Heuristische Suche in der Testfallgenerierung

Alexander Pretschner und Jan Philipps
Institut für Informatik, Technische Universität München
www4.in.tum.de/~{pretschn,philipps}

Zusammenfassung. Mit zunehmendem Einsatz von CASE-Tools bei der Entwicklung reaktiver Systeme rückt die Anwendbarkeit (teil)automatisierter Validierungs- und Verifikationstechniken in greifbare Nähe. Ausgehend von einem Modell des Systems, können Testsequenzen für das Modell selbst und –nach entsprechender Konkretisierung– für die Implementierung berechnet werden. Für funktionale Testfälle läßt sich dieses Problem auf die Suche nach ausgezeichneten Trajektorien im Zustandsraum zurückführen. Klassische heuristische Graphensuchverfahren sind hierbei vielversprechende Kandidaten für Effizienzsteigerungen. Wir demonstrieren diesen Ansatz am Beispiel der Best-First-Suche zur Kontrolle der Testfallgenerierung mit symbolischer Ausführung.

1 Einleitung

Verifikationstechniken wie Model Checking haben das Ziel, mit mathematischer Präzision und Vollständigkeit die Korrektheit eines Modells in bezug auf eine Eigenschaft nachzuweisen. Der große Vorteil des Model Checking ist, daß es sich — zumindest konzeptionell — um eine push-button-Technologie handelt, die außer der Eingabe einer zu verifizierenden Eigenschaft keinerlei Interaktion mit dem Benutzer verlangt. Der Nachteil ist, daß Model Checking aufgrund der Zustandsraumexplosion bis heute nicht für große industrielle Systeme einsetzbar ist. Außerdem ist der Nachweis der Korrektheit eines Modells nur ein erster Schritt: die Korrektheit der eigentlichen Implementierung muß ebenfalls überprüft werden.

Da das Problem der Zustandsexplosion direkt mit der angestrebten Vollständigkeit des Nachweises verbunden ist, bietet es sich an, auf eben diese Vollständigkeit durch eine explizite Einschränkung des Suchraums zu verzichten. Damit wird die Menge der Traces des Systems eingeschränkt (mathematisch findet eine Konkretisierung des Modells statt), und der Nachweis universeller Eigenschaften wie Invarianz, Sicherheit oder Lebendigkeit muß durch sorgfältig gewählte existentielle endliche Eigenschaften approximiert werden.

Im folgenden wird deshalb die Klasse zu verifizierender Eigenschaften auf existentielle Eigenschaften eingeschränkt, wie etwa "bringe das System in einen Zustand q". Offensichtlich lassen sich auch eigenschaftsunabhängige Überdeckungskriterien (Pfadüberdeckung, MC/DC) durch Mengen solcher existentieller Eigenschaften beschreiben. Testingenieure sind in der Lage, derartige funktionale Testfallspezifikationen aus der Spezifikation zu extrahieren; die resultierende endliche Menge endlicher Testsequenzen dient dann dem Test von Modell und Implementierung.

2 Beispiel

Zur Spezifikation reaktiver Systeme verwenden wir das CASE-Tool AutoFocus[1]. An die UML-RT angelehnte Beschreibungstechniken für Struktur, Verhalten (erweiterte endliche Zustandsmaschinen),

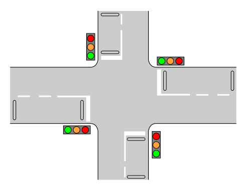

Abbildung 1: Ampelsteuerung.

Datentyp- und Funktionsdefinitionen sowie Interaktion (Sequenzdiagramme) erlauben eine konzise und übersichtliche Darstellung eines Systems.

Abbildung 2 zeigt Struktur und Teile des Verhaltens einer Ampelsteuerung für eine Kreuzung (Abbildung 1) mit zwei Induktionsschleifenpaaren für jede Richtung. In Abhängigkeit eines über die Zeit definierten Verhaltens der Induktionsschleifen sowie eines externen Schalters kann sich die Ampel in verschiedenen Modi befinden: Tages- und Nachtmodus sowie in einem Zustand, den das System bei Ausfall von essentiellen Systemteilen betritt (Blinken der gelben Leuchten). Dieser Zustand wird u.a. dann erreicht, wenn eine Induktionsschleife aufgrund eines ununterbrochenen Signals von zehn Minuten Dauer als defekt angenommen wird. Da sich im System andere Timing-Constraints im Sekundenbereich befinden, hat ein Testfall, der das System in diesen Zustand bringt, mindestens eine Länge von 600 Schritten. Das System enthält mehrere Timer und Zähler, die den Zustandsraum vergrößern. Deshalb kann dieser Trace nicht mit einem Model Checker wie SMV erzeugt werden.

[1] http://autofocus.in.tum.de

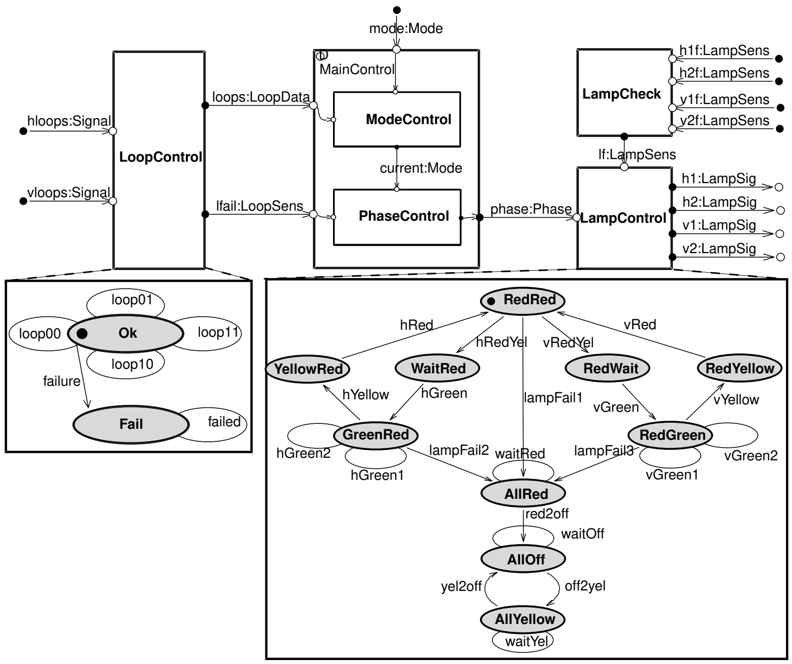

Abbildung 2: Ampelsystem. Struktur, Verhalten.

3 Testfallgenerierung

Unser Ansatz zur Testfallgenerierung basiert auf symbolischer Ausführung mit Constraint-Logikprogrammierung (CLP). Kurz gesagt, wird das AutoFocus-Modell zusammen mit einer Testfallsspezifikation, Umwelt- und Effizienz-Constraints in Form von Automaten, Formeln oder Sequenzdiagrammen nach CLP übersetzt. Die Umwelt- und Effizienz-Constraints dienen der oben erwähnten expliziten Beschneidung des Suchraums. Die Zustände dieses Systems werden dann mit verschiedenen Zustandsspeicherungs- und Suchstrategien (Wettbewerbsparallelität, Heuristik) aufgezählt, um einen Trace zu finden, der die Testfallspezifikation erfüllt.

Die Definition von Fitneßfunktionen für heuristische Suchverfahren wie A* oder best-first hingegen erlaubt es, diesen Testfall binnen einer Sekunde zu finden. Der Blinkmodus wird erreicht, wenn sich die Komponente *LoopControl* im Zustand *Fail* oder die Komponente *LampControl* im Zustand *AllYellow* befindet. Zwei Zähler in der Komponente *LampControl*, cnt_h und cnt_v für jedes Paar von Induktionsschleifen enthalten die Anzahl aufeinanderfolgender Aktivierungssignale der entsprechenden Schleife. Die Entscheidung während der symbolischen Ausführung, welcher Zustandsübergang im System als nächstes "ausprobiert" werden soll, kann dann mittels der Fitneßfunktionen $\varphi_1(t) = 1/(600 - cnt_h + 600 - cnt_v + \delta(dest(t), \sigma_{Fail}))$ oder $\varphi_2(t) = 1/(\delta(dest(t), \sigma_{AllYellow}))$ erfolgen, wobei *dest(t)* der Zielkontrollzustand von Transition t und $\delta(s_1, s_2)$ die minimale Anzahl von Transitionen ist, um vom Kontrollzustand s_1 zum Kontrollzustand s_2 zu gelangen. Für alle anderen Komponenten wird eine zufällige Reihenfolge der auszuwählenden Transition gewählt, wenn diese keinen Einfluß auf die Fitneß haben. Auf die Verallgemeinerung dieser Ideen für mehrere Komponenten (und das Problem einer adäquaten Komponentenordnung im erzeugten CLP-Code) kann hier aus Platzgründen nicht eingegangen werden.

4 Zusammenfassung

Testfallgenerierung für Modelle und Implementierungen reaktiver Systeme mit Model Checkern scheitert in der Praxis am Problem der Zustandsexplosion und konzeptionell am Zwang, endliche Mengen endlicher Systemtraces als Tests zu verwenden.

Unsere aktuellen Arbeiten befassen sich mit einem methodischen Übergang von universellen Eigenschaften zu existentiellen oder funktionalen Testfallsspezifikationen. Überdeckungskriterien sind wegen ihrer inhärenten Eigenschaftsunabhängigkeit nur eingeschränkt für einen möglichst vollständigen Test mit "sinnvollen" Testfällen geeignet; allerdings können mit ihnen mit wenig Aufwand große Testfallmengen erzeugt werden, um z.B. die Übereinstimmung von unabhängig und manuell erzeugtem Code mit als Spezifikation dienenden Systemmodellen zu überprüfen. Dies ist inbesondere bei Auftraggeber/Zulieferer-Beziehungen wünschenswert, wie sie in der Automobilindustrie vorherrschen.