9.2 Parallele und kommunizierende Prozesse

Durch den *Parallelloperator (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:

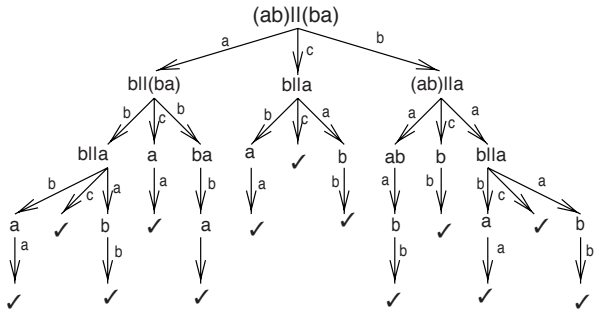
$$\begin{aligned} \gamma(a,b) &\equiv \gamma(b,a) \\ \gamma(\gamma(a,b),c) &\equiv \gamma(a,\gamma(b,c)) \end{aligned}$$

Der Parallelloperator 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 Paralleloperator 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 *Paralleloperator (merge)* $\|$, *Links-Merge-Operator (left merge)* $\between$ und *Kommunikations-Merge-Operator (communication merge)* | heißt PAP (*Prozessalgebra mit Parallelismus*).
In ihr sollen die neuen Paralleloperatoren stärker binden als $+$, d.h.: $a \between b + a \| b$ steht für $(a \between b) + (a \| b)$. Der Paralleloperator $\|$ 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

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

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.

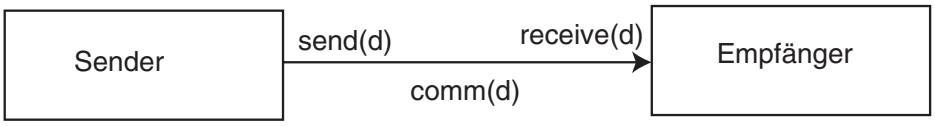


Abbildung 9.3: Kommunikation mit Sender und Empfänger

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:

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

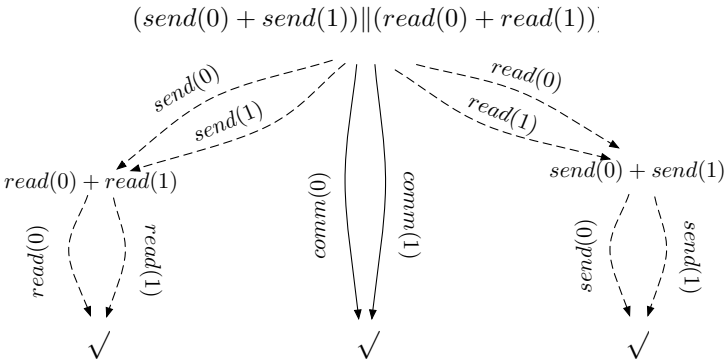
Das soll bedeuten: wenn a und b nicht kommunizieren, dann soll $\gamma(a, b) \equiv \delta$ gelten.
Die Operation Unterdrücken erzwingt Kommunikation. Beispielsweise kann $\partial_{\{a,b\}}(a \| b)$ nur als $\gamma(a, b)$ ausgeführt werden (falls $\gamma(a, b) \neq \delta$).

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* $s + t \leftrightarrow s' + t'$*,* $s \cdot t \leftrightarrow s' \cdot t'$*,* $s \| t \leftrightarrow s' \| t'$*,* $s \lfloor t \leftrightarrow s' \lfloor t'$*,* $s | t \leftrightarrow s' | t'$ *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.

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

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 $\gamma(a, b) = c$

9.4 Rekursion

Bislang wurden nur Prozesse endlicher Länge spezifiziert. Unendliche Prozesse, wie z.B. der Prozess *ababab*... mit dem Transitionssystem von Abb. 9.4 werden durch Rekursion definiert.

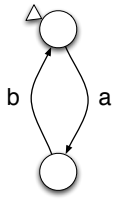


Abbildung 9.4: Transitionssystem für abab...

Konkret geschieht dies z.B. durch das folgende *rekursive Gleichungssystem*:

$$\begin{aligned} X &\stackrel{!}{=} aY \\ Y &\stackrel{!}{=} bX \end{aligned}$$

Dabei sind X und Y *Rekursionsvariablen*, die zwei Zustände des Prozesses repräsentieren.

Definition 9.29 *Eine* rekursive Spezifikation *für die Variablen $X_1, \ldots, X_n$ ist ein System von Gleichungen der Form:*

$$\begin{aligned} X_1 &\stackrel{!}{=} t_1(X_1, \ldots, X_n) \\ &\vdots \qquad\qquad \vdots \\ X_n &\stackrel{!}{=} t_n(X_1, \ldots, X_n) \end{aligned}$$

Hierbei sind die $t_i(X_1, \ldots, X_n)$ ACP-Terme ACP, welche die Variablen $X_1, \ldots, X_n$ enthalten dürfen.

Notationshinweis: Wir verwenden das Symbol $\stackrel{!}{=}$ in den Gleichungen $X_n \stackrel{!}{=} t_n(X_1, \ldots, X_n)$, um zu zeigen, dass es sich hierbei um etwas anderes handelt als um die Aussagen $s = t$ im Kalkül.

Wenn durch den Kontext eine Verwechslung ausgeschlossen ist, notieren wir aber $\stackrel{!}{=}$ auch als $=$.

Definition 9.30 *Prozesse $p_1, \ldots, p_n$ werden als eine* Lösung *einer rekursiven Spezifikation*

$$\{X_i \stackrel{!}{=} t_i(X_1, \ldots, X_n) \mid i \in \{1, \ldots, n\}\}$$

(modulo Bisimulations-Äquivalenz) bezeichnet, falls $p_i \leftrightarrows t_i(p_1, \ldots, p_n)$ für $i \in \{1, \ldots, n\}$ gilt.

Mit den Prozessen $p_1, \ldots, p_n$ sind hier ihre Prozessgraphen gemeint. Damit dann der Ausdruck $t_i(p_1, \ldots, p_n)$ Sinn ergibt, verstehen wir die Operationen $+$ und $\cdot$ auch als Operationen auf Prozessgraphen.
Wir wollen, dass Lösungen eindeutig bezüglich Bisimulations-Äquivalenz sind, d.h. falls $p_1, \ldots, p_n$ und $q_1, \ldots, q_n$ zwei solche Lösungen sind, dann erwarten wir $p_i \leftrightarrows q_i$ für alle $i \in \{1, \ldots, n\}$.
Dies muss aber nicht für jede Spezifikation der Fall sein.

Beispiel 9.31

a) $X \stackrel{!}{=} X$ hat alle Prozesse als Lösung.

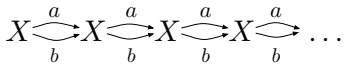
b) Alle Prozesse p mit $p \xrightarrow{a} \surd$ sind Lösungen von der rekursiven Spezifikation $\{X \stackrel{!}{=} a + X\}$.
Beispiel $p \equiv a + cd$: Die linke Seite ist nach Einsetzen ($X \equiv p \equiv a + cd$) bisimilar zur rechten Seite nach Einsetzen ($a + X \equiv a + a + X$).

c) Alle nichtterminierenden Prozesse sind Lösungen von der rekursiven Spezifikation $\{X \stackrel{!}{=} Xa\}$.

d) $\{X \stackrel{!}{=} aY, Y \stackrel{!}{=} bX\}$ hat als einzige Lösung den Prozess
$X \xrightarrow{a} Y \xrightarrow{b} X \xrightarrow{a} Y \xrightarrow{b} \ldots$ für X und
$Y \xrightarrow{b} X \xrightarrow{a} Y \xrightarrow{b} X \xrightarrow{a} \ldots$ für Y.

e) $\{X \stackrel{!}{=} Y, Y \stackrel{!}{=} aX\}$ hat als einzige Lösung $X \xrightarrow{a} X \xrightarrow{a} X \xrightarrow{a} X \xrightarrow{a} \ldots$ für X und Y.

f) $\{X \stackrel{!}{=} (a + b) \between X\}$ hat als einzige Lösung:



„Einzig“ bzw. „zwei verschiedene“ ist hier jeweils modulo Bisimulation zu verstehen.

Geschützte rekursive Spezifikation

Wir definieren jetzt einen Spezialfall rekursiver Spezifikation, um Existenz und Eindeutigkeit zu garantieren.

Definition 9.32 *Eine rekursive Spezifikation*

$$\{X_i \stackrel{!}{=} t_i(X_1, \ldots, X_n) \mid i \in \{1, \ldots, n\}\}$$

heißt geschützt (guarded)*, falls die rechten Seiten ihrer Gleichungen mit Hilfe der ACP-Kalküls in die folgende Form überführt werden können:*

$$a_1 \cdot s_1(X_1, \ldots, X_n) + \cdots + a_k \cdot s_k(X_1, \ldots, X_n) + b_1 + \cdots + b_\ell$$

Außerdem dürfen bei dieser Umformung Variable einer Rekursionsgleichung durch die entsprechende rechte Seite ersetzt werden.

Beispiel 9.33 Sei $\gamma(a,b) \equiv c$ und $\gamma(b,b) \equiv c$.
$\{X \stackrel{!}{=} Y \| Z,\ Y \stackrel{!}{=} Z + a,\ Z \stackrel{!}{=} bZ\}$ ist geschützt, denn:

- $Z \stackrel{!}{=} bZ$ hat schon die gewünschte Form.
- $Y \stackrel{!}{=} Z + a$ wird transformiert, indem Z durch bZ ersetzt wird.
- $X \stackrel{!}{=} Y \| Z$ wird transformiert, indem Y durch $bZ + a$ und Z durch bZ ersetzt wird und der so erhaltene Term $(bZ + a) \| bZ$ mittels der Axiome transformiert wird:

$$\begin{aligned} & (bZ + a) \| bZ \\ = \ & (bZ + a) \between bZ + bZ \between (bZ + a) + (bZ + a) | bZ \\ = \ & bZ \between bZ + a \between bZ + bZ \between (bZ + a) + bZ | bZ + a | bZ \\ = \ & b(Z \| bZ) + abZ + b(Z \| (bZ + a)) + c(Z \| Z) + cZ \end{aligned}$$

Satz 9.34 *Eine rekursive Spezifikation besitzt genau dann eine eindeutige Lösung (modulo Bisimulation), wenn sie geschützt ist.*

Zum Beispiel sind einige der rekursiven Spezifikationen von Beispiel 9.31 nicht geschützt. Wegen der Existenz und der Eindeutigkeit können wir von *der* Lösung sprechen.

Definition 9.35 *Für eine geschützte rekursive Spezifikation E über den Variablen $X_1, \ldots, X_n$ bezeichnet*

$$\langle X_i | E \rangle$$

die Lösung der Variablen $X_i, i = 1, \ldots, n$.

Transitionsregeln für Rekursion

$$\frac{t_i(\langle X_1|E\rangle, \ldots, \langle X_n|E\rangle) \stackrel{v}{\to} \surd}{\langle X_i|E\rangle \stackrel{v}{\to} \surd}$$

$$\frac{t_i(\langle X_1|E\rangle, \ldots, \langle X_n|E\rangle) \stackrel{v}{\to} y}{\langle X_i|E\rangle \stackrel{v}{\to} y}$$

d.h.: $\langle X_i|E\rangle$ übernimmt das Verhalten von $t_i(\langle X_1|E\rangle, \ldots, \langle X_n|E\rangle)$:

Beispiel 9.36
Sei E die geschützte rekursive Spezifikation $\{X = aY,\ Y = bX\}$. Der Prozessgraph von $\langle X|E\rangle$ ist:

Wir leiten her: $\langle X|E\rangle \xrightarrow{a} \langle Y|E\rangle$:

$$\frac{a \xrightarrow{a} \surd}{a \cdot \langle Y|E\rangle \xrightarrow{a} \langle Y|E\rangle} \qquad \frac{}{v \xrightarrow{v} \surd} \qquad v := a$$

$$\frac{x \xrightarrow{v} \surd}{x \cdot y \xrightarrow{v} y} \qquad v := a, \quad x := a, \quad y := \langle Y|E\rangle$$

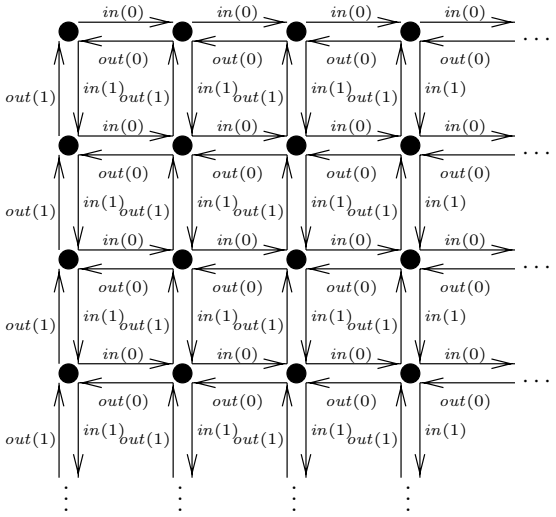
$$\frac{a \cdot \langle Y|E\rangle \xrightarrow{v} y}{\langle X|E\rangle \xrightarrow{v} y} \qquad v := a, \quad y := \langle Y|E\rangle$$

Aufgabe 9.37 Leiten Sie für die folgenden drei Prozess-Terme den Prozessgraphen ab:

a) $\langle X|X = ab\rangle$
b) $\langle X|X = aXb\rangle$
c) $\langle X|X = aXb + c\rangle$

Anmerkung: ACP mit geschützter Rekursion ist eine konservative Erweiterung von ACP und Bisimulation ist eine Kongruenzrelation.

Beispiel 9.38 Multimenge (bag) über $\{0, 1\}$.
Die Elemente 0 and 1 werden durch $in(0)$ und $in(1)$ in die Multimenge eingefügt und können durch $out(0)$ und $out(1)$ in beliebiger Reihenfolge entfernt werden.



Eine rekursive Spezifikation ist:

$$X \;=\; in(0)(X \| out(0)) + in(1)(X \| out(1))$$

Aufgabe 9.39 Geben Sie eine Bisimulations-Relation für Beispiel 9.38 an. Dabei ist das gezeichnete Transitionssystem mit dem Zustand links oben als Anfangszustand zu versehen. Die Bisimulation soll zwischen diesem Transitionssystem und dem Prozessgraphen einer Lösung $\langle X|E\rangle$ der rekursiven Spezifikation konstruiert werden.

Definition 9.40 *Sei E eine rekursive Spezifikation E der Form:*

$$\{X_i = t_i(X_1, \ldots, X_n) \mid i \in \{1, \ldots, n\}\}$$

Folgende Axiome gelten für ACP mit Rekursion:

- ***Rekursives Definitionsprinzip:***

$$\langle X_i|E\rangle \;=\; t_i(\langle X_1|E\rangle, \ldots, \langle X_n|E\rangle) \quad (i \in \{1, \ldots, n\}) \tag{RDP}$$

- ***Rekursives Spezifikationsprinzip:***

$$\begin{array}{l} \textit{Falls } y_i = t_i(y_1, \ldots, y_n) \textit{ für } i \in \{1, \ldots, n\}, \\ \textit{dann} \quad y_i \;=\; \langle X_i|E\rangle \qquad (i \in \{1, \ldots, n\}) \end{array} \tag{RSP}$$

Beispiel 9.41

$$\begin{array}{rcl} \langle Z \mid Z{=}aZ\rangle & \overset{\text{RDP}}{=} & a\langle Z \mid Z{=}aZ\rangle \\ & \overset{\text{RDP}}{=} & a(a\langle Z \mid Z{=}aZ\rangle) \\ & \overset{\text{A5}}{=} & (aa)\langle Z \mid Z{=}aZ\rangle \end{array}$$

Also gilt mit RSP (denn wir haben eine Gleichung mit Z der Form $x = aax$):

$$\langle Z \mid Z{=}aZ\rangle \;=\; \langle X \mid X{=}(aa)X\rangle$$

Weiter gilt:

$$\begin{array}{rcl} \langle Z \mid Z{=}aZ\rangle & \overset{\text{RDP}}{=} & a\langle Z \mid Z{=}aZ\rangle \\ & \overset{\text{RDP}}{=} & a(a\langle Z \mid Z{=}aZ\rangle) \\ & \overset{\text{RDP}}{=} & a(a(a\langle Z \mid Z{=}aZ\rangle)) \\ & \overset{\text{A5}}{=} & ((aa)a)\langle Z \mid Z{=}aZ\rangle \end{array}$$

Also gilt mit RSP (denn wir haben eine Gleichung mit Z der Form $y = aaay$):

$$\langle Z \mid Z{=}aZ\rangle \;=\; \langle Y \mid Y{=}((aa)a)Y\rangle$$

Also:

$$\begin{array}{rcl} \langle X \mid X{=}(aa)X\rangle & = & \langle Z \mid Z{=}aZ\rangle \\ & = & \langle Y \mid Y{=}((aa)a)Y\rangle \end{array}$$

Beispiel 9.42 Wir wollen für $\gamma(a,b) \equiv c$ die folgende Gleichung beweisen:

$$\partial_{\{a,b\}}(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) = \langle Z \mid Z{=}cZ\rangle$$

Ansatz: Es gilt:

$$\begin{array}{rl} & \langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle \\ = & \langle X \mid X{=}aX\rangle \between \langle Y \mid Y{=}bY\rangle \\ + & \langle Y \mid Y{=}bY\rangle \between \langle X \mid X{=}aX\rangle \\ + & \langle X \mid X{=}aX\rangle | \langle Y \mid Y{=}bY\rangle \\ = & a(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) \\ + & b(\langle Y \mid Y{=}bY\rangle \| \langle X \mid X{=}aX\rangle) \\ + & c(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) \end{array}$$

Aus

$$\begin{array}{rl} & \partial_{\{a,b\}}(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) \\ = & c \cdot \partial_{\{a,b\}}(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) \end{array}$$

folgt mit RSP das Ergebnis.

Theorem 9.43 *Der Kalkül ACP mit geschützter Rekursion ist korrekt bezüglich Bisimulation:*

$$s = t \;\Rightarrow\; s \leftrightarrows t$$

RSP ist **nicht** korrekt für ungeschützte Rekursion. Beispielsweise folgt mit RSP aus $t = t$, dass t das Muster der Spezifikation $X = X$ besitzt, d.h. dass t nach RSP eine Lösung ist:

$$t = \langle X \mid X{=}X\rangle$$

Da diese Gleichung für alle Prozess-Terme t gilt, erhalten wir also $t_1 = \langle X \mid X{=}X\rangle$ und $t_2 = \langle X \mid X{=}X\rangle$ und hieraus folgt direkt $t_1 = t_2$ und das für beliebige Terme t_1, t_2. Mit anderen Worten: Aus RSP würde für ungeschützte Rekursion folgen, dass alle Terme gleich sind, es also nur einen einzigen Term gibt.
Bemerkung: Der Kalkül ACP mit geschützter Rekursion ist jedoch *nicht* vollständig. Wir werden daher die Form der Rekursion weiter einschränken.

Definition 9.44 *Eine rekursive Spezifikation ist* linear *(in ACP) , wenn ihre rekursiven Gleichungen die folgende Form haben:*

$$X = a_1X_1 + \cdots + a_kX_k + b_1 + \cdots + b_l \qquad a_i, b_j \in A$$

Die linearen rekursiven Spezifikationen sind vollständig axiomatisierbar.

Theorem 9.45 (Vollständigkeit) *Das Kalkül ACP mit den Axiomen RDP, RSP und linearer rekursiver Spezifikation ist eine vollständige Axiomatisierung für Bisimulation.*

Umformungen in rekursiven Spezifikationen

Sei E, E' geschützte rekursive Spezifikationen, wobei E' aus E entsteht, indem die rechten Seiten der Gleichungen folgendermaßen modifiziert werden:

- Die Axiome für ACP mit geschützter Rekursion dürfen benutzt werden.
- Rekursionsvariablen dürfen durch die rechte Seiten ihrer Rekursionsgleichung ersetzt werden.

Dann kann $\langle X|E\rangle = \langle X|E'\rangle$ im Kalkül ACP mit geschützter Rekursion für alle Rekursionsvariablen abgeleitet werden.
Zum Beispiel kann

$$\langle X \mid X{=}aX{+}aX\rangle = \langle X \mid X{=}(aa)X\rangle$$

abgeleitet werden, indem

- erst A3 angewandt wird: $(x + x = x)$,
- dann X durch die rechte Seite aX ersetzt wird,
- und dann zuletzt A5 angewandt wird: $((xy)z = x(yz))$.

$$X{=}aX{+}aX \rightsquigarrow X{=}aX \rightsquigarrow X{=}a(aX) \rightsquigarrow X{=}(aa)X$$

9.5 Abstraktion

Wird ein spezifiziertes System implementiert, so führt es i.A. mehr Aktionen aus, als in der Spezifikation vorgegeben sind. Diese Aktionen sind zwar für den internen Ablauf wichtig, für den Benutzer oder Auftraggeber jedoch nicht relevant. Um dies zu beschreiben, werden *stille* (silent), *spontane* oder *interne* Aktionen eingeführt. Ihre Bezeichnung ist meist τ (Tau).
Der stille Systemschritt τ kann ohne Vorbedingung ausgeführt werden und terminiert dann erfolgreich:

$$\frac{}{\tau \xrightarrow{\tau} \surd}$$

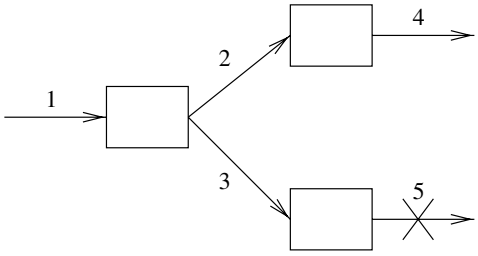
Der Kalkül ACP über A' mit τ-Transition und Abstraktionsoperator wird mit ACP_τ bezeichnet.
Die Transitionsregeln werden nun so erweitert, dass sie τ enthalten, d.h. die neue Menge von Aktionen ist $A' := A \cup \{\tau\}$ mit einer neuen Kommunikationsfunktion $\gamma : A' \times A' \to A \cup \{\delta\}$ derart, dass das Bild immer δ ist, falls im Argument ein τ steht, d.h. es findet keine Kommunikation mit der internen Aktion statt.
Der *Abstraktions-Operator* τ_I, mit $I \subseteq A$, benennt alle Aktionen aus I, die er als Argument führt, in τ um:

$$\frac{x \xrightarrow{v} \surd \ (v \notin I)}{\tau_I(x) \xrightarrow{v} \surd} \qquad \frac{x \xrightarrow{v} x' \ (v \notin I)}{\tau_I(x) \xrightarrow{v} \tau_I(x')}$$

$$\frac{x \xrightarrow{v} \surd \ (v \in I)}{\tau_I(x) \xrightarrow{\tau} \surd} \qquad \frac{x \xrightarrow{v} x' \ (v \in I)}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')}$$

Wenn in einer Folge abc von Aktionen b intern ist, dann ist $a\tau c$ (für die externe Spezifikation) äquivalent zu ac, d.h. stille Aktionen sind (extern) wirkungslos.
Wie das folgende Beispiel zeigt, kann nicht generell still gleich wirkungslos gesetzt werden.

Beispiel 9.46
Daten d werden über Kanal 1 empfangen ($r_1(d)$) und über Kanal 2 oder 3 weitergeleitet ($c_2(d), c_3(d)$). Im ersten Fall wird d über Kanal 4 weitergeleitet ($s_4(d)$), im zweiten Fall ist dies jedoch nicht möglich, da der Kanal 5 blockiert ist, d.h. $s_5(d)$ ist blockiert.



Dieses Verhalten wird durch folgenden Prozessausdruck beschrieben:

$$\partial_{\{s_5(d)\}}(r_1(d) \cdot (c_2(d) \cdot s_4(d) + c_3(d) \cdot s_5(d)))$$

$$= r_1(d) \cdot (c_2(d) \cdot s_4(d) + c_3(d) \cdot \delta)$$

(Die Umformung erfolgt mit den Regeln D1,D2,D4,D5 von Seite 211.)
Spezifiziert man $c_2(d)$ und $c_3(d)$ als interne Aktionen, so erhält man

$$r_1(d) \cdot (\tau \cdot s_4(d) + \tau \cdot \delta).$$

Werden beide stille Aktionen τ gestrichen, so erhält man mit $r_1(d) \cdot (s_4(d) + \delta)$ einen Prozess ohne Verklemmung. Er wird daher als nicht (Verhaltens-)äquivalent betrachtet.

Beispiel 9.47 Weitere nicht-äquivalente Prozessausdrücke:

- $a + \tau\delta$ und a
- $\partial_{\{b\}}(a + \tau b)$ und $\partial_{\{b\}}(a + b)$
- $a + \tau b$ und $a + b$

Wenn still nicht generell mit wirkungslos gleichgesetzt werden kann, dann stellt sich die Frage: Welche τ-Transitionen sind wirkungslos?
Antwort: Die τ-Transitionen, deren Streichung nicht das Verhalten ändert.
Zum Beispiel: $a + \tau(a + b)$ und $a + b$, denn nach der Ausführung von τ ist a immer noch ausführbar. Dies wird durch die folgende Definition der *Verzweigungs-Bisimulation* (branching bisimulation)formalisiert. (Hierbei bezeichne $\equiv$ die syntaktische Gleichheit.)

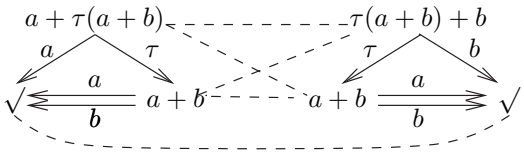
Definition 9.48 *Eine* Verzweigungs-Bisimulation *ist eine binäre Relation $\mathcal{B}$ auf Prozessen, für die mit $a \in A' = A \cup \{\tau\}$ gilt:*

1. *Wenn $p \mathcal{B} q$ und $p \xrightarrow{a} p'$, dann*
   - *entweder $a \equiv \tau$ und $p' \mathcal{B} q$*
   - *oder $q \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_0$ mit $p \mathcal{B} q_0$ und $q_0 \xrightarrow{a} q'$ mit $p' \mathcal{B} q'$*
2. *Wenn $p \mathcal{B} q$ und $q \xrightarrow{a} q'$, dann*
   - *entweder $a \equiv \tau$ und $p \mathcal{B} q'$*
   - *oder $p \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_0$ mit $p_0 \mathcal{B} q$ und $p_0 \xrightarrow{a} p'$ mit $p' \mathcal{B} q'$*
3. *Wenn $p \mathcal{B} q$ und p ist terminiert, dann $q \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_0$, so dass $p \mathcal{B} q_0$ und q_0 ist terminiert.*
4. *Wenn $p \mathcal{B} q$ und q ist terminiert, dann $p \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_0$, so dass $p_0 \mathcal{B} q$ und p_0 ist terminiert.*

Die τ-Folgen haben eine endliche Länge $n \geq 0$.
Zwei Prozesse p und q heißen verzweigungsbisimilar (branching bisimilar)*, in Zeichen $p \leftrightarrow_b q$, wenn es eine Verzweigungs-Bisimulations-Relation $\mathcal{B}$ gibt mit $p \mathcal{B} q$.*

Beispiel 9.49

$a + \tau(a + b) \underline{\leftrightarrow}_b \tau(a + b) + b$



Die Relation $\mathcal{B}$ ist definiert durch:

$a + \tau(a + b)\, \mathcal{B}\, \tau(a + b) + b$

$a + b\, \mathcal{B}\, \tau(a + b) + b$

$a + \tau(a + b)\, \mathcal{B}\, a + b$

$a + b\, \mathcal{B}\, a + b$

$\surd\, \mathcal{B}\, \surd$

Aufgabe 9.50

1. Geben Sie eine Verzweigungs-Bisimulations-Relation an, die zeigt dass $\tau(\tau(a + b) + b) + a$ und $a + b$ verzweigungsbisimilar sind.
2. Begründen Sie, warum $\tau a + \tau b$ und $a + b$ nicht verzweigungsbisimilar sind.

Die Verzweigungs-Bisimulations-Relation ist zwar eine Äquivalenzrelation, aber keine Kongruenz bezüglich BPA. Beispielsweise sind τa und a verzweigungsbisimilar, aber nicht $\tau a + b$ und $a + b$.

Ansatz: Einige Terme werden nicht mehr als äquivalent angesehen, indem wir die Bedingungen verschärfen.

Milner [Mil89] hat gezeigt, dass man dieses Problem dadurch lösen kann, dass eine Initialisierungsbedingung gefordert wird: Initiale τ-Transitionen werden nie eliminiert.

Dann wird in folgenden Fällen wird keine Äquivalenz erzeugt:

- $a + \tau\delta$ und a
- $\partial_{\{b\}}(a + \tau b)$ und $\partial_{\{b\}}(a + b)$
- $a + \tau b$ und $a + b$
- b und τb

Definition 9.51 *Eine* initiale Verzweigungs-Bisimulation (rooted branching bisimulation) *ist eine binäre Relation $\mathcal{B}$ auf Prozessen, für die mit $a \in A' = A \cup \{\tau\}$ gilt:*

1. *Wenn $p\, \mathcal{B}\, q$ und $p \xrightarrow{a} p'$, dann $q \xrightarrow{a} q'$ mit $p' \underline{\leftrightarrow}_b q'$.*
2. *Wenn $p\, \mathcal{B}\, q$ und $q \xrightarrow{a} q'$, dann $p \xrightarrow{a} p'$ mit $p' \underline{\leftrightarrow}_b q'$.*
3. *Wenn $p\, \mathcal{B}\, q$ und p terminiert, dann terminiert auch q.*
4. *Wenn $p\, \mathcal{B}\, q$ und q terminiert, dann terminiert auch p.*

Zwei Prozesse p und q heißen initial verzweigungsbisimilar (rooted branching bisimilar), *in Zeichen $p \underline{\leftrightarrow}_{rb} q$, wenn es eine initiale Verzweigungs-Bisimulations-Relation $\mathcal{B}$ gibt mit $p\, \mathcal{B}\, q$.*

Initiale Verzweigungs-Bisimulation ist (wie Verzweigungs-Bisimulation auch) eine Äquivalenzrelation. Das folgende Lemma zeigt, dass Initiale Verzweigungs-Bisimulation feiner als Verzweigungs-Bisimulation ist.

Lemma 9.52 *Zwischen den Äquivalenzen gelten die Beziehungen:*

$$\underline{\leftrightarrow} \quad \subseteq \quad \underline{\leftrightarrow}_{rb} \quad \subseteq \quad \underline{\leftrightarrow}_{b}$$

Falls τ in den Termen nicht vorkommt (wie z.B. in ACP), dann gilt sogar:

$$\underline{\leftrightarrow} \quad = \quad \underline{\leftrightarrow}_{rb} \quad = \quad \underline{\leftrightarrow}_{b}$$

Aufgabe 9.53 Bestimmen Sie für jedes der folgenden Paare von Prozess-Termen, ob es bisimilar, initial verzweigungsbisimilar oder verzweigungsbisimilar ist bzw. nicht ist - möglichst mit stichhaltiger Begründung:

a) $(a+b)(c+d)$ und $ac+ad+bc+bd$,
b) $(a+b)(c+d)$ und $(b+a)(d+c)+a(c+d)$,
c) $\tau(b+a)+\tau(a+b)$ und $a+b$,
d) $c(\tau(b+a)+\tau(a+b))$ und $c(a+b)$ sowie
e) $a(\tau b+c)$ und $a(b+\tau c)$.

9.5.1 Geschützte lineare Rekursion

Betrachten wir lineare Rekursion im Zusammenhang mit τ-Aktionen.
Alle Prozess-Terme τs stellen eine Lösung für die Spezifikation $X = \tau X$ dar, da $\tau s \underline{\leftrightarrow}_{rb} \tau\tau s$.
Die τ-Aktion reicht also nicht aus, um eine Eindeutigkeit zu garantieren. Wir müssen also den Begriff der geschützten Rekursion anpassen.

Definition 9.54 *1. Eine rekursive Spezifikation, in der τ-Transitionen erlaubt sind, ist* linear*, wenn ihre rekursiven Gleichungen die folgende Form haben:*

$$X \;=\; a_1X_1 + \cdots + a_kX_k + b_1 + \cdots + b_\ell \qquad\qquad (a_i, b_j \in A \cup \{\tau\})$$

2. Eine lineare rekursive Spezifikation E ist geschützt*, wenn es keine unendliche Folge von τ-Transitionen der folgenden Form gibt:*

$$\langle X|E\rangle \xrightarrow{\tau} \langle X'|E\rangle \xrightarrow{\tau} \langle X''|E\rangle \xrightarrow{\tau} \cdots$$

Die geschützten linearen rekursiven Spezifikationen sind genau diejenigen rekursiven Spezifikationen, die eine eindeutige Lösung modulo initialer Verzweigungs-Bisimulation haben. Wieder geben wir eine Axiomatisierung des erweiterten Kalküls ACP_τ an.
ACP_τ bezeichne jetzt den Kalkül ACP mit τ-Aktion, Abstraktionsoperator, geschützter linearer Rekursion und den folgenden Axiomen:

Axiome für Abstraktion (mit $v \in A' = A \cup \{\tau\}, I \subseteq A$)

B1	$v \cdot \tau$	$=$	v
B2	$v \cdot (\tau \cdot (x + y) + x)$	$=$	$v \cdot (x + y)$
TI1	$\tau_I(v)$	$=$	v $(v \notin I)$
TI2	$\tau_I(v)$	$=$	τ $(v \in I)$
TI3	$\tau_I(\delta)$	$=$	δ
TI4	$\tau_I(x + y)$	$=$	$\tau_I(x) + \tau_I(y)$
TI5	$\tau_I(x \cdot y)$	$=$	$\tau_I(x) \cdot \tau_I(y)$

Satz 9.55 *Initiale Verzweigungs-Bisimulation ist eine Kongruenzrelation für ACP_τ und geschützter linearer Rekursion.*

Theorem 9.56 *Der Kalkül ACP_τ mit Abstraktion und geschützter linearer Rekursion ist korrekt und vollständig in Bezug auf initiale Verzweigungsbisimulation.*

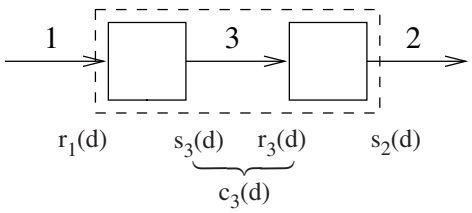
Aufgabe 9.57 Leiten sie $\tau_{\{b\}}(\langle X|X = aY, Y = bX\rangle) = \langle Z|Z = aZ\rangle$ in ACP_τ ab.

Beispiel 9.58 (Tandempuffer)
Konvention: Um Ausdrücke zu vereinfachen, wird in den folgenden Beispielrechnungen eine Lösung $\langle Q|Q = r_1 s_1 Q\rangle$ einer rekursiven Spezifikation abkürzend als Q geschrieben.
Betrachtet werden zwei in Serie angeordnete Puffer der Kapazität 1 die mit Kanälen 1, 2 und 3 verbunden sind. $r_i(d)$ bzw. $s_i(d)$ bezeichne das Schreiben bzw. Lesen vom Kanal i. Δ ist eine endliche Menge von Daten. Das Verhalten ist:

$$\begin{aligned} Q_1 &= \Sigma_{d\in\Delta} r_1\, s_3\, Q_1 \\ Q_2 &= \Sigma_{d\in\Delta} r_3\, s_2\, Q_2 \end{aligned}$$

wobei $d \in \Delta$ weggelassen wird, d.h. wir tun so, als ob Δ nur ein Element enthalten würde:

$$\begin{aligned} Q_1 &= r_1\, s_3\, Q_1 \\ Q_2 &= r_3\, s_2\, Q_2 \end{aligned}$$



Die Puffer Q_1 und Q_2 der Kapazität 1 arbeiten parallel und sind durch die Aktion $\gamma(s_3, r_3) = c_3$ synchronisiert:

$$\tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1))$$

Wir beweisen, dass das gemeinsame Verhalten dasjenige eines Puffers der Kapazität 2 ist:

$$\begin{aligned} X &= r_1 Y \\ Y &= r_1 Z + s_2 X \\ Z &= s_2 Y \end{aligned}$$

Dazu überlegen wir, was die Lösung dieses rekursiven Gleichungssystems E ist.
Als erstes berechnen wir:

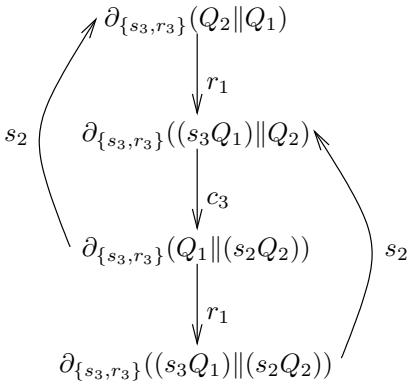
$$\begin{aligned} & Q_2 \| Q_1 \\ \overset{\text{M1}}{=} \; & Q_2 \between Q_1 + Q_1 \between Q_2 + Q_2 | Q_1 \\ \overset{\text{RDP}}{=} \; & (r_3 s_2 Q_2) \between Q_1 + (r_1 s_3 Q_1) \between Q_2 \\ & + (r_3 s_2 Q_2) | (r_1 s_3 Q_1) \\ \overset{\text{LM3,CM8}}{=} \; & r_3 \cdot ((s_2 Q_2) \| Q_1) + r_1 \cdot ((s_3 Q_1) \| Q_2) \\ & + \delta \cdot ((s_2 Q_2) \| (s_3 Q_1)) \\ \overset{\text{A7,A6}}{=} \; & r_3 \cdot ((s_2 Q_2) \| Q_1) + r_1 \cdot ((s_3 Q_1) \| Q_2) \end{aligned}$$

$$\begin{aligned} & \partial_{\{s_3,r_3\}}(Q_2 \| Q_1) \\ = \; & \partial_{\{s_3,r_3\}}(r_3 \cdot ((s_2 Q_2) \| Q_1) + r_1 \cdot ((s_3 Q_1) \| Q_2)) \\ = \; & \partial_{\{s_3,r_3\}}(r_3 \cdot ((s_2 Q_2) \| Q_1)) \\ & + \partial_{\{s_3,r_3\}}(r_1 \cdot ((s_3 Q_1) \| Q_2)) \\ = \; & \partial_{\{s_3,r_3\}}(r_3) \cdot \partial_{\{s_3,r_3\}}((s_2 Q_2) \| Q_1) \\ & + \partial_{\{s_3,r_3\}}(r_1) \cdot \partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2) \\ = \; & \delta \cdot \partial_{\{s_3,r_3\}}((s_2 Q_2) \| Q_1) \\ & + r_1 \cdot \partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2) \\ = \; & r_1 \cdot \partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2) \end{aligned}$$

Weiter rechnen wir:

$$\begin{aligned} \partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2) &= c_3 \cdot \partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2)) \\ \partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2)) &= r_1 \cdot \partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2)) \\ &\quad + s_2 \cdot \partial_{\{s_3,r_3\}}(Q_2 \| Q_1) \\ \partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2)) &= s_2 \cdot \partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2) \end{aligned}$$

Wir fassen die gerechneten Transitionsübergänge zusammen:



Die Axiome für die τ-Aktion und Abstraktion ergeben:

$$\begin{aligned}
& \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1)) \\
= \;& \tau_{\{c_3\}}(r_1 \cdot \partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2)) \\
= \;& r_1 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2)) \\
= \;& r_1 \cdot \tau_{\{c_3\}}(c_3 \cdot \partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \\
= \;& r_1 \cdot \tau \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \\
= \;& r_1 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2)))
\end{aligned}$$

Weiter rechnen wir:

$$\begin{aligned}
\tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \;=\; & \\
& r_1 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2))) \\
& + s_2 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1)) \\
\tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2))) \;=\; & \\
& s_2 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2)))
\end{aligned}$$

Damit erhalten wir als Lösung für die lineare rekursive Spezifikation E für einen Puffer der Kapazität 2:

$$\begin{aligned}
X &:= \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1)) \\
Y &:= \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \\
Z &:= \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2)))
\end{aligned}$$

9.5.2 Die Fairness-Regel

Es ist möglich, durch Abstraktion τ-Schleifen zu konstruieren: So führt $\tau_{\{a\}}(\langle X \mid X{=}aX \rangle)$ unendlich lange die τ-Aktion aus.

Definition 9.59 *Sei E eine geschützte lineare rekursive Spezifikation, C eine Teilmenge ihrer Variablen und $I \subset A$ eine Menge von Aktionen.*

- *C heißt* (Fairness-)Gruppe *(cluster) für I, falls für je zwei Rekursionsvariablen X und Y in C $\langle X|E\rangle \xrightarrow{b_1} \cdots \xrightarrow{b_m} \langle Y|E\rangle$ und $\langle Y|E\rangle \xrightarrow{c_1} \cdots \xrightarrow{c_n} \langle X|E\rangle$ für Aktionen $b_1, \cdots b_m, c_1 \cdots, c_n \in I \cup \{\tau\}$ gilt.*
- *a und aX heißen* Ausgang *(exit) der Gruppe C falls:*
  1. *a oder aX ein Summand auf der rechten Seite der Rekursionsgleichung für eine Rekursionsvariable in C ist, und*
  2. *im Fall aX zusätzlich $a \notin I \cup \{\tau\}$ oder $X \notin C$ gilt.*

Die Fairness-Regel CFAR: Falls X in einer Gruppe C für I mit Ausgängen $\{v_1 Y_1, \ldots, v_m Y_m, w_1, \ldots, w_n\}$ ist, dann gilt

$$\tau \cdot \tau_I(\langle X|E\rangle) = \tau \cdot \tau_I(v_1\langle Y_1|E\rangle + \cdots + v_m\langle Y_m|E\rangle + w_1 + \cdots + w_n)$$

Zur Erläuterung von CFAR sei E das Gleichungssystem:

$$\begin{aligned} X_1 &= aX_2 + b_1 \\ &\vdots \\ X_{n-1} &= aX_n + b_{n-1} \\ X_n &= aX_1 + b_n \end{aligned}$$

$\tau_{\{a\}}(\langle X_1|E\rangle)$ führt τ-Transitionen aus, bis eine Aktion b_i für $i \in \{1, \ldots, n\}$ ausgeführt wird.

Faire Abstraktion bedeutet, dass $\tau_{\{a\}}(\langle X_1|E\rangle)$ nicht für immer in einer τ-Schleife bleibt, d.h. irgend wann wird einmal ein b_i ausgeführt:

$$\tau_{\{a\}}(\langle X_1|E\rangle) \underline{\leftrightarrow}_{rb} b_1 + \tau(b_1 + \cdots + b_n)$$

Anfangs führt $\tau_{\{a\}}(\langle X_1|E\rangle)$ die Aktionen b_1 oder τ aus. Im letzteren Fall wird nach einer Reihe von möglichen τ-Aktionen ein b_i ausgeführt. Dies entspricht für $n = 3$ der Situation, dass in dem P/T-Netz von Abbildung 9.5 die mit f bezeichneten Transitionen b_1, b_2 und b_3 fair schalten (Definition 6.35 b)), d.h. der Zyklus der mit a bezeichneten Transitionen einmal verlassen wird.

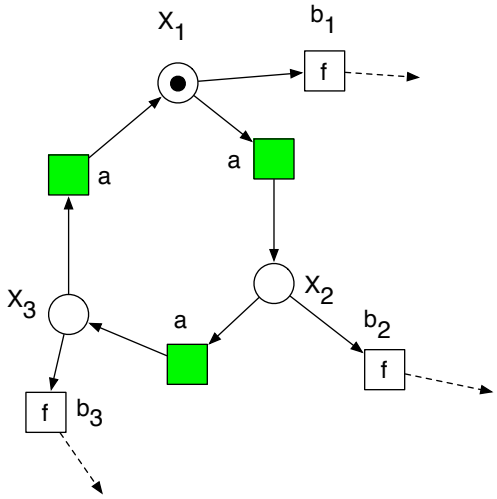


Abbildung 9.5: Faire Abstraktion als P/T-Netz mit fair schaltenden Transitionen.

Beispiel 9.60

$$X = heads \cdot X + tails$$

$\langle X|E\rangle$ stellt das Werfen einer idealen Münze dar, das mit *tails* endet. Von den Ergebnissen „Kopf“ wird abstrahiert: (head) $\tau_{\{heads\}}(\langle X|E\rangle)$.

$\{X\}$ ist die einzige Gruppe für $\{heads\}$ mit dem einzigen Ausgang *tails*. Daher erhält man mit der Regel CFAR:

$$\begin{aligned} \tau \cdot \tau_{\{heads\}}(\langle X|E\rangle) &= \tau \cdot \tau_{\{heads\}}(tails) \\ &= \tau \cdot tails \end{aligned}$$

und

$$\begin{aligned}\tau_{\{heads\}}(\langle X|E\rangle) &= \tau_{\{heads\}}(heads \cdot \langle X|E\rangle + tails)\\ &= \tau \cdot \tau_{\{heads\}}(\langle X|E\rangle) + tails\\ &= \tau \cdot tails + tails\end{aligned}$$

Anmerkung: Der Kalkül ACP_τ mit Abstraktion, geschützter linearer Rekursion und Fairnessregel ist korrekt und vollständig in Bezug auf initiale Verzweigungsbisimulation:

$$s = t \;\Leftrightarrow\; s \mathrel{\underline{\leftrightarrow}_{rb}} t$$