WSR 2004, Bad Honnef, 03.05.-05.05.2004

Reverse-Engineering durch Identifikation von Eingabedaten-Äquivalenzklassen aus Programmabläufen

Rainer Schmidberger
schmidrr@informatik.uni-stuttgart.de

se

Übersicht

Worum geht es?

Es geht um das **Wiederfinden und die Nach-dokumentation** von **Entscheidungstabellen** innerhalb eines bestehenden Programmcodes zum Zweck der besseren Wartung (Decision table recovery)

Vortragsaufbau

- ❑ Motivation und Ziele
- ❑ Was sind Entscheidungstabellen?
- ❑ Begriffe
- ❑ Konzept und grundsätzliches Vorgehen
- ❑ Deco: technische Umsetzung
- ❑ Ausblick

se

Hintergrund und Motivation

- Betrachtet werden Software-Systeme aus **betriebswirtschaftlicher Domäne**, die
  - ⇨ seit mehreren Jahren **in Betrieb** sind (und auch noch weiter bleiben sollen) und
  - ⇨ deren implementierte Geschäftslogik zu einem gewichtigen Teil aus **Entscheidungstabellen** besteht,
  - ⇨ die nicht zufriedenstellend dokumentiert sind.
- Die Menge dieser Systeme ist groß, typischerweise ist der Einsatz bei **Verwaltungen und Finanzdienstleistern**.
- Die heute verfügbare Methoden- und Werkzeug-unterstützung zum Reverse-Engineering von Entscheidungstabellen ist unbefriedigend.

Was sind Entscheidungstabellen?

	Regeln				
Bedingungen	Regel 1	Regel 2	Regel 3	...	*Regel k*
Bedingung 1	T	F	T	Beding-ungen ver-wenden Eingabe-daten	F
Bedingung 2	-	T	F		F
Bedingung i	F	-	-		-
Aktionen					
Aktion 1		x	x	Aktionen erzeugen Ausgabe-daten	x
Aktion 2	x		x		
Aktion j	x				

Don't care

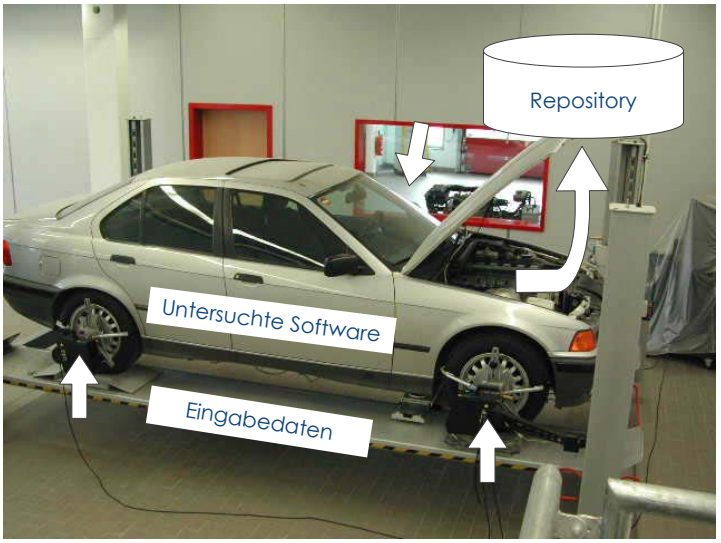
```
IF (Bedingung 1) AND (Not Bedingung i)
  Aktion 2
  Aktion j
END-IF
```

Laurence I. Press, „Conversation of Decision Tables to Computer programs", Comm. ACM 8, No. 6, 385-390 (Juni 1965)

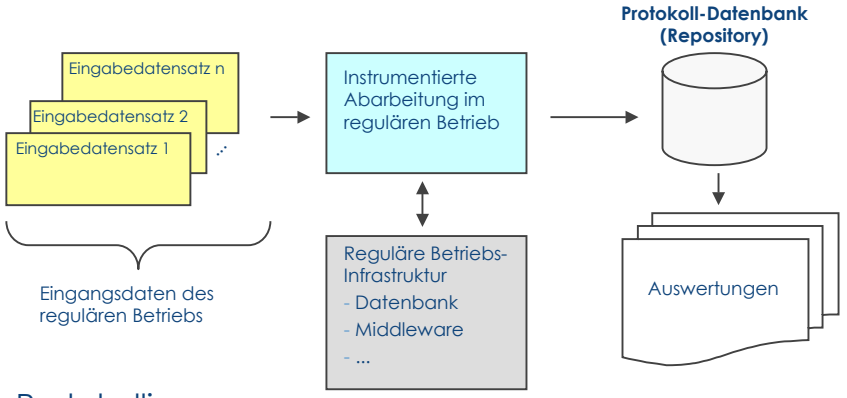
Ziele

- Automatisches Wiederfinden der Entscheidungstabellen, die dem Programmcode zugrundeliege (Decision table recovery).
- Auffinden von relevanten Sonderfällen.
- Auffinden von nicht verwendeten Programmteilen (nicht verwendet bezüglich einer gegebenen Menge von Eingabedaten).
- Die Adressaten sind die Analytiker oder Fachexperten

„Software Prüfstand“



Decision table Recovery



Protokollierung ...

- ... des durchlaufenen Pfads und
- ... aller angewendeten Terme incl. der jeweiligen Variablenwerte .

Begriffe (1)

- Mit dem **Eingabedatensatz** d_i durchläuft das Programm P deterministisch einen ganz bestimmten Programmpfad, den Pfad p_i.
- Zwei Eingabedatensätze d_i und d_j sind **schwach äquivalent**, wenn die beiden zugeordneten Pfade p_i und p_j gleich sind.
- Jeder Pfad ist bestimmt durch die Sequenz von **Entscheidungen mit deren Term**, die an den Verzweigungen getroffen werden.

Begriffe (2)

- Zwei schwach äquivalente Eingabedatensätze d_i und d_j sind **stark äquivalent**, wenn alle angewendeten Terme jeweils das gleiche Resultat ergeben.
- Der **annotierte Pfad** ist der um die ausgewerteten Terme erweiterte Programmpfad.
- **Dunkle Pfade** sind Programmpfade deren Eintrittspunkt bekannt ist, aber noch nicht durchlaufen wurden.

Grundidee

Wenn noch kein einziger Datensatz angewendet wurde, stellt das **gesamte Programm einen dunklen Pfad** dar.

Während eines Programmablaufs mit dem Eingabedatensatz d_i

⇨ erweist sich d_i als **stark äquivalent** zu einer früheren Ausführung

⇨ oder es wird ein bislang dunkler Pfad **erstmals durchlaufen**. Es entstehen an Verzweigungen **wieder dunkle Pfade**.

Vorgehen

- Das zu untersuchende Programm P wird **instrumentiert**.
- **P wird** mit den verfügbaren Eingabedatensätzen **betrieben**. **Protokollierung** des Programmpfads sowie der Terme (mit Resultat) **im Repository**.
- Repository-Auswertung:
  - ⇨ Gruppieren nach Eingabedatensätzen mit gleichen Pfaden (**schwache Äquivalenz**) und gleichen Term-Resultaten (**starke Äquivalenz**)
  - ⇨ Streichen durchlaufener dunkler Pfade
- Die Terme (Bedingungen) mit den Anweisungen (Aktionen) und Pfaden (Regeln) bilden dann die **Entscheidungstabelle**

Beispiel (1)

```
0010 IF A <= 27 OR >= 65
0020   IF INR >= 10 AND <= 20
0030*    keine MwSt.
0040   ELSE
0050*    MwSt. für Unterkunft
0060   END-IF
0070 ELSE
0080   IF INR >= 10 AND <= 20
0090*    MwSt. für Unterkunft
0100   ELSE
0110*    immer zzgl. MwSt.
0120   END-IF
0130 END-IF
0140*Ende
```

Eingabedatensatz d_1: A_1 = 42, INR_1 = 12

Annotierter Pfad ap_1 = {

0010 (A ≤ 27→F, A ≥ 65→F),

Dunkle Pfade:
ax_1 = { 0010 (A ≤ 27→*T*) }
ax_2 = { 0010 (A ≥ 65→*T*) }

0080 (INR ≥ 10→T, INR ≤ 20→T),

Dunkle Pfade:
ax_3 = { 0010 (A ≤ 27→F, A ≥ 65→F),
0080 (INR ≥ 10→F) }
ax_4 = { 0010 (A ≤ 27→F, A ≥ 65→F),
0080 (INR ≤ 20→F) }

0090 (MwSt. für Unterkunft),
0140 (Ende) }

Beispiel (2)

```
0010 IF A <= 27 OR >= 65
0020   IF INR >= 10 AND <= 20
0030*    keine MwSt.
0040   ELSE
0050*    MwSt. für Unterkunft
0060   END-IF
0070 ELSE
0080   IF INR >= 10 AND <= 20
0090*    MwSt. für Unterkunft
0100   ELSE
0110*    immer zzgl. MwSt.
0120   END-IF
0130 END-IF
0140*Ende
```

Eingabedatensatz d_2: A_2 = 18, INR_2 = 12

Annotierter Pfad ap_2 = {

0010 (A ≤ 27→T, A ≥ 65→F),

Dunkle Pfade:
ax_5 = { 0010 (A ≤ 27 → F) }
ax_6 = { 0010 (A ≥ 65 → T) }

0020 (INR ≥ 10→T, INR ≤ 20→T),

Dunkle Pfade:
ax_7 = { 0010 (A ≤ 27→T, A ≥ 65→F), 0020 (INR ≥ 10→F) }
ax_8 = { 0010 (A ≤ 27→T, A ≥ 65→F), 0020 (INR ≤ 20→F) }

0030 (keine MwSt.),
0140 (Ende) }

Beispiel: annotierte Pfade (unsortiert)

ap_1 = { 0010 (A ≤ 27→F, A ≥ 65→F), 0080 (INR ≥10→T, INR ≤ 20→T), 0090, 0140 }
ax_1 = { 0010 (A ≤ 27 → T) }
ax_2 = { 0010 (A ≥ 65 → T) }
ax_3 = { 0010 (A ≤ 27→F, A ≥ 65→F), 0080 (INR ≥10→F) }
ax_4 = { 0010 (A ≤ 27→F, A ≥ 5→F), 0080 (INR ≤ 20→F) }
ap_2 = {0010 (A ≤ 27→T, A ≥ 65→F), 0020 (INR ≥ 10→T, INR ≤ 20→T), 0030, 0140 }
ax_5 = { 0010 (A ≤ 27 → F) }
ax_6 = { 0010 (A ≥ 65 → T) }
ax_7 = { 0010 (A ≤ 27→T, A ≥ 65→F), 0020 (INR ≥ 10→F) }
ax_8 = { 0010 (A ≤ 27→T, A ≥ 65→F), 0020 (INR ≤ 20→F) }

⇨ Gruppieren nach Eingabedaten mit gleichen Pfaden (**schwache Äquivalenz**) und gleichen Term-Resultaten (**starke Äquivalenz**)

⇨ Streichen durchlaufener dunkler Pfade (z.B. ax_5)

Die resultierende Entscheidungstabelle

	Regeln						
Bedingungen	ap_2	ax_2	ax_7	ax_8	ap_1	ax_4	ax_3
A ≤ 27	T	-	T	T	F	F	F
A ≥ 65	F	T	F	F	F	F	F
I ≥ 10	T	-	F	-	T	-	F
I ≤ 20	T	-	-	F	T	F	-
Aktionen							
0010	x	x	x	x	x	x	x
0020	x		x	x			
0030	x						
0080					x	x	x
0090					x		
0130	x				x		

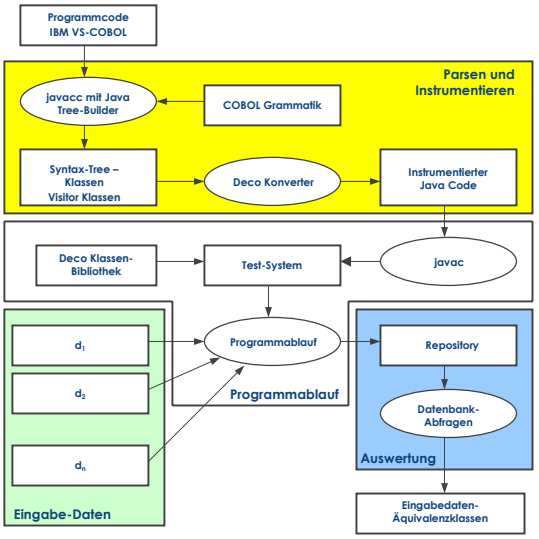
schwache Äquivalenz

schwache Äquivalenz

dunkle Pfade

dunkle Pfade

Deco: Technische Umsetzung



Einschränkungen

- Das Verfahren wird nur für Systeme sinnvoll anwendbar sein, bei denen die Eingabedaten in den Termen der Verzweigungen direkt verwendet werden
- Systeme mit stochastischen Ereignissen oder komplexer Anwender-Interaktion scheiden aus
- Der Programmcode muss vorliegen
- Das System muss im Betrieb einsetzbar sein; d.h. es müssen (möglichst viele) sinnvolle Eingabedaten vorliegen

Zusammenfassung und Aussichten

- Eingabedaten-Äquivalenzklassen können über das beschriebene Verfahren aus einem bestehenden Programmcode mit verfügbarer Grammatik sowie regulären Produktionsdaten gewonnen werden.
- Für COBOL-Programme wurde eine exemplarische Umsetzung erfolgreich implementiert.
- Der Einsatz in einem konkreten Projekt mit etwa 40 KLOC läuft derzeit.
- Automatische Behandlung von Schleifen wird noch weiter untersucht. Ebenso Auswertung von speziellen Code-Kommentaren.

Reverse-Engineering durch Identifikation von Eingabedaten-Äquivalenzklassen aus Programmabläufen

Vielen Dank!

Rainer Schmidberger
schmidrr@informatik.uni-stuttgart.de

se