9 Prozessalgebra

Die Prozessalgebra wurde aus der Automatentheorie entwickelt, um nebenläufige und reaktive Prozesse und Systeme beschreiben, modellieren und verifizieren zu können. Dabei wurden wie bei endlichen Automaten Zustände festgelegt und für Aktionen oder Folgen von Aktionen spezifiziert, welches der Nachfolgezustand ist. Die *elementare Prozessalgebra* entspricht der Modellierung eines einzigen Automaten. Im Unterschied zur traditionellen Automatentheorie wird aber eine algebraische Behandlung (vergleichbar zu den regulären Ausdrücken) und der Aspekt der Beobachtbarkeit und Verhaltensäquivalenz reaktiver Systeme betont. Nebenläufige und kommunizierende Systeme werden durch mehrere Automaten beschrieben, die mittels Rendezvous-Synchronisation gekoppelt werden.
Die Darstellung in diesem Kapitel stützt sich in erster Linie auf [Fok99] und teilweise auf [Mil99]. Weitere einschlägige Literaturquellen sind: [BW90], [BV95], [Mil89] und [Bae95].

9.1 Prozess-Algebra

In diesem Abschnitt werden Prozess-Terme und ihre Bedeutung definiert. Letzteres erfolgt durch spezielle Transitionssysteme wie in den folgenden Beispielen.

9.1.1 Prozess-Terme

Die syntaktische Darstellung von Prozessen geschieht durch *Prozess-Terme*. Die im folgenden definierte Menge BPA ist die Grundmenge des auf Seite 202 eingeführten gleichnamigen Kalküls (**b**asic **p**rocess **a**lgebra)[1].

Definition 9.1 *Die Menge der* elementaren Prozess-Terme *bzw.* Prozess-Ausdrücke BPA *wird aus* atomaren Aktionen $a \in A$ *sowie der* Auswahl *und* Hintereinanderausführung *gebildet:*

- Atomare Aktion*: Jedes $a \in A$ ist in BPA.*
- Auswahl*: Für alle $t_1, t_2 \in BPA$ ist $(t_1 + t_2) \in BPA$*
- Sequenz*: Für alle $t_1, t_2 \in BPA$ ist $(t_1 \cdot t_2) \in BPA$*
- *Nur nach diesen Regeln gebildete Terme liegen in* BPA.

Bindungsstärke: Der Sequenzoperator $\_ \cdot \_$ bindet stärker als der Auswahloperator $\_ + \_$. Durch diese Konvention können Klammern gespart werden. Der Sequenzoperator kann weggelassen werden, d.h. wir notieren kurz $(t_1 t_2)$ statt $(t_1 \cdot t_2)$.

[1] Genau genommen muss also die Menge BPA vom Kalkül BPA unterschieden werden.

9.1.2 Prozessgraphen

Wir verbinden im Folgenden mit jedem Term einen Prozess. Der Auswahl-Term $(t_1 + t_2)$ bedeutet dann: Es wird entweder t_1 oder t_2 ausgeführt. Die Sequenz $(t_1 \cdot t_2)$ bedeutet: Nach der ordnungsgemäßen Termination von t_1 wird t_2 ausgeführt.

Beispiel 9.2 Der BPA-Term $(((a + b) \cdot c) \cdot d)$ (oder kürzer $((a + b)c)d$) repräsentiert das Verhalten des linken Transitionssystems von Abbildung 9.1.

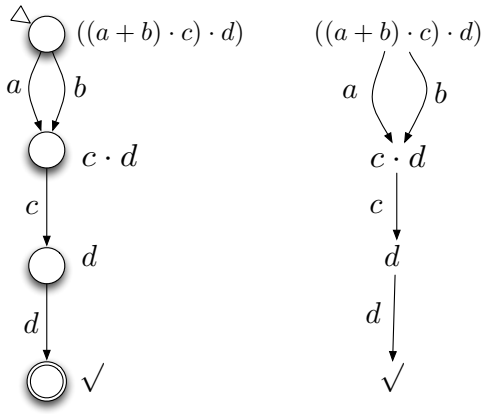


Abbildung 9.1: Prozessgraph von $((a + b)c)d$ als Transitionssystem

Als *Prozessgraphen* bezeichnen wir ein solches Transitionssystem, dessen Zustände Prozess-Terme sind. Ein solcher Prozess-Term repräsentiert das Verhalten des Transitionssystems mit ihm als Anfangszustand. Die Zustände eines Prozessgraphen beschreiben also das noch bevorstehende (Rest-)Verhalten, nachdem dieser Zustand erreicht wurde. Der Prozessgraph definiert eine *operationale Semantik* der Prozessalgebra.
Ein Prozessgraph wird formal über einen Kalkül definiert, wobei (A_0) das Axiom ist. Die Regeln definieren Transitionsübergänge zwischen Zuständen, die entweder Prozess-Terme oder das Symbol $\surd$ sind. Das Symbol $\surd$ kennzeichnet die korrekte Termination.

Definition 9.3 *Sei t ein Prozess-Term. Dann heißt das Transitionssystem $TS(t) = (S, A, tr, s_0, S^F)$* Prozessgraph von t, *wenn gilt:*

- *$S \subseteq BPA$ sind die mit dem* Transitionskalkül *aus t erreichbaren Prozess-Terme.*
- *A ist die Menge der atomaren Aktionen von t.*
- *tr die durch den* Transitionskalkül *eingeführte Transitionsrelation.*
- *$s_0 = t$ ist der einzige Anfangszustand.*
- *$S^F = \{\surd\}$ ist der einzige Endzustand.*

Abbildung 9.1 zeigt links einen Prozessgraphen als Transitionssystem mit Anfangs- und Endzustand. Prozessgraphen werden in den folgenden Beispielen wie in Abbildung 9.1

$$\frac{}{v \xrightarrow{v} \surd} \quad (A_0)$$

$$\frac{x \xrightarrow{v} \surd}{x + y \xrightarrow{v} \surd} \quad (T^{\surd}_{+R}) \qquad \frac{x \xrightarrow{v} x'}{x + y \xrightarrow{v} x'} \quad (T_{+R})$$

$$\frac{y \xrightarrow{v} \surd}{x + y \xrightarrow{v} \surd} \quad (T^{\surd}_{+L}) \qquad \frac{y \xrightarrow{v} y'}{x + y \xrightarrow{v} y'} \quad (T_{+L})$$

$$\frac{x \xrightarrow{v} \surd}{x \cdot y \xrightarrow{v} y} \quad (T^{\surd}_{\cdot}) \qquad \frac{x \xrightarrow{v} x'}{x \cdot y \xrightarrow{v} x' \cdot y} \quad (T_{\cdot})$$

Abbildung 9.2: BPA-Transitionsregeln ($v \in A, x, y, x'y' \in BPA$)

rechts dargestellt. Um die Graphiken übersichtlicher zu gestalten, wird oft der einzige Endzustand $\surd$ mehrfach gezeichnet. (Mehrfach gezeichnete Knoten bezeichnen aber immer nur ein einziges Objekt.)

Beispiel 9.4 Ableitung von $((a + b) \cdot c) \cdot d \xrightarrow{b} c \cdot d$ aus den Transitionsregeln:

$$\dfrac{\dfrac{\dfrac{\dfrac{b \xrightarrow{b} \surd}{a + b \xrightarrow{b} \surd} \qquad \dfrac{}{v \xrightarrow{v} \surd}}{(a + b) \cdot c \xrightarrow{b} c} \qquad \dfrac{y \xrightarrow{v} \surd}{x + y \xrightarrow{v} \surd}}{((a + b) \cdot c) \cdot d \xrightarrow{b} c \cdot d} \qquad \dfrac{x \xrightarrow{v} \surd}{x \cdot y \xrightarrow{v} y} \qquad \dfrac{x \xrightarrow{v} x'}{x \cdot y \xrightarrow{v} x' \cdot y}$$

(links: die Ableitung im Kalkül, rechts: die benutzten Regeln)

Die Operatoren $\cdot$ und $+$ werden in der offensichtlichen Weise auch auf Prozessgraphen erweitert. Wir betrachten also die alternative Komposition von Prozessgraphen $TS(s) + TS(t)$ und die sequentielle Komposition $TS(s) \cdot TS(t)$.

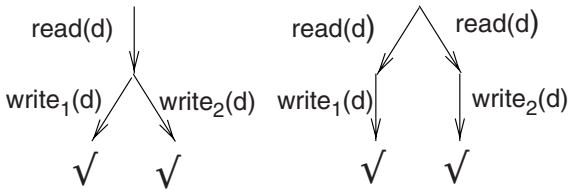
Aufgabe 9.5 Leiten Sie die Prozessgraphen der drei folgenden Prozessausdrücke mit dem Kalkül ab.

a) $((a + b)c + ac)d$
b) $(a(b + b))(c + c)$
c) $(b + a)(cd)$

9.1.3 Bisimulation und Äquivalenz

Um das Verhalten eines Systems mit dem Verhalten eines anderen Systems oder einer Spezifikation zu vergleichen, wurde der Begriff der wechselseitigen Simulation oder Bisimulation eingeführt.

Beispiel 9.6 Der linke Prozess liest d. Dann wird entschieden, ob d auf Platte 1 oder Platte 2 geschrieben wird. In dem anderen Prozess wird die Entscheidung vor dem Lesen getroffen.



Beide Prozesse haben die gleichen Schaltfolgen:

$$read(d)write_1(d) \qquad \text{und} \qquad read(d)write_2(d)$$

Die Prozesse sind daher „schaltfolgenäquivalent“ (trace equivalent).
Diese Art der Äquivalenz ist jedoch häufig nicht angemessen, z.B. wenn die Platte 1 ausfällt. Dann würde der erste Prozess d bei jedem Ablauf auf Platte 2 schreiben - im Gegensatz zum anderen Prozess, der in eine Verklemmung geraten kann.
Dies ist die Motivation, für einen auf „Bisimulation“ beruhenden Äquivalenzbegriff.

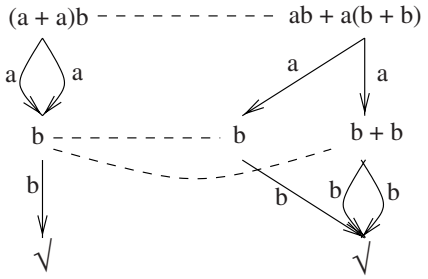
Definition 9.7 *Eine* Bisimulation *ist eine binäre Relation $\mathcal{B}$ auf BPA (d.h. $\mathcal{B} \subseteq BPA \times BPA$) mit folgenden Eigenschaften:*

1. *falls $p \mathcal{B} q$ und $p \xrightarrow{a} p'$, dann $q \xrightarrow{a} q'$ mit $p' \mathcal{B} q'$*
2. *falls $p \mathcal{B} q$ und $q \xrightarrow{a} q'$, dann $p \xrightarrow{a} p'$ mit $p' \mathcal{B} q'$*
3. *falls $p \mathcal{B} q$ und $p \xrightarrow{a} \surd$, dann $q \xrightarrow{a} \surd$*
4. *falls $p \mathcal{B} q$ und $q \xrightarrow{a} \surd$, dann $p \xrightarrow{a} \surd$*

Zwei Prozesse p und q heißen bisimilar *(Bezeichnung: $p \leftrightarrows q$), falls es eine Bisimulation $\mathcal{B}$ mit $p \mathcal{B} q$ gibt.*

Abweichend zu der Definition 2.4 wird in Definition 9.7 nicht der Anfangszustand in Relation gesetzt, so dass $\mathcal{B} = \emptyset$ stets eine mögliche Bisimulation ist. Da aber auch hier gilt, dass $(\mathcal{B} \cup \mathcal{B}')$ zweier Bisimulationen $\mathcal{B}$ und $\mathcal{B}'$ eine ist, können wir stets die maximale Bisimulation $\bigcup_{\mathcal{B} \text{ ist Bisimulation}} \mathcal{B}$ verwenden. Da dann zumindest $\surd \mathcal{B} \surd$ gilt, können wir den Fall $\mathcal{B} = \emptyset$ ausschließen.

Beispiel 9.8 Es gilt: $(a+a)b \leftrightarrow ab + a(b+b)$



Diese Bisimulation $\mathcal{B}$ wird durch folgende Paare definiert: $(a+a)b \; \mathcal{B} \; ab+a(b+b)$ sowie $b \; \mathcal{B} \; b$ und $b \; \mathcal{B} \; b+b$. In der Prozessalgebra ist es üblich, das immer geltende Paar $\surd \; \mathcal{B} \; \surd$ nicht explizit zu nennen.

Lemma 9.9 *Die Bisimulation ist eine Äquivalenzrelation.*

Aufgabe 9.10 Beweisen Sie die folgenden drei Bisimulationspaare mit Hilfe der Definition:

a) $((a+a)(b+b))(c+c) \; \mathcal{B} \; a(bc)$
b) $(a+a)(bc) + (ab)(c+c) \; \mathcal{B} \; (a(b+b))(c+c)$
c) $((a+b)c+ac)d \; \mathcal{B} \; (b+a)(cd)$

9.1.4 Bisimulation als Kongruenz

Dass die Bisimulation eine Äquivalenz ist, reicht uns aber noch nicht, wenn wir mit BPA kompositional argumentieren wollen: Angenommen wir haben ein System, dass wir durch die Auswahl $(s+t)$ beschreiben. Ersetzen wir das Subsystem s durch ein System s', von dem wir wissen, dass es sich genauso wie s verhält (d.h. es gilt $s \leftrightarrow s'$), dann soll sich am Verhalten des Gesamtsystems auch nichts verändern:

$$s \leftrightarrow s' \quad \text{impliziert} (s+t) \leftrightarrow (s'+t)$$

Dieses Konzept ist in der Algebra unter dem Begriff der *Kongruenz* bekannt.

Definition 9.11 *Eine Äquivalenzrelation $\simeq$ auf der Menge X heißt* Kongruenz*, wenn für alle $x, y, x', y' \in X$ und alle Operatoren f gilt:*

$$\textit{Wenn } x \simeq x' \textit{ und } y \simeq y', \textit{ dann auch } f(x,y) \simeq f(x',y').$$

Ganz allgemein wollen wir also, dass die Bisimulation eine Kongruenz bzgl. der BPA-Operatoren + und $\cdot$ sein soll:

$$\text{Wenn } s \leftrightarrow s' \text{ und } t \leftrightarrow t', \text{ dann auch } (s+t) \leftrightarrow (s'+t') \text{ und } (s \cdot t) \leftrightarrow (s' \cdot t').$$

Theorem 9.12 *Bisimulation ist eine Kongruenz auf BPA.*

Beweis: (Idee) Die Kongruenzeigenschaft folgt daraus, dass die Transitionsregeln in einer bestimmten Normalform (Panth-Form) sind. □

9.1.5 Der BPA-Kalkül

Mit oben stehender Definition ist es aufwendig zu überprüfen, ob zwei elementare Prozess-Terme bisimilar sind: es müssen zunächst die Prozessgraphen (die i.a. sehr groß werden) und dann zwischen ihnen die Bisimulationsrelation konstruiert werden.
Es wird daher jetzt ein Gleichungskalkül (**b**asic **p**rocess **a**lgebra) für eine Gleichheitsrelation zwischen elementaren Prozess-Termen eingeführt, der diese Aufgabe durch die Konstruktion von Ableitungen lösen soll.
In diesem Kalkül lässt sich fast wie gewohnt rechnen. Es ist allerdings zu beachten, dass die geltenden Axiome anders als die bekannter Algebren (wie z.B. Gruppen, Körper, Mengenalgebren) sind.

Axiome des BPA-Kalküls Für x, y, $z \in BPA$ gelte:

$$\begin{array}{lrcl} \text{A1} & x+y & = & y+x \\ \text{A2} & (x+y)+z & = & x+(y+z) \\ \text{A3} & x+x & = & x \\ \text{A4} & (x+y)\cdot z & = & x\cdot z+y\cdot z \\ \text{A5} & (x\cdot y)\cdot z & = & x\cdot(y\cdot z) \end{array}$$

Substitution Eine *Substitution* σ ist eine Abbildung der Variablen in Terme. Sie wird induktiv auf Terme erweitert, indem alle Variablen v des Terms durch $\sigma(v)$ ersetzt werden. Beispiel: für $\sigma(u) := (u+v)\cdot w, \sigma(v) := u+u, \sigma(w) := w$ gilt dann $\sigma(u\cdot v+u) = ((u+v)\cdot w)\cdot(u+u)+(u+v)\cdot w$.

Schlussregeln des BPA-Kalküls

- (Substitution) Für $s = t$ und eine Substitution σ gelte $\sigma(s) = \sigma(t)$.
- (Äquivalenz)
  - $t = t$ für alle $t \in BPA$
  - falls $s = t$, dann $t = s$
  - falls $s = t$ und $t = u$, dann $s = u$
- (Kontext)
  Falls $s = s'$ und $t = t'$, dann $s + t = s' + t'$ und $s \cdot t = s' \cdot t'$.

Um die Relation *bisimilar* mittels des BPA-Kalküls zu berechnen, muss natürlich bewiesen werden, dass zwei Terme genau dann bisimilar sind, wenn sie äquivalent sind. Traditionell wird diese Eigenschaft in zwei Teilen als *Korrektheit* und *Vollständigkeit* des BPA-Kalküls formuliert und bewiesen.

Definition 9.13 (Korrektheit (soundness), Vollständigkeit (completeness))
Das BPA-Kalkül heißt korrekt (sound)*, wenn für alle Terme* $s, t \in BPA$ *aus* $s = t$ *auch* $s \underline{\leftrightarrow} t$ *folgt.*
Das BPA-Kalkül heißt vollständig (complete)*, wenn für alle Terme* $s, t \in BPA$ *aus* $s \underline{\leftrightarrow} t$ *auch* $s = t$ *folgt.*

Satz 9.14 *Der BPA-Kalkül ist korrekt.*

Beweis: Es wird hier nur die Beweisstruktur dargestellt. Wie fast immer bei Beweisen für die Korrektheit eines Kalküls ist zu zeigen:

1.) Die Axiome sind korrekt.
2.) Die Regeln überführen korrekte Terme in korrekte Terme.

zu 1.): Erfolgt mit 2 a): Substitution

zu 2.):

a) Substitution: Es ist $\sigma(s) \underline{\leftrightarrow} \sigma(t)$ für jedes Axiom $s = t$ zu beweisen, wobei σ eine Substitution ist, die alle Variablen in s und t auf elementare Prozess-Terme abbildet.

   Dabei ist auf die Definition der Bisimulation zurückzugreifen. Dies wird hier nicht ausgeführt.

   Informell steht dahinter in Bezug auf die Axiome A1 ... A5 folgende Argumentation:

   - A1: Die Terme $s + t$ und $t + s$ stellen beide eine Auswahl zwischen s und t dar.
   - A2: Die Terme $(s + t) + u$ und $s + (t + u)$ stellen beide eine Auswahl zwischen s, t und u dar.
   - A3: Eine Auswahl zwischen t und t ist eine Wahl für t.
   - A4: Die Terme $(s + t) \cdot u$ und $s \cdot u + t \cdot u$ stellen beide eine Auswahl zwischen s und t dar, worauf u ausgeführt wird.
   - A5: Die Terme $(s \cdot t) \cdot u$ und $s \cdot (t \cdot u)$ stellen beide die Aktion s dar, gefolgt von t und dann von u.

b) Die Äquivalenzregeln gelten, da Bisimulation eine Äquivalenz ist.
c) Die Kontextregel gilt, da Bisimulation-Relation eine Kongruenz ist.

□

Aufgabe 9.15 Motivieren Sie, dass folgendes Distributivgesetz nicht gilt: $x \cdot (y + z) = x \cdot y + x \cdot z$.
Hinweis: Setzen Sie $x = read(d)$ usw. in obigem Beispiel.

Satz 9.16 *Der BPA-Kalkül ist vollständig.*

Beweis: Zu beweisen ist also, dass aus $s \underline{\leftrightarrow} t$ auch $s = t$ folgt. Zunächst wird der Beweis für die einfachere Relation $=_{\mathrm{AC}}$ bewiesen, d.h. $s \underline{\leftrightarrow} t \Rightarrow s =_{\mathrm{AC}} t$. Daraus wird dann die Behauptung des Satzes abgeleitet.
Per definitionem gelte $s =_{\mathrm{AC}} t$, wenn der Ausdruck nur mittels der Axiome $A1$ (Kommutativität) und $A2$ (Assoziativität) abgeleitet werden kann. Jede Äquivalenzklasse bezüglich dieser Relation wird durch einen Ausdruck aus „Summanden“ der Form $s_1 + \ldots + s_k$ dargestellt, wobei s_i entweder eine atomare Aktion a ist oder die Form $t_1 \cdot t_2$ hat.

Die übrigen Axiome werden in Ersetzungsregeln mit der Richtung „links nach rechts“ umgeformt:

$$\begin{array}{lrcl} \text{R1} & x+y & =_{\text{AC}} & y+x \\ \text{R2} & (x+y)+z & =_{\text{AC}} & x+(y+z) \\ \text{R3} & x+x & \to & x \\ \text{R4} & (x+y)\cdot z & \to & x\cdot z+y\cdot z \\ \text{R5} & (x\cdot y)\cdot z & \to & x\cdot(y\cdot z) \end{array}$$

Auf diese Weise erhalten wir ein Ersetzungskalkül, bei dem zwischen den Regelanwendungen Umformungen bzgl. der Relation $=_{\text{AC}}$ angewendet werden können (siehe Algorithmus 9.1).

Wir zeigen im Folgenden die Existenz von Normalformen:

Wendet man die Regeln dieses Kalküls immer wieder auf einen Prozess-Term $s \in BPA$ an, so gelangt man nach einer endlichen Zahl von Schritten zu einem Prozess-Term $t \in BPA$, der nicht weiter reduziert werden kann.

Man kann sogar zeigen, dass die Normalform trotz des Nicht-Determinismus eindeutig bestimmt ist. Dies folgt aus der Eigenschaft des Ersetzungskalküls *konfluent* zu sein – was aber hier nicht bewiesen wird.

Dass der Ersetzungskalkül *terminierend* ist, weist man mit einer *Gewichtsfunktion*

$$gew : BPA \mapsto \mathbb{N}$$

nach, die Eigenschaft hat, dass jede Reduktion das Gewicht echt verringert.

Wir definiert $gew : BPA \mapsto \mathbb{N}$ folgendermaßen ($v \in A, s, t \in BPA$):

$$\begin{array}{rcl} gew(v) & := & 2 \\ gew(s+t) & := & gew(s)+gew(t) \\ gew(s\cdot t) & := & gew(s)^2\cdot gew(t) \end{array}$$

Da $gew(s_1) > gew(s_2) > \ldots > gew(s_q) > \ldots$ für jede Reduktion $s_1, s_2, \ldots, s_q, \ldots$ gilt, kann es keine unendlichen Ableitungen geben.

Terme in Normalform haben die Struktur $t_1+\ldots+t_k$, wobei jedes t_i eine atomare Aktion $a \in A$ ist oder die Form $a \cdot s$ ($a \in A$, s in Normalform) hat. Durch Induktion über ihre Länge beweist man für Normalformen n und n':

$$n \underline{\leftrightarrow} n' \Rightarrow n =_{\text{AC}} n'$$

- Falls n einen Summanden der Form a enthält, dann gilt $n \xrightarrow{a} \surd$ und wegen $n \underline{\leftrightarrow} n'$ auch $n' \xrightarrow{a} \surd$. Also ist a auch in n' als Summand enthalten.
- Falls n einen Summanden der Form $a\cdot s$ enthält, dann gilt $n \xrightarrow{a} s$ und wegen $n \underline{\leftrightarrow} n'$ auch $n' \xrightarrow{a} t$ mit $s \underline{\leftrightarrow} t$. Also ist $a \cdot t$ in n' als Summand enthalten. Da s und t in Normalform, aber kleiner als n und n' sind, folgt durch Induktion $s =_{\text{AC}} t$.

Da somit n und n' dieselben Summanden haben, gilt $n =_{\text{AC}} n'$.

Um nun den Beweis von $s \underline{\leftrightarrow} t \Rightarrow s = t$ zu führen, sei $s \underline{\leftrightarrow} t$ angenommen.

s und t können durch das Ersetzungssystem zu Normalformen n und n' reduziert werden.
Dies könnte auch durch den Kalkül geschehen, d.h. es gilt: $s = n$ und $t = n'$.
Aus der Korrektheit des Kalküls folgt $s \underline{\leftrightarrow} n$ und $t \underline{\leftrightarrow} n'$, also insgesamt $n \underline{\leftrightarrow} n'$.
Für solche Normalformen wurde aber oben gezeigt: $n =_{\mathrm{AC}} n'$.
Damit ergibt sich insgesamt: $s = n =_{\mathrm{AC}} n' = t$, d.h. $s = t$. □

Anmerkung: Der Begriff *Normalform* beinhaltet, dass verschiedene Ableitungen auf die gleiche Form (modulo $=_{AC}$) führen. Dies ist insofern bemerkenswert, da die Konstruktion der Normalform i.a. nicht deterministisch abläuft.
Beispielsweise erlaubt der Term $t = ((a + a) + (b + b))c$ folgende Umformungen:

$$((a + a) + (b + b))c \xrightarrow{R3} (a + (b + b))c \xrightarrow{R3} (a + b)c \xrightarrow{R4} ac + bc$$

und

$$\begin{aligned} ((a + a) + (b + b))c &\xrightarrow{R4} (a + a)c + (b + b)c \\ &\xrightarrow{R4} (ac + ac) + (b + b)c \\ &\xrightarrow{R4} (ac + ac) + (bc + bc) \\ &\xrightarrow{R3} ac + (bc + bc) \\ &\xrightarrow{R3} ac + bc \end{aligned}$$

Die Ersetzungsfolgen sind hier sogar unterschiedlich lang.
Man kann jedoch für diesen Ersetzungskalkül zeigen, dass die Normalform (trotz des Nich-Determinismus) stets eindeutig bestimmt ist, man also von *der* Normalform sprechen kann.

Aus dem Beweis ergibt sich mit Algorithmus 9.1 ein Verfahren, mit dem man mit den Regeln R3, R4 und R5 von Seite 204 die Gültigkeit von $s \underline{\leftrightarrow} t$ bzw. $s = t$ entscheiden kann. Dieses Verfahren hat eine lineare Zeitkomplexität und ist sehr viel besser als eines, das auf der Definition der Bisimulation beruht.

Algorithmus 9.1 (Entscheiden von $s \underline{\leftrightarrow} t$ bzw. $s = t$)

Input - Zwei Prozess-Terme s und t.
Output - TRUE falls $s = t$; FALSE falls die Eigenschaft $s = t$ nicht erfüllt ist.

1. Wende die Regeln R3, R4 und R5 von Seite 204 solange wie möglich auf s an.
2. Nenne das Ergebnis n (n ist ein Prozess-Term in Normalform).
3. Wende die Regeln R3, R4 und R5 von Seite 204 solange wie möglich auf t an.
4. Nenne das Ergebnis n' (n' ist ein Prozess-Term in Normalform).
5. Falls $n =_{AC} n'$, gebe TRUE aus, sonst FALSE.
   (Dieser Schritt kann mit den Regeln R1 und R2 durchgeführt werden (wobei auf Termination zu achten ist) oder durch andere Verfahren der Textverarbeitung.)

Satz 9.17 *Der Algorithmus 9.1 entscheidet korrekt, ob zwei Prozess-Terme s und t aus BPA äquivalent sind.*

Beispiel 9.18

Zu entscheiden ist: $(a+a)(cd)+(bc)(d+d) \stackrel{?}{=} ((b+a)(c+c))d$

$s \equiv (\underline{a+a})(cd)+(bc)(d+d)$	$t \equiv ((b+a)(\underline{c+c}))d$
$\stackrel{\text{A3}}{\to}\ a(cd)+(bc)(\underline{d+d})$	$\stackrel{\text{A3}}{\to}\ \underline{((b+a)c)d}$
$\stackrel{\text{A3}}{\to}\ a(cd)+\underline{(bc)d}$	$\stackrel{\text{A5}}{\to}\ \underline{(b+a)(cd)}$
$\stackrel{\text{A5}}{\to}\ a(cd)+b(cd) \equiv n$	$\stackrel{\text{A4}}{\to}\ b(cd)+a(cd) \equiv n'$

Die beiden berechneten Normalformen n und n' sind äquivalent (modulo $=_{\text{AC}}$) und daher auch die Ausgangsterme s und t.

Aufgabe 9.19 Leiten Sie die folgenden drei Äquivalenzen mit dem Kalkül ab.

a) $((a+a)(b+b))(c+c) = a(bc)$

b) $(a+a)(bc)+(ab)(c+c) = (a(b+b))(c+c)$

c) $((a+b)c+ac)d = (b+a)(cd)$

9.2 Parallele und kommunizierende Prozesse

Durch den *Parallelоperator (merge)* $\|$ wird die parallele (besser: nebenläufige) Ausführung der beiden Prozesse dargestellt, die er als Argument hat.
Ein Term kann jetzt bspw. die Form $(a+b)\|(cd)$ haben.
In den folgenden Regeln sei $\{v,w\} \subseteq A$ und x, x', y, y' seien Prozess-Terme.

$$\frac{x \xrightarrow{v} \surd}{x\|y \xrightarrow{v} y} \qquad \frac{x \xrightarrow{v} x'}{x\|y \xrightarrow{v} x'\|y}$$

$$\frac{y \xrightarrow{v} \surd}{x\|y \xrightarrow{v} x} \qquad \frac{y \xrightarrow{v} y'}{x\|y \xrightarrow{v} x\|y'}$$

Zwei parallel ablaufende Prozesse kommunizieren mittels einer Kommunikationsfunktion.
Eine *Kommunikationsfunktion* $\gamma : A \times A \to A$ erzeugt für jedes Paar atomarer Aktionen a und b ihre Kommunikations-Aktion $\gamma(a,b)$.

Die Kommunikationsfunktion γ ist kommutativ und assoziativ:

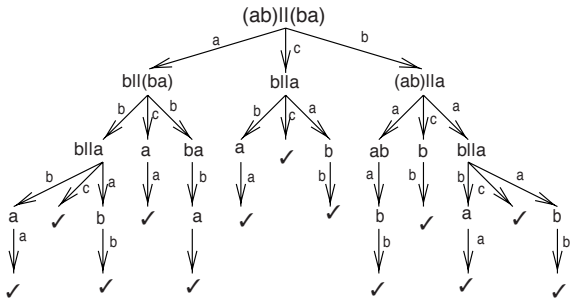
$$\begin{aligned} \gamma(a,b) &\equiv \gamma(b,a) \\ \gamma(\gamma(a,b),c) &\equiv \gamma(a,\gamma(b,c)) \end{aligned}$$

Der Parallelоperator kann eine solche Kommunikation enthalten. Sie ist eine unteilbare Aktion beider Prozesse.

$$\frac{x \xrightarrow{v} \surd \; y \xrightarrow{w} \surd}{x\|y \xrightarrow{\gamma(v,w)} \surd} \qquad \frac{x \xrightarrow{v} \surd \; y \xrightarrow{w} y'}{x\|y \xrightarrow{\gamma(v,w)} y'}$$

$$\frac{x \xrightarrow{v} x' \; y \xrightarrow{w} \surd}{x\|y \xrightarrow{\gamma(v,w)} x'} \qquad \frac{x \xrightarrow{v} x' \; y \xrightarrow{w} y'}{x\|y \xrightarrow{\gamma(v,w)} x'\|y'}$$

Beispiel 9.20 Der Prozessgraph von $(ab)\|(ba)$ mit $\gamma(x,y) = c$ für alle $x, y \in \{a,b\}$:



Links-Merge und Kommunikations-Merge .

Um eine Axiomatisierung mit dem Parralleloperator zu erhalten, werden zwei Hilfsoperatoren benötigt: *left-merge* x $\between$ y und *communication merge* $x|y$.
Der *Links-Merge-Operator (left merge)* $\between$ erlaubt die Ausführung der ersten Transition des ersten (linken) Argumentes:

$$\frac{x \xrightarrow{v} \surd}{x \between y \xrightarrow{v} y} \qquad \frac{x \xrightarrow{v} x'}{x \between y \xrightarrow{v} x' \| y}$$

Der *Kommunikations-Merge-Operator (communication merge)* | stellt die Kommunikation der beiden ersten Transitionen der beiden Argumente dar:

$$\frac{x \xrightarrow{v} \surd \; y \xrightarrow{w} \surd}{x|y \xrightarrow{\gamma(v,w)} \surd} \qquad \frac{x \xrightarrow{v} \surd \; y \xrightarrow{w} y'}{x|y \xrightarrow{\gamma(v,w)} y'}$$

$$\frac{x \xrightarrow{v} x' \; y \xrightarrow{w} \surd}{x|y \xrightarrow{\gamma(v,w)} x'} \qquad \frac{x \xrightarrow{v} x' \; y \xrightarrow{w} y'}{x|y \xrightarrow{\gamma(v,w)} x' \| y'}$$

Die Erweiterung der elementaren Prozessalgebra um die Operatoren *Parralleloperator (merge)* $\|$, *Links-Merge-Operator (left merge)* $\between$ und *Kommunikations-Merge-Operator (communication merge)* | heißt PAP (*Prozessalgebra mit Parallelismus*).
In ihr sollen die neuen Parralleloperatoren stärker binden als +, d.h.: $a \between b + a \| b$ steht für $(a \between b) + (a \| b)$. Der Parralleloperator $\|$ kann durch $\between$ und | ausgedrückt werden: $s \| t \leftrightarrows (s \between t + t \between s) + s|t$.

Anmerkung: PAP ist eine konservative Erweiterung von BPA, d.h. die neuen Transitionsregeln verändern nicht die alten. Anders ausgedrückt bedeutet dies, dass der auf BPA eingeschränkte Prozessgraph unverändert bleibt.

Satz 9.21 *Die Äquivalenzrelation Bisimulation ist eine Kongruenzrelation in PAP, d.h.: wenn* $s \leftrightarrows s'$ *und* $t \leftrightarrows t'$*, dann gilt:*

- $s + t \leftrightarrows s' + t'$,
- $s \cdot t \leftrightarrows s' \cdot t'$,
- $s \| t \leftrightarrows s' \| t'$,
- $s \between t \leftrightarrows s' \between t'$ *und*
- $s|t \leftrightarrows s'|t'$.

Axiome des PAP-Kalküls: Axiome A1, ..., A5 (Seite 202) und

$$
\begin{array}{lrcl}
\text{M1} & x\|y & = & (x \between y + y \between x) + x|y \\
 & & & \\
\text{LM2} & v \between y & = & v \cdot y \\
\text{LM3} & (v \cdot x) \between y & = & v \cdot (x\|y) \\
\text{LM4} & (x + y) \between z & = & x \between z + y \between z \\
 & & & \\
\text{CM5} & v|w & = & \gamma(v, w) \\
\text{CM6} & v|(w \cdot y) & = & \gamma(v, w) \cdot y \\
\text{CM7} & (v \cdot x)|w & = & \gamma(v, w) \cdot x \\
\text{CM8} & (v \cdot x)|(w \cdot y) & = & \gamma(v, w) \cdot (x\|y) \\
\text{CM9} & (x + y)|z & = & x|z + y|z \\
\text{CM10} & x|(y + z) & = & x|y + x|z
\end{array}
$$

Satz 9.22 *Der PAP-Kalkül ist korrekt, d.h.:* $s = t \;\Rightarrow\; s \leftrightarrows t$.

Beweisskizze: Da die Bisimulationsäquivalenz eine Kongruenz ist, genügt es für jedes Axiom $s = t$ die Relation $\sigma(s) \leftrightarrows \sigma(t)$ für alle Substitutionen von den Variablen aus s und t in Prozess-Terme zu beweisen.

Satz 9.23 *Der PAP-Kalkül ist vollständig, d.h.:* $s \leftrightarrows t \;\Rightarrow\; s = t$.

Beweisskizze: Dies kann man beweisen, indem man die Axiome für PAP in ein (Term-)Ersetzungskalkül modulo + verwandelt. Jeder Prozess-Term über PAP ist in Normalform reduzierbar. Wenn $s \leftrightarrows t$ ist, wobei s und t Normalformen s' und t' haben, dann gilt $s' =_{\mathrm{AC}} t'$, also auch $s = s' =_{\mathrm{AC}} t' = t$.

9.3 Abbruch und Unterdrücken

Abbruch (auch "deadlock") und Unterdrücken (auch „Verdecken", encapsulation) dienen dazu, Teile einer Kommunikation (wie $send(d)$ und das zugehörige $receive(d)$) zu einer Operation (z.B. $comm(d)$, vgl. Abb. 9.3) zu verschmelzen. Darüber hinaus können diese Operationen als Einzelaktionen unterbunden werden.

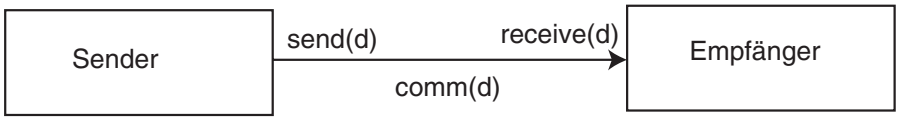


Abbildung 9.3: Kommunikation mit Sender und Empfänger

Der *Abbruchoperator* δ zeigt kein sichtbares Verhalten. Es gibt daher auch dazu keine Transitionsregel.

Die Operation *Unterdrücken* ∂_H , mit $H \subseteq A$, benennt alle Aktionen aus H, die bei ihm als Argument auftreten, in δ um:

$$\frac{x \xrightarrow{v} \surd \ (v \notin H)}{\partial_H(x) \xrightarrow{v} \surd} \qquad \frac{x \xrightarrow{v} x' \ (v \notin H)}{\partial_H(x) \xrightarrow{v} \partial_H(x')}$$

Der Bildbereich der Kommunikationsfunktion γ wird um δ erweitert:

$$\gamma : A \times A \to A \cup \{\delta\}.$$

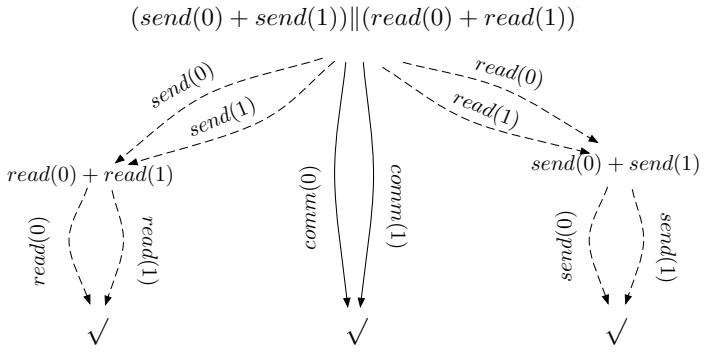
Das soll bedeuten: wenn a und b nicht kommunizieren, dann soll $\gamma(a, b) \equiv \delta$ gelten.
Die Operation Unterdrücken erzwingt Kommunikation. Beispielsweise kann $\partial_{\{a,b\}}(a \| b)$ nur als $\gamma(a, b)$ ausgeführt werden (falls $\gamma(a, b) \neq \delta$).

Definition 9.24 *Die Erweiterung des Kalküls PAP durch die nachstehenden Axiome für Abbruch und Verdeckung wird mit ACP bezeichnet (algebra of communicating processes).*

Axiome des Kalküls ACP ACP besteht aus den Axiomen von PAP und den folgenden für Abbruch und Unterdrücken:

$$
\begin{array}{lrcl}
\text{A6} & x + \delta & = & x \\
\text{A7} & \delta \cdot x & = & \delta \\
 & & & \\
\text{D1} & \partial_H(v) & = & v \ (v \notin H) \\
\text{D2} & \partial_H(v) & = & \delta \ (v \in H) \\
\text{D3} & \partial_H(\delta) & = & \delta \\
\text{D4} & \partial_H(x + y) & = & \partial_H(x) + \partial_H(y) \\
\text{D5} & \partial_H(x \cdot y) & = & \partial_H(x) \cdot \partial_H(y) \\
 & & & \\
\text{LM11} & \delta \between x & = & \delta \\
\text{CM12} & \delta | x & = & \delta \\
\text{CM13} & x | \delta & = & \delta
\end{array}
$$

Beispiel 9.25 Der Prozess-Term t zum einführenden Beispiel in Abb. 9.3 lautet:
$t \equiv \partial_{\{send(0),send(1),read(0),read(1)\}}((send(0) + send(1)) \| (read(0) + read(1)))$
mit $\gamma(send(d), read(d)) = comm(d)$ für $d \in \{0, 1\}$.
Die folgende Abbildung zeigt den zugehörigen Prozessgraphen, falls der Unterdrücken-Operator $\partial_{\{send(0),send(1),read(0),read(1)\}}$ weggelassen wird. Seine Hinzufügung bewirkt das Streichen der unterbrochenen Pfeile.



Theorem 9.26 *a) Bisimulation ist eine Kongruenz für ACP: wenn* $s \leftrightarrow s'$ *und* $t \leftrightarrow t'$*, dann* $s + t \leftrightarrow s' + t'$*,* $s \cdot t \leftrightarrow s' \cdot t'$*,* $s \| t \leftrightarrow s' \| t'$*,* $s \lfloor\!\lfloor t \leftrightarrow s' \lfloor\!\lfloor t'$*,* $s | t \leftrightarrow s' | t'$ *und* $\partial_H(s) \leftrightarrow \partial_H(s')$*.*

b) Der Kalkül ACP ist korrekt: $s = t \Rightarrow s \leftrightarrow t$*.*

c) Der Kalkül ACP ist vollständig: $s \leftrightarrow t \Rightarrow s = t$*.*

Beispiel 9.27 Seien $\gamma(a, b) \equiv c$ und $\gamma(a', b') \equiv c'$ zunächst die einzigen Kommunikationsaktionen zwischen Aktionen.

$$
\begin{array}{rl}
 & (a + a') \| (b + b') \\
\overset{\text{M1}}{=} & (a + a') \lfloor\!\lfloor (b + b') + (b + b') \lfloor\!\lfloor (a + a') + (a + a') | (b + b') \\
\overset{\text{LM4, CM9,10}}{=} & a \lfloor\!\lfloor (b + b') + a' \lfloor\!\lfloor (b + b') + b \lfloor\!\lfloor (a + a') + b' \lfloor\!\lfloor (a + a') + a|b + a|b' + a'|b + a'|b' \\
\overset{\text{LM2, CM5}}{=} & a \cdot (b + b') + a' \cdot (b + b') + b \cdot (a + a') + b' \cdot (a + a') + c + \delta + \delta + c' \\
\overset{\text{A6}}{=} & a \cdot (b + b') + a' \cdot (b + b') + b \cdot (a + a') + b' \cdot (a + a') + c + c'
\end{array}
$$

Sei nun $H = \{a, a', b, b'\}$.

$$
\begin{array}{rl}
 & \partial_H((a + a') \| (b + b')) \\
= & \partial_H(a \cdot (b + b') + a' \cdot (b + b') + b \cdot (a + a') + b' \cdot (a + a') + c + c') \\
\overset{\text{D1,2,4,5}}{=} & \delta \cdot \partial_H(b + b') + \delta \cdot \partial_H(b + b') + \delta \cdot \partial_H(a + a') + \delta \cdot \partial_H(a + a') + c + c' \\
\overset{\text{A6,7}}{=} & c + c'
\end{array}
$$

∂_H erzwingt also die Kommunikation zwischen a und b einerseits und zwischen a' und b' andererseits.

Aufgabe 9.28 Konstruieren Sie die Prozessgraphen zu folgenden Prozess-Termen:

a) $\partial_{\{a\}}(ac)$

b) $\partial_{\{a\}}((a + b)c)$

c) $\partial_{\{c\}}((a + b)c)$

d) $\partial_{\{a,b\}}((ab) \| (ba))$ mit $\gamma(a, b) = c$