Verhaltensorientierte Modellierung 3

Modul 12 (v0.9)

Kanonikvorlesung: Foundations of Computing

Heiko Mantel

MAIS, TU Darmstadt, WS10/11

Motivation

Möglichkeiten der Modellierung (vergleiche Modul 4)

- Angabe eines Modells in der Sprache der Mathematik
- Angabe eines Ausdrucks in einer Spezifikationssprache, deren Semantik definiert, welche Menge von Modellen durch den Ausdruck spezifiziert wird.

Fokus dieses Moduls

- Wie definiert man eine formale Spezifikationssprache für verhaltensorientierte Modelle?
  - Syntax und Semantik

Übersicht: Modul 12

Verhaltensorientierte Modellierung

- Mengen von Spuren
- Prozesse

Entwurf einer Spezifikationssprache für Prozesse

- primitive Prozessausdrücke
- schrittweise Ausführung von Aktionen
- nicht-deterministische Auswahl
- Rekursive Spezifikation von Prozessausdrücken
- Fixpunktoperatoren
- Sequentielle Komposition
- Parallele Komposition

Unterschiede zu Modul 10 und 11

Verhaltensorientierte Modellierung in Modul 10 und 11:

Wie wurde das Verhalten eines Systems modelliert?

- Transitionssysteme TS = (S,S0,E,T) wurden als Modelle verwendet.
- Eine solche Modellierung gibt an, welche Transitionen möglich sind. Weitergehende Informationen, z.B. über die Wahrscheinlichkeit der einzelnen Ereignisse in einem Zustand werden nicht berücksichtigt.

Wie wurden die möglichen Systemläufe modelliert?

- durch eine Menge von Spuren (endliche Folgen) oder
- durch eine Menge von Historien (unendliche Folgen)

Wie wurden diese Mengen definiert?

- Sie wurden durch das Transitionssystem induziert
  - Traces(TS) bzw. Hist(TS)

Man kann die möglichen Systemläufe auch direkt durch Mengen von Spuren oder Historien modellieren, d.h. ohne Angabe eines Transitionssystems.

Unterschiede zum vorigen Modul 2

Fokus in diesem Modul

Modellierung von Systemläufen

- Beschränkung auf endliche Systemläufe
- Spuren werden als Modelle verwendet.

Modellierung aller möglichen Verhalten eines Systems

Die Menge aller möglichen Systemläufe wird durch eine Menge von Spuren modelliert.

- genauer: durch eine Menge von Ereignisspuren

Modellierung von Systemverhalten 1

Definition (aus Modul 10)

Eine **Spur** (engl.: **trace**) ist eine endliche Folge von Ereignissen und Zuständen. Wenn die Folge nur Ereignisse enthält, so nennen wir die Spur auch **Ereignisspur**. Enthält die Folge nur Zustände, so nennen wir die Spur auch **Zustandsspur**.

Definition

Eine Spur t' heißt **Präfix** einer Spur t wenn es eine Spur t'' gibt, so dass $t = t'.t''$ gilt. Wir schreiben $t' \leq t$, um auszudrücken, dass t' ein Präfix von t ist.

Beispiel

- $() \leq$ (send(u,v,m1), send(u,v,m2), recv(u,v,m1))
- (send(u,v,m1)) $\leq$ (send(u,v,m1), send(u,v,m2), recv(u,v,m1))

Modellierung von Systemverhalten 2

Definition

Ein **Prozess** P ist ein Paar **(E,Tr)**, wobei

- **E** eine Menge von Ereignissen und
- **Tr** $\subseteq E^* \cup (E^* \times \{\surd\})$ eine nicht leere Menge von Ereignisspuren ist.

Die Menge **Tr muss unter Präfixbildung abgeschlossen sein**, d.h. $\forall t, t' \in E^* \cup (E^* \times \{\surd\}) : (t \in Tr \wedge t' \leq t) \Rightarrow t' \in Tr$ muss gelten.

Intuition

- Die Menge E enthält genau die Ereignisse, die Gegebenheiten modellieren, an denen das System beteiligt ist.
- Die Menge Tr enthält genau die Spuren, die Abläufe modellieren, die für das System möglich sind.

Beachte

Zustände werden nicht explizit modelliert.

Modellierung von Systemverhalten 3

Definition

Die Funktion α liefert für einen Prozess, die Menge von Ereignissen. Für $P = (E,Tr)$ gilt also $\alpha P = E$. Die Menge αP wird als **Alphabet von P** bezeichnet.

Definition

Die Funktion **traces** liefert für einen Prozess, die Menge von Spuren. Für $P = (E,Tr)$ gilt also $traces(P) = Tr$.

Wie modelliert man das interne Verhalten eines Systems?

- E wird so definiert, dass jede Eingabeaktion, jede interne Aktion und jede Ausgabeaktion durch jeweils ein Ereignis in E modelliert wird.

Wie modelliert man die Kommunikation an einer Schnittstelle?

- E wird so definiert, dass jede Eingabeaktion und jede Ausgabeaktion durch jeweils ein Ereignis in E modelliert wird.
- Interne Aktionen werden in diesem Fall nicht modelliert.

Übersicht: Modul 12

Verhaltensorientierte Modellierung

- Mengen von Spuren
- Prozesse

Entwurf einer Spezifikationssprache für Prozesse

- primitive Prozessausdrücke
- schrittweise Ausführung von Aktionen
- nicht-deterministische Auswahl
- Rekursive Spezifikation von Prozessausdrücken
- Fixpunktoperatoren
- Sequentielle Komposition
- Parallele Komposition

Entwurf einer Spezifikationssprache 1

Ziel

Eine formale Sprache zur Spezifikation von Prozessen.

Muss-Kriterien

- Die Sprache muss eine formale Sprache sein.
- Die Sprache muss eine formal definierte Semantik haben.

Kann-Kriterien

- Die Sprache sollte möglichst ausdrucksmächtig sein.
- Die Sprache sollte möglichst natürlich sein.
  - „natürlich“ ist nur schwer objektivierbar.

Entwurf einer Spezifikationssprache 2

Ansatz

- Entwurf einer Teilsprache für endliche Menge von Spuren.
  - STOP und SKIP
  - Aktionspräfixe
  - nicht-deterministische Auswahl
- Schrittweise Einführung von Operatoren
  - rekursive Definitionen
  - Sequentielle Komposition
  - modulare Spezifikation nebenläufiger Systeme
    - Synchronisation und Verschachtelung von Spuren

$STOP_E$

Was wird durch die Spur $\langle\rangle$ modelliert?

- Die Spur $\langle\rangle$ modelliert, dass gar nichts passiert ist.

Wie modelliert man ein System, das gar nichts tut?

Durch einen Prozessausdruck aus der Familie **$STOP_E$** wobei die Funktionen α und traces wie folgt auf Prozessausdrücke erweitert werden:

- **$\alpha\, STOP_E = E$** und
- **$traces(STOP_E) = \{ \langle\rangle \}$** .

Beachte

Für jedes Alphabet E gibt es einen Prozessausdruck $STOP_E$.

Beachte

Die Menge der Spuren eines Prozesses kann nicht leer sein, da unsere Definition von Prozessen dieses ausschließt.

$SKIP_E$

Was wird durch die Spur (√) modelliert?

- Die Spur (√) modelliert, dass das System terminiert, aber ansonsten nichts passiert ist.

Wie modelliert man ein System, das unmittelbar terminiert?

Durch einen Prozessausdruck aus der Familie **$SKIP_E$** wobei

- **$\alpha\ SKIP_E = E \cup \{\surd\}$** und
- **$traces(SKIP_E) = \{ (), (\surd) \}$** .

Beachte

- Für jedes Alphabet E, gibt es einen Prozessausdruck $SKIP_E$.
- $() \in traces(SKIP_E)$ muss gelten, da die Menge ansonsten nicht unter Präfixbildung abgeschlossen wäre.
- Die Prozessausdrücke $STOP_E$ und $SKIP_E$ sind nicht äquivalent.

$x \rightarrow P$ 1

Was wird durch eine Spur (x).t modelliert?

- Eine Spur (x).t modelliert, dass zunächst das Ereignis x passiert und danach der durch die Spur t modellierte Systemlauf erfolgt.

Syntax, Intuition und Aussprache

Der Prozessausdruck **(x → P)** spezifiziert einen Prozess, der sich zunächst am Ereignis x beteiligt und sich danach wie der Prozess verhält, der durch den Prozessausdruck P spezifiziert wird.

- (x → P) ist nur dann ein zulässiger Ausdruck, wenn $x \in \alpha P$ und $x \neq \surd$ gilt.
- (x → P) wird **„x vor P“** gesprochen.

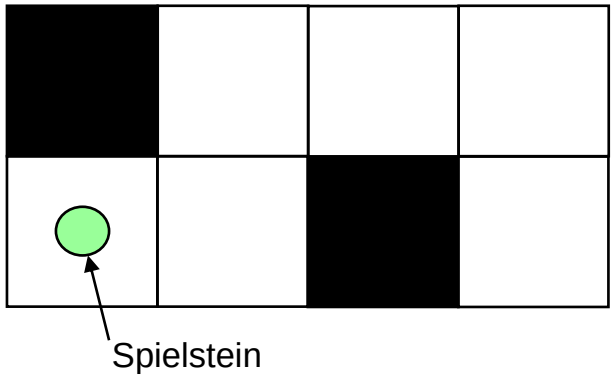
Semantik von x → P

- **α (x → P) = αP** und
- **traces((x → P)) = { () } ∪ { (x).t | t∈** traces(P) **}** .

$x \rightarrow P$

Beispiel

Wir wollen die möglichen Bewegungen eines Spielsteins auf einem Spielbrett durch einen Prozessausdruck spezifizieren, wobei der Spielstein nur nach oben oder nach rechts bewegt und dabei nicht auf schwarze Felder gezogen werden darf.
Die möglichen Bewegungen in folgender Situation:



werden durch folgenden Ausdruck spezifiziert:

- (right $\rightarrow$ (up $\rightarrow$ (right $\rightarrow$ (right $\rightarrow$ $STOP_{\{right,up\}}$)))).

$P \sqcap Q$ 1

Syntax, Intuition und Aussprache

Der Prozessausdruck **$(P \sqcap Q)$** spezifiziert einen Prozess, der sich entweder wie der Prozess verhält, der durch P spezifiziert wird, oder wie der Prozess, der durch Q spezifiziert wird.

- **$(P \sqcap Q)$** ist nur dann ein zulässiger Ausdruck, wenn $\alpha P = \alpha Q$ gilt.
- $(P \sqcap Q)$ wird **„P oder Q“** gesprochen.

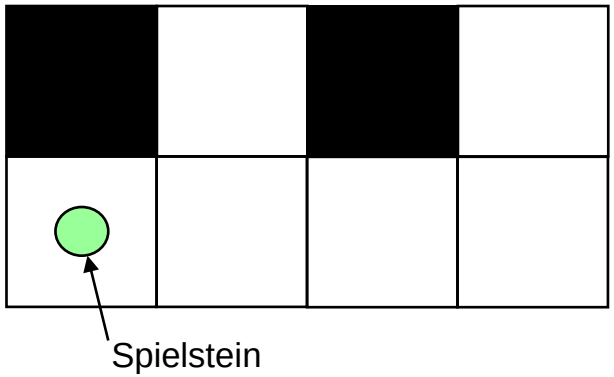
Semantik von $(P \sqcap Q)$

- **$\alpha (P \sqcap Q) = \alpha P = \alpha Q$** und
- **$traces((P \sqcap Q)) = traces(P) \cup traces(Q)$** .

$P \sqcap Q$ 2

Beispiel

Die möglichen Bewegungen, wenn man nur nach rechts und nach oben ziehen darf, in folgender Situation:



werden durch folgenden Ausdruck spezifiziert:

- $(right \rightarrow ((up \rightarrow STOP_{\{right,up\}})$
  $\sqcap (right \rightarrow (right \rightarrow (up \rightarrow STOP_{\{right,up\}}))))).$

Ausdrucksmächtigkeit 1

Theorem

Jeder Ausdruck der durch folgende BNF definierten Sprache

- P ::= $STOP_E$ | $SKIP_E$ | $(x \rightarrow P)$ | $(P \sqcap P)$

spezifiziert einen Prozess (E,Tr) mit endlicher Menge Tr.

Beweis

… wird hier ausgelassen und verbleibt als kleine Übung …

Ausdrucksmächtigkeit 2

Definition

Die Funktion **process** liefert für jedes Alphabet E und jede Spur $t \in E^* \cup (E^* \times \{\surd\})$ einen Prozessausdruck. Die Funktion ist rekursiv definiert:

- $process(E,t) = STOP_E$ wenn $t = ()$
- $process(E,t) = SKIP_E$ wenn $t = (\surd)$
- $process(E,t) = (x \rightarrow process(E,t'))$ wenn $t = (x).t'$ und $t \neq (\surd)$

Theorem

Für die Menge process(E,t) gilt:

- $E \cup \{\surd\} = \alpha(process(E,t)) \cup \{\surd\}$
- $t \in traces(process(E,t))$
- $traces(process(E,t)) = \{ t' \in E^* \cup (E^* \times \{\surd\}) \mid t' \leq t \}$

Beweis

… wird hier ausgelassen …

Ausdrucksmächtigkeit 3

Theorem

Jeder Prozess (E,Tr) mit endlicher Menge Tr kann durch einen Ausdruck in der folgenden Sprache spezifiziert werden:

- $P ::= STOP_E \mid SKIP_E \mid (x \rightarrow P) \mid (P \sqcap P)$

Beweis

- Sei $Tr \subseteq E^* \cup (E^* \times \{\surd\})$ nicht leer, endlich und unter Präfixbildung abgeschlossen.
- Wir beweisen die Aussage per Induktion über die Kardinalität von Tr.

Induktionsanfang ($|Tr| = 1$)

- Da Tr unter Präfixbildung abgeschlossen ist, muss $Tr=\{()\}$ gelten.
- Der Ausdruck $STOP_E$ spezifiziert daher den Prozess (E,Tr).

Induktionsschritt

- Die Aussage gelte für alle endlichen Menge von Spuren, die unter Präfixbildung abgeschlossen sind, und Kardinalität n haben.
- Wir betrachten eine beliebige Menge Tr mit $|Tr| = n+1 > 1$.

Ausdrucksmächtigkeit 4

Beweis (Fortsetzung)

- Da Tr endlich ist, gibt es mindestens eine Spur $t \in Tr$, die kein echtes Präfix einer anderen Spur in Tr ist.
- Die Menge $Tr\backslash\{t\}$ ist daher unter Präfixbildung abgeschlossen und hat die Kardinalität n.
- Per Induktionsannahme gibt es daher einen Prozessausdruck P, der den Prozess $(E, Tr\backslash\{t\})$ spezifiziert.
- Der Prozess (E, Tr) kann daher durch den Prozessausdruck $P \sqcap process(E,t)$ spezifiziert werden.

Erweiterung der Ausdruckmächtigkeit

Wie modelliert man ein unendlich lange laufendes System?

- Durch einen Prozess mit einer unendlichen Menge von Spuren.
- Ein solcher Prozess kann nicht in der bisher eingeführten Spezifikationssprache spezifiziert werden, wie die Theoreme zur Ausdrucksmächtigkeit zeigen.

Beispiel

Wir betrachten einen Taktgeber, der nicht terminieren soll.

- Als Alphabet wählen wir $\alpha CLOCK = \{ tick \}$, wobei das Ereignis **tick** einen einzelnen Takt modellieren soll.
- Folgende Spuren sind möglich:
  - **(), (tick), (tick, tick), (tick, tick, tick), (tick, tick, tick, tick),** …

Wie können wir die Spezifikationssprache erweitern, so dass auch Systeme mit unendlich vielen möglichen Systemläufen spezifiziert werden können?

Rekursive Gleichungssysteme 1

Ansatz

Ein Prozess wird durch einen Prozessausdruck aus der Sprache

- $P ::= STOP_E \mid SKIP_E \mid (x \rightarrow P) \mid (P \sqcap P) \mid Id$

und durch eine Menge von Gleichungen der Form

- $\{ id =_E P \mid id \in Id \}$

spezifiziert, wobei Id eine Menge von Prozessbezeichnern ist und E das Alphabet des durch id bezeichneten Prozesses angibt. $STOP_E$, $SKIP_E$, $\sqcap$ und Ereignisse dürfen nicht als Prozessbezeichner verwendet werden, d.h. $\forall E: STOP_E, SKIP_E, \sqcap \notin Id \wedge Id \cap E = \emptyset$.

Problem

Nicht jedes Gleichungssystem hat eine Lösung. Betrachte z.B.

- $INCONS\text{-}ALPHABET =_{\{x\}} (x \rightarrow STOP_{\{x\}})$
- $INCONS\text{-}ALPHABET =_{\{x,y\}} (x \rightarrow STOP_{\{x,y\}})$

Lösung und Konvention

Wir lassen nur Gleichungssysteme zu, in denen für jeden Prozessbezeichner $id \in Id$ genau eine Gleichung enthalten ist, in der id auf der linken Seite auftritt.

Rekursive Gleichungssysteme 2

Problem

Es gibt Gleichungssysteme, die mehr als eine Lösung haben.
Betrachte z.B. das Gleichungssystem

- $X =_E X$

oder das Gleichungssystem

- $X =_E Y$
- $Y =_E X$.

Beide Gleichungssysteme haben unendlich viele Lösungen.

Lösung und Konvention

Wir lassen nur Gleichungen zu, deren rechte Seite **„bewacht“** ist (engl.: **guarded**). D.h. für jedes Vorkommen eines Bezeichners id´ auf der rechten Seite einer jeden Gleichung $id =_E P$ muss es einen Teilausdruck $(x \rightarrow P')$ von P geben, der das Vorkommen von id´ enthält. Das Ereignis x heißt dann auch **Wächter** (engl.: **guard**).

Rekursive Gleichungssysteme 3

Beispiel

Das Gleichungssystem

- **Clock =$_{\{tick\}}$ (tick $\rightarrow$ Clock)**

ist zulässig für die Menge Id = { Clock } , da es zu jedem Bezeichner in Id (es gibt ja nur einen) genau eine Gleichung gibt, auf deren linker Seite der Bezeichner Clock auftritt, und da das Vorkommen von Clock auf der rechten Seite der einzigen Gleichung durch das Ereignis tick bewacht wird.

Rekursive Gleichungssysteme 4

Wie kann man Gleichungen in einer Spezifikation vermeiden?

- durch Einführen eines Fixpunktoperators

Syntax, Intuition und Aussprache

Sei F eine Funktion, die für jeden Prozessbezeichner id∈ Id einen Prozessausdruck liefert. Dann spezifiziert **μX:E.F(X)** einen Prozess X mit Alphabet E, der sich entsprechend des Ausdrucks F(X) verhält. Beachte, dass es sich hierbei um eine rekursive Definition handelt, da F(X) wiederum den Bezeichner X enthalten kann.

- μX:E.F(X) ist nur dann ein zulässiger Ausdruck, wenn F(X) nur Ereignisse enthält, die in der Menge E enthalten sind, und wenn jedes Vorkommen von X in F(X) bewacht ist.

Semantik von μX:E.F(X)

- **α (μX:E.F(X))** = E und
- **traces(μX:E.F(X))** ist die Menge der Spuren der Semantik der Lösung der Gleichung $X =_E F(X)$.

Beispiel

Der Taktgeber wird durch **μX:{tick}. (tick → X)** spezifiziert.

P ; Q

Syntax, Intuition und Aussprache

Der Prozessausdruck **(P ; Q)** spezifiziert einen Prozess, der sich zunächst wie der Prozess verhält, der durch P spezifiziert wird, und nachdem dieser Prozess terminiert wäre sich wie der Prozess verhält, der durch Q spezifiziert wird.

- **(P ; Q)** ist nur dann ein zulässiger Ausdruck, wenn $\alpha P = \alpha Q$ gilt.
- (P ; Q) wird **„P und danach Q“** gesprochen.

Semantik von (P ; Q)

- α **(P ; Q) =** α**P =** α**Q** und
- **traces((P ; Q)) = { s** $\in$ **traces(P)** | $\surd$ **kommt in s nicht vor }**
  $\cup$ **{ s.t** | **s.(**$\surd$**)** $\in$ **traces(P)** $\wedge$ **t** $\in$ **traces(Q)}** .

P || Q

1

Syntax, Intuition und Aussprache

Der Prozessausdruck **(P || Q)** spezifiziert einen Prozess, der die Prozesse, die durch P und Q spezifiziert werden, nebenläufig ablaufen lässt, wobei sich die Prozesse bei Ereignissen, die in beiden Alphabeten vorkommen synchronisieren.

- **(P || Q)** ist auch dann ein zulässiger Ausdruck, wenn $\alpha P \neq \alpha Q$!
- (P || Q) wird **„P parallel Q“** gesprochen.

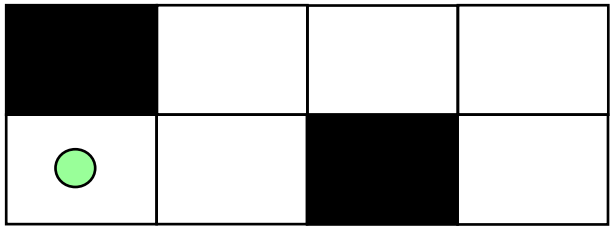
Semantik von (P || Q)

- α **(P || Q) =** α**P** $\cup$ α**Q** und
- **traces((P || Q)) = {** $t \in (\alpha P \cup \alpha Q)^*$ **|** $(t \upharpoonright (\alpha P)) \in$ **traces(P)** $\wedge$ $(t \upharpoonright (\alpha Q)) \in$ **traces(Q)}** .

P || Q

Beispiel

Wir teilen die Spezifikation in eine Spezifikation des Spielbretts und in eine Spezifikation der Regeln auf. Das Brett



kann für E = { right, left, up, down } wie folgt spezifiziert werden:

- **Board $=_E$ (right $\rightarrow$ Board-1)**
- **Board-1 $=_E$ ((left $\rightarrow$ Board) $\sqcap$ (up $\rightarrow$ Board-2))**
- **Board-2 $=_E$ ((down $\rightarrow$ Board-1) $\sqcap$ (right $\rightarrow$ Board-3))**
- **Board-3 $=_E$ ((left $\rightarrow$ Board-2) $\sqcap$ (right $\rightarrow$ Board-4))**
- **Board-4 $=_E$ ((left $\rightarrow$ Board-3) $\sqcap$ (down $\rightarrow$ Board-5))**
- **Board-5 $=_E$ (up $\rightarrow$ Board-4)**

und die Regel „ziehe nur nach rechts oder nach oben durch“

- **Rule $=_E$ ((right $\rightarrow$ Rule) $\sqcap$ (up $\rightarrow$ Rule))**.

Die möglichen Zugfolgen auf dem Spielbrett können dann mit Hilfe der parallelen Komposition wie folgt spezifiziert werden:

- **(Board || Rule)**

P ||| Q

1

Definition

Seien $t,u \in E^* \cup (E^* \times \{\surd\})$ zwei Spuren. Die **Menge der Verschachtelungen** von t und u ist wie folgt rekursiv definiert:

- Interleaving(t,u) = { t } falls u = () gilt
- Interleaving(t,u) = { u } falls t = () gilt
- Interleaving(t,u) = { (x).s | $s \in$ Interleaving(t´,u) }
  $\cup$ { (y).s | $s \in$ Interleaving(t,u´) }
  falls t = (x).t´ und u = (y).u´ gilt

P ||| Q

Syntax, Intuition und Aussprache

Der Prozessausdruck **(P ||| Q)** spezifiziert einen Prozess, der die durch P und Q spezifizierten Prozesse nebenläufig abarbeitet, **ohne** die Prozesse zu synchronisieren.

- **(P ||| Q)** ist nur dann ein zulässiger Ausdruck, wenn $\alpha P = \alpha Q$ gilt.
- (P ||| Q) wird **„P verschachtelt Q“** gesprochen.

Semantik von (P ||| Q)

- α **(P ||| Q) =** α**P =** α**Q** und
- **traces((P ||| Q)) = { s** $\in$ **(**α**P** $\cup$ α**Q)*** | $\exists$**t** $\in$ **traces(P):** $\exists$**u** $\in$ **traces(Q):**
  s $\in$ **Interleaving(t, u)}** .

Rückblick

Einige wesentliche Lernziele dieses Moduls

- Einsatz von Prozessen zur verhaltensorientierten Modellierung
- Definition einer einfachen, formalen Spezifikationssprache
  - Syntax und Semantik
- Kenntnis einiger Gefahren bei der Definition einer Spezifikationssprache wie z.B. bei rekursiven Definitionen
- Fähigkeit zur Spezifikation von Systemen in der definierten Spezifikationssprache

Literatur

C. A. R. Hoare

Communicating Sequential Processes; Prentice Hall 1985.

- enthält die hier definierte Spezifikationssprache

Einen genaueren Einblick in die unterliegende Theorie bieten z.B.

- Kapitel 2.8 - Lösungen rekursiver Gleichungssysteme
- Kapitel 3.9 - Verfeinerung der unterliegenden Modelle für Prozesse, die nicht deterministisch sind