Analyse von Message Sequence Charts

Philipp Lucas

Diplomarbeit in Informatik an der
Rheinisch-Westfälischen Technischen Hochschule
Aachen
vorgelegt der Fakultät für
Mathematik, Informatik und Naturwissenschaften

Lehrstuhl für Informatik II	Lehrstuhl für Informatik VII
Prof. Dr. rer. nat. Indermark	Prof. Dr. rer. nat. Thomas

Betreuer: Dipl.-Inform. Bollig

Diese Arbeit wurde vom Autor mit LaTeX 2$_\varepsilon$ erstellt. Dabei waren insbesondere die Makro-Pakte XY-pic von KRISTOFFER H. ROSE, GasTeX von PAUL GASTIN und MSC von SJOUKE MAUW und VICTOR BOS eine große Hilfe.

Ich danke MARTIN LEUCKER, der mich auf MSCs aufmerksam machte, und Prof. KLAUS INDERMARK und Prof. WOLFGANG THOMAS für Überlassung des Themas und Begutachtung der Arbeit.

Ich danke BENEDIKT BOLLIG, der geduldig und konstruktiv seine segnende Hand über diese Arbeit hielt.

Благодаря ВЕНЦИСЛАВА ВАСИЛЕВА, че през последните месци моят живот не се върти съвсем на диаграми и автомати.

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig erstellt und keine anderen Quellen und Hilfsmittel als die angezeigten benutzt habe und alle Zitate als solche gekennzeichnet habe.

Aachen, 3. Februar 2002.

Inhaltsverzeichnis

Abbildungsverzeichnis

Mathematische Vorbemerkungen

Zahlen

Um der Tradition genüge zu tun, wird 0 als natürliche Zahl betrachtet. Wir setzen $\mathbb{P} := \{x \in \mathbb{R} : x \geqslant 0\}$ und $\mathbb{T} := \mathbb{P} \cap \mathbb{Q}$. Die Menge aller *Intervalle* über $\mathbb{T}$, also aller nicht leeren, konvexen Teilmengen von $\mathbb{T}$, wird mit $\mathbb{I}$ bezeichnet, wobei wir für $x \in \mathbb{I}$ die üblichen Notationen wie $(5, 6]$, $[7, \infty)$ oder $[0, 0]$ verwenden.

Für ein $x \in \mathbb{P}$ ist der *ganzzahlige Anteil von* x $\lfloor x \rfloor := \max\{n \in \mathbb{N} : n \leqslant x\}$ und der *Bruchteil von* x $\mathfrak{f}(x) := x - \lfloor x \rfloor$.

Funktionen

Funktionsverkettung mit dem $\circ$-Operator ist von rechts nach links zu verstehen, also $(g \circ h)(x) = g(h(x))$. Wo nützlich, wird die Lambda-Schreibweise von Funktionen in der Form $\lambda x \in M.\varphi$ oder kurz $\lambda x.\varphi$ verwendet. Sollten bei einer Funktionsdefinition die Werte von Argumenten irrelevant sein, so wird dies mit _ (*Joker, don't-care*) angedeutet (Zum Beispiel kann die konstante 1-Funktion auf $\mathbb{N}$ als $\lambda\_.1$ geschrieben werden.). Die Identitätsfunktion auf M ist id_M, oder kurz id, falls M klar ist. Der Raum aller totalen Funktionen von M nach M ist $[M \to N]$.

Partielle Funktionen werden mit $\dashrightarrow$ statt $\to$ geschrieben. Ist der Wert einer partiellen Funktion f an der Stelle x undefiniert, so schreiben wir $f(x) = \bot$. Der Raum aller partiellen Funktionen von M nach M ist $[M \dashrightarrow N]$.

Ist $f : M \dashrightarrow N$ eine beliebige Funktion und sind $m \in M$, $n \in N$, so wird mit $f[m/n]$ die Funktion

$$\lambda m'. \begin{cases} n & m = m' \\ f(m') & \text{sonst} \end{cases}$$

bezeichnet. Statt $f[m/n][m'/n']$ schreibt man auch $f[m/n, m'/n']$, falls klar ist, daß $m \neq m'$; diese paarweise Verschiedenheit wollen wir immer voraussetzen, wenn wir diese Schreibweise benutzen. Weiter ist $f[X/n]$ die Kurzschreibweise von (mit $X = \{x_\nu\}$) $f[x_1/n, \ldots, x_{|X|}/n]$.

Sei $f \in [M \to N]$. Dann wird f auf natürliche Weise auf Tupel $V_1 \times M \times V_2$ fortgesetzt durch $f(v_1, m, v_2) = (v_1, f(m), v_2)$, oder auf Teilmengen $N \subseteq M$ durch $f(N) := \{f(n) : n \in N\}$.

Sind $f_1 : M_1 \dashrightarrow N_1$ und $f_2 : M_2 \dashrightarrow N_2$ beliebige Funktionen mit $M_1 \cap M_2 = \emptyset$, so definieren wir

$$f_1 \vee f_2 : M_1 \cup M_2 \dashrightarrow N_1 \cup N_2,\ x \mapsto \begin{cases} f_1(x) & x \in M_1 \\ f_2(x) & x \in M_2. \end{cases}$$

Relationen

Da Funktionen direkt als Relationen ansehbar sind, werden wir dies hier auch machen und z. B. $f \subseteq R$ für eine Funktion f und eine Relation R schreiben.

Die Verknüpfung zweier Relationen $R_1 \subseteq M_1 \times M_2$ und $R_2 \subseteq M_2 \times M_3$ ist definiert als $R_2 \circ R_1 := \{(m_1, m_3) : \exists m_2 \in M_2.(m_1, m_2) \in R_1 \wedge (m_2, m_3) \in R_2\}$. $R^1 := R$, $R^k := R \circ R^{k-1}$ für $k > 1$, $R^+ := \bigcup_{n=1}^{\infty} R^n$, $R^* := R^+ \cup \mathrm{id}$, $R^? := R \cup \mathrm{id}$. Für eine Relation $R \subseteq M_1 \times M_2$ gilt $\mathrm{range}(R) = \{n \in M_2 : (M_1 \times \{n\}) \cap R \neq \emptyset\}$ und $\mathrm{dom}(R) = \{n \in M_1 : (\{n\} \times M_2) \cap R \neq \emptyset\}$.

Eine *Halbordnung* ist hier eine irreflexive, transitive, asymmetrische binäre Relation auf einer Menge; ist $\prec \subseteq M^2$ für eine Menge M, so ist also $\prec$ eine Halbordnung auf M, falls $\prec \cap\, \mathrm{id} = \emptyset \wedge \prec^+ \subseteq \prec$, also falls

- $\neg \exists m \in M. m \prec m$ und
- $\forall m, m' \in M.((\exists m'' \in M. m \prec m'' \prec m') \rightarrow m \prec m')$.

Für $a \in M$ ist $\mathfrak{a}(a) := \{a' \prec a\}$. Da normalerweise klar ist, was die Grundmenge ist, wird diese Information bei der Angabe von Halbordnungen nicht erwähnt.

Ein Halbordnung $\prec$ (auf M), sodaß zusätzlich für alle $m, m' \in M$ mit $m \neq m'$ gilt $m \prec m'$ oder $m' \prec m$, heißt eine *totale Ordnung (auf M)*. Gilt für eine totale Ordnung $\prec'$ und eine Halbordnung $\prec$, daß $\prec' \supseteq \prec$, so heißt $\prec'$ eine *Linearisierung* von $\prec$. $\mathfrak{l}(\prec)$ ist die Menge aller Linearisierungen von $\prec$. Manchmal identifizieren wir eine totale Ordnung mit dem dadurch eindeutig bestimmten Wort, welches aus den Elementen des Universums in der geordneten Folge entspricht.[1]

Mengen

Die *Potenzmenge* einer Menge M, also die Menge aller Teilmengen von M, wird mit $\mathfrak{p}(M)$ bezeichnet.

$M \uplus N$ bedeutet die *disjunkte Vereinigung* der Mengen M und N. Dies läßt sich etwa darstellen durch $M \uplus N := (M \times \{0\}) \cup (N \times \{1\})$. Meistens identifizieren wir die Elemente aus M mit denen aus $M \times \{0\}$ in $M \uplus N$, und entsprechend für N. In den Fällen, in denen es als nützlich erscheint, ein $e \in M$ von $(e, 0) \in M \uplus N$ zu unterscheiden, so schreiben wir $e \sim (e, 0)$ (und entsprechend für $(e, 1)$) und setzen $\sim$ zur kleinsten $\sim$ enthaltenden Äquivalenzrelation fort.

[1] In dieser Arbeit werden keine Ordnungen auf überabzählbaren Mengen benutzt außer dem üblichen $<$ auf $\mathbb{R}$.

Sprachen

Für Mengen M, N setzen wir N^M als die Menge aller (totalen) Funktionen $M \to N$, wobei für $n \in \mathbb{N}$ wie üblich $M^n := M^{\{1,\dots,n\}}$. Die Tupel-Schreibweise wie etwa $(w_1, w_2) \in M^2$ wird bei endlichem Definitionsbereich benutzt. Für $w \in M^n$ ist w_i für $i \in \mathbb{N}$ mit $1 < i \leqslant n$ immer auf natürliche Weise definiert. Das einzige Element aus M^0 wird mit ε bezeichnet.[2] $M^+ := \bigcup_{i=1}^{\infty} M^i$, $M^* := M^+ \cup \{\varepsilon\}$. Elemente aus M^* heißen *Wörter*. Wortkonkatenation wird durch den binären Operator $\cdot$ ausgedrückt, der üblicherweise weggelassen wird:

$$w \cdot w' = \lambda n. \begin{cases} w_n & n \leqslant |w| \\ w'_{n-|w|} & n > |w|. \end{cases}$$

Ein *Alphabet* ist immer eine endliche, nicht-leere Menge. Ist Σ ein Alphabet, so ist $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. Für ein $w \in \Sigma^*$ ist $|w|$ die Anzahl der Elemente von w. Ist außerdem $a \in \Sigma$, so wird die Anzahl der Vorkommen von a in w mit $|w|_a$ bezeichnet.

Automaten

Ein *Automat* ist ein 5-Tupel $A := \langle Q, \Sigma, \delta, Q_0, Q_* \rangle$ mit $Q_0 \subseteq Q$ als Anfangszustandsmenge, $\delta : Q \times \Sigma_\varepsilon \to \mathfrak{p}(Q)$ als nichtdeterministischer Transitionsrelation und $Q_* \subseteq Q$ als Endzustandsmenge. In der graphischen Darstellung werden Endzustände doppelt umkreist gezeichnet und normale Zustände einfach, oder auch normale Zustände ohne Rand, Endzustände mit Kreis. Die Erweiterung von δ auf Wörter wird mit $\overline{\delta}$ bezeichnet.

Ist $\forall q \in Q.(\delta(q, \varepsilon) = \emptyset \wedge \forall a \in \Sigma.|\delta(q, a)| = 1)$, so schreiben wir δ auch in der dann isomorphen Form als Funktion $Q \times \Sigma \to Q$. Statt $\delta(q, a) \ni q'$ wird auch $q \to_A^a q'$ geschrieben, oder $q \to^a q'$, falls A klar ist. Mit der Erweiterung auf Wörter schreibt man dann auch in der Kurznotation $q \to_A^w q'$ oder $q \to^w q'$ falls $|w| \neq 1$. Gibt es irgendein ξ mit $q \to_A^\xi q'$, so schreibt man auch einfach $q \to_A q'$ (oder $q \to q'$). Die von einem Automaten A erkannte Sprache wird mit $\mathfrak{L}(A)$ bezeichnet. Zwei Automaten A_1, A_2 heißen *äquivalent* ($A_1 \equiv A_2$), falls $\mathfrak{L}(A_1) = \mathfrak{L}(A_2)$. Ist Q endlich, so heißt der Automat auch *endlich*.

Graphen

Ein *(gerichteter) Graph* ist ein Paar $G := (M, \longrightarrow)$ von einer Menge M und einem $\longrightarrow \subseteq M \times M$. Ist $N \subseteq M$, so heißt $(N, \longrightarrow \cap (N \times N))$ der *durch N induzierte Teilgraph*. Er wird auch als $G|_N$ geschrieben.

G heißt *(stark) zusammenhängend*, falls $\longrightarrow^+ = M \times M$. Ein *(endlicher) Weg* in G ist eine Folge $v := v_1 v_2 \dots v_n \in M^*$ mit $\forall i \in \{1, \dots, n-1\}.v_i \longrightarrow v_{i+1}$. Ist zusätzlich $v_n = v_1$, so heißt v eine *Schleife*; sind zusätzlich alle v_s für $1 \leqslant s < n$ disjunkt, so heißt v *einfache* Schleife. Ein Graph $(M, \longrightarrow)$ heißt *reflexiv*, falls $m \longrightarrow^+ m$ für jedes $m \in M$.

Für einen Graphen $(M, \longrightarrow)$ und eine Funktion $\rho : \longrightarrow \to \mathbb{R}$ heißt $(M, \longrightarrow, \rho)$ *gewichteter Graph*. Dann ist $\mathfrak{d}(v_1, v_2)$ für $v_1, v_2 \in M$ die *Länge des kürzesten Pfades von v_1 nach v_2*.

[2] Also $M^0 = N^0$ für alle Mengen M, N.

Sonstiges

Die *Landau-Symbole groß-Oh/klein-Oh* werden als $\mathfrak{O}/\mathfrak{o}$ geschrieben. Wie üblich benutzen wir Schreibweisen wie $f(n) = \mathfrak{O}(n^2)$ statt $f(n) \in \mathfrak{O}(n^2)$.

Ist B eine Aussage, die wahr oder falsch sein kann, so ist $B?e_1 : e_2$ (wie aus manchen Programmiersprachen bekannt) e_1, falls B wahr ist, und e_2, falls B falsch ist.

Kapitel 1

Einführung

Eine Sache, die sich aufklärt,
hört auf, uns etwas anzugehen.
FRIEDRICH NIETZSCHE, *Jenseits von Gut und Böse*, 1886

Message Sequence Charts (im weiteren Verlauf kurz *MSCs*) sind ein teilweise formales Beschreibungsmittel für die Modellierung verteilter Systeme.[1] Durch die graphische Darstellbarkeit ist eine Verständigung auch mit nicht fachlich vorbelasteten Personen möglich[2], durch eine formale Fassung der jeweiligen graphischen Darstellungsmerkmale ist dann eine weitere Untersuchung mit den üblichen Mitteln formaler Methoden möglich.

Da MSCs üblicherweise in den sehr frühen Phasen der Software-Entwicklung benutzt werden ([AY99, M01b, GM02]), können Fehler schon weit vor der Erstellung genauerer Architekturdokumente, der Implementierung und dem Test des fertigen Systems entdeckt werden, also zu einem Zeitpunkt, an dem die Behebung eines Fehlers noch relativ billig ist. Dabei wird meist von Datenübertragungen und quantitativem Zeitverhalten abstrahiert.

Wegen dieser Vorteile werden MSCs und verwandte Sprachen schon seit langer Zeit verwendet. Vorläufer dieser Notation lassen sich bis in die 1970er Jahre zurückverfolgen, nämlich zu den *Lamport-Diagrammen* aus [L78]. Schon relativ früh wurden Anstrengungen unternommen, diese Spezifikationsdokumente zu standardisieren, was seit Anfang der 1990er Jahre durch die *International Telecommunications Union (ITU)*[3] gesteuert wurde. Gegen Ende des 20. Jahrhunderts entwickelten sich dann die Methoden zum Umgang mit MSCs schnell weiter, und so kann man heute auf einen wohlverstandenen Mechanismus zurückgreifen. Eine kleine Einführung von einem eher praktischen Standpunkt aus wird in [RGG96] gegeben.

Leider ist die formale Fassung der Semantik von MSCs nicht vollständig auf alle Konstruktionsmerkmale ausgedehnt worden. Tatsächlich wurden zunächst die syntaktischen Strukturen der MSCs definiert, bevor man sich um die Semantik kümmerte, was zu diversen Mißverständnissen und Sackgassen in der Entwicklung führte (man betrachte dazu etwa [MR99, GRG93]; Das frühe Werk [LL93b]

[1] *Modellierung* ist hier zu verstehen als der Prozeß, der bestimmt, welches Verhalten ein System zeigen soll, nicht hingegen, wie es dies konkret macht.

[2] MSCs waren natürlich nicht die ersten graphischen Beschreibungstechniken. Statecharts ([H87]) oder UML ([RJB99]) (beispielsweise) bieten ebenfalls eine sehr ausgefeilte Syntax.

[3] Damals noch *Comité Consultatif International Téléphonique et Télégraphique (CCITT).*

trägt zurecht den Titel *What do Message Sequence Charts Mean?*). Das ist dann auf viele Weisen nachträglich geändert worden. Die Arbeiten der Eindhovener Gruppe um MAUW und RENIERS führten dann schließlich zur offiziellen Definition durch eine operationelle Semantik, wie sie im ITU-Standard [ITU99] nierdergelegt ist ([CEMR98]; s. auch [MR99] und [R99]). Andere Semantiken benutzten eine Abbildung in Petri-Netze ([GRG93]) oder in diverse Automaten (z. B. [LL93a, HMKT99, BLN02a]). Auch jetzt noch sind manche Teile der Syntax nicht vollständig semantisch erklärt, und manche Teile der offiziellen Semantik sind umstritten ([E00]). In dieser Arbeit müssen ebenfalls Teile des Standards behutsam angepaßt werden. Dies geschieht jedoch nie mutwillig, sondern nur an Stellen, an denen dies aufgrund von Unterspezifikation oder erforderlicher Erweiterung nötig ist, und wird eindeutig als Erweiterung oder Änderung benannt werden.

Hier soll zunächst für ein spezielles Merkmal von MSCs, nämlich die Spezifikation des zeitlichen Verhaltens, eine Übersetzung in *Timed Automata (Uhrenautomaten)* angegeben werden. Diese wurden von ALUR und DILL als natürliche Erweiterung endlicher Automaten um eine Bestimmung des Zeitverhaltens eingeführt und seitdem vielfach untersucht (für eine Übersicht betrachte man [A99]). Unter anderem wurde in [W94] die Übertragbarkeit in Logiken nachgewiesen. [BP99a] und [ACM97] geben eine Übertragbarkeit in etwas wie reguläre Ausdrücke an, [BP01] erreicht dabei die größte Nähe zu üblichen regulären Ausdrücken. [B98] gibt eine Version der traditionellen Iterationslemmata für Timed Automata, in [BPT01] erhalten wir eine Übertragung der Monoid-Erkennbarkeit. Somit scheint dieses Modell gut verstanden und mächtig genug, um bei der Untersuchung hilfreich zu sein.

Der Wert dieser Übersetzung liegt darin, daß wir für die Automaten die bekannten und beliebten Model-Checking-Algorithmen anwenden können und somit einen großen Apparat von Analysen für die gegebenen MSCs haben. *Model Checking* bedeutet dabei, daß wir ein MSC gegen eine Spezifikation in irgendeiner Logik testen, indem das MSC in einen Automaten A übersetzt wird, der die Sprache erkennt, die durch das MSC definiert wird, und die Spezifikation φ in einen Automaten $A_{\neg\varphi}$ übersetzt wird, der die Modellmenge von $\neg\varphi$ erkennt, und anschließend den Schnitt der Automaten auf Leerheit untersucht. Model Checking für Automaten wurde in [CE81] eingeführt.

Model Checking von (H)MSCs wurde auch in [AY99] vorgenommen, jedoch nicht mit Zeiten.

Betrachtungen über die Erweiterung von MSCs mit Zeitbeschränkungen wurden natürlich auch früher schon angestellt. In [L78] werden sehr ähnliche Diagramme vorgestellt; LAMPORT merkt dann an, daß ohne explizite Zeitmessungen ein Nutzer eines Systems keine Möglichkeit hat, festzustellen, ob ein Prozeß am Ende seiner Ausführung ist oder nur gerade eine Pause macht, und in der Tat sieht man als Außenstehender ein System ja genau dann als *abgestürzt* an, wenn er für eine gewisse Zeit keinerlei nach außen sichtbaren Aktionen durchführt. Schon hier wird also der Bedarf nach quantitativen Zeitspezifikationen deutlich.

In [AHP96] ist ein effektiver Mechanismus angegeben, mit dem bestimmte Einschränkungen nicht nur dargestellt, sondern auch überprüft werden können. Hier geht es darum, bei Einschränkungen an die Ausführungen eines einzelnen MSCs festzustellen, ob es überhaupt möglich ist, eine Implementierung zu erstellen, die die Spezifikation erfüllt.

[BAL97a] betrachtet ganz allgemein die Möglichkeiten, an Stellen in MSCs Referenzen auf Zeiten anbringen zu können und was dies jeweils nutzen mag. Insbesondere wird hier die Frage aufgeworfen, inwiefern wir überhaupt eine Beschreibung eines Systems durch ein MSC erwarten wollen. Besondere Schwierigkeiten ergeben sich hier bei *High-Level Message Sequence Charts (HMSCs)*, die eine vertikale Komposition einzelner MSCs beschreiben. Hier stellt sich, informell gesprochen, die Frage, ob Stoppuhren eines MSCs, das in einer Schleife eines HMSCs liegt, bei jedem Einlauf zurückgesetzt werden, überschrieben werden, oder weiterlaufen (für eine Diskussion dieses Problems siehe einführend [BAL97a] und weiter Kapitel 4 der vorliegenden Arbeit). Diese Schwierigkeiten sind vermutlich der Grund, warum in der Literatur dies bisher weniger untersucht wurde als das Zeitverhalten einzelner MSCs.

In letzter Zeit sind noch einige Erweiterungen von MSCs in der Literatur diskutiert worden. Zu nennen sind hier an erster Stelle die *Live Sequence Charts (LSCs)* aus [DH99, KW01], die nur angeben, welche Ereignisse stattfinden können, aber nicht mehr fordern, daß alle abgebildeten Ereignisse tatsächlich stattfinden. Insbesondere um die Stabilität des hier vorzustellenden Modells zu testen, soll es auf LSCs erweitert werden.

Ein weiteres wichtiges aktuelles Thema bei MSCs ist *Kompositionalit*. Grundsätzlich ist über die Möglichkeiten, MSCs zusammenzusetzen oder in irgendwie elementarere zu zerlegen schon einiges gesagt worden, etwa in [GMP01] oder [HM00]. Es soll gekärt werden, ob sich kompositionelle MSCs in der hier gegebenen Formalisierung ausdrücken lassen.

Dem kritischen Leser stellt sich vielleicht die Frage: „Wofür kann man das überhaupt brauchen?“ Der allgemeine Wert formaler Methoden in der Softwaretechnik braucht nicht betont zu werden, er liegt offenbar und wird von Praktikern wie Theoretikern eingesehen.[4] Praktiker haben aber manchmal andere Forderungen an formale Methoden. Insbesondere ist eine leichte Benutzbarkeit und leichte Erlernbarkeit für den Erfolg einer Methode sehr wichtig. Tatsächlich werden aber speziell MSCs auch in der Praxis eingesetzt. Für einen Überblick sei hier auf [MRW00] verwiesen. Man kann kritisch anmerken ([A97]), daß MSC-Spezifikationen leicht extrem unübersichtlich werden. Auch dagegen soll diese Arbeit wirken: Insbesondere die Vorstellung der LSCs kann helfen, Spezifikationen kürzer und somit leichter überblickbar zu halten.

Zum Abschluß sei der Leser auf das Symbolverzeichnis im Anhang hingewiesen. Es wurde versucht, durch Wahl der Schriftsätze und der Buchstaben die Bedeutungen der Bezeichnungen und Notationen möglichst intuitiv zu gestalten. Abgesehen von Fraktur und griechischen Buchstaben werden die verwendeten Zeichensätze nur in speziellen Themenbereichen bentutzt. Sollten dennoch Zweifelsfälle auftreten, sollten sie durch diesen Teil aufgelöst werden.

Teile dieser Arbeit werden auch in [L02] vorgestellt.

[4]Ein etwas älterer, aber sehr hübscher Artikel über die Relevanz formaler Methoden in der Praxis ist [CW96].

Kapitel 2

Message Sequence Charts

…“and what is the use of a book”, thought Alice,
“without pictures or conversation?”
LEWIS CARROL, *Alice’s Adventures in Wonderland*, 1865

In diesem Kapitel werden Message Sequence Charts vorgestellt und der übliche Weg, diese in größeren Modellierungen zusammenzufassen (High-Level Message Sequence Charts). Syntax und Semantik werden in Abgrenzung von den bisher üblichen Definitionen erklärt und motiviert. Anschließend wird eine semantisch äquivalente Charakterisierung von HMSCs über Automaten gegeben, wobei ein hinreichendes Kriterium für deren Endlichkeit angegeben wird und zitiert wird, daß dieses Kriterium nicht notwendig sein kann.

In diesem Kapitel wird das Zeitverhalten nicht betrachtet. Außer in einer Generalisierung bisheriger Definitionen liegt der Wert dieser Betrachtungen darin, daß in Kapitel 4 und Kapitel 5 die Erweiterungen der Syntax sich in ziemlich natürlichen Erweiterungen der nun vorzustellenden syntaktischen Konzepte niederschlagen.

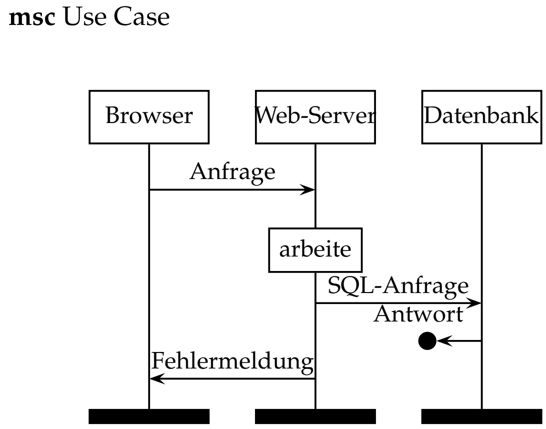
2.a Einfache Message Sequence Charts

„Message Sequence Charts are an [appealing | attractive] visual formalism…“ ist ein beliebter Einleitungssatz von Papern über MSCs. Die ästhetische Qualität der visuellen Darstellung soll hier nicht weiter untersucht werden, aber zunächst werden wir der Frage nachgehen, welche Vorteile die Tatsache hat, daß die Sprache eine graphische Syntax besitzt, und inwiefern man intuitiv eine nötige Formalisierung der Semantik aus den MSC-Bildern schon erfassen kann. Das ist insofern wichtig, als eine eindeutige graphische Repräsentation von aufwendigen Formalismen sicherlich viel zur notwendigen Akzeptanz der formalen Methoden in der Praxis beitragen kann. Anschließend werden die in dieser Arbeit benutzte mathematische Formalisierung vorgestellt und eine Semantikdefinition gegeben.

2.a.i Beispiele

Wir betrachten zunächst ein einzelnes Message Sequence Chart. Intuitiv soll ein MSC wie in Abbildung 2.1 bedeuten, daß ein System die als dieses MSC gegebene Spezifikation erfüllt, wenn die Abfolge der Ereignisse, die in diesem System

Abbildung 2.1 Ein MSC-Beispiel



stattfinden, der Abfolge entspricht, die durch das MSC vorgegeben ist. Dazu ist natürlich zunächst einmal zu untersuchen, was die Ereignisse des MSCs sind und welche Abfolge nun vorgegeben ist.

Ein *Prozeß* ist eine vertikale Linie mit nicht ausgefülltem Kästchen oben und ausgefülltem unten.[1] Den Namen des Prozesses schreiben wir in das obere Kästchen. In der Grundform von MSCs können in ihnen drei Ereignistypen auftreten:

- ▷ Nachrichten können *gesendet* und *empfangen* werden.
- ▷ Prozesse können *interne Aktionen* durchführen.

Interne Aktionen werden durch Kästchen auf der Prozeßlinie gekennzeichnet[2], Nachrichten durch einen Pfeil vom sendenden zum empfangenden Prozeß. Das Sende- und das Empfangsereignis sind im MSC da zu finden, wo dieser Pfeil die jeweilige Prozeßlinie einseitig tangiert; sollten andere Prozeßlinien zwischendurch gekreuzt werden, hat das keine Bedeutung.[3] Eine Nachricht muß nicht empfangen werden: Wenn das Kommunikationsmedium fehlerhaft ist, verschwindet die Nachricht einfach. Dies wird durch ein schwarzes Loch symbolisiert.[4]

Nachrichtenaustausch erfolgt asynchron.

Bei Nachrichten kann auch angegeben werden, was eigentlich gesendet wird. Das wird an den Nachrichtenpfeil geschrieben und soll Informationstransfer symbolisieren (vgl. auch die Diskussion über Daten in Abschnitt 2.d). Das nennt man die *Beschriftung* einer Nachricht. Obwohl Beschriftungen in unserem Modell auch erlaubt sind, werden sie in den Beispielen meistens nicht benutzt. Stattdessen numerieren wir die Nachrichten durch. Das dient dazu, in der Diskussion leichter auf die

[1] Der Standard erlaubt hier und bei anderen Elementen Variationen der graphischen Darstellung. Da diese hier aber ohnehin nur zur leichteren Erklärung von Beispielen gebraucht wird, wird auf solche Alternativen nicht weiter eingegangen.

[2] Da die Prozesse verteilt sind, können sie keine gemeinsamen internen Aktionen durchführen.

[3] Eine Nachricht von Prozeß 1 zu 3 kreuzend 2 ist unterscheidbar von zwei Nachrichten von 1 nach 2 und von 2 nach 3 durch Anzahl und Ort der Pfeilspitzen.

[4] Verfälschungen von Nachrichten gibt es nicht: Wenn etwas ankommt, ist es genau das, was gesendet wurde.

Unvollständige Nachrichten, also schwarze Löcher (und weiße, aus denen spontan Nachrichten herauskommen) unterstützt diese Arbeit erst ab Kapitel 5.

einzelnen Ereignisse Bezug nehmen zu können: So werden wir dem Senden und dem Empfangen der i-ten Nachricht die Ereignisnamen s_i bzw. r_i zuordnen. Das ist *keine* Änderung der MSCs, sondern dient ausschließlich der Übersichtlichkeit.

Auch Aktionen werden beschriftet: Man will schließlich angeben können, welche Aktion ausgeführt wird, und nicht nur, das ein Prozeß irgendetwas macht.

Man geht allgemein davon aus, daß die gesamte Kommunikation aller Prozesse in einem MSC abgebildet ist, daß also insbesondere kein weiterer Informationsfluß stattfindet und keine Synchronisierungen mehr existieren. Damit kann durchaus die Situation eintreten, daß eine Spezifikation über Use Cases durch MSCs erfordert, daß ein diese MSCs implementierendes System auch ein weiteres MSC implementiert, welches nicht erwünscht ist, aber wegen mangelnder Synchronisation nicht zu verhindern ist (Beispiel 2.2).

Die Abfolge von Ereignissen ist auf natürliche Weise definiert:

- ▹ Steht ein Ereignis e auf einer Prozeßlinie oberhalb eines f, so muß e vor f stattfinden.
- ▹ Das Senden einer Nachricht muß vor dem Empfangen stattfinden.
- ▹ Ansonsten ist keine Einschränkung an die Ordnung gegeben.[5]

Diese Erklärung zeigt, daß es leicht ist, beim Blick auf ein MSC zu verstehen, was damit gemeint ist, und daß man im Prinzip auch leicht eine formale Semantik aus der graphischen Syntax ableiten kann. Die Schwierigkeiten, die noch zu nennen sein werden, liegen nicht offenbar.

Beispiel 2.1 Betrachten wir etwa unser Beispiel aus Abbildung 2.1, so wird dadurch ein Use Case folgender Art ausgedrückt:[6]

1. Zuerst sendet ein Nutzer eine Anfrage an einen Web-Server, der beispielsweise das Web-Frontend eines Internet-Kaufhauses sein kann.
2. Das System verarbeitet die empfangene Nachricht und sendet eine gewisse Anfrage an eine Datenbank (zum Beispiel die Frage, ob ein Produkt lieferbar ist).
3. Die Datenbank schickt folgsam eine Antwort zurück, doch diese verschwindet auf unbekannte Weise.
4. Dem Kunden wird eine Fehlermeldung geschickt.

Damit scheint auch die Abfolge der Ereignisse vorgegeben. Doch halt!, wie kann der Web-Server wissen, daß die Antwort der Datenbank ihn niemals erreichen wird? Die graphische Darstellung ist trügerisch: Es ist überhaupt nicht sichergestellt, wie die Abfolge des Sendens der Datenbankantwort und der Fehlermeldung zueinander liegen, obwohl die vertikale Anordnung einen dazu verleiten mag, auch für voneinander unabhängige Nachrichten eine bestimmte Abfolge anzunehmen. Tatsächlich sind die Ereignisse jedoch ungeordnet, mithin also nebenläufig im eigentlichen Sinne. Da die Prozesse keine anderen Nachrichten austauschen als

[5] Ist $\prec$ die durch ein MSC definierte Halbordnung, so ist die graphische Darstellung des MSCs dem Graphen der Überdeckungsrelation von $\prec$ sehr ähnlich.

[6] Ausnahmsweise sind hier die Nachrichten beschriftet, um die praktische Anwendbarkeit in den Vordergrund zu rücken.

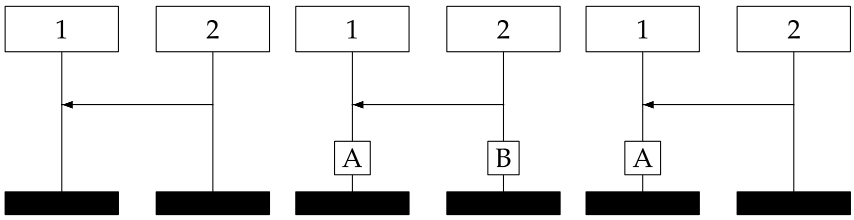
die angegebenen, ist für den Web-Server eine nicht ankommende Nachricht ununterscheidbar von einer, die noch ein wenig Zeit benötigt. Was wir gerne hätten, wäre eine Zeitbeschränkung der Art, daß 30 Sekunden nach dem Absenden der SQL-Anfrage die Fehlermeldung ausgegeben wird. Das erscheint sinnvoll, denn für den Benutzer ist eine lange Wartezeit äquivalent zum Absturz des Systems. Dies können wir offensichtlich nicht durch einfache Ordnungen modellieren. Wir haben also ein Problem.

Wenn man dieses Zeitverhalten modellieren kann, kann das angegebene Szenario erweitert werden mit einer Art Fallunterscheidung: „Prozeß Web-Server: Falls die Antwort doch vor Ablauf der 30 Sekunden eintrifft, schicke sie, nach einer geeigneten Konvertierung in HTML, an den Benutzer, ansonsten die Fehlermeldung.“ Das wäre dann schon ein sinnvoller und klarer Entwurf eines verteilten Systems.

Im nächsten Beispiel soll das Problem der Inferenz von MSCs wegen mangelnder Synchronisation noch kurz angesprochen werden.

Beispiel 2.2 Will man ein System implementieren, welches die beiden ersten MSCs in Abbildung 2.2 als Use Cases besitzt, wird dieses System auch das dritte MSC erlauben: Die Prozesse wissen nicht, welche internen Aktionen der jeweils andere durchführt. In [AEY00] wird dieses Thema eingehend untersucht.

Abbildung 2.2 Zwei MSCs, ein drittes erzeugend (Beispiel 2.2)



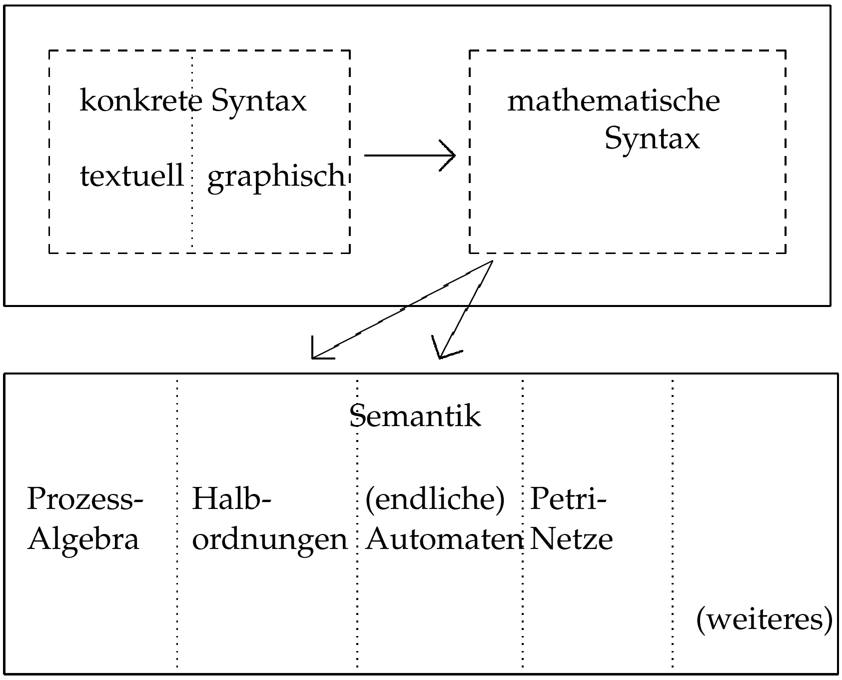
Es gibt noch einige andere Elemente in MSCs, die aber über den Horizont einer motivierenden Einführung hinausragen. Sofern sie durch das benutzte Modell unterstützt werden, werden sie rechtzeitig eingeführt. Andere Elemente werden in Abschnitt 2.d angesprochen.

Als nächstes müssen wir uns überlegen, welches formale Modell wir für die Untersuchung der MSCs heranziehen wollen.

2.a.ii Motivation der Formalisierung

Wie in Abbildung 2.3 angedeutet, gibt es zwei äquivalente Formen der konkreten Syntax. In dieser Arbeit werden wir ausschließlich die graphische Syntax benutzen. Um die Semantik zu erklären, kann man nun verschiedene Wege gehen. In [CEMR98] wird eine Prozeß-Algebra angeben, die fast direkt auf der konkreten textuellen Syntax aufsetzt: Für die (meisten der) Elemente der textuellen Syntax wird eine Übersetzungsfunktion gegeben, die dann gesamt einem MSC einen algebraischen Systemzustand zuordnet, der unter gewissen operationellen Regeln in andere Zustände übergehen kann.

Abbildung 2.3 Syntax und Semantik von MSCs

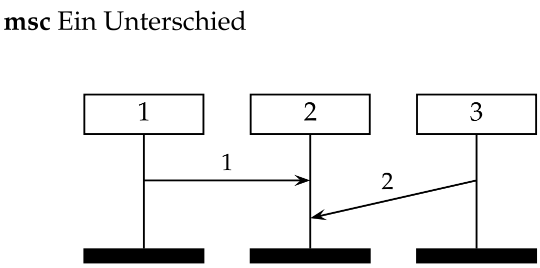


Die meisten Autoren hingegen abstrahieren auf informelle Weise von der konkreten graphischen Syntax, geben also eine mathematische Syntax an, die anhand von konkreten graphischen Beispielen erklärt wird.

Wir werden eine auf Halbordnungen beruhende mathematische Formalisierung benutzen. Das hat den Grund, daß dieser Mechanismus allgemein für die Modellierung verteilter Systeme häufig verwendet wird und auch in der Literatur über MSCs Verwendung gefunden hat (siehe auch [N01]). Es ist daher zu erklären, wie wir aus der graphischen Syntax die Halbordnung erhalten. Das ist jedoch nicht so einfach, wie nun gezeigt werden soll.

Beispiel 2.3 Man betrachte Abbildung 2.4. Die Autoren streiten sich nun darüber,

Abbildung 2.4 Ein MSC mit unklarer Bedeutung (Beispiel 2.3)



ob man in diesem Fall wirklich r_1 vor r_2 erzwingen soll, was ja nach unserer in Abschnitt 2.a.i genannten Interpretation (die der im Standard entspricht) notwendig ist. Offensichtlich gibt es für beide Antworten auf diese Frage gute Gründe:

▷ Davon ausgehend, daß Prozeß 2 keine Chance hat, die Ordnung von r_1 und r_2 zu erzwingen, weil er keinen Einfluß auf die Übertragungsgeschwindigkeit in den jeweiligen Kommunikationsmedien hat und darauf, ob s_1 vor s_2 stattfindet, sollte man keine Ordnung dieser beiden Elemente vorsehen.

▷ Davon ausgehend, daß die Vorteile der visuellen Spezifikation aufgehoben werden, wenn man erlaubt, daß Ereignisse nicht in der visuellen Ordnung stattfinden, könnte man die totale Ordnung der 2-Ereignisse für richtig halten.

Die erste Interpretation, bei der die abgebildete Ordnung als *visuelle Ordnung* bezeichnet wird, die angenommene hingegen als *kausale Ordnung*, setzt voraus, daß der Benutzer eine Vorstellung von der Architektur des Kommunikationsmediums hat. Weiterhin ermöglicht sie es nicht, bestimmte Ereignisfolgen auszuschließen, also für einige Ereignisse eben die visuelle Ordnung zu erzwingen (Falls es sich um einen Use-Case handelt, bei dem eine andere Reihenfolge einen Fehler bedeutet oder eine ganz andere Bedeutung für den Modellierer hat: Beispielsweise ist es in einem Auktionssystem durchaus wichtig, festzustellen, von welchem Prozess ein Gebot zuerst abgegeben wurde.). Im Gegensatz dazu kann man bei der zweiten Interpretation noch weitere MSCs definieren, die die anderen gewünschten Ordnungen abdecken, ist also flexibler und steht nicht im Widerspruch zu den Erwartungen des flüchtigen Benutzers. Das führt aber wiederum zu großen Mengen von MSCs, die nur dafür da sind, die diversen Ordnungen abzudecken. Die Übersichtlichkeit, eines der Hauptmerkmale von MSCs, leidet.

Beide Alternativen werden in der Literatur behandelt.

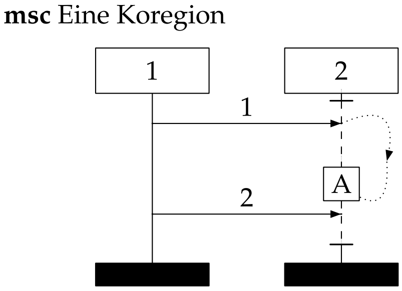
Der Standard definiert zur Auflösung dieses Problems zwei Mechanismen: *Koregionen* und *explizite Ordnungen*. Eine Koregion ist ein Abschnitt einer Prozeßlinie, in dem die Ordnungsrelation aufgehoben wird: Die Elemente dieses Abschnittes sind also ungeordnet, die anderen Elemente bleiben nach ihrer vertikalen Position geordnet. Eine explizite Ordnung hingegen führt eine beliebige Ordnung zwischen beliebigen zwei Elementen ein, sofern damit die Halbordnungseigenschaft nicht verletzt wird. Es ist dabei keine Voraussetzung an die in dieser Ordnung stehenden Elemente nötig, sodaß man auch nicht sinnvoll implementierbare Ordnungen angeben kann, etwa im zweiten MSC von Abbildung 2.2 $A \prec B$: Wie sollte man das in einer Implementierung sicherstellen?[7]

Graphisch wird eine explizite Ordnung dadurch dargestellt, daß zwischen zwei Ereignissen eine gepunktete Linie mit dem Richtungspfeil in der Mitte gezogen wird. Gemeint ist damit (Also: Unsere Übersetzung in die mathematische Syntax und die darauf aufbauende Semantik arbeiten unter dieser Annahme.), daß die Ordnung der Ereignisse sich nun nicht mehr als der transitive Abschluß der normalen Ordnungen ergibt, sondern nun wird zusätzlich die Menge aller solcher Linien herangezogen. Eine Koregion ist ein gestrichelter Teil einer Prozeßlinie.

[7] Daher behandeln wir hier die expliziten Ordnungen auch als Objekte an sich, als Teile der angegebenen Gesamtordnung. In [TDPVG01] wird stattdessen eine explizite Ordnung in eine leere Nachricht übersetzt. Dadurch kann man aber eine unvernünftige Ordnung , wie gerade beispielhaft angegeben, nicht mehr von einer angemessenen unterscheiden, was bei manchen späteren Resultaten wichtig werden wird.

Beispiel 2.4 In Abbildung 2.5 sehen wir sowohl eine Koregion als auch eine explizite Ordnung. Gemeint ist hier, daß die Aktion A auf jeden Fall erst durchgeführt wird, nachdem die Nachricht 1 eingetroffen ist, und daß irgendwann auch Nachricht 2 eintrifft, aber dies in keinem Zusammenhang mit den anderen Ereignissen auf Prozeß 2 steht.

Abbildung 2.5 Ein MSC mit anderen Ordnungen (Beispiel 2.4)



Man kann nun jede beliebige Ordnung dadurch erzeugen, daß man alle Prozesse vollständig auf eine Koregion legt und die gewünschte Ordnung nur durch die explizite Ordnungsrelation angibt. Das ist zwar komplizert, aber sehr allgemein. Darum werde ich für die Beispiele der zweiten obengenannten Interpretation folgen.

Die meisten Autoren betrachten bei ihren interessanten Resultaten nur MSCs mit totalen Ordnungen auf der Prozeßachse ([AHP96, LP97, M01a, MP01] und viele weitere). Ich denke dennoch, daß dem Benutzer die Freiheit gelassen werden sollte.[8]

Zusammengefasst: Wir betrachten nur die mathematische Formalisierung. Die graphische Syntax, die in den Beispielen vorkommen wird, wird im Einklang mit der Definition des Standards [ITU99] interpretiert, wobei Koregionen und explizite Ordnungen unterstützt werden.

2.a.iii Formalisierung

Definition 2.1 (MSC) (vgl. z. B. [MPS98, MP99, HMKT00a, BL01, AEY01])

(a) Ein *(einfaches) Message Sequence Chart (MSC) M* ist ein 7-Tupel

$$\langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell \rangle,$$

wobei gilt:

▹ $\mathcal{P} = \{1, \ldots, n\}$ für ein $n \in \mathbb{N}^{\geqslant 0}$ (eine Menge von *Prozeßnummern* oder *Prozessen*);

[8]Man könnte auch, gerade um Koregionen zu vermeiden, eine Übersetzung eines MSCs in einer erweiterten Syntax in mehrere MSCs mit nur einfachen Merkmalen vornehmen. Somit könnte man aus einem MSC mit einer 2-elementigen Koregion zwei MSCs mit beiden Reihenfolgen machen, und entsprechende Übersetzungen für gewisse Semantiken durchführen. Daß wir hier in einem einzelnen MSC die allgemeineren Merkmale unterstützen, wird erst auf Seite 60 klarwerden.

- ▷ $\mathcal{E} = \mathcal{S} \uplus \mathcal{R} \uplus \mathcal{A}$ mit $|\mathcal{E}| \in \mathbb{N}$ (eine endliche[9] Menge von *Ereignissen*, disjunkt aufgeteilt in die Mengen der *Sende-* und *Empfangsereignisse* sowie *(lokalen) Aktionen*);
- ▷ $o : \mathcal{E} \to \mathcal{P}$ (eine *Zuordnungsfunktion*), wobei $\mathcal{E}_i := \{e \in \mathcal{E} : o(e) = i\}$, $\mathcal{S}_i := \{e \in \mathcal{S} : o(e) = i\}, \mathcal{A}_i := \{e \in \mathcal{A} : o(e) = i\}$, $\mathcal{R}_i := \{e \in \mathcal{R} : o(e) = i\}$ für $i \in \mathcal{P}$;
- ▷ Λ (ein *Beschriftungsalphabet*);
- ▷ $m : \mathcal{S} \to \mathcal{R}$ (eine bijektive *Nachrichtenfunktion*);
- ▷ $\prec \subseteq \mathcal{E} \times \mathcal{E}$ mit $m \subseteq \prec$ (eine Halbordnung, die *Präzedenzordnung*), wobei $\prec_i := \prec \cap (\mathcal{E}_i \times \mathcal{E}_i)$ für $i \in \mathcal{P}$;
- ▷ $\ell : \mathcal{S} \cup \mathcal{A} \to \Lambda$ (eine *Beschriftungsfunktion*).

Wenn im folgenden einfach von einem MSC M die Rede ist, so vereinbaren wir, immer die Bezeichnungen dieser Definition zu unterlegen, sofern nichts anderes gesagt wird; entsprechend ist die Halbordnung eines MSCs M' dann $\prec'$ usw. In Darstellungen benennen wir die Elemente von $\mathcal{P}$ auch manchmal anders. Außerdem bezeichnen wir, wenn es sinnvoll erscheint, z. B. die Ereignismenge oder die Präzedenzordnung eines MSCs M mit $\mathcal{E}(M)$ bzw. $\prec(M)$.

(b) Ist $\prec = (\bigcup_{i=1}^{n} \prec_i \cup m)^+$, so heißt M *sinnvoll geordnet* oder kurz *sinnvoll* (vgl. Fußnote 7).

(c) M besitzt *FIFO-Kanäle*, falls für alle $(e_1, f_1), (e_2, f_2) \in m$ mit $o(e_1, f_1) = o(e_2, f_2)$ gilt: $(e_1 \prec e_2 \wedge f_1 \prec f_2) \vee (e_2 \prec e_1 \wedge f_2 \prec f_1)$.

(d) Eine Nachricht $(s, r) \in m$ mit $o(s) = o(r)$ heißt *Selbstnachricht*.

(e) Für feste $\mathcal{P}$ und Λ ist $\mathfrak{M}(\mathcal{P}, \Lambda)$ die Menge aller MSCs mit Prozeßmenge $\mathcal{P}$ und Nachrichtenalphabet Λ. Wie üblich lassen wir die Parameter ganz oder teilweise weg, wenn klar oder unerheblich ist, was genau gemeint ist.

Aus technischen Gründen wird folgende Definition später nützlich werden.

Definition 2.2 (aktive Prozesse) Für ein MSC M ist die Menge der *in M aktiven Prozesse ($\mathfrak{A}(M)$)* definiert als $\mathfrak{A}(M) := \{p : \exists e \in \mathrm{range}(m) \cup \mathrm{dom}(m).o(e) = p\}$.

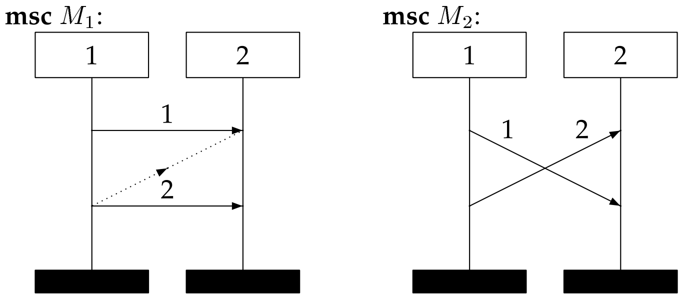
Als aktiv gelten also einfach alle Prozesse, die an Kommunikationen teilnehmen. Internen Aktionen wollen wir hier keine Beachtung schenken. Sinn dessen ist es, die Synchronisationsmöglichkeiten der Prozesse untereinander zu untersuchen, und Prozesse können sich (sinnvollerweise, vgl. Definition 2.1.b) nur über den Austausch von Nachrichten synchronisieren.

Nun kommen wir zur formalen Definition der Semantik von MSCs. Eine Linearisierung eines MSCs ist eine Reihenfolge der Ereignisse. Linearisierungen sind (vgl. Abschnitt 2.a.i) genau das, was uns an einem MSC interessiert: Die Semantik eines MSCs ist die Menge der Linearisierungen seiner Halbordnung. Allerdings interessiert es den Benutzer manchmal nicht, ob nun von zwei Nachrichten gleichen

[9]Man kann auch unendliche Ereignismengen erlauben (vgl. [GMP01, MP01]). Es ist davon auszugehen, daß Spezifikationen und Implementierungen, die noch nicht einmal endlich *repräsentierbar* sind, nur eine geringe Bedeutung haben. Für manche Anwendungen, für die man intuitiv unendliche MSCs gebrauchen könnte, kann man kompositionelle HMSCs verwenden (Abschnitt 5.b).

Inhaltes von einem Prozeß an denselben anderen nun die eine oder die andere zuerst ankommt.[10] Daher benutzen manche Autoren Ereignistypen der Art $\mathsf{S}_{p,q}(\lambda)$ für das Senden der Nachricht λ von p nach q und betrachten die Linearisierungen nur unter dem Aspekt der *Typen* der Ereignisse. Solche Autoren würden also die beiden MSCs aus Abbildung 2.6 als äquivalent betrachten in dem Sinne, daß sie die gleichen Linearisierungen erlauben.[11] Die Halbordnungen sind jeweils schon totale

Abbildung 2.6 Semantische Äquivalenz unter id oder σ



Ordnungen: $\ell(\prec(M_1)) = \{s_1 s_2 r_1 r_2\}$, $\ell(\prec(M_2)) = \{s_1 s_2 r_2 r_1\}$ Um beides zu erlauben, führen wir bei einem MSC eine zusätzliche Abbildung ein, die jedes Ereignis in ein Ereignistypalphabet abbildet. Durch geeignete Wahl dieses Alphabetes können beide Interpretationen behandelt werden, wie man gleich sehen wird.

Definition 2.3 (Ereignistyp) Sei M ein MSC (Definition 2.1).

(1) Ein $\pi : \mathcal{E} \to \Pi$ für ein Alphabet Π heißt *Ereignistypfunktion*. Π heißt *Ereignistypalphabet*.

(2) $\Sigma := \{\mathsf{A}_p(\lambda), \mathsf{S}_{p,q}(\lambda), \mathsf{R}_{p,q}(\lambda) : p, q \in \mathcal{P},\ \lambda \in \Lambda\}$ ist das *kanonische Ereignistypalphabet*. Die Funktion

$$\sigma : \mathcal{E} \to \Sigma,\ e \mapsto \begin{cases} \mathsf{A}_p(\lambda) & e \in \mathcal{A}_p,\ \ell(e) = \lambda \\ \mathsf{S}_{p,q}(\lambda) & e \in \mathcal{S}_p,\ o(m(e)) = q,\ \ell(e) = \lambda \\ \mathsf{R}_{p,q}(\lambda) & e \in \mathcal{R}_q,\ \sigma(m^{-1}(e)) = \mathsf{S}_{p,q}(\lambda) \end{cases}$$

ist die *kanonische Ereignistypfunktion* (vgl. auch Seite 87).

(3) Ist Λ bei den Nachrichten irrelevant, so ersetzt man $\mathsf{S}_{p,q}(\lambda)$ und $\mathsf{R}_{p,q}(\lambda)$ durch $\mathsf{S}_{p,q}$ bzw. $\mathsf{R}_{p,q}$. Ist Λ bei den Aktionen irrelevant, so ersetzt man $\mathsf{A}_p(\lambda)$ durch A_p.

Sofern nichts anderes gesagt wird, werden wir in den Beispielen die kanonische Ereignistypfunktion oder id benutzen.

Beispiel 2.5 Unter σ haben die MSCs aus Abbildung 2.6 jeweils die Linearisierungsmenge $\{\mathsf{S}_{1,2}\mathsf{S}_{1,2}\mathsf{R}_{1,2}\mathsf{R}_{1,2}\}$, obwohl die MSCs selbst nicht isomorph sind.

[10] Sobald man sich auf FIFO-MSCs ohne Koregionen beschränkt, kann das Problem nicht auftreten. Der Standard [CEMR98] erwähnt übrigens ausdrücklich das Überholen von Nachrichten als zulässig.
[11] Übrigens ist das linke MSC ein erstes Beispiel für ein nicht sinnvoll geordnetes MSC.

Die kanonische Ereignistypfunktion ist nun isomorph zur trivialen Funktion id für $\Pi := \mathcal{E}$, falls alle Nachrichten unterscheidbar sind. Das ist zum Beispiel der Fall, wenn wir fortschreitende Zeitstempel einfügen: Es ist egal, welche Werte diese haben, solange nur jeder einzelne Prozeß immer unterschiedliche Stempel an die Nachrichten fügt. Man würde dann das Senden einer Nachricht λ durch das Senden eines Paares (λ, α) mit Stempel α ausdrücken und Λ entsprechend wählen. Damit sind dann alle Nachrichten-Beschriftungstripel paarweise verschieden.

Definition 2.4 (Semantik) Sei π eine Ereignistypfunktion für ein MSC M. Die *Semantik von M unter π* ist dann $[\![M;\pi]\!] := \pi(\mathfrak{l}(\prec))$. Statt $[\![M;\mathrm{id}]\!]$ schreibt man auch nur $[\![M]\!]$.

Um die Terminologie später etwas zu erleichtern, werden nun noch einige Begriffe vorgestellt.

Definition 2.5 (MSC-Wort) Ein *MSC-Wort* ist ein Wort über Σ bei geeigneten $\mathcal{P}$ und Λ. Eine *MSC-Sprache* ist eine Menge von MSC-Wörtern über geeigneten $\mathcal{P}$ und Λ. Ein MSC-Wort heißt *konsistent*, falls es aus der Semantik eines MSCs stammt. Ein MSC-Wort heißt *linksgeschlossen*, falls es Präfix eines konsistenten Wortes ist.

In der Regel wird man einfach ein MSC-Wort hinschreiben und dann $\mathcal{P}$ und Λ stillschweigend voraussetzen als die kleinsten jeweiligen Mengen, für die die Definition erfüllt ist. Daher spezifizieren wir diese Parameter auch nicht weiter. Weiter schreiben wir auch $o(\alpha) = p$ für $\alpha \in \Sigma$, wobei o auf offensichtliche Weise definiert ist. Ist nur die Reihenfolge der Nachrichten interessant, benutzt man eine Notation mit s_i, r_i für Nachrichtenereignisse und A, B und dergleichen für Aktionen.

Beispiel 2.6 $\mathsf{S}_{1,2}\mathsf{R}_{1,2}$ ist konsistent. $\mathsf{S}_{1,2}\mathsf{R}_{1,2}\mathsf{S}_{2,1}$ ist nicht konsistent, aber linksgeschlossen, da es Präfix des konsistenten Wortes $\mathsf{S}_{1,2}\mathsf{R}_{1,2}\mathsf{S}_{2,1}\mathsf{R}_{2,1}$ ist. In der Kurzschreibweise schriebe man für diese drei Worte s_1r_1, $s_1r_1s_2$, $s_1r_1s_2r_2$, wodurch die Konsistenzeigenschaften ebenfalls klar werden. $s_1r_1r_2$ ist nicht linksgeschlossen.

Zwei kleine Hilfsresultate über Halbordnungen werden später noch gebraucht, weswegen sie hier eine Nummer bekommen sollen.

Lemma 2.1 Sei $\prec$ eine Halbordnung über einer Menge A.

(a) $\mathfrak{l}(\prec) \neq \emptyset$.

(b) Sei $B \subseteq A$ nach unten abgeschlossen (also $\forall b \in B. \mathfrak{a}(b) \subseteq B$). Sei $C = A \backslash B$. Seien $w' \in \mathfrak{l}(\prec \cap (B \times B))$ und $w'' \in \mathfrak{l}(\prec \cap (C \times C))$. Dann ist $w'w'' \in \mathfrak{l}(\prec)$.

Beweis:

(a) Das ist intuitiv klar; [S30] ist der nach meinem Wissen erste veröffentlichte Beweis.

(b) Falls $w'w'' \notin \mathfrak{l}(\prec)$, so gibt es $b \in B$ und $c \in C$ mit $c \prec b$, im Widerspruch zur Wahl von B. ⊛

Lemma 2.1.a bedeutet insbesondere, daß die Semantik eines MSCs nie leer ist.

Lemma 2.2 Sei $M \in \mathfrak{M}$. Gebe es keine $(s_1, r_1), (s_2, r_2), \ldots, (s_t, r_t) \in m$, sodaß

$$s_2 \prec r_1,\ s_3 \prec r_2, \ldots,\ s_t \prec r_{t-1}, s_1 \prec r_t.$$

Dann ist M *entschachtelbar*, d. h. es gibt ein $w \in [\![M]\!]$, sodaß für alle $(s, r) \in m$ gilt: Ist $w_i = s \wedge w_j = r$, so ist $w_g \in \mathcal{A}$ für alle $g \in \{i+1, \ldots, j-1\}$. Ist zusätzlich M sinnvoll geordnet und hat keine Selbstnachrichten, so kann man jeweils $j = i + 1$ wählen.

Beweis: Der folgende Algorithmus erzeugt ein Wort mit den gewünschten Eigenschaften:

(1) Setze $w = \varepsilon$.

(2) Ist $\mathcal{E} = \emptyset$, gib w aus.

(3) Solange es ein $\prec$-minimales $a \in \mathcal{A}$ gibt, füge a an w an und entferne a aus M.[12] Ist dann $\mathcal{E} = \emptyset$, so gib w aus.

(4) Wähle ein $r \in \mathcal{R}$, sodaß $\mathfrak{a}(r) \backslash \{m^{-1}(r)\} \subseteq \mathcal{A}$ und erweitere w um eine Linearisierung von $\mathfrak{a}(r)$. Entferne $\mathfrak{a}(r)$ aus M und fahre mit Schritt (2) fort.

Lemma 2.1.b stellt sicher, daß $w \in [\![M]\!]$. Daß w die andere geforderte Eigenschaft hat, ist klar.

Es ist also nur noch zu zeigen, daß Schritt (4) immer ein geeignetes r findet. Dazu spezifizieren wir diesen Schritt näher.

(4a) Sei $r \in \mathcal{R}$ beliebig.

(4b) Falls es ein $s' \in \mathfrak{a}(r) \cap \mathcal{S}$ gibt mit $m(s') \neq r$, so wiederhole Schritt (4b) mit $m(s')$ statt r fort.

(4c) Falls M sinnvoll geordnet ist, ist die Erweiterung von w ein $w'sr$ mit $m(s) = r$; ansonsten ist sie irgendeine Linearisierung von $\mathfrak{a}(r)$.

In Schritt (4a) wird immer ein r gefunden: Schritt (4) wird nur angefangen, falls es ein $e \in \mathcal{E}$ gibt. Ist $\mathcal{E} = \mathcal{A}$, so könnte Schritt (3) ein weiteres $\prec$-minimales $a \in \mathcal{A}$ finden, weil $\prec$ eine Halbordnung ist, würde also noch nicht verlassen. Terminiert die Schleife in Schritt (4b) nicht, so wird (4b) wegen der Endlichkeit von $\mathcal{R}$ mit einem $r \in \mathcal{R}$ mehrmals aufgerufen. Die Aufrufparameter dieser Schleife bilden dann mit ihren jeweiligen Sendungsereignissen einen Widerspruch zur Ordnungsvoraussetzung. In Schritt (4c) sind damit in $\mathfrak{a}(r)$ nur $m^{-1}(r)$ und gewisse $\mathcal{A}$-Elemente enthalten: Weitere $\mathcal{S}$-Elemente sind nicht enthalten, weil ansonsten Schritt (4b) nicht verlassen worden wäre, und wäre ein $\hat{r} \in \mathcal{R} \backslash \{r\}$ enthalten, so auch $m^{-1}(\hat{r}) \in \mathcal{S}$ wegen $m \subseteq \prec$.

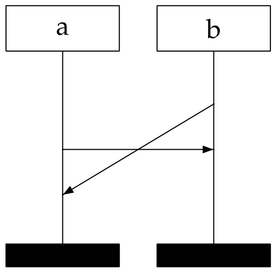
Ist das in Schritt (4c) ausgewählte $w'sr$ keine gültige Linearisierung, so gibt es Ereignisse $s \prec a_1 \prec^* a_b \prec r$ für $a_\nu \in \mathcal{A}$ in $\mathfrak{a}(r)$, was nur für $o(s) = o(a_b) = o(r)$ gelten kann oder eine nicht sinnvolle Ordnungsrelation voraussetzt, da eine transitive Abhängigkeit über m wegen der Zusammensetzung von $\mathfrak{a}(r)$ ausgeschlossen ist. ⊛

[12] a wird also aus $\mathcal{E}$ entfernt, und die nötigen Anpassungen an $\prec$, o und ℓ und gleich an m werden ebenfalls durchgeführt.

Beispiel 2.7

(a) Entschachtelbarkeit heißt, daß zu jedem Zeitpunkt während der Ausführung eines MSCs höchstens eine Nachricht übertragen wird.[13] Man könnte meinen, ein MSC mit FIFO-Kanälen, welches zusätzlich sinnvoll geordnet ist, erlaubte es, die Nachrichten zu entschachteln, d. h. eine Linearisierung zu erhalten, in der zwischen s_i und r_i keine anderen $e \in \mathcal{S} \cup \mathcal{R}$ vertreten sind. Das ist nicht der Fall, wie das MSC in Abbildung 2.7 demonstriert. Das MSC erfüllt aber auch nicht die Bedingung an die Ordnung aus Lemma 2.2: Die Bedingung ist nichts anderes als ein Ausschluß eines ebensolchen Kreuzens mit potentiell mehr Prozessen.

Abbildung 2.7 Ein nicht entschachtelbares, sinnvolles FIFO-MSC (Beispiel 2.7)



(b) Ein Überholen von Nachrichten verbietet offenbar eine Entschachtelung: Überholt Nachricht 1 Nachricht 2, ist also $s_1 \prec s_2$, aber $r_2 \prec r_1$, so gilt $s_1 \prec r_2$ und $s_2 \prec r_1$ wegen $m \subseteq \prec$.

Beispiel 2.8 Man kann leicht einem MSC-Wort (Definition 2.5) ein einzelnes, *kanonisches* ([AEY00]) MSC zuordnen, wenn man totale Ordnung auf den Prozessen und FIFO-Kanäle voraussetzt. Sei gegeben das MSC-Wort $\mathsf{S}_{2,1}\mathsf{R}_{2,1}\mathsf{S}_{2,3}\mathsf{R}_{2,3}$. Das MSC aus Abbildung 2.8.a läßt sich auf natürliche Weise daraus erzeugen: Man schreibe die Ereignisse jedes Prozesses untereinander auf die Linien und verbinde von oben nach unten zueinander passende Ereignisse. Die anderen Linearisierungen dieses MSCs, nämlich $\mathsf{S}_{2,1}\mathsf{S}_{2,3}\mathsf{R}_{2,1}\mathsf{R}_{2,3}$ und $\mathsf{S}_{2,1}\mathsf{S}_{2,3}\mathsf{R}_{2,3}\mathsf{R}_{2,1}$ führen genauso zu diesem MSC.[14] Mit Koregionen dagegen kann $\mathsf{S}_{2,1}\mathsf{R}_{2,1}\mathsf{S}_{2,3}\mathsf{R}_{2,3}$ auch Linearisierung des in Abbildung 2.8.b gegebenen MSCs sein, das aber auch $\mathsf{S}_{2,3}\mathsf{R}_{2,3}\mathsf{S}_{2,1}\mathsf{R}_{2,1}$ erlaubt.

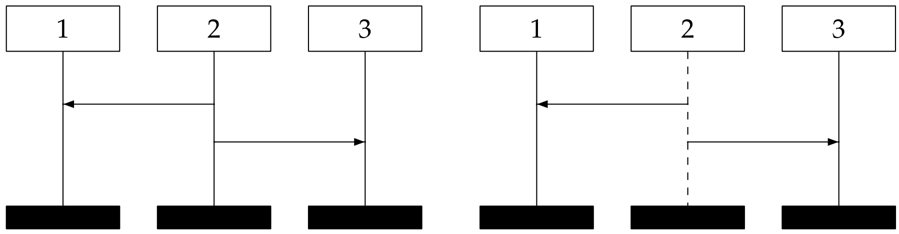
Tatsächlich sind zwei FIFO-MSCs mit Totalordnungen äquivalent, falls sie eine gemeinsame Linearisierung besitzen. Das ist natürlich ein wesentlicher Vorteil dieser Einschränkung. Deswegen wird in vielen Arbeiten genau das gefordert.

Wie erwähnt (Seite 20, Abbildung 2.3), gibt es verschiedene Semantiken für MSCs. Einige, die auch teilweise eine andere abstrakte Syntax verwenden, sollen jetzt kurz vorgestellt werden.

[13] Es liegt, in der Terminologie von [LM02], $\forall^1_{\text{glob}}$-Beschränktheit vor (vgl. Definition 2.14).

[14] Sei zu einem MSC-Wort w das Wort $\alpha(w,p)$ definiert als das Wort aus den Ereignissen w_ν mit $o(w_\nu) = p$, also die Projektion des Wortes auf einen bestimmten Prozeß. Bei allen Linearisierungen eines total geordneten FIFO-MSCs sind dieses Projektionen gleich, d. h. solche MSCs werden durch Angabe von α eindeutig bestimmt: $\forall w', w'' \in [\![M;\sigma]\!].\forall p \in \mathcal{P}.\alpha(w',p) = \alpha(w'',p)$.

Abbildung 2.8 Nicht-isomorphe MSCs ohne disjunkte Semantik



(a) (b)

Der Standard [CEMR98] definiert eine Prozeß-Algebra. Der Zustand eines Systems wird als algebraisches Objekt gegeben, wobei eine operationelle Semantik erklärt, welche Aktionen in diesem Zustand möglich sind und in welchen Zustand das System dann übergeht. Diese Semantik ist in meinen Augen ziemlich überladen, was sicher nicht verhindert werden kann (LADKIN und LEUE meinen in [LL93a], daß eine Prozeß-Algebra-Semantik nicht erstrebenswert sei, da sie zu kompliziert werde.). Daher wurde dieser Zugang hier nicht gewählt.

Auf ähnliche Weise wie in dieser Arbeit wurde kürzlich eine neue Semantik für MSCs von JONSSON und PADILLA in [JP01] gegeben. Diese Semantik betrachtet auch Daten und Variablen und die höheren Konstruktionsmechanismen, die auf Seite 63 vorgestellt werden. Statt mit HMSCs arbeiten die Autoren mit ähnlichen syntaktischen Elementen innerhalb einzelner MSCs. Der Ansatz ist sehr interessant und eine Vereinheitlichung der Untersuchungen des Zeitverhaltens und der Variablensemantik wäre sehr wertvoll.

Eine zu der hier vorgestellten sehr ähnliche, auf *pomsets (partially ordered multisets)* basierende Definition der Semantik von MSCs wird erklärt in [KL98, M01b, TDPVG01].

Die Erklärung der Semantik über Automaten, die wir später erreichen wollen, ist schon recht früh von LADKIN und LEUE für einzelne MSCs in Angriff genommen worden ([LL93a]). Es wird sich zeigen, daß die Erweiterung auf HMSCs allgemein Schwierigkeiten bringt, und die hier vorgestellten Erweiterungen die Automaten noch verkomplizieren.

Andere Semantiken, wie etwa eine über Petri-Netze erklärte ([GRG93]), konnten sich nie durchsetzen.

2.a.iv Schnitte

Zur Vorbereitung auf die Automaten brauchen wir nun noch gewisse Sachen, die ich in Analogie zu [HM00] benenne. Auch hier ist die Nomenklatur in der Literatur uneinheitlich. [LP97] verwendet *slice* für Schnittkanten (Definition 2.7) und *cut* für andere Objekte. Für die Schnittkanten ist bei manchen Autoren (z. B. [BL01, LM02]) der Ausdruck *Konfiguration* gebräuchlich, den ich hier aber nicht verwenden will, weil er später im Zusammenhang mit Automaten anders belegt werden wird. [AY99] benutzt *cut* für Schnittkanten, wohingegen in [M01b, K01] der Ausdruck *Ideal* benutzt wird. Für $|\mathfrak{c}(K)|$ (Definition 2.7.ii) wird in [LM02] das Symbol $\mathrm{gus}(K)$ verwendet (*globally unmatched sends*).

Definition 2.6 (Schnittpunkt) Ein *Schnittpunkt* eines MSCs M ist ein $C_i \subseteq \mathcal{E}_i$ für ein $i \in \mathcal{P}$, sodaß $\forall e \in C_i.\{e' : e' \prec_i e\} \subseteq C_i$. Die Menge aller C_i für ein festes i ist $\mathfrak{C}(M, i)$ (falls M klar ist, auch kurz $\mathfrak{C}_i$).

Ein Schnittpunkt gibt also an, wo wir in einem Prozeß bisher angelangt sind. Erweitern wir dies auf alle Prozesse eines MSCs, erhalten wir die folgende Definition.

Definition 2.7 (Schnittkante) Sei $M \in \mathfrak{M}(\{1, \ldots, n\})$.

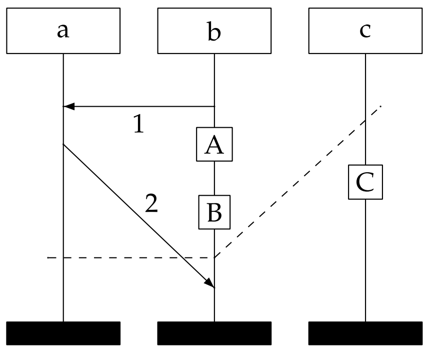
(i) Ein $K \in \mathcal{E}$ mit $\forall e \in K.\mathfrak{a}(e) \subseteq K$ ist eine *Schnittkante*. Die Menge aller Schnittkanten von M wird mit $\mathfrak{K}(M)$ bezeichnet.

(ii) Für ein $K \in \mathfrak{K}(M)$ ist die Menge aller *durch K geschnittenen Nachrichten* definiert als $\mathfrak{c}(K) := m \cap (K \times \mathcal{E} \backslash K)$.[15]

(iii) Zu einem $K \in \mathfrak{K}(M)$ ist die Aufspaltung in die einzelnen Schnittpunkte definiert als $K^\vee := (K \cap \mathcal{E}_1, \ldots, K \cap \mathcal{E}_n) \in \prod_{i=1}^n \mathfrak{C}_i$. Für $L := K^\vee$ setzen wir weiter $L^\wedge := K$.

Beispiel 2.9 Man kann sich eine Schnittkante als eine Linie durch ein MSC vorstellen, die einen bereits ausgeführten Teil von einem noch nicht in Angriff genommenen scheidet. In Abbildung 2.9 ist dies für die Schnittkante $K := \{s_1, r_1, s_2, A, B\}$, die gestrichelt dargestellt wird, aufgezeichnet.[16] Hier ist also $\mathfrak{c}(K) = \{(s_2, r_2)\}$.

Abbildung 2.9 Eine Schnittkante in einem MSC (Beispiel 2.9)



Lemma 2.3 Für ein MSC M ist $\mathfrak{K}(M)$ endlich. Insbesondere ist $|\mathfrak{K}(M)| \leqslant |\mathcal{E}|\,|\mathfrak{l}(\prec)|$.

Beweis: Für jede der $|\mathfrak{l}(\prec)|$ Linearisierungsmöglichkeiten gibt es $|\mathcal{E}|$ Präfixe. ⊛

Die Abschätzung ist in dem Sinne scharf, daß etwa ein MSC mit zwei unabhängigen Aktionen genau 4 Schnittkanten hat, im allgemeinen aber natürlich sehr schwammig, da ein Präfix einer Linearisierung i. d. R. auch eines einer anderen ist. Es ist also nach sinnvollen Abschätzungen zu suchen.

[15] Die andere Schnittrichtung kann wegen der Konsistenzbedingung in (i) nicht auftreten.

[16] Eine direkte Visualisierung einer Schnittkante durch eine entsprechende Linie ist wegen der Koregionen nicht immer möglich. Es soll hier auch nur zur Motivation dienen.

Leider stellen die üblichen Abschätzungen der Anzahl von Linearisierungspräfixen einer Halbordnung Anforderungen, die gerade von MSCs nicht erfüllt werden. So wird in [KR75] gezeigt (siehe auch [BPS96]), daß die allermeisten[17] Halbordnungen 3 Ebenen besitzen, also aus drei nicht-leeren Teilmengen X_1, X_2, X_3 unvergleichbarer Elemente bestehen, sodaß $\forall x_1 \in X_1, x_3 \in X_3 . x_1 < x_3$ gilt und $\neg\exists x_1 \in X_1, x_2 \in X_2, x_3 \in X_3 .(x_1 > x_2 \vee x_2 > x_3)$. Weiter wird gezeigt, daß für die allermeisten solcher Ordnungen auf einem Universum mit n Elementen X_1 und X_3 etwa je $\frac{n}{4}$ Elemente enthalten. In [B94] wird gezeigt, daß die allermeisten Ordnungen mit drei Ebenen $(1 + \mathfrak{o}(1))\eta^2|X_1|!|X_2|!|X_3|!$ Linearisierungen besitzen, wobei η etwa den Wert $3,46$ hat. Zusammen mit der Abschätzung der $|X_i|$ erhalten wir damit eine theoretische asymptotische Abschätzung.

Praktisch kann man davon ausgehen, daß real existierende MSCs eben *nicht* Halbordnungen mit Ebenenzahl 3 repräsentieren: Sobald wir in der Halbordnung eine Kette der Länge v haben, also Ereignisse $e_1 \prec e_2 \prec \ldots \prec e_v$, brauchen wir ja mindestens v Ebenen, da die Elemente einer einzelnen Ebene unvergleichbar sein sollen. Man wird in normalen MSCs sicherlich Ketten einer wesentlich höheren Länge als 3 treffen: Es reicht, wenn das Senden einer Nachricht vom Empfang einer anderen abhängt. Also sind MSCs vom reinen halbordnungstheoretischen Standpunkt her nur eine gewisse Randgruppe.

Auch andere Merkmale, die etwa Bedingungen an die Anzahl unvergleichlicher Elemente stellen ([E92]), erfüllen MSCs meist nicht. Es ist also nach Abschätzungen zu suchen, die die speziellen üblichen Charakteristika von MSCs berücksichtigen.

Zunächst habe ein MSC keine Koregionen, die Ereignisse jeder Prozesslinie seien also total geordnet. Das ist, wie gesagt, ein häufig vorausgesetztes Merkmal, verdient also spezielle Beachtung.

Lemma 2.4 Sei $M \in \mathfrak{M}$. Sei $\prec_p$ eine totale Ordnung auf $\mathcal{E}_p$ für jedes $p \in \mathcal{P}$. Dann ist $\forall p \in \mathcal{P}.|\mathfrak{C}_p| = 1 + |\mathcal{E}_p|$. Somit ist $|\mathfrak{K}| \leqslant \prod_{p\in\mathcal{P}}(1 + |\mathcal{E}_p|)$.

Beweis: Jedes $\prec_p$ hat genau eine Linearisierung. $\mathfrak{C}_p$ besteht also aus genau den Präfixen dieser Linearisierung. Da ein Wort der Länge z genau $z + 1$ verschiedene Präfixe hat, ergibt sich die Abschätzung für $\mathfrak{C}_p$.

Da jede Schnittkante K genau ein $|\mathcal{P}|$-Tupel von Schnittpunkten $K^\vee$ bestimmt und jedes $|\mathcal{P}|$-Tupel von Schnittpunkten J höchstens eine Schnittkante $J^\frown$ bestimmt, folgt die zweite Abschätzung. ⊛

In der Regel umfaßt $\mathfrak{K}$ weniger Elemente als einfach durch die Schnittpunkte herauskommen, da z. B. ein Schnittpunkt mit einem r_i nur dann in einem Schnitt auftreten kann, wenn s_i im Schnittpunkt des sendenden Prozesses enthalten ist. Im Extremfall, bei keiner Synchronisation, gilt jedoch Gleichheit, daher ist diese Schranke nicht verbesserbar.

Im übrigen ist festzustellen, daß bei etwa gleichvielen Ereignissen pro Prozess gilt:

$$\prod_{p\in\mathcal{P}}(1 + |\mathcal{E}_p|) \approx \prod_{p\in\mathcal{P}}(1 + \tfrac{|\mathcal{E}|}{|\mathcal{P}|}) \in \mathfrak{O}(|\mathcal{E}|^{|\mathcal{P}|})$$

mit eher kleinem $|\mathcal{P}|$.

[17] In dieser Diskussion soll das heißen, daß die Wahrscheinlichkeit, daß eine Halbordnung diese Eigenschaft besitzt, mit Zunahme ihrer Größe gegen 1 strebt. Der in den Originalen benutzte Begriff ist *almost all*.

Nun betrachten wir den Fall, daß ein MSCs sich in gewisse Ebenen aufteilen läßt. Ein Ebenenübergang sei dabei ein Punkt vollständiger Synchronisation: Die nächste Ebene darf erst betreten werden, sobald die vorhergehende vollständig abgearbeitet wurde.

Lemma 2.5 Sei $M \in \mathfrak{M}$. Lasse sich $\mathcal{E}$ partitionieren durch $\mathcal{E} = E_1 \uplus \ldots \uplus E_r$ mit $\forall i, j.(x \in E_i \wedge y \in E_j \wedge i < j \rightarrow x \prec y)$. Für jede Ebene E_z sei l_z die Anzahl ihrer Linearisierungen und sei c_z die Anzahl der Präfixe dieser Linearisierungen. Dann hat das MSC gesamt $\prod_{z=1}^{r} l_z$ Linearisierungen mit $1 + \sum_{z=1}^{r}(c_z - 1)$ Präfixen, also

$$|\mathfrak{K}| \leqslant (1 - r) + \sum_{z=1}^{r} c_z.$$

Beweis: Seien $W_z = \ell(\prec \cap (E_z \times E_z))$ für alle $z \in \{1, \ldots, r\}$. Dann ist genau dann $w \in [\![M]\!]$, falls $w = w^{(1)} \ldots w^{(r)}$ für gewisse $w^{(z)} \in W_z$ (vgl. Lemma 2.1). Die Anzahl der Linearisierungen folgt damit. Für die Abschätzung von $|\mathfrak{K}|$ ist zu beachten, daß ε ein Präfix eines jeden $w \in W_j$ ist, womit $w^{(1)} \ldots w^{(z)}$ und $w^{(1)} \ldots w^{(z)}\varepsilon$ nicht doppelt gezählt werden dürfen (daher die -1 in den Summanden). ⊛

Da in Lemma 2.5 keine weiteren Bedingungen an das MSC gegeben wurde, kann die Anzahl der Linearisierung pro Ebenen wieder sehr hoch sein. In jedem Fall ist $c_z \leqslant 2^{|E_z|}$, was gesamt die Schranke $|\mathfrak{K}| \leqslant \sum_{z=1}^{r} 2^{|E_z|}$ ergibt. Auch diese Schranke ist scharf: Schlechtestenfalls gilt tatsächlich Gleichheit. Analog zu Lemma 2.4 kann man totale Ordnungen in gewissen Ebenen fordern, um bessere Abschätzungen zu erhalten.

Jetzt lassen wir ganz allgemeine MSCs zu, mit der maximalen Größe von Koregionen als Parameter. Es gebe also keine Menge wenig geordneter Elemente auf einem Prozeß, die mehr Elemente umfaßt als durch ein gewisses z vorgegeben: Es kann auf jede Prozesslinie ein Gitter gelegt werden, dessen Gitterpunkte total geordnet sind und die nicht weiter als z auseinanderliegen. Das ist im Endeffekt eine Übertragung von Lemma 2.5 auf die einzelnen Prozesse.

Satz 2.1 Sei $M \in \mathfrak{M}$. Sei $z \in \mathbb{N}$. Für jedes $p \in \mathcal{P}$ gebe es Punkte $e_1 \prec \ldots \prec e_g \in \mathcal{E}_p$, sodaß $|E_1|, \ldots, |E_g| \leqslant z$ für $E_i := \{e \in \mathcal{E}_p : e_i \prec e \preceq e_{i+1}\}$ (für $1 < i < g$), $E_1 = \{e \in \mathcal{E}_p : e \preceq e_1\}$ und $E_g := \{e \in \mathcal{E}_p : e \succ e_g\}$ mit $\mathcal{E} = \bigcup_{i=1}^{g} E_i$. Dann ist

$$|\mathfrak{K}| \in \mathfrak{O}((|\mathcal{E}|2^z)^{|\mathcal{P}|}).$$

Beweis: Sei $p \in \mathcal{P}$ beliebig. Seien $e_1, \ldots, e_g$ und $E_1, \ldots, E_g$ die angegeben Ereignisse bzw. Ereignismengen. Wie im Beweis zu Lemma 2.5 sieht man, daß eine Linearisierung der Menge $\mathcal{E}_p$ aus Linearisierungen der Mengen $E_1, \ldots, E_g$ zusammengesetzt ist, also

$$|\mathfrak{C}_p| \leqslant 1 + \sum_{i=1}^{g}(2^{|E_i|} - 1) \leqslant g2^z.$$

Für $\mathfrak{K}$ beachte man, daß g für die einzelnen Prozesse verschieden sein mag. Für eine ganz genaue Abschätzung mag man z für jeden Prozeß setzen und g in Abhängigkeit dessen wählen, aber allgemein ist $g \leqslant |\mathcal{E}_p|$. Sei $f = \max_{p \in \mathcal{P}} |\mathcal{E}_p|$. Dann ist (Darstellung von $K \in \mathfrak{K}$ als $K^{\vee}$, vgl. Beweis zu Lemma 2.4)

$$|\mathfrak{K}| \leqslant \prod_{p \in \mathcal{P}} |\mathfrak{C}_p| \leqslant \prod_{p \in \mathcal{P}} f2^z \leqslant (f2^z)^{|\mathcal{P}|},$$

wobei $|\mathcal{P}|$ und z meist klein sind. ⊛

Damit ergibt sich eine akzeptable Komplexität auch bei Zulässigkeit von Koregionen.

2.a.v Konkatenation von MSCs

Wir wollen nun zwei MSCs hintereinander ausführen. Die MSCs M_1 und M_2 in

Abbildung 2.10 Zwei aneinanderzufügende MSCs

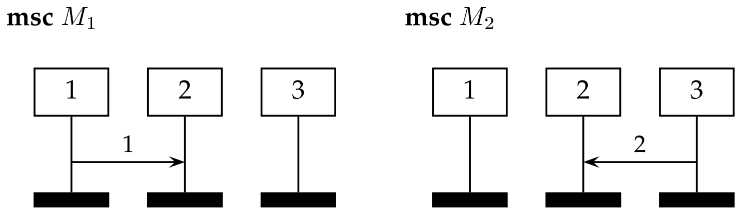


Abbildung 2.10 wolle man aneinanderfügen, d. h. es soll ein einzelnes MSC entstehen, welches die Linearisierungen erlaubt, die man erhielte, führte man zunächst M_1 und dann M_2 aus. Doch so klar dies erscheinen mag, steckt der Teufel wieder im Detail: Soll im entstehenden MSC $s_1 \prec s_2$ gelten oder nicht? Allgemein sind zwei Interpretationen der MSC-Konkatenation gebräuchlich:

(1) *Synchrone Konkatenation:* Das erste MSC muß vollständig abgearbeitet sein, bevor ein Prozeß zum zweiten fortschreitet.

(2) *Asynchrone Konkatenation:* Jeder einzelne Prozeß muß seinen Anteil am ersten MSC abarbeiten bevor er in den zweiten gehen darf.

Bei synchroner Konkatenation haben wir also $s_1 \prec s_2$, bei asynchroner nicht. Der Standard definiert einen Kompositionsoperator für die asynchrone Konkatenation und einen für die parallele Komposition (bei der die MSCs eben nicht nacheinander sondern nebeneinander ausgeführt werden; siehe auch Bemerkung nach Definition 2.9). Das ist insofern sinnvoll, als eine synchrone Konkatenation eine Synchronisation der verteilten Prozesse erfordert (wie es ihr Name schon andeutet). Das müßte also durch zusätzliche, nicht in den einzelnen MSCs dargestellte Nachrichten erreicht werden, was dem Gedanken der Vollständigkeit der Darstellung, der bei MSCs ja im Vordergrund steht, widerspräche (vgl. Diskussion auf Seite 19).

Auch bei asynchroner Konkatenation haben wir aber das gleiche Problem wie es auf Seite 22 angesprochen wurde: Warum muß unbedingt $r_1 \prec r_2$ gelten? Wir erlauben daher, *irgendeine* zusätzliche Ordnungsrelation anzugeben, mit der man die asynchrone Konkatenation, die synchrone und auch beliebige andere simulieren kann.

Definition 2.8 (Konkatenation)

(a) Seien $M', M'' \in \mathfrak{M}(\mathcal{P})$. Sei $k \subseteq \mathcal{E}' \times \mathcal{E}''$. Dann ist die *Konkatenation* oder *sequentielle Komposition von M' mit M'' unter k, $M' \circ^k M''$*, definiert als das MSC $M \in \mathfrak{M}(\mathcal{P})$ mit den folgenden Parametern:

▷ $\mathcal{E} := \mathcal{E}' \uplus \mathcal{E}''$, $\Lambda := \Lambda' \cup \Lambda''$,

▷ $o := o' \vee o''$, $\ell := \ell' \vee \ell''$, $m := m' \vee m''$,

▷ $\prec := (\prec' \cup \prec'' \cup k)^+$.

Ist $k \subseteq \bigcup_{i=1}^{n}(\mathcal{E}'_i \times \mathcal{E}''_i)$, so heißt die Konkatenation *sinnvoll* (vgl. Definition 2.1).

(b) $\circ^k$ ist linksassoziativ.

Man beachte, daß wir hier eine disjunkte Vereinigung der Ereignisse haben, und daß die Definition der weiteren Objekte sinngemäß im Sinne der Identifizierung wie angesprochen in der Vorbemerkung ist: Wir schreiben also $e \sim e'$ für alle e, e', die ursprünglich aus derselben Menge stammen. Somit sind auch Aussagen wie „$e \prec e$ in $M \circ^k M$" kein Problem: Gemeint ist, daß $e \prec e'$ für ein $e' \sim e$ mit $e \neq e'$, wobei $e' \not\prec e$ folgt.

Die explizite Assoziativität ist nötig, da k ja nur als Teilmenge der Ereignisse der rechts und links liegenden MSCs erklärt ist.

Die Einschränkung, daß beide MSCs die gleichen Prozesse besitzen müssen, ist keine echte. In Anwendungen können separate MSCs mit als verschieden gedachten Prozessen leicht durch Auffüllen mit nichts machenden Prozessen auf eine gemeinsame Prozeßmenge gebracht werden. Die Übersetzung in unsere mathematische Syntax ist also leicht durchführbar.

Man beachte, daß wir tatsächlich eine Halbordnungsrelation erhalten, egal wie k gewählt wird.

Durch das k in Definition 2.8 können wir das Konzept der Koregionen auf Kompositionen übertragen – Die Bildung der transitiven Hülle ist nötig, um die Transitivitätsbedingung der Halbordnungsdefinition einzuhalten. Unser k ähnelt der Abhängigkeitsrelation D, die in [TDPVG01] für die dortige Semantik mit pomsets eingeführt wird. Eine Hilfsfunktion der offiziellen Semantik in [CEMR98] benötigt ähnliche Parameter, ist dem MSC-Modellierer selbst jedoch nicht zugänglich.

Zur Vereinfachung seien für die üblichen Wahlen von k folgende Abkürzungen eingeführt.

Definition 2.9 (übliche Konkatenationen) Seien $M', M'' \in \mathfrak{M}(\mathcal{P})$.

(i) Die *synchrone Konkatenation von M' und M''*, $M' \oplus M''$, ist definiert als $M' \circ^k M''$ mit $k := \mathcal{E}' \times \mathcal{E}''$.

(ii) Die *(normale) asynchrone Konkatenation von M' und M''*, $M' \otimes M''$, ist definiert als $M' \circ^k M''$ mit $k := \bigcup_{i \in \mathcal{P}}(\mathcal{E}'_i \times \mathcal{E}''_i)$.

(iii) Die *gemäßigt asynchrone Konkatenation von M' und M''*, $M' \varovee M''$, ist definiert als $M' \circ^k M''$ mit $k := \bigcup_{i \in \mathcal{P}}(\mathcal{E}'_i \times (\mathcal{A}''_i \cup \mathcal{S}''_i))$.

(iv) Die *gleichgültige* (oder: *parallele*) *Konkatenation von M' und M''*, $M' \circledcirc M''$, ist definiert als $M' \circ^k M''$ mit $k := \emptyset$.

Die gleichgültige Konkatenation entspricht der parallelen Komposition des Standards [CEMR98].

In Beispielen werden wir, sofern nichts anderes gesagt wird, stillschweigend gemäßigt asynchrone Konkatenation benutzen. Dies erscheint natürlich, da wir so nicht erwarten, daß in zwei aufeinanderfolgenden MSCs an einen Prozeß gesandte Nachrichten in genau der Reihenfolge ankommen (gleiche Argumentation wie auf Seite 22).

Synchrone Konkatenation ist natürlich sehr schön. Das folgende kleine Lemma mag als Beispiel dienen.

Lemma 2.6 Sind $M', M'' \in \mathfrak{M}$, so ist $|\mathfrak{K}(M' \oplus M'')| = |\mathfrak{K}(M')| + |\mathfrak{K}(M'')| - 1$.

Beweis: klar. ⊛

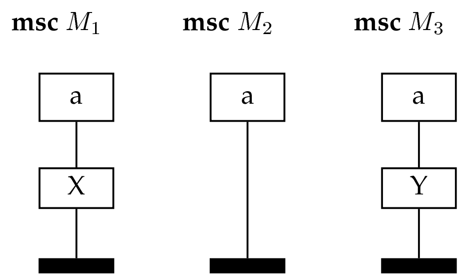
2.b High-Level Message Sequence Charts

2.b.i Formalisierung

In High-Level Message Sequence Charts geben wir strukturierte mögliche Abarbeitungsreihenfolgen einer Anzahl von MSCs an. Dafür wird eine Menge von MSCs in einem Graphen zusammengefasst, wobei die Kantenrelation bedeutet, daß ein System, welches sich innerhalb eines MSCs befindet, unter gewissen Umständen in ein folgendes eintreten kann.

Man beachte, daß unser alter Kompositionsoperator $\circ^k$ nicht hinreichend ist, um in der Realität vorkommende Spezifikationen auszudrücken, da wir auch Ereignisse in MSCs miteinander in Relation setzen wollen, die nicht direkt durch eine Erreichbarkeitsrelation miteinander verbunden sind, aber transitiv. Mit den MSCs aus Abbildung 2.11 könnte man $X \prec Y$ in $M_1 \circ^{k_1} M_2 \circ^{k_2} M_3$ fordern wollen. Dies geht, da $X \in \mathcal{E}(M_1 \circ^{\emptyset} M_2)$ ($k_1 = \emptyset$, da $\mathcal{E}(M_2) = \emptyset$) und somit $k_2 = \{(X, Y)\}$ möglich ist ($\circ$ ist linksassoziativ, Definition 2.8.b). In einem HMSC mit Schleifen haben

Abbildung 2.11 Motivation einer anderen Komposition



wir aber beliebig lange Pfade, aus denen *beliebig große* MSCs entstehen können. Um solche Modelle dennoch endlich repräsentieren zu können, definieren wir eine Relation k, die für die Paare von MSCs definiert ist, die irgendwie hintereinander in einem Lauf auftreten können. Dann sind bei der Konkatenation alle Ereignisse der MSCs in der k-Relation, also ihrem Parameter, die $\sim$-Repräsentanten in einer Inter-MSC-k-Relation haben. Dies wird bei der formalen Definition der Semantik in Definition 2.13 klarer werden.

Die Formalisierung von HMSCs ist in der Literatur nicht einheitlich, auch nicht bei den Autoren, die für MSCs ähnliche Formalisierungen gewählt haben. Definition 2.10 steht der aus [HMKT00a] nahe. Als andere Zugänge, die aber im Endeffekt auf das gleiche herauslaufen, seien hier repräsentativ [MPS98, M01b, GM02] genannt.

Definition 2.10 (HMSC) Ein *High-Level Message Sequence Chart (HMSC)* ist ein 8-Tupel

$$H := (N, \mathcal{P}, i_0, N_*, \Lambda, (M_i)_{i=1}^{|N|}, E, k),$$

wobei gilt:

- ▷ $N = \{1, \ldots, |N|\}$ für ein $|N| \in \mathbb{N}^{>0}$ (eine Menge von *Knotennummern* oder *Knoten*);
- ▷ $\mathcal{P}$, Λ wie in Definition 2.1;
- ▷ $i_0 \in N$ (ein *Startknoten*);
- ▷ $N_* \subseteq N$ (eine Menge von *Endknoten*);
- ▷ $M_i \in \mathfrak{M}(\mathcal{P}, \Lambda)$ (eine Menge von *Knoteninhalten*);[18]
- ▷ $E \subseteq N \times N$ (eine *Kantenrelation*), sodaß jedes M_i auf einem Pfad von M_{i_0} zu einem M_j mit $j \in N_*$ liegt;
- ▷ $k : E^+ \to \bigcup_{i=1}^m \mathcal{E}(M_i) \times \bigcup_{i=1}^m \mathcal{E}(M_i)$ mit $k(i,j) \subseteq \mathcal{E}(M_i) \times \mathcal{E}(M_j)$ (eine *Inter-MSC-Präzedenzrelation*).

Ist jedes $k(i,j)$ sinnvoll (Definition 2.8.a), so heißt H *sinnvoll geordnet*.

Wenn im folgenden einfach von einem HMSC H die Rede ist, so vereinbaren wir, immer die Bezeichnungen dieser Definition zu unterlegen, sofern nichts anderes gesagt wird. Entsprechend ist die Kantenrelation eines HMSCs H' dann E' usw.

Für feste $\mathcal{P}$ und Λ ist $\mathfrak{H}(\mathcal{P}, \Lambda)$ die Menge aller HMSCs mit Prozeßmenge $\mathcal{P}$ und Nachrichtenalphabet Λ. Wie üblich lassen wir die Parameter ganz oder teilweise weg, wenn klar oder unerheblich ist, was genau gemeint ist. Sollte es aus didaktischen Gründen sinnvoll erscheinen, wird statt $\{1, \ldots, |N|\}$ eine andere Indizierung der MSCs verwendet.

Wir setzen ferner $\mathcal{E}(H) := \bigcup_{i=1}^{|N|} \mathcal{E}(M_i)$, $\mathcal{S}(H) = \bigcup_{i=1}^{|N|} \mathcal{S}(M_i)$ etc.

Ein HMSC ist also eine im wesentlichen sequentielle Komposition von parallelen Einzelspezifikationen.[19] Das schaut ähnlich aus wie ein endlicher Automat. In der Tat kann man dies auch so formalisieren ([M01a, MM01]). Obwohl wir also einen anscheinend regulären Beschreibungsmechanismus haben, der reguläre Objekte verknüpft, ist die resultierende Sprache (die Menge der Linearisierungen) nicht unbedingt regulär: Das liegt daran, daß die Konkatenation im allgemeinen eben nicht regulär ist, sondern z. B. asynchron (Beispiel 2.14).

Zur Vereinfachung nehmen wir an, daß in jedem MSC ein Ereignis vorhanden ist. Es gibt also keine völlig leeren MSCs.

[18] Allgemeiner kann man auch in den Knoten HMSCs zulassen, indem man induktiv MSCs als HMSCs der Stufe 0 betrachtet und für HMSCs der Stufe n HMSCs der Stufen $0, \ldots, n-1$ als Knoteninhalte zuläßt. Praktisch kann man jedoch durch Flachklopfen ein in gewissem Sinne äquivalentes HMSC immer in unserer Form erhalten – Das wird offenbar werden, sobald die Semantik formal definiert ist.

[19] Kommunizierende Automaten sind im Gegensatz dazu parallele Kompositionen sequentieller Einzelspezifikationen.

Wie in Definition 2.9 werden wir auch für HMSCs die üblichen Konkatenationsoperatoren mit Symbolen bezeichnen.[20]

Definition 2.11 (übliche Konkatenationen) Für restliche Parameter wie in Definition 2.10 definieren wir:

(i) $k_\oplus := \lambda(i,j) \in E^+.(\mathcal{E}(M_i) \times \mathcal{E}(M_j))$.

(ii) $k_\otimes := \lambda(i,j) \in E^+.\bigcup_{p\in\mathcal{P}}((\mathcal{E}(M_i))_p \times (\mathcal{E}(M_j))_p)$.

(iii) $k_\oslash := \lambda(i,j) \in E^+.\bigcup_{p\in\mathcal{P}}((\mathcal{E}(M_i))_p \times ((\mathcal{A}(M_i))_p \cup (\mathcal{S}(M_j))_p))$.

(iv) $k_\circledcirc := \lambda(i,j) \in E^+.\emptyset$.

Wir wollen nun durch ein HMSC laufen wie durch einen beliebigen Graphen. Was dabei herauskommt, ist ein MSC, welches der Konkatenation der einzelnen MSCs entspricht. Dabei sind die transitiven Abhängigkeiten zu berücksichtigen, wie auf Seite 35 erläutert.

Definition 2.12 (Lauf) Sei $H \in \mathfrak{H}$. Ein *Lauf (durch H) von l_1 nach l_r* ist eine Folge $l := l_1 l_2 \ldots l_r \in N^+$ mit $(l_s, l_{s+1}) \in E$ für alle $s \in \{1, \ldots, r-1\}$. Gilt $i_0 = l_1$ und $l_r \in N_*$, so heißt l auch *Endlauf*. Die Menge aller Läufe durch H von i nach j ist $\mathfrak{L}(H,i,j)$, die aller Läufe $\mathfrak{L}(H)$, die aller Endläufe $\mathfrak{L}_*(H)$.

Man beachte, daß die Läufe zwar beliebig lang sein können, aber immer eine endliche Länge haben. Somit ist das, was entstehen soll, tatsächlich ein MSC, aber sofern das HMSC eine Schleife enthält, können MSCs mit beliebig vielen Ereignissen als Semantik von Läufen des HMSCs auftreten. Die gegebene Semantik ist nichts weiter als eine Anwendung einer Formalisierung der informell auf Seite 35 gegebenen Bedeutung von k.

Definition 2.13 (Semantik) Sei $H \in \mathfrak{H}$.

(a) Sei $l' \in \mathfrak{L}(H)$. Seien $1 \leqslant g_1 < g_2 < \ldots < g_r \leqslant |l'|$. Sei $l = l_1 \ldots l_r := l'_{g_1} \ldots l'_{g_r}$ mit $r > 0$.[21] Dann ist das *durch l definierte MSC* $\mathfrak{m}(l)$ definiert als

$$\mathfrak{m}(l) := M_{l_1} \circ^{\alpha(l_1,l_2)} M_{l_2} \circ^{\alpha(l_1 l_2,l_3)} \ldots \circ^{\alpha(l_1\ldots l_{r-1},l_r)} M_{l_r},$$

wobei $\alpha(l_1 \ldots l_s, l_{s+1})$ für $1 \leqslant s < r$ definiert ist wie folgt:

Seien $e \in \mathcal{E}(M_{l_1} \circ^{\alpha(l_1,l_2)} M_{l_2} \ldots \circ^{\alpha(l_1\ldots l_{s-1},l_s)} M_{l_s})$ und $f \in \mathcal{E}(M_{l_{s+1}})$ beliebig. Genau dann ist $(e,f) \in \alpha(l_1 \ldots l_s, l_{s+1})$, falls $e \sim e'$ für ein $e' \in \mathcal{E}(M_{l_\rho})$ $(\rho \in \{1,\ldots,s\})$ mit $(e',f) \in k(l_\rho, l_{s+1})$.

(b) Sei l wie in (a). Die *Semantik von l unter π* ist $[\![l;\pi]\!] := [\![\mathfrak{m}(l);\pi]\!]$. Statt $[\![l;\mathrm{id}]\!]$ schreibt man auch einfach $[\![l]\!]$.

(c) Die *Semantik von H unter π* ist definiert als $[\![H;\pi]\!] := \{[\![l;\pi]\!] : l \in \mathfrak{L}_*(H)\}$. Statt $[\![H;\mathrm{id}]\!]$ schreibt man auch einfach $[\![H]\!]$.

[20] Wenn man unterstellt, daß mit $\circ$ der Parameter k schon feststeht, kann man ein HMSC als rationalen Ausdruck über den MSCs ansehen ([M01b]). Da zur Definition der Semantik k aber bekannt sein muß, wäre die Definition der Semantik in der hier vorgestellten Allgemeinheit ebenfalls recht kompliziert.

[21] $l \in N^+$ entsteht also aus einem Lauf durch Wegstreichen von Knoten. Das wird aus technischen Gründen später wichtig werden. In Moment kann man sich die Definition einfach als nur für $l \in \mathfrak{L}(H)$ gültig vorstellen.

Wie man nun sieht, könnte man in der Semantik einen rekursiven Aufruf erlauben, indem die $\circ$-Argumente wieder über $[\![\cdot]\!]$ bestimmt werden. Somit könnte man größere Hierarchiestufen in HMSCs erlauben (vgl. Fußnote 18).

2.b.ii Wichtige Eigenschaften

Wir kommen nun zu diversen in der Literatur gebräuchlichen Eigenschaften, untersuchen deren Zusammenhänge in unserer Notation und Formalisierung und überlegen, welche Kriterien uns unserem Ziel, HMSCs in endliche Automaten zu übersetzen, näherbringen. Die Nomenklatur kann dabei nicht gleichzeitig mit allen bisherigen Veröffentlichungen übereinstimmen. Es wird daher darauf hingewiesen, welchen bisherigen Benennungen wir folgen und welche Mißverständnisse auftreten könnten.

Ein HMSC wird für unsere Analysen zugänglich sein, falls es nicht möglich ist, daß manche Prozesse andere überfahren.[22] Dazu stellen wir zunächst diverse in der Literatur vorgestellte Kriterien vor und erklären dann, warum diese mit unserer Verallgemeinerung nicht hinreichend sind. Wir beginnen mit einer beliebten Eigenschaft, welche zum Beispiel in [HMKT99] eine tragende Rolle spielt, aber auch in vielen anderen Veröffentlichungen unter diesem Namen bekannt ist.

Definition 2.14 (Beschränktheit) Sei $H \in \mathfrak{H}$. H heißt *b-beschränkt* für ein $b \in \mathbb{N}$, falls es keinen Lauf $l \in \mathfrak{L}(H)$ und keine Nachricht $(e, e') \in m(M_i)$ für ein MSC M_i gibt, sodaß $|\{(f, f') \in \mathfrak{c}(K) : f \sim e \wedge f' \sim e'\}| > b$ für ein $K \in \mathfrak{K}(\mathfrak{m}(l))$. Ist b egal, so heißt H auch einfach *beschränkt*. Ist H nicht beschränkt, so wird H *unbeschränkt* genannt.

Beschränktheit in diesem Sinne ist also vergleichbar mit der *Divergenz-Freiheit* im Sinne von [BAL97b]: Die Puffer aller Kanäle zwischen zwei Prozessen müssen beschränkt sein und dennoch alle Linearisierungen erlauben können. Wir werden unter Divergenz etwas anderes verstehen (Definition 2.16). Da die Schranke in Definition 2.14 nicht für das gesamte System gilt sondern nur für einen einzelnen Kanal, aber für alle Pfade gelten soll, würde b-Beschränktheit etwa zur $\forall^b_{\mathrm{loc}}$-Beschränktheit aus [LM02] äquivalent sein, aber dort betrachten die Autoren komplette Kanäle statt einzelner Nachrichten, weil die Nachrichten keine Unterscheidbarkeit besitzen. Konzeptionell sind die beiden Begriffe jedoch im wesentlichen gleichwertig. In [EMR97a] werden feinere Unterscheidungen der Kapazitätenbeschränkungen untersucht, die zur genaueren Bestimmung der bei der Implementierung nötigen Kommunikationskanäle hilfreich sind, aber für die theoretische Untersuchung in den folgenden Abschnitten nicht wichtig sind.

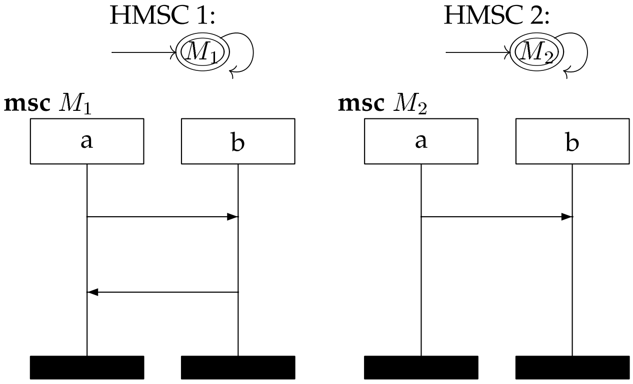
(b-)Beschränktheit ist nicht nur ein theoretisch wichtiges Kriterium, sondern auch praktisch nützlich: Kanalkapazität in verteilten Systemen ist ebenso eine Ressource wie Speicherplatz oder Rechenzeit auf einzelnen Maschinen. Der Bedarf an Kapazität ist daher möglichst zu minimieren, was bedeutet, daß HMSCs b-beschränkt mit kleinem b sein sollten.

Beispiel 2.10 Offensichtlich ist HMSC 1 in Abbildung 2.12 1-beschränkt, HMSC 2 hingegen unbeschränkt.[23] Auch bei sinnvollen Modellierungen können beschränk-

[22] Es handelt sich also um eine Art Fairness-Kriterium: Es ist nicht möglich, daß eine Menge von Prozessen ständig miteinander kommuniziert, ohne daß ein anderer Prozeß regelmäßig auch eine Aktion durchführen darf.

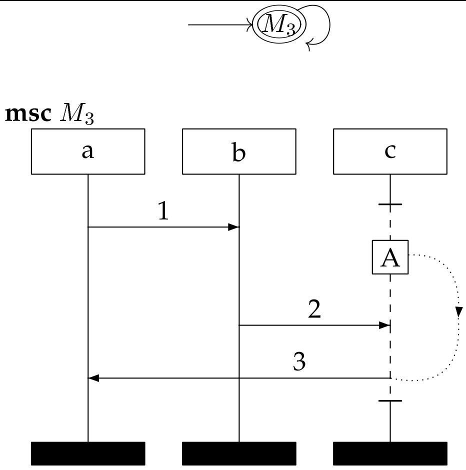
[23] Zur Erinnerung: Wenn nichts anderes gesagt wird, ist die Konkatenation gemäßigt asynchron.

Abbildung 2.12 (Un)beschränkte HMSCs (Beispiel 2.10)



te, aber nicht 1-beschränkte HMSCs auftreten. So ist das HMSC in Abbildung 2.13 nicht 1-beschränkt, da die Linearisierung $s_1 A s_3 r_3 s_1' | r_1 r_1' s_2 s_2' r_2 r_2' A' s_3' r_3'$ an der gekennzeichneten Stelle (Ereignisse des zweiten Durchlaufes durch M_2 sind zur Verdeutlichung strichliert.) zweimal Nachricht 1 schneidet. Daß das HMSC dennoch beschränkt ist (und zwar 2-beschränkt), kann man sich denken, einen formalen Beweis für einen allgemeineren Fall bringt Lemma 2.9.a.

Abbildung 2.13 Ein 2-beschränktes, nicht 1-beschränktes HMSC (Beispiel 2.10)



Es liegt offenbar, daß Beschränktheit zur Implementierung in einem System (Prozesse und Kommunikationssystem) mit endlich vielen Zuständen nötig ist.

Weiterhin recht beliebt ist die Methode, die möglichen Kommunikationspfade in einem Graphen zu sammeln und diesen auf Zusammenhang zu untersuchen.

Definition 2.15 (Graphen)

(a) Sei $M \in \mathfrak{M}$. Der *Kommunikationsgraph von M ($\mathscr{J}(M)$)* ist $(\mathfrak{A}(M), o(m))$ (Definition 2.2) (also $\mathscr{J}(M) = (\mathfrak{A}(M), \to)$ mit $\to = \{(p, q) : \exists e \in \mathcal{S}_p, e' \in \mathcal{R}_q . m(e) = e'\}$).

(b) Sei $H \in \mathfrak{H}$. Seien $l := l_1 \dots l_n l_1$ eine einfache Schleife in H und $M = \mathfrak{m}(l_1 \dots l_n)$. Dann ist der *Sendungsgraph* $\mathcal{T}(l)$ *von* l definiert als $\langle \mathcal{E}(M), \rightarrow \rangle$ mit

$$\rightarrow := \prec(M) \cup \bigcup_{j=1}^{n} k(l_j, l_1).$$

Wir setzen $\mathfrak{k}(x, y)$ als die minimale Anzahl von durch k erzeugten Kanten auf dem Weg von x nach y.[24]

(c) Sei $H \in \mathfrak{H}$. Seien $l := l_1 \dots l_n l_1$ eine einfache Schleife in H und $M = \mathfrak{m}(l_1 \dots l_n)$. Der *erweiterte Sendungsgraph* $\mathcal{L}(l)$ *von* l ist definiert als $\langle \mathcal{E}(M), \rightarrow \rangle$ mit

$$\rightarrow := \prec(M) \cup \bigcup_{i,j=1}^{n} k(l_i, l_j).$$

$\mathfrak{k}$ ist definiert wie in (b).

Definition 2.16 (Eigenschaften) Sei $H \in \mathfrak{H}$.

(a) Falls $\mathcal{J}(\mathfrak{m}(l))$ für jede einfache Schleife l von H zusammenhängend ist, heißt H *zusammenhängend*.

(b) Falls $\mathcal{T}(l)$ für jede einfache Schleife l von H zusammenhängend ist, heißt H *konvergent*.

(c) Falls $\mathcal{L}(l)$ für jede einfache Schleife l von H zusammenhängend ist, heißt H *schwach konvergent*, ansonsten *divergent*.

(d) Falls $\mathcal{T}(l)$ für jede einfache Schleife l von H reflexiv ist, so heißt H *reflexiv*.

(e) Falls für jede einfache Schleife $l := l_1 \dots l_r l_1$ von H für alle $e, e' \in \mathcal{E}(M_{l_1})$ in $\mathcal{T}(l)$ gilt $\mathfrak{k}(e, e') = 1$, heißt H *stark abhängig*.

Das sehr grobkörnige Kriterium des Zusammenhangs ist nach [BAL96] modelliert. Dort wird der Kommunikationsgraph als *coordination graph* bezeichnet. Auch [AY99, M01b] definiert *communication graphs*, die unseren Kommunikationsgraphen entsprechen. Dort wird ein in unserer Terminologie zusammenhängendes HMSC als *beschränkt* bezeichnet. Das reicht wegen der Verallgemeinerung der Ordnung und der Konkatenation in unserer Modellierung nicht.

Starke Abhängigkeit ist ein technisches Kriterium, das später wichtig werden wird, aber zugleich intuitiv leichter einsichtig als die direkten Graphenkriterien ist. Es besagt eigentlich nur, daß man in einem HMSC erst dann ein bereits besuchtes MSC wieder betreten darf, wenn man die vorherige Instanz vollständig ausgeführt hat (Lemma 2.15).

Beispiel 2.11 Betrachten wir nun HMSC H aus Abbildung 2.14. Es sind drei einfache Schleifen zu betrachten. Wir erhalten dabei die drei Kommunikationsgraphen in Abbildung 2.15. Offensichtlich sind alle zusammenhängend, womit H zusammenhängend ist. Wir stellen aber fest, daß bei Konkatenation mit geeignetem k niemand auf r_4 warten muß: Ist $(\{r_4\} \times \mathcal{E}) \cap k(2, i) = \emptyset$ für alle $i \in \{1, 2, 3\}$, so kann, mit ansonsten gemäßigt asynchroner Konkatenation, eine beliebige Menge

[24] Formal könnte man eine Bewertung der Kanten mit 0 normalerweise und mit 1 für die k-Kanten angeben und $\mathfrak{k}(x, y) := \mathfrak{d}(x, y)$ setzen.

Abbildung 2.14 Ein HMSC mit drei einfachen Schleifen

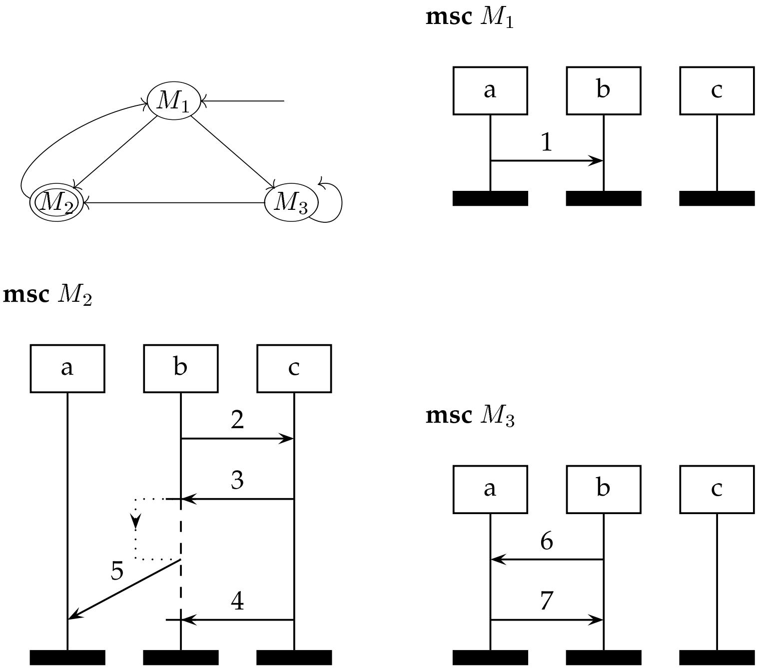
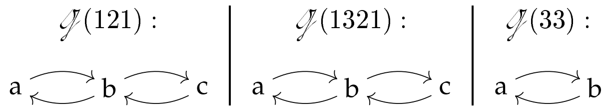


Abbildung 2.15 Die Kommunikationsgraphen für Abbildung 2.14



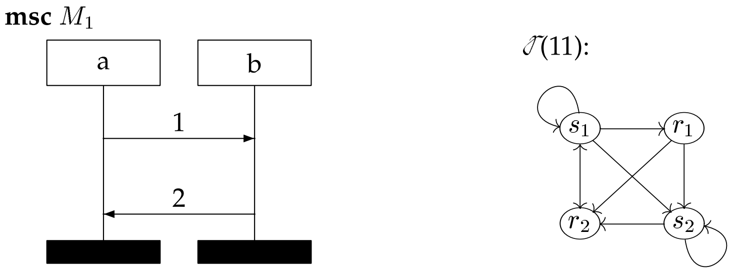
von s_4 in Schnittkanten von $(12)^*$ gesammelt werden, ohne dazugehörige r_4: $\mathfrak{c}$ ist also nicht nach oben beschränkt. Das heißt, daß das HMSC unbeschränkt ist (Definition 2.14).

Somit ist Zusammenhang kein hinreichendes Kriterium, obwohl es eigentlich ganz praktisch ausschaut. Das mag auch intuitiv einleuchten: M_2 sieht ziemlich kompliziert aus, führt aber nur zu 3 Pfeilen im Kommunikationsgraphen. Der eigentliche Punkt aber ist, daß die Parameter der *Konkatenation* im Kommunikationsgraphen nicht berücksichtigt werden.

Im Sendungsgraphen wird festgehalten, welche Ereignisse irgendwie voneinander abhängig sind. Das ist wesentlich feinkörniger als der Kommunikationsgraph und somit in unserem Fall angemessener. Der Zusammenhang dieser natürlichen Erweiterung des Kommunikationsgraphen ist zwar für die Übersetzbarkeit in Automaten hinreichend, aber die Bedingung kann noch abgeschwächt werden zur schwachen Konvergenz.

Beispiel 2.12 In Abbildung 2.16 ist ein HMSC mit dem Sendungsgraphen einer

Abbildung 2.16 Ein MSC und ein Sendungsgraph (Beispiel 2.12)



Schleife angegeben, gemäßigt asynchrone Konkatenation vorausgesetzt.[25]

Es gibt eine natürliche Hierarchie unter den Graphenkriterien, die auf der Ereignismenge aufsetzen (Lemma 2.7), die wir nun zu untersuchen beginnen.

Beispiel 2.13 Nicht jedes konvergente HMSC ist stark abhängig: Das HMSC aus Abbildung 2.13 (Beispiel 2.10) ist konvergent, aber in $\mathcal{T}(3\,3)$ führt der k-kürzeste Weg von r_2 nach s_1 über die beiden k-Kanten $r_2 \to A$ und $r_3 \to s_1$.

Starke Abhängigkeit ist also ein sehr starkes Kriterium. Betrachten wir Beispiel 2.13 weiter, so stellen wir fest, daß hier auch Nichtdeterminismus vorliegt in dem Sinne, daß man statt der Linearisierung $s_1 A s_3 r_3 s_1' \mathbf{r_1} \mathbf{r_1'} s_2 s_2' r_2 r_2' A' s_3' r_3'$ (Beispiel 2.10) auch $s_1 A s_3 r_3 s_1' \mathbf{r_1'} \mathbf{r_1} s_2 s_2' r_2 r_2' A' s_3' r_3'$ erhalten kann (Striche wieder nur für uns zur Verdeutlichung der verschiedenen Herkünfte). Es gibt also Linearisierungen, die sich nur in den Positionen zweier $\sim$-äquivalenter Ereignisse unterscheiden.

Es mag sein, daß in einer Anwendung dies nicht erwünscht ist. Dann kann man sich auf stark abhängige HMSCs beschränken, wie in Satz 2.9 zu sehen sein wird. Im allgemeinen erscheint die Eigenschaft der starken Abhängigkeit aber als extrem hart, und es wird sich herausstellen, daß sie nicht zwingend notwendig ist für die Übersetzbarkeit in Automaten.

Lemma 2.7 Sei $H \in \mathfrak{H}$. Es gilt:

$$H \text{ stark abhängig} \underset{\not\Longleftarrow}{\Longrightarrow} H \text{ konvergent} \underset{\not\Longleftarrow}{\Longrightarrow} H \text{ schwach konvergent}$$

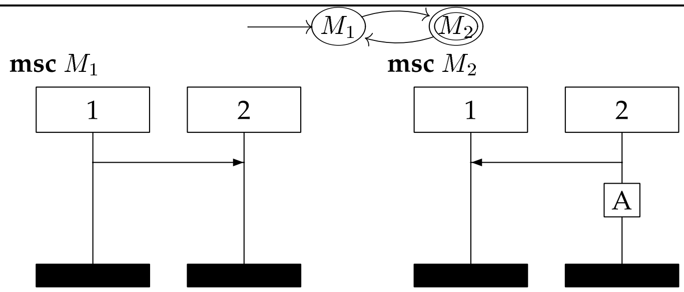
Beweis: Die Gültigkeit der behaupteten Implikationen ist klar. Für die andere Richtung betrachte man Beispiel 2.13 und das HMSC in Abbildung 2.17. ⊛

Die Untersuchung der Zusammenhänge setzen wir mit einem negativen Resultat fort. Intuitiv mag man meinen, es reiche hin, wenn auf jede Nachricht irgendjemand warte.

Definition 2.17 (Warte-Eigenschaft) Ein HMSC hat die *Warte-Eigenschaft*, falls für jede einfache Schleife l für jedes $e \in \mathcal{R}(M_{l_j})$ eines beliebigen l_j ein $e' \in \mathcal{E}(M_{l_i})$ für ein l_i existiert mit $(e, e') \in k(j, i)$.

[25]Es gibt insbesondere keine Kante von r_2 nach r_1. Bildeten wir hingegen $(\prec (M \circledvee M))/_\sim$ ab, so gäbe es eine solche Kante.

Abbildung 2.17 Ein weder konvergentes noch divergentes HMSC (Lemma 2.7)



Lemma 2.8 Warten und Zusammenhang sind voneinander unabhängig:

(a) Nicht jedes zusammenhängende HMSC erfüllt die Warte-Eigenschaft.

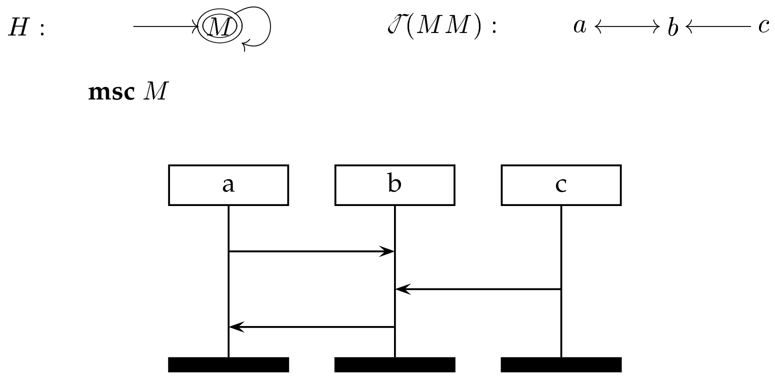
(b) Nicht jedes die Warte-Eigenschaft erfüllende HMSC ist zusammenhängend.

Beweis:

(a) Beispiel 2.11 (Abbildungen 2.14 und 2.15).

(b) In Abbildung 2.18 wird mit gemäßigt asynchroner Konkatenation offenbar die Warte-Eigenschaft erfüllt, aber der Kommunikationsgraph von MM ist nicht zusammenhängend. ⊛

Abbildung 2.18 Zum Beweis von Lemma 2.8



Weiter Resultate sind intuitiv leicht einsichtig.

Lemma 2.9 Sei $H \in \mathfrak{H}$.

(a) Ist H schwach konvergent, so ist H beschränkt.

(b) Ist H reflexiv und beschränkt, so muß H nicht schwach konvergent sein.

(c) Reflexivität und Beschränktheit sind voneinander unabhängig.

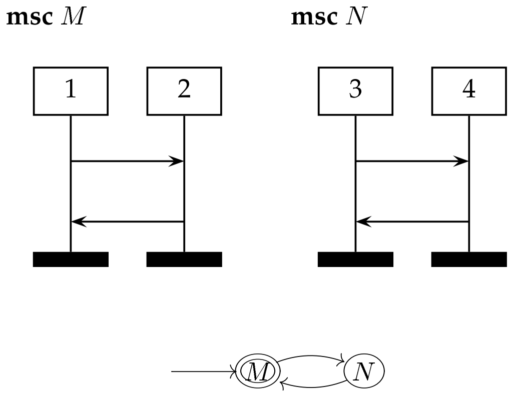
(d) Ist H schwach konvergent und sinnvoll, so ist H zusammenhängend. Ist H schwach konvergent und nicht sinnvoll, so muß H nicht zusammenhängend sein. Ist H zusammenhängend, so muß H nicht schwach konvergent sein.

Beweis:

(a) Die Aussage läßt sich viel leichter beweisen, wenn die Automaten definiert worden sind. Ich reiche die Argumentation dann nach.

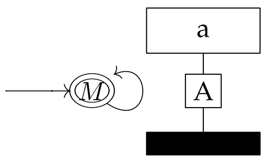
(b) Man betrachte das HMSC in Abbildung 2.19.

Abbildung 2.19 Ein 1-beschränktes, reflexives, divergentes HMSC



(c) HMSC 2 aus Abbildung 2.12 ist mit asynchroner Konkatenation offenbar unbeschränkt und reflexiv. Das HMSC aus Abbildung 2.20 mit gleichgültiger Konkatenation ist beschränkt und irreflexiv.

Abbildung 2.20 Ein beschränktes, irreflexives HMSC



(d) In Abbildung 2.12 ist mit HMSC 2 ein unzusammenhängendes HMSC angegeben, welches mit $k(2,2) = \{(r,s)\}$ konvergent ist (aber nicht sinnvoll). Ein zusammenhängendes, divergentes HMSC ist HMSC 1 aus Abbildung 2.12 mit $k(1,1) = \emptyset$ (vgl. auch Diskussion nach Lemma 2.10).

Sei nun H sinnvoll und schwach konvergent. Angenommen, H sei unzusammenhängend. Zu der schuldigen Schleife l betrachte den Kommunikationsgraphen.

Seien $p \neq q$ beliebige Knoten in $\mathscr{G}(l)$. Dann gibt es Ereignisse $e_p \in \mathcal{E}_p(M)$ und $e_q \in \mathcal{E}_q(M)$, sonst wären nicht $p, q \in \mathfrak{A}$. $\mathscr{G}(l)$ ist zusammenhängend, weswegen es einen Pfad $e_p \rightarrow_{\mathscr{G}(l)} x_1 \rightarrow_{\mathscr{G}(l)} x_2 \rightarrow_{\mathscr{G}(l)} \ldots \rightarrow_{\mathscr{G}(l)} x_b \rightarrow_{\mathscr{G}(l)} e_q$ gibt. Da H sinnvoll ist, kann man alle $x_\nu \rightarrow_{\mathscr{G}(l)} x_{\nu+1}$ mit $o(x_\nu) \neq o(x_{\nu+1})$ als Nachrichten wählen. Dadurch wird also jeweils eine Kante $o(x_\nu) \to_{\mathscr{G}(l)} o(x_{\nu+1})$ erzeugt. Also gibt es einen Pfad $p \to^?_{\mathscr{G}(l)} o(x_1) \to^?_{\mathscr{G}(l)} \cdots \to^?_{\mathscr{G}(l)} o(x_b) \to^?_{\mathscr{G}(l)} q$. Da $p \neq q$, ist dieser Pfad von Länge $\geqslant 1$, womit $p \to^+_{\mathscr{G}(l)} q$.

Sei nun $p \in \mathfrak{A}$ beliebig. Gibt es ein $q \in \mathfrak{A}$ mit $q \neq p$, so folgt $p \to^+_{\mathscr{G}(l)} q \to^+_{\mathscr{G}(l)} p$. Gibt es kein solches q, so folgt, da $p \in \mathfrak{A}$, daß es eine Selbstnachricht von p gibt, womit $p \to_{\mathscr{G}(l)} p$. ⊛

Das folgende Lemma ist aus Lemma 2.9 ausgelagert, weil es ein an sich schönes Resultat ist: Die Voraussetzungen des Zusammenhangs und des Konkatenationsart werden in der Literatur häufig ähnlich angenommen, und die Aktivitätsbedingung trifft man meistens nur deshalb nicht an, weil sie in Modellen ohne lokale Aktionen irrelevant ist.

Lemma 2.10 Sei H ein zusammenhängendes HMSC mit $k = k_{\circledvee}$. Sei für jede einfache Schleife l $\mathfrak{A}(\mathfrak{m}(l)) = \mathcal{P}$. Dann ist H schwach konvergent.

Beweis: Wir betrachten eine beliebige einfache Schleife l und ihren erweiterten Sendungsgraphen $G := \mathscr{G}(l)$.

Sei $p \in \mathcal{P}$. Wir zeigen zunächst, daß in G alle p-Ereignisse zusammenhängen. Nach der Definition von $k_{\circledvee}$ (Definition 2.11) ist klar, daß alle $\mathcal{S}_p \cup \mathcal{A}_p$-Ereignisse zusammenhängen. Betrachten wir nun ein $e \in \mathcal{R}_p$. Wieder wegen $k_{\circledvee}$ ist $\mathcal{S}_p \cup \mathcal{A}_p$ von e erreichbar. Für das $e' \in \mathcal{S}$ mit $m(e') = e$ setze $q := o(e')$. Wegen des Zusammenhangs gibt es dann ein $f \in \mathcal{R}_q$, von dem es einen Pfad zu e' gibt. Induktiv erhalten wir einen Kreis (wegen des Zusammenhanges) von Nachrichten und ein $g' \in \mathcal{S}_p$, wodurch ein Pfad $g' \to^+ e$ definiert wird. Somit sind alle Ereignisse der einzelnen Prozesse zusammenhängend.

Seien nun $p, q \in \mathcal{P}$ beliebig. Da alle Prozesse aktiv sind (Definition 2.2), gibt es aufgrund des Zusammenhangs eine Nachrichtenkette von p nach q. Mit der prozeßinternen Zusammenhangsbehauptung ergibt sich gesamt der Zusammenhang in $\mathscr{G}(l)$. ⊛

Lemma 2.10 gilt natürlich auch für alle Konkatenationsarten, die die Ereignisse stärker binden als gemäßigt asynchrone Konkatenation, insbesondere für $k_{\oplus}$ und $k_{\otimes}$ (Definition 2.11). Wir können die Anforderungen aber nicht ohne weiteres abschwächen: Sind nicht alle Prozesse aktiv, können wir wie in Abbildung 2.21 ein divergentes HMSC erhalten. Setzen wir bei dem HMSC in Abbildung 2.22 nur $k(M, M) = \{(r_2, s_1), (r_4, s_3)\}$, so ist es ebenfalls divergent ($(r_4, s_2) \in k_{\circledvee}$). Daß der Zusammenhang nötig ist, folgt aus Abbildung 2.19.

Abbildung 2.21 Ein divergentes HMSC unter $k_{\circledvee}$

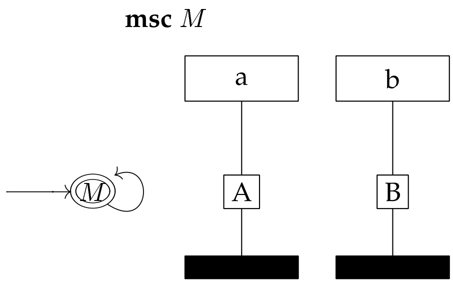
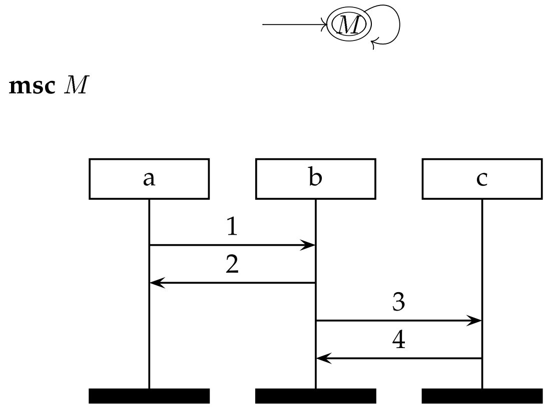


Abbildung 2.22 Ein divergentes HMSC ohne inaktive Prozesse

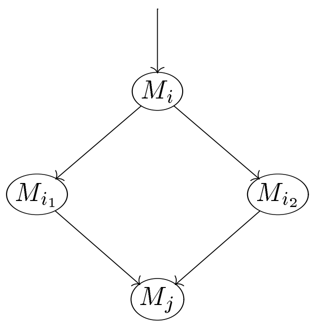


Ein letztes Kriterium benötigen wir später bei der Untersuchung des Determinismus der entstehenden Automaten.

Definition 2.18 (Diamanteigenschaft) Ein HMSC H hat die *Diamanteigenschaft*, falls es für alle paarweise verschiedenen Knoten $i, j, i_1, i_2 \in N$, für die es Läufe $l_1 i_1 \in \mathfrak{L}(H, i, i_1)$, $l_2 i_2 \in \mathfrak{L}(H, i, i_2)$, $l'_1 \in \mathfrak{L}(H, i_1, j)$, $l'_1 \in \mathfrak{L}(H, i_2, j)$ gibt, es für jedes $e \in \mathcal{E}(M_j)$ gewisse $e_1 \in \mathcal{E}(M_{i_1})$ und $e_2 \in \mathcal{E}(M_{i_2})$ gibt, sodaß $(e_1, e) \in k(i_1, j)$ und $(e_2, e) \in k(i_2, j)$.

Der Name kommt daher, daß wir in einem HMSC-Fragment wie in Abbildung 2.23 verlangen, daß ein Ereignis in M_j von M_i aus nur dann erreicht werden kann, wenn zuvor eines aus M_{i_1} oder M_{i_1} abgearbeitet wurde.[26] Der Sinn ist es, daß man sich beim Besuch einer Verzweigung entscheiden muß, welchen Weg man nimmt, bevor die beiden Pfade wieder zusammenlaufen.

Abbildung 2.23 Zur Erklärung von Definition 2.18 (Diamanteigenschaft)



Es wird sich als sinnvoll erweisen, sich auf schwach konvergente HMSCs zu beschränken. Das erscheint aus einem praktischen Gesichtspunkt heraus als sinnvoll (vgl. Diskussion am Anfang von Abschnitt 2.c.i): Irgendwann möchte man

[26] Für den Zweck, für den wir dies hier benutzen werden (Satz 2.9), könnte man die Voraussetzung noch leicht abschwächen: Auf irgendwelchen i_1, i_2 auf den Pfaden müssen obligatorische Ereignisse liegen. So ist es jedoch klarer.

ja die HMSCs implementieren, ansonsten stellte man keine Use Cases durch sie dar. Ein schwach konvergentes HMSC ist auf jeden Fall beschränkt (Lemma 2.9.a), was uns hier vor dem Problem bewahrt, von Implementierungen unbeschränkt große Nachrichtenpuffer fordern zu müssen. Also ist es sinnvoll, dies auszuschließen und im Falle eine Falles einfach den Benutzer zu warnen, daß ein mutmaßlich fehlerhaftes HMSC vorliegt. Auch beschränkte HMSCs können Probleme bereiten, wie anhand von Abbildung 2.19 gesehen (vgl. Motivation in Abschnitt 2.c.i). Die Verschärfung des Kriteriums der Beschränktheit auf schwache Konvergenz ist hingegen hinreichend für eine Übersetzbarkeit in Automaten,[27] denn damit können wir uns zu bestimmten Zeitpunkten eines Laufes immer sicher sein, daß bestimmte Ereignisse in jedem Falle schon durchgeführt geworden sein müssen (vgl. Diskussion auf Seite 52).

Natürlich muß man sich die Frage stellen, ob diese Einschränkung nicht praktisch eine zu große ist. Das ist sie nicht, denn überlegen wir uns, was es eigentlich bedeutet, ein divergentes HMSC implementieren zu wollen. Das heißt doch, daß ein Prozeß, der ein Ereignis gar nicht mehr ausführt, ununterscheidbar ist von einem, der diese Aktion zurückhält um sie dann spontan zu einem beliebigen Zeitpunkt nachzureichen. Das erscheint vom praktischen Gesichtspunkt her als wenig erstrebenswert.

2.c Automatentheoretische Semantik von (H)MSCs

Die Menge aller Linearisierungen ist eine Sprache. Was liegt näher, als diese Sprache durch einen Automaten zu erkennen? Das wollen wir in diesem Abschnitt beginnen, um einen leichten Übergang zu den Uhrenautomaten des nächsten Kapitels und der Übersetzung in solche im übernächsten Kapitel zu bringen.

2.c.i Regularität

Auf natürliche Weise kommt das Konzept der Regularität von Sprachen in unseren Untersuchungen vor. Das soll hier noch vertieft werden.

Wichtige Ergebnisse über *regular MSC languages* wurden in Arbeiten von Jesper Henriksen etc. veröffentlicht. So wird in [HMKT00a] gezeigt, daß Regularität von HMSCs unentscheidbar ist (Satz 2.3). In [HMKT00b] wurde gezeigt, daß sich für HMSCs eine zur Regularität äquivalente Charakterisierung über eine Logik geben läßt.

Beispiel 2.14 Es ist klar, daß die Menge aller Linearisierungen eines MSCs immer durch einen endlichen Automaten erkannt werden kann: Da es nur endlich viele Ereignisse gibt, ist die Menge aller Linearisierungen durch $|\mathcal{E}|!$ beschränkt, womit jedes MSC nur endlich viele Linearisierungen haben kann. Endliche Sprachen sind immer regulär und damit ist das möglich.

Bei HMSCs sieht das anders aus. Bei unbeschränkten HMSCs wie HMSC 2 in Abbildung 2.12 ist die Übersetzung nie möglich, wie man sich leicht überlegen kann: Man müßte beliebige Anzahlen von Ereignissen (nämlich den Sendungen der geschnittenen Nachrichten) in einer Kontrolle mithalten, was nicht mit nur endlich vielen Zuständen möglich ist.

[27] Sie ist nicht notwendig (Satz 2.3).

Auch bei beschränkten HMSCs kann man Probleme bekommen. Man betrachte nochmal das HMSC in Abbildung 2.19, welches wie gesagt 1-beschränkt ist. Trotzdem ist die Sprache nicht regulär: Mit einer Projektion auf das Teilalphabet $\{\mathsf{S}_{1,2}, \mathsf{S}_{3,4}\}$ erhalten wir die Sprache $\{w \in \{\mathsf{S}_{1,2}, \mathsf{S}_{3,4}\}^* : |w|_{\mathsf{S}_{1,2}} = |w|_{\mathsf{S}_{3,4}}\}$, die bekanntermaßen nicht regulär ist. Da reguläre Sprachen aber unter Substitution abgeschlossen sind ([HU00]), kann unsere Linearisierungssprache nicht selbst regulär sein.

Die Frage nach der Regularität von HMSCs wurde schon in manch einer Untersuchung betrachtet ([HMKT99, HMKT00a, HMKT00b, M01b, BLN02a]) und spielt auch ganz offensichtlich eine wichtige Rolle in unserem Vorhaben, eine automatentheoretische Beschreibung zu geben. Idealerweise möchten wir ein endliches[28] Transitionssystem als äquivalente Beschreibung der Semantik angeben (*Finite State Machine*, *FSM*), und dafür wären diskrete reguläre Sprachen erkennende endliche Automaten die erste Wahl.

Es ist jedoch klar, daß die Regularität von $[\![H;\pi]\!]$ als Sprache über Π von π abhängt (Definitionen 2.3 und 2.4). So erhalten wir durch Abbildung 2.19, wie gerade erklärt, eine irreguläre Sprache mit σ, aber offenbar ist $[\![M]\!] = \{a^{8n} : n \in \mathbb{N}\}$, falls $\pi = \lambda\_.a$. Wir untersuchen hier daher nur die Sprache als Sprache über $\mathcal{E}$ (also $\pi = \mathrm{id}$) und die Sprache erhalten durch die kanonische Ereignistypfunktion.

Definition 2.19 (Regularität) Sei $H \in \mathfrak{H}$. H heißt *π-regulär*, falls $[\![H;\pi]\!]$ regulär ist. Ist H σ-regulär, so heißt H *regulär*. Ist H id-regulär, so heißt H *vollständig regulär*.

Die Bezeichnung *vollständig regulär* ist wegen der Projektionseigenschaft der regulären Sprachen gewählt worden.

Lemma 2.11 Ein vollständig reguläres HMSC ist beschränkt.

Beweis: Klar: Durch Projektion auf das Senden und das Empfangen der schuldigen Nachricht bei einem unbeschränkten HMSC erhält man etwas wie $\{a^n b^n : n \in \mathbb{N}\}$, da jeder Anfangslauf zu einem Endlauf fortsetzbar ist. ⊛

Damit ist eine Hälfte des noch fehlenden Teiles zum Beweis von Lemma 2.9.a geleistet. Es ist nur noch zu zeigen, daß schwach konvergente HMSCs vollständig regulär sind.

Es ist klar, daß ein HMSC, was im wesentlichen ein endlicher Automat ist, alle regulären Sprachen erzeugen kann; Das soll hier aber dennoch erwähnt werden.

Satz 2.2 Sei L eine reguläre Sprache über Π. Dann gibt es ein $H \in \mathfrak{H}$ und ein π mit $[\![H;\pi]\!] = L$.

Beweis: Thompson-Konstruktion ([T68]). ⊛

[28]Daß endliche Systeme besser theoretisch zu erfassen sind, ist klar. In [LL93a] wird auch ein nettes praktisches Argument dafür gegeben, daß die Semantik *immer* durch ein endliches System dargestellt werden sollte: Implementiere man ein spezifiziertes System. Werde nun die Kommunikation gestört (Jemand zieht den Stecker bei einem Router.). Will man in einen konsistenten, vorherigen Zustand wiederaufsetzen, kann man dazu nur die endlich vielen verschiedenen Zustände der einzelnen Teilsysteme benutzen. „Hence, two such failures which result in the same local states to the processes are equivalent from the point of view of the potentially knowable state of the system. So each such equivalence class can be identified with a *global state*. Since there are finitely many finite-state processes, the global states are some equivalence. . . relation on a subset of the cartesian product of the state spaces of the individual processes. . . .“

Satz 2.3 Die folgende Frage ist im allgemeinen unentscheidbar:

> ***Gegeben:*** Ein HMSC H.
> ***Frage:*** Ist $[\![H]\!]$ regulär?

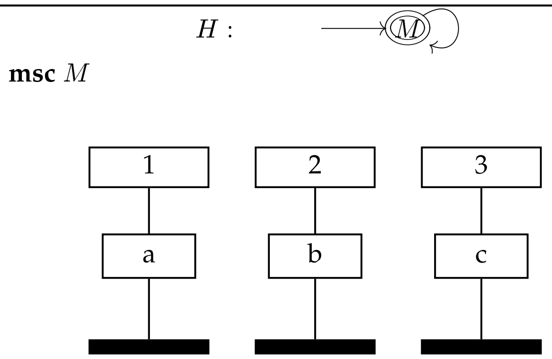
Beweis: Reduktion eines Spurproblems in [HMKT00a]. ⊛

Da schwache Konvergenz offenbar entscheidbar ist und hinreichend für die Regularität ist (Satz 2.8), stellt sie kein notwendiges Kriterium für die Übersetzbarkeit dar.

Bei sonst gleichen Parametern ist die Klasse der durch HMSCs erzeugten Sprachen eine echte Obermenge der regulären Sprachen (Satz 2.2).

Zu den kontextfreien Sprachen existiert leider kein so leichter Zusammenhang. Man kann sich überlegen, daß es kein $H \in \mathfrak{H}$ gibt, dessen Semantik isomorph zu der kontextfreien Sprache $\{a^n b^n : n \in \mathbb{N}\}$ ist; andererseits definiert H in Abbildung 2.24 mit $k_\otimes$ eine nicht-kontextfreie Sprache.[29]

Abbildung 2.24 Ein nicht kontextfreies HMSC



2.c.ii MSC-Automaten

Unser Hauptziel wird es sein, HMSC-Automaten zu definieren. Zum Aufwärmen untersuchen wir zunächst einfache MSCs. Das erlaubt es uns, ohne große Schwierigkeit deterministische Automaten anzugeben. Die Konstruktion entspricht einer Verallgemeinerung der in [AY99] angegebenen.

Definition 2.20 (MSC-Automat) Sei M ein MSC. Der *M-Automat* $\mathcal{A}(M)$ ist definiert als der endliche Automat

$$\left\langle \mathfrak{K}(M) \uplus \{q_s\}, \mathcal{E}, \delta, \{\emptyset\}, \{\mathcal{E}\} \right\rangle,$$

wobei die Transitionsfunktion δ gegeben ist durch

$$\begin{aligned} \delta(K,e) = & \begin{cases} K \cup \{e\} & e \notin K \wedge K \cup \{e\} \in \mathfrak{K} \\ q_s & \text{sonst,} \end{cases} \\ \delta(q_s, \_) = & \; q_s. \end{aligned}$$

[29] Angenommen, $[\![H]\!]$ sei kontextfrei. Sei p die Pumping-Lemma-Zahl. Da $a^p b^p c^p$ ein Kandidat ist, folgt der Widerspruch aus der Maximalspannweite des Mittelstücks.

q_s ist also ein Senkzustand. Daß der Automat alle Linearisierungen des MSCs erkennt, sollte klar sein. Um die Semantik zu erhalten, muß man also nur noch π anwenden.

Beispiel 2.15 Der Automat zum MSC aus Abbildung 2.5 ist in Abbildung 2.25 angegeben.

Abbildung 2.25 Der Automat zu Abbildung 2.5 (Beispiel 2.15)

$$
\begin{tikzcd}
 & & & \{s_1, r_1\} \arrow[r, "A"] \arrow[dr, "s_2"] & \{s_1, r_1, A\} \arrow[r, "s_2"] & \{s_1, s_2, r_1, A\} \arrow[dr, "r_2"] & \\
{} \arrow[r] & \emptyset \arrow[r, "s_1"] & \{s_1\} \arrow[ur, "r_1"] \arrow[r, "s_2"] & \{s_1, s_2\} \arrow[r, "r_1"] \arrow[dr, "r_2"] & \{s_1, s_2, r_1\} \arrow[r, "r_2"] \arrow[ur, "A"] & \{s_1, s_2, r_1, r_2\} \arrow[r, "A"] & \mathcal{E} \\
 & & & & \{s_1, s_2, r_2\} \arrow[ur, "r_1"] & &
\end{tikzcd}
$$

(Darstellung ohne Senkzustand und entsprechende Transitionen der Übersichtlichkeit wegen)

Zur Vorbereitung auf Definition 2.26, in der wir einen Automaten für ein HMSC erklären wollen, betrachten wir nun die direkte Komposition zweier Automaten ohne den Umweg über das MSC der Komposition.

Definition 2.21 (Kompositions-Automat) Seien M und M' MSCs mit $\mathcal{P} = \mathcal{P}'$. Sei $k \subseteq \mathcal{E} \times \mathcal{E}'$. Der *Kompositions-Automat* $\mathcal{A}(M, M', k)$ ist definiert als der endliche Automat

$$\big\langle (\mathfrak{K} \times \mathfrak{K}') \cup \{q_s\}, \mathcal{E} \uplus \mathcal{E}', \delta, \{(\emptyset, \emptyset)\}, \{(\mathcal{E}, \mathcal{E}')\} \big\rangle,$$

wobei die Transitionsfunktion δ gegeben ist wie folgt:

$$
\begin{aligned}
\delta((K, K'), e) = \; & \begin{cases} (K \cup \{e\}, K') & K \cup \{e\} \in \mathfrak{K} \wedge e \notin K \\ (K, K' \cup \{e\}) & K' \cup \{e\} \in \mathfrak{K}' \wedge e \notin K' \wedge \forall (e', e) \in k . e' \in K \\ q_s & \text{sonst,} \end{cases} \\
\delta(q_s, \_) = \; & q_s .
\end{aligned}
$$

Satz 2.4 Sei $M = M' \circ^k M''$. Dann gilt: $\mathcal{A}(M) \equiv \mathcal{A}(M', M'', k)$.

Beweis: folgt sofort aus den Definitionen 2.21, 2.8 und 2.20. ⊛

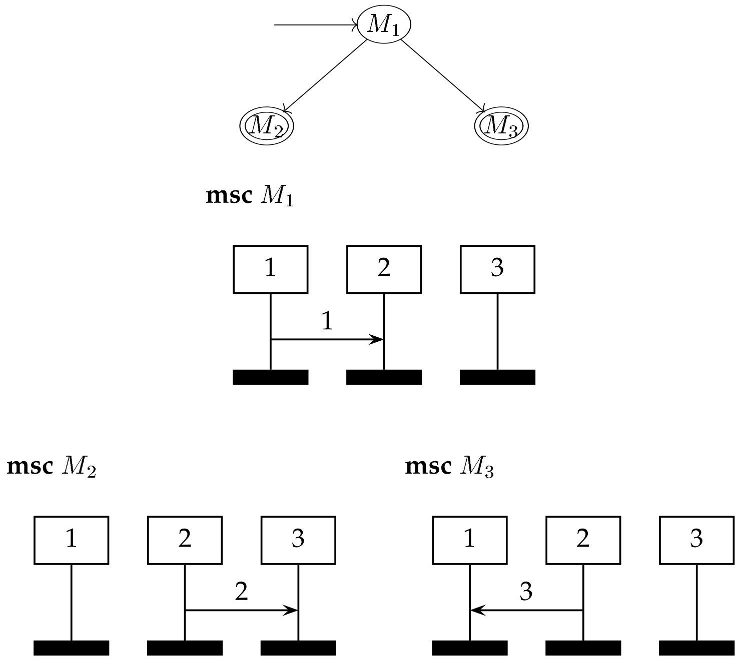
Die Komplexität der Automaten haben wir bereits bestimmt: Die Diskussion der Anzahl der Schnitte in Abschnitt 2.a.iv hat genau dies geleistet.

Alles, was nun folgt, ist Verallgemeinerung und Verfeinerung dessen. Wie in späteren Kapiteln zu sehen sein wird, muß man dann nur noch an manchen tiefliegenden Rädchen drehen, um Erweiterungen von (H)MSCs behandeln zu können, wohingegen die prinzipielle Automatenkonstruktion dem in den nächsten Unterabschnitten aufgezeigten Muster folgt.

2.c.iii HMSC-Automaten

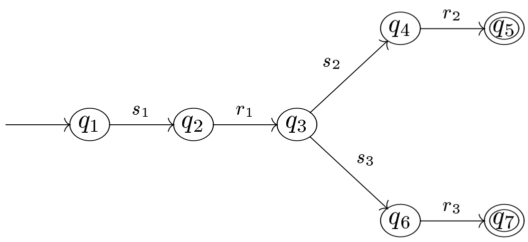
Beispiel 2.16 Um Definition 2.26 zu motivieren, betrachten wir kurz, was man intuitiv für Automaten basteln würde, hätte man ein HMSC vorliegen. Wir machen dies am Beispiel aus Abbildung 2.26, als Motivation Definition 2.21 unterlegend. Man könnte einen Automaten erzeugen, der etwa wie in Abbildung 2.27 aussieht.

Abbildung 2.26 Ein motivierendes HMSC (Beispiel 2.16)



Man beachte, daß dieser Automat deterministisch ist. Für schleifenfreie HMSCs

Abbildung 2.27 Ein möglicher Automat zu Abbildung 2.26



(Darstellung ohne Senkzustand und entsprechende Transitionen der Übersichtlichkeit wegen)

kann man dies offensichtlich immer erreichen[30]. In unserer Konstruktion wird dies nicht im allgemeinen der Fall sein. Zwar macht die Unterscheidung zwischen deterministischen und nichtdeterministischen endlichen Automaten von der prinzipiellen Darstellbarkeit und Analysierbarkeit keinen Sinn, aber wir müssen diese

[30]Es gibt nur endlich viele Pfade, also auch nur eine endliche Semantik.

Frage im Gedächtnis behalten, da die Unterscheidung bei Uhrenautomaten wesentlich ist (Satz 3.2).

Was machen wir bei Rekursion? Zunächst einmal scheint es so, als könne man keine endliche Beschreibung erzeugen. Das ist im allgemeinen auch richtig so, aber, wie angesprochen, beschränken wir uns auf schwach konvergente HMSCs (Definition 2.14). Dann klappt das, da wir nach einer gewissen Lauflänge immer sicher sein können, wie ein bestimmter Präfix-Teil des Laufes ausgesehen haben muß, und diesen dann entfernen können (Definition 2.25).

Definition 2.22 (Konfiguration) Sei H ein HMSC. Eine *Konfiguration von H* ist ein Paar (h, φ), wobei gilt:

(i) $h = h_1 \dots h_{|h|} \in N^*$ und

(ii) $\varphi = \varphi_1 \dots \varphi_{|h|}$, wobei $\forall\, j \in \{1, \dots, |h|\}. \varphi_j \in \mathfrak{K}(M_{h_j})$.

Wir setzen $\mathfrak{e}(\varphi) := \bigcup_{j=1}^{|\varphi|} \varphi_j$. Die Menge aller Konfigurationen sei $\mathcal{K}(H)$.

h gibt dabei an, welche Knoten gerade in Arbeit sind, und φ zeigt an, wo wir gerade in den jeweiligen Knoten sind.[31]

Diese Definition ist etwas komplizierter als die der entsprechenden (und gleichbezeichneten) Objekte in [AY99], da wir nicht unbedingt totale Ordnungen auf den Prozessen und (a)synchrone Konkatenation der MSCs voraussetzen.

Diese Konfigurationen werden die Zustände unseres Automaten bilden. Offensichtlich haben wir aber, sofern ein HMSC eine Schleife enthält, beliebig lange Pfade, also unendlich viele Konfigurationen. Wir können aber nur endlich viele davon gebrauchen. Dies erreichen wir dadurch, daß wir eine Äquivalenzrelation auf den Zuständen definieren, die für schwach konvergente HMSCs nur endlich viele Äquivalenzklassen besitzt.

Doch zunächst wollen wir auf natürliche Weise ein unendliches Transitionssystem definieren.

Definition 2.23 (Folgekonfiguration) Seien H ein HMSC und $(h, \varphi) \in \mathcal{K}(H)$ (Definition 2.22). Dann ist (h', φ') eine *Folgekonfiguration von (h, φ) unter $e \in \Sigma$*, falls eines gilt von:

(1)
- ▷ $h' = h \cdot l$ für ein $h_{|h|} l \in \mathfrak{L}(H, h_{|h|}, l_{|l|})$ und $|l| > 0$;
- ▷ $\varphi'_{|h'|} = \{e\}$; $\varphi'_g = \varphi_g$ für $g \leqslant |h|$, $\varphi'_g = \emptyset$ für $|h| < g < |h'|$;
- ▷ $\mathfrak{e}(\varphi') \in \mathfrak{K}(\mathfrak{m}(h'))$.

(2)
- ▷ $h' = h$;
- ▷ es gibt ein i mit $e \in \mathcal{E}(M_{h_i})$, $e \notin \varphi_i$, $\varphi'_j = \varphi_j$ für $j \neq i$ und $\varphi'_i = \varphi_i \cup \{e\}$;
- ▷ $\mathfrak{e}(\varphi') \in \mathfrak{K}(\mathfrak{m}(h'))$.

[31] Da die Ereignismengen disjunkt sind, bestimmt φ eindeutig h, falls jedes $\varphi_i \neq \emptyset$. Das muß aber nicht so sein, weswegen h notwendig ist, damit überhaupt nicht angearbeitete MSCs voneinander unterscheidbar sind.

Die Schritte der ersten Art sind also diejenigen, die von einem MSC zu einem nächsten übergehen, wobei dies nur klappt, wenn man dort auch etwas ausführen kann. Die andersartigen Schritte sorgen einfach dafür, daß man in einem MSC, das bereits in Arbeit ist, weiterkommt. In Abbildung 2.27 wären die Schritte für s_2 und s_3 Schritte erster Art, die anderen Schritte zweiter Art.

Man beachte, daß, um zu bestimmen, ob ein Schritt möglich ist, die Semantik des bisherigen Laufes herangezogen wird. Das erscheint natürlich. Wir ziehen aber nur die Teile des Laufes heran, die in der ersten Komponente der Konfiguration stehen. Wie später zu sehen sein wird (Definition 2.25, Lemma 2.13), muß das nicht der ganze Lauf sein, ja, es können sogar Lücken auftreten: h_i und h_{i+1} müssen nicht durch eine Kante miteinander verbunden sein. Aus diesem Grunde ist auch nicht $\mathfrak{L}(M)$ der Wertebereich von h und ist die Semantik-Definition 2.13.a nicht auf $\mathfrak{L}$ eingeschränkt.

Definition 2.24 (HMSC-Automat) Für ein $H \in \mathfrak{H}$ ist der *H-Automat* $\mathcal{A}(H)$ definiert als der Automat

$$\left\langle \mathcal{K}(H), \mathcal{E}, \{c_0\}, \delta, K_* \right\rangle,$$

wobei $c_0 = (i_0, \emptyset)$, $\delta(c, x) = \{c' : c' \text{ ist eine Folgekonfiguration von } c \text{ unter } x\}$ und $K_* := \{(hi, \varphi) : i \in N_* \wedge \varphi_j \text{ vollständig für alle } j \in \{1, \ldots, |hi|\}\} \subseteq \mathcal{K}(H)$.

Lemma 2.12 Sind $H \in \mathfrak{H}$, $l \in \mathfrak{L}(H)$ und $w \in [\![l]\!]$, so ist für jedes $i \in \{1, \ldots, |w| + 1\}$ $\{w_1, \ldots, w_{i-1}\} \in \mathfrak{K}(\mathfrak{m}(l_1 \ldots l_s))$ für ein geeignetes $s \in \{1, \ldots, |l|\}$.

Beweis: klar. ⊛

Satz 2.5 Für ein $H \in \mathfrak{H}$ und eine Ereignistypfunktion π ist $[\![H; \pi]\!] = \pi(\mathfrak{L}(\mathcal{A}(H)))$.

Beweis: Wir zeigen die Behauptung für $\pi := \mathrm{id}$. Im allgemeinen Fall folgt sie sofort daraus.

Sei $w \in [\![l]\!]$ für ein $l \in \mathfrak{L}_*(H)$. Falls wir zeigen können, daß $c_0 \rightarrow^w (l, \varphi) \in K_*$, so ist $[\![H]\!] \subseteq \mathfrak{L}(\mathcal{A}(H))$ bewiesen.

Wir beginnen damit, $c_0 \rightarrow^w (l, \varphi) \in K$ für beliebige $w \in [\![l]\!]$ für $l \in \mathfrak{L}(H, i_0, z)$ bei beliebigem z nachzuweisen, also nicht nur für Endläufe. Dies machen wir in erster Linie durch Induktion über $r := |l|$.

$r = 1$: Sei $M = M_{l_1}$. Dann ist $w \in [\![M]\!]$. Sei z minimal, sodaß es keinen nur Transitionen der zweiten Art verwendenden Pfad $c_0 \rightarrow^{w_1} c_1 \rightarrow^{w_2} \ldots \rightarrow^{w_{z-1}} c_{z-1} \rightarrow^{w_z} c_z$ gibt. Da nur Transitionen der zweiten Art verwendet werden, ist $c_{z-1} = (l_1, \{w_1, \ldots, w_{z-1}\})$. Daß nun w_z nicht gelesen werden kann, bedeutet, daß $\{w_1, \ldots, w_{z-1}, w_z\} \notin \mathfrak{K}(M)$. Das ist aber ein Widerspruch zu Lemma 2.12. Also gibt es einen Lauf wie gefordert und der Induktionsanfang ist fertig.

Induktionsschritt: Sei $w \in [\![l_1 \ldots l_r l_{r+1}]\!]$. Sei w_i das erste Element des letzten besuchten MSCs (also $i = \min\{j : w_j \notin \mathcal{E}(\mathfrak{m}(l_1 \ldots l_r))\}$) (Falls ein solches i nicht existiert, läßt sich direkt die Induktionsvoraussetzung anwenden und wir sind fertig.). Induktiv können wir voraussetzen, daß $c_0 \rightarrow^{w_1 \cdots w_{i-1}} (l_1 \ldots l_s, \hat{\varphi})$ für ein geeignetes $\hat{\varphi}$ und ein $s \leqslant r$.[32] Angenommen, von dort existiere keine w_i-Transition der ersten Art. Offenbar erfüllt aber $(l_1 \ldots l_s l_{s+1} \ldots l_r l_{r+1}, \varphi\{\emptyset\}^{r-s}\{w_i\})$ die ersten beiden Bedingungen dieser Transitionsart, sodaß $\bigcup_{j=1}^{s} \varphi_i \cup \{w_i\} \in \mathcal{K}(\mathfrak{m}(l_1 \ldots l_{r+1}))$ zu zeigen ist. Das folgt aber aus Lemma 2.12.

[32] Transitionsart 1 erlaubt es, einige MSCs zu überspringen. Damit muß nicht $s = r$ gelten. Unsere Induktionsrelation ist also $\{1, \ldots, r-1\} \curvearrowright r$.

Für den Induktionsschritt ist noch zu argumentieren, daß es auch Transitionen für die $w_{i+1}, \ldots, w_{|w|}$ gibt. Wir starten also in einem bestimmten c und wollen $w_{i+1} \ldots w_{|w|}$ mittels Transitionen der zweiten Art lesen. Daß das geht, folgt genau wie in der gleichen Situation des Induktionsanfangs.

Kehren wir zu dem w zurück, mit dem wir starteten. Hier ist l ein Endlauf, also $l \in \mathfrak{L}_*(H)$. Damit ist $(l, \varphi) \in K_*$, falls alle φ_ν vollständig sind. Wäre aber ein φ_ν nicht vollständig, fehlte also ein $e \in \mathcal{E}(M_{l_\nu})$, so wäre die Linearisierung des gemäß Definition 2.13 erzeugten MSCs nicht vollständig.

Gesamt ist $\mathfrak{L}((A)) \supseteq [\![H]\!]$ gezeigt. Es bleibt noch, $\mathfrak{L}(\mathcal{A}(H)) \subseteq [\![H]\!]$ zu zeigen.

Gebe es einen Pfad $c_0 \rightarrow^w (l, \varphi)$ für irgendein w. Es ist $\mathfrak{e}(\varphi) \in \mathfrak{K}(\mathfrak{m}(l))$ nach Definition 2.23. Offensichtlich kann w ergänzt werden zu einem $w' \in [\![l]\!]$. Ist $w \in \mathfrak{L}(\mathcal{A}(H))$, so gibt es einen solchen Pfad endend in (l, φ) mit $l_{|l|} \in N_*$ und $\mathfrak{e}(\varphi) = \mathcal{E}(\mathfrak{m}(l))$. Damit ist mit $w' := w\ w' \in [\![l]\!]$. l ist aber nach der ersten Transitionsart aus Definition 2.23 konstruiert, womit $l \in \mathfrak{L}_*(H)$, also $w \in [\![H]\!]$.

Damit sind beide Inklusionsrichtungen gezeigt und die Behauptung ist gesamt bewiesen. ⊛

Will man einen Automaten haben, der direkt π-Wörter erkennt, so kann man auch $\mathcal{A}$ erzeugen und anschließend alle Transitionsbuchstaben durch ihre π-Bilder ersetzen.

2.c.iv Endliche HMSC-Automaten

Das Problem ist, daß der soeben definierte Automat unendlich ist. Nun wollen wir aber nicht mehr benötigte Knoten aus der Knotenliste herausnehmen, um eine endliche Konfigurationsmenge zu erhalten. Zu diesem Zweck definieren wir die Funktion κ. Ziel von κ ist es, solche MSCs als irrelevant auszusortieren, die keine Bedingungen an nachfolgende nicht erfüllen. Zumindest für die vollständig erfüllten gilt dies.

Definition 2.25 (Konfigurationsbereinigungsfunktion) Sei (h, φ) eine Konfiguration. Dann ist $\kappa(h, \varphi) := (h', \varphi')$ definiert durch:

(i) $h' = h_{i_1} h_{i_2} \ldots h_{i_z} h_{|h|}$ mit $1 \leqslant i_1 < i_2 < \ldots < i_{z-1} < i_z < |h|$, und für ein $i \in \{1, \ldots, |h| - 1\}$ ist genau dann $i \notin \{i_1, \ldots, i_z\}$, falls $\varphi_i = \mathcal{E}(M_{h_i})$.

(ii) $\varphi' = \varphi_{i_1} \varphi_{i_2} \ldots \varphi_{i_z} \varphi_{|h|}$.

Grundsätzlich wird also jedes vollständige MSC entfernt, da dann davon ausgegangen wird, daß es nicht mehr benötigt wird (Falls wir κ im Automaten benutzen wollen, müssen wir das also noch sicherstellen.). Nur das letzte MSC kann man nicht entfernen, weil es angibt, wie wir den Lauf durch den Automaten fortsetzen können. In jedem Fall haben wir aber genau deshalb kein Problem mit unserer Transitions-Definition 2.23, weil wir für die Konkatenation unseren $\circ$-Operator dank der allgemeinen Definition der Ordnung k (Definition 2.10) auch für nicht hintereinanderliegende MSCs sinnvoll einsetzen können.

Es soll noch kurz erwähnt werden, daß κ tatsächlich Konfigurationen ergibt.

Lemma 2.13 Für ein $(h, \varphi) \in \mathcal{K}(H)$ ist $\kappa(h, \varphi) \in \mathcal{K}(H)$. Weiter ist $\kappa \circ \kappa = \kappa$.

Beweis: klar. ⊛

Jetzt sind wir, endlich, bereit, den Automaten zu definieren. Dies geschieht über eine Äquivalenzklassenbildung bzgl. κ. Trotz des Namens ist er nicht automatisch endlich – Wann das gilt, werden wir später untersuchen (Satz 2.8).

Definition 2.26 (HMSC-Automat) Sei H ein HMSC. Der *H-endlich-Automat* $\mathcal{F}(H)$ ist definiert als der Automat

$$\langle C, \mathcal{E}, \{c_0\}, \delta, C_* \rangle,$$

wobei gilt:

- ▷ $c_0 := (i_0, \emptyset) \in C$,
- ▷ $\delta(c, \xi) \subseteq C$ für alle $c \in C$ und $\xi \in \mathcal{E}$,
- ▷ $\delta(c, \xi) = \{\kappa(c') : c \to^{\xi}_{\mathcal{A}(H)} c'\}$;
- ▷ C ist die kleinste Menge, die dies erfüllt,

und $C_* = C \cap K_*$ mit K_* aus Definition 2.24.

Also resultiert ein Automat, der eine Menge an Konfigurationen als Zustände hat und mit der natürlichen Folgetransition aus Definition 2.23, die nach Lemma 2.13 anwendbar ist, übergeht bei nötigem Herausschneiden irrelevanter Knoten gemäß Definition 2.25, wobei C als Abschluß darunter definiert ist. Der Startzustand hat keine eingehenden Kanten.

Wir wollen nun zeigen, daß κ tatsächlich nichts an der Erkennungssemantik ändert. Dafür benötigen wir aber einige wenige Hilfsaussagen.

Lemma 2.14 Seien $H \in \mathfrak{H}$, $A = \mathcal{A}(H)$ und $B = \mathcal{F}(H)$.

(a) Gilt $q_1 \to^e_B q_2$ und $q_1 = \kappa(\hat{q}_1)$, so gibt es ein $\hat{q}_2$ mit $q_2 = \kappa(\hat{q}_2)$ und $\hat{q}_1 \to^e_A \hat{q}_2$.

(b) Gilt $q_1 \to^e_A q_2$, so geht $\kappa(q_1) \to^e_B \kappa(q_2)$.

(c) Genau dann ist $c \in K_*$, falls $\kappa(c) \in K_*$.

Beweis:

(a) Seien also $q_1, q_2, \hat{q}_1$ gegeben mit den Werten $q_1 = (h, \varphi)$ und $\hat{q}_1 = (\hat{h}, \hat{\varphi})$. Die B-Transition hat eine A-Transition als Grundlage, also $q_1 \to^e_A q$ mit $\kappa(q) = q_2$. Die Situation ist in Abbildung 2.28 dargestellt.

Sei $q_1 \to^e_A q$ zunächst nach dem ersten Fall aus Definition 2.23 gemacht, also $q = (hl, \varphi\psi)$ für gewisse l, ψ. Wir setzen nun $\hat{q}_2 := (\hat{h}l, \hat{\varphi}\psi)$ und zeigen, daß

$$(\hat{h}, \hat{\varphi}) \underset{A}{\xrightarrow{e}} (\hat{h}l, \hat{\varphi}\psi) \qquad (*)$$

möglich ist.

Angenommen, eine Transition $(*)$ existiere nicht. Da die ersten beiden Bedingungen an diese Transitionsart in Definition 2.23 erfüllt sind (Die angefügten Stücke l, ψ sind die gleichen wie in $q_1 \to_A q$ und $\hat{h}_{|\hat{h}|} = h_{|h|}$.), heißt das, daß

Abbildung 2.28 Zum Beweis von Lemma 2.14.a

$$\begin{tikzcd}
\hat{q}_1 \arrow[rr, dashed, "e", "A"'] \arrow[dd, "\kappa"'] & & \hat{q}_2 \arrow[dd, dashed, "\kappa"] \\
& q \arrow[dr, dotted, "\kappa"'] & \\
q_1 \arrow[ur, dotted, "e", "A"'] \arrow[rr, "e", "B"'] & & q_2
\end{tikzcd}$$

für $e \in \mathcal{E}(M_{l_{|l|}})$ $\mathfrak{e}(\hat{\varphi}) \cup \{e\} \notin \mathfrak{K}(\mathfrak{m}(\hat{h}l))$, also daß ein $e' \in \mathcal{E}(\mathfrak{m}(\hat{h}))$ mit $e' \prec e$ existiert, wobei $e' \notin \mathfrak{e}(\hat{\varphi})$. Da $e' \in \mathcal{E}(M_{\hat{h}_j})$ für ein j, wird diese Stelle durch κ nicht entfernt. Dieses nicht vollständige MSC ist also auch in q_1 vorhanden, im Widerspruch zur Annahme. Daß $\kappa(\hat{q}_2) = q_2$ folgt sofort, und die Behauptung ist für diesen Fall bewiesen.

Falls man eine Ereignistransition der anderen Art in A verwendet hat, läuft die Argumentation ganz analog durch.

(b) Die Situation ist ähnlich zu der in (a), wie durch Abbildung 2.29 deutlich wird.

Abbildung 2.29 Zum Beweis von Lemma 2.14.b

$$\begin{tikzcd}
q_1 \arrow[rr, "e", "A"'] \arrow[dd, "\kappa"'] & & q_2 \arrow[dd, "\kappa"] \\
& q \arrow[dr, dotted, "\kappa"'] & \\
\kappa(q_1) \arrow[ur, dotted, "e", "A"'] \arrow[rr, dashed, "e", "B"'] & & \kappa(q_2)
\end{tikzcd}$$

Werde e durch eine Transition der ersten Art gelesen. Seien also $q_1 = (\hat{h}, \hat{\varphi})$, $q_2 = (\hat{h}l, \hat{\varphi}\psi)$, $\kappa(q_1) = (h, \varphi)$. Sei $q = (hl, \varphi\psi)$ und sei $q' = \kappa(q)$. Analog zu (a) zeigt man, daß $\kappa(q_1) \rightarrow_A^e q$. Damit geht $\kappa(q_1) \rightarrow_B^e q'$. Daß $q' = \kappa(q_2)$, ist wieder klar, und damit ist dieser Fall abgeschlossen.

Der Fall der Transition zweiter Art läßt sich wieder vollkommen analog behandeln.

(c) klar. ⊛

Satz 2.6 Für ein $H \in \mathfrak{H}$ ist $\mathfrak{L}(\mathcal{A}(H)) = \mathfrak{L}(\mathcal{F}(H))$.

Beweis: Seien $A = \mathcal{A}(H)$ und $B = \mathcal{F}(H)$.

Sei zunächst $w := e_1 \ldots e_\nu \in \mathfrak{L}(\mathcal{A}(M))$ (Mit $\nu \geqslant 1$, sonst ist die Aussage trivial.). Dann gibt es einen w erkennenden A-Lauf $c_0 \rightarrow_A c_1 \rightarrow_A^{\nu-1} c_\nu$ mit $c_\nu \in K_*$. Da $\kappa(c_0) = c_0$, erhalten wir mit Lemma 2.14.b $c_0 \rightarrow_B^{e_1} \kappa(c_1)$. Wieder mit Lemma 2.14.b ergibt sich $\kappa(c_1) \rightarrow_B \kappa(c_2)$, und iterativ $c_0 \rightarrow_B^w \kappa(c_\nu)$. Abbildung 2.30 stellt diese Iteration bildlich dar. Mit Lemma 2.14.c erhalten wir $\kappa(c_\nu) \in K_* \cap C$ und somit $w \in \mathfrak{L}(\mathcal{F}(M))$.

Abbildung 2.30 Zum Beweis von Satz 2.6

$$
\begin{tikzcd}
c_0 \arrow[r, "e_1", "A"'] \arrow[d, "\kappa"] & c_1 \arrow[r, "e_2", "A"'] \arrow[d, "\kappa"] & c_2 \ldots \arrow[r, "e_\nu", "A"'] & c_\nu \arrow[d, "\kappa"] \\
c_0 \arrow[r, dashed, "e_1", "B"'] & \kappa(c_1) \arrow[r, dashed, "e_2", "B"'] & \kappa(c_2) \ldots \arrow[r, dashed, "e_\nu", "B"'] & \kappa(c_\nu)
\end{tikzcd}
$$

Abbildung 2.31 Zum Beweis von Satz 2.6, weiter

$$
\begin{tikzcd}
c_0 \arrow[r, dashed, "e_1", "A"'] \arrow[d, "\kappa"] & \hat{c}_1 \arrow[r, dashed, "e_2", "A"'] \arrow[d, dashed, "\kappa"] & \hat{c}_2 \ldots \arrow[r, dashed, "e_\nu", "A"'] & \hat{c}_\nu \arrow[d, dashed, "\kappa"] \\
c_0 \arrow[r, "e_1", "B"'] & c_1 \arrow[r, "e_2", "B"'] & c_2 \ldots \arrow[r, "e_\nu", "B"'] & c_\nu
\end{tikzcd}
$$

Zur Umkehrung: Sei $w := e_1 \ldots e_\nu \in \mathfrak{L}(\mathcal{F}(M))$. Dann gibt es also einen w erkennenden B-Lauf $c_0 \to_B c_1 \to_B^{\nu-1} c_\nu$ mit $c_\nu \in C_*$. Durch iterative Anwendung von Lemma 2.14.a ergibt sich $c_0 \to_A^w \hat{c}_\nu$ mit $\kappa(\hat{c}_\nu) = c_\nu$. Diese Iteration mag durch Abbildung 2.31 deutlich werden. Aus Lemma 2.14.c folgt wieder $\hat{c}_\nu \in K_*$ und also $w \in \mathfrak{L}(\mathcal{A}(H))$. $\circledast$

Satz 2.7 Für ein $H \in \mathfrak{H}$ und eine Ereignistypfunktion π ist $[\![H;\pi]\!] = \pi(\mathcal{L}(\mathcal{F}(H)))$.

Beweis: Sätze 2.5 und 2.6. $\circledast$

Nun ist noch zu klären, wann es sich tatsächlich um einen endlichen Automaten handelt, d. h. wann aus der iterativen Vorschrift der Erstellung der Zustandsmenge gemäß Definition 2.26 tatsächlich nur eine endliche Menge entsteht. Das ist durchaus nicht immer der Fall, wie man sich etwa leicht klarmachen kann, wenn man denn die Definition auf HMSC 2 aus Abbildung 2.12 anwendet.[33] Hauptpunkt dafür wird es sein, daß nur endliche Pfadlängen beibehalten werden: κ findet ab gewisser Längen immer vollständig abgearbeitete Knoten: Haben wir ein MSC noch nicht ganz abgearbeitet, kommt mit Sicherheit ein Punkt, an dem wir nicht mehr weiter im Pfad voranschreiten können, sondern ein altes MSC fertigmachen müssen.

Satz 2.8 Für ein schwach konvergentes $H \in \mathfrak{H}$ besitzt $\mathcal{F}(H)$ nur endlich viele Zustände.

Beweis: Sei K eine Schnittkante eines MSCs M. Angenommen, es existieren $x \notin K$ und $y \in K$. Wir werden zeigen, daß dann K nur $\alpha := \mathfrak{k}(x,y)$-mal in einer Konfiguration c, die in $\mathcal{A}(H)$ vorkommt, als Mitglied einer einfachen Schleife l um M auftreten kann. Nehmen wir also an, wir hätten $\alpha + 1$ Schleifen. Für den Pfad, der

[33]Wie gesagt ist die zu erkennende Sprache nicht regulär. Daher wäre es sehr verdächtig, lieferte $\mathcal{F}$ einen sie erkennenden endlichen Automaten.

$\mathfrak{k}(x, y)$ zugrundeliegt, seien $x_1, \ldots, x_\alpha$ die Ereignisse, in die k-Kanten einmünden (In $\mathcal{T}(l)$ haben wir also $x \preceq x'_1 \to x_1 \preceq x'_2 \to x_2 \ldots x'_\alpha \to x_\alpha \preceq y$, wobei die $\to$-Kanten nur aus k entstehen.). Sind $\varphi_{z_1}, \ldots, \varphi_{z_{\alpha+1}}$ die angenommenen Vorkommen von K in c, so muß wegen $y \in K$ $x_\alpha \in \varphi_{z_\alpha} = K$ gelten, und induktiv insbesondere $x \in \varphi_{z_1} = K$. Dies ist ein Widerspruch zur Voraussetzung, womit wir die angegebene Beschränkung haben.

Sei nun $K = \emptyset$ die Schnittkante eines MSCs M. Da jedes $x \in \mathcal{E}(M)$ von einem $y \in \mathcal{E}(M)$ abhängig ist, erhalten wir analog wie im obigen Fall eine Beschränkung des Auftretens dieses K.

Die einzigen Schnittkanten, die beliebig häufig in Konfigurationen, die Zustände von $\mathcal{A}(H)$ sind, auftreten können, sind vollständige. Diese werden jedoch irgendwann durch κ beseitigt, weswegen $\mathcal{F}(H)$ tatsächlich nur endlich viele Zustände besitzen kann. ⊛

Die Endzustandsmenge eines H-endlich-Automaten umfasst immer genau $|N_*|$ Zustände.

Rest zum Beweis von Lemma 2.9.b: Nach Sätzen 2.7 und 2.8 können die Linearisierungen eines schwach konvergenten HMSCs durch einen endlichen Automaten erkannt werden. Das HMSC ist also vollständig regulär. Aus Lemma 2.11 folgt dann die Beschränktheit.

Tatsächlich können wir sogar eine Schranke angeben: Ein konvergentes HMSC ist $\max\{\mathfrak{k}(x, y) : x, y \in M$ für ein M auf einer einfachen Schleife$\} + 1$-beschränkt (Beweis zu Satz 2.8). ⊛

Beispiel 2.17 In Abbildung 2.32 erhalten wir mit $k_{\heartsuit}$ die Abbildung 2.33 angegebenen Beziehungen (wobei zur leichteren Einsicht die Ereignisse des Folge-MSCs gestrichelt sind). Die Präzedenzordnungen der einzelnen MSC sind einfach und klar:

$$s_1 \prec r_1, \quad s_2 \prec r_2 \prec s_3 \prec r_3, \quad s_4 \prec r_4, \quad s_5 \prec r_5, \quad B \prec s_6 \prec r_6.$$

Nun sind wir bereit, den Automaten anzugeben. Er ist in Abbildung 2.34 dargestellt. Die Schnittkanten sind dabei zur Vereinfachung nur so notiert, daß man gerade weiß, was gemeint ist.

2.c.v Nachträge

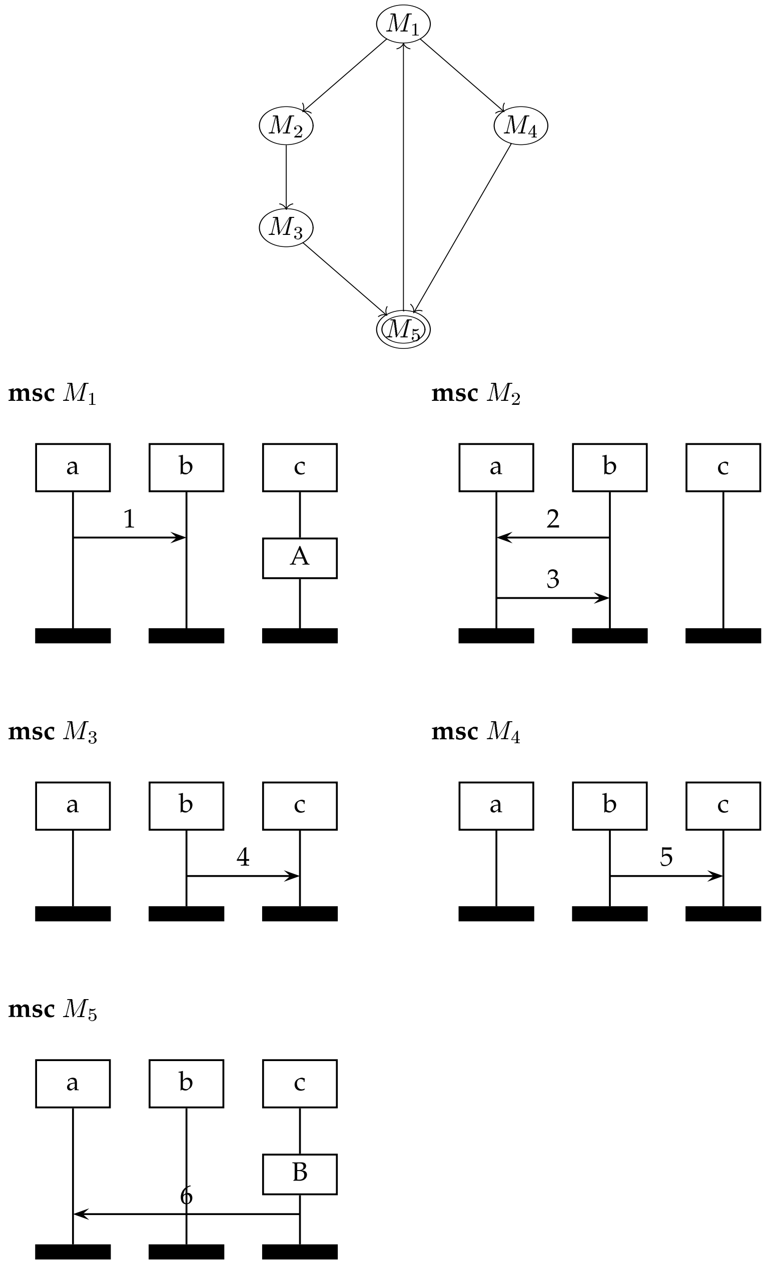
In einem stark abhängigen HMSC kann ein MSC nur dann wieder betreten werden, wenn jeder vorherige Besuch alle Ereignisse abgearbeitet hat.

Lemma 2.15 Ist H stark abhängig (Definition 2.16), so gibt es kein $(h, \varphi) \in \mathcal{K}(H)$, sodaß $h_i = h_j$ für gewisse $i < j$ mit $\varphi_i \neq \mathcal{E}(M_{h_i})$ und $\varphi_j \neq \emptyset$.

Beweis: Betrachten wir eine einfache Schleife $l := l_1 \ldots l_r l_1$ mit $l' := l_1 \ldots l_r$. Sei $K \in \mathcal{K}(\mathfrak{m}(l'))$ und sei $e_1 \in \mathcal{E}(M_{l_1})\backslash K$. Angenommen, es gebe ein $e_2 \in \mathcal{E}(M_{l_1})$, sodaß $K \cup \{e_2\} \in \mathcal{K}(\mathfrak{m}(l))$. Das geht aber nicht, weil im Sendungsgraphen von l e' und e'' existieren, sodaß $e_1 \to^* e' \to e'' \to^* e_2$ ($e' \to e''$ ist die eine k-Kante.). ⊛

Satz 2.9 Ist H stark abhängig (Definition 2.16) und erfüllt die Diamanteigenschaft (Definition 2.18), so ist $\mathcal{F}(H)$ deterministisch.

Abbildung 2.32 Ein in einen Automaten zu übersetzendes HMSC (Beispiel 2.17)



Beweis: Angenommen, $\mathcal{F}(H)$ sei nichtdeterministisch. Da nach Konstruktion keine ε-Transitionen vorliegen können, muß es einen Zustand mit verschiedenen e-Nachfolgern geben. Sei also (h, φ) eine Konfiguration mit den beiden e-Folgekonfigurationen $(h', \varphi') \neq (h'', \varphi'')$.

Seien beide Übergänge gemäß Version 2 aus Definition 2.23. Damit gibt es $i \neq j$ mit $h_i = h_j$ und $e \notin \varphi_i \cup \varphi_j$. Das führt zum Widerspruch mit Lemma 2.15.

Abbildung 2.33 k für Abbildung 2.32 (Beispiel 2.17)

i	j	$k(i,j)$
1	1	$s_1 \prec s'_1,\ A \prec A'$
1	2	$r_1 \prec s'_2,\ s_1 \prec s'_3$
1	3	$r_1 \prec s'_4$
1	4	$r_1 \prec s'_5$
1	5	$A \prec B',\ A \prec s'_6$
2	1	$r_2 \prec s'_1,\ s_3 \prec s'_1$
2	2	$s_2 \prec s'_2,\ r_3 \prec s'_2,\ r_2 \prec s'_3,\ s_3 \prec s'_3$
2	3	$s_2 \prec s'_4,\ r_3 \prec s'_4$
2	4	$s_2 \prec s'_5,\ r_3 \prec s'_5$
2	5	
3	1	$r_4 \prec A'$
3	2	$s_4 \prec s'_2$
3	3	$s_4 \prec s'_4$
3	4	$s_4 \prec s'_5$
3	5	$r_4 \prec B',\ r_4 \prec s'_6$
4	1	$r_5 \prec A'$
4	2	$s_5 \prec s'_2$
4	3	$s_5 \prec s'_4$
4	4	$s_5 \prec s'_5$
4	5	$r_5 \prec B',\ r_5 \prec s'_6$
5	1	$B \prec A',\ s_6 \prec A',\ r_6 \prec s'_1$
5	2	$r_6 \prec s'_3$
5	3	
5	4	
5	5	$B \prec B',\ B \prec s'_6,\ s_6 \prec B',\ s_6 \prec s'_6$

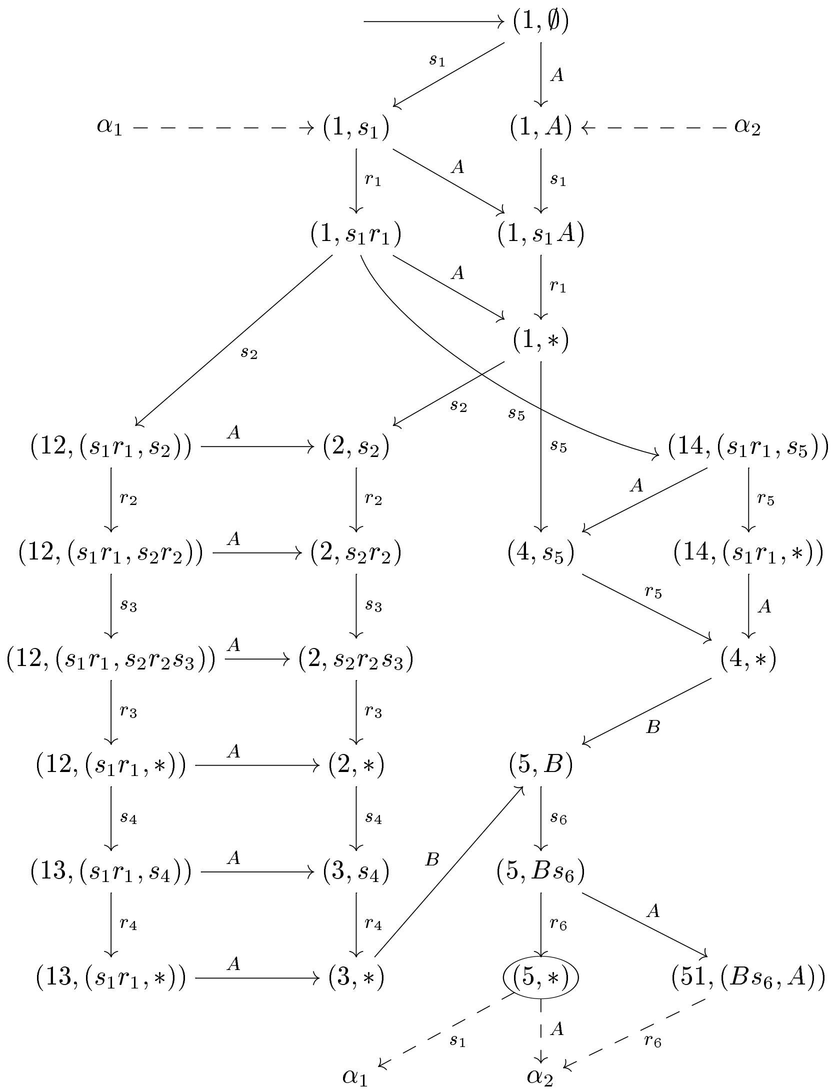
Ist ein Übergang gemäß Alternative 1 und einer gemäß 2, so folgt der Widerspruch genau analog.

Seien beide Folgekonfigurationen aus Alternative 1 konstruiert. Dann ist $h' \neq h''$. Wir haben also ein $j := h'_{|h'|} = h''_{|h''|}$ mit $e \in \mathcal{E}(M_j)$ und ein $i = h_\nu$ für ein ν, sodaß von i zwei verschiedene Pfade zu j führen (Generalvoraussetzung der Erreichbarkeit und Produktivität in Definition 2.10). Da $h' \neq h''$, gibt es ein $\mu > |h|$ mit $\mu < \min\{|h'|, |h''|\}$ und $h'_\mu \neq h''_\mu$. Da nach Konstruktion $\varphi'_\mu = \varphi''_\mu = \emptyset$, erfüllt H für diese Zweige nicht die Diamanteigenschaft, im Widerspruch zur Voraussetzung.⊞

Beispiel 2.18 Starke Abhängigkeit alleine ist nicht hinreichend für Nichtdeterminismus: Für das stark abhängige, aber nicht die Diamanteigenschaft erfüllende HMSC in Abbildung 2.35 mit $k_{\heartsuit}$ ist der entstehende Automat nichtdeterministisch.

Nachdem wir nun die Übersetzung hinter uns haben, wollen wir, wie versprochen, auf die eingangs gestellte Frage zurückkommen, warum Koregionen zu unterstützen sinnvoll ist. Offenbar kann man ein MSC mit Koregionen in mehrere übersetzen, die keine Koregionen mehr haben, indem man die jeweiligen Reihenfolgen in einzelnen MSCs sammelt. Dann könnte man für die Übersetzung in Automaten nur die einfacheren MSCs unterstützen und sich darauf zurückziehen, daß so wie ein komplexes MSC mehrere einfache erzeugt, man aus mehreren einfachen

Abbildung 2.34 Der Automat zu Abbildung 2.32 (Beispiel 2.17)

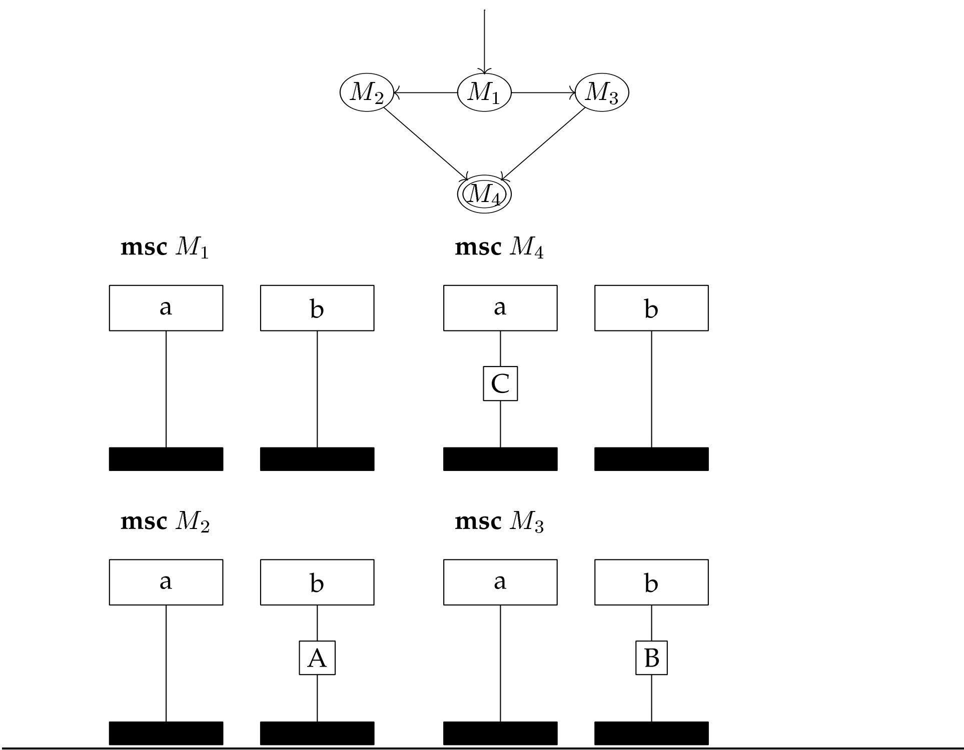


Automaten durch übliche $\cup$-Konstruktionen einen basteln kann, der dem komplexen MSC entspricht. Somit wäre die Konstruktionsvorschrift vielleicht kürzer oder einfacher.

Das stimmt. Nicht mehr sinnvoll ist eine solche Abwicklung der Alternativen jedoch bei unserem $\circ^k$-Operator für HMSCs. Da die Wahl der Reihenfolge bei nicht direkt voneinander abhängigen Ereignissen in verschiedenen MSCs die innerhalb einer Schleife liegen bei jedem Durchlauf durch die Schleife anders gewählt werden kann, erhielten wir ein unendlich großes HMSC. Daher sollten wir in diesem Falle keine Abwicklung vornehmen. Da unser Modell also die allgemeinen Ordnungsrelationen auf der Ebene der HMSCs nutzt, ist kein echter Vorteil mehr zu erzielen, verzichteten wir bei den einzelnen MSCs darauf.

Beispiel 2.19 Man betrachte das HMSC $\longrightarrow M_1 \longleftrightarrow M_2$. Angenommen, M_1

Abbildung 2.35 Ein stark abhängiges HMSC (Beispiel 2.18)



und M_2 bestünden aus den Ereignissen A bzw. B, bei $|\mathcal{P}| = 1$, die durch k nicht geordnet werden, also $k(1,2) = k(2,1) = \emptyset$. Dann sind die Linearisierungen AB, $AABB$, $ABAB$, $BBAA$ etc. erlaubt, allgemein alle $w \in \{A,B\}^+$ mit $|w|_A = |w|_B$. Das läßt sich nicht mehr mit endlich vielen MSCs in einem traditionellen HMSC mit nur asynchroner Konkatenation darstellen.

2.d Erweiterungen

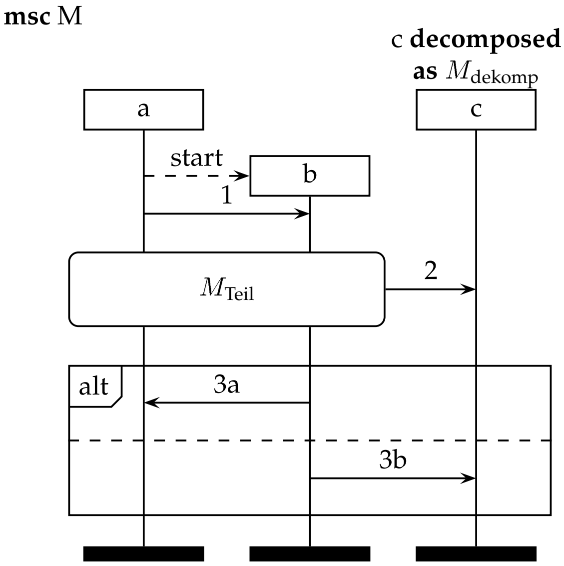
[ITU99] definiert weitere syntaktische Elemente, hauptsächlich zur verbesserten Strukturierung großer Dokumente und vieler einzelner MSCs. Es soll kurz begründet werden, warum diese hier nicht untersucht werden.

Zum einen kann man Daten und Variablen in diversen Situationen gebrauchen.[34] Allerdings gibt nicht nur [CEMR98] keine Semantik davon, sondern es ist auch, wie in [E00] dargelegt, durchaus umstritten, welche Semantik sinnvoll wäre. Weiter liegt einer der Vorteile von MSCs gerade in der Einfachheit der Darstellung. Das wird durch die Abstraktion von konkreten Daten erst ermöglicht. Daher werden wir Daten nur als Hindernis auf dem Weg zu unserem Ziel erachten und nicht näher darauf eingehen.[35]

[34] Man verschickt also nicht nur irgendwelche $\lambda \in \Lambda$, sondern z. B. in zwei Pfaden eines HMSCs x_1 und x_2, die vom empfangenden Prozeß als x empfangen werden. Dieser Prozeß kann dann x weiterverarbeiten, Verzweigungen im HMSC können in Abhängigkeit von x genommen werden usw.

[35] Diese Politik wird in der Literatur häufig verfolgt.

Abbildung 2.36 Ein MSC mit weiteren Konstruktionsmerkmalen



Weiter gibt es Konstruktionen innerhalb eines MSCs. In Abbildung 2.36 ist beispielhaft dargestellt, wie man ein MSC innerhalb eines anderen referenzieren kann (M_{Teil}). Das MSC M_{Teil} entspricht einer Auslagerung eines Teiles der Spezifikation zur Übersichtlichkeit. Ferner setzt sich der Prozeß c aus anderen zusammen, die in einem MSC M_{dekomp} definiert werden. Auch dies erhöht die Übersicht. Beides ist jedoch für uns nicht wichtig, wenn wir davon ausgehen, daß solche Definitionen flachgeklopft werden.[36] Eine Nachricht, die einen Prozeß überhaupt erst zum Leben erweckt, kann durch ein spezielles λ dargestellt werden.

Abbildung 2.36 gibt weiter ein Beispiel einer *inline-expression*. Hier wird ein Teil der Kommunikation gekennzeichnet, der zwei Alternativen beinhaltet. Gemeint ist, daß alle Spuren gültig sind, die entweder der einen Alternative folgen oder der anderen. Dies (und einige andere, ähnliche Ausdrücke) läßt sich mit HMSCs[37] ebenfalls behandeln, ohne daß die Übersichtlichkeit leidet, weswegen das hier nicht weiter betrachtet wird.

2.e Sequence Diagrams

In UML werden ebenfalls Diagramme definiert, die dazu dienen, Kommunikationen zu modellieren ([RJB99]). Die Sprache MSC muß sich daher auch daran messen lassen, inwiefern sie größere Vorteile bietet als *Sequence Diagrams* aus UML.[38]

Der UML-Standard definiert *Collaboration Diagrams (CDs)* und *Sequence Diagrams (SDs)* zur Modellierung von Kommunikationen, wobei diese verschiedene Sichten der gleichen Informationen darstellen sollen ([RJB99]). CDs konzentrieren

[36]Die Situation ist also hier anders als bei Koregionen (Seite 60).
[37]kHMSCs, vgl. Abschnitt 5.b.
[38]Dank sei MORITZ SCHNIZLER, RWTH Aachen, für Erklärungen zu SDs.

sich dabei auf die Beziehungen zwischen Objekten, wohingegen SDs die zeitliche Abfolge von Nachrichten in den Vordergrund stellen. MSCs konkurrieren mit SDs, die auch sehr ähnlich aussehen.[39] SDs kennen ebenfalls Nachrichten, die Prozesse erzeugen oder beenden, Parameter in Nachrichten, Bedingungen, Iterationen, Verzweigungen und etwas wie Zeitbeschränkungen im Sinne von Definition 4.1.

Ein wichtiges Problem bei CDs ist es, daß nur Reihenfolgen von Nachrichten modelliert werden können, nicht jedoch von einzelnen Ereignissen. Somit ist es nicht möglich, eine Folge der Art $s_1 \prec s_2 \prec r_1 \prec r_2$ zu modellieren, was in einem (sogar sinnvollen (Definition 2.1)) MSC durchaus möglich ist. Da CDs und SDs direkt ineinander übersetzbar sein sollen ([BRJ99]), müßte man auch bei SDs darauf verzichten. SDs bieten weiterhin dem Benutzer viele der anderen nicht in dieser Arbeit vorgestellten Elemente von (H)MSCs nicht.

Die MSC-Dokumente [ITU99, CEMR98] definieren eine konkrete Syntax und geben eine Semantik der meisten syntaktischen Elemente an, die für die fundamentalen Elemente der Intuition entspricht. Die semantische Diskussion von MSCs ist insgesamt weiter fortgeschritten als die über SDs. Daher gibt es Bestrebungen, für UML 2.0 einige der Charakteristika von MSCs zu übernehmen ([H01]). Eine komplette Eingliederung von MSCs ist nicht geplant, damit Umsteiger von UML 1 sich nicht allzusehr umzustellen brauchen und um die Integration in die vorhandenen weiteren UML-Diagramme nicht aufzuweichen. Im Gegensatz dazu hat sich im letzten Jahrzehnt gezeigt, daß der MSC-Standard sich schnell weiterentwickelt hat, um mit den Erweiterungen der Forschergemeinde Schritt zu halten. Man kann also davon ausgehen, daß SDs auch in Zukunft nicht offiziell soviele Modellierungsmöglichkeiten bieten werden wie MSCs.

Zusammengefasst kann man sagen, daß MSCs und SDs sich sehr ähnlich sind, aber SDs sich leichter in das sehr beliebte UML-Rahmenwerk einpassen lassen, wohingegen MSCs flexibler sind. Was praktisch wichtiger ist, muß im Einzelfall entschieden werden.

[39] Diese Wahl der Darstellung der Informationen liegt ja auch sehr nahe (vgl. auch [L78]).

Kapitel 3

Uhrenautomaten

tick tock tick tock
madness comes tonight
what's reality compared to me
PETER SPILLER/JÜRGEN JANSEN, *Timekiller*, 2001

In diesem Kapitel sollen die Grundlagen gelegt werden für die in Kapitel 4 nötige Betrachtung der Theorie der *Timed Automata (Uhrenautomaten)*. Sie wurden eingangs der 1990er Jahre von ALUR und DILL eingeführt; ein relativ frühes Papier ist [AD94]. Für eine Zusammenfassung späterer Resultate betrachte man [A99]. Sinnvolle Erweiterungen sind vor allem von der Gruppe um PETIT und BÉRARD vorgenommen worden.

Uhrenautomaten sind eine spezielle Form einer allgemeineren Klasse von Automaten, die WILKE in [W94] *Automaten auf bewerteten Wörtern* nennt.

3.a Grundlegende Definitionen

In diesem Abschnitt wird erklärt, mit welchen Objekten man sich im Zusammenhang mit Uhrenautomaten beschäftigt. Die Hauptidee ist dabei, ein Wort über einem üblichen Alphabet durch Zeitinformationen an den einzelnen Buchstaben zu erweitern. Damit soll ausgedrückt werden, daß ein bestimmter Buchstabe irgendetwas mit der Zeit zu tun hat. Hier (Dies ist auch die Interpretation, die in [AD94] nahegelegt wird.) wollen wir darunter verstehen, daß die Buchstaben gewisse *Ereignisse* sind, die zu bestimmten Zeiten *stattfinden*.

Definition 3.1 (Zeitsequenz) Ein $w \in \mathbb{P}^*$ heißt *Zeitsequenz*. $\mathbb{D}$ ist die Menge aller monoton steigenden Zeitsequenzen, also

$$\mathbb{D} := \{w \in \mathbb{P}^* : \forall i \in \{2, \ldots, |w|\}.w_i \geqslant w_{i-1}\}.$$

Für ein Alphabet Σ heißt ein $w \in (\Sigma \times \mathbb{P})^*$ *Intervallwort*. Die Menge aller Intervallwörter mit Zeitsequenzen aus $\mathbb{D}$ ist $\mathbb{W}(\Sigma)$ oder kurz $\mathbb{W}$. Man schreibt bewertete Wörter in diesem Sinne auch als $\frac{w}{x}$, wobei w die Σ^*-Komponente ist und $x \in \mathbb{D}$.

Es gibt durchaus auch andere Definitionen solcher Sequenzen, auch in diesem Zusammenhang: In [B98] werden die Sequenzen, in enger Analogie zu üblichen diskreten Wörtern, etwa als $a^1 b^{0,9} a^{2,4}$ geschrieben, um auszudrücken, daß ein a 1 Zeiteinheit lang dauert, ein nachfolgendes b 0,9 Einheiten und das letzte a 2,4 Einheiten.

Man kann Wörter auch als Funktionen $P \to \Sigma$ für ein Intervall $P \subseteq \mathbb{P}$ ansehen, mit gewissen weiteren Bedingungen.[1]

Wir wollen nun durch einen Automaten steuern, wann welche Worte erkannt werden. Dafür müssen wir Zeiten messen. Die dafür zuständigen Objekte werden *Uhren* genannt.

Definition 3.2 (Uhrstand) Sei X eine endliche Menge, deren Elemente *Uhren* heißen mögen. Ein *Uhrstand* ist eine Funktion $X \to \mathbb{P}$. Wir setzen $\mathbb{U}(X) := \mathbb{P}^X$ und einfach $\mathbb{U}$, falls X klar ist.

Wenn wir Bedingungen an die Uhren formulieren wollen, müssen wir eine Syntax dieser Bedingungen definieren und eine Semantik der Erfüllbarkeit.

Definition 3.3 (Uhrbedingung) Sei X eine Uhrmenge. Sei weiter $C \subseteq \mathbb{T}$ beliebig. Sei O eine Menge binärer Operationssymbole. Dann ist die Menge der *Uhrbedingungen über X, C und O*, $\mathfrak{B}(X, C, O)$, definiert als die kleinste Menge, die die Atome $\psi \in \{x \sim c,\ x - y \sim c : x, y \in X \wedge c \in C \wedge \sim \in O\}$ enthält und unter $\wedge$, $\vee$ und $\neg$ abgeschlossen ist. Wie üblich schreiben wir auch $\mathfrak{B}$, falls die Parameter klar sind, oder nennen nur Teile von ihnen.

Im folgenden sei O immer fest gewählt als $O := \{<, \leqslant, >, \geqslant, =, \neq\}$. Für theoretische Zwecke ist es einfacher, sich auf eine kleinere Menge zu beschränken (siehe auch Beweis zu Lemma 3.1), aber das soll im Moment nicht interessieren.

Wir interpretieren im folgenden die Symbole in der üblichen Weise und schreiben etwa $<$ sowohl für das syntaktische Objekt als auch für die üblicherweise mit dem Symbol $<$ bezeichnetet Relation auf $\mathbb{P}$.[2]

Definition 3.4 (Uhrbedingungssemantik) Seien $v \in \mathbb{U}(X)$ und $\psi \in \mathfrak{B}(X)$ beliebig. Dann gilt $v \models \psi$ genau falls

- ▷ $\psi = x \sim c$ und $v(x) \sim c$ oder
- ▷ $\psi = x - y \sim c$ und $v(x) - v(y) \sim c$ oder
- ▷ $\psi = \psi_1 \wedge \psi_2$ und $(v \models \psi_1) \wedge (v \models \psi_2)$ oder
- ▷ $\psi = \psi_1 \vee \psi_2$ und $(v \models \psi_1) \vee (v \models \psi_2)$ oder
- ▷ $\psi = \neg(\psi_1)$ und nicht $v \models \psi_1$.

Wir setzen $\mathfrak{M}(\psi) := \{v \in \mathbb{U}(X) : v \models \psi\}$.

Damit ist auch klar, daß Abkürzungen wie TRUE oder FALSE benutzbar sind, sobald $X \neq \emptyset$.

Für ein $i \in \mathbb{I}$ wird $x \in i$ verwendet, um die entsprechende Aufschlüsselung mit den O-Elementen zu vermeiden (statt $x < 4 \wedge \neg x < 1$ also $x \in [1, 4)$) (Offenbar erhöhen wir damit nicht die Ausdrucksstärke der Bedingungen.).

Uhren sollen wir auf verschiedene Werte auch setzen können. Dazu definieren wir allgemein *Updates* (Für eine allgemeinere Behandlung dieser konsultiere man [BDFP00a, B00, BDFP00c].).

[1] Rechtsseitig oder linksseitig stetig, stückweise konstant, höchstens endlich viele Unstetigkeitsstellen in jedem beschränkten Intervall (vgl. [ACM97]).

[2] Formal: Man definiere $[\![\sim]\!]$ für $\sim \in O$ in natürlicher Weise, setze $[\![\_]\!]$ an den geeigneten Stellen ein und vereinbare anschließend, $[\![\_]\!]$ dann wieder wegzulassen.

Definition 3.5 (Uhrupdate-Syntax) Sei X eine Uhrmenge.

(i) Ein *einfaches Update über* x für ein $x \in X$ ist eine Zeichenkette der Form $x \leftarrow c$ oder $x \leftarrow y$ für $y \in X$ und $c \in \mathbb{T}$.

(ii) Ein *Update über* X ist eine Menge γ von einfachen Updates, in der für jedes $x \in X$ genau ein einfaches Update über x vorkommt.

Ein Update soll eine Uhr (die auf der linke Seite des $\leftarrow$ stehende) updaten in dem Sinne, daß sie danach den rechts stehenden Wert bzw. den aktuellen Wert der rechts stehenden Uhr hat.

Definition 3.6 (Uhrupdate-Semantik) Sei $v \in \mathbb{W}(X)$.

(i) Sei γ ein einfaches Update über x. Dann ist $v' \in [\![\gamma]\!]v$ für ein $v' \in \mathbb{W}(X)$, falls gilt:

$$v'(x) = \begin{cases} c & \gamma = x \leftarrow c \\ v(y) & \gamma = x \leftarrow y. \end{cases}$$

(ii) Sei γ ein Update über X. Dann ist $[\![\gamma]\!]v$ definiert als dasjenige $v' \in \mathbb{W}(X)$, für das $v[x/v'(x)] \in [\![\gamma_x]\!]v$ für jedes $x \in X$, wobei γ_x das einfache Update über x aus γ sei. Wir schreiben dann auch einfach $v' = \gamma(v)$.

Man beachte, daß das Ergebnis eines Updates tatsächlich immer eindeutig ist.

Zu Übersetzung der MSCs können wir uns meistens auf gewisse Teilklassen der Updates beschränken, womit wir im direkten Fahrwasser von [AD94] sind.

Definition 3.7 (Rücksetzungsupdate) Sei γ ein Update über X. γ heißt *Rücksetzungsupdate*, falls alle $\gamma_x \in \gamma$ die Form $x \leftarrow 0$ oder $x \leftarrow x$ haben. Ein solches Update wird auch geschrieben als Menge aller x mit $x \leftarrow 0 \in \gamma$.

Nun wird die Syntax des eigentlichen Uhrenautomaten beschrieben. Es handelt sich dabei im wesentlichen um einen endlichen Automaten. Wir haben endlich viele Uhren, um endlich viele verschiedene Abstände messen zu können.[3] Die Bedingungen an die Uhren und die Fortschreibung werden auf natürliche Weise in die Transitionen aufgenommen.

Definition 3.8 (Uhrenautomat) Ein *Uhrenautomat* ist ein 6-Tupel

$$\langle Q, X, \Sigma, \delta, Q_0, Q_* \rangle,$$

wobei gilt:

- ▹ Q ist eine endliche Menge von *Zuständen*;
- ▹ X ist eine endliche Menge von *Uhren*;
- ▹ Σ ist ein *Eingabealphabet*;

[3] Eine mögliche Alternative wäre es, nur eine Uhr mitlaufen zu lassen, die die Gesamtzeit bestimmt, und sich dann verschiedene Zeitpunkte in Registern zu merken. Das ist vielleicht sogar ein intuitiv einsichtigerer Zugang (eine Stoppuhr mit Notizblatt), aber er ist äquivalent zum hier bevorzugten (mehrere Stoppuhren).

- $\triangleright$ $Q_0 \subseteq Q$ mit $Q_0 \neq \emptyset$ ist eine Menge von *Anfangszuständen*;
- $\triangleright$ $Q_* \subseteq Q$ ist eine Menge von *Endzuständen*;
- $\triangleright$ δ ist eine *Transitionsmenge*, wobei die $e \in \delta$ die Form $e = (s, \sigma, \psi, \gamma, s')$ haben, sodaß
  - $s \in Q$ der *Quellzustand*,
  - $\sigma \in \Sigma_\varepsilon$ der *gelesene Buchstabe der Eingabe*,
  - $\psi \in \mathfrak{B}(X)$ die *Transitions-Bedingung (Wächter)*,
  - γ das *Transitions-Update* gemäß Definition 3.5 und
  - $s' \in Q$ der *Zielzustand*

  sind.

Zur Vereinfachung setzen wir dabei C in $\mathfrak{B}(X, C, O)$ als genau die Menge der Konstanten, die in den Bedingungen vorkommt.

In einem Lauf durch einen Automaten müssen wir nun noch voraussetzen, daß die Wächter erfüllt sind und die Updates durchgeführt werden.

Für die Semantik ist jetzt nur noch zu untersuchen, welche Worte erkannt werden. Wir folgen dabei den üblichen Semantiken für endliche Automaten auf endlichen Wörtern (vgl. [HU00]): Ein Uhrenautomat A erkennt ein Wort $\genfrac{}{}{0pt}{}{w}{x}$, falls es einen Lauf für $\genfrac{}{}{0pt}{}{w}{x}$ gibt, der in einem $q \in Q_*$ endet.

Definition 3.9 (Semantik) Sei A ein Uhrenautomat. Sei $\genfrac{}{}{0pt}{}{w}{t} \in \mathbb{W}(\Sigma)$.

(a) $\genfrac{}{}{0pt}{}{w}{t}$ wird aus dem Zustand q und dem Uhrstand v erkannt, falls eines gilt von:

  (1) $\genfrac{}{}{0pt}{}{w}{t} = \varepsilon$ und $q \in Q_*$.

  (2) Es gibt eine Transition $(q, \varepsilon, \psi, \gamma, q')$ und ein $s \in \mathbb{P}$, sodaß $(v + s) \models \psi$ und $\genfrac{}{}{0pt}{}{w}{t-s}$ aus q' und $[\![\gamma]\!](v + s)$ erkannt wird.

  (3) $\genfrac{}{}{0pt}{}{w}{t} = \genfrac{}{}{0pt}{}{w_1}{s}\genfrac{}{}{0pt}{}{w'}{t'}$, und es gibt eine Transition $(q, w_1, \psi, \gamma, q')$, sodaß $(v + s) \models \psi$ und $\genfrac{}{}{0pt}{}{w'}{t'-s}$ aus q' und $[\![\gamma]\!](v + s)$ erkannt wird.

(b) A erkennt $\genfrac{}{}{0pt}{}{w}{t}$, falls $\genfrac{}{}{0pt}{}{w}{t}$ aus einem Zustand $q \in Q_0$ und dem Uhrstand $\lambda\_.0$ erkannt wird. $\mathfrak{L}(A)$ ist die Menge aller erkannten Worte.

(2) und (3) sind Transitionen, die in einen anderen Zustand übergehen. Man läßt s Sekunden in q verstreichen und geht dann ohne Zeitverzug in q' über.

Mit diesen Bezeichnungen schreibt man auch z. B. $q \rightarrow^{(w_1, t_1)} q'$, falls es entsprechende Transitionen gibt.

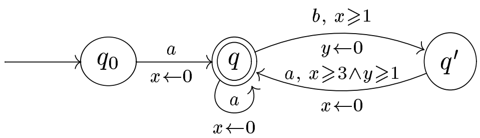
Die üblichen Termini für diskrete Automaten (beispielsweise *erreichbare* oder *produktive* Zustände) übertragen sich sinngemäß.

Auch andere Erkennungsmechanismen sind denkbar und werden in der Literatur behandelt, führen aber vielfach nicht zu Änderungen in der Erkennungsstärke. Insbesondere ist aber eine Erkennung unendlicher Wörter mit den üblichen Semantiken möglich.

Beispiel 3.1 Wir betrachten als Beispiel einen Automaten in graphischer Darstellung. Die Anfangszustände (aus Q_0) werden hierbei mit eingehendem Pfeil gezeichnet und Endzustände (aus Q_*) als Doppelkreise. Auf den Transitionen wird auf einer Seite das Element aus Σ_ε und die Bedingung an die Bewertung genannt (falls sie nicht TRUE ist, dann läßt man sie weg), auf der anderen werden die Rücksetzungen notiert. Offenbar erkennt der Automat die Sprache

$$\{ \genfrac{}{}{0pt}{}{w}{t} : w = a^+(ba^+)^* \wedge \forall w_i = b.(t_i \geqslant t_{i-1} + 1 \wedge t_{i+1} \geqslant \max\{t_i + 1, t_{i-1} + 3\})\}.$$

Abbildung 3.1 Ein erster Uhrenautomat (Beispiel 3.1)



3.b Einige Resultate

In diesem Abschnitt werden einige Resultate zitiert, die uns helfen werden, die in Kapitel 4 zu definierende Übersetzung richtig einzuordnen, oder an irgendeiner Stelle mal gebraucht werden.

Wir können nun in einem gewissen Sinne in Uhrenautomaten auf die Uhrvergleiche in den Beschränkungen verzichten. So ein Automat heißt *diagonalfrei*. Dieses *Folklore-Theorem* ([BDFP00a]) wird vielfach zitiert, aber selten bewiesen. Der hier gegebene Beweis benutzt die Idee aus [BDGP98].

Lemma 3.1 Zu jedem Uhrenautomaten A gibt es ein A' mit $\mathcal{L}(A) = \mathcal{L}(A')$, wobei A' keine Bedingungen der Form $x - y \sim c$ beinhaltet.

Beweis: Dieser Beweis beruht auf der Tatsache, daß $v \models \psi$ für ein ψ der Diagonalform immer $v + \alpha \models \psi$ für $\alpha > 0$ impliziert. Wir erweitern die Zustandsmenge Q also um einen Indikator, ob ein Uhrstand eine Diagonalbedingung erfüllt. Diesen Indikator müssen wir nur bei Updates aktualisieren. Wegen der Endlichkeit der Bedingungen haben wir weiterhin nur endlich viele Zustände.

Zur Vereinfachung nehmen wir an, daß alle Diagonalbedingungen nur $<$ benutzen. Das ist offenbar keine Beschränkung der Allgemeinheit.[4]

Sei $\langle Q, X, \Sigma, \delta, Q_0, Q_* \rangle$ ein Uhrenautomat mit Diagonalbedingungen, in denen als Konstanten $c_1, \ldots, c_r$ vorkommen. Sei $e = |X|^2 \cdot r$. Sei $B = \{\text{TRUE}, \text{FALSE}\}^e$. Wir ordnen nun jeder Diagonalbedingung $x - y < c_\nu$ eindeutig ein $e' \in \{1, \ldots, e\}$ zu und nennen sie $\psi_{e'}$.

Wir übersetzen diesen Automaten nun in $\langle Q', X, \Sigma, \delta', Q'_0, Q'_* \rangle$, wobei gilt:

- $Q' = Q \times B$, $Q'_* = Q_* \times B$,

[4] $a = b \leftrightarrow (\neg a < b \wedge \neg b < a)$ etc.

- ▷ $Q'_0 = Q \times \{(\lambda\_.0 \models \psi_1, \dots, \lambda\_.0 \models \psi_e)\}$,
- ▷ δ' enthält für jede Transition $(s, \xi, \psi, \gamma, s') \in \delta$ die folgenden Transitionen:
  - ▶ Sei ψ' so wie ψ, aber ohne Diagonalbedingungen. Für ein $v \in \mathfrak{M}(\psi)$ seien $r_1 = (v \models \psi_1, \dots, v \models \psi_e) \in B$, $r_2 = (\gamma(v) \models \psi_1, \dots, \gamma(v) \models \psi_e) \in B$. Dann enthält δ' die Transition $((s, r_1), \xi, \psi', \gamma, (s', r_2))$.
  - ▶ Ansonsten enthält δ' nichts.

Somit werden die Indikatoren mit den richtigen Werten initialisiert und bei jeder Transition geeignet fortgeschrieben. ⊛

Im weiteren Verlauf dieser Arbeit werden meistens direkt diagonalfreie Uhrenautomaten besprochen.

Es stellt sich nun die Frage, wie man allgemein Uhrenautomaten analysiert. Ein sehr beliebtes Hilfsmittel ist der *Region-Automat*, der ein endliches System ist, welches für gewisse Fragestellungen äquivalent zum Uhrenautomaten ist. Dazu untersuchen wir, wann Uhrstände äquivalent sind. Wir beachten zunächst, daß in einem Automaten offenbar nur gewisse Intervalle der Uhrstände von Bedeutung sind. Insbesondere steht das Ergebnis aller Vergleiche mit einer Uhr fest, wenn sie über den höchsten zum Vergleich herangezogenen Wert hinaus ist.[5] Weiter ist für einen Vergleich mit natürlichen Zahlen nur von Bedeutung, in welchem $\mathbb{N}$-Gitter des $\mathbb{R}^n$ wir uns befinden, und in der Nähe welcher Kanten (wird gleich klarer werden).

Definition 3.10 (Uhrstandsäquivalenz) Seien $v, v' \in \mathbb{V}(X)$. Sei $(c_x)_{x \in X}$ eine beliebige Familie reeller Zahlen.

Wir nennen v und v' *äquivalent bzgl.* (c_x) ($v \approx_{(c_x)} v'$) (oder, falls (c_x) klar ist, *äquivalent* ($v \approx v'$)), falls folgende Bedingungen erfüllt sind:

(i) Für jedes $x \in X$ ist $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ oder $v(x), v'(x) > c_x$.

(ii) Für alle $x, y \in X$ mit $v(x) \leqslant c_x \wedge v(y) \leqslant c_y$ gilt $\mathfrak{f}(v(x)) \leqslant \mathfrak{f}(v(y))$ genau falls $\mathfrak{f}(v'(x)) \leqslant \mathfrak{f}(v'(y))$.

(iii) Für alle $x \in X$ mit $v(x) \leqslant c_x$ gilt $\mathfrak{f}(v(x)) = 0$ genau falls $\mathfrak{f}(v'(x)) = 0$.

Wir werden im folgenden (c_x) immer als die Familie der für die einzelnen Uhren x maximalen in einem Automaten vorkommenden Konstanten wählen. Dabei sei vorausgesetzt, daß alle diese Konstanten ganzzahlig seien. Das ist keine Beschränkung, denn da die Zeiteinheit ohnehin abstrakt ist und alle Konstanten rational sind, kann man durch Multiplikation eine äquivalente Menge an Beschränkungen mit ganzzahligen Konstanten berechnen.

Definition 3.11 (Region-Automat) Sei A ein Uhrenautomat (Definition 3.8) ohne ε-Transitionen.[6] Wir modifizieren E dergestalt, daß alle Konstanten diskretisiert werden (geht, da das Universum $\mathbb{T} \subseteq \mathbb{Q}$ ist). Wir definieren dann das unendliche Transitionssystem

$$\mathfrak{T} := \left\langle Q \times \mathbb{V}(X), \Sigma_\varepsilon, Q_0 \times \{\lambda\_.0\}, \delta', Q_* \times \mathbb{V}(X) \right\rangle$$

[5] Man erinnere, daß wir uns auf diagonalfreie Automaten beschränken.

[6] Das ist allein eine Einschränkungn aus technischen Gründen, um die Definition einfacher zu gestalten. In Kapitel 4 werden nur solche Automaten entstehen.

durch

$$\begin{aligned}\delta' \quad &:= \quad \{((q_1, v_1), \xi, (q_2, v_2)) : (q_1, \xi, \psi, \gamma, q_2) \in \delta \wedge v_2 \models \psi \wedge v_2 = [\![\gamma]\!] v_1\} \\ &\cup \quad \{((q_1, v_1), \varepsilon, (q_1, v_1 + r)) : r \in \mathbb{P}\}.\end{aligned}$$

Sei

$$\delta'' = \{((q_1, v_1), \xi, (q_2, v_2)) : ((q_1, v_1), \varepsilon, (q_1, v_1 + r)), ((q_1, v_1 + r), \xi, (q_2, v_2)) \in \delta'\}.$$

Dann ist der *Region-Automat von A*

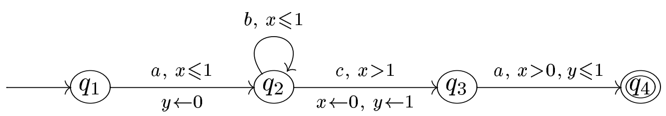
$$\langle Q \times (\mathbb{V}(X))/_{\approx}, \Sigma_\varepsilon, Q_0 \times (\{\lambda\_.0\})/_{\approx}, \delta''/_{\approx}, Q_* \times (\mathbb{V}(X))/_{\approx} \rangle.$$

Somit ist der Uhrenautomat in einen üblichen endlichen Automaten übersetzt worden, der die Grundlage weiterer Analysen bildet. Dabei wurden Zeitübergänge als ε-Übergänge ohne Zustandsveränderung modelliert. Anschließend werden ebenjene Übergänge in δ' übersprungen. Faßt man anschließend die Zustände zu den $\approx$-Äquivalenzklassen zusammen, so erhält man den angegebenen endlichen Automaten. Eine Region läßt sich dabei leicht in einer Intervallschreibweise definieren (Beispiel 3.2).

Der Uhrenautomat ist zu seinem Regionenautomaten im Prinzip äquivalent. Als wichtigstes sei hier die Erreichbarkeitsanalyse genannt: Ein Zustand q des Uhrenautomaten ist genau dann erreichbar, wenn ein Zustand des Region-Automatens mit q in der Zustandskomponente erreichbar ist ([A99]).

Beispiel 3.2 Die erreichbaren Zustände des Region-Automatens zum Uhrenautomat aus Abbildung 3.2 sind in Abbildung 3.3 dargestellt. Man kann daran erkennen, daß Zustand q_4 unerreichbar ist.

Abbildung 3.2 Ein zu untersuchender Uhrenautomat (Beispiel 3.2)



Ein wichtiges negatives Resultat müssen wir noch zitieren ([BDGP98]).

Satz 3.1 Die Sprache $S := \{\,{}^{w}_{t} \in \mathbb{W}(\{a\}) : t \in (2\mathbb{N})^*\}$ ist nicht durch einen diagonalfreien Uhrenautomaten ohne ε-Transitionen erkennbar, aber durch einen mit ε-Transitionen.

Beweis: Angenommen, es ginge mit einem solchen Automaten. Dann sei α die größte in ihm auftretende Zahl in den Uhrbedingungen der Transitionen. Für ein $\alpha' \in 2\mathbb{N}$ mit $\alpha' > \alpha$ ist ${}^{a}_{\alpha'} \in S$, weswegen es eine Transition (s, a, ψ, γ, s') geben muß mit $s \in Q_0$, $s' \in Q_*$ und $\lambda\_.\alpha' \models \psi$. Offensichtlich kann ψ aber nicht zwischen α' und $\alpha' + 1 \notin 2\mathbb{N}$ unterscheiden, da die Konstanten in ψ nach oben durch $\alpha < \alpha'$ beschränkt sind. Somit erkennt der Automat auch ${}^{a}_{\alpha'+1} \notin S$.

Ein erkennender Automat mit ε-Transition ist in Abbildung 3.4 angegeben. ⊛

Damit verbunden ist das folgende, bedauernswerte Resultat.

Abbildung 3.3 Der Region-Automat zu Abbildung 3.2 (Beispiel 3.2)

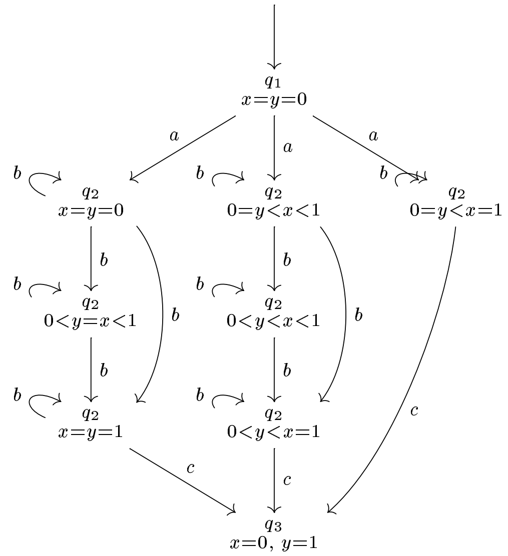
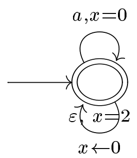


Abbildung 3.4 Zum Beweis von Satz 3.1



Satz 3.2 Die Klasse der deterministischen Uhrenautomaten ist echt erkennungsschwächer als die Klasse der nichtdeterministischen Uhrenautomaten.

Beweis: [AD94]. ⊛

Der übliche Weg, Sprachinklusion nachzuweisen (Determinisierung, Komplementierung, Schnitt, Leerheitstest) scheitert also bei nichtdeterministischen Automaten am ersten Schritt. Tatsächlich ist die Frage unentscheidbar. In [AFH98] werden gewisse Klassen deterministischer Automaten vorgestellt, die wir im nächsten Kapitel wiedertreffen werden.

Kapitel 4

Quantitative Zeitmessung in MSCs

Das also war des Pudels Kern!
JOHANN WOLFGANG VON GOETHE, *Faust*, 1808

Betrachten wir ein normales MSC (Definition 2.1). Wir wollen, wie gesagt, gewisse zusätzliche quantitative Informationen über die Ereignisse betrachten. Da die Semantik eines MSCs ja als eine diskrete Sprache erklärt ist (Definition 2.4), können wir auf einfache Weise diese im Sinne der Zeitsequenzen (Definition 3.1) um die Zusatzinformationen ergänzen. Somit würden wir den einzelnen Ereignissen die Zeiten zuordnen, an denen sie stattfinden.

Ein MSC kann als sehr einfaches HMSC angesehen werden. Daher werden wir uns an einigen Stellen, auch im nächsten Kapitel, darauf beschränken, HMSCs zu betrachten.

4.a MSCs mit Zeitbeschränkungen

Zunächst betrachten wir einzelne MSCs. Ähnlich wie in [AHP96] lassen wir Beschränkungen zu der Art, daß ein Ereignis e mindestens t_1, höchstens aber t_2 Zeiteinheiten[1] nach Ereignis e' stattfinden darf. Formal erweitern wir die Definition eines MSCs also um eine Abbildung, die (e, e') ein Intervall $[t_1, t_2]$ zuordnet.[2]

Definition 4.1 (MSC mit Timing)

[1]In dieser Arbeit wird auch der Begriff *Sekunden* benutzt, da er kürzer ist und dennoch jeder weiß, daß er in einem abstrakten Sinne gebraucht wird.

[2]Der Vollständigkeit wegen sei darauf aufmerksam gemacht, daß relativistische Effekte hier nicht auf die Zeitmessung einwirken sollten und Ereignisse zu genau definierten Zeitpunkten stattfinden. Insbesondere können wir also von *Nachzeitigkeit* im üblichen Sinne sprechen und haben die Sicherheit, daß, wenn die Ereignisse a und b auf einem Prozeß 4 Sekunden auseinanderliegen, sie dies auch auf einem anderen Prozeß tun.[3]

Da MSCs ohnehin dafür da sind, eine *einfache, intuitive* Modellierung zu ermöglichen, erscheint dies im höchsten Maße angemessen. Für eine weitergehende Rechtfertigung dieser Sichtweise sei auf [L78, W89] verwiesen.

[3]Das heißt nicht, daß alle Prozesse Zugriff auf eine globale Uhr hätten, sondern nur, daß ein Beobachter von außen eine überall gültige Uhr hat.

(a) Ein *MSC mit Timing* ist ein 8-Tupel

$$M := \langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell, \tau \rangle,$$

wobei $\mathfrak{u}(M) := \langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell \rangle$ ein MSC ist (Definition 2.1) und $\tau : \prec \to \mathbb{I}$. Die Menge aller solcher MSCs ist $\mathfrak{M}_\mathfrak{t}(\mathcal{P}, \Lambda)$, gegebenenfalls mit weniger Parametern, wie üblich.

(b) Die *Semantik von M* ist

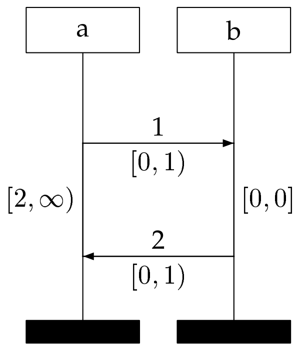
$$[\![M;\pi]\!] := \{ {}^{\pi(w)}_{\;\;x} : w \in \mathfrak{l}(\prec) \wedge {}^{w}_{x} \in \mathbb{W}(\mathcal{E}) \wedge \forall (w_i, w_j) \in \prec . x_j - x_i \in \tau(w_i, w_j) \}.$$

Statt $[\![M;\mathrm{id}]\!]$ schreibt man auch einfach $[\![M]\!]$.

(c) Für ein M wie unter (a) heißt $\mathfrak{u}(M)$ das *ungezeitete MSC zu M*.

Wir haben nun eine gänzlich andere Situation als bei normalen MSCs, da ein MSC mit Timing vielleicht überhaupt nichts bedeutet. Durch die Maßgabe, daß $\prec$ azyklisch ist (Definition 2.1), ist $[\![M;\pi]\!] \neq \emptyset$ für alle $M \in \mathfrak{M}$.[4] Für $M \in \mathfrak{M}_\mathfrak{t}$ kann aber $[\![M;\pi]\!] = \emptyset$ passieren: Das MSC in Abbildung 4.1 erlaubt keine zeitliche Abfolge der Ereignisse, die die Beschränkungen erfüllt, wie man sich leicht überlegen kann. (Für die graphische Syntax sei vereinbart, daß $\tau(e_1, e_2) = \mathbb{T}$ nicht gesondert aufgeführt wird und ansonsten $\tau(e_1, e_2)$ an die Linie von e_1 nach e_2 geschrieben wird – Als e_1 und e_2 mögen gelten die jeweils nächstmöglichen Punkte.)

Abbildung 4.1 Ein unerfüllbares MSC mit Timing



Definition 4.2 (Zeiterfüllbarkeit) Ein $M \in \mathfrak{M}_\mathfrak{t}$ heißt *zeiterfüllbar*, falls $[\![M]\!] \neq \emptyset$.

Betrachten wir zwei Ereignisse e_1, e_2 mit $\tau(e_1, e_2) = [a, b]$, die zu den Zeitpunkten t_1 bzw. t_2 beobachtet werden.[5] Gefordert ist also $t_2 - t_1 \leqslant b \wedge t_2 - t_1 \geqslant a$. Das ist nichts anderes als eine klassische Aufgabe der Linearen Programmierung, nämlich die Lösung der Gleichung $\left(\begin{smallmatrix} -1 & 1 \\ 1 & -1 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} t_1 \\ t_2 \end{smallmatrix}\right) \leqslant \left(\begin{smallmatrix} b \\ -a \end{smallmatrix}\right)$. Dies kann durch Berechnung von kürzesten Wegen in einem Graphen durchgeführt werden, wobei wir den Graphen durch das MSC bereits gegeben haben.

Um offene von abgeschlossenen Intervallschranken zu unterscheiden, müssen wir gewisse Tags einführen.

[4] Bei einem leeren MSC ist $[\![M;\pi]\!] = \{\varepsilon\}$.

[5] Wenn nicht anderes im Zusammenhang definiert wird, seien t_α immer die Beobachtungszeitpunkte von e_α.

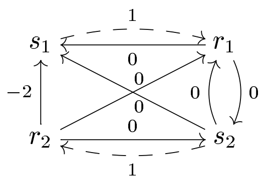
Definition 4.3 (Zeitgraph) Sei M ein MSC mit Timing. Wir definieren den *Zeitgraphen von M* als gewichteten Graphen $\langle \mathcal{E} \uplus \{\alpha\}, \longrightarrow, \rho \rangle$, und dafür die Markierungs-Funktion $b : \longrightarrow \to \{0, 1\}$ wie folgt:

- ▹ Sei $I = \tau(e, e')$ für $(e, e') \in \prec$. Dann gilt:
  - ▶ $(e', e) \in \longrightarrow$ mit $\rho(e', e) = -\inf I$. Falls $\inf I \in I$, so ist $b(e', e) = 0$, sonst $b(e', e) = 1$.
  - ▶ Falls $\sup I \in \mathbb{R}$, so ist $(e, e') \in \longrightarrow$ mit $\rho(e, e') = \sup I$, wobei $b(e, e') = 0$ falls $\sup I \in I$ und $b(e, e') = 1$ sonst.
  - ▶ Für alle $e \in \mathcal{E}$ ist $\alpha \to e$ mit $b(\alpha, e) = 1$ und $\rho(\alpha, e) = 0$.
- ▹ Ansonsten enthält $\longrightarrow$ nichts.

Dieser Graph entspricht etwa dem *constraint graph* aus [CLR90], außer, daß wir auch echte Ungleichungen betrachten, für welche wir die b-Markierung einführen.

Beispiel 4.1 Als Beispiel erhalten wir für das MSC aus Abbildung 4.1 den Graphen aus Abbildung 4.2 (ohne α dargestellt), wobei markierte Kanten gestrichelt sind. Man beachte die transitiven Kanten, die aus der Halbordnungseigenschaft resultieren.

Abbildung 4.2 Der Zeitgraph zu Abbildung 4.1 (Beispiel 4.1)



Satz 4.1 ([AHP96]) Ein MSC ist genau dann zeiterfüllbar (Definition 4.2), falls gilt:

- ▹ Der Zeitgraph enthält keine Schleife mit negativen Kosten.
- ▹ Jede Schleife mit den Kosten 0 enthält keine b-markierte Kante.

Dann läßt sich, falls keine Kante markiert ist, eine erfüllende Belegung in der Zeit $\mathfrak{O}(|\mathcal{E}| \cdot (|\mathcal{E}| + |\prec|))$ finden.

Beweis: Angenommen, es gebe eine Schleife mit negativen Kosten. Dann gibt es auch eine einfache Schleife $e_1 e_2 \ldots e_n e_1$ mit negativen Kosten ξ. Nach Konstruktion folgt aus $e_1 \longrightarrow^{\rho_1} e_2$, daß $t_2 - t_1 \leqslant \rho_1$ gelten muß. Mit $e_2 \longrightarrow^{\rho_2} e_3$ erhalten wir daraus $t_3 - t_1 \leqslant \rho_1 + \rho_2$. Gesamt ergibt sich $t_1 - t_1 \leqslant \xi < 0$, ein Widerspruch. Falls es eine Schleife mit den Kosten $\xi = 0$ gibt, so ist in derselben Situation $t_1 - t_1 \leqslant 0$. Wegen der Markierung ist aber eine der Ungleichungen scharf, was zu der scharfen Ungleichnung $t_1 - t_1 < 0$ führt. In jedem Fall erhalten wir eine unerfüllbare Forderung, womit M nicht zeiterfüllbar ist.

Zur Umkehrung: Sei keine Kante markiert (seien also alle Intervalle abgeschlossen). Für jedes $e_i \in \mathcal{E}$ setzen wir $t'_i := \mathfrak{d}(\alpha, e_i)$. Dies existiert, da kein Zykel negativ ist, und kann in der angegebenen Zeit berechnet werden ([CLR90]). Betrachten wir eine beliebige Kante $(e_1, e_2) \in \longrightarrow$, so ist offenbar $\mathfrak{d}(\alpha, e_2) \leqslant \mathfrak{d}(\alpha, e_1) + \rho(e_1, e_2)$, d. h. $t'_2 - t'_1 \leqslant \rho(e_1, e_2)$. Das ist genau das, was wir haben wollen, da keine Kante auf Zykeln markiert ist, also nie bei einer echten Ungleichung die Begrenzung erreicht wird. Damit sind die relativen Bedingungen mit den Zeitpunkten t'_i erfüllt. Eine Normalisierung $t_i := t'_i - \min_{e_j \in \mathcal{P}} t'_j$ ergibt damit gültige, erfüllende Zeitstempel. ⊛

In [AHP96] werden auch markierte Kanten näher untersucht.

Beispiel 4.2 Da der Graph in Abbildung 4.2 eine Schleife mit 0-Kosten und einer markierten Kante hat, ist das MSC nicht zeiterfüllbar, was wir ja bereits festgestellt haben. Wären dagegen die Beschränkungen der Nachrichtenlaufzeiten der Art $\tau(s_1, r_1) = \tau(s_2, r_2) = [0, 1]$, so gäbe es eine erfüllende Belegung: die kürzesten Wege würden berechnet zu $\mathfrak{d}(\alpha, s_1) = -2$, $\mathfrak{d}(\alpha, r_1) = -1$, $\mathfrak{d}(\alpha, s_2) = -1$, $\mathfrak{d}(\alpha, r_2) = 0$, was offenbar die Einschränkungen erfüllt. Um nun auf Zeiten aus $\mathbb{T}$ zu kommen, subtrahiert man einfach $-(-2)$ von allen und erhält eine sinnvolle Zeitbelegung.

Zur Übung stellen wir noch kurz fest, daß man gar keine Probleme hat, falls wir uns nur auf Einschränkungen der Nachrichtenlaufzeit beschränken und Nachrichten nicht verschachtelt werden müssen.

Lemma 4.1 Sei $M \in \mathfrak{M}_\mathrm{t}$, sodaß gilt:

(i) $\tau(e_1, e_2) = \mathbb{T}$ für alle $(e_1, e_2) \in \prec \setminus m$ und

(ii) $\mathfrak{u}(M)$ ist entschachtelbar (Lemma 2.2).

Dann ist M zeiterfüllbar und eine erfüllende Linearisierung kann in Polynomialzeit gefunden werden.

Beweis: Sei $w \in \mathfrak{l}(\prec)$, sodaß falls $w_i = s \wedge w_j = r \wedge m(s) = r$ stets $w_\nu \in \mathcal{A}$ für jedes $\nu \in \{i+1, \ldots, j-1\}$. So eine Linearisierung existiert nach Voraussetzung (ii). Wir setzen nun $t_1 := 0$ und für jedes $i \in \{2, \ldots, |w|\}$

$$t_i := \begin{cases} t_{i-1} & w_i \notin \mathcal{R} \\ t_j + \frac{1}{2}(\sup \tau(w_j, w_i) + \inf \tau(w_j, w_i)) & w_i \in \mathcal{R} \wedge m(w_j) = e. \end{cases}$$

Nach (i) sind damit alle Zeitbedingungen erfüllt.

Die Komplexitätsabschätzung folgt aus dem Beweis von Lemma 2.2: Die Hauptschleife wird höchstens $|\mathcal{E}|$-mal durchlaufen, Schritt (4b) hat eine Schleife, die durch $|\mathcal{R}|$ beschränkt ist. ⊛

Kurz soll noch die Konkatenation von einzelnen MSCs erklärt werden, obwohl die HMSC-Definition nicht darauf aufbaut.

Definition 4.4 (Konkatenation) Seien $M', M'' \in \mathfrak{M}_\mathrm{t}$ mit $\mathcal{P}' = \mathcal{P}''$. Seien $k \subseteq \mathcal{E}' \times \mathcal{E}''$ und $\tau_k : k \to \mathbb{I}$. Dann ist die *sequentielle Komposition von M' mit M'' unter k und τ_k*, $M' \circ^{(k,\tau_k)} M''$, definiert als das MSC mit Timing M mit den folgenden Parametern:

- ▷ $\mathcal{P} := \mathcal{P}' = \mathcal{P}''$, $\mathcal{E} := \mathcal{E}' \uplus \mathcal{E}''$, $\Lambda := \Lambda' \cup \Lambda''$;
- ▷ $o := o' \vee o''$, $\ell := \ell' \vee \ell''$, $m := m' \vee m''$;
- ▷ $\prec := (\prec' \cup \prec'' \cup k)^+$;
- ▷ $\tau := \lambda(x, y).((\tau' \vee \tau'' \vee \tau_k)(x, y) = \bot ? \mathbb{T} : (\tau' \vee \tau'' \vee \tau_k)(x, y))$.

4.b HMSCs mit Zeitbeschränkungen

Um auch MSC-übergreifende Beschränkungen modellieren zu können, erweist es sich als hilfreich, die zunächst sicherlich als kompliziert und vielleicht auch als unnötig erschienene Definition 2.10 der Inter-MSC-Präzedenz k genauso mit Zeitbeschränkungen zu annotieren wie die übliche Intra-MSC-Präzedenz $\prec$ (Definition 4.1).

4.b.i Formalisierung

Definition 4.5 (HMSC mit Timing) Sei $(N, \mathcal{P}, \Lambda, (\mathfrak{u}(M_i))_{i=1}^{|N|}, E, k, i_0, N_*) \in \mathfrak{H}$ (Definition 2.10) für gewisse $M_i \in \mathfrak{M}_\mathrm{t}$. Jedes M_i sei zeiterfüllbar. Dann heißt

$$H := (N, \mathcal{P}, \Lambda, (M_i)_{i=1}^{|N|}, E, k, i_0, N_*, \tau)$$

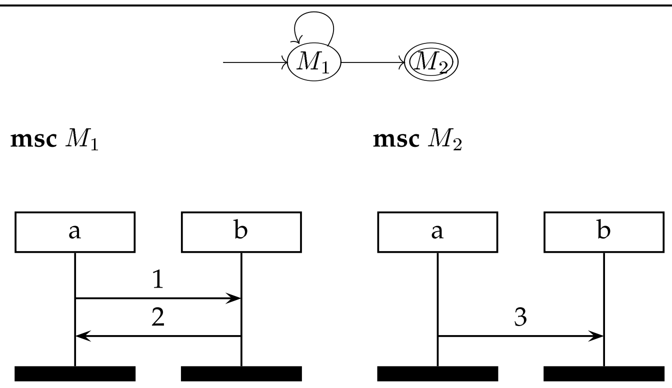
HMSC mit Timing, falls $\tau : \mathrm{range}(k) \to \mathbb{I}$. Die Menge aller solchen H ist $\mathfrak{H}_\mathrm{t}(\mathcal{P}, \Lambda)$ (mit den üblichen Kurzschreibweisen). $\mathfrak{u}(H) := (N, \mathcal{P}, \Lambda, (\mathfrak{u}(M_i))_{i=1}^{|N|}, E, k, i_0, N_*)$ heißt das *ungezeitete HMSC*.

Wenn im folgenden einfach von einem $H \in \mathfrak{H}_\mathrm{t}$ die Rede ist, so vereinbaren wir, immer die Bezeichnungen dieser Definition zu unterlegen, sofern nichts anderes gesagt wird. Entsprechend ist die Zeitbeschränkung eines $H' \in \mathfrak{H}_\mathrm{t}$ dann τ' usw.

Nun sind Läufe und Semantik eines HMSCs mit Timing zu definieren – Das geschieht in völliger Analogie zu den Definitionen 2.12 und 2.13. Aber dabei sind drei verschiedene mögliche, jede für sich sinnvolle, Definitionen der Semantik zu unterscheiden (Das Problem wurde teilweise bereits in [BAL97c] angesprochen.). Wir verlassen hier also den Bereich des intuitiv Erkennbaren!

Beispiel 4.3 Man betrachte das HMSC aus Abbildung 4.3. Angenommen, wir haben die Maßgabe $\tau(s_1, s_3) \in [1, 2]$.

Abbildung 4.3 Ein HMSC mit Semantikproblem (Beispiel 4.3)



(1) Wollen wir dann für *alle Vorkommen von* s_1, daß das nachfolgende s_3 1 bis 2 Zeiteinheiten danach kommt?

(2) Oder lassen wir den Zähler *nach jedem s_1 neu* laufen?

(3) Oder starten wir für *jedes einzelne Vorkommen von s_1* einen Zähler, die alle nacheinander bei s_3 abgefragt werden?

Alle Interpretationen sind in verschiedenen Fällen nützlich:

(1) Man will nicht, daß die beiden Prozesse sich ewig in M_1 aufhalten, sondern 1 bis 2 Zeiteinheiten nach Beginn von M_1 soll es mit M_2 losgehen.

(2) M_1 mag eine Herzschlag-Kommunikation bedeuten, die Prozeß 1 nach Belieben wiederholen kann, bis er sich entschließt, endlich die erlösende Nachricht 3 zu schicken. Dann interessiert nur, wie lange es vom letzten Vorkommen von s_2 an bis dann dauert.

(3) Die dritte Interpretation würde hier auch eine Beschränkung an die reine Anzahl der Sendungen und Empfänge bringen (was uns noch Probleme bereiten wird). Im allgemeinen kann man sich vorstellen, daß soetwas nützlich sein könnte. In diesem Beispiel sehen wir, daß diese Einschränkung es mit sich bringt, daß M_1, wenn wir denn wirklich eine genaue Entsprechung der Anzahlen wollen, genau einmal durchlaufen werden muß, der auf sich selbst zeigende Pfad von M_1 also nie genommen werden darf.

Wir definieren nun also die Semantik durch zusätzliche Bedingungen an die Zeiten x in $\genfrac{}{}{0pt}{}{w}{x}$. Das liegt auch insofern nahe, als wir später bei den Automaten ähnlich vorgehen werden, also nicht die Automaten auf analoge Weise neudefinieren, sondern die bestehenden für die ungezeiteten HMSCs erweitern.

Um die Notation zu erleichtern[6], sei $\text{ю}(w, i, X, Y)$ der Index des nächsten Vorkommens eines $x \in X$ in w nach Position i, sofern kein $y \in Y$ vorher vorkommt; oder $\perp$, falls es kein solches x gibt,[7] also z. B.

$$\text{ю}(efgf, 1, \{f\}, \emptyset) = 2,\ \text{ю}(efgf, 2, \{f\}, \{g\}) = \perp.$$

$\text{я}(w, e, X)$ für $e \notin X$ ist die Menge aller Paare (i, j) derart, daß $w_i = e, w_j \in X$ und j das $|w_1 \dots w_i|_e$-te Vorkommen eines $x \in X$ ist, zu welchen es vorher ein passendes e gab,[8] also z. B.

$$\text{я}(effgegfeeffe, e, \{f\}) = \{(1,2), (5,7), (8,10), (9,11)\}.$$

[6]Der Gewinn an Klarheit durch die Benutzung spezieller Zeichensätze für bestimmte Klassen von Objekten (hier: für technische Hilfsfunktionen) überwiegt meines Erachtens die Konfusion durch ungewohnte Symbole.

[7]Formal: $\text{ю}(w, i, X, Y) = i + \text{ю}'(w_{i+1} \dots w_{|w|}, X, Y)$ mit

$$\begin{aligned} \text{ю}'(xw', X, Y) &= \begin{cases} 1 & x \in X \\ \perp & x \in Y \setminus X \\ 1 + \text{ю}'(w', X, Y) & \text{sonst,} \end{cases} \\ \text{ю}'(\varepsilon, \_, \_) &= \perp. \end{aligned}$$

[8]Formal: Für w und V sei $I(w, V)$ die aufsteigend sortierte Liste von Indices i_ν mit $w_{i_\nu} \in V$. Dann ist $\text{я}(w, e, X) = \text{я}'(I(w, \{e\}), I(w, X))$ mit

$$\begin{aligned} \text{я}'(i\bar{i}, j\bar{j}) &= \begin{cases} \{(i,j)\} \cup \text{я}'(\bar{i}, \bar{j}) & i < j \\ \text{я}'(i\bar{i}, \bar{j}) & i > j, \end{cases} \\ \text{я}'(\varepsilon, \_) &= \emptyset, \\ \text{я}'(\_, \varepsilon) &= \emptyset. \end{aligned}$$

я behandelt also $\sim$-äquivalente Ereignisse gleich. Ist das nicht gewünscht, so muß hier я geändert werden, die auf я beruhende Semantikdefinition bleibt dann gleich.

Wir werden beide Funktionen außer für $\mathcal{E}$ als auch für Σ benutzen, mit analoger Bedeutung.

Definition 4.6 (Semantik) Sei $H \in \mathfrak{H}_\mathfrak{t}$.

(a) *Läufe, Endläufe,* $\mathfrak{L}(H)$ und $\mathfrak{L}_*(H)$ sind definiert wie in Definition 2.12.

(b) Sei $l \in N^*$ wie in Definition 2.13, bezogen auf $\mathfrak{u}(H)$. Sei $M = \mathfrak{m}(l)$. Dann ist die *$\natural$-Semantik von l unter π*, $[\![l;\pi]\!]_\natural$, definiert wie folgt:

(1) $[\![l;\pi]\!]_\star := \{ \genfrac{}{}{0pt}{}{\pi(w)}{t} : w \in [\![M]\!] \wedge \genfrac{}{}{0pt}{}{w}{t} \in \mathbb{W}$, sodaß für alle i die folgende Bedingung erfüllt ist$\}$:

$$\forall e \in \mathcal{E}.(j = \mathfrak{w}(w,i,\{e\},\emptyset) \to t_j - t_i \in \tau(w_i,e)).$$

(2) $[\![l;\pi]\!]_> := \{ \genfrac{}{}{0pt}{}{\pi(w)}{t} : w \in [\![M]\!] \wedge \genfrac{}{}{0pt}{}{w}{t} \in \mathbb{W}$, sodaß für alle i die folgende Bedingung erfüllt ist$\}$:

$$\forall e \in \mathcal{E}.(j = \mathfrak{w}(w,i,\{e\},\{w_i\}) \to t_j - t_i \in \tau(w_i,e))\}.$$

(3) $[\![l;\pi]\!]_1 := \{ \genfrac{}{}{0pt}{}{\pi(w)}{t} : w \in [\![M]\!] \wedge \genfrac{}{}{0pt}{}{w}{t} \in \mathbb{W}$, sodaß für alle $e, e' \in \mathcal{E}$ die folgende Bedingung erfüllt ist$\}$:

$$\forall (i,j) \in \mathfrak{s}(w,e,\{e'\}).t_j - t_i \in \tau(e,e')\}.$$

Falls τ für gewisse Ereignise nicht definiert ist, ist keine Bedingung zu erfüllen. $[\![H;\pi]\!]_\natural := \{[\![l;\pi]\!]_\natural : l \in \mathfrak{L}_*(H)\}$ für eine der drei Semantiken $\natural$. π wird für $\pi = \mathrm{id}$ wie üblich weggelassen. Ist klar oder unerheblich, welche Semantik wir benutzen, lassen wir den Index auch weg.

(c) H heißt *zeiterfüllbar (unter der Semantik $\natural$)*, falls $[\![H]\!] \neq \emptyset$ (bzw. $[\![H]\!]_\natural \neq \emptyset$).

Lemma 4.2 Sei $H \in \mathfrak{H}_\mathfrak{t}$.

(a) $[\![H]\!]_\star \subseteq [\![H]\!]_>$. Gleichheit gilt im allgemeinen nicht.

(b) $[\![H]\!]_1$ ist mit $[\![H]\!]_>$ und $[\![H]\!]_\star$ im allgemeinen nicht vergleichbar.

Beweis:

(a) In der Situation von Definition 4.6.b ist offenbar, falls $j = \mathfrak{w}(w,i,\{e\},\{w_i\}) \neq \bot$, auch $j = \mathfrak{w}(w,i,\{e\},\emptyset)$, womit die erste Aussage folgt.

Für die andere Aussage betrachte man das HMSC H in Abbildung 4.4. Es sei $\tau(A,B) = (0,1]$. Dann ist $\genfrac{}{}{0pt}{}{A}{0}\genfrac{}{}{0pt}{}{A}{1}\genfrac{}{}{0pt}{}{B}{2} \in [\![H]\!]_> \setminus [\![H]\!]_\star$.

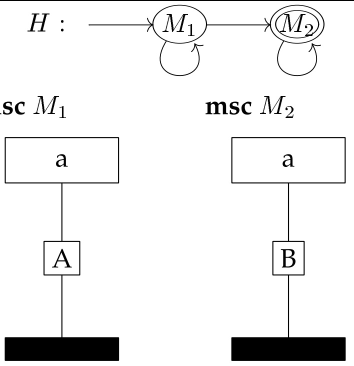
(b) Mit H aus (a) ist $\genfrac{}{}{0pt}{}{A}{0}\genfrac{}{}{0pt}{}{A}{1}\genfrac{}{}{0pt}{}{B}{1}\genfrac{}{}{0pt}{}{B}{2} \in [\![H]\!]_1 \setminus [\![H]\!]_>$, aber $\genfrac{}{}{0pt}{}{A}{0}\genfrac{}{}{0pt}{}{A}{0,5}\genfrac{}{}{0pt}{}{B}{1}\genfrac{}{}{0pt}{}{B}{3} \in [\![H]\!]_\star \setminus [\![H]\!]_1$. ⊛

BEN-ABDALLAH und LEUE führen in [BAL97a] gewisse Einschränkungen an die von ihren analysierten HMSCs auf, die wir in unserer Notation formalisieren können wie folgt.

Definition 4.7 (einfach gezeitet) Ein $H \in \mathfrak{H}_\mathfrak{t}$ heißt *einfach gezeitet*, falls für jede einfache Schleife $l := l_1 \ldots l_s l_1$ gilt, daß es kein $e \in \mathcal{E}(M_{l_1})$ gibt, sodaß $\tau(e,e') \neq \mathbb{T}$ für ein $e' \in \mathcal{E} \setminus \mathcal{E}(\mathfrak{m}(l))$.

Das ist offensichtlich entscheidbar.

Abbildung 4.4 Zum Beweis von Lemma 4.2



4.b.ii Automaten

Nun wollen wir also für die Objekte Automaten definieren, und zwar einen einzelnen Uhrenautomaten für je ein schwach konvergentes HMSC.[9] Betrachten wir zunächst Interpretation 2, weil dies die einfachste ist. Uns interessiert für jedes Paar von Ereignissen (e_1, e_2) nur, wann das letzte Mal e_1 vor e_2 stattfand, um $\tau(e_1, e_2)$ zu prüfen. Also halten wir entsprechend viele Uhren vor und merken uns in den Konfigurationen, welche Einschränkungen in der Folgezeit zu verifizieren sind.[10]

Definition 4.8 (HMSC-Automat) Sei $H \in \mathfrak{H}_\mathfrak{t}$. Sei $\mathcal{F}(\mathfrak{u}(H))$ gegeben als

$$\mathcal{F}(\mathfrak{u}(H)) = \left\langle C, \mathcal{E}, \delta, c_0, C_* \right\rangle.$$

Dann ist $\mathcal{F}(H)_>$ definiert als der Uhrenautomat

$$\left\langle C', X, \mathcal{E}, \delta', c'_0, C'_* \right\rangle,$$

wobei gilt:

- ▷ $C' = C \times (\mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{E}))$;
- ▷ $C'_* = C_* \times (\mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{E}))$;
- ▷ $c'_0 = (c_0, \emptyset)$;
- ▷ $X = \{x_{e_1,e_2} : e_1, e_2 \in \mathcal{E}\}$;
- ▷ ▶ Für jede Transition $\delta(s, e) = s'$ mit $e \in \mathcal{E}$ und jedes $W \in (\mathfrak{p}(\mathcal{E}))^2$ ist $((s, W), e, \psi, \gamma, (s', W')) \in \delta'$, wobei

$$\begin{aligned} \psi &= \bigwedge_{(e',e) \in W} x_{e',e} \in \tau(e', e), \\ \gamma &= \{x_{e,e'} : e' \in \mathcal{E}\} \end{aligned}$$

[9] Man könnte statt der hier vorzustellenden Automaten auch das Modell der *Concurrent Timed Automata* aus [LMST01] verwenden. Hier werden mehrere Uhrenautomaten nebeneinanderlaufengelassen, wobei bei Eingabewerten, die in allen ihren Eingabealphabeten liegen, auch alle reagieren müssen und insofern eine Synchronisation erzwungen wird. Dieser Zugang wurde hier nicht gewählt, weil mir die einfache Konstruktion des Automaten und die nachträgliche Erweiterung mit Uhren und Uhrbedingungen intuitiv einsichtiger erscheint.

[10] Zum Beispiel kann das allererste Ereignis nicht von irgendeiner Uhr abhängig gemacht werden, auch wenn es entsprechende Einschränkungen gibt, weil das Start-MSC von irgendeinem anderen Knoten erreicht werden kann. Also muß der Automat sich merken, welche Bedingungen zu überprüfen sind.

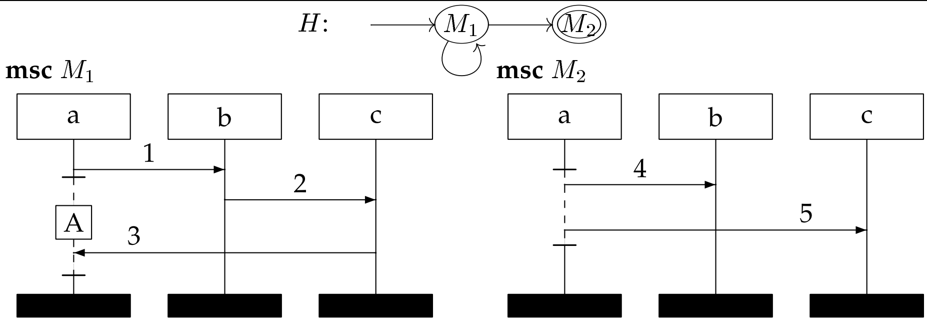
und $W' = (W \setminus (\mathcal{E} \times \{e\})) \cup (\{e\} \times \mathcal{E})$.

- Ansonsten enthält δ' nichts.

Jetzt und für immer gelte, daß alle Uhren und Marken, die nicht gebraucht werden, nicht zum Automaten gehören mögen. *Nicht gebraucht* werden eine Uhr oder ein Marker zu $(e, e') \in \mathcal{E} \times \mathcal{E}$, falls $\tau(e, e') = \perp$ oder $\tau(e, e') = \mathbb{T}$.

Beispiel 4.4 Es sei $\tau(s_1, r_1) = \tau(s_2, r_2) = (0, 1]$ in Abbildung 4.5. Abbildungen 4.6 und 4.7 geben die Automaten an. Im Beispiel werden die Zustände mit drei Komponenten geschrieben. Die ersten beiden entsprechen den Komponenten des jeweiligen Zustandes des normalen HMSC-endlich-Automaten, die dritte Komponente hält die Marken vor.

Abbildung 4.5 Ein zu übersetzendes HMSC mit Timing (Beispiel 4.4)



Es ist Zufall, daß beide Automaten gleichviele Zustände haben.

Kommen wir nun noch einmal zu Interpretation 1 aus Beispiel 4.3 zurück. Angenommen, wir haben eine Beschränkung der Art $\tau(e_1, e_2) = [a, b]$. Sei w_1 das erste Vorkommen von e_1, w_r das letzte in einer relevanten Sequenz. Dann ist offenbar dann die Beschränkung für alle Vorkommen erfüllt, wenn es für w_1 und w_r der Fall ist, da ein Intervall dicht ist. Wir brauchen also nur diese beiden Zeiten mitzunehmen, nehmen dafür die beiden Uhren x_{e_1,e_2} für das erste und y_{e_1,e_2} für das letzte Vorkommen.

Definition 4.9 (HMSC-Automat) Sei $H \in \mathfrak{H}_\mathfrak{t}$. Sei $\mathcal{F}(\mathfrak{u}(H))$ gegeben als

$$\mathcal{F}(\mathfrak{u}(H)) = \langle C, \mathcal{E}, \delta, c_0, C_* \rangle.$$

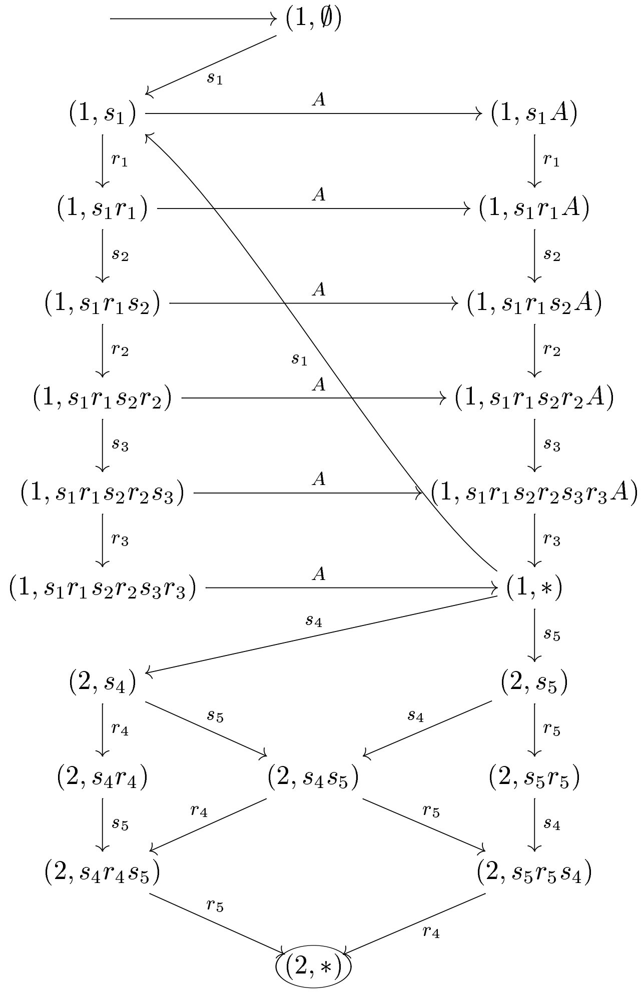
Dann ist $\mathcal{F}_\star(H)$ definiert als der Uhrenautomat

$$\langle C', X, \mathcal{E}, \delta', c_0', C_*' \rangle,$$

wobei gilt:

- $C' = C \times (\mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{E}))$;
- $C_*' = C_* \times (\mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{E}))$;
- $c_0' = (c_0, \emptyset)$;
- $X = \{x_{e_1,e_2}, y_{e_1,e_2} : e_1, e_2 \in \mathcal{E}\}$;

Abbildung 4.6 $\mathcal{F}(\mathfrak{u}(H))$ zu Abbildung 4.5 (Beispiel 4.4)



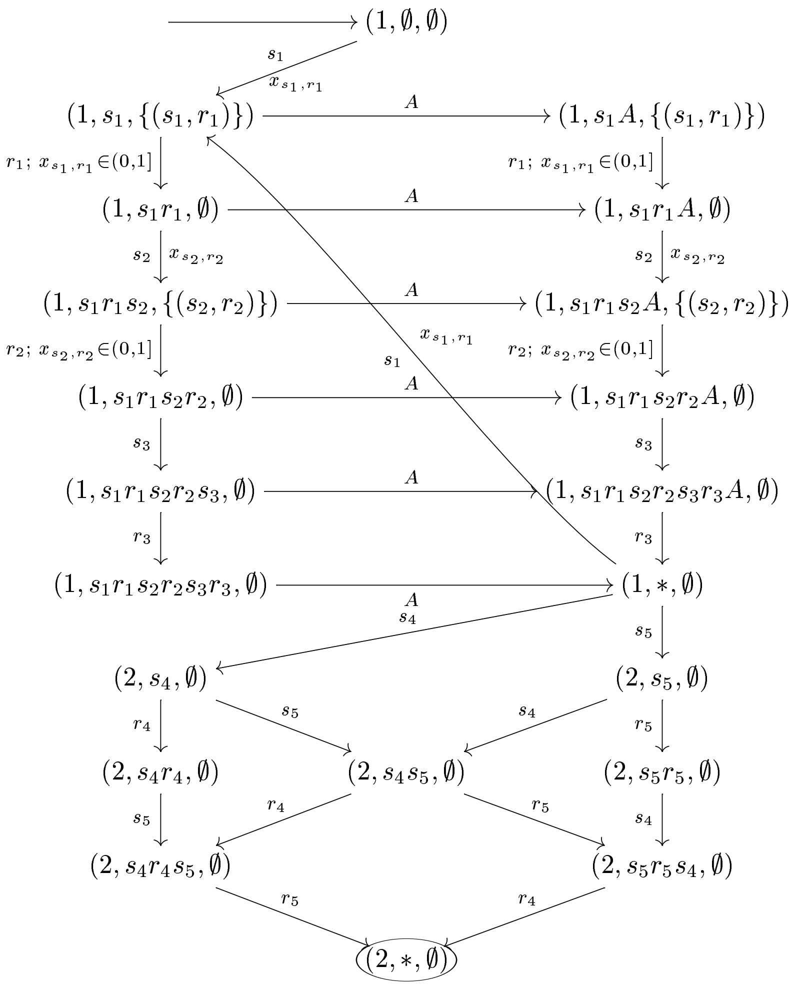
▷
- Für jede Transition $\delta(s, e) = s'$ mit $e \in \mathcal{E}$ und jedes $W \in (\mathfrak{p}(\mathcal{E}))^2$ ist $((s, W), e, \psi, \gamma, (s', W')) \in \delta'$, wobei

$$\begin{aligned}\psi &= \bigwedge_{(e',e)\in W} (x_{e',e} \in \tau(e, e') \wedge y_{e',e} \in \tau(e', e)),\\ \gamma &= \{y_{e,e'} : e' \in \mathcal{E}\} \cup \{x_{e,e'} : (e, e') \notin W \wedge e' \in \mathcal{E}\}\end{aligned}$$

  und $W' = (W \backslash (\mathcal{E} \times \{e\})) \cup (\{e\} \times \mathcal{E})$.
- Ansonsten enthält δ' nichts.

Interpretation 3 bereitet hingegen gewisse Probleme. Man betrachte das HMSC gegeben in Abbildung 4.8, wobei wir beispielhaft synchrone Konkatenation unterlegen. Es ist sicher leicht einsehbar, daß wir für H einen endlichen Automaten erstellen können (keine Probleme wie in Abbildung 2.19). Angenommen jedoch, wir hätten die Maßgabe $\tau(s_1, s_3) = [2, 5]$. Zählten wir nun die Vorkommen von s_1 (was für Interpretation 3 nötig ist), so könnte diese Anzahl unbeschränkt wachsen, und wir müßten eine nicht vor der Übersetzung bekannte Anzahl von Uhren

Abbildung 4.7 $\mathcal{F}_>(H)$ zu Abbildung 4.5 (Beispiel 4.4)



vorhalten. Auch Konvergenz ändert daran nichts: Wann immer irgendeine Schleife exisitiert, bekommen wir Probleme. Man beachte dazu [B95], wo bewiesen wird, daß die Projektion einer gezeiteten Sprache auf die diskrete Komponente regulär sein muß, damit ein Uhrenautomat die ursprüngliche Sprache erkennen kann.

Daher müssen wir diese Interpretation leider verwerfen. Diese prinzipielle Unmöglichkeit sollte allerdings auch nicht als Überraschung kommen: Sobald man anfängt zu zählen, kommt es bei endlichen Systemen generell zu Problemen.

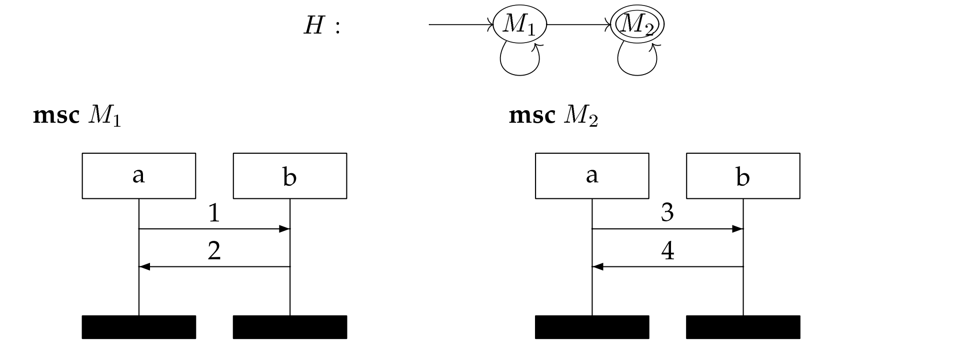
Daß man mit geeigneten Einschränkungen an die Ereignisse, die einer solchen Beschränkung unterworfen sein sollen, dennoch eine Übersetzung definieren kann, wird im Umfeld um Definition 4.18 besprochen.

Satz 4.2 Sei $H \in \mathfrak{H}_t$.

(a) $\mathcal{L}(\mathcal{F}_\star(H)) = [\![H]\!]_\star$, $\mathcal{L}(\mathcal{F}_>(H)) = [\![H]\!]_>$.

(b) Ist H schwach konvergent, so sind $\mathcal{F}_\star(H)$ und $\mathcal{F}_>(H)$ endlich.

Abbildung 4.8 Ein problematisches HMSC mit Timing



Beweis: (b) ist klar nach Satz 2.8, sodaß wir uns auf (a) beschränken. Hier läuft der Beweis für $\star$ gleich wie bei $>$, weswegen hier nur der $>$-Fall behandelt wird.

Sei $\genfrac{}{}{0pt}{}{w}{t} \in [\![H]\!]_>$ beliebig. Zu zeigen ist: $\genfrac{}{}{0pt}{}{w}{t} \in \mathcal{L}(\mathcal{F}_>(H))$. Nach Definition von $[\![H]\!]_>$ ist $w \in [\![\mathfrak{u}(H); \mathrm{id}]\!]$. Es gibt also einen Lauf $c_0 \to^{w_1} \to^{w_2} \ldots \to^{w_n} c_n$ in $\mathcal{F}(\mathfrak{u}(H))$ mit $c_n \in C_*$ (Bezeichnungen wie in Definition 4.8).

Sei nun j derart, daß es einen Lauf $c'_0 \to^{(w_1,t_1)} c'_1 \to^{(w_2,t_2)} \ldots \to^{(w_{j-1},t_{j-1})} c'_{j-1}$ gibt, sodaß c_{j-1} die C-Komponente von c'_{j-1} ist, es aber für jeden solchen Lauf keinen (w_j, t_j)-Nachfolger von c'_{j-1} gibt, dessen C-Komponente c_j ist. Falls das gezeigt ist, ist nur noch zu zeigen, daß es immer überhaupt einen Nachfolger gibt. Denn dann gibt es einen $\genfrac{}{}{0pt}{}{w}{t}$-Lauf in $\mathcal{F}_>(H)$ zu einem c'_n mit c_n in der C-Komponente, was nach Definition 4.8 $c'_n \in C'_*$ bedeutet. Somit würde $\genfrac{}{}{0pt}{}{w}{t}$ erkannt und $[\![H]\!]_> \subseteq \mathcal{L}(\mathcal{F}_>(H))$ wäre bewiesen.

Gebe es also ein j wie behauptet. Dann gilt für die w_j-Transitionen, daß eine Uhrbedingung nicht erfüllt ist. Sei etwa $x_{e',w_j} \notin \tau(e', w_j)$ für ein $(e', w_j) \in W$, wobei W die Uhrmarkenkomponente von c'_{j-1} sei. Daß $e' \in W$, heißt, daß $w_i = e'$ für ein $i < j$ (damit es reingekommen ist), und daß außerdem $w_{i'} \neq e'$ für $i < i' < j$ (damit es noch nicht rausgeflogen ist). Somit ist $j = \wp(w, i, \{w_j\}, \{e'\})$. $x_{e',w_j} \notin \tau(e', w_j)$ heißt dann, daß i die Bedingung aus Definition 4.6.b nicht erfüllt, im Widerspruch zur Voraussetzung. So ein j kann es also nicht geben.

Jetzt ist noch zu zeigen, daß es tatsächlich irgendeinen Lauf gibt. Das ist aber genauso klar.

Die Rückrichtung, $\mathcal{L}(\mathcal{F}_>(H)) \subseteq [\![H]\!]_>$, zeigt man analog. Damit ist die Gleichheit bewiesen und wir sind fertig. ⊛

Man beachte, daß die Hauptarbeit schon in Kapitel 2 geleistet wurde. Die Erweiterung mit den Zeitbeschränkungen geht sehr leicht.

4.c HMSCs mit Timern

4.c.i Einführung

Der Standard [ITU99, CEMR98] definiert auch *Timer*. Ein Prozeß kann über einen oder mehrere Timer[11] verfügen. Damit kann er, und nur er, verschiedene Operatio-

[11] Auch wir verwenden hier nicht den natürlicheren Ausdruck *Uhr*, sondern benutzen das vom Standard vorgegebene englische Wort, um Verwechslungen mit den *Uhren* der Uhrenautomaten zu vermei-

nen durchführen:

- Er kann ihn mit einem bestimmten Wert *initialisieren*, woraufhin der Timer beginnt, abzulaufen.
- Er kann das *Ablaufen beobachten*.
- Er kann den noch nicht abgelaufenen Timer *anhalten*.

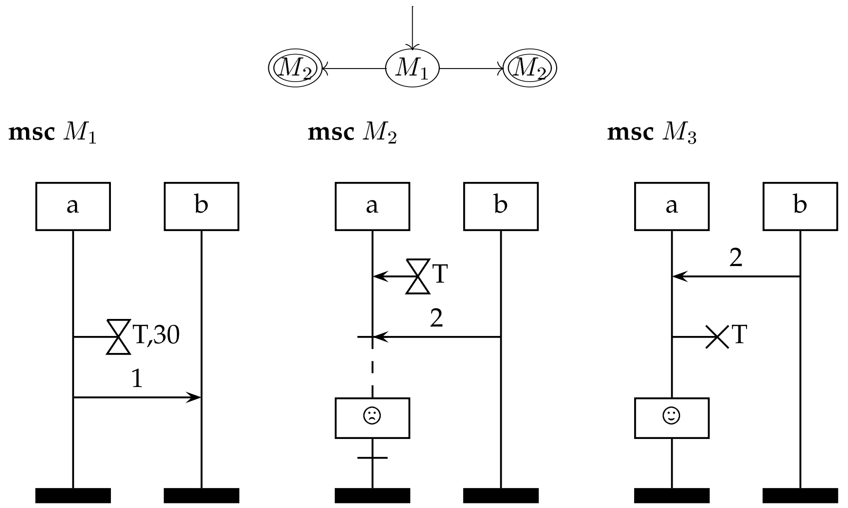
Mit „anhalten“ ist hier nicht gemeint, daß der Timer irgendwann kurz angehalten wird und danach wieder ans Laufen kommt, sondern er wird einfach abgeschaltet. „Ablaufen“ bedeutet, daß die Timer alle mit gleicher Geschwindigkeit (vgl. Fußnote 2) herunterzählen, bis sie auf 0 ankommen.

Beispiel 4.5 In einem MSC werden Zugriffe auf Timer durch drei Symbole ausgedrückt:

- Die Initialisierung durch einen Strich mit einer Uhr und der Initialisierungszeit.
- Der Ablauf durch einen Pfeil mit Uhr.
- Der Stopp durch ein Kreuz.

In Abbildung 4.9 wird deutlich, daß man mit Timern auf sehr natürliche und intuitiv einsichtigere Weise als mit Zeitbeschränkungen Sachverhalte modellieren kann wie etwa hier die Tatsache, daß nur eine gewisse Zeit auf eine Nachricht gewartet werden soll. Die Übersichtlichkeit in der graphischen Darstellung, wenn Zeiten zwischen weit auseinanderliegenden Ereignissen im Auge behalten werden müssen, leidet somit nicht, da man alles direkt im Blick hat.

Abbildung 4.9 Beispielhafte Benutzung von Timern (Beispiel 4.5)



den.

Für HMSCs wird es sich als sinnvoll erweisen, nicht bei allen Timern zu verlangen, daß diese am Schluß eines Durchlaufes abgelaufen oder angehalten worden sind. Wir notieren die, von denen wir soetwas zwingend verlangen, und bezeichnen sie als *Endtimer*. Damit kann man sowohl verlangen, daß ein Durchlauf durch ein MSC zu einer geeigneten Reaktion auf einen Timer führt, als auch, daß ein Timer nur in speziellen Situationen beachtet werden muß. In ersterem Fall müßte man ihn als Endtimer deklarieren, im anderen nicht und weiter nur etwa den Time-Out-Fall betrachten. Das ist standardmäßig nicht vorgesehen.

4.c.ii Formalisierung

Definition 4.10 (MSC mit Timern) Ein *MSC mit Timern* ist ein 12-Tupel

$$M := \langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell, \tau, \mathcal{T}, \vartheta, \chi, \mathcal{T}_! \rangle,$$

wobei gilt:

- ▷ $\mathcal{T}$ ist eine endliche Menge (eine Menge von *Timern*);
- ▷ $\mathcal{E} = \mathcal{S} \uplus \mathcal{R} \uplus \mathcal{A} \uplus \mathcal{I} \uplus \mathcal{C} \uplus \mathcal{O}$, mit *Timer-Initialisierungen* $\mathcal{I}$, *Timer-Löschungen* $\mathcal{C}$ und *Time-Outs* $\mathcal{O}$;
- ▷ $\vartheta : \mathcal{I} \to \mathbb{T}$ (eine *Zeit-Zuordnungsfunktion*);
- ▷ $\chi : \mathcal{I} \cup \mathcal{O} \cup \mathcal{C} \to \mathcal{T}$ (eine *Timer-Zuordnungsfunktion*);
- ▷ $o(e) = o(e')$ für alle $e, e' \in \mathcal{I} \cup \mathcal{T} \cup \mathcal{O}$ mit $\chi(e) = \chi(e')$
- ▷ $\mathcal{T}_! \subseteq \mathcal{T}$ (eine Menge der *Endtimer*),

und ansonsten $\langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell, \tau \rangle$ ein MSC mit Timing ist (Definition 4.1). Die Menge aller solcher MSCs ist $\mathfrak{M}_{\mathrm{tt}}(\mathcal{E}, \Lambda, \mathcal{T})$, gegebenenfalls mit weniger Parametern, wie üblich.

M heißt *timer-konsistent*, falls gilt:

- ▷ (Nur gesetzte Timer laufen ab, und zwar nur einmal.)

  $$\begin{aligned}&\forall e \in \mathcal{O} \cup \mathcal{C}. \exists e' \in \mathcal{I}.\\ &(\chi(e) = \chi(e') \land e' \prec e \land \forall e'' \in \mathcal{O} \cup \mathcal{C}.(\chi(e'') = \chi(e') \to (e'' \prec e' \lor e \prec e''))),\end{aligned}$$

  und

- ▷ (Kein Endtimer läuft am Schluß.)

  $$\forall T \in \mathcal{T}_!. \exists e \in \mathcal{C} \cup \mathcal{O}.(\chi(e) = T \land \forall e' \in \mathcal{I}.(\chi(e') = T \to e' \prec e)).$$

$\chi(e)$ zeigt also an, um welchen Timer es sich im Ereignis e handelt, $\vartheta(e)$, auf was er gesetzt wird.

Wir erlauben insbesondere, daß an die Timer-Ereignisse Zeitbeschränkungen gestellt werden können. Das ist sinnvoll, denn wenn wir mit einem Timer überprüfen wollen, ob man nach einer Nachricht länger als α Sekunden auf eine Antwort warten muß, ist es nicht hilfreich, wenn man den Timer auch erst 2α Sekunden nach dem Senden der Anfrage anstellen darf.

Die Interpretationen 1–3 aus dem vorigen Abschnitt (Seite 77) übertragen sich auf natürliche Weise, je nachdem, wie die Frage, was gemacht werden soll, wenn ein Timer, bevor er abläuft, mehrmals gesetzt wird, beantwortet wird. Es kommen hier noch weitere, ebenfalls natürliche Interpretationen hinzu:

(4) Man könnte ein Neu-Setzen eines Timers auch einfach für die quantitativen Aspekte ignorieren: Der Timer läuft halt schon.

(5) Man könnte das auch als ungültige Linearisierung betrachten und somit solche Transitionen verbieten.

Auch diese beiden Interpretationen lassen sich im gleichen Rahmenwerk behandeln. Der sittliche Nährwert einer genaueren Betrachtung ist jedoch sehr gering, weswegen ich darauf verzichte.

Zunächst wollen wir aber stillschweigend die alten Definitionen der Semantiken von MSCs und HMSCs sowie der Automaten um Timer-Events erweitern, indem diese wie lokale Aktionen behandelt werden (also wie in [CEMR98] vorgesehen). Diese Semantik werden wir von nun an mit $[\![\_]\!]_0$ bezeichnen. Man erachte also die Definitionen hinreichend als geändert und ergänze σ und Σ aus Definition 2.3 wie folgt:

$$\sigma(e) = \begin{cases} \mathsf{I}_T(\xi) & e \in \mathcal{I} \wedge \chi(e) = T \wedge \vartheta(e) = \xi \\ \mathsf{O}_T & e \in \mathcal{O} \wedge \chi(e) = T \\ \mathsf{C}_T & e \in \mathcal{C} \wedge \chi(e) = T \end{cases}$$

und $\Sigma \supseteq \{\mathsf{I}_T(\xi), \mathsf{C}_T, \mathsf{O}_T : T \in \mathcal{T},\ \xi \in \operatorname{range}(\vartheta)\}$.[12] Ebenso werden in Definitionen 2.9 und 2.11 die Timer-Events wie Elemente aus $\mathcal{A}$ behandelt. So wie wir für Nachrichten i die Ereignisse s_i und r_i benannt haben, werden wir für Timer-Ereignisse e $\sigma(e)$ zu ihrer Benennung benutzen, falls kein Mißverständnis möglich ist. Falls $\vartheta(e)$ durch $\chi(e)$ eindeutig bestimmt ist, schreiben wir kurz I_T.

Da etwas wie der Begriff der Timer-Konsistenz auch für MSC-Wörter nützlich ist, wollen wir ihn hier kurz ansprechen. Die Definition ist analog zu Definition 2.5.

Definition 4.11 (Konsistenz) Sei w ein konsistentes MSC-Wort (Definition 2.5, Bemerkung gerade eben).

(a) w heißt *timer-konsistent*, falls

$$\forall T \in \mathcal{T}. \forall w_i \in \sigma^{-1}\{\mathsf{O}_T, \mathsf{C}_T\}. \exists w_j \in \sigma^{-1}(\mathsf{I}_T). i = \mathfrak{w}(\sigma(w), j, \{\mathsf{O}_T, \mathsf{C}_T\}, \{\mathsf{I}_T\})),$$

und außerdem

$$\forall T \in \mathcal{T}_!. \forall w_i \in \sigma^{-1}\{\mathsf{I}_T\}. \mathfrak{w}(w, i, \{\mathsf{C}_T, \mathsf{O}_T\}, \emptyset) \neq \bot.$$

(b) w heißt *klammer-konsistent (bzgl. $\mathcal{T}_! \subseteq \mathcal{T}$)*, falls für alle Präfixe w' von w gilt, daß

$$\forall T \in \mathcal{T}. |\sigma(w')|_{\mathsf{I}_T} \geqslant |\sigma(w')|_{\mathsf{C}_T} + |\sigma(w')|_{\mathsf{O}_T},$$

und außerdem

$$\forall T \in \mathcal{T}_!. |\sigma(w)|_{\mathsf{I}_T} = |\sigma(w)|_{\mathsf{C}_T} + |\sigma(w)|_{\mathsf{I}_T}.$$

[12] Σ ist also jetzt auch von ϑ abhängig und nicht nur von den offensichtlichen Parametern $\Lambda, \mathcal{P}, \mathcal{T}$. Das ist aber kein Problem.

Beispiel 4.6 Sei T ein Endtimer. Ein w mit $\sigma(w) = \mathsf{I}_T\mathsf{I}_T\mathsf{C}_T\mathsf{C}_T$ ist klammer-konsistent, aber nicht timer-konsistent; falls $\sigma(w) = \mathsf{I}_T\mathsf{I}_T\mathsf{C}_T$, ist w timer-konsistent, aber nicht klammer-konsistent. $\sigma(w) = \mathsf{I}_T\mathsf{C}_T\mathsf{I}_T\mathsf{C}_T$ bedeutet beiderlei Konsistenz, $\sigma(w) = \mathsf{I}_T\mathsf{C}_T\mathsf{C}_T$ keinerlei.

Offenbar erzeugt ein timer-konsistentes MSC nur timer-konsistente Worte.

Die Erweiterung unserer Timer-Definitionen auf HMSCs geschieht nun ganz einfach, da wir keinerlei weitere Informationen brauchen.

Definition 4.12 (HMSC mit Timern) Ein

$$H := (N, \mathcal{P}, \Lambda, (M_i)_{i=1}^{|N|}, E, k, i_0, N_*)$$

heißt *HMSC mit Timern*, falls alle $M_i \in \mathfrak{M}_{\mathfrak{tt}}(\mathcal{P}, \Lambda, \mathcal{T})$ sind und ansonsten die Bedingungen aus Definition 4.5 gelten. $\mathfrak{u}(H)$ ist wieder das ungezeitete HMSC. Die Menge aller solchen H ist $\mathfrak{H}_{\mathfrak{tt}}(\mathcal{P}, \Lambda, \mathcal{T})$, gegebenenfalls mit weniger Parametern geschrieben.

Natürlich wollen wir, daß das entstehende HMSC in einem gewissen Sinne timer-konsistent ist. Eine Definition über die Konsistenz der benutzten MSCs (Definition 4.10) ist nicht so einfach möglich, da wir auch, ähnlich wie bei den Zeitbeschränkungen, ein MSC-übergreifendes Verhalten modellieren können sollten.

Definition 4.13 (Konsistenz) Sei $H \in \mathfrak{H}_{\mathfrak{tt}}$.

(a) H heißt *timer-konsistent*, falls jedes $w \in [\![H]\!]_0$ timer-konsistent ist.

(b) H heißt *klammer-konsistent*, falls jedes $w \in [\![H]\!]_0$ klammer-konsistent ist.

Die MSCs müssen also nicht timer-konsistent sein (Definition 4.10), um timer-konsistente HMSCs zu erlauben, da ansonsten etwas wie in Beispiel 4.5 nicht erlaubt wäre.

Nun können wir, mit unseren üblichen drei Interpretationen (Seite 77, vgl. Seite 87), die Semantiken eines $H \in \mathfrak{H}_{\mathfrak{tt}}$ angeben.

Definition 4.14 (Semantik) Sei $H \in \mathfrak{H}_{\mathfrak{tt}}$.

(a) *Läufe, Endläufe,* $\mathfrak{L}(H)$ und $\mathfrak{L}_*(H)$ sind definiert wie in Definition 4.6.

(b) Sei $l \in N^*$ wie in Definition 2.13, bezogen auf $\mathfrak{u}(H)$. Sei $M = \mathfrak{m}(l)$. Dann ist die $\natural$-*Semantik von* l *unter* π, $[\![l;\pi]\!]_\natural$, definiert wie folgt:

(1) $[\![l;\pi]\!]_\star := \{ {\pi(w) \atop t} : w \in [\![M]\!] \wedge {w \atop t} \in \mathbb{W}$, sodaß die folgenden Bedingungen erfüllt sind$\}$:

- ▷ Für alle i gilt: $\forall e \in \mathcal{E}.(j = \mathfrak{w}(w, i, \{e\}, \emptyset) \rightarrow t_j - t_i \in \tau(w_i, e))$.
- ▷ w ist timer-konsistent.
- ▷ Für alle i mit $\sigma(w_i) = \mathsf{I}_T$ gilt: Ist $j = \mathfrak{w}(\sigma(w), i, \{\mathsf{O}_T\}, \{\mathsf{C}_T\})$, so ist $t_j - t_i = \vartheta(w_i)$; ist $j = \mathfrak{w}(\sigma(w), i, \{\mathsf{C}_T\}, \{\mathsf{O}_T\})$, so ist $t_j - t_i < \vartheta(w_i)$; ist $j = \mathfrak{w}(\sigma(w), i, \{\mathsf{I}_T\}, \{\mathsf{O}_T, \mathsf{C}_T\})$, so ist $t_j - t_i < \vartheta(w_i)$.

(2) $[\![l;\pi]\!]_> := \{ {\pi(w) \atop t} : w \in [\![M]\!] \wedge {w \atop t} \in \mathbb{W}$, sodaß die folgenden Bedingungen erfüllt sind$\}$:

- ▷ Für alle i gilt: $\forall e \in \mathcal{E}.(j = \text{ю}(w, i, \{e\}, \{w_i\}) \rightarrow t_j - t_i \in \tau(w_i, e))$.
- ▷ w ist timer-konsistent.
- ▷ Für alle i mit $\sigma(w_i) = \mathsf{I}_T$ gilt: Ist $j = \text{ю}(\sigma(w), i, \{\mathsf{O}_T\}, \{\mathsf{C}_T, \mathsf{I}_T\})$, so ist $t_j - t_i = \vartheta(w_i)$; ist $j = \text{ю}(\sigma(w), i, \{\mathsf{C}_T\}, \{\mathsf{O}_T, \mathsf{I}_T\})$, so ist $t_j - t_i < \vartheta(w_i)$; ist $j = \text{ю}(\sigma(w), i, \{\mathsf{I}_T\}, \{\mathsf{O}_T, \mathsf{C}_T\})$ und ist $T \in \mathcal{T}_!$, so ist $t_j - t_i < \vartheta(w_i)$.

(3) $[\![l; \pi]\!]_1 := \{ \genfrac{}{}{0pt}{}{\pi(w)}{t} : w \in [\![M]\!] \wedge \genfrac{}{}{0pt}{}{w}{t} \in \mathbb{W}$, sodaß die folgenden Bedingungen erfüllt sind$\}$:

- ▷ Für alle $e, e' \in \mathcal{E}$ gilt: $\forall (i, j) \in \text{я}(w, \{e\}, \{e'\}).t_j - t_i \in \tau(e, e')$.
- ▷ w ist klammer-konsistent.
- ▷ Für alle $T \in \mathcal{T}$ und alle $(i, j) \in \text{я}(\sigma(w), \{\mathsf{I}_T\}, \{\mathsf{O}_T, \mathsf{C}_T\})$ gilt: Falls $\sigma(w_j) = \mathsf{C}_T$, so ist $t_j - t_i < \vartheta(w_i)$, ansonsten ist $t_j - t_i = \vartheta(w_i)$.

$[\![H; \pi]\!]_\natural := \{[\![l; \pi]\!]_\natural : l \in \mathfrak{L}_*(H)\}$ für eine der drei Semantiken $\natural$.

$\pi = \mathrm{id}$ wird weggelassen. Ist klar oder unerheblich, welche Semantik wir benutzen, lassen wir den Index auch weg.

(c) H heißt *zeiterfüllbar (unter der Semantik $\natural$)*, falls $[\![H]\!] \neq \emptyset$ (bzw. $[\![H]\!]_\natural \neq \emptyset$).

Der Leser wundert sich vielleicht, warum bei der Definition von $[\![\_]\!]_\star$ für alle Timer gefordert wurde, daß man ihr Ablaufen bemerkt, und nicht nur für Endtimer wie in $[\![\_]\!]_>$. Dazu muß man sich überlegen, daß $[\![\_]\!]_\star$ nach dem intuitiven Semantikvorschlag eine Aussage über *alle* zwischen zwei Abfragen liegenden Initialisierungen macht. Es wäre nun nicht angemessen, träfe man eine Aussage über Verträglichkeit eines Wortes mit den Zeitbedingungen nur in Abhängigkeit von den nicht unbemerkt abgelaufenen Timern, denn damit verlöre man eine wichtige Möglichkeit, von der Modellierung her als unerwünscht bewertete Abläufe auszuschliessen (Seite 91).

4.c.iii Automaten

Und nun werden wieder die Automaten definiert. Das Prinzip ist ähnlich wie in Abschnitt 4.b.ii. Bei den beiden auch dort vertretenen Interpretationen werden die Timer in kleinen Varianten übersetzt, weswegen die Betrachtung sinnvoll erscheint.

Zunächst wieder zu Interpretation 2 (Seite 77). Es reicht eine Uhr für einen Timer, da jeweils nur eine Setzung, die letzte, relevant ist. Im Gegensatz zu den Zeitbeschränkungen müssen wir hier aber darauf achten, daß eine Timer-Beschränkung, also ein Ablaufen oder ein Anhalten eines Timers, nur möglich ist, falls er vorher gesetzt wurde. Im Gegensatz zum vorherigen Verhalten ist also in einem solchen Falle nicht einfach keine Beschränkung zu untersuchen, sondern es handelt sich um einen Fehler, weswegen es keine einen Ablauf oder ähnliches akzeptierende Transition geben kann. Dies müssen wir also in unserer Menge W mithalten. Da die Timer aber auf verschiedene Werte gesetzt werden können, müssen wir uns überlegen, wie wir den aktuellen zu überprüfenden Wert mithalten können.[13] Dafür verwenden wir partielle Funktionen, die den zu überprüfenden Stand angeben.

[13] An dieser Stelle wäre eine Erweiterung der Uhrenautomaten mit allgemeinen Updates und runterzählenden Uhren sinnvoll ([W94, BDFP00b]). Es geht aber auch so und fügt sich besser in die bestehenden Ergebnisse über Uhrenautomaten ein.

Da es nur eine endliche Anzahl von Initialisierungsständen gibt, ist die Gesamtmenge der möglichen Funktion endlich, ändert also nicht an der Endlichkeit des Automaten.

Man beachte, daß die Uhren nur auf 0 gesetzt werden und dann nach oben hin laufen, womit sich die Bedingungen ergeben wie angegeben. Weiterhin können die Verpflichtungen bei den Automaten nicht beliebig sein, sondern müssen die Timer enthalten, die beendet werden müssen. Das muß in der Endzustandsmenge berücksichtigt werden.

Definition 4.15 (HMSC-Automat) Sei $H \in \mathfrak{H}_{\mathfrak{tt}}$. Sei $\mathcal{F}(\mathfrak{u}(H))$ gegeben als

$$\mathcal{F}(\mathfrak{u}(H)) = \langle C, \mathcal{E}, \delta, c_0, C_* \rangle.$$

Dann ist $\mathcal{F}_>(H)$ definiert als der Uhrenautomat

$$\langle C', X, \mathcal{E}, \delta', c'_0, C'_* \rangle,$$

wobei gilt (vgl. Definition 4.8):

▷ $C' = C \times (\mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{E})) \times [\mathcal{T} \dashrightarrow \mathrm{range}(\vartheta)]$;

▷ $C'_* = C_* \times (\mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{T} \setminus \mathcal{T}_!)) \times [\mathcal{T} \dashrightarrow \mathrm{range}(\vartheta)]$;

▷ $c'_0 = (c_0, \emptyset, \emptyset)$;

▷ $X = \{x_{e_1,e_2} : e_1, e_2 \in \mathcal{E}\} \cup \{x_T : T \in \mathcal{T}\}$;

▷ ▶ Für jede Transition $\delta(s, e) = s'$ mit $e \in \mathcal{E}$, jedes $W \in (\mathfrak{p}(\mathcal{E}))^2$ und jedes $\xi : \mathcal{T} \dashrightarrow \mathrm{range}(\vartheta)$ ist $((s, W, \xi), e, \psi, \gamma, (s', W', \xi)) \in \delta'$, wobei

$$\begin{aligned} \psi &= \bigwedge_{(e',e) \in W} x_{e',e} \in \tau(e', e), \\ \gamma &= \{x_{e,e'} : e' \in \mathcal{E}\} \end{aligned}$$

und $W' = (W \setminus (\mathcal{E} \times \{e\})) \cup (\{e\} \times \mathcal{E})$, und weiter gilt:

  * Falls $\sigma(e) = \mathsf{I}_T$, so ergänze γ zu $\gamma \cup \{x_T\}$ und setze $\xi' := \xi[T/\vartheta(e)]$, und ergänze ferner ψ zu $\psi \wedge x_T < \xi(T)$, falls $\xi(T) \neq \perp$ und $T \in \mathcal{T}_!$;
  * Falls $\sigma(e) = \mathsf{C}_T$, so muß $a := \xi(T) \neq \perp$ gelten, und ergänze ψ zu $\psi \wedge x_T < a$ und setze $\xi' := \xi[T/\perp]$;
  * Falls $\sigma(e) = \mathsf{O}_T$, so muß $a := \xi(T) \neq \perp$ gelten, und ergänze ψ zu $\psi \wedge x_T = a$ und setze $\xi' := \xi[T/\perp]$;
  * Sonst setze $\xi' := \xi$.

▶ Ansonsten enthält δ' nichts.

ξ fungiert also gleichzeitig als Marker für die Setzung der Timer: Falls $\xi(T) \neq \perp$, so läuft T gerade.

Beispiel 4.7 Als kleines Beispiel betrachten wir das HMSC in Abbildung 4.10, dessen Automat in Abbildung 4.11 gegeben ist. T sei ein Endtimer. In der Darstellung haben die Zustände vier Komponenten, wobei die ersten beiden der Konfiguration des normalen HMSC-endlich-Automaten entsprechen, die dritte die Marken-Menge ist und die vierte die partielle Funktion in Relationsschreibweise. Man beachte besonders, daß der $(1, *)$-Zustand kein Endzustand ist, weil T in ihm noch läuft. Das ist auch der Grund für die Bedingung $x_T < 10$ im I_T-Schritt: Wäre $T \notin \mathcal{T}_!$, so würde man das unbemerkte Ablaufen erlauben, diese Bedingung also weglassen.

Abbildung 4.10 Ein zu übersetzendes HMSC mit Timern (Beispiel 4.7)

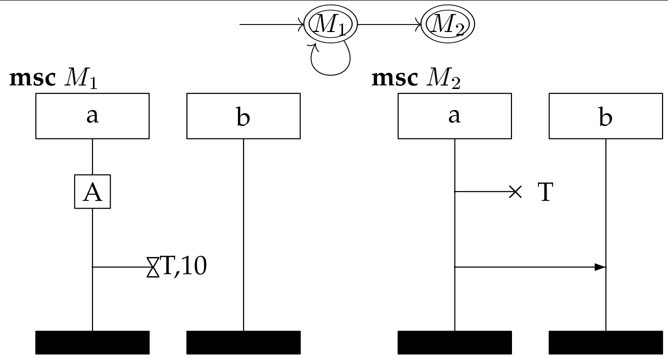
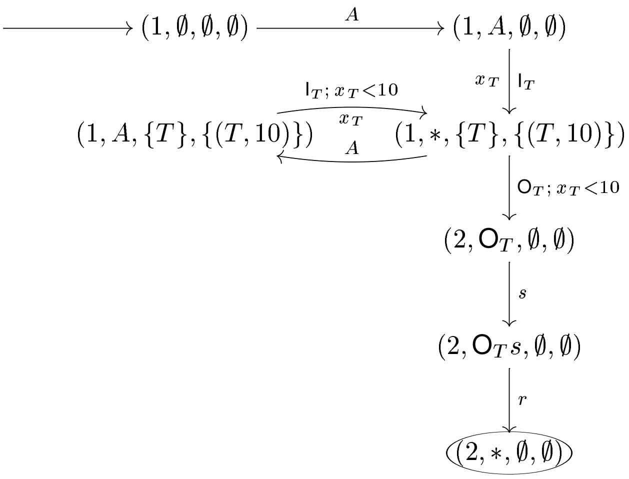


Abbildung 4.11 Der Automat zu Abbildung 4.10 (Beispiel 4.7)



Zu Interpretation 1: Es reichen wieder zwei Uhren wie für die Zeitbeschränkungen, die allerdings pro Timer-Setzung da sein müssen, nicht pro Timer: Man betrachte das Wort $\underset{1}{\mathsf{I}_T(5)}\,\underset{2}{\mathsf{I}_T(5)}\,\underset{3}{\mathsf{I}_T(3)}\,\underset{6}{\mathsf{O}_T}$. Merkt man sich nur den letzten und ersten Moment des Setzen des Timers, übersieht man, daß die zweite Setzung noch nicht gelöscht werde kann. Man kann allerdings auch feststellen, daß es bei dieser Semantik nicht sinnvoll ist, irgendwelchen Timern zu erlauben, abzulaufen, ohne daß dies in I_T bemerkt würde. Schließlich sollen *alle* Setzungen relevant sein, also muß man sie auch überprüfen. Daher ist die Erweiterung von ψ im I-Fall nicht von $T \in \mathcal{T}_!$ abhängig, wie auch schon bei der Semantik-Definition 4.14 motiviert.

Definition 4.16 (HMSC-Automat) Sei $H \in \mathfrak{H}_{\mathfrak{tt}}$. Sei $\mathcal{F}(\mathfrak{u}(H))$ gegeben als

$$\mathcal{F}(\mathfrak{u}(H)) = \langle C, \mathcal{E}, \delta, c_0, C_* \rangle.$$

Dann ist $\mathcal{F}_\star(H)$ definiert als der Uhrenautomat

$$\langle C', X, \mathcal{E}, \delta', c'_0, C'_* \rangle,$$

wobei gilt (vgl. Definition 4.9):

- ▷ $C' = C \times (\mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{I}))$;

▷ $C'_* = C_* \times (\mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{E}) \times \mathfrak{p}(\mathcal{I}\backslash\chi^{-1}(\mathcal{T}_!)))$;

▷ $c'_0 = (c_0, \emptyset)$;

▷ $X = \{x_{e_1,e_2}, y_{e_1,e_2} : e_1, e_2 \in \mathcal{E}\} \cup \{x_e, y_e : e \in \mathcal{I}\}$;

▷ ▶ Für jede Transition $\delta(s,e) = s'$ mit $e \in \mathcal{E}$ und jedes $W \in (\mathfrak{p}(\mathcal{E}))^2 \times \mathfrak{p}(T)$ ist $((s,W), e, \psi, \gamma, (s', W')) \in \delta'$, wobei

$$\begin{aligned}\psi &= \bigwedge_{(e',e)\in W} (x_{e',e} \in \tau(e,e') \wedge y_{e',e} \in \tau(e',e)),\\ \gamma &= \{y_{e,e'} : e' \in \mathcal{E}\} \cup \{x_{e,e'} : (e,e') \notin W \wedge e_i \in \mathcal{E}\}\end{aligned}$$

und $W' = (W\backslash(\mathcal{E} \times \{e\})) \cup (\{e\} \times \mathcal{E})$, und weiter gilt:

∗ Falls $\sigma(e) = \mathsf{I}_T$, so ergänze γ zu $\gamma \cup \{y_e\}$ falls $e \in W$ und sonst zu $\gamma \cup \{x_e, y_e\}$, und W' zu $W' \cup \{e\}$, und ψ zu

$$\psi \wedge \bigwedge_{e'\in W,\, \chi(e')=T} (x_{e'} < \vartheta(e') \wedge y_{e'} < \vartheta(e'));$$

∗ Falls $\sigma(e) = \mathsf{C}_T$, so muß $W \cap \chi^{-1}(T) \neq \emptyset$ gelten, und ergänze ψ zu

$$\psi \wedge \bigwedge_{e'\in W,\, \chi(e')=T} (x_{e'} < \vartheta(e') \wedge y_{e'} < \vartheta(e'))$$

und ändere W' zu $W'\backslash\chi^{-1}\{T\}$;

∗ Falls $\sigma(e) = \mathsf{O}_T$, so muß $W \cap \chi^{-1}(T) \neq \emptyset$ gelten, und ergänze ψ zu

$$\psi \wedge \bigwedge_{e'\in W,\, \chi(e')=T} (x_{e'} = \vartheta(e') \wedge y_{e'} = \vartheta(e'))$$

und ändere W' zu $W'\backslash\chi^{-1}\{T\}$.

▶ Ansonsten enthält δ' nichts.

Hier haben wir für die Timer keine Funktionskomponente, die den Timern ihren Initialisierungsstand zuweist, sondern mehrere Uhren. Die Komplexität wird also verlagert: Nicht so viel mehr Zustände wie in Definition 4.15, sondern mehr Uhren.

Kommen wir nun zu Interpretation 3 (vgl. Seite 82). Zunächst betrachten wir den generellen Fall einer solchartigen Interpretation bei irgendeinem HMSC.

Da wir bei einem Timer T mithalten müssen, wie häufig vorher I_T vorkam, um zu überprüfen, ob O_T stattfinden kann, müßten wir die Vorkommen zählen. Formal würden wir durch Projektion auf $\{\mathsf{I}_T, \mathsf{O}_T, \mathsf{C}_T\}$ und Umbenennen $\mathsf{I}_T \mapsto ($, $\mathsf{O}_T, \mathsf{C}_T \mapsto)$ die Sprache der vollständigen Klammerausdrücke erhalten,[14] die bekanntermaßen nicht regulär ist; somit kann im allgemeinen auch kein Uhrenautomat eine entsprechende Zählung durchführen ([B95, HU00]).

Allerdings setzen wir ja generell auf den endlichen Automaten auf, die in Kapitel 2 erzeugt wurden. Hier haben wir ja ebenfalls die Situation, daß eine Nachricht gleichhäufig empfangen wird wie gesendet, wodurch durch eine Projektion ebenfalls etwas wie ein vollständiger Klammerausdruck herauskommt. Wegen der schwache Konvergenz ist die Tiefe dieses Ausdrucks jedoch beschränkt, und so haben wir da doch keine Probleme. Für Timer müssen wir diese Tiefen-Beschränkung also auch ausdrücken.

[14] Wir interpretieren diese anders als die übliche Klammerung, da in $(_1(_2)_3)_4$ $\{(1,3),(2,4)\}$ gematcht wird statt $\{(2,3),(1,4)\}$ in der üblichen Semantik von Klammerausdrücken, aber es geht hier ja nur um die syntaktische Aussage.

Definition 4.17 (Timer-Beschränktheit) Sei $H \in \mathfrak{H}_{\mathfrak{tt}}$. Sei $\mathfrak{u}(H)$ schwach konvergent. Für jede einfache Schleife ll_1 sei

$$|\{e \in \mathcal{I}(\mathfrak{m}(l)) : \chi(e) = T\}| = |\{e \in \mathcal{O}(\mathfrak{m}(l)) \cup \mathcal{C}(\mathfrak{m}(l)) : \chi(e) = T\}|$$

für jedes $T \in \mathcal{T}$. Dann heißt H *timer-beschränkt*.

Analog könnte man auch diese Interpretation für Zeitbeschränkungen behandeln, wenn man voraussetzt, daß der Automat einfach gezeitet ist (Definition 4.7). Um die Definition nicht weiter zu verkomplizieren, betrachten wir hier aber nur Timer.

Definition 4.18 (HMSC-Automat) Sei $H \in \mathfrak{H}_{\mathfrak{tt}}$ Timer-beschränkt. Sei $\mathcal{F}(\mathfrak{u}(H))$ gegeben als

$$\mathcal{F}(\mathfrak{u}(H)) = \langle C, \mathcal{E}, \delta, c_0, C_* \rangle.$$

Sei $d = \sum_{l \text{ Schleife}, \sigma(e) = \mathsf{I}_T} \max\{\mathfrak{k}(e, e') : \sigma(e') \in \{\mathsf{O}_T, \mathsf{C}_T\} \text{ in } \mathscr{L}(l)\}$. Dann ist $\mathcal{F}_1(H)$ definiert als der Uhrenautomat

$$\langle C', X, \mathcal{E}, \delta', c'_0, C'_* \rangle,$$

wobei gilt:

- ▷ $C' = C \times [(\mathcal{T} \times \{1, \ldots, d\}) \dashrightarrow \mathrm{range}(\vartheta)]$;
- ▷ $C* = C \times \{\xi \in [(\mathcal{T} \times \{1, \ldots, d\}) \dashrightarrow \mathrm{range}(\vartheta)] : \forall T \in \mathcal{T}_!.\xi(T, \_) = \bot\}$;
- ▷ $c'_0 = (c_0, \emptyset)$;
- ▷ $X = \{x_{T,1}, \ldots, x_{T,d} : T \in \mathcal{T}\}$;
- ▷ ▶ Für $\delta(s, e) = s'$ und $\xi \in [(\mathcal{T} \times \{1, \ldots, d\}) \dashrightarrow \mathrm{range}(\vartheta)]$ setzen wir $((s, \xi), e, \psi, \gamma, (s', \xi')) \in \delta'$, wobei gilt:
  - ∗ Falls $\sigma(e) = \mathsf{I}_T$, so sei $a = \min\{b : \xi(T, b) = \bot\}$, und setze dann $\gamma := x_{T,a}$, $\xi' = \xi[(T, a)/\vartheta(e)]$ und $\psi :=$ TRUE.
  - ∗ Falls $\sigma(e) = \mathsf{O}_T$, so muß $\xi(T, 1) \neq \bot$ sein, und setze

$$\begin{aligned} \psi &:= x_{T,1} = \xi(T, 1), \\ \gamma &:= \{x_{T,1} \leftarrow x_{T,2}, \ldots, x_{T,d-1} \leftarrow x_{T,d}, x_{T,d} \leftarrow \bot\} \text{ und} \\ \xi' &:= \xi[(T, 1)/\xi(T, 2), \ldots, (T, a-1)/\xi(T, a), (T, a)/\bot].^{15} \end{aligned}$$

  - ∗ Falls $\sigma(e) = \mathsf{C}_T$, so muß $\xi(T, 1) \neq \bot$ sein, und setze

$$\begin{aligned} \psi &:= x_{T,1} < \xi(T, 1), \\ \gamma &:= \{x_{T,1} \leftarrow x_{T,2}, \ldots, x_{T,d-1} \leftarrow x_{T,d}, x_{T,d} \leftarrow \bot\} \text{ und} \\ \xi' &:= \xi[(T, 1)/\xi(T, 2), \ldots, (T, a-1)/\xi(T, a), (T, a)/\bot]. \end{aligned}$$

  - ∗ Ansonsten setze $\gamma := \emptyset$, $\psi :=$ TRUE und $\xi' := \xi$.

  ▶ Ansonsten enthält δ' nichts.

[15] Hier benutzen wir also nicht nur Rücksetzungsupdates, sondern die etwas allgemeinere Form. Man kann sich leicht überlegen, daß man mit einer zusätzlichen Marken-Komponente auch nur mit Rücksetzungsupdates auskäme.

Wir fordern also, im Einklang mit Definition 4.14, daß die Timer in genau der Reihenfolge abgefragt werden, wie gesetzt werden.

Satz 4.3 Sei $H \in \mathfrak{H}_{\mathrm{tt}}$.

(a) $\mathcal{L}(\mathcal{F}_\star(H)) = [\![H]\!]_\star$, $\mathcal{L}(\mathcal{F}_>(H)) = [\![H]\!]_>$.

(b) Ist H schwach konvergent, so sind $\mathcal{F}_\star(H)$ und $\mathcal{F}_>(H)$ endlich.

(c) Ist H timer-beschränkt, so ist $\mathcal{L}(\mathcal{F}_1(H)) = [\![H]\!]_1$.

Beweis: (b) ist klar, der Beweis zu (a) läuft genau wie in Satz 4.2. Für (c) ist das einzig interessante, ob wir mit den Uhren auskommen, d. h. ob in den l_T-Transitionen irgendwann mehr als d Uhren benötigt würden. Zum einen ist d wohldefiniert wegen der schwachen Konvergenz. Darum kann ein Timer nicht beliebig lange nach seiner Setzung abgefragt werden. Daß er überhaupt genügend häufig abgefragt wird, wird durch die Timer-Beschränktheit sichergestellt. Gesamt kommen wir mit d Uhren aus. ⊛

Was die Komplexität der Automaten betrifft, so ist davon auszugehen, daß $|\mathcal{T}|$ und $|\{e, e' : \tau(e, e') \notin \{\bot, \mathbb{N}\}\}|$ eher klein sind. Damit ist der Hauptkomplexitätsfaktor für die Zustandszahl $\mathfrak{p}(\mathcal{E})$. Schlimmstenfalls muß man hier tatsächlich mit $2^{|\mathcal{E}|}$ rechnen, aber in der Regel werden die meisten Zeiteinschränkungen trivial sein, sodaß in der Praxis der Komplexitätsunterschied zu den ungezeiteten Automaten nicht so groß sein wird.

Es ist klar, daß auf ganz natürliche Weise die Automaten auch mit unterschiedlichen Semantiken für das Verhalten bzgl. der Zeitbeschränkungen und bzgl. der Timer erstellt werden können. Ja man könnte sogar die Semantiken für jede Beschränkung oder je nach Timer unterschiedlich wählen – Das macht in dem hier vorgestellten Modell keine prinzipiellen Schwierigkeiten. Der hier gewählte Ansatz wurde vor allem aus didaktischen Gründen benutzt, da die formale Definition der Automaten mit diesen Wahlmöglichkeiten noch viel unübersichtlicher wäre als es jetzt schon der Fall ist.

Man beachte ferner, daß wir bei genügend wenigen Beschränkungen in der Situation der *Event Clock Automata* aus [AFH98] sind, die eine Determinisierung erlauben.

4.c.iv Austauschbarkeit von Beschränkungen/Timern

Der Leser mag sich nun fragen, warum man überhaupt die beiden unterschiedlichen Konstrukte der Zeitbeschränkungen und der Timer eingeführt hat.

Zunächsteinmal sollte der Standard in dieser Hinsicht unterstützt werden, und dieser definiert nun einmal Timer und auch, daß diese nur von einem Prozeß benutzt werden dürfen ([ITU99]). Um Beschränkungen an die Nachrichtenlaufzeit zu erhalten, sind wir daher auf die Zeitbeschränkungen unserer Notation angewiesen.

Will man Timer durch Zeitbeschränkungen ersetzen, so verliert man dadurch die Möglichkeit, gewisse Pfade in einem HMSC durch die Semantiken als ungültig zu verwerfen, nämlich die, wo ein Timer benutzt wird, ohne gesetzt zu sein. Mit Timern stehen uns außerdem Endtimer zur Verfügung.

Beschränkungen durch Timer zu ersetzen, ist auch nicht möglich, wie eben schon für Nachrichtenlaufzeiten erwähnt. Auf einer Prozeßlinie klappt das, wenn man verlangt, daß Timer-Benutzungen gleichzeitig mit anderen Ereignissen durchgeführt werden müssen. Das ist aber so nahe dran an den Zeitbeschränkungen, daß eine Unterscheidung sinnlos ist.

4.d Aktionsdauer

Als nächstes betrachten wir nun die Möglichkeit, eine Dauer von Aktionen zu betrachten. Das ist im Standard nicht vorgesehen und ist meines Wissens bisher auch noch nicht untersucht worden. Insbesondere soll das die Robustheit der Übersetzung verdeutlichen.

Stichwortartig werden nur die Erweiterungen der vorigen Definitionen angegeben. Die Nomenklatur ist der aus den Automaten-Definitionen 4.8, 4.9, 4.15, 4.16 und 4.18 nachempfunden. Eine Unterscheidung nach den verschiedenen Interpretationen (Seiten 77 und 87) ist sinnlos, da eine Aktion ohnehin erst dann wieder starten kann, sobald sie beendet ist.

(1) In einem MSC mit Timern (Definition 4.10) wird die Funktion τ erweitert um $\tau : \mathcal{A} \to \mathbb{T}$, und drückt dann die *Dauer einer Aktion* aus.[16]

(2) Für eine gültige Linearisierung eines MSCs oder die Semantik eines Laufes eines HMSCs fordern wir zusätzlich, daß während der Durchführung einer Aktion kein weiteres Ereignis auf dem Prozeß stattfindet, oder zumindest keines, was den Prozeß aktiv fordert, worunter wir alles außer Time-Outs und Nachrichtenempfängen verstehen. Für ein gültiges $\frac{w}{x}$ muß also gelten, je nach Wahl der Semantik:

   (i) Falls $w_i \in \mathcal{A}_p$, so ist $t_j > t_i + \tau(w_i)$ für alle $j > i$ mit $o(w_j) = p$.

   (ii) Falls $w_i \in \mathcal{A}_p$, so ist $t_j > t_i + \tau(w_i)$ für alle $j > i$ mit $o(w_j) = p$ und $w_j \notin \mathcal{R} \cup \mathcal{O}$.

(3) In die graphische Syntax fügen wir ein, daß man neben einer Aktion ihre Länge angeben kann; macht man dies nicht, werde als Länge 0 angenommen.

(4) In den Automaten fügen wir für jeden Prozeß p eine zusätzliche Uhr x_p ein. W wird um eine Komponente $W_\mathcal{A}$ aus $\mathfrak{p}(\mathcal{A})$ erweitert.[17] Offenbar ist $|W_\mathcal{A}| \leqslant |\mathcal{P}|$ in jedem erreichbaren Zustand.

(5) Wir erweitern die Transitionen für $e \in \mathcal{E}$ wie folgt:

   (a) Falls

      (i) $o(e) = p$ bzw.

      (ii) $o(e) = p$ und $e \notin \mathcal{R} \cup \mathcal{O}$,

   so sei A das eindeutig zu $p \in \mathcal{P}$ bestimmte $A \in W_\mathcal{A}$ mit $o(A) = p$, falls es existiert. Dann erweitere ψ zu $\psi \wedge x_p > \tau(A)$ und setze $W'_\mathcal{A} := W_\mathcal{A} \backslash \{A\}$. Falls dieses A nicht existiert, so läuft gerade keine Aktion und ψ und $W'_\mathcal{A}$ werden nicht angetastet.

   (b) Falls $e \in \mathcal{A}$, so erweitere $W'_\mathcal{A}$ um $\{e\}$ und erweitere γ zu $\gamma \cup \{x_p\}$.

[16] Eine Erweiterung auf $\tau : \mathcal{A} \to \mathbb{I}$ ist leicht möglich, falls man Mindest- und Höchstdauer anzeigen möchte.

[17] Es braucht für jeden Prozeß nur eine Uhr, da immer nur eine Aktion durchgeführt werden kann. Wir brauchen aber Marker für alle Aktionen, weil die Aktionen verschiedene Längen haben können.

Damit werden die Zeitbeschränkungen bzgl. der Anfangszeiten der Aktionen interpretiert.

Beispiel 4.8 Wir wollen uns nun ein Beispiel anschauen, in dem die in diesem Kapitel vorgestellten Konzepte benutzt werden. Modelliert werden soll folgende Kommunikation:

▷ Zwei Prozesse p und q sind beteiligt.

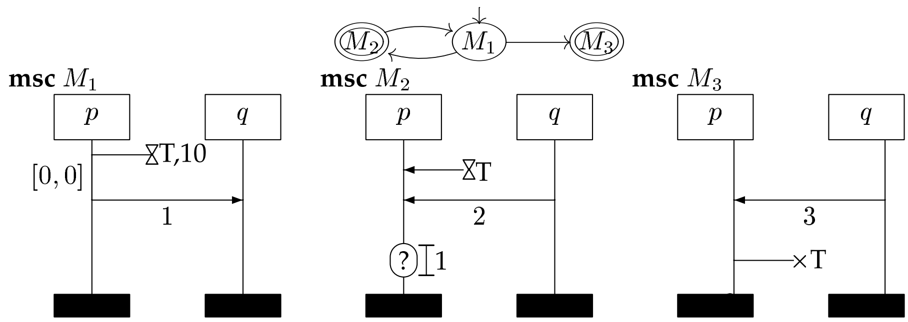
▷ Prozeß p sendet eine Nachricht an q und wartet 10 Sekunden lang auf Antwort.

▷ q braucht, falls er antwortet, höchstens 2 Sekunden dafür.

▷
- ▶ Falls sie vor 10 Sekunden kommt, ist die Kommunikation beendet.
- ▶ Falls sie später kommt, so überlegt sich p höchstens 1 Sekunde lang, ob ein neuer Versuch gestartet werden soll, und die Kommunikation ist dann dementsprechend entweder beendet oder beginnt von neuem.

Eine sinnvolle Modellierung besteht aus dem HMSC mit Timern und Aktionsdauern aus Abbildung 4.12. Wieder setzen wir gemäßigt asnychrone Konkatenation

Abbildung 4.12 Ein zu übersetzendes HMSC mit Aktionsdauern (Beispiel 4.8)



voraus und lassen alle trivialen Beschränkungen weg. Es fehlt damit nur noch die Angabe $\tau(r_1, s_2) = \tau(r_1, s_3) = [0, 2]$. Da wir die Nachrichten von q als äquivalent erachten, müssen wir dies durch die gewählte Funktion π ausdrücken. Die beiden alternativen Semantiken der Aktionsblockaden fallen hier zusammen.

Damit die Darstellung irgendwie nachvollziehbar bleibt, ist zunächst $\mathcal{F}_0(H)$ in Abbildung 4.13 angegeben. Nun erhalten wir gemäß unserer Vorüberlegung den durch Abbildung 4.14 hinreichend genau bestimmten Automaten $\mathcal{F}_>(H)$; wir benötigen dafür nur die Uhren $x_T, x_{r_1,s_2}, x_{r_1,s_3}, x_{|_T,s_1}, x_p$. Wir sehen, daß wir hier nur zwei Zustände mehr benötigen.

Man sieht hier sehr schön, daß eine sinnvolle Erweiterung der Modelierungsfähigkeit die Konstruktion der Automaten nur wenig verkompliziert. Will man nun ein Echtzeitsystem modellieren, bei welchem der Vorgang des Sendens einer Nachricht eine nicht vernachlässigbare Zeitspanne benötigt, so ist die nötige Erweiterung wieder sehr natürlich machbar.

Abbildung 4.13 Der Hilfsautomat zu Abbildung 4.12 (Beispiel 4.8)

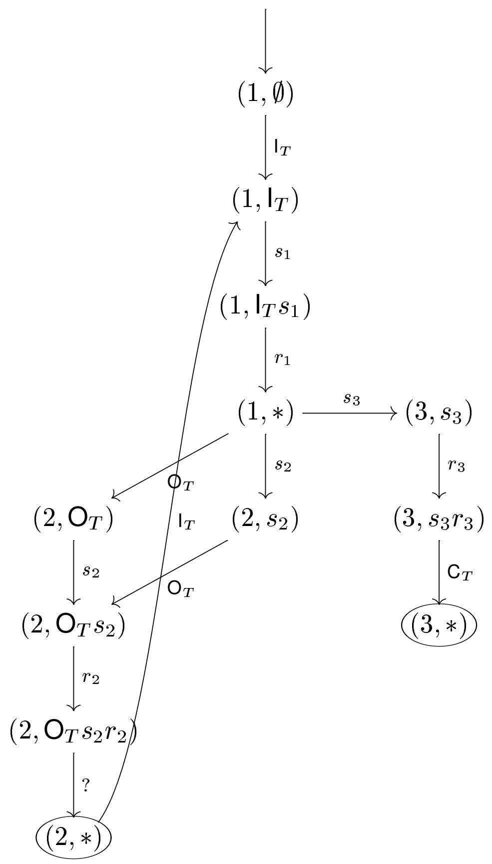
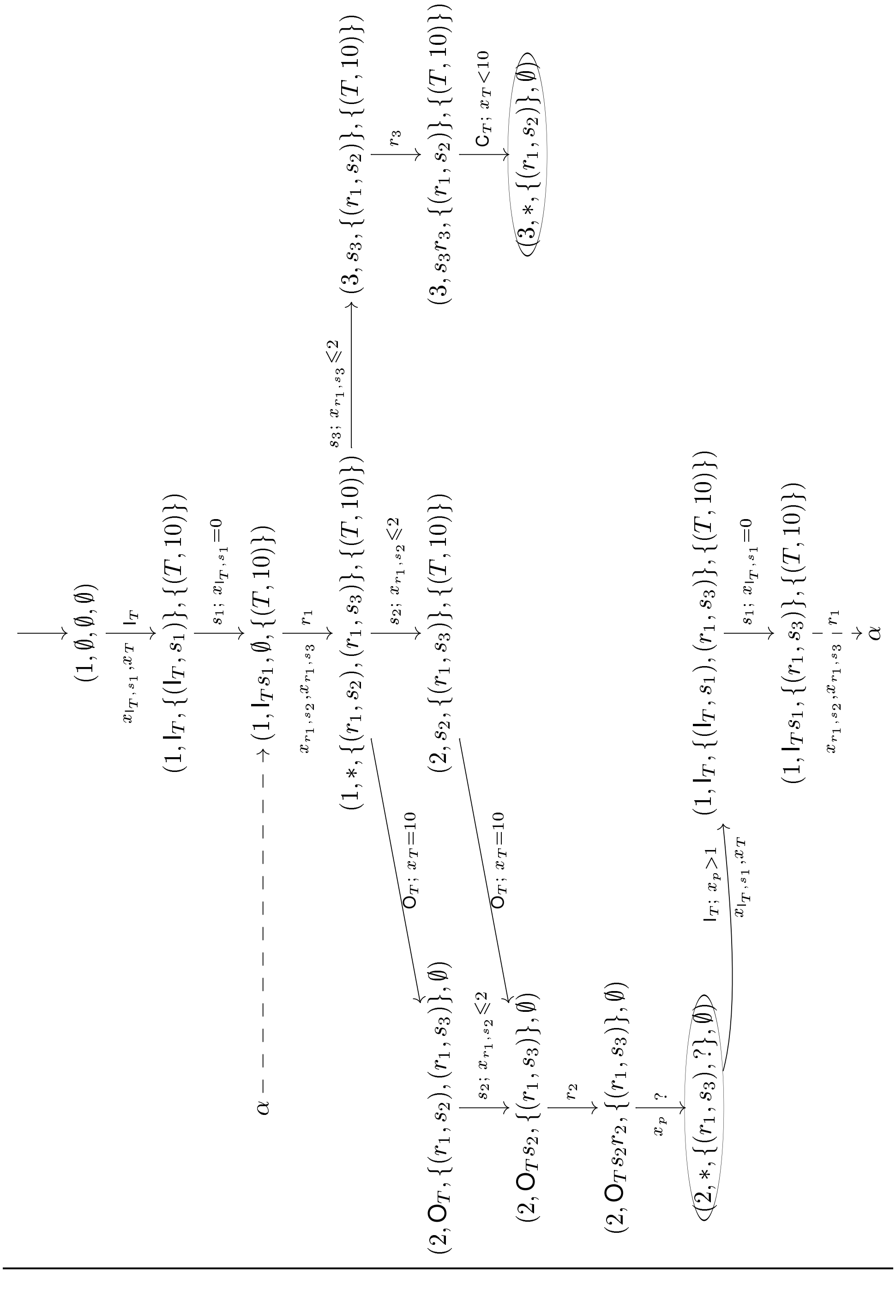


Abbildung 4.14 Der Automat zu Abbildung 4.12 (Beispiel 4.8)



Kapitel 5

Erweiterungen von MSCs

"Life," said Marvin, "don't talk to me about life."
DOUGLAS ADAMS, *The Hitch Hiker's Guide to the Galaxy*, 1979

In diesem Kapitel sollen diverse Erweiterungen der Grundformen der MSCs vorgestellt werden, nämlich Kompositionalität und Live Sequence Charts. Das ist im Standard nicht vorgesehen, eröffnet aber neue Einsatzbereiche und erlaubt leichtere und kürzere Modellierungen, wie man sehen wird. Insbesondere dient dies auch als Test für die Erweiterbarkeit der Modellierung aus Kapitel 2.

5.a Liveness

5.a.i Einführung

Neben der Unfähigkeit, gewisse Kommunikationsformen überhaupt ausdrücken zu können (was wir in Abschnitt 5.b untersuchen), ist ein weiteres Problem, daß alle Ereignisse in einem MSC stattfinden *müssen*. Unter Umständen möchte man aber beim Entwurf angeben, daß eine Nachricht vielleicht gesendet wird, vielleicht aber auch nicht. Natürlich kann man dazu ein HMSC mit mehreren Alternativen benutzen, doch die Einfachheit leidet.

Daher wurde das Konzept der *Liveness* eingeführt ([DH99, KW01]). Informell gesprochen fordert man nun, daß ein als *heiß* deklariertes Ereignis stattfinden muß, ein als *kalt* deklariertes hingegen nicht. Falls ein kaltes Ereignis stattfindet, muß es die üblichen Restriktionen der Ordnung und des Zeitverhaltens erfüllen. Wir führen ferner eine Wärmeleitungsrelation ein, die kalte Ereignisse miteinander assoziiert. Sie soll dafür da sein, daß man angeben kann, daß ein Ereignis nicht stattfinden muß, aber falls doch, es ein anderes bedingt. Damit kann etwa folgende informelle Forderung formalisiert werden: „Eine Nachricht muß nicht gesendet werden, aber wenn ja, wird sie auch empfangen, und umgekehrt. Wenn sie empfangen wird, führt der empfangende Prozeß auf jeden Fall eine Aktion durch, ansonsten nur vielleicht." Durch die Wärmeleitungsrelation können wir also sowohl Vor- als auch Nachbedingungen festlegen.

Definition 5.1 (LSC) Sei $\langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell \rangle$ ein MSC (Definition 2.1). Dann heißt

$$\langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell, \mathcal{H}, \rightsquigarrow \rangle$$

Live Sequence Chart (LSC), wobei gilt:

- ▷ $\mathcal{H} \subseteq \mathcal{E}$ (eine Menge der *heißen* Ereignisse), wobei $\mathcal{K} := \mathcal{E}\backslash\mathcal{H}$ die Menge der *kalten* Ereignisse ist und $m \cap ((\mathcal{K} \times \mathcal{H}) \cup (\mathcal{H} \times \mathcal{K})) = \emptyset$ gilt;
- ▷ $\rightsquigarrow \subseteq \mathcal{K} \times \mathcal{K}$ (eine *Wärmeleitungsrelation*) mit $\forall e \in \mathcal{S}.(e \rightsquigarrow m(e) \leftrightarrow m(e) \rightsquigarrow e)$

$\mathfrak{M}_\mathsf{l}$ ist analog zu $\mathfrak{M}$ definiert. Für ein $M \in \mathfrak{M}_\mathsf{l}$ ist $\varkappa(M) := \langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell \rangle \in \mathfrak{M}$ das *erhitzte LSC*.

$e \rightsquigarrow f$ bedeutet, daß das Vorkommen von e in einer Linearisierung das Vorkommen von f erzwingt. $\rightsquigarrow \subseteq \prec$ ist nicht gefordert. Insbesondere kann $e \rightsquigarrow f \rightsquigarrow e$ gelten. Das muß sogar der Fall sein, wenn $m(e) = f$ oder $m(f) = e$. Die Zusatzbedingung an m filtert die Inkonsistenzen heraus, die durch eine unsinnige Wärmeverteilung eingeführt würden. Beispiel 5.1 zeigt, was passieren kann, beachtete man dies nicht.

Die Semantik eines LSCs ist die Menge aller Linearisierungen von $\prec$, bezogen auf beliebige Mengen von Ereignissen, die alle heißen enthalten und mit $\rightsquigarrow$ konform gehen.

Definition 5.2 (LSC-Semantik) Sei $M \in \mathfrak{M}_\mathsf{l}$.

(a) Ein $\xi \subseteq \mathcal{E}$ mit $\xi \supseteq \mathcal{H}$ heißt *gleichmäßig warm*, falls $(\xi \times (\mathcal{E}\backslash\xi)) \cap \rightsquigarrow = \emptyset$. $\mathfrak{w}(M)$ ist die Menge aller gleichmäßig warmen Teilmengen von $\mathcal{E}$.

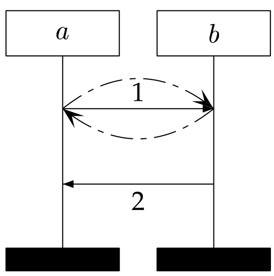
(b) Sei $\xi \in \mathfrak{w}(M)$. Sei M' das LSC, welches aus M entsteht, wenn man die Ereignismenge auf ξ einschränkt und die restlichen Parameter entsprechend angleicht. Dann setzen wir $M|_\xi := \varkappa(M')$.

(c) Die *Semantik von M unter π* ist

$$[\![M;\pi]\!] := \bigcup_{\xi \in \mathfrak{w}(L)} [\![M|_\xi;\pi]\!].$$

Ein $\pi = \mathrm{id}$ wird wie üblich weggelassen.

Abbildung 5.1 Ein erstes LSC (Beispiel 5.1)



Beispiel 5.1 Man betrachte das einfache LSC L in Abbildung 5.1. Wir werden $\rightsquigarrow$ durch strichpunktierte Linien mit dickem Pfeilkopf darstellen. Sollten kalte Ereignisse nicht Anfangs- oder Endpunkt einer $\rightsquigarrow$-Linie sein, wird die Partitionierung in

$\mathcal{K}$ und $\mathcal{H}$ gesondert angegeben. Ansonsten sind alle nichts mit $\rightsquigarrow$ zu tun habenden Ereignisse heiß.

Das LSC hat genau zwei gültige Linearisierungen: $s_1 r_1 s_2 r_2$ und $s_2 r_2$. $\mathcal{E}$ ist immer gleichmäßig warm. r_1 ist genau dann in einer gleichmäßig warmen Menge H', falls s_1 in ihr ist. Somit ist $\mathfrak{w}(M) = \{\mathcal{E}, \{s_2, r_2\}\}$. Die Linearisierung für $\mathcal{E}$ ist klar, bei $\{s_2, r_2\}$ muß $r_1 \prec s_2$ nicht beachtet werden, weil r_1 nicht zur betrachteten Ereignismenge gehört. Natürlich ist die Schnittkante $s_1|r_1 s_2 r_2$ nicht gleichmäßig warm. Das macht aber nichts, da diese Eigenschaft ja nur für die Gesamtmenge abgefragt wird: Daß zwischendurch ungleichmäßig warme Mengen auftreten, läßt sich nicht vermeiden.

Hätte man nur $s_1 \rightsquigarrow r_1$ gefordert, so wäre $\{r_1, s_2, r_2\}$ gleichmäßig warm. $[\![H]\!]$ würde also ein inkonsistentes Wort (Definition 2.5) enthalten. Daher wird soetwas in der Syntaxdefinition ausgeschlossen (Definition 5.1, Lemma 5.1).

Lemma 5.1 Ist $M \in \mathfrak{M}_\mathsf{l}$, so ist jedes $w \in [\![M]\!]$ konsistent.

Beweis: klar. ⊛

Definition 5.3 (LSC-Konkatenation) Seien $L', L'' \in \mathfrak{M}_\mathsf{l}$. Sei $k \subseteq \mathcal{E}' \times \mathcal{E}''$. Dann ist die *Konkatenation von L' mit L'' unter k*, $L' \circ^k L''$, definiert als das LSC $L \in \mathfrak{M}_\mathsf{l}$, sodaß $L = \langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell, \mathcal{H}, \rightsquigarrow \rangle$ mit $\rightsquigarrow = \rightsquigarrow' \cup \rightsquigarrow''$ und $\mathcal{H} = \mathcal{H}' \cup \mathcal{H}''$, und alles andere wie im MSC-Falle gilt (Definition 2.8).

Die Wärmeleitungsrelation wird also *nicht* über MSC-Grenzen ausgedehnt. Das hängt damit zusammen, daß eine solche Ausdehnung Schwierigkeiten mit sich bringt, die teilweise auf die gleiche Weise wie bei Timern in Kapitel 4 aufgelöst werden müssen und teilweise erhebliche Einschränkungen an eine solche Ausdehnung erfordern, um ein Analogon der Konvergenz sicherzustellen. Ich werde auf Seite 108 darauf zurückkommen.

High-Level LSCs sind genau analog zu High-Level MSCs definiert.

Definition 5.4 (HLSC) Die Syntax und Semantik von HMSCs (Definitionen 2.10, 2.12 und 2.13) überträgt sich vollständig auf HLSCs. $\mathfrak{m}(l)$ ist also das durch ein Lauffragment l definierte LSC. Die Menge aller HLSCs mit den Parametern $\mathcal{P}, \Lambda$ ist $\mathfrak{H}_\mathsf{l}(\mathcal{P}, \Lambda)$ (oder kürzer wie üblich).

5.a.ii Automaten

Man erkennt, daß die alten Definitionen der Schnittkanten und Konfigurationen (Definitionen 2.7 und 2.22) für uns nicht mehr zu gebrauchen sind, um im Sinne von Definition 2.23 gültige Konfigurationsübergänge zu bestimmen.

In den neuen Konfigurationen muß man, wie üblich, mithalten, welche Ereignisse keine Probleme mit der Präzedenz bereiten können. Dies erledigt, ebenfalls wie üblich, die Schnittkanten-Komponente. Sie wird geändert, wenn ein Ereignis stattfindet, aber auch, wenn ein kaltes eben definitiv als nicht-stattfindend festgehalten wird. Zusätzlich müssen wir jedoch noch Kompatibilität mit $\rightsquigarrow$ überprüfen. Dafür halten wir fest, welche Ereignisse tatsächlich stattfanden: Wir benutzen eine zusätzliche Komponente.

Definition 5.5 (LSC-Schnittkante) (vgl. Definition 2.7) Sei $M \in \mathfrak{M}_\mathfrak{l}$. Seien gegeben $\Xi \subseteq K \subseteq \mathcal{E}$. Dann heißt (K, Ξ) *LSC-Schnittkante von* M, falls es ein $\Xi' \in \mathfrak{w}(M)$ mit $\Xi' \cap (K \setminus \Xi) = \emptyset$ gibt, sodaß $\Xi \in \mathfrak{K}(M|_{\Xi'})$ (also insbesondere $\Xi \subseteq \Xi'$). Die Menge aller LSC-Schnittkanten von M ist $\mathfrak{K}_\mathfrak{l}(M)$.

Definition 5.6 (HLSC-Konfiguration) Sei $H \in \mathfrak{H}_\mathfrak{l}$. Eine *Konfiguration von* H ist ein Paar (h, φ), wobei gilt (vgl. Definition 2.22):

(i) $h = h_1 \dots h_{|h|} \in N^*$ und

(ii) $\varphi = \varphi_1 \dots \varphi_{|h|}$, wobei alle $(K_i, \Xi_i) := \varphi_i \in \mathfrak{K}_\mathfrak{l}(M_{h_i})$.

Die Menge aller Konfigurationen sei $\mathcal{K}_\mathfrak{l}(H)$. Wir setzen $\mathfrak{e}(\varphi) := (\bigcup_{i=1}^{|h|} K_i, \bigcup_{i=1}^{|h|} \Xi_i)$.

Beispiel 5.2 Damit das etwas klarer wird, soll ein einzelnes LSC als Illustration dienen. Wie in Beispiel 2.15 ist das auch hier zur Erläuterung da: Die eigentliche Übersetzung erhält HLSCs als Eingaben und läuft technisch etwas anders.

Abbildung 5.2 Ein Automat zu Abbildung 5.1 (Beispiel 5.2)

$$\begin{tikzcd}
 & (s_1, \emptyset) \arrow[r, "\varepsilon"] & (s_1 r_1, \emptyset) \arrow[r, "s_2"] & (s_1 r_1 s_2, s_2) \arrow[r, "r_2"] & (\mathcal{E}, s_2 r_2) \\
\arrow[r] & (\emptyset, \emptyset) \arrow[u, "\varepsilon"] \arrow[r, "s_1"] & (s_1, s_1) \arrow[r, "r_1"] & (s_1 r_1, s_1 r_1) \arrow[r, "s_2"] & (s_1 r_1 s_2, s_1 r_1 s_2) \arrow[r, "r_2"] & (\mathcal{E}, \mathcal{E})
\end{tikzcd}$$

Man betrachte das LSC aus Beispiel 5.1 (Abbildung 5.1). In Abbildung 5.2 sind die φ-Komponenten eines möglichen Automatens angegeben. Am Anfang ist noch kein Ereignis gelesen worden: Der Startknoten ist mit $(\emptyset, \emptyset)$ beschriftet. Dann kann auf die übliche Weise die Folge $s_1 r_1 s_2 r_2$ gelesen werden. Die Ereignisse werden in der K-Komponente wie auch in der Ξ-Komponente gespeichert. Um ein Ereignis e zu überspringen, benutzen wir ε-Transitionen. Hier wird e nur in K gespeichert, nicht jedoch in Ξ. In der oberen Reihe in Abbildung 5.2 wird die zweite Linearisierung des LSCs erkannt. In $(s_1, \emptyset)$ kann keine r_1-Transition erfolgen, weil dadurch eine ungültige Schnittkante entstünde: Die Ξ-Bedingung aus Definition 5.5 ist verletzt.

Angenommen, wir haben in einem LSC zwei Ereignisse $e' \prec e$ mit $e' \in \mathcal{K}$. Seien in einer Konfiguration beide noch nicht gelesen. Steht nun als nächstes e auf der Eingabe, so kann niemals mehr e' gelesen werden. Gilt desweiteren $e' \rightsquigarrow e''$, so kann damit auch e'' niemals gelesen werden. Es lohnt also, e' und e'' dementsprechend zu markieren – Tatsächlich wird das sich sogar als notwendig erweisen, um mit dem zu definierenden κ und der zu erklärenden l-Konvergenz endliche Automaten zu erhalten (Definitionen 5.8 und 5.10, Satz 5.2). Kehren wir zurück zu Beispiel 5.2, so heißt das, daß wir etwa eine s_2-Transition $(s_1, \emptyset) \to (s_1 r_1 s_2, s_2)$ erlauben sollten.

Sei allgemein $(h, \varphi) \in \mathcal{K}_\mathfrak{l}(H)$ für ein $H \in \mathfrak{H}_\mathfrak{l}$, wobei $\varphi_i = (K_i, \Xi_i)$ für jedes i. Sei $\varphi^{(\nu)} = \varphi_1^{(\nu)} \dots \varphi_{|\varphi|}^{(\nu)}$ eine Folge von LSC-Schnittkanten mit $\varphi_i^{(\nu)} = (K_i^{(\nu)}, \Xi_i^{(\nu)})$ und $\mathfrak{e}(\varphi^{(\nu)}) \in \mathfrak{w}(\mathfrak{m}(h))$, sodaß gilt:

(i) $\Xi_i^{(\nu)} \supseteq \Xi_i$, $K_i^{(\nu)} \supseteq K_i$,

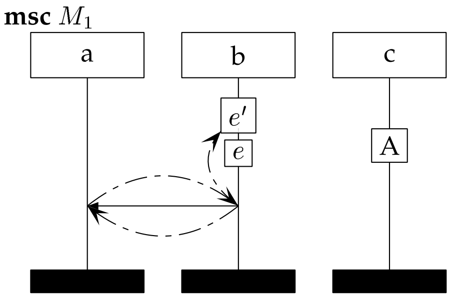
(ii) $\Xi_i^{(\nu)} \cap (K_i \backslash \Xi_i) = \emptyset$ und

(iii) $\forall e \in \mathcal{E}.\forall e' \prec e.\forall e'' \in \mathcal{K}.(e'' \rightsquigarrow^* e' \wedge e \in \Xi_i \wedge e' \notin \Xi_i \to e'' \in K_i^{(\nu)})$.

$\varphi^{(\nu)}$ repräsentiert also eine mögliche Vervollständigung von φ, berücksichtigend, daß alle tatsächlich gelesenen Ereignisse auch in $\varphi^{(\nu)}$ gelesen worden sind und alle definitiv nicht gelesenen auch dort nicht gelesen worden sind. Die dritte Bedingung gibt genau die Nicht-Lesbarkeit an: Wurde ein kaltes Ereignis schon nicht gelesen, das von einen anderen Ereignis (mittelbar oder direkt) gefordert wird, so kann dieses andere Ereignis nicht gelesen werden. Sei S die Menge aller solcher $\varphi^{(\nu)}$. Sei weiter $\varphi_i' = (((\bigcap_{\varphi^{(\nu)} \in S} K_i^{(\nu)}) \cap \mathcal{K}) \cup K_i, \Xi_i)$. Dann setzen wir $\mathfrak{u}(h, \varphi) = (h, \varphi')$.

Beispiel 5.3 Sei gegeben das LSC aus Abbildung 5.3. Nur A sei heiß.

Abbildung 5.3 Zu Beispiel 5.3



Offenbar ist

$$\mathfrak{w}(M_1) = \{\mathcal{E}, \{e', A\}, \{e', e, A\}, \{e', s, r, A\}, \{e, A\}, \{A\}\}.$$

Sei zunächst gegeben die Konfiguration $(1, \varphi)$ mit $\varphi = (\{e\}, \{e\}) =: (K, \Xi)$. Da e gelesen wurde, kann e' nicht mehr gelesen werden. Da s aber e' voraussetzt, kann auch s niemals mehr kommen, und r entsprechend auch nicht. Es sollte also $\mathfrak{u}(1, \varphi) = (1, (\{e, s, r, e'\}, \{e\}))$ gelten.

Bedingung (i) läßt von $\mathfrak{w}(M_1)$ noch

$$\{\{e', e, s, r, A\}, \{e', e, A\}, \{e, A\}\}$$

als Kandidaten für $\Xi_1^{(\nu)}$ übrig. Da $\{e\} \backslash \{e\} = \emptyset$, sagt Bedingung (ii) nichts aus. Bei Bedingung (iii) hingegen gilt zunächst, daß $e' \prec e$ mit $e' \rightsquigarrow^* e'$ und $e \in \Xi$, $e' \notin \Xi$. Damit muß $e' \in K_1^{(\nu)}$ für jedes ν der S-Menge gelten. Da weiter $s \rightsquigarrow e' \prec e$ und $r \rightsquigarrow s \rightsquigarrow e' \prec e$, muß auch $s, r \in K_1^{(\nu)}$ sein. Also muß jedes $\varphi_1^{(\nu)}$ von der Gestalt $(K_1^{(\nu)}, \Xi_1)$ sein mit $\{s, r, e'\} \cup \Xi_1^{(\nu)} \subseteq K_1^{(\nu)}$. Somit bleibt übrig:

$$S = \{(\mathcal{E}, \mathcal{E}), (\mathcal{E}, \{e', e, A\}), (\mathcal{E}, \{e, A\})\}.$$

Damit ist

$$\varphi_1' = (((\mathcal{E} \cap \mathcal{E} \cap \mathcal{E}) \cap \mathcal{K}) \cup K_1, \Xi_1) = (\{e, e', s, r\} \cup \{e\}, \{e\}) = (\{e, e', s, r\}, \{e\}),$$

also genau was wir wollten.

Gehen wir dagegen aus von der Konfiguration $(1, \varphi)$ mit $\varphi = (\{A, e', e\}, \{A, e'\})$. Sie mag dadurch entstanden sein, daß zunächst e' und A gelesen wurden und man sich dann entschied, e durch ε zu überspringen. Wir erwarten, daß $\mathfrak{r}(1, \varphi) = (1, \varphi)$. Bedingung (i) ergibt als Kandidaten für $\Xi_1^{(\nu)}$

$$\{\{e', e, s, r, A\}, \{e', e, A\}, \{e', A\}, \{e', s, r, A\}\}.$$

Bedingung (ii) besagt, daß $e \notin \Xi_1^{(\nu)}$, womit nur noch

$$\{\{e', A\}, \{e', s, r, A\}\}$$

übrigbleibt. Bedingung (iii) bestimmt nun nichts, aber die Schnittkantendefinition 5.5 erzwingt für $\{e', s, r, A\}$, daß e in der K-Komponente vorkommt. Es ergibt sich:

$$\begin{aligned} S \;=\{ \quad & (\{e', A\}, \{e', A\}), (\{e', e, A\}, \{e', A\}), (\{e', e, s, A\}, \{e', A\}), \\ & (\{e', e, s, r, A\}, \{e', A\}), (\{e', s, r, A, e\}, \{e', s, r, A\})\}. \end{aligned}$$

Es ist z. B. $(\{e', e, r, A\}, \{e', A\}) \notin S$, weil dies auch keine gültige Schnittkante ist. Damit ist $\varphi_1 = ((\{e', A\} \cap \mathcal{K}) \cup \{e', e, A\}, \{e', A\}) = \varphi_1$, ebenfalls wie erwartet.

Das ist jetzt bei der Definition der Transitionen zu nutzen.

Definition 5.7 (Folgekonfiguration) Seien H ein HLSC und $(h, \varphi) \in \mathcal{K}_\mathsf{l}(H)$ (Definition 2.22). Dann ist (h', φ') eine *Pseudo-Folgekonfiguration von* (h, φ) *unter* $e \in \mathcal{E}$, falls eines gilt von (vgl. Definition 2.23):

(1) ▹ $h' = h \cdot l$ für ein $h_{|h|}l \in \mathfrak{L}(H, h_{|h|}, l_{|l|})$ und $|l| > 0$;
▹ $\varphi'_{|h'|} = (\{e\}, \{e\})$, und $\varphi'_g = \varphi_g$ für $g \leqslant |h|$, $\varphi'_g = (\emptyset, \emptyset)$ für $|h| < g < |h'|$;
▹ $\mathfrak{e}(\varphi') \in \mathfrak{K}_\mathsf{l}(\mathfrak{m}(h'))$.

(2) ▹ $h' = h$;
▹ Es gibt ein i mit $(\nu, \nu') = \varphi_i$, $e \in \mathcal{E}(M_{h_i})$, $e \notin \nu$, $\varphi'_j = \varphi_j$ für $j \neq i$ und $\varphi'_i = (\nu \cup \{e\}, \nu' \cup \{e\})$,
▹ $\mathfrak{e}(\varphi') \in \mathfrak{K}_\mathsf{l}(\mathfrak{m}(h))$.

(h', φ') ist eine *Pseudo-Folgekonfiguration von* (h, φ) *unter* ε, falls eines gilt von:

(3) ▹ $h' = h \cdot l$ für ein $h_{|h|}l \in \mathfrak{L}(H, h_{|h|}, l_{|l|})$ und $|l| > 0$;
▹ $\varphi'_{|h'|} = (\emptyset, \{e\})$ für ein $e \in \mathcal{K}(M_{|h'|})$, und $\varphi'_g = \varphi_g$ für $g \leqslant |h|$, $\varphi'_g = (\emptyset, \emptyset)$ für $|h| < g < |h'|$;
▹ $\mathfrak{e}(\varphi') \in \mathfrak{K}_\mathsf{l}(\mathfrak{m}(h'))$.

(4) ▹ $h' = h$;
▹ Es gibt ein i mit $(\nu, \nu') = \varphi_i$, $e \in \mathcal{K}(M_{h_i})$, $e \notin \nu$, $\varphi'_j = \varphi_j$ für $j \neq i$ und $\varphi'_i = (\nu \cup \{e\}, \nu')$,
▹ $\mathfrak{e}(\varphi') \in \mathfrak{K}_\mathsf{l}(\mathfrak{m}(h))$.

(h', φ') ist eine *Folgekonfiguration von* (h, φ) *unter* $e \in \mathcal{E}_\varepsilon$, falls für eine Pseudo-Folgekonfiguration $(h', \hat{\varphi})$ von (h', φ') unter e gilt: $(h', \varphi') = \mathfrak{r}(h', \hat{\varphi}) \in \mathcal{K}_\mathsf{l}(H)$.

Die Transitionen sind also wie üblich, außer daß Ereignisse auch durch ε übersprungen werden können. Sie werden dann dennoch in der Konfiguration gespeichert, da sie später nicht mehr auftreten können, aber nicht in der Ξ-Komponente, die ja für die ordnungsgemäße Durchführung der Wärmeleitung zuständig ist.

Wir müssen uns wieder Gedanken um κ und hinreichende Kriterien für die Endlichkeit der Automaten machen. Da wir $\rightsquigarrow$ nicht über LSCs hinaus wirken lassen, macht κ keine Probleme. Es werden alle MSCs herausgeschnitten, die nicht mehr gebraucht werden. Ein MSC wird nicht mehr gebraucht, falls alle heißen Ereignisse abgearbeitet sind, und alle kalten entweder abgearbeitet wurden oder nicht mehr abgearbeitet werden könnten.[1] Da wir aber unsere Hilfsfunktion $\mathfrak{u}$ zur Verfüngung haben, können wir uns einfach auf die K-Komponenten als Kriterium stützen.

Definition 5.8 (HLSC-Konfigurationsbereinigungsfunktion) Sei $H \in \mathfrak{H}_{\mathsf{l}}$. Für ein $(h, \varphi) \in \mathcal{K}_{\mathsf{l}}(H)$ ist $\kappa(h, \varphi) = (h', \varphi')$, wobei gilt:

(i) $h' = h_{i_1} h_{i_2} \dots h_{i_z} h_{|h|}$ mit $1 \leqslant i_1 < i_2 < \dots < i_z < |h|$, und für ein $i \in \{1, \dots, |h| - 1\}$ ist genau dann $i \notin \{i_1, \dots, i_z\}$, falls $\varphi_i = (\mathcal{E}(M_{h_i}), \_)$.

(ii) $\varphi' = \varphi_{i_1} \varphi_{i_2} \dots \varphi_{i_z} \varphi_{|h|}$.

Definition 5.9 (HLSC-Automaten) Für ein $H \in \mathfrak{H}_{\mathsf{l}}$ sind $\mathcal{A}(H)$ und $\mathcal{F}(H)$ so definiert wie bei üblichen HMSCs (Definitionen 2.24 und 2.26). Die Endzustandsmenge von $\mathcal{A}(H)$ ist dabei $\{(hj, \varphi) : j \in N_* \text{ und } \varphi_i = (\mathcal{E}(M_{h_i}), \_) \text{ für jedes } i\}$.

Lemma 5.2 Sei $M \in \mathfrak{M}_{\mathsf{l}}$. Dann ist genau dann $\Xi \in \mathfrak{w}(M)$, falls $(\mathcal{E}, \Xi) \in \mathfrak{K}_{\mathsf{l}}(M)$.

Beweis: Sei zunächst $\Xi \in \mathfrak{w}(M)$. Seien $K = \mathcal{E}$ und $\Xi' = \Xi$. Dann ist $\Xi \subseteq K \subseteq \mathcal{E}$ und $\Xi' \cap (K \backslash \Xi) = \Xi \cap (\mathcal{E} \backslash \Xi) = \emptyset$. $\Xi' \in \mathfrak{w}(M)$ nach Annahme. Da $\Xi = \mathcal{E}(M|_{\Xi})$, ist jede Bedingung von Definition 5.5 für $(\mathcal{E}, \Xi)$ erfüllt.

Sei nun $(\mathcal{E}, \Xi) \in \mathfrak{K}_{\mathsf{l}}(M)$. Definition 5.5 fordert die Existenz eines gewissen $\Xi' \supseteq \Xi$, sodaß die dort angegebenen Bedingungen erfüllt sind. Angenommen, $\Xi' \supset \Xi$. Sei also $e \in \Xi' \backslash \Xi$. Dann ist $e \in \Xi' \cap (\mathcal{E} \backslash \Xi)$, also $\Xi' \cap (K \backslash \Xi) \neq \emptyset$ für $K := \mathcal{E}$. Also kann $\Xi' \supset \Xi$ nicht gelten. Daher müssen die Bedingungen aus Definition 5.5 für $\Xi' := \Xi$ erfüllt sein, insbesondere $\Xi' \in \mathfrak{w}(M)$, was zu zeigen war. ⊛

Der Vorteil des Ansatzes über die Schnittkanten und κ zeigt sich jetzt.

Satz 5.1 Für ein $H \in \mathfrak{H}_{\mathsf{l}}$ gilt:

(a) $\mathcal{L}(\mathcal{A}(H)) = [\![H]\!]$.

(b) $\mathcal{L}(\mathcal{F}(H)) = \mathcal{L}(\mathcal{A}(H))$.

Beweis:

(a) Der Beweis zu Satz 2.5 drückt sich durch, nur an manchen Stellen wird die Argumentation technisch etwas aufwendiger. Sei also wieder $w \in [\![l]\!]$ für ein $l \in \mathfrak{L}_*(H)$. Zu zeigen ist, daß $c_0 \rightarrow^w (l, \varphi) \in K_*$.

[1]Ließe man letzteres Kriterium weg, so führte man halb abgearbeitete MSCs mit nicht gelesenen kalten Ereignissen immer auf ewig mit.

Wir beginnen damit, $c_0 \rightarrow^w (l, \varphi) \in K$ für beliebige $w \in [\![l]\!]$ für $l \in \mathfrak{L}(H, i_0, z)$ bei beliebigem z nachzuweisen, also nicht nur für Endläufe. Dies machen wir in erster Linie durch Induktion über $r := |l|$.

$r = 1$: Sei $M = M_{l_1}$. Dann ist $w \in [\![M]\!]$. Sei z minimal, sodaß es keinen nur Transitionen der zweiten Art verwendenden Pfad $c_0 \rightarrow^{w_1} c_1 \rightarrow^{w_2} \ldots \rightarrow^{w_{z-1}} c_{z-1} \rightarrow^{w_z} c_z$ gibt. Da nur Transitionen der zweiten Art verwendet werden, ist $c_{z-1} = (l_1, (K, \{w_1, \ldots, w_{z-1}\}))$. Nun kann, nach Annahme, w_z nicht gelesen werden. Angenommen, $w_z \in K$. Da keine ε-Übergänge genommen wurden, muß w_z durch $\mathfrak{r}$ in K eingefügt worden sein. Damit ist $w_z \in \mathcal{K}$ und es muß nach Bedingung (iii) der $\mathfrak{r}$-Definition $w_z \rightsquigarrow^* e' \prec e$ für $e' \notin \Xi$, $e \in \Xi$ gegeben haben nach einer vorherigen w_j-Transition. Damit kann e' auch nachher nicht mehr vorkommen, mithin $w_{z'} \neq e'$ für $z' > z$. Also ist $\{w_1, \ldots, w_z, \ldots, w_{|w|}\} \notin \mathfrak{w}(M)$, im Widerspruch zu $w \in [\![M]\!]$ (Definition 5.2). Damit ist $w_z \notin K$. Da w_z nicht gelesen werden kann, muß die Bedingung $(K, \{w_1, \ldots, w_z\}) \in \mathfrak{K}_{\mathfrak{l}}(M)$ nicht erfüllt sein. Das ist aber ein Widerspruch. Also kann man ganz w lesen.

Induktionsschritt: Sei $w \in [\![l_1 \ldots l_r l_{r+1}]\!]$. Sei w_i das erste Element des letzten MSCs (also $i = \min\{j : w_j \notin \mathcal{E}(\mathfrak{m}(l_1 \ldots l_r))\}$) (Falls ein solches i nicht existiert, läßt sich direkt die Induktionsvoraussetzung anwenden und wir sind fertig.). Induktiv können wir voraussetzen, daß $c_0 \rightarrow^{w_1 \cdots w_{i-1}} (l_1 \ldots l_s, \hat{\varphi})$ für ein geeignetes $\hat{\varphi}$ und ein $s \leqslant r$. Angenommen, von dort existiere keine w_i-Transition der ersten Art. Offenbar erfüllt aber $(l_1 \ldots l_s l_{s+1} \ldots l_r l_{r+1}, \varphi(\emptyset, emptyset)^{r-s}\{w_i\})$ die ersten beiden Bedingungen der ersten Pseudo-Transitionsart. Da für w_i die Lauf-Komponente erweitert wird, muß es an der $\mathfrak{K}_{\mathfrak{l}}$-Komponente scheitern. Der Widerspruch folgt hier wieder mit Lemma 5.2.

Für den Induktionsschritt ist noch zu argumentieren, daß es auch Transitionen für die $w_{i+1}, \ldots, w_{|w|}$ gibt. Wir starten also in einem bestimmten c und wollen $w_{i+1} \ldots w_{|w|}$ mittels Transitionen der zweiten Art lesen. Daß das geht, folgt genau wie in der gleichen Situation des Induktionsanfangs.

Kehren wir zu dem w zurück, mit dem wir starteten. Hier ist l ein Endlauf, also $l \in \mathfrak{L}_*(H)$. Damit ist $(l, \varphi) \in K_*$, falls alle φ_ν vollständig sind. Im Gegensatz zu der Situation im Beweis zu Satz 2.5 muß das hier tatsächlich nicht der Fall sein. Dafür haben wir aber die ε-Transitionen: Es gibt einen Pfad

$$(l, \varphi) \xrightarrow{\varepsilon}{}^* (ll', \hat{\varphi}\hat{\varphi}') \in K_*,$$

denn ansonsten gäbe es wieder den Widerspruch wie eben.

Gesamt ist $\mathfrak{L}(\mathcal{A}(H)) \supseteq [\![H]\!]$ gezeigt. Es bleibt noch, $\mathfrak{L}(\mathcal{A}(H)) \subseteq [\![H]\!]$ zu zeigen. Das geht völlig analog zu der entsprechenden Richtung im Beweis zu Satz 2.5 mit Lemma 5.2. Damit sind beide Inklusionsrichtungen gezeigt und die Behauptung ist bewiesen.

(b) Satz 2.6 hat ausschließlich Lemma 2.14 benutzt. Wir sind also fertig, wenn wir die Aussagen aus diesem Lemma übertragen können.

Seien also $A = \mathcal{A}(H)$ und $B = \mathcal{F}(H)$. Aussage (c) ist auch hier absolut klar. Die technischen Schwierigkeiten werden offenbar, wenn wir uns beispielhaft Aussage (a) anschauen. Zu zeigen ist also: Gilt $q_1 \rightarrow^e_B q_2$ und $q_1 = \kappa(\hat{q}_1)$, so gibt es ein $\hat{q}_2$ mit $q_2 = \kappa(\hat{q}_2)$ und $\hat{q}_1 \rightarrow^e_A \hat{q}_2$. Nun setzt sich die vorausgesetzte B-Transition zwar wieder aus einer A-Transition und einer κ-Anwendung zusammen, aber die A-Transition ihrerseits entsteht aus einer Anwendung von $\mathfrak{r}$ auf eine Pseudo-Transition. Wir haben also die Situation in Abbildung 5.4, wobei p-Pfeile Pseudo-Transitionen bedeuten. Man kann das machen. Das ist

Abbildung 5.4 Zum Beweis von Satz 5.1

$$\begin{tikzcd}
\hat{q}_1 \arrow[rr, dashed, "e", "p"'] \arrow[ddd, "\kappa"'] & & q' \arrow[rr, dashed, "ч"] & & \hat{q}_2 \arrow[ddd, dashed, "\kappa"] \\
& & & & \\
& q \arrow[rr, dotted, "ч"] & & \hat{q} \arrow[dr, dotted, "\kappa"'] & \\
q_1 \arrow[ur, dotted, "e", "p"'] \arrow[rrrr, "e", "B"'] & & & & q_2
\end{tikzcd}$$

aber ein erheblicher technischer Aufwand, der wenig zur Erhellung beiträgt, weswegen ich hier darauf verzichte, das im einzelnen auszuführen. ⊛

Wann ist ein $\mathcal{F}(H)$ nun endlich? Vielleicht haben wir ja Glück, und schwache Konvergenz ist wieder hinreichend. Betrachten wir also ein nur aus dem LSC aus Abbildung 5.1 bestehendes HMSC mit Selbstschleife: $\longrightarrow M_1$, wenn M_1 dieses LSC ist. Wie man sich leicht überlegen kann, ist dieses HLSC mit asynchroner Konkatenation tatsächlich konvergent. Leider ist die Semantik unbeschränkt: Prozeß b kann unbeschränkt viele Nachrichten senden, ohne daß a irgendeine empfängt. Was die Konfiguration angeht, so sieht man, daß $ч(1, (\{s_2\}, \{s_2\})) = (1, (\{s_1 r_1 s_2\}, \{s_2\})$. Das ist eine gültige Konfiguration, da es die gleichmäßig warme Menge $\{s_2, r_2\}$ gibt, die die Bedingungen aus Definition 5.5 erfüllt. Beides gilt analog für mehrmaliges Durchlaufen von M_1. Wir haben also in $\mathcal{A}(H)$ die Transitionen

$$\to (1, (\emptyset, \emptyset)) \xrightarrow{s_2} (1, (\{s_1 r_1 s_2\}, \{s_2\}) \xrightarrow{s_2} (11, (\{s_1 r_1 s_2\}, \{s_2\})(\{s_1 r_1 s_2\}, \{s_2\})) \xrightarrow{s_2} \ldots,$$

an denen κ nichts ändern kann. Leider ist also auch $\mathcal{F}(H)$ unendlich.

Die Unbeschränktheit kommt hier daher, daß zwar gesamt Konvergenz herrscht, diese aber nur durch kalte Ereignisse entstanden ist. Wir müssen also fordern, daß schon die heißen Ereignisse alleine zur Konvergenz führen, also der heiße Teil des erweiterten Sendungsgraphen zusammenhängend ist.

Aber auch das reicht nicht. Angenommen, wir haben ein konvergentes HLSC und packen in eine Schleife zwei weitere Prozesse, von denen einer dem anderen eine Nachricht sendet. Diese beiden Ereignisse seien kalt. An der Konvergenz, so wie sie gerade vorgeschlagen wurde, ändert das nichts, aber dennoch kann das kalte Sendungsereignis beliebig häufig auftreten, ohne daß es einmal empfangen wird. Auch hier haben wir also Unbeschränktheit. Um das auszuschließen, verlangen wir, daß jedes kalte Ereigniss Bedingung eines heißen ist, daß also insbesondere heiße Ereignisse existieren. Dann müßte es ein heißes Ereignis geben, welches auf das kalte Empfangsereignis wartet, und da die heißen Ereignisse zusammenhängend sind, muß irgendwann der Empfang erfolgen oder durch ε übersprungen werden (falls die Nachricht nicht gesendet wurde). Das wird jetzt formalisiert.

Definition 5.10 (HLSC-Konvergenz) Sei $H \in \mathfrak{H}_\mathsf{l}$. Sei $\wp_l$ definiert wie in Definition 2.15.c. Ist für jede einfache Schleife l der Teilgraph $\wp_l(l)|_{\mathcal{H}}$ zusammenhängend und nicht leer, und ist ferner für alle $e \in \mathcal{K}$ und $e' \in \mathcal{H}$ in $\wp_l(l)$ e' von e erreichbar, so heißt H l-*konvergent*.

Satz 5.2 Für ein l-konvergentes $H \in \mathfrak{H}_\mathsf{l}$ ist $\mathcal{A}(H)$ endlich.

Beweis: Für die heißen Ereignisse ändert sich an der Argumentation im Beweis zu Satz 2.8 nichts: Alle müssen irgendwann gelesen werden, wenn ihre MSCs durchlaufen werden. Gelesene Ereignisse werden nicht nur in Ξ, sondern auch in K gespeichert. Für die kalten Ereignisse ergibt sich aus der Erreichbarkeitsbedingung in Definition 5.10, daß sie irgendwann durch $\varkappa$ in die K-Komponenten hineingeschrieben werden müssen, da sie durch spätere, heiße Ereignisse unmöglich gemacht werden. Insgesamt haben wir wiederum nur solche LSC-Schnittkanten in φ beliebig oft, die durch κ entfernt werden, was zu zeigen war. ⊛

Übrigens sieht man leicht, daß alle eingeführten Strukturen isomorph zu denen aus Kapitel 2 sind, wenn $\mathcal{H} = \mathcal{E}$. Es handelt sich also tatsächlich um eine Verallgemeinerung, und nicht um etwas völlig Unterschiedliches.

Wir erhielten erhebliche Probleme, dehnten wir auf naive Weise die Wärmeleitungsrelation auf Ereignisse verschiedener LSCs aus. Nehmen wir etwa in einem HLSC-Fragment $\circlearrowleft M \longrightarrow L$ ein $e \in \mathcal{K}(M)$ und ein $f \in \mathcal{K}(L)$ und fordern, f solle genau dann stattfinden, falls e stattfand.

Zunächst haben wir dann genau das gleiche Problem wie mit Timern: Ist für die Erkennbarkeit von f maßgeblich, ob e

- ▹ beim *letzten* M-Besuch,
- ▹ bei *allen vorherigen* M-Besuchen oder
- ▹ bei *irgendeinem vorherigen* M-Besuch

stattfand? Gibt es zusätzlich noch einen Pfad von N nach M, treten diese Fragen in die Zukunft gerichtet auf: Reicht es, daß e irgendwann später auftritt, oder beim nächsten Mal, oder bei allen folgenden Besuchen von M?

Ähnlich wie bei den Timern, bei denen wir nur teilweise forderten, daß sie am Schluß eines Durchlaufes nicht mehr laufen sollten, ist auch hier zu fragen, ob f nur dann von e abhängen darf, falls M überhaupt durchlaufen wurde, oder aber eine Festlegung der Wärmeleitung gleichzeitig eine Festlegung der Läufe ist: Somit könnte ein M-Besuch einen L-Besuch erzwingen, wie ein Timer den nachfolgenden Besuch eines MSCs mit Timern mit Time-Out oder Timer-Löschung erzwingen kann.

5.b Kompositionalität

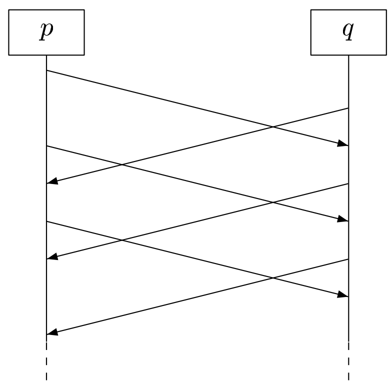
Wir wollen eines der Grundprinzipien der MSCs angreifen, nämlich die Forderung, ein solches müsse eine vollständige Kommunikation abbilden, also als einzelnes MSC implementierbar sein. Nun wollen wir auch zulassen (in gewissen Grenzen), daß ein MSC eine Abfolge von Kommunikationsereignissen spezifiziert, die alleine gesehen nicht sinnvoll ist. Das wurde vorher in [GMP01, MM01] untersucht. Die dort erreichten Ergebnisse werden an geeigneter Stelle angegeben.

In diesem Abschnitt wird keine Übersetzung in Automaten angegeben. Er dient nur dazu, zu demonstrieren, wie sich das Modell der (H)MSCs aus Kapitel 2 weiter an neu hinzukommende Forderungen an die Ausdrucksstärke der Spezifikationssprache anpassen läßt.

5.b.i Einführung

Beispiel 5.4 Um den Sinn dieser Verallgemeinerung einzusehen, betrachte man das klassische Beispiel aus [GMP01] und [HMKT99] in Abbildung 5.5. Hier sind

Abbildung 5.5 Ein nicht in MSCs abbildbares Kommunikations-Protokoll



zwei Prozesse in einer Kommunikation verstrickt, die man offensichtlich, sofern sie endlich ist, in einem MSC modellieren kann. Was aber, wenn man beliebig lange solcher Nachrichtenaustauschfolgen erlauben möchte? Es ist nicht möglich, dies durch eine Schleife in einem HMSC zu lösen, da wir die Kommunikation nicht in einzelne MSCs aufteilen können.[2] Nichtsdestotrotz läßt sich eine Finite State Machine angeben, die dieses Protokoll ausführt – Somit erscheint ein solches Protokoll durchaus als sinnvoll, und es wäre vorteilhaft, die Modellierungsfähigkeit von MSCs daraufhin auszudehnen. Das Problem ist es, daß innerhalb eines MSCs die Funktion m eine Bijektion $\mathcal{S} \to \mathcal{R}$ ist (Definition 2.1).

Definition 5.11 (kompositionelles MSC)

(a) Ein *kompositionelles Message Sequence Chart (kMSC)* M ist ein 7-Tupel

$$\langle \mathcal{P}, \mathcal{E}, o, \Lambda, m, \prec, \ell \rangle,$$

wobei $m : \mathcal{S} \dashrightarrow \mathcal{R}$ eine injektive *Nachrichtenfunktion* ist und alles andere wie in Definition 2.1 gilt. Wir setzen $\mathcal{S}\not\downarrow := \mathcal{S} \setminus \operatorname{dom}(m)$, $\mathcal{R}\not\downarrow := \mathcal{R} \setminus \operatorname{range}(m)$ und $\mathcal{E}\not\downarrow := \mathcal{S}\not\downarrow \cup \mathcal{R}\not\downarrow$.

(b) ([GMP01]) M heißt *linksgeschlossen*, falls m surjektiv ist, und *geschlossen*, falls m eine totale, bijektive Funktion ist.

(c) $\mathfrak{M}_\mathsf{k}(\mathcal{P}, \Lambda)$ ist analog zu $\mathfrak{M}(\mathcal{P}, \Lambda)$ definiert (Definition 2.1).

Ein linksgeschlossenes kMSC empfängt also keine nicht selbst gesendeten Nachrichten: $\mathcal{R}\not\downarrow = \emptyset$. In einem geschlossenen kMSC ist $\mathcal{R}\not\downarrow = \mathcal{S}\not\downarrow = \emptyset$.

Wir erklären *nicht* die Semantik von kMSCs, weil ein nicht geschlossenes kMSC nur eine Bedeutung als Teil eines größeren Modells wie eines HMSCs hat (kHMSCs

[2] Zu Abbildung 5.5 existiert keine Schnittkante $K \notin \{\emptyset, \mathcal{E}\}$ mit $\mathfrak{c}(K) = \emptyset$ (Definition 2.7).

werden in Definition 5.13 eingeführt.) und ein geschlossenes kMSC einfach ein MSC ist (vgl. jedoch die Diskussion um den Verknüpfungsparameter k_{k} auf Seite 112).

Man kann die Kompositionalität in standardmäßige MSCs einführen, indem man schwarze und weiße Löcher (Seite 18) erlaubt und diese miteinander verknüpft. Das ist aber eigentlich nur eine Frage der Visualisierung und daher für uns irrelevant (vgl. Seite 23).

Kompositionalität ist eine fundamentale Änderung der bisherigen Sichtweise eines MSCs. Somit müssen wir im Prinzip alle bisher gegebenen Definitionen neu gestalten. Leider werden damit schon einfache Sachen unentscheidbar: Erweitern wir die Definition der HMSCs auf den kompositionellen Fall (formal in Definition 5.13), so ist, wie in [GMP01] über das Post-Korrespondenz-Problem gezeigt wird, nicht entscheidbar, ob es einen Endlauf durch ein kHMSC gibt, bei dem eine bestimmte Nachricht empfangen wird (hier: Satz 5.3). Damit ist nicht zu erwarten, daß eine Übersetzung in endliche Automaten ohne große Einschränkungen gelingt.

Wir beginnen damit, die Konkatenation (vgl. Definition 2.8) auf kMSCs anzupassen. Angenommen, wir wollten die kMSCs M' und M'' miteinander in dieser Reihenfolge verknüpfen zu einem kMSC M. Die üblichen Parameter werden wir wie in Definition 2.8 setzen. Sei nun $e'' \in \mathcal{R}'' \lightning$. Falls es ein zu e'' passendes $e' \in \mathcal{S}' \lightning$ gibt, so kann man e' und e'' miteinander verknüpfen und also m in M um $m(e') = e''$ erweitern. Es ist daher zunächst festzulegen, *welche* Ereignisse zueinander passen: Da innerhalb eines kMSCs ein Ereignis als Sendeereignis erklärt werden kann, ohne den Prozeß zu spezifizieren, an den die Nachricht gerichtet ist, muß eine solche Festlegung bei der Konkatenation erfolgen. Der Standard erlaubt bei den Löchern, den eigentlichen Kommunikationspartner anzugeben, erzwingt das aber nicht.

Falls es allerdings noch ein zu e' passendes $\hat{e}'' \in \mathcal{R}'' \lightning$ gibt, so sollten wir weiterhin festlegen, mit welchem $\mathcal{R} \lightning$-Ereignis die Verknüpfung erfolgen soll, und ebenso, falls es ein weiteres passendes $\hat{e}' \in \mathcal{S}' \lightning$ gibt: Darf nur $m(e') = e''$ und $m(\hat{e}') = \hat{e}''$ gelten oder hat man auch die Möglichkeit, die Verbindung $m(e') = \hat{e}''$, $m(\hat{e}') = e''$ zu definieren, weil die Nachrichten in gewisser, durch die Modellierung bestimmter, Hinsicht als ähnlich erachtet werden?

Für die Konkatenation werden wir daher den weiteren Parameter k_{k} einführen, der genau diese Bestimmungen durchführt. In [GMP01] werden gewisse Annahmen über diese Festlegung getroffen. Zum einen werden Kommunikationsereignissen gewisse *types* zugeordnet, die besagen, welcher Prozess an welchen was sendet. Gematcht werden dann die Elemente mit gleichen Typen. Somit ist die Verbindbarkeit bereits in den einzelnen MSCs festgelegt. Bei passenden Typen wird immer das $\prec$-minimale Element zur Verbindung ausgewählt. Die folgende Formalisierung wird das Matching von Elementen verallgemeinern, aber die FIFO-Einschränkung beibehalten. Die Ergebnisse werden aber insofern verallgemeinert, als fehlgeleitete Nachrichten modelliert werden können. Ebenso haben wir durch unseren k-Operator die Möglichkeit, beliebige Ordnungen zwischen MSCs auszudrücken. Es ist daher nicht angebracht, von vorneherein anzunehmen, ein Match könne nicht in der anderen Richtung stattfinden, d. h. es könnte nicht ein $e'' \in \mathcal{S}'' \lightning$ mit einem $e' \in \mathcal{R}' \lightning$ zusammengebracht werden. Eine solche Nachricht, die von einem späteren MSC in ein früheres führt, wollen wir erlauben. Wenn man das nicht will, muß man $(e', e'') \in k$ setzen. [GMP01] erlaubt das nicht: Alle im Laufe durch ein kHMSC entstehenden MSCs müssen linksgeschlossen sein.

Während der Abfassung dieser Diplomarbeit wurde in [MM01] ein mächtigeres Modell von kHMSCs als in [GMP01] vorgestellt. Insbesondere wird die Verallgemeinerung, daß Nachrichten auch erst in späteren kMSCs gesendet werden durfen, behandelt (mit generellem FIFO-Matching). Die resultierenden Objekte werden *Extended Compositional Message Sequence Graphs (XMSGs)* genannt.

5.b.ii FIFO-Matching

Wie gesagt konzentrieren wir uns auf den Fall des FIFO-Matchings. Es ist sinnvoll, für die Definition wieder eine Hilfsfunktion zu benutzen: $\mathscr{u}_{\bowtie}(M, K)$ erhält als Argument ein kMSC M und binäre Ereignisrelation K, die angibt, welche Elemente miteinander assoziiert werden können. Zurückgegeben wird dann das dementsprechend entstehende kMSC. Dabei wird FIFO-Matching benutzt: Für jedes Sendeereignis[3] wird dasjenige $\prec$-minimale passende Empfangsereignis gewählt, welches noch nicht verbunden ist und gemäß K paßt, wobei die Reihenfolge der Betrachtung der Sendeereignisse ebenfalls $\prec$ folgt. Dabei wird eine Ordnung auf den Prozessen vorausgesetzt, um die ersten Sendeelemente herauszufinden, auch falls nicht alle total geordnet sind.[4] Diese Ordnung ist auf natürliche Weise durch die Definition von $\mathcal{P}$ als $\{1, \ldots, n\}$ für ein $n \in \mathbb{N}$ gegeben. Übrigens ist diese Ordnung irrelevant, falls *Eindeutigkeit* im Sinne von Definition 5.13.b gegeben ist. Es wird jedenfalls insbesondere die Existenz solcher $\prec$-minimaler Elemente vorausgesetzt, was im FIFO-Rahmen sinnvoll erscheint. Dabei müssen wir darauf achten, daß tatsächlich eine Halbordnung entsteht und das kMSC FIFO-Kanäle besitzt.

Beispiel 5.5 Im kMSC in Abbildung 5.6.a laufen einige Nachrichtenpfeile ins Leere. Das soll bedeuten, daß es sich bei den entsprechenden Ereignissen um $\mathcal{E}\lightning$-Ereignisse handelt. Die Pfeile drücken aus, wie die Verbindungsmöglichkeiten aussehen sollen. In den Beispielen wird das immer sofort klar werden.

Ein einfaches Verbinden der Nachrichten gemäß der suggerierten Verbindbarkeit würde zu einem Zykel führen. Die Funktion $\mathscr{u}_{\bowtie}$ würde eines der Sendeereignisse wegnehmen, ihm das passende Empfangsereignis zuordnen und das gleiche für die anderen beiden Ereignisse versuchen. Da dann keine Halbordnung entsteht, ist $\mathscr{u}_{\bowtie}$ hier nicht definiert.

Das kMSC in Abbildung 5.6.b ergibt zwar eine Halbordnung, aber kein FIFO-MSC.

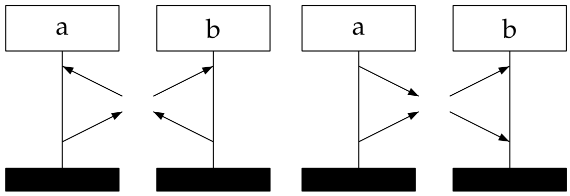
Definition 5.12 (Konkatenation) Seien M' und M'' kMSCs mit $\mathcal{P}' = \mathcal{P}''$. Sei weiter $k \subseteq \mathcal{E}' \times \mathcal{E}''$ und sei k_{K} eine beliebige Menge. . Dann ist die *Konkatenation* oder

[3] Zur Erinnerung: Im verwendeten Modell wird durch die Funktion λ bestimmt, welche Nachrichten übertragen werden, und λ ist ausschließlich auf $\mathcal{S}$ definiert (Definition 2.1).

[4] Formal:

$$
\begin{aligned}
\mathscr{u}_{\bowtie}(M, K) &:= \langle \mathcal{P}, \mathcal{E}, o, \Lambda, m', \prec', \ell \rangle \text{ falls es ein FIFO-MSC ist} \\
& \text{ wobei } (\prec', m') = \mathscr{u}'_{\bowtie}(\prec, m, K \cap (\mathcal{S}\lightning \times \mathcal{R}\lightning)), \\
\mathscr{u}'_{\bowtie}(\prec, m, \emptyset) &= (\prec, m), \\
\mathscr{u}'_{\bowtie}(\prec, m, K) &= \mathscr{u}'_{\bowtie}((\prec \cup \{(e, f)\})^{+}, m \cup \{(e, f)\}, \mathrm{kill}(K, e, f)) \\
& \text{ wobei } e = \min\{e' : (e', \_) \in K \land o(e') = \min\{o(e'') : (e'', \_) \in K\}\} \\
& \text{ und } f = \min\{f' : (e, f') \in K\}, \\
\mathrm{kill}(A, a, b) &= \{(c, d) \in A : a \neq c \land b \neq d\}.
\end{aligned}
$$

Abbildung 5.6 kMSCs ohne Möglichkeit der internen Verbindung



(a) (b)

sequentielle FIFO-Komposition von M' mit M'' unter k und k_{k}, $M' \circ_{\bowtie}^{(k,k_{\mathsf{k}})} M''$, definiert als

$$M' \circ_{\bowtie}^{(k,k_{\mathsf{k}})} M'' := u_{\bowtie}(u_{\bowtie}(M', k_{\mathsf{k}}) \circ^k M'', k_{\mathsf{k}}),$$

wobei $\circ^k$ auf kMSCs definiert ist analog zu $\circ^k$ auf MSCs (Definition 2.8). k_{k} wird als *Klebefunktion* bezeichnet. $\circ^{k,k_{\mathsf{k}}}$ ist linksassoziativ.

Informell gesprochen werden zwei kMSCs dadurch verbunden, daß zunächst im ersten gematcht wird, anschließend beide normal konkateniert werden und dann im entstehenden kMSC gematcht wird.

Im traditionellen Falle wurde in HMSCs der Parameter k durch die endliche Kantenrelation bestimmt. Ganz genauso können wir es hier aber nicht machen, da k als Definitionsbereich nur E^+ hatte. Der Grund ist eben, daß zu k das MSC-interne Äquivalent $\prec$ ist, wohingegen in Definition 5.11.a kein k_{k} entsprechender Parameter für einzelne MSCs vorgesehen ist. Außer m können aber auch noch Verbindungen von $\mathcal{E}\swarrow$-Ereignissen Nachrichten definieren, wie durch $u_{\bowtie}$ erklärt wurde.

Das kHMSC[5] in Abbildung 5.7 zeigt eine sinnvolle Anwendung einer Funktion mit erweitertem Definitionsbereich: Die Nachrichtenereignisse in kMSC M_3 können sowohl miteinander matchen als auch mit denen aus M_2 und M_4: Man erwartet, daß ein solches kHMSC sowohl die Linearisierung $s_{4|2}r_{4|2}s_{3|5}r_{3|5}$ als auch $s_{3|2}r_{3|2}$ erlaubt.[6]

Als weitere Begründung erinnere man sich, daß wir die Semantik eines einzelnen kMSCs nicht definiert haben, sondern über ein kHMSC $\longrightarrow M$ erklären müssen. Hier gibt es überhaupt kein Element in E, aber k_{k} muß trotzdem erklärt sein.

Wir werden k_{k} nur als Teilmenge über allen Ereignissen erklären. Das ist in diesem Fall möglich (vgl. Fußnote 7).

Definition 5.13 (kompositionelles HMSC mit FIFO-Matching)

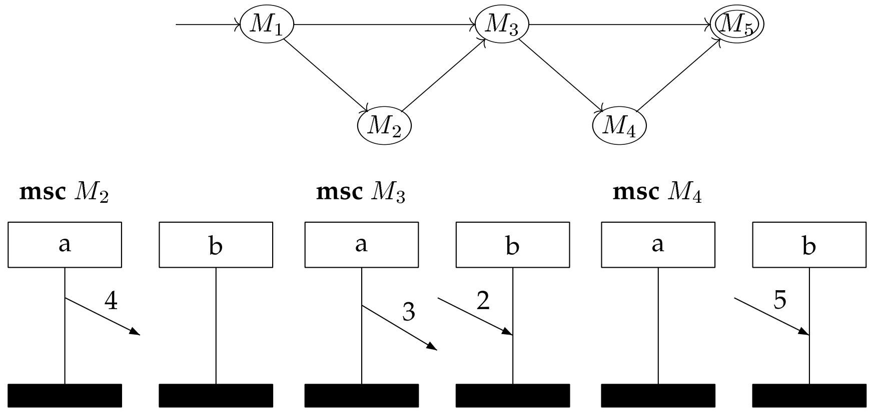
(a) Ein *kompositionelles High-Level Message Sequence Chart (kHMSC)* ist ein 9-Tupel

$$H := (N, \mathcal{P}, \Lambda, (M_i)_{i=1}^{|N|}, E, k, k_{\mathsf{k}}, i_0, N_*),$$

[5] M_1 und M_5 sind hier irrelevant und werden erst in Beispiel 5.6 definiert. Hier seien sie als leer anzunehmen.

[6] Die Notation wird also erweitert: $s_{3|2}$ heißt, daß es sich um das Sendeereignis von Nachricht 3 handelt, welche mit Nachricht 2 verbunden wird. Diese Zusatzinformation dient nur der Klarheit der Darstellung. Es handelt sich bei $r_{3|2}$ und $r_{4|2}$ in beiden Fällen um das gleiche Ereignis (nämlich r_2), aber sie sind dennoch insofern voneinander zu unterscheiden, als bei verschiedenen Beschriftungen von s_4 und s_2, etwa $\lambda' \neq \lambda''$, die Ereignisse die Typen $\mathsf{R}_{\mathsf{a,b}}(\lambda')$ bzw. $\mathsf{R}_{\mathsf{a,b}}(\lambda'')$ besitzen.

Abbildung 5.7 Zur Erklärung der Definition von k_{k} in kHMSCs



wobei alles Komponenten wie in Definition 2.10 sind, außer daß alle M_i kMSCs sind und $k_{\mathsf{k}} \subseteq \mathcal{S}\downarrow \times \mathcal{R}\downarrow$ ($\mathcal{S}\downarrow := \bigcup_{i=1}^{|N|} \mathcal{S}\downarrow (M_i)$ etc. wie üblich).

$\mathfrak{H}_{\mathsf{k}}(\mathcal{P}, \Lambda)$ entspricht $\mathfrak{H}(\mathcal{P}, \Lambda)$ für HMSCs (Definition 2.10).

(b) Ein kHMSC heißt *eindeutig*, falls aus $(e, f), (e, f') \in k_{\mathsf{k}}$ folgt $o(f) = o(f')$, und aus $(e, f), (e', f) \in k_{\mathsf{k}}$ folgt $o(e) = o(e')$; ansonsten heißt es *mehrdeutig*.

(c) Läufe etc. sind definiert wie bei HMSCs (Definition 2.12).

(d) Das *durch ein* $l := l_1 \ldots l_r \in \mathfrak{L}(H)$ *definierte kMSC mit FIFO-Matching* ist das kMSC $\mathfrak{m}_{\bowtie}(l) := M_{l_1} \circ_{\bowtie}^{\alpha(l_1,l_2),k_{\mathsf{k}}} M_{i_2} \circ_{\bowtie}^{\alpha(l_1 l_2,l_3),k_{\mathsf{k}}} \ldots \circ_{\bowtie}^{\alpha(l_1\ldots l_{r-1},l_r),k_{\mathsf{k}}} M_{l_r}$, wobei $\alpha(l_1 \ldots l_s, l_{s+1})$ definiert ist wie in Definition 2.13.[7]

(e) Wir setzen $[\![H;\pi]\!]_{\bowtie} := \{[\![l;\pi]\!] : l \in \mathfrak{L}_*(H) \wedge \mathfrak{m}_{\bowtie}(l)$ ist geschlossen$\}$, wobei π wie üblich ist und bei $\pi = \mathrm{id}$ weggelassen wird.

Aus jedem Lauf werden die resultierenden Objekte zur Bestimmung der Semantik ausgewählt, die MSCs ergeben, wohingegen in [MM01] gefordert wird, daß genau das für jeden Lauf der Fall ist. Der Unterschied ist aber nicht wesentlich. Ähnlich wie im Falle von HMSCs mit Timern, bei denen in einzelnen MSCs ja Timer ablaufen können, die nicht in ihm selbst in Gang gesetzt wurden (Beispiel 4.6), müssen wir hier aber die Konsistenz der resultierenden Worte verlangen (vgl. Definition 4.14), da man ansonsten nicht sinnvolle Linearisierungen erhält. Das wird durch die Forderung nach Geschlossenheit geleistet.

Eindeutigkeit ist eine sehr natürliche Forderung: Eine einmal abgesandte Nachricht kann nicht von verschiedenen Prozessen empfangen werden. Dennoch mag es sinnvoll sein, Mehrdeutigkeit zu erlauben, um eine gewisse Fehleranfälligkeit des Kommunikationsmediums zu erlauben.[8]

[7] Das ist, im Vergleich mit der sehr technischen Definition 2.13.a, entspannend. Das liegt daran, daß die Definition von $\mathfrak{u}_{\bowtie}$ nur wenige Parameter braucht. Das wiederum ist eine Folge der Tatsache, daß für Bindungen egal ist, welcher der beiden Bindungspartner aus einem früheren und welcher aus einem späteren MSC stammt. Beim normalen k jedoch treten häufig sowohl (e, e') als auch (e', e) auf, weswegen die Sammlung in einer großen Menge dann eine Auflösung bei der konkreten Konkatenation erfordern würde, die keine technischen Vorteile brächte.

[8] Das ist allerdings die einzige solchartige Stelle in der Modellierung.

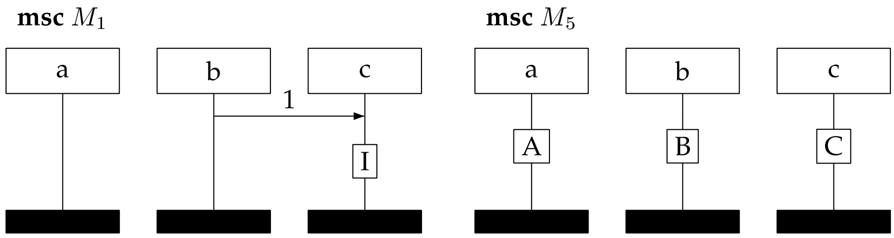
Beispiel 5.6 An einem Beispiel soll der Fortlauf der Konkatenation und des Matchings verdeutlicht werden. Gegeben sei das kHMSC aus Abbildung 5.7 erweitert um die Spezifikation in Abbildung 5.8. k ist definiert über gemäßigt asynchrone Konkatenation:

$$\begin{aligned} k(1,5) &= \{(I,C),(s_1,B)\}, \\ k(2,3) &= \{(s_4,s_2)\}, \\ k(2,5) &= \{(s_4,A)\}, \\ k(3,5) &= \{(s_3,A),(r_2,B)\}, \\ k(4,5) &= \{(r_5,B)\}. \end{aligned}$$

k_{k} ist so definiert, wie es die Pfeile suggerieren:

$$k_{\mathsf{k}} = \{(s_4,r_3),(s_4,r_5),(s_3,r_2),(s_3,r_5)\}.$$

Abbildung 5.8 Das kHMSC zu Beispiel 5.6



Der Lauf 135 resultiert in den kMSCs aus Abbildung 5.9 (k steht dabei für die α-Objekte aus Definition 5.13.). Der Lauf 1235 hingegen ergibt ein nicht geschlossenes kMSC (Abbildung 5.10), trägt also nichts zur Semantik bei, ebenso wie der Lauf 1345. Beide Läufe sind allerdings grundsätzlich als Präfixe eines weitergehenden Laufes möglich, nur halt nicht in diesem konkreten kHMSC.

Beispiel 5.7 Eine Motivation ergab sich ja durch Beispiel 5.4. Diese Kommunikation können wir tatsächlich in einem kompositionellen HMSC modellieren, wie Abbildung 5.11 zeigt. Dabei ist die Klebefunktion definiert, wie es durch die Pfeile suggeriert wird.

Der wesentliche Punkt in der Definition der Semantik ist die Definition von $\psi_{\bowtie}$. Es ist zu erwarten, daß andere Semantiken dadurch erhalten werden können, daß man eine Funktion ψ definiert, die einige passende Eigenschaften hat, und nicht sehr viele Änderungen and den höher liegenden Definitionen machen muß. Die Definition 5.12 wurde dementsprechend möglichst flexibel gestaltet: Die meiste Arbeit liegt auch bei der einfache Konkatenation bei der sicherlich weiter zu gebrauchenden $\circ^k$-Konkatenation und der anpassbaren ψ-Funktion.

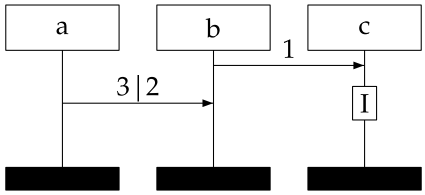
Wie versprochen, sind schon triviale Eigenschaften bei kHMSCs unentscheidbar. Der gegebene Beweis folgt einer Idee aus [GMP01], ist aber etwas einfacher, weil k_{k} passend definiert werden kann.

Satz 5.3 Die folgende Frage ist im allgemeinen unentscheidbar:

Abbildung 5.9 Der kurze Lauf durch das kHMSC aus Abbildung 5.8

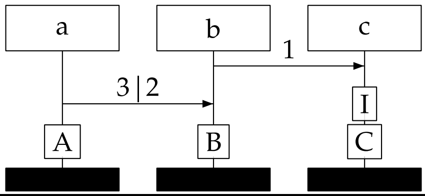
$u_{\bowtie}(M_1, k_{\mathsf{k}}) = \quad M_1,$

$\underbrace{u_{\bowtie}(u_{\bowtie}(M_1, k_{\mathsf{k}}) \circ^k M_3, k_{\mathsf{k}})}_{=:M'} =$



,

$u_{\bowtie}(M' \circ^k M_5, k_{\mathsf{k}}) =$



.

Gegeben: Ein kHMSC H und ein ungematchtes Sendeereignis e.
Frage: Gibt es ein $w \in [\![H]\!]_{\bowtie}$, sodaß e in w auftritt?

Beweis: (Reduktion vom Post-Korrespondenzproblem[9] [P46]) Angenommen, die genannte Frage sei entscheidbar. Sei $(\{w_1, \ldots, w_n\}, \{v_1, \ldots, v_n\})$ eine Instanz des Post-Korrespondenzproblemes über einem Alphabet Ω. Es ist bekannt, daß die Frage, ob zu diesen $w_\nu, v_\mu \in \Omega^*$ eine Folge $i_1, \ldots, i_g$ $(g \geqslant 0)$ mit $w_1 w_{i_1}, \ldots, w_{i_g} = v_1 v_{i_1}, \ldots, v_{i_g}$ existiert, unentscheidbar ist ([HU00]).

Sei $\mathcal{P} = \{r, s\}$. Nachrichten von s nach r kodieren die Worte. Sei dazu $i \in \{1, \ldots, n\}$ beliebig. Das MSC M_i umfasst $|w_i v_i|$ Ereignisse: $|w_i|$ Ereignisse auf s sind Sendungen, $|v_i|$ Ereignisse auf r sind Empfänge. Jede Prozesslinie ist total geordnet. Sei $(w_i)_\nu$ der Buchstabe $\beta \in \Omega$. Dann wird das ν-te Ereignis auf s in die Menge S_β aufgenommen. Entsprechendes gilt für die Ereignisse auf r und Mengen R_β. m ist total undefiniert. Andere Ereignisse existieren nicht.

Abbildung 5.12 zeigt die Kodierung von $w_i = abba$ und $v_i = bac$. $S_a = \{e_1, e_4\}$, $S_b = \{e_2, e_3\}$, $R_a = \{f_2\}$, $R_b = \{f_1\}$, $R_c = \{f_3\}$. $S_\alpha = R_\beta = \emptyset$ sonst.

Eine Lösung des PCPs soll einer Abfolge der M_i entsprechen. Dabei werden durch k_{k} genau die Ereignisse verbunden, die den gleichen Buchstaben kodieren. Die totale Ordnung auf den Prozessen sichert mit asynchroner Konkatenation und FIFO-Matching, daß gleiche Buchstaben auch an entsprechenden Positionen der Lösungswörter stehen: Könnte man eine a-Sendung und eine b-Sendung in beliebiger Reihenfolge empfangen, würde zu $(\{ab\}, \{ba\})$ eine Lösung gefunden. Formal haben wir also: $k'_{\mathsf{k}} = \{(e, f) : \exists \nu \in \Omega.(e \in S_\nu \wedge f \in R_\nu)\}$.

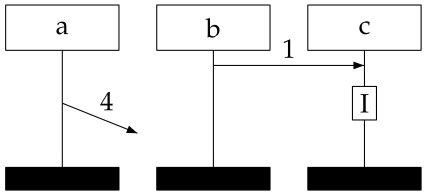
Die Nachricht, deren Empfang zu untersuchen ist, wird vor den M_i gesendet und nach ihnen empfangen. Dazu besitzen die MSCs M_0 und M_{n+1} je genau ein Ereignis: M_0 ein Sendeereignis e auf r und M'_0 ein Empfangsereignis e' auf s. Dann setzen wir $k_{\mathsf{k}} = \{(e, e')\} \cup k'_{\mathsf{k}}$, E und k sind definiert wie in Abbildung 5.13 angedeutet: $E = (\{0, 1, \ldots, n\} \times \{1, \ldots, n\}) \cup (\{1, \ldots, n\} \times \{1, \ldots, n, n+1\})$, k ist als $k_\otimes$ gegeben (Definition 2.11). ⊛

[9] Eine beliebte Beweis-Methode in der Theorie von HMSCs und ähnlichen Formalismen.

Abbildung 5.10 Ein unwichtiger Lauf durch das kHMSC aus Abbildung 5.8

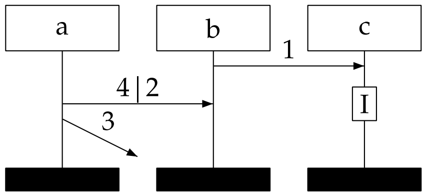
$$u_{\bowtie}(M_1, k_{\mathsf{k}}) = M_1,$$

$$\underbrace{u_{\bowtie}(u_{\bowtie}(M_1, k_{\mathsf{k}}) \circ^k M_2, k_{\mathsf{k}})}_{=:M'} =' $$



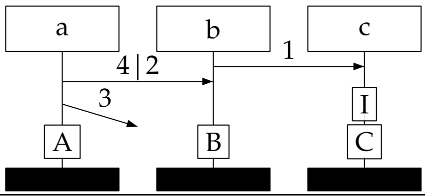
,

$$\underbrace{u_{\bowtie}(M' \circ^k M_3, k_{\mathsf{k}})}_{=:M''} =' $$



,

$$u_{\bowtie}(M'' \circ^k M_5, k_{\mathsf{k}}) = $$



.

Aus dem Beweis folgt natürlich sofort, daß auch die Frage, ob die Semantik eines kHMSCs leer ist, unentscheidbar ist.

Abbildung 5.11 Ein kompositionelles HMSC (Beispiel 5.7)

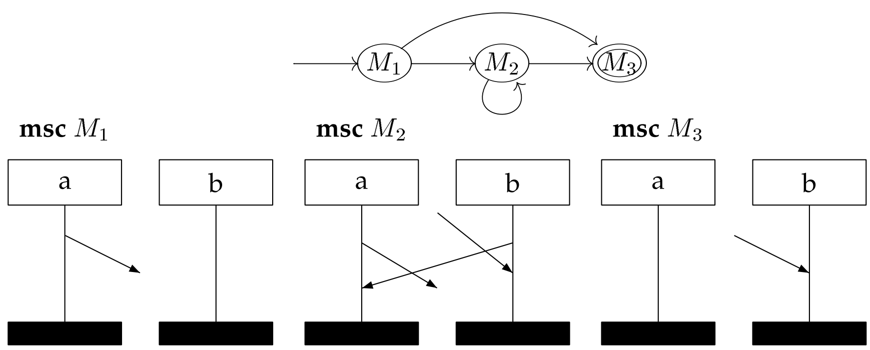


Abbildung 5.12 Kodierung eines Wortpaares des PCPs (Beweis zu Satz 5.3)

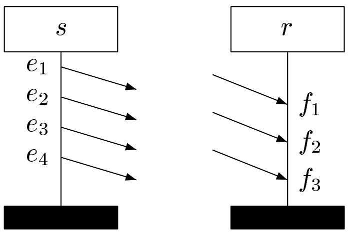
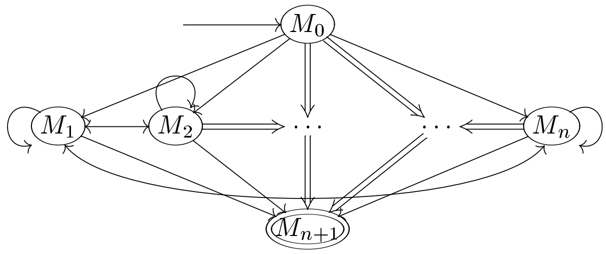


Abbildung 5.13 Das kHMSC zur Kodierung des PCPs (Beweis zu Satz 5.3)



Kapitel 6

Zusammenfassung

For the present we have journeyed far enough together,
and it is time to part. Yet before we do so,
we may well ask ourselves whether there is not some more general conclusion,
some lesson, if possible, of hope and encouragement,
to be drawn from the melancholy record of human error and folly
which has engaged our attention in this book.
JAMES GEORGE FRAZER, *The Golden Bough*, 1922

In dieser Arbeit wurden folgende relevante Fortschritte erzielt:

- Es wurden MSCs mit expliziten Ordnungen und Koregionen untersucht, wobei keine FIFO-Kanäle vorausgesetzt wurden. Die Konkatenation von MSCs wurde verallgemeinert, um auf spezifische Bedürfnisse einer Modellierung einzugehen.

- Es wurde eine Klasse von HMSCs identifiziert, die in endliche Automaten übersetzbar ist. Es läßt sich statisch entscheiden, ob ein HMSC zu dieser Klasse der *schwach konvergenten* HMSCs gehört. Gewisse andere Eigenschaften wurden benannt, die der schwachen Konvergenz ähnlich sind, und es wurden Beziehungen zwischen diesen Eigenschaften untersucht.

- Die Syntax und Semantik von (H)MSCs wurde um Zeitbeschränkungen und Timer erweitert. Das geschah, soweit möglich, im Einklang mit dem Standard.

- Die Übersetzung schwach konvergenter HMSCs in Automaten wurde so verändert, daß auch schwach konvergente HMSCs mit den neuen syntaktischen Elementen in Uhrenautomaten übersetzbar sind.

- Mit nur geringen Modifikation konnten die Ergebnisse auf Aktionsdauern erweitert werden. Das gibt Grund zur Hoffnung, daß das vorgestellte Modell robust genug ist, um auch für andere Erweiterungen von (H)MSCs als Grundlage zu dienen.

- Ebenso wurden Live Sequence Charts definiert. Auch hier konnte die Übersetzung in erkennende Automaten im wesentlichen parallel zu der normaler HMSCs durchgefürt werden, was wieder ein Hinweis auf gute Erweiterbarkeit des Modells ist.

▷ Abschließend wurden noch die nötigen Erweiterungen der Formalisierung durchgeführt, um kompositionelle MSCs in dieses Rahmenwerk einpassen zu können.

Welche zukünftige Arbeit kann nun darauf aufbauen?

▷ Als nächster logischer Schritt bietet sich natürlich an, das vorgestellte Verfahren in einem Model-Checker zu implementieren. Dabei lassen sich sicherlich sowohl bei der Konstruktion der Automaten als auch bei ihrer internen Repräsentation einige Möglichkeiten zur Effizienzsteigerung finden.

▷ Als weiterer Ansatzpunkt bieten sich die im Standard definierten, aber bisher nicht offiziell formal semantisch eingekreisten *Daten* an. Dem Ansatz von [JP01] folgend, könnte eine Behandlung von Daten in unserem Modell möglich und vielversprechend sein.

▷ Wie Lemma 3.1 zeigt, lassen sich Uhrenautomaten mit Diagonalbedingungen prinzipiell ebenso analysieren wie solche mit einfachen Bedingungen, wie sie in Kapitel 4 entstanden sind. Eine Erweiterung der Zeitbeschränkungen auf Uhrvergleiche in MSCs läßt sich also ebenfalls einfach mit Automaten untersuchen.

▷ Logische Charakterisierungen der Erweiterungen von MSCs fehlen bisher im wesentlichen. Es wäre nützlich, sie auch für den Fall (z. B.) kompositioneller timed LSCs zur Verfügung zu haben.

▷ Untersuchungen auf *non-local choice* ([BAL97b]), Pattern Matching ([GM02]) oder Inferenz ([AEY00]) sollten soweit möglich auf die hier vorgestellten Erweiterungen übertragen werden.

▷ Die HMSC-endlich-Automaten erkennen nur beschränkte Sprachen, wobei Beschränktheit nur einzelne Nachrichten betrachtet. Man könnte auch eingeschränktere Kommunikationskanäle im Sinne von [EMR97a] modellieren (Beschränktheit der Eingangs- oder Ausgangspuffer der Prozesse, globale Beschränktheit), indem man, ausgehend vom hier konstruierten Automaten, durch Betrachtung der Schnittkanten nicht gewünschte Zustände entfernt.

▷ Vielfach werden in der Literatur unendliche Läufe durch HMSCs untersucht. Dabei wird eine Büchi-Semantik unterlegt. Da auch für Uhrenautomaten Büchi-Erkennung untersucht wurde, könnte man die hier erlangten Ergebnisse auf ihre Übertragbarkeit hin untersuchen.

▷ Kompositionelle HMSCs sind nur in Grundzügen untersucht worden. Hier sollte man tiefere Untersuchungen durchführen. Insbesondere wurde die Vermutung aufgestellt, daß auch hier kleine Änderungen an den tiefliegenden Hilfsfunktionen gewünschte Ergebnisse auf der dem Modellierer zugänglichen Ebene hervorrufen können. Das sollte überprüft werden. Weiter kann man versuchen, eine Übersetzung in Automaten vorzunehmen.

Es bleibt also noch viel zu tun.

Symbolverzeichnis

Die Drucksachen, die uns stündlich begegnen,
sind nur selten in angenehmen Lettern und übersichtlich gedruckt.
Häufig sind sie lärmende Orgien wildgewordener Buchstaben
und selbstbewußt auftretender Wurstigkeit.
JAN TSCHICHOLD, *Erfreuliche Drucksachen durch gute Typographie*, 1960

In der folgenden Tabelle sollen sowohl echte Symbole aufgelistet werden wie $\mathfrak{M}$ als auch die üblichen Wahlen von Bezeichnern für Variablen und Strukturkomponenten wie ψ.

Die Sortierung erfolgt nach dem Schriftsatz.

Symbol	Bedeutung	Erklärung
γ	Uhrenupdate	Def. 3.5
ϑ	Zuordnungsfunktion $\mathcal{I} \to \mathbb{T}$	Def. 4.10
κ	Konfigurationsbereinigungsfunktion	Def. 2.25, 5.8
λ	Beschriftungsfunktion	Def. 2.1
π	Ereignistypfunktion	Def. 2.3
σ	kanonische Ereignistypfunktion	Def. 2.3, Seite 87
τ	Zeitfunktion	Def. 4.1, 4.5, 4.10, 4.12
φ	Schnittkantenfolge einer Konfiguration	Def. 2.22, 5.6
χ	Zuordnungsfunktion $\mathcal{I} \cup \mathcal{O} \cup \mathcal{C} \to \mathcal{T}$	Def. 4.10
ψ	Uhrbedingung	Def. 3.3
Λ	Beschriftungsalphabet	Def. 2.1
Ξ	echte Ereignisse einer LSC-Schnittkante	Def. 5.5
Π	Ereignistypalphabet	Def. 2.3
Σ	kanonisches Ereignistypalphabet	Def. 2.3, Seite 87
$\mathcal{A}$	Aktionsereignisse	Def. 2.1, 2.10
$\mathcal{C}$	Timer-Löschungen	Def. 4.10
$\mathcal{E}$	Ereignismenge	Def. 2.1, 2.10
$\mathcal{H}$	heiße Ereignisse	Def. 5.1
$\mathcal{I}$	Timer-Initialisierungen	Def. 4.10
$\mathcal{K}$	kalte Ereignisse	Def. 5.1
$\mathcal{O}$	Time-Outs	Def. 4.10
$\mathcal{P}$	Prozeßmenge	Def. 2.1, 2.10
$\mathcal{R}$	Empfangsereignisse	Def. 2.1, 2.10
$\mathcal{S}$	Sendeereignisse	Def. 2.1, 2.10
$\mathcal{T}$	Timer-Menge	Def. 4.10
$\mathcal{T}_!$	Endtimer-Menge	Def. 4.10
$\mathrm{dom}(R)$	Quellbereich einer Relation	Vorbem.
id_M	Identitätsfunktion auf M	Vorbem.
$\mathrm{range}(R)$	Bildbereich einer Relation	Vorbem.
k	HMSC-Ordnungsrelation	Def. 2.10
k_k	Klebefunktion	Def. 5.12
m	Nachrichtenfunktion	Def. 2.1
o	Zuordnungsfunktion $\mathcal{E} \to \mathcal{P}$	Def. 2.1
E	HMSC-Kantenrelation	Def. 2.10
		Fortsetzung. . .

Symbol	Bedeutung	Erklärung
N	HMSC-Knotennummern	Def. 2.10
N_*	HMSC-Endknotennummern	Def. 2.10
$L^\frown$	Schnittkante zu einer Schnittpunktmenge	Def. 2.7
$K^\vee$	Schnittpunktmenge zu einer Schnittkante	Def. 2.7
$\prec$	Präzedenzordnung	Def. 2.1
ℓ	Nachrichtenbeschriftung	Def. 2.1
$\models$	Modell-Relation für Uhrstände/-bedingungen	Def. 3.4
$[\![\gamma]\!]$	Semantik eines Updates	Def. 3.6
$[\![M;\pi]\!]$	Semantik eines MSCs etc. oder Laufes	Def. 2.4, 2.13, 5.2
$[\![H;\pi]\!]_>$	Semantik eines HMSCs mit Zeitverhalten	Def. 4.6, 4.14
$[\![H;\pi]\!]_1$	Semantik eines HMSCs mit Zeitverhalten	Def. 4.6, 4.14
$[\![H;\pi]\!]_\star$	Semantik eines HMSCs mit Zeitverhalten	Def. 4.6, 4.14
$[\![H;\pi]\!]_0$	Semantik eines HMSCs mit Zeitverhalten	Seite 87
$[\![H;\pi]\!]_\bowtie$	Semantik eines kHMSCs	Def.5.13
$\approx_{(c_x)}$	Uhrstandsäquivalenz bzgl. Grenzzahlen (c_x)	Def. 3.10
$\equiv$	Sprachäquivalenz zweier Automaten	Vorbem.
$\uplus$	disjunkte Vereinigung	Vorbem.
$\circ^k$	MSC- oder LSC-Konkatenation unter k	Def. 2.8, 5.3
$\circ_\bowtie^{k,k_\mathsf{k}}$	kMSC-Konkatenation unter k und k_k	Def. 5.12
$\oplus$	Synchrone Konkatenation	Def. 2.9, 2.11
$\otimes$	Asynchrone Konkatenation	Def. 2.9, 2.11
$\circledvee$	Gemäßigt asynchrone Konkatenation	Def. 2.9, 2.11
$\circledcirc$	Gleichgültige Konkatenation	Def. 2.9, 2.11
$\lfloor x \rfloor$	Ganzzahliger Anteil eines $x \in \mathbb{P}$	Vorbem.
$\mathcal{X}\swarrow$	Nicht-gematchte Ereignisse eines kHMSCs	Def. 5.11
$\rightsquigarrow$	Wärmeleitungsfunktion eines LSCs	Def. 5.1
$[M \to N]$	Raum aller totalen Funktionen von M nach N	Vorbem.
$[M \dashrightarrow N]$	Raum aller partiellen Funktionen von M nach N	Vorbem.
$\mathfrak{a}(a)$	Unterer Abschluß von a unter einer Halbordnung	Vorbem.
$\mathfrak{c}(K)$	Geschnittene Nachrichten einer Schnittkante	Def. 2.7
$\mathfrak{d}(v_1, v_2)$	Länge des kürzesten Pfades $v_1 \to v_2$	Vorbem.
$\mathfrak{e}(\varphi)$	Gesamtschnitt einer Konfiguration	Def. 2.22, 5.6
$\mathfrak{f}(x)$	Bruchteil eines $x \in \mathbb{P}$	Vorbem.
$\mathfrak{k}(x, y)$	k-Distanz im (erweiterten) Sendungsgraphen	Def. 2.15
$\mathfrak{l}$	Linearisierungsfunktion	Vorbem.
$\mathfrak{m}(x)$	Durch Lauf definiertes (k)MSC oder LSC	Def. 2.13, 5.4
$\mathfrak{m}_\bowtie(x)$	Durch Lauf definiertes FIFO-kMSC	Def. 5.13
$\mathfrak{o}$	Landau-Symbol *klein-Oh*	Vorbem.
$\mathfrak{p}(M)$	Potenzmenge einer Menge	Vorbem.
$\mathfrak{u}(M)$	Ungezeitetes MSC oder HMSC	Def. 4.1, 4.5, 4.12
$\mathfrak{w}(L)$	Gleichmäßig warme Ereignismengen eines LSCs	Def. 5.2
$\mathfrak{A}(M)$	in M aktive Prozesse	Def. 2.2
$\mathfrak{C}(M, i)$	Schnittpunktemenge	Def. 2.6
$\mathfrak{H}(\mathcal{P}, \Lambda)$	Menge aller HMSCs	Def. 2.10
$\mathfrak{H}_\mathsf{k}(\mathcal{P}, \Lambda)$	Menge aller kHMSCs	Def. 5.13
$\mathfrak{H}_\mathsf{l}(\mathcal{P}, \Lambda)$	Menge aller HLSCs	Def. 5.4
$\mathfrak{H}_\mathsf{t}(\mathcal{P}, \Lambda)$	Menge aller HMSCs mit Timing	Def. 4.5
$\mathfrak{H}_\mathsf{tt}(\mathcal{P}, \Lambda, \mathcal{T})$	Menge aller HMSCs mit Timern	Def. 4.12
$\mathfrak{K}(M)$	MSC-Schnittkantenmenge	Def. 2.7
$\mathfrak{K}_\mathsf{l}(M)$	LSC-Schnittkantenmenge	Def. 5.5
$\mathfrak{L}(H)$	Läufe durch ein HMSC	Def. 2.12
$\mathfrak{L}(H, i, j)$	Läufe durch ein HMSC von i nach j	Def. 2.12
$\mathfrak{L}_*(H)$	Endläufe durch ein HMSC	Def. 2.12
$\mathfrak{M}(\mathcal{P}, \Lambda)$	Menge aller MSCs	Def. 2.1
$\mathfrak{M}_\mathsf{k}(\mathcal{P}, \Lambda)$	Menge aller kMSCs	Def. 5.11
$\mathfrak{M}_\mathsf{l}(\mathcal{P}, \Lambda)$	Menge aller LSCs	Def. 5.1
$\mathfrak{M}_\mathsf{t}(\mathcal{P}, \Lambda)$	Menge aller MSCs mit Timing	Def. 4.1
$\mathfrak{M}_\mathsf{tt}(\mathcal{P}, \Lambda, \mathcal{T})$	Menge aller MSCs mit Timern	Def. 4.10
$\mathfrak{O}$	Landau-Symbol *gross-Oh*	Vorbem.
$\mathscr{B}(X, C, O)$	syntaktische Uhrbedingungen	Def. 3.3
$\mathscr{M}(\psi)$	Modellmenge einer Uhrbedingung	Def. 3.4
$\mathscr{W}(X)$	Menge von Uhrständen für Uhren X ($\mathbb{P}^X$)	Def. 3.2
$\mathscr{G}(M)$	Kommunikationsgraph eines MSCs	Def. 2.15

Fortsetzung…

Symbol	Bedeutung	Erklärung
$\mathscr{T}(l)$	Sendungsgraph eines HMSC-Laufes	Def. 2.15
$\mathscr{S}(l)$	Erweiterter Sendungsgraph eines HMSC-Laufes	Def. 2.15
$\mathcal{A}(M)$	Automat eines MSCs, HMSCs oder HLSCs	Def. 2.20, 2.24, 5.9
$\mathcal{A}(M, M', k)$	Kompositions-Automat zweier MSCs	Def. 2.21
$\mathcal{F}(H)$	Endlicher Automat eines HMSCs oder HLSCs	Def. 2.26, 5.9
$\mathcal{F}_{>}(H)$	Automat eines HMSCs mit Timern oder Timing	Def. 4.8, 4.15
$\mathcal{F}_{\star}(H)$	Automat eines HMSCs mit Timern oder Timing	Def. 4.9, 4.16
$\mathcal{F}_{1}(H)$	Automat eines HMSCs mit Timern oder Timing	Def. 4.18
$\mathcal{K}(H)$	Konfigurationsmenge eines HMSCs	Def. 2.22
$\mathcal{K}_{\mathsf{l}}(H)$	Konfigurationsmenge eines HLSCs	Def. 5.6
$\mathcal{L}(A)$	Sprache eines Automaten A	Vorbem., Def. 3.9
$\mathbb{D}$	Zeitsequenzen	Def. 3.1
$\mathbb{I}$	nicht-leere Intervalle über $\mathbb{T}$	Vorbem.
$\mathbb{P}$	$\{x \in \mathbb{R} : x \geqslant 0\}$	Vorbem.
$\mathbb{T}$	$\{x \in \mathbb{Q} : x \geqslant 0\}$	Vorbem.
$\mathbb{W}(\Sigma)$	Intervallwörter über $\mathbb{D}$ und Σ	Def. 3.1
$\mathsf{A}_p(\lambda)$	Aktionstyp	Def. 2.3
C_T	Timer-Löschungs-Typ	Seite 87
$\mathsf{I}_T(\xi)$	Timersetzungstyp	Seite 87
O_T	Time-Out-Typ	Seite 87
$\mathsf{R}_{p,q}(\lambda)$	Empfangstyp	Def. 2.3
$\mathsf{S}_{p,q}(\lambda)$	Sendetyp	Def. 2.3
$\text{ж}(\_)$	Erhitzungsfunktion	Def. 5.1, 5.2
$\text{ч}(h, \varphi)$	LSC-Konfigurations-Vervollständigungsfunktion	Seite 103
$\text{ц}_{\bowtie}(M, K)$	FIFO-kMSC aus M mit K-Verbindungen	Seite 111
$\text{ю}(w, i, X, Y)$	Index des ersten $x \in X$ vor $y \in Y$ ab i in w	Seite 78
$\text{я}(w, e, X)$	Matchmenge aller (e, x)-Paare aus w	Seite 78

Index

Literaturverzeichnis

[A97] Mark A. Ardis. Formal Methods of Telecommunication System Requirements: A Survey of Standardized Languages. *Annals of Software Engineering 3*, 3:157–187, 1997.

[A99] Rajeev Alur. Timed Automata. In *Proceedings of the 11th International Conference on Computer-Aided Verification (CAV'99)*, Band 1633 von *LNCS*, Seiten 8–22. Springer-Verlag, 1999.

[ACM97] Eugene Asarin, Paul Caspi, Oded Maler. A Kleene Theorem for Timed Automata. In *Proceedings of the 12th IEEE Symposium on Logic in Computer Science (LICS'97)*, 1997.

[AD94] Rajeev Alur, David L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994. (frühere Versionen: 17th ICALP, LNCS 443, 1990; Real Time: Theory in Practice, LNCS 600, 1991).

[AEY00] Rajeev Alur, Kousha Etessami, Mihalis Yannakakis. Inference of Message Sequence Charts. In *22nd International Conference on Software Engineering (CSE'2000)*, Seiten 304–313, Juni 2000.

[AEY01] Rajeev Alur, Kousha Etessami, Mihalis Yannakakis. Relizability and Verification of MSC Graphs. In *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming (ICALP'01)*, Band 2076 von *LNCS*, Seiten 797–808. Springer-Verlag, Juli 2001.

[AFH98] Rajeev Alur, Limor Fix, Thomas A. Henzinger. Event-Clock Automata: A Determinizable Class of Timed Automata. *Theoretical Computer Science*, 43(1):116–146, 1998.

[AHP96] Rajeev Alur, Gerard J. Holzmann, Doron A. Peled. An Analyzer for Message Sequence Charts. In *Proceedings of the 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, Band 1055 von *LNCS*, Seiten 35–48. Springer-Verlag, März 1996.

[AY99] Rajeev Alur, Mihalis Yannakakis. Model Checking of Message Sequence Charts. In *Proceedings of the 10th International Conference on Concurrency Theory (CONCUR'99)*, Band 1664 von *LNCS*, Seiten 114–129. Springer-Verlag, 1999.

[B94] Graham Brightwell. Linear Extensions of Random Orders. *Discrete Mathematics*, 125:87–96, 1994.

[B95] Béatrice Bérard. Untiming Timed Languages. *Information Processing Letters*, 55:129–135, 1995.

[B98] Danièle Beauqier. Pumping Lemmas For Timed Automata. In *Proceedings of the 1st International Conference on Foundations of Software Science and Computation Structures (FOSSACS'98)*, Band 1378 von *LNCS*, Seiten 81–94. Springer-Verlag, 1998.

[B00] Patricia Bouyer. A New Algorithm to Decide Emptiness of Updatable Timed Automata. Research Report LSV-00-9, ENS de Cachan, Oktober 2000.

[BAL96] Hanêne Ben-Abdallah, Stefan Leue. Syntactic Analysis of Message Sequence Chart Specifications. Technical Report 96-12, Department of Electrical and Computer Engineering, University of Waterloo, November 1996. *Langversion von [BAL97b].*

[BAL97a] Hanêne Ben-Abdallah, Stefan Leue. Expressing and Analysing Timing Constraints in Message Sequence Chart Specifications. Technical Report 97-04, Department of Electrical and Computer Engineering, University of Waterloo, 1997. *Langversion von [BAL97c].*

[BAL97b] Hanêne Ben-Abdallah, Stefan Leue. Syntactic Detection of Process Divergence and Non-Local Choice in Message Sequence Charts. In *Proceedings of the 3rd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'97)*, Band 1217 von *LNCS*, Seiten 259–274. Springer-Verlag, 1997. *Kurzversion von [BAL96].*

[BAL97c] Hanêne Ben-Abdallah, Stefan Leue. Timing Constraints in Message Sequence Chart Specifications. In *Proceedings of the 10th International Conference on Formal Description Techniques (FORTE/PSTV'97)*, November 1997. *Kurzversion von [BAL97a].*

[BDFP00a] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, Antoine Petit. Are Timed Automata Updatable? Research Report LSV-00-3, ENS de Cachan, Februar 2000. *Langversion von [BDFP00b].*

[BDFP00b] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, Antoine Petit. Are Timed Automata Updatable? In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV'2000)*, Band 1855 von *LNCS*, Seiten 464–479. Springer-Verlag, Juli 2000. *Kurzversion von [BDFP00a].*

[BDFP00c] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, Antoine Petit. Expressiveness of Updatable Timed Automata. In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS'2000)*, Band 1893 von *LNCS*, Seiten 232–242. Springer-Verlag, August 2000.

[BDGP98] Béatrice Bérard, Volker Diekert, Paul Gastin, Antoine Petit. Characterization of the Expressive Power of Silent Transitions in Timed Automata. *Fundamenta Informaticae*, 36(2):145–182, 1998.

[BL01] Benedikt Bollig, Martin Leucker. Modelling, Specifying and Verifying Message Passing Systems. In *Proceedings of the 8th International Symposium on Temporal Representation and Reasoning (TIME'01)*, Seiten 240–247, 2001.

[BLN02a] Benedikt Bollig, Martin Leucker, Thomas Noll. Generalised Regular MSC Languages. Technical Report 2002-03, RWTH Aachen, Januar 2002. *Langversion von [BLN02b]*, Erscheint.

[BLN02b] Benedikt Bollig, Martin Leucker, Thomas Noll. Generalised Regular MSC Languages. In *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'02)*, LNCS. Springer-Verlag, April 2002. *Kurzversion von [BLN02a]*, Erscheint.

[BP99a] Patricia Bouyer, Antoine Petit. Decomposition and Composition of Timed Automata. Research Report LSV-99-1, ENS de Cachan, Februar 1999. *Langversion von [BP99b].*

[BP99b] Patricia Bouyer, Antoine Petit. Decomposition and Composition of Timed Automata. In *Proceedings of the 26th International Colloquium on Automata, Languages, and Programming (ICALP'99)*, Band 1644 von *LNCS*, Seiten 210–219. Springer-Verlag, Juli 1999. *Kurzversion von [BP99a].*

[BP01] Patricia Bouyer, Antoine Petit. A Kleene/Büchi-like Theorem for Clock Languages. Research Report LSV-01-2, ENS de Cachan, Februar 2001.

[BPS96] Graham Brightwell, Hans Jürgen Prömel, Angelika Steger. The Average Number of Linear Extensions of a Partial Order. *Journal of Combinatorial Theory (A)*, 73:193–206, 1996.

[BPT01] Patricia Bouyer, Antoine Petit, Denis Thérien. An Algebraic Characterization of Data and Timed Languages. Research Report LSV-01-1, ENS de Cachan, Februar 2001.

[BRJ99] Grady Brooch, James Rumbaugh, Ivar Jacobson. *The Unified Modelling Language User Guide*. Addison Wesley Longman, 1999.

[CE81] Edmund M. Clark, E. Allen Emerson. Design and Synthesis of Synchronization Sceletons Usind Branching Time Temporal Logic. In *Proceedings of the Workshop on Logics of Programs*, Band 131 von *LNCS*, Seiten 52–71. Springer-Verlag, 1981.

[CEMR98] J. M. H. Cobben, A. Engels, Sjouke Mauw, Michel A. Reniers. *Formal Semantics of Message Sequence Charts*. ITU, 1998. *Anhang B von [ITU99]*.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[CW96] Edmund M. Clarke, Jeannette M. Wing. Formal Methods: State of the Art and Future Directions. Technical Report CMU-CS-96-178, CMU, September 1996.

[DH99] Werner Damm, David Harel. LSCs: Breathing Life into Message Sequence Charts. In *Proceedings of the 3rd IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'99)*. Kluwer, 1999.

[E92] Marcel Erné. The Number of Partially Ordered Sets with More Points than Incomparable Pairs. *Discrete Mathematics*, 105:49–60, 1992.

[E00] A. G. Engels. Design Decisions on Data and Guards in MSC2000. In *Proceedings of the 2nd Workshop on SDL and MSC (SAM'00)*, 2000.

[EMR97a] A. G. Engels, Sjouke Mauw, Michel A. Reniers. A Hierarchy of Communication Models for Message Sequence Charts. In *Proceedings of the 10th International Conference on Formal Description Techniques (FORTE/PSTV'97)*, Seiten 75–90, 1997. *Kurzversion von [EMR97b]*.

[EMR97b] A. G. Engels, Sjouke Mauw, Michel A. Reniers. A Hierarchy of Communication Models for Message Sequence Charts. Technical Report CS 97/11, TU Eindhoven, 1997. *Langversion von [EMR97a]*.

[GM02] Blaise Genest, Anca Muscholl. Pattern Matching and Membership for Hierarchical Message Sequence Charts. In *Proceedings of LATIN'02 (LATIN'02)*, LNCS. Springer-Verlag, 2002. Erscheint.

[GMP01] Elsa L. Gunter, Anca Muscholl, Doron A. Peled. Compositional Message Sequence Charts. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, Band 2031 von *LNCS*, Seiten 496–511. Springer-Verlag, 2001.

[GRG93] Peter Graubmann, Ekkart Rudolph, Jens Grabowski. Towards a Petri Net Based Semantics Definition for Message Sequence Charts. In *Proceedings of the 6th SDL Forum (SDL'93)*, Seiten 179–190. Amsterdam, North-Holland, 1993.

[H87] David Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274, 1987.

[H01] fflystein Haugen. From MSC-2000 to UML 2.0 – The Future of Sequence Diagrams. In *Proceedings of the 10th International SDL Forum (SDL2001)*, Band 2078 von *LNCS*, Seiten 38–51. Springer-Verlag, Juni 2001.

[HM00] Loïc Hélouët, Pierre Le Maigat. Decomposition of Message Sequence Charts. In *Proceedings of the 2nd Workshop on SDL and MSC (SAM'00)*, 2000.

[HMKT99] Jesper G. Henriksen, Madhavan Mukund, K. Narayan Kumar, P. S. Thiagarajan. Towards a Theory of Regular MSC Languages. Technical report, BRICS, Aarhus, Dezember 1999.

[HMKT00a] Jesper G. Henriksen, Madhavan Mukund, K. Narayan Kumar, P. S. Thiagarajan. On Message Sequence Graphs and Finitely Generated Regular MSC Languages. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP'00)*, Band 1853 von *LNCS*, Seiten 675–686. Springer-Verlag, Juli 2000.

[HMKT00b] Jesper G. Henriksen, Madhavan Mukund, K. Narayan Kumar, P. S. Thiagarajan. Regular Collections of Message Sequence Charts (Extended Abstract). In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS'00)*, Band 1893 von *LNCS*, Seiten 405–414. Springer-Verlag, 2000.

[HU00] John E. Hopcroft, Jeffrey D. Ulman. *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. Oldenbourg, 2000.

[ITU99] ITU. *Recommendation Z.120: Message Sequence Chart*, 1999.

[JP01] Bengt Jonsson, Gerardo Padilla. An Execution Semantics of MSC-2000. In *Proceedings of the 10th International SDL Forum (SDL2001)*, Band 2078 von *LNCS*, Seiten 365–378. Springer-Verlag, Juni 2001.

[K01] Dietrich Kuske. Another Step Towards a Theory of Regular MSC Languages. Technical Report 2001-36, Department of Mathematics and Computer Science, Universit of Leicester, 2001.

[KL98] Joost-Pieter Katoen, Lennard Lambert. Pomsets for Message Sequence Charts. In *Proceedings of the 8th German Workshop on Formal Description Techniques for Distributed Systems (FBT'98)*, Seiten 197–208. Shaker-Verlag, 1998.

[KR75] D. J. Kleitmann, B. L. Rothschild. Asymptotic Enumeration of Partial Orders on a Finite Set. *Transactions of the American Mathematical Society*, 205:205–220, 1975.

[KW01] Jochen Klose, Hartmut Wittke. An Automata Based Interpretation of Live Sequence Charts. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, Band 2031 von *LNCS*, Seiten 512–527, April 2001.

[L78] Leslie Lamport. Time, Clocks and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565, Juli 1978.

[L02] Philipp Lucas. Timed Semantics of Message Sequence Charts Based on Timed Automata. In *Proceedings of the Workshop of Theory and Practice of Timed Systems 2002 (TPTS'02)*, April 2002. Erscheint.

[LL93a] Peter B. Ladkin, Stefan Leue. Interpreting Message Sequence Charts. Technical Report 101, University of Stirling, März 1993. *Langversion von [LL93b]*.

[LL93b] Peter B. Ladkin, Stefan Leue. What Do Message Sequence Charts Mean? In *Proceedings of the 6th International Conference on Formal Description Techniques (FORTE'93)*, Seiten 301–316, Oktober 1993. *Kurzversion von [LL93a]*.

[LM02] Markus Lohrey, Anca Muscholl. Bounded MSC Communication. In *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'02)*, LNCS. Springer-Verlag, 2002. Erscheint.

[LMST01] Ruggero Lanotte, Andrea Maggiolo-Schettini, Simone Tini. Concurrency in Timed Automata. In *Proceedings of the 13th International Symposium on Fundamentals of Computation Theory (FCT'01)*, Band 2038 von *LNCS*, Seiten 240–251. Springer-Verlag, August 2001.

[LP97] Vladimir Levin, Doron Peled. Verification of Message Sequence Charts via Template Matching. In *Proceedings of the 7th International Conference on Theory and Practice of Software Development (TAPSOFT'97)*, Band 1214 von *LNCS*. Springer-Verlag, April 1997.

[M01a] P. Madhusudan. Reasoning about Sequential and Branching Behaviours of Message Sequence Charts. In *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming (ICALP'01)*, Band 2076 von *LNCS*, Seiten 809–820. Springer-Verlag, Juli 2001.

[M01b] Rémi Morin. On Regular Message Sequence Chart Languages and Relationships to Mazurkiewicz Trace Theory. In *Proceedings of the 4h International Conference on Foundations of Software Science and Computation Structures (FOSSACS'01)*, Band 2030 von *LNCS*, Seiten 332–346. Springer-Verlag, April 2001.

[MM01] P. Madhususan, B. Meenakshi. Beyond Message Sequence Graphs. In *21st International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2001)*, Band 2245 von *LNCS*, Seiten 256–267. Springer-Verlag, Dezember 2001.

[MP99] Anca Muscholl, Doron A. Peled. Message Sequence Graphs and Decision Problems on Mazurkiewicz Traces. In *Proceedings of the 24th International Symposium on Mathematical Foundations of Computer Science (MFCS'99)*, Band 1672 von *LNCS*, Seiten 81–91. Springer-Verlag, September 1999.

[MP01] Anca Muscholl, Doron Peled. From Finite State Communication Protocols to High-Level Message Sequence Charts. In *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming (ICALP'01)*, Band 2076 von *LNCS*, Seiten 720–731. Springer-Verlag, Juli 2001.

[MPS98] Anca Muscholl, Doron A. Peled, Zhendong Su. Deciding Properties for Message Sequence Charts. In *Proceedings of the 1st International Conference on Foundations of Software Science and Computation Structures (FOSSACS'98)*, Band 1278 von *LNCS*, Seiten 226–242. Springer-Verlag, 1998.

[MR99] Sjouke Mauw, Michel A. Reniers. Operational Semantics for MSC'96. *Computer Networks and ISDN Systems*, 31(17):1785–1799, 1999.

[MRW00] Sjouke Mauw, Michel A. Reniers, T. A. C. Willemse. Message Sequence Charts in the Software Engineering Process. Technical Report CS 00/12, TU Eindhoven, 2000.

[N01] Mogens Nielsen. Modelling with Partial Orders – Why and Why Not? (Extended Abstract). In *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming (ICALP'01)*, Band 2076 von *LNCS*, Seiten 61–63. Springer-Verlag, Juli 2001.

[P46] Emil L. Post. A Variant of a Recursively Unsolvable Problem. *Bulletin of the AMS*, 52:264–268, 1946.

[R99] Michel A. Reniers. *Message Sequence Charts: Syntax and Semantics*. Doktorarbeit, TU Eindhoven, 1999.

[RGG96] Ekkart Rudolph, Jens Grabowski, Peter Graubmann. Tutorial on Message Sequence Charts (MSC'96). In *Proceedings of the 10th International Conference on Formal Description Techniques (FORTE/PSTV'96)*, Oktober 1996.

[RJB99] James Rumbaugh, Ivar Jacobson, Grady Brooch. *The Unified Modelling Language Reference Manual*. Addison Wesley Longman, 1999.

[S30] Edward Szpilrajn. Sur l'extension de l'ordre partiel (Französisch). *Fundamenta Mathematicae*, 16:386–389, 1930.

[T68] K. Thompson. Expression Search Algorithms. *Communications of the ACM*, 11(6):419–422, 1968.

[TDPVG01] Stephan Tobies, Peter H. Deussen, Stephan Pietsch, Teophanis Vassillou-Gioles. Formal Test Purposes and Test System Validity, 2001. Eingereicht.

[W89] Glynn Winskel. An Introduction to Event Structures. In *REX Summerschool in Temporal Logic*, Band 354 von *Lecture Notes in Computer Science*, Seiten 364–397. Springer-Verlag, 1989.

[W94] Thomas Wilke. *Automaten und Logiken zur Beschreibung zeitabhängiger Systeme*. Doktorarbeit, Christian-Albrechts-Universität zu Kiel, 1994.