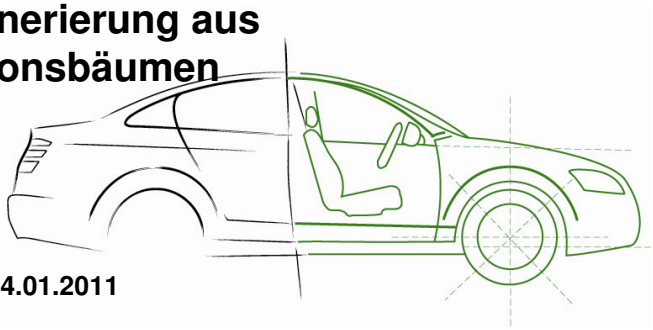
berner & mattner
optimizing your development

Sequenzgenerierung aus Klassifikationsbäumen



Peter M. Kruse, 24.01.2011

berner & mattner
optimizing your development

Inhalt

- Einleitung
- Stand von Wissenschaft und Technik
- Generierung von Testsequenzen mit der Klassifikationsbaum-Methode
  - Abhängigkeitsregeln
  - Generierungsvorschriften
  - Prototyp
- Zusammenfassung

Einleitung

- Klassifikationsbaum-Methode [1], CTE XL [3]
- Automatische Generierung von Testfallmengen für unterschiedliche Testobjekte
- Erstellung von Testsequenzen bislang nur manuell

→ Automatische Erzeugung sinnvoller, fehlersensitiver Testsequenzen

3

Stand von Wissenschaft und Technik

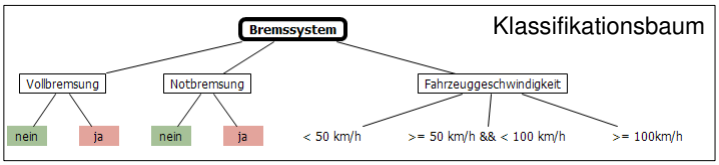
- Testsequenz-Generierung mittels Techniken des Modell-Checkings [4]
  - Gegenbeispiel-Generierung des Modell-Checkers zur Erzeugung relevanter Testsequenzen
- Temporale Logiken, *Linear Temporal Logic* (LTL) und *Computation Tree Logic* (CTL) [5]
- CCTL Logik zur Verifikation von K.-Baum-Testsequenzen [6]
  - Testsequenzen und Transitionen verifizieren mit Echtzeit-Modell-Checker
- Generierung von Testsequenzen basierend auf der Beschreibung endlicher Zustandsautomaten (FSM) [10]

4

berner & mattner
optimizing your development

Testsequenz-Generierung mit der K.-Baum-Methode

- Aktuell: regelbasierte Generierung von Testfallmengen für kombinatorisches Testen mit der Klassifikationsbaum-Methode



d1: high → Night
d2: Motorcycle OR Truck ⇔ medium

Abhängigkeitsregeln

pairwise(Proceeding Vehicle,Speed,Daylight);

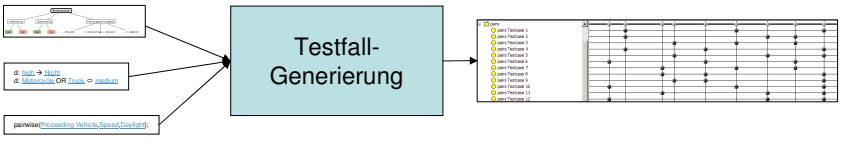
Generierungsvorschrift

5

berner & mattner
optimizing your development

Testsequenz-Generierung mit der K.-Baum-Methode

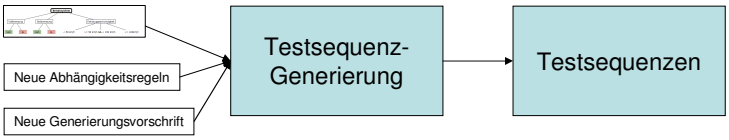
- Aktuell: regelbasierte Generierung von Testfallmengen für kombinatorisches Testen mit der Klassifikationsbaum-Methode



6

Testsequenz-Generierung mit der K.-Baum-Methode

- Erweiterung: regelbasierte Generierung von Test*sequenz*mengen für kombinatorisches Testen mit der Klassifikationsbaum-Methode



Abhängigkeitsregeln

Allgemein

- `Wenn` Klasse c_i aus Klassifikation C in Testschritt t_n `dann` muss Klasse c_j aus Klassifikation C in darauffolgendem Testschritt t_{n+1} der Testsequenz ausgewählt sein

Beispiel

- `Wenn` Fahrzeuggeschwindigkeit < 50 km/h in Testschritt t_n `dann` Fahrzeuggeschwindigkeit >= 50 km/h && < 100 km/h in darauffolgendem Testschritt t_{n+1}

Abhängigkeitsregeln

Allgemein

- Wenn $C=c_i$ in t_n, dann $C=c_j$ in t_{n+m}
- Wenn $C=c_i$ in t_n, dann $C=c_j$ in allen t_{n+1} bis t_{n+m}
- Wenn $C=c_i$ in t_n, dann $C=c_j$ in allen t_{n+m} bis t_{n+o}

Zusätzlich

- Kompositionen mit bekannten logischen Operatoren (*AND*, *OR*, *NOT*, *NAND*, *NOR*, *XOR*, ...)

Information

- Existierende Abhängigkeitsregeln bilden eine Untermenge der neuen Abhängigkeitsregeln mit $m = 0$ für beliebige t_n und t_{n+m}

9

Generierungsvorschriften

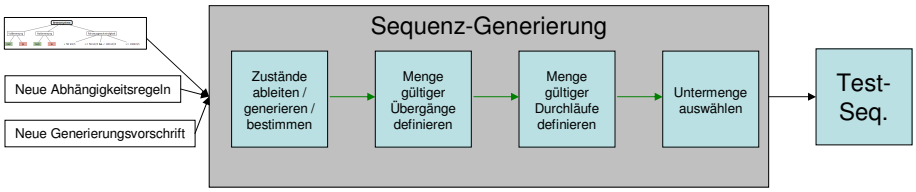
Parameter

- Gewünschter Abdeckungsgrad
- (Unter-)Menge aller Kombination von Klassen des K.-Baums
- Minimale und maximale Anzahl von Testschritten pro Testsequenz
- Definierte Menge von zugelassenen Klassenkombinationen für den ersten und den letzten Testschritt von Testsequenzen
- Maximale Anzahl von Testschritt-Wiederholungen (sowohl unmittelbar aufeinander folgende als auch innerhalb der gesamten Testsequenz)
- Maximale Anzahl von Klassen-Wiederholungen (sowohl unmittelbar aufeinander folgende als auch innerhalb der gesamten Testsequenz)

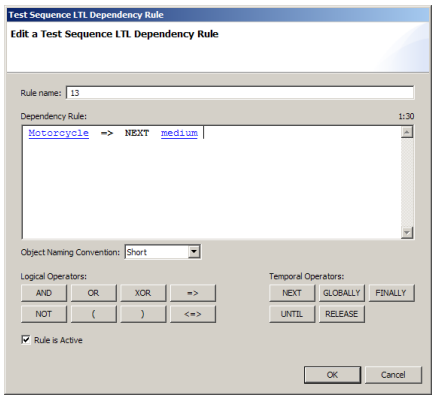
10

Prototyp

- Blick in die Blackbox



Prototyp: Abhängigkeitsregeln

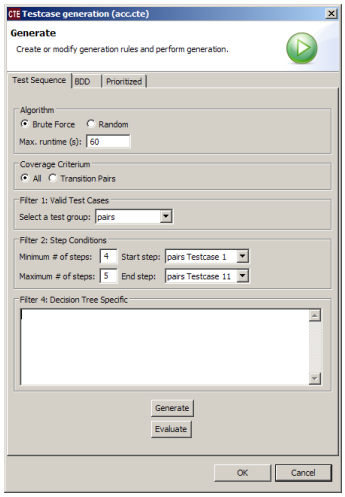


- Detailiertere Beschreibung möglich mittels LTL Regeln
- Ggf. zukünftige Vereinfachung
- Gelten pro Testsequenz
- Auch anwendbar auf manuell erstellte Testsequenzen

berner & mattner
optimizing your development

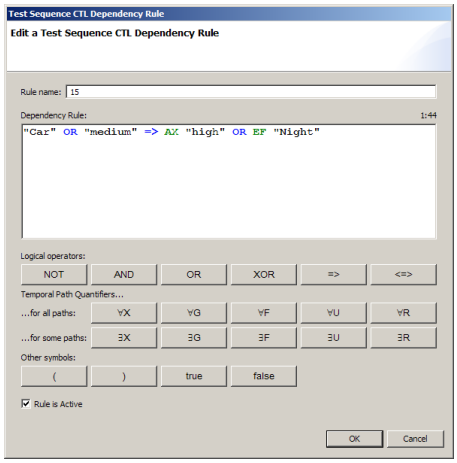
Prototyp: Generierungsvorschrift

- Algorithmus Auswahl / Laufzeit
- Gewünschter Abdeckungsgrad
- (Unter-)Menge aller Kombination von Klassen des K.-Baums
- Minimale und maximale Anzahl von Testschritten pro Testsequenz
- Definierte Menge von zugelassenen Klassenkombinationen für den ersten und den letzten Testschritt von Testsequenzen



berner & mattner
optimizing your development

Prototyp: Generierungsvorschrift



- Zusätzlich detailiertere Beschreibung möglich mittels CTL Regeln
- Gelten für Gesamtheit aller Testsequenzen
- Bedingt anwendbar auf Menge manuell erstellter Testsequenzen

Prototyp – aktuelle Einschränkungen

Folgende Parameter lassen sich nicht / nur eingeschränkt spezifizieren:

- Maximale Anzahl von Testschritt-Wiederholungen (sowohl unmittelbar aufeinander folgende als auch innerhalb der gesamten Testsequenz)
- Maximale Anzahl von Klassen-Wiederholungen (sowohl unmittelbar aufeinander folgende als auch innerhalb der gesamten Testsequenz)
- Gewünschter Abdeckungsgrad
- Transitionen

Zusammenfassung

- Ansatz für automatische Testsequenz-Generierung aus Klassifikationsbäumen
- neue Abhängigkeitsregeln
  - Obermenge der bestehenden Abhängigkeitsregeln
  - Erweitert um Ausdrucksmöglichkeiten für Zusammenhänge zwischen mehreren Testschritten
- neue Generierungsvorschriften
  - Parameter zur Definition des Testumfangs
  - Spezifikation Abdeckungsgraden und Wiederholungen
- Damit systematische Generierung von Testsequenzen zu Klassifikationsbäumen möglich

berner & mattner
optimizing your development

Ausblick

- Nach Abschluss der prototypischen Implementierung
  - Evaluierung des vorgestellten Ansatzes
  - Skalierbarkeit und Performance
- Danach reguläre Umsetzung der Testsequenz-Generierung im CTE XL Professional

berner & mattner
optimizing your development

Literatur

[1] M. Grochtmann and K. Grimm, “Classification trees for partition testing,” *Softw. Test., Verif. Reliab.*, vol. 3, no. 2, pp. 63–82, 1993.

[3] E. Lehmann and J. Wegener, “Test case design by means of the CTE XL,” *Proceedings of the 8th European International Conference on Software Testing, Analysis and Review (EuroSTAR 2000), Kopenhagen, Denmark, December*, 2000.

[4] M. P. Heimdahl, S. Rayadurgam, W. Visser, G. Devaraj, and J. Gao, “Auto-generating test sequences using model checkers: A case study,” in *3rd International Worshop on Formal Approaches to Testing of Software (FATES 2003)*, 2003.

[5] M. Y. Vardi, “Branching vs. linear time: Final showdown,” in *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS 2001, 2001, pp. 1–22.

[6] A. Krupp and W. Müller, “Modelchecking von Klassifikationsbaum-Testsequenzen,” 1 Apr. 2005, gI/ITG/GMM Workshop "Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen", München.

[10] H. Ural, “Formal methods for test sequence generation,” *Comput. Commun.*, vol. 15, pp. 311–325, June 1992.