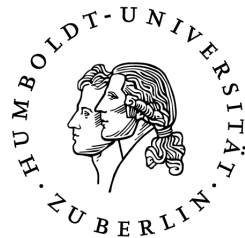


Aussagenlogische Testspezifikation



Peer Hausding
(10.06.2006)

Gliederung

Einführung

- Begriffe
- Test

Modellspezifikation

- AutoFocus

Transformation

- Spezifikation
- Testziel

Zusammenfassung

Quellen



Einführung

Aussagenlogik

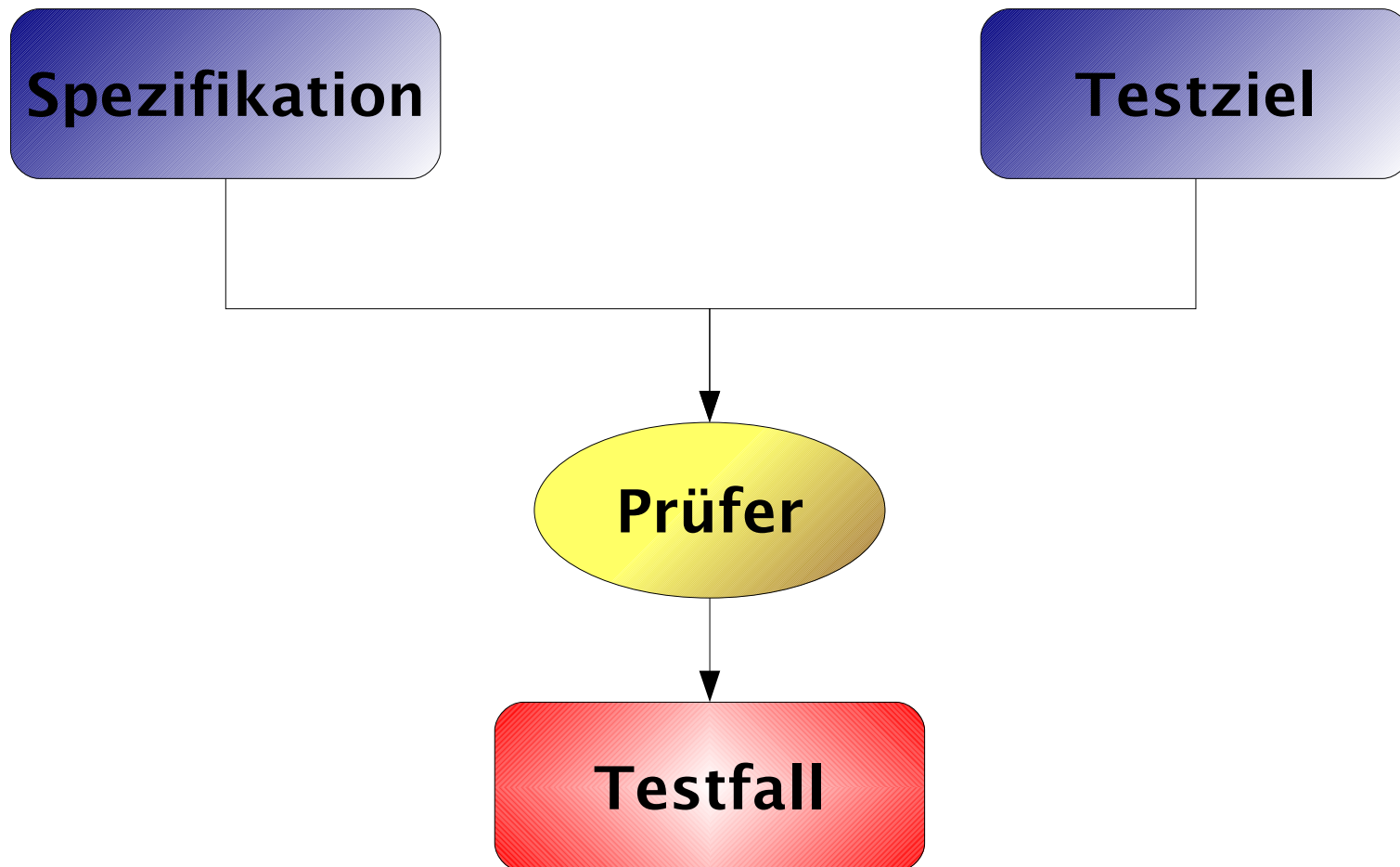
- logische Bewertung von Aussagen
- Aussagen bestehen aus mit Junktoren verknüpfte Atome (Elementaraussagen)

Prädikatenlogik

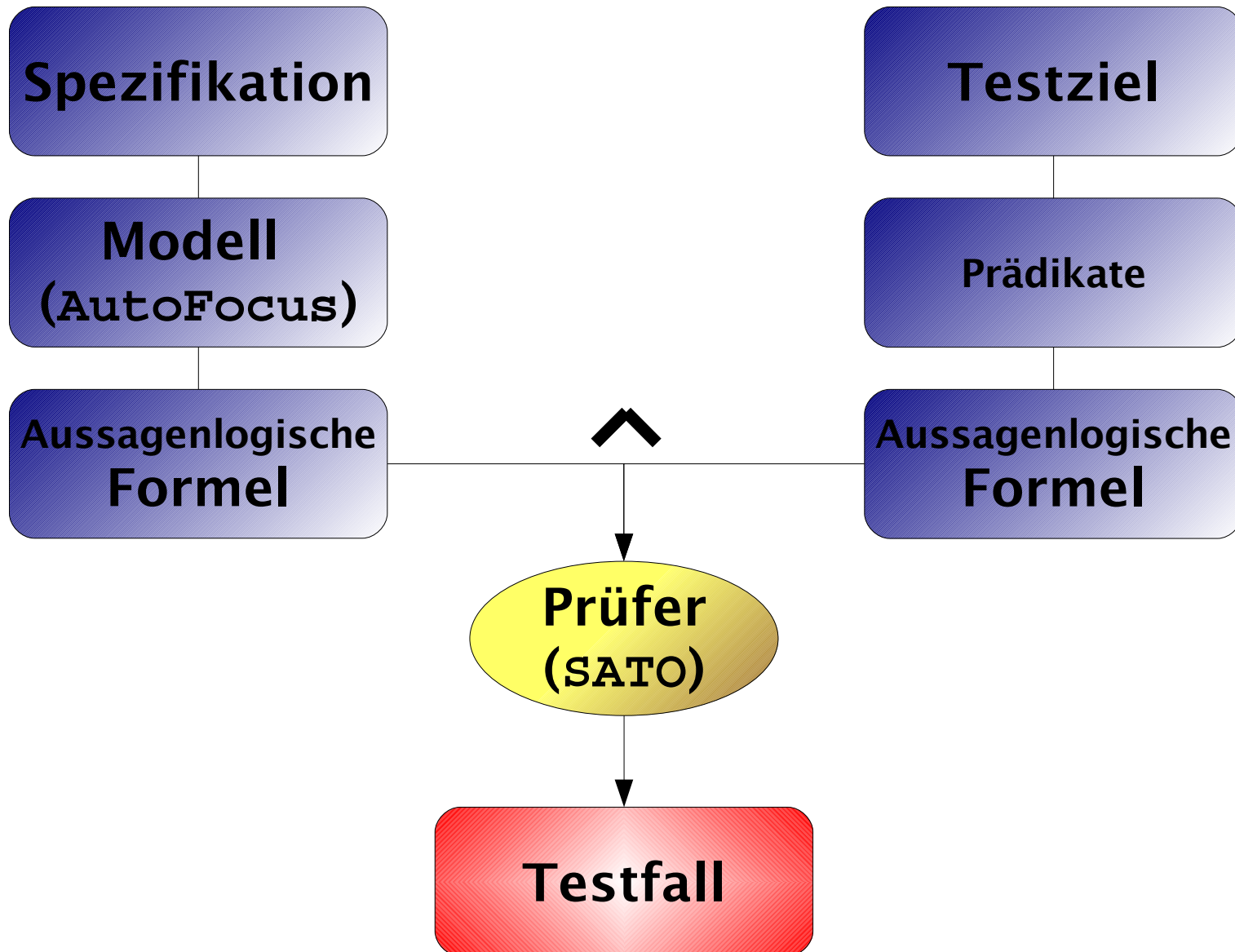
- eine Erweiterung der Aussagenlogik
- Untersuchen der inneren Struktur von Atomen

Einführung Test

Test



Einführung Test



Bounded Model Checking

- vollautomatische Verifikationstechnik
- nicht alle erreichbaren Zustände werden überprüft
- bestimmte Eigenschaft kann widerlegt werden, aber nicht verifiziert

Test anhand **aussagenlogischer Spezifikation** kann als nur testen, ob ein Aussage **NICHT** gilt, aber nicht die **Allgemeingültigkeit** überprüfen



Modellspezifikation

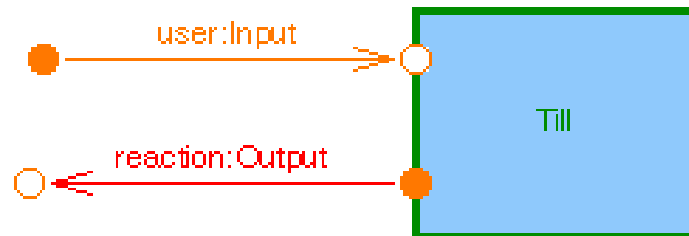
AutoFocus

- entwickelt am Lehrstuhl von Prof. Dr. Broy (Fakultät für Informatik der TU München)
- graphische Spezifikationssprache für eingebettete Systeme
- Anbindung von **QUEST** möglich
 - **QUEST** ermöglicht die Validierung & Verifizierung von Modellen in **AutoFocus**
 - zusätzliche Anbindungen möglich: SATO, SMV (**S**ymbolic **M**odel **V**erifier), CTE (**C**lassification-**T**ree **E**ditor), ...

Systemstrukturdiagramm

- Netzwerk von Komponenten eines Systems
- Dekompositionen (Subsysteme) möglich
- Kanäle als Verbindung/Übertragungsmedium
- Ports (ein- oder ausgehend)

Beispiel Geldautomat:

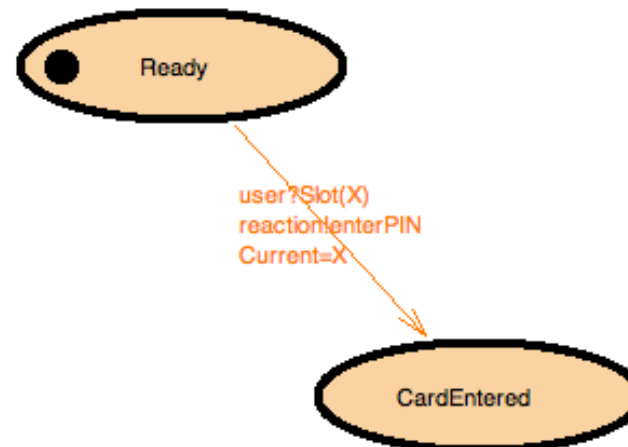


```
local Card Current=Invalid,  
local Int CurrentPIN=0
```

Zustandsübergangsdiagramm

- Verhalten der einzelnen Komponenten
- Zustände, Transitionen, Variablen
 - Precondition : Inputvariablen : Outputvariablen : Postcondition
 - `<pre> : <v_i>?<Input> : <v_o>!<Output> : <post>`

Beispiel Geldautomat:



Datentypendefinition

- Notation ist an Gofer (funktionale Programmiersprache) angelehnt

Beispiel Geldautomat:

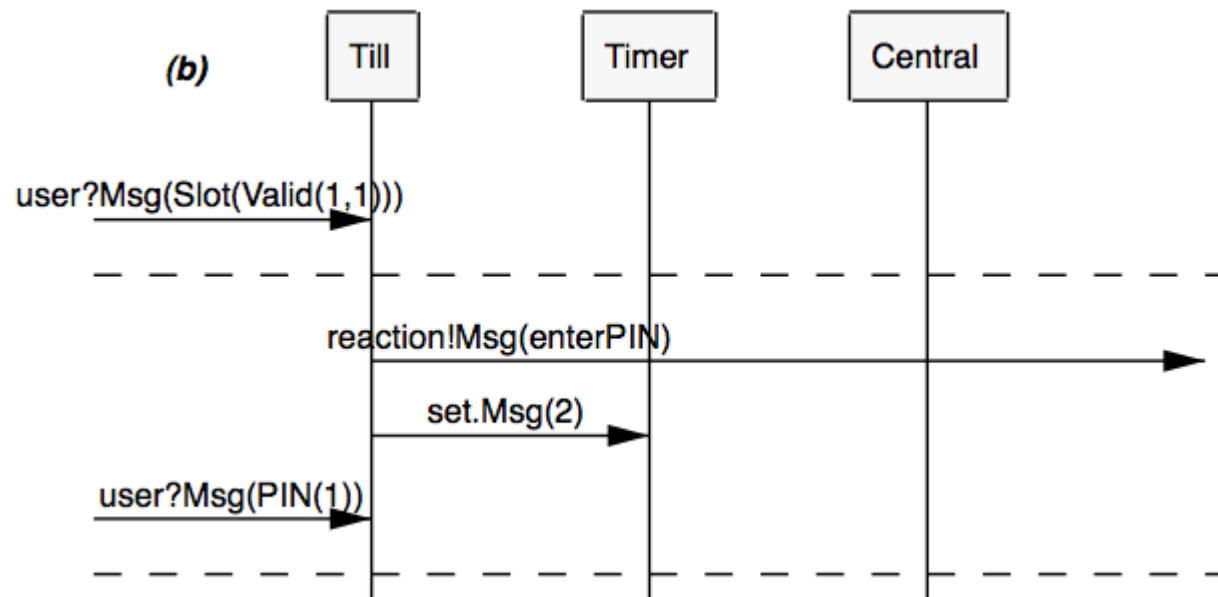
```
Card      = Invalid | Valid(getPIN:Int, Acc:Int)
Input     = Slot(Card) | FunKey(Action) | PIN(Int)
Output    = enterPIN | enterCard | enterAction
           timeoutError(Card) | byebye(money:Int, Card)
           ViewBal(Card) | errorWrongPIN
```

Message Sequence Chart (MSC)

- Standard der ITU
- textuell und graphisch (MSC/PR, MSC/GR)
- I/O Kommunikation zwischen Objekten in Abhängigkeit zur Modellzeit
 - Senden `<v_i>?Msg(<param>)`
 - Empfangen `<v_o>!Msg(<param>)`
 - Variablenwert ändern `<v>.Msg(<param>)`
- für Testziel/Testfall

MSC

Beispiel Geldautomat:





Transformation

Transformation

Aufstellen aussagenlogischer Formeln

- Formalismus zur Beschreibung von Systemmodellen
- Transformation des Systemmodells in eine Kripke Struktur

Testen mit SATO

- Implementation des Davis–Putnam–Verfahren (auf Unerfüllbarkeit testen)
- Eingabe der Spezifikation und des Testziels als aussagenlogische Formel in KNF

Kripke Struktur

- Kripke-Struktur M über AP ist ein Tupel $M = (S, S_0, R, L)$ mit:
 - S ist eine endliche Menge von Zuständen
 - $S_0 \subseteq S$ ist die Menge der Anfangszustände
 - $R \subseteq S \times S$ ist eine totale Transitionsrelation
 - $L: S \rightarrow 2^{AP}$ ist eine Beschriftungsfunktion, die jeden Zustand auf eine Menge von atomaren Aussagen (AP) abbildet
- Ein Pfad ist eine endliche Sequenz von Zuständen:
 - $\Pi : s_1 s_2 \dots$ mit $s_i \in S$ und $i \in \mathbb{N}$

Zustände (S)

- Kreuzprodukt der möglichen Zustände und aller Kanal- und Variablenbelegung

Beispiel Geldautomat:

$$\begin{aligned}\sum_{System} &= \sum_{Till} \times \dots \times \sum_{Channels} \\ \sum_{Till} &= \sum_{Till.ControlState} \times \sum_{Till.Current} \times \sum_{Till.CurrentPIN} \\ \sum_{Channels} &= \sum_{user} \times \sum_{reaction} \times \dots \\ \sum_{Till.CurrentPIN} &= B \times B \times B \times B \times B\end{aligned}$$

...

$$B = \{0, 1\}$$

Projektionsoperator Π

- bitweises Abbilden der Werte der Variablen, Kanäle und aktuellen Zustände

Beispiel Geldautomat:

$$\prod_{Till.CurrentPin_0} (s) \Leftrightarrow 0$$

$$\prod_{Till.CurrentPin_1} (s) \Leftrightarrow 0$$

...

Ausgangszustände ($I(s)$)

- Beschreiben der Startzustände, Werte der Variablen und der Kanäle

Beispiel Geldautomat:

$$\begin{aligned} I_{System}(s) = & \left(\prod_{Till.CurrentState_0} (s) \Leftrightarrow 0 \right) \wedge \left(\prod_{Till.State_1} (s) \Leftrightarrow 0 \right) \wedge \\ & \left(\prod_{Till.Current_0} (s) \Leftrightarrow 0 \right) \wedge \dots \wedge \left(\prod_{Till.Current_4} (s) \Leftrightarrow 0 \right) \wedge \\ & \left(\prod_{Till.CurrentPIN_0} (s) \Leftrightarrow 0 \right) \wedge \dots \wedge \left(\prod_{Till.CurrentPIN_3} (s) \Leftrightarrow 0 \right) \wedge \\ & \dots \end{aligned}$$

Transitionsrelation ($T(s,s')$)

- Konjunktion der Transitionsrelationen der Systemkomponenten
- Beschreiben jeder Transition t aus einem Zustand s in einen Zustand s' einer Komponente C :
 - $T_{C,t}(s,s')$

Beispiel Geldautomat:

$$T_{Till, CardIn}(s, s') = \left(\prod_{Till.CotrolState} (s) \equiv Ready^B \right) \wedge \\ \left(\prod_{Till.user} (s) \equiv Val^B \right) \wedge$$

...

... $\equiv X^B$ *bitweise Variablenvorbelegung*

Testziel

- Kodierung eines Prädikates Φ_i für ein bestimmtes Testziel
- Verbinden mit der aussagenlogischen Formel Ψ der Spezifikation
- das Ergebnis wird dann z.B. in **AutoFocus** als MSC ausgegeben



Zusammenfassung

Zusammenfassung

Aussagenlogische Testspezifikation

- Bilden von aussagenlogischen Formeln aus einer Spezifikation (z. B. **AutoFocus**)
- Formulieren von Testzielen als Prädikate
- Lösen der Formel in KNF mit einer Implementation des Davis–Putnam–Verfahren
- Analyse des Testfalls anhand des I/O traces (z. B. MSC)

Quellen

- G. Wimmel, H. Lötzbeyer, A. Pretschner, and O. Slotosch: Specification Based Test Sequence Generation with Propositional Logic. Special Issue on Specification Based Testing, Journal on Software Testing, Validation, and Reliability (STVR) 10(4):229–248, December 2000
- Hantao Zhang.
<http://www.cs.uiowa.edu/~hzhang/sato.html>
- Validas. Erfahrungsbericht Verifikationstechniken.
http://www.software-kompetenz.de/servlet/is/29286/Verifikationstechniken_2005.pdf?command=downloadContent&filename=Verifikationstechniken_2005.pdf

Quellen

- Heike Lötzbeyer. Techniken im Software-Test.
<http://www4.informatik.tu-muenchen.de/~pretschn/teaching/perlen-folien/Testen04072000.pdf>
- Wikimedia Foundation Inc.
<http://www.wikipedia.com>