Funktionale Spezifikation eines Kommunikationsprotokolls *

Katharina Spies

Institut für Informatik
Technische Universität München
Postfach 20 24 20, D-80290 München
spiesk@informatik.tu-muenchen.de

Mai 1994

Zusammenfassung

Als Fallstudie wird die funktionale Spezifikation eines einfachen Kommunikationsprotokolls erarbeitet. Dabei wird gezeigt, wie sich unterschiedliche Spezifikationsformalismen zur formalen Spezifikation eines Agenten-Netzwerkes unter Verwendung expliziter Zustände in die Focus-Methodik einbeziehen lassen. Üblicherweise werden die Agenten gemäß Focus durch zustandsbehaftete, stromverarbeitende Funktionen beschrieben. Es wird gezeigt, daß das Verhalten eines Agenten jeweils für einen ausgezeichneten Zustand mit Hilfe einer einfachen Tabelle spezifiziert werden kann, deren Semantik wiederum durch eine Menge von Funktionsgleichungen angegeben wird. Zusätzlich kann eine graphische Darstellung des Verhaltens eines Agenten in Form eines endlichen Automaten direkt aus der Tabellendarstellung abgeleitet werden.

1 Einleitung

Focus (siehe [BDD+92b] und [BDD+93]) ist eine Entwurfsmethodik zur schrittweisen, formalen Entwicklung verteilter Systeme. Sie ist formal ausgerichtet und mathematisch fundiert, so daß insbesondere Verfeinerungsschritte und der Nachweis der Korrektheit dieser Schritte systematisch und unter Verwendung mathematischer Beweismethoden durchgeführt werden können. Die Entwicklung eines Systems gemäß Focus ist in drei charakteristische Phasen aufgegliedert; auf diese Weise werden ausgehend von einer ersten Anforderungsspezifikation über den Entwurf des strukturellen, internen Aufbaus in der Designphase bis hin zu einem parallel ablaufenden Programm mehrere Entwicklungsschritte durchlaufen. Im Laufe der Entwicklung

*Diese Arbeit wurde unterstützt vom Sonderforschungsbereich 342 "Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen"

werden in den einzelnen Phasen unterschiedliche Formalismen eingesetzt, wobei jedoch der Übergang zwischen den verschiedenen Beschreibungsformen mühelos bewältigt werden kann.

Der strukturelle Aufbau eines Systems wird in der Designphase durch die Beschreibung eines Agenten-Netzwerkes angegeben, in dem die Agenten über Kanäle asynchron miteinander kommunizieren. Formal werden die einzelnen Agenten durch stromverarbeitende Funktionen spezifiziert, die üblicherweise durch Prädikate oder auch eine Reihe von Funktionsgleichungen definiert werden. Für die stromverarbeitenden Funktionen sind Kompositionsoperatoren definiert, so daß auch die Agenten-Netzwerke durch stromverarbeitende Funktionen spezifiziert werden. Zur Steigerung der Lesbarkeit solcher funktionaler Spezifikationen ist es möglich, in die Definition der Funktionen Zustände als zusätzliche Parameter aus einem explizit definierten Zustandsraum aufzunehmen.

Spezifikationen, die in dem so kurz vorgestellten Sinne erstellt wurden, sind aufgrund der speziellen Notationen und einer daraus resultierenden möglicherweise komplexen Darstellung für Nicht-Spezialisten oft schwer lesbar und somit auch nicht immer sofort nachvollziehbar. Aus diesem Grund erscheint es wichtig, zur Steigerung der Praktikabilität und folglich auch der Akzeptanz formaler Methoden – in diesem Fall Focus – andere bereits bekannte Formalismen und Techniken für die Erstellung der formalen Spezifikation anzubieten. Hierbei sollte jedoch gewährleistet sein, daß diese "neuen" Formalismen adäquat sind und zielgerichtet in der entsprechenden Methodik eingesetzt werden können.

Der Einsatz dieser Techniken ist in allen Phasen der Systementwicklung vorstellbar. In der hier vorgestellten Fallstudie wird gezeigt, wie sich Techniken wie Tabellenspezifikationen (vgl. [Par92], [Jan93] und auch [Fuc93]) und graphische Darstellungen wie die Graphen zu endlichen Automaten (siehe z.B. [Bra84] oder [HU88]) an die funktionale Spezifikation einzelner Agenten eines Netzwerks mit zustandsbehafteten, stromverarbeitenden Funktionen anbinden lassen. Die vorgestellte Entwicklung der formalen Spezifikation aus der informellen Beschreibung soll vor allem nachvollziehbar und leicht verständlich sein.

Die Fallstudie stützt sich auf das in [PP92] beschriebene Kommunikationsprotokoll, das vorschreibt, wie Daten von einem Erzeuger über ein Medium an einen Empfänger geschickt werden. Die Besonderheit des Protokolls besteht darin, daß die Erlaubnis zur Datenübertragung durch den Empfänger erteilt und wieder entzogen werden kann. Eine geringfügige Variante dieses Protokolls wird auch in [GG92] angegeben.

Die vorliegende Arbeit ist wie folgt gegliedert: In Kapitel 2 werden das Protokoll und das Agenten-Netzwerk, durch welches das Protokoll realisiert werden soll, informell beschrieben. In Kapitel 3 werden die Notationen und Techniken zur Erstellung funktionaler Spezifikationen mit der Focus-Methodik kurz vorgestellt. Kapitel 4 beinhaltet die Entwicklung der vollständigen formalen Spezifikation der einzelnen Agenten zur Realisierung des beschriebenen Protokolls in Form von einfachen Tabellen, die Beschreibung der Verbindung zu den stromverarbeitenden Funktionen und die Spezifikation des vollständigen Netzwerkes in der Focus-Notation durch stromverarbeitende Funktionen. Zusätzlich werden einige Prädikate angegeben, die Eigenschaften der einzelnen Agenten beschreiben. In Kapitel 5 erfolgt die Herleitung der graphischen Darstellung der Agenten in Form von Graphen endlicher Automaten direkt aus der in Kapitel 4 vorgestellten Tabellenspezifikation. Diese Darstellung wird unter anderem dafür genutzt, um Zustandsprädikate herzuleiten. Kapitel 6 beinhaltet einige Bemerkungen und stellt weitere Vorgehensweisen ausgehend von dem vorgestellten Beispiel vor.

2 Informelle Beschreibung des Protokolls

In diesem Abschnitt werden das Verhalten des Protokolls (siehe auch [PP92] und [GG92]) informell beschrieben und einige der geforderten Eigenschaften angegeben.

Eine typische Protokollsituation wird durch Abbildung 1 veranschaulicht (Einen knappen, allgemeinen Überblick über typische Protokollsituationen geben z.B. [JA90] und [BHS91]). In der oberen Schicht sind die beiden Protokolleinheiten – Erzeuger (*Producer*) und Verbraucher (*Consumer*) – angegeben, die über ein Medium miteinander kommunizieren. Der Erzeuger liefert Daten über die Verbindung PnB an das Medium, und diese werden dann an den Verbraucher weitergeleitet. Eine Besonderheit des hier beschriebenen Szenarios besteht darin, daß der Verbraucher das Medium dazu berechtigt, die Daten vom Erzeuger an den Verbraucher zu übertragen bzw. das Weiterleiten der Daten zu unterbinden. Die Übermittlung dieser Informationen an das Medium erfolgt über die Verbindung PnA und die Weiterleitung der Daten an den Verbraucher über die Verbindung AnP.

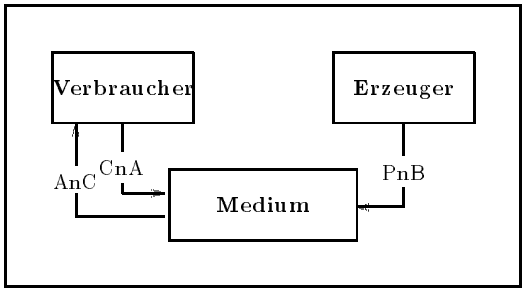

Abbildung 1: Erste Sicht auf das Protokoll

Die untere Schicht – das Medium – wird durch zwei Einheiten realisiert. Die Einheiten A und B realisieren die gemäß dem Protokollverhalten geforderte Übertragung der Daten und dabei insbesondere die Blockierung bzw. Erteilung der Erlaubnis zur Datenübertragung zwischen Erzeuger und Verbraucher. Auf diese Weise ergibt sich die in Abbildung 2 gezeigt Gesamtsituation für das Protokoll.

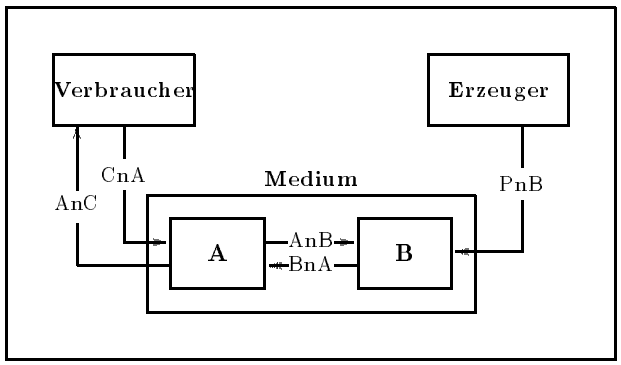

Abbildung 2: Detailliertere Sicht auf das Protokoll

Das zu spezifizierende System besteht somit aus den zwei Einheiten A und B, die in eine Umgebung eingebettet sind, in der es insbesondere den Erzeuger und den Verbraucher gibt.

Konkret spezifiziert werden in diesem Papier die Einheiten A und B.

Die zu übermittelnden Daten werden von dem Erzeuger an B gesandt; dieser schickt die Daten, falls die Weiterleitung der Daten aktuell zugelassen ist, an A, der sie dann an den Verbraucher weiterleitet. Daten, die der Erzeuger an B schickt, solange die Übertragung der Daten zwischen A und B nicht erlaubt ist, werden gespeichert und gehen somit nicht verloren. Die Datenübertragung zwischen den Einheiten A und B wird gemäß dem Protokoll dadurch realisiert, daß spezielle Nachrichten über die Verbindungen AnB und BnA ausgetauscht werden. Hierfür sind die Nachrichten C_{req} (*Connection request*) und R_{req} (*Release request*) für A und C_{cnf} (*Connection confirm*) und R_{cnf} (*Release confirm*) für B definiert.

Für das Verhalten der Agenten A und B wird durch das Protokoll folgendes festgelegt:

- Einheit A sendet C_{req} an B; hiermit wird die Erlaubnis zur Datenübertragung zwischen A und B erteilt.
- B bestätigt den Erhalt von C_{req} durch das sofortige Senden der Bestätigung C_{cnf} an A.
- Solange die Datenübertragung zwischen den beiden Einheiten zugelassen ist, können Daten von B nach A weitergeleitet werden.
- Einheit A initiiert das Verbot zur Datenübertragung durch das Senden von R_{req} an B.
- Einheit B bestätigt den Erhalt von R_{req} durch das Senden von R_{cnf} an A.

Mit den speziellen Nachrichten $\mathcal{E}$ und $\mathcal{V}$, die der Verbraucher an die Einheit A senden kann, erteilt der Verbraucher dem Medium die Berechtigung zur Datenübertragung bzw. entzieht diese Berechtigung wieder.

Die Spezifikation soll obige informelle Beschreibung formal beschreiben; dabei sollen auch Eigenschaften, wie z.B.

- das Versenden der Nachrichten $\mathcal{E}$ bzw. $\mathcal{V}$ durch den Empfänger muß innerhalb des Mediums zur Folge haben, daß die Berechtigung bzw. der Entzug dieser Berechtigung realisiert wird;
- die Nachrichten C_{req} und R_{req}, die A an B versendet, sollen auch mit den entsprechenden Bestätigungen beantwortet werden;
- Daten dürfen nur bei bestehender Übertragungserlaubnis weitergeleitet werden. Ist die Übertragung zum aktuellen Zeitpunkt untersagt, so darf keine Datenübertragung stattfinden;
- der Verbraucher erhält nur solche Daten, die vom Erzeuger versandt wurden; diese erhält er in der Reihenfolge, in der sie erzeugt wurden,

garantiert sein.

Das so beschriebene Verhalten soll zunächst mit den Formalismen, die gemäß der Designphase von Focus vorgesehen sind, spezifiziert werden. Damit ergibt sich das Netzwerk von Agenten (siehe auch Abbildung 3), in dem die Agenten A und B durch stromverarbeitende Funktionen spezifiziert werden. Agent A wird mit den Eingabekanälen CnA und BnA und den Ausgabekanälen AnC und AnB spezifiziert und Agent B mit den Eingabekanälen AnB und PnB und dem Ausgabekanal BnA. Zusätzlich wird Zeitverhalten durch die spezielle Aktion $\surd$ ("Zeit-Tick") modelliert. Mit $\surd$ wird das Fortschreiten der Zeit durch eine Zeiteinheit, deren Länge der Ausführungsdauer einer der anderen definierten Aktionen entspricht,

spezifiziert. Auf diese Weise wird explizit beschrieben, daß zum entsprechenden Zeitpunkt keine der Aktionen aus der definierten Aktionenmenge stattfindet. In dem hier beschriebenen Protokoll wird Zeitverhalten nur für die Agenten A, B und den Verbraucher modelliert; d.h. diese drei Komponenten des verteilten Systems sind über eine globale Uhr zusammengeschaltet. Weiterhin wird im Gegensatz zur standardmäßigen Notation der stromverarbeitenden Funktionen zugelassen, zur Spezifikation zustandsbehaftete Funktionen zu benutzen, die es erlauben, die Funktionen mit einem zusätzlichen Parameter zu versehen, so daß mit Variablen aus einem explizit definierten Zustandsraum gearbeitet werden kann.

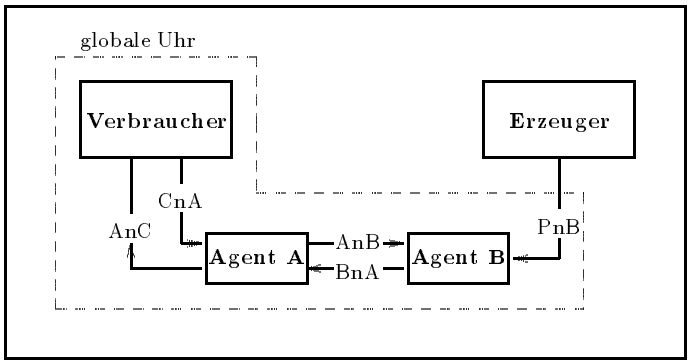

Abbildung 3: Protokoll als Agenten-Netzwerk

3 Mathematische Konzepte zur funktionalen Spezifikation

In diesem Kapitel werden einige der mathematischen Konzepte, die der Modellierung verteilter Systeme in der Designphase der Methodik Focus (siehe [BDD+92b] und [BDD+93]) zugrundeliegen, vorgestellt.

Für jeden Agenten des Systems wird eine endliche Anzahl von Ein- und Ausgabe-Kanälen definiert, über die Nachrichten asynchron ausgetauscht werden. Die Kommunikationsgeschichte der Kanäle wird mathematisch durch endliche oder unendliche Nachrichtenströme modelliert. In der Designphase wird ein Agent durch eine Menge stetiger stromverarbeitender Funktionen modelliert, die durch ein Prädikat über Funktionen definiert wird. Semantisch gesehen besteht kein Unterschied zwischen einem Agenten und einem Netzwerk von Agenten. Ein Netzwerk kann entweder durch eine Reihe von Funktionsgleichungen oder durch die Komposition von stromverarbeitenden Funktionen mit speziellen Operatoren zur sequentiellen und parallelen Komposition und Rückkopplung angegeben werden.

Ein Nachrichtenstrom über einem festgelegten Alphabet Act ist eine endliche oder unendliche Folge von Nachrichten. Für die Menge aller Ströme wird definiert:

$$Act^{\omega} \triangleq Act^{*} \cup Act^{\infty}$$

Hierbei bezeichnet Act^{*} die Menge der endlichen Ströme und Act^{∞} die Menge der unendlichen Ströme über dem Alphabet Act.

Für die Ströme sind weiterhin folgende Operationen und Relationen definiert:

$\langle\rangle$ bezeichnet den leeren Strom.

a bezeichnet den Strom, der nur aus der Aktion a besteht.

$x \circ y$ bezeichnet die Konkatenation der Ströme x und y. Hierbei gilt $x \circ y = x$, falls x unendlich ist.

$a\&y$ bezeichnet die Konkatenation von Aktion a und Strom y.

$\#x$ bezeichnet die Länge von Strom x; diese ergibt sich zu ∞ falls x unendlich ist.

$ft(x)$ bezeichnet die erste Aktion von Strom x; $ft(x)$ ist undefiniert, falls $x = \langle\rangle$ gilt.

$rt(x)$ bezeichnet den Rest von Strom x, also x ohne die erste Aktion; $rt(x)$ liefert den leeren Strom, falls $x = \langle\rangle$ gilt.

$M©x$ liefert den Strom, der sich durch das Herausfiltern der Aktionen der Menge M aus Strom x ergibt; es gilt: $M©\langle\rangle = \langle\rangle$.

$a\ in\ x$ zeigt an, ob die Aktion a im Strom x enthalten ist.

$x \sqsubseteq y$ beschreibt, daß Strom x ein Präfix von Strom y ist. Auf der Menge der Ströme ist hiermit eine partielle Ordnung definiert, d.h. die Menge der Ströme bilden zusammen mit der Präfix-Ordnung eine *cpo*.

$fst(...)$ liefert, angewendet auf ein Tupel von Strömen, das erste Element des Tupels. Entsprechend liefert $snd(\ldots)$ das zweite Element des Tupels.

Eine stromverarbeitende Funktion ist eine Funktion, die zu einem Tupel von Strömen als Eingabe ein Tupel von Strömen als Ausgabe liefert und bezüglich der Präfix-Ordung stetig ist. Formal wird das Verhalten einer Komponente über den Alphabeten In und Out mit n Eingabe- und m Ausgabe-Kanälen wie folgt angegeben:

$$P \ : \ (\ (In^{\omega})^{n} \ \rightarrow \ (Out^{\omega})^{m}\ ) \longrightarrow I\!B$$

Eine stromverarbeitende Funktion mit n Eingabe- und m Ausgabe-Kanälen nennen wir (n, m)-stellige Funktion. Das Prädikat P beschreibt folgende Menge stetiger stromverarbeitender Funktionen:

$$\{\ f \ : \ (In^{\omega})^{n} \ \rightarrow \ (Out^{\omega})^{m} \mid P(f) \ \wedge \ f \text{ ist stetig } \}$$

Auf funktionaler Ebene besteht mit Focus auch die Möglichkeit zur expliziten Modellierung von Zeitverhalten. Es wird eine diskrete, globale Zeit vorausgesetzt, so daß in jedem Zeitintervall eine Nachricht gesendet oder empfangen werden kann. Auf diese Weise entspricht jedes Element eines gezeiteten Stromes genau einem Kommunikationsereignis in einem Zeitinterval. Wird keine explizite Nachricht versandt, so wird dies durch das spezielle Ereignis $\surd$ ("Zeit-Tick") modelliert. Detaillierte Arbeiten zur Modellierung von Zeitverhalten in Focus finden sich in [Bro89] und [BD90].

Stromverarbeitende Funktionen können zusätzlich mit (Hilfs-)Argumenten parametrisiert werden, die keine Ströme sind. Auf diese Weise werden zustandsbehaftete, stromverarbeitende Funktionen definiert (siehe auch [DW92]). Dabei wird die Funktion als abstraktes Modell eines Agenten angesehen, der eine bestimmte Eingabe erhält, abhängig von seinem internen Zustand Ausgaben liefert und in einen Folgezustand übergeht. Mit der Definition eines expliziten Zustandsraumes *State* können dann Funktionen folgender Form verwendet werden:

$$F \ : \ State \rightarrow (\ (In^{\omega})^{n} \ \longrightarrow \ (Out^{\omega})^{m}\ )$$

Die Definition eines Agenten-Netzwerkes erfolgt mit Hilfe speziell definierter Operatoren zur Komposition stromverarbeitender Funktionen. Hier werden nur die Definitionen der Operatoren zur sequentiellen Komposition und für die Rückkopplung kurz angegeben.

Sequentielle Komposition Seien f eine (n,m)- und g eine (m,o)-stellige Funktion. Dann ist $f \circ g$ eine (n,o)-stellige Funktion, die definiert ist durch:

$$(f \circ g)(x) \;=\; g(f(x))$$

Rückkopplung Sei f eine (n,m)-stellige Funktion mit $n > 0$. Dann ist μf eine $(n-1,m)$-stellige Funktion so, daß sich der Wert von μf durch den kleinsten Fixpunkt einer Funktion g ergibt. Es gilt:

$$\begin{aligned}(\mu f)\,(x_1,\ldots,x_{n-1}) \;&=\; fix.g \\ \textbf{where}\quad g(y_1,\ldots,y_m) \;&=\; f(x_1,\ldots,x_{n-1},y_m)\end{aligned}$$

Detaillierte Informationen zu den hier vorgestellten mathematischen Konzepten der Focus-Methodik finden sich in [BDD+92b] und [BDD+93].

4 Spezifikation der Agenten

Gemäß den im vorangegangenen Kapitel 2 vorgestellten Konzepten zur funktionalen Spezifikation eines Agenten-Netzwerks wird das zu Beginn informell beschriebene Protokoll in diesem Abschnitt formal spezifiziert. Hierbei wird das beschriebene Medium bestehend aus den Agenten A und B als Agenten-Netzwerk durch zustandsbehaftete und gezeitete stromverarbeitende Funktionen und eine spezielle Tabellennotation funktional spezifiziert.

Dafür seien folgende Ein- und Ausgabe-Alphabete für die Agenten A und B definiert. Die Alphabete InA_C, InA_B, InB_A und InB_P sind den Kanälen CnA, BnA, AnB bzw. PnB und die Alphabete $OutA_C, OutA_B$ und $OutB_A$ den Kanälen AnC, AnB bzw. BnA zugeordnet.

$$\begin{array}{rclcrcl}
InA_C & \triangleq & \{\mathcal{E},\mathcal{V},\surd\} & & InA_B & \triangleq & \{C_{cnf},R_{cnf},\surd\}\cup D \\
InB_P & \triangleq & D & & InB_A & \triangleq & \{C_{req},R_{req},\surd\} \\
OutA_C & \triangleq & \{\surd\}\cup D & & OutA_B & = & InB_A \\
OutB_A & = & InA_B & & D & \triangleq & \text{Datenmenge}
\end{array}$$

Als Aktionsmenge zur Spezifikation des beschriebenen Protokolls ergibt sich somit:

$$Act \;\triangleq\; InA_C \cup InA_B \cup InB_P \cup InB_A \cup OutA_C$$

Agent A wird mit 4 und Agent B mit 3 internen Zuständen definiert, so daß sich folgende Zustandsräume ergeben:

$$\begin{aligned}AState \;&\triangleq\; \{As0,As1,As2,As3\} \\ BState \;&\triangleq\; \{Bs0,Bs1,Bs2\}\end{aligned}$$

Wie bereits in der Einleitung angedeutet wurde, soll bei der Darstellung der Spezifikationen vor allem darauf Wert gelegt werden, daß sie verständlich und auch für Nicht-Spezialisten leicht lesbar ist. Aus diesem Grund wird zunächst die funktionale Spezifikation für den Anfangszustand des Agenten A mit den üblicherweise in der Designebene von Focus vorgesehenen Notationen angegeben. Anschließend erfolgt die Darstellung dieser Spezifikation durch eine Tabelle, die dann in eine komprimierte Fassung gebracht wird, um die Lesbarkeit der Spezifikation zu steigern. Unter Zuhilfenahme dieser Darstellung werden anschließend die Agenten A und B vollständig formal spezifiziert, wobei jeweils die Darstellung mit stromverarbeitenden Funktionen zusätzlich angegeben wird, da durch sie die Semantik der Tabellen definiert wird.

4.1 Agent A

Agent A steht in direkter Verbindung zum Verbraucher und realisiert die Erteilung bzw. den Entzug der Erlaubnis zur Übertragung der Daten vom Erzeuger zum Verbraucher. Zusätzlich werden die von Agent B übertragenen Daten durch den Agenten A direkt an den Verbraucher weitergereicht. In diesem Sinne wird Agent A mit den beiden Eingabekanälen CnA und BnA und den beiden Ausgabekanälen AnC und BnA, siehe auch Abbildung 4, durch zustandsbehaftete und gezeitete, stromverarbeitende Funktionen spezifiziert. Das geforderte Verhalten von Agent A wird mit dem Zustandsraum $AState$ mit vier expliziten Zuständen beschrieben.

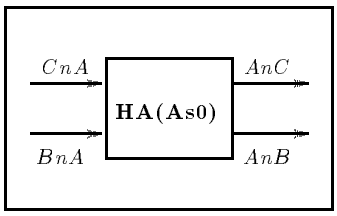

Abbildung 4: Agent A

Die Spezifikation erfolgt durch eine Menge von Funktionsgleichungen, die jeweils das Verhalten für einen der vier Zustände definieren. Die Vorgehensweise zur Darstellung mittels Tabellen und den speziellen komprimierten Tabellen wird nur für den Zustand $As0$ beschrieben. Anschließend wird das Verhalten für die drei verbleibenden Zustände in der so erklärten Tabellennotation spezifiziert.

Agent A wird durch die stromverarbeitende Funktion HA spezifiziert

$$HA \ : \ AState \ \rightarrow \ \Big[\ (InA_C^{\omega} \times InA_B^{\omega}) \ \rightarrow \ (OutA_C^{\omega} \times OutA_B^{\omega}) \ \Big]$$

Zustand As0: "Warten auf $\mathcal{E} \in InA_C$ "

Zustand $As0$ ist der Anfangszustand für den Agenten A. Er ist dadurch gekennzeichnet, daß keine Daten zwischen den Agenten A und B übertragen werden dürfen und somit keine Kommunikationsverbindung zwischen Erzeuger und Verbraucher existiert. Agent A wartet ausschließlich auf die Nachricht $\mathcal{E}$ des Verbrauchers, um dann die Erlaubnis zur Datenübertragung an B weiterzuleiten.

Erhält Agent A die Nachricht $\mathcal{E}$ als Eingabe über Kanal CnA, so sendet er sofort die Nachricht C_{req} an B über den Ausgabekanal AnB und geht in den Folgezustand $As1$ über. Wenn der Verbraucher in $As0$ entweder keine spezielle Nachricht ($\surd$) oder $\mathcal{V}$ an A schickt, so werden diese Nachrichten ignoriert, und Agent A verbleibt in Zustand $As0$.

Alle Nachrichten, die in Zustand $As0$ von Agent B an A gesendet werden, werden ignoriert und bewirken keine Ausgaben von Agent A. Da die Verbindung zwischen Erzeuger und Verbraucher im Zustand $As0$ noch nicht besteht, können auch keine Daten von A an den Verbraucher geschickt werden.

Die Spezifikation erfolgt in dem Sinne vollständig, daß alle mit den definierten Eingabealphabeten aktuell als erstes Element lesbaren Eingabekombinationen explizit behandelt werden. Damit ergeben sich folgende Funktionsgleichungen:

$\forall\ x \in InA_C^\omega,\ y \in InA_B^\omega\ , d \in D\ :$

$$
\begin{array}{lcl}
HA(As0)\,(\surd\ \&\ x, \surd\ \&\ y) & = & (\surd,\surd)\ \&\ HA(As0)\,(x,y)\ \wedge \\
HA(As0)\,(\surd\ \&\ x, C_{cnf}\ \&\ y) & = & (\surd,\surd)\ \&\ HA(As0)\,(x,y)\ \wedge \\
HA(As0)\,(\surd\ \&\ x, R_{cnf}\ \&\ y) & = & (\surd,\surd)\ \&\ HA(As0)\,(x,y)\ \wedge \\
HA(As0)\,(\surd\ \&\ x, d\ \&\ y) & = & (\surd,\surd)\ \&\ HA(As0)\,(x,y)\ \wedge \\
HA(As0)\,(\mathcal{E}\ \&\ x, \surd\ \&\ y) & = & (\surd, C_{req})\ \&\ HA(As1)\,(x,y)\ \wedge \\
HA(As0)\,(\mathcal{E}\ \&\ x, C_{cnf}\ \&\ y) & = & (\surd, C_{req})\ \&\ HA(As1)\,(x,y)\ \wedge \\
HA(As0)\,(\mathcal{E}\ \&\ x, R_{cnf}\ \&\ y) & = & (\surd, C_{req})\ \&\ HA(As1)\,(x,y)\ \wedge \\
HA(As0)\,(\mathcal{E}\ \&\ x, d\ \&\ y) & = & (\surd, C_{req})\ \&\ HA(As1)\,(x,y)\ \wedge \\
HA(As0)\,(\mathcal{V}\ \&\ x, \surd\ \&\ y) & = & (\surd,\surd)\ \&\ HA(As0)\,(x,y)\ \wedge \\
HA(As0)\,(\mathcal{V}\ \&\ x, C_{cnf}\ \&\ y) & = & (\surd,\surd)\ \&\ HA(As0)\,(x,y)\ \wedge \\
HA(As0)\,(\mathcal{V}\ \&\ x, R_{cnf}\ \&\ y) & = & (\surd,\surd)\ \&\ HA(As0)\,(x,y)\ \wedge \\
HA(As0)\,(\mathcal{V}\ \&\ x, d\ \&\ y) & = & (\surd,\surd)\ \&\ HA(As0)\,(x,y)
\end{array}
$$

Diese Spezifikation läßt sich auch mit Hilfe einer Tabelle formalisieren. (Arbeiten zu Tabellenspezifiaktionen sind beispielsweise [Par92] und [Jan93] oder auch [Fuc93] als weitere Fallstudie im Rahmen von Focus.) Eine Tabelle beschreibt das Verhalten in einem festgelegten Zustand. Für jeden der definierten Eingabekanäle ist jeweils eine Spalte vorgesehen, ebenso für jeden der vorgesehenen Ausgabekanäle. Diese beiden Klassen von Spalten werden durch "‖" voneinander getrennt; zuerst werden die Spalten zur Beschreibung der Eingaben und anschließend die Spalten zur Beschreibung der Ausgaben angegeben. Zusätzlich wird in der letzten Spalte der Folgezustand, durch ein "|" von den anderen Spalten abgetrennt, angegeben. In die einzelnen Spalten werden die Nachrichten eingetragen, die – bei den Eingabekanälen – aktuell als Eingabe vorliegen, und bei den Ausgabekanälen – bedingt durch die entsprechende Eingabe – als Ausgabe produziert werden. Zur Verbesserung der Lesbarkeit dieser Tabellen werden über die einzelnen Spalten die Kanalnamen und "As'" für den Folgezustand notiert. Wesentlich für das Verständnis dieser Tabellen ist auch, daß die beschriebenen Eingaben alle gleichzeitig vorliegen müssen, um die entsprechende Ausgabe zu produzieren. Mit diesen Erläuterungen ergibt sich zur Spezifikation von Zustand $As0$ aus den oben angegebenen Funktionsgleichungen sofort folgende Tabelle:

Zustand As0:

CnA	**BnA**	**AnC**	**AnB**	**As'**
$\surd$	$\surd$	$\surd$	$\surd$	$As0$
$\surd$	C_{cnf}	$\surd$	$\surd$	$As0$
$\surd$	R_{cnf}	$\surd$	$\surd$	$As0$
$\surd$	d	$\surd$	$\surd$	$As0$
$\mathcal{E}$	$\surd$	$\surd$	C_{req}	$As1$
$\mathcal{E}$	C_{cnf}	$\surd$	C_{req}	$As1$
$\mathcal{E}$	R_{cnf}	$\surd$	C_{req}	$As1$
$\mathcal{E}$	d	$\surd$	C_{req}	$As1$
$\mathcal{V}$	$\surd$	$\surd$	$\surd$	$As0$
$\mathcal{V}$	C_{cnf}	$\surd$	$\surd$	$As0$
$\mathcal{V}$	R_{cnf}	$\surd$	$\surd$	$As0$
$\mathcal{V}$	d	$\surd$	$\surd$	$As0$

Durch die Formalisierung der Spezifikation des Verhaltens in Zustand $As0$ mit der hier

angegebenen Tabelle reicht es aus, die für die Charakterisierung des Zustandes relevante Information anzugeben. Der Anwender ist in der Lage, diese Formalisierung vorzunehmen, ohne sich mit den speziellen Notationen, die für die stromverarbeitenden Funktionen definiert sind, auseinanderzusetzen. Die Umsetzung der Formalisierung als Tabelle in Funktionsgleichungen in der Form, in der sie oben angegeben sind, kann dann automatisch vorgenommen werden.

Aus der oben angegebenen Tabelle wird jedoch weiterhin ersichtlich, daß Agent A im Zustand $As0$ für zahlreiche Eingabekombinationen gleiches Verhalten zeigt. Es gibt für $As0$ im wesentlichen nur zwei relevante Verhaltensweisen: Über Kanal CnA erfolgt die Eingabe von $\mathcal{E}$ oder nicht. Dies wird auch bereits mit der informellen Beschreibung von Zustand $As0$ ersichtlich. Aus diesem Grund sollte es möglich sein, in der Formalisierung der Tabelle Schreibarbeit zu sparen, so daß ausschließlich die wesentlichen Charakteristika von Zustand $As0$ ersichtlich werden.

Zu diesem Zweck werden sogenannte *komprimierte Tabellen* definiert. Rein formal gesehen, werden Tabellen in derselben Notation notiert, wie es bereits oben beschrieben wurde. Spalten für die aktuellen Ein- und Ausgaben über die definierten Kanäle und den Folgezustand. Zusätzlich wird jedoch zugelassen, daß Variablen als Einträge in den Spalten auftreten dürfen. Die Definition des zulässigen Bereiches, für den die als Einträge benutzten Variablen definiert sind, erfolgt durch ein Prädikat, das in einem speziell hierfür vorgesehenen Rahmen zusätzlich zur Tabelle angegeben wird. Die komprimierte Tabelle zur Spezifikation von Zustand $As0$ ergibt sich damit wie folgt:

$$\boldsymbol{As = As0} \;\Rightarrow\quad \forall\ \gamma \in \{\surd, \mathcal{V}\},\ \beta \in \{\surd, C_{cnf}, R_{cnf}\} \cup D \quad :$$

CnA	**BnA**	**AnC**	**AnB**	**As'**
$\mathcal{E}$	β	$\surd$	C_{req}	$As1$
γ	β	$\surd$	$\surd$	$As0$

Für die Spezifikation der stromverarbeitenden Funktion HA ergibt sich mit dieser abkürzenden Schreibweise dann auch folgende Menge von Funktionsgleichungen:

$\forall\ \gamma \in \{\surd, \mathcal{V}\},\ \beta \in \{\surd, C_{cnf}, R_{cnf}\} \cup D\ .\ \forall\ x \in InA^{\omega}_{C},\ y \in InA^{\omega}_{B}$:

$$\begin{aligned} HA(As0)\,(\gamma\ \&\ x, \beta\ \&\ y) &= (\surd, \surd)\ \&\ HA(As0)\,(x, y)\ \wedge \\ HA(As0)\,(\mathcal{E}\ \&\ x, \beta\ \&\ y) &= (\surd, C_{req})\ \&\ HA(As1)\,(x, y) \end{aligned}$$

Mit dem so eingeführten Formalismus, den komprimierten Tabellen und den verkürzten Funktionsgleichungen wird jetzt die Spezifikation des Protokolls vollständig erarbeitet. Es werden immer zuerst die Tabellen und anschließend die Funktionsgleichungen angegeben.

Zustand As1: " Warten auf $C_{cnf} \in InA_B$; $\mathcal{V} \in InA_C$ möglich "

Zustand $As1$ ist dadurch gekennzeichnet, daß die Erlaubnis zur Datenübertragung durch den Verbraucher erteilt und von Agent A an B weitergeleitet wurde. Agent A wartet auf die Bestätigung, daß B die Nachricht C_{req} erhalten hat. Es können auch weiterhin noch keine Daten übertragen werden. Der Verbraucher kann die Erlaubnis zur Datenübertragung bereits wieder entziehen.

Agent A reagiert auf die Eingabe C_{cnf} von Agent B und auf Eingaben vom Verbraucher, die verschieden von $\mathcal{V}$ sind, mit der Ausgabe $\surd$ und dem Wechsel in den Folgezustand $As2$. Erhält A durch den Verbraucher die Eingabe $\mathcal{V}$, so erzeugt er die Ausgabe R_{req} an B und wechselt in den Folgezustand $As3$.

Solange Agent A von B eine Eingabe erhält, die verschieden von C_{cnf} ist, verbleibt er in Zustand $As1$ und liefert als Ausgaben nur $\surd$. Hierbei ist es wichtig, daß eine Eingabe $\mathcal{V}$ durch den Verbraucher nicht verloren geht, sondern gespeichert wird bis die Bestätigung C_{cnf} von Agent B vorliegt.

Als Spezifikation in Form einer komprimierten Tabelle ergibt sich damit für den Zustand $As1$:

$\boldsymbol{As = As1} \;\Rightarrow\quad \forall\, \gamma \in \{\surd, \mathcal{E}\},\ \beta \in \{\surd, R_{cnf}\} \cup D\ :$

CnA	**BnA**	**AnC**	**AnB**	**As'**
γ	C_{cnf}	$\surd$	$\surd$	$As2$
$\mathcal{V}$	C_{cnf}	$\surd$	R_{req}	$As3$
γ	β	$\surd$	$\surd$	$As1$
$\mathcal{V}\_$	β	$\surd$	$\surd$	$As1$

In dieser Tabelle wird die spezielle Notation $a\_$ für ein $a \in Act$ eingeführt, wodurch verdeutlicht werden soll, daß die Nachricht a gespeichert und somit nicht verarbeitet wird.

Die Tabelle entspricht folgenden Funktionsgleichungen:

$\forall\, \gamma \in \{\surd, \mathcal{E}\},\ \beta \in \{\surd, R_{cnf}\} \cup D\ .\ \forall\, x \in InA^{\omega}_{C},\ y \in InA^{\omega}_{B}\ :$

$$
\begin{aligned}
HA(As1)\,(\gamma\ \&\ x, C_{cnf}\ \&\ y) &= (\surd, \surd)\ \&\ HA(As2)\,(x,y)\ \wedge\\
HA(As1)\,(\mathcal{V}\ \&\ x, C_{cnf}\ \&\ y) &= (\surd, R_{req})\ \&\ HA(As3)\,(x,y)\ \wedge\\
HA(As1)\,(\gamma\ \&\ x, \beta\ \&\ y) &= (\surd, \surd)\ \&\ HA(As1)\,(x,y)\ \wedge\\
HA(As1)\,(\mathcal{V}\ \&\ x, \beta\ \&\ y) &= (\surd, \surd)\ \&\ HA(As1)\,(progress(\mathcal{V}\ \&\ x), y)
\end{aligned}
$$

Mit der in diesen Funktionsgleichungen benutzten Hilfsfunktion *progress* wird das Speichern einer Nachricht, die auf einem Eingabekanal vorliegt, realisiert. Mit ihr wird definiert, daß die erste Nachricht, die an einem Eingabekanal vorliegt, nicht entfernt wird, sondern der nächstfolgende "Zeittick", womit das Fortschreiten der Zeit gewährleistet wird. Die Hilfsfunktion *progress* wird für ein beliebiges Alphabet $Act_{\surd}$ formal definiert durch:

$$progress\ :\ Act^{\omega}_{\surd}\ \rightarrow\ Act^{\omega}_{\surd}$$

wobei gilt:

$$
\begin{aligned}
progress(\langle\rangle) &= \langle\rangle\\
progress(\surd \& x) &= x && \text{für } x \in Act^{\omega}_{\surd}\\
progress(a \& x) &= a \& progress(x) && \text{für } a \neq \surd \text{ und } x \in Act^{\omega}_{\surd}
\end{aligned}
$$

Zustand As2: " Warten auf $d \in InA_B$ oder $\mathcal{V} \in InA_C$ "

Im Zustand $As2$ dürfen Daten von Agent B an Agent A übertragen werden. Der Verbraucher kann die Erlaubnis zur Datenübertragung aber auch wieder entziehen.

Bei Eingabe d über Kanal BnA und beliebiger Eingabe über CnA, die verschieden von $\mathcal{V}$ ist, leitet Agent A das Datum an den Verbraucher über Kanal AnC weiter und verbleibt in Zustand $As2$. Liegen als Eingaben d und $\mathcal{V}$ vor, werden sowohl d weitergeleitet als auch die Nachricht R_{req} an B für den Entzug der Erlaubnis zur Datenübertragung ausgegeben, und Agent A geht in den Folgezustand $As3$ über.

Liegt kein Datum zur Übertragung vor, aber auf Kanal CnA die Nachricht $\mathcal{V}$, so wird R_{req} über AnB ausgegeben, und Agent A geht in den Zustand $As3$ über. Liegen weder ein Datum noch $\mathcal{V}$ vor, so schreitet nur die Zeit voran, und A verbleibt in Zustand $As2$.

Damit ergibt sich die Spezifikation:

$$\boldsymbol{As = As2} \;\Rightarrow\quad \forall\ \gamma \in \{\surd, \mathcal{E}\},\ \beta \in \{\surd, R_{cnf}, C_{cnf}\},\ d \in D\ :$$

CnA	**BnA**	**AnC**	**AnB**	**As'**
γ	d	d	$\surd$	$As2$
$\mathcal{V}$	d	d	R_{req}	$As3$
$\mathcal{V}$	β	$\surd$	R_{req}	$As3$
γ	β	$\surd$	$\surd$	$As2$

$$\forall\ \gamma \in \{\surd, \mathcal{E}\},\ \beta \in \{\surd, C_{cnf}, R_{cnf}\},\ d \in D\ .\ \forall\ x \in InA_C^{\omega},\ y \in InA_B^{\omega}\ :$$

$$\begin{aligned}
HA(As2)\,(\gamma\ \&\ x, d\ \&\ y) &= (d, \surd)\ \&\ HA(As2)\,(x,y)\ \wedge\\
HA(As2)\,(\mathcal{V}\ \&\ x, d\ \&\ y) &= (d, R_{req})\ \&\ HA(As3)\,(x,y)\ \wedge\\
HA(As2)\,(\mathcal{V}\ \&\ x, \beta\ \&\ y) &= (\surd, R_{req})\ \&\ HA(As3)\,(x,y)\ \wedge\\
HA(As2)\,(\gamma\ \&\ x, \beta\ \&\ y) &= (\surd, \surd)\ \&\ HA(As2)\,(x,y)
\end{aligned}$$

Zustand As3: " Warten auf $R_{cnf} \in InA_B$; $d \in InA_B$ und $\mathcal{E} \in InA_C$ möglich "

Zustand $As3$ ist dadurch gekennzeichnet, daß der Verbraucher die Erlaubnis zur Datenübertragung entzogen hat, Agent A diese Information an Agent B weitergeleitet hat und nun auf die Bestätigung R_{cnf} für den Erhalt von R_{req} durch Agent B wartet.

Erhält Agent A von B die Eingabe R_{cnf} und von dem Verbraucher eine Eingabe, die verschieden von $\mathcal{E}$ ist, so liefert er die Ausgabe $(\surd, \surd)$ und geht in den Anfangszustand $As0$ über. Wird zusätzlich zu R_{cnf} die Eingabe $\mathcal{E}$ durch den Verbraucher geliefert, wird C_{req} an B ausgegeben, und Agent A geht sofort in den Zustand $As1$ über.

Daten, die noch als Eingaben durch Agent B vorliegen, solange die Bestätigung noch nicht geschickt wurde, werden von Agent A an den Verbraucher weitergeleitet; Agent A verbleibt solange in Zustand $As3$. Die Eingabe $\mathcal{E}$ muß gespeichert werden, bis die Bestätigung R_{cnf} vorliegt, erst dann darf sie verarbeitet werden.

$\boldsymbol{As = As3} \;\Rightarrow\quad \forall \gamma \in \{\surd, \mathcal{V}\},\ \beta \in \{\surd, C_{cnf}\},\ d \in D\ :$

CnA	BnA	AnC	AnB	As'
γ	R_{cnf}	$\surd$	$\surd$	$As0$
$\mathcal{E}$	R_{cnf}	$\surd$	C_{req}	$As1$
γ	d	d	$\surd$	$As3$
$\mathcal{E}\_$	d	d	$\surd$	$As3$
$\mathcal{E}\_$	β	$\surd$	$\surd$	$As3$
γ	β	$\surd$	$\surd$	$As3$

$$\forall \gamma \in \{\surd, \mathcal{V}\},\ \beta \in \{\surd, C_{cnf}\},\ d \in D\ .\ \forall\, x \in InA^{\omega}_{C},\ y \in InA^{\omega}_{B}\ :$$

$$\begin{aligned}
HA(As3)\,(\gamma\ \&\ x, R_{cnf}\ \&\ y) &= (\surd, \surd)\ \&\ HA(As0)\,(x,y)\ \wedge\\
HA(As3)\,(\mathcal{E}\ \&\ x, R_{cnf}\ \&\ y) &= (\surd, C_{req})\ \&\ HA(As1)\,(x,y)\ \wedge\\
HA(As3)\,(\gamma\ \&\ x, d\ \&\ y) &= (d, \surd)\ \&\ HA(As3)\,(x,y)\ \wedge\\
HA(As3)\,(\mathcal{E}\ \&\ x, d\ \&\ y) &= (d, \surd)\ \&\ HA(As3)\,(progress(\mathcal{E}\ \&\ x), y)\ \wedge\\
HA(As3)\,(\mathcal{E}\ \&\ x, \beta\ \&\ y) &= (\surd, \surd)\ \&\ HA(As3)\,(progress(\mathcal{E}\ \&\ x), y)\ \wedge\\
HA(As3)\,(\gamma\ \&\ x, \beta\ \&\ y) &= (\surd, \surd)\ \&\ HA(As3)\,(x,y)
\end{aligned}$$

Mit der Definition der vier komprimierten Tabellen – zu jedem Zustand des für A definierten Zustandsraumes eine Tabelle – und den dazugehörigen Funktionsgleichungen ist das geforderte Verhalten von Agent A vollständig spezifiziert, d.h. für alle möglichen Eingabekombinationen wurde das geforderte Verhalten explizit spezifiziert.

4.1.1 Eigenschaften von Agent A

Der so spezifizierte Agent A soll u.a. folgende Eigenschaften erfüllen:

a1: Agent A schickt nie eine Nachricht C_{req} an den Agenten B, wenn dies nicht vorher durch ein $\mathcal{E}$ des Verbrauchers initiiert wurde.

$$\forall\, x \in InA^{\omega}_{C},\ y \in InA^{\omega}_{B}\ :$$
$$\Big[\ \forall\, b \sqsubseteq snd\ (\ HA(As0)(x,y)\ )\ :\ \ \#\{\mathcal{E}\}©x\ \ \geq\ \ \#\{C_{req}\}©b\ \Big]$$

Diese Eigenschaft gilt entsprechend auch für $\mathcal{V}$ und R_{req}.

a2: Die Daten-Ausgabe von Agent A an den Verbraucher ist immer Präfix der Daten-Eingabe durch B. Es werden also nur solche Daten an den Verbraucher weitergeleitet, die von B geschickt wurden.

$$\forall\, x \in InA^{\omega}_{C},\ y \in InA^{\omega}_{B}\ :$$
$$\Big[\ \forall\, a \sqsubseteq fst\ HA(As0)(x,y)\ :\ \ D©a\ \sqsubseteq\ D©y\ \Big]$$

a3: Schickt der Verbraucher genügend $\mathcal{E}$'s an Agent A, werden alle Daten, die Agent B an A sendet, schließlich an den Verbraucher übertragen.

Genügend $\mathcal{E}$'s bedeutet hier: Entweder schickt der Verbraucher ∞ viele $\mathcal{E}$ an Agent B, oder er schickt irgendwann nach $\mathcal{E}$ keine Verbotsnachricht $\mathcal{V}$ mehr. Dies wird durch das Prädikat $CInit$ beschrieben:

$$
\begin{aligned}
&CInit \ : \ \{\surd, \mathcal{E}, \mathcal{V}\}^{\omega} \rightarrow I\!B \\
&CInit(t) \ \triangleq \ \#\{\mathcal{E}\} \copyright t \ = \ \infty \ \vee \\
&\qquad \Big(\ \exists a \in \{\surd, \mathcal{E}, \mathcal{V}\}^{*}, \ b \in \{\surd, \mathcal{E}\}^{\omega} \ : \ t = a \ \circ \ \mathcal{E} \ \circ \ b \ \Big)
\end{aligned}
$$

Damit läßt sich diese Eigenschaft wie folgt formulieren:

$$
\begin{aligned}
&\forall \ x In A^{\omega}_{C}, \ y \in In A^{\omega}_{B} \ : \\
&\qquad \Big[\ CInit(x) \ \Rightarrow \ D \copyright fst \ HA(As0)(x, y) \ = \ D \copyright y \ \Big]
\end{aligned}
$$

4.2 Agent B

Agent B steht in direkter Verbindung mit dem Erzeuger und realisiert die Weiterleitung der hier erzeugten Daten. Diese Daten werden an Agent A weitergeleitet, wenn die Übertragung der Daten aktuell zugelassen ist. Zur Spezifikation dieses Verhaltens wird Agent B mit den beiden Eingabekanälen PnB und AnB und dem Ausgabekanal BnA, siehe auch Abbildung 5, definiert. Agent B wird in der gewohnten Weise als gezeitete, zustandsbehaftete stromverarbeitende Funktion formal spezifiziert, es werden für jeden der drei im Zustandsraum *BState* Zustände jeweils zunächst die komprimierte Tabelle und danach die zugehörigen Funktionsgleichungen angegeben.

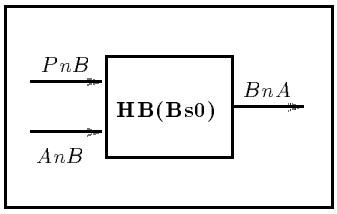

Abbildung 5: Agent B

Agent B wird mit der stromverarbeitenden Funktion HB spezifiziert:

$$
HB \ : \ BState \ \rightarrow \ \Big[\ (\ In B^{\omega}_{P} \times In B^{\omega}_{A} \ ) \ \rightarrow \ Out B^{\omega}_{A} \ \Big]
$$

Zustand Bs0: " Warten auf $C_{req} \in In B_{A}$ "

Im Anfangszustand $Bs0$ wartet Agent B auf die Erteilung der Erlaubnis zur Datenübertragung und damit auf die Nachricht C_{req} von Agent A. Sobald B diese Nachricht erhält, schickt er die Bestätigung C_{cnf} an A, und wechselt in den Folgezustand $Bs1$.

Der Erzeuger kann im Zustand $Bs0$ Daten erzeugen und an B schicken. Da diese jedoch noch nicht weitergeleitet werden können und nicht verloren gehen dürfen, werden sie gespeichert.

Als Tabelle ergibt sich damit:

$\boldsymbol{Bs = Bs0} \;\Rightarrow\quad \forall \alpha \in \{\surd, R_{req}\},\ d \in D\ :$

PnB	**AnB**	**BnA**	**Bs'**
$d\_$	α	$\surd$	$Bs0$
$d\_$	C_{req}	C_{cnf}	$Bs1$

$\forall\ \alpha \in \{\surd, R_{req}\},\ d \in D\ .\ \forall\ x \in InB_P^{\omega},\ y \in InB_A^{\omega}\ :$

$$\begin{aligned} HB(Bs0)\ (d\ \&\ x, \alpha\ \&\ y) &= \surd\ \&\ HB(Bs0)\ (d\ \&\ x, y)\ \wedge \\ HB(Bs0)\ (d\ \&\ x, C_{req}\ \&\ y) &= C_{cnf}\ \&\ HB(Bs1)(d\ \&\ x, y) \end{aligned}$$

Zustand Bs1: " Datenübertragung $d \in InB_P$; Warten auf $R_{req} \in InB_A$"

In Zustand $Bs1$ dürfen Daten an Agent A weitergeleitet werden, da der Verbraucher diese Erlaubnis erteilt und B die entsprechende Mitteilung von A bereits bestätigt hat. Diese Erlaubnis kann aber bereits auch wieder entzogen werden.

Agent B leitet die Daten, die vom Erzeuger geschickt werden, an A weiter. In jedem Zyklus soll mindestens ein Datum übertragen werden. Daher speichert B im Zustand $Bs1$ die Eingabe R_{req}, falls diese vorliegt, leitet die Eingabe d weiter und geht dann in den Folgezustand $Bs2$ über.

$\boldsymbol{Bs = Bs1} \;\Rightarrow\quad \forall \alpha \in \{\surd, C_{req}\},\ d \in D\ :$

PnB	**AnB**	**BnA**	**Bs'**
d	α	d	$Bs1$
d	$R_{req}\_$	d	$Bs2$

$\forall\ \alpha \in \{\surd, R_{req}\},\ d \in D\ .\ \forall\ x \in InB_P^{\omega},\ y \in InB_A^{\omega}\ :$

$$\begin{aligned} HB(Bs1)\ (d\ \&\ x, \alpha\ \&\ y) &= d\ \&\ HB(Bs1)\ (x, y)\ \wedge \\ HB(Bs1)\ (d\ \&\ x, R_{req}\ \&\ y) &= d\ \&\ HB(Bs2)\ (x, progress(R_{req}\ \&\ y)) \end{aligned}$$

Zustand Bs2: " Ausgabe der Bestätigung $R_{cnf} \in OutB_A$ "

Im Zustand $Bs2$ liegt die Mitteilung zum Entzug der Erlaubnis zur Datenübertragung von Agent A bei Agent B vor. Agent B schickt die Bestätigung R_{cnf} an A und geht dann wieder in den Anfangszustand $Bs0$ über. Im Zustand $Bs2$ kann B keine Daten an A weiterleiten; diese müssen gespeichert werden.

$\boldsymbol{Bs = Bs2} \;\Rightarrow\quad \forall \alpha \in \{\surd, C_{req}\},\ d \in D\ :$			
PnB	**AnB**	**BnA**	**Bs'**
$d\_$	R_{req}	R_{cnf}	$Bs0$
d	α	d	$Bs2$

$$\forall\ \alpha \in \{\surd, R_{req}\},\ d \in D\ .\ \forall\ x \in InB^{\omega}_{P},\ y \in InB^{\omega}_{A}\ :$$

$$\begin{aligned} HB(Bs2)\ (d\ \&\ x, R_{req}\ \&\ y) &= R_{cnf}\ \&\ HB(Bs0)(d\ \&\ x, y)\ \wedge \\ HB(Bs2)\ (d\ \&\ x, \alpha\ \&\ y) &= d\ \&\ HB(Bs2)(x, y) \end{aligned}$$

Zustand $Bs2$ ist ein Hilfszustand, der eingeführt wurde, um explizit zu verdeutlichen, daß in jedem Übertragungszyklus mindestens ein Datum übertragen wird. Als Eingabetupel liegt für diesen Zustand in jedem Fall (d, R_{req}) vor.

4.2.1 Eigenschaften von Agent B

Der so spezifizierte Agent B soll u.a. folgende Eigenschaften besitzen:

b1: Agent B schickt nie eine Bestätigung C_{cnf}, wenn von Agent A nicht vorher die Nachricht C_{req} geschickt wurde.

$$\forall\ x \in InB^{\omega}_{A},\ y \in InB^{\omega}_{P}\ :$$
$$\Big[\ \forall\ b \sqsubseteq HB(Bs0)(x, y)\ :\ \#\{C_{req}\}©x\ \geq\ \#\{C_{cnf}\}©b\ \Big]$$

Dies gilt entsprechend auch für R_{req} und R_{cnf}.

b2: Jedes C_{req} wird von Agent B schließlich mit der Bestätigung C_{cnf} beantwortet, wenn der Erzeuger nur genügend Daten liefert.

$$\forall\ x \in InB^{\omega}_{A},\ y \in InB^{\omega}_{P}\ :$$
$$\Big[\ \#y = \infty\ \Rightarrow\ \#\{C_{req}\}©x\ =\ \#\{C_{cnf}\}©HB(Bs0)(x, y)\ \Big]$$

Diese Eigenschaft gilt entsprechend für R_{req} und R_{cnf}.

b3: Zwischen zwei aufeinanderfolgenden Nachrichten R_{cnf} und C_{cnf} werden von Agent B keine Daten ausgegeben.

$$\forall\ x \in InB^{\omega}_{A},\ y \in InB^{\omega}_{P}\ :$$
$$\Big[\ \forall\ a, b \in OutB^{*}_{A},\ b \in^{*},\ c \in OutB^{\omega}_{A}\ :$$
$$a \circ R_{cnf} \circ b \circ C_{cnf} \circ c\ \sqsubseteq\ HB(Bs0)(x, y)\ \wedge\ \#\{R_{cnf}, C_{cnf}\}©b\ =\ 0\ \Rightarrow$$
$$\#\ D©b = 0\ \Big]$$

b4: Zwischen zwei aufeinanderfolgenden Nachrichten C_{cnf} und R_{cnf} wird von Agent B mindestens ein Datum ausgegeben, wenn der Erzeuger nur genügend Daten liefert.

$$\forall\, x \in InB_A^{\omega},\ y \in InB_P^{\omega}\ :$$
$$\Big[\ \forall\, a,b \in OutB_A^{*},\ c \in OutB_A^{\omega}\ :$$
$$a \circ C_{cnf} \circ b \circ R_{cnf} \circ c \ \sqsubseteq\ HB(Bs0)(x,y)\ \wedge\ \#\{R_{cnf},C_{cnf}\}\textcircled{c}b\ =\ 0$$
$$\wedge\ \#\ D = \infty\ \Rightarrow\ \#\ D\textcircled{c}b \geq 1\ \Big]$$

b5: Die Daten-Ausgabe von Agent B ist immer Präfix der Daten-Eingabe an B durch den Erzeuger.
Diese Eigenschaft entspricht Eigenschaft "**a2**" von Agent A und wird entsprechend mit der Funktion $HB(Bs0)$ formuliert.

b6: Alle vom Erzeuger gelieferten Daten werden schließlich von Agent B an Agent A weitergeleitet, vorausgesetzt, Agent A schickt genügend C_{req} an Agent B.

Die Forderung nach "genügend" C_{req}'s entspricht der Forderung zu der Anzahl der Eingaben $\mathcal{E}$ an Agent A von Eigenschaft "**a3**". Das Prädikat zur Formulierung dieser Eigenschaft erfolgt entsprechend zu "**a3**" mit $HB(Bs0)$ und den definierten Alphabeten.

4.3 Medium

Das Medium wird als Netzwerk, bestehend aus den vorher spezifizierten Agenten A und B, gemäß der funktionalen Spezifikation mit stromverarbeitenden Funktionen und den dafür definierten Kompositionsoperatoren spezifiziert.

Dafür werden die beiden Agentendefinitionen, also die beiden Funktionen HA und HB sukzessive durch sequentielle Komposition und Rückkopplung miteinander verknüpft, so daß schließlich nach außen hin das Medium als ein Agent mit einem Eingabe- und einem Ausgabekanal definiert ist. Nach außen hin werden auch die expliziten Zustände verborgen, so daß die Spezifikation des Mediums als reine stromverarbeitende Funktion erscheint.

Zunächst einmal läßt sich das Zusammenspiel von A, B, dem Erzeuger und dem Verbraucher mit der Funktion $F1$ (siehe Abbildung 6), definiert mit drei Eingabe- und zwei Ausgabekanälen und als sequentielle Komposition der Funktionen HA und HB angeben.

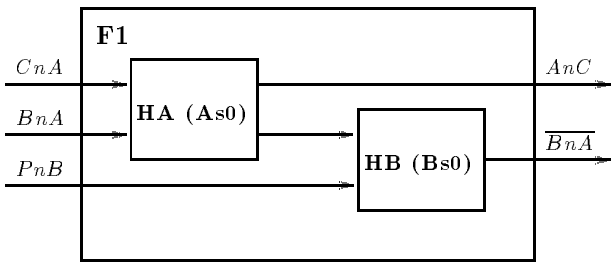

Abbildung 6: Sequentielle Komposition von A und B

Hierbei ist $F1$ wie folgt, mit den bereits definierten Alphabeten, definiert:

$$F1 \ : \ (\ InA_C^\omega \times InA_B^\omega \times InB_P^\omega \ ) \ \longrightarrow \ OutA_C^\omega \times OutB_A^\omega$$

$$\begin{array}{l} \forall \ x \in InA_C^\omega, \ y \in InA_B^\omega, \ z \in InB_P^\omega \ : \\ \quad F1(x,y,z) \ = \ (x',y') \\ \qquad \textbf{where} \quad x' \ = \ fst \ (\ HA \ (As0)(x,y) \ ) \\ \qquad\qquad\quad y' \ = \ HB(Bs0) \ (y'',z) \\ \qquad\qquad \textbf{where} \ \ y'' \ = \ snd(\ HA(As0) \ (x,y) \ ) \end{array}$$

Im nächsten Schritt wird zusätzlich Rückkopplung für die Realisierung der Verbindung zwischen den Agenten A und B durch die Kanal BnA ausgenutzt, siehe Abbildung 7.

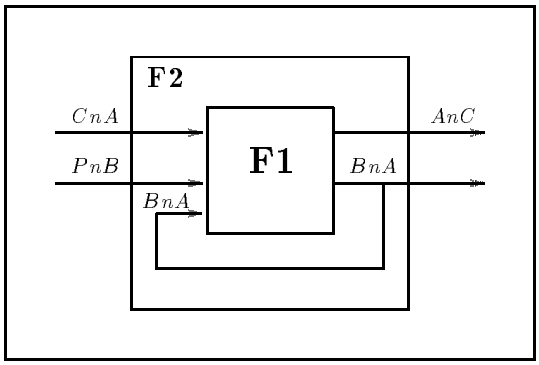

Abbildung 7: Rückkopplung über BnA

$F2$ ist dann wie folgt definiert:

$$F2 \ : \ InA_C^\omega \times InB_P^\omega \ \longrightarrow \ OutA_C^\omega \times OutB_A^\omega$$

$$\begin{array}{l} \forall \ x \in InA_C^\omega, \ y \in InB_P^\omega \ : \\ \quad F2 \ (x,y) \ = \ fix.g \\ \qquad \textbf{where} \ \ g(x,y) \ = \ F1 \ (x,y,z) \ \ \wedge \ \ z \in InA_B^\omega \end{array}$$

Im letzten Schritt wird das Medium als Agent mit 2 Eingabekanälen für die Eingaben durch den Verbraucher und den Erzeuger und einem Ausgabekanal für die Datenausgabe vom Medium an den Verbraucher definiert (siehe Abbildung 8). Die Realisierung des Mediums durch die beiden Agenten A und B und insbesondere die Rückkopplung verschwinden und sind nach außen nicht mehr sichtbar.

$$F3 \ : \ InA_C^\omega \times InB_P^\omega \ \longrightarrow \ OutA_C^\omega$$

$$\begin{array}{l} \forall \ x \in InA_P^\omega, \ y \in InB_C^\omega \ : \\ \quad F3 \ (x,y) \ = \ fst \ (\ F2 \ (x,y) \ ) \end{array}$$

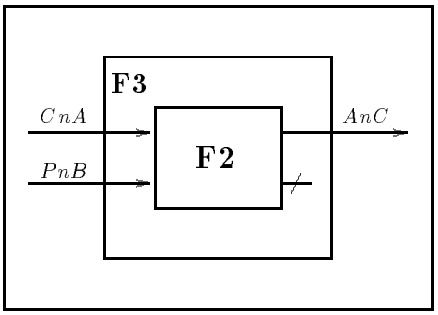

Abbildung 8: Spezifikation des Mediums durch $F3$

4.3.1 Eigenschaften des Mediums

Wie auch bereits bei den Agenten A und B sollen hier noch zwei Eigenschaften angegeben werden, die das so definierte Medium erfüllen soll. Diese Eigenschaften lassen sich jedoch i.w. aus den Eigenschaften der beiden Agenten ableiten.

m1: Der Verbraucher erhält nur solche Daten, die der Erzeuger auch gesendet hat. Die Datenausgabe des Mediums an den Verbraucher bildet immer ein Präfix der Daten-Eingabe des Erzeugers an das Medium.

$$\forall x \in InA_C^{\omega},\ y \in InB_P^{\omega}\ :\ \Big[\ \forall a \sqsubseteq F3\ (x,y)\ :\ D\copyright a\ \sqsubseteq\ D\copyright y\ \Big]$$

Diese Eigenschaft ergibt sich direkt aus **a2** von Agent A und **b5** von Agent B.

m2: Liefert der Erzeuger nur genügend viele Daten, und schickt der Verbraucher genügend $\mathcal{E}$'s an das Medium, so erhält der Verbraucher schließlich alle erzeugten Daten.

$$\forall x \in InA_C^{\infty},\ y \in InB_P^{\infty}\ :\ \Big[\ y = \infty\ \wedge\ CInit(x)\ \ \Rightarrow\ \ D\copyright F3(x,y)\ =\ D\copyright y\ \Big]$$

Das Prädikat $CInit$ wurde bereits bei der Formulierung von Eigenschaft **a3** des Agenten A definiert.

5 Darstellung der Agenten als endliche Automaten

In den vorangegangenen Abschnitten wurde anhand des vorliegenden Beispiels aufgezeigt, daß sich Tabellen gut eignen, um die relevanten Informationen in den einzelnen Teilen der Spezifikationen knapp aufzuschreiben und dennoch die angegebenen Formalisierungen als Spezifikationen mit den gemäß Focus standardmäßig vorgesehenen stromverarbeitenden Funktionen zu interpretieren.

Jede der angegebenen Tabellen beschreibt das für den jeweiligen Agenten geforderte Verhalten für einen der explizit definierten Zustände. Auf diese Weise wird das operationelle Vorgehen bei zustandsorientierten Spezifikationen besonders unterstrichen. Es kann jedoch auch von Interesse sein, eine Darstellung des Gesamtverhaltens eines Agenten anzugeben, die es ermöglicht, die für weitergehende Verfeinerungen und Verifikationen während des Entwicklungsvorganges eines Agenten wesentliche Informationen abzulesen. Zudem wird so die

Spezifikation einem weiteren Leserkreis erschlossen, dessen Spezialisierung auf diesen Formalismus ausgerichtet ist, so daß die angegebene Spezifikation nun für sie leichter zu lesen und auch zu verstehen ist.

Ein bekanntes mathematisches Modell für Agenten, die Informationen Zeichen für Zeichen einlesen und eine Ausgabe erzeugen, stellen die endlichen Automaten dar. Zudem werden Automaten oft als Beschreibungsmittel für Vorgänge genutzt, die sich auf eine bereits vorliegende Formalisierung in Form von Tabellen stützen. Zusätzlich werden endliche Automaten üblicherweise anschaulich durch Graphen dargestellt, so daß auch eine graphische Beschreibung leicht angegeben werden kann. Da die beiden vorher spezifizierten Agenten mit endlichen Zustandsräumen definiert sind, wird im folgenden die Darstellung als endlicher Automat in Form eines Graphen mit beschrifteten Kanten hergeleitet und angegeben (vgl. [Bra84] und [HU88]).

Üblicherweise wird ein endlicher Automat als Sechstupel $A = (I, O, Q, \delta, q_0, F)$ angegeben. Hierfür lassen sich die entsprechenden Informationen für die konkreten Automaten für die Agenten A und B mit geringen Anpassungen aus den bisher angegebenen Spezifikationen leicht ablesen. Die Ein- und Ausgabealphabete (I und O) und die Zustandsmenge (Q) ergeben sich mit der Spezifikation direkt. Der Anfangszustand (q_0) ist ebenfalls direkt bestimmbar. Die Zustandsübergangsfunktionen – die beiden Agenten sind deterministisch – kann leicht aus den bereits existierenden Tabellen hergeleitet werden. Da die Verhaltensweise des Mediums über prinzipiell unendliche Abläufe definiert wird, werden keine Zustände explizit als Endzustände ausgezeichnet.

Die graphische Darstellung eines endlichen Automaten erfolgt dadurch, daß die einzelnen Zustände als Kreise dargestellt werden, wobei der Anfangszustand besonders ausgezeichnet wird (hier als Doppelkreis). Zwischen den Zuständen, die gemäß der Spezifikation und der Zustandübergangsfunktion in Zustand/Folgezustand-Beziehung zueinander stehen, werden gerichtete Kanten gezogen. Diese Kanten können beschriftet werden. Hier besteht die Beschriftung aus der Angabe von Ein-/Ausgabenachrichten-Paaren bzw. Mengen solcher Paare, die diese Zustandsübergänge gemäß den Tabellen charakterisieren. Da die komprimierten Tabellen den endlichen Automaten zugrundeliegen, dürfen auch Variablen, deren Wertebereiche geeignet festgelegt sein müssen, in diesen Beschriftungen anstelle von Elementen aus den Ein-/Ausgabealphabeten benutzt werden.

5.1 Agent A

Der Automat zur Beschreibung von Agent A ist wie folgt definiert:

$$A_A \triangleq (\ I_A, O_A, Q_A, \delta_A, q_{0A}, F_A\ ) \qquad \text{mit} \quad \begin{array}{lcl} I_A & = & InA_C \times InA_B \\ O_A & = & OutA_C \times OutA_B \\ Q_A & = & AState \\ q_{0A} & = & As0 \\ F_A & = & \emptyset \end{array}$$

mit den bereits definierten Alphabeten $InA_C, InA_B, OutA_P$ und $OutA_B$ und der Zustandsmenge $AState$.

Eine Zustandsübergangsfunktion wird im allgemeinen wie folgt definiert:

$$\delta\ :\ Z \times E\ \rightarrow\ Z \times A \qquad \text{mit} \quad \begin{array}{lcl} Z & = & \text{endliche Zustandsmenge} \\ E & = & \text{Eingabe-Alphabet} \\ A & = & \text{Ausgabe-Alphabet} \end{array}$$

Für die Zustandübergangsfunktion zu Agent A ergibt sich dann gemäß den bisher angegebenen Spezifikationen sofort (mit $AState = \{As0, As1, As2, As3\}$):

$$\delta_A : AState \times (InA'_C \times InA_B) \rightarrow AState \times (OutA_C \times OutA_B)$$

wobei $InA'_C \triangleq InA_C \cup \{\mathcal{E}\_, \mathcal{V}\_\}$ gilt.

Die graphische Darstellung des Automaten für Agent A in der oben beschriebenen Art und Weise ist dann:

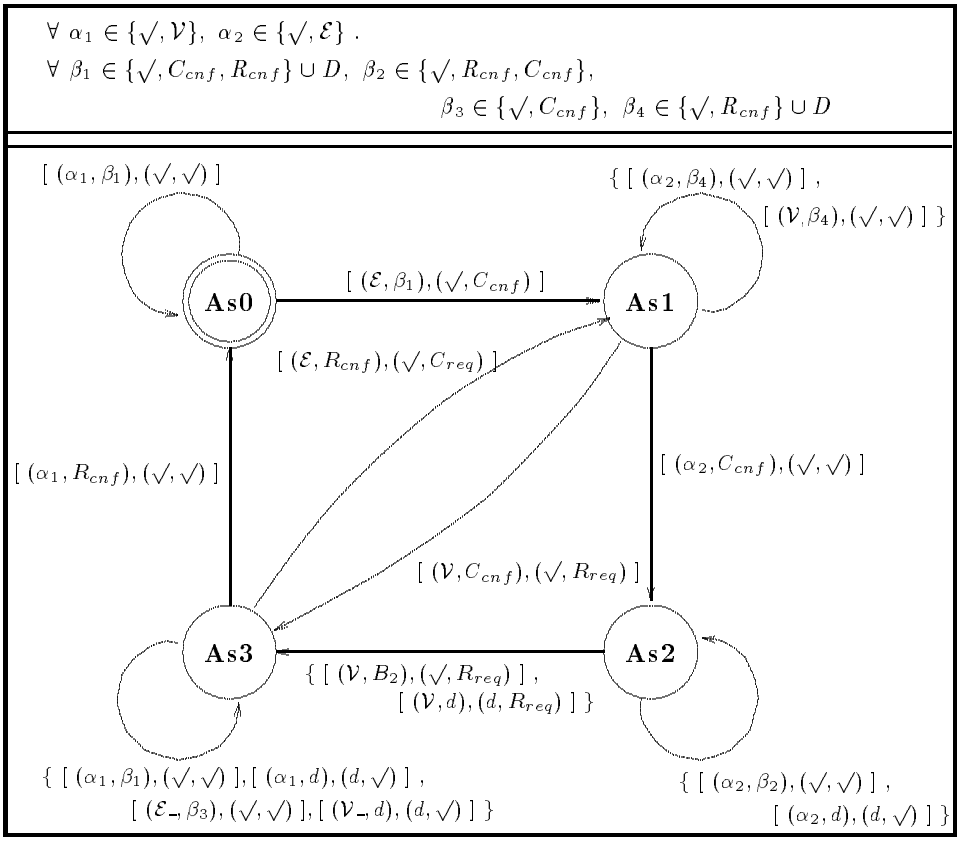

Abbildung 9: Automatendarstellung von Agent A

Zur vollständigen Definition müssen die Funktionsgleichungen zur Bestimmung der Zustandsübergangsfunktion angegeben werden:

$\forall x_1 \in \{\surd, \mathcal{V}\},\ x_2 \in \{\surd, \mathcal{E}\},\ d \in D$.
$\forall x_3 \in InA_B, x_4 \in \{\surd, C_{cnf}, R_{cnf}\}, x_5 \in \{\surd, C_{cnf}\}, x_6 \in \{\surd, R_{cnf}\} \cup D$:

$$\begin{aligned}
\delta_A(\ As0, (x_1, x_3)\ ) &= (\ As0, (\surd, \surd)\ ) \\
\delta_A(\ As0, (\mathcal{E}, x_3)\ ) &= (\ As1, (\surd, C_{cnf})\ ) \\
\delta_A(\ As1, (x_2, x_6)\ ) &= (\ As1, (\surd, \surd)\ ) \\
\delta_A(\ As1, (\mathcal{V}_, x_6)\ ) &= (\ As1, (\surd, \surd)\ ) \\
\delta_A(\ As1, (\mathcal{V}, C_{cnf})\ ) &= (\ As3, (\surd, R_{req})\ ) \\
\delta_A(\ As1, (x_2, C_{cnf})\ ) &= (\ As2, (\surd, \surd)\ )
\end{aligned}$$

$$
\begin{array}{lcl}
\delta_A(\ As2,(x_2,x_4)\ ) & = & (\ As2,(\surd,\surd)\ ) \\
\delta_A(\ As2,(x_2,d)\ ) & = & (\ As2,(d,\surd)\ ) \\
\delta_A(\ As2,(\mathcal{V},x_4)\ ) & = & (\ As3,(\surd,R_{req})\ ) \\
\delta_A(\ As2,(\mathcal{V},d)\ ) & = & (\ As3,(d,R_{req})\ ) \\
\delta_A(\ As3,(x_1,x_3)\ ) & = & (\ As3,(\surd,\surd)\ ) \\
\delta_A(\ As3,(x_1,d)\ ) & = & (\ As3,(d,\surd)\ ) \\
\delta_A(\ As3,(\mathcal{E}\_,x_5)\ ) & = & (\ As3,(\surd,\surd)\ ) \\
\delta_A(\ As3,(\mathcal{E}\_,d)\ ) & = & (\ As3,(d,\surd)\ ) \\
\delta_A(\ As3,(\mathcal{E},R_{cnf})\ ) & = & (\ As1,(\surd,C_{req})\ ) \\
\delta_A(\ As3,(x_1,R_{cnf})\ ) & = & (\ As0,(\surd,\surd)\ )
\end{array}
$$

5.2 Agent B

Der Automat zur Beschreibung von Agent B ist wie folgt definiert:

$$
A_B \triangleq (\ I_B,O_B,Q_B,\delta_B,q_{0B},F_B\ ) \qquad \text{mit} \quad \begin{array}{lcl} I_B & = & InB_P \times InB_A \\ O_B & = & OutB_A \\ Q_B & = & BState \\ q_{0B} & = & Bs0 \\ F_B & = & \emptyset \end{array}
$$

mit den bereits definierten Alphabeten InB_P, InB_A und $OutB_A$ und der Zustandsmenge $BState$.

Für die Zustandübergangsfunktion zu Agent B ergibt sich dann gemäß den bisher angegebenen Spezifikationen (mit $BState = \{Bs0, Bs1, Bs2\}$) sofort:

$$
\delta_B\ :\ BState \times (InB_P \times InB_A)\ \rightarrow\ BState \times OutB_A
$$

wobei $D' \triangleq D \cup \{d\_\}, \forall d \in D$ gilt.

Zur Definition der Zustandsübergangsfunktion ergeben sich die Funktionsgleichungen:

$\forall x_1 \in \{\surd, R_{req}\}, x_2 \in \{\surd, C_{req}\}, d \in D$:

$$
\begin{array}{lcl}
\delta_B(\ Bs0,(d,x_1)\ ) & = & (\ Bs0,\surd\ ) \\
\delta_B(\ Bs0,(d\_,C_{req})\ ) & = & (\ Bs1,C_{cnf}\ ) \\
\delta_B(\ Bs1,(d,x_2)\ ) & = & (\ Bs1,d\ ) \\
\delta_B(\ Bs1,(d,R_{req}\_)\ ) & = & (\ Bs2,d\ ) \\
\delta_B(\ Bs2,(d,x_2)\ ) & = & (\ Bs2,d\ ) \\
\delta_B(\ Bs2,(d\_,R_{req})\ ) & = & (\ Bs0,R_{cnf}\ )
\end{array}
$$

Die graphische Darstellung des Automaten für Agent B:

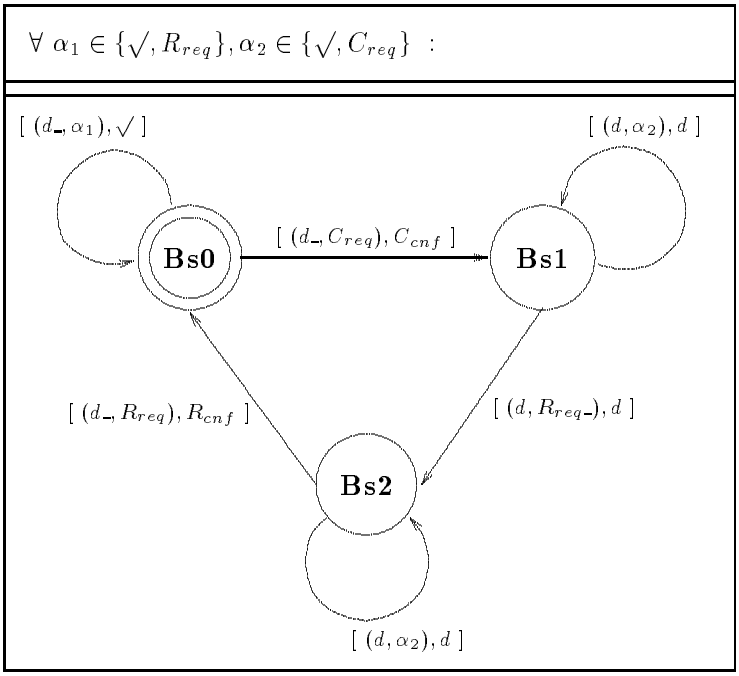

Abbildung 10: Automatendarstellung von Agent B

5.3 Herleitung von Zustandsprädikaten

Mit der Darstellung der Agenten als Automaten steht eine graphische Darstellung der Agenten zur Verfügung, die einen anschaulichen Überblick über das gesamte Ablaufverhalten eines Agenten liefert, und zudem die Anbindung eines weiteren bekannten Formalismus an die Spezifikationen mit stromverarbeitenden Funktionen darstellt. Zusätzlich wäre es jedoch wünschenswert, wenn auch Vorteile für die Formulierung von funktionalen Spezifikationen gemäß Standard-Focus aus den Darstellungen mit den hier angegebenen Beschreibungsformen gezogen werden könnten. Ein solcher Vorteil zeigt sich, wenn Prädikate über Kommunikationsgeschichten, die die explizit benutzten Zustände charakterisieren, formuliert werden sollen.

Mit den zustandsbehafteten, stromverarbeitenden Funktionen, wie sie hier benutzt werden, werden die Elemente des explizit definierten Zustandsraumes dazu benutzt, bestimmte Kommunikationgeschichten über die Eingaben durch Namen von zusätzlichen Parametern abzukürzen. Die standardmäßig in Focus benutzten stromverarbeitenden Funktionen sind ausschließlich über Bereichen von Strömen und somit nur über die Eingabeströme als Definitionsbereich definiert:

$$f\ :\ In_1^\omega \times \ldots \times In_n^\omega\ \rightarrow\ Out_1^\omega \times \ldots \times Out_m^\omega$$

Um von den in diesem Papier benutzten zustandsbehafteten, stromverarbeitenden Funktionen zu den üblichen stromverarbeitenden Funktionen zurückzukehren, müssen für die Agenten Prädikate über die Eingaben formuliert werden, mit denen die entsprechenden Zustände

charakterisiert werden können. Die Darstellung der Agenten mit Hilfe von Graphen mit beschrifteten Kanten erweist sich bei der Herleitung dieser Prädikate als äußerst nützlich, da die Eingaben, die zu den Zuständen führen, einfach an den Kanten abgelesen werden können.

Hier soll nun noch beispielhaft das Prädikat angegeben werden, das den Anfangszustand $As0$ von Agent A beschreibt:

$$
\begin{aligned}
P_{As0}\,(z_1, z_2) \;\triangleq\; & \forall z_1 \in InA_C^{\omega},\ z_2 \in InA_B^{\omega},\ t_1 \in\in InA_C^{*},\ t_3 \in InA_B^{*}\ : \\
& [\ z_1 = <>\ \wedge\ z_2 = <>\ ]\ \vee \\
& [\ z_1 = t_1 \& t_2\ \wedge\ t_2 \in \{\surd, \mathcal{V}\}\ \wedge \\
& \quad z_2 = t_3 \& t_4\ \wedge\ t_4 \in InA_B\ \wedge\ P_{As0}\,(t_1, t_3)\ ]\ \vee \\
& [\ z_1 = t_1 \& t_2\ \wedge\ t_2 \in \{\surd, \mathcal{V}\}\ \wedge\ z_2 = t_3 \& R_{cnf}\ \wedge\ P_{As3}\,(t_1, t_3)\ ]
\end{aligned}
$$

P_{As3} sei das entsprechend formulierte Prädikat zur Charakterisierung von Zustand $As3$.

Unter Zuhilfenahme derartiger Prädikate können dann auch die Eigenschaften, die die jeweilige Spezifikation eines Agenten erfüllen müssen, losgelöst von dem expliziten Zustandsraum, angegeben werden. Die für Agent A genannte Eigenschaft **a3** ist beispielsweise allgemein gehalten und stützt sich nicht auf spezielle Zustände ab. Da aber das Verhalten von Agent A bzgl. der Weiterleitung der Nachricht C_{req} als Reaktion auf $\mathcal{E}$ vor allem in den Zuständen $As0$ und $As3$ von Interesse ist, kann **a3** in eine diese beiden Zustände betreffende Eigenschaft umformuliert werden. Als Prädikat über Ströme und ohne explizit definierten Zustandsraum ergibt sich dann:

a3': In den Zuständen $As0$ und $As3$ produziert Agent A zu einer Eingabe $\mathcal{E}$ durch den Verbraucher in jedem Fall die Ausgabe C_{req} an Agent B.

$$
\begin{aligned}
& \forall\ z_1 \in InA_C^{\omega},\ z_2 \in InA_B^{\omega}\ .\ \forall\ t_1 \in InA_C^{*},\ t_3 \in InA_B^{*}\ : \\
& \quad [\ (\ z_1 = t_1\ \circ\ \mathcal{E}\ \&\ t_2\ \wedge\ z_2 = t_3\ \circ\ t_4\ )\ \wedge\ (\ P_{As0}\,(t_1, t_3)\ \vee\ P_{As3}\,(t_1, t_3)\ ) \\
& \qquad \wedge\ (\ HA'(z_1, z_2) = A_1\ \circ\ A_2\ \wedge\ A_1\ =\ HA'(t_1, t_3)\ )\ ] \\
& \quad \Rightarrow\ \ \#\{C_{req}\}\ \copyright\ A_2\ \geq\ 1
\end{aligned}
$$

Hierbei gilt: $HA'\ :\ InA_C^{\omega} \times InA_B^{\omega}\ \rightarrow\ OutA_C^{\omega} \times OutA_B^{\omega}$

6 Bemerkungen und Ausblick

In der vorliegenden Arbeit wurde eine vollständige funktionale Spezifikation eines einfachen Kommunikationsprotokolls erarbeitet. Es wurde beispielhaft gezeigt, wie weitere Spezifikationsformalismen neben den in Focus verwendeten stromverarbeitenden Funktionen zur Formulierung des in der informellen Beschreibung geforderten Verhaltens herangezogen werden können. Einen Überblick über weitere im Zusammenhang mit Focus erstellte Fallstudien liefern [BDD+92a] und [BFG+94].

Es wurde eine erste, formale Beschreibung der beiden Agenten, die das Medium in dem angegebenen Agenten-Netzwerk realisieren, durch Tabellen vorgenommen. Hierbei wurde für jeden der definierten Zustände jeweils eine Tabelle angegeben, die die für den Zustand relevanten Informationen knapp und leicht verständlich enthält. Die Anbindung an die stromverarbeitenden Funktionen erfolgte dadurch, daß die Tabelleneinträge in Funktionsgleichungen mit den entsprechenden aktuellen Ein- und Ausgaben umgesetzt wurden. Auf dieser Basis

ist es dann möglich, Prädikate über Eigenschaften der Funktionen zu formulieren, die die wesentlichen Anforderungen an das Protokoll beschreiben. Ebenso wurde das gesamte Medium als Netzwerk bestehend aus den sequentiell und mit Rückkopplung verknüpften Agenten A und B auf der Basis der stromverarbeitenden Funktionen spezifiziert.

Mit der Umsetzung der Tabellenspezifikationen in eine graphische Darstellung konnte gezeigt werden, daß auch eine Formalisierung durch endliche Automaten in dieser Fallstudie leicht an den Focus-Formalismus angebunden werden kann. Diese Darstellung erleichtert beispielsweise die Formulierung der zustandsbeschreibenden Prädikate auf der Basis der Kommunikationsgeschichten.

Auf diese Weise wurde beispielhaft gezeigt, daß erste Formalisierungen von informell beschriebenen Systemen auch von Nicht-Spezialisten der Focus-Notationen geschrieben werden können. Weitere Formalismen wie die hier entwickelten Tabellen können einfach und adäquat an die stromverarbeitenden Funktionen angebunden werden.

In diesem Zusammenhang stellen sich jedoch weitere interessante Aufgaben. Der hier an einem Beispiel entwickelte Tabellenformalismus sollte in eine allgemeine Form gebracht werden. Hiermit kann dann die Transformation der Tabellen in Funktionsgleichungen formal beschrieben werden. Ebenso sollte es mit der allgemeinen Tabellenform möglich sein, verschieden Tabellenarten zu klassifizieren. In dem hier gezeigten Beispiel wurden ausschließlich Aktionen bzw. Variablen, die Aktionen repräsentieren, als Tabelleneinträge benutzt. Es ist jedoch auch vorstellbar, daß der Entwickler z.B. Prädikate als Einträge verwenden möchte. Hierbei sollten auch Vergleiche mit anderen Formalisierungen von Tabellen, wie beispielsweise [Par92] und [Jan93], gezogen werden.

Eine weitere, interessante Frage besteht darin, ob sich die verschiedenen Darstellungen der Spezifikationen bei der Durchführung formaler Beweise als hilfreich erweisen. Hierbei gesammelte Erfahrungen können dazu verwendet werden, methodische Hinweise bzgl. der Nutzung verschiedener Formalismen zur korrekten Entwicklung eines Systems mit Focus anzugeben.

Danksagung

Zunächst möchte ich mich bei Manfred Broy und Franz Regensburger für Ihre Kommentare zu der Erstellung dieser Arbeit bedanken. Mein besonderer Dank gilt jedoch Bernhard Schätz. Aus den Diskussionen mit ihm ergaben sich zahlreiche Verbesserungen.

Literatur

[BD90] M. Broy and C. Dendorfer.
Functional Modelling of Operating System Structures by Timed Higher Order Stream Processing Functions.
SFB-Bericht 342/22/90 A, Technische Universität München, November 1990.

[BDD+92a] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber.
Summary of Case Studies in FOCUS — a Design Method for Distributed Systems.
SFB-Bericht 342/3/92 A, Technische Universität München, January 1992.

[BDD+92b] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber.
The Design of Distributed Systems — An Introduction to FOCUS.
SFB-Bericht 342/2/92 A, Technische Universität München, January 1992.

[BDD+93] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber.
The Design of Distributed Systems – An Introduction to FOCUS.

SFB-Bericht Nr. 342/2-2/92 A, Technische Universität München, Institut für Informatik, January 1993.

[BFG+94] M. Broy, M. Fuchs, T. F. Gritzner, B. Schätz, K. Spies, and K. Stølen.
Summary of Case Studies in FOCUS — a Design Method for Distributed Systems.
SFB-Bericht 342/13/94 A, Technische Universität München, June 1994.

[BHS91] F. Belina, D. Hogrefe, and A. Sarma.
SDL with Applications from Protocol Specification.
Hanser, 1991.

[Bra84] W. Brauer.
Automatentheorie: eine Einführung in die Technik endlicher Automaten.
Teubner, 1984.

[Bro89] M. Broy.
Functional Specification of Time Sensitive Communicating Systems.
In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems*, pages 151 – 179. Springer, LNCS 430, 1989.

[DW92] C. Dendorfer and R. Weber.
Development and Implementation of a Communication Protocol – An Exercise in Focus.
SFB-Report 342/4/92 A, Technische Universität München, März 1992.

[Fuc93] M. Fuchs.
Funktionale Spezifikation einer Geschwindigkeitsregelung.
SFB-Bericht 342/1/93 B, Technische Universität München, January 1993.

[GG92] U. Goltz and N. Götz.
Modelling a Simple Communication Protocol in a Language with Action Refinement.
unpublished, draft paper, 1992.

[HU88] John E. Hopcroft and Jeffrey D. Ullman.
Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie.
Addison-Wesley, 1988.

[JA90] B.N. Jain and A.K. Agrawala.
Open Systems Interconnection: Its Architectures and Protocols.
Elsevier, 1990.

[Jan93] R. Janicki.
Towards a Formal Semantics of Tables.
Technical Report 264, CRL, September 1993.

[Par92] D.L. Parnas.
Tabular Representation of Relations.
Technical Report 260, CRL, October 1992.

[PP92] G. M. Pinna and A. Poigné.
On the Nature of Events.
In I.M. Havel and V. Koubek, editors, *Proceedings of the 17th Symposium on the Mathematical Foundation of Computer Science (MFCS'92)*, pages 430 – 441. Springer, LNCS 629, 1992.