

## Überblick

# Automatentheorie und ihre Anwendungen

## Teil 2: endliche Automaten auf endlichen Bäumen

Thomas Schneider

7.–28. Mai 2014

- 1 Motivation: semistrukturierte Daten
- 2 Grundbegriffe
- 3 Anwendung 1: Einschließung bei Termersetzungssystemen
- 4 Charakterisierungen erkennbarer Baumsprachen
- 5 Top-down-Baumautomaten
- 6 Abschlusseigenschaften
- 7 Entscheidungsprobleme
- 8 Anwendung 2: XML-Schemasprachen

Und nun ...

Semistrukturierte Daten sind ...

- 1 Motivation: semistrukturierte Daten
- 2 Grundbegriffe
- 3 Anwendung 1: Einschließung bei Termersetzungssystemen
- 4 Charakterisierungen erkennbarer Baumsprachen
- 5 Top-down-Baumautomaten
- 6 Abschlusseigenschaften
- 7 Entscheidungsprobleme
- 8 Anwendung 2: XML-Schemasprachen

- 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..."}
```



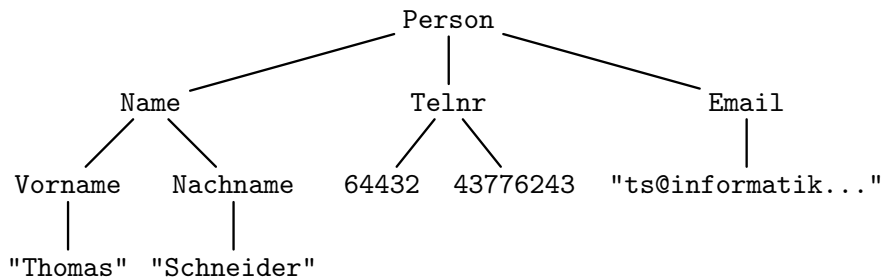
?

# Datenstruktur: Baum

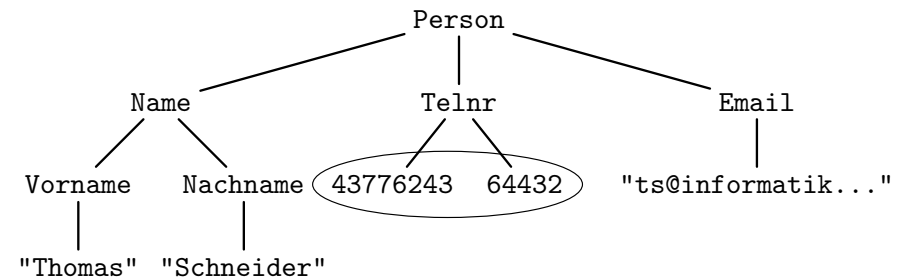
```
Person: {Name: {VN: "Thomas", NN: "Schneider"},
        Telnr: 64432,
        Telnr: 43776243,
        Email: "ts@informatik..."}
```

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

Repräsentation im Baum ist naheliegend:



Ist das derselbe oder ein anderer Baum?

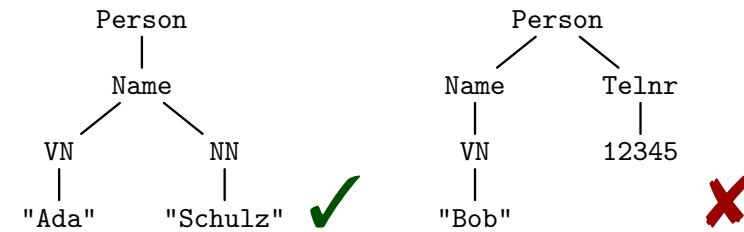


... sind wichtig für semistrukturierte Daten, weil sie ...

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

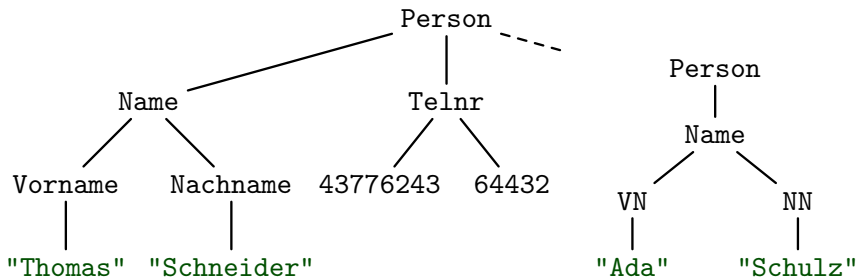
- Schemasprachen zur Beschreibung gültiger Bäume
- Validierung = Erkennen gültiger und ungültiger Bäume
- gängige Schemasprachen benutzen dafür Baumautomaten

Beispiel: jeder Name muss einen Vor- und Nachnamen haben



- beantworten Anfragen mit Daten aus gegebenen Bäumen

Beispiel: gib alle Namen von Personen zurück



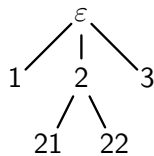
- 1 Motivation: semistrukturierte Daten
- 2 Grundbegriffe
- 3 Anwendung 1: Einschließung bei Termersetzungssystemen
- 4 Charakterisierungen erkennbarer Baumsprachen
- 5 Top-down-Baumautomaten
- 6 Abschlusseigenschaften
- 7 Entscheidungsprobleme
- 8 Anwendung 2: XML-Schemasprachen

# Positionen im Baum

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

Idee: Wurzel ist  $\epsilon$   
 $j$ -ter Nachfolger von  $p$  ist  $pj$

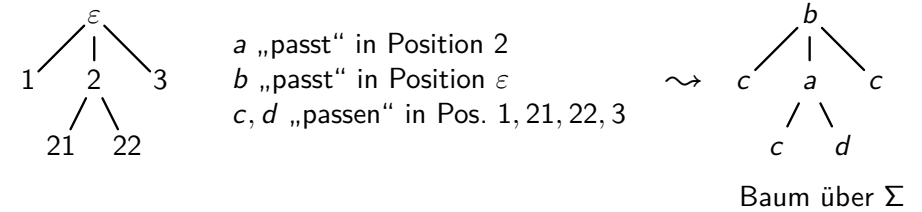
Beispiel:



# Alphabet mit Stelligkeit

- hier: **r-Alphabet**  $\Sigma$  (auf Englisch: *ranked alphabet*)
- nichtleere endliche Menge von Symbolen; jedem Symbol ist eine Stelligkeit  $\in \mathbb{N}_0$  zugeordnet
- $\Sigma_m =$  Menge der Symbole mit Stelligkeit  $m$
- Schreibweise:  $\Sigma = \{a_1/r_1, \dots, a_n/r_n\}$  heißt:  $\Sigma$  enthält die Symbole  $a_i$  mit Stelligkeit  $r_i, i = 1, \dots, n$

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



# Was genau ist eigentlich ein Baum?



?

# Was ist ein Baum?

**Definition 1**

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

- $P \subseteq \mathbb{N}^*$  eine nichtleere endl. präfix-abgeschlossene Menge ist,
- $t : P \rightarrow \Sigma$  eine Funktion ist mit den folgenden Eigenschaften.
  - 1 Wenn  $t(p) \in \Sigma_0$ , dann  $\{j \mid pj \in P\} = \emptyset$ .
  - 2 Wenn  $t(p) \in \Sigma_m, m \geq 1$ , dann  $\{j \mid pj \in P\} = \{1, \dots, m\}$ .

**Erklärungen:**

- $P$ : Menge der vorhandenen Positionen
- Präfix-Abgeschlossenheit: Baum ist wohlgeformt (z. B.: wenn Position 31 existiert, dann auch Position 3 und  $\epsilon$ )

# Was ist ein Baum?

## Definition 1

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

- $P \subseteq \mathbb{N}^*$  eine nichtleere endl. präfix-abgeschlossene Menge ist,
- $t : P \rightarrow \Sigma$  eine Funktion ist mit den folgenden Eigenschaften.
  - 1 Wenn  $t(p) \in \Sigma_0$ , dann  $\{j \mid pj \in P\} = \emptyset$ .
  - 2 Wenn  $t(p) \in \Sigma_m, m \geq 1$ , dann  $\{j \mid pj \in P\} = \{1, \dots, m\}$ .

## Bezeichnungen:

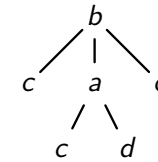
- $pj$  sind die **Kinder** von  $p$
- jedes Präfix von  $p$  ist ein **Vorgänger** von  $p$
- Knoten ohne Kinder (1) sind **Blätter**
- **Höhe** von  $T$ : Länge der längsten Position in  $P$

# Beispiel

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

$$P = \{\varepsilon, 1, 2, 3, 21, 22\}$$

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



- Höhe: 2
- Wurzelsymbol:  $b$
- Blätter: 1, 21, 22, 3

# Bottom-up-Baumautomaten

## Definition 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,
- $\Sigma$  ein  $r$ -Alphabet ist,
- $\Delta$  eine Menge von **Überführungsregeln** der Form

$$a(q_1, \dots, q_m) \rightarrow q$$

ist mit  $m \geq 0, a \in \Sigma_m, q, q_1, \dots, q_m \in Q$ , und

- $F \subseteq Q$  die Menge der **Endzustände** ist.

# Bottom-up-Baumautomaten: Intuitionen

**Zur Erinnerung:** NEBA ist ein Quadrupel  $\mathcal{A} = (Q, \Sigma, \Delta, F)$  mit

- $Q$ : endliche nichtleere Zustandsmenge
- $\Sigma$ :  $r$ -Alphabet
- $\Delta$ : Menge von Überführungsregeln  $a(q_1, \dots, q_m) \rightarrow q$   
 $(m \geq 0, a \in \Sigma_m, q, q_1, \dots, q_m \in Q)$
- $F \subseteq Q$ : Menge der Endzustände

**Bedeutung der Überführungsregeln:** wenn  $\mathcal{A}$

- in Position  $p$  Zeichen  $a$  liest
  - und in  $p$ 's Kindern Zustände  $q_1, \dots, q_m$  eingenommen hat,
- dann darf  $\mathcal{A}$  in  $p$  Zustand  $q$  einnehmen. (Skizze siehe Tafel) •

## Bottom-up-Baumautomaten: Intuitionen

Zur Erinnerung: NEBA ist ein Quadrupel  $\mathcal{A} = (Q, \Sigma, \Delta, F)$  mit

- $Q$ : endliche nichtleere Zustandsmenge
- $\Sigma$ :  $r$ -Alphabet
- $\Delta$ : Menge von Überführungsregeln  $a(q_1, \dots, q_m) \rightarrow q$   
( $m \geq 0, a \in \Sigma_m, q, q_1, \dots, q_m \in Q$ )
- $F \subseteq Q$ : Menge der Endzustände

↪ **Andere Betrachtungsweise:**

- $\mathcal{A}$  markiert Eingabebaum  $T$  von u. nach o. mit Zuständen
- $\mathcal{A}$  akzeptiert  $T$ , wenn  $\mathcal{A}$  in der Wurzel einen EZ einnimmt

## Bottom-up-Baumautomaten: Intuitionen

Zur Erinnerung: NEBA ist ein Quadrupel  $\mathcal{A} = (Q, \Sigma, \Delta, F)$  mit

- $Q$ : endliche nichtleere Zustandsmenge
- $\Sigma$ :  $r$ -Alphabet
- $\Delta$ : Menge von Überführungsregeln  $a(q_1, \dots, q_m) \rightarrow q$   
( $m \geq 0, a \in \Sigma_m, q, q_1, \dots, q_m \in Q$ )
- $F \subseteq Q$ : Menge der Endzustände

Was sind dann die Anfangszustände?

- $\ddot{U}$ -Regeln in  $\Delta$  mit  $m = 0$  deklarieren "zeichenspezifische" AZ:  
 $a \rightarrow q$  bedeutet: *% Kurzschreibweise für  $a() \rightarrow q$*   
 $\mathcal{A}$  darf in mit  $a$  markierten Blättern in  $q$  starten

## Berechnungen (analog zu NEAs)

### Definition 3

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

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

Also gilt:

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

## Berechnungen, Akzeptanz und erkannte Sprache

### Definition 3

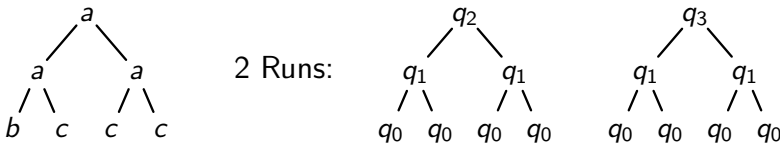
Sei  $\mathcal{A} = (Q, \Sigma, \Delta, F)$  ein NEBA und  $T = (P, t)$  ein  $\Sigma$ -Baum.

- **Berechnung (Run)** von  $\mathcal{A}$  auf  $T$  ist eine Fkt.  $r : P \rightarrow Q$  mit:
  - Wenn  $t(p) = a \in \Sigma_0$  und  $r(p) = q$ , dann  $a \rightarrow q \in \Delta$ .
  - Wenn  $t(p) = b \in \Sigma_m$  ( $m \geq 1$ ) und  $r(p) = q$  und wenn  $r(p1) = q_1, \dots, r(pm) = q_m$ , dann  $b(q_1, \dots, q_m) \rightarrow q \in \Delta$ .
- Ein Run  $r$  von  $\mathcal{A}$  auf  $T$  ist **akzeptierend**, wenn  $r(\varepsilon) \in F$ .
- $\mathcal{A}$  **akzeptiert**  $T$ , wenn es einen akz. 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  $\mathcal{A} = (\{q_0, \dots, q_3\}, \Sigma, \Delta, \{q_2\})$   
 mit  $\Delta = \{ b \rightarrow q_0, \quad c \rightarrow q_0,$   
 $a(q_0, q_0) \rightarrow q_1,$   
 $a(q_1, q_1) \rightarrow q_2, \quad a(q_1, q_1) \rightarrow q_3 \}.$

- Dann gibt es auf dem Baum



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

## 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

$$\Delta = \{ \quad c \rightarrow q_c, \quad a(q_c, q_d) \rightarrow q_f,$$

$$\quad d \rightarrow q_d,$$

$$\quad d \rightarrow q_f,$$

$$b(q_f) \rightarrow q_f, \quad a(q_f, q_f) \rightarrow q_f \}$$

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

Beispielbaum und -run: siehe Tafel

## Erkennbare Baumsprache

### Definition 4

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

## Determinismus

### Definition 5

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

Enthält  $\Delta$  für jedes  $a \in \Sigma_m$  und alle  $(q_1, \dots, q_m) \in Q^m$  **höchstens eine**<sup>1</sup> Regel  $a(q_1, \dots, q_m) \rightarrow q$

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

↪ 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?

<sup>1</sup>„höchstens eine“ ist günstiger als „genau eine“: vermeidet Papierkorbzustd.



# Potenzmengenkonstruktion

# Und nun ...

## Antwort

Ja, DEBAs und NEBAs sind gleichmächtig.

## Satz 6

Sei  $L$  eine erkennbare Baumsprache.

Dann gibt es einen DEBA, der  $L$  erkennt.

Beweis: siehe Tafel.

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

- 1 Motivation: semistrukturierte Daten
- 2 Grundbegriffe
- 3 Anwendung 1: Einschließung bei Termersetzungssystemen
- 4 Charakterisierungen erkennbarer Baumsprachen
- 5 Top-down-Baumautomaten
- 6 Abschlusseigenschaften
- 7 Entscheidungsprobleme
- 8 Anwendung 2: XML-Schemasprachen

# Motivation: Termersetzungssysteme ...

- sind ein mächtiges Berechnungsmodell (Turing-vollständig)
- liegen u. a. Logik- und funktionaler Programmierung zugrunde

## Beispiel 7 (symbolische Addition natürlicher Zahlen)

- $0, 1, 2, \dots$  werden dargestellt als  $0, succ(0), succ(succ(0)), \dots$
- Ersetzungsregeln:

$$\begin{aligned}
 R1 \quad & plus(0, y) \rightarrow y \\
 R2 \quad & plus(succ(x), y) \rightarrow succ(plus(x, y))
 \end{aligned}$$

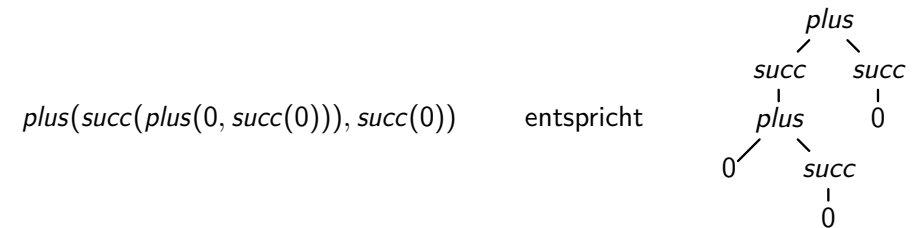
- Anwendung auf Beispielausdruck:

$$\begin{aligned}
 plus(succ(plus(0, succ(0))), succ(0)) &\stackrel{R1}{\mapsto} plus(succ(succ(0)), succ(0)) \\
 &\stackrel{R2}{\mapsto} succ(plus(succ(0), succ(0))) \\
 &\stackrel{R2}{\mapsto} succ(succ(plus(0, succ(0)))) \\
 &\stackrel{R1}{\mapsto} succ(succ(succ(0)))
 \end{aligned}$$

# Grundbegriffe

- **r-Alphabet**  $\Sigma$  wie bekannt (Symbole mit Stelligkeiten)
- endliche Menge  $V$  von **Variablen** (0-stellig)
- **Terme** über  $\Sigma, V$ , geschrieben  $\mathcal{T}(\Sigma, V)$ :
  - Alle 0-stelligen Symbole sind Terme:  $\Sigma_0 \cup V \subseteq \mathcal{T}(\Sigma, V)$
  - Wenn  $m \geq 1$ ,  $a \in \Sigma_m$  und  $t_1, \dots, t_m \in \mathcal{T}(\Sigma, V)$ , dann  $a(t_1, \dots, t_m) \in \mathcal{T}(\Sigma, V)$ .
- **Grundterme**: variablenfreie Terme

**Beobachtung:** Terme sind nur eine andere Schreibweise für Bäume.





## Regeln und deren Anwendung

Termersetzungsgesetze sind Paare  $\ell \rightarrow r$  mit

$$\ell, r \in \mathcal{T}(\Sigma, V) \quad (+ \text{ gewisse Einschränkungen, wg. Terminierung})$$

Regelanwendung:

- **Substitution:** partielle Abbildung  $\sigma : V \rightarrow \mathcal{T}(\Sigma, V)$

Bsp.: wenn  $t = plus(0, y)$  und  $\sigma = \{y \mapsto succ(0)\}$ ,  
dann  $t\sigma = plus(0, succ(0))$

- Term  $t$  **passt in** Term  $s$ , wenn es Subst.  $\sigma$  gibt mit  $t\sigma = s$ .

Bsp.:  $plus(0, y)$  passt in  $plus(0, succ(0))$  mit  $\sigma = \{y \mapsto succ(0)\}$

- $\ell \rightarrow r$  auf  $t$  **anwendbar**, wenn  $\ell$  in einen Subterm  $t'$  von  $t$  passt.

**Ergebnis:**  $t$ , wobei  $t'$  durch  $r\sigma$  ersetzt wurde

Bsp.: Anwendung von  $plus(0, y) \rightarrow y$  auf  $succ(plus(0, succ(0)))$   
ergibt  $succ(succ(0))$

## Und nun ...

- 1 Motivation: semistrukturierte Daten
- 2 Grundbegriffe
- 3 Anwendung 1: Einschließung bei Termersetzungssystemen
- 4 Charakterisierungen erkennbarer Baumsprachen
- 5 Top-down-Baumautomaten
- 6 Abschlusseigenschaften
- 7 Entscheidungsprobleme
- 8 Anwendung 2: XML-Schemasprachen

## Regelanwendbarkeit mittels Baumautomaten entscheiden

Zentrales Entscheidungsproblem für TE-Systeme:

**Einschließungsproblem (Encompassment)**

Gegeben Grundterm  $t$  und Regel  $\ell \rightarrow r$ ,  
gibt es einen Subterm  $t'$  von  $t$ , in den  $\ell$  passt? ( $t$  **schließt**  $\ell$  **ein**.)

Genauer: gibt es Subterm  $t'$  von  $t$  und Substitution  $\sigma$  mit  $\ell\sigma = t'$ ?

**Satz 8**

Wenn in  $\ell$  jede Variable höchstens einmal vorkommt,  
dann ist  $L = \{t \mid t \text{ schließt } \ell \text{ ein}\}$  NEBA-erkennbar.

$L$  wird sogar durch einen NEBA mit  $O(|\ell|)$  Zuständen erkannt.

Beweis: siehe Tafel. ●

Lineare Determinisierung ähnlich wie im 1. Teil, Textsuche.

## 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

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

$\rightsquigarrow L(\mathcal{A}) = \{T \mid \text{alle Wurzel-Blatt-Pfade in } T \text{ haben gerade Länge}\}$ .  
 $\neq \{T \mid T \text{ hat gerade Höhe}\}$  (Bsp. s. Tafel) ●

**Frage:** Sind die folgenden Baumsprachen erkennbar?

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

$$L_2 = \{T \mid T \text{ ist vollständiger Binärbaum}\}$$
 ●

Antwort: **Nein.** Idee s. Tafel ●

## Pumping-Lemma: Hilfsbegriffe

- **Variable:** zusätzliches nullstelliges Symbol  $x \notin \Sigma_0$
- **(unärer) Kontext:**  
Baum über  $\Sigma \cup \{x\}$ , in dem ein Blatt mit  $x$  markiert ist
- **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
  - $C^n$  induktiv definiert:

$$C^0 = C_0$$

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

## Pumping-Lemma

### Satz 9 (Pumping-Lemma)

Wenn  $L$  eine erkennbare Baumsprache über dem  $r$ -Alphabet  $\Sigma$  ist, dann gibt es eine Konstante  $p \in \mathbb{N}$ , so dass für alle Bäume  $T \in L$  mit  $\text{Höhe}(T) \geq p$  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: siehe Tafel.



## Anwendung des Pumping-Lemmas

Zur Erinnerung:

### Satz 9 (Pumping-Lemma)

Wenn  $L$  eine erkennbare Baumsprache über dem  $r$ -Alphabet  $\Sigma$  ist,

dann gibt es eine Konstante  $p \in \mathbb{N}$ , so dass für alle Bäume  $T \in L$  mit  $\text{Höhe}(T) \geq p$  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  $i \in \mathbb{N}$ .

Benutzen Kontraposition:

Wenn es für alle Konstanten  $p \in \mathbb{N}$  einen Baum  $T \in L$  mit  $\text{Höhe}(T) \geq p$  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. ◀ Bsp.: s. Tafel

## Der Satz von Myhill-Nerode für Baumsprachen

Ziel: notwendige und hinreichende Bedingung für Erkennbarkeit

### Definition 10

Sei  $L$  eine Baumsprache über  $\Sigma$ .

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

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

### Satz 11 (Myhill-Nerode)

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

Ohne Beweis. Beispiel siehe Tafel.

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? ☺

- 1 Motivation: semistrukturierte Daten
- 2 Grundbegriffe
- 3 Anwendung 1: Einschließung bei Termersetzungssystemen
- 4 Charakterisierungen erkennbarer Baumsprachen
- 5 Top-down-Baumautomaten
- 6 Abschlusseigenschaften
- 7 Entscheidungsprobleme
- 8 Anwendung 2: XML-Schemasprachen



## Nein, nur die Runs werden „umgedreht“.

### Idee: Top-down-Baumautomaten

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

### Definition 12

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,
- $\Sigma$  ein  $r$ -Alphabet ist,
- $\Delta$  eine Menge von Überführungsregeln der Form

$$(a, q) \rightarrow (q_1, \dots, q_m)$$

ist mit  $m \geq 0$ ,  $a \in \Sigma_m$ ,  $q, q_1, \dots, q_m \in Q$ , und

- $I \subseteq Q$  die Menge der Anfangszustände ist.

## Berechnungen, Akzeptanz und erkannte Sprache

### Definition 13

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

- **Berechnung (Run)** von  $\mathcal{A}$  auf  $T$  ist eine Fkt.  $r : P \rightarrow Q$  mit:
  - $r(\varepsilon) \in I$
  - Wenn  $t(p) = a \in \Sigma_0$  und  $r(p) = q$ , dann  $(a, q) \rightarrow () \in \Delta$ .
  - Wenn  $t(p) = a \in \Sigma_m$  ( $m \geq 1$ ) und  $r(p) = q$  und wenn  $r(p1) = q_1, \dots, r(pm) = q_m$ , dann gibt es eine Regel  $(a, q) \rightarrow (q_1, \dots, q_m) \in \Delta$ .
- $\mathcal{A}$  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  $\Delta$  müssen nur erlauben, von der Wurzel bis zu allen Blättern „durchzukommen“.

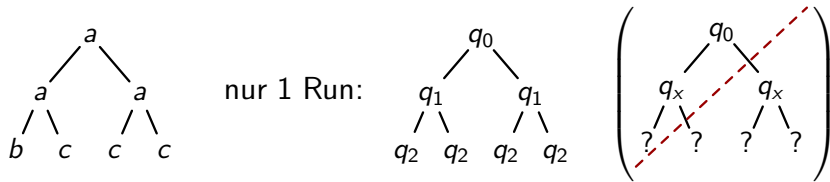
## Beispiel 1

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

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

$$\Delta = \left\{ \begin{array}{l} (a, q_0) \rightarrow (q_1, q_1), \quad (b, q_2) \rightarrow (), \\ (a, q_1) \rightarrow (q_2, q_2), \quad (c, q_2) \rightarrow (), \\ (a, q_0) \rightarrow (q_x, q_x) \end{array} \right\}$$

- Dann gibt es auf dem Baum



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

## Beispiel 2

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

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

$$\Delta = \left\{ \begin{array}{l} (a, q_0) \rightarrow (q_0, q_0), \quad b(q_0) \rightarrow q_0, \quad (c, q_c) \rightarrow (), \\ (a, q_0) \rightarrow (q_c, q_d), \quad (d, q_d) \rightarrow (), \\ (d, q_0) \rightarrow () \end{array} \right\}$$

$(a, q_0) \rightarrow (q_d, q_d)$  überflüssig:  $(a, q_0) \rightarrow (q_0, q_0)$  und  $(d, q_0) \rightarrow ()$

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

$$\Delta = \left\{ \begin{array}{l} c \rightarrow q_c, \quad b(q_f) \rightarrow q_f, \quad a(q_c, q_d) \rightarrow q_f, \\ d \rightarrow q_d, \quad a(q_f, q_f) \rightarrow q_f, \\ d \rightarrow q_f \end{array} \right\}$$

## Was sagt uns das über das Verhältnis NETDBAs : NEBAs?

## Determinisierung von NETDBAs

Sie sind gleichmächtig!

Satz 14

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

Beweis: siehe Tafel.

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

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

$$\Delta = \left\{ \begin{array}{l} (a, q_0) \rightarrow (q_0, q_0), \quad b(q_0) \rightarrow q_0, \quad (c, q_c) \rightarrow (), \\ (a, q_0) \rightarrow (q_c, q_d), \quad (d, q_d) \rightarrow (), \\ \blacktriangle \text{ Nichtdeterminismus!} \quad (d, q_0) \rightarrow () \end{array} \right\}$$

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

$$\Delta = \left\{ \begin{array}{l} c \rightarrow q_c, \quad b(q_f) \rightarrow q_f, \quad a(q_c, q_d) \rightarrow q_f, \\ d \rightarrow q_d, \quad a(q_f, q_f) \rightarrow q_f, \\ d \rightarrow q_f \end{array} \right\}$$

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 Notation für Terme von Folie 32)

**Frage:** Welcher DETDBA erkennt  $L$ ?

**Antwort:** Keiner!

### Lemma 15

*$L$  wird von keinem DETDBA erkannt.*

Beweis: siehe Tafel. ●

### Satz 16

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

## Und nun ...

- 1 Motivation: semistrukturierte Daten
- 2 Grundbegriffe
- 3 Anwendung 1: Einschließung bei Termersetzungssystemen
- 4 Charakterisierungen erkennbarer Baumsprachen
- 5 Top-down-Baumautomaten
- 6 Abschlusseigenschaften
- 7 Entscheidungsprobleme
- 8 Anwendung 2: XML-Schemasprachen

## Zur Erinnerung

Die Menge der erkennbaren Baumsprachen ist abgeschlossen unter

- **Vereinigung**, wenn gilt:  
Falls  $L_1, L_2$  erkennbar, so auch  $L_1 \cup L_2$ .
- **Komplement**, wenn gilt:  
Falls  $L$  erkennbar, so auch  $\bar{L}$ .
- **Durchschnitt**, wenn gilt:  
Falls  $L_1, L_2$  erkennbar, so auch  $L_1 \cap L_2$ .

### Quiz

Unter welchen Operationen gilt Abgeschlossenheit?

- Vereinigung?
- Komplement?
- Durchschnitt?

## Abgeschlossenheit

### Satz 17

*Die Menge der NEBA-erkennbaren Sprachen ist abgeschlossen unter den Operationen  $\cup, \cap, \bar{\phantom{x}}$ .*

**Beweis:** Direkte Konsequenz aus den folgenden Lemmata. □

## Abgeschlossenheit unter Vereinigung

### Lemma 18

Seien  $\mathcal{A}_1, \mathcal{A}_2$  NEBAs über  $\Sigma$ .

Dann gibt es einen NEBA  $\mathcal{A}_3$  mit  $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ .

**Beweis:** 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.

► *Idee wie für Wortautomaten: vereinige  $\mathcal{A}_1$  und  $\mathcal{A}_2$ .*

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

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

## Abgeschlossenheit unter Komplement

### Lemma 19

Sei  $\mathcal{A}$  ein NEBA über  $\Sigma$ .

Dann gibt es einen NEBA  $\mathcal{A}^c$  mit  $L(\mathcal{A}^c) = \overline{L(\mathcal{A})}$ .

**Beweis:**

► *Idee wie für Wortautomaten:*

- Umwandlung in DEBA
- Vertauschen von End- und Nicht-Endzuständen

O. B. d. A. sei  $\mathcal{A} = (Q, \Sigma, \Delta, F)$  ein DEBA

mit **genau einem** Run pro Eingabebaum (Satz 6).

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

## Abgeschlossenheit unter Durchschnitt

... folgt direkt aus der Abgeschlossenheit unter  $\cap$  und  $\bar{\phantom{x}}$ :

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

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

## Und nun ...

- 1 Motivation: semistrukturierte Daten
- 2 Grundbegriffe
- 3 Anwendung 1: Einschließung bei Termersetzungssystemen
- 4 Charakterisierungen erkennbarer Baumsprachen
- 5 Top-down-Baumautomaten
- 6 Abschlusseigenschaften
- 7 Entscheidungsprobleme
- 8 Anwendung 2: XML-Schemasprachen



## Das Leerheitsproblem

### Zur Erinnerung:

Gegeben: NEBA  $\mathcal{A}$

Frage: Gilt  $L(\mathcal{A}) = \emptyset$ ?

Mengenschreibweise:  $\{\text{NEBA } \mathcal{A} \mid L(\mathcal{A}) = \emptyset\}$

### Satz 20

*Das Leerheitsproblem für NEBAs ist entscheidbar.*

Beweis: siehe Tafel. ●

**Komplexität:** P-vollständig (Wegsuche in Hypergraphen)

## Das Zugehörigkeitsproblem

für NEAs: „Wortproblem“

### Zur Erinnerung:

Gegeben: NEBA  $\mathcal{A}$  und Baum  $T = (P, t)$

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

Mengenschreibweise:  $\{(\mathcal{A}, T) \mid T \in L(\mathcal{A})\}$

### Satz 21

*Das Zugehörigkeitsproblem für NEBAs ist entscheidbar.*

Beweis: Reduktion zum Leerheitsproblem – siehe Tafel. ●

**Komplexität:** P-vollständig

(folgt aus Reduktion per Produktautomat

alternativ: modifizierte Wegsuche in Hypergraphen)

## Das Äquivalenzproblem

### Zur Erinnerung:

Gegeben: NEBAs  $\mathcal{A}_1, \mathcal{A}_2$

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

Mengenschreibweise:  $\{(\mathcal{A}_1, \mathcal{A}_2) \mid L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$

### Satz 22

*Das Äquivalenzproblem ist entscheidbar.*

Beweis: analog zum Beweis für NEAs. Benutze:

$$L(\mathcal{A}_1) = L(\mathcal{A}_2) \Leftrightarrow (L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)}) \cup (\overline{L(\mathcal{A}_1)} \cap L(\mathcal{A}_2)) = \emptyset$$

**Komplexität:** für NEBAs **EXPTIME**-vollständig, für DEBAs in **P**

## Das Universalitätsproblem

### Zur Erinnerung:

Gegeben: NEBA  $\mathcal{A}$

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

Mengenschreibweise:  $\{\mathcal{A} \mid L(\mathcal{A}) = \mathcal{T}(\Sigma)\}$

### Satz 23

*Das Universalitätsproblem ist entscheidbar.*

Beweis: analog zum Beweis für NEAs. Benutze:

$$L(\mathcal{A}) = \mathcal{T}(\Sigma) \Leftrightarrow \overline{L(\mathcal{A})} = \emptyset$$

**Komplexität:** für NEBAs **EXPTIME**-vollständig, für DEBAs in **P**



# Und nun ...

# Zur Erinnerung: semistrukturierte Daten

- 1 Motivation: semistrukturierte Daten
- 2 Grundbegriffe
- 3 Anwendung 1: Einschließung bei Termersetzungssystemen
- 4 Charakterisierungen erkennbarer Baumsprachen
- 5 Top-down-Baumautomaten
- 6 Abschlusseigenschaften
- 7 Entscheidungsprobleme
- 8 Anwendung 2: XML-Schemasprachen

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?)

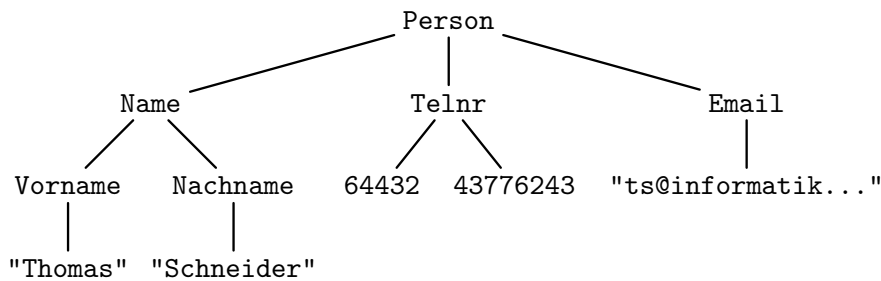
Beispiel:

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

## Repräsentation im Baum

## Größeres Bsp.: XML-Dokument für Konferenzprogramm

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

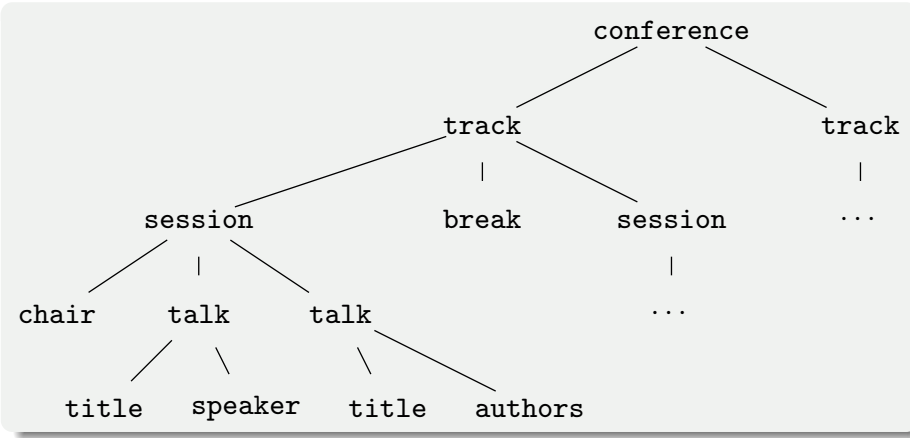


```
<conference>
  <track>
    <session>
      <chair> F. Angorn </chair>
      <talk>
        <title> The Pushdown Hierarchy </title>
        <speaker> D.J. Gaugal </speaker>
      </talk>
      <talk>
        <title> Trees Everywhere </title>
        <authors> B. Aum, T. Rees </authors>
      </talk>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

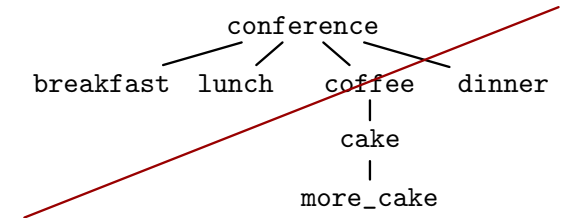
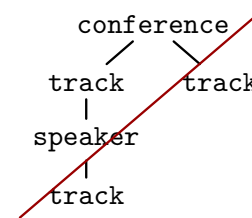
aus *Tree Automata Techniques and Applications*, S. 230

## Zugehöriger Baum

## Was ist ein gültiges Konferenzdokument?



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



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

## 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 hat einen Vorsitzenden (Chair).
- Jeder Vortrag hat einen Titel und
  - Autoren (falls es sich um einen Konferenzbeitrag handelt)
  - oder Vortragenden (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  $\{\text{conference, track, } \dots \}$ .
- Eine solche Beschreibung wird auch **Schema** genannt.
- Ein Dokument ist **gültig** für ein Schema, wenn sein Baum zur Baumsprache des Schemas gehört.

### Ziel dieses Abschnitts

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

# Entscheidungsprobleme für XML-Schemasprachen

# Dokumenttypdefinitionen (DTDs)

Entscheidungsprobleme, die wir für NEBAs kennen gelernt haben, 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 gegebene Schemata dieselbe Menge von gültigen Dokumenten?

- sind eine Möglichkeit, gültige Dokumente zu beschreiben
- sind kontextfreie Grammatiken (kfG), deren rechte Regelseiten reguläre Ausdrücke enthalten können
- die 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)>
]>
```

... beschreibt Bäume, in denen

- jeder conference-Knoten **beliebige Stelligkeit!**
  - **mindestens ein track-Kind** oder
  - **mindestens ein session-Kind** hat und
  - zwischen session-Kindern break-Geschwister erlaubt sind
- jeder track-Knoten ...

## Wir brauchen Symbole mit beliebiger Stelligkeit!

Erweitern unser r-Alphabet:

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

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

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

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

## Weitere Begriffe

- **Höhe, Tiefe, Teilbaum:** wie für Bäume mit Stelligkeit
- $a(T_1, \dots, T_n)$ : Baum mit  $a$  in Wurzel und Teilbäumen  $T_1, \dots, T_n$  direkt darunter
- **Hecke (Hedge):** Folge  $T_1 \dots T_n$  von Bäumen o. S.

### ↪ induktive Charakterisierung von Bäumen o. S.:

- Jede Folge von Bäumen o. S. ist eine Hecke. (Das schließt die leere Folge ein.)
- Wenn  $h$  eine Hecke und  $a \in \Sigma$  ein Symbol ist, dann ist  $a(h)$  ein Baum o. S. ◀ Bsp.: s. Tafel •
- $h = \varepsilon \rightsquigarrow$  schreiben  $a$  statt  $a(\varepsilon)$
- $H(\Sigma)$ : Menge aller Hecken über  $\Sigma$

## Heckenautomaten

- ... sind Bottom-up-Automaten auf Bäumen o. S.
- Können sie analog zu NEBAs definiert werden? **Nein:**
  - Weil Stelligkeit von  $a \in \Sigma$  nicht festgelegt ist, brauchten wir 1 Regel  $a(q_1, \dots, q_m) \rightarrow q$  **pro**  $m \geq 0$
  - **Abhilfe:** Nutzen reguläre Ausdr. über  $Q$  in linken Regelseiten

### Definition 24

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

- $Q, \Sigma, F$  wie für NEBAs definiert sind und
- $\Delta$  eine Menge von Überführungsregeln der Form

$$a(R) \rightarrow q$$

ist, wobei  $a \in \Sigma$  und  $R \subseteq Q^*$  eine reg. Sprache über  $Q$  ist.

## Berechnungen, Akzeptanz und erkannte Sprache

### Definition 25

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

- **Berechnung (Run)** von  $\mathcal{A}$  auf  $T$  ist eine Fkt.  $r : P \rightarrow Q$  mit:  
Wenn  $t(p) = a$ ,  $r(p) = q$  und  $m = \text{Anzahl von } p\text{'s Kindern}$ , dann gibt es  $a(R) \rightarrow q$  in  $\Delta$  mit  $r(p_1) \dots r(p_m) \in R$ .
- Ein Run  $r$  von  $\mathcal{A}$  auf  $T$  ist **akzeptierend**, wenn  $r(\varepsilon) \in F$ .
- $\mathcal{A}$  **akzeptiert**  $T$ , wenn es einen akz. Run von  $\mathcal{A}$  auf  $T$  **gibt**.
- Die von  $\mathcal{A}$  **erkannte Sprache** ist  $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$ .

**Anmerkung und Beispiel:** siehe Tafel •

## 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 break (#PCDATA)>
  <!ELEMENT title (#PCDATA)>
]>
```

### Zugehörige erweiterte kontextfreie Grammatik:

```
conference → track+ + (session (break + ε))+
track → (session (break + ε))+
session → chair talk+
talk → (title authors) + (title speaker)
chair → DATA
break → DATA
title → DATA
```

## Umwandlung der Beispiel-DTD in einen NEHA (2)

## Erweiterte kontextfreie Grammatik (Wdhlg.):

```

conference → track+ + (session (break + ε))+
track      → (session (break + ε))+
session    → chair talk+
talk       → (title authors) + (title speaker)
chair      → DATA
break      → DATA
title      → DATA

```

Rechte Regelseiten heißen **Inhaltsmodell**.

Beispielableitung: siehe Tafel

## Umwandlung der Beispiel-DTD in einen NEHA (3)

## Erweiterte kontextfreie Grammatik (Wdhlg.):

```

conference → track+ + (session (break + ε))+
track      → (session (break + ε))+
⋮
title      → DATA

```

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

```

Σ = {conference, track, session, talk, chair, ..., DATA}
Q = Σ
F = {conference}
Δ = { conf(track+ + (session (break + ε))+) → conf,
      track((session (break + ε))+) → track,
      ⋮
      title(DATA) → title,
      DATA() → DATA }

```

## Formalisierung von DTDs und den zugehörigen NEHAs

## Definition 26

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

- einem Alphabet o. S.  $\Sigma$ ,
- einem **Startsymbol**  $s \in \Sigma$  und
- einer Abbildung  $\Delta : \Sigma \rightarrow$  reguläre Ausdrücke über  $\Sigma$ :  
( $\Delta$  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

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

## Lokale Sprachen

## Definition 27

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

## Frage:

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

## Antwort:

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

Weil DTDs „immer nur eine Ebene nach unten schauen“  
(Nicht ausdrückbar:  
„alle Sitzungen jeder Konf. haben zusammen  $\geq 5$  Vortragende“)

## Deterministische Inhaltsmodelle

Die W3C<sup>a</sup>-Empfehlung für XML fordert, dass Inhaltsmodelle **deterministische reguläre Ausdrücke** sind.

<sup>a</sup>World Wide Web Consortium, int. Agentur für WWW-Standards

Regulärer Ausdruck  $r$  über  $\Sigma$  ist **deterministisch**, falls

- für jedes Wort  $w$  über  $\Sigma$  und jeden Buchstaben  $a$  in  $w$  höchstens ein Vorkommen von  $a$  in  $r$  existiert, auf den  $a$  passt.
  - Dann lässt sich in Poly-zeit ein äquivalenter DEA konstruieren.
- ↪ Stellt sicher, dass das Zugehörigkeitsproblem für DTDs in Poly-zeit lösbar ist.

## Deterministische Inhaltsmodelle – Beispiel

Betrachte die Zeile

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

und die zugehörige Regel

```
talk → (title authors) + (title speaker)
```

Für Wörter über  $\Sigma$ , 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  $\Sigma$ .

- Markiere das  $i$ -te Vorkommen jedes Buchstaben  $a$  in  $r$  mit  $a_i$ .
- Bsp.:  $(a + b)^* b(ab)^* \rightsquigarrow (a_1 + b_1)^* b_2(a_2 b_3)^* =: r'$ .
- $r$  ist deterministisch, wenn  $L(r')$  keine zwei Wörter  $ua_i v$  und  $ua_j w$  mit  $i \neq j$  enthält.

Etwas Notation:

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

### Definition 28

Ein **deterministischer RA (DRA)** ist ein RA  $r$  über  $\Sigma$ , so dass für alle Wörter  $u, v, w \in \Sigma'^*$  und Zeichen  $a \in \Sigma$  mit  $ua_i v, ua_j w \in L(r')$  gilt:  $i = j$ . Bsp.: siehe Tafel. •

## Was nützen uns nun DRAs?

### Satz 29

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

(Ohne Beweis.)

### Folgerung 30

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

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

**Und dieses Resultat garantiert nun ... ?**

## Deterministische DTDs sind effizient!

### Satz 31

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 2 gegeb. Schemata dieselbe Menge von gültigen Dok.-en?

## 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 32

- 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.


Literatur für diesen Teil (1)



**Vielen Dank.**

-  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)  
Abschnitt 3.4.2 (Einschließung bei Termersetzungssystemen)  
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 (2)

-  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.