Anforderungsverifikation von Komponenten- und Konnektormodellen am Beispiel Autonom Fahrender Autos

Malte Heithoff, Bernhard Rumpe, Michael von Wenckstern
Software Engineering, RWTH Aachen Universität, Deutschland

1 Einleitung

Aufgrund des hohen Konkurrenzdrucks in der Automobilindustrie erfolgen Entwicklungen im Bereich des autonomen Fahrens in immer kürzeren Abständen und Funktionen in diesem Gebiet müssen ständig angepasst werden. Die dabei entwickelten Systeme werden immer komplexer und greifen immer tiefer in sicherheitskritische Bereiche ein, somit muss deren Sicherheit validiert werden. Das Ziel ist nicht mehr nur die Unterstützung des Fahrers (Level-3), sondern ein völlig vom Menschen unabhängig fahrendes Auto (Level-5). Die Software hierfür muss ständig weiterentwickelt werden, es findet also ständig Software-Evolution statt.

Für die Modellierung eines solchen Systems wird meist Simulink, ein Modellierungstool für C&C (Komponenten und Konnektoren) benutzt. Zur Validierung der Anforderungen können Tests oder auch beschränktes Model-Checking (BMC) genutzt werden. BMC wird oft zur Falsifizierung benutzt, um schnell mögliche Fehler finden zu können ([2], [1]). Dieses Paper stellt ein Verfahren vor, um C&C-Modelle mittels BMC zu validieren. Dabei wird ein Beispiel aus dem Bereich des autonomen Fahrens verwendet.

2 Modellierung

Abb. 1 zeigt eine C&C-Komponente und soll für den Fall, dass sich auf der Spur vor dem eigenen Auto ein weiteres befindet, folgende Sicherheits-Anforderung erfüllen: *Die Distanz zum vorausfahrenden Fahrzeug muss immer mindestens* $2m$ *betragen.* Die Komponente erhält die Distanz zum nächsten Fahrzeug d als Eingabe und gibt eine Beschleunigung a als Ausgabe zurück. Falls das vorausfahrende Auto weniger als $30m$ entfernt ist, wird mit $-3\ m/s^2$ abgebremst, ansonsten wird mit $3\ m/s^2$ beschleunigt. Unter der Bedingung, dass die Maximalgeschwindigkeit des eigenen Autos nicht mehr als $9\ m/s$ ($\approx 30\ km/h$) und die maximale Beschleunigungs- und Bremskraft der beiden Fahrzeuge nicht mehr als $3\ km/h$ bzw. weniger als $-3\ km/h$ beträgt, wird die Sicherheits-Anforderung erfüllt (Graph ist im Beispiel endlich; kann mit normalen (unbeschränktem) Model-Checking verifiziert werden). Diese Funktion könnte also in bestimmten Szenarien in Stadtbereichen (z.B. 30er-Zonen) freigeschaltet werden.

Für die Modellierung der Anforderungen wurden Eingabe/Ausgabe-Automaten (I/O-Automat) gewählt. Die Automaten sollen dazu genutzt werden das Verhalten einer Komponente zu simulieren; bei erfolgreicher Simulation erfüllt die Komponente die Bedingung. Der Automat akzeptiert, wenn er während des Laufes einen Endzustand erreicht, ansonsten verwirft er. Der Anforderungs-Automat besitzt Ein- und Ausgabe und kann daher beliebiges Verhalten fordern bzw. simulieren. Da die Anforderungen meist unterspezifiziert sind, müssen die Automaten nicht-deterministisch sein: z.B. steht $s \xrightarrow[\{a<0\}]{[true]} t$ für eine Transition, bei der nicht-deterministisch entschieden wird, welcher Wert für $a < 0$ gewählt wird. Zusätzlich repräsentiert in diesem Paper eine Ausgabe $\{\%\}$ jede mögliche Ausgabe, d.h. sie kann alle Ausgaben simulieren. Abb. 2 zeigt die Sicherheits-Anforderung als nicht-deterministische I/O-Automaten. Der Automat benötigt eine Startbedingung, um zu entscheiden, wann diese Anforderung überprüft werden soll: Da es sich hier um den besonderen Fall einer Invariante handelt, soll die Eigenschaft immer überprüft werden $[true]$.

Die Funktion soll nun auch außerhalb von 30er-Zonen freigeschaltet werden, die Maximalgeschwindigkeit beträgt nicht mehr $9\ m/s^2$, sondern $14\ m/s^2$ ($\approx 50\ km/h$). Die bisherige C&C-Komponente erfüllt unter der neuen Bedingung die Anforderung nicht mehr (siehe Fehlerfall in Abbildung 3) und muss angepasst werden. In einer Implementierung der neuen Anforderung könnte in einer möglichen Evolution der Komponente der Sicherheitsabstand auf $60m$ erhöht worden sein. Die Validierung dieser Komponentenversion ergibt, dass diese die Anforderung wieder erfüllt.

3 Vorgehen

Zur Validierung einer C&C-Komponente wird die Komponente sowie der zugehörige Anforderungs-Automat benötigt. Bevor die eigentliche Simulation beginnen kann, muss die C&C-Komponente zuerst in folgenden Schritten in ein Modell transformiert werden, auf dem der Simulationsalgorithmus arbeiten kann. Wir benutzen dafür endliche Transitions-Systeme. ① Dazu wird aus der Komponente erst ein Kontrollflussgraphen berechnet [4] [7], der genau einen Durchlauf der Komponente beschreibt. ②

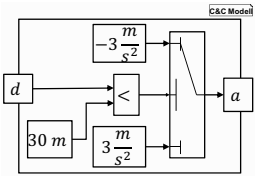

Abb. 1: C&C-Komponente vor der Evolution.

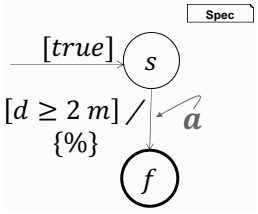

Abb. 2: Sicherheits-Anforderung

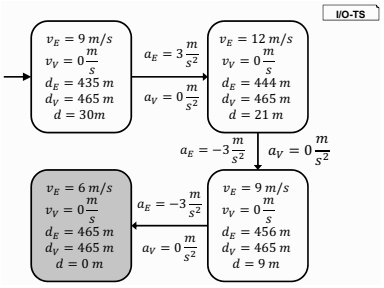

Abb. 3: Gegenbeispiel

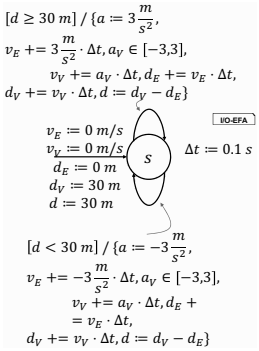

Abb. 4: Automat mit integrierter Umwelt[1].

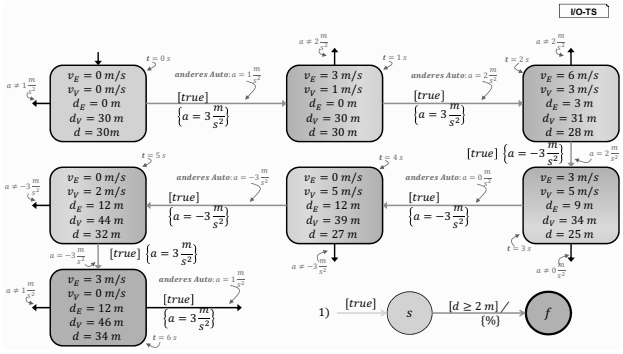

Abb. 5: Ausschnitt aus entfalteten I/O-TS aus Abb.4.

Der Kontrollflussgraph wird dann in einen Automaten mit internen Variablen (ehemals Speicherbausteine in der Komponente) überführt [6]. Dieser Automat beschreibt das vollständige Verhalten der Komponente: Für jede Wertebelegung der Eingabe-Ports wird eine Ausgabe berechnet. ③ In diesen Automaten wird nun noch die Umwelt, wie Geschwindigkeiten oder Distanzen, integriert (siehe Abb. 4). ④ Dann werden alle internen Variablen in einzelne Zustände entfaltet [3] und man erhält ein I/O-Transitions-System (I/O-TS), bei dem jede Transition einen Zeitschritt darstellt (siehe Abb. 5). Da dies ggf. ein unendlich großes I/O-TS erzeugen würde, wird hier nur bis zu einer bestimmten Tiefe entfaltet (BMC). Auf diesem kann (ähnlich zu [5]) der Simulationsalgorithmus ausgeführt werden: Für alle Zustände aus dem I/O-TS, für welche die Startbedingung des Verifikations-Automaten zutrifft, wird überprüft, ob die in Relation stehenden Zustände bei gleicher Eingabe eine gleiche Ausgabe erzeugen können und ob diese zu einem Endzustand führen. Transition a des Automaten in Abb. 2 könnte jede Transition in dem I/O-TS aus Abb. 5 simulieren, weshalb die Anforderung für diese Zustände erfüllt wird. Zur Überprüfung der dabei entstehenden Erfüllbarkeitsprobleme wird Microsofts Z3-Solver benutzt.

4 Herausforderungen

Um Anforderungen an die Umwelt (z.B. die eigene Geschwindigkeit oder die Distanz zu Hindernissen) verifizieren zu können, muss diese mit modelliert werden. Die Berechnung dieser Informationen erfolgt über mittels Differenzengleichungen approximierter Funktionen, welche entsprechende physikalische Gesetzmäßigkeiten beschreiben. Um etwa die Geschwindigkeit zu modellieren, wird für jeden Zeitschritt die Funktion $v := v + a \cdot \Delta t$ berechnet. Hierbei steht Δt für die Dauer eines Zeitschrittes. Dies hat auch zur Folge, dass Informationen wie Geschwindigkeit oder Distanz in Schritt ④ mit entfaltet werden müssen, was wiederum zu einem sehr großen Transitions-System führt (Zustandsraumexplosionsproblem [3]). Um die Anzahl an Zuständen klein zu halten, kann Δt variiert werden. Ein kleines Δt modelliert die Umwelt sehr genau, während die Wahl eines großen Δt bedeutet, dass weniger Zustände berechnet werden. Falls die Verifikation für ein zu großes Δt fehlschlägt, dann könnte die Wahl eines kleineren Δt zu einem positiven Ergebnis führen.

5 Zusammenfassung

Das hier vorgestellte Verfahren kann eingesetzt werden, um schnell ungewolltes Verhalten von C&C-Komponenten zu identifizieren. Damit einhergehend ist es dafür geeignet die Aufgabe der Validierung von Komponentenevolution zu übernehmen: In der Softwareevolutionsphase kann so validiert werden, ob eine neue Komponentenversion noch immer alle an sie gerichteten Anforderungen erfüllt.

Literatur

[1] E. C. Armin Biere, Alessandro Cimatti and Y. Zhu. Symbolic model checking without bdds. In *TACAS*, 1999.

[2] H. v. M. Armin Biere, Marijn Heule and T. Walsh. Handbook of satisfiability. 2009.

[3] C. Baier and J.-P. Katoen. *Principles of model checking.* MIT Press, 2008.

[4] D. Ernst. Transformation von MontiArc-Modellen zu Kontrollflussgraphen. Bachelorarbeit, 2016.

[5] B. Rumpe, C. Schulze, M. v. Wenckstern, J. O. Ringert, and P. Manhart. Behavioral Compatibility of Simulink Models for Product Line Maintenance and Evolution. In *SPLC*. ACM, 2015.

[6] I. Shumeiko. Strategies to Reduce Variable Unfoldings in I/O-EFA Simulation Preorder Algorithm. Masterarbeit, 2017.

[7] S. Tolksdorf. Kontrollflussgraphenanalyse für das Verifikationstool. Bachelorarbeit, 2016.

[1] a ist hier Output, der Rest sind interne Variablen. a_V ist die Beschleunigung des vorausfahrenden Fahrzeuges, v_E und v_V sind die Geschwindigkeiten des eigenen bzw. des vorausfahrenden Fahrzeuges und d_E und d_V sind zurückgelegte Distanzen der beiden Fahrzeuge