Dynamische Äquivalenzklassen im Klassifikationsbaum für zustandsbehaftete Systeme

Harald Cichos und Andy Schürr
Fachgebiet Echtzeitsysteme
Technische Universität Darmstadt
Merckstraße 25
D-64283 Darmstadt
{harald.cichos,andy.schuerr}@es.tu-darmstadt.de

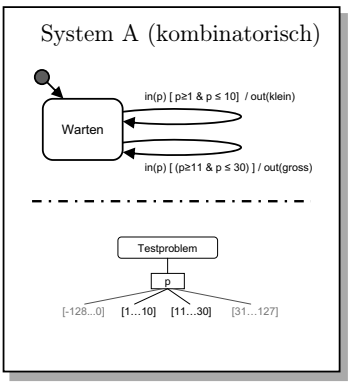
Abstract: Die Klassifikationsbaummethode ist eine weit verbreitete funktionsorientierte Methode zum Test von kombinatorischen Systemen. In der vorliegenden Arbeit wird eine Erweiterung der Klassifikationsbaummethode vorgestellt, die es ermöglicht, gültige und konkrete Testsequenzen auch für zustandsbehaftete Systeme abzuleiten. Zu diesem Zweck werden die im Klassifikationsbaum enthaltenen Äquivalenzklassen in Abhängigkeit vom Zustand des verhaltensbeschreibenden Testmodells aus diesem dynamisch erzeugt.

1 Einleitung

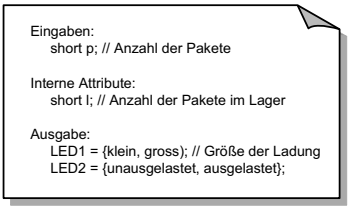
In der Praxis erfreuen sich funktionsorientierte Testmethoden, auch bekannt als Black-Box-Tests, großer Beliebtheit [Lig02]. Die Klassifikationsbaummethode (KBM) [GG06] ist ein solcher Ansatz, und erlaubt die systematische Herleitung von Eingabedaten für Testfälle, die anschließend auf das zu testende System angewendet werden. Bisher wird die KBM hauptsächlich zum Testen von kombinatorischen Systemen eingesetzt, da sie sich nur beschränkt zum Testen von zustandsbehafteten Systemen eignet. Begründet wird die Tatsache dadurch, dass sich bisher die durch einen Zustandswechsel erfolgten Änderungen im Systemverhalten nicht auf die Äquivalenzklassen im Klassifikationsbaum übertragen lassen. So unterstützt die KBM derzeit nur Äquivalenzklassen mit festen Wertebereichen, die sich nicht an den aktuellen Zustand anpassen können. Damit die KBM zukünftig auch auf zustandsbehaftete Systeme effizient anwendbar ist, müssen die internen Attribute und der Zustand eines Systems bei der Bildung von Äquivalenzklassen mit in Betracht gezogen werden. Da ein solches Vorgehen in der Regel sehr aufwändig und fehleranfällig ist, bietet sich eine Automatisierung an.
Wir schlagen daher vor, die zu einem Systemzustand passenden Äquivalenzklassen eines Klassifikationsbaums aus einem endlichen Automaten automatisch abzuleiten. Dadurch lässt sich die Komplexität und Fehleranfälligkeit beim Erzeugen des Klassifikationsbaums erheblich reduzieren. Anschließend können die für die Testfälle benötigten Testdaten aus den für den Zustand gültigen Eingabeäquivalenzklassen systematisch oder intuitiv ausgewählt und auf das zu testende System angewendet werden. In dieser Arbeit wird unser Ansatz näher erläutert und dessen Vorteile vorgestellt.

2 Einführung des Anwendungsbeispiels

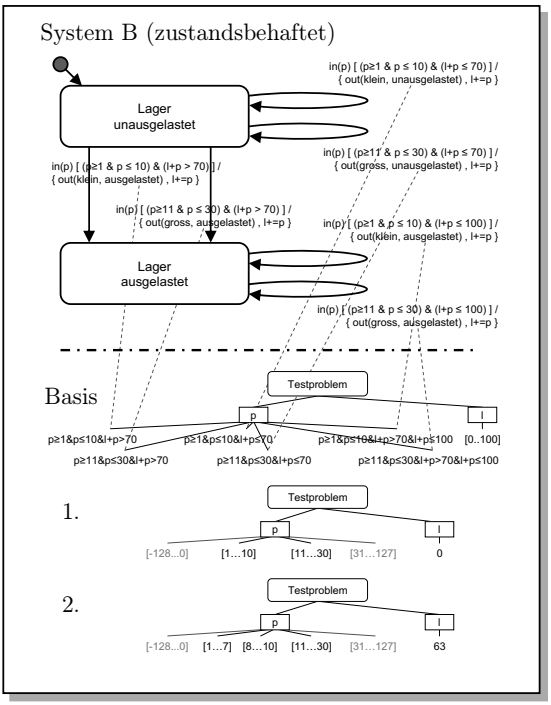
Zum besseren Verständnis des Ansatzes werden in diesem Kapitel zwei Systeme aus dem Transport-Logistik-Bereich vorgestellt (siehe Abbildung 1). Beide Systeme überwachen die Einlagerung von Paketen in ein Lagerhaus. Kleine Ladungen bestehen aus 1 bis 10 Paketen. Große Ladungen hingegen aus 11 bis 30 Paketen. Beide Systeme sollen als Ausgabe anzeigen, ob es sich um eine kleine oder große Ladung handelt. Zusätzlich dazu besitzt System `B` die Aufgabe die Auslastung des Lagers anzuzeigen. Dabei gilt das Lager erst bei mehr als 70 eingelagerten Paketen als ausgelastet. Eine formale Spezifikation der beiden Systeme ist in Abbildung 1 dargestellt. Bei System `A` handelt sich um ein kombinatorisches System, das auch als einfache Funktion interpretiert werden kann. System `B` hingegen ist ein zustandsbehaftetes System, da es sich entsprechend der Anzahl bereits eingelagerter Pakete unterschiedlich verhält. Für diese beiden Systeme soll im Folgenden der jeweils zugehörige Klassifikationsbaum erstellt werden. Im Anschluss könnten dann aus den Klassifikationsbäumen fehlersensitive Testfälle für die zu testenden Systeme erstellt werden.



a) Formale Spezifikation eines kombinatorischen Systems mit dem dazu passenden Klassifikationsbaum.



c) Ergänzende textuelle Spezifikation zu den zwei Systemen A und B.



b) Formale Spezifikation eines zustandsbehafteten Systems mit zwei aus dem Basis-Klassifikationsbaum abgeleiteten und dem internen Attribut entsprechenden Klassifikationsbäumen.

Abbildung 1: Formale Spezifikation zweier Systeme mit zugehörigen Klassifikationsbäumen.

3 Klassifikationsbaummethode

Beim Black-Box-Test darf das zu testende System nur über seine Schnittstellen getestet werden. Demzufolge liegt die einzige Möglichkeit eine Implementierung auf Konformität mit ihrer Spezifikation zu überprüfen in der Untersuchung des vollständigen Eingaberaumes. Jedoch ist das in realen Softwaresystemen aufgrund der damit verbunden Kosten selten praktikabel. Daher wird der Eingabewertebereich typischerweise in Äquivalenzklassen partitioniert. Dabei enthält eine Eingabeäquivalenzklasse nur solche Werte, die gleiches Ausgabeverhalten aufweisen, welches aber mehrere als äquivalent angesehene Ausgabewerte umfassen darf [Lig02]. Die Bildung von Äquivalenzklassen findet auch innerhalb der KBM Anwendung. Dabei wird zunächst die Spezifikation eines kombinatorischen Systems analysiert, um die testrelevanten Eingabeparameter zu identifizieren. Zugleich werden die für das Testproblem unwichtigen Informationen abstrahiert, um dessen Komplexität zu senken. Anschließend wird der Wertebereich aller testrelevanten Eingabeparamter anhand von Aspekten in Äquivalenzklassen unterteilt. Diese Unterteilung geschieht systematisch sowie schrittweise und führt letztendlich zu einer grafisch darstellbaren Baumstruktur, dem Klassifikationsbaum (siehe Abbildung 1a). Zum Erstellen eines Testfalls muss jedem Eingabeparameter aus einer seiner Äquivalenzklassen ein repräsentativer Wert zugewiesen werden. Zum Erstellen von fehlersensitiven Testfällen eignen sich erfahrungsgemäß die Grenzwerte der Äquivalenzklassen. Durch das Bilden von Äquivalenzklassen können redundante Testfälle vermieden werden, was zu einer Reduktion der Anzahl zu betrachtender Testfälle führt. Weitere Vorteile der KBM sind die Visualisierung des Testproblems und das mögliche Aufdecken von Unklarheiten in der funktionalen Spezifikation während der Erstellung des Baumes.
Die KBM setzt auf der Category-Partition-Method [OB88] auf und diente ursprünglich dem Erstellen abstrakter Testfälle. Sie wurde erst in einer späteren Arbeit [LB03] um die Eingaben von konkreten Testdaten erweitert. Die Kombination von Klassifikationsbäumen und zustandsbehafteten Systemen wurde erstmals in [OMS09] angedacht. Zum Themengebiet passende Arbeiten [WS08a, WS08b, Wei08] beschäftigten sich aber schon vorher mit der Bildung von Äquivalenzklassen und deren Einsatz beim Test von zustandsbehafteten Systemen. Der Classification-Tree-Editor (CTE) ist ein eigens für die KBM entwickeltes Werkzeug, das den Tester beim Erstellen eines Klassifikationsbaums unterstützt [GG06, LW00]. Der CTE erlaubt das Erstellen von Testsequenzen durch das Zusammenfassen von Testfällen, wobei ein Testfall jeweils die Voraussetzung für den nachfolgenden Testfall schafft. Allerdings passen sich die Äquivalenzklassen während einer Testsequenz nicht den internen Attributwerten eines zustandsbehafteten Systems an. Folglich kann der Repräsentant einer Äquivalenzklasse bei Vorliegen eines Zustandswechsels im Folgezustand ein anderes Verhalten verursachen, als das Verhalten der durch ihn charakterisierten Äquivalenzklasse. Aufgrund dessen sollten mit dem CTE erstellte Testsequenzen für zustandsbehaftete Systeme als potentiell fehlerhaft angesehen werden. Auch lassen sich mit dem CTE Bedingungen formulieren, die sich auf Äquivalenzklassen unterschiedlicher Eingabeparameter beziehen. So kann angegeben werden, dass eine Äquivalenzklasse nicht mehr ausgewählt werden darf, wenn zuvor einem anderen Parameter ein gewisser Wert zugewiesen wurde. Allerdings lassen sich diese Bedingungen nicht testfallübergreifend formulieren, weshalb sie für zustandsbehaftete Systeme ungeeignet sind.

4 Erweiterung auf zustandsbehaftete Systeme

Bei dem Versuch, für die beiden Systeme `A` und `B` (siehe Abbildung 1) jeweils einen Klassifikationsbaum zu erstellen, wird deutlich, dass die KBM in ihrer derzeitigen Form zwar für kombinatorische Systeme geeignet ist, nicht aber für zustandsbehaftete Systeme. Damit die KBM auch auf zustandsbehaftete Systeme angewendet werden kann, müssen sich die Äquivalenzklassen eines Klassifikationsbaums dynamisch an den Zustand und an die internen Attribute des Systems anpassen. Da die Abhängigkeiten zwischen dem Zustand bzw. den internen Attributen und den Äquivalenzklassen der Eingabewerte sehr komplex ausfallen können und daher für eine manuelle Herleitung zu fehleranfällig sind, sollten solche Äquivalenzklassen automatisch berechnet werden. Mit dynamisch hergeleiteten, dem Systemzustand entsprechenden Äquivalenzklassen sind Testsequenzen mit konkreten und gültigen Grenzwerten als Eingabedaten für zustandsbehaftete Systeme mittels der KBM möglich. Für das automatische Herleiten der Äquivalenzklassen wird jedoch eine formale Spezifikation des zu testenden Systems vorausgesetzt, beispielsweise ein Zustandsautomat (siehe Abbildung 1b).
Durch die Hinzunahme eines Zustandsautomaten gehört die KBM zukünftig zu den zustandsbasierten Testmethoden, die wiederum den modellbasierten Testverfahren zugeordnet werden. Zustandsbasierte Testmethoden werden eingesetzt, wenn sich neben den Eingabewerten auch der aktuelle Zustand des zu testenden Systems auf das Systemverhalten auswirkt. Bisher wurden zustandsbasierte Testmethoden vor allem zur Strukturüberdeckung eingesetzt. Durch die Kombination mit der Äquivalenzklassenbildung bietet sich jedoch das Testen von Grenzwerten an. Ein weiterer Vorteil dieser Kombination ist zudem, dass im hinzugenommenen Zustandsautomaten das erwartete Testverhalten enthalten ist, wodurch dieser auch als Testorakel eingesetzt werden kann. Bisher wird diese Funktion vom Tester übernommen. Schließlich eignet sich die Kombination, um die üblicherweise nicht spezifizierten, ungültigen Äquivalenzklassen automatisch zu identifizieren (siehe die äußeren Äquivalenzklassen des Klassifikationsbaums in Abbildung 1a) und damit, durch die Wahl eines entsprechenden Repräsentanten, das System auf unnötig implementierte Funktionen zu überprüfen.
Aus dem Zustandsautomaten kann für jeden Zustandsübergang die Äquivalenzklasse derjenigen Werte berechnet werden, die für den Zustandsübergang benötigt werden (siehe Basis-Klassifikationsbaum in Abbildung 1b). Für diese Aufgabe können insbesondere Verfahren der symbolischen Ausführung Anwendung finden [Kne92]. Diese sind in der Lage, selbst noch bei Zustandsübergängen mit komplexen Verzweigungsbedingungen (Kontrollflussartige Strukturen) die korrekten Äquivalenzklassen zu bestimmen. In einem zustandsbehafteten System wirken sich der Zustand und die internen Attribute in der Regel auf die Äquivalenzklassen der Eingabeparameter aus. Beispielsweise ist in Tabelle 1 gut zu erkennen, welche Auswirkung der Wert des internen Attributs `l` des Systems `B` auf die Äquivalenzklassen des Eingabeparameters `p` hat. In Abbildung 1b sind zum besseren Verständnis noch zwei Klassifikationsbäume dargestellt, deren Äquivalenzklassen jeweils an den Wert 0 bzw. 63 des internen Attributs `l` dynamisch angepasst wurden.
Da interne Attribute nicht direkt von außen beeinflussbar sind, kann deren Auswirkung auf die Äquivalenzklassen der Eingabeparameter leicht aufgelöst werden. So lässt sich die Äquivalenzklasse eines abhängigen Eingabeparameters sofort mit dem Wert des zugehöri-

ÄK des Eingabeparameters *p* in Abhängigkeit vom Wert des Attributs *l*	Zustand	ÄK für Attribut *l*
[1 … 10] , [11 … 30]	unausgelastet	[0 … 40]
[1 … 10] , [11 … (70 – *l*)] , [(71 – *l*) … 30]	unausgelastet	[41 … 59]
[1 … 10] , [11 … 30]	unausgelastet	60
[1 … (70 – *l*)] , [(71 – *l*) … 10] , [11 … 30]	unausgelastet	[61 … 69]
[1 … 10] , [11 … 30]	unausgelastet	70
[1 … 10] , [11 … (100 – *l*)]	ausgelastet	[71 … 89]
[1 … (100 – *l*)]	ausgelastet	[90 … 99]

Tabelle 1: Die Äquivalenzklassen des Eingabeparameters `p` des zustandsbehafteten Systems `B` (siehe Abbildung 1b) verändern sich in Abhängigkeit vom Wert des internen Attributs `l`.

gen internen Attributs verrechnen. Je mehr Abhängigkeiten durch das Festlegen von Werten aufgelöst wurden, desto übersichtlicher wird die Darstellung des dynamischen Klassifikationsbaums. Durch das Auflösen der Abhängigkeiten können auch mehrere Äquivalenzklassen eines Eingabeparameters zusammenfallen. In Abhängigkeit davon, für welche Äquivalenzklasse zuerst Repräsentanten gewählt und somit die Abhängigkeiten aufgelöst werden, kann der Klassifikationsbaum vorübergehend unterschiedliche Äquivalenzklassen ausbilden. Zusätzlich zu der Abhängigkeit zwischen den Eingabeparametern und den internen Attributen, können die Eingabeparameter aber auch untereinander voneinander abhängen. Diese Abhängigkeit lässt sich durch Vorbedingungen lösen. Diese werden den dynamischen Äquivalenzklassen eines abhängigen Eingabeparameter beigefügt und ermöglichen bei Auswahl eines Repräsentanten die Auflösung der Abhängigkeit.
Testsequenzen können mit dieser Erweiterung für zustandsbehaftete Systeme sowohl manuell als auch automatisch erstellt werden. Das manuelle Erstellen einer Testsequenz verläuft wie folgt:

Ausgehend vom Startzustand werden die Äquivalenzklassen berechnet, die die Bedingungen der ausgehenden Zustandsübergänge erfüllen. Anschließend werden, abhängig vom gewünschten Folgezustand, die Repräsentanten ausgewählt. Ausgehend vom erreichten Folgezustand wird der zuvor getätigte Schritt solange wiederholt, bis das Testziel der Testsequenz erreicht ist.

Ein Testziel kann beispielsweise das Erreichen eines Zustandes darstellen. Testsequenzen zu einem bestimmten Testziel lassen sich aber auch automatisch erstellen. Durch den Einsatz von Model-Checkern [GH99] lassen sich beispielsweise die kürzesten Testsequenzen zu einem Testziel automatisch berechnen. Unabhängig davon, auf welche Art die Testsequenzen erstellt wurden, können diese anschließend durch die KBM so modifiziert werden, dass die Testsequenzen gegenüber Fehlern sensitiver werden. Zu diesem Zweck müssen die Eingabewerte der sequentiell ablaufenden Testfälle durch die Grenzwerte der zugehörigen Äquivalenzklasse ausgetauscht werden. Allerdings ist dabei darauf zu achten, dass das eigentliche Verhalten der Testsequenz nicht verändern wird.

5 Zusammenfassung und Ausblick

In dieser Arbeit haben wir einen Ansatz beschrieben, wie sich die Klassifikationsbaummethode auf zustandsbehaftete Systeme effizient anwenden lässt. Dafür wird die Methode um eine Zustandsautomaten erweitert, der gerade der formalen Spezifikation des zu testenden Systems entspricht. Aus diesem Verhaltensmodell heraus kann der Eingabewertebereich in die für einen Systemzustand gültigen Äquivalenzklassen zerlegt werden. Durch die Auswahl und Festlegung von Grenzwerten als Repräsentanten der Äquivalenzklassen können dann insbesondere fehlersensitive und weniger redundante Testfälle für das zustandsbehaftete System abgeleitet werden. Aufgrund der mit der Berechnung der Äquivalenzklassen verbundenen hohen Komplexität und derer dynamischen Darstellung soll in weiteren Arbeiten ein Werkzeug entwickelt werden, das diese Aufgaben übernimmt.

Literatur

[GG06] M. Grochtmann und K. Grimm. Classification Trees for Partition Testing. *Software Testing, Verification and Reliability*, 3(2):63–82, 2006.

[GH99] A. Gargantini und C. Heitmeyer. Using Model Checking to Generate Tests from Requirements Specifications. *SIGSOFT Software Eng. Notes*, 24(6):146–162, 1999.

[Kne92] R. Kneuper. Validation und Verifikation von Software durch symbolische Ausführung. *Testen, Analysieren und Verifizieren von Software, Informatik Aktuell. Springer*, 1992.

[LB03] S. Lützkendorf und K. Bothe. Attributierte Klassifikationsbäume zur Testdatenbestimmung. *Fachgruppe TAV der Gesellschaft für Informatik*, 2003.

[Lig02] P. Liggesmeyer. *Software-Qualität*. Spektrum, Akad. Verlag, 2002.

[LW00] E. Lehmann und J. Wegener. Test Case Design by means of the CTE XL. *Proceedings of the 8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000)*, 2000.

[OB88] T. J. Ostrand und M. J. Balcer. The Category-Partition Method for Specifying and Generating Functional Tests. *Commun. of the ACM*, 31(6):676–686, 1988.

[OMS09] S. Oster, F. Markert und A. Schürr. Integrated Modeling of Software Product Lines with Feature Models and Classification Trees. In *Proceedings of the 13th Int. SPL Conf.*, MAPLE 2009 Workshop. Springer, 2009.

[Wei08] S. Weißleder. Partition-Oriented Test Generation. *GI Jahrestagung*, 1:199–204, 2008.

[WS08a] S. Weißleder und H. Schlingloff. Deriving Input Partitions from UML Models for Automatic Test Generation. In *Models in Software Eng.: Workshops MODELS 2007*, Seiten 151–163. Springer, 2008.

[WS08b] S. Weißleder und H. Schlingloff. Quality of Automatically Generated Test Cases based on OCL Expressions. *ICST*, 2008.