Entwurf und Verifikation nachrichtenbasierter verteilter Algorithmen durch verteilende Verfeinerung

Bodo Hohberg, Wolfgang Reisig, Bixia Wu
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany

Abstract

Um Entwurf und Verifikation komplizierter verteilter Algorithmen leichter und verständlicher zu machen, wird oft eine Verfeinerungsmethode verwendet. Dabei wird ein einfacher Algorithmus, der gewünschte Eigenschaften erfüllt, schrittweise zu einem komplizierten Algorithmus verfeinert. In jedem Schritt sollen die gewünschten Eigenschaften erhalten bleiben.

Für nachrichtenbasierte verteilte Algorithmen, die durch Petrinetze modelliert werden, haben wir eine neue Verfeinerungsmethmode entwickelt. Wir beginnen mit einem Anfangsalgorithmus, der Aktionen enthält, die gemeinsame Aufgaben mehrerer Agenten beschreiben. In jedem Schritt verfeinern wir eine dieser Aktionen zu einem Netz, das nur solche Aktionen enthält, die die Aufgaben einzelner Agenten beschreiben. Jeder Schritt ist also eine Verteilung einer unverteilten Aktion, also eine verteilende Verfeinerung.

Die Arbeit klärt den Zusammenhang von Eigenschaften des Verfeinerungsnetzes und den bei der Verfeinerung gültig bleibenden Eigenschaften des verfeinerten Algorithmus. Hierbei sind Kausalitäten im Verfeinerungsnetz von entscheidender Bedeutung.

Die Anwendung der Methode wird in der Arbeit an anschaulichen Beispielen demonstriert.

Stichworte: Petrinetz, Transitionsverfeinerung, verteilte Abläufe, Halbordnungssemantik, verteilte Algorithmen, Kausalität, Nachrichten, Agenten, Entwurf, Verifikation.

1. Einführung

Durch die zur Zeit immer weiter fortschreitende Arbeitsteilung in den verschiedensten Bereichen und die Entwicklung von Kommunikationstechnik, die eine Arbeitsteilung technisch unterstützt, wird die Frage nach einer theoretischen Grundlage für das korrekte Verteilen von Aufgaben immer wichtiger. Wie müssen Teilaufgaben z.B. so auf verschiedene Personen verteilt werden, dass sie gemeinsam ein vorgegebenes Ziel erreichen, ohne sich ununterbrochen neu abzustimmen. Ein noch relativ einfacher Bereich von Aufgaben lässt sich durch 1-beschränkte S/T-Netze modellieren. Aber noch nicht einmal für diese noch überschaubare Netzklasse liegen bisher theoretische Aussagen vor, wann und wie gemeinsame Aufgaben von Agenten auf die einzelnen Agenten verteilt werden können. In dieser Arbeit werden wir dieses Problem untersuchen.

Im Folgenden werden wir zunächst anhand eines Beispiels die Ideen und den wesentlichen Inhalt der Arbeit erläutern. Das Beispiel nennen wir *Dienstreise* (Abb. 1.1). Ein *Chef* und sein *Mitarbeiter* machen gemeinsam eine Dienstreise, um z.B. Geschäftsverträge abzuschließen. Wenn die beiden bereit sind, dann werden sie zusammen abfahren und gemeinsam Verträge abschließen. Anschließend werden sie zusammen zurückkommen.

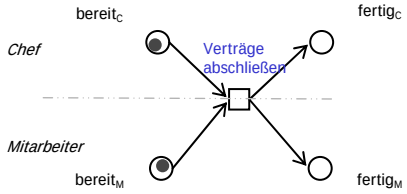


Abb. 1.1 *Gemeinsame Dienstreise* (System Σ_1)

Angenommen, beide sollen getrennt die Dienstreise antreten, aber dasselbe Ziel wie bei der gemeinsamen Aktion erreichen. In diesem Fall hat jeder einen Teil der gesamten Aufgaben zu erledigen. Jeder wird abfahren, wenn er bereit ist. Jeder wird zurückkommen, wenn er mit seiner eigenen Teilaufgabe fertig ist. In Abb. 1.2 ist ein System *getrennte Diestreise* angegeben. Also, in dieser Arbeit möchten wir eine gemeinsame Aktion von mehreren Agenten zu lokalen Aktionen einzelner Agenten verteilen.

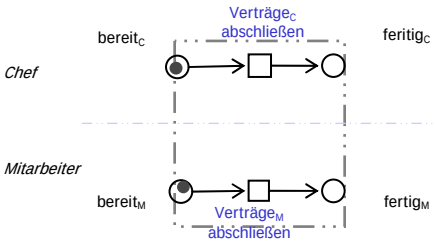


Abb. 1.2 *Getrennte Diestreise* (System Σ_1')

Was soll beim Verteilen erfüllt werden? Eine Mindestanforderung ist, jeder muss seine Teilaufgabe machen, d.h. es soll nicht vorkommen, dass der eine seine Teilaufgabe macht aber der andere nicht. Bei der *Diestreise* sollen entweder beide oder keiner die Diestreise antreten. Diese Bedingung ist die sogenannte *Blockbedingung* (siehe [Peu01]). Im Beispiel von Abb. 1.2 ist offenbar die Blockbedingung erfüllt.

In Abb. 1.3 ist ein etwas komplizierteres System dargestellt. Der *Chef* muss vor der Reise noch etwas vorbereiten. Ausserdem kann er noch eine andere Aufgabe erledigen.

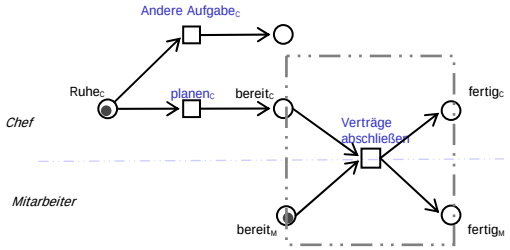


Abb. 1.3 Ein weiteres System Σ_2

In diesem Fall wird die Blockbedingung nicht erfüllt, wenn wir die gemeinsame Aktion noch so verteilten wie in Abb. 1.2, weil es dann passieren kann, dass der *Chef* eine andere Aufgabe macht und der Mitarbeiter alleine verreist. In diesem Fall müssen wir die gemeinsame Aktion wie folgt verteilen (Abb. 1.4): Wenn der *Chef* tatsächlich verreist, soll er an seinen *Mitarbeiter* eine SMS schicken. Der *Mitarbeiter* kann nur verreisen, wenn er diese SMS bekommen hat. Dann gilt die Blockbedingung wieder.

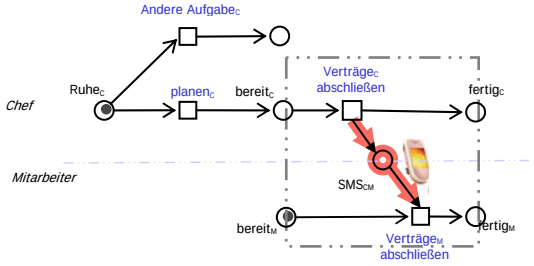


Abb. 1.4 Eine Verteilung Σ_2' vom System Σ_2 in Abb. 1.3

Also, ob die Blockbedingung erfüllt ist, das hängt von der Umgebung der zu verteilenden Aktion ab. Durch Nachrichten können wir erreichen, dass die Blockbedingung erfüllt ist.

Außer der Blockbedingung müssen wir noch beachten, dass alle gewünschten Eigenschaften des Systems erhalten bleiben. Angenommen, im System in Abb. 1.3 ist die Aufgabe des *Mitarbeiters* die Verträge zu behandeln und die Aufgabe des *Chefs* zu unterschreiben. Das System hat damit folgende Eigenschaft: Wenn der *Chef* unterschrieben hat, dann hat der *Mitarbeiter* sicherlich die Verträge schon behandelt. Angenommen, diese Eigenschaft ist gewünscht und soll nach dem Verteilen erhalten bleiben. Die Verteilung in Abb. 1.4 erfüllt zwar die Blockbedingung, aber die oben genannte Eigenschaft bleibt nicht erhalten, weil es sein kann, dass der *Chef* schon unterschrieben aber der *Mitarbeiter* die Verträge noch nicht behandelt hat.

In Abb. 1.5 ist eine Verteilung angegeben, die die Blockbedingung erfüllt und das Erhaltenbleiben der Eigenschaft sichert. Das Erhaltenbleiben der Eigenschaften wird durch Reihenfolge der Aktionen unterschiedlicher Agenten garantiert. Die Reihenfolge der Aktionen wird wiederum durch Kommunikation zwischen den Agenten realisiert.

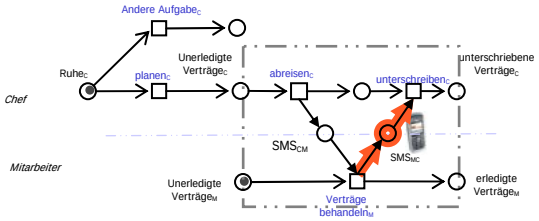


Abb. 1.5 Eine weitere Verteilung Σ_2'' von Σ_2 in Abb. 1.3

In dieser Arbeit suchen wir einen Verfeinerungsbegriff, mit dem wir einen Verfeinerungsschritt zum Verteilen einer unverteilten Transition verifizieren können. Daher muss dieser Verfeinerungsbegriff eigenschafts-haltend sein. Ein Ziel dieser Arbeit ist also eine Spezialisierung der Transitionsverfeinerung zu finden, die das Verteilen erlaubt und eigenschafts-haltend ist. Die Transitionsverfeinerung mit der Blockbedingung in der Literatur [Peu01] erlaubt schon das Verteilen. Wir ergänzen diese Transitionsverfeinerung um notwendige Synchronisationsbedingungen und können dann verteilende Verfeinerungsschritte verifizieren.

Ein weiteres Ziel dieser Arbeit ist zu untersuchen, wie man eine unverteilte Transition verteilen soll, damit die Blockbedingung erfüllt ist. Wir haben herausgefunden, dass die Gültigkeit der Blockbedingung außer mit der Umgebung der zu verteilenden Transition, insbesondere noch mit den Kausalitäten im Verfeinerungsnetz zusammen hängt. Wir beweisen mehrere Kriterien, mit deren Hilfe bestimmt werden kann, welche Verfeinerungsnetze bei einer konkreten Umgebung die Gültigkeit der Blockbedingung garantieren. Bei der Auswahl der Verfeinungsnetze geht es darum, mit einem Minimum an Kommunikation auszukommen

Natürlich interessieren wir uns weiter für folgendes Problem: Welche Bedingungen soll das Verfeinerungsnetz erfüllen, damit eine gewünschte Eigenschaft erhalten bleibt. Genauer formuliert, welche Kausalitäten des Verfeinerungsnetzes garantieren das Erhaltenbleiben einer gewünschten Eigenschaft.

Unser eigentlicher Zweck ist, eine Methode zum Entwurf und zur Verifikation verteilter Algorithmen durch verteilende Verfeinerung zu finden.

Für komplexe verteilte Algorithmen ist es leichter, durch Verfeinerung schrittweise den Algorithmus zu verifizieren, als direkt den Algorithmus zu verifizieren.

Bei unserer verteilenden Verfeinerung verifizieren wir die das Erhaltenbleiben der gewünschten Eigenschaften des gesamten Systems durch den Nachweis der Gültigkeit der erforderlichen Reihenfolge-Bedingungen im lokalen Verfeinerungsnetz. Die Gültigkeit einer Reihenfolge-Bedingung im lokalen Verfeinerungsnetz ist meist leicht nachzuweisen. Daher ist die Verifikation mit unserer Methode leicht durchzuführen.

Bei unserer Methode ist die Reihenfolge der Aktionen entscheidend. Petrinetze und deren verteilten Abläufe zeigen gerade die kausalen Ordnungen der Aktionen. Also, Petrinetze bieteten genau die richtige Grundlage für diese Arbeit.

Die Arbeit ist folgendermaßen strukturiert: Im Abschnitt 2 werden die notwendigen Grundlagen der Arbeit angegeben. Insbesondere wird der Begriff Kausalitäten im System eingeführt. Im Abschnitt 3 geht es um Transitionsverfeinerung für ein Petrinetz. Zunächst wird die Transitionverfeinerung mit Blockbedingung definiert und danach wird analysiert, welche Eigenschaften bei dieser Verfeinerung erhalten bleiben und welche nicht. In Abschnitt 4 wird der Begriff Agentensystem eingeführt. Direkt danach wird im Abschnitt 5 die verteilende Verfeinerung behandelt. Der Begriff Agentensystem wird direkt vor dem Abschnitt verteilende Verfeinerung eingeführt, damit der enge Zusammenhang zwischen den Definitionen Agentensystem und verteilende Verfeinerung klar wird. Im Abschnitt 6 geht es um Kriterien für die Blockbedingung. Und im Abschnitt 7 wird die Methode zum Entwurf und zur Verifikation verteilter Algorithmen durch verteilende Verfeinerung und zwei Anwendungsbeispiele vorgestellt. An dem Beispiel *Fahrradausleihe* kann man gleichzeitig sehen, wie ein Service als ein Agent abgespalten werden kann und wie dabei die Schnittstelle des Services geplant wird.

2. Grundlagen

In diesem Abschnitt werden wir die für diese Arbeit notwendigen Grundlagen vorstellen – Petrinetzmodell verteilter Algorithmen, ihre Halbordnungssemantik und ihre temporal-logischen Eigenschaften und Kausalitäten im Netz

2.1 Petrinetze

In diesem Abschnitt definieren wir elementare Petrinetze (S/T-Netze ohne Kantengewichtung) (vgl. [Peu01]). Der Leser, der mit dieser Petrinetzklasse vertraut ist (z.B. aus [Rei98]), kann diesen Abschnitt überspringen.

Definition 2.1.1(*Petrinetz*)
Ein *Petrinetz* (kurz: *Netz*) $N=(P,T,F)$ besteht aus zwei nicht-leeren, disjunkten Mengen P und T und der Flussrelation $F \subseteq (P\times T) \cup (T\times P)$, so dass für jedes $t\in T$ die Mengen $\{ p\in P \mid (p,t) \in F \}$ und $\{ p\in P \mid (t,p) \in F \}$ nicht leer und endlich sind. Die Elemente von P, T und F nennen wir Stellen, Transitionen bzw. Kanten des Netzes.

Für jedes $x\in P \cup T$ nennen wir die Menge $^\bullet x = \{ y\in P \cup T \mid (y,x) \in F \}$ den Vorbereich von x und die Menge $x^\bullet = \{ y\in P \cup T \mid (x,y) \in F \}$ den Nachbereich von x.

Für eine technisch angenehme Präsentation halten wir in dieser Arbeit eine universelle Stellenmenge $\wp$ fest und nehmen an, dass alle in dieser Arbeit betrachteten Petrinetze nur Stellen aus dieser Menge haben, d.h. $P\subseteq \wp$.

Definition 2.1.2 (*Markierung*)
Eine Markierung von $\wp$ ist eine Funktion $M: \wp \to \mathbb{N}$, die höchstens endlich vielen Stellen eine von Null verschiedene Zahl zuordnet. Eine Markierung des Netzes N ist eine Markierung, für die $M(p)=0$ für alle $p\in \wp \setminus P$ gilt.

Jeder Transition t ordnen wir die beiden Markierungen t^- und t^+ zu, die folgendermaßen definiert sind:

$$t^-(p)=\begin{cases}1, p\in {}^\bullet t\\ 0, sonst\end{cases} \quad \text{und} \quad t^+(p)=\begin{cases}1, p\in t^\bullet\\ 0, sonst\end{cases}$$

Wir definieren die Addition von Markierungen punktweise, d.h. für zwei Markierungen M_1 und M_2 des Netzes N definieren wir die Summe $M_1 + M_2$: $\wp \to \mathbb{N}$ durch $(M_1+M_2)(p)=M_1(p)+M_2(p)$ für alle $p\in \wp$. Analog vergleichen wir Markierungen punktweise und schreiben $M_1 \le M_2$, falls $M_1(p) \le M_2(p)$ für alle $p\in \wp$.

Eine Markierung wird durch das Schalten einer Transition verändert. Eine Transition t ist *aktiviert* in der Markierung M, wenn auf jeder Stelle im Vorbereich von t mindestens eine Marke liegt, d.h. $t^- \le M$. Eine aktivierte Transition kann schalten. Das führt zur Markierung M', die durch $M' + t^- = M + t^+$ gegeben ist (Es wird vermieden, eine partielle Subtraktion von Markierungen zu definieren). Den gerade beschriebenen Vorgang nennen wir *Schritt* und bezeichnen ihn durch $M \xrightarrow{t} M'$. Eine Folge von Schritten $M_0 \xrightarrow{t_1} M_1 \ \dots \ \xrightarrow{t_n} M_n$ heißt *ein sequentieller Ablauf in N.* Eine Markierung M' ist von der Markierung M aus *erreichbar*, wenn es eine endliche, ggf. leere Folge von Schritten $M_0 \xrightarrow{t_1} M_1 \ \dots \ \xrightarrow{t_n} M_n$ mit $M = M_0$ und $M' = M_n$ gibt.

Zwei verschiedene Transitionen t_1 und t_2 sind *konkurrent* zueinander, wenn ihre Vorbereiche nicht disjunkt sind. Eine Transition t_1 ist eine *aktivierte* konkurrente Transition zu einer anderen Transition t_2 in einer Markierung M, wenn sie konkurrent zu t_2 ist und beide in M aktiviert sind.

2.2 Verteilte Abläufe

Wir definieren eine Halbordnungssemantik für Petrinetze mit Hilfe verteilter Abläufe. Ein verteilter Ablauf wird durch ein Kausalnetz dargestellt. Ein Kausalnetz ist ein Petrinetz mit besonderen Eigenschaften. Es ist azyklisch, d.h. die Flußrelation enthält keinen Kreis. Außerdem ist ein Kausalnetz an Stellen unverzweigt, d.h. jede Stelle hat höchstens eine eingehende und eine ausgehende Kante. Die Stellen eines Kausalnetzes nennen wir Bedingungen und die Transitionen nennen wir Ereignisse.

Definition 2.2.1 (*Kausalnetz*)
Ein Netz $K = (B, E, <)$ ist ein Kausalnetz, wenn folgende Bedingungen gelten:

i. K ist an den Stellen unverzweigt, d.h. $| {}^\bullet b | \le 1$ und $|b^\bullet| \le 1$ für jede Bedingung $b\in B$.

ii. Die Menge $^\circ K = \{ b\in B \mid {}^\bullet b = \emptyset \}$ ist nicht leer und endlich.

iii. Die Relation $<$ ist azyklisch. Dadurch ist die reflexive transitive Hülle von $<$ eine Halbordnung, die wir mit $\le$ bezeichnen.

iv. Jedes Element von $B\cup E$ hat nur endlich viele Vorgänger bzgl. der Halbordnung $\le$.

Wir definieren nun einige Begriffe, um Aussagen über Kausalnetze zu treffen. Ein Schnitt ist eine maximale aber endliche Menge von paarweise unabhängigen Bedingungen bzgl. der Kausalordnung $\leq$ eines Kausalnetzes. Zwei von einander unabhängige Ereignisse bzgl. $\leq$ nennen wir auch zueinander nebenläufig. Die transitive Hülle von $\prec$ bezeichnen wir mit $<$.

Definition 2.2.2 (*co-Menge, Schnitt, nebenläufige Ereignisse*)

Sei $K = (B, E, \prec)$ ein Kausalnetz. Weiter seien $x,y \in B \cup E$.
$x \underline{\text{co}}\ y$ dgw. $\neg(x<y)$ und $\neg(y<x)$.

Eine Menge $C \subseteq B$ ist eine *co-Menge* von K, gdw. für je zwei Bedingungen $b_1, b_2 \in C$ gilt $b_1 \underline{\text{co}}\ b_2$.
Eine maximale, endliche *co*-Menge C ist ein *Schnitt*.
Zwei verschiedende Ereignisse $e_1, e_2 \in E$ mit $e_1 \underline{\text{co}}\ e_2$ heißt zueinander *nebenläufig* (*parallel*).
Kausalität wird also auch benötig, um über nebenläufige (parallele) und nicht nebenläufige Ereignisse Aussagen machen zu können.

Für jedes Ereignis $e \in E$ sind die Menge $^\bullet e$ und $e^\bullet$ *co*-Mengen. Die Menge $^\circ K = \{ b \in B \mid {}^\bullet b = \emptyset \}$ ist ein Schnitt. Diesen Schnitt nennen wir den Anfangsschnitt von K. Die Menge $K^\circ = \{ b \in B \mid b^\bullet = \emptyset \}$ ist eine *co*-Menge. Wir nennen K° das Ende des Kausalnetzes. Das Ende eines Kausalnetzes K ist genau dann ein Schnitt, wenn K endlich ist.

Ein *co*-Menge M' eines Kausalnetzes K ist durch Schalten des Ereignisses $e \in E$ von einer *co*-Menge M erreichbar, wenn ${}^\bullet e \subseteq M$ und $e^\bullet \cap M = \emptyset$ und $M' = M \setminus {}^\bullet e \cup e^\bullet$ gilt. Wir schreiben $M \xrightarrow{e} M'$. Wenn ein Ereignis e mit $M \xrightarrow{e} M'$ existiert, schreiben wir $M \rightarrow M'$ und für die reflexive transitive Hülle von $\rightarrow$ schreiben wir $\rightarrow^*$. Ein Schnitt C' (eine maximale *co*-Menge) ist von einem Schnitt C eines Kausalnetzes erreichbar, wenn $C \rightarrow^* C'$ gilt. (Erreichbarkeit wird benutzt, um über temporal-logische Eigenschaften reden zu können.)

Um einen verteilten Ablauf eines Systems zu modellieren, brauchen wir noch eine Funktion, die jedes Ereignis des Kausalnetzes als Schalten einer Transition des Systems identifiziert und jede Bedingung mit einer Stelle des Systems.

Definition 2.2.3 (*N-Beschriftung*)

Sei $N = (P,T,F)$ ein Netz und $K = (B, E, \prec)$ ein Kausalnetz.

Eine Funktion $r: B \cup E \rightarrow P \cup T$ ist eine *N-Beschriftung* von K, wenn die Funktion r Bedingungen auf Stellen und Ereignisse auf Transitionen abbildet.

Durch eine *N*-Beschriftung können wir jeder endlichen *co*-Menge C von K kanonisch eine Markierung von Σ zuordnen. Wir bezeichnen diese Markierung mit $r(C)$ und definieren $r(C): \wp \rightarrow \mathbb{N}$ mit $r(C)(p) = |\{b \in C \mid r(b) = p\}|$.

Definition 2.2.4 (*Verteilter Ablauf in einem Netz*)

Seien $N=(P,T,F)$ ein Netz, $K = (B, E, \prec)$ ein Kausalnetz, $r: B \cup E \rightarrow P \cup T$ mit $r: B \rightarrow P$, $r: E \rightarrow T$ eine *N*-Beschriftung von K.
Das Paar $\rho = (K, r)$ heißt *ein verteilter Ablauf in N*, gdw. für jedes Ereignis $e \in E$ gilt: Wenn $r(e) = t$, dann ist $r({}^\bullet e) = {}^- t$ und $r(e^\bullet) = t^+$.

Wir verlangen hier, dass jedes Ereignis e dem Schalten einer Transition t des Petrinetzes entspricht, d.h. e wird auf t abgebildet und der Vor- und Nachbereich von e auf den Vor- bzw. Nachbereich von t. Wir nennen e ein *Auftreten* der Transition t in diesem verteilten Ablauf in N.

Definition 2.2.5 (*Sequentialisierung eines verteilten Ablaufs im Netz N*)
Seien $N=(P,T,F)$ ein Netz, (K, r) ein verteilter Ablauf in N, wobei $K=(B,E,\prec)$. Sei weiter $K^\circ = C_0 \xrightarrow{e_1} C_1 \xrightarrow{e_2} C_2$ ... ein sequentieller Ablauf in K mit $\{e_1, e_2, ...\} = E$. Dann sagen wir, $r(C_0) \xrightarrow{r(e_1)} r(C_1) \xrightarrow{r(e_2)} r(C_2)$... ist eine *Sequentialisierung* von (K, r).

Bemerkung 2.2.6
Seien N ein Netz, (K, r) ein verteilter Ablauf in N.
Dann gilt: Jede Sequentialisierung von (K, r) ist ein sequentieller Ablauf in N.

Definition 2.2.7 (*Fairness im Ablauf*)
Seien $N=(P,T,F)$ ein Netz, $t \in T$.

i. Sei $w = M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2$ ... ein unendlicher sequentieller Ablauf in N. w *verletzt Fairness* von t (w ist *unfair zu t*), falls t in unendlich vielen Markeirungen M_i aktiviert ist und $t = t_i$ für nur endlich viele i.
ii. Ein sequentieller Ablauf w in N *respektiert Fairness* von t (w ist *fair zu t*), falls w nicht unfair zu t ist.
iii. Ein verteilter Ablauf (K, r) in N *respektiert Fairness* von t, falls jede Sequentialisierung von (K, r) Fairness von t respektiert.

Grafisch wird eine *faire* Transition mit „φ“ beschriftet.

Definition 2.2.8 (*System*)
Seien $N=(P,T,F)$ ein Netz, m_0 eine Markierung von N, $T_f \subseteq T$. $\Sigma = (N, m_0)$ mit T_f nennen wir ein *System*. Wir nennen m_0 die *Anfangsmarkierung* von Σ und T_f *die Menge der fairen Transitionen* von Σ.

Eine Markierung M ist *erreichbar in* Σ, wenn M von m_0 aus erreichbar ist.

Für alle Systeme nehmen wir an, dass jede aktivierte Transition irgendwann schaltet oder eine zu der Transition in Konflikt stehende Transition schaltet (d.h. keine Transition von Σ ist in K° aktiviert). Das ist die sogenannte *progress*-Annahme.

Definition 2.2.9 (*Verteilter Ablauf eines Systems*)
Sei $\Sigma = (N, m_0)$ mit T_f ein System, wobei $N=(P,T,F)$.
Ein verteilter Ablauf (K, r) in N ist *ein verteilter Ablauf von* Σ, gdw. (i) $r(^\circ K) = m_0$; (ii) Für alle Transtionen t aus T und alle endlichen *co*-Mengen $C \subseteq K^\circ$ gilt ${}^- t \not\leq r(C)$; (iii)(K,r) respektiert die Fairness aller Transitionen aus T_f.

Mit Bedingung (i) fordern wir, dass der Anfangsschnitt auf die Anfangsmarkierung des Petrinetzes abgebildet wird. In Bedingung (ii) verlangen wir, dass die *progress*-Annahme erfüllt wird. Mit Bedingung (iii) fordern wir, dass die *Fairness*-Annahme jeder fairen Transition erfüllt wird.

Sei $\rho = (K, r)$ ein verteilter Ablauf von Σ und sei C ein Schnitt des Kausalnetzes K. Wenn wir alle Elemente des Kausalnetzes weglassen, die grösser als C sind, dann erhalten wir ein *Anfangsstück* von ρ.

Bemerkung: Eine Markierung M von Σ ist genau dann erreichbar, wenn es einen verteilten Ablauf von Σ mit einem Schnitt C gibt, so dass $r(C)=M$.

Definition 2.2.10 (*1-beschränktes System*)
Ein System ist *1-beschränkt*, wenn auf keiner Stelle jemals mehr als eine Marke liegt, d.h. für jede erreichbare Markierung M und für jede Stelle $p \in \wp$ gilt $M(p) \leq 1$.

2.3 Eigenschaften von verteilten Algorithmen

Zur Beschreibung von Eigenschaften verteilter Algorithmen benutzen wir Formeln einer temporalen Logik (vgl. [Kin95], [Peu01]), die auf die vorher definierten Systeme abgestimmt ist. Wir geben hier nur die Syntax und die Semantik dieser Formeln an.

Terme

Mit Zustandaussagen formalisieren wir Aussagen über Zustände eines Systems. Die Grundbausteine der Zustandsaussagen sind Terme. Im Einführungsbeispiel *Dienstreise* (siehe Abb. 1.1) ist $bereit_C$ ein Stellenname des Systems. Dann ist $\#\{bereit_C\}$ ein Term. $\#\{bereit_C\} \geq 0$ ist eine mit Hilfe dieses Terms formulierte Zustandsaussage, die ausdrückt, dass die Anzahl der Marken auf der Stelle $bereit_C$ grösser oder gleich 0 ist. Sei m eine Variable der Sorte $\mathbb{N}$, wobei $\mathbb{N}$ die Menge der natürlichen Zahlen bezeichnet. Dann ist $\#\{ bereit_C \} \geq m$ eine Zustandsaussage, die ausdrückt, dass die Anzahl der Marken auf der Stelle $bereit_C$ grösser als m ist. m ist dabei ebenfalls ein Term.

Sei P eine Menge und V eine Menge von Variablen. Dann wird die Menge der Terme $\mathbf{ZT}(P,V)$ über P und V induktiv wie folgt definiert:

1. Für $n \in \mathbb{N}$ gilt $n \in \mathbf{ZT}(P,V)$.
2. Für $v \in V$ gilt $v \in \mathbf{ZT}(P,V)$.
3. Für jede endliche Menge $Z \subseteq P$ gilt $\#Z \in \mathbf{ZT}(P,V)$.
4. Für $u, u' \in \mathbf{ZT}(P,V)$ gilt auch $u+u' \in \mathbf{ZT}(P,V)$.

Eine Belegung für eine Variablenmenge V ist eine Funktion $\beta: V \to \mathbb{N}$, die jeder Variablen eine natürliche Zahl zuordnet. Die Terme werden in einer Markierung M unter einer Belegung β durch β_M: $\mathbf{ZT}(P,V) \to \mathbb{N}$ interpretiert, wobei β_M induktiv definiert ist durch:

1. $\beta_M (n) = n$ für $n \in \mathbb{N}$.
2. $\beta_M (v) = \beta(v)$ für $v \in V$.
3. $\beta_M (\#Z) = \sum_{s \in Z} M(s)$ für $Z \subseteq P$.
4. $\beta_M (u+u') = \beta_M (u) + \beta_M (u')$ für $u, u' \in \mathbf{ZT}(P,V)$.

Zustandsaussagen

Für zwei Terme $u, u' \in \mathbf{ZT}(P,V)$ ist $u \geq u'$ eine Zustandsaussage. Aus diesen Zustandsaussagen bilden wir mit den üblichen logischen Operatoren und Quantoren die Menge der Zustandsaussagen.

Definition 2.3.1 (*Zustandsaussage*)
Sei P eine Menge und V eine Menge von Variablen. Wir definieren die Menge $\mathbf{ZA}(P,V)$ der Zustandsaussagen über P und V wie folgt:

1. Wenn $u, u' \in \mathbf{ZT}(P,V)$, dann ist $u \geq u' \in \mathbf{ZA}(P,V)$.
2. Wenn $\varphi, \varphi' \in \mathbf{ZA}(P,V)$, dann ist $\varphi \wedge \varphi' \in \mathbf{ZA}(P,V)$.
3. Wenn $\varphi \in \mathbf{ZA}(P,V)$, dann ist $\neg\varphi \in \mathbf{ZA}(P,V)$.
4. Wenn $\varphi \in \mathbf{ZA}(P,V)$ und $v \in V$, dann ist $(\exists v\, \varphi) \in \mathbf{ZA}(P,V)$.

Wir interpretieren eine Zustandsaussage in einer Markierung M mit einer Belegung β der Variablen.

Definition 2.3.2 (*Gültigkeit einer Zustandsaussage*)
Eine Zustandsaussage φ ist genau dann in einer Markierung M und einer Belegung β gültig (Notation: $M,\beta \vDash \varphi$), wenn

1. $\varphi = (u \geq u')$ mit $u, u' \in \mathbf{ZT}(P,V)$, und $\beta_M (u) \geq \beta_M(u')$ oder
2. $\varphi = \varphi_1 \wedge \varphi_2$ und $M,\beta \vDash \varphi_1$ und $M,\beta \vDash \varphi_2$ oder
3. $\varphi = \neg\varphi_1$ und es gilt $M,\beta \vDash \varphi_1$ nicht oder
4. $\varphi = \exists v\, \varphi_1$ und es gibt eine Belegung β' mit $\beta(v')=\beta'(v')$ für alle $v' \neq v$, so dass $M, \beta' \vDash \varphi_1$.

Eine Zustandsaussage φ ist gültig in einer Markierung M (Notation: $M \vDash \varphi$), wenn φ in M unter jeder Belegung β gültig ist.

Temporale Aussagen

Zustandsaussagen sind die Grundbausteine von temporalen Aussagen. Temporale Aussagen interpretieren wir über verteilten Abläufen. Wir definieren zunächst nur den temporalen Operator $\Box$ (*always*) und bauen alle temporalen Aussagen induktiv aus $\Box$, $\wedge$ und $\neg$ auf. Wir führen dann andere temporale Operatoren als Abkürzungen von Kombinationen aus $\Box$, $\wedge$ und $\neg$ ein. Die Gültigkeit einer temporalen Aussage wird wieder induktiv über den Aufbau der Formeln definiert.

Definition 2.3.3 (*Gültigkeit einer temporalen Aussage*)
Seien $\rho = (K, r)$ ein verteilter Ablauf eines Systems $\Sigma = (N, m_0)$, C ein Schnitt aus ρ und β eine Belegung der Variablenmenge V.

1. Wenn $\varphi \in \mathbf{ZA}(P,V)$, dann gilt $\rho,C,\beta \vDash \varphi$ genau dann, wenn $r(C),\beta \vDash \varphi$.
2. Es gilt genau dann $\rho,C,\beta \vDash \Box\varphi$, wenn für jeden Schnitt C' mit $C \to^* C'$ aus ρ gilt $r(C'),\beta \vDash \varphi$.
3. Es gilt genau dann $\rho,C,\beta \vDash \varphi \wedge \varphi'$, wenn $\rho,C,\beta \vDash \varphi$ und $\rho,C,\beta \vDash \varphi'$.
4. Es gilt genau dann $\rho,C,\beta \vDash \neg\varphi$, wenn nicht $\rho,C,\beta \vDash \varphi$ gilt.
5. Es gilt genau dann $\rho,C,\beta \vDash \exists v\, \varphi$, wenn es eine Belegung β' gibt, mit

$\beta(v') = \beta'(v')$ für alle $v' \neq v$, so dass $\rho, C, \beta' \models \varphi$.

Eine temporale Aussage φ gilt im Ablauf ρ, wenn für jede Belegung β gilt: $\rho, {}^{\circ}K, \beta \models \varphi$. Die Aussage gilt im System Σ (Notation: $\Sigma \models \varphi$), wenn sie in jedem Ablauf gilt.

Die logischen Operatoren $\vee, \rightarrow, \leftrightarrow$ sind die üblichen Abkürzungen.

Für eine Zustandsaussage φ steht $\Diamond\varphi$ abkürzend für $\neg\Box\neg\varphi$. Für zwei Zustandsaussagen φ und φ' steht $\varphi \triangleright \varphi'$ abkürzend für $\Box(\varphi \rightarrow \Diamond\varphi')$. Wir nennen dies eine *leadsto-Aussage im verteilten Ablauf*. Sie bedeutet, dass in einem verteilten Ablauf auf einen Zustand, der φ erfüllt, ein Zustand folgt, der φ' erfüllt. In einem System gilt $\varphi \triangleright \varphi'$, wenn in jedem Ablauf $\varphi \triangleright \varphi'$ gilt.

Sei φ eine Zustandsaussage. Wenn $\Box\varphi$ in einem System gilt, dann heißt $\Box\varphi$ eine Invariante des Systems. In diesem Fall gilt φ in jeder erreichbaren Markierung des Systems. Besonders einfach zu handhabende Invarianten sind Stelleninvarianten. Einer Stelleninvariante entsprechend, gibt es eine Menge von Stellen, für die sich die Summe der Marken, die auf diesen Stelle liegen, beim Schalten nicht verändert.

In dieser Arbeit verwenden wir auch folgende Notation: Wenn p ein Stellenname ist, dann drücken wir mit p aus, dass mindestens eine Marke auf der Stelle p liegt. Wenn $Q=\{q_1, q_2, \ldots q_n\}$ eine Menge von Stellen ist, schreiben wir auch Q für $q_1 \wedge q_2 \wedge \ldots \wedge q_n$.

2.4 Kausalitäten im System

In der Einleitung wurde gezeigt, dass einige gewünschte Eigenschaften eines Systems bei der Verfeinerung einer Transition nur erhalten bleiben, wenn bestimmte Kausalitäten im Verfeinerungsnetz gelten. Die Aufgabe des folgenden Abschnittes ist es, einen geeigneten Kausalitätsbegriff für Systeme zu definieren. Grundlage dieser Definition ist die Kausalität in Abläufen. Weiter zeigt dieser Abschnitt, wie für 1-beschränkte Systeme unmittelbar aus der Struktur des Netzes des Systems die Gültigkeit einer Kausalitätsaussage für dieses System abgelesen werden kann.

In Abb. 2.4.1 ist ein weiteres *Dienstreise*-System Σ_3 dargestellt. In diesem System werden der *Chef* und der *Mitarbeiter* immer wieder Dienstreise unternehmen. Jedesmal wird zuerst der *Mitarbeiter* einen Vertrag behandeln (Transition *Vertrag-behandeln*$_M$), danach wird der *Chef* unterschreiben (Transition *unterschreiben*$_C$). Das nennen wir eine *Kausalität im System*. Im Folgenden werden wir genauer erklären, was dieser Begriff Kausalität im System in dieser Arbeit bedeutet. Wir fangen mit dem Begriff *Kausalität im Ablauf* an.

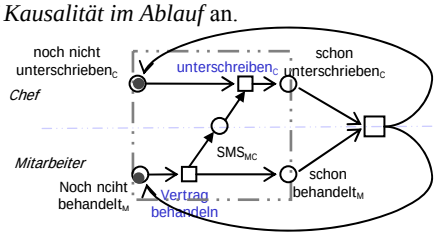


Abb. 2.4.1 System Σ_3

Wir sagen, in einem Ablauf gilt *Vertrag-behandeln*$_M$ bereitet *unterschreiben*$_C$ vor (Notation: *Vertrag-behandeln*$_M$ ◄ *unterschreiben*$_C$), gdw. es in diesem Ablauf zu jedem Auftreten von *unterschreiben*$_C$ ein Auftreten von *Vertrag-behandeln*$_M$ gibt, mit: a) Dieses Auftreten von *Vertrag-behandeln*$_M$ liegt vor dem Auftreten von *unterschreiben*$_C$; b) Zwischen den beiden Auftreten gibt es kein anderes Auftreten von *unterschreiben*$_C$.

In Abb. 2.4.2 ist ein Ablauf vom System Σ_3 dargestellt.

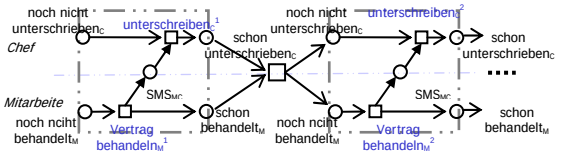


Abb. 2.4.2 Ablauf des Systems Σ_3 (Abb. 2.4.1)

In diesem Ablauf ist *unterschreiben*$_C^2$ ein Auftreten der Transition *unterschreiben*$_C$. Vor *unterschreiben*$_C^2$ liegt ein Auftreten *Vertrag-behandeln*$_M^2$ der Transition *Vertrag-behandeln*$_M$. Zwischen beiden gibt es kein Auftreten der Transition *unterschreiben*$_C$. Deshalb ist *Vertrag-behandeln*$_M^2$ für *unterschreiben*$_C^2$ das richtige Auftreten von *Vertrag-behandeln*$_M$, das die oben genannten Bedingungen a) und b) erfüllt.

Dagegen ist *Vertrag-behandeln*$_M^1$ zwar auch ein Auftreten von *Vertrag-behandeln*$_M$ und vor dem *unterschreiben*$_C^2$, aber nicht das richtige Auftreten von *Vertrag-behandeln*$_M$ für *unterschreiben*$_C^2$, weil zwischen *Vertrag-behandeln*$_M^1$ und *unterschreiben*$_C^2$ ein Auftreten *unterschreiben*$_C^1$ der Transition *unterschreiben*$_C$ liegt. *unterschreiben*$_C^2$ wurde von *Vertrag-behandeln*$_M^2$ vorbereitet und nicht von *Vertrag-behandeln*$_M^1$.

In einem System gilt *Vertrag-behandeln*$_M$◄*unterschreiben*$_C$, gdw. in jedem Ablauf *Vertrag-behandeln*$_M$◄*unterschreiben*$_C$ gilt.

In Abb. 2.4.3 ist ein weiteres System Σ_4 dargestellt. Σ_4 hat zwei Abläufe. In beiden Abläufen kommt die Transition *a* jeweils einmal vor. Aber nur in einem der beiden Abläufe tritt die Transition *b* vor dem Auftreten von *a* auf. Deshalb gilt in Σ_4 nicht *b* bereitet *a* vor, also $\Sigma_4 \not\models b \blacktriangleleft a$.

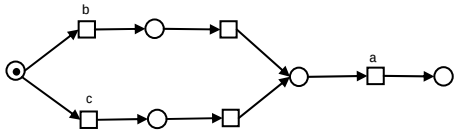


Abb. 2.4.3 System Σ_4

Definition 2.4.1 (*Kausalrelation ´bereitet vor´*)

Sei $\Sigma=((P,T,F), m_0)$ *ein System, seien* $x, y \in P \cup T$.

i. *Sei* (K, r) *ein Ablauf von* Σ. $x \blacktriangleleft y$ gilt *in* K (*Notation*: $K \models x \blacktriangleleft y$), *gdw. zu jedem* $s \in K$ *mit* $r(s)=y$ *gibt es ein* $s' \in K$ *mit* $r(s')=x$ *und* $s'<s$ *und für jedes* $s'' \in K$ *mit* $s'<s''<s$ *gilt* $r(s'') \neq y$.

ii. $x \blacktriangleleft y$ gilt *in* Σ (*Notation*: $\Sigma \models x \blacktriangleleft y$), *gdw. in jedem Ablauf* $x \blacktriangleleft y$ *gilt*.

Nun überlegen wir, wie wir direkt aus der Struktur des Netzes ablesen können, ob eine Kausalität im System gilt. Hier geben wir nur eine Ableseregel für den einfachsten

Fall an, dass das Netz ein Kausalnetz ist. Ableseregeln für kompliziertere Fälle siehe [Wu07].

Satz 2.4.2 (*Ableseregel der Relation ◄ zwischen Stellen bzw. Transitionen in Kausalnetzen*)
Sei Netz $N=(P,T,F)$ *ein Kausalnetz. Sei* $\Sigma=(N, m_0)$ *ein System mit:* $m_0(p)=1$ *gdw.* $p\in{}^\circ N$ (*genau die Stellen ohne Vorgänger sind markiert*). *Weiter seien* $x, y\in P\cup T$.
Dann gilt: $\Sigma \vDash x \blacktriangleleft y \leftrightarrow x\ F^+\ y$.
Beweis:
Da N ein Kausalnetz ist, hat Σ nur einen Ablauf. (N, i) beschreibt den Ablauf von Σ, wobei $i(x)=x$ für jedes $x\in N$. x, y kommen im Ablauf genau einmal vor. Die Kausalrelation $<$ im Ablauf ist gleich F^+. Also gilt wegen Def. 2.4.1 die Behauptung $\Sigma \vDash x \blacktriangleleft y$, *gdw.* $x\ F^+\ y$. □

Wenn eine Transition aus $\bullet$*Vertrag-schon-behandelt*$_M$ schaltet, dann wird danach die Stelle *Vertrag-schon-behandelt*$_M$ markiert, d.h. *Vertrag-schon-behandelt*$_M$ gilt. Analog, wenn eine Transition aus $\bullet$*schon-unterschrieben*$_C$ schaltet, dann wird danach die Stelle *schon-unterschrieben*$_C$ markiert, d.h. *schon-unterschrieben*$_C$ gilt. Wir beschreiben mit einer Formel $\bullet$*Vertrag-schon-behandelt*$_M\blacktriangleleft\bullet$*schon-unterschrieben*$_C$ das folgende: Jedesmal bevor eine Transition auftritt, deren Schalten dazu führt, dass *schon-unterschrieben*$_C$ gilt, ist eine Transition aufgetreten, deren Schalten dazu führt, dass *Vertrag-schon-behandelt*$_M$ gilt.

Im Folgenden definieren wir Kausalitäten der Form $\bullet p\blacktriangleleft\ \bullet q$ eines Systems.

Definition 2.4.3 ($\bullet p\blacktriangleleft\ \bullet q$)
Sei $\Sigma=((P,T,F), m_0)$ *ein System, seien* $p,q\in P$.
i. *Sei* (K, r) *ein Ablauf von* Σ. $\bullet p\blacktriangleleft\ \bullet q$ gilt *in* K (*Notation*: $K \vDash \bullet p \blacktriangleleft \bullet q$), *gdw. zu jedem* $e\in K$ *mit* $r(e)\in\bullet q$ *gibt es ein* $e'\in K$ *mit* $r(e')\in\bullet p$ *mit: entweder* $e'=e$; *oder* $e'<e$ *und für jedes* $e''\in K$ *mit* $e'<e''<e$ *gilt* $r(e'')\notin\bullet q$.
ii.$\bullet p \blacktriangleleft \bullet q$ gilt *in* Σ (*Notation*: $\Sigma \vDash \bullet p \blacktriangleleft \bullet q$), *gdw. in jedem Ablauf* $\bullet p\blacktriangleleft\ \bullet q$ *gilt*.

Allgemein gibt es zwischen zwei Stellen p und q die Relationen $p\bullet\blacktriangleleft q\bullet$, $\bullet p\blacktriangleleft\ \bullet q$, $p\bullet\blacktriangleleft\ \bullet q$ und $\bullet p\blacktriangleleft q\bullet$.

Definition 2.4.4
Sei $\Sigma=((P,T,F), m_0)$ *ein System, seien* $p,q\in P$ *und* $x=\bullet p$ *oder* $x=p\bullet$ *und* $y=\bullet q$ *oder* $y=q\bullet$.
i. *Sei* (K, r) *ein Ablauf von* Σ. $x\blacktriangleleft\ y$ gilt *in* K (*Notation*: $K\vDash x\blacktriangleleft y$), *gdw. zu jedem* $e\in K$ *mit* $r(e)\in y$ *gibt es ein* $e'\in K$ *mit* $r(e')\in x$ *mit: entweder* $e'=e$; *oder* $e'<e$ *und für jedes* $e''\in K$ *mit* $e'<e''<e$ *gilt* $r(e'')\notin y$.
ii. $x\blacktriangleleft\ y$ gilt *in* Σ (*Notation*: $\Sigma \vDash x\blacktriangleleft\ y$), *gdw. in jedem Ablauf* $x\blacktriangleleft\ y$ *gilt*.

Bemerkung 2.4.5
Sei $\Sigma=((P,T,F), m_0)$ *ein System. Weiter seien* $p,q\in P$, *sei* $x=\bullet p$ *oder* $x=p\bullet$, *sei* $y=\bullet q$ *oder* $y=q\bullet$. *Seien* $x=\{t\}$, $y=\{t'\}$, *wobei* $t, t'\in T$.
Dann gilt: $\Sigma \vDash x\blacktriangleleft\ y \leftrightarrow (t=t') \vee \Sigma \vDash t\blacktriangleleft\ t'$.

3. Transitionsverfeinerungen

Als Vorbereitung auf die in dieser Arbeit definierte verteilende Transitionsverfeinerung betrachten wir die von S. Peuker [Peu01] vorgestellte Transitionsverfeinerung mit Blockbedingung. Eine Transitionsverfeinerung mit Blockbedingung erhält wichtige Eigenschaften des verfeinerten Systems. Welche Eigenschaften sicher erhalten bleiben und einige Eigenschaften, die verloren gehen können, werden angegeben.

3.1 Transitionsverfeinerung mit Blockbedingung

In diesem Abschnitt wird der Begriff der Transitionsverfeinerung mit Blockbedingung angegeben. Zunächst wird der Begriff der Substitution in einem System definiert. Anschließend wird der Begriff der Substitution in einem Ablauf definiert. Danach wird definiert, wann eine Substitution eines Systems die Blockbedingung erfüllt.

Es sei noch einmal an die Bedeutung der Blockbedingung erinnert, die wir bereits in der Einleitung beschrieben haben. Das Konzept einer gemeinsammen Dienstreise eines Chefs mit seinem Mitarbeiter hatte folgende Form (Abb. 3.1.1):

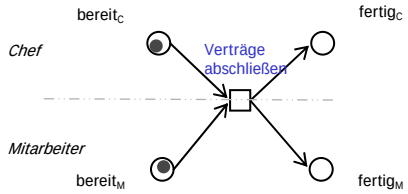


Abb. 3.1.1 *Gemeinsame Dienstreise* (System Σ_1)

Wenn *Chef* und *Mitarbeiter* getrennt diese Dienstreise unternehmen (Abb. 3.1.2), sollte weiter gesichert bleiben, dass beide die Dienstreise antreten, weiter alle beabsichtigten Verträge abgeschlossen werden und beide irgendwann zurückkommen. Die Transition darf zwar in mehrere Einzelaktionen zerlegt werden. Aber es muss gesichert werden, dass alle Einzelaktionen ausgeführt werden.

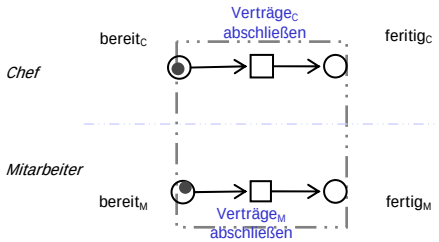


Abb. 3.1.2 *Getrennte Diestreise* (System Σ_1')

Definition 3.1.1 (*Ersetzbarkeit, Ersetzungsnetz*)
Sei Σ *ein System,* $t\in T$. *Sei* N_t *ein Netz.*
t *ist* ersetzbar *durch* N_t *in* Σ, *wenn:*

a) $P_{N_t} \cap P = {}^\bullet t \cup t^\bullet$, $T_{N_t} \cap T = \emptyset$ *und* ${}^\bullet({}^\bullet t) = \emptyset$ *in* N_t.

b) Für jeden Ablauf (K_{N_t}, r_{N_t}) *von* (N_t, t^-) *gilt:* K_{N_t} *ist endlich und* $r_{N_t}(K_{N_t}^\circ) = t^+$.

Wir nennen N_t *ein* Ersetzungsnetz *für t.*

Definition 3.1.2 ($t \to N_t$ *Substitution in einem System*)
Sei $\Sigma = (N, m_0)$ *ein System mit* $N = (P, T, F)$, $t \in T$. *Sei* $N_t = (P_{N_t}, T_{N_t}, F_{N_t})$ *ein Netz mit: t ist ersetzbar durch* N_t *in* Σ.

$\Sigma' =_{def} (N', m_0)$, *wobei:*

$$N' =_{def} (P', T', F');$$
$$P' =_{def} P \cup P_{N_t};$$
$$T' =_{def} T \cup T_{N_t} \setminus \{t\};$$
$$F' =_{def} (F \cup F_{N_t}) \cap (P' \times T' \cup T' \times P').$$

Σ' *heißt die* $t \to N_t$ Substitution *von* Σ. *Notation:* $\Sigma' = \Sigma(N_t \backslash t)$.
$\Sigma \to \Sigma'$: *Die entsprechende* Systemtransformation *von* Σ *nach* Σ'.

Im Folgenden definieren wir die Blockbedingung mit Hilfe des Begriffes Ablauf-Substitution. Die Ablauf-Substitution bedeutet folgendes: Sei K ein verteilter Ablauf eines Systems, das eine Transition t enthält. Wir ersetzen jedes Auftreten von t in K durch einen Ablauf von (N_t, t^-). Der erhaltene Ablauf ist eine $t \to N_t$ Substitution von K. Die Ablauf-Substitution wird durch die folgenden zwei Definitionen formuliert.

Definition 3.1.3 (*Passender Ablaufblock*)
Sei Σ *ein System, sei* $t \in T$ *eine Transition und ersetzbar durch* N_t *in* Σ. *Sei* (K, r) *ein Ablauf von* Σ, *sei e ein t-Auftreten in K. Sei* (K_{N_t}, r_{N_t}) *ein Ablauf von* (N_t, t^-).

(K_{N_t}, r_{N_t}) *heißt passend zu e in K, wenn:*

a) ${}^\bullet e \cup e^\bullet = {}^\circ K_{N_t} \cup K_{N_t}^\circ = K \cap K_{N_t}$

(beide Abläufe K und K_{N_t} *haben genau die Bedingungen gemeinsam, die im Vor- und Nachbereich von e enthalten sind);*

b) Für jedes $b \in {}^\bullet e \cup e^\bullet$ *gilt:* $r(b) = r_{N_t}(b)$

(Diese Bedingungen entsprechen den gleichen Stellen aus N_t *und damit aus* ${}^\bullet t \cup t^\bullet$*).*

Definition 3.2.4 (*Ablauf-Substitution*)
Sei Σ *ein System, sei* $t \in T$ *eine Transition und ersetzbar durch* N_t *in* Σ. *Sei* (K, r) *ein Ablauf von* Σ, *wobei* $K = (B, E, <)$. $E(t) := \{ e \in E \mid r(e) = t \}$. *Für jedes* $e \in E(t)$, *sei* (K_e, r_e) *ein Ablauf von* (N_t, t^-) *und passend zu e in K, wobei* $K_e = (B_e, E_e, <_e)$. *Für alle* $e \in E(t)$, *seien* K_e *paarweise disjunkt (keine Bedingungen und Ereignisse in den eingesetzten Abläufen gleich benannt).*

$K' =_{def} (B', E', <')$, *wobei:*

$$B' =_{def} B \cup \Big(\bigcup_{e \in E(t)} B_e \Big),$$

$$E' =_{def} E \cup \Big(\bigcup_{e \in E(t)} (E_e \setminus \{e\}) \Big),$$

$$<' =_{def} < \cup \Big(\bigcup_{e \in E(t)} (<_e \setminus ({}^\bullet e \times \{e\} \cup \{e\} \times e^\bullet)) \Big).$$

$$r'(s) =_{def} \begin{cases} r_e(s), & \text{falls } s \in K_e, e \in E(t), \\ r(s), & \text{sonst.} \end{cases}$$

(K', r') *heißt eine* $t \to N_t$ Substitution *von* (K, r).

K_e *heißt* ein Auftreten von N_t in K'.

Bedingungen und Ereignisse aus K sind auch Bedingungen und Ereignisse von K'. Die restlichen Bedingungen und Ereignisse aus K' sind Auftreten von Stellen bzw. Transitionen des Ersetzungsnetzes N_t.

Bemerkung 3.1.5
Sei $\Sigma = ((P, T, F), m_0)$ *ein System, sei* $t \in T$ *ersetzbar durch* N_t *in* Σ, *wobei* $N_t = (P_{N_t}, T_{N_t}, F_{N_t})$. *Sei* (K, r) *ein Ablauf von* Σ, *wobei* $K = (B, E, <)$. *Sei* (K', r') *eine* $t \to N_t$ *Substitution von* (K, r), *wobei* $K' = (B', E', <')$.
Dann gilt:

i. Für jedes $s \in B \cup E$ *mit* $r(s) \neq t$ *gelten* $s \in B' \cup E'$ *und* $r'(s) = r(s)$.

ii. Für jedes $s \in (B' \cup E') \setminus (B \cup E)$ *gilt* $r'(s) \in (P_{N_t} \cup T_{N_t}) \setminus ({}^\bullet t \cup t^\bullet)$.

Definition 3.1.6 (*Block-Bedingung*)
Sei $\Sigma' = \Sigma(N_t \setminus t)$.

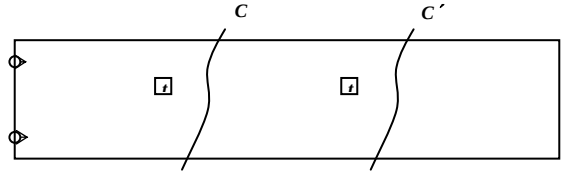
i. Ein Ablauf (K', r') *von* Σ' erfüllt die Block-Bedingung, *gdw. es einen Ablauf* (K, r) *von* Σ *gibt, so dass* (K', r') *eine* $t \to N_t$ *Substitution von* (K, r) *ist.*

ii. Σ' erfüllt die Block-Bedingung, *gdw. jeder Ablauf von* Σ' *die Blockbedingung erfüllt.*

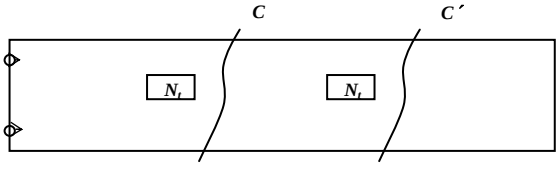
3.2 Eigenschaften der Blockbedingung

In diesem Abschnitt betrachten wir, welche temporal logische Eigenschaften bei einem Verfeinerungsschritt erhalten bleiben, wenn die Blockbedingung erfüllt ist. Der Abschnitt fasst die wichtigen Aussagen zur Blockbedingung aus [Peu01] zusammen.

In Abb. 3.2.1 (a) wird ein Ablauf K eines Systems vor der Verfeinerung illustriert, wobei $\boxed{t}$ für ein Auftreten von t in K steht und C, C' für Schnitte in K stehen. In Abb. 3.2.1 (b) wird ein Ablauf K' des neuen Systems nach der Verfeinerung illustriert, der K entspricht (d.h. K' ist eine $t \to N_t$ Substitution von K), wobei $\boxed{N_t}$ für ein Auftreten von N_t in K' steht.



(a) Veranschaulichung des Auftretens von t in einem Ablauf K von Σ und von Schnitten C und C' mit $C < C'$



(b) Veranschaulichung des Auftretens von Abläufen von (N_t, t^-) in dem zugeordneten Ablauf K' von Σ' und der Schnitte C und C'

Abb. 3.2.1 Veranschaulichung einer $t \to N_t$ Substitution in einem Ablauf K

Die Blockbedingung sichert folgende Eigenschaften:
- Jeder Schnitt von K ist auch ein Schnitt von K'. In Abb. 3.2.1 ist der Schnitt C in K auch ein Schnitt in K'.
- Wenn ein Schnitt C in (K, r) auf eine Markierung m abgebiledet wird, dann wird C in (K', r') auch auf die gleiche Markierung m abgebildet, d.h. $r(C) = r'(C)$.
- Wenn in K gilt: C' ist von C erreichbar, dann gilt in K'auch: C' ist von C erreichbar.

Aus den oben genannten Eigenschaften, die die Blockbedingung erhält, kann man weitere Eigenschaften ableiten, die die Blockbedingung erhält.

- $\Diamond\varphi$ (d.h. in jedem Ablauf gibt es einen Zustand, in dem φ gilt);

- $\Box\Diamond\varphi$ (d.h. in jedem Ablauf gibt es immer wieder einen Zustand, in dem φ gilt),

wobei φ eine Zustandsaussage ist.

3.3 Was bleibt nicht erhalten

Nun fragen wir uns: Welche Eigenschaften können in einem Verfeinerungsschritt verloren gehen, auch wenn die Blockbedingung erfüllt ist?

Lebendigkeitseigenschaft:
Lebendigkeitseigenschaften gehen verloren, weil nach der Verfeinerung neue Zustände mit neuen Eigenschaften entstehen. Im System *gemeinsame Dienstreise* vor der Verfeinerung (siehe Abb. 1.3) gilt offensichtlich die folgende Eigenschaft:

$$bereit_M \ \triangleright \ bereit_C \qquad \text{(L)}$$

In Abb. 1.4 (nach der Verfeinerung) gilt das nicht mehr, weil jetzt der Zustand erreichbar ist, in dem $bereit_M$, $fertig_C$ gelten. Auf diesen Zustand folgt nicht mehr $bereit_C$.

Sicherheitseigenschaft:
Im System *gemeinsame Dienstreise* vor der Verfeinerung (siehe Abb. 1.3) gilt offensichtlich die folgende Eigenschaft:

$$\Box\ \neg\,(bereit_M \wedge fertig_C) \qquad \text{(S)}$$

(*Chef* will nicht ohne den *Mitarbeiter* Verträge abschließen).

Nach der Verfeinerung (siehe Abb. 1.4) kann der Mitarbeiter nach der Rückkehr des Chefs losfahren. Also, auch die Sicherheitseigenschaft (S) geht verloren.

4. Agentensysteme

Um formalisieren zu können was es heißt, bei einer Transitionsverfeinerung eine gemeinsame Aufgabe mehrer Agenten auf die einzelnen Agenten aufzuteilen, wird in diesem Abschnitt geklärt, was Agenten eines Netzsystems, eines Netzes mit Anfangsbelegung der Stellen, sein sollen. Ein Agent wird durch eine Teilmenge von Stellen des Netzes angegeben. Sein Zustand bei einem Ablauf ist durch die Belegung dieser Stellen gegeben. Weiter wird definiert, was Nachrichtenstellen sind.

Im Einführungsbeispiel *Diestreise* gibt es im System Σ_2' (siehe Abb. 1.4) zwei Agenten *Chef* und *Mitarbeiter*, die durch Nachrichten kommunizieren. Die Stellen $bereit_C$ und $fertig_C$ gehören zu dem Agenten *Chef*. Die Stellen $bereit_M$ und $fertig_M$ gehören zu dem Agenten *Mitarbeiter*. Die Stelle SMS_{CM} ist ebendfalls eine Stelle des *Mitarbeiters*, denn sie nimmt eine Nachricht des *Chefs* an den *Mitarbeiter* auf. Damit ist sie auch eine Kanalstelle.

Wir definieren Agentensysteme in zwei Schritten: In der ersten Definition betrachten wir die Struktur von Netzen mit Agenten. In der zweiten Definition formulieren wir die Anforderungen an die von uns weiter untersuchten Agentensysteme.

Definition 4.1 (*Netz mit Agenten*)
Sei $N=(P,T,F)$ ein Netz. Sei A eine Partition von P, dann heißt N ein Netz mit Agentenmenge *A. Die Elemente von A nennen wir* Agenten*.*

Die folgende Definition klärt, wann eine Transition verteilt bzw. unverteilt ist. Anschließend definieren wir Kanalstellen. Im Einführungsbeispiel *Diestreise* betrifft die Transition *Verträge-abschließen* im System Σ_2 (siehe Abb. 1.3) beide Agenten, sie ist *unverteilt*. Die Transition *Verträge-abschließen*$_C$ im System Σ_2' (siehe Abb. 1.4) betrifft dagegen nur den Agenten *Chef*. Sie ist schon *verteilt* (lokal zu Agenten *Chef*). Wenn alle Transitionen eines Systems verteilt sind, dann ist dieses System verteilt, sonst ist es nicht verteilt.

Definition 4.2 (*verteilt, unverteilt*)
Sei $N=(P,T,F)$ ein Netz mit Agentenmenge A und sei $t\in T$.
i. Sei $a\in A$ ein Agent.
t ist lokal in *a, gdw. $\bullet t\subseteq a$ (t liest nur von Stellen von a).*
ii. t ist verteilt*, gdw. es ein $a\in A$ gibt, so dass t lokal in a ist.*
iii. t ist unverteilt*, gdw. t nicht verteilt ist.*
iv. N ist verteilt*, gdw. jedes $t\in T$ verteilt ist.*

Definition 4.3 (*Kanalstelle*)
Sei $N=(P,T,F)$ ein Netz mit Agentenmenge A und sei $a\in A$ ein Agent.

$p\in P$ ist eine Kanalstelle für *a, gdw. 1. $p\in a$; 2. $p\bullet\subseteq$ {lokale Transitionen von a} (nur a darf die Nachricht empfangen).*

Kanalstellen dienen zur Synchronisation der Arbeit unterschiedlicher Agenten.

Im System Σ_2' (siehe Abb. 1.4) beschreibt die Stelle SMS_{CM} eine Nachricht des *Chefs* an den *Mitarbeiter*. Eine Nachricht ist immer von einem Agenten für einen anderen Agenten. Also, der Sender und der Empfänger einer Nachricht sind immer unterschiedliche Agenten.

In dieser Arbeit betrachten wir nur einsbeschränkte Systeme, weil nur in diesen Systemen Kausalitätsaussagen hinreichend einfach bewiesen werden können.

Definition 4.4 (*Agentensystem*)
Sei $\Sigma=(N, m_0)$ ein System mit: $N=(P,T,F)$ ist ein Netz mit Agentenmenge A.
Σ ist ein Agentensystem*, gdw. Σ 1-beschränkt ist.*

5. Verteilende Verfeinerung

In diesem Abschnitt werden wir den Begriff *verteilende Verfeinerung* definieren. Wir untersuchen solche Verfeinerungsschritte, in denen eine unverteilte Transition t zu einem Netz N_t verfeinert wird, das die durch die Transition t modellierten Aktionen auf Transitionen der beteiligten Agenten verteilt. Wir werden diskutieren, welche Bedingungen ein solcher Verfeinerungsschritt erfüllen soll, damit die Korrektheit des Systems bei dem Verfeinerungsschritt erhalten bleibt.

In der Einleitung haben wir schon am System in Abb. 1.5 gesehen, dass eine Transitionsverfeinerung die Blockbedingung und kausale Bedingungen im Ersetzungsnetz erfüllen muss, damit gewünschte Eigenschaften des Systems erhaltenbleiben. Bei einer Verfeinerung zur Verteilung einer unverteilten Transition mehrerer Agenten sind diese kausalen Bedingungen im Ersetzungsnetz kausale Abhängigkeiten zwischen Transitionen unterschiedlicher Agenten. Die kausalen Abhängigkeiten der Aktionen der Agenten nennen wir *Synchronisationsbedingung*.

Im Folgenden werden wir zunächst *verteilende Systemtransformation* definieren und danach die Definition für die Synchronisationsbedingungen angeben und schließlich die verteilende Verfeinerung bezüglich einer Synchronisationsbedingung definieren.

5.1 Verteilende Systemtransformation

In diesem Abschnitt werden wir definieren, was wir als verteilende Transitionsverfeinerung in einem Agentensystem verstehen wollen.

Sei Σ ein Agentensystem mit Agentenmenge A und einer Transition t. Weiter sei $\Sigma' = \Sigma(N_t \setminus t)$. Wir betrachten zunächst, welche Bedingungen das Netz N_t noch erfüllen soll, damit Σ' ein System mit Agenten ist (1-Beschränktheit wird noch nicht gefordert). Danach betrachten wir, was die Agenten von Σ' sind.

Als erstes fordern wir, dass das Teilsystem (N_t, t^-) auch ein Agentensystem ist und die Zuordnung der Stellen aus $^\bullet t \cup t^\bullet$ zu den Agenten erhalten bleibt.

Bei einer formalen Definition der Forderung, dass die Zuordnung der Stellen aus $^\bullet t \cup t^\bullet$ zu Agenten bei der Verfeinerung erhalten bleibt, ist zu beachten, dass Agenten Stellenmengen sind. Diese Stellenmengen können sich bei der Verfeinerung vergrößern. Was gleich bleibt, ist die Zugehörigkeit zur gleichen Menge oder zu unterschiedlichen Mengen. Gehörten p und q vor der Verfeinerung zum gleichen Agenten, so gehören p und q auch nach der Verfeinerung zum gleichen Agenten.

Eine Agentenmenge A ist eine Zerlegung von Agentenstellen in eine Menge von Klassen. Diese Menge von Klassen A bilden die Äquivalenzklassen einer Äquivalenzrelation $\sim_A$. Zwei Stellen p, q sind äquivalent, d.h. $p \sim_A q$, gdw. sie beide zu dem selben Agenten gehören. Im Einführungsbeispiel *Dienstreise* in Abb. 1.3 ist die Agentenmenge $A = \{$ *Chef, Mitarbeiter* $\}$. Die Stellen $bereit_C$, $fertig_C$ gehören zu dem selben Agenten *Chef*, deshalb gilt $bereit_C \sim_A fertig_C$. Die Stellen $bereit_C$, $bereit_M$ gehören nicht zu dem selben Agenten, deshalb gilt $\neg (bereit_C \sim_A bereit_M)$.

Die Zerlegung der Agentenstellen A_{N_t} von (N_t, t^-) soll der Zerlegung der Agentenstellen A von Σ nicht widersprechen, d.h. wenn zwei Stellen in Σ zu dem selben Agenten gehören, dann gehören sie in (N_t, t^-) auch zu dem selben Agenten. Z.B. in Σ gehören $bereit_C$ und $fertig_C$ zu dem selben Agenten, sie gehören in (N_t, t^-) auch zusammen (siehe Abb. 1.4), also $bereit_C \sim_{A_{Nt}} fertig_C$.

Notation ($\sim_A$)
Sei A eine Zerlegung der Menge S.
Die Relation $\sim_A \subseteq S \times S$ *ist wie folgt definiert:*

Für $s,t \in S$, $s \sim_A t$ *gdw.* $\exists\ a \in A$: $s,t \in a$.

Definition 5.1.1 (*Ersetzbarkeit in einem Agentensystem*)
Sei Σ *ein Agentensystem mit Agentenmenge A und sei* $t \in T$ *eine Transition von* Σ. *Sei* (N_t, t^-) *ein Agentensystem mit Agentenmenge* A_{N_t}.

t *heißt* t im Agentensystem Σ *durch* N_t ersetzbar, *gdw.*
i. t *ist im System* Σ *durch* N_t *ersetzbar;*
ii. Für jedes $p,q \in {}^\bullet t \cup t^\bullet$ *gilt:* $p \sim_A q$ *gdw.* $p \sim_{A_{Nt}} q$ (d.h. p, q gehören in Σ zu dem selben Agenten gdw. sie in (N_t, t^-) zu dem selben Agenten gehören).

Die Agentenmenge A' von Σ' berechnen wir wie folgt: Zunächst ist eine Äquivalenzrelation $\sim'$ aus der Äquivalenzrelation $\sim_A$ und der Äquivalenzrelation $\sim_{A_{Nt}}$ abzuleiten (und zwar ist $\sim' =$ *die transitive Hülle von* $\sim_A \cup \sim_{A_{Nt}}$). Und dann ist aus dieser Äquivalenzrelation die Zerlegung A' zu definieren. A' ist die Agentenmenge des neuen Systems Σ'.

Wir sagen, das System Σ' ist eine $t \rightarrow N_t$ Substitution des Agentensystems Σ und $\Sigma \rightarrow \Sigma'$ ist eine Agentensystem-Transformation vom Agentensystem Σ nach SystemΣ'. Weil t unverteilt und N_t verteilt ist, nennen wir $\Sigma \rightarrow \Sigma'$ eine *verteilende Systemtransformation*.
Σ' ist im Allgemeinen nur ein System mit Agenten, weil Σ' nicht 1-beschränkt sein muss.

Notation ($A_\sim$)
Sei $\sim \subseteq S \times S$ *eine Äquivalenzrelation.*
Für $s \in S$, $[s] =_{def} \{\ t \in S \mid s \sim t\ \}$.
$A_\sim =_{def} \{\ [s] \mid s \in S\ \}$.

Für $\Sigma(N_t \setminus t)$ sind das neue Netz und die Agentenmenge zu definieren.

Definition 5.1.2 ($t \rightarrow N_t$ *Substitution in einem Agentensystem und Verteilende Systemtransformation*)
Sei Σ *ein Agentensystem mit Agentenmenge A und sei* $t \in T$ *eine Transition von* Σ. *Sei* (N_t, t^-) *ein Agentensystem mit*

Agentenmenge A_{N_t} mit: t ist im Agentensystem Σ durch N_t ersetzbar.

Ein System Σ' mit Agentenmenge A' heißt eine $t \to N_t$ Substitution des Agentensystems Σ *gdw. i. System Σ' ist eine $t \to N_t$ Substitution des Systems Σ; ii. $A' = A_{\sim'}$, wobei: $\sim' =_{def}$ die transitive Hülle von ($\sim_A \cup \sim_{A_{Nt}}$).*

$\Sigma \to \Sigma'$ heißt Agentensystem-Transformation *von Agentensystem Σ.*

$\Sigma \to \Sigma'$ heißt eine verteilende Systemtransformation, wenn t unverteilt und N_t verteilt ist. □

Wir betrachten in dieser Arbeit im Folgenden nur noch die Verfeinerung von unverteilten Transitionen zweier Agenten. Weiter gehen wir davon aus, dass jeder der beiden Agenten seine Arbeit im Verfeinerungsnetz erst beginnt, wenn alle Voraussetzungen für seine Arbeit erfüllt sind. Das modellieren wir dadurch, dass die beiden Agenten mit genau einer Starttransition beginnen, die jeweils alle Marken von $^\bullet t$, die dem Agenten gehören, konsumiert.

Definition 5.1.3 (*Standardverfeinerungsnetz für zwei Agenten*)
Sei t eine unverteilte Transition der beiden Agenten a und b. Ein Verfeinerungsnetz N_t ist ein Standardverfeinerungsnetz, *wenn für N_t gilt: Es gibt genau eine Transition, die auf Stellen von a zugreift und genau eine Transition, die auf Stellen von b zugreift.*

Alle bisher abgebildeten Verfeinerungsnetze waren Standardverfeinerungsnetze.

5.2 Synchronisations-Bedingung und verteilende Verfeinerung für Standardverfeinerungsnetze

In diesem Abschnitt werden wir den Begriff *Synchronisationsbedingung* einer unverteilten Transition und den Begriff *verteilende Verfeinerung* bzgl. einer Synchronisationsbedingung angeben.

Zur Veranschaulichung der wichtigsten Eigenschaften von Standardverfeinerungsnetzen können wir uns auf Beispiele beschränken, in denen die zu verfeinernde Transition t genau eine Stelle von a und genau eine Stelle von b im Vorbereich bzw. im Nachbereich hat. Das System in Abb. 5.2.1 enthält solche Transition t.

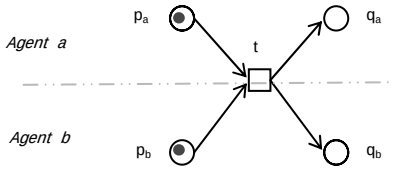


Abb. 5.2.1 Illustration für Synchronisationsbedingungen

Für Standardverfeinerungsnetze von Transition t mit $^\bullet t = \{p_a, p_b\}$ und $t^\bullet = \{ q_a, q_b \}$ gilt immer: $p_a{}^\bullet \blacktriangleleft {}^\bullet q_a$ bzw. $p_b{}^\bullet \blacktriangleleft {}^\bullet q_b$, d.h. der Agent a (bzw. b) konsumiert erst alle ihm gehörenden Marken aus dem Vorbereich bevor er ein Ergebnis bereitstellt, also eine Stelle aus $t^\bullet$ belegt. Bei Standardverfeinerungsnetzen bleiben also die Kausalitäten $p_a{}^\bullet \blacktriangleleft {}^\bullet q_a$ immer erhalten.

Bemerkung 5.2.1
Seien Σ ein Agentensystem mit Agentenmenge A, t eine unverteilte Transition von Σ, N_t ein Standardverfeinerungnsnetz für t. Es gelte $p \sim_A q$, wobei $p \in {}^\bullet t$ und $q \in t^\bullet$.
Dann gilt $(N_t, t^-) \models p^\bullet \blacktriangleleft {}^\bullet q$.

Im System in Abb. 5.2.1 gelten weiter folgende Kausalitäten: $p_a{}^\bullet \blacktriangleleft p_b{}^\bullet$, $p_a{}^\bullet \blacktriangleleft {}^\bullet q_b$, ${}^\bullet q_a \blacktriangleleft {}^\bullet q_b$, $p_b{}^\bullet \blacktriangleleft p_a{}^\bullet$, $p_b{}^\bullet \blacktriangleleft {}^\bullet q_a$ und ${}^\bullet q_b \blacktriangleleft {}^\bullet q_a$.

Alle diese Kausalitäten beschreiben Reihenfolgebeziehungen zwischen Aktionen, die Stellen unterschiedlicher Agenten belegen. Wir nennen sie *Synchronisationsbedingungen*. Bei der Verfeinerung einer Transition t durch ein Nezt N_t können im lokalen Teilsystem (N_t, t^-) einige gewünschte Synchronisationsbedingungen erhalten bleiben andere können wegfallen.

Im Beispiel *gemeinsame Dienstreise* gilt $bereit_C{}^\bullet \blacktriangleleft {}^\bullet fertig_M$, d.h. der *Chef* und der *Mitarbeiter* schließen Verträge ab, nachdem der *Chef* bereit ist. Danach kommt der *Mitarbeiter* von der Reise zurück. Genauso gilt $bereit_M{}^\bullet \blacktriangleleft {}^\bullet fertig_C$.

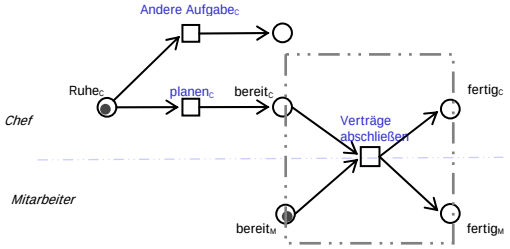


Abb. 5.2.2 Eine gemeinsame Dienstreise Σ_2

Im verfeinerten System gilt offensichtlich weiter $bereit_C{}^\bullet \blacktriangleleft {}^\bullet fertig_M$. Dagegen gilt nicht mehr $bereit_M{}^\bullet \blacktriangleleft {}^\bullet fertig_C$, da der *Chef* Verträge abschließen und zurückkommen kann, bevor der *Mitarbeiter* Verträge abschließt.

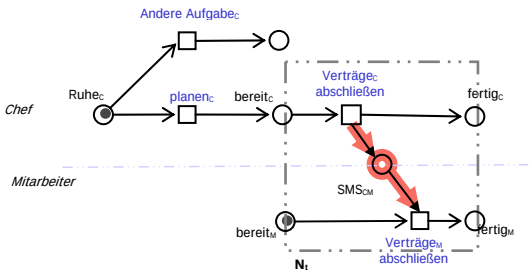


Abb. 5.2.3 Eine Verteilung Σ_2' von Σ_2 in Abb. 5.2.2

Die Synchronisation zwischen unterschiedlichen Agenten lokal in einem Ersetzungsnetz sichert das Erhaltenbleiben dieser Synchronisation im ganzen System.

Im Folgenden werden wir zunächst die Definition für Synchronisationsbedingungs-Formeln angeben. Dann definieren wir die Gültigkeit einer Synchronisationsbedingung im Ersetzungsnetz.

Definition 5.2.2 (*Synchronisationsbedingungs-Formeln*)
Sei Σ *ein Agentensystem mit Agentenmenge A und sei* $t \in T$ *unverteilt.*
Die Menge $\mathcal{P}_s$ *der* Synchronisationsbedingungs-Formeln *von t wird wie folgt induktiv definiert:*
a) $true \in \mathcal{P}_s$ *;*
b) Seien $p, q \in {}^\bullet t \cup t^\bullet$ *mit* $\neg (p \sim_A q)$.

$({}^\bullet p \blacktriangleleft {}^\bullet q) \in \mathcal{P}_s$ *, wenn* $p, q \in t^\bullet$*;*

$(p^\bullet \blacktriangleleft q^\bullet) \in \mathcal{P}_s$ *, wenn* $p, q \in {}^\bullet t$ *;*

$(p^\bullet \blacktriangleleft {}^\bullet q) \in \mathcal{P}_s$ *, wenn* $p \in {}^\bullet t, q \in t^\bullet$ *.*

c) $(p_{s1} \wedge p_{s2}) \in \mathcal{P}_s$ *, wenn* $p_{s1}, p_{s2} \in \mathcal{P}_s$ *.*

Definition 5.2.3 (Gültigkeit einer Synchronisationsbedingungs-Formel)
Sei $\Sigma \rightarrow \Sigma'$ *mit* $\Sigma' = \Sigma(N_t \backslash t)$ *eine verteilende Systemtransformation, sei* $p_s \in \mathcal{P}_s$ *eine Synchronisationsbedingung von t.*
Die Gültigkeit *von* p_s *im System* (N_t, t^-) *wird wie folgt induktiv definiert:*
Fall a) $p_s \equiv true$:
Die Synchronisationsbedingung p_s *ist in* (N_t, t^-) *erfüllt.*
Fall b) p_s *ist eine kausale Bedingung vom System* (N_t, t^-)
(d.h. $p_s = {}^\bullet p \blacktriangleleft {}^\bullet q$ *oder* $p_s = p^\bullet \blacktriangleleft q^\bullet$ *oder* $p_s = p^\bullet \blacktriangleleft {}^\bullet q$*):*
Die Synchronisationsbedingung p_s *ist in* (N_t, t^-) *erfüllt, gdw.* $(N_t, t^-) \models p_s$.
Fall c) $p_s = p_{s1} \wedge p_{s2}$, *wobei* $p_{s1}, p_{s2} \in \mathcal{P}_s$:
Die Synchronisationsbedingung p_s *ist in* (N_t, t^-) *erfüllt, gdw.* p_{s1} *und auch* p_{s2} *in* (N_t, t^-) *erfüllt sind.*

Wie schon erwähnt sind für das System Σ_2 in Abb. 5.2.2 z.B. die folgenden Formeln Synchronisationsbedingungen der Transition *Verträge-abschließen*:

$$p_{s1} = bereit_C {}^\bullet \blacktriangleleft {}^\bullet fertig_M$$

$$p_{s2} = bereit_M {}^\bullet \blacktriangleleft {}^\bullet fertig_C.$$

Für das Verfeinerungsnetz N_t (siehe Abb. 5.2.3) gilt in (N_t, t^-) die Synchronisationsbedingung p_{s1} aber nicht p_{s2}.

Angenommen, p_s ist eine Synchronisationsbedingungs-Formel einer unverteilten Transition *t*. Wenn das neue System Σ' nach der Verfeinerung die Blockbedingung erfüllt und die Synchronisationsbedingung p_s im Ersetzungsnetz N_t erfüllt ist, dann ist Σ' eine verteilende Verfeinerung von Σ bzgl. p_s .

Definition 5.2.4 (*Verteilende Verfeinerung bzgl.* p_s)
Sei $\Sigma \rightarrow \Sigma'$ *mit* $\Sigma' = \Sigma(N_t \setminus t)$ *eine verteilende Systemtransformation, sei* $p_s \in \mathcal{P}_s$ *.*
Σ' *heißt eine* verteilende Verfeinerung *bzgl.* p_s *von* Σ*, gdw.* Σ' *die Blockbedingung erfüllt und die Synchronisationsbedingung* p_s *in* (N_t, t^-) *erfüllt ist.*

In unserem Beispiel ist Σ_2' in Abb. 5.2.3 bzgl. p_{s1} eine verteilende Verfeinerung von Σ_2 aber bzgl. p_{s2} keine verteilende Verfeinerung von Σ_2.

5.3 Erhaltenbleiben von Eigenschaften

Aus dem Abschnitt 3.3 wissen wir, dass bei der Transitionsverfeinerung mit der Blockbedingung Sicherheits- und Lebendigkeitseigenschaften verloren gehen können. In einem Ablauf K' der durch $t \rightarrow N_t$ Substitution aus einem Ablauf K von Σ entsteht, können neue Schnitte auftreten, die es vorher in K nicht gab. Diese neuen Schnitte waren für das Verlorengehen dieser Eigenschaften verantwortlich. Wir müssen uns also die möglichen neuen Schnitte in Abläufen K' von Σ' ansehen und nach Synchronisationsbedingungen suchen, die unerwünschte Schnitte ausschließen.

Im Beispiel *Dienstreise* in Abb. 5.2.2, Abb. 5.2.3 ist der unerwünschte Schnitt möglich, in dem $fertig_C$ und $bereit_M$ gelten (siehe Abb. 5.3.1, Abb. 5.3.2), da der *Chef* Verträge abschließen kann, bevor der *Mitarbeiter* Verträge abschließt. Also, bei dieser Verfeinerung geht die Relation $bereit_M < fertig_C$ im Ablauf K_2' verloren.

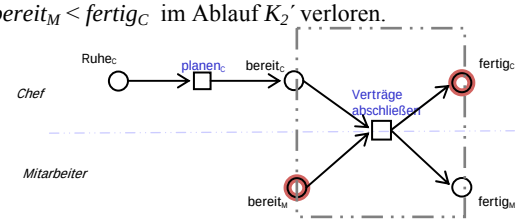


Abb. 5.3.1 Ein Ablauf K_2 von Σ_2 in Abb. 5.2.2

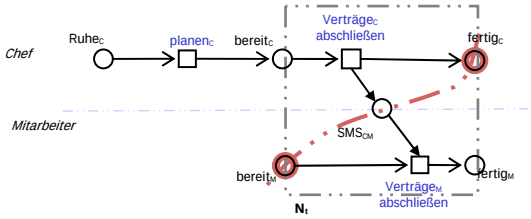


Abb. 5.3.2 Ein Ablauf K_2' von Σ_2' in Abb. 5.2.3

1. Erhaltenbleiben der Kausalordnung

Die folgenden Sätze klären den Zusammenhang zwischen Synchronisationsbedingungen, möglichen neuen Schnitten und dem Erhaltenbleiben der Kausalordnung in den Abläufen.

Wir notieren die Menge aller Stellen aus ${}^\bullet t$ für einen Agenten *a* mit $({}^\bullet t)_a$, d.h. $({}^\bullet t)_a = {}^\bullet t \cap a$.

Notation $(({}^\bullet t)_a \blacktriangleleft ({}^\bullet t)_b)$
Sei $\Sigma = ((P,T,F), m_0)$ *ein Agentensystem. Seien* $t \in T$ *eine unverteilte Transition der beiden Agenten a und b,* N_t *ein Standardverfeinerungsnetz für t.*
$(N_t, t^-) \models ({}^\bullet t)_a \blacktriangleleft ({}^\bullet t)_b$ *gdw.* $\forall\, p \in ({}^\bullet t)_a$ *und* $\forall\, q \in ({}^\bullet t)_b$ *gilt* $(N_t, t^-) \models p \blacktriangleleft q$.

Satz 5.3.1 (Erhaltenbleiben der Kausalordnung zwischen Bedingungen und Ereignissen)
Seien Σ *ein Agentensystem, t eine unverteilte Transition von Agenten a und b in* Σ*,* N_t *ein Standardverfeinerungsnetz von t. Erfülle* $\Sigma' = \Sigma(N_t \setminus t)$ *die Blockbedingung. Weiter seien* (K, r), (K', r') *jeweils Abläufe von* Σ, Σ' *mit:* (K', r')

ist eine $t \to N_t$ Substitution von (K, r), *wobei* $K = (B, E, \prec)$, $K' = (B', E', \prec')$.

Gelten $(\bullet t)_a \blacktriangleleft (\bullet t)_b$ *und* $(\bullet t)_b \blacktriangleleft (\bullet t)_a$ *in* (N_t, t^-), *so bleibt die Kausalitätsrelation* $<$ *von* K *zwischen Bedingungen und Ereignissen in* K' *erhalten* (*d.h.* $\forall\, s,t \in B \cup E: s \prec^+ t \to s \prec'^+ t$).

Beweis:
Neue Kausalordnungen können nicht hinzukommen, weil t stärker synchronisiert als N_t. Es reicht also zu zeigen, dass *co*-Mengen M' aus K' mit $M' \subseteq B$ auch *co*-Mengen in K sind.
Jede *co*-Menge kann zu maximalen *co*-Mengen, also Schnitten ergänzt werden. Es reicht also alle Schnitt in K' zu betrachten.
Wegen der 1-Beschränktheit von Agentennetzen schneidet jeder Schnitt C' in K' höchstens einen Ablauf von (N_t, t^-). Schneidet C' keinen Ablauf von (N_t, t^-), so ist C' ein Schnitt von K und wir sind fertig.

Im anderen Fall ist $M = C' \cap B$ eine *co*-Menge aus K'. Es ist zu zeigen, dass M auch *co*-Menge von K ist, also in einem Schnitt von K enthalten ist. Der Ablauf von (N_t, t^-) der geschnitten wird, sei K_{N_t} ($^\circ K_{N_t} = \bullet e$, $K_{N_t}{}^\circ = e^\bullet$, wobei $r(e) = t$).
Da wir nur Standardersetzungsnetze betrachten, haben vor C' bereits a) entweder t_{Start_a} oder t_{Start_b} geschaltet oder b) t_{Start_a} und t_{Start_b} haben geschaltet.
Fall a) t_{Start_b} hat noch nicht geschaltet.
Wegen $(\bullet t)_b \blacktriangleleft (t^\bullet)_a$ gilt $C' \cap K_{N_t}{}^\circ = \emptyset$ und $(\bullet e)_a = (^\circ K_{N_t})_a \to^* C' \setminus M$. Daraus folgt, dass $M \cup (\bullet e)_a$ ein Zustand von K ist.
Fall b) Da t_{Start_a} und t_{Start_b} vor C' geschaltet haben gilt, dass $C' \cap \bullet e = \emptyset$. Von dem Ablauf K_{N_t} haben bereits einige Transitionen geschlatet. Durch das Schalten der restlichen Transitionen dieses Ablaufs werden nur Bedingungen $K_{N_t}{}^\circ = e^\bullet$ belegt, d.h. es wird ein Schnitt erhalten, der M enthält und in K enthalten ist. □

2. Erhaltenbleiben der Kausalitäten im System

Folgerung 5.3.2 (von Satz 5.3.1, Erhaltenbleiben von Kausalitäten zwischen Stellen und Transitionen bei der verteilenden Verfeinerung)
Seien Σ *ein Agentensystem,* t *eine unverteilte Transition von Agenten* a *und* b *in* Σ, N_t *ein Standardverfeinerungsnetz von* t. $\Sigma' = \Sigma(N_t \setminus t)$ *erfülle die Blockbedingung.*

Gelten $(\bullet t)_a \blacktriangleleft (\bullet t)_b$ *und* $(\bullet t)_b \blacktriangleleft (\bullet t)_a$ *in* (N_t, t^-), *so gilt die Relation* $\blacktriangleleft$ *zwischen Stellen und Transitionen des Systems* Σ *auch im System* Σ'.

Beweis: Die Definition der Relation $\blacktriangleleft$ im System basiert alleine auf den Kausalitäten in den Abläufen der Systeme. Nach Satz 5.3.1 bleiben alle Kausalitäten in den Abläufen erhalten, falls $(\bullet t)_a \blacktriangleleft (\bullet t)_b$ und $(\bullet t)_b \blacktriangleleft (\bullet t)_a$ für die Verfeinerung gelten. □

3. Erhaltenbleiben der Lebendigkeitseigenschaften

Satz 5.3.3
Seien Σ *ein Agentensystem,* t *eine unverteilte Transition von Agenten* a *und* b *in* Σ, N_t *ein Standardverfeinerungsnetz von* t. *Erfülle* $\Sigma' = \Sigma(N_t \setminus t)$ *die Blockbedingung. Weiter seien* $p,q \in P$ *mit* $q \notin \bullet t$ *oder* $q \in (\bullet t)_b$ *mit* $(N_t, t^-) \models t_{Start_a} \blacktriangleleft t_{Start_b}$, *wobei* t_{Start_a} *und* t_{Start_b} *jeweils die Anfangstransition von* a *und* b *in* N_t *sind.*

Gelten $(\bullet t)_a \blacktriangleleft (t^\bullet)_b$ *und* $(\bullet t)_b \blacktriangleleft (t^\bullet)_a$ *in* (N_t, t^-), *so bleibt beim Verfeinerungsschritt* $(\Sigma \to \Sigma')$ *die Lebendigkeitseigenschaft* $p \triangleright q$ *erhalten.*

Beweis: Wir beweisen hier für den Fall $q \notin \bullet t$. Beweis für den Fall $q \in (\bullet t)_b$ mit $(N_t, t^-) \models t_{Start_a} \blacktriangleleft t_{Start_b}$ ist analog.
Wie im Beweis von Satz 5.3.1 betrachten wir die Schnitte C' von Σ' etwas genauer. Seien (K, r), (K', r') jeweils Abläufe von Σ, Σ' mit: (K', r') ist eine $t \to N_t$ Substitution von (K, r), wobei $K = (B, E, \prec)$, $K' = (B', E', \prec')$. Angenommen, in K gilt $p \triangleright q$. Zu zeigen ist: In K' gilt $p \triangleright q$.
Sei C ein Schnitt in K', in dem p gilt, d.h. es gibt ein $b_p \in C$ mit $r'(b_p) = p$. (1)
Zu zeigen ist: In K' gibt es einen von C erreichbaren Schnitt C', in dem q gilt. (2)
Falls C ein Schnitt von K ist, dann gilt aus der Voraussetzung: In K gibt es einen von C erreichbaren Schnitt C', in dem q gilt. Wegen Bem. 3.1.5 gilt: C' ist ein Schnitt von K' und in K' von C erreichbar. Also, (2) gilt.
Falls C kein Schnitt von K ist, dann gibt es in K' ein N_t-Auftreten K_{N_t} mit: K_{N_t} wird von C geschnitten. Vor C haben entweder die beiden Anfangstransitionen t_{Start_a} und t_{Start_b} im Verfeinrungsnetz N_t von Agenten a und b geschaltet, oder nur eine von den beiden t_{Start_a} und t_{Start_b} geschaltet.
Fall 1. Vor C haben t_{Start_a} und t_{Start_b} geschaltet.
Dann ist in K' ein Schnitt C_1 von C erreichbar, wobei $C_1 = (C \cap B) \cup M$, $M = \{\, b \in K_{N_t}{}^\circ \mid \exists\, b' \in C: b' \prec'^+ b \}$. C_1 ist ein Schnitt von K und in C_1 gilt p, da $b_p \in C \cap B$ gilt. Wegen Voraussetzung gilt: In K gibt es einen von C_1 erreichbaren Schnitt C', in dem q gilt. Wegen Bem. 3.1.5 gilt: C' ist eine Schnitt von K', in K' von C_1 erreichbar und in C' gilt q. C' ist auch von C erreichbar. Daraus folgt, (2) gilt.
Fall 2. Vor C hat nur eine von t_{Start_a} und t_{Start_b} geschaltet. Angenommen, t_{Start_a} hat geschaltet und t_{Start_b} noch nicht.
Sei $C = (C \cap B) \cup N$. Weiter sei $C_0 = (C \cap B) \cup (^\circ K_{N_t})_a$ mit $r'((^\circ K_{N_t})_a) = (\bullet t)_a$. Dann gilt: In K' ist C von C_0 erreichbar und C_0 ist ein Schnitt von K und in C_0 gilt p. Aus der Voraussetzung gibt es in K einen von C_0 erreichbaren Schnitt C', in dem q gilt, d.h. es gibt ein $b_q \in C'$ mit $r(b_q) = q$. Angenommen, C' wird durch das Schalten von $e_1, e_2, \ldots, e_n$ in K erreicht, d.h. $C_0 \xrightarrow{e_1} C_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} C_n$, wobei $C_n = C'$.
Sei $e \in E$ mit $\bullet e = {}^\circ K_{N_t}$ und $e^\bullet = K_{N_t}{}^\circ$. Für $i = 0,1,..,n$, $C_i' := C_i \setminus (^\circ K_{N_t})_a \cup N$, falls für jedes $1 \le j \le i$ $e_j \ne e$ gilt; $C_i' := C_i$, falls es ein $1 \le j \le i$ gibt, so dass $e_j = e$ gilt. Wegen $b_q \in C_n$ und

$q \notin {}^\bullet t$ gilt $b_q \in C_n'$. Falls in C_0 q gilt (n=0), dann gilt q auch in C. Daraus folgt: (2) gilt. Für den anderen Fall brauchen wir zu zeigen, C_n' ist in K' von C erreichbar. Wir zeigen durch Induktion: Für jedes $k=0,1,..,n$ gilt:

C_k' ist in K' von C erreichbar. (3)

1. Für $k=0$ gilt offensichtlich: $C_0'=C$ ist in K' von C erreichbar.

2. Für $k=i-1$ gelte (3), d.h. $C \rightarrow^* C_{i-1}'$. Zu zeigen ist: Für $k=i$ gilt (3), d.h. $C \rightarrow^* C_i'$.

a) Falls $e_i=e$: Dann gelten $C_{i-1}'=C_{i-1} \setminus ({}^\circ K_{N_t})_a \cup N$ und $C_i'=C_i$. Wegen $C_{i-1} \xrightarrow{e} C_i$ gelten ${}^\circ K_{N_t} = {}^\bullet e \subseteq C_{i-1}$ und $K_{N_t}{}^\circ = e^\bullet \subseteq C_i$. Daraus folgt: In K' gilt $C_{i-1} \setminus ({}^\circ K_{N_t})_a \cup N \rightarrow^* C_i$, d.h. $C_{i-1}' \rightarrow^* C_i'$. Aus der Induktionsvoraussetzung folgt, C_i' ist von C erreichbar.

b) Falls $e_i \neq e$ und e nicht vor e_i schaltet, d.h. $e_j \neq e$ für jedes $j=0,1,..,i-1$. Dann gelten $C_{i-1}'=C_{i-1} \setminus ({}^\circ K_{N_t})_a \cup N$ und $C_i'=C_i \setminus ({}^\circ K_{N_t})_a \cup N$. Wegen $C_{i-1} \xrightarrow{e_i} C_i$ und ${}^\bullet e_i \cap ({}^\circ K_{N_t})_a = \emptyset$ gilt: e_i ist auch in $C_{i-1} \setminus ({}^\circ K_{N_t})_a \cup N$ aktiviert und

$C_{i-1} \setminus ({}^\circ K_{N_t})_a \cup N \xrightarrow{e_i} C_i \setminus ({}^\circ K_{N_t})_a \cup N$, d.h. $C_{i-1}' \xrightarrow{e_i} C_i'$. Aus der Induktionsvoraussetzung folgt, C_i' ist von C erreichbar.

c) Falls e vor e_i schaltet, d.h. $e_j=e$ für ein $1 \leq j \leq i-1$. Dann gelten $C_{i-1}'=C_{i-1}$ und $C_i'=C_i$. Wegen $C_{i-1} \xrightarrow{e_i} C_i$ nach Bem. 3.1.5 gilt in K': $C_{i-1} \rightarrow^* C_i$, d.h. $C_{i-1}' \rightarrow^* C_i'$. Aus der Induktionsvoraussetzung folgt, C_i' ist von C erreichbar. □

Folgerung 5.3.4 (von Satz 5.3.3)

$\Sigma' = \Sigma(N_t \setminus t)$ erfülle die Blockbedingung. Seien $R,Q \subseteq P$ mit $Q \cap {}^\bullet t = \emptyset$.

Gelten $({}^\bullet t)_a \blacktriangleleft (t^\bullet)_b$ und $({}^\bullet t)_b \blacktriangleleft (t^\bullet)_a$ in (N_t, t^-), so bleibt $R \triangleright Q$ beim Verfeinerungsschritt $(\Sigma \rightarrow \Sigma')$ erhalten.

Die folgende Definition werden wir verwenden, um eine Art von Eigenschaften eines Systems zu formulieren, die für Existenz eines verteilenden Verfeinerungsnetzes wichtig ist (siehe Abschnitt 6.3).

Definition 5.3.5 (leadsto ohne Zugriff auf eine (oder mehrere) Stellen)

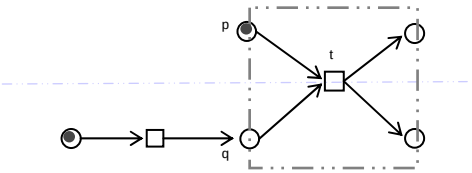
Sei $\Sigma=((P,T,F), m_0)$ ein System und seien $p,q \subseteq P$.

i. Sei (K, r) ein Ablauf von Σ. $p(\triangleright \backslash p)q$ gilt in K, gdw. für jeden Schnitt C von K mit $b \subseteq C$ und $r(b)=p$ gibt es einen von C erreichbaren Schnitt C', der b enthält und in dem q gilt.

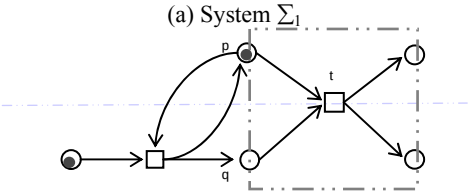
ii. $p(\triangleright \backslash p)q$ gilt in Σ, gdw. in jedem Ablauf von Σ $p(\triangleright \backslash p)q$ gilt.

Der Einfachheit halber schreiben wir auch $p(\triangleright \backslash p)q$ anstelle $\{p\}$ $(\triangleright \backslash \{p\})$ $\{q\}$, wobei $p,q \in P$ und P die Stellenmenge eines Systems Σ ist.

Beispiel:
In Abb. 5.3.3 sind zwei Systeme Σ_1 und Σ_2 angegeben. $p(\triangleright \backslash p)$ q gilt in Σ_1 aber nicht in Σ_2. $p \triangleright q$ gilt in beiden Systemen.



(a) System Σ_1



(b) System Σ_2

Abb. 5.3.3 Illustration für $p(\triangleright \backslash p)q$

Bemerkung 5.3.6

Sei $\Sigma=((P,T,F), m_0)$ ein System und seien $p,q \subseteq P$.

Es gilt $p(\triangleright \backslash p)q \rightarrow p \triangleright q$ aber nicht $p \triangleright q \rightarrow p(\triangleright \backslash p)q$.

Beweis: durch Beispiel in Abb. 5.3.3.

Bemerkung 5.3.7

Sei $\Sigma=((P,T,F), m_0)$ ein System und seien $p,q \subseteq P$.

Dann gilt: $\Sigma \models \Box(p \rightarrow q) \rightarrow \Sigma \models p(\triangleright \backslash p)q$.

Folgerung 5.3.8 (von Satz 5.3.3)

$\Sigma' = \Sigma(N_t \setminus t)$ erfülle die Blockbedingung. Seien $p,q \in P$ mit $q \notin {}^\bullet t$.

Gelten $({}^\bullet t)_a \blacktriangleleft (t^\bullet)_b$ und $({}^\bullet t)_b \blacktriangleleft (t^\bullet)_a$ in (N_t, t^-), so bleibt $p(\triangleright \backslash p)q$ beim Verfeinerungsschritt $(\Sigma \rightarrow \Sigma')$ erhalten.

Beweis:

Angenommen, $\Sigma \models p(\triangleright \backslash p)q$ gilt. Zu zeigen ist, $\Sigma' \models p(\triangleright \backslash p)q$ gilt.

Angenommen, C ist ein Schnitt in einem Ablauf (K', r') von Σ', in dem p gilt, d.h. es gibt ein $b_p \in C$ mit $r'(b_p)=p$. Zu zeigen ist: Es gibt einen von C erreichbaren Schnitt C' in K', der b_p enthält und in dem q gilt.

Wie im Beweis von $p \triangleright q$ (Satz 5.3.3) die drei Fälle von Schnitten betrachten. Wenn vorher $p \triangleright q$ mit $b_p \in C_0$, $C_0 \xrightarrow{e_1} C_1 \xrightarrow{e_2} \ldots \xrightarrow{e_k} C_k$ und $b_q \in C_k$ mit $r(b_q)=q$ galt, dann gilt jetzt zusätzlich für die e_i mit $r(e_i)=t_i$ und $p \notin {}^\bullet t_i$. Also gilt das jetzt auch für Σ'. □

Lemma 5.3.9

Seien Σ ein Agentensystem, t eine unverteilte Transition von Agenten a und b in Σ, N_t ein Standardverfeinerungsnetz von t. $\Sigma' = \Sigma(N_t \setminus t)$ erfülle die Blockbedingung.

i. Gilt ${}^\bullet p \blacktriangleleft q^\bullet$ mit $p \in ({}^\bullet t)_a$ und $q \in (t^\bullet)_b$ in (N_t, t^-), so bleibt $p \triangleright q$ bei $(\Sigma \rightarrow \Sigma')$ erhalten.

ii. Gilt ${}^\bullet p \blacktriangleleft q^\bullet$ mit $p \in ({}^\bullet t)_a$ und $q \in ({}^\bullet t)_b$ in (N_t, t^-), so bleibt $p \triangleright q$ bei $(\Sigma \rightarrow \Sigma')$ erhalten.

Beweis: Wir beweisen hier nur (*i*). Beweis für (*ii*) ist analog.

Seien (K, r), (K', r') Abläufe von Σ, Σ' mit: (K', r') ist eine $t \rightarrow N_t$ Substitution von (K, r), wobei $K=(B, E, \prec)$, $K'=(B', E', \prec')$.

Angenommen, in K glit $p \triangleright q$. (1)

Zu zeigen ist: Auch in K' gilt $p \triangleright q$. Sei C ein Schnitt von K', in dem p gilt, d.h. es gibt ein $b_p \in B'$ mit $r'(b_p) = p$. Wir brauchen nur zu zeigen:

Es gibt einen von C erreichbaren Schnitt C', in dem q gilt. (2)

Es gibt folgende drei Fälle.

1. Direkt auf b_p folgt ein N_t-Auftreten K_{N_t}, d.h. $b_p \in {}^\circ K_{N_t}$.

Dann gibt es in K' ein $b_q \in B'$ mit: $r'(b_q) = q$ und $b_p \in K_{N_t}{}^\circ$.

Wegen ($p^\bullet \blacktriangleleft {}^\bullet q$) gilt $b_p \prec'^{+} b_q$. Daraus folgt: Es gibt einen von C erreichbaren Schnitt C', der b_q enthält. Also, (2) gilt.

2. Direkt auf b_p folgt kein N_t –Auftreten und C ist ein Schnitt von K.

Dann gibt es in K einen von C erreichbaren Schnitt C', in dem q gilt (wegen (1)). C' ist auch ein Schnitt von K', in dem q gilt und C' ist auch in K' von C erreichbar (wegen Bem. 3.1.5). Daraus folgt: (2) gilt.

3. Direkt auf b_p folgt kein N_t –Auftreten und C ist kein Schnitt von K.

Dann gibt es ein N_t-Auftreten K_{N_t} in K', das von C geschnitten wird. Es gibt zwei Fälle: $b_p \in C$ liegt entweder a) vor K_{N_t} oder b) nach K_{N_t}.

Fall a): Es gibt $b_1, b_2 \in {}^\circ K_{N_t}$, $b' \in C$ mit $b_p \prec'^{+} b_1$ und $b_2 \prec'^{+} b'$.

Dann gibt es ein $b_p' \in {}^\circ K_{N_t}$ mit $r'(b_p') = p$ und $b_p \prec'^{+} b_p'$.

Aus 1 folgt: (2) gilt.

Fall b): Es gibt $b_1, b_2 \in {}^\circ K_{N_t}$, $b' \in C$ mit $b' \prec'^{+} b_1$ und $b_2 \prec'^{+} b_p$.

Dann gibt es in K' einen von C erreichbaren Schnitt C', der b_p enthält und ein Schnitt von K ist. Aus 2 folgt: (2) gilt.

Daraus folgt die Behauptung. □

Das folgende Lemma werden wir beim Anwendungsbeispiel (siehe Abschnitt 7.1) verwenden.

Lemma 5.3.10

Seien Σ ein Agentensystem mit Agentenmenge A, t eine unverteilte Transition in Σ, N_t ein Standardverfeinerungsnetz von t. Seien $p,q \in P$ mit $p \sim_A q$ und $\Sigma \models p+q \leq 1$. Erfülle $\Sigma' = \Sigma(N_t \setminus t)$ die Blockbedingung.

Dann *bleibt $p \triangleright q$ bei $(\Sigma \rightarrow \Sigma')$ erhalten.*

Beweis: siehe [Wu07].

4. Erhaltenbleiben der Sicherheitseigenschaften

Lemma 5.3.11

$\Sigma' = \Sigma(N_t \setminus t)$ erfülle die Blockbedingung. Seien $p,q \in P$.

Gelten $({}^\bullet t)_a \blacktriangleleft (t^\bullet)_b$ und $({}^\bullet t)_b \blacktriangleleft (t^\bullet)_a$ in (N_t, t^-), so bleibt im Verfeinerungsschritt $(\Sigma \rightarrow \Sigma')$ die Sicherheitseigenschaft $\Box \neg (p \wedge q)$ erhalten.

Beweis: Angenommen, in Σ gilt: $\Box \neg (p \wedge q)$ (1)

Zu zeigen ist: In Σ' gilt $\Box \neg (p \wedge q)$ (2)

Seien (K, r), (K', r') Abläufe von Σ, Σ' mit: (K', r') ist eine $t \rightarrow N_t$ Substitution von (K, r), wobei $K = (B, E, \prec)$, $K' = (B', E', \prec')$. Seien $b_p, b_q \in B'$ beliebige Auftreten von p, q in K', d.h. $r'(b_p) = p$, $r'(b_q) = q$.

Dann gilt (wegen Bem. 3.1.5): $b_p, b_q \in B$ und $r(b_p) = r'(b_p) = p$, $r(b_q) = r'(b_q) = q$. Wegen (1) gilt in K: $\neg (b_p \underline{\text{co}}\ b_q)$. Dann gilt: entweder $b_p \prec^+ b_q$ oder $b_q \prec^+ b_p$. In K' gilt weiter: $b_p \prec'^{+} b_q$ falls $b_p \prec^+ b_q$; $b_q \prec'^{+} b_p$ falls $b_q \prec^+ b_p$ (wegen Satz 5.3.1). D.h. in K' gilt $\neg (b_p \underline{\text{co}}\ b_q)$. Daraus folgt: (2) gilt. □

Aus dem Satz 5.3.1 können wir die folgende Bemerkung (Bem. 5.3.12) erhalten. Diese Bemerkung impliziert: Wenn eine Transition t_2' im System Σ vor der Verfeinerung eine nicht aktivierte Konflikt-Transition von t_2 ist (d.h. ${}^\bullet t_2 \cap {}^\bullet t_2' \neq \emptyset$ und $\Box ({}^\bullet t_2 \rightarrow \neg {}^\bullet t_2')$, dann ist t_2' auch im System $\Sigma' = \Sigma(N_{t_1} \setminus t_1)$ nach der Verfeinerung von t_1 eine nicht aktivierte Konflik-Transition von t_2.

Beim Nachweis der Blockbedingung müssen wir oft zeigen, dass eine konkurrente Transition nicht aktiviert ist (vgl. Abschnitt 6.1, 6.2 und 6.4). Bei der Vertauschbarkeit der Verfeinerungsschritte (vgl. Abschnitt 6.5) müssen wir dann entsprechend zeigen, dass eine vor einer Verfeinerung nicht aktivierte konkurrente Transition nach der Verfeinerung auch nicht aktiviert ist. Dort werden wir diese Bemerkung verwenden.

Bemerkung 5.3.12

$\Sigma' = \Sigma(N_t \setminus t)$ erfülle die Blockbedingung. Seien $Q \subseteq P$, $p \in P$.

Gelten $({}^\bullet t)_a \blacktriangleleft (t^\bullet)_b$ und $({}^\bullet t)_b \blacktriangleleft (t^\bullet)_a$ in (N_t, t^-), so bleibt $\Box(Q \rightarrow \neg p)$ beim Verfeinerungsschritt $(\Sigma \rightarrow \Sigma')$ erhalten.

Beweis: siehe [Wu07].

Lemma 5.3.13

Seien Σ ein Agentensystem, t eine unverteilte Transition von Agenten a und b in Σ, N_t ein Standardverfeinerungsnetz von t. $\Sigma' = \Sigma(N_t \setminus t)$ erfülle die Blockbedingung.

i. Gilt $p^\bullet \blacktriangleleft {}^\bullet q$ mit $p \in ({}^\bullet t)_a$ und $q \in (t^\bullet)_b$ in (N_t, t^-), so bleibt $\Box \neg (p \wedge q)$ bei $(\Sigma \rightarrow \Sigma')$ erhalten.

ii. Gilt ${}^\bullet p \blacktriangleleft {}^\bullet q$ mit $p \in (t^\bullet)_a$ und $q \in (t^\bullet)_b$ in (N_t, t^-), so bleibt $\Box (q \rightarrow p)$ bei $(\Sigma \rightarrow \Sigma')$ erhalten.

iii. Gilt $p^\bullet \blacktriangleleft q^\bullet$ mit $p \in ({}^\bullet t)_a$ und $q \in ({}^\bullet t)_b$ in (N_t, t^-), so bleibt $\Box (p \rightarrow q)$ bei $(\Sigma \rightarrow \Sigma')$ erhalten.

Beweis: siehe [Wu07].

Das folgende Lemma werden wir beim Anwendungsbeispiel (siehe Abschnitt 7.1) verwenden.

Lemma 5.3.14

Seien Σ ein Agentensystem, t eine unverteilte Transition von Agenten a und b in Σ, N_t ein Standardverfeinerungsnetz von t. Weiter seien $p,q,r \in P$ mit $p \in ({}^\bullet t)_a$, $q \in (t^\bullet)_b$ und $\Sigma \models p+q+r \leq 1$. $\Sigma' = \Sigma(N_t \setminus t)$ erfülle die Blockbedingung.

Gilt $p^\bullet \blacktriangleleft {}^\bullet q$ in (N_t, t^-), so bleiben $\Box \neg (p \wedge r)$ und $\Box \neg (q \wedge r)$ bei $(\Sigma \rightarrow \Sigma')$ erhalten.

Beweis: siehe [Wu07].

6. Nachweis der Blockbedingung bei der verteilenden Verfeinerung

- Grundtypen von Verfeinerungsnetzen

Die Blockbedingung ist eine Voraussetzung für eine korrekte Transitionsverfeinerung. Beim Entwurf fragt man deshalb, wie man ein Verfeinerungsnetz konstruiert, um die

Blockbedingung zu garantieren. Und bei der Verifikation fragt man, wann die Blockbedingung erfüllt ist. Dafür brauchen wir Kriterien für die Blockbedingung.

Ob die Blockbedingung bei einem Verfeinerungsschritt erfüllt ist, das hängt von der Umgebung der zu verfeinernden Aktion ab. Wir geben die Kriterien für unterschiedliche Umgebungen an, die besagen, unter welchen Bedingungen ein Ersetzungsnetz die Blockbedingung garantiert.

Wir haben drei Grundtypen von Ersetzungsnetzen herausgearbeitet, die in Abhängigkeit von den vorangegangenen Absprachen der Agenten zu verwenden sind. Bei dem ersten Grundtyp (*abgesprochene gemeinsame Aktion* genannt) haben sich die Agenten vorher gegenseitig informiert. Bei dem zweiten Grundtyp (*erbetene Zusammenarbeit*), hat ein Agent dem anderen Agenten eine Nachricht geschickt und bei dem dritten Grundtyp (*erwartete Zusammenarbeit*) ist ohne vorbereitende Kommunikation durch den Algorithmus gesichert, dass die Aktion gemeinsam ausgeführt wird.

Bei der Verfeinerung einer Transition t muss beachtet werden, dass alle Stellen aus dem Nachbereich von t kausal nach den Stellen aus dem Vorbereich von t belegt werden. Die drei Typen von Ersetzungsnetzen schwächen diese Eigenschaft unterschiedlich stark ab. Die abgesprochene gemeinsame Aktion benötigt keine Kommunikation im Ersetzungsnetz und schwächt damit die Kausalität zwischen der Belegung des Vorbereichs und Nachbereichs von t am stärksten ab. Die erwartete Zusammenarbeit erfordert im Ersetzungsnetz zwei Nachrichten und erhält vollständig die Kausalitätseigenschaft. Um im verteilten Algorithmus mit möglichst wenigen Kommunikationen auszukommen, sollte immer versucht werden, den Typ von Verfeinerungsnetzen zu benutzen, der die wenigsten Kommunikationen enthält.

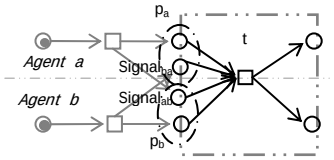
In den Abschnitten 6.1, 6.2 und 6.4 werden diese drei Grundtypen diskutiert.

Im Abschnitt 6.3 werden wir eine notwendige und hinreichende Anforderung an die Umgebung einer unverteilten Transition angeben, die erfüllt sein muß, damit überhaupt eine verteilende Verfeinerung existiert.

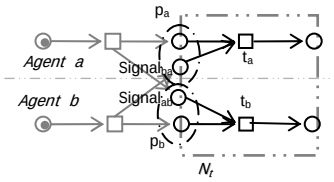
Eine weitere interessante Frage bei der Verfeinerung ist die Vertauschbarkeit von Verfeinerungsschritten. Angenommen, der Anfangsalgorithmus Σ_0 hat n zu verfeinernden Transitionen $t_1,t_2,..,t_n$. Dann ist der Verfeinerungsprozess $\Sigma_0 \rightarrow \Sigma_1 \rightarrow ... \rightarrow \Sigma_n$, wobei $\Sigma_k = \Sigma_{k-1}(N_{t_k} \backslash t_k)$, $k=1,2,..,n$. Wann ist die Reihenfolge von Verfeinerungsschritten beliebig? Für jeden Verfeinerungsschritt $\Sigma_k = \Sigma_{k-1}(N_{t_k} \backslash t_k)$, $k=1,2,..,n$, müssen wir für die Blockbedingung die Umgebung von t_k in Σ_{k-1} analysieren. Im Verfeinerungsprozess wird Σ_k immer komplizierter und die Analyse für Σ_k wird immer schwerer. Es wäre viel einfacher, wenn wir für jede Transition t_k nur den Verfeinerungsschritt $(\Sigma_0 \rightarrow \Sigma_k')$ mit $\Sigma_k' = \Sigma_0(N_{t_k} \backslash t_k)$ zu betrachten brauchen. Unter welcher Bedingung geht das? Diese Frage werden wir im Abschnitt 6.5 diskutieren und ein Kriterium für die Vertauschbarkeit von Verfeinerungsschritten beweisen. Leider ist es uns bisher nicht gelungen, ein so allgemeines Kriterium für die Vertauschbarkeit zu finden, dass es für alle Transitionen in den beiden abschließenden Beispielen anwendbar ist. Dadurch sind die Beweise für die Beispiele leider etwas umfangreicher als sie sein müssten.

6.1 Der erste Grundtyp: Abgesprochene Zusammenarbeit

Dieser Grundtyp passt zu dem folgenden Fall: Beide Agenten haben vorher (vor der Transition t) gegenseitig Nachrichten geschickt. D.h. jeder Agent muss warten, bis er ein Signal von seinem Partner – dem anderen Agenten – bekommen hat. Erst danach kann er die Aufgabe für t machen. In diesem Fall brauchen die beiden im Ersetzungsnetz (also Verfeinerungsnetz) keine Kommunikation mehr, um die Blockbedingung zu erfüllen.



(a) Vor der Verfeinerung von t



(b) Nach der Verfeinerung von t

Abb. 6.1.1 Der erste Grundtyp: *Abgesprochene Zusammenarbeit*

Als Verfeinerungsnetze lassen wir wieder nur Standardverfeinerungsnetze zu, d.h. jeder Agent hat genau eine Anfangstransition im Verfeinerungsnetz. Wieder notieren wir die Menge aller Stellen aus $\bullet t$ für einen Agenten a mit $(\bullet t)_a$, also $(\bullet t)_a := \bullet t \cap a$. Analog $(t\bullet)_a := t\bullet \cap a$, $(\bullet t)_b := \bullet t \cap b$ und $(t\bullet)_b := t\bullet \cap b$.

Satz 6.1.1
Seien Σ ein Agentensystem, t eine unverteilte Transition von Agenten a und b in Σ, N_t ein Standardverfeinerungsnetz von t. In Σ gelte: i. $(\bullet t)_a \leftrightarrow (\bullet t)_b$; ii. Es gibt keine konkurrente Transition zu t. Dann erfüllt $\Sigma' = \Sigma(N_t \backslash t)$ *die Blockbedingung.*

Bevor wir den Satz beweisen, geben wir eine Bemerkung an, die wir beim Nachweis der Gültigkeit der Blockbedingung oft verwenden können. Diese Bemerkung zeigt, dass immer nur die Transitionen von N_t zu betrachten sind, die Stellen aus $\bullet t$ im Vorbereich haben.

Bemerkung 6.1.2 *Sei $\Sigma \rightarrow \Sigma'$ mit $\Sigma' = \Sigma(N_t \backslash t)$ eine verteilende Systemtransformation. K' sei ein Ablauf von Σ'. C sei Schnitt in K'. In C beginnt ein Ablauf von (N_t, t^-), wenn in C ein Anfangsstück K_A eines Ablaufs von (N_t, t^-) mit $^\circ K_A \cap K_A^\circ = \emptyset$ beginnt, d.h. jede Marke aus $^\circ K_A$ wurde bereits im Anfangsstück von K_A konsumiert.*
Beweis: Für (N_t, t^-) ist das nach Voraussetzung erfüllt. In Σ' gilt das weiter, da keine Transition von $\Sigma \backslash \{t\}$ konkurrent zu einer Transition t' von N_t mit $\bullet t' \cap \bullet t = \emptyset$ sein

kann, wegen $P \cap P_{t\text{-}intern} = \emptyset$, wobei $P_{t\text{-}intern} = P_{N_t} \setminus (\bullet t \cup t\bullet)$. Das Anfangsstück K_A muss also wie in (N_t, t^-) zu einem Ablauf von (N_t, t^-) vervollständigt werden. □

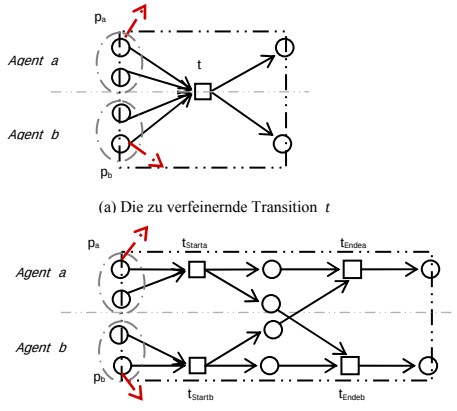
Wir werden hier in dieser Arbeit nur den Beweis für den Satz 6.1.1 angeben. Beweisideen für die weiteren Kriterien sind ähnlich (siehe [Wu07]).

Beweis von Satz 6.1.1:

Zu zeigen ist: Jeder Ablauf von Σ' wird aus einem Ablauf von Σ erhalten, indem jedes Auftreten von t durch einen vollständigen Ablauf von (N_t, t^-) ersetzt wird. Wir nehmen an, es gibt einen Ablauf (K', r') von Σ', für den das nicht der Fall ist. Diese Annahme ist zum Widerspruch zu führen.

Aus der Annahme folgt: (K', r') enthält einen unvollständigen Ablauf von (N_t, t^-). Es gibt einen Schnitt C, in dem dieser unvollständige Ablauf beginnt, also t_{Start_a} oder t_{Start_b} schaltet. Aus Symmetriegründen können wir annehmen, dass in C t_{Start_a} schaltet, also $M \subseteq C$ mit $r'(M) = \bullet t_{Start_a}$. Nach Voraussetzung gilt in C $(\bullet t)_a \rightarrow (\bullet t)_b = \bullet t_{Start_b}$. Da es nach Voraussetzung keine konkurrenten Transtionen zu t und t_{Start_b} gibt, schaltet auch t_{Start_b} im Schnitt C. Nach Bem. 6.1.2 ist das ein Widerspruch dazu, dass in C kein vollständiger Ablauf von (N_t, t^-) beginnt.
Daraus folgt die Behauptung. □

Im Satz 6.1.1 wird gefordert, dass die Agenten a und b keine Konflikt-Transitionen zu t haben. Diese Forderung kann wie folgt abgeschwächt werden: Die Agenten a und b haben zwar Konflikt-Transitionen zu t, diese Konflikt-Transitionen sind aber nicht aktiviert. Außerdem verlangen wir nicht $(\bullet t)_a \rightarrow (\bullet t)_b$ und $(\bullet t)_b \rightarrow (\bullet t)_a$, sondern nur $(\bullet t)_a \triangleright (\bullet t)_b$ und $(\bullet t)_b \triangleright (\bullet t)_a$. In diesem abgeschwächten Fall erfüllt das erhaltene System nach der Verfeinerung die Blockbedingung, wenn das Ersetzungsnetz zusätzlich noch die Kausalitäten $(\bullet t)_a \blacktriangleleft (t\bullet)_b$ und $(\bullet t)_b \blacktriangleleft (t\bullet)_a$ erfüllt (siehe Abb. 6.1.2).



(a) Die zu verfeinernde Transition t

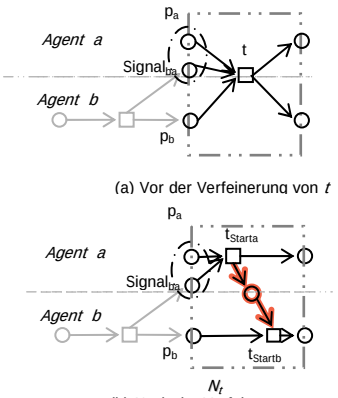
(b) Das Ersetzungsnetz N_t

Abb. 6.1.2 Illustration für Satz 6.1.4 (Abschwächung von Satz 6.1.2)

Satz 6.1.3 (*Abschwächung von Satz 6.1.1*)
Seien Σ ein Agentensystem, t eine unverteilte Transition von Agenten a und b in Σ, N_t ein Standardverfeinerungsnetz von t. In Σ gelte: i. $(\bullet t)_a \triangleright (\bullet t)_b$ und $(\bullet t)_b \triangleright (\bullet t)_a$; ii. Es gibt keine aktivierte konkurrente Transition zu $(\bullet t)_a$ und $(\bullet t)_b$. Dann erfüllt $\Sigma' = \Sigma(N_t \setminus t)$ *die Blockbedingung, wenn in* (N_t, t^-) $(\bullet t)_a \blacktriangleleft (t\bullet)_b$ *und* $(\bullet t)_b \blacktriangleleft (t\bullet)_a$ *gelten.*

6.2 Der zweite Grundtyp: erbetene Mithilfe

Dieser Grundtyp passt zu dem folgenden Fall: Einer (z.B. Agent b) der beiden Agenten schickt vor der Transition t eine Nachricht an a. Der Agent a muss warten, bis er ein Signal von seinem Partner bekommen hat. Erst danach kann er die Aufgabe für t machen. In diesem Fall muss dieser Agent a im Ersetzungsnetz eine Nachricht an b schicken, damit b nur seine Arbeit beginnt, wenn a seine Arbeit erledigen wird, also die Blockbedingung erfüllt ist.



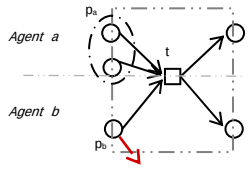
(a) Vor der Verfeinerung von t

(b) Nach der Verfeinerung von t

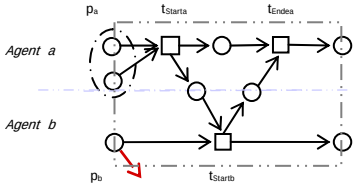
Abb. 6.2.1 Der zweite Grundtyp: *Erbetene Mithilfe*

Satz 6.2.1
Seien Σ ein Agentensystem, t eine unverteilte Transition von Agenten a und b in Σ, N_t ein Standardverfeinerungsnetz von t. In Σ gelte: i. $(\bullet t)_a \rightarrow (\bullet t)_b$; ii. Es gibt keine konkurrente Transition zu t. Dann erfüllt $\Sigma' = \Sigma(N_t \setminus t)$ *die Blockbedingung, wenn in* (N_t, t^-) $t_{Start_a} \blacktriangleleft t_{Start_b}$ *gilt, wobei* t_{Start_a} *und* t_{Start_b} *jeweils die Anfangstransition von a und b in N_t sind.*

Im Satz 6.2.1 wird gefordert, dass der Agent b keine Konflikt-Transitionen zu t hat. Diese Forderung kann wie folgt abgeschwächt werden: Der Agent b hat zwar Konflikt-Transitionen zu t, diese Konflikt-Transitionen sind aber nicht aktiviert. Und die Bedingung $(\bullet t)_a \rightarrow (\bullet t)_b$ an die Umgebung im Satz 6.2.1 kann zu $(\bullet t)_a \triangleright (\bullet t)_b$ abgeschwächt werden. In diesem abgeschwächten Fall erfüllt das erhaltene System nach der Verfeinerung die Blockbedingung, wenn das Ersetzungsnetz zusätzlich noch die Synchronisationsbedingungen $(\bullet t)_a \blacktriangleleft (t\bullet)_b$ und $(\bullet t)_b \blacktriangleleft (t\bullet)_a$ erfüllt (siehe Abb. 6.2.2).



(a) Die zu verfeinernde Transition t



(b) Das Ersetzungsnetz N_t

Abb. 6.2.2 Illustration für Satz 6.2.2 (Abschwächung von Satz 6.2.1)

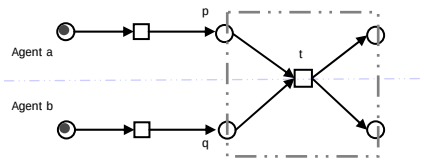
Satz 6.2.2 (*Abschwächung von Satz 6.2.1*)
Seien $\sum$ ein Agentensystem, t eine unverteilte Transition von Agenten a und b in $\sum$, N_t ein Standardverfeinerungsnetz von t. In $\sum$ gelte: i. $(\bullet t)_a \triangleright (\bullet t)_b$; ii. Es gibt keine aktivierte konkurrente Transition zu $(\bullet t)_a$ und $(\bullet t)_b$. Dann erfüllt *$\sum' = \sum(N_t \setminus t)$ die Blockbedingung, wenn in (N_t, t^-) $(\bullet t)_a \blacktriangleleft (t^\bullet)_b$, $(\bullet t)_b \blacktriangleleft (t^\bullet)_a$ und $t_{Start_a} \blacktriangleleft t_{Start_b}$ gelten, wobei t_{Start_a} und t_{Start_b} jeweils die Anfangstransition von a und b in N_t sind.*

6.3 Grenzen für verteilende Verfeinerungen

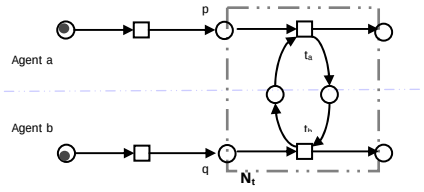
Die bisher bewiesenen Kriterien zeigen notwendige Anforderungen an die Umgebung einer unverteilten Transition t, die erfüllt sein müssen, damit eine bestimmte Form von Verfeinerungen, die Blockbedingung garantiert. Wir gehen jetzt der Frage nach, wie eine notwendige und hinreichende Anforderung aussieht, damit überhaupt eine verteilende Verfeinerung existiert. Die Beantwortung dieser Frage ist für den schrittweisen Entwurf verteilter Algorithmen wichtig. Wenn der Entwurf mit einem einfachen unverteilten Agentensystem beginnt, dann muss das System für alle unverteilten Transitionen zumindestens diese schwächste Bedingung erfüllen. Aus der hinreichenden Bedingung lassen sich weitere Kriterien zum Nachweis der Blockbedingung ableiten, indem man diese Bedingung verschärft. Satz 6.4.3 ist ein Beispiel dafür.

Vorher geben wir noch eine Bemerkung an, die besagt, nach dem Verteilen einer unverteilten Transition t können Token an den Stellen aus $\bullet t$ unterschiedlicher Agenten nicht mehr gleichzeitig konsumiert werden.

Bemerkung 6.3.1 (*Gleichzeitige Nutzung von $(\bullet t)_a$ und $(\bullet t)_b$ gehen verloren*)
Sei $\sum' = \sum(N_t \setminus t)$, wobei N_t ein Standardverfeinenrungsnetz von t ist. Nach der Verfeinerung gilt für alle $p \in (\bullet t)_a$ und $q \in (\bullet t)_b$ höchstens $p \blacktriangleleft q$ oder $q \blacktriangleleft p$.
Beweis: Falls beides gilt, dann besitzt (N_t, t^-) keinen Ablauf, der in t^+ endet, weil dann das System blockiert. □



(a) System $\sum$



(b) System $\sum(N_t \setminus t)$

Abb. 6.3.1 Illustration für Bem. 6.3.1

Wie in Abschnitt 5.3 bereits definiert, für Stellen p,q eines Systems $\sum$ gilt $p(\triangleright\setminus p)q$ in einem Ablauf (K, r) von $\sum$, gdw. für jeden Schnitt C von K, in dem p markiert ist (d.h. es gibt ein $b \in C$ mit $r(b)=p$), gibt es eine *co*-Menge von K, die von $C\setminus\{b\}$ erreichbar ist und in der q markiert ist.

Satz 6.3.2 (*notwendiges und hinreichendes Kriterium für die Blockbedingung*)
Im System $\sum$ existiert für eine unverteilte Transition t der Agenten a und b genau dann eine verteilende Verfeinerung, wenn $(\bullet t)_a$ $(\triangleright\setminus(\bullet t)_a)$ $(\bullet t)_b$ oder $(\bullet t)_b$ $(\triangleright\setminus(\bullet t)_b)$ $(\bullet t)_a$ gilt.
Beweis:

($\leftarrow$):
Aus Symmetrie-Gründen zeigen wir nur für die erste Bedingung, dass die Blockbedingung gilt.
Wir konstruieren ein Verfeinerungsnetz (siehe Abb. 6.3.2), das folgende Bedingungen erfüllt: i. $t_{a1} \blacktriangleleft t_{b1}$, wobei t_{a1} und t_{b1} jeweils die Anfangstransition von a und b in N_t sind; ii. $(\bullet t)_a \blacktriangleleft (t^\bullet)_b$ und $(\bullet t)_b \blacktriangleleft (t^\bullet)_a$; iii. t_{b1} ist *fair*. Wir zeigen, diese Verfeinerung erfüllt die Blockbedingung.
Nach Voraussetzung können in jedem Ablauf die Marken auf $(\bullet t)_a$ festgehalten werden, bis $(\bullet t)_b$ markiert ist. Das heißt, für einen Schnitt C eines Ablaufs (K', r') von $\sum'$, in dem $(\bullet t)_a$ markiert ist, d.h. es gibt ein $M_a \subseteq C$ mit $r'(M_a)=(\bullet t)_a$ und eine *co*-Menge M' mit $C\setminus M_a \rightarrow^* M'$, in der $(\bullet t)_b$ markiert ist, d.h. es gibt ein $M_b \subseteq M'$ mit $r(M_b)=(\bullet t)_b$. Dann wird t_{b1} irgendwann schalten, da $(\bullet t)_b$ wegen $(\bullet t)_a(\triangleright\setminus(\bullet t)_a)(\bullet t)_b$ immer wieder markiert wird und t_{b1} *fair* ist.

($\rightarrow$):
Falls beide Bedingungen nicht gelten, ist nachzuweisen, dass dann die Blockebedingung nicht gelten kann.
Da das Verfeinerungsnetz verteilt ist, gilt im Verfeinerungsnetz höchstens $(\bullet t)_a{}^\bullet \blacktriangleleft (\bullet t)_b{}^\bullet$ oder $(\bullet t)_b{}^\bullet \blacktriangleleft (\bullet t)_a{}^\bullet$ (wegen Bem. 6.3.1). Es sind also zwei Fälle zu unterscheiden: (a) Es gilt nicht $(\bullet t)_b{}^\bullet \blacktriangleleft (\bullet t)_a{}^\bullet$. (b) Es gilt nicht $(\bullet t)_a{}^\bullet \blacktriangleleft (\bullet t)_b{}^\bullet$.
Zu (a): (Fall (b) folgt aus Symmetrie-Gründen)
Falls (a) gilt, haben alle Verfeinerungsnetze die folgende Form: $\bullet t_{Start_a} = (\bullet t)_a$, weil keine interne Stelle von (N_t, t^-) markiert ist und $\neg((\bullet t)_b{}^\bullet \blacktriangleleft (\bullet t)_a{}^\bullet)$, also nach Schalten von t_{Start_b} keine Stelle aus $\bullet t_{Start_a}$ belegt werden kann, wobei

t_{Start_a} und t_{Start_b} jeweils die Anfangstransition von *a* und *b* in N_t sind.
Falls *t* in Σ jemals aktiviert wird, gibt es Abläufe, in denen ein Schnitt *C* existiert mit: $(\bullet t)_a$ ist in *C* belegt. In jedem dieser Schnitte kann t_{Start_a} schalten. Damit entsteht jeweils ein Schnitt C_1 mit $C<C_1$, in dem ein Ablauf von (N_t, t^-) begonnen hat. $(\bullet t)_a$ ist nicht mehr markiert und es gilt $M=C\setminus(\bullet t)_a \subseteq C_1$. Solange $(\bullet t)_b$ nicht belegt wird, kann durch Transitionen von N_t keine Stelle von Σ belegt werden (wegen $(\bullet t)_a \blacktriangleleft (t\bullet)_b$ und $(\bullet t)_b \blacktriangleleft (t\bullet)_a$). Wenn die Blockbedingung gelten soll, dann wird aber immer irgendwann $(\bullet t)_a$ belegt (also durch Transition von Σ), d.h. es gilt $(\bullet t)_a (\triangleright\setminus(\bullet t)_a)(\bullet t)_b$. Widerspruch zur Voraussetzung. □

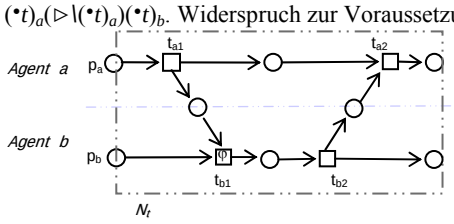


Abb. 6.3.2 Illustration für den Beweis von Satz 6.3.2

Bemerkung 6.3.3
Falls für eine unverteilte Transition t der Agenten a und b ein Verfeinerungsnetz existiert, das die Blockbedingung erfüllt, existiert immer auch ein Verfeinerungnetz, das $(\bullet t)_a \blacktriangleleft (t\bullet)_b$ *und* $(\bullet t)_b \blacktriangleleft (t\bullet)_a$ *erfüllt.*
Beweis: Durch das Schalten der beiden Starttransitionen in den Standardverfeinerungsnetzen ist gesichert, dass ein begonnener Ablauf von (N_t, t^-) vollständig ausgeführt wird (Bem. 6.1.2). Damit kann an das Verfeinerungsnetz eine zusätzliche Ausgabesynchronisation ergänzt werden, die $(\bullet t)_a \blacktriangleleft (t\bullet)_b$ und $(\bullet t)_b \blacktriangleleft (t\bullet)_a$ sichert. □

6.4 Der dritte Grundtyp: Erwartete Mithilfe

Dieser Grundtyp passt zu dem folgenden Fall: Keiner von den beiden Agenten hat vorher (vor der Transition *t*) eine Nachricht geschickt. In diesem Fall müssen beide Agenten im Ersetzungsnetz noch mehr kommunizieren, damit die Blockbedingung erfüllt ist.

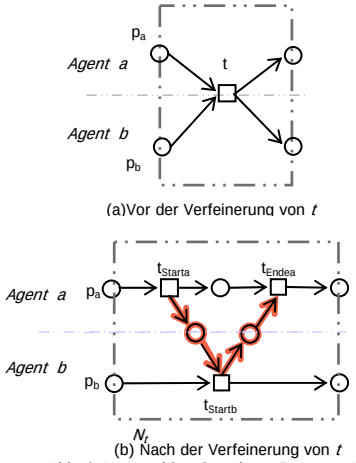


Abb. 6.4.1 Der dritte Grundtyp: *Erwartete Mithilfe*

Satz 6.4.1
Seien Σ *ein Agentensystem, t eine unverteilte Transition von Agenten a und b in* Σ*,* N_t *ein Standardverfeinerungsnetz von t. In* Σ *gelte: i.* $(\bullet t)_a \triangleright (\bullet t)_b$ *; ii. Es gibt keine konkurrente Transition zu t.* Dann erfüllt $\Sigma' = \Sigma(N_t \setminus t)$ *die Blockbedingung, wenn in* (N_t, t^-) $(\bullet t)_a \blacktriangleleft (t\bullet)_b$, $(\bullet t)_b \blacktriangleleft (t\bullet)_a$ *und* $t_{Start_a} \blacktriangleleft t_{Start_b}$ *gelten, wobei* t_{Start_a} *und* t_{Start_b} *jeweils die Anfangstransition von a und b in* N_t *sind.*

Im Satz 6.4.1 wird gefordert, dass Agenten *a* und *b* keine Konflikt-Transitionen zu *t* haben. Diese Forderung kann wie folgt abgeschwächt werden: Agent *b* hat zwar Konflikt-Transitionen zu *t*, die Transition *t* ist aber *fair*. Und Agent *a* hat zwar Konflikt-Transitionen, diese Konflikt-Transitionen sind aber nicht aktiviert. In diesem abgeschwächten Fall erfüllt das erhaltene System nach der Verfeinerung die Blockbedingung, wenn das Ersetzungsnetz zusätzlich noch die Bedingung erfüllt, dass die Anfangstransition vom Agenten *b fair* ist

Bemerkung 6.4.2
In Σ *gelte: i.* $(\bullet t)_a \triangleright (\bullet t)_b$*; ii. Es gibt keine aktivierte Konkurrente Transition zu* $(\bullet t)_a$*, falls alle Stellen von* $(\bullet t)_a$ *markiert sind; iii. t ist fair.*
So gilt in Σ*:* $(\bullet t)_a (\triangleright \setminus (\bullet t)_a) \triangleright (\bullet t)_b$.
Beweis: Folgt daraus, dass es keine aktivierte konkurrente Transition zu $(\bullet t)$ gibt. □

Aus Bem. 6.4.2 und Satz 6.4.1 folgt unmittelbar das im folgenden Satz formulierte Kriterium.
Satz 6.4.3 (*Abschwächung von Satz 6.4.1*)
Seien Σ *ein Agentensystem, t eine unverteilte Transition von Agenten a und b in* Σ*,* N_t *ein Standardverfeinerungsnetz von t. In* Σ *gelte: i.* $(\bullet t)_a \triangleright (\bullet t)_b$ *; ii. Es gibt keine aktivierte Konkurrente Transition zu* $(\bullet t)_a$*, falls alle Stellen von* $(\bullet t)_a$ *markiert sind; iii. t ist fair.* Dann erfüllt $\Sigma' = \Sigma(N_t \setminus t)$ *die Blockbedingung, wenn in* (N_t, t^-) $(\bullet t)_a \blacktriangleleft (t\bullet)_b$, $(\bullet t)_b \blacktriangleleft (t\bullet)_a$ *und* $t_{Start_a} \blacktriangleleft t_{Start_b}$ *gelten und in* (N_t, t^-) t_{Start_b} *fair ist, wobei* t_{Start_a} *und* t_{Start_b} *jeweils die Anfangstransition von a und b in* N_t *sind.*

Aus Satz 6.4.3 können wir das folgende noch allgemeinere Kriterium erhalten.

Folgerung 6.4.4 (*von Satz 6.4.3*)
Seien Σ *ein Agentensystem, t eine unverteilte Transition von Agenten a und b in* Σ*,* N_t *ein Standardverfeinerungsnetz von t. In* (N_t, t^-) *gelte:* $(\bullet t)_a \blacktriangleleft (t\bullet)_b$*,* $(\bullet t)_b \blacktriangleleft (t\bullet)_a$ *und* $t_{Start_a} \blacktriangleleft t_{Start_b}$*, wobei* t_{Start_a}*,* t_{Start_b} *jeweils die Anfangstransition von a und b in* N_t *sind.*
$\Sigma' = \Sigma(N_t \setminus t)$ erfüllt *die Blockbedingung, gdw.* (i) *In* Σ *gilt* $(\bullet t)_a$ $(\triangleright\setminus(\bullet t)_a)$ $(\bullet t)_b$*;* (ii) *Entweder* a) *es gibt an* $(\bullet t)_b$ *keine aktivierte konkurrente Transition, falls* $(\bullet t)_b$ *markiert sind; oder* b) *es gibt an* $(\bullet t)_b$ *aktivierte konkurrente Transitionen und* t_{Start_b} *ist fair.*
Beweis:
(←):
Angenommen, die Bedingungen (i) und (ii) sind erfüllt. Zu zeigen ist: $\Sigma' = \Sigma(N_t \setminus t)$ erfüllt die Blockbedingung. Der

Beweis dafür ist analog zu dem Beweis für (←) von Satz 6.3.2.

(→):

Angenommen, $\Sigma' = \Sigma(N_t \backslash t)$ erfüllt die Blockbedingung. Zu zeigen ist: (i) und (ii) gelten.

Angenommen, (i) gilt nicht. Dann gibt es einen Schnitt C in einem Ablauf (K', r') von $\Sigma' = \Sigma(N_t \backslash t)$, in dem $(\bullet t)_a$ gilt, d.h. $M_a \subseteq C$ mit $r'(M_a) = (\bullet t)_a$, und auf den ein Auftreten von t_{Start_a} folgt, aber es gibt keinen Schnitt C', der von C erreichbar ist und M_a enthält und in dem $(\bullet t)_b$ gilt. Dann ist in K' die Blockbedingung nicht erfüllt. Das ist ein Widerspruch dazu, dass $\Sigma(N_t \backslash t)$ die Blockbedingung erfüllt. Daraus folgt: (i) gilt.

Da (i) gilt, gibt es einen von C erreichbaren Schnitt C', der M_a enthält und in dem $(\bullet t)_b$ gilt. Wenn es an $(\bullet t)_b$ eine aktivierte konkurrente Transition gibt, aber t_{Start_b} nicht *fair* ist, dann ist nicht garantiert, dass t_{Start_b} irgendwann schalten wird und damit auch nicht garantiert, dass K' die Blockbedingung erfüllt. Daraus folgt: (ii) gilt. □

Zusammengefasst, beim ersten Grundtyp ist die Anforderung an die Umgebung sehr streng. Dafür ist die Anforderung an das Ersetzungsnetz sehr schwach: Es benötigt gar keine Kommunikation für die Blockbedingung. Beim zweiten Grundtyp ist die Anforderung an die Umgebung etwas schwächer. Dann benötigt aber das Ersetzungsnetz eine Kommunikation für die Blockbedingung. Beim dritten Grundtyp ist die Anforderung an die Umgebung noch schwächer. Dann benötigt das Ersetzungsnetz noch mehr Kommunikationen für die Blockbedingung.

6.5 Vertauschbarkeit von Verfeinerungsschritten

Gibt es in einem unverteilten System zwei zu verteilende Transitionen, so stellt sich die Frage, ob beide gleichzeitig verteilt werden dürfen. Wenn das der Fall ist, dann darf für beide Transitionen die Gültigkeit der Blockbedingung im unverteilten System nachgewiesen werden.

Satz 6.5.1 (*Vertauschbarkeit*)

Sei Σ_0 ein Agentensystem, seien t_1, $t_2 \in T$ unverteilte Transitionen von Agenten a und b in Σ_0, N_{t_1} und N_{t_2} jeweils Standardverfeinerungsnetze von t_1 und t_2. Es gelte entweder $\bullet t_1 \cap \bullet t_2 = \emptyset$ oder $\bullet t_1 \cap \bullet t_2 = (\bullet t_1)_a = (\bullet t_2)_a$. Für k=1,2 erfülle $\Sigma_k = \Sigma_0(N_{t_k} \backslash t_k)$ die Blockbedingung und in (N_{t_k}, t_k^-) gelte: $(\bullet t_k)_a \blacktriangleleft (t_k \bullet)_b$, $(\bullet t_k)_b \blacktriangleleft (t_k \bullet)_a$ und $t_{k_{Start_a}} \blacktriangleleft t_{k_{Start_b}}$, wobei $t_{k_{Start_a}}$, $t_{k_{Start_b}}$ jeweils die Anfangstransition von a und b in N_{t_k} sind.

Dann erfüllen $\Sigma_1(N_{t_2} \backslash t_2)$ *und* $\Sigma_2(N_{t_1} \backslash t_1)$ *die Blockbedingung.*

Beweis:

Aus Symmetrie-Gründen beweisen wir nur, dass für $\Sigma_1(N_{t_2} \backslash t_2)$ die Blockbedingung erfüllt ist.

Nach der Folgerung 4.4.4 gilt in Σ_0: (i) $(\bullet t_2)_a (\triangleright \backslash (\bullet t_2)_a)(\bullet t_2)_b$ und (ii) entweder a) an $(\bullet t_2)_b$ gibt es keine aktivierte konkurrente Transition oder b) an $(\bullet t_2)_b$ gibt es aktivierte konkurrente Transition und $t_{2_{Start_b}}$ ist *fair*.

Im Folgenden zeigen wir, in Σ_1 gelten (i) und (ii) weiter. Daraus folgt dann nach der Folgerung 4.4.4, dass auch die Blockbedingung erhalten bleibt.

Da $\Sigma_1 = \Sigma_0(N_{t_1} \backslash t_1)$ die Blockbedingung erfüllt und in (N_{t_1}, t_1^-) $(\bullet t_1)_a \blacktriangleleft (t_1 \bullet)_b$ und $(\bullet t_1)_b \blacktriangleleft (t_1 \bullet)_a$ gelten und $(\bullet t_2)_b \cap \bullet t_1 = \emptyset$ gilt, bleibt $(\bullet t_2)_a (\triangleright \backslash (\bullet t_2)_a)(\bullet t_2)_b$ im Verfeinerungsschritt $\Sigma_0 \rightarrow \Sigma_1$ erhalten (wegen Folgerung 5.3.8). Daraus folgt: (i) gilt in Σ_1 weiter.

Angenommen, in Σ_0 gilt (ii) a), d.h. $\Sigma_0 \models \Box((\bullet t_2)_b \rightarrow \neg p)$ für ein $p \in \bullet t_{2_{Start_b}}'$, wobei $t_{2_{Start_b}}'$ eine konkurrente Transition von $t_{2_{Start_b}}$ ist. Wegen Bemerkung 5.3.12 bleibt $\Box((\bullet t_2)_b \rightarrow \neg p)$ im Verfeinerungsschritt $\Sigma_0 \rightarrow \Sigma_1$ erhalten. Also, (ii) a) gilt in Σ_1 weiter.

Angenommen, in Σ_0 gilt (ii) b). Dann ist $t_{2_{Start_b}}$ *fair*.

Also, in Σ_1 gelten (i) und entweder (ii) a) oder (ii) b). □

An folgendem Beispiel ist leicht zu sehen, dass die Voraussetzung $\bullet t_1 \cap \bullet t_2 = \emptyset$ oder $\bullet t_1 \cap \bullet t_2 = (\bullet t_1)_a = (\bullet t_2)_a$ notwendig ist. Angenommen *Chef* und *Mitarbeiter* planen zwei gemeinsame Dienstreisen. Wenn die Dienstreisen einzeln begonnen werden dürfen, dann ist ferstzulegen, wer entscheidet, welche Diestreise zuerst begonnen wird. Es wird meistens der *Chef* sein.

7. Entwurf und Verifikation verteilter Algorithmen durch verteilende Verfeinerung und Anwendungsbeispiele

In diesem Abschnitt werden wir unsere Methode zum Entwurf und zur Verifikation durch verteilende Verfeinerung vorstellen. Wir werden anhand zweier nicht trivialer dennoch verständlicher und anschaulicher Anwendungsbeispiele unsere Methode demonstrieren. Als Anwendungsbeispiele für die Verifikation werden wir den *asymmetrischen verteilten Mutex* – verifizieren. Als Anwendungsbeispiel für den Entwurf werden wir den verteilten *crosstalk* Algorithmus durch verteilende Verfeinerung entwerfen.

Bei *Mutex* betreffen die gewünschten Eigenschaften die Stellen von Agenten. Daher ist es unwichtig, was die zu verteilenden Transitionen für Teilaufgaben modellieren. Die Eigenschaften des *crosstalk* beziehen sich dagegen auf die Teilaufgaben der zu verteilenden Transitionen. Daher ist es wichtig, was diese Transitionen für Teilaufgaben modellieren. Das ist der schwerere Fall. Die gewünschten Eigenschaften werden durch Reihenfolge-Relationen der Aktionen (Kausalitäten im System) formuliert.

7.1 Verifikation verteilter Algorithmen durch verteilende Verfeinerung

Wir können einen verteilten Algorithmus folgendermaßen verifizieren. Als erstes ist die algorithmische Idee des gegebenen Algorithmus zu bestimmen. Danach sucht man nach einem Modell, das die Grundidee des Algorithmus darstellt. Dieses Modell wird zu einem unverteilten Agentensystem erweitert, das jetzt schon die

algorithmische Idee veranschaulicht. Von diesem Netz ist zu beweisen, dass es die geforderten Eigenschaften des gegebenen Algorithmus erfüllt. Jetzt sind nur noch die im Netz enthaltenen unverteilten Transitionen korrekt zu verteilen. Es wird ein verteilter korrekter Algorithmus erhalten, der mit dem gegebenen Algorithmus übereinstimmt.

Diese Verifikationsmethode hat noch den Vorteil, dass neben dem Beweis des gegebenen Algorithmus auch gut nachvollziehbare Entwurfsschritte für den Algorithmus erhalten werden.

Im Folgenden demonstrieren wir, wie sich verteilte Algorithmen mit Hilfe der verteilenden Verfeinerung verifizieren lassen.

Beispiel: *Fahrradausleihe* Σ_0

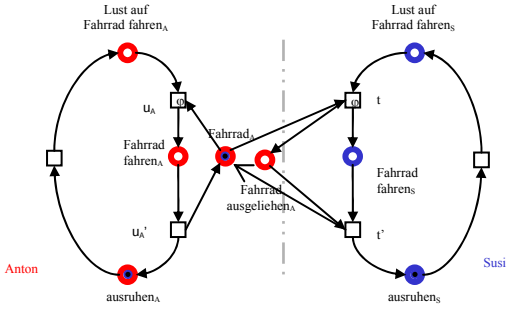


Abb. 7.1.1 *Fahrradausleihe* Σ_0

Anton und *Susi* sind Geschwister. *Anton* hat ein Fahrrad (Stelle $Fahrrad_A$). Wenn er Lust hat (Stelle *Lust-auf-Fahrrad-fahren*$_A$), kann er Fahrrad fahren (Stelle *Fahrrad-fahren*$_A$). Danach wird er sich ausruhen (Stelle *ausruhen*$_A$). *Susi* hat kein Fahrrad. Sie kann das Fahrrad aber von *Anton* ausleihen, wenn sie auch mal Fahrrad fahren möchte. Wenn *Anton* aber zufällig auch selber das Fahrrad braucht, kann er es benutzen. Aber *Anton* soll nicht immer wieder nur selbst das Fahrrad benutzten, er soll *Susi* irgendwann einmal das Fahrrad ausleihen (das nennt man *fairness*). Wenn *Susi* das Fahrrad benutzt, soll sie danach sofort das Fahrrad zurückgeben, weil das Fahrrad *Anton* gehört.

Das System hat folgende Eigenschaften:
Lebendigkeitseigenschaften:

$$Lust\text{-}auf\text{-}Fahrrad\text{-}fahren_A \triangleright Fahrrad\text{-}fahren_A \quad (L1)$$

$$Lust\text{-}auf\text{-}Fahrrad\text{-}fahren_S \triangleright Fahrrad\text{-}fahren_S \quad (L2)$$

D.h. jeder wird irgendwann das Fahrrad benutzen, wenn er Bedarf hat.
Sicherheitseigenschaft:

$$\Box\ \neg (Fahrrad\text{-}fahren_A \wedge Fahrrad\text{-}fahren_S) \quad (S)$$

D.h. die beiden werden nie gleichzeitig das Fahrrad benutzen.

Im System gibt es zwei Agenten: den Agenten *Anton* und die Agentin *Susi*. Die linken Stellen sind Stellen von *Anton*, die rechten Stellen sind Stellen von *Susi*. Die Stelle Fahrrad gehört zum Agenten *Anton*, weil das Fahrrad *Anton* gehört. Die Stelle *Fahrrad-ausgeliehen*$_A$ modelliert, ob das Fahrrad ausgeliehen ist.

Die Transition u_A ist lokal von Anton, weil das Fahrrad von *Anton* ist. Analog ist die Transition u_A' lokal. Die Transition *t* ist von *Anton* und *Susi* zusammen auszuführen, sie ist also unverteilt. Analog ist die Transition *t'* auch unverteilt. Das System ist also unverteilt. Wir beginnen mit diesem System Σ_0, verteilen die unverteilten Transitionen *t, t'* nacheinander. Schließlich erhalten wir den verteilten *Mutex*-Algorithmus Σ_*.

Zunächst geben wir einige Eigenschaften von Σ_0 an. In Σ_0 gelten:

$$\Box\ (ausruhen_S + Lust\text{-}auf\text{-}Fahrrad\text{-}fahren_S + Fahrrad\text{-}fahren_S = 1) \quad (7\text{-}1\text{-}1)$$

$$\Box\ (ausruhen_A + Lust\text{-}auf\text{-}Fahrrad\text{-}fahren_A + Fahrrad\text{-}fahren_A = 1) \quad (7\text{-}1\text{-}2)$$

$$\Box\ (Fahrrad\text{-}fahren_A + Fahrrad_A + Fahrrad\text{-}fahren_S = 1) \quad (7\text{-}1\text{-}3)$$

$$\Box\ (Fahrrad\text{-}fahren_A + Fahrrad_A + Fahrrad\text{-}ausgeliehen_A = 1) \quad (7\text{-}1\text{-}4)$$

$$\Box\ (Fahrrad\text{-}fahren_S \rightarrow Fahrrad\text{-}ausgeliehen_A) \quad (7\text{-}1\text{-}5)$$

$$Lust\text{-}auf\text{-}Fahrrad\text{-}fahren_S \triangleright Fahrrad_A \quad (7\text{-}1\text{-}6)$$

Lemma 7.1.1 Σ_0 *erfüllt (S), (L1) und (L2).*

Die Verteilung der unverteilten Transition *t*

Im ersten Schritt verteilen wir die unverteilte Transition *t* zu einem verteilten Ersetzungsnetz N_t (Abb. 7.1.2).

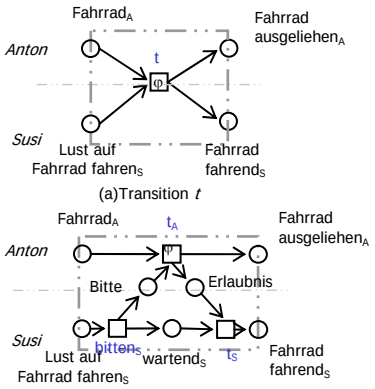


Abb. 7.1.2 Verteilung von *t*

Kausalität:
Im lokalen Teilsystem (N_t, t^-) gelten offensichtlich die folgende Kausalität $(\bullet t)_A \blacktriangleleft (t\bullet)_S$ und $(\bullet t)_S \blacktriangleleft (t\bullet)_A$, wobei die Indizes *A* und *S* jeweils Agenten *Anton* und *Susi* bezeichnen.

Blockbedingung:
Die Umgebung von *t* erfüllt die folgenden Bedingungen:

In Σ_0 gilt: $Lust\text{-}auf\text{-}Fahrrad\text{-}fahren_S \triangleright Fahrrad_A$ (Formel (7-1-6)); *t* hat an der Stelle *Lust-auf-Fahrrad-fahren*$_S$ keine Konflikt-Transition, an der Stelle $Fahrrad_A$ zwar eine Konflikt-Transition aber auch eine *Fairness*-Annahme. Das Ersetzungsnetz N_t für *t* erfüllt die folgenden Bedingungen:

In (N_t, t^-) gelten: Anfangstransition *bitten*$_S$ von *Susi* $\blacktriangleleft$ Anfangstransition t_A von *Anton*; $(\bullet t)_A \blacktriangleleft (t\bullet)_S$ und $(\bullet t)_S \blacktriangleleft (t\bullet)_A$; Anfangstransition t_A von *Anton* ist *fair*.
Wir benutzen den dritten Grundtyp (vgl. Satz 6.4.3 in Abschnitt 6.4) zum Nachweisen, dass die Blockbedingung erfüllt ist.

Lemma 7.1.2 $\Sigma_1=\Sigma_0(N_t \setminus t)$ *erfüllt die Blockbedingung.*

Erhaltenbleiben von Eigenschaften:
Wir benutzen die Sätze über das Erhaltenbleiben der Eigenschaften durch Verfeinerung mit Synchronisationsbedingungen aus Abschnitt 5.3 zum Nachweisen, dass alle gewünschten Eigenschaften erhalten bleiben.

Lemma 7.1.3
Beim Verfeinerungsschritt $(\Sigma_0 \rightarrow \Sigma_1)$ *bleiben (S), (L1) und (L2) erhalten.*

Lemma 7.1.4
Bei $(\Sigma_0 \rightarrow \Sigma_1)$ *bleiben die folgenden Eigenschaften erhalten:*
(*i*) $\Box(Fahrrad\text{-}fahren_S \rightarrow Fahrrad\text{-}ausgeliehen_A)$;
(*ii*) $\Box\neg(Lust\text{-}auf\text{-}Fahrrad\text{-}fahren_S \wedge Fahrrad\text{-}fahren_S)$;
(*iii*) $\Box\neg(Lust\text{-}auf\text{-}Fahrrad\text{-}fahren_A \wedge Fahrrad\text{-}fahren_A)$;
(*iv*) $\Box(Fahrrad\text{-}fahren_A + Fahrrad_A + Fahrrad\text{-}fahren_S \leq 1)$.
Beweis: siehe [Wu07].

Lemma 7.1.4 (*i*) werden wir zum Nachweis der Blockbedingung bei der Verteilung von *t'* benutzen. Lemma 7.1.4 (*ii*)-(*iv*) werden wir zum Nachweis des Erhaltenbleiben von Eigenschaften bei der Verteilung von *t'* benutzen.

In Abb. 7.1.3 ist das erhaltene System Σ_1 nach der Verteilung von *t* angegeben.

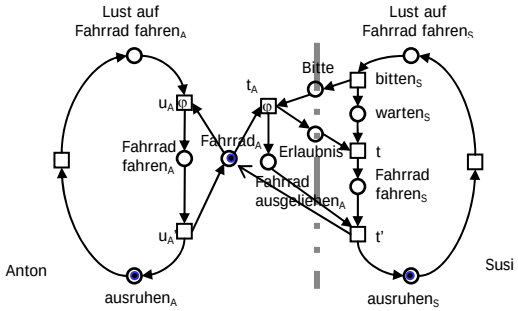
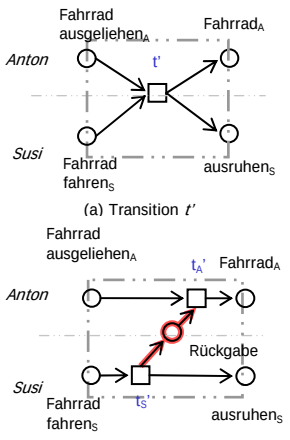


Abb. 7.1.3 System Σ_1 (nach der Verteilung von *t*)

Die Verteilung der unverteilten Transition *t´*

Wir verteilen die unverteilte Transition *t´* zu einem verteilten Ersetzungsnetz $N_{t'}$ (Abb. 7.1.4).



(b) Ersetzungsnetz $N_{t'}$ für *t'*
Abb. 7.1.4 Verteilung von *t'*

Synchronisationsbedingung:
Im lokalen Teilsystem $(N_{t'}, t'^{-})$ gilt offensichtlich die folgende Synchronisationsbedingung:

$$Fahrrad\text{-}fahren_S \bullet \blacktriangleleft \bullet Fahrrad_A$$

Blockbedingung:
Die Umgebung von *t´* erfüllt die folgenden Bedingungen: In Σ_1 gilt: $\Box(Fahrrad\text{-}fahren_S \rightarrow Fahrrad\text{-}ausgeliehen_A)$; *t´* hat an der Stelle $Fahrrad\text{-}ausgeliehen_A$ keine Konflikt-Transition. Das Ersetzungsnetz $N_{t'}$ für *t´* erfüllt die folgende Bedingung: In $(N_{t'}, t'^{-})$ gilt: Anfangstransition t_S' von *Susi* ◄ Anfangstransition t_A' von *Anton*.
Wir benutzen den zweiten Grundtyp (vgl. Satz 6.2.1 in Abschnitt 6.2) zum Nachweisen, dass die Blockbedingung erfüllt ist.

Lemma 7.1.5 $\Sigma_*=\Sigma_1(N_{t'} \setminus t')$ *erfüllt die Blockbedingung.*

Erhaltenbleiben von Eigenschaften:
Wir benutzen die Sätze über Erhaltenbleiben der Eigenschaften durch Verfeinerung mit Synchronisationsbedingung aus Abschnitt 5.3 zum Nachweisen, dass alle gewünschten Eigenschaften erhalten bleiben.

Lemma 7.1.6
Beim Verfeinerungsschritt $(\Sigma_1 \rightarrow \Sigma_*)$ *bleiben (S), (L1) und (L2) erhalten.*
Beweis: Aus Lemma 5.3.14 folgt: Bei $(\Sigma_1 \rightarrow \Sigma_*)$ bleibt (*S*) erhalten (wegen (7-1-3), Lemma 7.1.4 (*iv*)). Aus Lemma 5.3.10 folgt: Bei $(\Sigma_1 \rightarrow \Sigma_*)$ bleiben (*L1*) und (*L2*) erhalten (wegen (7-1-1), (7-1-2) und Lemma 7.1.4 (*ii*),(*iii*)). □

In Abb. 7.1.5 ist das erhaltene System Σ_* nach der Verteilung von *t´* angegeben. Σ_* ist schon unser Ziel-Algorithmus – ein verteilter *Mutex*-Algorithmus. In diesem System gelten alle gewünschten Eigenschaften.

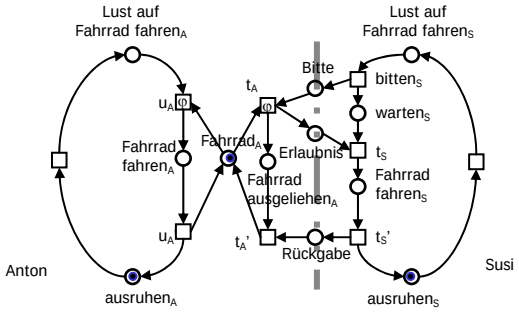


Abb. 7.1.5 *Verteilte Fahrradausleihe* Σ_*

Satz 7.1.7 Σ_* *erfüllt (S), (L1) und (L2).*
Beweis: Folgt direkt aus Lemma 7.1.1, Lemma 7.1.3 und Lemma 7.1.6. □

7.2 Entwurf verteilter Algorithmen durch verteilende Verfeinerung

Der Entwurf durch verteilende Verfeinerung umfasst drei Schritte:
1. Die Grundidee des Algorithmus wird durch ein einfaches System, das Grundmodell, dargestellt. Die geforderten Eigenschaften des Systems werden formuliert.

2. Das Grundmodell wird um eine algorithmische Idee zu einem Algorithmus erweitert und zwar so, dass ein unverteiltes Agentensystem (Anfangsalgorithmus) entsteht. Die Gültigkeit der geforderten Eigenschaften des Systems wird bewiesen.
3. Die unverteilten Transitionen werden so verfeinert, dass die gewünschten Eigenschaften erhalten bleiben.

Zur Illustration des Entwurfs wird in diesem Abschnitt der *Crosstalk*-Algorithmus durch verteilende Verfeinerung erzeugt. Hierbei werden drei unverteilte Transitionen verteilt. Die Korrektheit des Entwurfs wird bewiesen.

1. Grundidee des *crosstalk*-Algorithmus

Wir betrachten zunächst das folgende System (vgl. Abb. 7.2.1): Zwei Partner – der linke Agent *l* und der rechte Agent *r* – werden wiederholend gemeinsame Aktionen zur Kommunikation ausführen. Sie haben drei unterschiedliche Aufgaben: $Aufgabe_1$, $Aufgabe_2$ und $Aufgabe_3$.

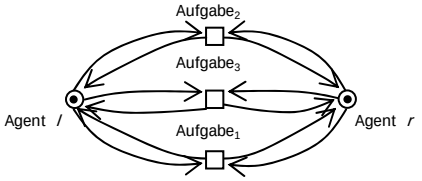


Abb. 7.2.1 *Crosstalk*

Jede der drei Aufgaben wird vollständig bearbeitet, bevor die nächste Aufgabe begonnen wird. Bei dem *crosstalk*-Algorithmus sind die drei Aufgaben:
$Aufgabe_1$ besteht aus folgenden Aktionen: der Agent *r* sendet eine Nachricht (Aktion $senden_r$); der Agent *l* antwortet auf diese Nachricht (Aktion $antworten_l$); der Agent *r* empfängt die Antwort (Aktion $empfangen_r$).
$Aufgabe_2$ besteht aus folgenden Aktionen: der Agent *l* sendet eine Nachricht (Aktion $senden_l$); der Agent *r* antwortet auf diese Nachricht (Aktion $antworten_r$); der Agent *l* empfängt die Antwort (Aktion $empfangen_l$).
$Aufgabe_3$ besteht aus folgenden Aktionen: beide Agenten senden jeweils eine Nachricht (Aktionen $senden_l$, $senden_r$); beide Agenten antworten jeweils auf die Nachricht (Aktionen $antworten_l$, $antworten_r$); beide Agenten empfangen jeweils die Antworten (Aktionen $empfangen_l$, $empfangen_r$).

Dieses Grundmodell des *crosstalk*-Algorithmus beschreibt die einzelnen Aktionen der Agenten verbal. Um die geforderten Eigenschaften des Algorithmus formalisieren zu können, müssen wir zunächst definieren, wie wir die Aufgaben der einzelnen Agenten formal darstellen.

2. Spezifikation von Aufgaben eines unverteilten Systems mit Agenten und Kanälen

In einem unverteilten System modelliert eine unverteilte Transition mehrere Aufgaben unterschiedlicher Agenten. Zum Beispiel enthält die unverteilte Transition *Verträge-behandeln-und-unterschreiben* im Einführungsbeispiel *Dienstreise* (siehe Abschnitt 1) zwei Aufgaben der Agenten *Mitarbeiter* und *Chef*. Um alle Teilaufgaben einer Transition beschreiben zu können, haben wir entweder lange Namen verwendet oder im Text genauer die Aufgaben einer unverteilten Transition beschrieben.

Aufgabenspezifikation

Eine Aufgabenspezifikation soll z.B. $senden_l$ **;** $empfangen_r$ sein. Ein Agent *l* hat die Aufgabe *´senden´*. Danach soll ein Agent *r* die Aufgabe *´*$empfangen_r$*´* haben. Aufgaben sind also Bezeichner. Agenten sind die Agenten des Agentensystems. Die Reihenfolge der Aufgaben wird durch Symbole festgelegt. Die Aufgabenspezifikation $senden_l$ **,** $empfangen_r$ besagt, die Agenten *l* und *r* senden beide in beliebiger Reihenfolge.

Aufgabenbezeichnung: Bezeichner
Agent: Bezeichner
Elementare Aufgabe: $\text{Aufgabenbezeichnung}_{\text{Agent}}$
Term: elementare Aufgabe **|** {Aufgabenspezifikation }
Sequenz: Term **|** Sequenz **;** Term
Aufgabenspezifikation: Sequenz **|** Aufgabenspezifikation **,** Sequenz

Eine etwas umfassendere Aufgabenspezifikation ist:
{ $senden_l$ **,** $senden_r$} **;** { $antworten_l$ **,** $antworten_r$} **;** {$empfangen_l$ **,** $empfangen_r$}

Elementare Aufgaben **(**Aufgabenspezifikation **)** $=_{def}$ Menge der in der Aufgabenspezifikation auftretenden elementaren Aufgaben.

Semantik einer Aufgabenspezifikation bzw. einer Transition

Wir verwenden Aufgabenspezifikationen als Namen von Transitionen. Damit ordnen wir den Transitionen eine Semantik zu. Das Ziel der Aufgabenspezifikation ist es, die elementaren Aufgaben der Agenten festzulegen und die Reihenfolge anzugeben, in der diese Aufgaben erfüllt werden sollen. Falls eine Verfeinerung diese Reihenfolgebeziehungen beachtet, werden diese Reihenfolgebeziehungen der ◄ Relation entsprechen. Deshalb werden wir die ◄ Relation zur Bezeichnung der Reihenfolgebeziehungen verwenden.

Für *a, b*∈ elementare Aufgaben **(***Aufgabenspezifikation* **)** gilt *a*◄*b* gdw. es existiert ein Teilausdruck der *Aufgabenspezifikation* der Form *Sequenz* **;** *Term* mit *a* ∈ elementare Aufgaben **(***Sequenz***)**, *b* ∈ elementare Aufgaben **(***Term***).** Gilt *t´*◄ *t* und *a*∈ elementare Aufgaben (*t*), so wird definiert *t´*◄*a*. Gilt *t*◄ *t´* und *a*∈elementare Aufgaben (*t*), so wird definiert *a*◄ *t´*.

Beispiel für ein einfaches unverteiltes Netz:

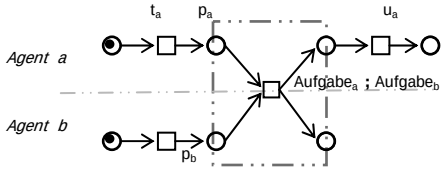


Abb. 7.2.2 Ein System ∑ mit einer unverteilten Transition $Aufgabe_a$ **;** $Aufgabe_b$

Es gilt z.B.: t_a◄$Aufgabe_a$, $Aufgabe_a$◄$Aufgabe_b$,t_a◄$Aufgabe_b$, $Aufgabe_b$◄u_a.

Verfeinerung konform zur Aufgabenspezifikation

Damit die Eigenschaften eines unverteilten Systems erhalten bleiben, muss ein Verfeinerungsnetz N_t einer unverteilten Transition t die durch den Namen t gegebene Aufgabenspezifikation beachten.

Im Beispiel der Transition $Aufgabe_a$ **;** $Aufgabe_b$ geschieht das z.B. durch folgendes Verfeinerungsnetz:

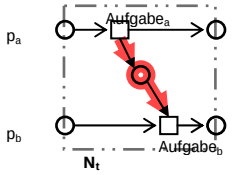


Abb. 7.2.3 Ersetzungsnetz für die Transition $Aufgabe_a$ **;** $Aufgabe_b$ in Abb. 7.2.2

Die Relation $Aufgabe_a \blacktriangleleft Aufgabe_b$ wird durch dieses Verfeinerungsnetz erhalten. Außerdem haben wir eine eineindeutige Abbildung der elementaren Aufgaben der Spezifikation (also $Aufgabe_a$, $Aufgabe_b$) in die Menge der Transitionen des Verfeinerungsnetzes, die in unserem Beispiel die Identität ist. Leider kann nicht immer die Identität als Abbildung benutzt werden. Das kommt daher, dass zwei verschiedene Aufgabenspezifikationen die gleiche elementare Aufgabe enthalten dürfen. Die Verfeinerungsnetze zweier Transitionen dürfen aber nicht den gleichen Namen für eine Transition verwenden. In unseren Beispielen werden wir dieses Problem umgehen, indem wir den Namen elementarer Aufgaben mit keinem, einem oder mehreren Strichen versehen, wenn wir sie als Namen von Transitionen in Verfeinerungsnetzen verwenden.

Definition 7.2.1 (*Verfeinerungsnetz konform zu einer durch den Namen der Transition gegebenen Aufgabenspezifikation*)
Sei t der Name einer Transition, wobei t syntaktisch eine Aufgabenspezifikation ist. Ein Verfeinerungsnetz N_t ist konform *zur Aufagbenspezifikaiton t, wenn es eine eineindeutige Abbildung c elementare Aufgaben(t) in Transitionen(N_t) gibt mit : sind a,b∈elementare Aufgaben (t) und $a \blacktriangleleft b$, so gilt im Verfeinerungsnetz $c(a) \blacktriangleleft c(b)$.*

Durch die Festlegung, dass eine Aufgabenspezifikation durch den Namen der Transitionen gegeben wird, vermeiden wir eine Erweiterung der Netzdefinition.

Durch das Verwenden der Namen elementarer Aufgaben als Namen der Transitionen in Verfeinerungsnetzen (gegebenfalls ergänzt um ein oder mehrere Striche) vermeiden wir die explizite Angabe der Abbildung zwischen den elementaren Aufgaben und den Transitionen.

Bisher können wir nur Aussagen darüber machen, welche temporal-logischen Eigenschaften eines Systems Σ im verfeinerten System $\Sigma' = \Sigma(N_t \setminus t)$ erhalten bleiben (siehe Abschnitt 5). Jetzt ist es auch möglich festzustellen, welche Aussagen über elementare Aufgaben einer Transition t erhalten bleiben.

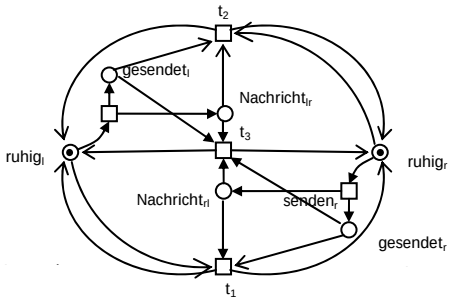
Satz 7.2.2
Seien Σ ein Agentensystem, t eine unverteilte Transition von Agenten a und b in Σ. Sei N_t ein Standardverfeinerungsnetz von t und konform zur Aufgabenspezifikation t. Es gelte für $\Sigma' = \Sigma(N_t \setminus t)$ die Blockbedingung. In (N_t, t^-) gelten $(\bullet t)_a \blacktriangleleft (t\bullet)_b$ und $(\bullet t)_b \blacktriangleleft (t\bullet)_a$. Sei e eine elementare Teilaufgabe von t, sei c(e) der Transitionsname von e in N_t. Weiter sei $t' \in T$ mit $t' \neq t$.

i. *Wenn $\Sigma \models t' \blacktriangleleft e$ gilt und es ein $p \in \bullet t$ mit $\Sigma \models t' \blacktriangleleft p$ und $(N_t, t^-) \models p \blacktriangleleft c(e)$ gibt, dann gilt $\Sigma' \models t' \blacktriangleleft e$.*
ii. *Sei e´ eine weitere Aufgabe von t, d.h. e´ ∈*elementare Aufgaben(*t*). *Dann bleibt $e \blacktriangleleft e'$ beim Verfeinerungsschritt ($\Sigma \rightarrow \Sigma'$) erhalten.*
iii. *Sei e´ eine Aufgabe, $T_{e'} = \{ t_{e'} \in T \mid e' \in$* elementare Aufgaben($t_{e'}$)*$\}$ mit $T_{e'} \neq \emptyset$ und $t' \notin T_{e'}$. Dann bleiben $t' \blacktriangleleft e'$ und $e' \blacktriangleleft t'$ beim Verfeinerungsschritt ($\Sigma \rightarrow \Sigma'$) erhalten.*
iv. *Seien e_1 und e_2 Aufgaben, $T_{e_1} = \{ t_{e_1} \in T \mid e_1 \in$* elementare Aufgaben($t_{e_1}$)*$\}$, $T_{e_2} = \{t_{e_2} \in T \mid e_2 \in$* elementare Aufgaben($t_{e_2}$)*$\}$ mit: T_{e_1}, T_{e_1} sind nicht leer und enthalten t nicht. Dann bleibt $e_1 \blacktriangleleft e_2$ beim Verfeinerungsschritt ($\Sigma \rightarrow \Sigma'$) erhalten.*

Beweisidee: $(\bullet t)_a \blacktriangleleft (t\bullet)_b$ und $(\bullet t)_b \blacktriangleleft (t\bullet)_a$ sichern, dass $<$ in allen Abläufen erhalten bleibt. Bei der konformen Verfeinerung von *t* gelten zusätzlich in jedem Ablauf für jede $t \rightarrow N_t$ Ersetzung die zwischen den Teilaufgaben geforderten Relationen. (ein kompleter Beweis siehe [wu07]).

3. Anfangsalgorithmus

In Abb. 7.2.4 ist ein etwas konkreteres Modell Σ_0 unter folgender Entwurfsentscheidung angegeben: Es wird für beide Agenten jeweils nur eine Sendeoperation modelliert. Ein Agent kann nur im *ruhig*-Zustand eine Nachricht senden, und danach steht er im Zustand *gesendet*. Dann modelliert die Transition t_1 in Σ_0 $antworten_l$ **;** $empfangen_r$, und die Transition t_2 modelliert $antworten_r$ **;** $empfangen_l$, die Transition t_3 modelliert
$\{antworten_l, antworten_r\}$**;** $\{empfangen_l, empfangen_r\}$.



Abkürzungen für die Namen der Transitionen:
t_1=$antworten_l$; $empfangen_r$
t_2=$antworten_r$; $empfangen_l$
t_3={$antworten_l$, $antworten_r$}; {$empfangen_l$, $empfangen_r$}

Abb. 7.2.4 System Σ_0

Seien $l=\{ruhig_l, gesendet_l, Nachricht_{rl}\}$, $r=\{ruhig_r, gesendet_r, Nachricht_{lr}\}$. Weiter sei die Agentenmenge $A=\{l, r\}$.

In Σ_0 gilt für den linken Agenten *l* die folgende Invariante:

$$ruhig_l + gesendet_l = 1 \qquad (7\text{-}2\text{-}1)$$

(d.h. der Agent *l* steht entweder im Zustand $ruhig_l$ oder im Zustand $gesendet_l$).

Analog gilt für den rechten Agenten *r*:

$$ruhig_r + gesendet_r = 1. \quad (7\text{-}2\text{-}2)$$

Außerdem sind die Kanalstellen $Nachricht_{rl}$ und $Nachricht_{lr}$ 1-beschränkt durch folgende Invarianten:

$$ruhig_r + Nachricht_{rl} = 1 \quad (7\text{-}2\text{-}3)$$

(d.h. entweder steht *r* im Zustand $ruhig_r$ oder es gibt eine Nachricht für *l* $Nachricht_{rl}$) und

$$ruhig_l + Nachricht_{lr} = 1. \quad (7\text{-}2\text{-}4)$$

Daraus folgt: Das System $\sum_0$ ist ein Agentensystem.

Wir brauchen noch folgende Formeln:

$$Nachricht_{lr} \leftrightarrow gesendet_l \quad (7\text{-}2\text{-}5)$$

(wegen Invarianten (7-2-1) und (7-2-4))

$$Nachricht_{rl} \leftrightarrow gesendet_r \quad (7\text{-}2\text{-}6)$$

(wegen Invarianten (7-2-2) und (7-2-3)).

Das System $\sum_0$ enthält drei unverteilte Transitionen t_1, t_2 und t_3, ist also ein unverteiltes System. Wir werden durch verteilende Verfeinerung dieses System zu einem verteilten System verfeinern. Das erhaltene System ist gerade der verteilte *crosstalk*-Algorithmus (System $\sum_*$).

Bei *crosstalk* ist die folgende Eigenschaft gewünscht: Ein Agent, z.B. der Agent *l*, soll die Kommunikation nicht beenden, bevor *r* die gesendete Nachricht empfangen hat. Dann soll gelten:
Vor einer Antwortoperation eines Agenten liegt eine Sendeoperation des anderen Agenten, d.h.

$$senden_l \blacktriangleleft antworten_r; \quad (K1)$$

$$senden_r \blacktriangleleft antworten_l. \quad (K2)$$

Vor einer Empfangsoperation eines Agenten liegt eine Antwortoperation des anderen Agenten, d.h.

$$antworten_r \blacktriangleleft empfangen_l; \quad (K3)$$

$$antworten_l \blacktriangleleft empfangen_r. \quad (K4)$$

Damit gilt: Zwischen zwei Sendeoperationen eines Agenten gibt es immer eine Antwortaktion des anderen Agenten und zwischen zwei Antwortoperationen eine Sendeoperation.

Das folgende Lemma besagt, dass der Anfangsalgorithmus $\sum_0$ die gewünschten Eigenschaften (*K1*)-(*K4*) erfüllt.

Lemma 7.2.3 *$\sum_0$ erfüllt (K1), (K2), (K3) und (K4).*
Beweis: Zu (*K1*): In $\sum_0$ gilt: $senden_l \blacktriangleleft t_2$ und $senden_l \blacktriangleleft t_3$. Nur *elementare Aufgaben* (t_2) und *elementare Aufgaben* (t_3) enthalten Aufgabe $antworten_r$. Daraus folgt: $\sum_0$ erfüllt (*K1*). Beweis für (*K2*)-(*K4*) siehe [Wu07].

Im Folgenden werden wir nacheinander die drei Transitionen t_1, t_2 und t_3 verteilen. Da bei unseren verteilenden Verfeinerungen hier die Kausalordnungen vollständig erhalten bleiben, wissen wir nach dem Satz *Reihenfolge von Verfeinerungsschritten* im Abschnitt 6, dass die Reihenfolge, in der die Transitionen verfeinert werden, den Entwurf der Verfeinerungsnetze nicht beeinflußt, d.h. wir erhalten immer dieselben Verfeinerungsnetze. Wir können deshalb den Entwurf wie folgt durchführen: Zunächst werden wir jede einzelne Transition t_1, t_2 und t_3 in $\sum_0$ verfeinern und die Korrektheit der Verfeinerungen nachweisen. Dann werden wir alle drei Transitionen gleichzeitig in $\sum_0$ jeweils durch die entsprechenden Verfeinerungsnetze ersetzen und ein korrektes verteiltes System $\sum_*$´ erhalten.

4. Die Verteilung der unverteilten Transition t_1

Wir verfeinern jetzt die Transition t_1, also $antworten_l$ **;** $empfangen_r$. Wenn *r* schon eine Nachricht $Nachricht_{rl}$ gesendet hat, und im Zustand $gesendet_r$ steht, und *l* im Zustand $ruhig_l$ steht, dann werden die beiden Agenten diese Transition t_1 gemeinsam ausführen. Danach werden sie in den Zustand $ruhig_l$ bzw. $ruhig_r$ zurück gehen. In Abb. 7.2.5 wird t_1 isoliert dargestellt.

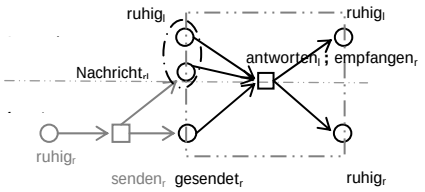


Abb. 7.2.5 Die zu verfeinernde Transition t_1

Wir werden den Entwurf der Verteilung von t_1 wie folgt durchführen: Zunächst werden wir die Umgebung der zu verfeinernden Transition analysieren und dann die Sätze aus dem Abschnitt 6 benutzen, um die Kausalitäten zu finden, die das Ersetzungsnetz erfüllen soll, damit die Blockbedingung erfüllt werden kann. Dann konstruieren wir ein Ersetzungsnetz, das diese Kausalitäten erfüllt.

In $\sum_0$ gilt (wegen Formel (7-2-6)): $ruhig_l \wedge Nachricht_{rl} \rightarrow gesendet_r$. Der Agent *r* hat an der Stelle $gesendet_r$ eine alternative Transition t_3. Aus $ruhig_l$ folgt $\neg\ gesendet_l$ (wegen Invariante (7-2-1)). Und $gesendet_l \in {}^\bullet t_3$. Daraus folgt: t_3 ist nicht aktiviert.

Nun können wir den Satz 6.2.2 (Abschwächung von Satz 6.2.1, Grundtyp 2) benutzen. Laut diesem Satz garantiert ein Standardverfeinerungsnetz die Blockbedingung, wenn es die folgenden Kausalitäten erfüllt:
- Anfangs-Transition von Agent *l* ◂ Anfangs-Transition von Agent *r*.
- $({}^\bullet t_1)_l \blacktriangleleft (t_1{}^\bullet)_r$ und $({}^\bullet t_1)_r \blacktriangleleft (t_1{}^\bullet)_l$, d.h. ein Agent kann erst im Nachbereich von t_1 Token legen, wenn sein Partner schon seine Anfangs-Transition ausgeführt hat.

Das in Abb. 7.2.6 angegebene Standardverfeinerungsnetz N_{t_1} erfüllt offensichlich die oben genannten Bedingungen.

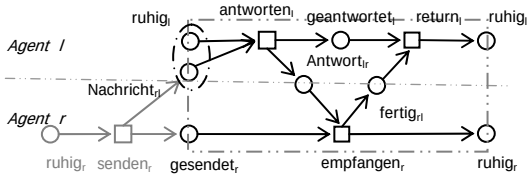


Abb. 7.2.6 Verfeinerungsnetz N_{t_1} von t_1

Satz 6.2.2 sichert die Gültigkeit der Blockbedingung. Zusammen mit $({}^\bullet t_1)_l \blacktriangleleft (t_1{}^\bullet)_r$ und $({}^\bullet t_1)_r \blacktriangleleft (t_1{}^\bullet)_l$ erhalten wir das folgende Lemma.

Lemma 7.2.4
Verfeinerungsschritt $(\sum_0 \rightarrow \sum_1)$ *mit* $\sum_1 = \sum_0(N_{t_1} \setminus t_1)$ *erfüllt die Blockbedingung und in* (N_{t_1}, t_1^-) *gelten* $({}^\bullet t_1)_l \blacktriangleleft (t_1{}^\bullet)_r$ *und* $({}^\bullet t_1)_r \blacktriangleleft (t_1{}^\bullet)_l$.

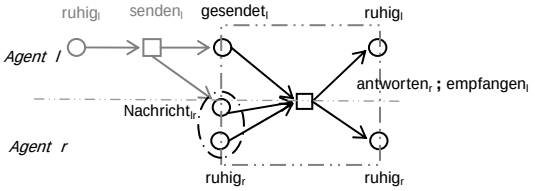
Offensichtlich gilt die folgende Aussage.

Lemma 7.2.5

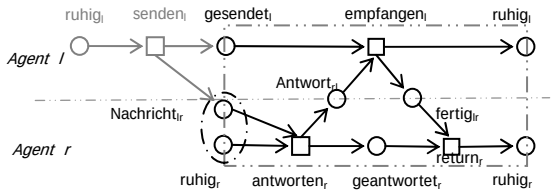
N_{t_1} ist konform zur Aufagbenspezifikaiton t_1 und in (N_{t_1}, t_1^-) gilt $Nachricht_{rl} \blacktriangleleft antworten_l$.

5. Die Verteilung der unverteilten Transition t_2

Analog zu der Verteilung von t_1 verfeinern wir jetzt die Transition t_2, also $antworten_r$ **;** $empfangen_l$. Wenn l schon an seinen Partner r eine Nachricht $Nachricht_{lr}$ gesendet hat, und im Zustand $gesendet_l$ steht, und r im Zustand $ruhig_r$ steht, dann werden die beiden Agenten diese Transition t_2 gemeinsam ausführen. Danach werden sie in den Zustand $ruhig_l$ bzw. $ruhig_r$ zurück gehen.



(a) Die zu verfeinernde Transition t_2



(b) Verfeinerungsnetz N_{t_2} von t_2

Abb. 7.2.7 Verteilung von t_2

Wir werden wieder zunächst die Umgebung der zu verfeinernden Transition analysieren, und dann die Sätze aus dem Abschnitt 6 verwenden, um ein geeignetes korrektes Ersetzungsnetz zu finden.

In Σ_0 gilt: $ruhig_r \wedge Nachricht_{lr} \rightarrow gesendet_l$ (wegen $Nachricht_{lr} \rightarrow gesendet_l$). Die alternative Transition t_3 an der Stelle $gesendet_l$ ist nicht aktiviert, weil $gesendet_r \in {}^\bullet t_3$ aber $gesendet_r$ gilt nicht (da $ruhig_r$ gilt). Nach dem Satz 6.2.2 (Abschwächung von Satz 6.2.1) garantiert das in Abb. 7.2.7(b) angegebene Standardverfeinerungsnetz N_{t_2} die Blockbedingung.

Damit erhalten wir die folgenden zwei Lemmata.

Lemma 7.2.6

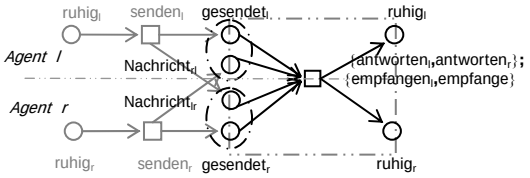
Verfeinerungsschritt $(\Sigma_0 \rightarrow \Sigma_2)$ mit $\Sigma_2 = \Sigma_0(N_{t_2} \setminus t_2)$ erfüllt die Blockbedingung und in (N_{t_2}, t_2^-) gelten $({}^\bullet t_2)_l \blacktriangleleft (t_2{}^\bullet)_r$ und $({}^\bullet t_2)_r \blacktriangleleft (t_2{}^\bullet)_l$.

Lemma 7.2.7

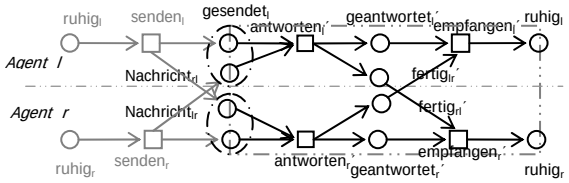
N_{t_2} ist konform zur Aufagbenspezifikaiton t_2 und in (N_{t_2}, t_2^-) gilt $Nachricht_{lr} \blacktriangleleft antworten_r$.

6. Die Verteilung der unverteilten Transition t_3

Im Folgenden verfeinern wir die Transition t_3, also $\{antworten_l$ **,** $antworten_r\}$**;** $\{empfangen_l$ **,** $empfangen_r\}$. Wenn der Agent l schon an seinen Partner eine Nachricht $Nachricht_{lr}$ gesendet hat, und im Zustand $gesendet_l$ steht, und auch der Agent r schon an seinen Partner eine Nachricht $Nachricht_{rl}$ gesendet hat, und im Zustand $gesendet_r$ steht, dann werden die beiden Agenten diese Transition t_3 gemeinsam ausführen. Danach werden sie in den Zustand $ruhig_l$ bzw. $ruhig_r$ zurück gehen.



(a) Die zu verfeinernde Transition t_3



(b) Verfeinerungsnetz N_{t_3} von t_3

Abb. 7.2.8 Verteilung von t_3

Wieder analysieren wir zunächst die Umgebung der zu verfeinernden Transition, und dann verwenden wir die Sätze aus dem Abschnitt 6, um ein korrektes Ersetzungsnetz zu entwerfen.

In Σ_0 gilt: $gesendet_l \wedge Nachricht_{rl} \rightarrow gesendet_r \wedge Nachricht_r$ (wegen $gesendet_l \rightarrow Nachricht_{lr}$ und $Nachricht_{rl} \rightarrow gesendet_r$). Damit gilt auch $gesendet_l \wedge Nachricht_{rl} \triangleright gesendet_r \wedge Nachricht_r$. Umgekehrt gilt auch $gesendet_r \wedge Nachricht_{lr} \triangleright gesendet_l \wedge Nachricht_{rl}$. Die alternative Transition t_2 von Agent l an der Stelle $gesendet_l$ ist nicht aktiviert, weil $ruhig_r \in {}^\bullet t_2$ aber $ruhig_r$ gilt nicht (da $gesendet_r$ gilt). Die andere alternative Transition t_1 von Agent l an der Stelle $Nachricht_{rl}$ ist auch nicht aktiviert, weil $ruhig_l \in {}^\bullet t_1$ aber $ruhig_l$ gilt nicht (da $gesendet_l$ gilt). Analog sind die beiden alternativen Transitionen t_1 und t_2 von Agent r jeweils an der Stelle $gesendet_r$ und der Stelle $Nachricht_{lr}$ nicht aktiviert. Nach dem Satz 6.1.3 (Abschwächung von Satz 6.1.1, Grungtyp 1) garantiert das in Abb. 7.2.8(b) angegebene Standardverfeinerungsnetz N_{t_3} die Blockbedingung.

Damit erhalten wir folgende zwei Lemmata.

Lemma 7.2.8

Verfeinerungsschritt $(\Sigma_0 \rightarrow \Sigma_3)$ mit $\Sigma_3 = \Sigma_0(N_{t_3} \setminus t_3)$ erfüllt die Blockbedingung und in (N_{t_3}, t_3^-) gelten $({}^\bullet t_3)_l \blacktriangleleft (t_3{}^\bullet)_r$ und $({}^\bullet t_3)_r \blacktriangleleft (t_3{}^\bullet)_l$.

Lemma 7.2.9

N_{t_3} ist konform zur Aufagbenspezifikaiton t_3 und in (N_{t_3}, t_3^-) gelten $Nachricht_{lr} \blacktriangleleft antworten_r´$ und $Nachricht_{rl} \blacktriangleleft antworten_l´$.

Das folgende Lemma besagt, dass beim Verfeinerungsschritt ($\Sigma_0 \rightarrow \Sigma_3$) die Voraussetzungen für die Umgebung für den Nachweis der Blockbedingung bei verteilender Verfeinerung von t_1 und von t_2 erhalten bleiben.

Lemma 7.2.10
i. *Beim Verfeinerungsschritt* ($\Sigma_0 \rightarrow \Sigma_3$) *bleiben folgende Eigenschaften erhalten:*

- $ruhig_l \wedge Nachricht_{rl} \triangleright gesendet_r$;

- $ruhig_r \wedge Nachricht_{lr} \triangleright gesendet_l$.

ii. *In* Σ_3 *gibt es*
- *keine neue aktivierte Transition zu* t_1 *und*
- *auch keine neue aktivierte Transition zu* t_2.

Beweis: siehe [Wu07].

7. Der verteilte Zielalgorithmus

Wir ersetzen jetzt alle drei Transitionen t_1, t_2 und t_3 in Σ_0 jeweils durch ihre Verfeinerungsnetze N_{t_1}, N_{t_2} und N_{t_3}. Das erhaltene System ist Σ_*' (Abb. 7.2.9).

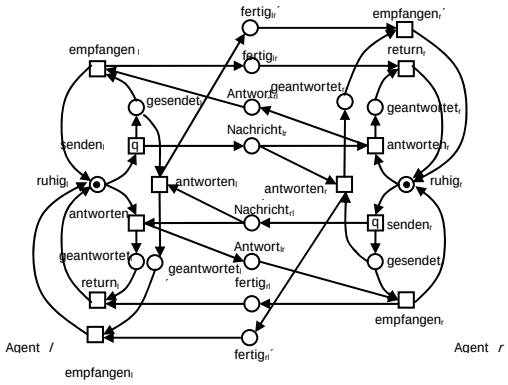


Abb. 7.2.9 System Σ_*'

Lemma 7.2.11
Sei $\sigma = \Sigma_0 \rightarrow \Sigma_3 \rightarrow \Sigma_1' \rightarrow \Sigma_2'$ *mit* $\Sigma_3 = \Sigma_0(N_{t_3} \backslash t_3)$, $\Sigma_1' = \Sigma_3(N_{t_1} \backslash t_1)$, $\Sigma_2' = \Sigma_1'(N_{t_2} \backslash t_2)$. *Jeder Verfeinerungsschritt von* σ *erfüllt die Blockbedingung.*
Beweis: Wegen Lemma 7.2.10 nach Satz 6.2.2 erfüllt $\Sigma_3(N_{t_1} \backslash t_1)$ und $\Sigma_3(N_{t_2} \backslash t_2)$ die Blockbedingung. Nach Satz 6.5.1 erfüllt jeder Verfeinerungsschritt von $\Sigma_3 \rightarrow \Sigma_1' \rightarrow \Sigma_2'$ die Blockbedingung. q.e.d.

Satz 7.2.12 Σ_*' *erfüllt (K1), (K2), (K3) und (K4).*
Beweis: Wir zeigen hier, Σ_*' erfüllt (*K1*). Der Beweis für (*K2*) - (*K4*) ist analog (siehe [Wu07]).
Wegen Lemma 7.2.11 gilt:
Jeder Verfeinerungsschritt von $\Sigma_0 \rightarrow \Sigma_3 \rightarrow \Sigma_1' \rightarrow \Sigma_2'$ mit $\Sigma_3 = \Sigma_0(N_{t_3} \backslash t_3)$, $\Sigma_1' = \Sigma_3(N_{t_1} \backslash t_1)$, $\Sigma_2' = \Sigma_1'(N_{t_2} \backslash t_2)$ und $\Sigma_2' = \Sigma_*'$ ist Halbordnung vollständig erhaltend. (1)
Wegen Lemma 7.2.3 gilt (*K1*) in Σ_0. Wir brauchen nur zu zeigen, dass (*K1*) bei jedem Verfeinerungsschritt von $\Sigma_0 \rightarrow \Sigma_3 \rightarrow \Sigma_1' \rightarrow \Sigma_2'$ erhalten bleibt.
Zu ($\Sigma_0 \rightarrow \Sigma_3$) :
In Σ_0 gilt: $senden_l \blacktriangleleft Nachricht_{lr}$. (2)
In (N_{t_3}, t_3^-) gilt $Nachricht_{lr} \blacktriangleleft antworten_r'$(wegen Lemma 7.2.9). Daraus folgt (wegen Satz 7.2.2(*i*)): (*K1*) bleibt erhalten.
Zu ($\Sigma_3 \rightarrow \Sigma_1'$): Es gilt: $antworten_r \notin$ elementare Aufgaben(t_1).Wegen (1) gilt nach Satz 7.2.2(*iii*): (*K1*) bleibt erhalten.
Zu ($\Sigma_1' \rightarrow \Sigma_2'$): Wegen (1) gilt nach Folgerung 5.3.2: Bei jedem Verfeinerungsschritt von $\Sigma_0 \rightarrow \Sigma_3 \rightarrow \Sigma_1'$ bleibt $senden_l \blacktriangleleft Nachricht_{lr}$ erhalten. Aus (2) folgt: In Σ_1' gilt $senden_l \blacktriangleleft Nachricht_{lr}$. In (N_{t_2}, t_2^-) gilt
$Nachricht_{lr} \blacktriangleleft antworten_r$(wegen Lemma 7.2.7). Daraus folgt (wegen Satz 7.2.2(*i*)): (*K1*) bleibt erhalten.
Daraus folgt: In Σ_*' gilt (*K1*). □

Wir machen von diesem System Σ_*' aus eine äquivalente System-Transformation und erreichen dann schon den Zielalgorithmus Σ_* - *crosstalk*-Algorithmus(Abb. 7.2.10). Z. B. haben $empfangen_r'$ und $return_r$ die gleiche Wirkung. Bei ihrer Identifikation fallen die Stellen $fertig_{lr}'$ und $fertig_{lr}$ und auch die Stellen $geantwortet_r$ und $geantwortet_r'$ zusammen. Es gibt für derartige Vereinfachungen, bei denen zwei Aktionen mit der gleichen Wirkung identifiziert werden, gut bekannte Algorithmen.

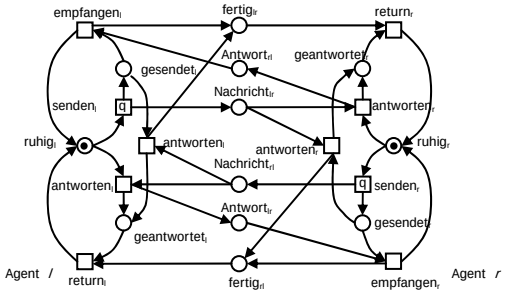


Abb. 7.2.10 Zielalgorithmus Σ_* – verteilter *crosstalk*-Algorithmus

Der entworfene Algorithmus erfüllt die geforderten Eigenschaften. Es ist verteilt, lebendig und es gelten die Sicherheitseigenschaften: Zwischen zwei Sendeoperationen eines Agenten gibt es immer eine Antwortaktion des anderen Agenten und zwischen zwei Antwortoperationen eine Sendeoperation.

8. Zusammenfassung und Ausblick

Zum Schluss fassen wir kurz zusammen, was wir erreicht haben, und machen einen Ausblick darauf, was wir weiter untersuchen können.

Wir haben den Verfeinerungsbegriff Verteilende Verfeinerung eingeführt und entsprechend passende Begriffe Agentensystem, verteilt, unverteilt, und Synchronisationsbedingung definiert. Unser Verfeinerungsbegriff erlaubt neue Nebenläufigkeiten und ist eigenschafts-haltend.

Wir haben drei Grundtypen von Verfeinerungsnetzen gefunden, die wir sowohl beim Nachweis der Gültigkeit der Blockbedingung als auch beim Entwurf des Verfeinerungsnetzes, das die Blockbedingung garantiert, verwenden können.

Weiterhin haben wir eine Methode zum Entwurf und zur Verifikation nachrichtenbasierter verteilter Algorithmen durch verteilende Verfeinerung entwickelt. Mit dieser

Methode sind der Entwurf und die Verifikation leicht und verständlich.

Im Folgenden geben wir einige offene und weiter zu untersuchende Probleme an.

1. Verteilen einer unverteilten Transition von n Agenten in Agentensystemen.

In dieser Arbeit haben wir die Verteilung unverteilter Transitionen von zwei Agenten behandelt. Darauf basierend können wir die Verteilung unverteilter Transitionen von n Agenten untersuchen. Für den Fall von n Agenten gibt es z.B. den folgenden Weg. Angenommen, es gibt n Agenten a_1, a_2, ..., a_n. Zunächst betrachten wir a_1, a_2,..., a_{n-1} als einen Agenten a_{n-1}' und verteilen die unverteilte Transition der beiden Agenten a_{n-1}' und a_n. Wir erhalten dann ein System, in dem es unverteilte Transitionen von Agenten a_1, a_2,..., a_{n-1} gibt. Und dann verteilen wir diese unverteilten Transitionen folgendermaßen: Wir betrachten a_1,...,a_{n-2} als einen Agenten a_{n-2}', und verteilen jede unverteilte Transition von a_{n-2}' und a_{n-1} usw. Am Ende erhalten wir ein System, in dem es nur verteilte Transitionen von einzelnen Agenten a_1, a_2,..., a_n gibt.

2. Verteilende Verfeinerung im Modell Algebraische Petrinetze

Komplexe verteilte Algorithmen werden oft durch Algebraische Petrinetze modelliert (über Modellierung und Verifikation verteilter Algorithmen mit Algebraischen Petrinetzen, siehe [Rei98]). Damit wir direkt im Modell Algebraische Petrinetze die Ideen von verteilender Verfeinerung anwenden können, um einen verteilten Algorithmus zu verifizieren, müssen wir noch den Begriff verteilende Verfeinerung von Elementaren Petrinetzen in Algebraische Petrientze übertragen. In Algebraischen Petrinetzen modelliert eine Transition durch Variablen eine Menge von Aktionen. Die verteilende Verfeinerung im Modell Algebraische Petrinetze basiert auf der verteilenden Verfeinerung im Modell elemantare Petrinetze. Die Verteilung einer Transition im Algebraischen Petrinetz entspricht im elementaren Petrinetz den Verteilungen aller Aktionen dieser Transition. Beim Verteilen einer Transition müssen wir sichern, dass jede Aktion dieser Transition korrekt verteilt wird, d.h. beim Verteilen die Blockbedingung und die Synchronisationsbedingung erfüllt sind. Die Blockbedingung für Algebraische Petrinetze wurde schon in der Literatur [Peu01] formalisiert. Eine weitere Arbeit ist die Formalisierung der Synchronisationsbedingung für Algebraische Petrinetze und der Begriff verteilende Verfeinerung für Algebraische Petrinetze.

3. Erhaltenbleiben gewünschter Eigenschaften und Kausalitäten im Verfeinerungsnetz

Beim Verteilen einer unverteilten Transition t müssen wir außer der Blockbedingung noch garantieren, dass gewünschte Eigenschaften erhalten bleiben. Dafür müssen wir noch herausfinden, welche Kausalitäten im Verfeinerungsnetz das Erhaltenbleiben einer gewünschten Eigenschaft garantieren. Direkt aus der Arbeit erhalten wir die folgende Tabelle (Tabelle 8.1). In der ersten Spalte steht die zu erhaltende temporal-logische Eigenschaft ϕ und in der zweiten Spalte steht die Kausalität im Verfeinerungsnetz, die das Erhaltenbleiben von ϕ garantiert. Die Eigenschaften von der 2. Zeile bis zu der 6. Zeile in der Tabelle sind über *Interface*-Stellen von t (also $^\bullet t \cup t^\bullet$) dargestellt. Zeile 7 bis 10 machen Aussagen über Eigenschaften, die über Stellen von ganzen Netz dargestellt sind.

Tabelle 8.1 Erhaltenbleiben von Eigenschaften und Kausalitäten im Standardverfeinerungsnetz

	zu erhaltende Eigenschaft	Kausalität im Verfeinerungs Netz	Referenz
1	$p \rhd q$, wobei $p \sim_A q$ (d.h. p,q gehören zu dem selben Agenten).	keine	Lemma 5.3.10
2	$p \rhd q$, wobei $p \in {}^\bullet t$, $q \in t^\bullet$ und $\neg(p \sim_A q)$	$p^\bullet \blacktriangleleft {}^\bullet q$	Lemma 5.3.9
3	$p \rhd q$, wobei $p,q \in {}^\bullet t$, $\neg(p \sim_A q)$	$p^\bullet \blacktriangleleft q^\bullet$	Lemma 5.3.9
4	$\Box\neg(p \wedge q)$, wobei $p \in {}^\bullet t$, $q \in t^\bullet$ und $\neg(p \sim_A q)$	$p^\bullet \blacktriangleleft {}^\bullet q$	Lemma 5.3.13
5	$\Box(p \rightarrow q)$, wobei $p,q \in {}^\bullet t$, $\neg(p \sim_A q)$	$p^\bullet \blacktriangleleft q^\bullet$	Lemma 5.3.13
6	$\Box(p \rightarrow q)$, wobei $p,q \in t^\bullet$, $\neg(p \sim_A q)$	${}^\bullet q \blacktriangleleft {}^\bullet p$	Lemma 5.3.13
7	$R \rhd Q$, wobei $P,Q \subseteq P$, $Q \cap {}^\bullet t = \emptyset$	$({}^\bullet t)_a \blacktriangleleft (t^\bullet)_b$ und $({}^\bullet t)_b \blacktriangleleft (t^\bullet)_a$	Lemma 5.3.3
8	$\Box\neg(p \wedge q)$, wobei $p,q \in P$	$({}^\bullet t)_a \blacktriangleleft (t^\bullet)_b$ und $({}^\bullet t)_b \blacktriangleleft (t^\bullet)_a$	Lemma 5.3.11
9	$\Box(Q \rightarrow \neg p)$, wobei $Q \subseteq P$, $p \in P$	$({}^\bullet t)_a \blacktriangleleft (t^\bullet)_b$ und $({}^\bullet t)_b \blacktriangleleft (t^\bullet)_a$	Bemerkung 5.3.12
10	$x \blacktriangleleft y$, wobei $x,y \in P \cup T$	$({}^\bullet t)_a \blacktriangleleft (t^\bullet)_b$ und $({}^\bullet t)_b \blacktriangleleft (t^\bullet)_a$	Folgerung 5.3.2

P,T: Stellen- bzw. Transitionsmenge eines Systems Σ mit Agentenmenge A. $t \in T$: Eine unverteilte Transition von Agenten a und b.

Literaturverzeichnis

[AL91] Martin Abadi and Leslie Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82: 253 – 284, 1991.

[BDE93] E. Best, R. R. Devillers, J. Esparza. General Refinement and Recursion Operators for the Petri Box Calculus. In P. Enjalbert, A. Finkel, K. W. Wagner (Eds.): STACS 93, *10th Annual Symposium on Theoretical Aspects of Computer Science*, Bd. 665 aus *Lecture Notes in Computer Science*, Seiten 130-140. Springer-Verlag, 1993.

[BGV90] Wilfried Brauer, Robert Gold, Walter Vogler: A survey of behaviour and equivalence preserving refinements of Petri nets. In: G. Rozenberg editor, *Advances in Petri Nets*, Bd. 483 aus *Lecture*

Notes in Computer Science, Seiten 1-46. Springer-Verlag, 1990.

[Bro97] Manfred Broy. Compositional refinement of interactive systems. *Journal of the ACM*, Volume 44, Number 6, Seiten 850-891. 1997.

[BS96] R. J. R. Back and K. Sere. Superposition refinement of reactive systems. *Formal Aspects of Computing*, 8: 324-346, 1996.

[CM88] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addision-Wesley, 1988.

[Des97] Jörg Desel. How distributed algorithms play the token game. In: C. Fesksa, M. Jantzen und R. Valk, Hrsg., *Foundations of Computer Science – Potential, Theory, Cognition*, Bd. 1337 aus *Lecture Notes in Computer Science*, Seiten 297-306. Springer-Verlag, 1997.

[dRe98] Willem-Paul de Roever and Kai Engelhardt. *Data Refinement: Model-oriented Proof Methods and their Comparison*, volume 47 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.

[HHS87] C. A. R. Hoare, Jifeng He and Jedd W. Sanders. Prespecification in data refinement. *Information Processing Letters*, 25: 71-76, 1987.

[Hoa99] C. A. R. Hoare. Theories of Programming: Top-Down and Bottom-Up and Meeting in the Middle. In E. R. Olderog, B. Steffen (Eds.): *Correct System Design, Recent Insight and Advances*, Bd. 1710 aus *Lecture Notes in Computer Science*, Seiten 3-28. Springer-Verlag, 1999.

[Kin95] Ekkart Kindler. *Modularer Entwurf verteilter Systeme mit Petrinetzen*, Bd. 1 aus Edition VERSAL. Bertz Verlag, 1995. Dissertation, Technische Universität München.

[LL90] Leslie Lamport and Nancy Lynch. Distributed computing: Models and methods. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantiks, Chapter 18, pages 1158-1199. Elsevier Science Publishers B. V., 1990.

[Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[ME92] Carroll Morgan and Trevor Vickers (Eds.). *On the Refinement Calculus*. FACIT. Springer-Verlag, 1992.

[MP92] Zohar Manna und Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems. Specification*. Springer-Verlag, 1992.

[Peu01] Sibylle Peuker. *Halbordnungsbasierte Verfeinerung zur Verifikation verteilter Algorithmen*. Dissertation, Humboldt-Universität zu Berlin. 2001.

[Rei85] Wolfgang Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[Rei87] Wolfgang Reisig. A Strong Part of Concurrency. In: G. Rozenberg editor, *Advances in Petri Nets*, Bd. 266 aus *Lecture Notes in Computer Science*, Seiten 238-272. Springer-Verlag, 1987.

[Rei95] Wolfgang Reisig. Petri Net Models of Distriduted Algorithms. In: Jan van Leeuven, Hrsg., *Computer Science Today. Recent Trends and Developments*, Bd. 1000 aus *Lecture Notes in Computer Science*, Seiten 441-454. Springer-Verlag, 1995.

[Rei98] Wolfgang Reisig. *Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets.* Springer-Verlag, 1998.

[RK97] Wolfgang Reisig und Ekkart Kindler. Verification of Distributed Algorithms with Algebraic Petri Nets. In: C. Fesksa, M. Jantzen und R. Valk, Hrsg., *Foundations of Computer Science – Potential, Theory, Cognition*, Bd. 1337 aus *Lecture Notes in Computer Science*, Seiten 297-306. Springer-Verlag, 1997.

[SL90] A. U. Shankar, S. S. Lam. Construction of Network Protocols by Stepwise Refinement. In J, W. de Bakker, W. P. de Roever and G. Rozenberg (Eds.): *Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness*, Bd. 430 aus *Lecture Notes in Computer Science*, Seiten 669-695. Springer-Verlag, 1990.

[Val79] R. Valette. Analysis of Petri Nets by stepwise refinement. *Journal of Computer and System Sciences*, 18: 35-46, 1979.

[vGG01] R. J. van Glabbeek, Ursula Goltz: Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, Volume 37, Number 4/5: 229-327, 2001.

[vGG89] R. J. van Glabbeek and U. Goltz. Refinement of actions in causality based models. In J, W. de Bakker, W. P. de Roever and G. Rozenberg (Eds.): *Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness*, Bd. 430 aus *Lecture Notes in Computer Science*, Seiten 267-300. Springer-Verlag, 1990.

[Vog87] W. Vogler: Behaviour Preserving Refinement of Petri Nets. In G. Tinhofer, G. Schmidt (Eds.): *Graphtheoretic Concepts in Computer Science*, Bd. 246 aus *Lecture Notes in Computer Science*, Seiten 82-93. Springer-Verlag, 1987.

[Vog92] W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets*. *Lecture Notes in Computer Science* Vol. 625. Springer-Verlag, 1992.

[Wal95] Rolf Walter. *Petrinetzmodelle verteilter Algorithmen. Beweistechnik und Intuition,* Bd. 2

aus Edition VERSAL. Bertz Verlag, 1995. Dissertation, Humboldt-Universität Berlin.

[Wu07] Bixia Wu. *Entwurf und Verifikation von Petrinetzmodellen verteilter Algorithmen durch Verfeinerung unverteilter Algorithmen*. Dissertation, Humboldt-Universität zu Berlin. 2007.

[WWV+97] M. Weber, R. Walter, H. Völzer, T. Vesper, W. Reisig, S. Peuker, E. Kindler, J. Freiheit, and J. Desel. *DAWN: Petrinetzmodelle zur Verifikation Verteilter Algorithmen.* Informatik-Berichte 88, Humboldt-Universität zu Berlin, Dezember 1997.