Seminar – Theorie kommunizierender Systeme
Einführung in den π-Kalkül

Christian Müller

17. Oktober 2004

Inhaltsverzeichnis

1 Einleitung

In Robin Milners Buch „communication and mobile systems: the π-calculus“ wird über die Hälfte des Textes darauf verwendet, das Kalkül *CCS* zu entwickeln. Die zahlreichen Beispiele der ersten Kapitel zeigen, daß das Kalkül schon recht ausdrucksstark ist. Aus welchem Grund sollen wir, nachdem wir viel Zeit und Aufwand für *CCS* investiert haben, zu einem neuen Kalkül – dem π-Kalkül – übergehen? Der Wechsel soll anhand eines Beispiels motiviert werden.

2 Mautsystem I

Abbildung 1 stellt zur Wiederholung die wesentlichen Elemente von CCS dar: die Syntax, den Handshake als formale Reaktion und Verwendung rekursiver Definitionen.

$$\alpha ::= x \mid \overline{x} \mid \tau$$

$$P ::= \sum_{i \in I} \alpha_i.P_i \quad | \quad P_1 | P_2 \quad | \quad \text{new}\, x\, P \quad | \quad A\langle \vec{a} \rangle$$

$$(a.A + M) \mid (\overline{a}.B + N) \rightarrow A \mid B$$

$$A(\vec{a}) \stackrel{\text{def}}{=} P_A \Rightarrow A\langle \vec{b} \rangle \equiv \{^{\vec{b}}/_{\vec{a}}\} P_A$$

Abbildung 1: Kernelemente aus CCS

Mithilfe dieses Kalküls soll im folgenden ein Mautsystem entworfen werden.

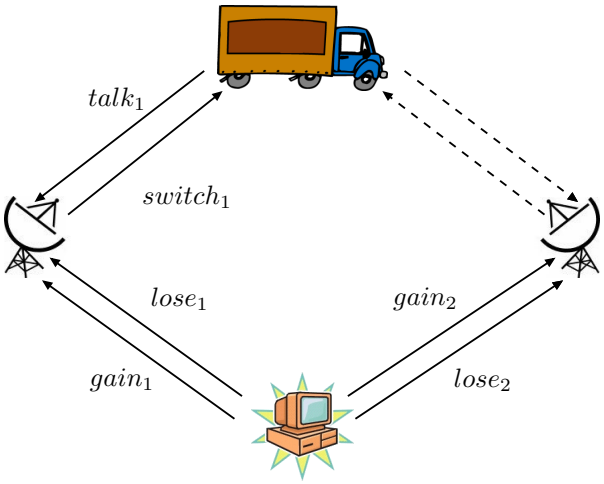


Abbildung 2: Mautsystem I

Die Struktur des Systems ist in Abbildung 2 dargestellt. Es besteht aus vier Komponenten:

1. ein LKW ($Truck_i$), der mit einem Transmitter kommuniziert.

2. ein Transmitter ($Trans_1$), der mit dem LKW verbunden ist.

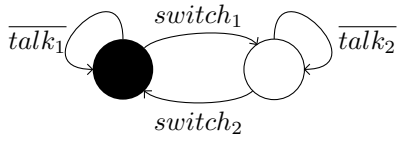
3. ein Transmitter ($Trans_2$), der bereit ist, Kommunikation aufzunehmen.

4. ein Kontroller ($Cnrtl$), der initiieren kann, daß der LKW die Verbindung zu $Trans_1$ beendet und mit $Trans_2$ aufgenommen wird.

Der LKW soll zu einem Zeitpunkt stets nur mit einem Transmitter kommunizieren können. Aufgabe des Kontrollers ist es diese Invariante zu erhalten und sicherzustellen, das der LKW beim Wechsel zu einem anderen Transmitter nicht aus dem System verloren geht. Beginnen wir mit der Modellierung des LKWs. Der LKW ist in der Lage ein *talk* Signal an den Transmitter zu versenden. Sobald er ein *switch* Signal empfängt, geht er in einen Zustand über, indem er mit dem anderen Transmitter kommuniziert. Da unser System aus zwei Transmittern besteht, kann der LKW zwei Zustände annehmen, die anhand von zwei verschränkt rekursiven Gleichungen wie folgt beschrieben werden können:

$$\begin{aligned} Truck_1 &\stackrel{\text{def}}{=} \overline{talk_1}.Truck_1 + switch_1.Truck_2 \\ Truck_2 &\stackrel{\text{def}}{=} \overline{talk_2}.Truck_2 + switch_2.Truck_1 \end{aligned}$$

In CCS war es sehr nützlich, sich die Gleichungen als Transitionssysteme vorzustellen. Daher werden die Transitionssysteme zu den Gleichungen im folgenden jeweils zusätzlich angegeben. Für den LKW besteht das System aus zwei Knoten. Der Zustand, in dem sich das System befindet ist immer schwarz markiert.



Schauen wir uns den Transmitter an. $Trans_1$ kann ein Signal über $talk_1$ empfangen und verändert dabei seinen Zustand nicht. Sobald der Transmitter ein $lose_1$ Signal vom Kontroller erhält, versendet er an den LKW ein $switch_1$ Signal. Bevor also die Verbindung zwischen $Trans_1$ und LKW gekappt wird, wird der LKW aufgefordert, seine Kommunikationskanäle zu wechseln. $Trans_2$ ist analog aufgebaut, befindet sich aber in einem anderen Zustand. Wenn $Trans_2$ das Signal $gain_2$ – also die Aufforderung mit dem LKW zu kommunizieren – erhalten hat, verhält er sich genau wie $Trans_1$. Diese informelle Beschreibung kann durch folgende CCS Gleichungen beschrieben werden:

$$\begin{aligned} Trans_1 &\stackrel{\text{def}}{=} talk_1.Trans_1 + lose_1.\overline{switch_1}.gain_1.Trans_1 \\ Trans_2 &\stackrel{\text{def}}{=} talk_2.Trans_2 + lose_2.\overline{switch_2}.gain_2.Trans_2 \end{aligned}$$

In den zugehörigen Transitionsgraphen in Abbildung 3 sind die Zustände, in denen sich die Transmitter befinden wieder schwarz markiert.

Fehlt also noch der Kontroller. Sobald dieser dem Transmitter mitteilt, daß er den Kontakt zum LKW abbrechen soll, muß sichergestellt werden, daß der andere Transmitter die Kommunikation übernimmt. Also muß er direkt nach

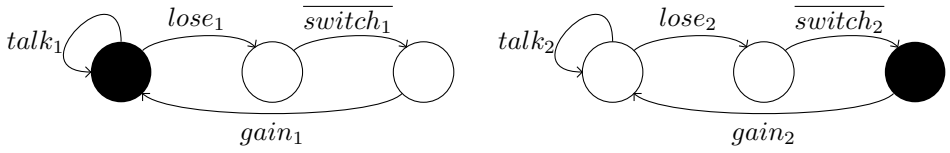
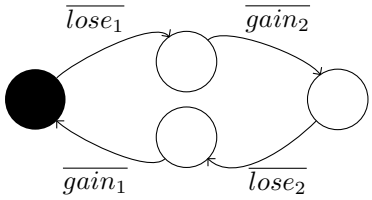


Abbildung 3: Transmitter

einer *lose*-Nachricht eine *gain*-Nachricht versenden. Dieses zyklische Wechselspiel kann durch zwei verschränkt rekursive Gleichungen bzw. ein Transitionssystem beschrieben werden:

$$\begin{aligned} Cntrl_1 &\stackrel{\text{def}}{=} \overline{lose_1}.\overline{gain_2}.Cntrl_2 \\ Cntrl_2 &\stackrel{\text{def}}{=} \overline{lose_2}.\overline{gain_1}.Cntrl_1 \end{aligned}$$



Damit sind die Komponenten des Systems vollständig beschrieben. Das Mautsystem aus Abbildung 2 entsteht durch die parallele Komposition der vier Einzelteile. Den Zustand, indem der LKW mit $Trans_1$ verbunden ist, erhalten wir als:

$$System_1 \stackrel{\text{def}}{=} Truck_1|Trans_1|Cntrl_1|gain_2.Trans_2$$

Mithilfe der CCS Reaktionsregeln läßt sich nun das Übergabeprotokoll zwischen den Transmittern formal beschreiben:

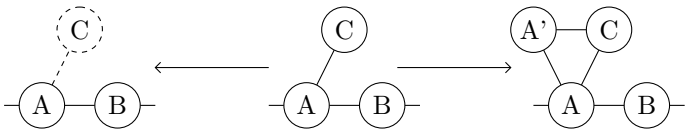
$$\begin{aligned} &System_1 \\ &\quad \rightarrow^\star \quad Truck_1|talk_1.Trans_1 + lose_1 \ldots|\overline{lose_1}.\overline{gain_2}.Cntrl_2|gain_2.Trans_2 \\ &\quad \rightarrow^\star \quad Truck_1|\overline{switch_1}.gain_1.Trans_1|\overline{gain_2}.Cntrl_2|gain_2.Trans_2 \\ &\quad \rightarrow^\star \quad switch_1.Truck_2|\overline{switch_1} \ldots Trans_1|Cntrl_2|Trans_2 \\ &\quad \rightarrow^\star \quad Truck_2|Trans_2|Cntrl_2|gain_1.Trans_1 \\ &\quad \rightarrow^\star \quad System_2 \end{aligned}$$

Unsere Modellierung in CCS scheint also erfolgreich gewesen zu sein. Aber wenn wir das System genauer betrachten, stellt sich heraus, daß es extrem unflexibel ist. Als erstes wurden zur Kommunikation zwischen den Transmittern und dem LKW feste Frequenzen gewählt. Werden diese Frequenzen gestört, so bricht die Verbindung zusammen, ohne das wir eine Möglichkeit haben, den Ausfall zu verhindern. Der zweite, vielleicht entscheidendere, Nachteil unseres Systems ist, daß es sehr statisch modelliert ist. Sollte sich der Betreiber entscheiden, weitere Transmitter aufzustellen, so müssen alle LKWs in die Werkstatt. Die

Kommunikationswege sind fest in die Onboard-Unit des LKWs integriert, was ein Update erforderlich macht, sobald die Infrastruktur erweitert wird. Es wäre daher sehr vorteilhaft, die Frequenzen die Kommunikationskanäle der Onboard-Unit abstrakt zu halten. Doch läßt sich diese Flexibilität in CCS überhaupt darstellen? Die Antwort lautet nein, und leitet zum nächstem Abschnitt, der Mobilität, über.

3 Mobilität

Zentrale Möglichkeit zur Kommunikation in CCS ist der *Handshake*. Zwei parallel ablaufende Prozesse können sich über ein gemeinsames Label synchronisieren, was formal durch die Reaktion $\overline{a}.A|a.B \rightarrow A|B$ beschrieben wird. In ein System verbundener Prozesse, können sich dadurch neue Komponenten einklinken. Ebenso können bestehende Komponenten im Laufe der Reaktion aus dem System entfernt werden.



Aber in CCS gibt es keine Möglichkeit zwischen bestehenden Komponenten eine neue Verbindung aufzubauen. Hierfür müßte den Komponenten mitgeteilt werden, wie die Verbindung heißt. In CCS können wir jedoch bei einem Handshake keine Informationen übertragen. Diese Einschränkung verhindert auch, daß im Mautsystem der Kontroller zentral die Frequenzen verwalten kann. In einem Kalkül zur Beschreibung von Kommunikation muß zwangsläufig eine Möglichkeit bestehen, Informationen zu übertragen. Diese Motivation führt uns zum π-Kalkül.

4 Der formale Rahmen des π-Kalkül

4.1 Die Syntax

Die Hinzunahme von Informationsübertragung verläuft auf formaler Ebene äußerst unspektakulär. Bei der Syntax ändern sich im Vergleich zu CCS nur die Aktionspräfixe, so daß wir zur folgenden Definition des π-Kalküls gelangen:

$x, y, z, ... \in \mathcal{N}$			unendliche Menge an Namen
π	::=	$x(y)$	empfange y über x
		$\overline{x}\langle y \rangle$	sende y über x
		τ	unsichtbare Aktion
P	::=	$\sum_{i \in I} \pi_i.P_i \quad \mid \quad P_1 \mid P_2 \quad \mid \quad \text{new}\, x\, P \quad \mid \quad A\langle \vec{a} \rangle$	

Die Syntax bietet uns eine nichtdeterministische Auswahl $\sum_{i \in I} \pi_i.P_i$, welche über die Aktionspräfixe π_i gesteuert werden kann. Die parallele Komposition $P_1|P_2$ beschreibt die nebenläufige Ausführung zweier Prozesse. Mit new $x\, P$ kann ein privater Name x eingeführt, also ein neuer Namensraum erzeugt werden, und $A\langle \vec{x} \rangle$ erlaubt uns die Verwendung rekursiver Definitionen [1].
Bei den Aktionspräfixen bleibt die unsichtbare Aktion τ, also eine spontane Aktion ohne Interaktion mit der Außenwelt, erhalten. Die beiden anderen Aktionen wurden zu Kanälen erweitert. Der Handshake wird wie bisher über gegensätzliche Labels x und $\overline{x}$ eingeleitet. Nach dem Andocken können jedoch Namen gesendet oder empfangen werden. (y) stellt den Empfänger dar und $\langle a \rangle$ den Sender. Aber wieso werden *Namen* übertragen? Der π-Kalkül entscheidet sich für Namen, die diese einen geeigneten Abstraktionsgrad besitzen, um Informationsübertragung möglichst allgemein zu beschreiben. Namen können als Zeiger auf beliebige Werte fungieren. In späteren Kapiteln zeigt Milner, wie im π-Kalkül die natürlichen Zahlen kodiert werden können, d.h. Namen können auch direkt zur Bildung von Werten eingesetzt werden.

Der Syntax muß nun Leben eingehaucht werden. Konkret sind wir an einem formalen System interessiert, welches die Kommunikation zwischen Prozessen beschreibt. Wir wir aus CCS schon wissen, kann der Handshake nur stattfinden, wenn sich zwei Prozesse in einer parallelen Komposition direkt gegenüber stehen. Wir benötigen folglich ein Werkzeug, welches uns in die Lage versetzt, Prozessausdrücke so zu manipulieren, daß eine Reaktion stattfinden kann. Für diese Transformation müssen bestimmte Eigenschaften gelten, damit wir die Semantik unserer Ausdrücke nicht versehentlich verändern.

4.2 Die strukturelle Kongruenz

Bevor wir die Umformungsregeln aufstellen, soll der allgemeine Rahmen abgesteckt werden. Dazu definieren wir zuerst den Begriff des *Kontextes*.

Definition (Prozesskontext)

$$\mathcal{C} ::= [\,] \quad \mid \quad \pi.\mathcal{C} + M \quad \mid \quad \text{new}\, x\, \mathcal{C} \quad \mid \quad \mathcal{C}|P \quad \mid \quad P|\mathcal{C}$$

Hierbei stellt $\mathcal{C}[Q]$ den Prozessausdruck dar, der sich aus dem Einsetzten von Q in den Kontext $\mathcal{C}$ ergibt.

[1] In Milners Buch gehören rekursive Definitionen nicht zur Syntax des π-Kalküls. Allerdings stellt er in Kapitel 9.4 dar, wie man sie mit anderen Mitteln kodieren kann. Um uns auf die zentrale Änderung, nämlich die Erweiterung der Präfixe, konzentrieren zu können, behalten wir die aus CCS bekannten Gleichungen vorerst bei.

Ein Kontext ist also einfach ein Ausdruck, der an einer Stelle eine Lücke ausweist. Mithilfe von Kontexten definieren wir, was wir unter einer *Prozesskongruenz* verstehen wollen.

Definition (Prozesskongruenz)

Sei $\cong$ eine beliebige Äquivalenzrelation über den Prozessausdrücken $\mathcal{P}^\pi$. Dann ist $\cong$ genau dann eine *Prozesskongruenz*, wenn die folgende Implikation gilt:

$$\forall P, Q \in \mathcal{P}^\pi \, \forall \mathcal{C} : \; P \cong Q \;\Rightarrow\; \mathcal{C}[P] \cong \mathcal{C}[Q]$$

Nur zur Wiederholung: eine Äquivalenzrelation, ist eine Relation, die reflexiv, symmetrisch und transitiv ist. Entscheidender ist die zweite Eigenschaft. Angenommen zwei Prozessausdrücke P und Q sind äquivalent zueinander. Dann kann man sie in jeden beliebigen Kontext einsetzten und erhält wieder äquivalente Ausdrücke. Insbesondere bedeutet dies: Wenn wir innerhalb eines Ausdrucks $\mathcal{C}[P]$, den Unterausdruck P, durch einen äquivalenten Ausdruck Q ersetzten, so ist $\mathcal{C}[P] \cong \mathcal{C}[Q]$. Also ist eine Prozesskongruenz zur Manipulation von Ausdrücken geeignet und wir definieren unser Werkzeug $\equiv$.

Definition (Strukturelle Kongruenz)

Die *strukturelle Kongruenz* $\equiv$ ist eine Prozesskongruenz, die durch folgende Gleichung gegeben ist:

1. Umbenennung gebundener Variablen
2. $\sum_{i \in I} \pi_i.P_i \equiv \sum_{i \in I} .\pi_{\mu(i)}.P_{\mu(i)} \quad \mu \in I \to I$
3. $P \mid Q \equiv Q \mid P \qquad (P \mid Q) \mid R \equiv P \mid (Q \mid R)$
4. $\mathrm{new}\, x\, (P|Q) \equiv P|\mathrm{new}\, x\, Q \qquad$ falls $x \notin \mathrm{fn}(P)$
5. $A(\vec{a}) \stackrel{\mathrm{def}}{=} P_A \;\Rightarrow\; A\langle\vec{b}\rangle \equiv \{^{\vec{b}}/_{\vec{a}}\}P_A$
6. $P|0 \equiv P \qquad \mathrm{new}\, x\, 0 \equiv 0 \qquad \mathrm{new}\, xy\, P \equiv \mathrm{new}\, yx\, P$

Die Gleichungen sind identisch zu denen aus CCS. Im folgenden sollen nochmals die wesentlichen Aspekte der Definition hervorgehoben werden.

1. Die Namen gebundener Variablen sind nicht relevant und können beliebig manipuliert werden. In CCS traten Binder in rekursiven Gleichungen und bei *new* auf. Im π-Kalkül stellt das Empfangen über einen Kanal eine zusätzliche Bindung dar. In $x(y).P$ wird die Variable y gebunden.
2. $\mu \in I \to I$ ist eine beliebige Permutation der Indexmenge. Innerhalb von Summen erlaubt es uns die strukturelle Kongruenz beliebig umzuordnen, um Prozesse reaktionsbereit zu machen.
3. Auch innerhalb einer parallelen Komposition spielt die Reihenfolge keine Rolle. Wir können somit Prozesse, die über einen gemeinsamen Kanal verfügen, so platzieren, daß sie direkt nebeneinander liegen.

4. Die Operation des Zusammenschiebens kann allerdings durch den *new* Binder blockiert werden. Glücklicherweise verfügen wir über Regel 1 und können den gebundenen Namen so wählen, daß er nicht in P vorkommt. Danach können wir den Binder gefahrlos nach außen ziehen.

5. Der Ausdruck $A\langle\vec{b}\rangle$ weißt gewisse Ähnlichkeit zu einem Prozeduraufruf auf. An seiner Stelle wird die Definition eingesetzt, in der die formalen Parameter durch die Aktualparameter $\vec{b}$ ersetzt werden.

6. Diese Gleichungen erlauben uns kleine kosmetische Veränderungen an den Ausdrücken vorzunehmen. Bspw. kann der inaktive Prozess 0 in einer parallelen Komposition entsorgt werden.

Mithilfe der strukturellen Kongruenz konnten wir die Prozessausdrücke so positionieren, daß eine Reaktion ablaufen kann. Wie sieht aber die Reaktion aus?

4.3 Reaktion

Definition (Reaktion):

Die *Reaktionsrelation* $\rightarrow$ ist durch das folgende Inferenzsystem definiert:

$$(x(y).P + M)|(\overline{x}\langle m\rangle.Q + N) \rightarrow \{^m/_y\}P|Q \qquad \text{(React)}$$

$$\frac{Q \equiv P \qquad P \rightarrow P' \qquad P' \equiv Q'}{Q \rightarrow Q'} \qquad \text{(Struct)}$$

$$\tau.P + M \rightarrow P \qquad \text{(Tau)}$$

$$\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q} \qquad \text{(Par)}$$

$$\frac{P \rightarrow P'}{\text{new}\, x\, P \rightarrow \text{new}\, x\, P'} \qquad \text{(Res)}$$

React ist die zentrale Regel, die Informationsübertragung modelliert. Wenn zwei Summanden parallel komponierter Summen über einen gemeinsamen Kanal verfügen, kann Kommunikation stattfinden. Die Reaktionsregel verwirft alle weiteren Alternativen und ersetzt im Prozess P den abstrakten Namen y durch die Nachricht m. In dieser Substitution manifestiert sich die grundlegende Neuerung. Der Handshake bewirkt also nicht mehr nur, daß sich die Prozesse synchronisieren, sondern auch, daß sich der Zustand des Empfängerprozesses deutlich verändert.
In vielen Fällen ist es notwendig, die Prozessausdrücke zuerst umzuformen, bevor eine Reaktion stattfinden kann. Daher wurde das Inferenzsystem um die Regel Struct ergänzt, die die strukturelle Kongruenz $\equiv$ mit der Reaktionsrelation $\rightarrow$ verbinden. Die Regel Tau ermöglicht, daß ein Prozess ohne Einwirkung von außen eine Aktion durchführt. Das Umsortieren in parallelen Kompositionen und das Nachaußenziehen von *new* Bindern macht natürlich nur dann

Sinn, wenn dadurch eine Reaktion ermöglicht wird. Die Regeln PAR und RES erlauben es uns, auch innerhalb bestimmter Konstrukte Reaktionen durchzuführen.

Betrachten wir mit unserem neuen Wissen noch einmal kurz die strukturelle Kongruenz. Warum wurde sie genauso gewählt, wie sie ist? Warum wurde zum Beispiel nicht eine zusätzliche Regel der Form

$$\pi.\text{new}\, x\, P + M \;\equiv\; \text{new}\, x\, (\pi.P + M) \qquad \text{falls } x \notin \text{fn}(M) \cup \text{fn}(\pi) \tag{1}$$

aufgenommen? Da sie analog zur dritten Regel von $\equiv$ aufgebaut ist, scheint nichts dagegen zu sprechen. Auf der anderen Seite muß man sich dann allerdings die Frage gefallen lassen, wieso wir unser System unnötig aufblähen sollten. Tatsächlich ist $\equiv$ genauso definiert, daß sie jede potentielle Reaktion ermöglicht. In der Regel (1) wird der *new* Binder vom Aktionspräfix π verdeckt und kann deshalb die Reaktion auf keine Weise beeinflussen. Die strukturelle Kongruenz ist exakt auf die Reaktionsmöglichkeiten zugeschnitten, so daß man durch zusätzliche Erweiterungen nichts gewinnt.

5 Mautsystem II

Der erste Entwurf des Mautsystems hat sich als sehr statisch herausgestellt. Wir hatten keine Möglichkeit die Frequenzen für die Kommunikation zwischen LKW und Transmittern zentral zu verwalten. In CCS fehlte die Möglichkeit flexibel neue Verbindungen zwischen den bestehenden Komponenten zu erzeugen. Mit dem π-Kalkül, können wir unser System so anpassen, daß es die beiden gewünschten Eigenschaften hat:

1. Frequenzen sollen nicht fest in den LKW und die Transmitter kodiert, sondern dynamisch zugeteilt werden.
2. Nach dem die Verbindung zwischen einem Transmitter und dem LKW getrennt wurde, soll der Kontroller entscheiden können, welcher Transmitter die Kommunikation übernimmt.

5.1 Neues Spiel, neues Glück

Wie man in Abbildung 4 sieht, bleibt der Zusammenhang zwischen den einzelnen Komponenten erhalten. Über die Kanäle können aber jetzt Informationen versendet werden.

Schauen wir uns an, wie sich die neuen Gleichungen für die einzelnen Komponenten ergeben:

$$\begin{aligned}
Truck(talk, switch) &\stackrel{\text{def}}{=} \overline{talk}.Truck\langle talk, switch\rangle + switch(t,s).Truck\langle t,s\rangle \\
Trans_i(talk, switch) &\stackrel{\text{def}}{=} talk.Trans_i\langle talk, switch\rangle \\
&\quad + lose_i(t,s).\overline{switch}\langle t,s\rangle.gain_i(t,s).Trans_i\langle t,s\rangle \\
Cntrl(l) &\stackrel{\text{def}}{=} \text{new } t\ s\ \bar{l}\langle t,s\rangle. \\
&\quad (\overline{gain_1}\langle t,s\rangle.Cntrl\langle lose_1\rangle + \overline{gain_2}\langle t,s\rangle.Cntrl\langle lose_2\rangle) \\
System_1 &\stackrel{\text{def}}{=} Truck\langle t_1, s_1\rangle | Trans_1\langle t_1, s_1\rangle \\
&\quad | Cntrl\langle lose_1\rangle | gain_2(t,s).Trans_2\langle t,s\rangle
\end{aligned}$$

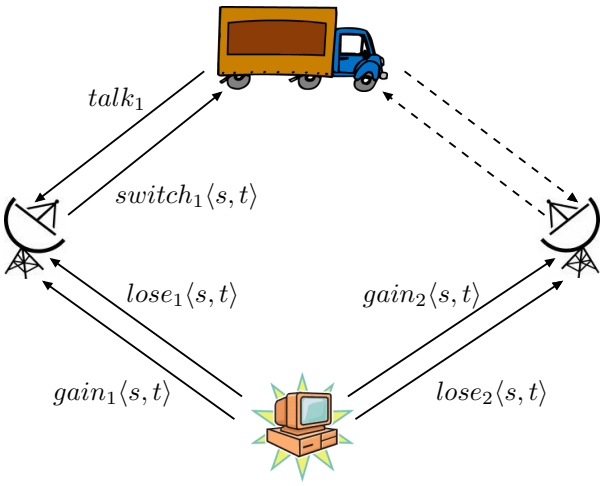


Abbildung 4: Mautsystem II

In der Definition des LKWs können wir die Kommunikationskanäle *talk* und *switch* abstrakt halten. Wie vorher auch, verändert sich der Zustand nicht, wenn ein *talk* Signal zum Transmitter gesendet wird. Die *switch* Nachricht überführt den LKW jedoch nicht mehr in einen fest verdrahteten Folgezustand. Vielmehr bekommt der LKW mit dieser Nachricht mitgeteilt, auf welche Frequenzen er umschalten soll. Da wir die Menge $\mathcal{N}$ der Namen als unendlich vorausgesetzt haben, kann der LKW unendlich viele Zustände annehmen, also mit beliebig vielen verschiedenen Transmittern kommunizieren. Der Transmitter ist ebenfalls über die Kommunikationskanäle *talk* und *switch* parametrisiert. Wenn er ein *talk* Signal empfängt verändert er seinen Zustand nicht. Mit der *switch* Nachricht versendet der Transmitter die Information über die Kanäle, die ihm durch *lose* vom Kontroller mitgeteilt wurde. Er leitet also diese Information nur weiter und fungiert somit als Router zwischen LKW und Kontroller. Nachdem er die Verbindung mit dem LKW beendet hat, ist er wieder aufnahmebereit. Im Unterschied zum ersten System teilt der Kontroller dem Transmitter mit, über welche Frequenzen er mit dem LKW kommunizieren soll. Der Kontroller scheint sich vom statischen Umschalter zum dynamischen Multitalent gemausert zu haben. Wie ist mit den wenigen syntaktischen Erweiterungen solch ein drastischer semantischer Unterschied zu erreichen?
Durch unser zweites Ziel haben wir dem Kontroller eine gewissen Entscheidungsfreiheit gewährt. Da aus der Struktur der Gleichung nicht mehr hervorgeht, welchen Transmitter der Kontroller bei einem Übergang auswählt, speichern wir diese Information unter dem Namen l. Wenn ein *gain* Signal gesendet wurde, geht der Kontroller in Rekursion mit dem Wissen, welche Verbindung er im nächsten Zyklus beenden kann. Für unser Mautsystem ist also bspw. sichergestellt, das die Verbindung, die über $gain_1$ aufgebaut wurde, über $lose_1$ gekappt werden kann. Die interessanteste Neuerung ist der *new* Binder. Mit ihm können ständig neue Frequenzen erzeugt werden und über *gain* und *lose* den anderen Komponenten bekannt gegeben werden. Das Gesamtsystem besteht aus der parallelen Komposition der vier Komponenten, wobei die Verbindung zwischen $Trans_1$ und LKW explizit angegeben werden muß.

5.2 Formale Übergänge und das Versenden von Namen

Auf den ersten Blick scheint es unsicher zu sein, die Namen t und s einfach nach außen zu versenden. Doch zeigt sich, daß die strukturelle Kongruenz gewährleistet, daß die privaten Namen nicht beliebig nach außen offengelegt werden. Schauen wir uns hierzu ein kleines Beispiel an:

$$P \stackrel{\text{def}}{=} \underbrace{x(z)}_{Q_1} \mid a(y).\underbrace{\overline{y}\langle b\rangle}_{Q_2} \mid \text{new}\, x\, (\overline{a}\langle x\rangle)$$

Wir wollen vermeiden, daß nach dem Versenden des privaten Namens x über a Prozess Q_1 mit Prozess Q_2 kommunizieren kann. Dies würde unserer Idee des Scoping widersprechen. Genau dieses Verhalten wird durch die vierte Regel der strukturellen Kongruenz unterbunden. Die Reaktion läuft deshalb wie folgt ab:

$$\begin{array}{lll} P & \stackrel{\equiv}{\rightarrow} & x(z) \mid \text{new}\, x\, (a(y).\overline{y}\langle b\rangle \mid \overline{a}\langle x\rangle) \\ & \stackrel{\text{React}}{\rightarrow} & x(z) \mid \text{new}\, x\, (\overline{x}\langle b\rangle \mid 0) \\ & \stackrel{\equiv}{\rightarrow} & x(z) \mid \text{new}\, x\, (\overline{x}\langle b\rangle) \\ & \stackrel{\equiv}{\rightarrow} & \text{new}\, x'\, (x(z) \mid \overline{x'}\langle b\rangle) \\ & \stackrel{\text{React}}{\not\rightarrow} & \end{array}$$

Es wird also kein Name aus dem Scope heraus gesendet, sondern der Scope bei Bedarf ausgedehnt. Hierbei stellt die strukturelle Kongruenz zusätzlich sicher, daß keine freien Namen gekapert werden. Wir können uns also beruhigt zurücklehnen und uns das Mautsystem II in Aktion anschauen.

Der Wechsel des LKWs von $Trans_1$ zu $Trans_2$ geschieht im neuen Mautsystem folgendermaßen:

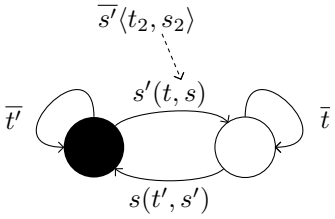
$$\begin{array}{ll} System_1 & \\ \rightarrow^* & \text{new}\, t\, s\, \overline{lose_1}\langle t,s\rangle.(\ldots) | lose_1(t,s).\overline{s_1}\langle t,s\rangle.gain_1(t,s).Trans_1\langle t,s\rangle + \ldots \\ \rightarrow^* & \text{new}\, t_2\, s_2\, (\overline{gain_1}\langle t_2,s_2\rangle.Cntrl\langle lose_1\rangle + \overline{gain_2}\langle t_2,s_2\rangle.Cntrl\langle lose_2\rangle \\ & | gain_2(t,s).Trans_2\langle t,s\rangle | \overline{s_1}\langle t_2,s_2\rangle.gain_1(t,s).Trans_1\langle t,s\rangle | Truck\langle t_1,s_1\rangle) \\ \rightarrow^* & Cntrl\langle lose_2\rangle | Trans_2\langle t_2,s_2\rangle \\ & | \overline{s_1}\langle t_2,s_2\rangle.gain_1(t,s).Trans_1\langle t,s\rangle | s_1(t,s).Truck\langle t,s\rangle + t_1.Truck\langle t,s\rangle \\ \rightarrow^* & Cntrl\langle lose_2\rangle | Trans_2\langle t_2,s_2\rangle | Truck\langle t_2,s_2\rangle | gain_1(t,s).Trans_1\langle t,s\rangle \\ \rightarrow^* & System_2 \end{array}$$

Es sollte keine Schwierigkeiten bereiten, die Reaktionen auszurechnen, die ablaufen, wenn die Frequenzen auf einem Transmitter ausgetauscht werden, ohne daß ein Wechsel stattfindet.

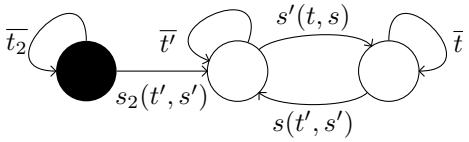
6 Endlich(e) Transitionssysteme?

Wann werden *endlich* Transitionssysteme zu den Gleichungen gezeigt? Sie erleichterten uns das Verständnis des ersten Mautsystems deutlich. Es wäre durchaus nützlich, wenn sich die Komponenten des zweiten Systems auf ähnlich Weise

graphisch darstellen ließen. Zuerst müssen wir uns aber klar darüber werden, wie sich der Übergang von CCS zum π-Kalkül auf die Transitionssysteme auswirkt. Angenommen wir würden für den LKW das System aus Abschnitt 2 übernehmen und nur die Markierung an den Verbindungen modifizieren.



Die Abbildung zeigt den LKW, der gerade eine *switch*-Nachricht empfängt. In CCS entsprach eine Reaktion einem Zustandsübergang im Diagramm. Es wurde also der rechte Knoten schwarz gefärbt. Dies reicht aber im π-Kalkül offensichtlich nicht aus. In der REACT Regel findet nämlich eine Substitution statt. Leider läßt sich diese schwer im Transitionssystem darstellen. Wenn wir nämlich die Kantenmarkierungen ersetzten, können wir nicht mehr zum Ausgangszustand zurückkehren. Eine Möglichkeit ist, analog zu den Prozessausdrücken, das System einmal aufzufalten und die Substitution nur auf einen Teil anzuwenden. Nach der Reaktion erhält man dadurch folgendes System:



Für jeden Zustand, den das System einnehmen kann, sieht das Transitionssystem etwas anders aus. Da wir für den LKW aber schon wissen, daß er unendlich viele Zustände einnehmen kann, erhalten wir ein unendliches Transitionssystem. Die statischen und *endlichen* Systeme, die wir in CCS verwendet haben, helfen uns im π-Kalkül nicht mehr weiter. Hier würde man eine mächtigere, aber auch kompliziertere, Art von Automaten benötigen. Die Automaten in CCS wurden nur zur Verdeutlichung der rekursiven Gleichungen verwendet. Für den π-Kalkül belassen wir es bei den Gleichungen und bei den Strukturübersichten, die wir für das zweite Mautsystem gesehen haben, und verschwenden keine Mühe auf die Entwicklung komplexer Automaten.

7 Replikation

Rekursive Definitionen erlauben eine elegante Modellierung von komplexen Systemen. Allerdings können sie manche Beweise deutlich erschweren. Dadurch, daß man im π-Kalkül Namen übertragen kann, ist es möglich zu einem kompakteren Formalismus überzugehen: der *Replikation*. Wir streichen zunächst die rekursiven Definitionen aus unserem formalen System und nehmen folgende Ergänzungen vor:

$$P ::= \ldots \mid !P$$

$$!P \equiv P \mid !P$$

$$\mathcal{C} ::= \ldots \mid !\mathcal{C}$$

Syntaktisch besteht die Replikation aus einem Prozessausdruck vor den ein Ausrufungszeichen gesetzt wird. Die Semantik wird über die strukturelle Kongruenz definiert. $!P$ kann den Prozessausdruck P beliebig oft replizieren, wobei die einzelnen Instanzen parallel komponiert werden. Interessanterweise wurde wieder die strukturelle Kongruenz zur Definition eines zentralen semantischen Konzepts eingespannt. Sie ist jetzt dafür zuständig Prozesse reaktionsbereit zu machen, das Scoping zu erweitern und bei Bedarf neue Kopien eines Prozesses mithilfe der Replikation zu erzeugen. Dies ist mit ein Grund, weshalb das Inferenzsystem aus Kapitel 12 des Buches, sehr komplex wird, da es ohne $\equiv$ definiert ist.

8 Ausblick

Die Semantik der Replikation legt nahe, daß sie Rekursion subsumiert. Allerdings trügt der Schein oft, weshalb Kapitel 9.5 einer Kodierung rekursiver Definitionen mit Replikation und dem polyadischen π-Kalkül gewidmet ist. Für das Mautsystem haben wir den letzteren einfach verwendet. Die beiden Namen t und s wurden über einen einzelnen Kanal gesendet. Die formale Rechtfertigung findet man im Kapitel 9.4. Das Verhalten von Systemen, also wie sich bspw. eine Implementierung in Bezug zu ihrer Spezifikation verhält, wurde ebenfalls noch nicht betrachtet. Die Lektüre der verbleibenden Buchkapitel bietet also noch viele interessante Einsichten. Die Absicht dieser Ausarbeitung war es dem Leser eine Intuition vom π-Kalkül zu vermitteln und zur weiteren Erkundung der neuen Landschaft einzuladen.

9 Literatur

- Milner R., Parrow, J. and Walker D., *A calculus of mobile processes, Parts I and II*, Information and Computation, 100, 1, 1992, pp1-77

  Originalartikel zum π-Kalkül

- Milner, R., *communication and mobile systems: the π-calculus*, Cambridge University Press, 1999

  Grundlage des Seminars

- Sangiorgi, D. and Walker, D., *The π-calculus: A Theory of Mobile Processes*, Cambridge University Press, 2001

  Sehr ausführliches Buch über fast alle Bereiche des *pi*-Kalküls

- Milner, R., *What's in a name*, January 2003, http://www.cl.cam.ac.uk/users/rm135/wosname.pdf

  Netter kleiner Artikel über die grundlegende Philosophie des π-Kalküls mit Konzentration auf den Aspekt der Namen

10 Anhang

Dieser Abschnitt behandelt einige weiterführende Aspekte der strukturellen Kongruenz. Da diese nicht grundlegend für das Verständnis des π-Kalküls sind, wurden sie in der Präsentation nicht angesprochen. Zum Einstimmen wollen wir ein kleines Lemma zur strukturellen Kongruenz beweisen.

Lemma 1: $x \notin fn(Q) \Rightarrow \text{new}\, x\, Q \,|\, Q$

Beweis: $\text{new}\, x\, Q \equiv \text{new}\, x\, Q|0 \equiv Q|\text{new}\, x, 0 \equiv Q|0 \equiv Q$

Dies ist ein weiterer Beweis dafür, daß wir $\equiv$ zwar sehr kompakt gewählt haben, es aber keine Schwierigkeit bereitet weitere Eigenschaften abzuleiten.

Lemma 2:

Jeder Prozess P, der keine Replikation enthält, ist strukturell kongruent zu einer Standardform, d.h.

$$\forall P \in \hat{\mathcal{P}}^{\pi} : P \equiv \text{new}\, \vec{x}\, (N_1 | \ldots | N_n)$$

Wobei $\hat{\mathcal{P}}^{\pi}$ die Menge der Prozessausdrücke ohne Replikation ist.

Lemma 3:

Jeder Prozess P ist strukturell kongruent zu einer Standardform, d.h.

$$\forall P \in \mathcal{P}^{\pi} : P \equiv \text{new}\, \vec{x}\, (N_1 | \ldots | N_n)$$

Den Beweis zu Lemma 2 findet man im Buch auf Seite 32. Lemma 3 entspricht Proposition 9.13 auf Seite 90.

Definition (erweiterte Standardform)

Ein Prozessausdruck

$$P \stackrel{\text{def}}{=} \text{new}\, x_{1k}\, (M_1 | \ldots | M_n | !Q_1 | \ldots | !Q_m)$$

ist in *erweiterter Standardform* (ESF), wenn

1. alle seine Unterausdrücke in Standardform sind
2. für die Binder $\text{new}\, x_1 \ldots_k$ aller Unterausdrücke Q von P gilt: $\{x_1, \ldots, x_k\} \subseteq \text{fn}(Q)$

Insbesondere bedeutet dies, wenn P in ESF ist und einen Unterausdruck $\pi.Q$ enthält, dann ist Q auch in ESF.

Lemma 4

Jeder Prozess P ist strukturell kongruent zu einer ESF.

Beweis:

1. Aus Lemma 3 wissen wir, daß alle Prozessausdrücke strukturell kongruent zu einer Standardform sind. Des weiteren ist $\equiv$ eine Prozesskongruenz. Also können alle Unterausdrücke durch ihre Standardform ersetzt werden.
2. Mit Lemma 1 können alle x_i, die nicht frei in P vorkommen, gelöscht werden.

□

Lemma 5

$P \equiv Q$ ist entscheidbar für alle $P, Q \in \hat{\mathcal{P}}^{\pi}$

Beweis:
Mit Lemma 4 können P und Q in die ESF P' und Q' überführt werden. Die ESF ist nicht eindeutig bzgl. der Ordnung und der Namen der Variablen in einer Bindung, der Ordnung innerhalb einer parallelen Komposition und der Ordnung innerhalb einer Summe. Da die Ausdrücke endlich sind, gibt es nur endlich viele Unterausdrücke und endlich viele Variablen. Es gibt daher nur endlich viele Möglichkeiten zur Umordnung und Umbenennung. Wenn P' und Q' durch Umordnung und Umbenennung gebundener Variablen syntaktisch gleich gemacht werden können, so gilt $P \equiv Q$.

□

Satz (Entscheidbarkeit von $\equiv$)

$P \equiv Q$ ist entscheidbar für alle $P, Q \in \mathcal{P}^{\pi}$.

Überlegen wir uns zuerst, was durch die Hinzunahme der Replikation im Vergleich zu Lemma 5 schwerer wird. Wegen der Gleichung $!P \equiv P|!P$ können durch die Replikation beliebig viele Prozessausdrücke P erzeugt werden. Wenn wir die Ausdrücke, nachdem wir sie in EFS gebracht haben, syntaktisch vergleichen wollen, dann müssen wir dieses Auffalten soweit wie möglich rückgängig machen. Dies kann durch die folgende Inferenzregel ausgedrückt werden:

$$\frac{P \equiv Q}{P|!Q \rightarrow_C !Q} \qquad (\textsc{Compress})$$

Wenn wir diese Komprimierung, so oft wie möglich angewendet haben, können wir den Test auf syntaktische Gleichheit modulo Umbenennung und Umordnung analog zu Lemma 5 durchführen. In der Regel Compress müssen wir entscheiden können, ob $P \equiv Q$ gilt. Dummerweise wollen wir gerade diese Eigenschaft zeigen. Hier hilft jedoch ein Beweis über Induktion.

Definition:

Sei $P \in \mathcal{P}^\pi$. Die Funktion $\#_! \in \mathcal{P}^\pi \rightarrow \mathbb{N}$ gibt die maximale Anzahl von Replikationen an, die sich auf einem Pfad im Syntaxbaum zu P befinden.

Beweis (Entscheidbarkeit von $\equiv$)

Sei $P \in \mathcal{P}^\pi$ in ESF. Es reicht zu zeigen, daß wir P soweit wie möglich komprimieren können, d.h. $\exists P' \in \mathcal{P}^\pi : P \rightarrow_C^\star P' \wedge P' \rightarrow_C^\star P'$.

Beweis mit Induktion über $\#_!(P)$.

$\#_!(\mathbf{P}) = \mathbf{0}$:
Folgt mit Lemma 5.

$\#_!(\mathbf{P}) = \mathbf{1}$:
Also enthält P einen Unterausdruck der Form $!Q$. Da $\#_!(Q) = 0$, können wir $P \rightarrow_C^\star P'$ bilden, so daß P' nicht mehr weiter komprimiert werden kann.

$\#_!(\mathbf{P}) > \mathbf{1}$:
P enthält Unterausdrücke der Form $!Q$, wobei $\#_!(Q) = \#_!(P) - 1$ gilt. Wegen der Induktionsannahme können wir Q soweit wie möglich komprimieren. Analog zum Beweis von Lemma 5 können wir nun die strukturelle Kongruenz zwischen Q und parallel komponierten Unterausdrücken von P entscheiden und somit P soweit wie möglich komprimieren.

□