Automatentheorie und ihre Anwendungen
Teil 2: endliche Automaten auf endlichen Bäumen

Wintersemester 2018/19 Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)

http://tinyurl.com/ws1819-autom

Überblick

Und nun …

Semistrukturierte Daten sind …

- ein Datenmodell zur Beschreibung von **Entitäten** und **Attributen**,

  das weniger formale Struktur voraussetzt als z. B. relationale Datenbanken
- ein Vorläufer von XML
- gut geeignet, um
  - Dokumentansichten (z. B. Webseiten) und
  - strukturierte Daten (z. B. Datenbank-Tabellen)

  zu repräsentieren und miteinander zu verbinden

Merkmale semistrukturierter Daten

Daten werden in Entitäten (Einheiten) zusammengefasst

- Markierung von Entitäten durch Tags
- Bildung von Hierarchien
- Gruppieren ähnlicher Entitäten
- Entitäten derselben Gruppe können verschiedene (oder keine) Attribute haben
- Reihenfolge der Attribute *kann* eine Rolle spielen (Mengen oder Listen z. B. von Telefonnummern?)

Beispiel:

```
Person: {Name: {VN: "Thomas", NN: "Schneider"},
         Telnr: 64432,
         Telnr: 43776243,
         Email: "ts@informatik..."}
```

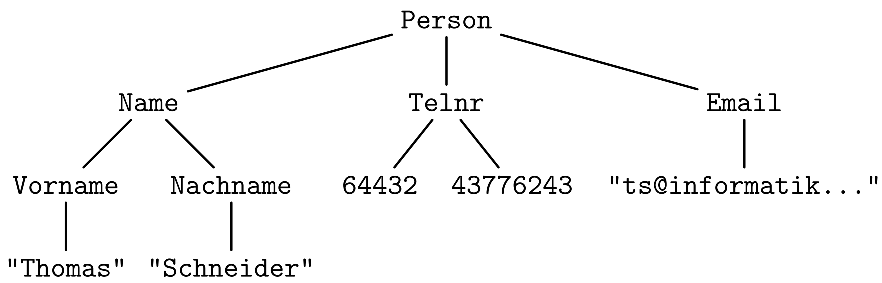

Datenstruktur: Baum

?

Datenstruktur: Baum

```
Person: {Name: {VN: "Thomas", NN: "Schneider"},
         Telnr: 64432,
         Telnr: 43776243,
         Email: "ts@informatik..."}
```

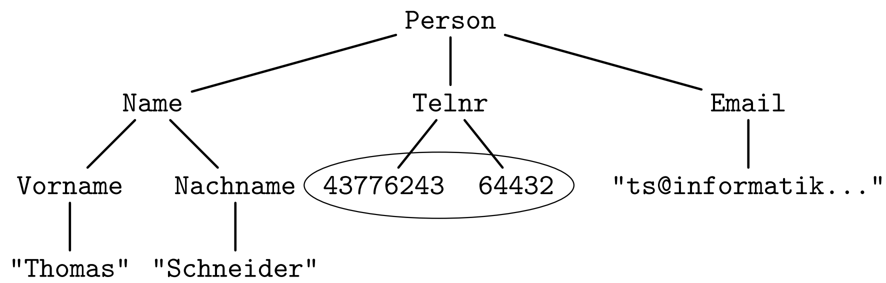
Repräsentation im Baum ist naheliegend:



Datenstruktur: Baum

```
Person: {Name: {VN: "Thomas", NN: "Schneider"},
         Telnr: 64432,
         Telnr: 43776243,
         Email: "ts@informatik..."}
```

Ist das derselbe oder ein anderer Baum?



Automaten auf endlichen Bäumen

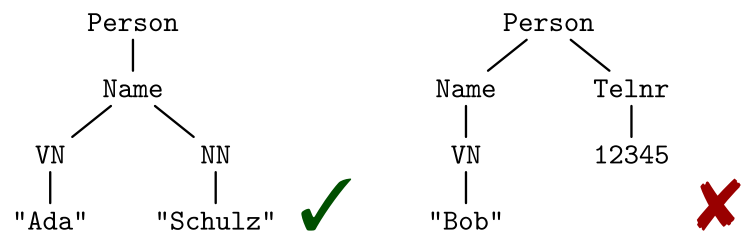
. . . sind wichtig für semistrukturierte Daten, weil sie . . .

- XML-Schemasprachen und -validierung zugrunde liegen
- XML-Anfragesprachen auf ihnen aufgebaut sind

XML-Schemasprachen und -validierung

- Schemasprachen zur Beschreibung gültiger Bäume
- Validierung = Erkennen gültiger und ungültiger Bäume
- gängige Schemasprachen benutzen dafür Baumautomaten

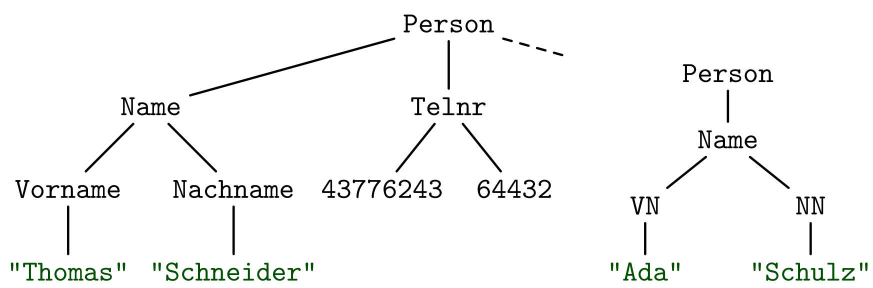
Beispiel: jeder Name muss einen Vor- und Nachnamen haben



XML-Anfragesprachen

- beantworten Anfragen mit Daten aus gegebenen Bäumen

Beispiel: gib alle Namen von Personen zurück



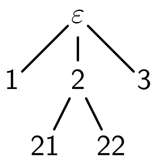
Und nun . . .

Positionen im Baum

- **positive** natürliche Zahlen: $\mathbb{N}_+$
- **Position:** Wort $p \in \mathbb{N}_+^*$

Idee: Wurzel ist ε
j-ter Nachfolger von p ist pj

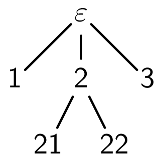
Beispiel:



Alphabet mit Stelligkeit

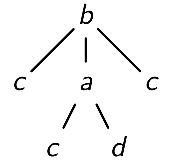
- hier: **r-Alphabet** Σ (auf Englisch: *ranked alphabet*)
- nichtleere endliche Menge von Symbolen;
  jedem Symbol ist eine Stelligkeit $\in \mathbb{N}$ zugeordnet
- Σ_m = Menge der Symbole mit Stelligkeit m
- Schreibweise: $\Sigma = \{a_1/r_1, \dots, a_n/r_n\}$ heißt:
  Σ enthält die Symbole a_i mit Stelligkeit r_i, $i = 1, \dots, n$

Beispiel: $\Sigma = \{a/2,\ b/3,\ c/0,\ d/0\}$



a „passt" in Position 2
b „passt" in Position ε
c, d „passen" in Pos. $1, 21, 22, 3$

$\rightsquigarrow$



Baum über Σ

Was ist nun ein Baum?

?

Was ist ein Baum?

Definition 2.1

Ein **endlicher geordneter Baum** über dem r-Alphabet Σ ist ein Paar $T = (P, t)$, wobei

(1) $P \subseteq \mathbb{N}_+^*$ eine nichtleere endl. präfix-abgeschlossene Menge ist,

(2) $t : P \to \Sigma$ eine Funktion ist mit den folgenden Eigenschaften.

  (a) Wenn $t(p) \in \Sigma_0$, dann $\{j \mid pj \in P\} = \emptyset$.

  (b) Wenn $t(p) \in \Sigma_m$, $m \geqslant 1$, dann $\{j \mid pj \in P\} = \{1, \dots, m\}$.

Erklärungen:

(1) P: Menge der vorhandenen Positionen

Präfix-Abgeschlossenheit: Baum ist wohlgeformt
(z. B.: wenn Position 31 existiert, dann auch Position 3 und ε)

(2) **(a)** und **(b)** sagen: Stelligkeit des Zeichens an Position p muss mit der Anzahl der Kinder von p übereinstimmen.

Was ist ein Baum?

Bezeichnungen

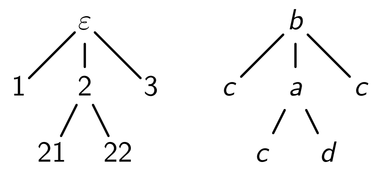
- Position p hat **Kinder** $p1, p2, \ldots$;
  p ist deren **Elternteil**
- jedes Präfix von p ist ein **Vorgänger** von p;
  p ist **Nachfolger** eines jeden Präfixes von p
- **Blatt:** Knoten ohne Kinder
- **Höhe von p** in T:
  Länge des längsten Pfades von p zu einem Blatt
- **Höhe von T:** Höhe von ε in T

Beispiel

$$\Sigma = \{a/2,\ b/3,\ c/0,\ d/0\}$$
$$P = \{\varepsilon, 1, 2, 3, 21, 22\}$$

$$t(\varepsilon) = b,\ t(1) = c,\ t(2) = a,\ t(3) = c,\ t(21) = c,\ t(22) = d$$

Positionen P Baum $T = (P, t)$



andere Schreibweise:

$$b(ca(cd)c)$$

($\approx$ in-order-Tiefensuche)

- Höhe: 2
- Blätter: 1, 21, 22, 3
- 21 ist Kind von 2 und hat Vorgänger 2, ε

Bottom-up-Baumautomaten

Definition 2.2

Ein **nichtdet. Bottom-up-Automat auf endl. geord. Bäumen (NEBA)** ist ein Quadrupel $\mathcal{A} = (Q, \Sigma, \Delta, F)$, wobei

- Q eine endliche nichtleere **Zustandsmenge** ist,
- Σ ein r-Alphabet ist,
- Δ eine Menge von **Überführungsregeln** der Form

  $$a(q_1, \ldots, q_m) \to q$$

  ist mit $m \geqslant 0$, $a \in \Sigma_m$, $q, q_1, \ldots, q_m \in Q$, und
- $F \subseteq Q$ die Menge der **akzeptierenden Zustände** ist.

Bottom-up-Baumautomaten: Intuitionen

Bedeutung der Überführungsregeln $a(q_1, \ldots, q_m) \to q$:

- Wenn $\mathcal{A}$ in Position p Zeichen a liest
- und in p's Kindern Zustände $q_1, \ldots, q_m$ eingenommen hat,

dann darf $\mathcal{A}$ in p Zustand q einnehmen. **T 2.1**

$\rightsquigarrow$ **Andere Betrachtungsweise:**

- $\mathcal{A}$ markiert Eingabebaum T **bottom-up** mit Zuständen
- $\mathcal{A}$ akzeptiert T, wenn $\mathcal{A}$ in der Wurzel einen akzeptierenden Zustand einnimmt

Was sind dann die Anfangszustände?

- Ü-Regeln $a() \to q$ deklarieren „zeichenspezifische" AZ:
  $\mathcal{A}$ darf in mit a markierten Blättern in q starten
- Kurzschreibweise: $a \to q$

Berechnungen (analog zu NEAs)

Definition 2.3

Sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$ ein NEBA und $T = (P, t)$ ein Σ-Baum.

- Ein **Run** von $\mathcal{A}$ auf T ist eine Fkt. $r : P \to Q$ mit:
  - Wenn $t(p) = a \in \Sigma_0$ und $r(p) = q$, dann $a \to q \in \Delta$.
  - Wenn $t(p) = b \in \Sigma_m$ $(m \geqslant 1)$ und $r(p) = q$ und wenn $r(p1) = q_1, \quad \dots, \quad r(pm) = q_m$, dann $b(q_1, \dots, q_m) \to q \in \Delta$.

Also gilt:

- Blatt mit a kann q nur zugewiesen kriegen, wenn $a \to q \in \Delta$.
- Nicht-Blatt mit b, dessen Kinder $q_1, \dots, q_m$ haben, kann q nur zugew. kriegen, wenn $b(q_1, \dots, q_m) \to q \in \Delta$.

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.3

Sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$ ein NEBA und $T = (P, t)$ ein Σ-Baum.

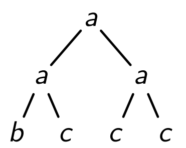
- Ein **Run** von $\mathcal{A}$ auf T ist eine Funktion $r : P \to Q$ mit:
  - Wenn $t(p) = a \in \Sigma_0$ und $r(p) = q$, dann $a \to q \in \Delta$.
  - Wenn $t(p) = b \in \Sigma_m$ $(m \geqslant 1)$ und $r(p) = q$ und wenn $r(p1) = q_1, \quad \dots, \quad r(pm) = q_m$, dann $b(q_1, \dots, q_m) \to q \in \Delta$.
- Ein Run r von $\mathcal{A}$ auf T ist **erfolgreich**, wenn $r(\varepsilon) \in F$.
- $\mathcal{A}$ **akzeptiert** T, wenn es einen erfolgreichen Run von $\mathcal{A}$ auf T **gibt**.
- Die von $\mathcal{A}$ **erkannte Sprache** ist $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$.

Beispiel 1

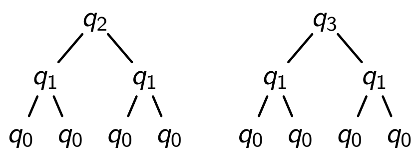
- Sei $\Sigma = \{a/2, b/0, c/0\}$ und $\mathcal{A} = (\{q_0, \dots, q_3\}, \Sigma, \Delta, \{q_2\})$

$$\text{mit } \Delta = \{\ b \to q_0, \quad c \to q_0, \quad a(q_0, q_0) \to q_1, \quad a(q_1, q_1) \to q_2, \quad a(q_1, q_1) \to q_3\ \}.$$

- Dann gibt es auf dem Baum



2 Runs:



- $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \text{alle Pfade in } T \text{ haben Länge } 2\}$
- **Anmerkung:** Da Σ nur ./2 und ./0 enthält: $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid T \text{ ist der vollständige Binärbaum der Tiefe } 2\}$

Beispiel 2

Sei $\Sigma = \{a/2, \ b/1, \ c/0, \ d/0\}$. Welcher NEBA erkennt $\{T \text{ über } \Sigma \mid \text{ jedes } c\text{-Blatt hat ein rechtes } d\text{-Geschwister}\}$?

$\mathcal{A} = (\{q_c, q_d, q_f\}, \Sigma, \Delta, \{q_f\})$ mit

$$\Delta = \{\ c \to q_c, \quad d \to q_d, \quad d \to q_f, \quad a(q_c, q_d) \to q_f, \quad a(q_f, q_f) \to q_f, \quad b(q_f) \to q_f \ \}$$

Übergang $a(q_d, q_d) \to q_f$ ist überflüssig: $d \to q_f$ und $a(q_f, q_f) \to q_f$.

Beispielbaum und -run: siehe Tafel **T 2.2**

Erkennbare Baumsprache

Definition 2.4

Eine Menge L von (endlichen geordneten) Bäumen über Σ ist eine **erkennbare Baumsprache**, wenn es einen NEBA $\mathcal{A}$ gibt mit $L(\mathcal{A}) = L$.

Determinismus

Definition 2.5

Sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$ ein NEBA.

Enthält Δ für jedes jedes $a \in \Sigma_m$ und alle $(q_1, \dots, q_m) \in Q^m$ **höchstens eine**[1] Regel $a(q_1, \dots, q_m) \to q$

dann ist $\mathcal{A}$ ein **deterministischer endlicher Baumautomat (DEBA)**.

$\leadsto$ Nachfolgezustand für jedes $(m+1)$-Tupel $a(q_1, \dots, q_m)$ ist eindeutig bestimmt (wenn er existiert)

- Jeder DEBA ist ein NEBA, aber nicht umgekehrt (z. B. die vergangenen 2 Beispiele).

Frage

Sind DEBAs und NEBAs gleichmächtig?

[1] hier „höchstens eine“ statt „genau eine“: vermeidet Papierkorbzustand

Potenzmengenkonstruktion

Antwort: Ja!

Satz 2.6

Für jeden NEBA $\mathcal{A}$ gibt es einen DEBA $\mathcal{A}^d$ mit $L(\mathcal{A}^d) = L(\mathcal{A})$.

Beweis: (analog zur Potenzmengenkonstr. für NEAs)
Sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$. Konstruieren $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, F^d)$:

- $Q^d = 2^Q$ (Potenzmenge der Zustandsmenge)
- $F^d = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$
- $a(S_1, \dots, S_m) \to S \in \Delta^d$ **gdw.** $S = \{q \mid \exists q_1 \in S_1, \dots, \exists q_m \in S_m : a(q_1, \dots, q_m) \to q \in \Delta\}$

$\mathcal{A}^d$ ist DEBA (klar) mit $L(\mathcal{A}^d) = L(\mathcal{A})$. **T 2.3** $\square$

Auch für NEBAs kann die Potenzmengenkonstruktion im schlimmsten Fall zu exponentiell vielen Zuständen führen.

Und nun …

Nichterkennbare Baumsprachen: Intuitionen

Beispiel:

- r-Alphabet $\Sigma = \{a/2,\ b/1,\ c/0\}$
- Baumautomat $\mathcal{A} = (\{q_0, q_1\}, \Sigma, \Delta, \{q_0\})$ mit

$$\Delta = \{\ c \to q_0,\quad b(q_0) \to q_1,\quad a(q_0, q_0) \to q_1,\quad b(q_1) \to q_0,\quad a(q_1, q_1) \to q_0\ \}.$$

$\rightsquigarrow L(\mathcal{A}) = \{T \mid \text{alle Wurzel-Blatt-Pfade in } T \text{ haben gerade Länge}\}.$

$\neq \{T \mid T \text{ hat gerade Höhe}\}$ **T 2.4**

Frage: Sind die folgenden Baumsprachen (über Σ) erkennbar?

$L_1 = \{T \mid T \text{ hat gerade Höhe}\}$

$L_2 = \{T \mid T \text{ ist vollständiger Binärbaum}\}$ **T 2.4 Forts.**

Antwort: **Nein.** **T 2.4 Forts.**

Pumping-Lemma: Hilfsbegriffe

Einsetzen von Bäumen ineinander:

- **Variable**: zusätzliches nullstelliges Symbol $x \notin \Sigma_0$
- **(unärer) Kontext**:
  Baum über $\Sigma \cup \{x\}$, in dem ein Blatt mit x markiert ist **T 2.5**
- **trivialer Kontext C_0**: Kontext der Höhe 0 ($\Rightarrow$ nur Wurzel)
- **Einsetzen** von Bäumen/Kontexten in Kontexte:
  - $C[T]$ = der Baum/Kontext, den man aus C erhält, indem man die Position von x mit Baum/Kontext T ersetzt **T 2.5 Forts.**
  - C^n induktiv definiert:

$$C^0 = C_0$$
$$C^{n+1} = C^n[C]$$

Pumping-Lemma

Satz 2.7 (Pumping-Lemma)

Sei L eine NEBA-erkennbare Baumsprache über dem r-Alphabet Σ.

Dann gibt es eine Konstante $k \in \mathbb{N}$,
so dass für alle Bäume $T \in L$ mit Höhe$(T) \geqslant k$ gilt:

Es gibt Kontexte C, D mit $D \neq C_0$ und Baum V mit $T = C[D[V]]$,
so dass $C[D^i[V]] \in L$ für alle $i \in \mathbb{N}$.

Beweis des Pumping-Lemmas

Beweis. Sei L eine erkennbare Baumsprache,
und sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$ ein NEBA mit $L(\mathcal{A}) = L$.

Wir wählen $k = |Q|$.

Sei $T = (P, t) \in L$ ein Baum mit Höhe $\geq k$,
und sei r ein akzeptierender Run von $\mathcal{A}$ auf T.

Wegen Höhe $\geq k$ gibt es in T einen Pfad mit $\geq k + 1$ Knoten.
Darauf gibt es also zwei Positionen $p_1 \neq p_2$ mit demselben Zustand,
d. h. $r(p_1) = r(p_2) = q$ für ein $q \in Q$.

O. B. d. A. ist $p_2 = p_1 p_3$ für ein $p_3 \neq \varepsilon$. **T 2.6**

Beweis des Pumping-Lemmas

Seien nun:

$$\begin{aligned} U &= T_{p_1} \\ C &= \text{derjenige Kontext mit } C[U] = T \\ V &= T_{p_2} \\ D &= \text{derjenige Kontext mit } U = D[V] \end{aligned}$$

Weil $p_1 \neq p_2$, ist D nichttrivial, also $D \neq C_0$ wie gefordert.

Dann gilt zunächst $T = C[D[V]]$. **T 2.6 Forts.**

Noch zu zeigen: $T_i := C[D^i[V]] \in L$ für alle $i \geq 0$.

Beweis des Pumping-Lemmas

Noch zu zeigen: $T_i := C[D^i[V]] \in L$ für alle $i \geq 0$.

1. Fall: $i = 0$, also $T_0 = C[V]$. **T 2.6 Forts.**

Definieren Run r_0 positionsweise:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \textit{ kein } \text{Nachfolger von } p_1 \text{ ist} \\ r(p_2p') & \text{falls } p = p_1p' \end{cases} \qquad (*)$$

Leicht zu prüfen: r_0 ist ein **Run** von $\mathcal{A}$ auf T_0.

r_0 ist **erfolgreich:** wegen $(*)$ ist $r_0(\varepsilon) = r(\varepsilon)$.

Also $T_0 \in L$.

Beweis des Pumping-Lemmas

Noch zu zeigen: $T_i := C[D^i[V]] \in L$ für alle $i \geq 0$.

2. Fall: $i \geq 1$. **T 2.6 Forts.**

Definieren Run r_i positionsweise:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \textit{ kein } \text{Nachfolger von } p_1 \text{ ist} \\ r(p_1p') & \text{falls } p = p_1p_3^jp',\ p' \textit{ kein } \text{NF von } p_3 \text{ und} \\ & \qquad\qquad\qquad p \textit{ kein } \text{NF von } p_1p_3^i \\ r(p_2p') & \text{falls } p = p_1p_3^ip' \end{cases}$$

Wie im 1. Fall: r_i ist **erfolgreicher Run** von $\mathcal{A}$ auf T_i.

Also $T_i \in L$. □

Anwendung des Pumping-Lemmas

Benutzen Kontraposition (siehe Kapitel „endliche Wörter“):

Wenn es **für alle** Konstanten $k \in \mathbb{N}$
einen Baum $T \in L$ mit Höhe$(T) \geqslant k$ **gibt**, so dass es
für alle Kontexte C, D mit $D \neq C_0$ und Bäume V mit $T = C[D[V]]$
ein $i \in \mathbb{N}$ **gibt** mit $C[D^i[V]] \notin L$,

dann ist L **keine** erkennbare Baumsprache.

T 2.7

Der Satz von Myhill-Nerode für Baumsprachen

Ziel: notwendige **und** hinreichende Bedingung für Erkennbarkeit

Definition 2.8

Sei L eine Baumsprache über Σ.

Zwei Σ-Bäume T_1, T_2 sind ***L*-äquivalent** (Schreibw.: $T_1 \sim_L T_2$), wenn für alle Σ-Kontexte C gilt:

$$C[T_1] \in L \quad \text{genau dann, wenn} \quad C[T_2] \in L$$

Satz 2.9

$L \subseteq \Sigma^*$ is NEBA-erkennbar gdw. $\sim_L$ endlichen Index hat.

T 2.8

Auch für Baumsprachen gilt: endlicher Index n von $\sim_L$
= minimale Anzahl von Zuständen in einem DEBA, der L erkennt

Und nun . . .

Drehen wir jetzt alles um? ☺

$\overset{?}{\rightsquigarrow}$

Top-down-Baumautomaten

. . . weisen der Wurzel einen Startzustand zu
und arbeiten sich dann von oben nach unten zu den Blättern durch:

Definition 2.10

Ein **nichtdet. Top-down-Automat auf endl. geord. Bäumen (NETDBA)** ist ein Quadrupel $\mathcal{A} = (Q, \Sigma, \Delta, I)$, wobei

- Q eine endliche nichtleere **Zustandsmenge** ist,
- Σ ein r-Alphabet ist,
- Δ eine Menge von **Überführungsregeln** der Form

$$(a, q) \to (q_1, \ldots, q_m)$$

  ist mit $m \geqslant 0$, $a \in \Sigma_m$, $q, q_1, \ldots, q_m \in Q$, und
- $I \subseteq Q$ die Menge der **Anfangszustände** ist.

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.11

Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA und $T = (P, t)$ ein Σ-Baum.

- **Berechnung (Run)** von $\mathcal{A}$ auf T ist eine Fkt. $r : P \to Q$ mit:
  - $r(\varepsilon) \in I$
  - Wenn $t(p) = a \in \Sigma_m$ $(m \geqslant 1)$ und $r(p) = q$ und wenn $r(p1) = q_1, \quad \dots, \quad r(pm) = q_m$, dann gibt es eine Regel $(a, q) \to (q_1, \dots, q_m) \in \Delta$.
  - Wenn $t(p) = a \in \Sigma_0$ und $r(p) = q$, dann $(a, q) \to () \in \Delta$.
- $\mathcal{A}$ **akzeptiert** T, wenn es einen Run von $\mathcal{A}$ auf T **gibt**.
- Die von $\mathcal{A}$ **erkannte Sprache** ist $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$.

Beachte: Keine Endzustände nötig – die Regeln in Δ müssen nur erlauben, von der Wurzel bis zu allen Blättern „durchzukommen“.

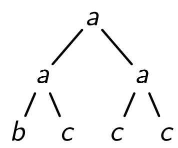
Beispiel 1

- Sei $\Sigma = \{a/2, \; b/0, \; c/0\}$ und

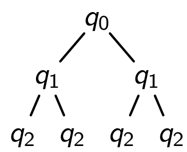
$\mathcal{A} = (\{q_0, q_1, q_2, q_x\}, \; \Sigma, \; \Delta, \; \{q_0\})$ mit

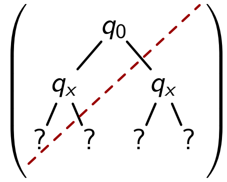
$$\Delta = \{ \; (a, q_0) \to (q_1, q_1), \quad (b, q_2) \to (), \quad (a, q_1) \to (q_2, q_2), \quad (c, q_2) \to (), \quad (a, q_0) \to (q_x, q_x) \; \}$$

- Dann gibt es auf dem Baum



nur 1 Run:





- $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \text{alle Pfade in } T \text{ haben Länge } 2\}$

Beispiel 2

Sei $\Sigma = \{a/2, \; b/1, \; c/0, \; d/0\}$. Welcher NETDBA erkennt $L_{\mathrm{cd}} = \{T \text{ über } \Sigma \mid \text{ jedes } c\text{-Blatt hat ein rechtes } d\text{-Geschwister}\}$?

NETDBA $\quad \mathcal{A} = (\{q_0, q_c, q_d\}, \Sigma, \Delta, \{q_0\})$ mit

$$\Delta = \{ \; (a, q_0) \to (q_0, q_0), \quad b(q_0) \to q_0, \quad (c, q_c) \to (), \quad (a, q_0) \to (q_c, q_d), \quad (d, q_d) \to (), \quad (d, q_0) \to () \; \}$$

Vergleiche mit dem NEBA $\quad \mathcal{A} = (\{q_c, q_d, q_f\}, \Sigma, \Delta, \{q_f\})$ mit

$$\Delta = \{ \; a(q_f, q_f) \to q_f, \quad b(q_f) \to q_f, \quad c \to q_c, \quad a(q_c, q_d) \to q_f, \quad d \to q_d, \quad d \to q_f \; \}$$

Was sagt uns das über das Verhältnis NETDBAs : NEBAs?

NETDBAs vs. NEBAs

NETDBAs und NEBAs sind gleichmächtig!

Satz 2.12

$\{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NETDBA}\} \quad = \quad \{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NEBA}\}.$

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NET**D**BA.
Konstruieren NEBA $\mathcal{A}^\uparrow = (Q, \Sigma, \Delta^\uparrow, F^\uparrow)$ mit:

$$\Delta^\uparrow = \{a(q_1, \dots, q_m) \to q) \mid (a, q) \to (q_1, \dots, q_m) \in \Delta\}$$
$$F^\uparrow = I$$

Dann ist jeder Run von $\mathcal{A}$ auf einem Σ-Baum T auch ein **erfolgreicher** Run von $\mathcal{A}^\uparrow$ auf T und umgekehrt.

Daraus folgt $L(\mathcal{A}^\uparrow) = L(\mathcal{A})$.

Rückrichtung analog. □

Determinisierung von NETDBAs

Erinnerung an Beispiel 2: Sei $\Sigma = \{a/2,\ b/1,\ c/0,\ d/0\}$ und $L_{\text{cd}} = \{T \text{ über } \Sigma \mid \text{ jedes } c\text{-Blatt hat ein rechtes } d\text{-Geschwister}\}$.

NETDBA $\mathcal{A} = (\{q_0, q_c, q_d\}, \Sigma, \Delta, \{q_0\})$ mit

$$\Delta = \{\ (a, q_0) \to (q_0, q_0),\quad b(q_0) \to q_0,\quad (c, q_c) \to (),$$
$$(a, q_0) \to (q_c, q_d),\qquad (d, q_d) \to (),$$
$$(d, q_0) \to ()\ \}$$

▲ Nichtdeterminismus!

Wir wissen ja, wie man Nichtdeterminismus „loswird“. **Oder?**

Determinisierung von NETDBAs?

Betrachte

- $\Sigma = \{a/2,\ b/0,\ c/0\}$ und
- die erkennbare Baumsprache $L = \{a(bc), a(cb)\}$.

(denke an die alternative Schreibweise von Folie 18)

Frage: Welcher DETDBA erkennt L?

Anwort: Keiner!

Lemma 2.13
L wird von keinem DETDBA erkannt.

Beweis: siehe Tafel. **T 2.9**

Korollar 2.14
Es gibt erkennbare Baumsprachen,
die nicht von einem DETDBA erkannt werden.

Und nun . . .

Operationen auf Baumsprachen

Zur Erinnerung: die Menge der erkennbaren Baumsprachen heißt abgeschlossen unter . . .

- **Vereinigung**, falls gilt:
  Falls L_1, L_2 erkennbar, so auch $L_1 \cup L_2$.
- **Komplement**, falls gilt:
  Falls L erkennbar, so auch $\overline{L}$.
- **Schnitt**, falls gilt:
  Falls L_1, L_2 erkennbar, so auch $L_1 \cap L_2$.

Quiz
Unter welchen Operationen sind die NEBA-erkennbaren Sprachen abgeschlossen?
Vereinigung?
Komplement?
Schnitt?

Abgeschlossenheit

Satz 2.15
Die Menge der NEBA-erkennbaren Sprachen ist abgeschlossen unter den Operationen $\cup, \cap, \overline{}$.

Direkte Konsequenz aus den folgenden Lemmata.

Abgeschlossenheit unter Vereinigung

Lemma 2.16
Seien $\mathcal{A}_1, \mathcal{A}_2$ NEBAs über Σ.
Dann gibt es einen NEBA $\mathcal{A}_3$ mit $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Beweis. analog zu NEAs:

Seien $\mathcal{A}_i = (Q_i, \Sigma, \Delta_i, F_i)$ für $i = 1, 2$.
O. B. d. A. gelte $Q_1 \cap Q_2 = \emptyset$.

Konstruieren $\mathcal{A}_3 = (Q_3, \Sigma, \Delta_3, F_3)$ wie folgt.

- $Q_3 = Q_1 \cup Q_2$
- $\Delta_3 = \Delta_1 \cup \Delta_2$
- $F_3 = F_1 \cup F_2$

Dann gilt: $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ □

Abgeschlossenheit unter Komplement

Lemma 2.17
Sei $\mathcal{A}$ ein NEBA über Σ.
Dann gibt es einen NEBA $\mathcal{A}^c$ mit $L(\mathcal{A}^c) = \overline{L(\mathcal{A})}$.

Beweis: analog zu NEAs:

- Umwandlung in DEBA
- Vertauschen von akzeptierenden und nicht-akz. Zuständen

Sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$.

Nach Satz 2.6 gibt es **D**EBA $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, F^d)$ mit $L(\mathcal{A}^d) = L(\mathcal{A})$ und **genau einem** Run pro Eingabebaum.

Dann erkennt $\mathcal{A}^c = (Q^d, \Sigma, \Delta^d, Q^d \setminus F^d)$ die Sprache $\overline{L(\mathcal{A})}$. □

Abgeschlossenheit unter Schnitt

Lemma 2.18
Seien $\mathcal{A}_1, \mathcal{A}_2$ NEBAs über Σ.
Dann gibt es einen NEBA $\mathcal{A}_3$ mit $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

…folgt direkt aus der Abgeschlossenheit unter $\cup$ und $\overline{}$:

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

Alternative: Konstruktion des Produktautomaten wie für NEAs (vermeidet exponentielle "Explosion")

Abgeschlossenheit unter Verkettungsoperationen

Randbemerkung:
Man kann Analoga zu $\cdot$ und $*$ für Baumsprachen definieren:

Seien L, L_1, L_2 Baumsprachen.

Bezeichne **Con(L)** die Menge aller Kontexte, die man aus Bäumen in L erhält, indem man ein Blattsymbol durch x ersetzt.

- $L_1L_2 = \{C[T] \mid T \in L_1,\ C \in \mathrm{Con}(L_2)\}$
- $L^* = \{C_1[C_2[\dots[C_n[T]]\dots]] \mid T \in L,\ C_1,\dots,C_n \in \mathrm{Con}(L),\ n \geqslant 0\}$

Abgeschlossenheit unter $\cdot, *$ kann man dann wie für NEAs zeigen, aber mit mehr technischem Aufwand (Eliminierung ε-Kanten …)

Und nun …

Das Leerheitsproblem

Eingabe: NEBA (oder DEBA) $\mathcal{A}$
Frage: Ist $L(\mathcal{A}) = \emptyset$?
d. h. $\mathbf{LP}_{\mathbf{NEBA}} = \{\mathcal{A} \mid \mathcal{A}\ \text{NEBA},\ L(\mathcal{A}) = \emptyset\}$ (analog für DEBAs)

Satz 2.19
$\mathrm{LP}_{\mathrm{NEBA}}$ und $\mathrm{LP}_{\mathrm{DEBA}}$ sind entscheidbar und P-vollständig.

Beweis.

- Entscheidbarkeit in Polyzeit analog zu NEAs:
  prüfe, ob ein akz. Zustand erreichbar ist (nächste Folie)
- P-Härte:
  Reduktion von „Solvable Path Systems“
  ($\approx$ Erreichbarkeit in Hypergraphen mit ternärer Kantenrelation),
  siehe [Comon et al. 2008, Exercise 1.19]

Das Leerheitsproblem

Polynomialzeitalgorithmus:

- Berechne Menge der erreichbaren Zustände
- Prüfe, ob ein akzeptierender Zustand darin ist.

Sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$.

Konstruieren Menge $R \subseteq Q$ wie folgt:

- $R := \{q \mid a \to q \in \Delta \text{ für ein } a \in \Sigma_0\}$
- Wenn es $q_1, \dots, q_m \in R$ und $a \in \Sigma_m$ gibt mit $a(q_1, \dots, q_m) \to q \in \Delta$ und $q \notin R$, dann $R := R \cup \{q\}$.
- Wiederhole letzten Schritt, bis sich R nicht mehr ändert.

Leicht zu sehen: Berechnung endet nach $\leq |Q|$ vielen Schritten $\rightsquigarrow$ R ist in Polyzeit berechenbar.

Noch zu zeigen: $L(\mathcal{A}) = \emptyset$ gdw. $R \cap F = \emptyset$ **T 2.10** □

Das Wortproblem alias Zugehörigkeitsproblem

Eingabe: NEBA (oder DEBA) $\mathcal{A}$, Baum T über Σ

Frage: Ist $T \in L(\mathcal{A})$?

d. h. $\textbf{WP}_{\textbf{NEBA}} = \{(\mathcal{A}, T) \mid \mathcal{A} \text{ NEBA}, T \in L(\mathcal{A})\}$ (analog f. DEBAs)

Satz 2.20

WP_{NEBA} und WP_{DEBA} sind entscheidbar und in **P**.

Beweis. analog zu NEAs, per Reduktion zum LP:

$$T \in L(\mathcal{A}) \text{ gdw. } L(\mathcal{A}) \cap L(\mathcal{A}_T) \neq \emptyset,$$

wobei $\mathcal{A}_T$ ein DEBA mit $L(\mathcal{A}_T) = \{T\}$ ist (konstruiere selbst!) □

(WP_{NEBA} ist LOGCFL-vollständig. (zwischen **NL** und **P**)
WP_{DEBA} ist in LOGDCFL. (Genaue Komplexität ist offen!)
$\text{WP}_{\text{DETDBA}}$ ist **L**-vollständig.)

Das Äquivalenzproblem

Eingabe: NEBAs (oder DEBAs) $\mathcal{A}_1, \mathcal{A}_2$

Frage: Ist $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?

d. h. $\textbf{ÄP}_{\textbf{NEBA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \text{ NEBAs}, L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$ etc.

Satz 2.21

ÄP_{NEBA} und ÄP_{DEBA} sind entscheidbar.
ÄP_{NEBA} ist **ExpTime**-vollständig; ÄP_{DEBA} ist in **P**.

Beweis.

- Entscheidbarkeit analog zu NEAs, per Reduktion zum LP:

$$L(\mathcal{A}_1) = L(\mathcal{A}_2) \text{ gdw. } L(\mathcal{A}_1) \triangle L(\mathcal{A}_2) = \emptyset$$

- obere Schranken: Automat für $L(\mathcal{A}_1) \triangle L(\mathcal{A}_2)$ ist exponentiell in der Größe der Eingabe-NEBAs / polynomiell für DEBAs
- **ExpTime**-Härte: Reduktion vom Universalitätsproblem (F. 59) □

Das Universalitätsproblem

Eingabe: NEBA (oder DEBA) $\mathcal{A}$

Frage: Ist $L(\mathcal{A}) = \mathcal{T}(\Sigma)$? ($\mathcal{T}(\boldsymbol{\Sigma}) = \{T \mid T \text{ ist ein } \Sigma\text{-Baum}\}$)

d. h. $\textbf{UP}_{\textbf{NEBA}} = \{\mathcal{A} \mid \mathcal{A} \text{ NEBA}, L(\mathcal{A}) = \mathcal{T}(\Sigma)\}$ (analog f. DEBAs)

Satz 2.22

UP_{NEBA} und UP_{DEBA} sind entscheidbar.
UP_{NEBA} ist **ExpTime**-vollständig; UP_{DEBA} ist in **P**.

Beweis:

- Entscheidbarkeit & obere Schranken per Red. zum ÄP:

$$L(\mathcal{A}) = \mathcal{T}(\Sigma) \text{ gdw. } L(\mathcal{A}) = L(\mathcal{A}_\Sigma),$$

  wobei $\mathcal{A}_\Sigma$ DEBA mit $L(\mathcal{A}_\Sigma) = \mathcal{T}(\Sigma)$ (konstruiere selbst!)

- **ExpTime**-Härte: Red. vom WP für lin. platzbeschränkte alternierende TM (s. a. [Comon et al. 2008, § 1.7]) □

Überblick Entscheidungsprobleme für NEBAs/DEBAs

Problem	entscheidbar?	für DEBAs effizient lösbar?	für NEBAs effizient lösbar?
LP	✓	✓	✓
WP	✓	✓	✓
ÄP	✓	✓	✘*
UP	✓	✓	✘*

* nachweislich! (da **ExpTime** $\neq$ **P**)

Und nun ...

Zur Erinnerung: semistrukturierte Daten

Daten werden in Entitäten (Einheiten) zusammengefasst

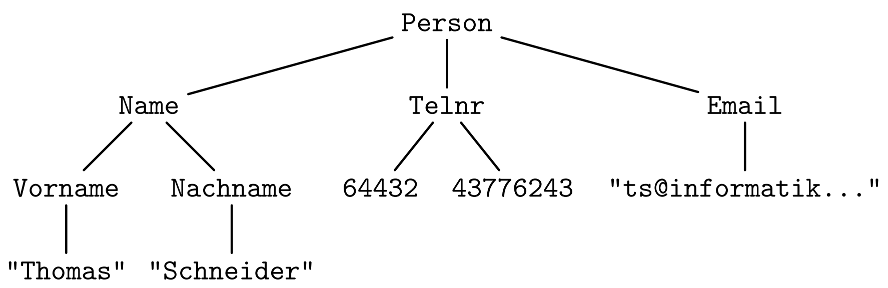
- Markierung von Entitäten durch Tags
- Bildung von Hierarchien
- Gruppieren ähnlicher Entitäten
- Entitäten derselben Gruppe können verschiedene (oder keine) Attribute haben
- Reihenfolge der Attribute *kann* eine Rolle spielen (Mengen oder Listen z. B. von Telefonnummern?)

Beispiel:

```
Person: {Name: {VN: "Thomas", NN: "Schneider"},
         Telnr: 64432,
         Telnr: 43776243,
         Email: "ts@informatik..."}
```


Repräsentation im Baum

```
Person: {Name: {VN: "Thomas", NN: "Schneider"},
         Telnr: 64432,
         Telnr: 43776243,
         Email: "ts@informatik..."}
```

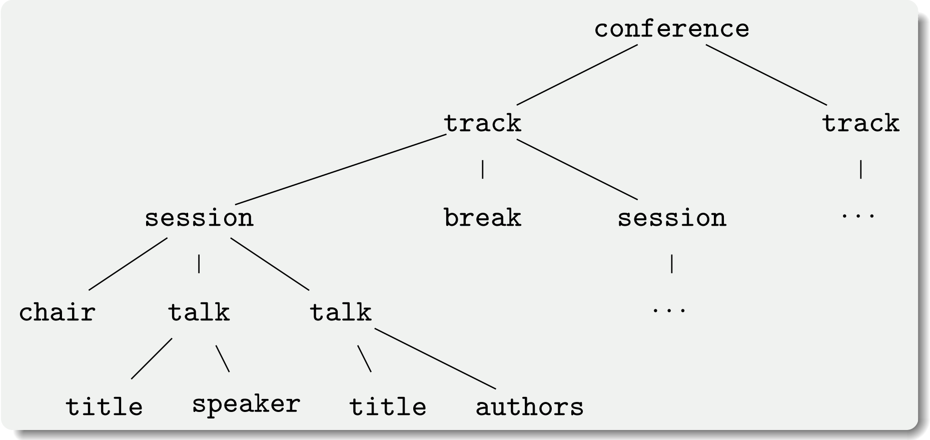


Größeres Bsp.: XML-Dokument für Konferenzprogramm

```
<conference>
  <track>
    <session>
       <chair> F. Angorn </chair>
       <talk>
         <title> The Pushdown Hierarchy </title>
         <speaker> D.J. Gaugal </speaker>
       </talk>
       <talk>
         <title> Trees Everywhere </title>
         <authors> B. Aum, T. Rees </authors>
       </talk>
    </session>
    <break> Coffee </break>
    <session>
       ....
    </session>
  </track>
  <track>
    ....
  </track>
</conference>
```

aus *Tree Automata Techniques and Applications*, S. 230

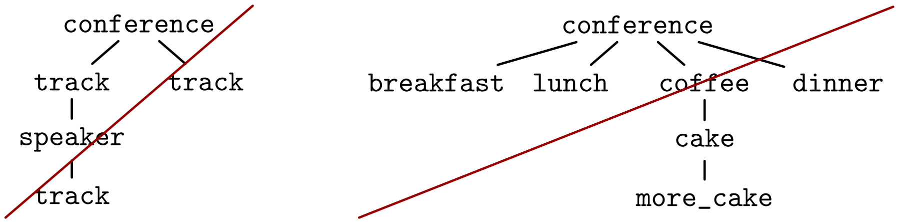
Zugehöriger Baum



aus *Tree Automata Techniques and Applications*, S. 230

- **Ab jetzt:** wir beschreiben nur die Struktur, ignorieren die Daten

Was ist ein gültiges Konferenzdokument?



Mögliche Anforderungen an gültige Konferenzdokumente

- Eine Konferenz kann in mehrere Blöcke (Tracks) geteilt sein.
- Jeder Block (oder die Konf. selbst, wenn sie keine Blöcke hat) ist in Sitzungen aufgeteilt.
- Jede Sitzung hat einen oder mehrere Vorträge.
- Jede Sitzung wird von einer Person geleitet (Chair).
- Jeder Vortrag hat einen Titel und
  - Autor_innen (falls es sich um einen Konferenzbeitrag handelt)
  - oder Vortragende_n (falls es ein eingeladener Vortrag ist).
- Zwischen den Sitzungen kann es Pausen geben.

Gültige Dokumente als Baumsprachen!

Die gelisteten Anforderungen beschreiben eine **Baumsprache** über dem Alphabet $\{\texttt{conference}, \texttt{track}, \texttt{session}, \dots\}$.

Eine solche Beschreibung wird auch **Schema** genannt.

Ein Dokument ist **gültig** für ein Schema, wenn sein Baum zur Baumsprache des Schemas gehört.

Ziele dieses Abschnitts

- Vorstellen von XML-Schemasprachen
- Diskutieren von Verbindungen zur Automatentheorie
- Untersuchen der Ausdrucksstärke von Schemasprachen
- und ihre **Entscheidungsprobleme**

Entscheidungsprobleme für XML-Schemasprachen

Die bekannten Entscheidungsprobleme entsprechen natürlichen Fragen für XML-Dokumente und -Schemasprachen:

Zugehörigkeitsproblem

Ist ein gegebenes Dokument gültig für ein gegebenes Schema?
(im Bsp.: erfüllt ein gegebenes Konf.-dokument die Anforderungen?)

Leerheitsproblem

Gibt es für ein gegebenes Schema gültige Dokumente?
(Enthält das gegebene Schema keinen „Widerspruch"?)

Äquivalenzproblem

Haben zwei Schemata dieselben gültigen Dokumente?
(Wichtig bei der Vereinfachung von Schemata)

Dokumenttypdefinitionen (DTDs)

DTDs sind ein Standard zur Beschreibung gültiger Dokumente

Eine DTD ist eine kontextfreie Grammatik (kfG), deren rechte Regelseiten reguläre Ausdrücke enthalten können

Ableitungsbäume der kfG bilden die Baumsprache, die durch die DTD bestimmt wird

Beispiel-DTD für Konferenzdokumente

```
<!DOCTYPE CONFERENCE [
  <!ELEMENT conference (track+|(session,break?)+)>
  <!ELEMENT track      ((session,break?)+)>
  <!ELEMENT session    (chair,talk+)>
  <!ELEMENT talk       ((title,authors)|(title,speaker))>
  <!ELEMENT chair      (#PCDATA)>
  <!ELEMENT break      (#PCDATA)>
  <!ELEMENT title      (#PCDATA)>
  <!ELEMENT authors    (#PCDATA)>
  <!ELEMENT speaker    (#PCDATA)>
]>
```

, $\widehat{=}$ Verkettung

Beschreibt Bäume, in denen z. B. jeder `conference`-Knoten

- ein oder mehrere `track`-Kinder hat oder
- ein oder mehrere `session`-Kinder hat, zwischen denen einzelne `break`-Geschwister stehen dürfen

$\rightsquigarrow$ beliebige Stelligkeit!

Wir brauchen Symbole mit beliebiger Stelligkeit!

Erweitern unser r-Alphabet:

- U: Menge von **Symbolen ohne Stelligkeit**
- $\Sigma = U \;\cup\; \bigcup_{i \geqslant 0} \Sigma_i$

Endlicher geordneter Baum $T = (P, t)$ über Σ:

- $P \subseteq \mathbb{N}_+^*$ nichtleere endl. präfix-abgeschlossene Menge
- $t : P \to \Sigma$ Funktion mit
  1. Wenn $t(p) \in \Sigma_m$, dann $\{j \mid pj \in P\} = \{1, \ldots, m\}$.
  2. Wenn $t(p) \in U$, dann $\{j \mid pj \in P\} = \{1, \ldots, k\}$ für ein $k \geqslant 0$.

Beschränken uns auf den Fall ohne Stelligkeit (o. S.): $\Sigma = U$

Weitere Begriffe

- **Höhe, Tiefe, Teilbaum**: wie für Bäume mit Stelligkeit
- $a(T_1 \cdots T_n)$: Baum mit a in Wurzel und Teilbäumen $T_1, \ldots, T_n$ direkt darunter
- **Hecke (Hedge)**: Folge $T_1 \cdots T_n$ von Bäumen
  leere Hecke: ε
- $H(\Sigma)$: Menge aller Hecken über Σ

$\rightsquigarrow$ **induktive Charakterisierung von Bäumen:**

- Jede Folge von Bäumen ist eine Hecke.
- Wenn h eine Hecke und $a \in \Sigma$ ein Symbol ist, dann ist $a(h)$ ein Baum

$h = \varepsilon \quad \rightsquigarrow \quad$ schreiben a statt $a(\varepsilon)$

Beispiele für Hecken und Bäume

Sei $\Sigma = \{a, b, c\}$.

ε ist eine Hecke
$\rightsquigarrow$ $a = a()$ ist ein Baum
$\rightsquigarrow$ aa ist eine Hecke
$\rightsquigarrow$ $b(aa)$ ist ein Baum
$\rightsquigarrow$ $ab(aa)c$ ist eine Hecke
$\rightsquigarrow$ $a(ab(aa)c)$ ist ein Baum **T 2.11**

$(a(c(b)cb(ab))$ ist ein Baum **T 2.11 Forts.**

Heckenautomaten

…sind Bottom-up-Automaten auf Bäumen ohne Stelligkeit

Können sie analog zu NEBAs definiert werden?

Nein: Weil Stelligkeit von $a \in \Sigma$ nicht festgelegt ist, brauchten wir 1 Regel $a(q_1, \ldots, q_m) \to q$ **pro $m \geqslant 0$.** **T 2.12**

Abhilfe: Nutzen **reguläre Ausdrücke** über Q in linken Regelseiten

Definition 2.23

Ein **nichtdeterministischer endlicher Heckenautomat (NEHA)** ist ein Quadrupel $\mathcal{A} = (Q, \Sigma, \Delta, F)$, wobei

- Q, Σ, F wie für NEBAs definiert sind und
- Δ eine Menge von Überführungsregeln der Form
$$a(R) \to q$$
ist, wobei $a \in \Sigma$ und $R \subseteq Q^*$ eine reg. Sprache über Q ist.

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.24

Sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$ ein NEHA und $T = (P, t)$ ein Σ-Baum o. S.

- **Berechnung (Run)** von $\mathcal{A}$ auf T ist eine Fkt. $r : P \to Q$ mit:
  Wenn $t(p) = a$, $r(p) = q$ und $m =$ Anzahl von p's Kindern, dann gibt es $a(R) \to q$ in Δ mit $r(p1) \cdots r(pm) \in R$.

Anmerkungen

- Wenn p Blattposition mit Markierung a (d. h. $t(p) = a$), dann darf $a(R) \to q$ nur angewendet werden, wenn $\varepsilon \in R$.
- Repräsentation der reg. Sprache $R \subseteq Q^*$: NEAs, DEAs oder reg. Ausdrücke
  - das ist egal für die Mächtigkeit von NEHAs,
  - aber nicht für Entscheidungsverfahren und deren Komplexität!

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.24 (Fortsetzung)

Sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$ ein NEHA und $T = (P, t)$ ein Σ-Baum o. S.

- **Berechnung (Run)** von $\mathcal{A}$ auf T ist eine Fkt. $r : P \to Q$ mit:

  Wenn $t(p) = a$, $r(p) = q$ und $m =$ Anzahl von p's Kindern,
  dann gibt es $a(R) \to q$ in Δ mit $r(p1) \cdots r(pm) \in R$.

- Ein Run r von $\mathcal{A}$ auf T ist **erfolgreich**, wenn $r(\varepsilon) \in F$.
- $\mathcal{A}$ **akzeptiert** T, wenn es erfolgreichen Run von $\mathcal{A}$ auf T **gibt**.
- Die von $\mathcal{A}$ **erkannte Sprache** ist
  $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$.

Beispiel

Sei $\Sigma = \{a, b, c\}$ und T ein Baum o. S. über Σ.

Der **tiefste gemeinsame Vorgänger** zweier Positionen p_1, p_2 in T ist die Position p, die das **längste gemeinsame Präfix** von p_1, p_2 ist.

Schreibweise: $p = \mathbf{tgV}(p_1, p_2)$ **T 2.13**

$$\begin{aligned} \boldsymbol{L} \; = \{T = (P,t) \mid{} & \text{es gibt } p_1, p_2 \in P \text{ mit} \\ & t(p_1) = t(p_2) = b \text{ und } t(\mathrm{tgV}(p_1,p_2) = c)\} \end{aligned}$$

T 2.13 Forts.

Idee für einen Baumautomaten:

- Gehe in q_b, sobald b gesehen.
  Propagiere q_b nach oben.
- Gehe in q_c, wenn c gesehen und in 2 Kindern q_b.
  Propagiere q_c nach oben.
- Akzeptiere, wenn Wurzel mit q_c markiert.

Beispiel

$$\begin{aligned} \boldsymbol{L} \; = \{T = (P,t) \mid{} & \text{es gibt } p_1, p_2 \in P \text{ mit} \\ & t(p_1) = t(p_2) = b \text{ und } t(\mathrm{tgV}(p_1,p_2) = c)\} \end{aligned}$$

$\mathcal{A} = (Q, \Sigma, \Delta, F)$ mit $Q = \{q_0, q_b, q_c\}$, $F = \{q_c\}$ und

$$\Delta = \{ \underbrace{\begin{matrix} a(Q^*) \to q_0 \\ b(Q^*) \to q_b \\ c(Q^*) \to q_0 \end{matrix}}_{\substack{\text{„Anfangszustand“/} \\ \text{noch kein } b \text{ gefunden}}} \quad \underbrace{\begin{matrix} a(Q^* q_b Q^*) \to q_b \\ c(Q^* q_b Q^*) \to q_b \\ c(Q^* q_b Q^* q_b Q^*) \to q_c \end{matrix}}_{\substack{\text{Propagiere } q_b/ \\ \text{gehe in } q_c}} \quad \underbrace{\begin{matrix} a(Q^* q_c Q^*) \to q_c \\ b(Q^* q_c Q^*) \to q_c \\ c(Q^* q_c Q^*) \to q_c \end{matrix}}_{\text{Propagiere } q_c} \}$$

T 2.13 Forts.

Umwandlung der Beispiel-DTD in einen NEHA (1)

```
<!DOCTYPE CONFERENCE [
  <!ELEMENT conference (track+|(session,break?)+)>
  <!ELEMENT track      ((session,break?)+)>
  <!ELEMENT session    (chair,talk+)>
  <!ELEMENT talk       ((title,authors)|(title,speaker))>
  <!ELEMENT chair      (#PCDATA)>
  ...
  <!ELEMENT speaker    (#PCDATA)>
]>
```

Zugehörige erweiterte kontextfreie Grammatik:

$$\begin{array}{lcl} \texttt{conference} & \to & \texttt{track}^+ + (\texttt{session}\ (\texttt{break} + \varepsilon))^+ \\ \texttt{track} & \to & (\texttt{session}\ (\texttt{break} + \varepsilon))^+ \\ \texttt{session} & \to & \texttt{chair}\ \texttt{talk}^+ \\ \texttt{talk} & \to & (\texttt{title}\ \texttt{authors}) + (\texttt{title}\ \texttt{speaker}) \\ \texttt{chair} & \to & \texttt{DATA} \\ \ldots & & \\ \texttt{speaker} & \to & \texttt{DATA} \end{array}$$

Umwandlung der Beispiel-DTD in einen NEHA (2)

Erweiterte kontextfreie Grammatik (Wdhlg.):

$$
\begin{array}{lcl}
\texttt{conference} & \to & \texttt{track}^+ + (\texttt{session}\ (\texttt{break} + \varepsilon))^+ \\
\texttt{track} & \to & (\texttt{session}\ (\texttt{break} + \varepsilon))^+ \\
\texttt{session} & \to & \texttt{chair}\ \texttt{talk}^+ \\
\texttt{talk} & \to & (\texttt{title}\ \texttt{authors}) + (\texttt{title}\ \texttt{speaker}) \\
\texttt{chair} & \to & \texttt{DATA} \\
\dots & & \\
\texttt{title} & \to & \texttt{DATA}
\end{array}
$$

Startsymbol: hier `conference`

Ableitungsschritt:

- Wähle mit ℓ beschriftetes Blatt, $\ell \in \Sigma$
- Wähle Regel $\ell \to R$ (R: reg. Sprache über Σ, **Inhaltsmodell**)
- Wähle $a_1 \cdots a_n \in R$ und füge Kinder $a_1, \dots, a_n$ zu ℓ hinzu

Beispielableitung: siehe Tafel **T 2.14**

Umwandlung der Beispiel-DTD in einen NEHA (3)

Erweiterte kontextfreie Grammatik (Wdhlg.):

$$
\begin{array}{lcl}
\texttt{conference} & \to & \texttt{track}^+ + (\texttt{session}\ (\texttt{break} + \varepsilon))^+ \\
\texttt{track} & \to & (\texttt{session}\ (\texttt{break} + \varepsilon))^+ \\
\vdots & & \\
\texttt{title} & \to & \texttt{DATA}
\end{array}
$$

Zugehöriger NEHA: $\mathcal{A} = (Q, \Sigma, \Delta, F)$ mit

$$
\begin{array}{lcl}
\Sigma & = & \{\texttt{conference}, \texttt{track}, \texttt{session}, \texttt{talk}, \texttt{chair}, \dots, \texttt{DATA}\} \\
Q & = & \Sigma \\
F & = & \{\texttt{conference}\} \\
\Delta & = & \{\ \texttt{conf}\big(\mathit{track}^+ + (\mathit{session}\ (\mathit{break} + \varepsilon))^+\big) \to \mathit{conf}, \\
 & & \quad \texttt{track}\big((\mathit{session}\ (\mathit{break} + \varepsilon))^+\big) \to \mathit{track}, \\
 & & \quad \vdots \\
 & & \quad \texttt{title}\big(\mathit{DATA}\big) \to \mathit{title}, \\
 & & \quad \texttt{DATA}\big(\big) \to \mathit{DATA}\ \}
\end{array}
$$

Präzise Definition DTD & zugehöriger NEHA

Definition 2.25

Eine **Dokumenttypdefinition (DTD)** ist ein Tupel $D = (\Sigma, s, \Delta)$ mit

- einem Alphabet Σ (ohne Stelligkeit)
- einem **Startsymbol** $s \in \Sigma$ und
- einer Abbildung $\Delta : \Sigma \to$ reguläre Ausdrücke über Σ

(Δ entspricht einer Menge von Regeln – die Folge der Symbole in den Kindern jedes Knotens mit $a \in \Sigma$ muss in $L(\Delta(a))$ sein.)

Zugehöriger NEHA: $\mathcal{A}_D = (Q_D, \Sigma, \Delta_D, F_D)$ mit

- $Q_D = \Sigma$
- $F_D = \{s\}$
- $\Delta_D = \{a(\Delta(a)) \to a \mid a \in \Sigma\}$

Lokale Sprachen

Definition 2.26

- Die von einer DTD D **erzeugte Sprache** ist $L(\mathcal{A}_D)$.
- Eine Baumsprache über Σ heißt **lokal**, wenn sie von einer DTD über Σ erzeugt wird.

Frage:
Wie verhalten sich lokale Sprachen zu NEHA-erkennbaren Spr.?
(Gibt es NEHAs, die keine DTD beschreiben?)

Anwort:

Nicht jede NEHA-erkennbare Sprache ist lokal. (Ja.)

Weil DTDs „immer nur eine Ebene nach unten schauen“ **T 2.15**

(Nicht ausdrückbar:
„alle Sitzungen jeder Konf. haben zusammen $\geqslant 5$ Vortragende“)

Deterministische Inhaltsmodelle

Die W3C[a]-Empfehlung für XML fordert,
dass Inhaltsmodelle **deterministische reguläre Ausdrücke** sind.

[a] *World Wide Web Consortium*, int. Agentur für WWW-Standards

Regulärer Ausdruck r über Σ ist **deterministisch**, falls

- für jedes Wort $w \in \Sigma^*$ und jeden Buchstaben a in w höchstens ein Vorkommen von a in r existiert, auf das a passt.
- Dann lässt sich in Polyzeit ein äquivalenter DEA konstruieren.

$\rightsquigarrow$ Stellt sicher, dass das Zugehörigkeitsproblem für DTDs in Polyzeit lösbar ist.

Deterministische Inhaltsmodelle – Beispiel

Betrachte die Zeile

```
<!ELEMENT talk            ((title,authors)|(title,speaker))>
```

und die zugehörige Regel

$$\texttt{talk} \quad \rightarrow \quad (\texttt{title authors}) + (\texttt{title speaker})$$

Für Wörter über Σ, die mit dem Buchstaben `title` beginnen, ist nicht klar, welchem Vorkommen von `title` im Inhaltsmodell dieser Buchstabe entspricht!

Jetzt genau: deterministische reguläre Ausdrücke

Idee: Sei r ein RA über Σ.

- Markiere das i-te Vorkommen jedes Buchstaben a in r mit a_i.
- Bsp.: $(a+b)^*b(ab)^* \quad \rightsquigarrow \quad (a_1+b_1)^*b_2(a_2b_3)^* =: r'$.
- r ist deterministisch, wenn $L(r')$ keine zwei Wörter ua_iv und ua_jw mit $i \neq j$ enhält.

Etwas Notation:

- RA r über $\Sigma \quad \rightsquigarrow \quad$ markierter RA $\boldsymbol{r'}$ über $\boldsymbol{\Sigma'}$
- wie üblich: $L(r) \subseteq \Sigma^*$ und $L(r') \subseteq \Sigma'^*$

Definition 2.27

Ein **deterministischer RA (DRA)** ist ein RA r über Σ, so dass
für alle Wörter $u, v, w \in \Sigma'^*$ und Zeichen $a \in \Sigma$
mit $ua_iv,\ ua_jw \in L(r')$ gilt: $\quad i = j$. **T 2.16**

Was nützen uns nun DRAs?

Satz 2.28

Zu jedem DRA r kann man in Polynomialzeit einen DEA $\mathcal{A}$ mit $L(\mathcal{A}) = L(r)$ konstruieren.

(Ohne Beweis.)

Folgerung 2.29

Zu jeder deterministischen DTD kann man in Polynomialzeit einen äquivalenten NEHA(DEA) konstruieren.

NEHA(DEA): $\mathcal{A} = (Q, \Sigma, \Delta, F)$, bei dem für alle $a(R) \rightarrow q \in \Delta$ R als DEA gegeben ist.

Und dieses Resultat garantiert nun ...?

Deterministische DTDs sind effizient!

Satz 2.30

Für deterministische DTDs sind in Polynomialzeit lösbar:

- das Zugehörigkeitsproblem
- das Leerheitsproblem
- das Äquivalenzproblem

(Ohne Beweis.)

Zur Erinnerung:

- **Zugehörigkeitsproblem** (Gültigkeit)
  Ist ein gegebenes Dokument gültig für ein gegebenes Schema?
- **Leerheitsproblem** (Widerspruchsfreiheit)
  Gibt es für ein gegebenes Schema gültige Dokumente?
- **Äquivalenzproblem**
  Haben zwei Schemata dieselben gültigen Dokumente?

Sind deterministische DTDs schwächer als allgemeine?

1. Im Allgemeinen **ja**,
2. **aber** es ist entscheidbar, ob eine gegebene DTD äquivalent zu einer deterministischen DTD ist:

Satz 2.31

1. Nicht jede reg. Sprache wird durch einen DRA beschrieben:

$$\{L(r) \mid r \text{ ist DRA}\} \subset \{L(r) \mid r \text{ ist RA}\}$$

2. Das folgende Problem ist in Polynomialzeit entscheidbar.

   Gegeben: DEA $\mathcal{A}$
   Frage: Gibt es einen DRA r mit $L(r) = L(\mathcal{A})$?

   Wenn ein solcher DRA existiert,
   dann kann er in Exponentialzeit konstruiert werden.

Zusammenfassung für deterministische DTDs

Deterministische DTDs ...

- sind **echt schwächer als NEHAs**, weil sie
  - nur **lokale Sprachen** beschreiben
    (sie können keine Bedingungen über Knoten ausdrücken, die durch einen Pfad der Länge > 1 getrennt sind);
  - nur **DRAs** auf rechten Regelseiten erlauben.
- Dafür sind **die wichtigen Entscheidungsprobleme effizient lösbar**.

Ausblick: Lockern der Einschränkungen

Extended DTDs (EDTDs)

- führen durch eine einfache syntaktische Erweiterung aus den lokalen Sprachen heraus
- sind fast äquivalent zu NEHAs
  (beschränkt auf Sprachen, in denen alle Bäume dasselbe Wurzelsymbol haben)
- haben ein in Polynomialzeit lösbares Zugehörigkeits- und Leerheitsproblem

Weitere Einschränkung von EDTDs

- garantiert auch ein in Polynomialzeit lösbares Äquivalenzproblem
- liegt **XML Schema** zugrunde

Damit sind wir am Ende dieses Kapitels.

Vielen Dank.

Literatur für diesen Teil (Basis)

Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, Marc Tommasi.
Tree Automata Techniques and Applications.
http://tata.gforge.inria.fr Nov. 2008.
Kapitel 1
Abschnitt 2.4 (Verbindung zu kontextfreien Wortsprachen)
Abschnitte 8.2.1, 8.2.2, 8.7 (Heckenaut. und XML-Schemasprachen)

Meghyn Bienvenu.
Automata on Infinite Words and Trees.
Vorlesungsskript, Uni Bremen, WS 2009/10.
http://www.informatik.uni-bremen.de/tdki/lehre/ws09/automata/automata-notes.pdf
Kapitel 3

Literatur für diesen Teil (weiterführend)

Anne Brüggemann-Klein, Derick Wood.
One-Unambiguous Regular Languages.
Information and Computation, 142:1998, S. 182–206.
http://dx.doi.org/10.1006/inco.1997.2695
Grundlegende Resultate für deterministische reguläre Ausdrücke.