Eine Evaluationssuite zur schnellen Bewertung von Matlab/Simulink-Modelcheckern

Jacob Palczynski Bastian Schlich
Stefan Kowalewski
{palczynski, schlich, kowalewski}@informatik.rwth-aachen.de

Abstract:
Der Beitrag stellt eine Evaluationssuite für Modelchecker für Matlab/Simulink vor. Die Suite besteht aus einer Simulink-Bibliothek, einer Sammlung von Test-Modellen und einem Vorgehensmodell. Sie dient zur schnellen Beurteilung der Eignung von Matlab/Simulink-Modelcheckern für konkrete Anforderungen, bzw. zum Vergleich der Leistungsfähigkeit verschiedener Modelchecker. Um erste Ergebnisse zur Anwendbarkeit zu erhalten, wurde die Suite selbst anhand zweier existierender Modelchecker erprobt.

1 Einführung

Im Zuge des zunehmenden Einsatzes von Matlab/Simulink als Software-Entwicklungswerkzeug in der Automobilindustrie wird auch der Einsatz von Modelchecking [CGP99] zur Fehlersuche in den Modellen interessant. Unternehmen, die entsprechende, kommerziell verfügbare Werkzeuge ausprobieren wollten, berichten aber, dass sich der Anwendungsaufwand auf der Basis von Vertriebspräsentationen und Herstellerangaben kaum abschätzen lässt. Der Grund ist, dass sich erst im praktischen Einsatz zeigt, welche Anpassungen und Umformulierungen der Modelle für eine erfolgreiche Verarbeitung durch das Werkzeug wirklich notwendig sind. Um diese Informationen schneller verfügbar zu machen und so eine Eignungsbewertung von in Betracht gezogenen Modelcheckern für Matlab/Simulink zu ermöglichen, wurde auf Anfrage und in Zusammenarbeit mit einem Unternehmen aus der Automobilindustrie die hier vorgestellte Evaluierungssuite entwickelt.
Ein Grund für die beschriebene Problematik ist, dass die verfügbaren Matlab/Simulink-Modelchecker relativ neue Entwicklungen sind und noch nicht beliebige Modelle in jedweder Ausprägung verarbeiten können. Vielmehr muss man davon ausgehen, dass einzelne Simulink-Sprachelemente oder bestimmte Verwendungen und Kombinationen nicht angenommen werden. Häufig lässt sich dann zwar eine semantikerhaltende Umformulierung finden, die der Modelchecker versteht. Da dies aber mit dem erwähnten, schlecht abschätzbaren Aufwand verbunden ist, besteht ein wichtiges Evaluationskriterium darin, welche Syntax unmittelbar vom Modelchecker verarbeitet werden kann.
Die zweite interessante Frage ist die nach der Effizienz des Modelcheckers, mit anderen Worten, wie große Modelle in welcher Zeit und mit welchem Speicheraufwand verifiziert werden können. Unsere Evaluationssuite adressiert beide Aspekte. Im Hinblick auf

die Effizienz werden skalierbare Beispiele bereitgestellt, mit denen der Anwender die Leistungsfähigkeit des Modelcheckers ausloten kann.
Die Evaluierungssuite ist modular aufgebaut und kann durch Hinzufügen von weiteren Simulink-Modellen leicht erweitert werden. Die zur Suite gehörenden Modelle und eine detaillierte Beschreibung findet man in [Pal05]. Der Rest des Beitrags enthält eine kurze Vorstellung der technischen Realisierung und berichtet über erste Anwendungserfahrungen bei der Erprobung der Suite an zwei kommerziell verfügbaren Modelcheckern. Wir schließen mit einem Ausblick auf mögliche Erweiterungen.

2 Evaluationssuite

Die Evaluationssuite besteht aus einer Blockbibliothek, den Modellen und einem Vorgehensmodell. Die durch Subsysteme modularisierte Blockbibliothek enthält konfigurierbare Simulink-Subsysteme mit eigenen Maskendialogen und bildet die Grundlage der Suite. Mit Hilfe dieser Bibliothek kann man schnell unterschiedlich parametrisierte Modelle für die Suite erstellen und bestehende anpassen (siehe Abb. 1).

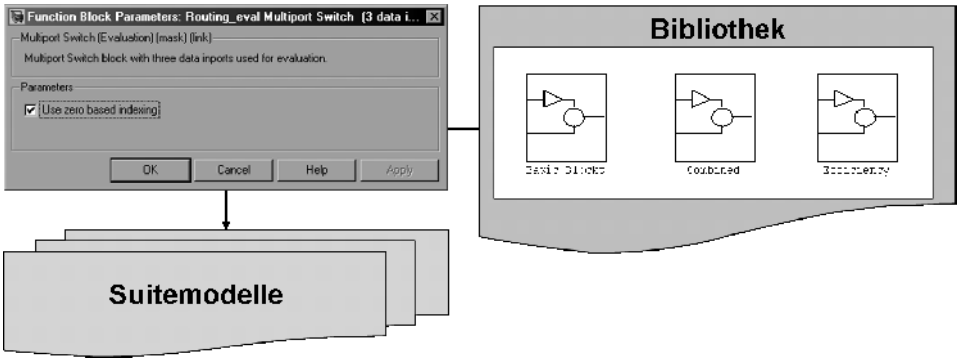

Abbildung 1: Die Subsysteme werden aus der Bibliothek in ein Modell eingebunden und über Maskendialoge konfiguriert.

Aus den Subsystemen werden die Modelle für die Evaluationssuite erstellt, die als Eingabe für den zu bewertenden Modelchecker dienen. Die Suite konzentriert sich auf die Fragen, welche syntaktischen Konstrukte der Modelchecker akzeptiert und wie effizient er arbeitet. Dementsprechend werden die Modelle in ein Syntax- und in ein Effizienzmodul eingeteilt (Abb. 2), die jeweils aus weiteren Submodulen bestehen.

Das Syntaxmodul (ca. 300 Modelle) umfasst drei Submodule, mit denen die Akzeptanz (1) von Blöcken und Ausprägungen, die sich durch die Parametrisierung ergeben, (2) von Datentypen und (3) der möglichen Arten, Funktionen zu implementieren (Simulink-Diagramme, Stateflow-Charts, S-Functions), untersucht wird.
Ein Beispiel für die erste Gruppe sind die Modelle zum Sum-Block. Mögliche Ausprägungen dieses Blocks ergeben sich aus der Anzahl der Eingänge, der jeweiligen Operation

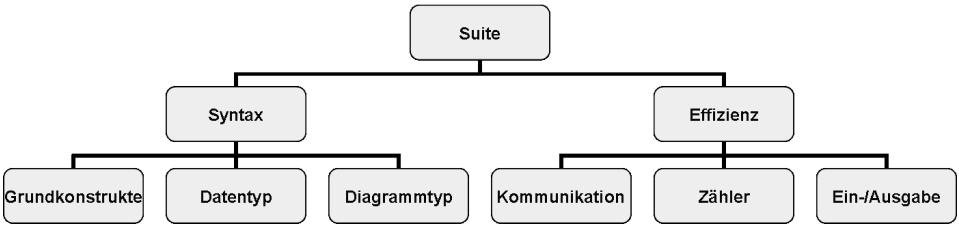

Abbildung 2: Struktur der Evaluationssuite.

jedes Eingangs ($+$, $-$ und | als Platzhalter bei runden Summenblöcken) und der Form des Blocks (rechteckig oder rund). Sum-Blöcke mit einem Eingang summieren, bzw. subtrahieren die Komponenten eines vektoriellen Eingangs, die mit mehreren Eingängen wenden die definierten Operationen auf die Eingänge an. Die Standardausprägung ist ein rechteckiger Block mit zwei $+$-Eingängen. Daraus ergeben sich drei Mengen von Ausprägungen, mit einem Eingang, mit zwei und mit mehr als zwei Eingängen. In der Bibliothek sind Subsysteme mit Sum-Blöcken mit einem, zwei und drei Eingängen hinterlegt. Die Parameter Operationen und Form Können über den Maskendialog eingestellt werden; die Suite enthält entsprechende Modelle.

Das Effizienz-Modul (74 Modelle) stellt drei Aufgabenstellungen (kommunizierende Prozesse, Zähler, Auswahl aus Eingängen) zur Verfügung, die auf verschiedene Arten skalierbar sind. Zu jeder Aufgabe gehören Spezifikationen, die die Modelle erfüllen.
Eine dieser Aufgaben, das Zähler-Submodul, stellen wir exemplarisch vor. Bei den Modellen dieses Submoduls wird eine Variable von einem Anfangswert a bis zu einem Endwert b inkrementiert. In der Bibliothek liegen dafür drei verschiedene Implementierungen vor, eine besteht aus einem „reinen“ Simulink-Subsystem, eine aus einem Stateflow-Chart und eine aus einer in C geschriebenen S-Function. Drei Parameter können bei der Erstellung eines Modells gesetzt werden, neben a und b das Inkrement i.
Im Zähler-Submodul liegen jeweils 17 Modelle vor, die auf der Simulink- und auf der Stateflow-Implementierung basieren. Die Parameter sind mit $a = 0$, $b = 4^n$ mit $n \in \{0, \ldots, 16\}$ und $i = 1$ belegt. Der bei der Verifikation des Zählermoduls zu durchsuchende Zustandsraum kann also über die Wahl von n eingestellt werden. Bei Bedarf können aus den Bibliotheksmodellen weitere Modellvariationen erstellt und die Parameter über den Maskendialog passend gewählt werden.
Die Spezifikation für das Zähler-Submodul lautet: *Es existiert ein Lauf des Systems, so dass end immer 1 ist, bis die Variable counter die Grenze b erreicht*, in CTL: $\mathsf{E}\,(end = 1\ \mathsf{U}\ counter = b)$. Dabei ist *end* eine boolesche Variable im Modell, die auf 1 gesetzt wird, wenn die Zählervariable den Wert b erreicht oder überschreitet.

Vorgehensmodell Zur Evaluierung eines Modelcheckers ist es nicht notwendig, alle 350 Modelle aus der Suite anzuwenden. In vielen Fällen erlaubt das Ergebnis für ein Modell Rückschlüsse darauf, welche Ergebnisse für eine Menge anderer Modelle zu erwarten sind, z.B. weil gleiche Syntaxelemente benutzt werden. Um den Anwender entsprechend durch

den Evaluationsprozess zu leiten, wurde ein Vorgehensmodell entwickelt. Es liegt in graphischer (Ablaufdiagramm) und in textueller Form vor.
Ziel des Vorgehensmodells ist, dass der Anwender nur die aussagekräftigen Tests durchführt und unnötige Tests vermeidet. Wir haben Block-Kategorien und Blöcke jeder Kategorie aufsteigend nach mathematischer und syntaktischer Komplexität geordnet. Die einzelnen Ausprägungen jedes Blocks und die Datentypen wurden der Komplexität nach absteigend geordnet. Diese Ordnungen bestimmen die Reihenfoge der Modelle im Vorgehensmodell. Beispielsweise werden Blöcke mit logischen Funktionen früher als Blöcke mit trigonometrischen Funktionen überprüft. Damit werden grundlegendere Blöcke zuerst abgehandelt.
Ist ein Test erfolgreich, können einfachere Ausprägungen des gleichen Blocks übersprungen werden. Schlagen alle Test mit einem Block oder Datentyp fehl, werden Modelle, die diesen verwenden, ausgelassen. Im Datentyp-Submodul beispielsweise existieren 38 Modelle, die sich aber im günstigsten Fall durch zwei Tests abdecken lassen.
Zu dem Vorgehensmodell gehört eine Tabelle, mit der die Ergebnisse der Tests erfasst werden können. Insbesondere sind hier auch die Abhängigkeiten zwischen diesen dokumentiert. Liegt ein Ergebnis vor, ergeben sich daraus oft weitere Resultate, die dann in der Tabelle festgehalten werden können, ohne dass die entsprechenden Test durchgeführt werden müssen.

3 Erste Anwendungserfahrungen

Um zu prüfen, ob man mit der Evaluationssuite das in der Einführung formulierte Problem angemessen lösen kann, wurde die Suite und das Vorgehensmodell auf zwei kommerzielle Modelchecker für Simulink, OSC EmbeddedValidator (EV, [OSC05]) und TNI Safety-Checker Blockset (SCB, [TNI05]), angewendet. Unser Ziel war dabei festzustellen, ob Simulink-Modelchecker, die grundsätzlich verschiedene Ansätze verwenden, mit der Suite sinnvoll behandelt werden können. Im Vordergrund stand nicht die Evaluierung der Modelchecker.
EV führt die Verfikation auf dem aus einem Simulink-Subsystem mit dSPACE TargetLink generiertem C-Code aus. Die Anforderungen an das System werden aus sogenannten Pattern zusammengestellt. SCB führt die Verifikation direkt auf den Simulink-Subsystemen durch und enthält Simulink-Blöcke für die Formulierung der Beweisforderungen und für Annahmen über Signale.
Der Aufwand, um die Modelle für die Eingabe vorzubereiten, variiert stark für die Modelchecker. Bei unseren beiden Kandidaten beschränkte sich die Vorbereitung auf das Hinzufügen von einigen Blöcken. Es lohnt sich unserer Meinung nach nicht, diesen Schritt zu automatisieren, da sich die Benutzerschnittstellen der Modelchecker stark unterscheiden.
Bei der Syntaxanalyse von EV stellten wir fest, dass einige Blöcke, bzw. Block-Ausprägungen, nicht akzeptiert wurden; einige andere wurden akzeptiert, aber nicht korrekt behandelt. Die Einschränkungen resultieren zum Teil aus den Einschränkungen durch TargetLink, da es dort nicht zu jedem Simulink-Block ein Pendant gibt.
Bei den Effizienzanalysen mit beiden Modelcheckern haben wir die maximale Laufzeit

auf 1 Stunde begrenzt, um die Messungen in annehmbarer Zeit durchzuführen. Wir haben zu jedem Modell einen Beweis mit eventuellem Gegenbeispiel und einen ohne anfertigen lassen.
Das Submodul, das die in Simulink implementierten Zähler enthält, bereitete beiden Modelcheckern syntaktisch keine Probleme. Allerdings brach SCB Beweise, die einen Zählermaximalwert von über 2500 aufwiesen, mit der Meldung „Maximum search depth reached“ ab. Mit dem EV konnten wir diesbezüglich keine Probleme feststellen. Die Stateflow-Modelle des Zählers wurden von SCB nicht akzeptiert. EV verarbeitete diese geringfügig langsamer als die Simulink-Modelle.
Für weitere Ergebnisse der Erprobung der Suite sei auf [Pal05] verwiesen.

4 Fazit

Mit der Evaluationssuite liegt eine Sammlung von Simulink-Modellen vor, mit der die syntaktischen Fähigkeiten und die Effizienz von Matlab/Simulink-Modelcheckern bewertet werden können. Ein Vorgehensmodell hilft, die Evaluierung schnell durchzuführen und nicht aussagekräftige Tests zu vermeiden.
Die Suite erhebt keineswegs den Anspruch auf Vollständigkeit. Angesichts der ständigen Erweiterungen der Simulink-Produktpalette halten wir ein solches Vorhaben für nicht realisierbar. In einigen Punkten lässt sich die Suite noch ausbauen. Mit der aktuellen Version der Suite können zum Beispiel nicht alle syntaktischen Einschränkungen der Modelchecker erkannt werden. Auch werden die Fähigkeiten bezüglich der Eingabe der zu verifizierenden Eigenschaften nur in Verbindung mit der Effizienzanalyse untersucht.
Da die Suite modular aufgebaut ist, lassen sich weitere Analysen leicht integrieren. Denkbar sind Erweiterungen durch Tests zu domänenspezifischen Modellen. Dafür müssen domänentypische Modelle analysiert und ihre Essenz in Suitemodellen erfasst werden.
Die Erprobung der Evaluationssuite hat gezeigt, dass die Suite prinzipiell geeignet ist, Stärken und Schwächen von Modelcheckern für Simulink schnell zu erkennen und zu vergleichen. Die im Rahmen der Erprobung zu beobachtenden Eigenschaften der beiden verwendeten Modelchecker zeigen, dass große Unterschiede zwischen den Werkzeugen existieren und dass solche Vergleiche durchaus sinnvoll sind.

Literatur

[CGP99] Edmund M. Clarke, Orna Grumberg und Doron A. Peled. *Model Checking*. MIT Press, 1999.

[OSC05] OSC–Embedded Systems AG. *EmbeddedValidator 2.0 User Guide*, 2005.

[Pal05] Jacob Palczynski. Anforderungen an einen Modelchecker für Matlab/Simulink. Diplomarbeit, RWTH Aachen, 2005. `http://www-i11.informatik.rwth-aachen.de/?id=eval_sl-mc`.

[TNI05] TNI-Software. *Safety-Checker Blockset V2.2 User's Manual*, 2005.