Prioritätszählerautomaten und die Synchronisation von Halbspursprachen

Von der Fakultät Informatik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der Naturwissenschaften
genehmigte Abhandlung

von Klaus Reinhardt aus Stuttgart

Hauptberichter: Prof. Dr. V. Diekert
Mitberichter: Prof. Dr. K.-J. Lange

Tag der mündlichen Prüfung: 7. Juli 1994

Stuttgart, 1994

Vorwort

Mein besonderer Dank gilt Prof. Dr. Volker Diekert, der mein Interesse für Halbspuren geweckt hat, für die intensive Unterstützung und Betreuung meiner Arbeit und Prof. Dr. Klaus-Jörn Lange für die Übernahme des Mitberichtes.
Ebenso danke ich Michael Bertol, Werner Ebinger, Anca Muscholl, Wolfgang Reissenberger und Pierre-Andre Wacrenier für viele interessante Diskussionen und Prof. Dr. Michel Latteux, der mich auf das Problem der Maximalität aufmerksam gemacht hat.

Zusammenfassung

In dieser Arbeit wird gezeigt daß das Leerheitsproblem für die von einem Prioritätsmulticounterautomaten erkannte Sprache und damit auch das Halteproblem für Prioritätsmulticounterautomaten entscheidbar ist. Hierbei wird vorausgesetzt, daß das Erreichbarkeitsproblem für Petrinetze, bei denen die inhibitorischen Kanten auf eine bestimmte Art und Weise angeordnet sind (siehe [Rei94]), entscheidbar ist. Die Beziehungen der von diesen Automaten erkannten Prioritätsmulticountersprachen zu anderen Sprachklassen wird beschrieben und es wird gezeigt, daß die Sprache $\{(a^n b)^m \mid n, m \in \mathbb{N}\}$ keine Prioritätsmulticountersprache ist.

Damit werden notwendige und hinreichende Kriterien für solche Semikommutationssysteme entwickelt, bei denen eine Halbspursprache, d.h. die die Hülle einer regulären Sprache unter diesem Semikommutationssystem, von einem Prioritätsmulticounterautomaten erkannt und damit die Maximalität und die Synchronisierbarkeit mit einer regulären Sprache entschieden werden kann. Ferner werden Kriterien für Semikommutationssysteme entwickelt, bei denen die Maximalität und die Synchronisierbarkeit von Halbspursprachen unentscheidbar ist.

Für Halbspuren werden Kriterien für die Inklusion, die Synchronisierbarkeit, für die Frage, ob die Synchronisation von Halbspuren wieder eine Halbspur ist und für die Existenz einer nicht konfluenten Situation gezeigt und deren Komplexität ermittelt. Gleiches erfolgt für diese Eigenschaften der Halbspuren eines Semikommutationssystems im allgemeinen.

Inhaltsverzeichnis

1 Einleitung

Bei der formalen Beschreibung von Prozessen spielt die Reihenfolge der einzelnen Aktionen bereits bei sequentiellen Systemen eine große Rolle; z.B. bei der Optimierung von Programmcodes ist es wichtig zu betrachten, welche der einzelnen Anweisungen mit welchen anderen vertauscht werden können und welche abhängig sind. Beispielsweise sind die Zuweisungen $m := n + o$ und $o := n * p$ abhängig, wohingegen die Zuweisungen $m := n + o$ und $p := n * o$ unabhängig sind. Um dies mit mathematischen Instrumenten beschreiben zu können, werden Monoide von Spuren betrachtet. Die Spur, welche die möglichen Ablauffolgen eines Prozesses beschreibt, ist eine Äquivalenzklasse von durch Vertauschen unabhängiger Aktionen auseinander ableitbaren Ablauffolgen. Durch Spuren und Spursprachen kann man somit gültige Evaluierungen eines Prozesses beschreiben.
Da die Computertechnik mit der Schaltgeschwindigkeit von Halbleiterelementen auf Grenzen stößt, kommt die Forschung in der letzten Zeit von nur sequentiell arbeitenden Computern immer mehr zu parallelen Systemen. Durch die nun auftretende Nebenläufigkeit von Aktionen in einem parallelen Prozess erhält die richtige Behandlung der Reihenfolge eine ganz entscheidende Bedeutung.
Beim modularen Aufbau von Prozessen ist es unumgänglich, daß die Unterprozesse Daten miteinander austauschen. Zu diesem Zweck muß eine Synchronisation der Unterprozesse stattfinden. Produziert z.B. ein Unterprozess in Aktionen mit der Bezeichnung p Daten einer bestimmten Art und ein anderer Unterprozess konsumiert in Aktionen mit der Bezeichnung c Daten dieser Art, so ist c abhängig von p, d.h. findet eine Aktion p vor c statt, so kann deren Reihenfolge nicht vertauscht werden.
Umgekehrt aber kann, falls eine Aktion c vor p stattfinden kann, deren Reihenfolge vertauscht werden. Da dies mit den symmetrischen Kommutationen nicht beschrieben werden kann, werden in dieser Arbeit Semikommutationssysteme betrachtet. Mögliche Ablauffolgen werden nun durch Halbspuren beschrieben. Interessante Fragestellungen sind dabei, ob eine gegebene Ablauffolge in einer gegebenen Halbspur liegt, ob gegebene Halbspuren synchronisierbar sind, ob deren Synchronisation wieder eine Halbspur ist (siehe Kapitel 4) und ob zwei Halbspuren, die Teilmenge einer gegebenen Halbspur sind, immer synchronisierbar sind (Konfluenz siehe Kapitel 6). Auch möchte man bei gegebenen Semikommutationssystemen bereits im Vorfeld erkennen können, welche Fälle darin auftreten können, z.B. ob es darin Halbspuren gibt, deren Synchronisation selbst keine Halbspur mehr ist.
Für (endliche) Halbspuren sind alle diese Fragen entscheidbar und die Erkennung der gültigen Ablauffolgen kann durch einen endlichen Automaten erfolgen. Für Halbspursprachen werden die Fragen jedoch i.a. unentscheidbar. In einigen Fällen können verschiedene Arten von Zählerautomaten verwendet werden (siehe Kapitel 5).

Eine andere mächtige Beschreibungsmethode für nebenläufige Systeme, welche viel weiter verbreitet ist, sind die von C. A. Petri Anfang der 60'er Jahre vorgestellten Petrinetze. Eine wichtige Fragestellung bei Petrinetzen ist, ob eine bestimmte Konfiguration von einer Startkonfiguration aus erreichbar ist. Bei Petrinetzen ohne inhibitorische Kanten ist die Bedingung dafür, daß eine Aktion stattfinden kann, immer nur das Vorhandensein von Marken auf Stellen, die z.B. irgendwelche Resourcen repräsentieren. Für solche Petrinetze ist bereits bekannt, daß die Frage der Erreichbarkeit entscheidbar ist. In vielen Fällen will man jedoch eine Bedingung formulieren können, welche besagt, daß das Vorhandensein einer Marke eine Aktion verhindert. Dies kann nur durch eine inhibitorische Kante ausgedrückt werden.
In [Rei94] wird gezeigt, daß das Erreichbarkeitsproblem bei denjenigen Petrinetzen entscheidbar ist, bei denen die Lage der inhibitorischen Kanten eine Anordnung (Indizierung) der Stellen mit folgender Eigenschaft ermöglicht: Hat eine Transition t eine inhibitorische Kante von einer Stelle s_j, so hat sie auch inhibitorische Kanten von allen kleineren Stellen s_i mit $i \leq j$. Ein Spezialfall davon ist, daß nur eine Stelle auf Null getestet werden kann, d.h. daß nur von einer Stelle inhibitorische Kanten ausgehen.
In Kapitel 3 wird der dazu korrespondierende Prioritätsmulticounterautomat beschrieben, bei dem die Anordnung der Zähler der Anordnung der Stellen im Petrinetz entspricht und dessen Leerheits- bzw. Halteproblem dem Erreichbarkeits problem im Petrinetz entspricht. Dieser Prioritätsmulticounterautomat wird später in Kapitel 5 verwendet um bestimmte Halbspursprachen zu erkennen.

2 Begriffe

2.1 Grundbegriffe

Wichtige Bezeichnungen aus [HU79] werden im folgenden zusammengefaßt beschrieben:
Für ein *Alphabet*, d.h. eine Menge von Zeichen Σ ist

$$\Sigma^n := \{x_1x_2...x_n \mid x_i \in \Sigma \text{ für } 1 \leq i \leq n\}$$

die Menge der *Wörter* der Länge $|x_1x_2...x_n| := n$. Die Menge aller Wörter über Σ ist $\Sigma^* := \bigcup_{n\in\mathbb{N}} \Sigma^n$ und $\Sigma^+ := \bigcup_{n\in\mathbb{N}^+} \Sigma^n$, wobei $\mathbb{N} = \{0, 1, ...\}$ die Menge der natürlichen Zahlen, $\mathbb{N}_+ = \{1, 2, ...\}$ die Menge der positiven natürlichen Zahlen und $\mathbb{Z} = \{..., -1, 0, 1, ...\}$ die Menge der ganzen Zahlen ist. Das Leerwort ist λ mit $|\lambda| = 0$.
Eine Menge von Wörtern $L \subseteq \Sigma^*$ nennt man *Sprache*. Für Wörter $x = x_1...x_n$ und $y = y_1...y_m$ ist die *Konkatenation* $xy := x_1...x_ny_1...y_m$ und die *Spiegelung* $x^R := x_n...x_2x_1$. Σ bildet also mit der Konkatenation und dem neutralen Element λ ein *Monoid*. Die Konkatenation für Sprachen L_1 und L_2 ist $L_1L_2 := \{w_1w_2 \mid w_1 \in L_1 \text{ und } w_2 \in L_2\}$; der *Sternoperator* * ist für Sprachen definiert durch $L^* := \bigcup_{n\in\mathbb{N}} L^n$ und analog dazu $L^+ := \bigcup_{n\in\mathbb{N}^+} L^n$. Die Anzahl der Vorkommen eines Zeichens x in einem Wort w ist $|w|_x$.
Ein Wort $\alpha\beta\gamma$ kann mittels einer *Produktionsregel* $\beta \to \delta$ in einer Menge P von Produktionsregeln abgeleitet werden zu einem Wort $\alpha\delta\gamma$. Wir schreiben dafür $\alpha\beta\gamma \underset{P}{\Longrightarrow} \alpha\delta\gamma$. Es gilt $w_0 \overset{*}{\underset{P}{\Longrightarrow}} w_n$, falls $w_1, ...w_{n-1}$ existieren mit $w_0 \underset{P}{\Longrightarrow} w_1 \underset{P}{\Longrightarrow} ...w_n$ und falls auch noch $n > 0$, so gilt $w_0 \overset{+}{\underset{P}{\Longrightarrow}} w_n$. Durch eine *Grammatik* $G = (V, \Sigma, P, S)$ mit der Variablenmenge $V = \{S, ...\}$ und den Produktionsregeln

$$P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma \setminus \{S\})^*$$

wird die Sprache $L(G) := \{x \in \Sigma^* \mid S \overset{*}{\underset{P}{\Longrightarrow}} x\}$ erzeugt. Gilt $P \subseteq V \times (\Sigma^+V \cup \Sigma) \cup \{S \to \lambda\}$, so nennt man G und damit auch $L(G)$ *regulär*, im Fall

$$P \subseteq V \times (V \cup \Sigma)^+ \cup \{S \to \lambda\}$$

kontextfrei und im Fall $P \subseteq V \times (\Sigma^*V\Sigma^* \cup \Sigma) \cup \{S \to \lambda\}$ *linear*.

Die *Projektion* eines Wortes oder einer Sprache auf ein Alphabet A ist der Homomorphismus Π_A mit $\Pi_A(a) = a$, falls $a \in A$ und $\Pi_A(a) = \lambda$, falls $a \notin A$.
Die Zahl $ggT(n, m) \in \mathbb{N}$ ist der *größte gemeinsame Teiler* zweier natürlicher Zahlen n und m.
Für Mengen ist $|M|$ die Anzahl der Elemente in M.

2.2 Spuren und Kommutationen

Ein *Abhängigkeitsaphabet* über einem Alphabet A ist ein Paar (A, D) mit einer reflexiven und symmetrischen *Abhängigkeitsrelation* $D \subseteq A^2$. Durch dieses ist ein *Kommutationssystem* $C := \{ab \rightarrow ba \mid (a,b) \in A^2 \setminus D\}$ definiert. *Spuren* sind Äquivalenzklassen der Form $[w] := \{v \mid w \overset{*}{\underset{C}{\Longrightarrow}} v\}$; über diesen entsteht ein Monoid $M(A, D)$ mit der Multiplikation $[w] \cdot [u] = [wu]$ und dem neutralen Element $[\lambda]$. Eine Spur $[a_1...a_n]$ wird repräsentiert als ein Graph mit Knoten, welche mit $a_1, ...a_n$ markiert sind, und gerichteten Kanten von einem mit a_i markierten Knoten zu einem mit a_j markierten Knoten, für $(a_i, a_j) \in D$ und $i < j$.
Diese freien, partiell kommutativen Monoide wurden zuerst in der Mathematik von P. Cartier und D. Foata [CF69] betrachtet. In der Informatik wurden sie erstmals von A. Mazurkiewicz [Maz77] zu Beschreibung von Schaltfolgen in 1-sicheren Petrinetzen verwendet.

2.3 Halbspuren und Semikommutationen

Die *Semikommutationen* wurden erstmals von M.Clerbout in ihrer Dissertation [Cle84] als Verallgemeinerung der partiellen Kommutation definiert (siehe auch M. Clerbout und M. Latteux [CL87]) und eine Menge von Resultaten entstanden in letzter Zeit auf diesem Gebiet, z.B. [CG94], [CGL$^+$92], [Gon93], [Lat92], [RW91]. Mit Semikommutationen ist es nun z.B. auch möglich Systeme mit Produzenten und Konsumenten zu beschreiben und in verschiedenen Arbeiten wie [HK89], [Och90] und [Och92] stellen sich Semikommutationen bei Beschreibung von Petrinetzen als nützlich heraus.
Zu jedem *Halbabhängigkeitsalphabet* (A, SD) mit einer reflexiven, aber möglicherweise asymmetrischen *Halbabhängigkeitsrelation* $SD \subseteq A \times A$ ist das dazugehörige *Semikommutationssystem* definiert durch $SC = \{ab \rightarrow ba \mid (a,b) \notin SD\}$. Eine *Halbspur* über (A, SD) ist definiert als die Menge der Wörter $[u\rangle = \{w \in A^* \mid u \overset{*}{\underset{SC}{\Longrightarrow}} w\}$, welche von einem Wort $u \in A^*$ durch Anwendung von Ableitungsregeln in SC abgeleitet werden können. Gibt es mehrere Halbabhängigkeitsalphabete (A_i, SD_i), so werden die zugehörigen Halbspuren $[u\rangle_i$ ebenfalls indiziert.
Es ist leicht zu sehen, daß zwei Halbspuren genau dann gleich sind, wenn sie die gleiche Spur im Monoid $M(A, D)$ mit $D = SD \cup SD^{-1}$ repräsentieren. Dies legt nahe eine Halbspur $[u\rangle$ über (A, SD) durch den Abhängigkeitsgraphen der Spur $[u]$ in $M(A, SD \cup SD^{-1})$ darzustellen. Zur Darstellung der Halbabhängigkeitsstruktur ersetzen wir jede Kante $a \rightarrow b$ durch eine gepunktete Kante $a \dashrightarrow b$, wenn $(a,b) \notin SD$. Die gepunkteten Kanten nennen wir *weiche Kanten*. Die anderen Kanten $a \rightarrow b$ mit $(a,b) \in SD$ nennen wir *harte Kanten*, da sie eine nicht abänderbare Reihenfolge ausdrücken. Ein Semikommutationssystem kann als *Graphersetzungssystem* betrachtet werden, bei dem eine weiche Kante $a \dashrightarrow b$ abgeleitet werden kann zu einer harten Kante $a \leftarrow b$, wobei die Bedingung, daß keine Kreise entstehen, eingehalten werden muß.

Die Hülle einer Sprache L unter einem Semikommutationssystem SC ist die Menge $f_{SC}(L) = \{w \in A^* \mid \exists u \in L, u \stackrel{*}{\underset{SC}{\Longrightarrow}} w\}$ der Wörter, welche von einem Wort der Sprache abgeleitet werden können ($f_{SC}(w) = f_{SC}(\{w\})$). Unter einer *Halbspursprache* verstehen wir die Hülle $f_{SC}(R)$ einer regulären Sprache R.

Ein Semikommutationssystem SC ist die *Komposition* zweier Semikommutationssysteme SC_1 und SC_2, wenn $\forall w\ f_{SC}(w) = f_{SC_1}(f_{SC_2}(w))$ gilt.

Um komplexe Systeme in modularer Weise aufbauen zu können, benötigen wir die Synchronisation von Halbspuren in der gleichen Weise, wie sie von Mazurkiewicz in [Maz87] für die symmetrischen partiellen Kommutationen verwendet werden.

Im folgenden werden wir $(A, SD) = (\bigcup_i A_i, \bigcup_i SD_i)$ als Synchronisationshalbabhängigkeitsalphabet verwenden. Die *Synchronisation von Halbspuren* ist

$$[u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots \parallel [u_m\rangle_m = \{w \in A^* \mid \forall 0 < i \leq m \ \ \Pi^A_{A_i}(w) \in [u_i\rangle_i\},$$

wobei $\Pi^A_{A_i}(w)$ die *Projektion* der Wörter w über A_i mittels des Homomorphismus $\Pi^A_{A_i}$ mit für alle $a \in A$ wenn $a \in A_i$ dann $\Pi^A_{A_i}(a) = a$ sonst $\Pi^A_{A_i}(a) = \lambda$ ist. Wir nennen Halbspuren *synchronisierbar*, wenn ihre Synchronisation nicht die leere Menge ist. Allgemein ist die *Synchronisation* von Sprachen über dem Alphabet A definiert durch $L_1 \parallel L_2 \parallel \ldots \parallel L_m = \{w \in A^* \mid \forall 0 < i \leq m \ \ \Pi^A_{A_i}(w) \in L_i\}$.

Bemerkung: Es ist leicht zu sehen, daß die Synchronisation über einem einzelnen Alphabet gerade der Schnitt ist:
Für $(A_1, SD_1) = (A_2, SD_2)$ gilt $[u\rangle_1 \parallel [v\rangle_2 = [u\rangle_1 \cap [v\rangle_2$

2.4 Komplexitätsklassen

Bei der Analyse eines Problems genügt es nicht nur festzustellen, ob ein Problem von einem Computer bzw. einer Turingmaschine (siehe [HU79]) entschieden werden kann. In der Komplexitätstheorie untersucht man daher den Platz- und den Zeitbedarf in Abhängigkeit von der Eingabelänge, welche benötigt wird, ein Problem (z.B. die Zugehörigkeit eines Wortes zu einer Sprache) zu lösen und teilt die Probleme (Sprachen) in Komplexitätsklassen ein:

$DSPACE(S(n))$ (bzw. $NSPACE(S(n))$) ist die Klasse der Sprachen, die von einer deterministischen (bzw. nichtdeterministischen) Turingmaschine erkannt werden, die bei einer Eingabe der Länge n auf jeder Berechnung nur $S(n)$ Platz benötigt.
$DTIME(\mathrm{T(n)})$ (bzw. $NTIME(\mathrm{T(n)})$) ist die Klasse der Sprachen, die von einer deterministischen (bzw. nichtdeterministischen Turingmaschine erkannt werden, die bei einer Eingabe der Länge n auf jeder Berechnung nur T(n) Schritte benötigt.

Besondere Komplexitätsklassen sind

$$\begin{aligned} NLOGSPACE &:= NSPACE(\log n) \\ P &:= \bigcup_{c>1} DTIME(n^c) \\ NP &:= \bigcup_{c>1} NTIME(n^c) \\ PSPACE &:= \bigcup_{c>1} DSPACE(n^c) \\ EXPTIME &:= \bigcup_{c>1} DTIME(2^{n^c}) \end{aligned}$$

2.4.1 Reduktion und Orakel

Orakel machen es möglich bei der Lösung eines Problems A ein anderes Problem B als bereits kostenlos gelöst zu betrachten. Eine Möglichkeit, A auf B zu reduzieren, ist die *Turingreduktion* ($A \leq_T B$), welche von einer Orakelturingmaschine ausgeführt wird, die ein Ausgabeband und einen Fragezustand mit zwei Antwortzuständen besitzt. In welchem der Antwortzustände die Rechnung fortgesetzt wird, wenn die Orakelturingmaschine in den Fragezustand geht, hängt davon ab, ob der Inhalt des Ausgabebandes in der Orakelsprache liegt. Eine Rechnung, die von der Eingabe überprüfen soll, ob sie in A liegt, kann das Orakel B mehrfach benutzen und ist nicht an die Antwort des Orakels gebunden, d.h. sie muß nicht akzeptieren, wenn das gefragte Wort im Orakel liegt. Ist C eine Sprachklasse, so ist C^A die Klasse der Sprachen, die mit entsprechenden Orakelmaschinen mit Orakel A berechnet werden können; so ist beispielsweise $NP^A = \{L \mid L \leq_T^{np} A\}$. Die *polynomielle Hierarchie* ist als Orakelhierarchie definiert durch $\Sigma_0^P := P$ und $\Sigma_{k+1}^P := NP^{\Sigma_k^P}$ für $k > 0$.
In dieser Arbeit verwenden wir jedoch nur die many-one-Reduktion, die von einem Transducer durchgeführt wird, der das Orakel nur einmal benutzt; das Ergebnis ist die Antwort des Orakels, d.h. der Inhalt des Ausgabebandes muß genau dann in B sein, wenn die Eingabe in A liegt.
Ein *logarithmischer Transducer* $T = (Z, \Sigma, \Gamma, \Delta, \delta, z_0, E)$ ist eine deterministische Turingmaschine mit einem (Zweiweg)Eingabeband mit Eingabealphabet Σ, einem Einwegausgabeband mit Ausgabealphabet Γ, auf das nur geschrieben aber nicht gelesen werden darf, einem logarithmisch platzbeschränkten Arbeitsband mit Bandalphabet Δ, einer Übergangsfunktion $\delta : (Z \times \Sigma \times \Delta) \mapsto (Z \times \Delta \times \Gamma^*)$, einem Anfangszustand $z_0 \in Z$ und den Endzuständen $E \subseteq Z$.
Mit $f_T(x)$ bezeichnen wir das Wort, das nach Erreichen eines Endzustandes auf dem Ausgabeband steht.
Es gilt $L \leq^{log} L'$, wenn es einen logarithmischen Transducer T mit $x \in L$ gdw. $f_T(x) \in L'$ gibt.
Ein *endlicher Transducer* $T = (Z, \Sigma, \Gamma, \delta, z_0, E)$ (gemäß [Ber79] auch *rationaler Transducer*) ist ein Automat mit einem Einwegeingabeband mit Eingabealphabet Σ, einem Einwegausgabeband mit Ausgabealphabet Γ, einer Übergangsfunktion $\delta : (Z \times \Sigma) \mapsto (Z \times \Gamma^*)$ (bzw. einer Übergangsrelation $\delta \subset (Z \times \Sigma) \times (Z \times \Gamma^*)$, falls

T nichtdeterministisch ist), einem Anfangszustand $z_0 \in Z$ und den Endzuständen $E \subseteq Z$.
Für einen deterministischen Transducer T ist $f_T(x) \in \Gamma^*$ das Wort, das nach Durchlaufen der Eingabe $x \in \Sigma^*$ auf dem Eingabeband auf dem Ausgabeband steht, wenn ein Endzustand erreicht wurde.
Bei einem nichtdeterministischen Transducer ist f_T eine *reguläre Relation* (auch *rationale Relation*), d.h. sie kann von einem nichtdeterministischen endlichen Automaten mit zwei Eingabebändern erkannt werden.
Es gilt $L \leq^f L'$ ($L \leq^{nf} L'$), wenn es einen deterministischen (nichtdeterministischen) endlichen Transducer T mit Einwegeingabeband und Einwegausgabeband gibt mit $x \in L$ gdw. es einen Ausgabewert $f_T(x) \in L'$ gibt ($\exists y \in L'$ und $(x, y) \in f_T$).
Für eine Sprachklasse S ist

$$\begin{aligned} LOG(S) &:= \{Y \mid \exists X \in S \text{ mit } Y \leq^{log} X\}, \\ co\text{–}S &:= \{Y \mid \overline{Y} \in S\}, \end{aligned}$$

eine Sprache V heißt *vollständig für S bezüglich einer Reduktion '$\leq$'*, wenn $V \in S$ und $\forall X \in S\ X \leq V$ gilt. Eine Sprachklasse S ist *abgeschlossen* bezüglich einer Reduktion '$\leq$', wenn für alle $L \leq L' \in S$ auch $L \in S$ liegt.
Eine Sprachklasse S ist eine *abstrakte Familie von Sprachen*[1] (*AFL*), wenn S unter Vereinigung, Konkatenation, dem Sternoperator und *rationaler Transduktion* ($\leq^{nf}$) abgeschlossen ist. Beispielsweise ist die Klasse *CFL* der kontextfreien Sprachen eine *AFL*.
Eine Äquivalenzrelation δ über Σ^* ist eine *Kongruenz*, wenn für alle $w, x, y, z \in \Sigma^*$ mit $[w]_\delta = [x]_\delta$ und $[y]_\delta = [z]_\delta$ auch $[wy]_\delta = [xz]_\delta$ gilt. Die *Semi-Dyck-Sprache* ${D'_n}^*$ über dem Alphabet $\{a_1, ...a_n, b_1, ..., b_n\}$ ist die Sprache der Wörter, die kongruent mit dem leeren Wort λ sind bezüglich der Kongruenz δ' mit $[a_i b_i]_{\delta'} = [\lambda]_{\delta'}$ für alle $i \leq n$. Die Semi-Dyck-Sprache

$${D'_n}^* := \{w \in \{a_1, ...a_n, b_1, ..., b_n\} \mid [w]_{\delta'} = [\lambda]_{\delta'}\}$$

kann auch als Klammersprache bezeichnet werden, wobei a_i eine öffnende Klammer und b_i die dazugehörige schließende Klammer ist.
Für $n > 1$ ist ${D'_n}^*$ vollständig für die Klasse *CFL* bezüglich rationaler Transduktion. Die Sprache ${D'_1}^*$ ist hingegen nur vollständig bezüglich rationaler Transduktion für die Klasse *ROCL* (siehe [Ber79]) der Sprachen, die von einem Automaten mit einem schwachen Zähler erkannt werden können. Die Sprache $({D'_1}^* c)^*$ ist vollständig bezüglich rationaler Transduktion für die Klasse *OCL* (siehe [Ber79]) der Sprachen, die von einem Automaten mit einem starken Zähler erkannt werden können und die Sprache $\{w\$w^R \mid w \in \{a, b\}^*\}^*$ ist vollständig bezüglich rationaler Transduktion für die Klasse *LIN* der linearen Sprachen.

[1] Eine genauere Bezeichnung ist 'volle abstrakte Familie von Sprachen'. Nicht volle AFL's, bei denen die rationalen Transduktionen nicht löschend sein dürfen, kommen in dieser Arbeit nicht vor.

2.4.2 Schaltkreiskomplexitätsklassen

Unterhalb von P wurden viele Komplexitätsklassen mit verschiedenen Arten von parallelen Random Access Maschinen, verschiedenen Arten von Schaltkreisen, verschiedenen sequenziellen Modellen mit unterschiedlichen Akzeptanzbedingungen z.B. Alternierung oder Zählkomplexitätsklassen und die logarithmische Hülle über formalen Sprachklassen z.B. $LOG(CFL)$ betrachtet und viele Beziehungen zwischen diesen Modellen gefunden (siehe z.B. [LR94], [LR90], [Par90], [Rei92a], [Rei90], [SV84], usw.). Von Interesse ist in dieser Arbeit die Klasse TC^0, in der diejenigen Sprachen liegen, welche von einer uniformen Familie von Schaltkreisen aus Threshold-Gattern mit konstanter Tiefe und polynomieller Größe erkannt werden können, d.h. für jede Eingabelänge existiert ein Schaltkreis in dieser Familie mit der Eigenschaft, daß für alle Eingaben dieser Länge der Ausgang des Schaltkreises genau dann 1 ist, wenn die Eingabe in der Sprache liegt. Als Uniformitätsbedingung verwenden wir die *DLOGTIME-Uniformität* [BIS90]; dies bedeutet, daß die Schaltkreise einer solchen Familie durch Ausdrücke beschrieben werden und die Sprache dieser Ausdrücke

$$\{\langle c, i, y\rangle \mid |y| = n \text{ und das } i\text{-te Zeichen des } n\text{-ten Ausdrucks ist } c\}$$

von einer deterministischen logarithmisch zeitbeschränkten Turingmaschine erkannt werden kann. Threshold-Schaltkreise sind auch ein Modell für neuronale Netze. In TC^0 liegen genau diejenigen Sprachen, welche von einer parallelen Random Access Maschine, welche zusätzlich über die Majoritätsfunktion verfügt, in konstanter Zeit berechnet werden kann [Par90]. Nach [BIS90] ist TC^0 die Klasse der Sprachen, die durch Logik erster Stufe mit zusätzlichem Mayoritätsquantor ($FO{+}MAJ$) beschrieben werden können.
NC^k (AC^k) ist die Klasse der Sprachen, die von einer uniformen Familie von Schaltkreisen aus $\wedge$, $\vee$ und $\neg$ -Gattern mit der Tiefe $O(log^k n)$ und polynomieller Größe mit beschränktem (bzw. unberschränktem) Fan-In erkannt werden können [Ruz81]. Die Probleme in $NC = \bigcup_k NC^k$ betrachtet man häufig als die effizient parallelisierbaren Probleme. Zwar ist nicht bekannt, ob die Klassen sich trennen lassen (z.B. ist nicht bekannt ob $TC^0 \neq NP$), es wird jedoch angenommen, daß viele dieser Klassen verschieden sind, was eine genaue Einteilung von Problemen in diese Klassen interessant macht.

2.5 Petrinetze mit inhibitorischen Kanten

Üblicherweise bedeutet eine positiv gewichtete Kante von einer Stelle zu einer Transition in einem Petrinetz, daß die Transition nur dann schalten darf, wenn genügend Marken auf der Stelle liegen. Im Gegensatz dazu bedeutet eine *inhibitorische Kante*, daß die Transition nur dann schalten darf, wenn *keine* Marke auf der Stelle liegt.
Ein solches Petrinetz beschreiben wir durch ein 6-Tupel (S, T, W, I, m_0, m_e) mit der Gewichtsfunktion $W \in \mathbb{N}^{S\times T\cup T\times S}$, den inhibitorischen Kanten $I \subseteq S \times T$ und den Anfangs- und Endmarkierungen $m_0, m_e \in \mathbb{N}^S$. Die inhibitorischen Kanten werden wir in den Bildern durch ——● darstellen.
Eine Transition $t \in T$ kann eine Markierung m zu der Markierung m' schalten, d.h. $m[t\rangle m'$, wenn $m - W(., t) = m' - W(t, .) \in \mathbb{N}^S$ und $\forall s \in S\ (s, t) \in I \rightarrow m(s) = 0$. Eine Transitionsfolge $w = t_1...t_n \in T^*$ kann m_0 nach m_n schalten, d.h. $m_0[w\rangle m_n$, wenn $m_1, ...m_{n-1}$ existieren mit $m_0[t_1\rangle m_1[t_2\rangle ...[t_n\rangle m_n$.
Das *Erreichbarkeitsproblem* für ein Petrinetz (S, T, W, I, m_0, m_e) ist es, zu entscheiden, ob ein $w \in T^*$ existiert mit $m_0[w\rangle m_e$.
Existieren $s, s' \in S$ und $t, t' \in T$ mit den inhibitorischen Kanten $(s, t), (s', t') \in I$, welche unabhängig liegen, d.h. es gilt $(s, t'), (s', t) \notin I$, so wird das Erreichbarkeitsproblem i.a. unentscheidbar. Ist dies jedoch nicht der Fall, so gilt für alle $s, s' \in S$ und $t, t' \in T$ mit $(s, t), (s', t') \in I$ die Bedingung $(s, t') \in I$ oder $(s', t) \in I$. D.h. für alle $s, s' \in S$ gilt

$$T(s) := \{t \mid (s, t) \notin I\} \subseteq T(s') := \{t \mid (s', t) \notin I\} \text{ oder } T(s) \supseteq T(s').$$

Also lassen sich die Stellen mit dieser Teilmengenbeziehung linear anordnen und zwar so, daß jede Stelle zu all denjenigen Transitionen eine inhibitorische Kante hat, zu denen eine bezüglich der Anordnung größere Stelle eine inhibitorische Kante hat. Diese Ordnung wird in [Rei94] beim Übergang von Petrinetzen mit inhibitorischen Kanten zu geschachtelten Petrinetzen ausgenutzt um die Entscheidbarkeit in diesem Fall zu zeigen.

2.6 Multimengen und Vektoren

Für zwei Mengen A, B ist A^B die Menge der Funktionen $f : B \mapsto A$ von B nach A. Eine solche Funktion mit $f(b_1) = a_1, f(b_2) = a_2, ...$ läßt sich darstellen als Menge $\{b_1 \mapsto a_1, b_2 \mapsto a_2, ...\}$ oder als B-Vektor $\begin{pmatrix} a_1 \\ a_2 \\ \vdots \end{pmatrix}$. Eine Multimenge über B ist eine Funktion in $\mathbb{N}^B$.
Eine Funktion, die auf Multimengen abbildet $f : A \mapsto \mathbb{N}^B$ betrachten wir als Multimenge in $\mathbb{N}^{B \times A}$ mit $f(b, a) = f(a)(b)$, $f(., a) = f(a)$ und $f(b, .) = \{a \mapsto f(b, a) \mid a \in A\}$ oder als $B \times A$-Matrix.
Ist $A \subseteq B$ so gilt $\mathbb{N}^A \subseteq \mathbb{N}^B$, wobei wir eine Funktion an nicht definierten Stellen als um 0 erweitert denken. Es gilt $\mathbb{N}^A \cap \mathbb{N}^B = \mathbb{N}^{A \cap B}$.
Die Einschränkung einer Multimenge $f \in \mathbb{N}^B$ auf A ist $f \mid_A := \{b \mapsto n \mid b \in A \wedge f(b) = n\}$. Dies bedeutet, daß für alle f und A die 3 Aussagen $f \in \mathbb{N}^A$, $f \mid_A = f$ und $\forall x \notin A\ f(x) = 0$ äquivalent sind.
Die Addition zweier Multimengen $f, g \in \mathbb{N}^A$ erklärt sich durch $(f+g)(a) = f(a) + g(a)$ und die Skalarmultiplikation mit $n \in \mathbb{N}$ durch $(fn)(a) = (nf)(a) = nf(a)$.

Sei $M \subseteq \mathbb{N}^A$ eine endliche Menge von Multimengen, so ist die Identitätsfunktion $id_M \in M^M \subseteq \mathbb{N}^{A \times M}$ mit $\forall m \in M\ id_M(., m) = id_M(m) := m$ eine $A \times M$-Matrix, bei der ein Spaltenvektor bei einer Position für ein $m \in M$ der A-Vektor m selbst ist.
Sei $f \in \mathbb{N}^M$ eine Multimenge von Multimengen. Entsprechend der Multiplikation der Matrix id_M mit einem M-Vektor $f \in \mathbb{N}^M$ definieren wir

$$M \cdot f := id_M \cdot f = \sum_{m \in M} mf(m) \ \in \mathbb{N}^A.$$

Diese Multiplikation $M \cdot f := \sum_{m \in M} mf(m)$ wird später bei der geschachtelten Beschreibung von Schaltfolgen wichtig.
Ist $f \in \mathbb{N}^{M'}$ mit $M \subseteq M'$, so ist $M \cdot f := M \cdot (f \mid_M)$.
Sei z.B. $M := \{\{a_1 \mapsto 3, a_2 \mapsto 2\}, \{a_1 \mapsto 1, a_2 \mapsto 5\}\} = \{\begin{pmatrix} 3 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 5 \end{pmatrix}\}$ und $f := \{\begin{pmatrix} 3 \\ 2 \end{pmatrix} \mapsto 4, \begin{pmatrix} 1 \\ 5 \end{pmatrix} \mapsto 7\}$, so ist $M \cdot f = \begin{pmatrix} 3 & 1 \\ 2 & 5 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 7 \end{pmatrix} = \begin{pmatrix} 19 \\ 43 \end{pmatrix} =$
$\{a_1 \mapsto 19, a_2 \mapsto 43\}$.
Für eine endliche Menge M von Multimengen ist $M^* := \{M \cdot f \mid f \in \mathbb{N}^M\}$ die Menge aller Linearkombinationen.

2.7 Geschachtelte Petrinetze

Die Idee eines geschachtelten Petrinetzes ist, daß eine Transition selbst eine Kette (als Liste zwischen [und] dargestellt) hintereinandergeschalteter Unternetze mit inneren Stellen ist, welche vor und nach dem Schalten dieser Transition frei von Marken sein müssen.
Wesentlicher Unterschied zu den Netzsystemen in [Kie89] ist dabei, daß Marken auf äußeren Stellen mit in die Unternetze genommen und und übrige Marken auf diesen Stellen wieder nach außen genommen werden können. (Dies läßt sich mit der Verwendung globaler Variablen in Unterprogrammen vergleichen.)
Die Kantengewichte (welche zuvor mit W bezeichnet wurden) sind nun keine festgelegten Größen außerhalb der Transitionen, sondern sie werden im Innern einer Transition bestimmt. Zu diesem Zweck enthält jede Transition eine semilineare Menge von Multimengen, durch welche der benötigte Zusammenhang von äußerem Schaltverhalten und der inneren Schaltbedingung hergestellt wird. Zusätzlich übernimmt diese semilineare Menge die Aufgabe die Schaltfolgen zu restringieren.
Eine Darstellung als Vektoren (wie sie bei einer hypothetischen Implementation gebraucht würde) brächte einen erheblichen formalen Aufwand zur Berechnung der jeweils benötigten Koordinaten mit sich. Aus diesem Grund wurde die Schreibweise als Multimenge gegenüber der Darstellung als Vektor bevorzugt.

Definition Ein geschachteltes Petrinetz (S_T, T) besteht aus einer Menge T von Transitionen, welche im eigentlichen Sinne Transitionsketten sind, und den Stellen S_T. Dabei gilt:

(1) Eine *Elementartransition* hat die Form $t = ([\,], c, P)$ mit der leeren Liste $[\,]$ von Unternetzen, der *Konstantenmultimenge* $c \in \mathbb{N}^{S_T \times \{0,1\}}$, welche festlegt, wieviele Marken mindestens von den Stellen abgezogen werden und auf die Stellen gelegt werden müssen, und den linear unabhängigen (und endlich vielen) *Periodenmultimengen*

$$P \subseteq \{p \in \mathbb{N}^{S_T \times \{0,1\}} \mid p(0) = p(1)\},$$

welche beschreiben, wieviele Marken zusätzlich abgezogen und wieder aufgelegt werden können. (Bei einer Elementartransition muß dieser Zusatz jeweils gleich sein, damit das Schaltverhalten nur durch c bestimmt ist. Prinzipiell genügt für eine Elementartransition auch $P = \emptyset$.)

(2) Das Schaltverhalten einer Elementartransition:
Die Multimengen $m_0, m_1 \in \mathbb{N}^{S_T}$ seien Markierungen; es sei $c \notin P$ (man denke sich c als markiert). Die Elementartransition t kann mittels eines *Pfades* $f \in \mathbb{N}^{\{c\} \cup P}$ mit $f(c) = 1$ von der Markierung m_0 zu der Markierung m_1 schalten, d.h. $m_0[t\rangle m_1$ mittels f, wenn $p := (\{c\} \cup P) \cdot f = c + P \cdot f$ und $m_0 - p(0) = m_1 - p(1) \in \mathbb{N}^{S_T}$. (Es scheint hier überflüssig zu sein, daß die Markierungen vor und nach dem Schalten sowohl durch einen größeren Pfad f als auch durch eine größere Differenz $m_0 - p(0)$

erhöht werden können, doch wir werden im Beispiel 3 Seiten weiter sehen, daß im allgemeinen Fall beides separat möglich sein muß.)
Der Pfad f bescheibt also mit Hilfe der in Unterkapitel 2.6 beschriebenen Multiplikation, wie oft die Transition schaltet (in diesem Fall gerade $f(c) = 1$ mal) und von jeder Periode, wie oft sie hinzukommt.

(3) Schaltfolgen:
Für ein Netz (S, T) mit $S = S_T$ ist $Q := \bigcup\limits_{(K,c,P)\in T} \{c\} \cup P$ die disjunkte Vereinigung aller Mengen von Konstanten- und Periodenmultimengen, die die Transitionen in T beschreiben (man denke sich die Multimengen in $\{c\} \cup P$ mit $t = (K, c, P) \in T$ mit t markiert). Eine Transitionsfolge $w = t_1...t_n \in T^*$ kann nun mittels eines Pfades $f \in \mathbb{N}^Q$ von der Markierung $m_0 \in \mathbb{N}^S$ zu der Markierung $m_n \in \mathbb{N}^S$ schalten, d.h. $m_0[w\rangle m_n$ mittels f, wenn der Pfad in einzelne Schaltvorgänge zerlegt werden kann, d.h. wenn

$$\exists m_1, ...m_{n-1} \in \mathbb{N}^S, f_1, ...f_n \in \mathbb{N}^Q \; f = \sum_{i=1}^{n} f_i \wedge \forall 1 \leq i \leq n \; m_{i-1}[t_i\rangle m_i \text{ mittels } f_i.$$

(Dies gilt ungeachtet dessen, ob die t_i Elementartransitionen oder 'komplexe' Transitionen sind.)

(4) Im allgemeinen Fall hat eine 'komplexe' Transition n Netze (S_i, T_i) mit mindestens so vielen Stellen wie außen, d.h.

$$\forall 1 \leq i \leq n \;\; S_{T_i} = S_i \supseteq S_T,$$

als Unternetze. Diese Transition hat die Form $t = ([(S_1, T_1)...(S_n, T_n)], c, P)$ mit

$$\{c\} \cup P \subseteq \mathbb{N}^{S_T \times \{0,1\} \cup \bigcup_{i=1}^{n} (S_i \times \{2i, 2i+1\} \cup Q_i \times \{-i\})}$$

mit $Q_i := \bigcup\limits_{(K',c',P')\in T_i} \{c'\} \cup P'$ und

$$\forall p \in P \;\; p(0) = p(2) \wedge p(2n+1) = p(1) \wedge \forall 2 \leq i \leq n \;\; p(2i-1) = p(2i).$$

(Wegen $p(2i) \in \mathbb{N}^{S_i}$ und $p(2i-1) \in \mathbb{N}^{S_{i-1}}$ für $2 \leq i \leq n$ bedeutet $p(2i-1) = p(2i)$, daß der Wert $p(s, 2i-1) = p(s, 2i) \in \mathbb{N}^{S_{i-1} \cap S_i}$ liegt, also für $s \in S_{i-1} \setminus S_i \cup S_i \setminus S_{i-1}$ gleich 0 ist.) Eine Multimenge in $c + P^*$ beschreibt somit, wieviele Marken die Transition insgesamt zu Beginn ihres Schaltens abzieht (markiert mit 0), wieviele sie zu Beendigung ihres Schaltens auflegt (markiert mit 1) und für jedes *Unternetz* (S_i, T_i), wieviele Marken zu Beginn einer Schaltfolge des Unternetzes zur Verfügung gestellt werden (markiert mit $2i$), wieviele Marken genau beim Ende einer Schaltfolge des Unternetzes wieder vorhanden sein müssen (markiert mit $2i + 1$) und den Pfad, der die Schaltfolge des Unternetzes quantitativ beschreibt (markiert mit $-i$). Dabei darf wiederum die Differenz der Marken am Ende eines

Unternetzes zu den Marken am Anfang des folgenden Unternetzes nicht durch Erhöhen einer Periode verändert werden; dies wird durch $p(2i-1) = p(2i)$ ausgedrückt. Gleiches gilt für die Differenz der Marken, welche die Transition insgesamt zu Beginn ihres Schaltens abzieht, zu den Marken am Anfang des ersten Unternetzes ($p(0) = p(2)$) und die Differenz der Marken, welche die Transition insgesamt zu Beendigung ihres Schaltens auflegt, zu den Marken am Ende des letzten Unternetzes ($p(2n+1) = p(1)$).
Die Periodenmultimengen in P sind linear unabhängig, somit ist die Darstellung eines $p \in c + P^*$ eindeutig und f mit $p = (\{c\} \cup P) \cdot f = c + P \cdot f$ ist eindeutig festgelegt.

(5) Die Transition t kann mittels eines Pfades $f \in \mathbb{N}^{\{c\} \cup P}$ mit $f(c) = 1$ von der Markierung m_0 zu der Markierung m_1 schalten, d.h. $m_0[t\rangle m_1$ mittels f, wenn $p := (\{c\} \cup P) \cdot f = c + P \cdot f$, $m_0 - p(0) = m_1 - p(1) \in \mathbb{N}^{S_T}$ und

$$\forall 1 \leq i \leq n \ \exists w_i \in T_i^* \ p(2i)[w_i\rangle p(2i+1) \text{ mittels } p(-i).$$

Die Menge $c + P^*$ dient sowohl der Restriktion von möglichen Schaltfolgen in Unternetzen, als auch der Spezifikation der (nicht explizit genannten) Zwischenübergänge von $\mathbb{N}^{S_T}$ nach $\mathbb{N}^{S_1}$, von $\mathbb{N}^{S_i}$ nach $\mathbb{N}^{S_{i+1}}$ und von $\mathbb{N}^{S_n}$ nach $\mathbb{N}^{S_T}$. $p(-i)$ ist die Summe der Pfade im i-ten Unternetz; f beschreibt also durch rekursive Anwendung der in Unterkapitel 2.6 beschriebenen Multiplikation die Summe der Pfade in jeder Unternetzebene.
Da gleichzeitiges Vergrößern von m_0 und m_1 die Differenz $m_0 - p(0) = m_1 - p(1)$ positiv läßt und die innere Schaltbedingung nicht verändert, gilt die Monotonie.

(6) Kommt eine Stelle vor und nach einem solchen Zwischenübergang in t vor, so ist sie dort als *frei* zu bezeichnen: $\forall 0 < i < n \ F_{2i+1} := F_{2i+2} := S_i \cap S_{i+1}$. Die Menge der freien Stellen F_2 am Anfang des ersten Unternetzes und und F_{2n+1} am Ende des letzten Unternetzes sind, falls sie nicht anders festgelegt werden, definiert durch $F_2 := S_T \cap S_1$ und $F_{2n+1} := S_n \cap S_T$. Stellen in $\overline{F_{2i}} := S_i \setminus F_{2i}$ bzw. $\overline{F_{2i+1}} := S_i \setminus F_{2i+1}$ sind dagegen fest spezifiziert.

(7) Die *erweiterte Erreichbarkeitsmenge* für eine Transition t ist

$$ER_t := \{(m_0, m_1, f) \mid m_0[t\rangle m_1 \text{ mittels } f\}.$$

Die *Erreichbarkeitsmenge* $R_t := \{(m_0, m_1) \mid \exists f \ m_0[t\rangle m_1$ mittels $f\}$ und die Pfadmenge

$$E_t := \{f \mid \exists m_0, m_1 \ m_0[t\rangle m_1 \text{ mittels } f\}$$

sind Projektionen von ER_t. Für eine Menge T gilt entsprechend $ER_T := \bigcup_{t \in T} ER_t$ analog dazu E_T, R_T und für ein Netz (S, T) ist entsprechend [Hau90]

$$ER_{(S,T)} := \{(m_0, m_1, f) \mid \exists w \in T^* \ m_0[w\rangle m_1 \text{ mittels } f\}$$

und $E_{(S,T)} := \{f \mid \exists w \in T^* \; \exists m_0, m_1 \; m_0[w\rangle m_1 \text{ mittels } f\} \supseteq E_T$.
Es gilt also $ER_{(K,c,P)} =$

$$\{(m_0, m_1, f) \mid \quad f(c) = 1 \text{ und für } p \text{ mit } p = (\{c\} \cup P) \cdot f = c + P \cdot f \text{ gilt} \\ m_0 - p(0) = m_1 - p(1) \geq \emptyset \wedge \\ \forall (S_i, T_i) \in K \; (p(2i), p(2i+1), p(-i)) \in ER_{(S_i,T_i)} \; \}$$

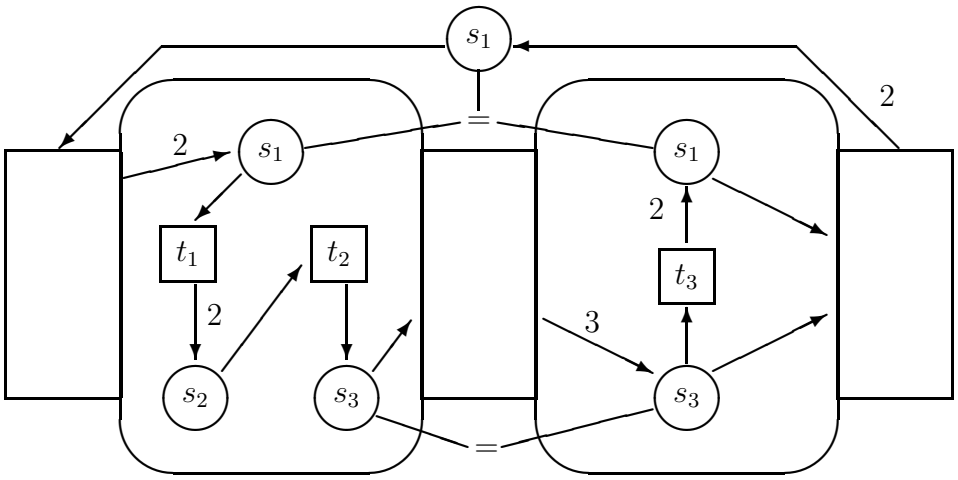
und $E_{(K,c,P)} =$

$$\{f \mid \quad f(c) = 1 \text{ und für } p \text{ mit } p = (\{c\} \cup P) \cdot f = c + P \cdot f \text{ gilt} \\ \forall (S_i, T_i) \in K \; (p(2i), p(2i+1), p(-i)) \in ER_{(S_i,T_i)} \; \}.$$

(8) Zwei Transitionsmengen T, T' heißen *erreichbarkeitsäquivalent*, wenn $R_T = R_{T'}$.

(9) Das *Schaltbarkeitsproblem* für eine Transitionenmenge T ist zu entscheiden, ob R_T nicht leer ist.

Beispiel:
Sei $S_T = \{s_1\}$ und $T = \{t\}$ bestehend aus der in folgendem Bild mit s_1 dargestellten Transition $t = ([(\{s_1, s_2, s_3\}, \{t_1, t_2\})(\{s_1, s_3\}, \{t_3\})], c, P)$ spezifiziet durch $c(0) = (1), c(1) = (2), c(2) = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}, c(3) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, c(4) = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, c(5) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ und $c(c_1, -1) = c(c_2, -1) = c(c_3, -2) = c(p_{10}, -2) = c(p_{11}, -2) = 0$. Die Transitionen in den Unternetzen $t_1 = ([\,], c_1, \{\})$, $t_2 = ([\,], c_2, \{\})$, $t_3 = ([\,], c_3, \{p_{10}, p_{11}\})$ sind spezifiziert durch $c_1 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 0 & 0 \end{pmatrix}, c_2 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$ und $c_3 = \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}$.



Deren Periodenvektoren $p_{10} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, p_{11} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$ von t_3 und die Periodenvektoren $P = \{p_1 ... p_9\}$ von t mit
$p_1 = \{(s_1, 0) \mapsto 1, (s_1, 2) \mapsto 1\}, p_2 = \{(s_1, 5) \mapsto 1, (s_1, 1) \mapsto 1\}$,
$p_3 = \{(s_1, 3) \mapsto 1, (s_1, 4) \mapsto 1\}, p_4 = \{(s_3, 3) \mapsto 1, (s_3, 4) \mapsto 1\}$,
$p_5 = \{(c_1, -1) \mapsto 1\}, p_6 = \{(c_2, -1) \mapsto 1\}, p_7 = \{(c_3, -2) \mapsto 1\}$,
$p_8 = \{(p_{10}, -2) \mapsto 1\}, p_9 = \{(p_{11}, -2) \mapsto 1\}$
gehen nicht in die bildliche Darstellung ein und sind bei diesem Beispiel so gewählt, daß die Schaltfolgen nicht durch sie restringiert werden.
Ein Schaltvorgang wird in der Reihenfolge von links nach rechts interpretiert.
Es gilt z.B. $(1)[t\rangle(8)$ mittels $\{c \mapsto 1, p_1 \mapsto 0, p_2 \mapsto 6, p_3 \mapsto 1, p_4 \mapsto 1, p_5 \mapsto 1, p_6 \mapsto 2, p_7 \mapsto 3, p_8 \mapsto 0, p_9 \mapsto 0\} = (1, 0, 6, 1, 1, 1, 2, 3, 0, 0)$ mit der folgenden Interpretation:

> Zu Beginn zieht t eine Marke von s_1 ab und stellt 2 Marken auf s_1 im ersten Unternetz zur Verfügung. Einmaliges Schalten ($p_5 \mapsto 1$) von t_1 zieht eine dieser Marken ab und legt 2 Marken auf s_2. Zweimaliges Schalten von t_2 ($p_6 \mapsto 1$) bewegt diese beiden Marken nach s_3, wovon eine gefordert wird und die andere zusätzlich ist ($p_4 \mapsto 1$). Die übrige Marke auf s_1 ist ebenfalls zusätzlich ($p_3 \mapsto 1$). (Auf s_2 darf ohnehin keine zusätliche Marke bleiben.) Im zweiten Unternetz stehen nun 4 Marken auf s_3 zu Verfügung (eine davon ist zusätzlich). Da genau eine dieser Marken übrig bleiben muß, schaltet t_3 drei mal ($p_7 \mapsto 3$) und legt dabei 6 Marken zu der einen hinzu. Zuletzt legt t 8 Marken (2 obligatorische und 6 zusätzliche wegen $p_2 \mapsto 6$) auf s_1 ab.

Der Schaltvorgang $(1)[t\rangle(8)$ kann aber auch mittels $(1, 0, 6, 1, 1, 1, 2, 3, 9, 5)$ stattfinden, wobei lediglich zwischendurch mehr abgezogen und wieder aufgelegt wird, was sich wieder ausgleicht.
Auch kann $(2)[t\rangle(9)$ mittels $(1, 0, 6, 1, 1, 1, 2, 3, 0, 0)$ oder auch mittels $(1, 1, 7, 2, 1, 1, 2, 3, 12, 5)$, hierbei ist die Erhöhung der Perioden p_1, p_2, p_3, p_8 und p_9 freiwillig. $(2)[t\rangle(15)$ kann mittels $(1, 1, 13, 0, 5, 3, 6, 7, 0, 0)$ schalten, hierbei ist die Erhöhung der Perioden p_1 und p_2 erzwungen. Durch zweimaliges Schalten könnte z.B. auch $(1)[tt\rangle(15)$ mittels $(2, 0, 12, 2, 2, 2, 4, 6, 25, 10)$ gelten.
Wir verwenden die folgenden Ergebnisse aus [Rei94]:

Theorem 2.1 *Zu jeder Transitionenmenge T kann effektiv ein erreichbarkeitsäquivalentes T' konstruiert werden mit der Eigenschaft, daß für alle Transitionen $t = (K, c, P) \in T'$ die folgenden 6 Bedingungen gelten:*

1. *Für alle $(S_i, T_i) \in K$ hat T_i selbige Eigenschaft, d.h. die Eigenschaft ist rekursiv über der Schachtelungsstruktur.*

2. *$\forall (S_i, T_i) \in K \ \forall p \in \{c\} \cup P \ p(2i) - (Q_i \cdot (p(-i)))(0) = p(2i + 1) - (Q_i \cdot (p(-i)))(1)$, d.h. für jeden Periodenvektor stimmt in jedem Unternetz die*

Differenz der Markierungen an Anfang und Ende mit der Differenz der Markierungen, die durch den Pfad im Unternetz entstehen, überein.

3. $\forall (S_i, T_i) \in K \ \forall s \in F_{2i} \ \sum_{p \in P} p(s, 2i) \geq 1 \wedge \forall s \in F_{2i+1} \ \sum_{p \in P} p(s, 2i+1) \geq 1$, *d.h. freie Stellen können beliebig erhöht werden.*

4. $\forall (S_i, T_i) \in K \ \forall q \in Q_i \ \sum_{p \in P} p(q, -i) \geq 1$, *d.h. alle Durchläufe durch Transitionen und alle Perioden in Unternetzen können beliebig oft verwendet werden.*

5. $\forall (S_i, T_i) \in K \ \exists (x_i, x_i + \Delta_i, \alpha_i), (x'_i + \Delta'_i, x'_i, \alpha'_i) \in ER_{(S_i, T_i)}$ *mit*
$$\forall s \in \overline{F_{2i}} \ \Delta_i(s) \geq 1 \wedge x_i(s) \leq c(s, 2i) \wedge$$
$$\forall s \in \overline{F_{2i+1}} \ \Delta'_i(s) \geq 1 \wedge x'_i(s) \leq c(s, 2i+1),$$
d.h. falls die freien Stellen genügend hoch sind, so gibt es eine Schaltfolge, bei der die nicht freien Stellen erhöht, bzw. verkleinert werden.

6. $\{c \mapsto 1\} \in E_t$, *d.h. c selbst beschreibt schon eine erlaubte Schaltfolge. (Darüber hinaus gibt es gemäß Lemma 2.1 beliebig hohe Pfade in t, d.h. es gilt* $\forall g \in \mathbb{N}^P \ \exists f' \in \mathbb{N}^P \ f' \geq g \ \wedge \ \{c \mapsto 1\} + f' \in E_t$.*)*

Korollar 2.1 *[Rei94] Das Schaltbarkeitsproblem für geschachtelte Petrinetze ist entscheidbar.*

Lemma 2.1 *[Rei94] Unter der Voraussetzung, daß die Bedingungen 1 bis 5 für* $t = (K, c, P)$ *gelten, gilt:*

$$\forall f \in \mathbb{N}_+^P, g \in \mathbb{Z}^P \exists k \ \geq 2 \ \{c \mapsto 1\} + kf, \{c \mapsto 1\} + kf + g \in E_t$$

Die Bedingung sagt, daß jede Pfaddifferenz g durch eine andere Pfaddifferenz f ausgeglichen werden kann, wenn man diese nur oft genug anwendet und dabei alle Perioden beliebig erhöhen kann.

Wenn die inhibitorischen Kanten so liegen, daß sich die Stellen so anordnen lassen, daß jede Stelle zu all denjenigen Transitionen eine inhibitorische Kante hat, zu denen eine bezüglich der Anordnung größere Stelle eine inhibitorische Kante hat, dann kann dazu ein Menge von geschachtelten Transitionen T' konstruiert werden für die $R_{T'}$ genau dann nicht leer ist, wenn ein $w \in T^*$ existiert mit $m_0[w\rangle m_e$. Damit kann das Erreichbarkeitsproblem für (S, T, W, I, m_0, m_e) mit Hilfe von Korollar 2.1 entscheiden werden.

Theorem 2.2 *[Rei94] Gilt für ein Petrinetz* (S, T, W, I, m_0, m_e) *die Bedingung*

$$\exists g \in \mathbb{N}_+^S \ \forall s, s' \in S \ g(s) \leq g(s') \rightarrow (\forall t \in T \ (s', t) \in I \rightarrow (s, t) \in I),$$

so ist das Erreichbarkeitsproblem für (S, T, W, I, m_0, m_e) *entscheidbar.*

3 Prioritätsmulticounterautomaten

In diesem Kapitel wird beschrieben, für welche Art von Counterautomaten entscheidbar ist, ob die von einem gegeben Counterautomaten akzeptierte Sprache leer ist.
Bekannt ist die Entscheidbarkeit der Erreichbarkeit einer akzeptierenden Konfiguration, falls die Leerheit bei keinem Zähler unterschieden werden kann (man bezeichnet diese als *schwache Zähler*). In diesem Fall ist sie äquivalent zum Erreichbarkeitsproblem in Petrinetzen bzw. in Vektoradditionssystemen ([May84], [Kos84]).
Man kann leicht sehen, daß das Ereichbarkeitsproblem unentscheidbar wird, wenn man die Leerheit für zwei Zähler testen kann, da nun ein wiederholtes, kontrolliertes 'Umschaufeln' zwischen diesen beiden Zählern möglich wird. Mit dieser Methode kann eine Turingmaschine simuliert und damit das Halteproblem auf das Erreichbarkeitsproblem reduziert werden (siehe [Min71]).
Multicountersprachen (Sprachen die von Multicounterautomaten erkannt werden) mit keinem oder nur einem Nulltest bilden keine AFL (abstrakte Familie von Sprachen), da sie nicht unter Sternbildung abgeschlossen sind. Mit der folgenden Verallgemeinerung wird jedoch eine AFL gebildet:
Außer dem Nulltest von Zähler 1 kann auch, falls Zähler 1 Null ist, Zähler 2 auf Leerheit getestet werden oder allgemein, falls die Zähler 1 bis $k-1$ den Wert Null haben, kann Zähler k auf Leerheit getestet werden. Kontrolliertes 'Umschaufeln' wird dabei nur in einer Richtung möglich. Formal läßt sich dies wie folgt fassen:

Definition
Ein *Prioritätsmulticounterautomat* ist ein Einwegautomat beschrieben durch ein 6-Tupel

$$A = (k, Z, \Sigma, \delta, z_0, E)$$

mit Zustandsmenge Z, Eingabealphabet Σ, Übergangsrelation

$$\delta \subseteq (Z \times (\Sigma \cup \{\lambda\}) \times \{0 \dots k\}) \times (Z \times \{-1, 0, 1\}^k),$$

Anfangszustand z_0, akzeptierende Zustände $E \subseteq Z$, Konfigurationenmenge $C_A = Z \times \Sigma^* \times \mathbb{N}^k$, Anfangskonfiguration $\sigma_A(x) = \langle z_0, x, 0^k \rangle$ und Konfigurationsübergangsrelation

$$\langle z, ax, n_1, ..., n_k \rangle \ \vdash_A \ \langle z', x, n_1 + i_1, ..., n_k + i_k \rangle$$

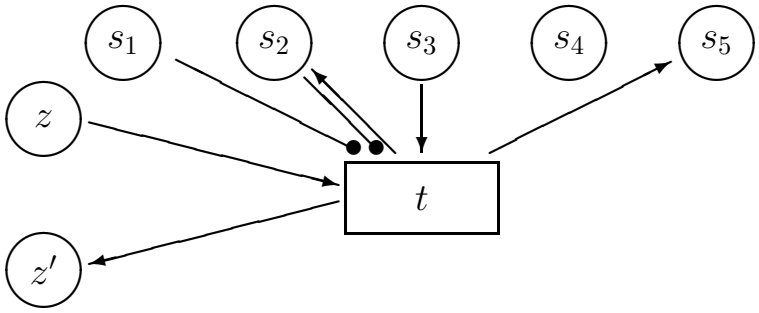
gdw. $z, z' \in Z, a \in \Sigma \cup \{\lambda\}, \langle (z, a, j), (z', i_1, ...i_k) \rangle \in \delta, \ \ \forall i \leq j \ n_i = 0$ (und $n_{j+1} > 0$ oder $j = k$)[2].

[2]Automaten gemäß der Definition mit bzw. ohne diesen Zusatz lassen sich leicht ineinander überführen

Bemerkung: Die von dem Prioritätsmulticounterautomaten A erkannte Sprache ist $L(A) = \{w \mid \exists z_e \in Z\ \exists n_1, ..., n_k \in \mathbb{N}\ \langle z_0, w, 0, ..., 0\rangle \ \vdash_A^* \ \langle z_e, \lambda, n_1, ..., n_k\rangle$. Ein Prioritätsmulticounterautomat kann so umgewandelt werden, daß er nur einen akzeptierenden Zustand z_e besitzt und vor dem Akzeptieren alle Zähler löscht. Dies geschieht dadurch, indem man man von jedem der bisherigen Endzustände einen λ-Übergang in einen neuen Zwischenzustand macht, in welchem die Zähler in einer λ-Schleife dekrementiert werden und mit einem weiteren Nulltest (mit $j = k$) bei einem weiteren λ-Übergang in den neuen Endzustand z_e geht. Wir können dann die von dem Prioritätsmulticounterautomaten A erkannte Sprache als $L(A) = \{w \mid \langle z_0, w, 0, ..., 0\rangle \ \vdash_A^* \ \langle z_e, \lambda, 0, ..., 0\rangle\}$ betrachten.

Theorem 3.1 *Für einen Prioritätsmulticounterautomaten ist entscheidbar, ob die erkannte Sprache leer ist.*

Beweis: Zu einem Prioritätsmulticounterautomat mit einem Zustand kann ein Petrinetz (S, T, W, I, m_0, m_1) konstruiert werden mit den Stellen $S := \{1...k\} \cup Z$, den Transitionen $T = \delta$, den Gewichten W mit
$W(z, ((z', a, j), (z'', V))) := 1$ falls $z = z'$ und $:= 0$ sonst,
$W(((z', a, j), (z'', V)), z) := 1$ falls $z = z''$ und $:= 0$ sonst,
$W(i, ((z', a, j), (z'', V))) := 1$ falls $V(i) = -1$ und $:= 0$ sonst und
$W(((z', a, j), (z'', V)), i) := 1$ falls $V(i) = 1$ und $:= 0$ sonst,
den inhibitorischen Kanten $I := \{(i, ((z', a, j), (z'', V))) \mid i \leq j\}$, der Anfangsmarkierung $m_0 := \{z_0 \mapsto 1\}$ und der Endmarkierung $m_1 := \{z_e \mapsto 1\}$, in welchem genau dann m_1 von m_0 erreichbar ist, wenn die von dem Prioritätsmulticounterautomaten akzeptierte Sprache nicht leer ist. Aufgrund von Theorem 2.2 mit $g(i) = i$ für $i \leq k$ und $g(z) = k + 1$ für $z \in Z$ ist dies entscheidbar.
Beispiel: Sei $k = 5$ und die Transition $t = ((z, a, 2), (z', \langle 0, 1, -1, 0, 1\rangle)) \in \delta$, so sieht das betreffende Teilnetz wie folgt aus:



■

Eine bestimmte Eingabe w kann in die Zustände eines Prioritätsmulticounterautomat kodiert werden, welcher die $|w|$-fache Anzahl von Zuständen besitzt und mit λ-Übergängen den ursprünglichen Prioritätsmulticounterautomat auf w simuliert. Die akzeptierte Sprache ist $\{\lambda\}$ (und nicht $\{\}$) gdw. der ursprünglichen Prioritätsmulticounterautomat w akzeptiert; somit folgt:

Korollar 3.1 *Für einen Prioritätsmulticounterautomaten A ist das Halteproblem (d.h. die Frage $\langle z_0, \lambda, 0, ..., 0\rangle \vdash_A^* \langle z_e, \lambda, 0, ..., 0\rangle$) entscheidbar.*

3.1 Multicounter- und Prioritätsmulticountersprachen

In [Gon93] definiert D. Gonzalez die Klasse MC der *Multicountersprachen* als kleinste Familie, welche unter Schnitt und rationaler Transduktion abgeschlossen ist und die Semi-Dyck-Sprache $D_1'^*$ enthält. Die Sprachen in MC können gerade von einem Multicounterautomaten, der keinen Zähler auf Leerheit testen kann, erkannt werden (siehe [Gre78] Seite 316). Wegen z.B. $\{a^n b^n \mid n \geq 0\}^* \notin MC$ laut [Gre78] ist MC nicht unter Sternbildung abgeschlossen.

Definition Die Klasse PMC der Prioritätsmulticountersprachen ist der Abschluß (von $\{\Sigma^*\}$) unter rationaler Transduktion, Sternbildung und Schnitt mit der Semi-Dyck-Sprache $D_1'^*$.

PMC ist eine AFL. Die Sprachen in PMC können gerade von einem Prioritätsmulticounterautomaten erkannt werden.
Es ist leicht zu sehen, daß ein Counterautomat, der bei 2 Zählern 0 testen kann, die Sprache $\{(a^n b)^m \mid n, m \in \mathbb{N}\}$ erkennen kann, ein Prioritätsmulticounterautomat ist aber schwächer:

Theorem 3.2 *Die Sprache $\{(a^n b)^m \mid n, m \in \mathbb{N}\}$ kann nicht von einem Prioritätsmulticounterautomaten erkannt werden.*

Beweis: Angenommen ein Prioritätsmulticounterautomat A mit k Zählern könnte die Sprache $\{(a^n b)^m \mid n, m \in \mathbb{N}\}$ erkennen, so könnte ein Multicounterautomat M mit m Zählern und beliebigem Nulltest von einem Prioritätsmulticounterautomaten P, der $k+2m$ Zähler besitzt, mit der folgenden Methode simuliert werden, wobei die ersten k Zähler den Zählern von A entsprechen und zwei der weiteren Zähler jeweils einem Zähler von M entsprechen:
P rät zuerst in einer λ-Schleife den maximalen Zählerinhalt n, den ein Zähler von M während der simulierten Rechnung haben wird. Dabei beginnt P den Automaten A auf $a^n b$ zu simulieren und gleichzeitig wird für jeden Zähler von M der 'Gegenzähler' (d.h. einer der beiden entsprechenden Zähler) mit n gefüllt. Nun simuliert P den Automaten M, wobei jeder Gegenzähler immer in entgegengesetzter Richtung verändert wird, d.h. wird z.B. der z-te Zähler mit $z \leq m$ inkrementiert, so wird der entsprechende $k+z$-te Zähler ebenfalls inkrementiert und der entsprechende 'Gegenzähler' (mit der Nummer $k+m+z$) dekrementiert. Wenn M einen Nulltest ausführt, so prüft P, ob der entsprechende Gegenzähler n hat, indem er, während er diesen n-fach dekrementiert, A auf $a^n b$ weitersimuliert und anschließend, während er den Gegenzähler wieder n-fach inkrementiert, A nochmals auf $a^n b$ weitersimuliert. Die Simulation von A kann nur akzeptieren,

wenn n jedesmal richtig geraten wird. Somit ist sichergestellt daß vor und nach dem Nulltest der Inhalt des Gegenzählers n und der Zähler selbst daher 0 ist.
P akzeptiert, wenn die Simulationen von M und A akzeptieren.
Wegen Theorem 3.1 könnte somit die von M erkannte Sprache auf Leerheit überprüft werden, was ein Widerspruch ist. ∎
Zu einer Vielzahl von Klassen existiert gemäß [Ber79] und [Gre78] (siehe auch Diagramm auf der übernächsten Seite) eine Sprache, die vollständig bezüglich rationaler Transduktion in dieser Klasse ist. Dabei entspricht jedes Zeichen des Alphabetes einer Operation auf dem Speicher der Automaten, die die Sprachklasse erkennen können, so entsprechen z.B. die öffnenden- und schließenden Klammern der Dyck-Sprachen den zugehörigen *push*- und *pop*-Operationen der Kellerautomaten. Die Wörter der Sprache entsprechen den korrekten Protokollen des Speichers (siehe auch Kellerprotokolle in [Rei89]).
Die Sprache kann von einem Automaten erkannt werden, der diese Operationen direkt ausführt. Umgekehrt kann jede andere Sprache der Klasse reduziert werden mittels eines Transducers, der den dazugehörigen Automaten simuliert und die ausgeführten Operationen auf das Ausgabeband schreibt. Die Simulation entspricht genau dann einer Rechnung des Automaten, wenn das ausgegebene Wort in der vollständigen Sprache liegt. Für k-PMC lautet die Sprache wie folgt:
Sei $PD_0 := \{\lambda\}$, $\Sigma_k := \{a_1, b_1, c_1, ..., a_k, b_k, c_k\}$ und $PD_k :=$

$$(\{w \in (\Sigma_k \setminus \{c_k\})^* \mid \Pi_{\Sigma_{k-1}}(w) \in PD_{k-1} \ \wedge \\ |w|_{a_k} = |w|_{b_k} \ \wedge \ \forall uv = w \ |u|_{a_k} \geq |u|_{b_k}\}c_k)^*.$$

Die Sprache PD_k ist vollständig bezüglich rationaler Transduktion in der Klasse k-PMC der von einem Prioritätsmulticounterautomaten mit k Zählern erkannten Sprachen. Hierbei entspricht a_i dem Inkrementieren des Zählers i, b_i dem Dekrementieren des Zählers i und c_i dem Nulltest des Zählers i und allen Zählern $l \leq i$, da aus $wc_iv \in PD_k$ folgt daß $\forall l \leq i \ |w|_{a_l} = |w|_{b_l}$ gilt.

3.2 Die Inklusionsstruktur der Sprachklassen

Theorem 3.3 *Für $k > 0$ sind die Klassen k-PMC und LIN unvergleichbar und für $k > 1$ sind die Klassen k-PMC und CFL unvergleichbar*

Beweis: '$\not\subseteq$': Die Sprachen $\{a^nb^n \mid n \in \mathbb{N}\}^*$ bzw. $\{a^nb^nc^n \mid n \in \mathbb{N}\}$ sind in 1-PMC bzw. 2-PMC aber nicht linear bzw. kontextfrei.
'$\not\supseteq$': Die Sprache $\{w\$w^R \mid w \in \{a, b\}^*\}$ ist linear. Könnte ein Prioritätsmulticounterautomat mit k Zählern die Sprache erkennen, so könnte ein solcher auch die Sprache

$$\{a^{n_1}ba^{n_2}...ba^{n_{k+1}}\$a^{n_{k+1}}b...a^{n_2}ba^{n_1} \mid \forall i \leq k+1 \ n_i \in \mathbb{N}\} = \\ \{w\$w^R \mid w \in \{a, b\}^*\} \cap \{w \in \{a, b, \$\}^* \mid |w|_b = 2k\}$$

erkennen, was nach folgendem Lemma 3.1 nicht möglich ist. ∎

Lemma 3.1 $\{a^{n_1}ba^{n_2}...ba^{n_{k+1}}\$a^{n_{k+1}}b...a^{n_2}ba^{n_1} \mid \forall i \leq k+1\ n_i \in \mathbb{N}\} \notin k\text{-}PMC.$

Beweis: Eine Prioritätsmulticounterautomat A mit k Zählern, der die Sprache $\{a^{n_1}ba^{n_2}...ba^{n_{k+1}}\$a^{n_{k+1}}b...a^{n_2}ba^{n_1} \mid \forall i \leq k+1\ n_i \in \mathbb{N}\}$ erkennt, kann gemäß Theorem 3.1, Theorem 2.2 und Theorem 2.1 umgewandelt werden in ein geschachteltes Petrinetz $(\emptyset, T)$, das für alle $t \in T$ die Bedingungen 1-6 erfüllt. Da die Anzahl $2k$ der Transitionen, die dem Lesen eines b's entsprechen, beschränkt ist (gleiches gilt für \$), müssen sich die Transitionen, die vor und nach einem solchen Schaltvorgang schalten, in verschiedenen Unternetzen schalten. Aus diesem Grund beschreibt jeder Schaltpfad $f \in E_T$ u.a. eindeutig ein Wort der Sprache und die Konfiguration von A bei Lesen von \$, d.h. es existieren Homomorhpismen $h_a : \mathbb{N}^Q \mapsto \mathbb{N}^{k+1}$ und $h_k : \mathbb{N}^Q \mapsto \mathbb{N}^k$ mit $Q := \bigcup_{(K,c,P)\in T} \{c\} \cup P$, wobei für alle $f \in E_T$ in dem durch f beschriebenen Wort $h_a(f)(i) = n_i$ für alle $i \leq k+1$ gilt und der j-te Zähler beim lesen von \$ den Wert $h_k(f)(j)$ hat.
Um alle Wörter der Sprache zu erreichen, muß ein $t = (K, c, P) \in T$ existieren, mit den Periodenmultimengen $p_1...p_{k+1} \in P$, wobei

$$f_a(\{p_1 \mapsto 1\}), ..., f_a(\{p_{k+1} \mapsto 1\})$$

linear unabhängig sind. Andererseits müssen aber

$$f_k(\{p_1 \mapsto 1\}), ..., f_k(\{p_{k+1} \mapsto 1\})$$

linear abhängig sein und somit existiert ein $g \in \mathbb{Z}^{\{p_1,...p_{k+1}\}}$ mit $f_k(g) = \overline{0}$ und $f_a(g) \neq \overline{0}$. Nach Lemma 2.1 folgt nun für $f = \{1\}^P$ die Existenz eines k mit

$$\{c \mapsto 1\} + kf, \{c \mapsto 1\} + kf + g \in E_t.$$

Diese Pfade beschreiben zwei verschiedene Wörter der Sprache, bei denen A beim Lesen von \$ in der gleichen Konfiguration ist. Durch falsches Zusammensetzen der Worthälften erhält man damit ein Wort, das der Prioritätsmulticounterautomat A fälschlicherweise akzeptiert. ∎

Da die Sprache $\{a^{n_1}ba^{n_2}...ba^{n_{k+1}}\$a^{n_{k+1}}b...a^{n_2}ba^{n_1} \mid \forall i \leq k+1\ n_i \in \mathbb{N}\}$ leicht von einem Zählerautomaten mit $k+1$ blinden Zählern erkannt werden kann, folgt:

Korollar 3.2 *Die Hierarchien k-BLIND und k-PBLIND gemäß [Gre78] und die Hierarchie k-PMC sind echt.*

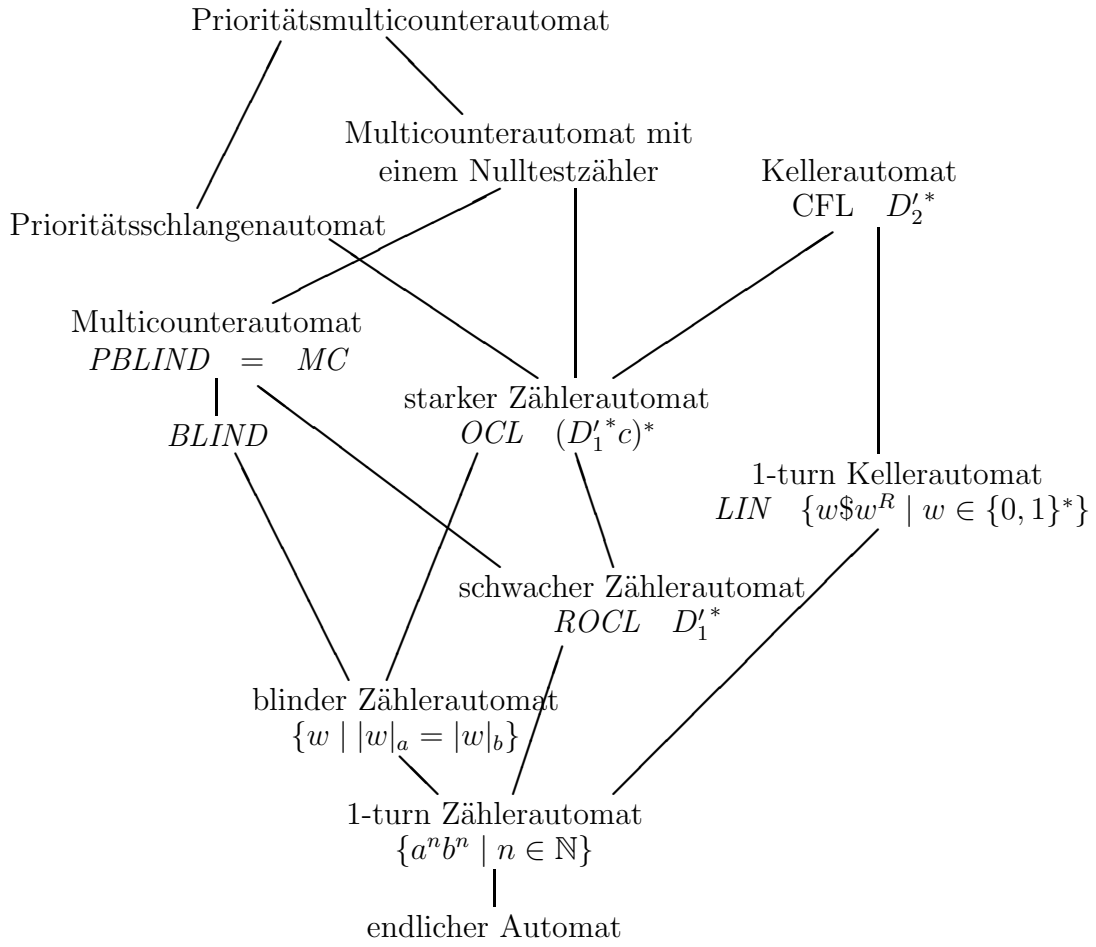
Bemerkung: Sei $\Sigma_l := \{a_0, b_0, c_0, ..., a_l, b_l, c_l\}$ und $LD_{k,l} :=$

$$\{w \in \Sigma_l^* \mid \Pi_{\Sigma_k}(w) \in PD_k \ \wedge\ \forall i \leq l\ |w|_{a_i} = |w|_{b_i} \ \wedge\ \forall uv = w\ |u|_{a_i} \geq |u|_{b_i}\}.$$

Die Sprache $LD_{k,l}$ kann von einem Prioritätsmulticounterautomaten, welcher überhaupt nur bei den ersten k Zählern einen Nulltest machen darf (aber weitere $l-k$ Zähler haben kann), erkannt werden. Vermutlich ergibt sich dabei mit

dieser Einschränkung eine echte Hierarchie. Es ist aber nicht einmal bekannt, ob ein Prioritätsmulticounterautomat Sprachen erkennen kann, die kein Multicounterautomat, der bei genau einem seiner Zähler Null testen kann, erkennen kann.

Ein Überblick über Sprachklassen, bei denen das Leerheitsproblem entscheidbar ist, deren Automaten und die dazugehörigen, unter rationaler Transduktion vollständigen Sprachen gibt folgendes Inklusionsdiagramm:



ss

Vermutung: Das Halteproblem für einen Prioritätsmulticounterautomaten mit einem zusätzlichem Keller, der bei einem Nulltest leer sein muß ist ebenfalls entscheidbar.
Die Klasse der von solchen Automaten erkannten Sprachen enthält alle Sprachklassen des Diagramms.

Eine andere Möglichkeit zur Definition von Zählerautomaten besteht darin *blinde Zähler* (vergleiche BLIND in [Gre78]) zu verwenden, deren Inhalt auch negativ sein kann. Bei einem solchen Zähler kann man außer Inkrement und Dekrement

anstelle eines Nulltest einen Natürlichkeitstest (d.h. Zahlerinhalt ≥ 0) als Instruktion erlauben.
Einen *Prioritätsnatürlichkeitstestautomaten* kann man analog zum Prioritätsmulticounterautomaten definieren; die einzigen Unterschiede sind dabei die neue Konfigurationenmenge $C_A = Z \times \Sigma^* \times \mathbb{Z}^k$ und die Konfigurationsübergangsrelation $\langle z, ax, n_1, ..., n_k \rangle \vdash_A \langle z', x, n_1 + i_1, ..., n_k + i_k$ welche jetzt die Bedinung $z, z' \in Z, a \in \Sigma, \langle (z, a, j), (z', i_1, ...i_k) \rangle \in \delta, \ \forall i \leq j \ n_i \geq 0$ hat.
Ein solcher Automat kann einen Prioritätsmulticounterautomaten mit der doppelten Anzahl von Zählern simulieren. Dabei wird ein Zähler durch zwei Zähler mit umgekehrtem Vorzeichen (Inkrementieren und Dekrementieren vertauscht) simuliert. Einer der Zähler wird bei jedem Schritt auf ≥ 0 geprüft wird und der Nulltest wird durch einen Natürlichkeitstest des 'Gegenzählers' simuliert.
Umgekehrt kann ein solcher Automat auch von einem Prioritätsmulticounterautomaten mit der doppelten Anzahl von Zählern simuliert werden. Der Inhalt eines Zählers wird als Differenz zweier Zähler dargestellt und für jede Veränderung des simulierten Zählers wird nichtdeterministisch entweder der positive Teil oder der negative Teil in umgekehrter Weise verändert. Der Natürlichkeitstest wird durch einen Nulltest des negativen Teils simuliert.
Im Gegensatz zu Prioritätsmulticounterautomaten läßt sich vermuten, daß zugelassen werden kann, daß die ersten zwei oder vielleicht auch drei (aber keinesfalls vier) Zähler unabhängig von ihrer Priorität auf Natürlichkeit getestet werden können, ohne daß das Leerheitsproblem dadurch unentscheidbar wird. Ein solcher Automat kann Sprachen erkennen (siehe später in Kapitel 5), die nicht in *PMC* liegen.

4 Die Synchronisation von Halbspuren

4.1 Die Inklusion von Halbspuren

Eine grundsätzliche Fragestellung für eine gültige Ablauffolge eines konkurrierenden Systems ist, ob eine andere Ablauffolge ebenfalls gültig ist. Ist das System durch partielle Kommutation beschrieben, so entspricht diese Fragestellung dem folgenden Spuräquivalenzproblem: Gegeben seien ein symmetrisches Semikommutationssystem C und zwei Wörter w, w'; gilt $w \overset{*}{\underset{C}{\Longrightarrow}} w'$?
Daß das Spuräquivalenzproblem in TC^0 liegt, wurde in [AG91] gezeigt. Dort wird für jedes abhängige Paar (a, b) in Logik erster Stufe mit zusätzlichem Majoritätsquantor beschrieben, ob es für jede Postion in w eine entsprechende Position in w' mit der gleichen Anzahl vorhergehender a's und b's gibt. Nach einem Resultat von [BIS90] liegt dies folglich in TC^0.
Die Struktur S eines Wortpaares w, w' für ein festes Halbabhängigkeitsalphabet ist

$$S = \langle \{1...k\}, \Xi_{w,x_1}, ..., \Xi_{w,x_n}, \Xi_{w',x_1}, ..., \Xi_{w',x_n} \rangle$$

mit dem Universum $\{1...k\}$ und Prädikaten

$$\Xi_{w,x_1}(i) \equiv \text{'Das } i\text{-te Zeichen in } w \text{ ist } x_1\text{'}.$$

Die Struktur S eines Wortpaares w, w' mit einem Halbabhängigkeitsalphabet (A, SD) mit $A = \{a_1, ..., a_{|A|}\}$ ist

$$S = \langle \{1...k\}, \Xi_w, \Xi_{w'}, \Xi_{SD} \rangle$$

mit dem Universum $\{1...k\}$ mit $|w|, |w'|, |A| \leq k$ und Prädikaten

$$\Xi_w(i, j) \equiv \text{'Das } i\text{-te Zeichen in } w \text{ ist } a_j\text{'}$$
$$\text{und } \Xi_{SD}(i, j) \equiv (a_i, a_j) \in SD.$$

In der Logik erster Stufe können zusätzlich die Prädikate $=, \leq$ und BIT mit

$$BIT(i, j) \equiv \text{'Das } i\text{-te Bit in der Binärdarstellung von } j \text{ ist 1'},$$

die logisch Junktoren $\wedge, \vee \neg$ und die Quantoren $\forall$ und $\exists$ für Variablen über dem Universum verwendet werden.
Steht auch der *Majoritätsquantor*

$$Mi : \phi(i) \equiv \text{ 'Für mehr als die Hälfte aller möglichen } i\text{'s ist } \phi(i) \text{ wahr'}$$

zur Verfügung, so kann damit auch das Prädikat

$$j = \#i : \phi(i) \equiv \text{ '}j \text{ ist die Zahl der Werte } i \text{ für die } \phi(i) \text{ gilt'}$$

ausgedrückt werden.
Die gleiche Methode wie in [AG91] kann auch auf asymmetrische Semikommutationssysteme angewendet werden:

Theorem 4.1 *Das folgende Halbspur-Inklusionsproblem ist in TC^0: Gegeben seien ein asymmetrisches Semikommutationssystem SC durch das Halbabhängigkeitsalphabet (A, SD) und zwei Wörter w, w'; gilt $w \stackrel{*}{\underset{SC}{\Longrightarrow}} w'$ bzw. $[w'\rangle \subseteq [w\rangle$?*

Beweis: Nach folgende Projektionslemma (siehe [Cle84]) gilt: Für ein Semikommutationssystem SC gilt

$$w \stackrel{*}{\underset{SC}{\Longrightarrow}} w' \Leftrightarrow \forall (a,b) \in A^2, \Pi_{ab}(w) \stackrel{*}{\underset{SC}{\Longrightarrow}} \Pi_{ab}(w').$$

In analoger Weise wie im Spuräquivalenzfall können wir für jedes (asymmetrisch) abhängige Paar $(a_i, a_j) \in SD$ beschreiben, ob es für jede Position l in w eine entsprechende Position m in w' mit der gleichen Anzahl vorhergehender a_i's und einer höchstens so großen Anzahl vorhergehender a_j's gibt. Um sicherzustellen, daß w' keine zusätzlichen Zeichen enthält, wird zusätzlich beschrieben, daß es für jede Position m in w' eine entsprechende Position l in w mit der gleichen Anzahl vorhergehender a_i's gibt.
Das beschreibende Prädikat lautet

$$\begin{aligned}
&\forall i\ \forall j\ (\neg \Xi_{SD}(i,j)\ \vee\ \forall l\ \exists m\ \exists k_1\ (k_1 = \#l' : l' \leq l\ \wedge\ \Xi_w(l',i)) \wedge \\
&\qquad\qquad (k_1 = \#m' : m' \leq m\ \wedge\ \Xi_{w'}(m',i)) \wedge \\
&\qquad\qquad \exists k_2\ \exists k_3\ (k_2 = \#l' : l' \leq l\ \wedge\ \Xi_w(l',j)) \wedge \\
&\qquad\qquad\qquad (k_3 = \#m' : m' \leq m\ \wedge\ \Xi_{w'}(m',j)) \wedge \\
&\qquad\qquad\qquad (k_3 \leq k_2) \wedge \\
&\forall i\ \forall m\ \exists l\ \exists k_1\ (k_1 = \#l' : l' \leq l\ \wedge\ \Xi_w(l',i)) \wedge \\
&\qquad\qquad (k_1 = \#m' : m' \leq m\ \wedge\ \Xi_{w'}(m',i))
\end{aligned}$$

und nach [BIS90] liegt die Inklusion von Halbspuren folglich in TC^0. ∎

Es ist leicht zu sehen, daß die Inklusion von Halbspuren auch für ein festes SD in TC^0 liegt. Bereits für $SD = \{(1,0)\}$ liegt die Inklusion von Halbspuren jedoch nicht in AC_0, da gemäß [AG91] der Majoritätsquantor in konstanter Tiefe durch

$$w \in MAJ \Leftrightarrow \bigvee_{i < \frac{|w|}{2}} [1^{|w|-i}0^i\rangle \subseteq [w\rangle$$

darauf reduziert werden kann.

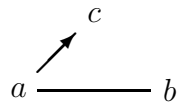
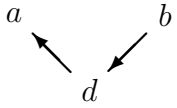
4.2 Die Synchronisierbarkeit von Halbspuren

Gegeben seien m Halbabhängigkeitsalphabete (A_i, SD_i) mit $0 < i \leq m$ und die Halbspuren $[u_1\rangle_1, [u_2\rangle_2, \ldots [u_m\rangle_m$. Die Bedingung

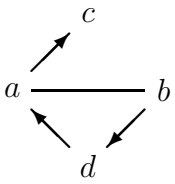
$$\forall i,j \leq m\ \forall a \in A_i \cap A_j\ \ |\ u_i\ |_a = |\ u_j\ |_a$$

ist notwendig für Synchronisierbarkeit. Dies zu überprüfen liegt in TC^0. Doch dies ist nicht hinreichend.

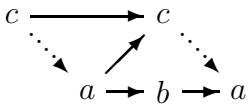
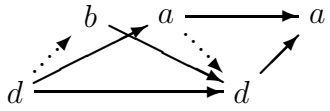
Beispiel: Betrachte die folgenden Halbabhängigkeitsalphabete

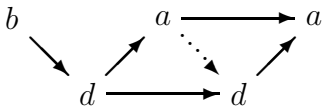
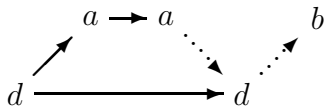
$(A_1, SD_1) =$  und $(A_2, SD_2) =$ 

mit dem durch Vereinigung erhaltenen

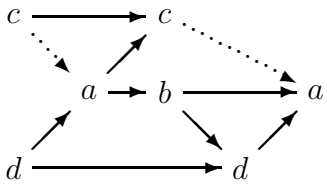
Halbabhängigkeitsalphabet $(A, SD) =$ 

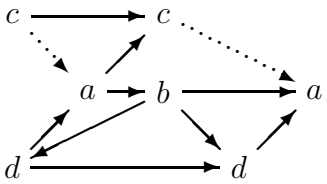
und die Halbspuren

$[cacba\rangle_1 =$  $[dbada\rangle_2 =$ 

$[bdada\rangle_2 =$  $[daadb\rangle_2 =$ 

Da in der Synchronisation die Ordnung jedes abhängigen Paares erhalten bleiben muß, kann man sie durch die Vereinigung beschreiben. Beispielsweise für $[cacba\rangle_1 \parallel [dbada\rangle_2$ erhalten wir

$[cdacbda\rangle =$ 



Andererseits erhalten wir für
$[cacba\rangle_1 \parallel [bdada\rangle_2$ den Graphen

und daher ist $[cacba\rangle_1 \parallel [bdada\rangle_2 = \emptyset$. Daraus läßt sich ersehen, daß Halbspuren genau dann synchronisierbar sind, wenn die Vereinigung ihrer Graphen keinen Kreis aus harten Kanten enthällt. (Störende weiche Kanten können herumgedreht werden). Dies führt zu den folgenden zwei Theoremen.

Theorem 4.2 *Das folgende Problem ist NLOGSPACE-vollständig: Gegeben seien m Halbabhängigkeitsalphabete (A_i, SD_i) mit $0 < i \leq m$ und die Halbspuren $[u_1\rangle_1, [u_2\rangle_2, \ldots [u_m\rangle_m$. Gilt $[u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots [u_m\rangle_m \neq \emptyset$?*

Beweis: Nach der deterministischen Überprüfung von $\forall i, j \leq m\ \forall a \in A_i \cap A_j \mid u_i \mid_a = \mid u_j \mid_a$ rät eine nichtdeterministische logarithmisch platzbeschränkte Turingmaschine einen Kreis und überprüft jede Verbindung indem sie die Reihenfolge und die Abhängigkeit des jeweiligen Paares in jeder Halbspur überprüft.

Nach [Imm88] und [Sze88] liegt das Komplement, d.h. die Nichtexistenz eines solchen Kreises ebenfalls in NLOGSPACE. Die NLOGSPACE-Härte folgt aus Theorem 4.3. ■

Theorem 4.3 *Das folgende Problem ist NLOGSPACE-vollständig: Gegeben seien ein Halbabhängigkeitsalphabet (A, SD) und die Halbspuren $[u\rangle, [v\rangle$ mit*

$$\forall a \in A \ |u_i|_a = |u_j|_a.$$

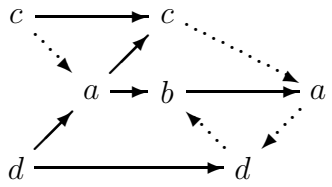
Gilt $[u\rangle \cap [v\rangle \neq \emptyset$?

Beweis: Das Problem ist in NLOGSPACE wegen Theorem 4.2. Das monotone Grapherreichbarkeitsproblem für gerichtete Graphen ist bekanntlicherweise NLOGSPACE-vollständig; für einen gegebenen Graphen

$$G = (\{s = a_1, a_2, \ldots a_n = e\}, R)$$

mit der Eigenschaft $(a_i, a_j) \in R \Rightarrow i < j$ soll ermittelt werden, ob ein gerichteter Pfad von s nach e existiert. Dies kann man auf das Synchronisierbarkeitsproblem reduzieren, indem man die Kante (e, s) zu dem Abhängigkeitsgraphen $(\{s, a_2, a_3 \ldots a_{n-1}, e\}, R)$ hinzufügt ($SD := R \cup \{(e, s)\}$) und nach der Synchronisierbarkeit von $[esa_2a_3 \ldots a_{n-1}\rangle$ und $[a_2a_3 \ldots a_{n-1}es\rangle$ fragt. Es existiert ein gerichteter Pfad von s nach e, gdw.die Halbspuren nicht synchronisierbar sind, weil ein Kreis aus harten Kanten vorhanden ist. ■

Für die Beschreibung modularer Systeme ist es nützlich, wenn die Synchronisation von Halbspuren immer eine Halbspur ist, wenn die Halbspuren synchronisierbar sind. Doch dies ist nicht immer der Fall, wie man am Beispiel $[cacba\rangle_1 \parallel [daadb\rangle_2$ sehen kann. Wir erhalten hierbei



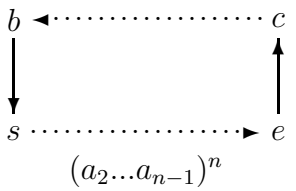
und daher gilt $[cacba\rangle_1 \parallel [daadb\rangle_2 = [cdacbad\rangle_1 \cup [cdacdba\rangle_2$ da entweder die weiche Kante $a \dashrightarrow d$ oder die weiche Kante $d \dashrightarrow b$ umgedreht werden muß. Es ist leicht zu sehen, daß die Synchronisation von Halbspuren eine Halbspur ist, gdw.die Vereinigung ihrer Graphen keinen Kreis aus harten Kanten und keinen Kreis mit mindestens 2 weichen Kanten, welche unabhänging voneinander umgedreht werden können, besitzt. Damit kommt man zu den zwei folgenden Theoremen.

Theorem 4.4 *Das folgende Problem ist NLOGSPACE-vollständig: Gegeben seien m Halbabhängigkeitsalphabete (A_i, SD_i) mit $0 < i \leq m$ und die Halbspuren $[u_1\rangle_1, [u_2\rangle_2, \ldots [u_m\rangle_m$. Gilt $\exists w \in A^*$ mit $[u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots [u_m\rangle_m = [w\rangle$?*

Beweis: Eine nichtdeterministische logarithmisch platzbeschränkte Turingmaschine kann zu einer weichen Kante einen Kreis aus harten Kanten finden, welcher es verbieten würde, daß diese Kante umgedreht wird. Mit Hilfe der Technik von [Imm88] und [Sze88] kann sie somit auch für eine weiche Kante überprüfen, ob sie umgedreht werden kann. Somit kann sie auch einen Kreis mit zwei weichen Kanten raten und überprüfen, ob beide umgedreht werden können. Unter abermaliger Anwendung von [Imm88] und [Sze88] kann sie, nachdem sie zuvor die Synchronsierbarkeit festgestellt hat, überprüfen, ob kein solcher Kreis existiert. Die NLOGSPACE-Härte folgt aus Theorem 4.5. ∎

Theorem 4.5 *Das folgende Problem ist NLOGSPACE-vollständig: Gegeben seien ein Halbabhängigkeitsalphabet* (A, SD) *und die Halbspuren* $[u\rangle, [v\rangle$ *mit* $[u\rangle \cap [v\rangle \neq \emptyset$*. Gilt* $\exists w \in A^*$ *mit* $[u\rangle \cap [v\rangle = [w\rangle$*?*

Beweis: Das Problem ist in NLOGSPACE wegen Theorem 4.4. Das monotone Grapherreichbarkeitsproblem für gerichtete Graphen kann man auf das Problem reduzieren, indem man die zusätzlichen Knoten b und c und die Kanten $(b,s),(b,c),(e,c)$ und (e,s) zu dem Graphen hinzufügt und bei dem so erzeugten Abhängigkeitsgraphen $(\{b,c,s,a_2,a_3,\ldots a_{n-1},e\}, SD)$ überprüft, ob die Synchronisation der Halbspuren $[cbs(a_2a_3\ldots a_{n-1})^n e\rangle$ und $[s(a_2a_3\ldots a_{n-1})^n ecb\rangle$ eine Halbspur ist.



Die Synchronisation ist genau dann die Halbspur $[bs(a_2a_3\ldots a_{n-1})^n ec\rangle$, wenn ein Pfad aus harten Kanten von s nach e existiert. ∎

4.3 Die Synchronisierbarkeit in Semikommutationssystemen

Es ist leicht zu sehen, daß nur ein Kreis aus harten Kanten von mindestens zwei Halbspuren für die Nichtsynchronisierbarkeit von Halbspuren für die $\forall i,j \leq m\ \forall a \in A_i \cap A_j \mid u_i \mid_a = \mid u_j \mid_a$ gilt, verantwortlich sein kann. (Ein solcher Kreis kann auch aus einer einzigen symmetrischen Kante bestehen.) Dieser Kreis muß sich natürlich im Abhängigkeitsgraphen widerspiegeln und sich aus Kanten aus mindestens zwei der Abhängigkeitsgraphen zusammensetzen.

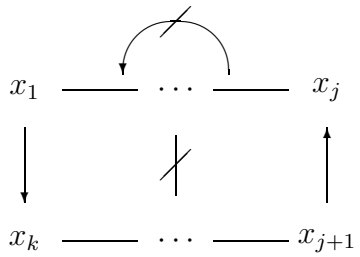
Theorem 4.6 *Gegeben seien* m *Halbabhängigkeitsalphabete* (A_i, SD_i) *mit* $0 < i \leq m$*. Die folgenden Aussagen sind äquivalent:*

i) $\forall u_1 \in A_1^*, \ldots u_m \in A_m^*, (\forall i,j \leq m \ \forall a \in A_i \cap A_j \mid u_i \mid_a = \mid u_j \mid_a) \Rightarrow [u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots [u_m\rangle_m \neq \emptyset$

ii) $\neg\exists 1 < k, i \neq j, C = \{(x_1,x_2), \ldots (x_{k-1},x_k), (x_k,x_1)\}$ *mit* $C \subseteq SD \wedge C \cap SD_i \neq \emptyset \wedge C \cap SD_j \neq \emptyset$

Auch diese Eigenschaft ist NLOGSPACE-vollständig.
Das nächste Theorem besagt, daß Halbabhängigkeitsalphabete genau dann die Eigenschaft haben, daß es vorkommen kann, daß die Synchronisation zweier Halbspuren nicht leer aber trotzdem keine Halbspur ist, wenn in der Vereinigung dieser Halbabhängigkeitsalphabete ein Kreis derart existiert, daß

- die Knoten verschieden sind,
- zwei gerichtete Kanten (x_1, x_k) und (x_{j+1}, x_j) in Gegenrichtung zeigen,
- diese beiden Kanten nicht durch eine Sehne getrennt sind,
- keine Sehne rückwärts verläuft und
- der Kreis in jedem der Halbabhängigkeitsalphabete an einer Stelle unterbrochen ist oder an einer Stelle eine gerichtete Kante in Gegenrichtung besitzt, doch im diesem Fall muß an dieser Stelle eine Kante in einem der anderen Halbabhängigkeitsalphabete existieren, in welchem der Kreis an einer anderen Stelle unterbrochen ist oder eine Kante in Gegenrichtung besitzt.



Theorem 4.7 *Gegeben m Halbabhängigkeitsalphabete* (A_i, SD_i) *mit* $0 < i \leq m$. *Die folgenden Aussagen sind äquivalent:*

i) $\forall u_1 \in A_1^*, \ldots u_m \in A_m^* \exists w \in A^*$ *mit* $[u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots [u_m\rangle_m = [w\rangle$ *oder* $= \emptyset$

ii) $\neg\exists 0 < j < k, (x_1,x_2), \ldots, (x_{k-1},x_k) \in SD \cup SD^{-1}$ *mit*
$\forall n \neq p \Rightarrow x_n \neq x_p \wedge$
$(x_1,x_k), (x_{j+1},x_j) \in SD \setminus SD^{-1} \wedge$
$\forall n,p$ *mit* $0 < n \leq j < p \leq k, n+1 < p < n+k-1 \Rightarrow$
$(x_n,x_p) \notin SD \cup SD^{-1} \wedge$
$\forall n,p$ *mit* $1 < n+1 < p \leq k \Rightarrow (x_p,x_n) \notin SD \wedge$

$$\forall i \leq m \exists n \leq k, \Big((x_{n+1}, x_n) \notin SD_i \wedge \\ ((x_n, x_{n+1}) \notin SD_i \vee \\ \exists l \leq m, \neq i, p \leq k, \neq n, (x_{p+1}, x_p) \notin SD_l \wedge \\ (x_n, x_{n+1}) \in SD_l \cup SD_l^{-1}) \Big)$$

Das einfachste Beispiel hierfür ist $(A_1, SD_1) = (\{a, b\}, \{(a, b)\})$ und $(A_2, SD_2) = (\{a, b\}, \{(b, a)\})$ in diesem Fall gilt $[ba\rangle_1 \parallel [ab\rangle_2 = \{ab, ba\} \neq [u\rangle$ für jedes u.
Betrachten wir nur Halbspuren über einem Halbabhängigkeitsalphabet, so erhalten wir, daß es vorkommen kann, daß die Synchronisation von Halbspuren keine Halbspur aber auch nicht leer ist, gdw.ein Kreis aus verschiedenen Knoten mit zwei gerichteten Kanten in jeder Richtung derart existiert, daß Sehnen nur in der Richtung von einem Teil, von dem aus zwei dieser Kanten wegzeigen zu einem Teil zu dem die beiden anderen hinzeigen (mit anderen Worten Sehnen, die diese Kanten 'abkürzen') vorkommen .

Korollar 4.1 *Gegeben ein Halbabhängigkeitsalphabet* (A, SD).
Die folgenden Aussagen sind äquivalent:

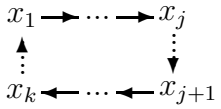
i) $\forall u, v \in A^* \exists w \in A^*$ *mit* $[u\rangle \parallel [v\rangle = [w\rangle$ *oder* $= \emptyset$

ii) $\neg \exists 0 < j < k, n \neq p \neq j, k, (x_1, x_2), \ldots (x_{k-1}, x_k) \in SD \cup SD^{-1}$ *mit*
$\forall n \neq p \Rightarrow x_n \neq x_p \wedge$
$(x_1, x_k), (x_{j+1}, x_j) \in SD \setminus SD^{-1} \wedge$
$(x_{n+1}, x_n), (x_{p+1}, x_p) \in SD^{-1} \setminus SD \wedge$
$\forall n, p$ *mit* $0 < n \leq j < p \leq k, n + 1 < p < n + k - 1 \Rightarrow$
$(x_n, x_p) \notin SD \cap SD^{-1} \wedge$
$\forall n, p$ *mit* $1 < n + 1 < p \leq k \Rightarrow (x_p, x_n) \notin SD$

So gibt es gerade die beiden grundlegenden Beispiele für Halbabhängigkeitsalphabete (alle anderen lassen sich durch Zusammenfassen von Knoten darauf reduzieren):

Im ersten Beispiel würde jede eventuelle Sehne die Situation $[cabd\rangle \parallel [bdca\rangle = [abdc\rangle \cup [dcab\rangle \neq [u\rangle$ für jedes u verhindern. Im zweiten Beispiel 'funktioniert' die Situation $[dcab\rangle \parallel [bdca\rangle = [abdc\rangle \cup [cdba\rangle \neq [u\rangle$ für jedes u unabhängig von der Existenz der Kante von a nach d.
Nun kommen wir zum Beweis von Theorem 4.7:
Beweis: $\neg ii \Rightarrow \neg i$: Betrachte den kürzesten Kreis dieser Art und wähle $u_i =$

$x_{n+1} \dots x_k x_1 x_2 \dots x_n$ so, daß $(x_{n+1}, x_n) \notin SD_i$ und $(x_n, x_{n+1}) \notin SD_i \vee \exists l \leq m, \neq i, (x_n, x_{n+1}) \in (SD_l \cup SD_l^{-1}) \cap [u_l\rangle_l$. Somit hat die Synchronisation einen Kreis mit den weichen Kanten (x_k, x_1) und (x_j, x_{j+1}) und keinen Kreis aus harten Kanten.
$\neg i \Rightarrow \neg ii$: Wähle v, v', u_i (Längen-)minimal mit $\forall i \Pi^A_{A_i}(v) = u_i, \forall i \Pi^A_{A_i}(v') = u_i$ und $\neg \exists w$ mit $\forall i\ \Pi^A_{A_i}(w) = u_i, v, v' \in [w\rangle$. Konstruiere G als die Vereinigung aller Graphen von $[u_i\rangle_i$. Konstruiere G' aus G indem jedes $b \dashrightarrow a$ durch $a \longrightarrow b$ ersetzt wird, falls $a = x_1 \longrightarrow x_2 \longrightarrow \dots x_j = b$ in G. Wegen v, v' enthält der Graph G' keinen Kreis aus harten Kanten. Somit muß G' einen Kreis mit mindestens zwei weichen Kanten besitzen,

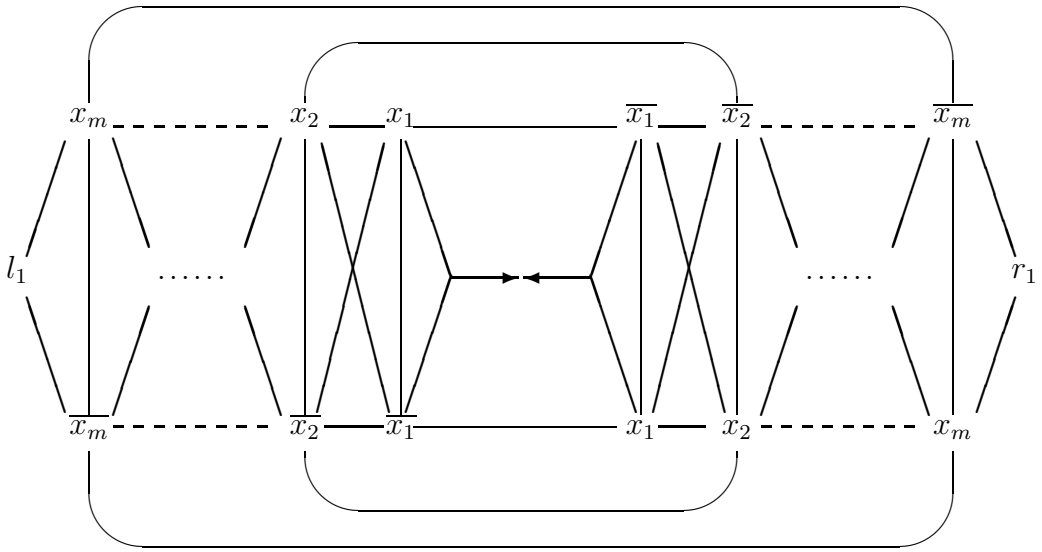


da ansonsten G' den Graphen für $[w\rangle$ darstellen würde. G' muß ohnehin die harten Kanten von G enthalten, denn es gilt $\forall i\ \Pi^A_{A_i}(w) = u_i$ und Kreise mit einer weichen Kante verschwinden bei der Konstruktion von G'. Der kürzeste Kreis dieser Art hat kein Sehne.
Mit der im folgenden beschriebenen iterativen Methode können wir einen Kreis mit mindestens zwei weichen Kanten finden, dessen harte Kanten nicht neu, d.h. aus G sind und der nur neue Sehnen (aus $G' \setminus G$) besitzt, welche die beiden weichen Kanten nicht trennen und in Richtung des Kreises verlaufen:
Wir ersetzen eine neue Kante im Kreis durch den Pfad originaler Kanten, durch den diese Kante erzeugt wurde. Führt dies zu einer unerlaubten Sehne, so können wir immer einen kürzeren Kreis finden, der mindestens zwei weiche Kanten und keine unerlaubte Sehne besitzt. Da sich in einem solchen Schritt die Anzahl der weichen Kanten oder die Anzahl der neuen harten Kanten im Kreis vermindert, muß diese Methode terminieren.
Aufgrund der Minimalität der u_i gibt es keine anderen Knoten in G' außer denen im Kreis und kein Element von A kommt mehrfach vor. Ferner kann der Kreis aufgrund der Minimalität keine weiche Sehne besitzen und daher kann es in SD keine Sehne in Gegenrichtung geben. Jede Kante im Kreis muß von einem der $[u_i\rangle_i$'s stammen. Doch der Kreis muß für jedes $[u_i\rangle_i$ an irgendeiner Stelle offen sein und dies könnte eine Stelle sein, an der $[u_i\rangle_i$ eine harte Kante hätte beitragen können. In diesem Fall muß die Kante von einem anderen $[u_l\rangle_l$ geliefert werden. Dieses $[u_l\rangle_l$ muß jedoch an einer anderen Selle offen sein. Genau dies drückt die Formel in $\neg ii$ aus. ■

Theorem 4.8 *Es ist* **co-NP***-vollständig zu entscheiden, ob für Semikommutationssysteme bzw. für ein Semikommutationssystem die Aussagen von Theorem 4.7 bzw. Korollar 4.1 erfüllt sind.*

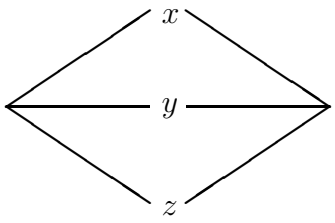
Beweis: Es ist leicht zu sehen, daß das Problem in **co-NP** liegt. Zum Beweis der Härte kann fast die gleiche Graphkonstruktion wie in [DOR94] für ein Halbabhängigkeitsalphabet (A, SD) verwendet werden. Der einzige Unterschied ist, daß

anstelle der zwei dort verwendeten gerichteten Kanten jeweils ein zusätzlicher Knoten und zwei zu diesem gerichtete Kanten einzusetzten sind.
Gegeben sei ein Ausdruck $e = c_1 \wedge \ldots \wedge c_n$ in konjunktiver 3-Normalform über den Variablen $x_1, ..., x_m$. Wir konstruieren nun daraus einen Graphen für (A, SD), der genau 4 gerichtete Kanten besitzt. Das Halbabhängigkeitsalphabet (A, SD) besitzt in diesem Fall die Eigenschaft (ii) in Korollar 4.1 genau dann nicht, wenn ein Kreis ohne Sehnen aus verschiedenen Knoten mit zwei gerichteten Kanten in jeder Richtung im Graphen existiert. Durch die Konstruktion wird dies genau dann der Fall sein, wenn der Ausdruck e erfüllbar ist.
Zunächst konstruieren wir einen Graphen $G(x)$ mit $x = (x_1, ..., x_m)$ wie im folgendem Bild:



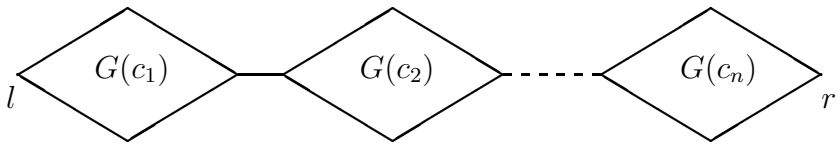
Jede Belegung der Variablen $x_1, ..., x_m$ entspricht eindeutig einem Pfad ohne Sehnen von l_1 nach r_1, der durch die Knoten verläuft, welche mit einer Variablen benannt sind, die mit 'wahr' belegt ist.
Zu jeder Klausel $c = (x \vee y \vee z)$ konstruieren wir den Graphen

$G(c) \;=$

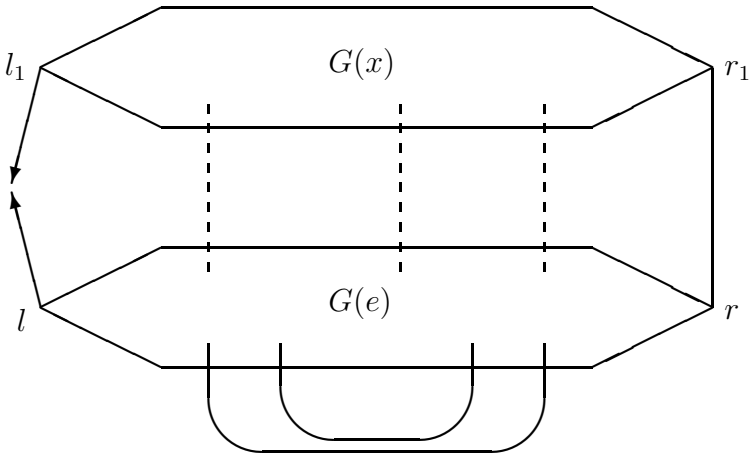


und zu dem Ausdruck $e = c_1 \wedge \cdots \wedge c_n$ den Graphen

$G(e) \;=$



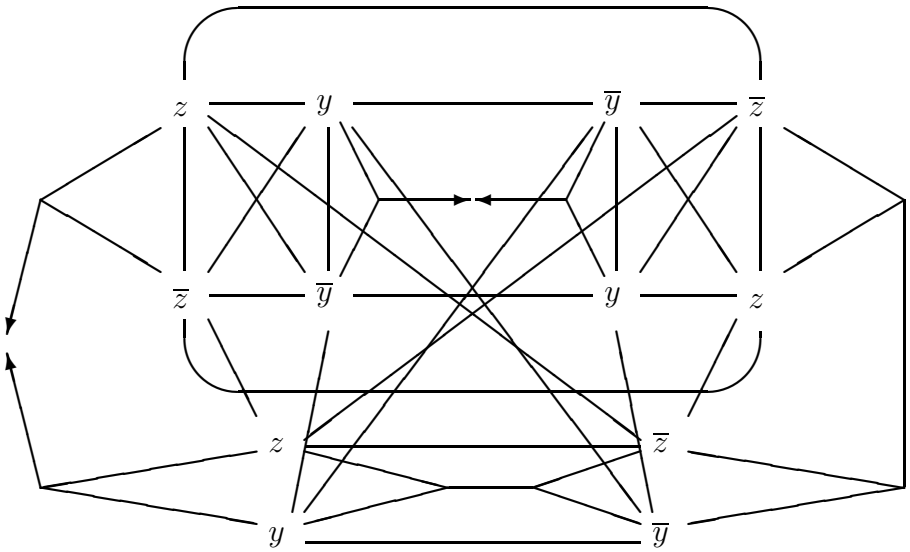
als disjunkte Vereinigung $G(c_1) \dot\cup \cdots \dot\cup G(c_n)$, wobei zusätzlich für alle $1 \leq i < n$ das rechte Ende von $G(c_i)$ mit dem linken Ende von $G(c_{i+1})$ verbunden wird. Das linke Ende von $G(e)$ nennen wir l und das rechte r. Existiert eine erfüllende Belegung der Variablen $x_1, ..., x_m$, so ist in jeder Klausel eine Variable 'wahr' und somit existiert auch ein Pfad ohne Sehnen von l_1 nach r_1, der nur durch Knoten verläuft, welche mit keiner Variablen benannt sind, die nicht mit 'wahr' belegt ist.
Den gesamten Graphen für e, welcher die Form



hat, erhalten wir durch die disjunkte Vereinigung $G(x) \dot\cup G(e)$ und durch Hinzufügen einer symmetrischen Kante zwischen r_1 und r und gerichteten Kanten von l und l_1 zu einem zusätzlichen Punkt. Weiterhin werden für alle $1 \leq i \leq m$ alle mit x_i benannten Knoten mit allen mit $\overline{x_i}$ benannten Knoten verbunden.
Existiert eine erfüllende Belegung der Variablen $x_1, ..., x_m$, so setzen sich die beiden entsprechenden Pfade in $G(x)$ und $G(e)$ mit den drei hinzugefügten Kanten zu einem Kreis ohne Sehnen mit zwei gerichteten Kanten in jeder Richtung zusammen.
Umgekehrt ergibt sich aus der Existenz eines solchen Kreises eine erfüllende Belegung von e wie folgt: Ein solcher Kreis muß durch die beiden gerichteten Kanten in $G(x)$ verlaufen. Setzt sich der Kreis auf einer Seite o.B.d.A über den mit x_1 benannten Knoten fort, so muß er dies auch auf der anderen Seite tun, da er sonst eine Sehne von x_1 nach $\overline{x_1}$ hätte. Damit ist ausgeschlossen, daß der Kreis durch einen mit $\overline{x_1}$ benannten Knoten verlaufen kann und das bisher beschriebene Kreisstück muß auf beiden Seiten entweder durch x_2 nach $\overline{x_2}$ fortgestzt werden. Hier wiederholt sich die Argumentation und allgemein gilt für alle $1 \leq i \leq m$, daß

der Kreis entweder nicht durch mit x_i oder nicht durch mit $\overline{x_i}$ benannte Knoten verlaufen kann. Da das Kreisstück durch $G(e)$ für alle Klauseln durch einen mit einer darin vorkommenden Variablen benannten Knoten gehen muß, erfüllt die entsprechende Belegung alle Klauseln von e. ■

Beispiel: Sei $e = ((z \vee y) \wedge (\overline{z} \vee \overline{y})$, so ergibt sich der folgende Graph:



Analog zu einem anderen Resultat in [DOR94] ist es Σ_2^P-vollständigzu entscheiden, ob die Richtung von asymmetrischen Kanten so verändert werden kann, daß die Synchronisation zweier Halbspuren immer eine Halbspur ist, wenn sie nicht leer ist:

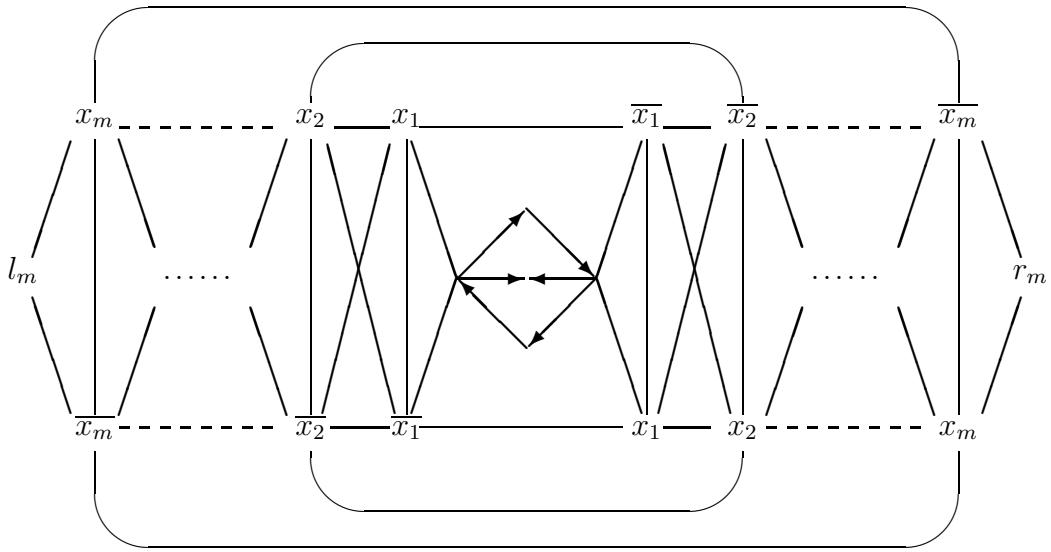
Theorem 4.9 *Das folgende Problem ist Σ_2^P-vollständig: Gegeben zwei Abhängigkeitsalphabete (A, D) und (A, D') mit $D' \subseteq D$.*
Gibt es ein SC mit $D = \{(a,b) \in A \times A \mid ab \Longrightarrow ba \notin SC \cap SC^{-1}\}$ und $D' = \{(a,b) \in A \times A \mid ab \Longrightarrow ba \notin SC \cup SC^{-1}\}$ und $\forall u, v \in A^* \exists w \in A^*$ mit $[u\rangle \parallel [v\rangle = [w\rangle$ oder $= \emptyset$?

Beweis: Durch Raten von SC und eine Frage an ein **co-NP**-Orakel entsprechend dem letzten Theorem kann das Problem in Σ_2^P entschieden werden.
Da, wie aus dem Beweis von Theorem 4.7 zu sehen ist, die Richtung der meisten Kanten in einem sehr großen Kreis die Bedingung nicht beeinflusst, kann in diesem Fall nicht die gleiche Graphkonstruktion wie in [DOR94] zum Beweis der Härte verwendet werden. Daher verwenden wir nun eine andere Konstruktion, welche Gebrauch von der Richtung erlaubter Sehnen im Kreis macht.
Nach [Sto77], [Wra77]) ist die Allgemeingültigkeit beschränkt quantifizierter Boolescher Formeln in konjunktiver 3-Normalform vollständig für die Klasse $\Pi_2^P = Co - \Sigma_2^P$. Wir werden nun diese auf das Problem reduzieren.

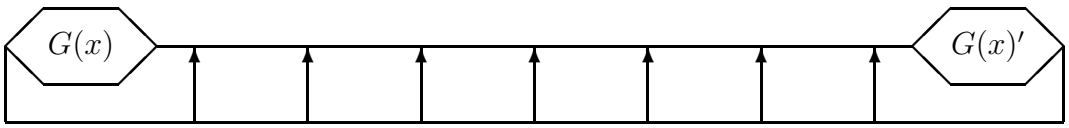
Gegeben sei die Formel $f = \forall x_1, \ldots, x_k \exists x_{k+1}, \ldots, x_m \; e =$

$$\forall x_1, \ldots, x_k \exists x_{k+1}, \ldots, x_m (c_1 \wedge \ldots \wedge c_n).$$

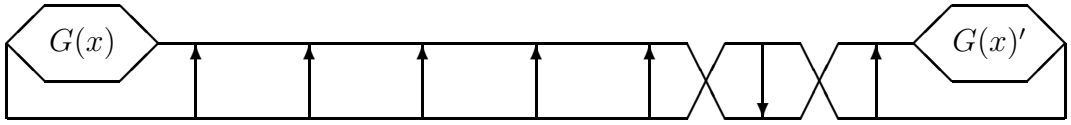
Zunächst konstruieren wir einen Graphen $G(x)$ mit $x = (x_1, ..., x_m)$ wie im folgendem Bild:



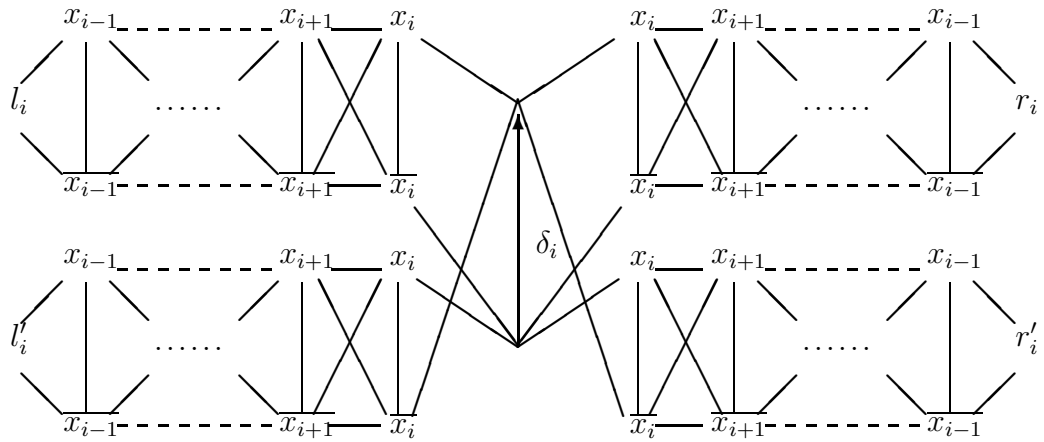
Die mit Pfeilen dargestellten Kanten sind die zu richtenden Kanten in $D \setminus D'$. Das 'Herzstück' von $G(x)$ besteht aus zwei Knoten, welche auf 3 Wegen mit Zwischenknoten über jeweils zwei zu richtende Kanten verbunden sind. Sind in zwei der Verbindungen alle Kanten gleichgerichtet oder sind in zwei der Verbindungen je zwei verschieden gerichtete Kanten, so wird durch diese bereits ein Kreis ohne Sehnen mit zwei gerichteten Kanten in jeder Richtung gebildet. Interessant ist daher nur der Fall, daß wie im obigem Bild eine der Verbindungen Kanten in verschiedener Richtung besitzt, eine der Verbindungen Kanten in der einen Richtung besitzt und eine der Verbindungen Kanten in der anderen Richtung besitzt. Die Aufgabe des Restes von $G(x)$ ist es wie im vorangehenden Beweis unbeabsichtigte Kreise zu verhindern und damit eine eindeutige Entsprechung von Kreisen und Belegungen herzustellen.
Ein Graph mit $G(x)$ und einer dazu identischen Kopie $G(x)'$ läßt genau dann die Konstruktion des Semikommutationssystems SC mit der gewünschten Eigenschaft nicht zu, wenn zu jeder Ausrichtung der Kanten in $D \setminus D'$ ein Kreis wie im folgendem Bild existiert, bei dem alle Sehnen in gleicher Weise gerichtet sind.



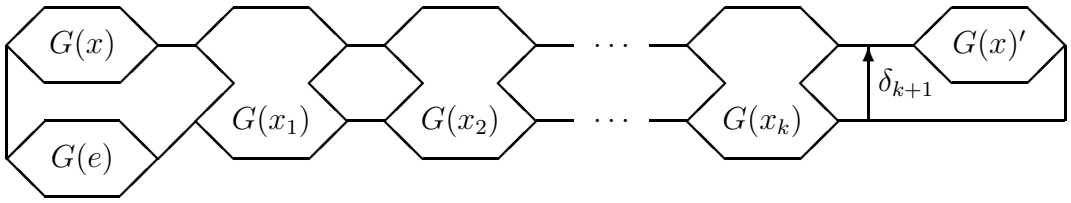
Für die andere Richtung einer Kante muß der Kreis einen anderen Weg nehmen, wie z.B. folgendes Bild zeigt:



Um die Richtung der Kante mit der Variablenbelegung zu koppeln konstruieren wir für alle $1 \leq i \leq k$ folgenden Graphen $G(x_i)$ durch welchen 2 Kreisstücke verlaufen:



Die Indizes werden dabei molulo m betrachtet. Wie zuvor gilt für alle $1 \leq j \leq m$, daß der Kreis entweder nicht durch mit x_j oder nicht durch mit $\overline{x_j}$ benannte Knoten verlaufen kann. Die Richtung der Sehne δ_i hängt davon ab, ob der Kreis durch x_i oder durch $\overline{x_i}$ verläuft. Den gesamten Graphen für f, welcher die Form



mit $G(e)$ wie im vorherigen Beweis hat, erhalten wir durch disjunkte Vereinigung

$$G(x) \dot\cup G(x)' \dot\cup G(e) \dot\cup G(x_1) \dot\cup \ldots \dot\cup G(x_k)$$

der Graphen, durch Hinzufügen einer weiteren zu richtenden Kante δ_{k+1} mit zwei zusätzlchen Knoten und von symmetrischen Verbindungskanten vom Endpunkt von δ_{k+1} nach r_k und nach l'_m, vom Anfangspunkt von δ_{k+1} nach r'_k und nach r'_m, für alle $1 \leq i < k$ von r_i nach l_{i+1} und von r'_i nach l'_{i+1}, von r_m nach l_1, von r' nach l'_1 und von l_m nach l'. Weiterhin werden wie im vorherigen Beweis für alle $1 \leq i \leq m$ alle mit x_i benannten Knoten mit allen mit $\overline{x_i}$ benannten Knoten verbunden und dadurch unbeabsichtigte Kreise verhindert.

Für jede Orientierung der gerichteten Kanten wird für alle $1 \leq i \leq k$ die Variable x_i als 'wahr', wenn δ_i und δ_{k+1} die gleiche Richtung haben und ansonsten als 'falsch' belegt. Ist die Formel f wahr, so kann für jede Orientierung der gerichteten Kanten die dadurch bestimmte Belegung der Varaiablen $x_1, ..., x_k$ zu einer erfüllenden Belegung von e erweitert werden und somit existiert der Kreis, bei dem alle Sehnen in gleicher Weise gerichtet sind.
Umgekehrt folgt daraus, daß bei allen Orientierungen der zu richtenden Kanten ein Kreis existiert, bei dem alle Sehnen in gleicher Weise gerichtet sind, daß jede Belegung der Varaiablen $x_1, ..., x_k$ zu einer dem Kreis entsprechenden erfüllenden Belegung von e erweitert werden kann und damit f wahr ist. ■

Bemerkung: Gegeben $2m$ Abhängigkeitsalphabete (A_i, D_i) und (A_i, D'_i) mit $D'_i \subseteq D_i$ für alle $0 < i \leq m$. Es ist leicht zu sehen, daß genau dann Semikommutationssysteme SC_i mit

$$D_i = \{(a,b) \in A_i \times A_i \mid ab \Longrightarrow ba \notin SC_i \cap SC_i^{-1}\} \text{ und}$$
$$D'_i = \{(a,b) \in A_i \times A_i \mid ab \Longrightarrow ba \notin SC_i \cup SC_i^{-1}\}$$

für $0 < i \leq m$ und $\forall u_1 \in A_1^*, \ldots u_m \in A_m^*$

$$(\forall i,j \leq m \forall a \in A_i \cap A_j \mid u_i \mid_a = \mid u_j \mid_a) \Rightarrow [u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots [u_m\rangle_m \neq \emptyset$$

existieren, wenn gilt:

$$\neg\exists 1 < k, i \neq j, C = \{(x_1,x_2), \ldots (x_{k-1},x_k), (x_k,x_1)\} \text{ mit}$$
$$C \subseteq \bigcup_l D'_l \cup \{(x_k,x_1)\} \wedge (x_k,x_1) \in \bigcup_l D_l \wedge C \cap D_i \neq \emptyset \wedge C \cap D_j \neq \emptyset.$$

Dies bedeutet, daß die Richtung von asymmetrischen Kanten so verändert werden kann, daß Halbspuren, deren Buchstabenzahlen auf der Schnittmenge übereinstimmen, immer synchronisierbar sind, d.h. die Richtung von asymmetrischen Kanten so verändert werden kann, daß kein gerichteter Kreis, der sich aus Kanten aus mindestens zwei der Abhängigkeitsgraphen zusammensetzt, existiert, gdw. kein solcher Kreis existiert, in dem höchstens eine der Kanten asymmetrisch ist.
Die Eigenschaft liegt somit in NLOGSPACE.

4.4 Das lokale Überprüfen der Synchronisierbarkeit von Halbspuren

In [DV89] wird eine leicht zu überprüfende (TC^0), notwendige Bedingung für die Synchronisierbarkeit von Spuren beschrieben und es wird ein Kriterium angegeben, mit dessen Hilfe für Abhängigkeitsalphabete entschieden werden kann, ob diese Bedingung auch hinreichend ist. Dieses Kriterium nennt man die *local checking property*. Sie ist **co-NP**-vollständig[DOR94]. Das gleiche machen wir nun für Halbspuren. Falls die zwei Halbspuren $[u_1\rangle_1, [u_2\rangle_2$ synchronisierbar sind, so gilt $\Pi^{A_1}_{A_1\cap A_2}(u_1) \stackrel{*}{\underset{SC'}{\Longrightarrow}} \Pi^{A_2}_{A_1\cap A_2}(u_2)$ für das Semikommutationssystem SC' mit $SD' = SD_1 \cap SD_2^{-1}$. Gemäß Theorem 4.1 liegt das Überprüfen dieser Bedingung in TC^0.

Lemma 4.1 *Die Bedingung ist hinreichend, falls die Komposition der Semikommutationssysteme SC_1 und SC_2^{-1} ein Semikommutationssystem ist, dies bedeutet $f_{SC_1} \circ f_{SC_2^{-1}} = f_{SC'}$ für ein Semikommutationssystem SC'.*

Beweis: Wenn die Komposition der Semikommutationssysteme SC_1 und SC_2^{-1} ein Semikommutationssystem ist, so ist deren Komposition SC bestimmt durch $SD' = SD_1 \cap SD_2^{-1}$. Sei $A_1 \dot\cup \{a_1, \ldots a_n\} = A_2 \dot\cup \{b_1, \ldots b_m\}$. Falls $\Pi^{A_1}_{A_1\cap A_2}(u_1) \stackrel{*}{\underset{SC'}{\Longrightarrow}} \Pi^{A_2}_{A_1\cap A_2}(u_2)$ so gilt auch $u_1 a_1^{|u_2|_{a_1}} \ldots a_n^{|u_2|_{a_n}} \stackrel{*}{\underset{SC'}{\Longrightarrow}} u_2 b_1^{|u_1|_{b_1}} \ldots b_m^{|u_1|_{b_m}}$, da $(A_1 \times \{a_1, \ldots a_n\}) \cap SD_1 = \emptyset$ und $(A_2 \times \{b_1, \ldots b_m\}) \cap SD_2 = \emptyset$, und damit die jeweils nicht vorkommenden Buchstaben unabhängig sind, also gilt auch

$$u_1 a_1^{|u_2|_{a_1}}, \ldots a_n^{|u_2|_{a_n}} \stackrel{*}{\underset{SC_1}{\Longrightarrow}} s \stackrel{*}{\underset{SC_2^{-1}}{\Longrightarrow}} u_2 b_1^{|u_1|_{b_1}}, \ldots b_m^{|u_1|_{b_m}}$$

und folglich ist s eine Synchronisation von $[u_1\rangle_1$ und $[u_2\rangle_2$. ∎

Theorem 4.10 *Die* local checking property *für Synchronisierbarkeit von zwei Halbspuren ist* **co-NP***-vollständig.*

Beweis: Ein Kriterium dafür, ob die Komposition zweier Semikommutationssysteme ein Semikommutationssystem ist, findet sich in [RW91]. Es ist leicht zu sehen, daß dieses Kriterium in **co-NP** ist und da das Kriterim für Konfluenz, von welchem in [DOR94] gezeigt wurde, daß es ebenfalls **co-NP**-vollständig ist, ein Spezialfall davon ist, ist dieses ebenfalls **co-NP**-vollständig. ∎

5 Die Synchronisation und die Maximalität von Halbspursprachen

Es ist leicht zu sehen, daß für reguläre Stringsprachen R_1, R_2 die Synchronisation ebenfalls eine reguläre Sprache (mittels eines Produktautomaten) ist und die Synchronisierbarkeit daher leicht (NLOGSPACE) zu entscheiden ist. Andererseits ist bei regulären Halbspursprachen die Synchronisierbarkeitfrage d.h. $f_{SC}(R_1) \parallel f_{SC}(R_2) \neq \emptyset$ oder die Frage $f_{SC}(R_1) \parallel R_2 \neq \emptyset$ im allgemeinen unentscheidbar. In diesem Kapitel soll nun untersucht werden, welche Eigenschaft SD (bzw. SC) haben muß, damit diese Frage entscheidbar wird.
Desweiteren ist die Frage nach der Maximalität einer regulären Sprache bezüglich eines Semikommutationssystems von Interesse. Die Menge der maximalen Wörter einer Sprache L ist

$$max_{SC}(L) := \{u \in L \mid \forall v \in L \; v \overset{*}{\underset{SC}{\Longrightarrow}} u \longrightarrow u \overset{*}{\underset{SC}{\Longrightarrow}} v\}.$$

Es ist leicht zu sehen, daß die Frage, ob L *maximal* ist, d.h. ob $max_{SC}(L) = L$ äquivalent ist zur Frage $f_{SC}(L') \cap L = \emptyset$ mit

$$L' := \{w \mid \exists u \in L \; u \underset{SC \cap SC^{-1}}{\overset{*}{\Longrightarrow}} v \underset{SC \setminus SC^{-1}}{\Longrightarrow} w\}.$$

Die Leerheit und die Unendlichkeit von Halbspursprachen sind bereits an der zugrundeliegenden regulären Sprache mit geringer Komplexität (NLOGSPACE) zu entscheiden. Auch das Wortproblem ist sowohl für ein festes Semikommutationssystem (NLOGSPACE) als auch uniform (laut [BMS82] NP-vollständig) entscheidbar. Universalität, Äquivalenz, Inklusion, Erkennbarkeit und Disjunktheit sind jedoch wie in [Iba78] und [AH89] gezeigt bereits für symmetrische Kommutationssysteme im allgemeinen unentscheidbar. Die Maximalität hingegen ist eine triviale Eigenschaft für symmetrische Kommutationssysteme.
Ist SC konfluent (siehe nächstes Kapitel), so ist gemäß [RW91] die Komposition von SC und SC^{-1} ein Semikommutationssystem und damit die Frage $f_{SC}(R_1) \parallel f_{SC}(R_2) \neq \emptyset$ äquivalent zu der Frage $f_{SC \cup SC^{-1}}(R_1) \parallel R_2 \neq \emptyset$. Somit genügt es in diesem Fall die Synchronisierbarkeit einer Halbspursprache mit einer regulären Sprache zu betrachen.

5.1 Zählerhalbabhängigkeiten

In [CG90] wird gezeigt, daß sich Semikommutationssysteme mit Hilfe der Dekomposition in atomare Semikommutationssysteme zerlegen lassen. Atomare Semikommutationssysteme haben die Form $\overline{SD} = A_1 \times A_2$. Ist $|A_1| = 1$ oder $|A_2| = 1$, so ist $\overline{SD}$ eine atomare Zählersemikommutation, d.h. das Element in A_1 bzw. A_2 kann mit Hilfe eines Zählers an eine andere Stelle bewegt werden.

Definition Entsprechend der Definition von *Zählersemikommutationen*, welche in [Gon93] definiert wurden, ist SD eine *Zählerhalbabhängigkeit* (*CSD*), falls wenn $(a, b) \in SD$ und $(c, d) \in SD$, so auch $(a, d) \in SD$ und $(c, b) \in SD$.

Theorem 5.1 *[Gon93]* $f_{SC}(MC) \subseteq MC \Leftrightarrow SD$ *ist eine CSD.*

D.h. die Multicountersprachen sind abgeschlossen unter der Hüllenbildung mit einer Zählersemikommutation. Wegen der Entscheidbarkeit des Leerheitsproblems für Multicountersprachen folgt:

Korollar 5.1 *[Gon93] Ist SD ist eine CSD, so sind für reguläre Stringsprachen* R_1, R_2 *die Fragen* $f_{SC}(R_1) \cap R_2 = \emptyset$*? und* $max_{SC}(R_1) = R_1$ *entscheidbar.*

Da die Komposition zweier CSD SD_1 und SD_2^{-1} ebenfalls eine CSD ist, gilt:

Korollar 5.2 *[Gon93] Sind* SD_1 *und* SD_2 *CSD's, so ist für reguläre Stringsprachen* R_1, R_2 *die Frage* $f_{SC_1}(R_1) \cap f_{SD_2}(R_2) = \emptyset$*? entscheidbar.*

5.2 Prioritätszählerhalbabhängigkeiten

Da wir nun mit dem Prioritätsmulticounterautomaten ein stärkeres Instrument zur Verfügung haben, definieren wir im folgenden die *Prioritätszählerhalbabhängigkeiten* (*PCSD*):

Definition Ein Relation SD ist eine PCSD, wenn sie die Eigenschaft hat, daß die Relationen $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ und $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'' \underset{SC \setminus SC^{-1}}{\Longrightarrow} w''' \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ in PMC liegen, d.h. die Relationen können von einem Prioritätsmulticounterautomaten mit 2 Eingabebändern erkannt werden.

Durch Raten eines Wortes w in der regulären Sprache R, gleichzeitige Simulation des endlichen Automaten für R auf w und gleichzeitige Simulation des Prioritätsmulticounterautomaten für $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ folgt:

Theorem 5.2 *Jede PCSD SD hat die Eigenschaft, daß* $f_{SC}(R)$ *und* $\{w \mid \exists u \in R\ u \overset{*}{\underset{SC}{\Longrightarrow}} w'' \underset{SC \setminus SC^{-1}}{\Longrightarrow} w' \overset{*}{\underset{SC}{\Longrightarrow}} w\}$ *für jede reguläre Stringsprache R in PMC liegen.*

Korollar 5.3 *Für jede PCSD SD ist für reguläre Stringsprachen* R_1, R_2 *die Frage* $f_{SC}(R_1) \cap R_2 = \emptyset$*? entscheidbar.*

Beweis: Wegen $f_{SC}(R_1) \in PMC$ und der Abgeschlossenheit unter Schnitt mit regulären Sprachen existiert ein Prioritätsmulticounterautomat, der $f_{SC}(R_1) \cap R_2$ erkennt und dessen Leerheitsproblem wegen Theorem 3.1 entscheidbar ist. ■

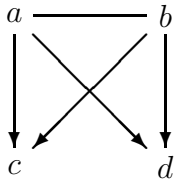
Korollar 5.4 *Für jede PCSD SD ist für eine reguläre Stringsprache L die Frage $max_{SC}(L) = L$ entscheidbar.*

Beweis: Durch Raten eines Wortes w in der regulären Sprache L, gleichzeitige Simulation des endlichen Automaten für L auf w und gleichzeitige Simulation des Prioritätsmulticounterautomaten für $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'' \underset{SC\setminus SC^{-1}}{\Longrightarrow} w''' \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ aus Theorem 5.2 folgt $f_{SC}(L') \in PMC$ für $L' := \{w \mid \exists u \in L\ u \overset{*}{\underset{SC\cap SC^{-1}}{\Longrightarrow}} v \underset{SC\setminus SC^{-1}}{\Longrightarrow} w\}$. Wegen der Abgeschlossenheit unter Schnitt mit regulären Sprachen existiert ein Prioritätsmulticounterautomat, der $f_{SC}(L') \cap L$ erkennt. Dessen Leerheitsproblem ist äquivalent zu der Frage $max_{SC}(L) = L$ und ist wegen Theorem 3.1 entscheidbar. ■

Bemerkung: Beim Beweis jeder der folgenden hinreichenden Bedingungen für PCSD's können wir uns o.B.d.A auf das Erkennen der Relation $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ beschränken.
Beweis: Es existiert, wie für jede der folgenden hinreichenden Bedingungen für PCSD's gezeigt wird, ein Prioritätsmulticounterautomat für die Sprache $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$, der für jedes Zeichen a zu jeder Zeit die Differenz der bisher gelesenen Vorkommen von a in w und w' durch Zählerinhalte repräsentiert.
Um $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'' \underset{SC\setminus SC^{-1}}{\Longrightarrow} w''' \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ zu erkennen, wird die Anzahl der Zustände dieses Prioritätsmulticounterautomat verdoppelt. Nur wenn eine asymmetrische Vertauschung stattfindet, kann von der ersten Hälfte der Zustände in die zweite Hälfte übergegangen und nur dort später akzeptiert werden. Das Auftreten einer asymmetrischen Vertauschung kann, wie bei der weiteren Konstruktion ersichtlich, durch das Überprüfen eines Zählers auf $\neq 0$ festgestellt werden. ■

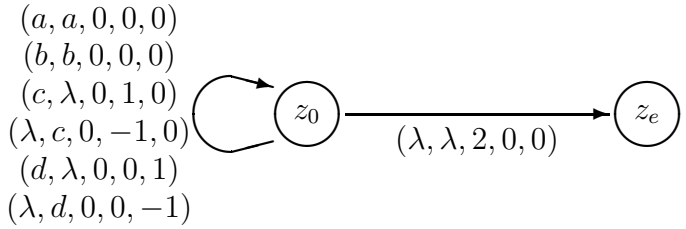
Korollar 5.5 *Jede CSD ist eine PCSD.*

Beispiel: Betrachten wir das folgende Halbabhängigkeitsalphabet SD:

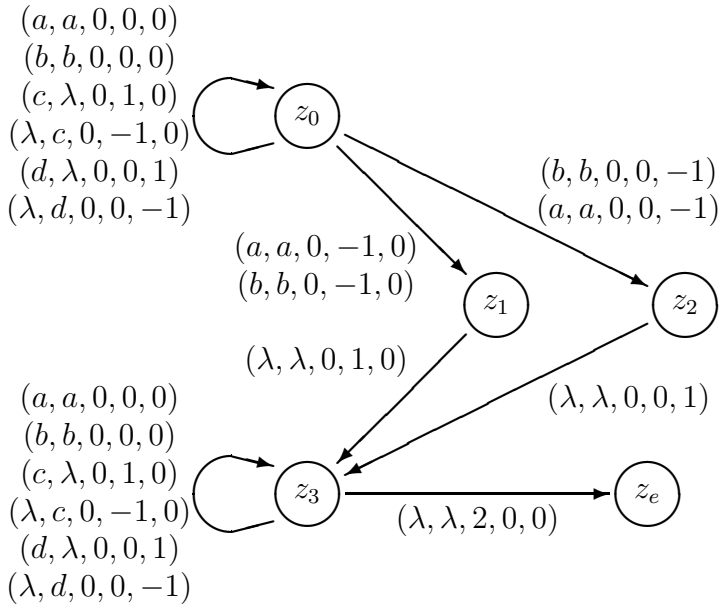


Es ist die Komposition der beiden atomaren Zählerhalbabhängigkeiten

und die Relation $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ kann von folgendem Prioritätsmulticounterautomaten erkannt werden:



Die erste Stelle der die Übergänge beschreibenden Tupel bezeichnet das gelesene Zeichen des ersten Eingabebandes, die zweite Stelle der Tupel bezeichnet das gelesene Zeichen des zweiten Eingabebandes, die dritte Stelle gibt die Zahl der auf Null getesteten Zähler an und alle weiteren Stellen bezeichnen die Veränderung der Zählerinhalte. Der angegebene Automat zählt die nach vorne bewegten c's bzw. d's in Zähler 1 bzw. 2; a's und b's können nur simultan auf beiden Bändern gelesen werden. Der Automat kann nur akzeptieren, wenn beide Zähler durch den Nulltest kommen (2 in dritter Stelle). Der Automat akzeptiert z.B. $(cdabdc, abdccd)$, nicht aber (ac, ca).
Um $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'' \underset{SC \setminus SC^{-1}}{\Longrightarrow} w''' \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ zu erkennen muß während einem simultanen Lesen von a oder b auf beiden Eingabebändern sichergestellt sein, daß einer der beiden Zähler nicht Null enthält, dies geschieht durch dekrementieren und anschließendes Inkrementieren in folgendem Automaten:

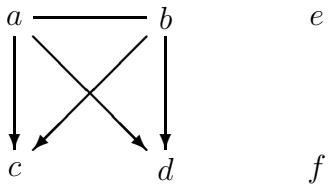


Der Automat akzeptiert z.B. $(acdcb, adcbc)$, nicht aber $(acdcb, adccb)$.

Theorem 5.3 *Ist SD eine PCSD über dem Alphabet A, so auch über $A' \supseteq A$.*

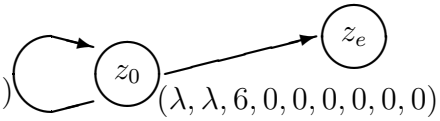
Beweis: Jeder zusätzliche Buchstabe in $A' \setminus A$ kann mit allen anderen Buchstaben kommutieren. Zu diesem Zweck werden jeweils 2 zusätzliche Zähler verwendet, mit deren Hilfe die Differenz der Anzahl der Vorkommen dieser Buchstaben in den bisher gelesenen Präfixen von w und w' gemerkt wird. Diese Zähler müssen nie zwischendurch auf 0 getestet werden. ■

Beispiel: Betrachten wir das folgende Halbabhängigkeitsalphabet SD:



Die Relation $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ kann nun von folgendem Prioritätsmulticounterautomaten erkannt werden:

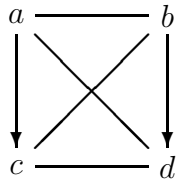
$$\begin{array}{ll}
(\lambda, e, 0, 0, 0, 1, 0, 0, 0) & (a, a, 0, 0, 0, 0, 0, 0, 0) \\
(e, \lambda, 0, 0, 0, 0, 1, 0, 0) & (b, b, 0, 0, 0, 0, 0, 0, 0) \\
(\lambda, \lambda, 0, 0, 0, -1, -1, 0, 0) & (c, \lambda, 0, 1, 0, 0, 0, 0, 0) \\
(\lambda, f, 0, 0, 0, 0, 0, 1, 0) & (\lambda, c, 0, -1, 0, 0, 0, 0, 0) \\
(f, \lambda, 0, 0, 0, 0, 0, 0, 1) & (d, \lambda, 0, 0, 1, 0, 0, 0, 0) \\
(\lambda, \lambda, 0, 0, 0, 0, 0, -1, -1) & (\lambda, d, 0, 0, -1, 0, 0, 0, 0)
\end{array}$$



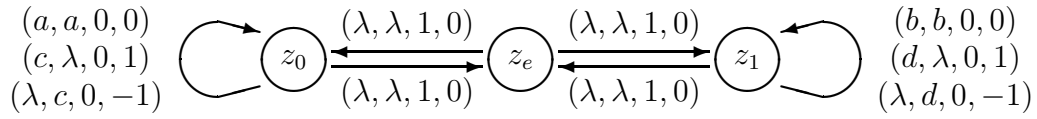
Theorem 5.4 *Seien SD_1 und SD_2 PCSD's über den disjunkten Alphabeten A_1 und A_2, so ist auch $SD = SD_1 \cup SD_2 \cup A_1 \times A_2 \cup A_2 \times A_1$ über $A = A_1 \cup A_2$ eine PCSD.*

Beweis: Ein Prioritätsmulticounterautomat simuliert auf A_1 den Prioritätsmulticounterautomaten für SD_1 und auf A_2 den Prioritätsmulticounterautomaten für SD_2. Beim Übergang zwischen diesen Simulationen werden alle Zähler auf 0 getestet. ■

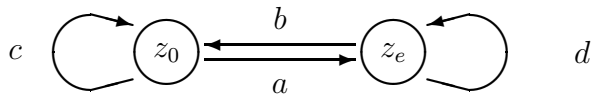
Beispiel: Betrachten wir das folgende Halbabhängigkeitsalphabet SD:



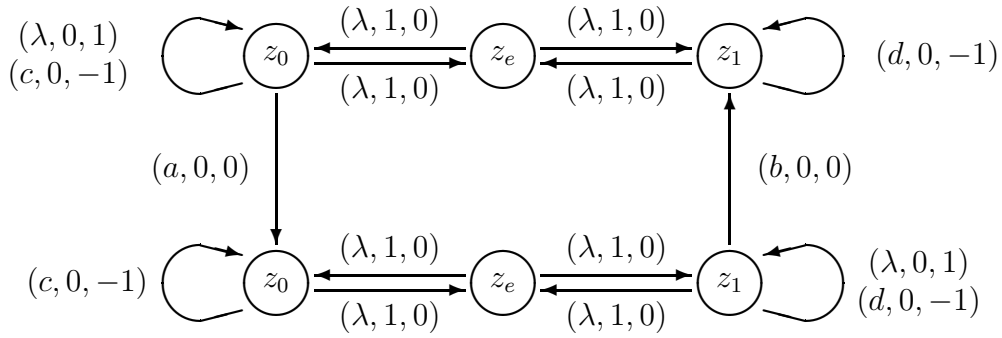
Der Prioritätsmulticounterautomat, der die Relation $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ erkennt, setzt sich aus dem Automaten für $SD_0 = \{(a, c)\}$, der die Zustände z_0 und z_e besitzt, und dem Automaten für $SD_1 = \{(b, d)\}$, der die Zustände z_1 und z_e besitzt, zusammen:



Die Sprache $L = c^*a(d^*bc^*a)^*$ wird vom endlichen Automaten



erkannt. Die Sprache $f_{SC}(L) = c^*ac^*(d^*bd^*c^*ac^*)^*$ wird von folgendem Prioritätsmulticounterautomaten gemäß Theorem 5.2 erkannt, welcher beide Automaten auf dem gleichen geratenen Eingabewort w simuliert, wodurch das zweite Eingabeband des simulierten Prioritätsmulticounterautomaten zum einzigen Eingabeband wird und die erste Komponente der Übergangstripel wegfällt.



Mit den soweit charakerisierten PCSD und Theorem 5.14 erhalten wir bereits einen alternativen Beweis für folgendes Resultat in [AH89]:

Theorem 5.5 *[AH89] Für ein symmetrisches Kommutationssystem C ist die Frage $f_C(R_1) \cap f_C(R_2) = \emptyset$? für reguläre Sprachen R_i genau dann entscheidbar, wenn $\overline{D}$* diagonal *ist, d.h. sind $(a, b), (b, c)$ und (c, d) in $\overline{D}$, so sind auch (a, c) oder (b, d) in $\overline{D}$.*

Beweis: Nach [Wol65] ist ein Graph genau dann diagonal, wenn er ein *transitiver Wald* ist, d.h. er entsteht aus einem Wald durch (transitives) Hinzufügen von Kanten von allen Knoten zu allen ihren Vorgängerknoten. Ein solcher transitiver Wald ist entweder nicht zusammenhängend (d.h. er enthält mehrere transitive Bäume) oder er besteht nur aus einem einzigen transitiven Baum, dessen Wurzelknoten mit allen anderen Knoten (welche ebenfalls einen transitiven Wald bilden) verbunden ist. Ist also $\overline{D}$ diagonal und somit ein transitiver Wald, so ist entweder $\overline{D}$ nicht zusammenhängend oder D enthält einen isolierten Knoten. Daraus läßt sich erkennen, daß D entweder wegen Theorem 5.4 (falls $\overline{D}$ nicht zusammenhängend ist) oder wegen Theorem 5.3 (falls D einen isolierten Knoten enthält) eine PCSD ist.
Ist $\overline{D}$ nicht diagonal, so besitzt D einen induzierten Teilgraphen der Form • — • — • — • oder der Form • — • • — •. Aufgrund der Unentscheidbarkeitsbedingung 1 in Theorem 5.14 ist die Frage $f_C(R_1) \cap R_2 = \emptyset$? und somit die Frage $f_C(R_1) \cap f_C(R_2) = \emptyset$? für reguläre Sprachen R_i in diesem Fall unentscheidbar. ■

Bemerkung: Nach [Wol65] gilt also für jeden nichtleeren, symmetrischen Graphen D:

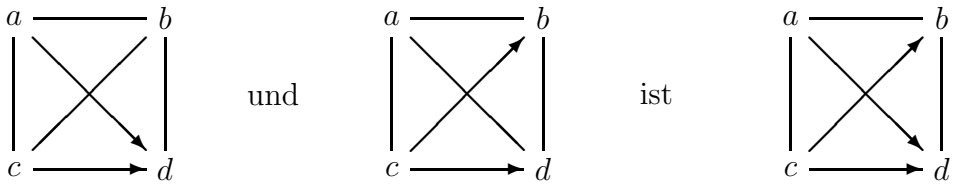
1. $\overline{D}$ ist nicht zusammenhängend oder
2. D enthält einen induzierten Teilgraphen der Form • — • — • — • oder
3. D enthält einen induzierten Teilgraphen der Form • — • • — • oder
4. D enthält einen isolierten Knoten.

In [Sei74] und unabhängig davon in [Teo93] wird gezeigt, daß die Graphen, die keinen induzierten Teilgraphen der Form • — • • — • besitzen, gerade die *Cographen* sind, d.h. die Graphen, die nur aus einem Einzelknotnen bestehen oder durch Komplementbildung und disjunkte Vereinigung anderer Cographen gebildet werden können. Somit gilt für jeden nichtleeren, symmetrischen Graphen D:

1. $\overline{D}$ ist nicht zusammenhängend oder
2. D enthält einen induzierten Teilgraphen der Form • — • — • — • oder
3. D ist nicht zusammenhängend oder
4. D besteht aus einem einzigen Knoten.

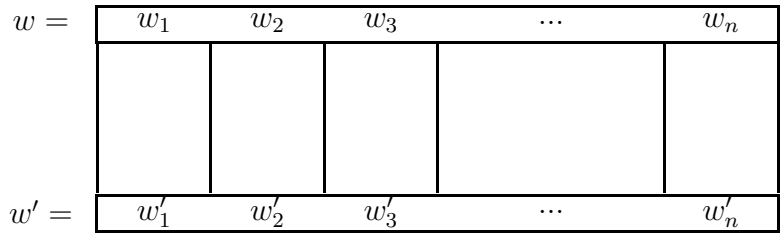
Es ist leicht zu sehen, daß die beiden Resultate direkt auseinander folgen:
Enthält D einen induzierten Teilgraphen der Form • — • • — •, so ist D entweder nicht zusammenhängend oder mit der kürzesten Verbindung wird ein induzierter Teilgraph der Form • — • — • — • gebildet. Enthält D einen isolierten Knoten, so ist D nicht zusammenhängend oder besteht aus einem einzigen Knoten. Ist $\overline{D}$ nicht zusammenhängend, so enthält D einen isolierten Knoten oder die Zusammenhangskomponenten (mindestes zwei) sind größer und somit enthält D einen induzierten Teilgraphen der Form • — • • — •.

Wir wollen nun noch weitere PCSD's finden. Leider sind die PCSD's nicht unter der gleichen Komposition wie allgemeine Semikommutationssysteme abgeschlossen, wie folgendes Beispiel zeigt: Die Komposition der beiden PCSD's



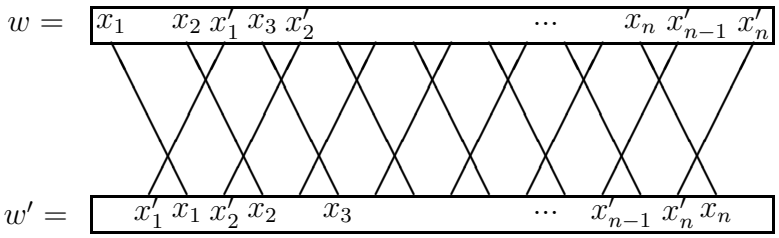
und nach Theorem 5.9 keine PCSD. Aus diesem Grund definieren wir nun eine Verschärfung der Kompositionsbedingung in [RW91]:

Definition $SD = SD_1 \cap SD_2$ ist die *starke Komposition* von SD_1 und SD_2 über dem Alphabet A, wenn $\forall w, w'$ mit $w' \in f_{SC}(w)$ $\exists w_1 ... w_n = w, w'_1 ... w'_n = w'$ $\forall i \leq n$ $w'_i \in f_{SC_1}(w_i) \vee w'_i \in f_{SC_2}(w_i)$.



Theorem 5.6 *$SD = SD_1 \cap SD_2$ ist die* starke Komposition *von SD_1 und SD_2 über dem Alphabet A, gdw* $\neg\exists x_1, x'_1, ...x_n, x'_n \in A$ $((x_1, x'_1) \in SD_1 \backslash SD_2 \vee (x_2, x'_1) \in SD_1 \setminus SD_2) \wedge ((x_n, x'_n) \in SD_2 \setminus SD_1 \vee (x_n, x'_{n-1}) \notin SD_2 \setminus SD_1) \wedge \forall i < n$ $((x_i, x'_i) \notin SD \wedge (x_{i+1}, x'_i) \notin SD)$.

Beweis: Müssen zwei Paare (x_1, x'_1) bzw.(x_2, x'_1) und (x_n, x'_n) bzw.(x_n, x'_{n-1}) in verschieden Semikommutationssysteme abgeleitet werden, so ist das dann und nur dann möglich, wenn keine Verschränkung der folgenden Art vorliegt:



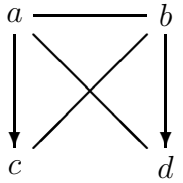
■

Korollar 5.6 *$SD = SD_1 \cap SD_2$ ist die* starke Komposition *von SD_1 und SD_2 über dem Alphabet A, wenn* $\forall a, b, c \in A$ $(a, b) \in SD_1 \vee ((a, c) \in SD_2 \wedge (c, b) \in SD_2)$.

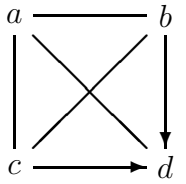
Theorem 5.7 *Seien SD_1 und SD_2 PCSD's über dem Alphabet A, so ist auch die* starke Komposition *von SD_1 und SD_2 über dem Alphabet A (falls diese existiert) eine PCSD.*

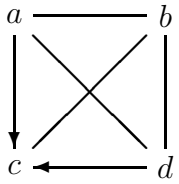
Beweis: Ein Prioritätsmulticounterautomat simuliert nichtdeterministisch entweder den Prioritätsmulticounterautomaten für SD_1 oder den Prioritätsmulticounterautomaten für SD_2. Beim Übergang zwischen den Simulationen werden alle Zähler auf 0 getestet. ■

Beispiel: Betrachten wir das folgende Halbabhängigkeitsalphabet SD:



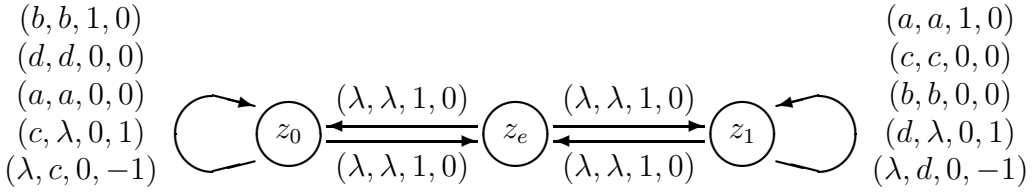
Es ist die starke Komposition der beiden PCSD's

$SD_1 :=$ 

und 

$=: SD_0$

Der Prioritätsmulticounterautomat, der die Relation $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ erkennt, setzt sich wieder aus den Automaten für SD_1 und SD_2 zusammen:



5.2.1 Hinreichendes Kriterium

Auch mit der starken Komposition werden noch nicht alle PCSD's erfasst, da dabei nicht die Möglichkeit besteht den Inhalt eines Zählers, der in beiden Simulationen die gleiche Aufgabe erfüllt, beizubehalten. Eine allgemeinere Definition, welche dies ermöglicht, ist folgende:

Theorem 5.8 *Eine Relation SD über dem Alphabet A ist eine PCSD, wenn*

$$
\begin{aligned}
&\exists\ s \in \mathbb{N}, C_1, B_1, D_1, ..., C_s, B_s, D_s \subseteq A \text{ und } g \in \mathbb{N}^{\{l,r\}\times A} \text{ injektiv}\\
&\quad \text{mit } \overline{SD} = \bigcup_{i\le s} \overline{SD_i} \text{ mit}\\
&\quad \forall i \le s\ \overline{SD_i} = (C_i \times B_i \cup D_i \times C_i) \setminus id_A \text{ und } B_i \cup D_i \subseteq C_i \text{ und}\\
&\quad \forall i, j \le s\ \forall b \in B_i \setminus B_j, d \in D_i \setminus D_j, a \in B_i \cap B_j, e \in D_i \cap D_j\\
&\qquad\qquad \longrightarrow max(g(l,b), g(r,d)) < min(g(l,a), g(r,e)) \text{ und}\\
&\qquad \forall b, d, e \in A\ \ (d,b) \in \overline{SD_i}\ \wedge\ (d,e) \in \overline{SD_j}\\
&\qquad\qquad\qquad \longrightarrow d \in D_i \cap D_j\ \vee\ b, e \in B_i \cap B_j \text{ und}\\
&\qquad \forall d, b, e \in A\ (d,b) \in \overline{SD_i}\ \wedge\ (e,b) \in \overline{SD_j}\\
&\qquad\qquad\qquad \longrightarrow d, e \in D_i \cap D_j\ \vee\ b \in B_i \cap B_j.
\end{aligned}
$$

Beweis: Anschaulich gesehen ist klar: Wenn ein $w \in A^*$ abgeleitet werden kann zu w^R, so müssen alle dabei beteiligten Kommutationen in einem $\overline{SD_i}$ liegen. Dies sagt folgender Hilfssatz:
Hilfssatz: Für jede geordnete Menge $\{a_1, ..., a_n\}$ mit

$$S(a_1, ..., a_n) := \{(a_i, a_j) \mid i < j \le n\} \subseteq \overline{SD}$$

existiert ein Index $i \le s$ mit $S(a_1, ..., a_n) \subseteq \overline{SD_i}$.
Der Beweis durch Induktion: Die Behauptung ist klar für $\emptyset, \{a_1\}, \{a_1, a_2\}$. Sei

$$I := \{i \le s \mid \exists j < n\ (a_j, a_n) \in \overline{SD_i}\},$$

so folgt aus

$$\{a_1...a_{n-1}\} \times \{a_n\} \subseteq \bigcup_{i \in I} \overline{SD_i}$$

entweder $\forall i \in I\ a_n \in B_i$ oder $\exists i \in I\ a_n \notin B_i$, woraus wegen

$$\forall a_j, a_n \text{ mit } j,k < n\ \forall l \in I\ (a_j, a_n) \in \overline{SD_l}\ \wedge\ (a_k, a_n) \in \overline{SD_i} \\ \longrightarrow a_j, a_k \in D_l \cap D_i\ \vee\ a_n \in B_l \cap B_i$$

$\exists i \in I\ \{a_1...a_{n-1}\} \subseteq D_i\ \wedge\ a_n \in C_i$ folgt. Im zweiten Fall ist die Behauptung bereits erfüllt. Im ersten Fall gilt mit analoger Argumentation $\forall i \in I'\ a_1 \in D_i$ mit $I' := \{i \leq s \mid \exists 1 < j \leq n\ (a_1, a_j) \in \overline{SD_i}\}$. Daher existiert ein $i_0 \in I \cap I'$ mit $a_1 \in D_{i_0}$ und $a_n \in B_{i_0}$. Nach Induktion existiert ein $j\ (\in I)$ mit $S(a_2, ..., a_n) \subseteq \overline{SD_j}$ und ein $k\ (\in I')$ mit $S(a_1, ..., a_{n-1}) \subseteq \overline{SD_k}$. Sei

$$m := min\{l \mid 2 \leq l \leq n-1, a_l \notin D_j \cap D_k\},$$

so folgt daraus für alle m' mit $m < m' < n$ wegen $(a_m, a_{m'}) \in \overline{SD_j} \cap \overline{SD_k}$, daß $m' \in B_j \cap B_k$. Es gilt also

$$a_2, ..., a_{m-1} \in D_j \cap D_k \text{ und } a_{m+1}, ..., a_{n-1} \in B_j \cap B_k$$

und außerdem gilt $a_1 \in D_k$ und $a_n \in B_j$.
Sei o.B.d.A. $g(r, a_1) > g(l, a_n)$. Es muß ein $h \in I$ existieren mit $a_n \in B_h$ und $a_m \in C_h$. Wäre $a_1 \in D_{i_0} \setminus D_h$, so würde nach Definition aus $a_n \in B_{i_0} \cap B_h$ die falsche Ungleichung $g(r, a_1) < g(l, a_n)$ folgen. Folglich muß auch $a_1 \in D_h$ gelten. Existiert ein $a_p \in D_k \setminus D_h$ mit $2 \leq p \leq m-1$, so gilt wegen $a_1 \in D_k \cap D_h$ die Ungleichung $g(r, a_1) > g(r, a_p)$ und somit ist $a_1 \in D_j$ (sonst würde aus $a_1 \in D_k \setminus D_j$ und $a_p \in D_k \cap D_j$ wieder die gegenteilige Ungleichung gelten) und j ist der gesuchte Index i des Hilfssatzes. Existiert ein $a_p \notin B_h$ mit $m+1 \leq p \leq n-1$, so gilt analog dazu $g(l, a_n) < g(l, a_p)$ und somit ist $a_n \in B_k$ und k ist der gesuchte Index i des Hilfssatzes. Ansonsten ist h der gesuchte Index i des Hilfssatzes. □

Ein Prioritätsmulticounterautomat M mit s Zuständen und $max\{g(x, a) \mid x \in \{l, r\}, a \in A\}$ Zählern erkennt $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ wie folgt:
Befindet sich M im Zustand q_i und hat M den Präfix v von w und den Präfix v' von w' gelesen, so dürfen nur die Zähler mit den Nummern $g(l, b)$ und $g(r, d)$ für alle $b \in B_i$ und $d \in D_i$ einen Wert $c(g(l, b))$ bzw. $c(g(r, d))$ größer als 0 haben (alle anderen müssen den Wert 0 haben) und es gilt

$$\forall a \in A\ |v|_a + c(g(l, a)) = |v'|_a + c(g(r, a))$$

. Im Zähler $g(l, a)$ wird also die Anzahl der sich nach links bewegenden a's gezählt. Die Zählerinhalte $c(g(l, a))$ und $c(g(r, a))$ können jederzeit simultan dekrementiert werden.

Geht M in den Zustand q_j über, so müssen dabei alle Zähler mit Nummern aus

$$\{g(l,b) \mid b \in B_i \setminus B_j\} \cup \{g(r,d) \mid d \in D_i \setminus D_j\}$$

auf 0 getestet werden, was möglich ist, weil diese Zählernummern gemäß Definition kleiner als die in

$$\{g(l,b) \mid b \in B_i \cap B_j\} \cup \{g(r,d) \mid d \in D_i \cap D_j)$$

sind. Sind alle Zähler 0, so kann der Automat die gelesene Eingabe akzeptieren.
A kann im Zustand q_i ein Zeichen $d \in D_i$ auf dem ersten Eingabeband überlesen und dabei $c(g(r,d))$ inkrementieren oder ein Zeichen $b \in B_i$ auf dem zweiten Eingabeband überlesen und dabei $c(g(l,b))$ inkrementieren oder ein Zeichen $b \in B_i$ auf dem ersten Eingabeband überlesen und dabei $c(g(l,b))$ dekrementieren oder ein Zeichen $d \in D_i$ auf dem zweiten Eingabeband überlesen und dabei $c(g(r,d))$ dekrementieren oder ein Zeichen $c \in C_i$ simultan auf beiden Eingabebändern überlesen.
Damit wird garantiert, daß A nur solche Paare w, w' akzeptiert, bei denen bei der Ableitung von w nach w' Zeichen in einem D_i bzw. B_i nur über solche Zeichen nach vorne bzw. nach hinten vertauscht wurden, mit denen sie nicht abhängig sind.
Es ist nun noch zu zeigen, daß w' nicht von w abgeleitet werden kann, wenn M zwischendurch nicht weiterlesen kann.
Sei $B := \{b \in A \mid |v|_b < |v'|_b\}$ die Menge der Zeichen, welche im bisher gelesenen Teil von w seltener vorkommen als im bisher gelesenen Teil von w' und $D := \{d \in A \mid |v|_d > |v'|_d\}$ entsprechend umgekehrt. Für den Fall, daß w (bzw. w') bereits zu Ende gelesen wurde und M nicht weiterlesen kann, muß noch ein Zeichen, welches nicht in D (bzw. B) ist, im Rest des jeweils anderen Wortes stehen, dann kann w' aber nicht von w ableitbar sein, da die Anzahl dieses Zeichens nicht in w und w' übereinstimmt.
Seien also $x, x' \in A$ mit $w = vxu$ und $w' = v'x'u'$ die jeweils nächsten Zeichen. Mittels λ-Übergängen sind alle Zustände q_i mit $D \subseteq D_i$ und $B \subseteq B_i$ erreichbar, da die Zähler in $\{g(r,d), g(l,b) \mid d \in D, b \in B\}$ bei einem solchen Übergang nicht auf 0 zu testen sind. Ferner existiert mindestens ein solches q_m. Da Zeichen aus D von v nach u' und Zeichen aus B von u nach v' gelangen müssen, gilt

$$D \times \{x'\} \cup \{x\} \times B \subseteq \overline{SD}.$$

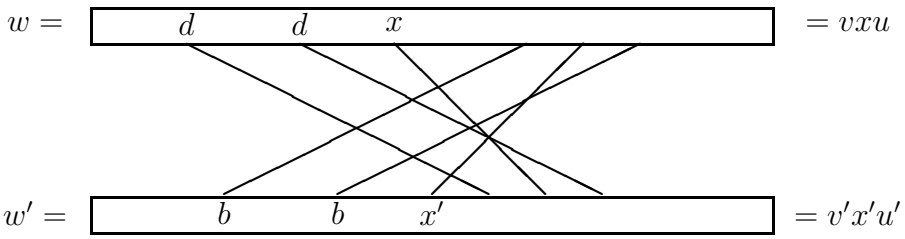
Wäre $x = x'$, so existiert gemäß obenstehendem Hilfssatz (für eine geordnete Menge $\{a_1, ..., a_n\} = D \cup \{x\} \cup B$ mit $x = a_m$ und $a_i \in D$ für $i < m$ und $a_i \in B$ für $i > m$) ein i mit

$$S(D, x, B) \subseteq \overline{SD_i}$$

und so könnte M das Zeichen x simultan auf beiden Bändern lesen.

Sonst gilt $x, x' \notin D \cup B$, da M sonst x oder x' lesen könnte, daher gilt $(x, x') \in \overline{SD}$. Insgesamt gilt also

$$D^2,\ B^2,\ (D \cup \{x\}) \times (B \cup \{x'\}) \subseteq \overline{SD} \cup id.$$



(1.) Der Fall $\exists d_1 \neq d_2, b_1 \neq b_2$

$$(d_1, d_2) \in (D \cup \{x\})^2 \cap SD \ \wedge \ (b_1, b_2) \in (B \cup \{x'\})^2 \cap SD$$

kann aus folgenden Grund nicht auftreten: Aus $(d_1, b_1) \in \overline{SD_i} \wedge (d_1, b_2) \in \overline{SD_j} \wedge (d_2, b_2) \in \overline{SD_k}$ für gewisse i, j, k folgt, daß $d_1 \in D_i \cap D_j \ \vee \ b_1, b_2 \in B_i \cap B_j$ und, wegen $(b_1, b_2) \in (B \cup \{x'\})^2 \cap SD$ muß $d_1 \in D_i \cap D_j$ gelten. Analog dazu gilt auch $b_2 \in B_j \cap B_k$. Es gilt aber $d_1 \notin D_k$ und $b_2 \notin B_i$, d.h. $d_1 \in D_j \setminus D_k$ und $b_2 \in B_j \setminus B_i$. Also müsste $g(r, d_1) < g(l, b_2)$ und $g(l, b_2) < g(r, d_1)$ gelten, was ein Widerspruch ist.
(2.) Für den Fall

$$(D \cup \{x\})^2 \cap SD = (B \cup \{x'\})^2 \cap SD = \emptyset$$

existiert gemäß dem Hilfssatz ein i mit

$$S(D, x, x', B) \subseteq \overline{SD_i}$$

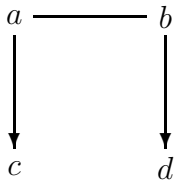
und falls $D \cup \{x\} \subseteq D_i$ oder $B \cup \{x'\} \subset B_i$, so kann M entweder x oder x' lesen. Existiert jedoch o.B.d.A ein $d \in D \cup \{x\} \setminus D_i$, so gilt $x' \in B_i$ wegen $(d, x') \in \overline{SD_i}$ und somit kann M x' lesen.
(3.) Bleibt also o.B.d.A. der Fall

$$(D \cup \{x\})^2 \cap SD = \emptyset \ \wedge \ (b_1, b_2) \in (B \cup \{x'\})^2 \cap SD.$$

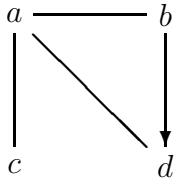
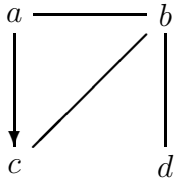
Gemäß dem Hilfssatz existiert ein $i \leq s$ mit $S(D, x, B) \subseteq \overline{SD_i}$ und ein j mit $S(D, x, x') \subseteq \overline{SD_j}$. Aus $(x, b_1) \in \overline{SD_i} \wedge (x, b_2) \in \overline{SD_j}$ oder $(x, b_2) \in \overline{SD_i} \ \wedge \ (x, b_1) \in \overline{SD_j}$ folgt $x \in D_i \cap D_j \ \vee \ b_1, b_2 \in B_i \cap B_j$. Und da wegen $(b_1, b_2) \in (B \cup \{x'\})^2 \cap SD$ die zweite Möglichkeit wegfällt, kann M im Zustand q_i das Zeichen x lesen.

■

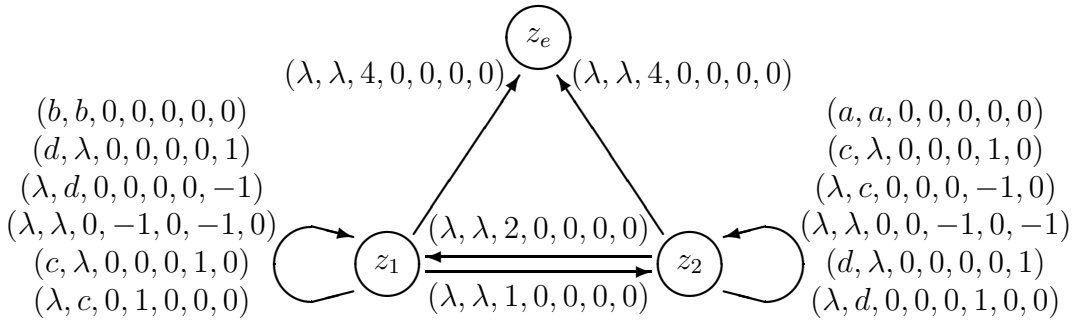
Beispiel: Betrachten wir das folgende Halbabhängigkeitsalphabet SD:



Es ist der Schnitt der beiden PCSD's

$SD_1 :=$  und  $=: SD_2$

mit $C_1 = \{b, c, d\}, B_1 = \{c\}, D_1 = D_2 = \{c, d\}, B_2 = \{d\}, C_2 = \{a, c, d\}$ und $g(l, c) = 1, g(l, d) = 2, g(r, c) = 3, g(r, d) = 4$. Damit ergibt sich nach der Konstruktion in Theorem 5.8 der folgende Prioritätsmulticounterautomat, der die Relation $\{(w, w') \mid w \overset{*}{\underset{SC}{\Longrightarrow}} w'\}$ erkennt:



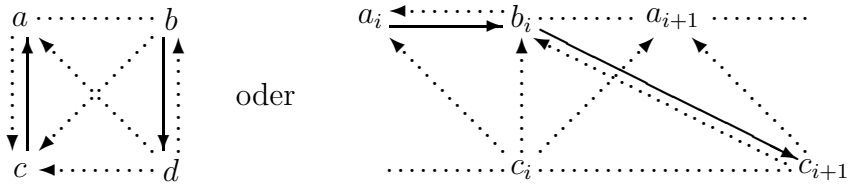
Bemerkung: Bei dem so konstruierten Prioritätsmulticounterautomat lassen sich auch leicht durch Hinzufügen weiterer Zustände die λ-Übergänge vermeiden. Der in Theorem 5.8 konstruierte Prioritätsmulticounterautomat hat die Eigenschaft, daß von jedem Zustand in jeden anderen Zustand übergegangen werden kann, wenn die entsprechende Nulltestbedingung erfüllt ist und daß der Verwendungszweck für jeden Zähler eineindeutig ist. Es ist nicht bekannt, ob mit dieser Eigenschaft eine Einschränkung der charakterisierten PCSD's verbunden ist.

5.2.2 Notwendiges Kriterium

Eine notwendige Bedingung für eine PMCD gibt der folgende Satz:

Theorem 5.9 *SD ist keine PCSD, wenn SD einen induzierten Teilgraphen der Form $(\{a, b, c, d\}, E)$ mit $\{(c, a), (b, d)\} \subseteq E \subseteq \overline{\{(a, d), (c, d), (c, b)\}}$ mit $|E| = 4$ oder einen induzierten Teilgraphen der Form $(\{a_1, b_1, c_1, ..., a_n = a_0, b_n = b_0, c_n = c_0\}, E)$ oder $(\{a_1, b_1, c_1, ..., a_n = a_0, b_n = b_0, c_n = c_0\}, E^{-1})$ mit $\{(a_i, b_i), (b_i, c_{i+1}) \mid i < n\} \subseteq E \subseteq \overline{\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}}$ enthält.*

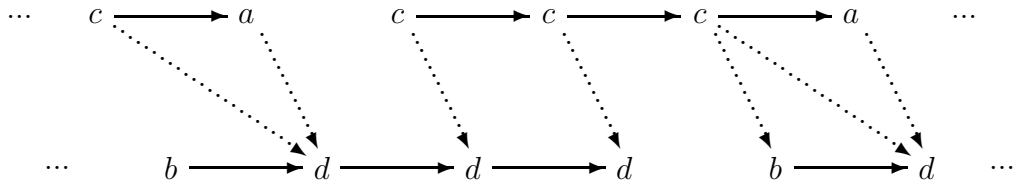
Das negative Kriterium sind also die Teilgraphen



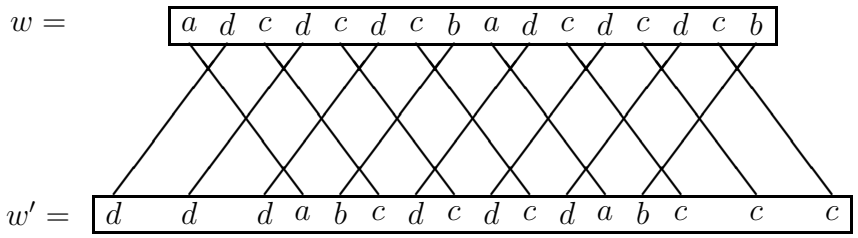
wobei gepunktete Linien eventuelle (nicht störende) Abhängigkeiten beschreiben.
Beweis: Angenommen SD ist eine PCSD und zu jeder regulären Sprache R gäbe es somit einen Prioritätsmulticounterautomaten, der $f_{SC}(R)$ erkennt, so können folgende Fälle auftreten:
(1.) Enthält SD einen induzierten Teilgraphen der Form $(\{a,b,c,d\},E)$ mit $\{(c,a),(b,d)\} \subseteq E \subseteq \overline{\{(a,d),(c,d),(c,b)\}}$, so könnte ein Prioritätsmulticounterautomat die Sprache

$$\begin{aligned} & f_{SC}((a(dc)^*b)^+) \ \cap \ d^*a(b(cd)^*a)^*bc^* \ \cap \\ & \{d^{n_1}ab(cd)^{n_2}ab(cd)^{n_3}... \mid \forall j \ \sum_{i=1}^{j} n_{2i-1} - n_{2i} \geq 0\} \cap \\ & \{d^{n_1}ab(cd)^{n_2}ab(cd)^{n_3}... \mid \forall j \ \sum_{i=1}^{j} n_{2i} - n_{2i+1} \geq 0\} \\ = \ & \{d^n a(b(cd)^n a)^m bc^n \mid n,m \in \mathbb{N}\} \end{aligned}$$

erkennen, indem er zusätzlich einen endlichen Automaten und zwei Ein-Zählerautomaten (ohne Nulltest) simuliert. Dies ist aber ein Widerspruch, da diese rational äquivalent zu der Sprache $\{(a^nb)^m \mid n,m \in \mathbb{N}\}$ ist, welche wegen Theorem 3.2 nicht von einem Prioritätsmulticounterautomat erkannt werden kann.
Ein Teilstück mitten aus der Spur $[w\rangle$ mit $w \in (a(dc)^*b)^+$ kann dabei z.B. die Form



haben, wobei die gestrichelten Pfeile mögliche weiche Kanten darstellen, welche bei der durch



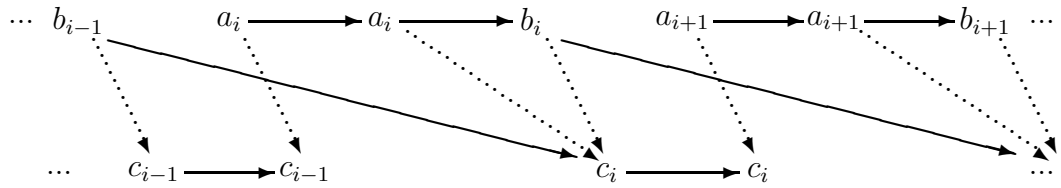
dargestellten Ableitung zu einem Wort in $d^*a(b(cd)^*d)^*c^*$ umgedreht werden müssen.
Durch den Schnitt mit den Sprachen

$$\{d^{n_1}ab(cd)^{n_2}ab(cd)^{n_3}... \mid \forall j \sum_{i=1}^{j} n_{2i-1} - n_{2i} \geq 0\} \text{ und}$$
$$\{d^{n_1}ab(cd)^{n_2}ab(cd)^{n_3}... \mid \forall j \sum_{i=1}^{j} n_{2i} - n_{2i+1} \geq 0\},$$

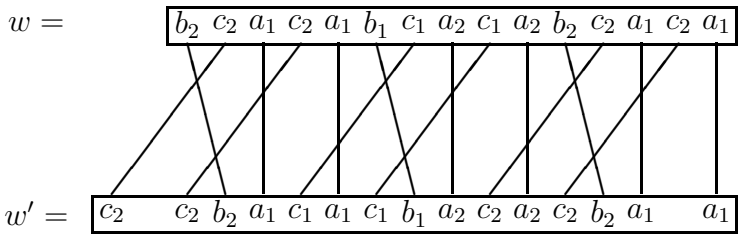
welche jeweils mit einem Zähler ohne Nulltest erkannt werden können, wird erreicht, daß sich ein d nicht vorwärts über ein b und ein c nicht rückwärts über ein a bewegt werden kann, obwohl das Semikommutationssystem dies zulassen würde. Auf diese Weise kann die gleiche Länge der Blöcke erzwungen werden.
(2.) Enthält SD einen induzierten Teilgraphen der Form
$(\{a_1, b_1, c_1, ..., a_n = a_0, b_n = b_0, c_n = c_0\}, E)$ mit
$\{(a_i, b_i), (b_i, c_{i+1}) \mid i < n\} \subseteq E \subseteq \overline{\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}}$, so könnte ein Prioritätsmulticounterautomat die Sprache

$$\begin{aligned}
& f_{SC}(b_n((c_na_1)^+b_1(c_1a_2)^+b_2...(c_{n-1}a_n)^+b_n)^+(c_na_1)^+) \cap \\
& c_n^+b_n((a_1c_1)^+b_1(a_2c_2)^+b_2...(a_nc_n)^+b_n)^+a_1^+ \cap \\
& \{c_n^{n_1}b_n(a_1c_1)^{n_2}b_1(a_2c_2)^{n_3}b_2... \mid \forall j \sum_{i=1}^{j} n_{2i-1} - n_{2i} \geq 0\} \cap \\
& \{c_n^{n_1}b_n(a_1c_1)^{n_2}b_1(a_2c_2)^{n_3}b_2... \mid \forall j \sum_{i=1}^{j} n_{2i} - n_{2i+1} \geq 0\} \\
= \quad & \{c_n^kb_n((a_1c_1)^kb_1(a_2c_2)^kb_2...(a_nc_n)^kb_n)^ma_1^k \mid m, k \in \mathbb{N}\}
\end{aligned}$$

erkennen, was ein Widerspruch ist, da diese ebenfalls rational äquivalent zu der Sprache $\{(a^nb)^m \mid n, m \in \mathbb{N}\}$ ist.
Ein Teilstück mitten aus der Spur $[w\rangle$ mit
$w \in b_n(c_na_1)^+b_1(c_1a_2)^+b_2...(c_{n-1}a_n)^+b_n)^+(c_na_1)^+$ kann dabei z.B. die Form

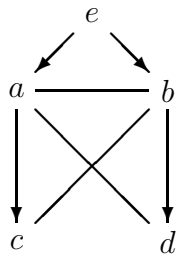


haben.

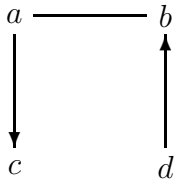


Durch die Verwendung der beiden Zähler ohne Nulltest wird in diesem Fall erreicht, daß sich ein c_i nicht vorwärts über ein b_{i+1} und ein a_{i+1} nicht rückwärts über ein b_i bewegt werden kann, obwohl das Semikommutationssystem dies zulassen würde. ■

Völlig unklar ist, ob $SD_3 :=$



eine PCSD ist. Keine PCSD nach Theorem 5.9 ist $SD_4 :=$



Für eine reguläre Sprache R kann $f_{SC_4}(R)$ bzw. $f_{SC_3}(R)$ von einem blinden Zählerautomaten erkannt werden, der auf zwei bzw. drei Zählern unabhängig einen Natürlichkeitstest durchführen kann. Unter der Annahme, daß das Halteproblem für diese Automaten entscheidbar ist, sind auch die Fragen $f_{SC}(R_1) \cap R_2 = \emptyset?$ und $max_{SC}(R_1) = R_1$ für $SD = SD_4$ bzw. $SD = SD_3$ entscheidbar.

5.3 Unentscheidbare Fälle

Mit Theorem 5.9 ist noch nicht bewiesen, daß die Fragen $f_{SC}(R_1) \cap R_2 = \emptyset?$ und $max_{SC}(R_1) = R_1$ für diese Halbabhängigkeitsalphabete nicht entscheidbar sind. Für eine stärkere Einschränkung des induzierten Teilgraphen, bei der die Verwendung der beiden Zähler wie im Beweis von Theorem 5.9 nicht nötig ist, weil das Semikommutationssystem die unerwünschten Verschiebungen ohnehin nicht zuläßt, sind beide Fragen unentscheidbar. Zum Beweis wird folgendes Lemma benötigt:

Lemma 5.1 *Für die Eingabe $R \subset \mathbb{N}^3$, $|R| < \infty$ und $a, e \in \mathbb{N}$ ist nicht entscheidbar, ob $m, a = n_1, ..., n_m = e \in \mathbb{N}$ existieren mit*

$$\forall i < m \exists (r_1, r_2, r_3) \in R \; n_i/r_1 = n_{i+1}/r_2 \in \mathbb{N} \; \wedge \; \neg n_i/r_3 \in \mathbb{N}.$$

Beweis: Bekannt ist aus [Min71] die Unentscheidbarkeit des Halteproblems für Multicounterautomaten mit Nulltest[3]. Dieses werden wir nun mittels Primzahl-

[3]Ein Zähler kann die Codierung eines Halbbandes einer Turingmaschine enthalten.

codierung der Konfigurationen reduzieren. Im Gegensatz zur Reduktion auf eine Ein-Registermaschine in [Min71] muß nun auch der Zustand mitcodiert werden. Gegeben sei ein Multicounterautomat $A = (k, Z, \delta, z_0, z_t)$ mit k Zählern, der Zustandsmenge $Z = \{z_0, z_1, ...z_t\}$, der Übergangsrelation

$$\delta \subseteq (Z \times \{1, ...k\}) \times (Z \times \{0,1\}^k \times \{0,1\}^k),$$

dem Anfangszustand z_0, dem Endzustand z_t, der Konfigurationenmenge $C_A = Z \times \mathbb{N}^k$, der Anfangs- und Endkonfiguration $\langle z_0, 0^k\rangle$ und $\langle z_t, 0^k\rangle$ und der Konfigurationsübergangsrelation

$$\langle z, c+d\rangle \underset{A}{\vdash} \langle z', c+i\rangle \text{ gdw. } z, z' \in Z, \exists j \leq k\ \langle (z,j),(z',i,d)\rangle \in \delta \text{ und } n_j = 0,$$

d.h. j ist die Nummer des Zählers, der bei diesem Übergang von z nach z' auf Null getestet wird und i bzw. d sind Vektoren in $\{0,1\}^k$ um die die Zählerinhalte inkrementiert bzw. dekerementiert werden. (Wir können o.B.d.A. davon ausgehen, daß A seine Zähler auf 0 bringt, bevor er akzeptiert. Will A in einem Schritt mehrere Zähler auf 0 testen, so kann er dies auch sequentiell mit zusätzlichen Zwischenzuständen tun. Will A keinen Zähler auf 0 testen, so kann er einen Zähler, der nie verändert wird, auf 0 testen.)
Seien $p_0, p_1, ..., p_{k+t}$ die ersten $k+t+1$ Primzahlen. Wir wollen nun die Konfigurationen $\langle z_{k_l}, c_l\rangle$ des Automaten in Primzahlprodukte $n_l = p_{k_l} \prod_{h=1}^{k} p_{t+h}^{c_l(h)}$ codieren. Setze $a := p_0$ für die Anfangskonfiguration $\langle z_0, 0^k\rangle$, $e := p_t$ für die Endkonfiguration $\langle z_t, 0^k\rangle$ und $R :=$

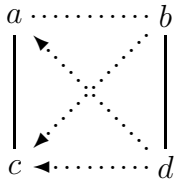
$$\{(r_1, r_2, p_{t+j}) \mid \langle (z_v, j), (z_n, i, d)\rangle \in \delta\ \wedge\ r_1 = p_v \prod_{h=1}^{k} p_{t+h}^{d(h)}\ \wedge\ r_2 = p_n \prod_{h=1}^{k} p_{t+h}^{i(h)}\}.$$

Die Division einer Konfigurationscodierung durch r_1 bzw. durch r_2 entspricht dem Entfernen des Zustandes und des Inkrements der Zähler (bzw. in umgekehrter Weise des Dekrements); die Nichtteilbarkeit durch $r_3 = p_{t+j}$ entspricht dem Nulltest des j-ten Zählers. Eine Konfigurationsfolge $\langle z_{k_1}, c_1\rangle = \langle z_0, 0^k\rangle, \langle z_{k_2}, c_2\rangle$ bis $\langle z_{k_m}, c_m\rangle = \langle z_t, 0^k\rangle$ existiert genau dann wenn $a = n_1, ..., n_m = e \in \mathbb{N}$ existieren mit $n_l = p_{k_l} \prod_{h=1}^{k} p_{t+h}^{c_l(h)}$ für alle $1 \leq l \leq m$. ∎

Wir können nun verschiedene Kriterien für die Unentscheidbarkeit der Frage $max_{SC}(L) = L$? angeben.

Theorem 5.10 *Hat SD einen induzierten Teilgraphen der Form* $(\{a,b,c,d\}, E)$ *mit* $\{(c,a),(a,c),(d,b),(b,d)\} \subseteq E \subseteq \overline{\{(a,d),(c,d),(c,b)\}}$, $a \neq c$, $b \neq d$ *und existiert eine nicht symmetrische Kante* $(f,e) \in SD \setminus SD^{-1}$, *so ist für reguläre Stringsprachen L die Frage* $max_{SC}(L) = L$ *unentscheidbar.*

Das Kriterium ist also der Teilgraph



wobei gepunktete Linien eventuelle (nicht störende) Abhängigkeiten beschreiben.

Beweis: Sei $R \subset \mathbb{N}^3$ mit $|R| < \infty$ und $a, e \in \mathbb{N}$ wie in Lemma 5.1 gegeben. Die Sprache

$$R_{c,d} := \{(c^{r_1}d^{r_2})^{mk} \mid (r_1 m, r_2 m, r_3) \in R \ \wedge \ ggT(r_1, r_2) = 1 \ \wedge \ \neg k/r_3 \in \mathbb{N}\}$$

ist regulär, da R endlich ist. Es gilt für alle $n_i, n_{i+1} \in \mathbb{N}$

$$\exists (r_1, r_2, r_3) \in R \ n_i/r_1 = n_{i+1}/r_2 \in \mathbb{N} \ \wedge \ \neg n_i/r_3 \in \mathbb{N}$$

genau dann wenn

$$\exists w \in R_{c,d} \ |w|_d = n_i \ \wedge \ |w|_c = n_{i+1}.$$

Ferner ist $R_{c,d}$ aus folgendem Grund maximal: Ist $(d, c) \notin SD$, so ist die Maximalität trivial, andernfalls folgt aus $(c^{r_1}d^{r_2})^l \stackrel{*}{\underset{SC}{\Longrightarrow}} (c^{r_1'}d^{r_2'})^{l'}$ wegen $r_1 l = r_1' l'$ und $r_2 l = r_2' l'$ daß $r_1 r_2' = r_1' r_2$ gilt. Es gilt aber $r_1 \geq r_1'$ und $r_2 \geq r_2'$, da der erste c-Block und der letzte d-Block nur kürzer werden können. Wegen $ggT(r_1, r_2) = 1$ muß also $r_1 = r_1'$ und $r_2 = r_2'$ gelten.
Sei

$$L = feb(adR_{c,d}cb)^+ a \cup efbd^{n_1} a(b(cd)^* a)^* bc^{n_m} a.$$

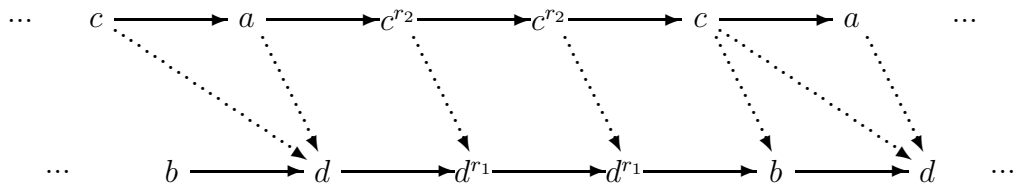
Da die c's und d's die durch ba bzw. ab getrennten Blöcke nicht verlassen können, sind die beiden Teilmengen der Sprache L jeweils maximal. Die einzige Möglichkeit, daß L selbst nicht maximal ist, besteht also darin, daß ein Wort aus dem Teil $feb(adR_{c,d}cb)^+ a$ zu einem Wort in $efbd^{n_1} a(b(cd)^* a)^* bc^{n_m} a$ abgeleitet werden kann. Könnte man entscheiden, ob die Sprache L maximal ist, so würde dies Lemma 5.1 widersprechen, da $n_2, ...n_{m-1}$ mit der Bedingung genau dann existieren, wenn ein Wort w aus $feb(adR_{c,d}cb)^+ a$ abgeleitet werden kann zu einem Wort

$$w' = efbd^{n_1} ab(cd)^{n_2} ab(cd)^{n_3} a...b(cd)^{n_{m-1}} abc^{n_m} a$$

in $efbd^{n_1} a(b(cd)^* a)^* bc^{n_m} a$. Die n_i d's eines i-ten Blockes und die n_{i+1} d's des folgenden $i+1$-ten Blockes von w' befinden sich in w im gleichen Teilwort in $R_{c,d}$, wodurch für alle n_i, n_{i+1} die Bedingung

$$\exists (r_1, r_2, r_3) \in R \ n_i/r_1 = n_{i+1}/r_2 \in \mathbb{N} \ \wedge \ \neg n_i/r_3 \in \mathbb{N}$$

erfüllt ist. Ein Teilstück mitten aus der Spur $[w\rangle$ wird exemplarisch dargestellt durch

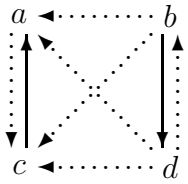


wobei die gepunkteten Pfeile mögliche weiche Kanten darstellen, welche bei der Ableitung umgedreht werden. ■

Durch das Zusammenfassen von Zeichen ergibt sich anderes Kriterium.

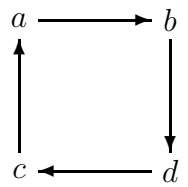
Theorem 5.11 *Hat SD einen induzierten Teilgraphen der Form* $(\{a,b,c,d\},E)$ *mit* $\{(c,a),(b,d)\} \subseteq E \subseteq \overline{\{(a,d),(c,d),(c,b),(a,b)\}}$, $a \neq c$, $b \neq d$ *und existiert* $(f,e) \in SD \setminus SD^{-1}$, *so ist für reguläre Stringsprachen L die Frage* $max_{SC}(L) = L$ *unentscheidbar.*

Das Kriterium ist also der Teilgraph



Beweis: Sei h ein Homomorphismus mit $h(a) = ac, h(c) = ca, h(b) = bd, h(d) = db$. Anstelle der Sprache L im Beweis von Theorem 5.10 verwenden wir nun $h(L)$. Wird im einem Wort $h(w)$ in $h(L)$ ein Teilstück ac zu ca oder ein Teilstück db zu bd abgeleitet, so wird dabei entweder das Bild eines Wortes w' erzeugt, welches aus w durch Ersetzen von a durch c bzw. von d durch b entsteht, oder es wird ein Wort erzeugt, welches kein Bildwort ist und von dem auch kein Bildwort mehr abgeleitet werden kann. In beiden Fällen kann damit kein Wort in $h(L)$ erreicht werden, da die Anzahl der a's und b's in jedem Wort von L gleich sind. Somit verhalten sich die gepaarten Symbole in gleicher Weise wie die Urbilder im Beweis von Theorem 5.10. ■

Theorem 5.12 *Hat SD einen induzierten Teilgraphen der Form*



(d.h. $(\{a,b,c,d\},\{(a,b),(b,c),(c,d),(d,a)\})$*), so ist für reguläre Stringsprachen L die Frage* $max_{SC}(L) = L$ *unentscheidbar.*

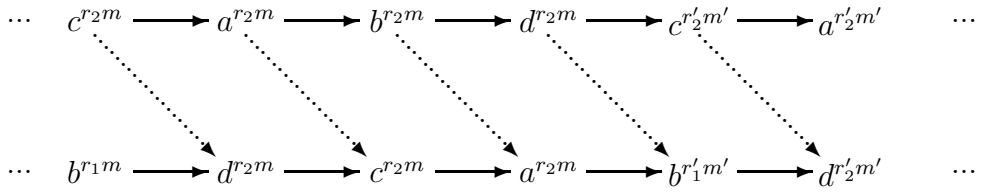
Beweis: Analog zum Beweis von Theorem 5.10 sei

$$R_{b,c} := \{(b^{r_1}c^{r_2})^{mk} \mid (r_1 m, r_2 m, r_3) \in R \ \wedge \ ggT(r_1, r_2) = 1 \ \wedge \ \neg k/r_3 \in \mathbb{N}\}.$$

Wir betrachten die Maximalität der Sprache

$$L = (R_{b,c}(ad)^+(bc)^+(da)^+)^*(cb)^+ \ \cup \ b^{n_1}((cd)^+(ac)^+(ba)^+(db)^+)^*c^{n_m}.$$

Ein Teilstück aus der Spur $[w\rangle$ für ein Wort $w \in (R_{b,c}(ad)^+(bc)^+(da)^+)^*(cb)^+$ wird dabei exemplarisch dargestellt durch



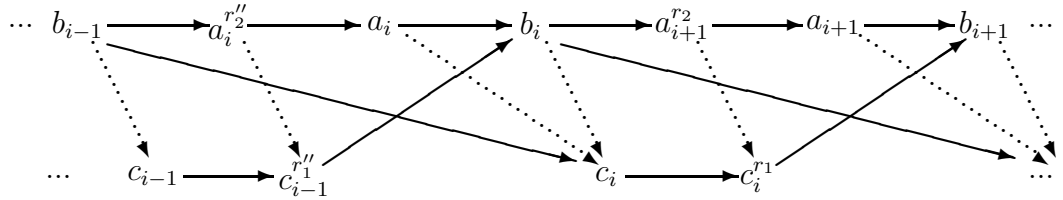
wobei die Ableitung zu einem $w' \in b^{n_1}((cd)^+(ac)^+(ba)^+(db)^+)^*c^{n_m}$ nur dann erfolgreich sein kann, wenn $r_2 m = r'_1 m'$ ist. ■

Theorem 5.13 *Hat SD einen induzierten Teilgraphen der Form*
$(\{a_1, b_1, c_1, ..., a_n = a_0, b_n = b_0, c_n = c_0\}, E)$ *oder*
$(\{a_1, b_1, c_1, ..., a_n = a_0, b_n = b_0, c_n = c_0\}, E^{-1})$ *mit*
$\{(a_i, b_i), (b_i, a_{i+1}), (c_i, b_{i+1}), (b_i, c_{i+1}) \mid i < n\} \subseteq E \subseteq$
$\overline{\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}}$ *oder*
$\{(a_i, b_i), (b_i, a_{i+1}), (c_i, a_{i+1}), (b_i, c_{i+1}) \mid i < n\} \subseteq E \subseteq$
$\overline{\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}}$ *oder* $\forall i < n \ a_i \neq a_{i+1}$ *und*
$\{(a_i, b_i), (c_i, a_{i+1}), (b_i, c_{i+1}) \mid i < n\} \subseteq E \subseteq \overline{\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}}$
und existiert $(f, e) \in SD \setminus SD^{-1}$*, so ist für reguläre Stringsprachen* L *die Frage* $max_{SC}(L) = L$ *unentscheidbar.*

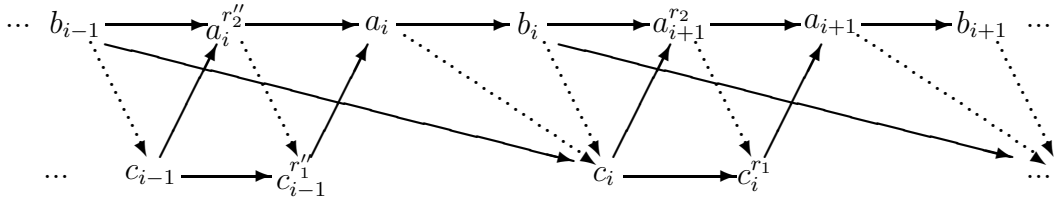
Beweis: Analog zum Beweis von Theorem 5.10, wobei die Sprache

$$\begin{aligned} L = & feb_n(c_n R_{a_1,c_n} a_1 b_1 (c_1 a_2)^+ b_2 ... (c_{n-1} a_n)^+ b_n)^+ (c_n a_1)^+ \cup \\ & ef c_n^{n_1} b_n ((a_1 c_1)^+ b_1 (a_2 c_2)^+ b_2 ... (a_n c_n)^+ b_n)^+ a_1^{n_m} \end{aligned}$$

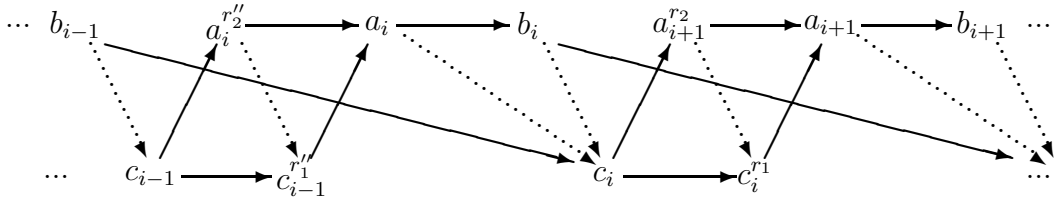
verwendet wird und ein Teilstück von $[w\rangle$ im Fall
$\{(a_i, b_i), (b_i, a_{i+1}), (c_i, b_{i+1}), (b_i, c_{i+1}) \mid i < n\} \subseteq E$ die Form



im Fall $\{(a_i, b_i), (b_i, a_{i+1}), (c_i, a_{i+1}), (b_i, c_{i+1}) \mid i < n\} \subseteq E$ die Form

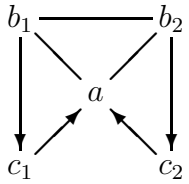


und im Fall $\forall i < n\ a_i \neq a_{i+1}$ und $\{(a_i, b_i), (c_i, a_{i+1}), (b_i, c_{i+1}) \mid i < n\} \subseteq E$ die Form



haben können. ■

Beispiel:



Mit der gleichen Beweismethode kann auch die Unentscheidbarkeit der Frage $f_{SC}(R_1) \cap R_2 = \emptyset$? für zwei reguläre Sprachen gezeigt werden. Die Bedingung der Existenz einer asymmetrischen Kante $(f, e) \in SD \setminus SD^{-1}$ ist dabei nicht mehr erforderlich.

Theorem 5.14 *Hat SD einen induzierten Teilgraphen der Form*

1. $(\{a, b, c, d\}, E)$ *mit* $\{(c, a), (a, c), (d, b), (b, d)\} \subseteq E \subseteq \overline{\{(a, d), (c, d), (c, b)\}}$, $a \neq c$, $b \neq d$ *oder*

2. $(\{a, b, c, d\}, E)$ *mit* $\{(c, a), (b, d)\} \subseteq E \subseteq \overline{\{(a, d), (c, d), (c, b), (a, b)\}}$, $a \neq c$, $b \neq d$ *oder*

3. $(\{a, b, c, d\}, \{(a, b), (b, c), (c, d), (d, a)\})$ *oder*

4. $(\{a_1, b_1, c_1, ..., a_n = a_0, b_n = b_0, c_n = c_0\}, E)$ *oder*
   $(\{a_1, b_1, c_1, ..., a_n = a_0, b_n = b_0, c_n = c_0\}, E^{-1})$ *mit*
   $\underline{\{(a_i, b_i), (b_i, a_{i+1}), (c_i, b_{i+1}), (b_i, c_{i+1}) \mid i < n\}} \subseteq E \subseteq$
   $\overline{\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}}$ *oder*
   $\underline{\{(a_i, b_i), (b_i, a_{i+1}), (c_i, a_{i+1}), (b_i, c_{i+1}) \mid i < n\}} \subseteq E \subseteq$
   $\overline{\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}}$ *oder*
   $\forall i < n\ a_i \neq a_{i+1}$ *und*

$$\underline{\{(a_i, b_i), (c_i, a_{i+1}), (b_i, c_{i+1}) \mid i < n\}} \subseteq E \subseteq$$
$$\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}$$

so ist für zwei reguläre Sprachen die Frage $f_{SC}(R_1) \cap R_2 = \emptyset$? unentscheidbar.

Beweis: Im Fall 1 verwenden wir für wie in Lemma 5.1 gegebenes $R \subset \mathbb{N}^3$ mit $|R| < \infty$ und $a, e \in \mathbb{N}$ die Sprachen

$$R_1 := b(adR_{c,d}cb)^+a \text{ und } R_2 := bd^{n_1}a(b(cd)^*a)^*bc^{n_m}a.$$

Könnte man entscheiden, ob $f_{SC}(R_1) \cap R_2 = \emptyset$ gilt, so würde dies Lemma 5.1 widersprechen, da $n_2, ...n_{m-1}$ mit der Bedingung

$$\exists (r_1, r_2, r_3) \in R \; n_i/r_1 = n_{i+1}/r_2 \in \mathbb{N} \; \wedge \; \neg n_i/r_3 \in \mathbb{N}$$

genau dann existieren, wenn ein Wort w aus R_1 abgeleitet werden kann zu einem Wort

$$w' = bd^{n_1}ab(cd)^{n_2}ab(cd)^{n_3}a...b(cd)^{n_{m-1}}abc^{n_m}a$$

in R_2.
Analog dazu wird im Fall 2

$$R_1 := h(b(adR_{c,d}cb)^+a) \text{ und } R_2 := h(bd^{n_1}a(b(cd)^*a)^*bc^{n_m}a),$$

im Fall 3

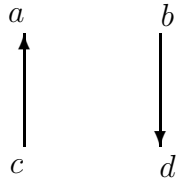
$$R_1 := (R_{b,c}(ad)^+(bc)^+(da)^+)^*(cb)^+ \text{ und } R_2 := b^{n_1}((cd)^+(ac)^+(ba)^+(db)^+)^*c^{n_m}$$

und im Fall 4

$$R_1 := b_n(c_nR_{a_1,c_n}a_1b_1(c_1a_2)^+b_2...(c_{n-1}a_n)^+b_n)^+c_nR_{a_1,c_n}a_1 \text{ und}$$
$$R_2 := c_n^{n_1}b_n((a_1c_1)^+b_1(a_2c_2)^+b_2...(a_nc_n)^+b_n)^+a_1^{n_m}$$

verwendet. ∎

Die Frage $f_{SC}(R_1) \cap f_{SC}(R_2) = \emptyset$? bzw. $f_{SC}(R_1) \parallel f_{SC}(R_2) \neq \emptyset$? nach der Synchronisierbarkeit zweier Halbspursprachen ist in dem durch Fall 2 beschriebenen Beispiel



offensichtlich entscheidbar, da in diesem Fall SC konfluent ist, und somit gemäß [RW91] die Komposition von SC und SC^{-1} ein Semikommutationssystem und damit die Frage $f_{SC}(R_1) \parallel f_{SC}(R_2) \neq \emptyset$ äquivalent zu der Frage $f_{SC \cup SC^{-1}}(R_1) \cap R_2 \neq \emptyset$ ist, was leicht zu entscheiden ist, da in diesem Fall $SD \cap SD^{-1}$ die Identität ist.
Für andere Teile der Bedingung läßt sich aber, wie man durch Betrachten der Spurgraphen der vorhergehenden Beweise erkennen kann, der Unentscheidbarkeitsbeweis in gleicher Weise führen:

Theorem 5.15 *Hat SD einen induzierten Teilgraphen der Form*

1. $(\{a, b, c, d\}, E)$ *mit* $\{(c, a), (a, c), (d, b), (b, d)\} \subseteq E \subseteq \overline{\{(a, d), (c, d), (c, b)\}}$, $a \neq c$, $b \neq d$ *oder*

2. $(\{a_1, b_1, c_1, ..., a_n = a_0, b_n = b_0, c_n = c_0\}, E)$ *oder*
   $(\{a_1, b_1, c_1, ..., a_n = a_0, b_n = b_0, c_n = c_0\}, E^{-1})$ *mit*
   $\underline{\{(a_i, b_i), (b_i, a_{i+1}), (c_i, b_{i+1}), (b_i, c_{i+1}) \mid i < n\}} \subseteq E \subseteq$
   $\overline{\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}}$ *oder*
   $\underline{\{(a_i, b_i), (b_i, a_{i+1}), (c_i, a_{i+1}), (b_i, c_{i+1}) \mid i < n\}} \subseteq E \subseteq$
   $\overline{\{(a_i, c_i), (b_i, c_i), (a_{i+1}, c_i) \mid i < n\}}$

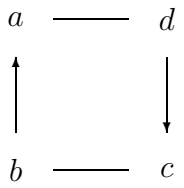
so ist für zwei reguläre Sprachen die Frage $f_{SC}(R_1) \parallel f_{SC}(R_2) = \emptyset$? unentscheidbar.

6 Die Konfluenz von Semikommutationssystemen

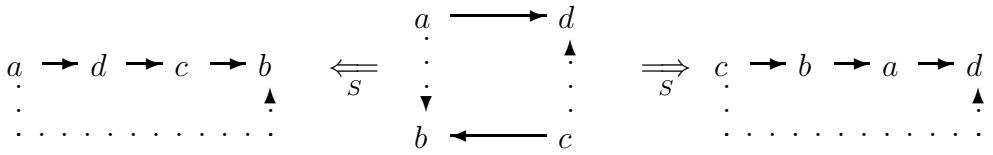
Definition Ein Semikommutationssystem SC heißt *konfluent*, wenn für alle $u, v, w \in A^*$ mit $u \overset{*}{\underset{SC}{\Longleftarrow}} v \overset{*}{\underset{SC}{\Longrightarrow}} w$ ein $v' \in A^*$ existiert mit $u \overset{*}{\underset{SC}{\Longrightarrow}} v' \overset{*}{\underset{SC}{\Longleftarrow}} w$. Dies bedeutet, daß falls $[u\rangle \subseteq [v\rangle$ und $[w\rangle \subseteq [v\rangle$, so ist $[u\rangle \cap [w\rangle \neq \emptyset$.

Die Eigenschaft der Konfluenz ist besonders deswegen von Interesse, da ein System, das zugleich *noethersch* ist, die Möglichkeit bietet das Wortproblem durch Berechnen der irreduzieblen Normalform zu lösen. Ein System $S \subseteq M \times M$ heißt *noethersch*, wenn es keine unendliche Kette $t_1 \underset{S}{\Longrightarrow} t_2 \underset{S}{\Longrightarrow} t_3 \ldots$ gibt.
Im Gegensatz zu Semi-Thue-Systemen ist die Konfluenz endlicher noetherscher längenreduzierender Spurersetzungssysteme i.A. unentscheidbar [NO88].
Ein Beispiel für ein nicht konfluentes Semikommutationssystem ist folgender Abhängigkeitsgraph:

$(A, SD) \quad =$

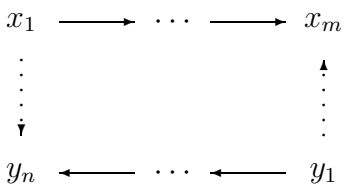


Bei diesem gilt $adcb \overset{}{\underset{SC}{\Longleftarrow}} acbd \overset{}{\underset{SC}{\Longrightarrow}} cbad$, aber $adcb$ und $cbad$ sind irreduzibel, wie folgendes Bild zeigt:



Allgemein hat eine Halbspur v die Möglichkeit zu einer nicht konfluenten Situation, d.h. $\exists u, w \in A^*$ mit $u \overset{*}{\underset{SC}{\Longleftarrow}} v \overset{*}{\underset{SC}{\Longrightarrow}} w$ und $\neg\exists v' \in A^*$ mit $u \overset{*}{\underset{SC}{\Longrightarrow}} v' \overset{*}{\underset{SC}{\Longleftarrow}} w$, genau dann, wenn v einen sehnenlosen, gerichteten Kreis aus harten Kanten und mindestens zwei weichen Kanten in umgekehrter Richtung besitzt, wie folgende Abbildung zeigt:

$[v\rangle =$

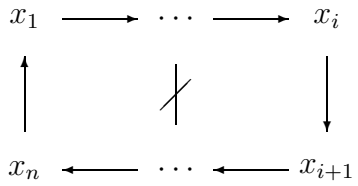


Theorem 6.1 *Das folgende Problem ist NLOGSPACE-vollständig: Gegeben seien ein Halbabhängigkeitsalphabet (A, SD) und $v \in A^*$.*
Gilt $\exists u, w \in A^*$ mit $u \overset{*}{\underset{SC}{\Longleftarrow}} v \overset{*}{\underset{SC}{\Longrightarrow}} w$ und $\neg\exists v' \in A^*$ mit $u \overset{*}{\underset{SC}{\Longrightarrow}} v' \overset{*}{\underset{SC}{\Longleftarrow}} w$?

Beweis: Eine nichtdeterministische logarithmisch platzbeschränkte Turingmaschine rät zwei weiche Kanten $a \dashrightarrow b$ und $c \dashrightarrow d$ und überprüft, ob es einen Weg aus harten Kanten und umgekehrten weichen Kanten von a nach d und von c nach b gibt. (Es darf auch $a = d$ oder $c = b$ gelten aber natürlich nicht beides.) Mit Hilfe der Technik von [Imm88] und [Sze88] wird noch überprüft ob es keinen Weg aus harten Kanten von a nach b und von c nach d gibt.
Die NLOGSPACE-Härte folgt durch Reduktion des monotonen Grapherreichbarkeitsproblems für gerichtete Graphen. Es werden zusätzliche Kanten (b, s) und (e, b) und ein zusätzlicher Knoten b hinzugefügt und bei dem so erzeugten Abhängigkeitsgraphen $(\{b, s, a_2, a_3, \ldots a_{n-1}, e\}, SD)$ überprüft, ob es eine nicht konfluente Situtation für $[sb(a_2a_3 \ldots a_{n-1})^n e\rangle$ gibt. ■

Ein allgemeines Kriterium für Konfluenz ist folgendes:

Theorem 6.2 *[DOR94] Sei (A, SD) ein Halbabhängigkeitsalphabet mit dem Semikommutationssystem $SC = \{ab \Longrightarrow ba \mid (a, b) \in A \times A \setminus SD\}$. SC ist nicht konfluent gdw. (A, SD) einen gerichteten Kreis $(x_1, \ldots, x_n)$ aus verschiedenen Knoten mit $n \geq 3$, $(x_i, x_{i+1}) \in SD$ für alle $i \bmod n$ und mindestens 2 gerichtete Kanten enthält, der aber keine ungerichtete Sehne $(x_i, x_j) \in SD \cap SD^{-1}$ mit $2 \leq \mid j - i \mid \leq n - 2$ besitzt.*



Da gemäß [RW91] ein Semikommutationssystem SC genau dann konfluent ist, wenn die Komposition von SC und SC^{-1} ein Semikommutationssystem ist, folgt dies auch aus folgender allgemeineren Aussage:

Theorem 6.3 *[RW91] Für Halbabhängigkeitsalphabete (A, SD_1) und (A, SD_2) gilt:*
Die Komposition der beiden Semikommutationssysteme ist kein Semikommutationssystem gdw. $(A, SD_1 \cup SD_2^{-1})$ einen gerichteten Kreis $(x_1, \ldots, x_n)$ aus verschiedenen Knoten mit $n \geq 3$, $(x_i, x_{i+1}) \in SD_1 \cup SD_2^{-1}$ für alle $i \bmod n$ und beiden Kanten $(x_0, x_n) \in SD_2 \setminus SD_1$ und $(x_i, x_{i+1}) \in SD_1 \setminus SD_2$ enthält, der aber keine Sehne $(x_h, x_j) \in SD_1 \cap SD_2$ mit $0 \leq h \leq i$ und $i + 1 \leq j \leq n$ besitzt.

Aus der Symmetrie in Theorem 6.2 folgt:

Korollar 6.1 *[DOR94] Ein Semikommutationssystem SC ist konfluent gdw. $SC^{-1} = \{ba \Longrightarrow ab \mid ab \Longrightarrow ba \in SC\}$ konfluent ist.*

Theorem 6.2 ist eine Verallgemeinerung eines Resultates aus [MO87], welches besagt, daß ein Semikommutationssystem ohne symmetrische Regeln konfluent ist gdw. die Unabhängigkeitsrelation transitiv ist. Das Resultat verallgemeinert auch [Ott89], da sich die Frage nach der Existenz eines endlichen konfluenten noetherschen Spurersetzungssystems, welches ein Spurmonoid auf einem anderen Spurmonoid mit größerer Abhängigkeitsrelation definiert, auf die Existenz eines konfluenten Semikommutationssystems reduzieren läßt.

Theorem 6.4 *[DOR94] Seien $M = M(A, D)$, $M' = M(A, D')$ zwei Spurmonoide mit $D' \subseteq D$. Sei $R \subseteq M \times M$ ein endliches konfluentes noethersches Spurersetzungssystem $M/R = M'$ und sei $S = \{ab \Longrightarrow ba \in R \mid a, b \in A\}$, so gilt $M/S = M/R = M'$ und S ist ein endliches konfluentes noethersches Spurersetzungssystem.*

Mit Hilfe der Graphkonstruktion, mit welcher bestimmte beschränkt quantifizierter Boolescher Formeln, welche gemäß [Wra77] Σ_2^P-vollständig sind, reduziert werden, erhalten wir in [DOR94] folgende Resultate:

Theorem 6.5 *Die folgenden zwei äquivalenten Probleme sind Σ_2^P-vollständig:*

1. *Gegeben zwei Halbanhängigkeitsalphabete (A, D) und (A, D') mit $D' \subseteq D$.*

   Gibt es ein konfluentes Semikommutationssystem SC mit
   $D = \{(a, b) \in A \times A \mid ab \Longrightarrow ba \notin SC \cap SC^{-1}\}$ und
   $D' = \{(a, b) \in A \times A \mid ab \Longrightarrow ba \notin SC \cup SC^{-1}\}$?

2. *Gegeben seien zwei Spurmonoide $M = M(A, D)$ und $M' = M(A, D')$ mit $D' \subseteq D$.*

   Gibt es ein endliches konfluentes noethersches Spurersetzungssystem $R \subseteq M \times M$ mit $M/R = M'$?

Sei die Anzahl der asymmetrischen Abhängigkeiten durch eine Konstante k beschränkt, bzw. $|D \setminus D'| \leq k$. Für $k = 1$ sind beide Probleme trivial.

Theorem 6.6 *Sei $k \geq 2$ konstant, so ist das Problem aus Theorem 6.5 beschränkt auf $|D \setminus D'| \leq k$ co-NP-vollständig.*

Noch spezieller gilt: Es ist co-NP-vollständig für eine Eingabe der Form $D = D' \cup \{(a, b), (b, a), (c, d), (d, c)\}$ zu entscheiden, ob ein endliches konfluentes noethersches Spurersetzungssystem $R \subseteq M(A, D) \times M(A, D)$ mit der Eigenschaft $M(A, D)/R = M(A, D')$ existiert.

Theorem 6.7 *Auch für den Spezialfall, daß höchstens 2 asymmetrische Regeln vorhanden sind, ist es co-NP-vollständig zu entscheiden, ob ein Semikommutationssystem konfluent ist.*

Mit Theorem 6.3 gilt auch folgendes:

Korollar 6.2 *Es ist es co-NP-vollständig zu entscheiden, ob für Halbabhängigkeitsalphabete* (A, SD_1) *und* (A, SD_2) *die Komposition der beiden Semikommutationssysteme ein Semikommutationssystem ist.*

Literatur

[AG91] C. Àlvarez and J. Gabarró. The parallel complexity of two problems on concurrency. *Information Processing Letters*, 38:61–70, 1991.

[AH89] IJ. J. Aalbersberg and H. J. Hoogeboom. Characterizations of the decidability of some problems for regular trace languages. *Mathematical Systems Theory*, 22:1–19, 1989.

[Ber79] J. Berstel. *Transductions and context-free languages*. Teubner Studienbücher, Stuttgart, 1979.

[BIS90] D.M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. of Comp. and Syst. Sciences*, 41:274–306, 1990.

[BMS82] A. Bertoni, G. Mauri, and N. Sabadini. Equivalence and membership problems for regular trace languages. In *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, number 140 in Lecture Notes in Computer Science, pages 61–71, Berlin-Heidelberg-New York, 1982. Springer.

[CF69] P. Cartier and D. Foata. *Problèmes combinatoires de commutation et réarrangements*. Number 85 in Lecture Notes in Mathematics. Springer, Berlin-Heidelberg-New York, 1969.

[CG90] M. Clerbout and D. Gonzalez. Decomposition of semi commutations. In B. Rovan, editor, *Proceedings of the 15th Symposium on Mathematical Foundations of Computer Science (MFCS'90), Banská Bystrica (Czechoslovakia) 1990*, number 452 in Lecture Notes in Computer Science, pages 209–216, Berlin-Heidelberg-New York, 1990. Springer.

[CG94] M. Clerbout and D. Gonzalez. Atomic semi commutations. *Theoretical Computer Science*, 123:259–272, 1994.

[CGL+92] M. Clerbout, D. Gonzalez, M. Latteux, E. Ochmanski, Y. Roos, and P.A. Wacrenier. Recognizable morphisms on semi commutations. Tech. Rep. LIFL I.T.-238, Université des Sciences et Technologies de Lille (France), 1992.

[CL87] M. Clerbout and M. Latteux. Semi-Commutations. *Information and Computation*, 73:59–74, 1987.

[Cle84] M. Clerbout. *Commutations Partielles et Familles de Langages*. Thèse, Université des Sciences et Technologies de Lille (France), 1984.

[Die90] V. Diekert. *Combinatorics on Traces*. Number 454 in Lecture Notes in Computer Science. Springer, Berlin-Heidelberg-New York, 1990.

[DOR94] V. Diekert, E. Ochmański, and K. Reinhardt. On confluent semi-commutation systems – decidability and complexity results. *Information and Computation*, 110:164–182, 1994. A preliminary version was presented at ICALP'91, Lecture Notes in Computer Science 510 (1991).

[DV89] V. Diekert and W. Vogler. On the synchronization of traces. *Mathematical Systems Theory*, 22:161–175, 1989. A preliminary extended abstract appeared at MFCS 88, Lecture Notes in Computer Science 324 (1988) 271-279.

[Gon93] D. Gonzalez. *Décomposition de semi-commutations*. Thèse, Université des Sciences et Technologies de Lille (France), 1993.

[Gre78] S. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoret. Comput. Sci.*, 7:311–324, 1978.

[Hau90] D. Hauschildt. *Semiliearity of the Reachability Set is Decidable for Petri Nets*. Dissertation, Universität Hamburg, 1990.

[HK89] D. V. Hung and E. Knuth. Semi-commutations and Petri nets. *Theoretical Computer Science*, 64:67–81, 1989.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[Iba78] O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the Association of Computing Machinery*, 25:116–133, 1978.

[Imm88] N. Immerman. Nondeterministic space is closed under complement. *SIAM Journal on Computing*, 17(5):935–938, 1988.

[Kie89] A. Kiehn. *A Structuring Mechanism for Petri Nets*. Dissertation, Universität München, 1989.

[Kos84] S.R. Kosaraju. Decidability of reachability in vector addition systems. In *Proceedings 14th Ann. ACM STOC*, pages 267–281, 1984.

[Lat92] M. Latteux. Some results on semi-commutations. In M. Ito, editor, *Proceedings of the International Colloquium on Words, Languages and Combinatorics (Kyoto 1990)*. World Scientific, Singapore, 1992. Appeared also as Technical Report I.T 201 Université de Lille.

[LR90] K.-J. Lange and P. Rossmanith. *Two Results on Unambiguous Circuits*. SFB-Bericht 342/3/90 A, I9006, Institut für Informatik, Technische Universität München, 1990.

[LR94] K.-J. Lange and K. Reinhardt. Empty alternation. In B. Rovan, editor, *Proc. of 19th MFCS*, number 841 in LNCS, pages 494–503, Kosice, Slovakia, August 1994. Springer-Verlag.

[May84] E. Mayr. An algorithm for the general Petri net reachability problem. *Siam J. Comput.*, 13:441–459, 1984.

[Maz77] A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.

[Maz87] A. Mazurkiewicz. Trace theory. In W. Brauer et al., editors, *Petri Nets, Applications and Relationship to other Models of Concurrency*, number 255 in Lecture Notes in Computer Science, pages 279–324, Berlin-Heidelberg-New York, 1987. Springer.

[Min71] M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1971.

[MO87] Y. Métivier and E. Ochmański. On lexicographic semi-commutations. *Information Processing Letters*, 26:55–59, 1987.

[NO88] P. Narendran and F. Otto. Preperfectness is undecidable for Thue systems containing only length-reducing rules and a single commutation rule. *Information Processing Letters*, 29:125–130, 1988.

[Och90] E. Ochmański. Semi-Commutation and Petri Nets. In V. Diekert, editor, *Proceedings of the ASMICS workshop Free Partially Commutative Monoids, Kochel am See 1989*, Report TUM-I9002, Technical University of Munich, pages 151–166, 1990.

[Och92] E. Ochmański. Modelling concurrency with semi-commutations. In I. M. Havel and V. Koubek, editors, *Proceedings of the 17th Symposium on Mathematical Foundations of Computer Science (MFCS'92), Prague, (Czechoslovakia), 1992*, number 629 in Lecture Notes in Computer Science, pages 412–420, Berlin-Heidelberg-New York, 1992. Springer.

[Ott89] F. Otto. On deciding confluence of finite string rewriting systems modulo partial commutativity. *Theoret. Comput. Sci.*, 67:19–36, 1989.

[Par90] Ian Parberry. A primer on the complexity theory of neural networks. In R.B. Banerji, editor, *Formal Techniques in Artificial Intelligence*, Amsterdam, 1990. North-Holland.

[Rei89] K. Reinhardt. *Hierarchien mit alternierenden Kellerautomaten, alternierenden Grammatiken und finiten Transducern.* Diplomarbeit, Universität Stuttgart, Breitwiesenstr. 22, D-70565 Stuttgart, September 1989.

[Rei90] K. Reinhardt. Hierarchies over the context-free languages. In J. Dassow and J. Kelemen, editors, *Proc. of 6th IMYCS*, number 464 in LNCS, pages 214–224. Springer-Verlag, 1990.

[Rei92a] K. Reinhardt. Counting and empty alternating pushdown automata. In J. Dassow, editor, *Developments in Theoretical Computer Science: Proc. of 7th IMYCS*, number 6 in Topics in Computer Mathematics, pages 123–132. Gordon and Breach Science Publishes S.A., 1992.

[Rei92b] K. Reinhardt. Sorting *in-place* with a *worst case* complexity of $n \log n - 1.3n + o(\log n)$ comparisons and $\varepsilon\, n \log n + o(1)$ transports. In Ibaraki et al., editor, *Proceedings of the 3rd International Symposium on Algorithms and Computation*, number 650 in LNCS, pages 489–498. Springer-Verlag, 1992.

[Rei94] K. Reinhardt. *Das Erreichbarkeitsproblem bei Petrinetzen mit inhibitorischen Kanten.* Manuscript, 1994.

[Ruz81] W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–383, 1981.

[RW91] Y. Roos and P. A. Wacrenier. Composition of two semi commutations. In A. Tarlecki, editor, *Proceedings of the 16th Symposium on Mathematical Foundations of Computer Science (MFCS'91), Kazimierz Dolny (Poland) 1991*, number 520 in Lecture Notes in Computer Science, pages 406–414, Berlin-Heidelberg-New York, 1991. Springer.

[Sei74] D. Seinsche. On a property of the class of n-colorable graphs. *Journal of Combinatorial Theory B*, B(16):191–193, 1974.

[Sto77] L. J. Stockmeyer. The polynomial time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[SV84] L. Stockmeyer and U. Vishkin. Simulation of parallel random access machines by circuits. *SIAM Journal on Computing*, 13(2):409–422, May 1984.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Teo93] Dan Teodosiu. *Bereichseigenschaften komplexer Spuren.* Diplomarbeit, Universität Stuttgart, 1993.

[Wol65] E.S. Wolk. A note on 'the comparability graph of a tree. In *Proc. Amer. Math. Soc. 16*, pages 17–20, 1965.

[Wra77] C. Wrathall. Complete sets and the polynomial hierarchie. *Theoretical Computer Science*, 3:23–33, 1977.

Index