikation erfüllen, so erfüllt das Modell die Spezifikation.

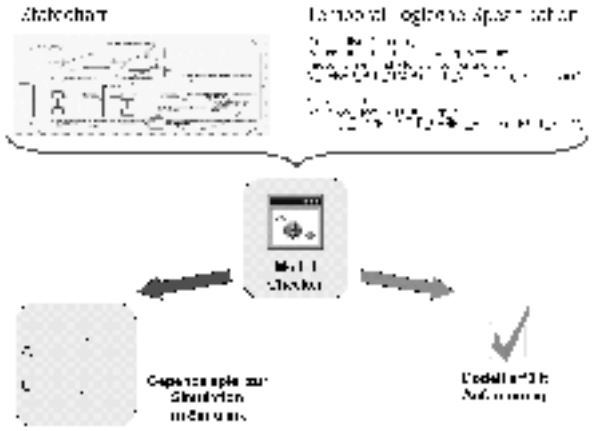

Abbildung 1 Model Checking

Der Zeit und Platzbedarf des expliziten Verfahrens wächst linear mit der Größe des Modells und der Länge der temporallogischen Spezifikation, wobei die Anzahl der Zustände eines Modells exponentiell mit der Anzahl der parallelen Systemkomponenten wächst. In der Literatur ist dieses Problem als „State Explosion Problem“ bekannt. Aus dem genannten Grund sind mit dem expliziten Verfahren keine Nachweise von Eigenschaften für industrierelevante Modelle möglich, da diese einen zu großen Zustandsraum aufweisen.
Im genannten Ansatz werden die Zustände des Modells explizit aufgezählt und anschließend überprüft, welche Teilformeln der temporallogischen Spezifikation in welchen Zuständen gelten. Beim symbolischen Model Checking wird jeder Teilformel der tempo-

rallogischen Spezifikation eine Menge zugeordnet, die jeweils die Zustände repräsentiert, in der die Teilformel gilt. Diese Mengen können sehr effizient dargestellt werden, indem die booleschen charakteristischen Funktionen der Mengen durch BDD (Binary Decision Diagrams) Darstellungen repräsentiert werden.

3 Praxiserfahrungen

Zur Evaluation der Leistungsfähigkeit und zur Ermittlung der praktischen Verfügbarkeit wurden bei DaimlerChrysler mehrere Statechart-Modelle aus verschiedenen Entwicklungsprojekten ausgewählt. Die Modelle waren mit Statemate oder mit StateFlow erstellt. Zu den einzelnen Modellen wurden Eigenschaften identifiziert, die nachgewiesen werden sollten. Neben dem konkreten Nachweis der Eigenschaften wurden folgende allgemeine Erfahrungen festgestellt.

Die zutreffende Formalisierung einer umgangssprachlichen Eigenschaft ist eine nicht zu unterschätzende Aufgabenstellung. Da eine Verifikationsaufgabe, also ein mathematisch korrekter Nachweis, davon abhängt, muss um so mehr darauf geachtet werden, dass die Formulierung einer nachzuweisenden Eigenschaft durch einen temporal-logischen Ausdruck die intendierte Eigenschaft tatsächlich abbildet. Dennoch wurden bei der Evaluation des Model Checkers wiederholt auf, dass Formalisierungen fehlerhaft vorgenommen. Diese Fehler wurden meistens dadurch offenbar, dass die mit dem Model Checker geführten Nachweise fehlschlugen. Im Zuge der Ursachenermittlung bemerkt, dass das Modell korrekt, aber die Formalisierung eine andere, als die intendierte Eigenschaft beschrieb. Diese fehlerhaft falschen Nachweise sind für die Aufgabenstellung der Verifikation vergleichsweise unkritisch.

Eine große Gefahr besteht jedoch in fehlerhaft richtigen Nachweisen, also solchen, in denen der Model Checker eine temporal-logische Formel nachweist, diese jedoch die falsche Eigenschaft beschreibt. In diesem Fall werden kritische Fehler aufgrund von „Bedienungsfehlern“ beim Model Checking nicht erkannt. Eine Absicherung der Formalisierung ist also beim Einsatz eines Model Checkers dringend angeraten. Die Verifikation von Modellen nur durch spezialisierte Mitarbeiter zusammen mit den eigentlichen Funktionsentwicklern kann beispielsweise eine sinnvolle organisatorische Maßnahme sein.

Die Bearbeitung der Beispiele hat weiterhin gezeigt, dass die volle Ausdrucksmächtigkeit einer temporalen Logik häufig nicht benötigt wird. Ausgehend von einer Stufung Aussagen-, Prädikaten- und Modallogik ist festzustellen, dass die Formulierung von Invarianten oft ausreichend ist. Viele Eigenschaften konnten durch Ausdrücke, die keine zeitlichen Abhängigkeiten beschreiben, sondern ausschließlich Aussagen über alle Zeitpunkte treffen, erfasst werden. Hier sind die Komplexität und damit die Gefahr von fehlerhaft richtigen Nachweisen in der Regel reduziert.

Eine weitere Absicherung der Nachweise besteht in der Verwendung vordefinierter Aussagen-Muster. Diese Pattern erfassen für eine Anwendung oder Domäne typische Aussagen und erlauben durch die wiederholte Anwendung einen sichereren Gebrauch von Formeln. In der Bearbeitung der Beispiele wurde deutlich, dass eine eingeschränkte Menge von Mustern zur Formulierung der relevanten Eigenschaften ausreichten. Die

Fehlerquelle falscher Formalisierung konnte so ausgeschlossen werden. Die Wirksamkeit der Pattern motivierte die in Kapitel 4 vorgestellten Arbeiten.

4 Zielgerichteter Einsatz von Pattern

Die korrekte Formulierung der Beweisaufgabe hängt wesentlich von der korrekten Interpretation der zu beweisenden informellen Eigenschaft ab. Drei verschiedene Interpretationen lassen sich für die informelle Aussage 'Immer P dann auch Q' finden:

- aussagen-logisch Wenn P dann Q
- temporal-logisch Wenn in einem Lauf Ereignis P auftritt, muss auch Ereignis Q eintreten
- kausal-logisch Das Ereignis Q muss in einem Lauf zeitlich nach dem Ereignis P auftreten

Um nun die Spezifikation von Beweisaufgaben sicher zu gestalten, wurden Pattern für formalisierte Eigenschaften entwickelt, mit denen Beweisverpflichtungen korrekt formuliert werden können ([IO01], [DAC97]).
Je nach Beweisverpflichtung und Kontext sind verschiedene Eigenschaftsmuster wünschenswert. Eine Klassifikation der verschiedenen Pattern hilft, das geeignete Pattern für eine anstehende Beweisverpflichtung zu identifizieren [Bi00]. Weiterhin lassen sich vorhandene Sammlungen auf Vollständigkeit und Konsistenz überprüfen. In der Praxis zeigte sich, dass Pattern für Observer Automaten und Invarianzprüfung zur Formulierung der Anforderungen genügen. Zudem besitzen sie eine deutlich bessere Laufzeit und Speicher Performanz. In der Klassifikation wurden deshalb diese wichtigen Muster erfasst (siehe Abbildung 2).

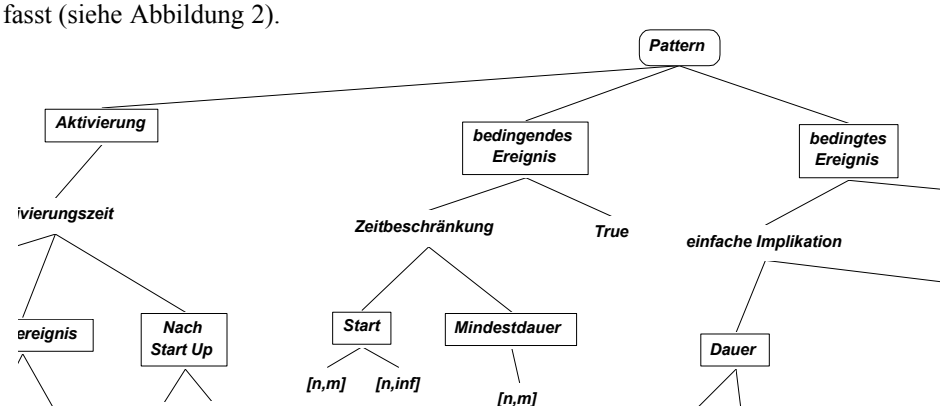

Abbildung 2 Pattern Klassifikation (Ausschnitt)

Als wichtigste Gruppe von Abfragen wurden Beweisverpflichtungen zu Implikationen erkannt. Das Eintreffen eines Ereignisse im allgemeineren Sinne, also bestimmte markante Geschehnisse im Simulationslauf des Modells, beschränkt das Auftreten weiterer Ereignisse. Damit erhalten wir die Grundunterscheidung beteiligter Ereignisse in bedingende, deren Eintreffen vorausgesetzt wird, und bedingte, über deren Auftreten dann weitere Aussagen und Einschränkungen gemacht werden. Spezifische Eigenschaften

dieser Ereignisse wie: Ihre Dauer, zeitliche Reihenfolge zum implizierenden Vorgänger etc. dienen dann zur weiteren Einordnung des Pattern.
Zusätzlich zu diesen Merkmalen der beteiligten Ereignisse ist noch festzulegen, wann die geforderte Eigenschaft geprüft werden soll, (Aktivierung). Dies kann unmittelbar nach dem Beginn des Laufes starten, oder auch erst nach einer gewissen Initialisierungszeit, immer wieder überprüft oder nur einmalig gecheckt werden.
Ausgehend von der obigen Klassifikation der Eigenschaftsmuster wurde ein Wizard konzipiert und konkrete Handlungsanweisungen abgeleitet, welcher den Benutzer bei der Formulierung seiner Beweisaufgabe über Pattern unterstützt. Durch einfache Fragen nach der Anzahl der beteiligten Ereignisse, deren Dauer und Reihenfolge, wird der Benutzer immer näher zu den geeigneten Abfragen geleitet.

5 Ausblick und offene Probleme

Model Checking im industriellen Einsatz ist heute möglich. Verbesserungspotenzial hinsichtlich der zuverlässigeren Formalisierung kann durch eine Klassifikation von Pattern genutzt werden, die künftig auch in die marktgängigen Verifikationsumgebungen aufgenommen werden sollten.

Wir bedanken uns bei den Herren M. Finkbohner und A. Nägele für die zahlreichen und instruktiven Diskussionen, ohne die diese Arbeit nicht hätte entstehen können.

Literatur und Referenzen

[Be00] P. Bechberger: Modellbasierte Software-Entwicklung für Steuergeräte Automobiltechnische. Zeitschrift/Motortechnische Zeitschrift (ATZ/MTZ), Sonderausgabe 'Automotive Electronics', Jan. 2000

[Bi00] F. Bitsch: Classification of Safety Requirements for Formal Verification of Software Models of Industrial Automation Systems. Proc. of 13th Intern. Conf. on Software Engineering and Their Application 2000, CNAM-Paris

[CGP99] E.M. Clarke, O.Grumberg, D.A Peled: Model Checking. MIT Press, Cambridge, MA, 1999

[Co99] Conrad, M.;H. Dörr, I. Fey, A.Yap: Model Based Generation and Structured Representation of Test Scenarios. Proc. of the Workshop on Software-Embedded Systems Testing WSEST '99), Maryland, USA, 1999.

[DAC97] M. Dwyer, G. Avrun, J. Corbett A System of Specification-Patterns http://www.cis.ksu.edu/santos/spec-patterns, '97

[IO01] I-Logix, OSC: Pattern Library, User Guide zum Statemate MAGNUM Addin 'Certifier'. Andover, Oldenburg 01