FoSAP-Panikzettel

Caspar Zecha, Philipp Schröer, Fabian Meyer,
Christoph von Oy, Tobias Polock

5. April 2018

Inhaltsverzeichnis

1 Grundlagen

- $\mathbb{N} = \mathbb{N} \cup \{0\}$
- Alphabet $\Sigma = \{Symbole, Buchstaben, Zeichen\}$
- Wörter bestehen aus diesem Alphabet (z.B. u, v, w)
- $|w|$ = Länge des Wortes w
- ε ist das leere Wort; $|\varepsilon|$=0
- $|w|_a$ = Anzahl der a in w
- Menge von Wörtern bilden eine Sprache (z.B. Σ^*)
- Sprache $\overline{L} = \Sigma^* \setminus L$
- u ist ein Prä-/In-/Suffix von v, falls $v = uw$ / $v = w_1uw_2$ / $v = wu$ gilt

2 DFAs

Ein DFA $A = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- Menge der Zustände Q
- Alphabet Σ
- Transitionsfunktion $\delta : Q \times \Sigma \to Q$
- Anfangszustand q_0

- Menge der Endzustände F mit $F \subseteq Q$

Ein Lauf $(r_0, a_1, r_1, a_2, r_2, \ldots, a_n, r_n)$ mit $\delta(r_{i-1}, a_i) = r_i$ und $r_0 = q_0$ wird genau dann akzeptiert, wenn $r_n \in F$.

$L(A) = \{w \in \Sigma^* | A \text{ akzeptiert } w\}$
Die Erreichbarkeitsrelation ist $q \xrightarrow{w} q'$ und beschreibt die Zustände, die man über einen "Pfad" im DFA erreichen kann.

2.1 Produktkonstruktion

Für $A_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1)$ und $A_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$ ist $A = (Q_1 \times Q_2, \Sigma, \delta, (q_{0,1}, q_{0,2}), F)$ mit Transitionsfunktion

$$\delta : (Q_1 \times Q_2) \times \Sigma \to Q_1 \times Q_2, \delta((r_1, r_2), a) \mapsto (\delta_1(r_1, a), \delta_2(r_2, a))$$

sowie mit $F = F_1 \times F_2$ / $(Q_1 \times F_2) \cup (F_1 \times Q_2)$ / $F_1 \times (Q_2 \setminus F_2)$ die Produktkonstruktion von A_1 und A_2.

2.2 Abschlusseigenschaften

- Ist L DFA-erkennbar, so ist auch $\overline{L}$ DFA-erkennbar (über Vertauschung von End- und nicht Endzuständen)
- $L_1 \cap L_2$ (über Produktkonstruktion mit $(q_1, q_2) \in F \Leftrightarrow q_1 \in F_1 \wedge q_2 \in F_2$)
- $L_1 \cup L_2$ (über Produktkonstruktion mit $(q_1, q_2) \in F \Leftrightarrow q_1 \in F_1 \vee q_2 \in F_2$)
- Konkatenation: LK (Hintereinanderausführung beider Automaten)
- Iteration: $L^* = \cup_{n \in (N)} L^n$, $L^0 = \{\varepsilon\}$, $\varnothing^* = \{\varepsilon\}$

3 NFAs

Ein NFA $A = (Q, \Sigma, \Delta, q_0, F)$ besteht aus

- Menge der Zustände Q
- Alphabet Σ
- Transitionsrelation $\Delta \subseteq Q \times \Sigma \times Q$
- Anfangszustand q_0
- Menge der Endzustände F mit $F \subseteq Q$

Erreichbarkeitsmengen: $E(A, w) := \{q \in Q | q_o \xrightarrow{w} q\}$

3.1 ε-NFAs

Ein ε-NFA $A{=}(Q, \Sigma, \Delta, q_0, F)$ mit

- Menge der Zustände Q
- Alphabet Σ
- Transitionsrelation $\Delta : Q \times (\Sigma \cup \{\varepsilon\}) \times Q$
- Anfangszustand q_0
- Menge der Endzustände F mit $F \subseteq Q$

Ein Lauf wird ohne ε angegeben.

3.2 Äquivalenz von DFAs und NFAs (Potenzmengenkonstruktion)

Jeder DFA ist offensichtlich ein NFA. Und zu jedem NFA $(Q, \Sigma, \Delta, q_0, F)$ kann man wie folgt einen äquivalenten DFA $(Q', \Sigma, \delta', q_0', F')$ konstruieren:

1. Setze die Zustandsmenge des DFAs $Q' := \mathrm{Pot}(Q)$
2. $\delta'(q', a) = \{q \in Q' \mid \exists p \in q' : (p, a, q) \in \Delta\}$
3. $q_0' = \{q_0\}$
4. $F' = \{q \in Q' | F \cap q \neq \emptyset\}$

Allgemein gilt: L ist DFA-erkennbar $\Leftrightarrow$ L ist NFA-erkennbar $\Leftrightarrow$ L ist ε-NFA-erkennbar $\Leftrightarrow$ L ist FA-erkennbar

3.3 Reduzierte Automaten

Reduzierte Automaten enthalten keine Zustände, die nicht vom Startzustand erreicht werden können.

3.4 Transformation von ε-NFAs in normale NFAs

1. Ersetze jede ε-Transition von q auf q' so, dass die Erreichbarkeitsmengen von q gleich der von q' ist. (Man "rückverlagert"die Transitionen also) (Formal: $\Delta' := \{(q, a, q') \in Q \times \Sigma \times Q \mid \mathcal{A} : q \xrightarrow{a} q'\}$)
2. Wenn eine ε-Transition von q nach $q' \in F$ geht, füge q zu F hinzu

3.5 Komplement

Sei A ein ε-NFA. Führe folgende Schritte aus um $L(\overline{A})$ zu finden:

1. Konstruiere zu A äquivalenten NFA A'
2. Konstruiere zu A' äquivalenten DFA A'' mit der Potenzmengenkonstruktion
3. Konstruiere DFA A''' mit $L(A''') = \overline{L(A'')}$

4 Reguläre Sprachen

Reguläre Sprachen lassen sich durch reguläre Ausdrücke beschreiben und sind genau die FA-erkennbaren Sprachen.

4.1 Sprachen von regulären Ausdrücken

Rekursive Definition mit r, s reguläre Ausdrücke und $a \in \Sigma$:

$$\begin{aligned} L(\emptyset) &:= \emptyset \\ L(\varepsilon) &:= \{\varepsilon\} \\ L(a) &:= \{a\} \end{aligned}$$

$$\begin{aligned} L((r+s)) &:= L(r) \cup L(s) \\ L((r \cdot s)) &:= L(r)L(s) \\ L(r^*) &:= L(r)^* \end{aligned}$$

4.2 Von einem regulären Ausdruck zu einem ε-NFA

Siehe Beispiel 4.13 in den Vorlesungsfolien für ein einfaches Schema. Lässt sich auch einfach selber herleiten.

4.3 Vom NFA zum regulären Ausdruck

4.3.1 Formal

$r_X(p, q)$ ist ein regulärer Ausdruck welcher die Sprache $\{w \in \Sigma^* \mid p \xrightarrow[X]{w} q\}$ beschreibt. $p \xrightarrow[X]{w} q$ meint dabei $p \xrightarrow{w} q$ nur unter Betrachtung von Zuständen aus X. Berechne die Ausdrücke $r_Q(q_0, q), q \in F$, dann ist $L(\sum r_Q(q_0, q)) = L(A)$. Die regulären Ausdrücke werden wie folgt rekursiv berechnet: Wähle $x \in X$:

$$r_X(q, q') = r_{X \setminus x}(q, q') + r_{X \setminus x}(q, x) r_{X \setminus x}(x, x)^* r_{X \setminus x}(x, q')$$

4.3.2 Graphisch

Man arbeitet auf dem Transitionsgraphen des NFAs

1. Füge einen neuen Startzustand hinzu, der mit einer ε-Transition auf den orginalen Startzustand zeigt.
2. Füge einen neuen Endzustand hinzu. Jeder orginale Endzustand zeigt nun mit einer ε-Transition auf den neuen Endzustand.

3. Schreibe auf jeder Transition, die mehr als einen Buchstaben hat, diese in einen regulären Ausdruck (dh. aus a, b wird $a + b$).

4. Nehme einen der orginalen Zustände herraus.

5. Wähle ein Paar von Zuständen so, dass der zweite vom ersten über den herraugenommenen Knoten erreichbar ist.

6. Füge, falls nicht vorhanden eine Transition vom ersten zum zweiten Knoten ein, und beschrifte sie mit ε.

7. Erweitere den regulären Ausdruck auf den neuen Transition um einen neuen +-Term, der wie folgt aussieht $rs * t$ wobei r der Ausdruck vom ersten zum herrausgenommen Knoten, s der Ausdruck vom herrausgenommenen zum herrausgenommenen Knoten, t der Ausdruck vom herrausgenommenen zum zweiten Knoten ist (der entstandene Ausdruck kann vereinfacht werden).

8. Wiederhole Schritte 5 bis 7 für jedes weitere Knotenpaar.

9. Wiederhole Schritte 4 bis 8 für jeden orginalen Knoten.

Jetzt sollte nur noch der neue Start und Endknoten übrig sein. Der Reguläre Ausdruck auf der Transition ist das Ergebnis.

4.4 Pumping-Lemma für reguläre Sprachen

Wird benutzt, um Regularität von Sprachen zu widerlegen.

Kernidee: Wenn ein Automat Wörter akzeptiert, deren Länge größer als die Anzahl der Zustände sind, gibt es irgendwo einen Kreis im Automaten.

Wenn L eine reguläre Sprache ist, besitzt sie eine Pumping-Zahl $n \geq 1$, sodass jedes Wort $w \in L$ mit $|w| \geq n$ zerlegbar ist in $w = xyz$ mit:

1. $y \neq \varepsilon$
2. $|xy| \leq n$
3. $xy^k z \in L, k \in \mathbb{N}$

Wenn man widerlegen soll, dass eine Sprache regulär ist, sollte man prüfen, ob man nicht leicht einen Widerspruch mit dem Pumping-Lemma herleiten kann. Dabei beißt sich dann meist die dritte Aussage mit den ersten zweien.

Beachte: Mit dem Pumping-Lemma lässt sich **nicht** beweisen, dass Sprachen regulär sind!

5 Algorithmen auf DFAs/NFAs

5.1 Myhill-Nerode Äquivalenz

Äquivalenzrelation $\sim$ auf Wörter einer Sprache L. Es gilt $x \sim y \Leftrightarrow$ für alle $w \in \Sigma^*$: $(xw \in L \Leftrightarrow yw \in L)$ für $x, y \in L$. Anschaulich: x und y sind äquivalent, falls sie sich genau durch die gleichen $w \in \Sigma^*$ ergänzen lassen sodass $xw, yw \in L$.

Der Index $\text{index}(L) \coloneqq \text{index}(\sim_L)$ einer Sprache ist die Anzahl ihrer Äquivalenzklassen. Notation der Äquivalenzklassen: $w/_L$ bezeichnet die Äquivalenzklasse mit dem Repräsentanten w auf L.

Eine Sprache ist genau dann regulär, wenn $\text{index}(L) < \infty$ und es ist $\text{index}(L) \leq |Q|$ für jede Zustandsmenge Q eines Automaten, der L erkennt.

5.2 Minimale/Myhill-Nerode DFAs

Der Myhill-Nerode DFA ist wie folgt definiert:

- $Q \coloneqq \Sigma^*/_L = \{v/_L \mid v \in \Sigma^*\}$
- $\delta(v/_L, a) = va/_L$
- $q_0 \coloneqq \varepsilon/_L$
- $F \coloneqq \{v/_L \mid v/_L \cap L \neq \emptyset\}$

Da für jeden DFA $\mid Q \mid \geq \text{index}(L)$ gilt, ist der Myhill-Nerode DFA ein minimaler DFA. Alle minimalen DFAs sind bis auf Isomorphie eindeutig.

Der Minimierungsalgorithmus für einen DFA $\mathcal{A}$ geht wie folgt:

1. Reduziere den DFA auf die erreichbaren Zustände
2. Berechne die Äquivalenzklassen von $\mathcal{A}$ mittels des Verfeinerungsalgorithmus
3. Berechne den Myhill-Nerode-DFA

Der Verfeinerungsalgorithmus ist der folgende:

1. Teile Q in die beiden Partitionen F und $Q \setminus F$ auf
2. Wenn es in einer Partition 2 Zustände p, q gibt, so dass $\delta(p, a)$ und $\delta(q, a)$ in verschiedenen Klassen sind, so unterteile die Klasse.
3. Wenn noch eine Verfeinerung möglich, gehe zu 2

5.3 Unendlichkeitsproblem

$L(\mathcal{A})$ unendlich $\Leftrightarrow \exists q \in Q, w \in \Sigma^* \setminus \{\varepsilon\} : \mathcal{A} : q_0 \xrightarrow{*} q \xrightarrow{w} q \xrightarrow{*} F$

5.4 Inklusionsproblem

Es ist $L_1 \subset L_2 \Leftrightarrow L_1 \cap \overline{L_2} = \emptyset$

5.5 Äquivalenzproblem

$L_1 = L_2 \Leftrightarrow L_1 \subset L_2 \wedge L_2 \subset L_1$

6 Kontextfreie Sprachen

6.1 Kontextfreie Grammatiken

Eine Kontextfreie Grammatik $G = (N, \Sigma, P, S)$ besteht aus

1. Einer Menge von Nichtterminalsymbolen N
2. Einer Menge von Terminalsymbolen Σ
3. Einer Menge von Produktionsregeln P der Form

   $$A \to \alpha$$

   mit $A \in N$ und $\alpha \in (N \cup \Sigma)^*$
4. Einem Startsymbol $S \in N$

Eine Ableitung eines Wortes $w \in \Sigma^*$ ist eine Folge von Ersetzungen von Nichtterminalen nach den Produktionsregeln um von S zu w zu kommen.

Die Sprache einer Kontextfreien Grammatik ist die Menge aller Wörter, die in dieser Grammatik abgeleitet werden können.

6.2 Chomsky-Normalform

Bei einer Grammatik in Chomsky-Normalform (kurz CNF) gibt es nur Produktionsregeln der Form

$$A \to a$$

oder

$$A \to BC$$

mit $A, B, C \in N, a \in \Sigma$.

6.2.1 Von kontextfreier Grammatik zu CNF

1. Elimination der ε-Regeln
2. Terminalsymbole nur in Regeln der Form $A \to a$
3. Elimination der Regeln $A \to B$
4. Elimination der Regeln $A \to A_1 \dots A_n$ mit $n > 2$

6.3 Pumping-Lemma für kontextfreie Sprachen

Genau wie für reguläre Sprachen gibt es für kontextfreie Sprachen ein Pumping-Lemma, mit dem Kontextfreiheit widerlegt werden kann. Wie zuvor lässt sich damit jedoch **nicht** Kontextfreiheit zeigen!

Die Kernidee ist ähnlich zu der des Pumping-Lemmas für reguläre Sprachen: Wenn das Wort länger als die Anzahl der Nichtterminale in CNF ist, dann müssen sich Teilbäume der Ableitung wiederholen.

Sei $L \subseteq \Sigma^*$ kontextfrei. Dann gibt es eine Zahl $n \geq 1$, so dass jedes Wort $z \in L$ mit $|z| \geq n$ zerlegbar ist in Wörter $u, v, w, x, y \in \Sigma^*$ mit

$$z = uvwxy$$

die folgende Eigenschaften haben

1. $vx \neq \varepsilon$
2. $|vwx| \leq n$
3. $uv^k wx^k y \in L$ für alle $k \geq 0$

6.4 Kellerautomanten

Kellerautomaten akzeptieren kontextfreie Sprachen. Ein Kellerautomat $\mathcal{A}$ ist als 7-Tupel definiert: $\mathcal{A} = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ mit

- Die endliche Zustandsmenge Q
- Das Eingabealphabet Σ
- Das Stapelalphabet Γ
- Die Transitionsrelation $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$.
  Bspw. Transition (p, a, Z, q, γ): Im Zustand p, bei Eingabesymbol a und Stapelsymbol Z gehe in den Zustand q über und lege alle Symbole in $\gamma \in \Gamma^*$ auf den Stapel, so dass das erste Symbol von γ oben auf dem Stapel liegt.
- Der Anfangszustand $q_0 \in Q$
- Das Stapelanfangssymbol $Z_0 \in \Gamma$
- Die Endzustände $F \subseteq Q$

Ein solcher Kellerautomat kann sich in einem aktuellem "Ausführungszustand" befinden, der *Konfiguration*. Die Konfiguration κ ist definiert als

$$\kappa = (\underbrace{q}_{\text{Zustand}}, \underbrace{\gamma}_{\text{Stapelinhalt}}, \underbrace{w}_{\text{Rest des Eingabeworts}})$$

PDAs sind nichtdetermenistisch und können ε-Transitionen haben. Es gibt jedoch auch deterministische PDAs (DPDAs), jedoch bilden die DPDA erkennbaren Sprachen eine echte Teilmenge der von PDAs erkennbaren.

Ein PDA der mit leerem Stapel akzeptiert ist ein 6-Tupel $(Q, \Sigma, \Gamma, \Delta, q_0, Z_0)$ und akzeptiert ein Wort, falls es einen Lauf gibt, der in einem Zustand mit mit leerem Stapel endet. Eine Sprache ist genau dann PDA-erkennbar, wenn sie erkennbar ist durch einen PDA, der mit leerem Stapel akzeptiert.

6.5 Von einer Grammatik in CNF zum PDA

Für eine Grammatik (N, Σ, P, S) in CNF:
Setze

$$Q \coloneqq \{q_0\}$$
$$\Delta \coloneqq \{(q_0, \varepsilon, q_0, BC) \mid A \to BC \in P\} \cup \{(q_0, a, A, q_0, \varepsilon) \mid A \to a \in P\}$$

Dann ist $(Q, \Sigma, N, \Delta, q_0, S)$ ein PDA, der mit leerem Stapel akzeptiert und die selbe Sprache erkennt.

6.6 Von PDAs zu Grammatiken

Zu einem PDA, der mit leerem Stapel akzeptiert, wollen wir eine Grammatik konstruieren.

1. S ist ein neues Symbol
2. $N \coloneqq \{S\} \cup \{[pZq] \mid p, q \in Q, Z \in \Gamma\}$
3. $\{[pZp_m] \to \sigma[p_0 Z_1 p_1][p_1 Z_2 p_2] \dots [p_{m-1} Z_m p_m] \mid$
   $(p, \sigma, Z, p_0, Z_1 \dots Z_m) \in \Delta, m \geq 1, p_1, \dots, p_m \in Q\}$
   $\cup \{[pZ_0 p_0] \to \sigma \mid (p, \sigma, Z, p_0, \varepsilon) \in \Delta\}$
   $\cup \{S \to [q_0 Z_0 q] \mid q \in Q\}$

6.7 CYK-Algorithmus

Der CYK-Algorithmus löst das Wortproblem für eine kontextfreie Sprache in Chomsky-Normalform in $\mathcal{O}(n^3)$ Für ein Wort $w = a_1 \dots a_n$.
Distanz $d = j - i$ Mengen: $N_{i,j} = \{A \in N \mid A \overset{*}{\to} a_i \dots a_j\}$

1. Berechne alle $N_{i,j}$ mit $d = 0$
2. Berechne für alle $N_{i,j}$ mit $d = 1$ (dh. für alle $A \in N_{i,j}$ gibt es eine Regel $A \to BC$ und ein k mit $i \leq k < j$, so dass $B \in N_{i,k}$ und $C \in N_{k+1,j}$)
3. Wiederhole Schritt 2 mit jeweis incrementierter Distanz, bis $d + 1 = |w|$
4. Überprüfe $S \in N_{1,n}$

6.8 Markierungsalgorithmus

Ein "terminierendes Nichtterminal" ist ein Nichtterminal, aus welchem sich ein Wort ableiten lässt, oder eine Satzform bestehend aus Zeichen aus Σ und terminierenden Nichtterminalen.

Der Markierungsalgorithmus markiert nun rekursiv die terminierenden Nichtterminale. Wenn S kein terminierendes Nichtterminal ist, ist die Sprache leer.
Die Laufzeit ist quadratisch, das Problem lässt sich aber auch in Linearzeit lösen.

6.9 Weitere Abschlusseigenschaften

Das Unendlichkeitsproblem lässt sich in Polynomialzeit lösem (Pumping-Lemma). Das Äquivalenzproblem, das Inklusionsproblem, das Universilatitätsproblem ($L(\mathcal{G}) = \Sigma^*$), das Durchschnittsproblem ($L(\mathcal{G}) \cap L(\mathcal{G}') = \emptyset$), das Regularitätsproblem (Ist $L(\mathcal{G})$ regulär?) und das Eindeutigkeitsproblem sind unentscheidbare Probleme.

7 Kontextsensitive Grammatiken

Kontextsensitive Grammatiken sind eine Erweiterung von kontextfreien Grammatiken, bei denen Regeln der Form

$$\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$$

mit $\alpha_1, \alpha_2, \beta \in (\Sigma \cup N)^*, A \in N$ erlaubt sind.

Das Wortproblem ist für kontextsensitive Sprachen entscheidbar. Das Leerheitsproblem ist jedoch unentscheidbar. Die kontextsensitiven Grammatiken entsprechen im Automatenmodell den linear beschränkten Automaten (eine Turingmaschine mit linear zur Eingabelänge beschränkten Band).

8 Allgemeine Grammatiken

Allgemeine Grammatiken sind eine Erweiterung von kontextsensitiven Grammatiken, bei denen Regeln der Form

$$\alpha \to \beta$$

mit $\alpha, \beta \in (\Sigma \cup N)^*$ erlaubt sind.

Das Wortproblem für allgemeine Grammatiken ist unendscheidbar. Die allgemeinen Grammatiken entsprechen im Automatenmodell den Turingmaschinen.

8.1 Chomsky-Hierarchie

Die Chomsky-Hierarchie klassiert unsere Sprachen in einer Hierarchie:

- Allgemeine Grammatiken sind vom Typ 0

- Kontextsensitive Grammatiken sind vom Typ 1
- Kontextfreie Grammatiken sind vom Typ 2
- Rechtslineare Grammatiken (erkennen reguläre Sprachen) sind vom Typ 3

	allgemein	kontextsensitiv	kontextfrei	regulär
Wortproblem	U	E	E	E
Leerheit	U	U	E	E
Äquivalenz	U	U	U	E

E = entscheidbar, U = unentscheidbar

9 Nebenläufige Systeme

9.1 Transitionssysteme

Transitionssysteme dienen zur Modelierung von Prozessen. Sie sind ähnlich aufgebaut wie ein ε-NFA nur ohne Endzustände.
Der Prozess kann sich in einem Zustand befinden - und beginnt immer in einem festen Startzustand - um dann von dort aus Aktionen ausführen. Für die Interpretation des Modelles sind nicht nur mögliche Abfolgen von Aktionen wichtig, die der Prozess theoretisch vom Startzustand aus durchlaufen kann, sondern auch die dabei durchlaufenen Zustände.
Das heißt, wenn zwei Transitionssysteme als ε-NFAs aufgefasst, die selbe Sprache erzeugen würden, könnte das Verhalten des modellierten Prozesses unterschiedlich sein.

9.1.1 Das freie Produkt von Transitionssystemen

Das freie Produkt von Transitionssystemen modelliert die parallele Ausführung dieser Systeme.
Für Transitionssysteme $A_1, \ldots, A_n$ mit $A_i = (Q_i, \Sigma_i, q_{i,0}, \Delta_i)$ ist das freie Produkt definiert als

$$\prod_{i=1}^{n} \mathcal{A}_i \coloneqq \mathcal{A}_1 \times \ldots \times \mathcal{A}_n \coloneqq (Q, \Sigma, q_0, \Delta)$$

mit

$$Q \coloneqq \bigtimes_{i=1}^{n} Q_i$$
$$\Sigma \coloneqq \bigtimes_{i=1}^{n} (\Sigma_i \cup \{\varepsilon\})$$
$$q_0 \coloneqq (q_{1,0}, \ldots, q_{n,0})$$
$$\Delta \coloneqq \{((q_1, \ldots, q_n), (a_1, \ldots, a_n), (q'_1, \ldots, q'_n)) | \forall 1 \leq i \leq n : (q_i, a_i, q'_i) \in \Delta_i \vee (q_i = q'_i \wedge a_i = \varepsilon)\}$$

Also kommen wir von einem Zustand in einen anderen, wenn es in jedem einzelnen Faktor-Automaten eine entsprechende Transition gibt, oder wir nichts ändern (ε).

9.1.2 Synchronisierte Produkte

Idee: Erlaube im Produkt nur gewisse Transitionen um die Synchronisierung zu steuern und sinnlose Läufe auszuschließen. Ein Synchronisierungsschema ist eine Teilmenge $S \subset \Sigma$. Das synchronisierte Produkt mit Synchronisierungsschema S ist das Transitionssystem, das aus dem freien Produkt entsteht, wenn man nur Transitionen (q, a, q') mit $a \in S$ zulässt:

$$(\mathcal{A}_1 \times \ldots \times \mathcal{A}_n \mid S)$$

Enthält S die Transitionen $(a, \varepsilon), (\varepsilon, b)$, so können wir die Transition (a, b) hinzufügen oder weglassen, ohne das System zu wesentlich zu verändern. Dadurch kann jedoch das Synchronisierungsschemata verkleinert werden.

9.2 Prozesskalküle

Wir beschreiben die Transitionsrelationen nun als Gleichungssystem. Für die Transitionen $(q, a_1, q_1), \ldots, (q, a_n, q_n)$ von q fügen wir folgende Gleichungen hinzu.

$$q \equiv a_1.q_1 + \ldots + a_n.q_n$$

Durch Einsetzen von Gleichungen kann man weitere Gleichungen ableiten. Zur Modellierung von Nebenläufigkeit fügen wir eine neue zweistellige Operation $\mid$ hinzu, wobei $A \mid B$ bedeutet, dass A und B nebenläufig ausgeführt werden.

9.3 Petrinetze

Idee: Prozesse verbrauchen und erzeugen lokal Ressourcen Ein Petrinetz $N = (P, T, F)$ mit

1. Menge an Plätzen P
2. Menge an Transitionen T
3. Flussrelation $F \subseteq (P \times T) \cup (T \times P)$

Vorbereich einer Transition $t \in T$:	${}^\bullet t := \{p \in P \mid (p, t) \in F\}$
Vorbereich eines Platzes $p \in P$:	${}^\bullet p := \{t \in T \mid (t, p) \in F\}$
Nachbereich einer Transition $t \in T$:	$t^\bullet := \{p \in P \mid (t, p) \in F\}$
Nachbereich eines Platzes $p \in P$:	$p^\bullet := \{t \in T \mid (p, t) \in F\}$

Markierung eines Petrinetzes $M : P \to \mathbb{N}$ ordnet jedem Platz $p \in P$ eine Markenzahl $M(p)$ zu.
Haben wir die Plätze fest aufgezählt z.B. $(P_1, \ldots, P_n)$ so können wir eine Markierung als Vektor in $\mathbb{R}^n$ auffassen $M = (M(p_1), \ldots M(p_n))$

$t \in T$ ist aktiviert, wenn für alle $p \in {}^\bullet t$ gilt $M(p) > 0$

t schaltet von M nach M' (geschrieben $M \xrightarrow{t}_N M'$) wenn t in M aktiviert ist für M' gilt

$$M'(p) \begin{cases} M(p) - 1 & p \in {}^\bullet t \setminus t^\bullet \\ M(p) + 1 & p \in t^\bullet \setminus {}^\bullet t \\ M(p) & \text{sonst} \end{cases}$$

Ein Lauf von M nach M' der länge l ist eine Folge $M_0, P_1, \ldots, P_l, M_l$ mit

$$l \geq 0, M = M_0, M = M_l$$

und

$$M_0 \xrightarrow{t_1}_N M_1 \xrightarrow{t_2}_N M_2 \ldots M_{l-1} \xrightarrow{t_l}_N M_l$$

Die Notation ist ähnlich zur Erreichbarkeitsrelation von NFAs.

Eine Markeirung M ist erreichbar wenn gilt $M_0 \xrightarrow{*}_N M$ wobei M_0 die Startmarkierung des Netzes ist.

M ist eine Verklemmung von N wenn es keine Transition $t \in T$ gibt, die aktiv ist.

(N, M) ist beschränkt, wenn nur endlich viele Markeirung von M erreicht werden können.

Wenn man Markierungen als Knoten in einen Baum einzeichnet, in dem die Kinder alle Markierungen sind, die mit einem Lauf der länge 1 erreicht werden können, kann man für beschrkänkte Netze entscheiden ob $M \xrightarrow{*}_N M'$ gilt.