

Semantiken für Statecharts

Benjamin Braatz, Markus Klein

GraGra-/PN-AG, 22.5.2003

Inhalt

- Klassische Statecharts:
 - Zustände und Konfigurationen
 - Konfigurations-Transformationen
 - Ereignisse und Aktionen
 - Schritte und Zeitmodelle
- Objekt-Orientierte Statecharts:
 - Syntax für Objekt-Orientierte Statecharts
 - Semantik mit Abstract State Machines
 - Semantik mit Graph-Transformation

Benjamin Braatz, Markus Klein

2

GraGra-AG

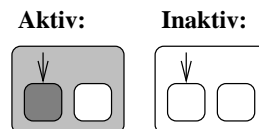
Semantiken für Statecharts

Zustände und Konfigurationen

- **Einfache Zustände:**

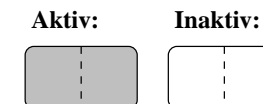


- **OR-Zustände:**



Genau ein Unterzustand eines aktiven OR-Zustands ist aktiv.
 Wenn ein OR-Zustand neu aktiv wird, wird der initiale Unterzustand aktiv.

- **AND-Zustände:**

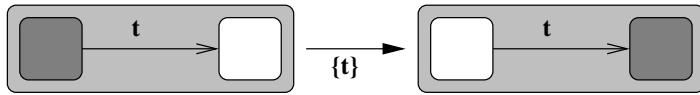


Alle Unterzustände eines aktiven AND-Zustands sind aktiv.

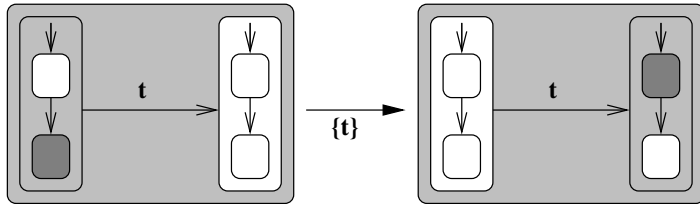
- **Konfigurationen:** Alle legalen Mengen aktiver Zustände inklusive des **Root-Zustandes**
- **History-Konnektoren:** Zuletzt aktiver statt initialer Zustand
 Shallow: Zuletzt aktiver Zustand wird initialisiert
 Deep: Der zuletzt aktive Zustand wird rekursiv aktiviert

Konfigurations-Transformationen

- Einzelne Transitionen zwischen einfachen Zuständen:



- Einzelne Transitionen zwischen OR-Zuständen:



- Relevanz einer Transitionsmenge für eine Konfiguration:

Alle Quell-Zustände in der Konfiguration enthalten

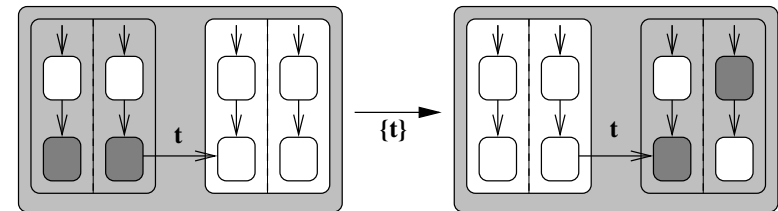
- Konsistenz einer Transitionsmenge:

Keine überlappenden Transitionen

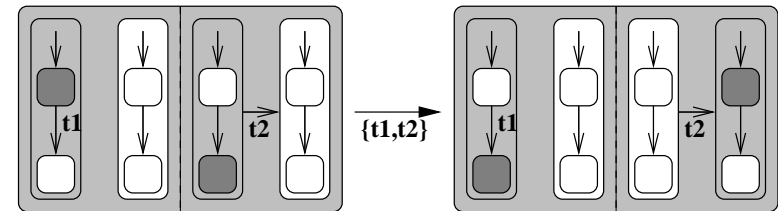
- Prioritäten bei zueinander inkonsistenten Transitionen:

- Klassisch: Transitionen mit höherem Scope haben Priorität
- UML: Transitionen mit niedrigerem Quell-Zustand haben Priorität

- Transitionen über Zustandsgrenzen hinweg:



- Mehrere Transitionen:

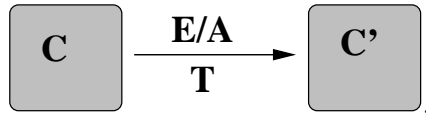


Ereignisse und Aktionen

- Gegeben: **Ereignisse** Σ und **negierte Ereignisse** $\neg\Sigma$
- **Trigger und Aktionen für Transitionen:**
 $Trigger(t) \subset \Sigma \cup \neg\Sigma$, $Actions(t) \subset \Sigma$
- **Aktivierung von Transitions Mengen für Ereignismengen:**
Selbst-Triggerung im nächsten Schritt [Harel/Naamad]:
 - Positive Trigger aller Transitionen in der Ereignismenge
 - Negative Trigger aller Transitionen nicht in der Ereignismenge
 - Aktionen erst im nächsten Schritt sichtbar
 Selbst-Triggerung im gleichen Schritt [Pnueli/Shalev]:
 - Ereignismenge plus Aktionen anderer Transitionen
 - Daher: Keine Negation einer Aktion im Trigger
 - Kausalitätsprinzip zwischen Transitionen

Schritte und Zeit-Modelle

- Bestehen aus Konfigurations-Transformationen

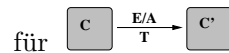


wobei T

- für C relevant und konsistent ist,
- zu C' schaltet,
- für E aktiviert und maximal priorisiert ist,
- eine maximale Anzahl Transitionen enthält und
- die Aktionen A auslöst.

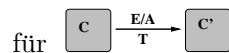
- **Synchrone Zeit:**

$$(C, E.F.in, out) \rightarrow_{sync} (C', (F \cup A).in, A.out)$$



- **Selbst-Triggerung im gleichen Schritt [Pnueli/Shalev]:**

$$(C, E.in, out) \rightarrow_{PS} (C', in, A.out)$$



- **Kompositionale Semantik nach [Lüttgen/von der Beeck/Cleveland]:**

Information über Zusammensetzung von Schritten aus einzelnen Transitionen in **Mikro-Konfigurationen**

- **Schritte** schalten zwischen Konfigurationen mit **Eingabe** und **Ausgabe**

$$in \in (\mathcal{P}(\Sigma))^* \text{ and } out \in (\mathcal{P}(\Sigma))^*$$

- **Selbst-Triggerung im nächsten Schritt [Harel/Naamad]:**

- **Asynchrone Zeit:**
Mikro-Schritte:

$$(C, E.in, out) \rightarrow_{\mu} (C', A.in, A.out)$$



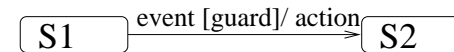
- **Super-Schritte:**

$$(C, E.in, out) \rightarrow_{sup} (C', in, A.out) \text{ für}$$

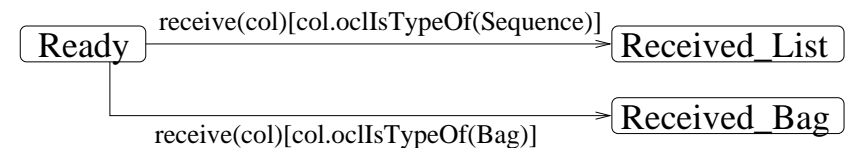
$$(C, E.in, out) \rightarrow_{\mu}^+ (C', in, A_n \dots A_1.out) \text{ und } A = \bigcup_{i=1, \dots, n} A_i$$

Syntax für OO-Statecharts

- Guards und Finale Zustände



- Beispiel



Abstract State Machines

Update	$f(\bar{s}) := t$
Conditional	if g then R else S
Blocks	do-in-parallel R_1, \dots, R_k enddo (Regeln müssen konsistent sein)
Sequential Composition	seq R, S endseq
Quantor	forall v with $g(v)$ do $R(v)$ (Konsistenz)
Loop List	loop v through list X $R(v)$

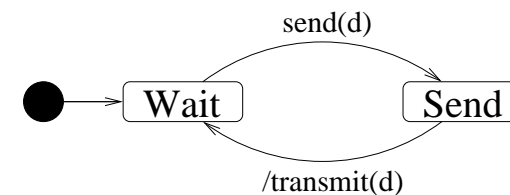
Actions

Call Action	ActionRule ($call_{op[args]}$) $outQueue(O) := outQueue(O) \uplus \{op_O[args]\}$
Send Action	ActionRule ($send_e$) $outQueue(O) := outQueue(O) \uplus \{e\}$
ReturnAction	ActionRule ($send_{return_{op(a)}}$) $outQueue(O) := outQueue(O) \uplus$ $\{trigsusy(op).return_{op(a)}\}$
Assignment	$att := exp$
Void Action	ASM Regel skip

Definition: Interaktive ASM

- Die Menge der Nachrichtennamen, **MsgNm**, enthält endliche Sequenzen $n_1.n_2.\dots.n_k$, wobei n_1 bis n_{k-2} Namen von Subsystemen sind. n_{k-1} ist der Name eines Objekts und n_k ist der lokale Name der Nachricht. (Operationen, Signale, Return-Nachrichten)
- Eine *interaktive ASM* $iA = (A, in, out)$ besteht aus einer ASM A , und zwei Mengen von Multimengennamen in, out , wobei die Regeln in A nur Elemente zu out hinzufügen dürfen – es sei denn, $elem \in out$ und $elem \in in$.

Beispiel: Sender

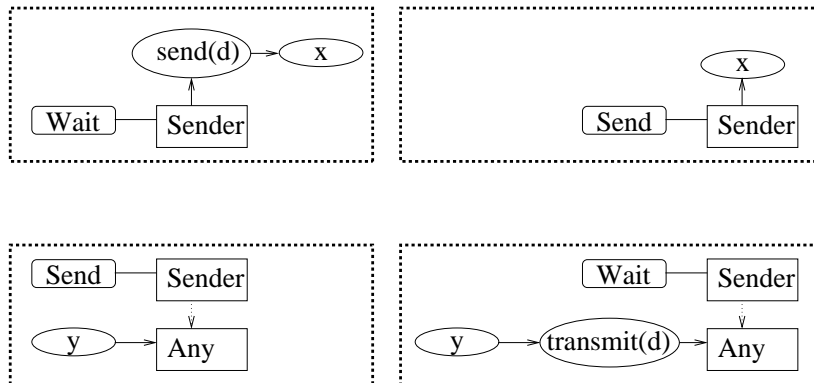


```

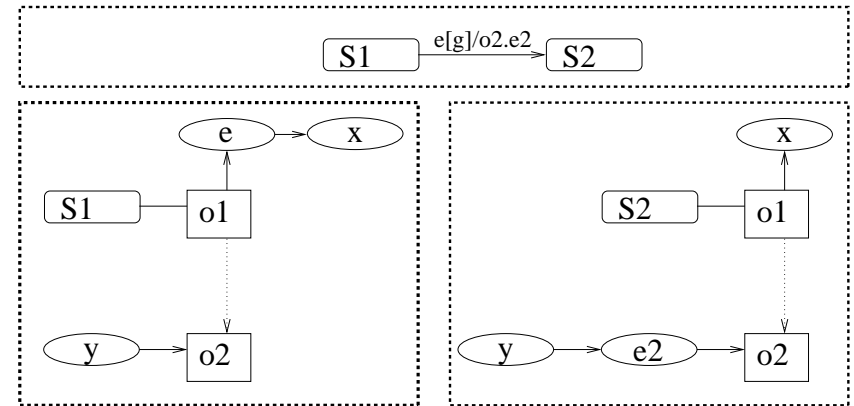
case currState of
  {Initialsender}: do currState:={Wait}
  {Wait}: do choose  $e : e \in inQueue(sender)$ 
    do-in-parallel
       $inQueue(sender) = inQueue(sender) \setminus \{e\}$ 
      if mname( $e$ ) = send then
        do-in-parallel
          currState:={Send}
           $d := Args(e)$ 
        enddo
      enddo
    enddo
  {Send}: do-in-parallel
    currState:={Wait}
     $outQueue(sender) := outQueue(sender) \uplus \{transmit(d)\}$ 
  enddo

```

Regeln für Beispiel Sender



Graph-Transformations basierte Semantik



Übersicht

ASMs	<ul style="list-style-type: none"> • keine Transitionen über Grenzen von zusammengesetzten Zuständen • synchrone und asynchrone Kommunikation
GraTra	<ul style="list-style-type: none"> • Zustände sind nur Namen • asynchrone Kommunikation

Literatur

- [HN96]** David Harel, Amnon Naamad: The STATEMATE Semantics of Statecharts, ACM Transactions on Software Engineering and Methodology, Vol. 5, No. 4, 1996
- [Joh02]** Sebastian John: Steps for Statecharts, TU Berlin – Fak. IV Bericht Nr. 2003-04
- [LBC00]** Gerald Lüttgen, Michael von der Beeck, Rance Cleaveland: A Compositional Approach to Statecharts Semantics, ICASE Report No. 2000-12

- [KGK02]** Sabine Kuske, Martin Gogolla, Hans-Jörg Kreowski, Ralf Kollmann: An Integrated Semantics for UML Class, Object and State Diagrams Based on Graph Transformation, 3rd International Conference on Integrated Formal Methods (IFM 2002)
- [Jür02]** Jan Jürjens: A UML Statecharts Semantics with Message-Passing, Symposium of Applied Computing (SAC 2002)