Modellierung, Spezifikation und Semantik

Zusammenfassung

Fabian Damken

8. November 2023

TECHNISCHE UNIVERSITÄT DARMSTADT

Inhaltsverzeichnis

1 Einführung

1.1 Modellierungsbeispiele

Ein Modell ist, wie in den folgenden Beispielen deutlich zu erkennen ist, absichtlich nicht originalgetreu, und unterschiedliche Modelle heben unterschiedliche Aspekte der Realität hervor.

Liniennetzplan im Personennahverkehr

Hervorgehoben

- Linien
- Haltestellen
- Tarifzonen

Weggelassen

- Verbindungen außerhalb des abgebildeten Bereiches
- Andere Verkehrsmittel
- Geographische Gegebenheiten (z.B. Entfernungen, Höhenunterschiede, ...)

Abfahrtsplan im Personennahverkehr

Hervorgehoben

- Abfahrtszeiten
- Nächste Haltestellen
- Linie
- Aktuelle Haltestelle
- Richtung (Zielhaltestelle)

Weggelassen

- Gegenrichtung
- Andere Haltestellen
- Andere Busse an der aktuellen Haltestelle

1.2 Beziehung Realität $\leftrightarrow$ Modell

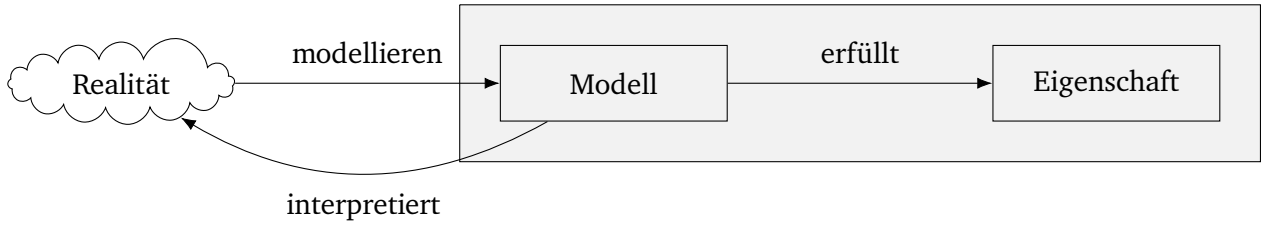

Abbildung 1.1: Beziehung Realität $\leftrightarrow$ Modell

1.3 Formale Modelle

Erklärung formaler Modelle anhand eines Beispiels:

> „Ein Mann steht mit einem Wolf, einer Ziege und einem Kohlkopf am linken Ufer eines Flusses, den er überqueren will. Er hat ein Boot, das gerade groß genug ist, ihn und ein weiteres Objekt zu transportieren, so dass er immer nur eines der drei mit sich hinübernehmen kann. Falls der Mann allerdings den Wolf mit der Ziege oder die Ziege mit dem Kohlkopf an einem Ufer zurücklässt, wird einer gefressen.“ [Kastens, Kleine Büning: Modellierung, 2008]

1.3.1 Modellierung

***Identifikation* relevanter Objekte im Szenario**

- Mann, Wolf, Ziege, Kohlkopf

***Modellierung* der Objekte durch eine Menge von Symbolen** Symbolmenge:

$$O = \{m, w, z, k\}$$

***Identifikation* relevanter Aspekte einer Situation im Szenario**

- Welches Objekt befindet sich an welchem Ufer?

***Modellierung* einer Situation durch ein Paar (L, R) von Mengen**

- Die Menge L enthält genau die Symbole, welche Objekte modellieren, welche sich momentan am linken Ufer befinden.
- Die Menge R enthält genau die Symbole, welche Objekte modellieren, welche sich momentan am rechten Ufer befinden.

***Identifikation* von Beschränkungen an möglichen Situationen**

- Keines der Objekte kann sich an mehr als einem Ufer befinden
- Der Mann und der Wolf können nicht verschwinden

***Modellierung* möglicher Zustände durch Menge von Paaren** Zustandsmenge:

$$Z \coloneqq \{(L, R) \mid L \subseteq O \land R \subseteq O \land L \cap R = \emptyset \land \{m, w\} \subseteq L \cup R\}$$

***Identifikation* relevanter Aktionen im Szenario**

- Transport von Objekten mit dem Boot zum anderen Ufer

***Identifikation* von Beschränkungen an relevanten Aktionen**

- Das Boot kann nicht ohne den Mann fahren
- Das Boot kann den Mann und maximal ein weiteres Objekt aufnehmen

***Modellierung* relevanter Aktionen durch eine Menge von Termen** Ereignismenge:

$$E \coloneqq \{\text{fahre}(T) \mid T \subseteq O \land m \in T \land |T| \leq 2\}$$

***Identifikation* der Vorbedingungen der Aktionen**

- Die durch fahre(T) modellierte Aktion ist zulässig, wenn alle Objekte in T an einem Ufer sind und nicht Wolf und Ziege oder Ziege und Kohlkopf zurückbleiben

***Identifikation* von Nachbedingungen der Aktionen**

- Nach dem Ereignis fahre(T) befinden sich die Objekte in T am anderen Ufer

***Modellierung* der Effekte von Aktionen durch Zustandsübergänge** Zustandsübergangsrelation $\to \subseteq Z \times E \times Z$

$$\begin{aligned}\to \coloneqq \{((L, R), \text{fahre}(T), (L', R')) \mid\ & (T \subseteq L \land L' = L \setminus T \land R' = R \cup T \land \{z, w\} \not\subseteq L' \land \{z, k\} \not\subseteq L') \\ & \lor (T \subseteq R \land R' = R \setminus T \land L' = L \cup T \land \{z, w\} \not\subseteq R' \land \{z, k\} \not\subseteq R')\}\end{aligned}$$

Interpretation Der Zustand (L, R) resultiert durch das Ausführen der Aktion fahre(T) im Zustand (L', R').

Notation (Zustandsübergang) Schreibe $(L, R) \xrightarrow{\text{fahre}(T)} (L', R')$ statt $((L, R), \text{fahre}(T), (L', R')) \in \to$

2 Grundlagen

2.1 Mengen

Teilmenge und Gleichheit

Definition (Teilmenge) Eine Menge N ist eine Teilmenge einer Menge M ($N \subseteq M$) gdw. für jedes $a \in N$ auch $a \in M$ gilt.

Definition (Gleichheit) Zwei Mengen M, N sind gleich ($M = N$), gdw. $M \subseteq N$ und $N \subseteq M$ gelten.

Kartesisches Produkt

Definition (Kartesisches Produkt) Das kartesische Produkt zweier Mengen M, N ist die Menge aller Paare bestehend aus einem Element aus M als erstes Element und einem Element aus N als zweites Element, also $M \times N \coloneqq \{(x, y) \mid x \in M \land y \in N\}$.

Vereinigung, Schnitt und Differenz

Definition (Vereinigung, Schnitt, Differenz)

Vereinigung Die Vereinigung zweier Mengen M, N ist $M \cup N \coloneqq \{x \mid x \in M \lor x \in N\}$.

Schnitt Der Schnitt zweier Mengen M, N ist $M \cap N \coloneqq \{x \mid x \in M \land x \in N\}$. Gilt $M \cap N = \emptyset$, so sind M und N disjunkt.

Differenz Die Differenz zweier Mengen M, M ist $M \setminus N \coloneqq \{x \mid x \in M \land x \notin N\}$.

Disjunkte Vereinigung

Definition (Disjunkte Vereinigung) Sei $I \coloneqq \{1, \cdots, n\}$ eine Indexmenge mit $n > 1$. Dann ist die disjunkte Vereinigung von Mengen $M(1), \cdots, W(n)$ wie folgt definiert:

$$W(1) \uplus \cdots \uplus W(n) \coloneqq \{(i, w) \mid i \in I \land w \in W(i)\}$$

Mehrstellige Operationen

Definition (Kartesische Produkte) Das kartesische Produkt $\times_{i\in\{1,\cdots,n\}M(i)}$ einer endlichen Folge $M(1),\cdots,M(n)$ von Mengen ist wie folgt induktiv definiert:

$$\times_{i\in\{1,\cdots,n\}}M(i) \coloneqq \begin{cases}\{()\} & \text{falls } n=0\\ M(1) & \text{falls } n=1\\ (\times_{j\in\{1,\cdots,n-1\}}M(j))\times M(n) & \text{falls } n>1\end{cases}$$

Die Elemente des kartesischen Produkts $\times_{i\in\{1,\cdots,n\}}M(i)$ heißen n-Tupel.

Notation (Kartesische Produkte)

- Def.: $M(1)\times\cdots\times M(n) \iff \times_{i\in\{1,\cdots,n\}}M(i)$
- Def.: $M^n \iff \times_{i\in\{1,\cdots,n\}}M(i)$ falls $M(i)=M$ für alle $i\in\{1,\cdots,n\}$

Notation (Tupel)

- Def.: $(m_1,m_2,\cdots,m_n) \iff (((m_1,m_2),\cdots),m_n)$

Definition (Vereinigung/Schnitt) Die Vereinigung $\bigcup_{i\in\{1,\cdots,n\}}M(i)$ und der Schnitt $\bigcap_{i\in\{1,\cdots,n\}}M(i)$ einer nichtleeren, endlichen Folge $M(1),\cdots,M(n)$ von Mengen ist induktiv definiert:

$$\bigcup_{i\in\{1,\cdots n\}}M(i) \coloneqq \begin{cases}\emptyset & \text{falls } n=0\\ M(1) & \text{falls } n=1\\ (\bigcup_{j\in\{1,\cdots,n-1\}}M(j))\cup M(n) & \text{falls } n>1\end{cases}$$

$$\bigcap_{i\in\{1,\cdots n\}} \coloneqq \begin{cases}M(1) & \text{falls } n=1\\ (\bigcap_{j\in\{1,\cdots,n-1\}}M(j))\cap M(n) & \text{falls n > 1}\end{cases}$$

Notation (Vereinigung/Schnitt)

- Def.: $M(1)\cup\cdots\cup M(n) \iff \bigcup_{i\in\{1,\cdots,n\}}M(i)$
- Def.: $M(1)\cap\cdots\cap M(n) \iff \bigcap_{i\in\{1,\cdots,n\}}M(i)$

Potenzmengen

Definition (Protenzmengen) Die Potenzmenge einer Menge M ist die Menge aller Teilmengen von M, also $\mathcal{P}(M)\coloneqq\{N\mid N\subseteq M\}$.

Indexmengen

Beschreibung Indexmengen können verwendet werden, wenn Elemente einer Menge vervielfacht werden sollen.

Beispiel In einem Doppelkopfspiel repräsentiert die unterspezifzierte Menge *SPIELSET* alle möglichen Kartentypen und die Menge *INDEX* $:= \{1, 2\}$ eine Indexmenge. Dann modelliert die Menge *KARTEN* $:=$ *SPIELSET* $\times$ *INDEX* die Menge aller Karten, wobei jeder Kartentyp zweimal vorhanden ist. Bspw. modelliert $(\text{König}, 2)$ den zweiten König im Spiel.

Notation (Indexmengen)

- $T_i \in$ *KARTEN* $\iff (T, i) \in$ *KARTEN*, $T \in$ *SPIELSET*, $i \in$ *INDEX*

Kardinalität

Definition (Kardinalität) Die Kardinalität $|M|$ einer endlichen Menge M ist die Anzahl ihrer Elemente.

Beobachtungen

- $|M| \leq |M \cup N|$, $|N| \leq |M \cup N|$, $|M| + |N| \geq |M \cup N|$
- $|M| \geq |M \cap N|$, $|N| \geq |M \cap N|$
- $|M| - |N| \leq |M \setminus N|$, $|M| \geq |M \setminus N|$
- $|M| \cdot |N| = |M \times N|$
- $2^{|M|} = |\mathcal{P}(M)|$
- Unabhängig von der Kardinalität einer nichtleeren Menge M haben die Mengen M^+ und M^* unendlich viele Elemente.
- $(|B| + 1)^{|D|} = |D \rightharpoonup B|$

Multimengen

Definition (Multimengen) Eine Multimenge über einer Menge M ist eine Funktion $m : M \to \mathbb{N}_0$.

Verwendung Multimengen eignen sich, um Ressourcen zu modellieren (bspw. der Preis von etwas).

(Partiell) Geordnete Mengen

Definition ((Partiell) Geordnete Mengen) Eine partiell geordnete Menge ist ein Paar $(M, \leq)$, wobei

- M eine Menge ist, die Trägermenge genannt wird, und
- $\leq \subseteq M \times M$ eine partielle Ordnung ist.

Eine geordnete Menge ist eine partiell geordnete Menge $(M, \leq)$, wobei $\leq$ eine totale Ordnung ist.

Visualisierung (Hasse-Diagramme) Partiell geordnete Mengen können durch Hasse-Diagramme als Graph visualisiert werden. Das Hasse-Diagramm für eine partiell geordnete Menge $(M, \leq)$ zeichnet ergibt sich wie folgt:

- Jedes $m \in M$ ist ein Knoten.
- Wenn $m_1 \leq m_2$ für zwei Knoten $m_1, m_2 \in M$ gilt, dann
  - wird der Knoten m_2 oberhalb von m_1 gezeichnet und
  - falls es kein $m \in M$ mit $m_1 \leq m, m \leq m_2, \neg m \neq m_1$ und $m \neq m_2$ gibt, dann werden m_1 und m_2 mit einer Kante verbunden.

Obere/Untere Schranken

Definition (Obere/Untere Schranken) Sei $(M, \leq)$ eine geordnete Menge.

- Eine Funktion $\sqcup : M \times M \to M$ heißt *Vereinigungsoperator* gdw. $m_1 \sqcup m_2$ die *kleinste obere Schranke* für $m_1, m_2 \in M$ ist, d.h.
  - $m_1 \leq (m_1 \sqcup m_2)$ und $m_2 \leq (m_1 \sqcup m_2)$ und
  - für alle $m \in M$, wenn $m_1 \leq m$ und $m_2 \leq m$, dann $(m_1 \sqcup m_2) \leq m$.
- Eine Funktion $\sqcap : M \times M \to M$ heißt *Schnittoperator* gdw. $m_1 \sqcap m_2$ die *größte unter Schranke* für $m_1, m_2 \in M$ ist, d.h.
  - $(m_1 \sqcap m_2) \leq m_1$ und $(m_1 \sqcap m_2) \leq m_2$ und
  - für alle $m \in M$, wenn $m \leq m_1$ und $m \leq m_2$, dann $m \leq (m_1 \sqcap m_2)$.

Verbände

Definition (Verbände) Ein Verband ist ein Tupel $(M, \leq, \sqcup, \sqcap)$, sodass

- $(M, \leq)$ eine geordnete Menge ist,
- $\sqcup$ ein Vereinigungsoperator auf $(M, \leq)$ ist und
- $\sqcup$ ein Schnittoperator auf $(M, \leq)$ ist.

2.2 Relationen

Relationen

Definition (Relationen) Eine Relation über $M(1), \cdots, M(n)$ ist eine Menge R von n-Tupeln aus dem Wertebereich $M(1) \times \cdots \times M(n)$, d.h. $R \in \mathcal{P}(M(1) \times \cdots \times M(n))$.

Notation (Relationen)

- Def.: $R(s_1, \cdots, s_n) \iff (s_1, \cdots, s_n) \in R$
- Def.: $s_1 R s_2 \iff R(s_1, s_2)$ falls die Relation zweistellig ist

Eigenschaften

Definition (Relationseigenschaften) Eine zweistellige Relation $R \subseteq M \times M$ heißt...

reflexiv	$\forall x \in M : xRx$
symmetrisch	$\forall x, y \in M : xRy \implies yRx$
asymmetrisch	$\forall x, y \in M : xRy \implies \neg yRx$
antisymmetrisch	$\forall x, y \in M : (xRy \wedge yRx) \implies x = y$
transitiv	$\forall x, y, z \in M : (xRy \wedge yRz) \implies xRz$
alternativ	$\forall x, y \in M : xRy \vee yRx$
Äquivalenzrelation	reflexiv $\wedge$ symmetrisch $\wedge$ transitiv
Quasiordnung	reflexiv $\wedge$ transitiv
(partielle) Ordnung	reflexiv $\wedge$ antisymmetrisch $\wedge$ transitiv
totale/lineare Ordnung	reflexiv $\wedge$ antisymmetrisch $\wedge$ transitiv $\wedge$ alternativ

Definition (Wohlfundiert) Eine zweistellige Relation $\prec \subseteq D \times D$ auf einer Menge D heißt wohlfundiert gdw. es keine unendlich absteigende Kette für $\prec$ gibt.

Transitive Hülle

Definition (Transitive Hülle) Die transitive Hülle eine zweistelligen Relation $R \subseteq D \times D$ auf einer Menge D ist die kleinste Relation $R^* \subseteq D \times D$, sodass

- $\forall d_1, d_2 \in D : d_1 R d_2 \implies d_1 R^* d_2$
- $\forall d_1, d_2, d_3 \in D : d_1 R^* d_2 \wedge d_2 R^* d_3 \implies d_1 R^* d_3$

Theorem Sei $\prec \subseteq D \times D$ eine wohlfundierte Relation auf D. Dann gilt:

- $\prec$ ist irreflexiv, d.h. $\forall d \in D : \neg dRd$ und
- $\prec^*$ ist eine wohlfundierte Relation.

Prädikate

Definition (Prädikate) Ein Prädikat über einer Menge M ist eine Funktion $p : M \to \mathbb{B}$.

Beobachtung

- Die *charakteristische Funktion* einer Teilmenge $K \subseteq M$ ist das folgende Prädikat über M:

$$\mathcal{X}(m) \coloneqq \begin{cases} w & \text{falls } m \in K \\ f & \text{falls } m \notin K \end{cases}$$

- Ein Prädikat $p : M \to \mathbb{B}$ induziert zwei Mengen:

$$W_p \coloneqq \{m \in M \mid p(m) = \mathfrak{w}\}$$
$$F_p \coloneqq \{m \in M \mid p(m) = \mathfrak{f}\}$$

2.3 Funktionen

Funktionen

Definition (Funktion) Eine Funktion f von D nach B ist eine Relation über $D \times B$, wobei es für jedes $d \in D$ höchstens ein $b \in B$ mit $(d, b) \in f$ geben darf. Die Menge D heißt *Definitionsbereich* von f, die Menge B heißt *Bildbereich* von f.
Die *Menge aller Funktionen* von D nach B wird mit $D \rightharpoonup B$ bezeichnet.

Notation (Funktionen)

- $f(d) = b \iff (d, b) \in f$
- $f(d) \uparrow \iff$ es gibt kein $b \in B$ sodass $(d, b) \in f$
- $f : D \rightharpoonup B \iff f \in D \rightharpoonup B$
- $f : D \to B \iff f \in D \to B$

Eigenschaften

Definition (Totale Funktion) Eine Funktion f heißt totale Funktion gdw. es für jedes $d \in D$ ein $b \in B$ gibt, sodass $(d, b) \in f$ gilt.
Die *Menge aller totalen Funktionen* von D nach B wird mit $D \to B$ bezeichnet.

Definition (Indempotente Funktionen) Sei M eine Menge.
Eine Funktion $f : \mathcal{P}(M) \to \mathcal{P}(M)$ heißt idempotent gdw. $\forall Q \subseteq M : (f(Q) = f(f(Q))$.

Definition (Monotone Funktionen) Sei M eine Menge.
Eine Funktion $f : \mathcal{P}(M) \to \mathcal{P}(M)$ heißt monoton gdw. $\forall Q, Q' \subseteq M : (Q \subseteq Q' \implies f(Q) \subseteq f(Q'))$ gilt.

Definition (Extensive Funktionen) Sei M eine Menge.
Eine Funktion $f : \mathcal{P}(M) \to \mathcal{P}(M)$ heißt extensiv gdw. $\forall Q \subseteq M : (Q \subseteq f(Q))$ gilt.

Hüllenoperator

Definition (Hüllenoperator) Sei M eine Menge.
Eine Funktion $f : \mathcal{P}(M) \to \mathcal{P}(M)$ heißt Hüllenoperator gdw. folgende Bedingungen gelten:

$$\begin{aligned} &\forall Q \subseteq M : (Q \subseteq f(Q)) && \text{(Extensivität)} \\ \wedge &\forall Q, Q' \subseteq M : (Q \subseteq Q' \implies f(Q) \subseteq f(Q')) && \text{(Monotonie)} \\ \wedge &\forall Q \subseteq M : (f(Q) = f(f(Q))) && \text{(Idempotenz)} \end{aligned}$$

Unendlich absteigende Ketten

Definition (Unendlich absteigende Ketten) Sei $\prec \subseteq D \times D$ eine binäre Relation auf einer Menge D.
Eine unendliche Folge $f : \mathbb{N} \to D$ heißt unendlich absteigende Kette für $\prec$ gdw. $\forall i \in \mathbb{N} : f(i+1) \prec f(i)$.

2.4 Folgen

Endliche Folgen

Definition (Endliche Folgen) Die Menge aller nichtleeren, endlichen Folgen über einer Menge M ist die Menge $M^+ \coloneqq \bigcup_{i \in \mathbb{N}} M^i$.

Definition (Menge aller endlichen Folgen) Die Menge aller endlichen Folgen über einer Menge M ist die Menge $M^* \coloneqq M^+ \cup \{()\}$.

Notation (Endliche Folgen)

- „(“ markiert den Anfang einer Folge.
- „)“ markiert das Ende einer Folge.
- „,“ trennt einzelne Elemente in der Folge voneinander.

Unendliche Folgen

Definition (Unendliche Folgen) Die Menge aller unendlichen Folgen über M ist die Menge $M^\infty \coloneqq \mathbb{N} \to M$.

2.5 Konzepte zur Modellierung

Symbole

Definition (Symbole) Symbole sind beliebige Zeichenketten, welche verwendet werden, um Objekte, Subjekte o.ä. aus der echten Welt zu repräsentieren. Beispielsweise kann *Lautsprecher* definiert werden als Symbol, welches einen Lautsprecher darstellt.

> **Warning:** Symbole müssen immer explizit definiert werden. Implizite Definitionen über den Namen des Symbols sind nicht zulässig.

Wertebereiche

Definition (Wertebereiche) Ein Wertebereich ist eine Menge von Werten, welche im Sinne des Modells als gleichartig angesehen werden. Wird ein Wert aus dem Wertebereich W gefordert, kann jedes Element aus W verwendet werden.

Verwendung Wertebereiche werden zur Strukturierung von Symbolmengen genutzt. Symbole werden hierfür als Werte betrachtet.

Angabemöglichkeiten von Wertebereichen

- Extensional (Aufzählung aller Elemente), z.B.: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
- Intensional (Deklarative Bedingung), z.B.: $\{\alpha \in \mathbb{N} \mid \alpha \text{ ist Prim}\}$
- Induktive Definition einer Familie von Mengen $(M(i))_{i\in\mathbb{N}}$, z.B.:

$$M(1) \coloneqq \{1\}$$
$$M(i) \coloneqq \begin{cases} \{k \in \mathbb{N} \mid k \in M(i-1) \vee k = i\} & \text{falls } i \text{ eine Quadratzahl ist} \\ \{k \in \mathbb{N} \mid k \in M(i-1)\} & \text{falls } i \text{ keine Quadratzahl ist} \end{cases}$$

Notation (Wertebereiche) $\mathbb{N}$ ist die Menge der natürlichen Zahlen *ohne Null* und $\mathbb{N}_0 \coloneqq \mathbb{N} \cup \{0\}$ die Menge der natürlichen Zahlen *mit Null*.

> **Warning:** Nicht jede intensionale Definition einer Menge führt zu einer Menge. Die Menge muss wohldefiniert sein. Beispiel: $M \coloneqq \{x \mid x \notin M\}$
> Eine Menge ist immer wohldefiniert, wenn in der Bedingung nur bereits definierte Begriffe genutzt werden.

Unterspezifizierte Wertebereiche Ein Wertebereich ist unterspezifiziert, wenn die genauen Elemente nicht angegeben werden. Beispielsweise kann ein Wertebereich *LIEDER* definiert werden, welche alle Lieder enthält, die genauen Elemente werden aber nicht gelistet

Konvention Innerhalb einer Modellierung können Konventionen/Vereinfachungen eingeführt werden.
Beispiel: Im folgenden wird anstatt „die durch das Symbol v modellierte Person“ verkürzend „die Person v“ verwendet.

2.6 Aussagen- und Prädikatenlogik

Siehe APL Zusammenfassung (`https://www.dmken.com/redmine/documents/6`).

2.6.1 Präzedenzenordnung

Die Präzedenzenordnung in absteigender Reihenfolge ist ($\wedge$, $\vee$ und $\implies$ sind linksassoziativ):

1. $\neg$
2. $\wedge$
3. $\vee$
4. $\implies$, $\iff$
5. $\forall$, $\exists$

2.7 Formale Sprachen

Siehe AFE Zusammenfassung (`https://www.dmken.com/redmine/documents/7`).

2.7.1 Definitionen

Konkatenation

Definition (Wort-Konkatenation) Die Konkatenation zweier Wörter $u = (a_1, \cdots, a_n)$ und $v = (b_1, \cdots, b_m)$ ist das Wort $u \circ v = (a_1, \cdots, a_n, b_1, \cdots, b_m)$.

Definition (Sprach-Konkatenation) Die Konkatenation zweier Sprachen L_1 und L_2 ist die Sprache $L_1 \circ L_2 = \{v \circ w \,|\, v \in L_1 \wedge w \in L_2\}$.

Definition (i-Fache Sprach-Konkatenation)

$$L^1 = L$$
$$L^{i+1} = \{v \circ w \,|\, v \in L \wedge w \in L^i\}$$

3 Modellierung

3.1 Vorgehen

1. Identifikation relevanter Dinge
2. Modellierung der relevanten Dinge durch Symbole
3. Strukturierung der Symbolmengen durch Wertebereiche
4. Modellierung von Beziehungen durch Relationen
5. Modellierung von Eigenschaften durch Funktionen
6. Modellierung der Anforderungen durch Relationen

3.2 Angemessenheit

Ein (formales) Modell ist angemessen, wenn Beobachtungen, welche am Modell gemacht werden, auch in der Realität gültig sind. Hierzu müssen alle relevanten Aspekte der Realität im Modell wiedergegeben sein.

Die Angemessenheit eines Modells hängt auch davon ab,

- wie eine Beobachtung interpretiert wird und
- welche Fragestellungen von Interesse sind.

Die Argumentation, ob ein Modell angemessen ist, stellt die Verbindung zwischen Modell und Realität her und kann daher nur informell geschehen.

4 Programmiersprachen

4.1 IMP Syntax/Semantik

IMP

- Einfache imperative Programmiersprache
- Sequentielle Sprache mit Verzweigungen und Schleifen

Wertebereiche

Num Ganzen Zahlen

Definition: $\mathit{Num} \coloneqq \mathbb{N}_0 \cup \{-n \mid n \in \mathbb{N}\}$

Intuition: Symbole, welche ganze Zahlen repräsentieren

Bool Wahrheitswerte

Definition: $\mathit{Bool} \coloneqq \{\texttt{true}, \texttt{false}\}$

Intuition: Symbole, welche Wahrheitswerte repräsentieren

Var Programmvariablen

Definition: Unterspezifiziert

Intuition: Programmvariablen

Konvention: m, n bezeichnen Elemente aus *Num*, t bezeichnet Element aus *Bool*, X, Y bezeichnen Elemente aus *Var*

AExp Arithmetischen Ausdrücke

Definition: $\begin{aligned}\langle a \rangle ::= \; & n \\ \mid \; & X \\ \mid \; & (\langle a \rangle \oplus \langle a \rangle) \\ \mid \; & (\langle a \rangle \ominus \langle a \rangle) \\ \mid \; & (\langle a \rangle \odot \langle a \rangle)\end{aligned}$

Intuition: Arithmetische Ausdrücke

BExp Boolesche Ausdrücke

Definition: $\begin{aligned}\langle b \rangle ::= \; & \texttt{true} \\ \mid \; & \texttt{false} \\ \mid \; & (\langle a \rangle \; \texttt{eq} \; \langle a \rangle) \\ \mid \; & (\langle a \rangle \; \texttt{leq} \; \langle a \rangle) \\ \mid \; & \texttt{not} \; \langle b \rangle \\ \mid \; & (\langle b \rangle \; \texttt{and} \; \langle b \rangle) \\ \mid \; & (\langle b \rangle \; \texttt{or} \; \langle b \rangle)\end{aligned}$

Intuition: Boolesche Ausdrücke

Com Kommandos

Definition: $\langle c \rangle ::=$ `skip`
| X `:=` $\langle a \rangle$
| $\langle c \rangle$ `;` $\langle c \rangle$
| `if` $\langle b \rangle$ `then` $\langle c \rangle$ `else` $\langle c \rangle$ `fi`
| `while` $\langle b \rangle$ `do` $\langle c \rangle$ `od`

Intuition: Kommandos von IMP

Konvention: „Programm“ wird synonym für „Kommando“ verwendet

4.2 Syntaktische Korrektheit/Gleichheit

Definition (Korrektheit) Ein Wort c ist ein syntaktisch korrektes Programm, gdw. es in der Grammatik von *Com* ableitbar ist.

Definition (Gleichheit) Zwei Programme c_1 und c_2 sind syntaktisch gleich, gdw. die die gleichen Ableitungen haben (so sind $3 \oplus 5$ und $5 \oplus 3$ nicht syntaktisch gleich).

4.3 Zustände

Definition (Zustand) Ein Zustand eines Programms ist eine Funktion $\sigma : \mathit{Var} \to \mathit{Num}$. Die Menge aller Zustände wird mit Σ bezeichnet.

Intuition: Ein Zustand ordnet jeder Programmvariablen (aus *Var*) einen Wert (aus *Num*) zu.

Definition (Wertsetzungen) Sei $\sigma \in \Sigma$ ein Zustand. Dann ist $\sigma[X \backslash n]$ der Zustand, der der Programmvariablen X den Wert n und jeder anderen Variablen Y den Wert $\sigma(Y)$ zuweist.

4.4 Substitutionen

Definition (Substitution) Eine Substitution ist eine Funktion, welche eine endliche Menge von Metavariablen als Definitionsbereich hat und jedem Element aus dieser Menge genau ein Element aus dem Bildbereich zuordnet.

Definition (Grundsubstitution) Der Bildbereich keine Metavariablen.

Notation (Substitution) $[X_1 \mapsto t_1, \cdots, X_n \mapsto t_n]$ besagt, dass der Definitionsbereich $\{X_1, \cdots, X_n\}$ auf den Bildbereich $\{t_1, \cdots, t_n\}$ abgebildet wird (X_1 auf t_1, X_2 auf t_2 usw.).

Definition (Anwendung) Die Anwendung einer Substitution η auf einen Ausdruck α (geschrieben $\alpha\eta$) ergibt einen Ausdruck β, indem jedes (freies) Auftreten von X durch $\eta(X)$ ersetzt.

Beispiel

$$((2 \oplus 1) \odot (Y \ominus X))[X \mapsto (1 \oplus Z)] = ((2 \oplus 1) \odot (Y \ominus (1 \oplus Z)))$$

4.5 Auswertungssemantik

4.5.1 Urteil

Definitionen

Definition (Urteil) Ein Urteil ist ein Schema für Ausdrücke, welcher Metavariablen als atomare Ausdrücke enthalten kann. Ein Urteil formalisiert einen gegebenen intuitiven Sachverhalt.

Definition (Urteil-Instanz) Ein Ausdruck ξ ist eine Instanz des Urteils ζ gdw. ξ und ζ gleich sind oder ζ durch ersetzen der Metavariablen zu ξ umgeformt werden kann.

Definition (Urteil-Grundinstanz) Eine Instanz ξ des Urteils ζ ist eine Grundinstanz, wenn ξ keine Metavariablen enthält.

Urteile

Definition (Urteil für *AExp*) Das Urteil $\langle a, \sigma \rangle \Downarrow n$ besagt, dass

- ein arithmetischer Ausdruck $a \in \mathit{AExp}$
- in einem Zustand $\sigma \in \Sigma$
- zu einem Wert $n \in \mathit{Num}$ auswertet.

Definition (Urteil für *BExp*) Das Urteil $\langle b, \sigma \rangle \Downarrow t$ besagt, dass

- ein boolescher Ausdruck $b \in \mathit{BExp}$
- in einem Zustand $\sigma \in \Sigma$
- zu einem Wert $t \in \mathit{Bool}$ auswertet.

Definition (Urteil für *Com*) Das Urteil $\langle c, \sigma \rangle \to \sigma'$ besagt, dass

- ein Kommando $c \in \mathit{Com}$
- in einem Zustand $\sigma \in \Sigma$
- zu einem Zustand $\sigma' \in \Sigma$ auswertet.

4.5.2 Kalkül

Notation (Kalkülregeln)

$$\text{r-name}\ \frac{\zeta_1, \cdots, \zeta_n}{\zeta}\ \Phi_1, \cdots, \Phi_m$$

r-name Name der Regel

$\zeta_1, \cdots, \zeta_n$ Prämissen; endliche Liste von Instanzen von Urteilen, welche leer sein kann

ζ Konklusion; instantiiertes Urteil

$\Phi_1, \cdots, \Phi_m$ Seitenbedingungen; endliche Liste von Bedingungen, welche leer sein kann

Instanziierung

Definition (Kalkülregel-Instanz) Eine Regel

$$\text{r-name}\ \frac{\xi_1, \cdots, \xi_n}{\xi}$$

ist die Instanz (bzw. Grundinstanz) einer Kalkülregel

$$\text{r-name}\ \frac{\zeta_1, \cdots, \zeta_n}{\zeta}\ \Phi_1, \cdots, \Phi_n$$

wenn es eine Substitution (bzw. Grundsubstitution) η gibt, sodass $\xi = \zeta\eta$ und $\xi_1 = \zeta_1\eta, \cdots, \xi_n = \zeta_n\eta$ gilt und die Instanzen $\Phi_1\eta, \cdots, \Phi_n\eta$ erfüllt sind.

4.5.3 Kalkülregeln

Kalkül für *AExp*: $\mathcal{A}$

Dieser Kalkül in *AExp* wird zur Herleitung von Urteilen der Form $\langle a, \sigma\rangle \Downarrow n$ genutzt und $\mathcal{A}$ genannt.

$$\text{r}\oplus\ \frac{\langle a_1, \sigma\rangle \Downarrow n_1 \quad \langle a_2, \sigma\rangle \Downarrow n_2}{\langle (a_1 \oplus a_2), \sigma\rangle \Downarrow n}\ n = n_1 + n_2$$

$$\text{r}\ominus\ \frac{\langle a_1, \sigma\rangle \Downarrow n_1 \quad \langle a_1, \sigma\rangle \Downarrow n_2}{\langle (a_1 \ominus a_2), \sigma\rangle \Downarrow n}\ n = n_1 - n_2$$

$$\text{r}\odot\ \frac{\langle a_1, \sigma\rangle \Downarrow n_1 \quad \langle a_2, \sigma\rangle \Downarrow n_2}{\langle (a_1 \cdot a_2), \sigma\rangle \Downarrow n}\ n = n_1 \cdot n_2$$

$$\text{rVar}\ \frac{}{\langle X, \sigma\rangle \Downarrow n}\ n = \sigma(X)$$

$$\text{rNum}\ \frac{}{\langle n, \sigma\rangle \Downarrow n}$$

Kalkül für *BExp*: $\mathcal{B}$

Dieser Kalkül in *BExp* wird zur Herleitung von Urteilen der Form $\langle b, \sigma\rangle \Downarrow t$ genutzt und $\mathcal{B}$ genannt.

$$\text{rfalse}\ \frac{}{\langle \texttt{false}, \sigma\rangle \Downarrow \texttt{false}}$$

$$\text{rtrue}\ \frac{}{\langle \texttt{true}, \sigma\rangle \Downarrow \texttt{true}}$$

$$\text{reqt}\ \frac{\langle a_1,\sigma\rangle \Downarrow n_1 \quad \langle a_2,\sigma\rangle \Downarrow n_2}{\langle (a_1 \texttt{eq} a_2),\sigma\rangle \Downarrow \texttt{true}}\ n_1 = n_2 \qquad \text{randt}\ \frac{\langle b_1,\sigma\rangle \Downarrow \texttt{true} \quad \langle b_2,\sigma\rangle \Downarrow \texttt{true}}{\langle (b_1 \texttt{and} b_2),\sigma\rangle \Downarrow \texttt{true}}$$

$$\text{reqf}\ \frac{\langle a_1,\sigma\rangle \Downarrow n_1 \quad \langle a_2,\sigma\rangle \Downarrow n_2}{\langle (a_1 \texttt{eq} a_2),\sigma\rangle \Downarrow \texttt{false}}\ n_1 \neq n_2 \qquad \text{randf1}\ \frac{\langle b_1,\sigma\rangle \Downarrow \texttt{false}}{\langle (b_1 \texttt{and} b_2),\sigma\rangle \Downarrow \texttt{false}}$$

$$\text{rleqt}\ \frac{\langle a_1,\sigma\rangle \Downarrow n_1 \quad \langle a_2,\sigma\rangle \Downarrow n_2}{\langle (a_1 \texttt{leq} a_2),\sigma\rangle \Downarrow \texttt{true}}\ n_1 \leq n_2 \qquad \text{randf2}\ \frac{\langle b_2,\sigma\rangle \Downarrow \texttt{false}}{\langle (b_1 \texttt{and} b_2),\sigma\rangle \Downarrow \texttt{false}}$$

$$\text{rleqf}\ \frac{\langle a_1,\sigma\rangle \Downarrow n_1 \quad \langle a_2,\sigma\rangle \Downarrow n_2}{\langle (a_1 \texttt{leq} a_2),\sigma\rangle \Downarrow \texttt{false}}\ n_1 > n_2 \qquad \text{rort1}\ \frac{\langle b_1,\sigma\rangle \Downarrow \texttt{true}}{\langle (b_1 \texttt{or} b_2),\sigma\rangle \Downarrow \texttt{true}}$$

$$\text{rnott}\ \frac{\langle b,\sigma\rangle \Downarrow \texttt{false}}{\langle \texttt{not} b,\sigma\rangle \Downarrow \texttt{true}} \qquad \text{rort2}\ \frac{\langle b_2,\sigma\rangle \Downarrow \texttt{true}}{\langle (b_1 \texttt{or} b_2),\sigma\rangle \Downarrow \texttt{true}}$$

$$\text{rnotf}\ \frac{\langle b,\sigma\rangle \Downarrow \texttt{true}}{\langle \texttt{not} b,\sigma\rangle \Downarrow \texttt{false}} \qquad \text{rorf}\ \frac{\langle b_1,\sigma\rangle \Downarrow \texttt{false} \quad \langle b_2,\sigma\rangle \Downarrow \texttt{false}}{\langle (b_1 \texttt{or} b_2),\sigma\rangle \Downarrow \texttt{false}}$$

Kalkül für *Com*: $\mathcal{C}$

Dieser Kalkül in *Com* wird zur Herleitung von Urteilen der Form $\langle c,\sigma\rangle \to \sigma'$ genutzt und $\mathcal{C}$ genannt.

$$\text{rsk}\ \frac{}{\langle \texttt{skip},\sigma\rangle \to \sigma} \qquad \text{rift}\ \frac{\langle b,\sigma\rangle \Downarrow \texttt{true} \quad \langle c_1,\sigma\rangle \to \sigma'}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{fi},\sigma\rangle \to \sigma'}$$

$$\text{r:=}\ \frac{\langle a,\sigma\rangle \Downarrow n}{\langle X := a,\sigma\rangle \to \sigma'}\ \sigma' = \sigma[X\backslash n] \qquad \text{riff}\ \frac{\langle b,\sigma\rangle \Downarrow \texttt{false} \quad \langle c_2,\sigma\rangle \to \sigma'}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{fi},\sigma\rangle \to \sigma'}$$

$$\text{r;}\ \frac{\langle c_1,\sigma\rangle \to \sigma'' \quad \langle c_2,\sigma''\rangle \to \sigma'}{\langle c_1; c_2,\sigma\rangle \to \sigma'} \qquad \text{rwhf}\ \frac{\langle b,\sigma\rangle \Downarrow \texttt{false}}{\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od},\sigma\rangle \to \sigma}$$

$$\text{rwht}\ \frac{\langle b,\sigma\rangle \Downarrow \texttt{true} \quad \langle c,\sigma\rangle \to \sigma'' \quad \langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od},\sigma''\rangle \to \sigma'}{\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od},\sigma\rangle \to \sigma'}$$

4.5.4 Herleitbarkeit

Definition (Herleitbarkeit) Eine Instanz ξ eines Urteils ist in einem Kalkül herleitbar gdw. eine der folgenden Bedingungen erfüllt ist:

- Es existiert eine Kalkülregel der Form

$$\text{r-name}\ \frac{}{\zeta}\ \Phi_1, \cdots, \Phi_n$$

  und eine Substitution η, sodass $\zeta\eta = \xi$ und $\Phi_1\eta, \cdots, \Phi_n\eta$ erfüllt sind.

- Es existiert eine Kalkülregel der Form

$$\text{r-name}\ \frac{\zeta_1, \cdots, \zeta_n}{\zeta}\ \Phi_1, \cdots, \Phi_n$$

  und eine Substitution η, sodass $\zeta\eta = \xi$ und $\Phi_1\eta, \cdots, \Phi_n\eta$ erfüllt sind und die Instanzen $\zeta_1\eta, \cdots, \zeta_n\eta$ der Prämisse herleitbar sind.

4.5.5 Semantische Äquivalenz

Definition (Äquivalenz von booleschen Ausdrücken) Zwei boolesche Ausdrücke $b_1, b_2 \in \mathit{BExp}$ (die Metavariablen enthalten dürfen) sind zueinander semantisch äquivalent gdw. für alle Grundsubstitutionen η, deren Definitionsbereich alle Metavariablen von b_1 und b_2 enthält, für alle Zustände $\sigma \in \Sigma$ und für alle $t \in \mathit{Bool}$ gilt:

$$\langle b_1\eta, \sigma\rangle \Downarrow t \text{ herleitbar} \iff \langle b_2\eta, \sigma\rangle \Downarrow t \text{ herleitbar}$$

Definition (Äquivalenz von arithmetischen Ausdrücken) Zwei arithmetische Ausdrücke $a_1, a_2 \in \mathit{AExp}$ (die Metavariablen enthalten dürfen) sind zueinander semantisch äquivalent gdw. für alle Grundsubstitutionen η, deren Definitionsbereich alle Metavariablen von b_1 und b_2 enthält, und für alle Zustände $\sigma \in \Sigma$ und für alle $n \in \mathit{Num}$ gilt:

$$\langle a_1\eta, \sigma\rangle \Downarrow n \text{ herleitbar} \iff \langle a_2\eta, \sigma\rangle \Downarrow n \text{ herleitbar}$$

Definition (Äquivalenz von Kommandos) Zwei Kommandos $c_1, c_2 \in \mathit{Com}$ (die Metavariablen enthalten dürfen) sind zueinander semantisch äquivalent gdw. für alle Grundsubstitutionen η, deren Definitionsbereich alle Metavariablen von b_1 und b_2 enthält, für alle Zustände $\sigma \in \Sigma$ und für alle $\sigma' \in \Sigma$ gilt:

$$\langle c_1\eta, \sigma\rangle \to \sigma' \text{ herleitbar} \iff \langle c_2\eta, \sigma\rangle \to \sigma' \text{ herleitbar}$$

Notation (Äquivalenz von Programmen) Seien $c_1, c_2 \in \mathit{Com}$. Dann sagt $c_1 \sim c_2$ aus, dass c_1 und c_2 zueinander äquivalent sind.

4.5.6 Termbeschreibungen

Regeln

Definition (Regelterm) Ein Regelterm ist ein Ausdruck der Form r-name$(\xi, (\xi_1, \cdots, \xi_n))$, wobei

r-name der Name der Regel,

ξ eine Grundinstanz des Konklusionsurteils ist und

$(\xi_1, \cdots, \xi_n)$ eine endliche Liste von Grundinstanzen von Prämissenurteilen ist, die auch leer sein kann.

Somit sind Regelterme eine andere Schreibweise für Instanzen von Kalkülregeln.

Definition (Regeltermmenge) Die durch eine Kalkülregel

$$\text{r-name } \frac{\zeta_1, \cdots, \zeta_n}{\zeta} \; \Phi_1, \cdots, \Phi_m$$

repräsentierte Mengen an Regeltermen ist, für alle Grundsubstitutionen η, definiert als

$$\begin{aligned}\text{RTerme(r-name)} \coloneqq \{&\text{r-name}(\zeta\eta, (\zeta_1\eta, \cdots, \zeta_n\eta)) \\ &\mid \zeta_1\eta, \cdots, \zeta_n\eta \text{ enthalten keine Metavariablen und } \Phi_1\eta, \cdots, \Phi_m\eta \text{ sind erfüllt}\}\end{aligned}$$

Herleitungen

Definition (Herleitungsterm) Sei ξ eine Instanz eines Urteils ζ. Die Herleitungen von ξ in einem Kalkül $\mathcal{K}$ (kurz: K-Herleitung von ξ) sind folgendermaßen definiert:
Eine $\mathcal{K}$-Herleitung von ξ ist ein Term der Form r-name$(\xi, (\mathcal{H}_1, \cdots, \mathcal{H}_n))$, wobei es in $\mathcal{K}$ eine Regel der Form

$$\text{r-name } \frac{\zeta_1, \cdots, \zeta_n}{\zeta} \ \Phi_1, \cdots, \Phi_m$$

und eine Substitution η gibt, sodass

- $\xi = \zeta\eta$ und
- $\Phi_1\eta, \cdots, \Phi_m\eta$ erfüllt sind und
- $(\mathcal{H}_1, \cdots, \mathcal{H}_n)$ eine (möglicherweise leere) Liste von Herleitungen ist, sodass, für jedes $i \in \{1, \cdots, n\}$, $\mathcal{H}_i$ eine Herleitung von $\zeta_i\eta$ ist.

Definition (Herleitungsterm 2) Seien $\xi, \xi_1, \cdots, \xi_k$ Instanzen von Urteilen $\zeta, \zeta_1, \cdots, \zeta_k$. Eine Herleitung von ξ aus $\xi_1, \cdots, \xi_k$ in einem Kalkül $\mathcal{K}$ (kurz: $\mathcal{K}$-Herleitung von ξ aus $\xi_1, \cdots, \xi_k$) ist entweder

1. der Term ξ, wobei $\xi \in \{\xi_1, \cdots, \xi_k\}$ gilt, oder
2. ein Term der Form r-name$(\xi, (\mathcal{H}_1, \cdots, \mathcal{H}_n))$, wobei
   - es in $\mathcal{K}$ eine Regel folgender Form gibt:

     $$\text{r-name } \frac{\zeta_1, \cdots, \zeta_n}{\zeta} \ \Phi_1, \cdots, \Phi_m$$

   - und es eine Substitution η gibt, sodass
     - $\xi = \zeta\eta$ und
     - $\Phi_1\eta, \cdots, \Phi_m\eta$ erfüllt sind und
     - $(\mathcal{H}_1, \cdots, \mathcal{H}_n)$ eine (möglicherweise leere) Liste von Herleitungen ist, sodass, für jedes $i \in \{1, \cdots, n\}$, $\mathcal{H}_i$ ein Herleitung von $\zeta_i\eta$ aus $\xi_1, \cdots, \xi_k$ ist.

Notation (Herleitungen)

- Def.: $\mathcal{H} \Vdash_{\mathcal{K}} \xi \iff \mathcal{H}$ ist eine $\mathcal{K}$-Herleitung von ξ
- Def.: $\Vdash_{\mathcal{K}} \xi \iff \xi$ hat eine $\mathcal{K}$-Herleitung

Ergibt sich der Kalkül aus dem Kontext, so kann das tiefgestellte $\mathcal{K}$ weg gelassen werden.

Notation (Menge aller $\mathcal{K}$-Herleitungen von ξ) Die Menge aller $\mathcal{K}$-Herleitungen wird mit $DER_{\mathcal{K}}(\xi)$ bezeichnet.
Ergibt sich der Kalkül aus dem Kontext, so kann das tiefgestellte $\mathcal{K}$ weg gelassen werden.

Notation (Menge aller $\mathcal{K}$-Herleitungen) Die Menge aller $\mathcal{K}$-Herleitungen wird mit $DER_{\mathcal{K}}$ bezeichnet.
Ergibt sich der Kalkül aus dem Kontext, so kann das tiefgestellte $\mathcal{K}$ weg gelassen werden.

4.6 Alternative (operationelle) Semantik

4.6.1 Urteil

Definition (1. Urteil für *AExp*) Das Urteil $\langle a, \sigma \rangle \to_1 a'$ besagt, dass

- ein arithmetischer Ausdruck $a \in \mathit{AExp}$
- in einem Zustand $\sigma \in \Sigma$
- in einem primitiven Berechnungsschritt
- zu einem Ausdruck $a' \in \mathit{AExp}$ reduziert wird.

Definition (Auswertungsurteil für *AExp*) Das Urteil $\langle a, \sigma \rangle \Rightarrow n$ besagt, dass

- ein arithmetischer Ausdruck $a \in \mathit{AExp}$
- in einem Zustand $\sigma \in \Sigma$
- zu einem Wert $n \in \mathit{Num}$ auswertet.

Definition (Urteil für *Com*) Das Urteil $\langle c, \sigma \rangle \to_1 \langle c', \sigma' \rangle$ besagt, dass

- ein Kommando $c \in \mathit{Com}$
- in einem Zustand $\sigma \in \Sigma$
- in einem primitiven Berechnungsschritt
- zu einem Paar $\langle c', \sigma' \rangle$ reduziert wird, wobei
  - $\sigma' \in \Sigma$
  - $c' \in \mathit{Com} \cup \{\epsilon\}$
  - und die Terminierung eines Programms durch ϵ modelliert wird.

Definition (Auswertungsurteil für *Com*) Das Urteil $\langle c, \sigma \rangle \Rightarrow \sigma'$ besagt, dass

- ein Kommando $c \in \mathit{Com}$
- in einem Zustand $\sigma \in \Sigma$
- zu einem Zustand $\sigma' \in \Sigma$ auswertet.

4.6.2 Kalkülregeln

Kalkül für *AExp*

Die folgenden Regeln formulieren einen kleinschrittigen Kalkül, welcher Ausdrücke in *AExp* in primitiven Berechnungsschritten auswertet.

Basisregeln

$$\text{arNum}\ \frac{}{\langle n, \sigma\rangle \to_1 n}$$

$$\text{arVar}\ \frac{}{\langle X, \sigma\rangle \to_1 n}\ n = \sigma(X)$$

Regeln für $\oplus$

$$\text{ar}\oplus 1\ \frac{\langle a_1, \sigma\rangle \to_1 a_1'}{\langle (a_1 \oplus a_2), \sigma\rangle \to_1 (a_1' \oplus a_2)}\ a_1 \notin \mathit{Num}$$

$$\text{ar}\oplus 2\ \frac{\langle a_2, \sigma\rangle \to_1 a_2'}{\langle (n_1 \oplus a_2), \sigma\rangle \to_1 (n_1 \oplus a_2')}\ n_1 \in \mathit{Num} \wedge a_2 \notin \mathit{Num}$$

$$\text{ar}\oplus 3\ \frac{}{\langle (n_1 \oplus n_2), \sigma\rangle \to_1 n}\ n_1, n_2 \in \mathit{Num} \wedge n = n_1 + n_2$$

Regeln für $\ominus$

$$\text{ar}\ominus 1\ \frac{\langle a_1, \sigma\rangle \to_1 a_1'}{\langle (a_1 \ominus a_2), \sigma\rangle \to_1 (a_1' \ominus a_2)}\ a_1 \notin \mathit{Num}$$

$$\text{ar}\ominus 2\ \frac{\langle a_2, \sigma\rangle \to_1 a_2'}{\langle (n_1 \ominus a_2), \sigma\rangle \to_1 (n_1 \ominus a_2')}\ n_1 \in \mathit{Num} \wedge a_2 \notin \mathit{Num}$$

$$\text{ar}\ominus 3\ \frac{}{\langle (n_1 \ominus n_2), \sigma\rangle \to_1 n}\ n_1, n_2 \in \mathit{Num} \wedge n = n_1 - n_2$$

Regeln für $\odot$

$$\text{ar}\odot 1\ \frac{\langle a_1, \sigma\rangle \to_1 a_1'}{\langle (a_1 \odot a_2), \sigma\rangle \to_1 (a_1' \odot a_2)}\ a_1 \notin \mathit{Num}$$

$$\text{ar}\odot 2\ \frac{\langle a_2, \sigma\rangle \to_1 a_2'}{\langle (n_1 \odot a_2), \sigma\rangle \to_1 (n_1 \odot a_2')}\ n_1 \in \mathit{Num} \wedge a_2 \notin \mathit{Num}$$

$$\text{ar}\odot 3\ \frac{}{\langle (n_1 \odot n_2), \sigma\rangle \to_1 n}\ n_1, n_2 \in \mathit{Num} \wedge n = n_1 \cdot n_2$$

Regeln für Auswertungen

$$\text{ar} \Rightarrow \text{a1}\ \frac{\langle a, \sigma\rangle \to_1 n}{\langle a, \sigma\rangle \Rightarrow n}\ n \in \mathit{Num}$$

$$\text{ar} \Rightarrow \text{a2}\ \frac{\langle a, \sigma\rangle \to_1 a' \quad \langle a', \sigma\rangle \Rightarrow n}{\langle a, \sigma\rangle \Rightarrow n}\ a' \notin \mathit{Num}$$

Kalkül für *BExp*

Kalkül für *Com*

4.7 Beweistechniken

4.7.1 Äquivalenz zweier Programme

Fallunterscheidung

Theorem

$$\texttt{while}\ b\ \texttt{do}\ c\ \texttt{od} \sim \texttt{if}\ b\ \texttt{then}\ c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi}$$

Beweis

- Es ist die Definition von semantischer Äquivalenz in *Com* zu zeigen.
- Seien $\sigma, \sigma' \in \Sigma$ beliebige Zustände und η eine beliebige Grundsubstitution, deren Definitionsbereich b und c enthält.
- Somit sind folgende Teilaussagen zu Beweisen:
  1. Wenn $\langle(\texttt{while}\ b\ \texttt{do}\ c\ \texttt{od})\eta, \sigma\rangle \to \sigma'$ herleitbar ist, dann ist auch $\langle(\texttt{if}\ b\ \texttt{then}\ c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi})\eta, \sigma\rangle \to \sigma'$ herleitbar.
  2. Wenn $\langle(\texttt{if}\ b\ \texttt{then}\ c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi})\eta, \sigma\rangle \to \sigma'$ herleitbar ist, dann ist auch $\langle(\texttt{while}\ b\ \texttt{do}\ c\ \texttt{od})\eta, \sigma\rangle \to \sigma'$ herleitbar.
- $\to$ Fallunterscheidung

Beweis von Aussage 1

- Angenommen, $\langle(\texttt{while}\ b\ \texttt{do}\ c\ \texttt{od})\eta, \sigma\rangle \to \sigma'$ sei herleitbar.
- Es gibt zwei Möglichkeiten für die letzte Regel in der Herleitung:
  1. „rwhf“ ist die letzte Regel
  2. „rwht“ ist die letzte Regel
- $\to$ Fallunterscheidung

Beweis von Fall 1.1

- Da „rwhf“ die letzte Regel in der Herleitung ist, muss die Herleitung die folgende Form haben:

$$\text{rwhf}\ \frac{\begin{array}{c}\mathcal{H}_1\\ \vdots\\ \langle b\eta, \sigma\rangle \Downarrow \texttt{false}\end{array}}{\langle(\texttt{while}\ b\ \texttt{do}\ c\ \texttt{od})\eta, \sigma\rangle \to \sigma}$$

- Es gibt somit eine Herleitung $\mathcal{H}_1$ von $\langle b\eta, \sigma\rangle \Downarrow \texttt{false}$ und es gilt $\sigma = \sigma'$.

- Mit Hilfe der Herleitung $\mathcal{H}_1$ kann folgende Herleitung von $\langle(\texttt{if}\ b\ \texttt{then}\ c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi})\eta,\sigma\rangle \to \sigma'$ konstruiert werden:

$$\text{riff}\ \frac{\begin{array}{c}\mathcal{H}_1\\ \vdots\\ \langle b\eta,\sigma\rangle \Downarrow \texttt{false}\end{array}\quad \text{rsk}\ \frac{}{\langle(\texttt{skip})\eta,\sigma\rangle \to \sigma}}{\langle(\texttt{if}\ b\ \texttt{then}\ c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi})\eta,\sigma\rangle \to \sigma}$$

Beweis von Fall 1.2

- Da „rwht“ die letzte Regel in der Herleitung ist, muss die Herleitung die folgende Form haben:

$$\text{rwht}\ \frac{\begin{array}{c}\mathcal{H}_1\\ \vdots\\ \langle b\eta,\sigma\rangle \Downarrow \texttt{true}\end{array}\quad \begin{array}{c}\mathcal{H}_2\\ \vdots\\ \langle c\eta,\sigma\rangle \to \sigma''\end{array}\quad \begin{array}{c}\mathcal{H}_3\\ \vdots\\ \langle(\texttt{while}\ b\ \texttt{do}\ c\ \texttt{od})\eta,\sigma''\rangle \to \sigma'\end{array}}{\langle(\texttt{while}\ b\ \texttt{do}\ c\ \texttt{od})\eta,\sigma\rangle \to \sigma'}$$

- Mit Hilfe der Herleitungen $\mathcal{H}_1$, $\mathcal{H}_2$ und $\mathcal{H}_3$ kann folgende Herleitung von $\langle(\texttt{if}\ b\ \texttt{then}\ c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi})\eta,\sigma\rangle \to \sigma'$ konstruiert werden:

$$\text{rift}\ \frac{\begin{array}{c}\mathcal{H}_1\\ \vdots\\ \langle b\eta,\sigma\rangle \Downarrow \texttt{true}\end{array}\quad \text{r;}\ \frac{\begin{array}{c}\mathcal{H}_2\\ \vdots\\ \langle c\eta,\sigma\rangle \to \sigma''\end{array}\quad \begin{array}{c}\mathcal{H}_3\\ \vdots\\ \langle(\texttt{while}\ b\ \texttt{do}\ c\ \texttt{od})\eta,\sigma''\rangle \to \sigma'\end{array}}{\langle(c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi})\eta,\sigma\rangle \to \sigma'}}{\langle(\texttt{if}\ b\ \texttt{then}\ c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi})\eta,\sigma\rangle \to \sigma'}$$

Beweis von Aussage 2

- Angenommen, $\langle(\texttt{if}\ b\ \texttt{then}\ c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi})\eta,\sigma\rangle \to \sigma'$ sei herleitbar.
- Es gibt zwei Möglichkeiten für die letzte Regel der Herleitung:
  1. „riff“ ist die letzte Regel
  2. „rift“ ist die letzte Regel
- $\to$ Fallunterscheidung

Beweis von Fall 2.1

- Da „riff“ die letzte Regel der Herleitung ist, muss die Herleitung die folgende Form haben:

$$\text{riff}\ \frac{\begin{array}{c}\mathcal{H}_1\\ \vdots\\ \langle b\eta,\sigma\rangle \Downarrow \texttt{false}\end{array}\quad \text{rsk}\ \frac{}{\langle(\texttt{skip})\eta,\sigma\rangle \to \sigma}}{\langle(\texttt{if}\ b\ \texttt{then}\ c;\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \texttt{else skip fi})\eta,\sigma\rangle \to \sigma}$$

- Somit gilt für „riff“, dass $\sigma' = \sigma$.
- Mit Hilfe der Herleitung $\mathcal{H}_1$ kann folgende Herleitung von $\langle(\texttt{while}\ \ b\ \ \texttt{do}\ \ c\ \ \texttt{od})\eta, \sigma\rangle \to \sigma'$ konstruiert werden:

$$\text{rwhf}\ \frac{\begin{array}{c}\mathcal{H}_1\\ \vdots\\ \langle b\eta, \sigma\rangle \Downarrow \texttt{false}\end{array}}{\langle(\texttt{while}\ \ b\ \ \texttt{do}\ \ c\ \ \texttt{od})\eta, \sigma\rangle \to \sigma}$$

Beweis von Fall 2.2

- Da „rift“ die letzte Regel der Herleitung ist, muss die Herleitung die folgende Form haben:

$$\text{riff}\ \frac{\begin{array}{c}\mathcal{H}_1\\ \vdots\\ \langle b\eta, \sigma\rangle \Downarrow \texttt{true}\end{array}\quad \text{r;}\ \frac{\begin{array}{c}\mathcal{H}_1\\ \vdots\\ \langle c\eta, \sigma\rangle \to \sigma''\end{array}\quad \begin{array}{c}\mathcal{H}_3\\ \vdots\\ \langle(\ \texttt{while}\ \ b\ \ \texttt{do}\ \ c\ \ \texttt{od})\eta, \sigma''\rangle \to \sigma'\end{array}}{\langle(c;\ \ \texttt{while}\ \ b\ \ \texttt{do}\ \ c\ \ \texttt{od else skip fi})\eta, \sigma\rangle \to \sigma'}}{\langle(\texttt{if}\ \ b\ \ \texttt{then}\ \ c;\ \ \texttt{while}\ \ b\ \ \texttt{do}\ \ c\ \ \texttt{od else skip fi})\eta, \sigma\rangle \to \sigma'}$$

- Mit Hilfe der Herleitungen $\mathcal{H}_1$, $\mathcal{H}_2$ und $\mathcal{H}_3$ kann folgende Herleitung von $\langle(\texttt{while}\ \ b\ \ \texttt{do}\ \ c\ \ \texttt{od})\eta, \sigma\rangle \to \sigma'$ konstruiert werden:

$$\text{rwht}\ \frac{\begin{array}{c}\mathcal{H}_1\\ \vdots\\ \langle b\eta, \sigma\rangle \Downarrow \texttt{true}\end{array}\quad \begin{array}{c}\mathcal{H}_2\\ \vdots\\ \langle c\eta, \sigma\rangle \to \sigma''\end{array}\quad \begin{array}{c}\mathcal{H}_3\\ \vdots\\ \langle(\texttt{while}\ \ b\ \ \texttt{do}\ \ c\ \ \texttt{od})\eta, \sigma''\rangle \to \sigma'\end{array}}{\langle(\texttt{while}\ \ b\ \ \texttt{do}\ \ c\ \ \texttt{od})\eta, \sigma\rangle \to \sigma'}$$

□

4.7.2 Nichtterminierung eines Programms

Widerspruchsbeweis

Theorem Es gibt keine Zustände $\sigma, \sigma' \in \Sigma$, sodass $\langle\ \texttt{while true do skip od}, \sigma\rangle \to \sigma'$ herleitbar ist.
Intuition: Das Programm terminiert nie.

Beweis

- Seien $\sigma, \sigma' \in \Sigma$ beliebig.
- Sei $\mathcal{H}$ die minimale Herleitung von $\langle\ \texttt{while true do skip od}, \sigma\rangle \to \sigma'$, d.h. es gibt keine Herleitung mit weniger Regeln als $\mathcal{H}$.

- Da „rwht“ die letzte Regel der Herleitung sein muss, hat die Herleitung die folgende Form:

$$\text{rwht}\ \frac{\text{rtrue}\ \dfrac{}{\langle \texttt{true}, \sigma\rangle \Downarrow \texttt{true}} \quad \text{rsk}\ \dfrac{}{\langle \texttt{skip}, \sigma\rangle \to \sigma} \quad \langle\ \texttt{while true do skip od}, \sigma\rangle \to \sigma' \overset{\mathcal{H}_1}{\vdots}}{\langle\ \texttt{while true do skip od}, \sigma\rangle \to \sigma'}$$

- Somit gilt $\sigma'' = \sigma$.
- Da $\mathcal{H}_2$ wiederum eine Herleitung von $\langle\ \texttt{while true do skip od}, \sigma\rangle \to \sigma'$ darstellt, aber weniger Schritte hat als $\mathcal{H}$, ist $\mathcal{H}$ nicht die minimale Herleitung. $\lightning$

$\Longrightarrow$ Das Programm terminiert nie.

□

4.7.3 Induktionsprinzipien

Prinzip der wohlfundierte Induktion

Theorem Sei $P \subseteq D$ eine einstellige Relation auf einer Menge D und $\prec \subseteq D \times D$ eine wohlfundierte Relation auf D.

$$\begin{aligned}&\text{Wenn gilt: } \forall d \in D : (\forall d' \in D : ((d' \prec d \implies P(d')) \implies P(d)))\\ &\text{dann gilt: } \forall d \in D : P(d)\end{aligned}$$

Intuition: Wenn aus „P gilt für alle d', die kleiner sind als d“ folgt, dass „P für d gilt“, dann gilt P für alle Elemente aus D.

Induktion auf den natürlichen Zahlen

Theorem Sei $P \subseteq \mathbb{N}_0$ eine einstellige Relation über den natürlichen Zahlen.

$$\begin{aligned}&\text{Wenn gilt: } P(0)\\ &\qquad\qquad \wedge \forall n \in \mathbb{N}_0 : (P(n) \implies P(n+1))\\ &\text{dann gilt: } \forall n \in \mathbb{N}_0 : P(n)\end{aligned}$$

Instanziierung der wohlfundierten Induktion Die Relation $\prec \subseteq \mathbb{N}_0 \times \mathbb{N}_0$ sei definiert als $\prec := \{(m, n) \in \mathbb{N}_0 \times \mathbb{N}_0 \mid m + 1 = n\}$.

Somit wird das Beweisprinzip der wohlfundierten Induktion folgendermaßen instantiiert:

$$\begin{aligned}&\text{Wenn gilt: } \forall n \in \mathbb{N}_0 : (\forall n' \in \mathbb{N}_0 : ((n' + 1 = n \implies P(n')) \implies P(n)))\\ &\text{Dann gilt: } \forall n \in \mathbb{N}_0 : P(n)\end{aligned}$$

Da $n'+1 = n$ für $n = 0$ niemals gelten kann und die Implikation somit immer gilt, wird eine Fallunterscheidung vorgenommen:

$$\begin{aligned}\text{Wenn gilt: } & \forall n' \in \mathbb{N}_0 : ((n'+1=0 \implies P(n)) \implies P(0)) \\ & \wedge \forall n \in \mathbb{N}_0 : (n \neq 0 \implies ((\forall n' \in \mathbb{N}_0 : ((n'+1=n \implies P(n')) \implies P(n))))) \\ \text{dann gilt: } & \forall n \in \mathbb{N}_0 : P(n)\end{aligned}$$

Dies kann vereinfacht werden zu:

$$\begin{aligned}\text{Wenn gilt: } & \top \implies P(0) \\ & \wedge \forall n'' \in \mathbb{N}_0 : (n''+1 \neq 0 \implies ((\forall n' \in \mathbb{N}_0 : ((n'+1=n''+1 \implies P(n')) \implies P(n''+1))))) \\ \text{dann gilt: } & \forall n \in \mathbb{N}_0 : P(n)\end{aligned}$$

Da $n''+1 \neq 0$ immer gilt, kann wieder vereinfacht werden und das Prinzip der wohlfundierten Induktion damit endgültig für das Prinzip der Induktion über den natürlichen Zahlen instantiiert werden:

$$\begin{aligned}\text{Wenn gilt: } & P(0) \\ & \wedge \forall n'' \in \mathbb{N}_0 : (P(n'') \implies P(n''+1)) \\ \text{dann gilt: } & \forall n \in \mathbb{N}_0 : P(n)\end{aligned}$$

Prinzip der Strukturelle Induktion

Das Prinzip der strukturellen Induktion instantiiert das Prinzip der wohlfundierten Induktion für Strukturen wie *AExp*, *BExp* oder *Com*. Diese werden in den folgenden Abschnitten vorgestellt.

Das Prinzip der strukturellen Induktion für *Com* wird hier bewusst ausgelassen.

Strukturelle Induktion für *AExp*

Sei $P \subseteq \mathit{AExp}$ eine einstellige Relation über *AExp*.

Theorem

$$\begin{aligned}\text{Wenn gilt: } & \forall n \in \mathit{Num} : P(n) \\ & \wedge \forall X \in \mathit{Var} : P(X) \\ & \wedge \forall a_1, a_2 \in \mathit{AExp} : (P(a_1) \wedge P(a_2) \implies P((a_1 \oplus a_2))) \\ & \wedge \forall a_1, a_2 \in \mathit{AExp} : (P(a_1) \wedge P(a_2) \implies P((a_1 \ominus a_2))) \\ & \wedge \forall a_1, a_2 \in \mathit{AExp} : (P(a_1) \wedge P(a_2) \implies P((a_1 \odot a_2))) \\ \text{dann gilt: } & \forall a \in \mathit{AExp} : P(a)\end{aligned}$$

Instanziierung der wohlfundierten Induktion

Definition (Direkte Teilausdrücke) Die Ausdrücke $a_1, a_2 \in \mathit{AExp}$ sind die direkten Teilausdrücke der Ausdrücke $(a_1 \oplus a_2)$, $(a_1 \ominus a_2)$ und $(a_1 \odot a_2)$. Die Ausdrücke n und X haben keine direkten Teilausdrücke.

Die Relation $\prec \subseteq \mathit{AExp} \times \mathit{AExp}$ sei definiert als

$$\prec := \{(a_1, a_2) \in \mathit{AExp} \times \mathit{AExp} \mid a_1 \text{ ist direkter Teilausdruck von } a_2\}$$

.

Somit wird das Beweisprinzip der wohlfundierten Induktion folgendermaßen instantiiert:

Wenn gilt: $\forall a \in \mathit{AExp} : (\forall a' \in \mathit{AExp} : ((a' \text{ ist direkter Teilausdruck von } a \implies P(a')) \implies P(a)))$
dann gilt: $\forall a \in \mathit{AExp} : P(a)$

Da für alle $n \in \mathit{Num}$ und alle $X \in \mathit{Var}$ die Teilausdrucksbedingung nicht gilt und somit die erste Implikation immer Wahr, wird folgende Fallunterscheidung durchgeführt. Auch kann die Bedingung der direkten Teilausdrücke mit obiger Definition verworfen und durch eine Fallunterscheidung implementiert werden:

Wenn gilt: $\forall n \in \mathit{Num} : P(n)$	(n hat keine direkte Teilausdrücke)
$\wedge \forall X \in \mathit{Var} : P(X)$	(X hat keine direkte Teilausdrücke)
$\wedge \forall a_1, a_2 \in \mathit{AExp} : (P(a_1) \wedge P(a_2) \implies P(a_1 \oplus a_2))$	(1. Fall für direkte Teilausdrücke)
$\wedge \forall a_1, a_2 \in \mathit{AExp} : (P(a_1) \wedge P(a_2) \implies P(a_1 \ominus a_2))$	(2. Fall für direkte Teilausdrücke)
$\wedge \forall a_1, a_2 \in \mathit{AExp} : (P(a_1) \wedge P(a_2) \implies P(a_1 \odot a_2))$	(3. Fall für direkte Teilausdrücke)
dann gilt: $\forall a \in \mathit{AExp} : P(a)$	

Damit wurde die wohlfundierte Induktion für die strukturelle Induktion für *AExp* instantiiert.

Strukturelle Induktion für *BExp*

Sei $P \subseteq \mathit{BExp}$ eine einstellige Relation über *BExp*.

Theorem

Wenn gilt: $P(\texttt{true})$
$\wedge P(\texttt{false})$
$\wedge \forall a_1, a_2 \in \mathit{AExp} : P((a_1 \texttt{ eq } a_2))$
$\wedge \forall a_1, a_2 \in \mathit{AExp} : P((a_1 \texttt{ leq } a_2))$
$\wedge \forall b_1 \in \mathit{BExp} : (P(b_1) \implies P(\texttt{not } b_1))$
$\wedge \forall b_1, b_2 \in \mathit{BExp} : (P(b_1) \wedge P(b_2) \implies P((b_1 \texttt{ and } b_2)))$
$\wedge \forall b_1, b_2 \in \mathit{BExp} : (P(b_1) \wedge P(b_2) \implies P((b_1 \texttt{ or } b_2)))$
dann gilt: $\forall b \in \mathit{BExp} : P(b)$

Instanziierung der wohlfundierten Induktion

Definition (Direkte Teilausdrücke) Die Ausdrücke $b_1, b_2 \in$ *BExp* sind die direkten Teilausdrücke der Ausdrücke `not` b_1, b_1 `and` b_2 und b_1 `or` b_2. Die Ausdrücke $true$, $false$, a_1 `eq` a_2 und a_1 `leq` a_2 ($a_1, a_2 \in$ *AExp*) haben keine direkten Teilausdrücke (in *BExp*).

Die Relation $\prec \subseteq$ *BExp* $\times$ *BExp* sei definiert als

$$\prec := \{(b_1, b_2) \in \textit{BExp} \times \textit{BExp} \mid b_1 \text{ ist direkter Teilausdruck von } b_2\}$$

.

Somit wird das Beweisprinzip der wohlfundierten Induktion folgendermaßen instantiiert:

Wenn gilt: $\forall b \in \textit{BExp} : (\forall b' \in \textit{BExp} : ((b \text{ ist direkter Teilausdruck von } b' \implies P(b')) \implies P(b)))$
dann gilt: $\forall b \in \textit{BExp} : P(b)$

Durch eine Fallunterscheidung an den Elementen der Struktur *BExp* ergibt sich das folgende Prinzip der strukturellen Induktion für *BExp*:

Wenn gilt: $P(\texttt{true})$
$\wedge\, P(\texttt{false})$
$\wedge\, \forall a_1, a_2 \in \textit{AExp} : P((a_1 \texttt{ eq } a_2))$
$\wedge\, \forall a_1, a_2 \in \textit{AExp} : P((a_1 \texttt{ leq } a_2))$
$\wedge\, \forall b_1 \in \textit{BExp} : (P(b_1) \implies P(\texttt{not } b_1))$
$\wedge\, \forall b_1, b_2 \in \textit{BExp} : (P(b_1) \wedge P(b_2) \implies P((b_1 \texttt{ and } b_2)))$
$\wedge\, \forall b_1, b_2 \in \textit{BExp} : (P(b_1) \wedge P(b_2) \implies P((b_1 \texttt{ or } b_2)))$
dann gilt: $\forall b \in \textit{BExp} : P(b)$

Damit wurde die wohlfundierte Induktion für die strukturelle Induktion für *BExp* instantiiert.

Prinzip der Induktion über Herleitung

Das Prinzip der Induktion über Herleitung (Herleitungsinduktion) instantiiert das Prinzip der wohlfundierten Induktion für Kalküle wie $\mathcal{A}$, $\mathcal{B}$ oder $\mathcal{C}$. Diese werden in den folgenden Abschnitten vorgestellt.

Definition (Direkte Teilherleitung) Die direkten Teilherleitungen einer Herleitung r-name$(\xi, (\mathcal{H}_1, \cdots, \mathcal{H}_n))$ sind die Herleitungen $\mathcal{H}_1, \cdots, \mathcal{H}_n$.

Induktion über Herleitung für $\mathcal{C}$

Sei $P \subseteq \mathcal{C}$ eine einstellige Relation über $\mathcal{C}$.

Theorem

Wenn gilt: $\forall \sigma \in \Sigma :$

$$P(\text{rsk}((\langle \texttt{skip}, \sigma\rangle \to \sigma), ()))$$

$$\wedge \forall \sigma \in \Sigma :$$

$$\forall X \in \mathit{Var} : \forall a \in \mathit{AExp} : \forall n \in \mathit{Num} :$$

$$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{A}}(\langle a, \sigma\rangle \Downarrow n) :$$

$$P(\text{r:=}((\langle X := a, \sigma\rangle \to \sigma[X \backslash n]), (\mathcal{H}_1))$$

$$\wedge \forall \sigma, \sigma', \sigma'' \in \Sigma :$$

$$\forall c_1, c_2 \in \mathit{Com} :$$

$$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{C}}(\langle c_1, \sigma\rangle \to \sigma'') : \forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_2, \sigma'' \to \sigma') :$$

$$P(\mathcal{H}_1) \wedge P(\mathcal{H}_2)$$

$$\implies P(\text{r;}((\langle c_1; c_2, \sigma\rangle \to \sigma'), (\mathcal{H}_1, \mathcal{H}_2)))$$

$$\wedge \forall \sigma, \sigma' \in \Sigma :$$

$$\forall b \in \mathit{BExp} : \forall c_1, c_2 \in \mathit{Com} :$$

$$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{true}) :$$

$$\forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_1, \sigma\rangle \to \sigma') :$$

$$P(\mathcal{H}_2)$$

$$\implies P(\text{rift}((\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{fi}, \sigma\rangle \to \sigma'), (\mathcal{H}_1, \mathcal{H}_2)))$$

$$\wedge \forall \sigma, \sigma' \in \Sigma :$$

$$\forall b \in \mathit{BExp} : \forall c_1, c_2 \in \mathit{Com} :$$

$$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{false}) :$$

$$\forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_2, \sigma\rangle \to \sigma') :$$

$$P(\mathcal{H}_2)$$

$$\implies P(\text{riff}((\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{fi}, \sigma\rangle \to \sigma'), (\mathcal{H}_1, \mathcal{H}_2)))$$

$$\wedge \forall \sigma, \sigma', \sigma'' \in \Sigma :$$

$$\forall b \in \mathit{BExp} : \forall c \in \mathit{Com} :$$

$$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{true}) :$$

$$\forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma'') :$$

$$\forall \mathcal{H}_3 \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}, \sigma''\rangle \to \sigma') :$$

$$P(\mathcal{H}_2) \wedge P(\mathcal{H}_3)$$

$$\implies P(\text{rwht}((\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}, \sigma\rangle \to \sigma'), (\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)))$$

$$\wedge \forall \sigma \in \Sigma :$$

$$\forall b \in \mathit{BExp} : \forall c \in \mathit{Com} :$$

$$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{false}) :$$

$$P(\text{rwhf}((\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}, \sigma\rangle \to \sigma), (\mathcal{H}_1)))$$

dann gilt: $\forall \mathcal{H} \in \mathit{DER}_{\mathcal{C}} : P(\mathcal{H})$

Instanziierung der wohlfundierten Induktion Die Relation $\prec \subseteq \mathit{DER}_{\mathcal{C}} \times \mathit{DER}_{\mathcal{C}}$ sei definiert als

$$\prec := \{(\mathcal{H}_1, \mathcal{H}_{\in}) \in \mathit{DER}_{\mathcal{C}} \times \mathit{DER}_{\mathcal{C}} \mid \mathcal{H}_1 \text{ ist direkte Teilherleitung von } \mathcal{H}_2\}$$

Somit wird das Beweisprinzip der wohlfundierten Induktion folgendermaßen instantiiert:

Wenn gilt: $\forall \mathcal{H} \in \mathit{DER}_{\mathcal{C}} : (\forall \mathcal{H}' \in \mathit{DER}_{\mathcal{C}} : ((\mathcal{H} \text{ ist direkte Teilherleitung von } \mathcal{H}' \implies P(\mathcal{H}')) \implies \mathcal{H}))$
dann gilt: $\forall \mathcal{H} \in \mathit{DER}_{\mathcal{C}} : P(\mathcal{H})$

Durch Fallunterscheidungen für alle Regeln aus $\mathcal{C}$ ergeben sich viele Einzelbedingungen (bspw. hat die Regel

„rsk“ keine direkten Teilherleitungen, weshalb sich die Regel stark vereinfacht):

Wenn gilt: $\forall \sigma \in \Sigma :$ (Zustände)
$P(\text{rsk}((\langle \texttt{skip}, \sigma\rangle \to \sigma), ()))$ (Konklusion)
$\wedge \forall \sigma \in \Sigma :$ (Zustände)
$\forall X \in \mathit{Var} : \forall a \in \mathit{AExp} : \forall n \in \mathit{Num} :$ (Ausdrucksbestandteile)
$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{A}}(\langle a, \sigma\rangle \Downarrow n) :$ (Kalkülfremde Herleitungen)
$P(\text{r:=}((\langle X \coloneqq a, \sigma\rangle \to \sigma[X \backslash n]), (\mathcal{H}_1))$ (Konklusion)
$\wedge \forall \sigma, \sigma', \sigma'' \in \Sigma :$ (Zustände)
$\forall c_1, c_2 \in \mathit{Com} :$ (Ausdrucksbestandteile)
$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{C}}(\langle c_1, \sigma\rangle \to \sigma'') : \forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_2, \sigma'' \to \sigma'\rangle :$ (Herleitungen in Prämisse)
$P(\mathcal{H}_1) \wedge P(\mathcal{H}_2)$ (Prämisse)
$\implies P(\text{r;}((\langle c_1; c_2, \sigma\rangle \to \sigma'), (\mathcal{H}_1, \mathcal{H}_2)))$ (Konklusion)
$\wedge \forall \sigma, \sigma' \in \Sigma :$ (Zustände)
$\forall b \in \mathit{BExp} : \forall c_1, c_2 \in \mathit{Com} :$ (Ausdrucksbestandteile)
$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{true}) :$ (Kalkülfremde Herleitungen)
$\forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_1, \sigma\rangle \to \sigma') :$ (Herleitungen in Prämisse)
$P(\mathcal{H}_2)$ (Prämisse)
$\implies P(\text{rift}((\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ fi}, \sigma\rangle \to \sigma'), (\mathcal{H}_1, \mathcal{H}_2)))$ (Konklusion)
$\wedge \forall \sigma, \sigma' \in \Sigma :$ (Zustände)
$\forall b \in \mathit{BExp} : \forall c_1, c_2 \in \mathit{Com} :$ (Ausdrucksbestandteile)
$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{false}) :$ (Kalkülfremde Herleitungen)
$\forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_2, \sigma\rangle \to \sigma') :$ (Herleitungen in Prämisse)
$P(\mathcal{H}_2)$ (Prämisse)
$\implies P(\text{riff}((\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ fi}, \sigma\rangle \to \sigma'), (\mathcal{H}_1, \mathcal{H}_2)))$ (Konklusion)
$\wedge \forall \sigma, \sigma', \sigma'' \in \Sigma :$ (Zustände)
$\forall b \in \mathit{BExp} : \forall c \in \mathit{Com} :$ (Ausdrucksbestandteile)
$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{true}) :$ (Kalkülfremde Herleitungen)
$\forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma'') :$ (Herleitungen in Prämisse)
$\forall \mathcal{H}_3 \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma''\rangle \to \sigma') :$ (Herleitungen in Prämisse)
$P(\mathcal{H}_2) \wedge P(\mathcal{H}_3)$ (Prämisse)
$\implies P(\text{rwht}((\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma\rangle \to \sigma'), (\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)))$ (Konklusion)
$\wedge \forall \sigma \in \Sigma :$ (Zustände)
$\forall b \in \mathit{BExp} : \forall c \in \mathit{Com} :$ (Ausdrucksbestandteile)
$\forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{false}) :$ (Kalkülfremde Herleitungen)
$P(\text{rwhf}((\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma\rangle \to \sigma), (\mathcal{H}_1)))$ (Konklusion)
dann gilt: $\forall \mathcal{H} \in \mathit{DER}_{\mathcal{C}} : P(\mathcal{H})$

Damit wurde die wohlfundierte Induktion für die Induktion über Herleitungen für $\mathcal{C}$ instantiiert.

Regelinduktion

Das Prinzip der Regelinduktion definiert ein Induktionsprinzip über Regeln eines Kalküls. Es kann beispielsweise für Kalküle wie $\mathcal{A}$, $\mathcal{B}$ oder $\mathcal{C}$ instantiiert werden.

Das Prinzip Regelinduktion für $\mathcal{B}$ und $\mathcal{C}$ wird hier bewusst ausgelassen.

Beweisprinzip Sei $\mathcal{K}$ ein Kalkül zur Herleitung von Instanzen eines Urteils und P eine einstellige Relation auf der Menge $I_{\mathcal{K}}$ aller Instanzen dieses Urteils.

Wenn für jede Kalkülregel

$$\text{r-name}\ \frac{\zeta_1, \cdots, \zeta_n}{\zeta}\ \Phi_1, \cdots, \Phi_m$$

in $\mathcal{K}$ und jeder Substitution η, sodass $\Phi_1\eta, \cdots, \Phi_m\eta$ erfüllt sind, gilt, dass

$$P(\zeta_1\eta) \wedge \cdots \wedge P(\zeta_n\eta) \implies P(\zeta\eta)$$

dann gilt auch

$$\forall \xi \in I_{\mathcal{K}} : P(\xi)$$

Regelinduktion für $\mathcal{A}$

Theorem

$$\begin{aligned}
\text{Wenn gilt: } & \forall n \in \mathit{Num} : P(n) \\
& \wedge \forall X \in \mathit{Var} : P(X) \\
& \wedge \forall a_1, a_2 \in \mathit{AExp} : P(a_1) \wedge P(a_2) \implies P((a_1 \oplus a_2)) \\
& \wedge \forall a_1, a_2 \in \mathit{AExp} : P(a_1) \wedge P(a_2) \implies P((a_1 \ominus a_2)) \\
& \wedge \forall a_1, a_2 \in \mathit{AExp} : P(a_1) \wedge P(a_2) \implies P((a_1 \odot a_2)) \\
\text{dann gilt: } & \forall a \in \mathit{AExp} : P(a)
\end{aligned}$$

Welches wiederum der strukturellen Induktion für *AExp* entspricht.

4.8 Deterministische Auswertung

4.8.1 ...von Ausdrücken in *AExp*

Theorem Für alle $a \in \mathit{AExp}$, $m, m' \in \mathit{Num}$ und $\sigma \in \Sigma$ gilt:
Wenn $\langle a, \sigma\rangle \Downarrow m$ und $\langle a, \sigma\rangle \Downarrow m'$ herleitbar sind, dann gilt $m = m'$.

Beweis Es wird das Prinzip der strukturellen Induktion verwendet mit der einstelligen Relation

$$\begin{aligned}
P(a) \coloneqq \forall a \in \mathit{AExp} : \forall m, m' \in \mathit{Num} : \forall \sigma \in \Sigma : \\
((\langle a, \sigma\rangle \Downarrow m) \text{ herleitbar} \wedge (\langle a, \sigma\rangle \Downarrow m') \text{ herleitbar}) \implies m = m'
\end{aligned}$$

Fall „Num“ Sei $n \in \mathit{Num}$ beliebig.
Die Herleitungen von $\langle n, \sigma\rangle \Downarrow m$ und $\langle n, \sigma\rangle \Downarrow m'$ können nur die folgende Form haben:

$$\text{rNum}\ \frac{}{\langle n, \sigma\rangle \Downarrow n}$$

Womit $m = m'$ gelten muss.

Fall „Var“ Sei $X \in \mathit{Var}$ beliebig.
Die Herleitungen von $\langle X, \sigma\rangle \Downarrow m$ und $\langle n, \sigma\rangle \Downarrow m'$ können nur die folgende Form haben:

$$\text{rVar}\ \frac{}{\langle X, \sigma\rangle \Downarrow n}\ n = \sigma(X)$$

Da $\sigma(\cdot)$ eine Funktion ist, muss somit $m = m'$ gelten.

Fall „$\oplus$“ Seien $a_1, a_2 \in \mathit{AExp}$ beliebig und gelte $P(a_1) \wedge P(a_2)$.
Die Herleitungen von $\langle (a_1 \oplus a_2), \sigma\rangle \Downarrow m$ und $\langle (a_1 \oplus a_2), \sigma\rangle \Downarrow m'$ können nur die folgende Form haben:

$$\text{r}\oplus\ \frac{\begin{array}{cc}\mathcal{H}_1 & \mathcal{H}_2\\ \vdots & \vdots\\ \langle a_1, \sigma\rangle \Downarrow n_1 & \langle a_2, \sigma\rangle \Downarrow n_2\end{array}}{\langle (a_1 \oplus a_2), \sigma\rangle \Downarrow n}\ n = n_1 + n_2$$

Da a_1 und a_2 deterministisch auswerten, muss $m = m'$ gelten.

Fall „$\ominus$“ Seien $a_1, a_2 \in \mathit{AExp}$ beliebig und gelte $P(a_1) \wedge P(a_2)$.
Die Herleitungen von $\langle (a_1 \ominus a_2), \sigma\rangle \Downarrow m$ und $\langle (a_1 \ominus a_2), \sigma\rangle \Downarrow m'$ können nur die folgende Form haben:

$$\text{r}\ominus\ \frac{\begin{array}{cc}\mathcal{H}_1 & \mathcal{H}_2\\ \vdots & \vdots\\ \langle a_1, \sigma\rangle \Downarrow n_1 & \langle a_2, \sigma\rangle \Downarrow n_2\end{array}}{\langle (a_1 \ominus a_2), \sigma\rangle \Downarrow n}\ n = n_1 - n_2$$

Da a_1 und a_2 deterministisch auswerten, muss $m = m'$ gelten.

Fall „$\odot$“ Seien $a_1, a_2 \in \mathit{AExp}$ beliebig und gelte $P(a_1) \wedge P(a_2)$.
Die Herleitungen von $\langle (a_1 \odot a_2), \sigma\rangle \Downarrow m$ und $\langle (a_1 \odot a_2), \sigma\rangle \Downarrow m'$ können nur die folgende Form haben:

$$\text{r}\odot\ \frac{\begin{array}{cc}\mathcal{H}_1 & \mathcal{H}_2\\ \vdots & \vdots\\ \langle a_1, \sigma\rangle \Downarrow n_1 & \langle a_2, \sigma\rangle \Downarrow n_2\end{array}}{\langle (a_1 \odot a_2), \sigma\rangle \Downarrow n}\ n = n_1 \cdot n_2$$

Da a_1 und a_2 deterministisch auswerten, muss $m = m'$ gelten.
Somit gilt $\forall a \in \mathit{AExp} : P(a)$ und damit sind Auswertungen in *AExp* deterministisch.

□

4.8.2 ... von Programmen in *Com*

Theorem Für alle $c \in \mathit{Com}$ und alle Zustände $\sigma, \sigma', \sigma'' \in \Sigma$ gilt:
Wenn $\langle c, \sigma\rangle \to \sigma'$ und $\langle c, \sigma\rangle \to \sigma''$ herleitbar sind, dann gilt $\sigma' = \sigma''$.

Beweis Es wird das Prinzip der Induktion über Herleitung verwendet mit der einstelligen Relation

$$P(\mathcal{H}) \coloneqq \forall c \in \mathit{Com} : \forall \sigma, \sigma', \sigma'' \in \Sigma : (\mathcal{H} \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma') \implies \forall \mathcal{H}' \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma'') : \sigma' = \sigma'')$$

Fall „rsk“ Es ist zu zeigen, dass $\forall \sigma^* \in \Sigma : P(\mathrm{rsk}((\langle \texttt{skip}, \sigma^*\rangle \to \sigma^*), ()))$ gilt.
Seien $\sigma^*, \sigma, \sigma', \sigma'' \in \Sigma$, $c \in \mathit{Com}$ beliebig, sodass $\mathrm{rsk}((\langle \texttt{skip}, \sigma^*\rangle \to \sigma^*), ()) \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma')$ gilt.
Somit muss $c = \texttt{skip}$, $\sigma = \sigma^*$ und $\sigma' = \sigma^*$ gelten. Damit gilt auch $\sigma = \sigma'$.
Es ist noch zu zeigen, dass $\forall \mathcal{H}' \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{skip}, \sigma\rangle \to \sigma'')$.
Sei $\mathcal{H}' \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{skip}, \sigma\rangle \to \sigma'')$ beliebig.
Da es nur eine Regel gibt, welche `skip` enthält, muss $\mathcal{H}' = \mathrm{rsk}((\langle \texttt{skip}, \sigma\rangle \to \sigma), ())$ gelten. Durch $\mathcal{H}' \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{skip}, \sigma\rangle \to \sigma'')$ gilt somit $\sigma = \sigma''$.
Damit folgt aus $\sigma = \sigma'$, dass auch $\sigma' = \sigma''$ gilt.

Fall „r:=“ Es ist zu zeigen, dass $\forall \sigma^* \in \Sigma : \forall X \in \mathit{Var} : \forall a \in \mathit{AExp} : \forall n \in \mathit{Num} : \forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{A}} : (\langle a, \sigma^*\rangle \Downarrow n) : P(\mathrm{r{:=}}((\langle X \coloneqq a, \sigma^*\rangle \to \sigma^*[X \backslash n]), (\mathcal{H}_1)))$ gilt.
Seien $\sigma^*, \sigma, \sigma', \sigma'' \in \Sigma$, $X \in \mathit{Var}$, $a \in \mathit{AExp}$, $n \in \mathit{Num}$ und $\mathcal{H}_1 \in \mathit{DER}_{\mathcal{A}}(\langle a, \sigma^*\rangle \Downarrow n)$ beliebig, sodass $\mathrm{r{:=}}((\langle X \coloneqq a, \sigma^*\rangle \to \sigma^*[X \backslash n]), (\mathcal{H}_1)) \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma')$ gilt.
Somit muss $c = (A \coloneqq a)$, $\sigma = \sigma^*$ und $\sigma' = \sigma^*[X \backslash n]$ gelten.
Es ist noch zu zeigen, dass $\forall \mathcal{H}' \in \mathit{DER}_{\mathcal{C}}(\langle A \coloneqq a, \sigma\rangle \to \sigma'')$ gilt.
Da es nur eine Regel gibt, welche $A \coloneqq a$ enthält, muss $\mathcal{H}' = \mathrm{r{:=}}((\langle A \coloneqq a, \sigma\rangle \to \sigma''), (\mathcal{H}'_1))$ gelten, wobei $\mathcal{H}'_1 \in \mathit{DER}_{\mathcal{A}}(\langle a, \sigma\rangle \Downarrow m)$. Durch $\mathcal{H}' \in \mathit{DER}_{\mathcal{C}}(\langle A \coloneqq a, \sigma\rangle \to \sigma'')$ gilt somit $\sigma'' = \sigma[X \backslash m]$.
Da die Auswertung von Ausdrücken in *AExp* deterministisch ist, gilt $n = m$.
Damit folgt aus $\sigma = \sigma^*$, dass $\sigma'' = \sigma^*[X \backslash n] = \sigma'$, dass auch $\sigma' = \sigma''$ gilt.

Fall „r;“ Es ist zu zeigen, dass $\forall \sigma^*, \sigma^{*'}, \sigma^{*''} \in \Sigma : \forall c_1, c_2 \in \mathit{Com} : \forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{C}}(\langle c_1, \sigma^*\rangle \to \sigma^{*''}) : \forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_2, \sigma^{*''}\rangle \to \sigma^{*'}) : P(\mathcal{H}_1) \wedge P(\mathcal{H}_2) \implies P(\mathrm{r;}((\langle c_1; c_2, \sigma^*\rangle \to \sigma^{*'}), (\mathcal{H}_1, \mathcal{H}_2)))$ gilt.
Seien $\sigma^*, \sigma^{*'}, \sigma^{*''}, \sigma, \sigma', \sigma'' \in \Sigma$, $c_1, c_2, c \in \mathit{Com}$, $\mathcal{H}_1 \in \mathit{DER}_{\mathcal{C}}(\langle c_1, \sigma^*\rangle \to \sigma^{*''})$, $\mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_2, \sigma^{*''}\rangle \to \sigma^{*'})$ beliebig, sodass $\mathrm{r;}((\langle c_1; c_2, \sigma^* \to \sigma^{*'}), (\mathcal{H}_1, \mathcal{H}_2)) \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma')$ und $P(\mathcal{H}_1)$ und $P(\mathcal{H}_2)$ gelten.
Somit muss $c = (c_1; c_2)$, $\sigma = \sigma^*$ und $\sigma' = \sigma^{*'}$ gelten.
Es ist noch zu zeigen, dass $\forall \mathcal{H}' \in \mathit{DER}_{\mathcal{C}}(\langle c_1; c_2, \sigma\rangle \to \sigma')$ gilt.
Da die Komposition zweier Ausdrücke nur in einer Regel vorkommt, muss $\mathcal{H}' = \mathrm{r;}((\langle c_1; c_2, \sigma\rangle \to \sigma'), (\mathcal{H}'_1, \mathcal{H}'_2))$ mit $\mathcal{H}'_1 \in \mathit{DER}_{\mathcal{C}}(\langle c_1, \sigma\rangle \to \sigma''')$ und $\mathcal{H}'_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_2, \sigma'''\rangle \to \sigma'')$ gelten für ein $\sigma''' \in \Sigma$.
Aus $P(\mathcal{H}_1)$, $\mathcal{H}_1 \in \mathit{DER}_{\mathcal{C}}(\langle c_1^*, \sigma^*\rangle \to \sigma^{*''})$, $\mathcal{H}'_1 \in \mathit{DER}_{\mathcal{C}}(\langle c_1^*, \sigma\rangle \to \sigma''')$ und $\sigma^* = \sigma$ folgt, dass $\sigma''' = \sigma^{*''}$.
Aus $P(\mathcal{H}_2)$, $\mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_2^*, \sigma^{*''}\rangle \to \sigma^{*'})$, $\mathcal{H}'_2 \in \mathit{DER}_{\mathcal{C}}(\langle c_2^*, \sigma'''\rangle \to \sigma'')$ und $\sigma^{*''} = \sigma'''$ folgt, dass $\sigma^{*'} = \sigma''$.
Da $\sigma^{*'} = \sigma'$, gilt $\sigma' = \sigma''$.

Fall „rift“ Analog ...

Fall „riff“ Analog ...

Fall „rwht“ Es ist zu zeigen, dass $\forall \sigma^*, \sigma^{*'}, \sigma^{*''} \in \Sigma : \forall b \in \mathit{BExp} : \forall c^* \in \mathit{Com} : \forall \mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{true}) : \forall \mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma^*\rangle \to \sigma^{*''}) : \forall \mathcal{H}_3 \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma^{*''}\rangle \to \sigma^{*'}) : P(\mathcal{H}_2) \wedge P(\mathcal{H}_3) \implies P(\text{rwht})((\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma^*\rangle \to \sigma^{*'}), (\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3))$ gilt.

Seien $\sigma^*, \sigma^{*'}, \sigma^{*''}, \sigma, \sigma', \sigma'' \in \Sigma$, $b \in \mathit{BExp}$, $c^*, c \in \mathit{Com}$, $\mathcal{H}_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{true})$, $\mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma^*\rangle \to \sigma^{*''})$ und $\mathcal{H}_3 \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma^{*''}\rangle \to \sigma^{*'})$ beliebig, sodass $\text{rwht}((\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma^*\rangle \to \sigma^{*'}), (\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)) \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma')$, $P(\mathcal{H}_1)$ und $P(\mathcal{H}_2)$ gelten.

Somit muss $c = (\texttt{while } b \texttt{ do } c \texttt{ od})$, $\sigma = \sigma^*$ und $\sigma' = \sigma^{*'}$ gelten.

Es ist noch zu zeigen, dass $\forall \mathcal{H}' \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{while } b \texttt{ do } c^* \texttt{ od}, \sigma\rangle \to \sigma'')$ gilt.

Da die Auswertung von Ausdrücken in *BExp* deterministisch ist, ist $\langle b, \sigma\rangle \Downarrow \texttt{false}$ nicht herleitbar. Somit muss $\mathcal{H}' = \text{rwht}((\langle \texttt{while } b \texttt{ do } c \texttt{ od}), (\mathcal{H}'_1, \mathcal{H}'_2, \mathcal{H}'_3), \sigma\rangle \to \sigma')$ gelten, wobei $\mathcal{H}'_1 \in \mathit{DER}_{\mathcal{B}}(\langle b, \sigma\rangle \Downarrow \texttt{true})$, $\mathcal{H}'_2 \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma''')$ und $\mathcal{H}'_3 \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma'''\rangle \to \sigma'')$ für ein $\sigma''' \in \Sigma$.

Aus $P(\mathcal{H}_2)$, $\mathcal{H}_2 \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma^*\rangle \to \sigma^{*''})$, $\mathcal{H}'_2 \in \mathit{DER}_{\mathcal{C}}(\langle c, \sigma\rangle \to \sigma''')$ und $\sigma = \sigma^*$ folgt, dass $\sigma''' = \sigma^{*''}$.

Aus $P(\mathcal{H}_3)$, $\mathcal{H}_3 \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma^{*''}\rangle \to \sigma^{*'})$, $\mathcal{H}'_3 \in \mathit{DER}_{\mathcal{C}}(\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma'''\rangle \to \sigma'')$ und $\sigma''' = \sigma^{*''}$ folgt, dass $\sigma'' = \sigma^{*'}$.

Da $\sigma^{*'} = \sigma'$, gilt $\sigma' = \sigma''$.

Fall „rwhf“ Analog …

Somit gilt $\forall \mathcal{H} \in \mathit{DER}_{\mathcal{C}} : P(\mathcal{H})$.

□

4.9 Kalküle als Spezifikationssprache

4.9.1 Induktiv definierte Mengen

Ein Kalkül $\mathcal{K}$ definiert induktiv die Menge $I_{\mathcal{K}} \coloneqq \{\xi \mid \Vdash_{\mathcal{K}} \xi\}$.

In den folgenden Abschnitten werden weitete Möglichkeiten aufgezeigt, wie die Menge $I_{\mathcal{K}}$ charakterisiert werden kann:

- als Menge aller Instanzen eines Urteils, welche in $\mathcal{K}$ herleitbar sind,
- als Schnittmenge aller unter $\mathcal{K}$ abgeschlossenen Mengen und
- als Ergebnis der Anwendung des Hüllenoperators $R^*_{\mathcal{K}}$ auf $\emptyset$.

4.9.2 Operatoren

Definition (Operatoren) Sei $\mathcal{K}$ ein Kalkül zur Herleitung von Instanzen eines Urteils.

Die Operatoren $\hat{R}_{\mathcal{K}}$, $\hat{R}^i_{\mathcal{K}}$, $\overline{R}_{\mathcal{K}}$ und $R^*_{\mathcal{K}}$ auf Mengen von Instanzen dieses Urteils werden wie folgt durch $\mathcal{K}$

spezifiziert:

$$\begin{aligned}
\hat{R}_{\mathcal{K}}(Q) &\coloneqq \{\xi \mid \exists \xi_1, \cdots, \xi_n : \exists \text{r-name} : (\text{r-name}(\xi, (\xi_1, \cdots, \xi_n)) \in \text{RTerme}(\text{r-name}) \wedge \{\xi_1, \cdots, \xi_n\} \subseteq Q)\} \\
\overline{R}_{\mathcal{K}}(Q) &\coloneqq Q \cup \{\xi \mid \exists \xi_1, \cdots, \xi_n : \exists \text{r-name} : (\text{r-name}(\xi, (\xi_1, \cdots, \xi_n)) \in \text{RTerme}(\text{r-name}) \wedge \{\xi_1, \cdots, \xi_n\} \subseteq Q)\} \\
&= Q \cup \hat{R}_{\mathcal{K}}(Q) \\
\hat{R}^0_{\mathcal{K}}(Q) &\coloneqq Q \\
\hat{R}^{i+1}_{\mathcal{K}}(Q) &\coloneqq \hat{R}_{\mathcal{K}}(\hat{R}^i_{\mathcal{K}}(Q)) \\
R^*_{\mathcal{K}}(Q) &\coloneqq \bigcup_{i \in \mathbb{N}_0} \hat{R}^i_{\mathcal{K}}(Q)
\end{aligned}$$

Ergibt sich der Kalkül aus dem Kontext, kann das tiefgestellte $\mathcal{K}$ fallen gelassen werden.

Theorem $\hat{R}_{\mathcal{K}}$ und $\overline{R}_{\mathcal{K}}$ sind monoton.

Theorem $\overline{R}_{\mathcal{K}}$ ist extensiv.

Theorem $R^*_{\mathcal{K}}$ ist ein Hüllenoperator.

Theorem Es gilt $R^*_{\mathcal{K}}(\emptyset) = I_{\mathcal{K}}$.

4.9.3 Abschlusseigenschaften

Definition (Abgeschlossenheit unter Regeln) Eine Kalkülregel

$$\text{r-name}\ \frac{\zeta_1, \cdots, \zeta_n}{\zeta}\ \Phi_1, \cdots, \Phi_m$$

definiert für eine Menge Q die Eigenschaft

$$\forall \eta : (\Phi_1 \eta \wedge \cdots \wedge \Phi_m \eta \wedge \{\zeta_1 \eta, \cdots, \zeta_m \eta\} \subseteq Q) \implies \zeta \eta \in Q$$

Eine Menge Q heißt *abgeschlossen unter r-name* gdw. obige Formel für Q erfüllt ist.

Definition (Abgeschlossenheit unter Kalkül) Sei $\mathcal{K}$ ein Kalkül zur Herleitung von Instanzen eines Urteils.
Eine Menge Q ist *abgeschlossen unter* $\mathcal{K}$ („Q ist $\mathcal{K}$-abgeschlossen“) gdw. Q unter jeder Kalkülregel in $\mathcal{K}$ abgeschlossen ist.

Theorem Sei $\mathcal{K}$ ein Kalkül zur Herleitung von Instanzen eines Urteils.
Eine Menge Q von Instanzen dieses Urteils ist unter $\mathcal{K}$ abgeschlossen gdw. $\hat{R}_{\mathcal{K}}(Q) \subseteq Q$.

Definition (Abschlusseigenschaft) Sei M eine Menge. Eine einstellige Relation P auf $\mathcal{P}(M)$ heißt Abschlusseigenschaft gdw. $\forall Q \subseteq M : \exists Q' \subseteq M : (Q \subseteq Q' \wedge P(Q'))$

Theorem Die durch ein Kalkül $\mathcal{K}$ für eine Menge Q spezifizierte Eigenschaft „Q ist abgeschlossen unter $\mathcal{K}$“ ist eine Abschlusseigenschaft.

Theorem Sei $\mathcal{K}$ ein Kalkül zur Herleitung von Instanzen eines Urteils.

- Die Menge $I_\mathcal{K}$ ist $\mathcal{K}$-abgeschlossen.
- Wenn Q unter $\mathcal{K}$ abgeschlossen ist, dann gilt $I_\mathcal{K} \subseteq Q$.
- $\bigcap\{Q \mid Q \text{ ist abgeschlossen unter } \mathcal{K}\} = I_\mathcal{K}$

5 Formale Modellierung in der Softwareentwicklung

Siehe SE Zusammenfassung (`https://www.dmken.com/redmine/documents/12`).

5.1 Formale Modellierung

5.1.1 Spezifikationssprachen

Spezifikationssprache Formale Notation zur Spezifikation von Modellen

Syntax Ausdrücke, welche als Spezifikation zulässig sind

Spezifikation Modell oder eine Menge von Modellen

Semantik Welches Modell/welche Modelle eine Spezifikation beschreibt
Abbildung in die Sprache der Mathematik

Beispiele Use Cases, Aktionsdiagramme, Klassendiagramme, ...

Eine Spezifikation heißt inkonsistent, wenn die die leere Menge von Modellen beschreibt.

Klassen von Spezifikationssprachen

Die Syntax ist in beiden Klassen (formal und semi-formal) eine formale Notation. Die Klassen unterscheiden sich nur in ihrer Semantik.

Formal Für jeden gültigen Ausdruck ist die Semantik eindeutig spezifiziert.

- Selektive Einsetzung bei bspw. sehr kritischen Komponenten
- Auch für die Entwicklungsdokumentation

Semi-Formal Es gibt gültige Ausdrücke, für die die Semantik nicht eindeutig spezifiziert ist (Interpretationspielraum).

- Einsetzbar für weniger kritische Komponenten
- Bei unkritischen Komponenten sind auch informelle Beschreibungen in Ordnung

5.1.2 Einsatz

Formale Spezifikation der Anforderungen

- Kommunikation zwischen Auftraggebern und Entwicklern
- Vermeidung von Missverständnissen

Formale Spezifikation während der Entwicklung

- Kommunikation zwischen den Entwicklern
- Vermeidung von Missverständnissen
- Generierung von Programmteilen (bspw. Tests) aus Spezifikation ist möglich

Formale Verifikation

- Überprüfung von Systemeigenschaften (bspw. auf Sicherheit)
- Fehlervermeidung durch mathematische Beweise

5.2 Formale Softwareentwicklung

Um Software vollständig formal zu Entwickeln gibt es zwei Ansätze, beide haben dabei den selben Startpunkt und das gleiche Ziel.

- Transformationsbasierter Ansatz

  Startpunkt Formale Spezifikation der Anforderungen

  Entwicklung Die Spezifikation wird durch Anwendung von Transformationen schrittweise modifiziert, wobei relevante Teile erhalten bleiben

  Zielpunkt Spezifikation, welche einem Programm entspricht

- Erfinde-und-Verifiziere Ansatz

  Startpunkt Formale Spezifikation der Anforderungen

  Entwicklung Erfindung einer neuen Spezifikation und Verifikation, dass alle relevanten Teile vorhanden sind

  Zielpunkt Spezifikation, welche dem Programm entspricht

5.3 Formale Verifikation

- Bedingungen
  - Kalküle zur mathematischen Beweisführung (bspw. Hoare-Logik, Refinement Calculus, ...)
- Erleichterungen
  - Automatisierung der Beweisprüfung (Ist der Beweis korrekt?)
  - Automatisierung der Beweissucher (automatische oder halbautomatische Theorembeweiser)

5.3.1 Einsatzmöglichkeiten formaler Verifikation

- Codeebene (sowohl in Programmier- als auch Maschinensprachen)
- Modellebene
  - Nachweis, dass ein Modell gegebene Eigenschaften erfüllt
  - Nachweis, dass Modele in einer gegebenen Beziehung zueinander stehen

5.3.2 Beispiel: Verifikation auf Codeebene (Hoare-Logik)

Hoare-Tripel: $\{P\} \quad C \quad \{Q\}$

P Vorbedingung (prädikatenlogische Formel)

C Programm (imperative Programmiersprache)

Q Nachbedingung (prädikatenlogische Formel)

Beispiele

- Gilt $\{x = 11\} \quad x \coloneqq (x \oplus 2) \quad \{x = 13\}$?
- Gilt $\{x > y\} \quad \texttt{while} \ (x \ \texttt{leq} \ 0) \ \texttt{do} \ x \coloneqq (x \ominus 1) \ \texttt{od} \quad \{x > y\}$?
- Gilt $\{x < y, y > 0\} \quad \texttt{while} \ (x\texttt{leq}0) \ \texttt{do} \ x \coloneqq (x \oplus 1) \ \texttt{od} \quad \{x > y\}$?

6 Verhaltensorientierte Modellierung

Erinnerung: Modelle sind absichtlich nicht originalgetreu, sondern heben bestimmte Aspekte der Realität hervor, beispielsweise die Daten, die Architektur oder das Verhalten des Systems. Die verhaltensorientierte Modellierung beschäftigt sich mit dem Hervorheben des Verhaltens der Realität/des Modells.

6.1 Komponenten

Bei der verhaltensorientierten Modellierung wird zwischen folgenden, grundlegenden Komponenten unterschieden (diese werden in den folgenden Abschnitten näher erläutert):

Zustände Eine Momentaufnahme des Systems während der Ausführung

Ereignisse Eine Gegebenheit, welche einen Zustandsübergang (Transition) auslösen kann

Transitionen Ein Übergang von einem Zustand in einen anderen (wird von einem Ereignis ausgelöst)

Spuren und Historien Mögliche Abläufe des Systems

6.1.1 Zustände

Modellierung durch eine Zustandsmenge S, welche wie folgt definiert werden kann:

- Die Menge verbleibt unterspezifiziert
- Die Zustände werden als Symbol spezifiziert und S ist die Menge dieser Symbole
- Die Zustände werden durch mathematische Konzepte (bspw. Funktionen) spezifiziert und S ist die Menge dieser Konzepte

Beispiel Sei *PRODUKTE* die Menge aller Produkte eines Bewertungssystems und

$$\mathit{BEWERTUNG} := \{1, 2, 3, 4, 5, 6\}$$

die Menge der möglichen Bewertungen, wobei die Symbole den intuitiven Schulnoten entsprechen.

Die möglichen Zustände des Bewertungssystems werden nun als Funktion $S : \mathit{PRODUKTE} \to \mathit{BEWERTUNG}^*$ spezifiziert, wobei ein Produkt mehrfach bewertet werden kann. Eine Liste von Bewertungen kann durch $S[\xi \backslash \zeta]$ ausgetauscht werden, wobei $\xi \in \mathit{PRODUKTE}$ und $\zeta \in \mathit{BEWERTUNG}*$.

6.1.2 Ereignisse

Modellierung durch eine Ereignismenge E, welche wie folgt definiert werden kann:

- Die Menge verbleibt unterspezifiziert
- Die Ereignisse werden als Symbol spezifiziert und S ist die Menge dieser Symbole
- Die Ereignisse werden durch mathematische Konzepte (bspw. Funktionen) spezifiziert und S ist die Menge dieser Konzepte

Beispiel Seien alle Definitionen aus dem vorigen Beispiel gültig.
Die möglichen Ereignisse eines Systems werden durch die Menge

$$E \coloneqq \{\text{bewerte}(\xi, \Delta) \mid \xi \in \textit{PRODUKTE} \land \Delta \in \textit{BEWERTUNG}\}$$

definiert, wobei das Ereignis bewerte(ξ, Δ) dem Produkt ξ eine neue Bewertung Δ hinzufügt.

6.1.3 Transitionen/Zustandsübergänge

Transitionen können beispielsweise durch Tupel der Form (s, e, s') modelliert werden, wobei

$e \in E$ das auslösende Ereignis darstellt,

$s \in S$ der Zustand vor dem Geschehen von e ist und

$s' \in S$ den nächsten (nach dem Geschehen von e) Zustand darstellt.

Diese Tupel werden in einer Menge $\rightarrow \subseteq S \times E \times S$ gesammelt und als „Transitionsrelation“ bezeichnet.
Dabei gibt es folgende Möglichkeiten zur Spezifikation der Transitionsrelation:

- Die möglichen Transitionen werden explizit aufgeführt
- Die Menge wird deklarativ spezifiziert

Eine unterspezifizierte Transitionsrelation ist nicht zulässig.

Beispiel Seien alle Definitionen aus dem vorigen Beispiel gültig.
Die möglichen Zuständsübergänge können folgendermaßen spezifiziert werden:

$$\rightarrow \coloneqq \{(S, \text{bewerte}(\xi, \Delta), S') \mid \xi \in \textit{PRODUKTE} \land \Delta \in \textit{BEWERTUNG} \land S' = S[\xi \backslash S(\xi).(\Delta)]\}$$

6.1.4 Transitionssysteme

Definition (Transitionssystem) Ein Transitionssystem ist ein Tupel $(S, S_0, E, \rightarrow)$, wobei

S die Menge der Zustände,

$S_0 \subseteq S$ die Menge der Start-/Anfangszustände,

E die Ereignismenge und

$\rightarrow \subseteq S \times E \times S$ die Transitionsrelation darstellt.

Notation (Transitionen) $(s, e, s') \in \rightarrow \quad \Longleftrightarrow \quad s - e \rightarrow s'$

Beispiel Seien alle Definitionen aus dem vorigen Beispiel gültig.
Dann ergibt sich folgendes Transitionssystem $\mathcal{T} = (S, S_0, E, \rightarrow)$:

$$
\begin{aligned}
S &\text{ Siehe Beispiel „Zustände“} \\
S_0 &:= \mathit{PRODUKTE} \times \{()\} && \text{(Alle Produkte starten ohne Bewertung)} \\
E &\text{ Siehe Beispiel „Ereignisse“} \\
\rightarrow &\text{ Siehe Beispiel „Transitionen“}
\end{aligned}
$$

Spuren von Transitionssystemen

Sei $\mathcal{T} = (S, S_0, E, \rightarrow)$ ein Transitionssystem.

Definition (Spuren von Transitionssystemen) Die durch $\mathcal{T}$ induzierte Menge von Spuren $\text{Traces}(\mathcal{T}) \subseteq (S \cup E)^*$ ist die kleinste Menge, sodass

$$
\begin{cases}
(s) \in \text{Traces}(\mathcal{T}) & \text{wenn } s \in S_0 \\
t.(s, e, s') \in \text{Traces}(\mathcal{T}) & \text{wenn } t.(s) \in \text{Traces}(\mathcal{T}) \text{ und } s - e \rightarrow s'
\end{cases}
$$

Intuition: Intuitiv bedeutet dies, dass die Menge $\text{Traces}(\mathcal{T})$ alle möglichen Wege durch das Transitionssystem inklusive der Zustände und der Ereignisse enthält.

Definition (Ereignis-/Zustandsspuren von Transitionssystemen) Die durch $\mathcal{T}$ induzierte Menge von Zustandsspuren $\text{S-Traces}(\mathcal{T}) \subseteq S^*$ und die Menge von Ereignisspuren $\text{E-Traces}(\mathcal{T}) \subseteq E^*$ sind wie folgt definiert:

$$
\begin{aligned}
\text{S-Traces}(\mathcal{T}) &:= \{t \upharpoonright S \mid t \in \text{Traces}(\mathcal{T})\} \\
\text{E-Traces}(\mathcal{T}) &:= \{t \upharpoonright E \mid t \in \text{Traces}(\mathcal{T})\}
\end{aligned}
$$

Historien von Transitionssystemen

Sei $\mathcal{T} = (S, S_0, E, \rightarrow)$ ein Transitionssystem.

Definition (Historien von Transitionssystemen) Die durch $\mathcal{T}$ induzierte Menge von Historien $\text{Hist}(\mathcal{T}) \subseteq (S \cup E)^\infty$ ist wie folgt definiert:

$$
\text{Hist}(\mathcal{T}) := \{h : \mathbb{N}_0 \rightarrow (S \cup E) \mid h(0) \in S_0 \wedge \forall n \in \mathbb{N}_0 : h(2n) - h(2n+1) \rightarrow h(2n+2)\}
$$

Intuition: Intuitiv bedeutet dies, dass die Menge $\text{Hist}(\mathcal{T})$ alle unendlichen Wege durch das Transitionssystem enthält, wobei jede Historie mit einem Startzustand startet, gefolgt von einem Ereignis, gefolgt von einem Zustand, ...

Definition (Ereignis-/Zustandshistorien von Transitionssystemen) Die durch $\mathcal{T}$ induzierte Menge von Zustandshistorien S-Hist$(\mathcal{T}) \subseteq S^\infty$ und die Menge von Ereignishistorien E-Hist$(\mathcal{T}) \subseteq E^\infty$ sind wie folgt definiert:

$$\text{S-Hist}(\mathcal{T}) \coloneqq \{h : \mathbb{N}_0 \to S \mid \exists h' \in \text{Hist}(\mathcal{T}) : \forall n \in \mathbb{N}_0 : h(n) = h'(2n)\}$$
$$\text{E-Hist}(\mathcal{T}) \coloneqq \{h : \mathbb{N}_0 \to S \mid \exists h' \in \text{Hist}(\mathcal{T}) : \forall n \in \mathbb{N}_0 : h(n) = h'(2n+1)\}$$

6.1.5 Spuren

Definition (Spur) Eine Spur ist eine endliche Folge von Zuständen und Ereignissen.
Enthält die Spur nur Ereignisse, so heißt sie Ereignisspur.
Enthält die Spur nur Zustände, so heißt sie Zustandsspur.

Definition (Konkatentation) Die Konkatenation von zwei Spuren $t_1, t_2 \in (S \cup E)^*$ ist die wie folgt definierte Spur $t_1.t_2 \in (S \cup E)^*$:

$$t_1.t_2 \coloneqq \begin{cases} t_1 & \text{wenn } t_2 = () \\ (t_1.t_2, q) & \text{wenn } t_2 = (t_2', q) \end{cases}$$

Definition (Projektion) Die Projektion einer Spur $t \in (S \cup E)^*$ nach einer Menge $Q \subseteq S \cup E$ ist die wie folgt definierte Spur:

$$t \restriction Q \coloneqq \begin{cases} () & \text{wenn } t = () \\ (t' \restriction Q, q) & \text{wenn } t = t'.(q) \text{ und } q \in Q \\ t' \restriction Q & \text{wenn } t = t'.(q) \text{ und } q \notin Q \end{cases}$$

Intuition: Intuitiv bedeutet dies, dass die Projektion eine Teilspur aus t extrahiert mit den Elementen, welche in Q vorhanden sind (in der Reihenfolge, in der sie in t vorkommen). Dies kann beispielsweise mit $Q = E$ dazu genutzt werden, eine beliebige Spur in eine Ereignisspur umzuwandeln. Warnung: Hierbei können wichtige Daten verloren gehen.

Definition (Präfix) Eine Spur t' heißt Präfix einer Spur t gdw. es eine Spur t'' gibt, sodass $t = t'.t''$ gilt. Notation: $t' \leq t \iff t'$ ist Präfix von t

Definition (Verschachtlungsmenge) Seien $t, u \in E^* \cup (E^* \times \{\surd\})$ zwei Spuren.
Die Menge der Verschachtlungen Interleaving(t, u) von t und u ist wie folgt rekursiv definiert (in der Definition als „IV“ abgekürzt):

$$\text{IV}(t,u) \coloneqq \begin{cases} \{t\} & \text{falls } u = () \\ \{u\} & \text{falls } t = () \\ \{(x).s \mid s \in \text{IV}(t', u)\} \cup \{(y).s \mid s \in \text{IV}(t, u')\} & \text{falls } t = (x).t' \wedge u = (y).u' \end{cases}$$

Beispiel

Spur Die Spur

$$
\begin{aligned}
&(\{(\text{HANDY}, ()), (\text{LAPTOP}, ())\},\\
&\text{bewerte}(\text{HANDY}, 2),\\
&\{(\text{HANDY}, (2)), (\text{LAPTOP}, ())\},\\
&\text{bewerte}(\text{LAPTOP}, 5),\\
&\{(\text{HANDY}, (2)), (\text{LAPTOP}, (5))\},\\
&\text{bewerte}(\text{HANDY}, 4)\\
&\{(\text{HANDY}, (2, 4)), (\text{LAPTOP}, (5))\})
\end{aligned}
$$

ist wie folgt zu interpretieren:

1. Das System startet im Anfangszustand.
2. Es wird eine Bewertung 2 zum Produkt „HANDY“ hinzugefügt, an dem Produkt „LAPTOP“ wird nichts geändert.
3. Es wird eine Bewertung 5 zum Produkt „LAPTOP“ hinzugefügt, an dem Produkt „HANDY“ wird nichts geändert.
4. Es wird eine Bewertung 4 zum Produkt „HANDY“ hinzugefügt, an dem Produkt „LAPTOP“ wird nichts geändert.

Ereignisspur Die Ereignisspur

$$
\begin{aligned}
&(\text{bewerte}(\text{HANDY}, 2),\\
&\text{bewerte}(\text{LAPTOP}, 5),\\
&\text{bewerte}(\text{HANDY}, 4))
\end{aligned}
$$

ist wie folgt zu interpretieren:

1. Ein Nutzer bewertet das Produkt „HANDY“ mit einer 2.
2. Ein Nutzer bewertet das Produkt „LAPTOP“ mit einer 5.
3. Ein Nutzer bewertet das Produkt „HANDY“ mit einer 4.

Zustandsspur Die Zustandsspur

$$
\begin{aligned}
&(\{(\text{HANDY}, ()), (\text{LAPTOP}, ())\},\\
&\{(\text{HANDY}, (2)), (\text{LAPTOP}, ())\},\\
&\{(\text{HANDY}, (2)), (\text{LAPTOP}, (5))\},\\
&\{(\text{HANDY}, (2, 4)), (\text{LAPTOP}, (5))\})
\end{aligned}
$$

ist wie folgt zu interpretieren:

1. Das System startet im Anfangszustand.
2. Es wird eine Bewertung 2 zum Produkt „HANDY“ hinzugefügt.
3. Es wird eine Bewertung 5 zum Produkt „LAPTOP“ hinzugefügt.
4. Es wird eine Bewertung 4 zum Produkt „HANDY“ hinzugefügt.

6.1.6 Historien

Definition (Historie) Eine Historie ist eine unendliche Folge von Ereignissen und Zuständen. Historien können zur Modellierung von nicht-terminierenden Ausführungen und zur Modellierung von terminierenden Ausführungen eingesetzt werden.

Enthält die Historie nur Ereignisse, so heißt sie Ereignishistorie.
Enthält die Historie nur Zustände, so heißt sie Zustandshistorie.

Definition (Terminierende Historien) Der Haupteinsatz von Historien ist die Modellierung von Modellen, welche nicht Terminieren.

Durch die Einführung eines Ereignisses $\surd$ ist es möglich, auch terminierende Modell zu spezifizieren:

- Das Ereignis $\surd$ modelliert die Terminierung einer Ausführung.
- Tritt $\surd$ zum ersten Mal an einer Stelle n auf ($\mathit{hist}(n) = \surd$), so gilt für alle $n' > n$, dass $\mathit{hist}(n') = \mathit{hist}(n' - 1)$, beziehungsweise $\mathit{hist}(n') = \surd$. Somit ist der letzte Zustand der Historie an der Stelle $n - 1$, welcher unverändert bleibt.

6.2 Modulare Modellierung

Mit der Modellierung von nebenläufigen Komponenten ist eine modulare Modellierung möglich, wodurch

- das Modell strukturiert und
- die Komplexität des Modellierens reduziert wird.

Hierzu ist eine Definition der Komposition von Modellen nötig und es muss klar definiert sein, wann die Komposition zweier angemessener Modelle wiederum ein angemessenes Modell ergibt.

Varianten nebenläufiger Ausführung:

Synchrone Ausführung Die Systemkomponenten führen gleichzeitig einen Berechnungsschritt durch.

Asynchrone Ausführung Die Systemkomponenten führen unabhängig voneinander ihre Berechnungsschritte durch.

6.2.1 Ausführung ohne Kommunikation (Produktkomposition)

Dieser Abschnitt beschäftigt sich mit der Produktkomposition zweier nebenläufiger Komponenten, welche genutzt werden kann, wenn die Komponenten nicht miteinander kommunizieren.

Seien $\mathcal{T}^1 = (S^1, S_0^1, E^1, \rightarrow^1)$ und $\mathcal{T}^2 = (S^2, S_0^2, E^2, \rightarrow^2)$ zwei Transitionssysteme.

Synchrone Produktkomposition

Definition (Synchrone Produktkomposition) Die synchrone Produktkomposition von $\mathcal{T}^1$ und $\mathcal{T}^2$ ergibt das Transitionssystem $\mathcal{T} = (S, S_0, E, \rightarrow)$, wobei

$$
\begin{aligned}
S &:= S^1 \times S^2 \\
S_0 &:= S_0^1 \times S_0^2 \\
E &:= E^1 \times E^2 \\
\rightarrow &:= \{((s_1, s_2), (e_1, e_2), (s_1', s_2')) \in S \times E \times S \mid (s_1, e_1, s_1') \in \rightarrow^1 \wedge (s_2, e_2, s_2') \in \rightarrow^2\}
\end{aligned}
$$

Intuition: Damit $\mathcal{T}$ einen Schritt machen kann, müssen $\mathcal{T}^1$ und $\mathcal{T}^2$ jeweils einen Schritt machen.

Asynchrone Produktkomposition

Definition (Asynchrone Produktkomposition) Die asynchrone Produktkomposition von $\mathcal{T}^1$ und $\mathcal{T}^2$ ergibt das Transitionssystem $\mathcal{T} = (S, S_0, E, \rightarrow)$, wobei

$$
\begin{aligned}
S &:= S^1 \times S^2 \\
S_0 &:= S_0^1 \times S_0^2 \\
E &:= E^1 \cup E^2 \\
\rightarrow &:= \{((s_1, s_2), e, (s_1', s_2')) \in S \times E \times S \\
&\qquad \mid ((s_1, e, s_1') \in \rightarrow^1 \wedge e \in E^1 \wedge s_2' = s_2) \vee ((s_2, e, s_2') \in \rightarrow^2 \wedge e \in E^2 \wedge s_1' = s_1)\}
\end{aligned}
$$

Intuition: Damit $\mathcal{T}$ einen Schritt machen kann, muss entweder $\mathcal{T}^1$ oder $\mathcal{T}^2$ einen Schritt machen.

Theorem Sei $\mathcal{T} = (S, S_0, E, \rightarrow)$ die asynchrone Produktkomposition von $\mathcal{T}^1$ und $\mathcal{T}^2$.
Wenn $E^1 \cap E^2 = \emptyset$ gilt, dann gilt:

$$
\begin{aligned}
\rightarrow^1 &= \{(s_1, e_1, s_1') \in S^1 \times E^1 \times S^1 \mid \exists s_2 \in S^2 : ((s_1, s_2), e_1, (s_1', s_2)) \in \rightarrow\} \\
\rightarrow^2 &= \{(s_2, e_2, s_2') \in S^2 \times E^2 \times S^2 \mid \exists s_1 \in S^1 : ((s_1, s_2), e_2, (s_1, s_2')) \in \rightarrow\}
\end{aligned}
$$

6.3 Ausführung mit Kommunikation

Sollen zwei Komponenten innerhalb eines Modells miteinander kommunizieren, so gibt es folgende Möglichkeiten zur Kommunikation:

Shared Memory Eine Komponente aktualisiert dem gemeinsamen Speicher; alle anderen Komponenten sehen den aktualisierten Speicher
Problematik: Umgang mit Schreibkonflikte

Message Passing Eine Komponente sendet eine Nachricht, eine anderen Komponente empfängt die Nachricht

Seien $\mathcal{T}^1 = (S^1, S_0^1, E^1, \rightarrow^1)$ und $\mathcal{T}^2 = (S^2, S_0^2, E^2, \rightarrow^2)$ zwei Transitionssysteme.

6.3.1 Shared Memory (asynchron)

Definition (Speicherkomponierbarkeit) Zwei Transitionssysteme $\mathcal{T}^1$ und $\mathcal{T}^2$ heißen speicherkomponierbar gdw. sich die Zustände in S^1 und S^2 in einen lokalen und einen globalen Teil zerlegen lassen, d.h. wenn es S_L^1, S_L^2 und S_G gibt, sodass $S^1 = S_L^1 \times S_G$ und $S^2 = S_L^2 \times S_G$ gelten.

Der Wertebereich S_G modelliert den gemeinsamen Speicher, die Wertebereiche S_L^1 und S_L^2 modellieren jeweils den lokalen Speicher von $\mathcal{T}^1$ und $\mathcal{T}^2$.

Definition (Shared Memory Komposition) Seien $\mathcal{T}^1$ und $\mathcal{T}^2$ speicherkomponierbar.

Dann ergibt die Shared Memory Komposition von $\mathcal{T}^1$ und $\mathcal{T}^2$ das Transitionssystem $\mathcal{T} = (S, S_0, E, \to)$, wobei

$$
\begin{aligned}
S &:= S_L^1 \times S_L^2 \times S_G \\
S_0 &:= \{(s_L^1, s_L^2, s_G) \in S \mid (s_L^1, s_G) \in S_0^1 \wedge (s_L^2, s_G) \in S_0^2\} \\
E &:= E^1 \cup E^2 \\
\to &:= \{((s_L^1, s_L^2, s_G), e, (s_L^{1'}, s_L^{2'}, s_G')) \in S \times E \times S \\
&\qquad \mid (((s_L^1, s_G), e, (s_L^{1'}, s_G')) \in \to^1 \wedge e \in E^1 \wedge s_L^{2'} = s_L^2) \vee (((s_L^2, s_G), e, (s_L^{2'}, s_G')) \in \to^2 \wedge e \in E^2 \wedge s_L^{1'} = s_L^1)\}
\end{aligned}
$$

6.3.2 Message Passing (asynchron)

Definition (Message Passing Komposition) Die Message Passing Komposition von $\mathcal{T}^1$ und $\mathcal{T}^2$ ergibt das Transitionssystem $\mathcal{T} = (S, S_0, E, \to)$, wobei

$$
\begin{aligned}
S &:= S^1 \times S^2 \\
S_0 &:= S_0^1 \times S_0^2 \\
E &:= E^1 \cup E^2 \\
\to &:= \{((s_1, s_2), e, (s_1', s_2')) \in S \times E \times S \\
&\qquad \mid ((s_1, e, s_1') \in \to^1 \wedge (s_2, e, s_2') \in \to^2 \wedge e \in E^1 \cap E^2) \\
&\qquad\quad \vee ((s_1, e, s_1') \in \to^1 \wedge e \in E^1 \setminus E^2 \wedge s_2' = s_2) \\
&\qquad\quad \vee ((s_2, e, s_2') \in \to^2 \wedge e \in E^2 \setminus E^1 \wedge s_1' = s_1)\}
\end{aligned}
$$

Intuition: Gemeinsame Ereignisse verursachen einen einen synchronen Zustandsübergang in beiden Komponenten; lokale Ereignisse verursachen nur einen Übergang in einer Komponente.

6.3.3 Shared Memory (synchron)

6.4 Formale Spezifikationssprache

6.4.1 Vergleich Transitionssystem

- Eine Modellierung durch Transitionssysteme geben nur an, welche Transitionen möglich sind, geben aber keine weiterführenden Informationen

- Systemläufe werden durch Spuren und Historien modelliert (durch induzierte Mengen auf dem Transitionssystem)
- Mögliche Systemläufe können auch direkt durch Mengen von Spuren/Historien modelliert werden

6.4.2 Prozesse

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

E eine Menge von Ereignissen und

$\mathit{Tr} \subseteq E^* \cup (E^* \times \{\surd\})$ eine nichtleere Menge von Ereignisspuren ist.

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

Intuition:

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

Definition (Prozessalphabet) Die Funktion $a(\cdot)$ liefert für einen Prozess die Menge an Ereignissen. Für $P = (E, \mathit{Tr})$ gilt somit $a(P) = E$. Die Menge $a(P)$ wird als Alphabet von P bezeichnet.

Definition (Prozessspuren) Die Funktion traces$(\cdot)$ liefert für einen Prozess die Menge an Spuren. Für $P = (E, \mathit{Tr})$ gilt somit $a(P) = \mathit{Tr}$.

Modellierung des internen Verhaltens E wird so definiert, dass jede Ein- und Ausgabeaktion und jede interne Aktion durch ein Ereignis in E modelliert wird.

Kommunikation an einer Schnittstelle E wird so definiert, dass jede Ein- und Ausgabeaktion durch ein Ereignis in E modelliert wird. Interne Aktionen werden hierbei nicht modelliert.

6.4.3 Modellierung von Anforderungen

Definition (Anforderungsmodellierung) $PsatS$ spezifiziert, dass der durch P spezifizierte Prozess die durch S modellierte Eigenschaft hat.

- P ist ein Prozessausdruck
- S ist ein Prädikat auf Spuren, d.h. $S : E^* \to \mathbb{B}$
- $PsatS$ wird „P erfüllt S“ gesprochen

Semantik:

$$\forall \mathit{tr} \in \text{traces}(P) : S(\mathit{tr})$$

Definition (Anforderungsmodellierung mit Reihenfolge) Seien E ein Alphabet und $F, G \subseteq E$ beliebig. *Psat* $\text{BEFORE}_{F,G}$ spezifiziert, dass vor der Ausführung eines Ereignisses aus G zuvor ein Ereignis aus F geschehen muss, wobei das Prädikat $\text{BEFORE}_{F,G}$ wie folgt spezifiziert ist:

$$\text{BEFORE}_{F,G}(tr) \coloneqq (tr \restriction G \neq ()) \implies (tr \restriction F \neq ())$$

6.4.4 Entwurf einer Spezifikationssprache

Ziel Eine formale Sprache zur Spezifikation von Prozessen

Muss-Kriterien formal; formal definierte Semantik

Kann-Kriterien möglichst ausdrucksmächtig; möglichst natürlich (nur schwer objektivierbar)

Anatz

- Entwurf einer Teilsprache für endliche Mengen von Spuren
  - *STOP* und *SKIP*
  - Aktionspräfixe
  - nicht-deterministische Auswahl
- Schrittweise Einführung von Operatoren
  - rekursive Definitionen
  - sequentielle Komposition
  - modulare Spezifikation nebenläufiger Systeme (Synchronisation und Verschachtlung)

Ausdrucksstärke

Dieser Abschnitt beschäftigt sich mit der Ausdrucksstärke der Prozessausdruckssprache, wobei die Sprache auf

$$\begin{aligned}\langle P\rangle ::= \ & \mathit{STOP}_E \\ \mid \ & \mathit{SKIP}_E \\ \mid \ & (X \to \langle P\rangle) \\ \mid \ & (\langle P\rangle \sqcap \langle P\rangle)\end{aligned}$$

beschränkt wird.

Theorem Jeder Ausdruck der Prozessausdruckssprache spezifiziert einen Prozess (E, Tr) mit endlicher Menge *Tr*.

Definition (Alphabet/Spur zu Prozessausdruck) Die Funktion „process“ liefert für jedes Alphabet E und jede Spur $t \in E^* \cup (E^* \times \{\surd\})$ einen Prozessausdruck.

Die Funktion ist folgendermaßen rekursiv definiert:

$$\text{process}(E,t) \coloneqq \begin{cases} \mathit{STOP}_E & \text{wenn } t = () \\ \mathit{SKIP}_E & \text{wenn } t = (\surd) \\ (x \to \text{process}(E,t')) & \text{wenn } t = (x).t' \wedge x \neq \surd \end{cases}$$

Theorem Für den Prozessausdruck process(E, t) gilt:

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

Theorem Jeder Prozess (E, Tr) mit endlicher und unter Präfixbildung abgeschlossener Menge *Tr* kann durch einen Ausdruck in der oben definierten Prozessausdruckssprache spezifiziert werden.

Syntax/Semantik

Prozessausdruck $STOP_E$

Definition (Prozessausdruck $STOP_E$) Für jedes Alphabet E gibt es einen Prozessausdruck $STOP_E$, welcher ein System modelliert, welches nichts tut.

Semantik:

$$a(STOP_E) \coloneqq E$$
$$\text{traces}(STOP_E) \coloneqq \{()\}$$

Prozessausdruck $SKIP_E$ (Terminierung)

Definition (Prozessausdruck $SKIP_E$) Für jedes Alphabet E gibt es einen Prozessausdruck $SKIP_E$, welcher ein System modelliert, welches terminiert aber sonst nichts tut.

Semantik:

$$a(SKIP_E) \coloneqq E$$
$$\text{traces}(SKIP_E) \coloneqq \{(), (\surd)\}$$

$() \in \text{traces}(SKIP_E)$ muss gelten, da die Menge sonst nicht unter der Präfixbildung abgeschlossen wäre.

Warning: Die Prozessausdrücke $STOP_E$ und $SKIP_E$ sind nicht äquivalent.

Prozessausdruck $(x \to P)$ (Reihenfolge)

Definition (Prozessausdruck $(x \to P)$) Der Prozessausdruck $(x \to P)$ spezifiziert einen Prozess, der sich zunächst am Ereignis x beteiligt und danach die der Prozessausdruck P verhält.

- $(x \to P)$ ist nur dann zulässig, wenn $x \in a(P)$ und $x \neq \surd$ gilt.
- $(x \to P)$ wird „x vor P“ gesprochen

  Semantik:

$$a((x \to P)) \coloneqq a(P)$$
$$\text{traces}((x \to P)) \coloneqq \{()\} \cup \{(x).t \mid t \in \text{traces}(P)\}$$

Prozessausdruck $(P \sqcap Q)$ (nichtdeterministische Wahl)

Definition (Prozessausdruck $(P \sqcap Q)$) Der Prozessausdruck $(P \sqcap Q)$ spezifiziert einen Prozess, der sich entweder wie der Prozessausdruck P oder wie der Prozessausdruck Q verhält, wobei die Entscheidung nicht von dem System beeinflusst werden kann.

- $(P \sqcap Q)$ ist nur dann zulässig, wenn $a(P) = a(Q)$ gilt.
- $(P \sqcap Q)$ wird „P oder Q“ gesprochen

  Semantik:

$$a((P \sqcap Q)) \coloneqq a(P) = a(Q)$$
$$\text{traces}((P \sqcap Q)) \coloneqq \text{traces}(P) \cup \text{traces}(Q)$$

Prozessausdruck $(\sqcap_{j \in I} P(j))$ (n-stellige nichtdeterministische Wahl)

Definition (Prozessausdruck $(\sqcap_{j \in I} P(j))$) Der Prozessausdruck $(\sqcap_{j \in I} P(j))$ spezifiziert einen Prozess, der sich wie einer der Prozessausdrücke $P(j)$ verhält, wobei die Entscheidung nicht von dem System beeinflusst werden kann.

- $(\sqcap_{j \in I} P(j))$ ist nur dann zulässig, wenn $\forall i, j \in I : a(P(i)) = a(P(j))$ gilt.

  Semantik:

$$a((\sqcap_{j \in I} P(j))) \coloneqq a(P(j)) \text{ (für ein beliebiges)} j \in I$$
$$\text{traces}((\sqcap_{j \in I} P(j))) \coloneqq \bigcup_{j \in I} \text{traces}(P(j))$$

Prozessausdruck $(P \,\square\, Q)$ (deterministische Wahl)

Definition (Prozessausdruck $(P \,\square\, Q)$) Der Prozessausdruck $(P \,\square\, Q)$ spezifiziert einen Prozess, der sich entweder wie der Prozessausdruck P oder wie der Prozessausdruck Q verhält, wobei die Entscheidung von dem System beeinflusst werden kann.

- $(P \,\square\, Q)$ ist nur dann zulässig, wenn $a(P) = a(Q)$ gilt.
- $(P \,\square\, Q)$ wird „P Wahl Q“ gesprochen

  Semantik:

$$a((P \,\square\, Q)) \coloneqq a(P) = a(Q)$$
$$\text{traces}((P \,\square\, Q)) \coloneqq \text{traces}(P) \cup \text{traces}(Q)$$

Prozessausdruck $(\square_{j \in I} P(j))$ (n-stellige deterministische Wahl)

Definition (Prozessausdruck $(\square_{j \in I} P(j))$) Der Prozessausdruck $(\square_{j \in I} P(j))$ spezifiziert einen Prozess, der sich wie einer der Prozessausdrücke $P(j)$ verhält, wobei die Entscheidung von dem System beeinflusst werden kann.

- $(\square_{j \in I} P(j))$ ist nur dann zulässig, wenn $\forall i, j \in I : a(P(i)) = a(P(j))$ gilt.

  Semantik:

$$a((\square_{j \in I} P(j))) \coloneqq a(P(j)) \text{ (für ein beliebiges)} j \in I$$
$$\text{traces}((\square_{j \in I} P(j))) \coloneqq \bigcup_{j \in I} \text{traces}(P(j))$$

Prozessausdruck $(P; Q)$ (Konkatenation)

Definition (Prozessausdruck $(P; Q)$) Der Prozessausdruck $(P; Q)$ spezifiziert einen Prozess, der sich erst wie der Prozessausdruck P verhält und anschließend wie der Prozessausdruck Q.

- $(P; Q)$ ist nur dann zulässig, wenn $a(P) = a(Q)$ gilt.
- $(P; Q)$ wird „P und danach Q“ gesprochen

  Semantik:

$$a((P; Q)) \coloneqq a(P) = a(Q)$$
$$\text{traces}((P; Q)) \coloneqq \{s \in \text{traces}(P) \mid a \restriction \{\surd\} = ()\} \cup \{s.t \mid s.(\surd) \in \text{traces}(P) \wedge t \in \text{traces}(Q)\}$$

Prozessausdruck $(P \parallel Q)$ (Synchrone Komposition)

Definition (Prozessausdruck $(P \parallel Q)$) Der Prozessausdruck $(P \parallel Q)$ spezifiziert einen Prozess, der die durch P und Q spezifizierten Prozesse nebenläufig synchron ablaufen lässt.

- $(P \parallel Q)$ ist nur dann zulässig, wenn $a(P) \neq a(Q)$ gilt.
- $(P \parallel Q)$ wird „P parallel Q“ gesprochen

  Semantik:

$$a((P \parallel Q)) := a(P) \cup a(Q)$$
$$\text{traces}((P \parallel Q)) := \{t \in (a(P) \cup a(Q))^* \mid (t \restriction (a(P))) \in \text{traces}(P) \wedge (t \restriction (a(Q))) \in \text{traces}(Q)\}$$

Prozessausdruck $(P \,|||\, Q)$ (Asynchrone Komposition)

Definition (Prozessausdruck $(P \,|||\, Q)$) Der Prozessausdruck $(P \,|||\, Q)$ spezifiziert einen Prozess, der die durch P und Q spezifizierten Prozesse nebenläufig asynchron ablaufen lässt.

- $(P \,|||\, Q)$ ist nur dann zulässig, wenn $a(P) = a(Q)$ gilt.
- $(P \,|||\, Q)$ wird „P verschachtelt Q“ gesprochen

  Semantik:

$$a((P \,|||\, Q)) := a(P) = a(Q)$$
$$\text{traces}((P \,|||\, Q)) := \{s \in (a(P) \cup a(Q))^* \mid \exists t \in \text{traces}(P) : \exists u \in \text{traces}(Q) : s \in \text{Interleaving}(t, u)\}$$

Prozessausdruck $\mu X : E.F(X)$ (Fixpunktoperator) Zur Vermeidung von rekursiven Gleichungssystemen kann ein Fixpunktoperator eingeführt werden.

Definition (Fixpunktoperator) Sei F eine Funktion, die für jeden Prozessbezeichner *id* $\in$ *ID* einen Prozessausdruck liefert.
Dann spezifiziert der Prozessausdruck $\mu X : E.F(X)$ einen Prozess X mit Alphabet E, der sich entsprechend des Ausdrucks $F(X)$ verhält.

- $\mu X : E.F(X)$ ist nur dann zulässig, wenn $F(X)$ nur Ereignisse enthält, die in der Menge E vorhanden sind und wenn jedes Vorkommen von X in $F(X)$ bewacht ist.

  Semantik:

$$a(\mu X : E.F(X)) := E$$
$$\text{traces}(\mu X : E.F(X)) := \text{Menge der Spuren zur Lösung der Gleichung } X =_E F(X)$$

Unendliche Systeme

Unendliche Systeme lassen sich beispielsweise durch eine unendliche Menge von Spuren modellieren.

Rekursive Gleichungssysteme

Ansatz Ein Prozess wird durch mehrere Prozessausdrücke aus der oben spezifizierten Prozessausdruckssprache, welche die genannten Seitenbedingungen erfüllen, 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 Prozess angibt.

Ereignisse dürfen eben so wenig wie Bestandteile der Sprache (bspw. $STOP_E$) als Bezeichner verwendet werden.

> **Warning:** Das Gleichungssystem darf für jeden Prozessbezeichner $id \in ID$ nur genau eine Gleichung enthalten, in der *id* auf der linken Seite auftritt.

> **Warning:** Das Gleichungssystem darf nur Gleichungen enthalten, deren rechte Seite „bewacht“ ist. Das heißt, für jedes Vorkommen eines Bezeichners id' auf der rechten Seite einer Gleichung $id =_E P$ muss es einen Teilausdruck $(x \to P')$ geben, der das Vorkommen von id' enthält. Das Ereignis x heißt Wächter.

Semantik Sei *ID* eine Menge von Prozessbezeichnern, *GS* eine Menge von Gleichungen der Form $\{id =_E P \mid id \in ID\}$, sodass *GS* genau eine Gleichung für jeden Prozessbezeichner $id \in ID$ enthält und jede Gleichung in *GS* bewacht ist.

Die Semantik eines Prozessausdruck *id* unter *GS* ist wie folgt definiert:

$$a(id) \coloneqq E \qquad ((id =_E P) \in GS)$$

$$\text{traces}(id) \coloneqq Tr \qquad ((id =_E P) \in GS \text{ und } Tr \text{ Lösung von } \text{traces}(id) = \text{traces}(P))$$

6.5 Modellierung von Systemeigenschaften

Zur Modellierung von Anforderungen wird angenommen, dass bestimmte Strukturen im Modell definiert werden (bspw. Zustände durch Funktionen), auf denen die Anforderungen aufbauen.

Sei $\mathcal{T} = (S, S_0, E, \to)$ ein Transitionssystem, welches das Systemverhalten als Systemmodell darstellt. Anforderungen an das System können nur mit Hilfe prädikatenlogischer Formeln auf dem Transitionssystem modelliert werden.

Die Modellierung der Anforderungen durch ein Prädikaten K ist angemessen, wenn gilt

- $K(\mathcal{T})$ gilt für ein Transitionssystem, $\mathcal{T} = (S, S_0, E, \to)$ gdw.
- das durch $\mathcal{T}$ modellierte System die durch K modellierte Anforderung erfüllt.

Siehe auch 6.4.3 für Notationen zur Anforderungsmodellierung auf Prozessausdrücken.

6.5.1 Beispiel: Modellierung einer Robotersteuerung

6.5.2 Beispiel: Sicherheitslücke in einem Kommunikationsprotokoll