Sprachtheoretische Semantik von Interaktionsausdrücken

Christian Heinlein, Abt. DBIS

Juni 1997

1. Einleitung

Dieser Bericht definiert *Interaktionsausdrücke*, wie sie in den Berichten „Grundlagen von Interaktionsausdrücken“ (UIB 97-09) und “Interaction Expressions − A Powerful Formalism for Describing Inter-Workflow Dependencies” (UIB 97-04) beschrieben sind, mittels ihrer *sprachtheoretischen Semantik*.

Nach einer kurzen Definition der wichtigsten Grundbegriffe werden in Abschnitt 2 zunächst *reguläre Ausdrücke*, die eine Teilmenge von Interaktionsausdrücken darstellen, mittels ihrer sprachtheoretischen Semantik definiert. Anschließend wird diese Definition auf elementare Interaktionsausdrücke erweitert und ein erster Versuch zur Definition von Quantoren unternommen.

Abschnitt 3 zeigt jedoch Schwierigkeiten und Grenzen dieser „traditionellen“ Vorgehensweise auf, so daß in Abschnitt 4 eine „revidierte“ Definition von Interaktionsausdrücken gegeben wird, die im wesentlichen auf einer sauberen Trennung von *vollständigen* und *teilweisen* Worten basiert.

Alle wesentlichen Definitionen werden jeweils durch Beispiele illustriert, um die abstrakten Konzepte ein wenig „greifbarer“ zu machen.

2. Traditioneller Ansatz

2.1 Grundbegriffe

Gegeben sei eine Menge Σ von *Aktionen* (vgl. Grundlagenbericht), die auch *Alphabet* genannt wird. Formal ist eine Aktion a ein Tupel der Gestalt $(a_0, a_1, \ldots, a_n)$ mit $n \in \mathbb{N}_0$, wobei a_0 der *Name* und $a_1, \ldots, a_n$ die (optionalen) *Parameter* der Aktion sind. Der Einfachheit halber wird eine parameterlose Aktion (a_0) jedoch als „Prozedur“ a_0 und eine parametrisierte Aktion $(a_0, a_1, \ldots, a_n)$ als „Funktion“ $a_0(a_1, \ldots, a_n)$ geschrieben.

Ein *Wort* der *Länge* n (über dem Alphabet Σ) ist ein n-Tupel ‘$a_1 \ldots a_n$’ mit $a_1, \ldots, a_n \in \Sigma$ ($n \in \mathbb{N}_0$). Das Wort ‘’ der Länge 0 heißt auch *leeres Wort*. Die Länge eines Wortes w wird mit $|w|$ bezeichnet.

Die Menge aller Worte über dem Alphabet Σ wird mit Σ^* bezeichnet.

2.2 Reguläre Ausdrücke

Interaktionsausdrücke stellen eine Erweiterung von regulären Ausdrücken dar.[1] Daher scheint es naheliegend, die sprachtheoretische Semantik regulärer Ausdrücke geeignet zu erweitern, um eine entsprechende Semantik für Interaktionsausdrücke zu erhalten.

Die Semantik eines regulären Ausdrucks x wird gewöhnlich als *Sprache* $L(x)$, d. h. als die Menge aller von diesem Ausdruck *akzeptierten Worte*, definiert. Für einen beliebigen regulären Ausdruck x (der nicht gleich der leeren Menge ist, vgl. Fußnote) läßt sich $L(x)$ wie folgt rekursiv definieren:

- Für einen leeren Ausdruck $x = \lambda$:
  $L(x) = \{ \text{‘’} \}$.
- Für einen atomaren Ausdruck $x = a$:
  $L(x) = \{ \text{‘}a\text{’} \}$.
- Für eine Disjunktion $x = y \mid z$:
  $L(x) = L(y) \cup L(z) = \{ w \mid w \in L(y) \vee w \in L(z) \}$.
- Für eine sequentielle Komposition $x = y - z$:
  $L(x) = L(y)\,L(z) = \{ uv \mid u \in L(y), v \in L(z) \}$.
- Für eine sequentielle Iteration $x = y*$:
  $L(x) = L(y)* = \bigcup_{n=0}^{\infty} L(y)^n = \bigcup_{n=0}^{\infty} \{ u^1 \ldots u^n \mid u^1, \ldots, u^n \in L(y) \}$.

Hierbei steht $uv = \text{‘}u_1 \ldots u_m v_1 \ldots v_n\text{’}$ für die *Konkatenation* der beiden Worte $u = \text{‘}u_1 \ldots u_m\text{’}$ und $v = \text{‘}v_1 \ldots v_n\text{’}$. Entsprechend steht $u^1 \ldots u^n$ für die Konkatenation der n Worte $u^1, \ldots, u^n$ ($n > 0$) bzw. für das leere Wort ($n = 0$). Bei den hochgestellten Zahlen handelt es sich hier also nicht um Exponenten, sondern um Indizes. Beachte, daß mit u_i (Index unten) Aktionen $\in \Sigma$ und mit u^i (Index oben) Worte $\in \Sigma^*$ bezeichnet werden.

Beispiele

a) $L(a - b) = \{ uv \mid u \in L(a), v \in L(b) \} = \{ uv \mid u \in \{ \text{‘}a\text{’} \}, v \in \{ \text{‘}b\text{’} \} \} = \{ \text{‘}a\text{’}\text{‘}b\text{’} \} = \{ \text{‘}ab\text{’} \}$.

b) $L(a - b \mid c) = L(a - b) \cup L(c) = \{ \text{‘}ab\text{’} \} \cup \{ \text{‘}c\text{’} \} = \{ \text{‘}ab\text{’}, \text{‘}c\text{’} \}$.

c) $L((a - b \mid c)*) = \bigcup_{n=0}^{\infty} \{ u^1 \ldots u^n \mid u^1, \ldots, u^n \in L(a - b \mid c) \} =$
$\bigcup_{n=0}^{\infty} \{ u^1 \ldots u^n \mid u^1, \ldots, u^n \in \{ \text{‘}ab\text{’}, \text{‘}c\text{’} \} \} = \{ \text{‘’}, \text{‘}ab\text{’}, \text{‘}c\text{’}, \text{‘}abab\text{’}, \text{‘}abc\text{’}, \text{‘}cab\text{’}, \text{‘}cc\text{’}, \ldots \}$.

2.3 Elementare Interaktionsausdrücke

Die obigen Definitionen lassen sich wie folgt für Interaktionsausdrücke erweitern:

- Für eine Konjunktion $x = y \,\&\, z$:
  $L(x) = L(y) \cap L(z) = \{ w \mid w \in L(y), w \in L(z) \}$.
- Für eine parallele Komposition $x = y + z$:
  $L(x) = L(y) \otimes L(z) = \{ w \mid w \in u \otimes v, u \in L(y), v \in L(z) \}$.

[1] … wenn man von der leeren Menge ∅ absieht, die formal ein regulärer Ausdruck, jedoch kein Interaktionsausdruck ist.

- Für eine parallele Iteration $x = y\,\#$:

$$\mathrm{L}(x) = \mathrm{L}(y)\,\# = \bigcup_{n=0}^{\infty} \bigotimes^{n} \mathrm{L}(y) = \bigcup_{n=0}^{\infty} \{\, w \mid w \in u^1 \otimes \ldots \otimes u^n,\, u^1, \ldots, u^n \in \mathrm{L}(y) \,\}.$$

Hierbei ist $u \otimes v$ definiert als die Menge aller Worte w, die durch „Mischen“ oder *Überlagern* der beiden Worte u und v entstehen:

$$u \otimes v = \{\, u^1 v^1 \ldots u^n v^n \mid n \in \mathbb{N},\, u^1, v^1, \ldots, u^n, v^n \in \Sigma^*,\, u^1 \ldots u^n = u,\, v^1 \ldots v^n = v \,\}.$$

Beachte wiederum, daß $u^1, v^1, \ldots$ nicht einzelne Aktionen, sondern Teilworte von u bzw. v sind, die ggf. auch leer sein können.

$u^1 \otimes \ldots \otimes u^n$ steht entsprechend für die Überlagerung der n Worte $u^1, \ldots, u^n$ und kann für $n > 2$ induktiv wie folgt definiert werden:

$$u^1 \otimes \ldots \otimes u^n = (u^1 \otimes \ldots \otimes u^{n-1}) \otimes \{\, u^n \,\}.$$

Für die beiden Grenzfälle $n = 0$ und $n = 1$ ist $u^1 \otimes \ldots \otimes u^n$ als $\{\, \text{‘’} \,\}$ bzw. $\{\, u^1 \,\}$ definiert.

Abgeleitete Operatoren

Die verbleibenden Operatoren, nämlich Synchronisation und Multiplikatoren, lassen sich gemäß ihrer Definition auf bereits definierte Operatoren zurückführen. Für die Booleschen Multiplikatoren $\big|$ und $\&$ erhält man z. B.:

$$\mathrm{L}\left(\bigg|_{i=m}^{n} x_i \right) = \bigcup_{i=m}^{n} \mathrm{L}(x_i);$$

$$\mathrm{L}\left(\&_{i=m}^{n} x_i \right) = \bigcap_{i=m}^{n} \mathrm{L}(x_i).$$

Diese Definitionen lassen sich prinzipiell auf den Fall $n = \infty$ verallgemeinern:

$$\mathrm{L}\left(\bigg|_{i=m}^{\infty} x_i \right) = \bigcup_{i=m}^{\infty} \mathrm{L}(x_i);$$

$$\mathrm{L}\left(\&_{i=m}^{\infty} x_i \right) = \bigcap_{i=m}^{\infty} \mathrm{L}(x_i).$$

Damit hat man prinzipiell die Möglichkeit, auch die beiden Iterationen wie folgt auf „elementarere“ Operatoren zurückzuführen:

$$x^* = \bigg|_{n=0}^{\infty} \underline{\quad}_{i=1}^{n} x;$$

$$x\,\# = \bigg|_{n=0}^{\infty} +_{i=1}^{n} x.$$

Wie man leicht sieht, erhält man auf diese Weise dieselben Formeln für $\mathrm{L}(x^*)$ und $\mathrm{L}(x\,\#)$ wie bei den oben angegebenen „direkten“ Definitionen.

Beispiele

a) $\mathrm{L}(a + b) = \mathrm{L}(a) \otimes \mathrm{L}(b) = \{\, \text{‘}a\text{’} \,\} \otimes \{\, \text{‘}b\text{’} \,\} = \text{‘}a\text{’} \otimes \text{‘}b\text{’} =$
$\{\, u^1 v^1 \ldots u^n v^n \mid n \in \mathbb{N},\, u^1 \ldots u^n = \text{‘}a\text{’},\, v^1 \ldots v^n = \text{‘}b\text{’} \,\} = \{\, \text{‘}ab\text{’}, \text{‘}ba\text{’} \,\}.$

b) $L((a + b)\#) = \bigcup_{n=0}^{\infty} \{ w \mid w \in u^1 \otimes \ldots \otimes u^n, u^1, \ldots, u^n \in L(a + b) \} =$
$\bigcup_{n=0}^{\infty} \{ w \mid w \in u^1 \otimes \ldots \otimes u^n, u^1, \ldots, u^n \in \{ \text{‘}ab\text{’}, \text{‘}ba\text{’} \} \} = \{ w \mid w_a = w_b \},$
wobei w_a bzw. w_b die Anzahl der a’s bzw. b’s im Wort w bezeichne.

c) $L((a - b)\#) = \bigcup_{n=0}^{\infty} \{ w \mid w \in u^1 \otimes \ldots \otimes u^n, u^1, \ldots, u^n \in L(a - b) \} =$
$\bigcup_{n=0}^{\infty} \{ w \mid w \in u^1 \otimes \ldots \otimes u^n, u^1, \ldots, u^n \in \{ \text{‘}ab\text{’} \} \} = \{ w \mid w_a = w_b \wedge \forall\, i = 1, \ldots, |w|: w_a^i \geq w_b^i \},$
wobei w^i hier das Präfix der Länge i von w bezeichne.
Für die Worte w muß also gelten, daß jedes Präfix von w mindestens so viele a’s wie b’s enthält, d. h. daß es in w zu jedem b ein „zugehöriges“ vorausgehendes a geben muß.

2.4 Quantoren

Durch die Verallgemeinerung der endlichen auf unendliche Boolesche Multiplikatoren wurde bereits ein erster Schritt unternommen, um die Semantik von Quantoren zu definieren.

Um Quantoren nun auf unendliche Multiplikatoren zurückführen zu können, ist es zweckmäßig, eine Menge Π einzuführen, die *alle* denkbaren Werte π enthält, die als *aktuelle Parameter* von Aktionen auftreten können. Diese Menge läßt sich wie folgt aus der Menge Σ aller Aktionen ableiten:

$$\Pi = \bigcup_{a \in \Sigma} \pi(a),$$

wobei $\pi(a)$ für eine Aktion $a = a_0(a_1, \ldots, a_n)$ als $\{ a_1, \ldots, a_n \}$ definiert ist. Unter der Annahme, daß diese Menge (höchstens) *abzählbar* ist − was für praktische Anwendungen sicherlich realistisch ist −, lassen sich ihre Elemente π auch in einer bestimmten (willkürlichen) Reihenfolge $\pi_1, \pi_2, \ldots$ *anordnen*.

Unter Zuhilfenahme dieser Menge Π läßt sich nun der Quantor $\bigsqcup_p$ wie folgt definieren:

$$\bigsqcup_p x(p) = \bigsqcup_{i=1}^{\infty} x(\pi_i).$$

Schwierigkeiten bereitet jedoch der Quantor $\mathop{+}_p$. Die rein formale Umformung

$$\mathop{+}_p x(p) = \mathop{+}_{i=1}^{\infty} x(\pi_i)$$

stellt natürlich keine Definition dar, da die Bedeutung eines unendlichen Multiplikators $\mathop{+}_{i=1}^{\infty}$ bisher nicht definiert wurde. Gerade diese Definition erweist sich jedoch als schwierig.

Zum Vergleich betrachte man die Definition einer unendlichen Reihe $s = \sum_{i=1}^{\infty} s_i$, die intuitiv „einfach“ eine Summe mit unendlich vielen Gliedern darstellt, formal jedoch als *Grenzwert* einer Folge von *endlichen Summen* definiert werden muß:

$$s = \lim_{n \to \infty} \sum_{i=1}^{n} s_i.$$

Die Tatsache, daß dieser Grenzwert u. U. gar nicht existiert und der Wert der unendlichen Reihe damit undefinert ist, macht klar, daß „unendliche Ausdrücke“ nicht immer sinnvoll definiert werden können.

In Analogie zur Definition einer unendlichen Reihe werden unendliche Multiplikatorausdrücke und Quantoren später ebenfalls mit Hilfe von Grenzwerten definiert. Zuvor sollen jedoch einige Schwierigkeiten und Grenzen der bisher definierten Semantik aufgezeigt werden.

3. Schwierigkeiten und Grenzen des traditionellen Ansatzes

3.1 Vollständige versus unvollständige Worte

Beim traditionellen *Worterkennungsproblem* ist die Frage zu beantworten, ob ein gegebenes Wort w mit einer bestimmten Länge n in der Sprache $\mathrm{L}(x)$ enthalten ist, die durch einen gegebenen Ausdruck (oder eine Grammatik) x erzeugt wird. So muß beispielsweise ein Compiler für eine Programmiersprache (unter anderem) entscheiden, ob ein ihm vorgelegtes Programm korrekt ist oder nicht.

Im Kontext von Interaktionsausdrücken liegt jedoch typischerweise keine vollständige Folge von Aktionen vor, die *nachträglich* auf Korrektheit überprüft werden soll. Vielmehr werden Aktionen *sukzessive* ausgeführt, und es muß zu jedem Zeitpunkt die Frage beantwortet werden, welche Aktionen im *nächsten* Schritt zulässig sind. Da eine einmal begonnene Aktionsfolge prinzipiell jederzeit fortgesetzt werden kann, kann man konsequenterweise auch keine Aussage über ihre *Länge* machen. (Es gibt kein End of file wie bei einer Datei, das anzeigt, daß das Programm hier zu Ende ist.) Somit ist der traditionell in Automaten gebräuchliche Begriff des Endzustands in diesem Kontext bedeutungslos: Die Menge der Endzustände (oder akzeptierenden Zustände) ist entweder leer oder gleich der Menge aller Zustände.

Diese unterschiedliche Sichtweise sollte sich auch in der Semantik von Interaktionsausdrücken widerspiegeln. Beispielsweise sollte ein Ausdruck wie $(a - b)*$ nicht nur die Worte ‘’, ‘ab’, ‘$abab$’, …, sondern auch die Worte ‘a’, ‘aba’, … akzeptieren.

Dies läßt sich prinzipiell dadurch erreichen, daß man nach der Bestimmung der Menge $\mathrm{L}(x)$ für einen gegebenen Ausdruck x die Menge aller Präfixe

$$\mathrm{P}(x) = \{\, u \mid \exists\, v\colon uv \in \mathrm{L}(x) \,\}$$

bestimmt. Wenn dann zu einem bestimmten Zeitpunkt t bereits die Aktionen $a_1, \ldots, a_n$ ausgeführt worden sind, so ist eine weitere Aktion a_{n+1} zu diesem Zeitpunkt genau dann zulässig, wenn das Wort ‘$a_1 \ldots a_n a_{n+1}$’ in der Menge $\mathrm{P}(x)$ enthalten ist.

3.2 Unerfüllbare Teilausdrücke

Intuitiv − insbesondere wenn man mit der Theorie der formalen Sprachen nicht vertraut ist − würde man vermutlich erwarten, daß ein Ausdruck der Gestalt $x = a - y$ im ersten Schritt die Aktion a akzeptiert, *unabhängig* davon, wie der Teilausdruck y beschaffen ist. In den „meisten“ Fällen trifft dies auch zu, denn es gilt ja:

$$\mathrm{L}(x) = \mathrm{L}(a - y) = \mathrm{L}(a)\,\mathrm{L}(y) = \{\, \text{‘}a\text{’}w \mid w \in \mathrm{L}(y) \,\}.$$

Da alle Worte aus $\mathrm{L}(x)$ mit a beginnen, gilt offensichtlich ‘a’ $\in \mathrm{P}(x)$, d. h. a wird als erste Aktion akzeptiert − sofern es überhaupt Worte in $\mathrm{L}(x)$ gibt! Andernfalls, d. h. wenn $\mathrm{L}(x) = \varnothing$ ist, so ist auch $\mathrm{P}(x) = \{\, u \mid \exists\, v\colon uv \in \mathrm{L}(x) \,\} = \varnothing$, und der Ausdruck x akzeptiert − im Gegensatz zur gerade erwähnten „intuitiven Semantik“ − überhaupt keine Aktionen! Dieser Fall tritt hier genau dann ein, wenn $\mathrm{L}(y) = \varnothing$ ist, was z. B. für $y = b \,\&\, c$ oder auch $y = b - c \,@\, c - b$ zutrifft.

3.3 Quantoren

Ein ähnliches Problem ergibt sich im Zusammenhang mit Quantoren. Intuitiv sollte der Ausdruck $x = \mathop{+}_{p} a(p)$ beliebige Folgen von $a(p)$'s (mit paarweise verschiedenen Werten von p) akzeptieren. Da die Menge der möglichen Werte von p jedoch potentiell unendlich groß ist, ist intuitiv ebenso einleuchtend, daß es wohl keine *endliche* Folge von Aktionen gibt, die diesen Ausdruck *vollständig* erfüllt. Somit wäre auch hier $\mathrm{L}(x) = \mathrm{P}(x) = \varnothing$, und der Ausdruck würde nichts akzeptieren.

3.4 Ausweg

Das zuletzt genannte Problem ließe sich prinzipiell durch die Einführung *unendlicher* Worte lösen, das Dilemma im Zusammenhang unerfüllbarer Teilausdrücke jedoch nicht.

Beiden Problemen liegt jedoch eine gemeinsame Ursache zugrunde − die Präfixbildung wird unter Umständen „zu spät" durchgeführt: Wenn $\mathrm{L}(x) = \varnothing$ ist, dann ist auch $\mathrm{P}(x) = \varnothing$, obwohl der Ausdruck x intuitiv u. U. gewisse Teilworte akzeptieren sollte.

Andererseits darf die Präfixbildung auch nicht „zu früh" erfolgen. Würde man sie beispielsweise auf jeden Teilausdruck einzeln anwenden, d. h. letztlich die Definition $\mathrm{L}(a) = \{\, \text{‘}a\text{’} \,\}$ durch $\mathrm{L}(a) = \{\, \text{‘’}, \text{‘}a\text{’} \,\}$ ersetzen, so ergäbe sich z. B.

$$\mathrm{L}(a - b) = \{\, uv \mid u \in \mathrm{L}(a), v \in \mathrm{L}(b) \,\} = \{\, uv \mid u \in \{\, \text{‘’}, \text{‘}a\text{’} \,\}, v \in \{\, \text{‘’}, \text{‘}b\text{’} \,\} \,\} = \{\, \text{‘’}, \text{‘}b\text{’}, \text{‘}a\text{’}, \text{‘}ab\text{’} \,\},$$

was natürlich ebenfalls jeder Intuition widerspricht.

Der Ausweg aus diesem Dilemma besteht darin, für jeden Ausdruck x sowohl eine *Präfix-basierte* als auch eine auf *vollständigen Worten* basierende Definition zu geben und diese geeignet zu kombinieren. Dies ist Gegenstand des folgenden Abschnitts.

4. Teilwortorientierte Semantik

Wie bereits angedeutet, wird in diesem Abschnitt für jeden Ausdruck x sowohl eine Menge $\mathrm{C}(x)$ der von diesem Ausdruck akzeptierten *vollständigen Worte* (complete words) als auch eine Menge $\mathrm{P}(x)$ der von ihm akzeptierten *Teilworte* (partial words bzw. Präfixe) definiert. Durch eine geeignete „Verflechtung" der Definitionen wird sichergestellt, daß die Präfixbildung weder „zu früh" noch „zu spät" durchgeführt wird und die resultierende Semantik somit dem im Grundlagenbericht vorgestellten intuitiven Ausführungsmodell von Interaktionsausdrücken entspricht bzw. dieses präzisiert.

4.1 Basisoperatoren

Für einen Interaktionsausdruck x werden die Mengen $\mathrm{C}(x)$ und $\mathrm{P}(x)$ wie folgt definiert:

- Für einen leeren Ausdruck $x = \lambda$:
  $\mathrm{C}(x) = \{\, \text{‘’} \,\}$;
  $\mathrm{P}(x) = \{\, \text{‘’} \,\}$.
- Für einen atomaren Ausdruck $x = a$:
  $\mathrm{C}(x) = \{\, \text{‘}a\text{’} \,\}$;
  $\mathrm{P}(x) = \{\, \text{‘’}, \text{‘}a\text{’} \,\}$.

- Für eine Disjunktion $x = y \mid z$:
  $C(x) = C(y) \cup C(z) = \{ w \mid w \in C(y) \vee w \in C(z) \}$;
  $P(x) = P(y) \cup P(z) = \{ w \mid w \in P(y) \vee w \in P(z) \}$.
- Für eine Konjunktion $x = y \,\&\, z$:
  $C(x) = C(y) \cap C(z) = \{ w \mid w \in C(y), w \in C(z) \}$;
  $P(x) = P(y) \cap P(z) = \{ w \mid w \in P(y), w \in P(z) \}$.
- Für eine sequentielle Komposition $x = y - z$:
  $C(x) = C(y)\,C(z) = \{ uv \mid u \in C(y), v \in C(z) \}$;
  $P(x) = P(y) \cup C(y)\,P(z) = P(y) \cup \{ uv \mid u \in C(y), v \in P(z) \}$.
- Für eine sequentielle Iteration $x = y*$:
  $C(x) = C(y)* = \bigcup_{n=0}^{\infty} C(y)^n = \bigcup_{n=0}^{\infty} \{ u^1 \ldots u^n \mid u^1, \ldots, u^n \in C(y) \}$;
  $P(x) = C(y)*\,P(y) = \bigcup_{n=0}^{\infty} C(y)^n\,P(y) = \bigcup_{n=0}^{\infty} \{ u^1 \ldots u^{n+1} \mid u^1, \ldots, u^n \in C(y), u^n \in P(y) \}$.
- Für eine parallele Komposition $x = y + z$:
  $C(x) = C(y) \otimes C(z) = \{ w \mid w \in u \otimes v, u \in C(y), v \in C(z) \}$;
  $P(x) = P(y) \otimes P(z) = \{ w \mid w \in u \otimes v, u \in P(y), v \in P(z) \}$.
- Für eine parallele Iteration $x = y\#$:
  $C(x) = C(y)\# = \bigcup_{n=0}^{\infty} \bigotimes^{n} C(y) = \bigcup_{n=0}^{\infty} \{ w \mid w \in u^1 \otimes \ldots \otimes u^n, u^1, \ldots, u^n \in C(y) \}$;
  $P(x) = P(y)\# = \bigcup_{n=0}^{\infty} \bigotimes^{n} P(y) = \bigcup_{n=0}^{\infty} \{ w \mid w \in u^1 \otimes \ldots \otimes u^n, u^1, \ldots, u^n \in P(y) \}$.

Man beachte, daß die Definitionen für $C(x)$ und $P(x)$ weitgehend „unabhängig“ voneinander sind; lediglich bei den sequentiellen Operatoren tritt (erwartungsgemäß) ein Querbezug auf.

Beachte außerdem, daß die Definitionen für $C(x)$ mit den alten Definitionen für $L(x)$ übereinstimmen.

Beispiele

a) $P(a - b) = P(a) \cup \{ uv \mid u \in C(a), v \in P(b) \} = \{ \text{‘’}, \text{‘}a\text{’} \} \cup \{ uv \mid u \in \{ \text{‘}a\text{’} \}, v \in \{ \text{‘’}, \text{‘}b\text{’} \} \} =$
$\{ \text{‘’}, \text{‘}a\text{’} \} \cup \{ \text{‘}a\text{’}, \text{‘}ab\text{’} \} = \{ \text{‘’}, \text{‘}a\text{’}, \text{‘}ab\text{’} \}$.

b) $P(a + b) = P(a) \otimes P(b) = \{ \text{‘’}, \text{‘}a\text{’} \} \otimes \{ \text{‘’}, \text{‘}b\text{’} \} =$
$(\text{‘’} \otimes \text{‘’}) \cup (\text{‘’} \otimes \text{‘}b\text{’}) \cup (\text{‘}a\text{’} \otimes \text{‘’}) \cup (\text{‘}a\text{’} \otimes \text{‘}b\text{’}) = \{ \text{‘’} \} \cup \{ \text{‘}b\text{’} \} \cup \{ \text{‘}a\text{’} \} \cup \{ \text{‘}ab\text{’}, \text{‘}ba\text{’} \} =$
$\{ \text{‘’}, \text{‘}a\text{’}, \text{‘}b\text{’}, \text{‘}ab\text{’}, \text{‘}ba\text{’} \}$.

c) $P((a - b)*) = \bigcup_{n=0}^{\infty} \{ u^1 \ldots u^{n+1} \mid u^1, \ldots, u^n \in C(a - b), u^{n+1} \in P(a - b) \} =$
$\bigcup_{n=0}^{\infty} \{ u^1 \ldots u^{n+1} \mid u^1, \ldots, u^n \in \{ \text{‘}ab\text{’} \}, u^{n+1} \in \{ \text{‘’}, \text{‘}a\text{’}, \text{‘}ab\text{’} \} \} =$
$\{ \text{‘’}, \text{‘}a\text{’}, \text{‘}ab\text{’}, \text{‘}aba\text{’}, \text{‘}abab\text{’}, \ldots \}$.

d) $P((a - b)\#) = \bigcup_{n=0}^{\infty} \{ w \mid w \in u^1 \otimes \ldots \otimes u^n, u^1, \ldots, u^n \in P(a - b) \} =$
$\bigcup_{n=0}^{\infty} \{ w \mid w \in u^1 \otimes \ldots \otimes u^n, u^1, \ldots, u^n \in \{ \text{‘’}, \text{‘}a\text{’}, \text{‘}ab\text{’} \} \} = \{ w \mid \forall\, i = 1, \ldots, |w| \colon w_a^i \geq w_b^i \}$.

e) $C(b \,\&\, c) = C(b) \cap C(c) = \{ \text{‘}b\text{’} \} \cap \{ \text{‘}c\text{’} \} = \varnothing$;

$P(b \,\&\, c) = P(b) \cap P(c) = \{ \text{‘’}, \text{‘}b\text{’} \} \cap \{ \text{‘’}, \text{‘}c\text{’} \} = \{ \text{‘’} \}$.

f) $C(a - (b \,\&\, c)) = \{ uv \mid u \in C(a), v \in C(b \,\&\, c) \} = \{ uv \mid u \in \{ \text{‘}a\text{’} \}, v \in \varnothing \} = \varnothing$;

$P(a - (b \,\&\, c)) = P(a) \cup \{ uv \mid u \in C(a), v \in P(b \,\&\, c) \} = P(a) \cup \{ uv \mid u \in \{ \text{‘}a\text{’} \}, v \in \{ \text{‘’} \} \} = \{ \text{‘’}, \text{‘}a\text{’} \} \cup \{ \text{‘}a\text{’} \} = \{ \text{‘’}, \text{‘}a\text{’} \} = P(a)$.

4.2 Monotonieregeln

Aus den genannten Definitionen folgen unmittelbar die folgenden „Monotonieregeln“ für beliebige Ausdrücke x, y und z:

a) $\text{‘’} \in P(x)$;

b) $P(y - z) \supset P(y)$;

c) $C(y - z) \supset C(y) \Leftrightarrow \text{‘’} \in C(z)$;

d) $P(y + z) \supset P(y)$;

e) $C(y + z) \supset C(y) \Leftrightarrow \text{‘’} \in C(z)$.

Beweis

a) Rekursiv durch Anwenden der Definitionen.

b) Unmittelbar aus der Definition von $P(y - z)$.

c) Falls $\text{‘’} \in C(z)$, so folgt aus der Definition von $C(y - z)$:
$C(y - z) \supset \{ uv \mid u \in C(y), v = \text{‘’} \} = \{ u \mid u \in C(y) \} = C(y)$.
Falls $\text{‘’} \notin C(z)$, wähle ein Wort $u \in C(y)$ mit *minimaler* Länge l. Aus der Definition von $C(y - z)$ folgt zusammen mit $\text{‘’} \notin C(z)$, daß alle Worte $w \in C(y - z)$ eine Länge *größer* als l haben. Da u nur die Länge l hat, muß somit $u \notin C(y - z)$ gelten, woraus die Behauptung $C(y - z) \not\supset C(y)$ folgt.

d) Wegen (a) gilt $\text{‘’} \in P(z)$, und somit folgt ähnlich wie bei (c):
$P(y + z) \supset \{ w \mid w \in u \otimes v, u \in P(y), v = \text{‘’} \} = \{ w \mid w \in \{ u \}, u \in P(y) \} = P(y)$.

e) Analog zu (c).

4.3 Multiplikatoren

Endliche Multiplikatoren

Endliche Multiplikatoren werden gemäß ihrer Definition auf Basisoperatoren zurückgeführt:

$\overset{n}{\sim} x = x \sim \ldots \sim x$ (n-mal x, $n \in \mathbb{N}$);

$\overset{n}{\sim} x = \lambda$ für $n \leq 0$;

$\underset{k=m}{\overset{n}{\sim}} x_k = x_m \sim \ldots \sim x_n$ ($m, n \in \mathbb{Z}$).

Hierbei steht ~ für einen der binären Basisoperatoren –, +, | oder &.

Unendliche Multiplikatoren

Die Semantik unendlicher Multiplikatoren wird durch *Grenzwerte von Mengenfolgen* definiert.

Eine Folge $(M_n)_{n=1}^{\infty}$ von Mengen M_n hat hierbei den Grenzwert

- $\lim\limits_{n\to\infty} M_n = \bigcup\limits_{n=n_0}^{\infty} M_n = \{\, v \mid \exists\, n \geq n_0 \colon v \in M_n \,\}$,
  falls die Folge (M_n) ab einem bestimmten $n_0 \in \mathbb{N}$ *monoton wachsend* ist, d. h. falls $M_{n+1} \supset M_n$ für alle $n \geq n_0$ gilt;
- $\lim\limits_{n\to\infty} M_n = \bigcap\limits_{n=n_0}^{\infty} M_n = \{\, v \mid \forall\, n \geq n_0 \colon v \in M_n \,\}$,
  falls die Folge (M_n) ab einem bestimmten $n_0 \in \mathbb{N}$ *monoton fallend* ist, d. h. falls $M_{n+1} \subset M_n$ für alle $n \geq n_0$ gilt;
- $\lim\limits_{n\to\infty} M_n = \varnothing$,
  falls die Folge (M_n) *nicht monoton* ist, d. h. falls es kein $n_0 \in \mathbb{N}$ gibt, ab dem die Folge (M_n) monoton wachsend oder monoton fallend ist.

Für einen Ausdruck $x = \mathop{\sim}\limits_{i=1}^{\infty} x_i$ mit $\sim \in \{\,|, \&, -, +\,\}$ werden die Mengen $\mathrm{C}(x)$ und $\mathrm{P}(x)$ „abstrakt“ wie folgt definiert:

$$\mathrm{C}(x) = \lim_{n\to\infty} \mathrm{C}(y_n),$$

$$\mathrm{P}(x) = \lim_{n\to\infty} \mathrm{P}(y_n),$$

mit $y_n = \mathop{\sim}\limits_{i=1}^{n} x_i$.

Im folgenden wird für die einzelnen Operatoren diskutiert, wie diese Grenzwerte konkret zu interpretieren sind.

Disjunktion

Für eine unendliche Disjunktion $x = \coprod\limits_{i=1}^{\infty} x_i$, d. h. $y_n = \coprod\limits_{i=1}^{n} x_i$ gilt:

$$\mathrm{C}(y_n) = \mathrm{C}\left(\coprod_{i=1}^{n} x_i\right) = \bigcup_{i=1}^{n} \mathrm{C}(x_i),$$

und somit für alle $n \geq 1$:

$$\mathrm{C}(y_{n+1}) = \mathrm{C}(y_n) \cup \mathrm{C}(x_{n+1}) \supset \mathrm{C}(y_n),$$

d. h. die Mengenfolge $(\mathrm{C}(y_n))$ ist ab $n_0 = 1$ monoton wachsend, woraus folgt:

$$\mathrm{C}(x) = \lim_{n\to\infty} \mathrm{C}(y_n) = \bigcup_{n=1}^{\infty} \mathrm{C}(y_n) = \bigcup_{n=1}^{\infty} \bigcup_{i=1}^{n} \mathrm{C}(x_i) = \bigcup_{i=1}^{\infty} \mathrm{C}(x_i).$$

Ganz analog ergibt sich auch:

$$\mathrm{P}(x) = \bigcup_{i=1}^{\infty} \mathrm{P}(x_i).$$

Konjunktion

Dual zur Disjunktion ergibt sich für eine unendliche Konjunktion $x = \mathop{\&}_{i=1}^{\infty} x_i$:

$$C(x) = \bigcap_{i=1}^{\infty} C(x_i);$$

$$P(x) = \bigcap_{i=1}^{\infty} P(x_i).$$

Kompositionen

Für eine unendliche sequentielle *oder* parallele Komposition $x = \mathop{\pm}_{i=1}^{\infty} x_i$, d. h. $y_n = \mathop{\pm}_{i=1}^{n} x_i$ folgt aus der Monotonieregel (b) bzw. (d) für alle $n \geq 1$:

$$P(y_{n+1}) = P(y_n \pm x_{n+1}) \supset P(y_n),$$

d. h. die Mengenfolge $(P(y_n))$ ist ab $n_0 = 1$ monoton wachsend. Somit gilt:

$$P(x) = \lim_{n\to\infty} P(y_n) = \bigcup_{n=1}^{\infty} P(y_n).$$

Falls es außerdem ein $n_0 \in \mathbb{N}$ gibt, so daß $C(x_i)$ für alle $i \geq n_0$ das leere Wort ‘’ enthält − d. h. wenn nur *endlich* viele $C(x_i)$ das leere Wort *nicht* enthalten −, so gelten die gleichen Überlegungen aufgrund der Monotonieregel (c) bzw. (e) auch für die Mengenfolge $(C(y_n))$ (für $n \geq n_0$), und es gilt:

$$C(x) = \lim_{n\to\infty} P(y_n) = \bigcup_{n=n_0}^{\infty} C(y_n).$$

Andernfalls, d. h. wenn *unendlich* viele $C(x_i)$ das leere Wort nicht enthalten, ist die Mengenfolge $(C(y_n))$ auf jeden Fall nicht monoton wachsend. Wenn sie (ab einem bestimmten n_0) monoton fallend wäre, wäre der Grenzwert $C(x)$ als unendlicher Durchschnitt $\bigcap_{n=n_0}^{\infty} C(y_n)$, andernfalls als leere Menge zu interpretieren.

Da jedoch *unendlich viele* Teilausdrücke x_i nur durch Worte *positiver* Länge erfüllbar sind, würde man intuitiv in *keinem* dieser beiden Fälle erwarten, daß der Gesamtausdruck x durch ein Wort endlicher Länge erfüllt werden kann. Das Resultat des unendlichen Durchschnitts sollte also ebenfalls gleich der leeren Menge sein, so daß sich die Frage erübrigt, unter welchen Umständen die Mengenfolge $(C(y_n))$ monoton fallend ist.

Zum Beweis dieser Vermutung betrachte man die folgende Funktion l, die einem beliebigen Ausdruck z die minimale Länge seiner vollständigen Worte zuordnet:

$$l(z) = \min \{ |w| \mid w \in C(z) \}.$$

Offensichtlich erfüllt l die beiden folgenden Aussagen:

$$l(z) \geq 1, \text{ wenn } \text{‘’} \notin C(z);$$

$$l(y \pm z) = l(y) + l(z).$$

Daraus folgt:

$$l(y_n) = l\left(\mathop{\pm}_{i=1}^{n} x_i\right) = \sum_{i=1}^{n} l(x_i) \to \infty \text{ für } n \to \infty,$$

da $l(x_i) \geq 1$ für unendlich viele i gilt. Wenn es nun ein Wort $u \in \bigcap_{n=n_0}^{\infty} C(y_n)$ gäbe, so würde für jedes $n \geq n_0$ gelten:

$u \in C(y_n) \Rightarrow l(y_n) = \min \{ |w| \mid w \in C(y_n) \} \leq |u|$,

d. h. $l(y_n)$ wäre durch $|u|$ beschränkt, was einen Widerspruch zu $l(y_n) \to \infty$ darstellt.

Zusammenfassend gilt für eine unendliche sequentielle oder parallele Komposition $x = \overset{\infty}{\underset{i=1}{\pm}} x_i$:

$C(x) = \bigcup_{n=n_0}^{\infty} C\left(\overset{n}{\underset{i=1}{\pm}} x_i \right)$, falls für alle $i \geq n_0$ gilt: ‘’ $\in C(x_i)$;

$C(x) = \varnothing$, falls kein derartiges $n_0 \in \mathbb{N}$ existiert;

$P(x) = \bigcup_{n=1}^{\infty} P\left(\overset{n}{\underset{i=1}{\pm}} x_i \right)$.

4.4 Quantoren

Nach der Definition unendlicher Multiplikatoren lassen sich nun Quantoren unter Zuhilfenahme der in Abschnitt 2.4 eingeführten Menge Π einfach wie folgt definieren:

$$\left| \underset{p}{} x(p) = \overset{\infty}{\underset{i=1}{\left|\right.}} x(\pi_i); \right.$$

$$\underset{p}{\&} x(p) = \overset{\infty}{\underset{i=1}{\&}} x(\pi_i);$$

$$\underset{p}{+} x(p) = \overset{\infty}{\underset{i=1}{+}} x(\pi_i).$$

Rein theoretisch ist diese Definition natürlich auch für die sequentielle Komposition anwendbar. Allerdings würde ihre Bedeutung dann von der *Anordnung* der Werte π_i abhängen, was für praktische Anwendungen natürlich sinnlos ist (vgl. auch Abschnitt 10.6 im Grundlagenbericht).

Der Quantor $\underset{p}{@}$ wird im Abschnitt 4.5 definiert.

Beispiele

a) $C\left(\left| \underset{p}{} a(p) \right. \right) = C\left(\overset{\infty}{\underset{i=1}{\left|\right.}} a(\pi_i) \right) = \bigcup_{i=1}^{\infty} C(a(\pi_i)) = \bigcup_{i=1}^{\infty} \{ \text{‘}a(\pi_i)\text{’} \} = \{ \text{‘}a(\pi_1)\text{’}, \text{‘}a(\pi_2)\text{’}, \ldots \}$;

$P\left(\left| \underset{p}{} a(p) \right. \right) = P\left(\overset{\infty}{\underset{i=1}{\left|\right.}} a(\pi_i) \right) = \bigcup_{i=1}^{\infty} P(a(\pi_i)) = \bigcup_{i=1}^{\infty} \{ \text{‘’}, \text{‘}a(\pi_i)\text{’} \} = \{ \text{‘’}, \text{‘}a(\pi_1)\text{’}, \text{‘}a(\pi_2)\text{’}, \ldots \}$.

b) $C\left(\underset{p}{+} a(p) \right) = C\left(\overset{\infty}{\underset{i=1}{+}} a(\pi_i) \right) = \varnothing$,
da kein $C(a(\pi_i))$ das leere Wort ‘’ enthält, d. h. unendliche viele $C(a(\pi_i))$ das leere Wort nicht enthalten;

$$P\left(\underset{p}{+} a(p) \right) = P\left(\overset{\infty}{\underset{i=1}{+}} a(\pi_i) \right) = \bigcup_{n=1}^{\infty} P\left(\overset{n}{\underset{i=1}{+}} a(\pi_i) \right) =$$

$$\bigcup_{n=1}^{\infty} \{ \text{‘}w_1 \ldots w_n\text{’} \mid \{ w_1, \ldots, w_n \} = \{ a(\pi_1), \ldots, a(\pi_n) \} \} = \{ a(\pi_1), a(\pi_2), \ldots \} *.$$

c) $\mathrm{C}\left(\underset{p}{+} a(p) - b\right) = \left\{ uv \mid u \in \mathrm{C}\left(\underset{p}{+} a(p)\right), v \in \mathrm{C}(b) \right\} = \{ uv \mid u \in \varnothing, v \in \{ \text{‘}b\text{’} \} \} = \varnothing;$

$$\mathrm{P}\left(\underset{p}{+} a(p) - b\right) = \mathrm{P}\left(\underset{p}{+} a(p)\right) \cup \left\{ uv \mid u \in \mathrm{C}\left(\underset{p}{+} a(p)\right), v \in \mathrm{P}(b) \right\} =$$

$$\mathrm{P}\left(\underset{p}{+} a(p)\right) \cup \{ uv \mid u \in \varnothing, v \in \{ \text{‘’}, \text{‘}b\text{’} \} \} = \mathrm{P}\left(\underset{p}{+} a(p)\right) \cup \varnothing = \mathrm{P}\left(\underset{p}{+} a(p)\right).$$

4.5 Synchronisation

Obwohl die Synchronisation aufgrund ihrer großen praktischen Bedeutung im Grundlagenbericht wie ein Basisoperator behandelt wird, ist sie aus theoretischer Sicht ein abgeleiteter Operator, der wie folgt auf Basisoperatoren zurückgeführt werden kann.

Alphabet eines Ausdrucks

Der Begriff des *Alphabets* eines Ausdrucks wurde im Grundlagenbericht informell als die Menge aller Aktionen dieses Ausdrucks definiert.

Formal läßt sich das Alphabet $\alpha(x)$ eines Ausdrucks x wie folgt rekursiv definieren:

- Für einen atomaren Ausdruck $x = a$:
  $\alpha(x) = \{ a \}$.
- Für einen Ausdruck der Gestalt $x = y*$ oder $x = y\#$:
  $\alpha(x) = \alpha(y)$.
- Für einen Ausdruck der Gestalt $x = y \sim z$ mit einem beliebigen binären Operator ~:
  $\alpha(x) = \alpha(y) \cup \alpha(z)$.
- Für einen Ausdruck der Gestalt $x = \underset{i=1}{\overset{\infty}{\sim}} x_i$:
  $\alpha(x) = \underset{i=1}{\overset{\infty}{\cup}} \alpha(x_i)$.

Binäre Synchronisation

Bereits im Grundlagenbericht wurde der Ausdruck $y \mathbin{@} z$ definiert als:

$$y \mathbin{@} z = (y + (z_1 \mid z_2 \mid \ldots)*) \mathbin{\&} (z + (y_1 \mid y_2 \mid \ldots)*),$$

wobei $\{ z_1, z_2, \ldots \} = \alpha(z) \setminus \alpha(y)$ und $\{ y_1, y_2, \ldots \} = \alpha(y) \setminus \alpha(z)$ gilt.

Endliche Multiplikatoren

Ein endlicher Multiplikator-Ausdruck $\overset{n}{@}\, x$ oder $\underset{k=m}{\overset{n}{@}}\, x(k)$ kann wie in Abschnitt 4.3 auf den binären Operator @ zurückgeführt werden.

Unendliche Multiplikatoren

Die Semantik einer unendlichen Synchronisation $x = \overset{\infty}{\underset{i=1}{@}} x_i$ wird schrittweise wie folgt definiert.

Für $i \in \mathbb{N}$ sei die Menge A_i definiert als das *Komplement* des Alphabets des Teilausdrucks x_i bzgl. des Alphabets des Gesamtausdrucks x, d. h.:

$$A_i = \alpha(x) \setminus \alpha(x_i).$$

Wie üblich bezeichne dann A_i^* die Menge aller Worte über der Menge A_i, d. h.:

$$A_i^* = \{ \text{‘}w_1 \ldots w_n\text{’} \mid w_1, \ldots, w_n \in A_i, n \in \mathbb{N}_0 \}.$$

Die Menge C_i bzw. P_i entstehe durch Überlagern dieser Worte mit den vom Teilausdruck x_i akzeptierten vollständigen bzw. teilweisen Worten:

$$C_i = \mathrm{C}(x_i) \otimes A_i^*;$$

$$P_i = \mathrm{P}(x_i) \otimes A_i^*.$$

C_i bzw. P_i enthält somit genau die (vollständigen bzw. teilweisen) Worte, die der Ausdruck x_i „im Kontext“ der Synchronisation x akzeptiert. (Ein Teilausdruck einer Synchronisation soll ja die Aktionen, die nur in anderen Teilausdrücken vorkommen, zu jedem Zeitpunkt akzeptieren.)

Nach diesen Vorbereitungen werden die Mengen $\mathrm{C}(x)$ und $\mathrm{P}(x)$ dann wie folgt definiert:

$$\mathrm{C}(x) = \bigcap_{i=1}^{\infty} C_i;$$

$$\mathrm{P}(x) = \bigcap_{i=1}^{\infty} P_i.$$

Quantoren

Der Quantor $\underset{p}{@}$ läßt sich nun analog zu den übrigen Quantoren definieren:

$$\underset{p}{@}\, x(p) = \overset{\infty}{\underset{i=1}{@}}\, x(\pi_i).$$

Beispiele

a) $x = a - b \;@\; a - c = a - b + c\,* \;\&\; a - c + b\,*;$

$\mathrm{C}(x) = \mathrm{C}(a - b + c\,*) \cap \mathrm{C}(a - c + b\,*) = (\mathrm{C}(a - b) \otimes \mathrm{C}(c)\,*) \cap (\mathrm{C}(a - c) \otimes \mathrm{C}(b)\,*) =$
$(\{ \text{‘}ab\text{’} \} \otimes \{ \text{‘}c\text{’}^n \mid n \in \mathbb{N}_0 \}) \cap (\{ \text{‘}ac\text{’} \} \otimes \{ \text{‘}b\text{’}^n \mid n \in \mathbb{N}_0 \}) =$
$\{ \text{‘}c\text{’}^{n_1}\text{‘}a\text{’}\text{‘}c\text{’}^{n_2}\text{‘}b\text{’}\text{‘}c\text{’}^{n_3} \mid n_1, n_2, n_3 \in \mathbb{N}_0 \} \cap \{ \text{‘}b\text{’}^{n_1}\text{‘}a\text{’}\text{‘}b\text{’}^{n_2}\text{‘}c\text{’}\text{‘}b\text{’}^{n_3} \mid n_1, n_2, n_3 \in \mathbb{N}_0 \} =$
$\{ \text{‘}abc\text{’}, \text{‘}acb\text{’} \}.$

b) $x = \underset{p}{@}\, x(p)$ mit $x(p) = a(p) - b;$
$x = \overset{\infty}{\underset{i=1}{@}}\, x(\pi_i) = \overset{\infty}{\underset{i=1}{@}}\, x_i$ mit $x_i = x(\pi_i) = a(\pi_i) - b;$

$\alpha(x_i) = \{ a(\pi_i), b \};$
$\alpha(x) = \alpha\left(\overset{\infty}{\underset{i=1}{@}}\, x_i \right) = \bigcup_{i=1}^{\infty} \alpha(x_i) = \bigcup_{i=1}^{\infty} \{ a(\pi_i), b \} = \{ a(\pi) \mid \pi \in \Pi \} \cup \{ b \};$
$A_i = \alpha(x) \setminus \alpha(x_i) = \{ a(\pi) \mid \pi \in \Pi, \pi \neq \pi_i \};$

$C_i = \mathrm{C}(x_i) \otimes A_i^* = \{ \text{‘}a(\pi_i)b\text{’} \} \otimes A_i^*;$

$P_i = \mathrm{P}(x_i) \otimes A_i^* = \{\ \text{‘’},\ \text{‘}a(\pi_i)\text{’},\ \text{‘}a(\pi_i)b\text{’}\ \} \otimes A_i^*$;

$\mathrm{C}(x) = \bigcap_{i=1}^{\infty} C_i = \varnothing$, denn:
Für jedes Wort $w \in C_i$ gilt, daß $a(\pi_i)$ in w vor b auftritt. Somit müßte in einem Wort $w \in \mathrm{C}(x) = \bigcap_{i=1}^{\infty} C_i$ *jedes* $a(\pi_i)$ vor b auftreten, was für endliche Worte nicht möglich ist.

$\mathrm{P}(x) = \bigcap_{i=1}^{\infty} P_i = \{\ \text{‘}a(p_1)\ldots a(p_n)\text{’} \mid p_1, \ldots, p_n \in \Pi$ paarweise verschieden, $n \in \mathbb{N}_0\ \}$.

5. Ausblick

Die in diesem Bericht vorgestellte Semantik dient natürlich einerseits dazu, die präzise Bedeutung eines einzelnen Ausdrucks festzulegen. Auf der anderen Seite kann sie dazu verwendet werden, Aussagen über die Gleichheit oder Äquivalenz zweier Ausdrücke zu machen:

> Zwei Ausdrücke x und y heißen genau dann *gleich* oder *äquivalent*, wenn $\mathrm{C}(x) = \mathrm{C}(y)$ und $\mathrm{P}(x) = \mathrm{P}(y)$ gilt.

Beispielsweise kann man zeigen, daß die Ausdrücke $x\#$, $x*\#$, $x\#*$ und $x\#\#$ alle äquivalent sind. Ebenso sind aber auch $+_p a(p)$ und $+_p a(p) - b$ äquivalent (vgl. Beispiele in Abschnitt 4.4), obwohl ihr Alphabet verschieden ist! Da das Alphabet eines Ausdrucks jedoch von Bedeutung sein kann, wenn er als Teilausdruck einer Synchronisation verwendet wird, definieren wir noch einen zweiten Gleichheitsbegriff:

> Zwei Ausdrücke x und y heißen genau dann *absolut gleich* oder *absolut äquivalent*, wenn $\mathrm{C}(x) = \mathrm{C}(y)$, $\mathrm{P}(x) = \mathrm{P}(y)$ und $\alpha(x) = \alpha(y)$ gilt.

Nur wenn zwei Teilausdrücke absolut äquivalent sind, kann man den einen gefahrlos durch den anderen ersetzen, ohne dadurch die Bedeutung des Gesamtausdrucks zu verändern.

Aussagen über die Äquivalenz von Ausdrücken können u. U. für ihre (effiziente) Implementierung nützlich sein. In Analogie zur Anfrageoptimierung in (relationalen) Datenbanksystemen könnte man auch für Interaktionsausdrücke versuchen, einen Algorithmus zu formulieren, der versucht, zu einem vorgegebenen Ausdruck einen möglichst effizient implementierbaren äquivalenten Ausdruck zu finden. Die aus dem DB-Bereich bekannten Probleme (insbesondere die Frage der sinnvollen Begrenzung des „Suchraums“) dürften sich jedoch übertragen, so daß diese Aufgabe sicherlich sehr schwierig zu lösen sein dürfte.

Äquivalenzregeln können aber auch dazu benutzt werden, den Aufwand für eine Implementierung zu reduzieren. So wird in der derzeit entwickelten Implementierung beispielsweise die parallele Iteration wie folgt auf einen Quantorausdruck zurückgeführt:

$$x\# = +_p (x \mid \lambda),$$

wobei p ein „anonymer“ Parameter ist, der im Ausdruck x nicht vorkommt.