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`

2019-02-01

Teil 2: endliche Bäume

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

8:30

20. 11. 18 **S. 215** (F. 85)

Überblick

Überblick

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

8:30

Und nun ...

Und nun ...
1. *Motivation: semistrukturierte Daten*
2. Grundbegriffe
3. Charakterisierungen erkennbarer Baumsprachen
4. Top-down-Baumautomaten
5. Abschlusseigenschaften
6. Entscheidungsprobleme
7. *Anwendung: XML-Schemasprachen*

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

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

8:31

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

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

8:32

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..."}
```


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..."}

8:32

Datenstruktur: Baum

?

2019-02-01

Teil 2: endliche Bäume
└*Motivation: semistrukturierte Daten*
└Datenstruktur: Baum

Datenstruktur: Baum

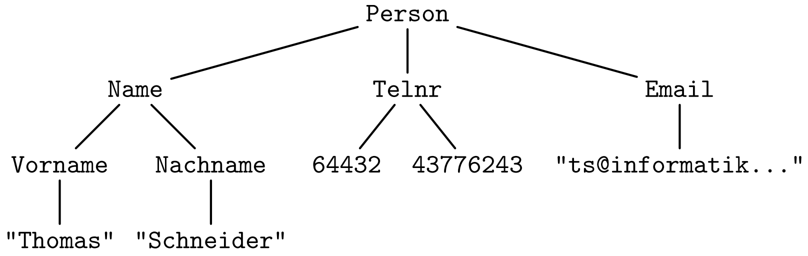
?

8:33

Datenstruktur: Baum

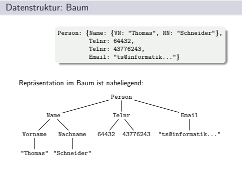
```
Person: {Name: {VN: "Thomas", NN: "Schneider"},
         Telnr: 64432,
         Telnr: 43776243,
         Email: "ts@informatik..."}
```

Repräsentation im Baum ist naheliegend:



2019-02-01

Teil 2: endliche Bäume
└ *Motivation: semistrukturierte Daten*
  └ Datenstruktur: Baum

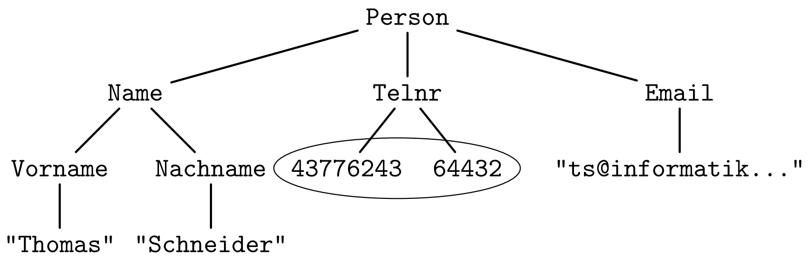


8:33

Datenstruktur: Baum

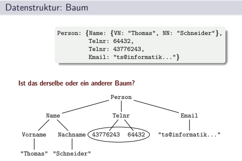
```
Person: {Name: {VN: "Thomas", NN: "Schneider"},
         Telnr: 64432,
         Telnr: 43776243,
         Email: "ts@informatik..."}
```

Ist das derselbe oder ein anderer Baum?



2019-02-01

Teil 2: endliche Bäume
└─*Motivation: semistrukturierte Daten*
└─Datenstruktur: Baum



8:34

Automaten auf endlichen Bäumen

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

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

Automaten auf endlichen Bäumen
... sind wichtig für semistrukturierte Daten, weil sie ...
- XML-Schemasprachen und -validierung zugrunde liegen
- XML-Anfragesprachen auf ihnen aufgebaut sind

8:35

XML-Schemasprachen und -validierung

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

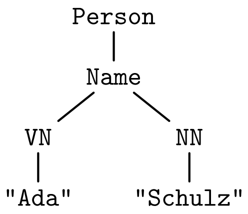
XML-Schemasprachen und -validierung
- Schemasprachen zur Beschreibung gültiger Bäume
- Validierung = Erkennen gültiger und ungültiger Bäume
- gängige Schemasprachen benutzen dafür Baumautomaten

8:35

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

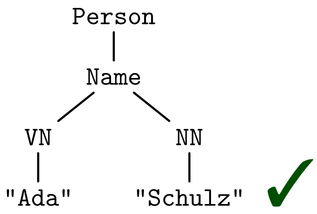
Beispiel: jeder Name muss einen Vor- und Nachnamen haben

8:35

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



2019-02-01

Teil 2: endliche Bäume
└ *Motivation: semistrukturierte Daten*
└ XML-Schemasprachen und -validierung

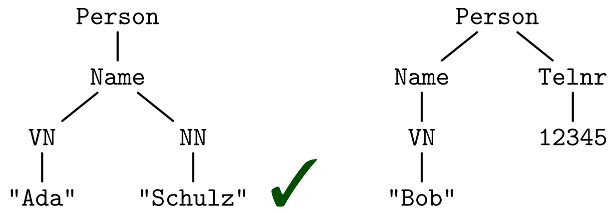
XML-Schemasprachen und -validierung
- Schemasprachen zur Beschreibung gültiger Bäume
- Validierung = Erkennen gültiger und ungültiger Bäume
- gängige Schemasprachen benutzen dafür Baumautomaten
Beispiel: jeder Name muss einen Vor- und Nachnamen haben

8:35

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



2019-02-01

Teil 2: endliche Bäume
└*Motivation: semistrukturierte Daten*
└XML-Schemasprachen und -validierung

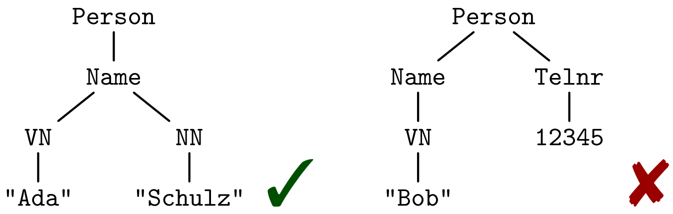
XML-Schemasprachen und -validierung
- Schemasprachen zur Beschreibung gültiger Bäume
- Validierung = Erkennen gültiger und ungültiger Bäume
- gängige Schemasprachen benutzen dafür Baumautomaten
Beispiel: jeder Name muss einen Vor- und Nachnamen haben

8:35

XML-Schemasprachen und -validierung

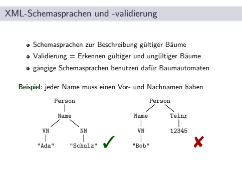
- Schemasprachen zur Beschreibung gültiger Bäume
- Validierung = Erkennen gültiger und ungültiger Bäume
- gängige Schemasprachen benutzen dafür Baumautomaten

Beispiel: jeder Name muss einen Vor- und Nachnamen haben



2019-02-01

Teil 2: endliche Bäume
└*Motivation: semistrukturierte Daten*
└XML-Schemasprachen und -validierung



8:35

XML-Anfragesprachen

- beantworten Anfragen mit Daten aus gegebenen Bäumen

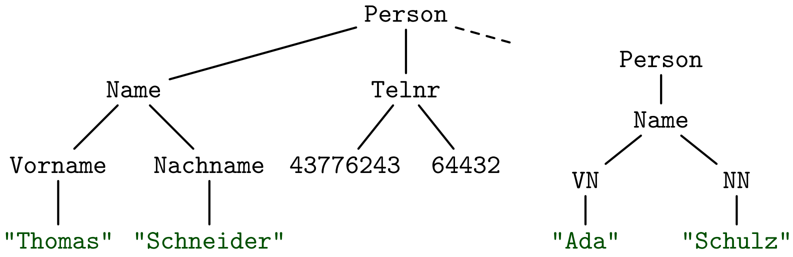
8:36

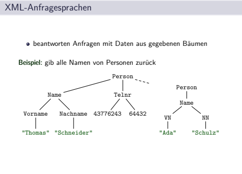
XML-Anfragesprachen
- beantworten Anfragen mit Daten aus gegebenen Bäumen

XML-Anfragesprachen

- beantworten Anfragen mit Daten aus gegebenen Bäumen

Beispiel: gib alle Namen von Personen zurück





8:36

Und nun . . .

Und nun . . .

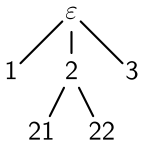
1. *Motivation: semistrukturierte Daten*
2. Grundbegriffe
3. Charakterisierungen erkennbarer Baumsprachen
4. Top-down-Baumautomaten
5. Abschlusseigenschaften
6. Entscheidungsprobleme
7. *Anwendung: XML-Schemasprachen*

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:



8:37

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, $i = 1, \dots, n$

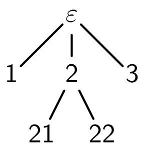
Alphabet mit Stelligkeit
- hier: **r-Alphabet** Σ (auf Englisch: *ranked alphabet*)
- nichtleere endliche Menge von Symbolen; jedem Symbol ist eine Stelligkeit $\in \mathbb{N}$ zugeordnet
- Σ_m = Menge der Symbole mit Stelligkeit m
- Schreibweise: $\Sigma = \{a_1/r_1, \dots, a_n/r_n\}$ heißt: Σ enthält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

8:38

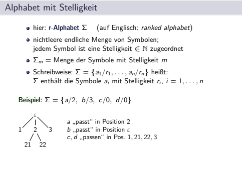
Alphabet mit Stelligkeit

- 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. $1, 21, 22, 3$

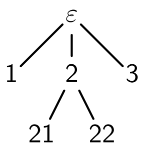


8:38

Alphabet mit Stelligkeit

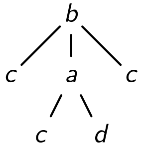
- hier: **r-Alphabet** Σ (auf Englisch: *ranked alphabet*)
- nichtleere endliche Menge von Symbolen;
  jedem Symbol ist eine Stelligkeit $\in \mathbb{N}$ zugeordnet
- Σ_m = Menge der Symbole mit Stelligkeit m
- Schreibweise: $\Sigma = \{a_1/r_1, \dots, a_n/r_n\}$ heißt:
  Σ enthält die Symbole a_i mit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

2019-02-01

Teil 2: endliche Bäume
└Grundbegriffe
└Alphabet mit Stelligkeit

8:38

Was ist nun ein Baum?

?

2019-02-01

Teil 2: endliche Bäume
└Grundbegriffe
└Was ist nun ein Baum?

Was ist nun ein Baum?

?

8:42

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) $t : P \longrightarrow \Sigma$ 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 $\{j \mid pj \in P\} = \{1, \ldots, m\}$.

8:42

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 $\{j \mid pj \in P\} = \{1, \ldots, m\}$.

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 $\{j \mid pj \in P\} = \{1, \ldots, m\}$.

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.

8:42

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 $\{j \mid pj \in P\} = \{1, \ldots, m\}$.

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

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

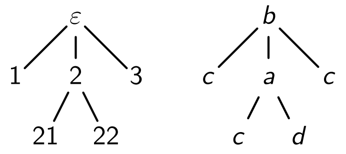
8:45

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

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

8:46

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

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.

8:50

Keine Anfangszustände – wir sehen gleich, warum.

Bottom-up-Baumautomaten: Intuitionen

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

- 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. **T 2.1**

Bottom-up-Baumautomaten: Intuitionen
Bedeutung der Überführungsregeln $a(q_1, \dots, q_m) \to q$:
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. T 2.1

8:52 bis 8:56

Fragen: Was sind dann die Anfangszustände?

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

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

8:52 bis 8:56

Fragen: Was sind dann die Anfangszustände?

Bottom-up-Baumautomaten: Intuitionen

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

- 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. **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?

Bottom-up-Baumautomaten: Intuitionen
Bedeutung der Überführungsregeln $a(q_1, \dots, q_m) \to q$:
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. 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?

8:52 bis 8:56

Fragen: Was sind dann die Anfangszustände?

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: $a \to q$

8:52 bis 8:56

Fragen: Was sind dann die Anfangszustände?

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 $a \to q \in \Delta$.

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 $a \to q \in \Delta$.

8:56 $\to$ 5 min Puffer $\to$ 9:01

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 $r(p1) = q_1, \quad \dots, \quad r(pm) = q_m,$

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 $r(p1) = q_1, \dots, r(pm) = q_m,$

8:56 → 5 min Puffer → 9:01

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 $b(q_1, \dots, q_m) \to q \in \Delta$.

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 $r(p1) = q_1, \dots, r(pm) = q_m$, dann $b(q_1, \dots, q_m) \to q \in \Delta$.

8:56 → 5 min Puffer → 9:01

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 $a \to q \in \Delta$.

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 $r(p1) = q_1, \dots, r(pm) = q_m$, dann $b(q_1, \dots, q_m) \to q \in \Delta$.
Also gilt:
Blatt mit a kann q nur zugewiesen kriegen, wenn $a \to q \in \Delta$.

8:56 $\to$ 5 min Puffer $\to$ 9:01

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 $b(q_1, \dots, q_m) \to q \in \Delta$.

8:56 $\to$ 5 min Puffer $\to$ 9:01

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 $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$.

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 $r(p1) = q_1, \dots, r(pm) = q_m$, 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 $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$.

9:04

Beispiel 1

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

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

Beispiel 1
Sei $\Sigma = \{a/2, b/0, c/0\}$ und $\mathcal{A} = (\{q_0, \dots, q_3\}, \Sigma, \Delta, \{q_2\})$
mit $\Delta = \{ b \to q_0, c \to q_0, a(q_0, q_0) \to q_1, a(q_1, q_1) \to q_2, a(q_1, q_1) \to q_3 \}$.

9:05

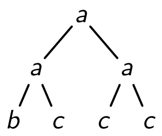
Fragen: Welche Runs gibt es? Was ist die erkannte Sprache?

Beispiel 1

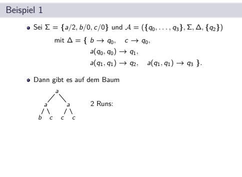
- Sei $\Sigma = \{a/2, b/0, c/0\}$ und $\mathcal{A} = (\{q_0, \dots, q_3\}, \Sigma, \Delta, \{q_2\})$

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

- Dann gibt es auf dem Baum



2 Runs:



9:05

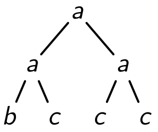
Fragen: Welche Runs gibt es? Was ist die erkannte Sprache?

Beispiel 1

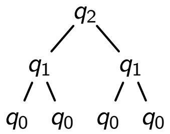
- Sei $\Sigma = \{a/2, b/0, c/0\}$ und $\mathcal{A} = (\{q_0, \ldots, q_3\}, \Sigma, \Delta, \{q_2\})$

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

- Dann gibt es auf dem Baum

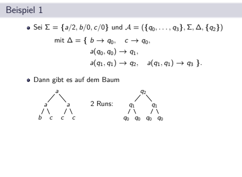


2 Runs:



2019-02-01

Teil 2: endliche Bäume
└Grundbegriffe
└Beispiel 1



9:05

Fragen: Welche Runs gibt es? Was ist die erkannte Sprache?

Beispiel 1

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

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

- Dann gibt es auf dem Baum

$$\begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & a & & & & a & \\ / & & \backslash & & / & & \backslash \\ b & & c & & c & & c \end{array}$$

2 Runs:

$$\begin{array}{ccccccc} & & & q_3 & & & \\ & & / & & \backslash & & \\ & q_1 & & & & q_1 & \\ / & & \backslash & & / & & \backslash \\ q_0 & & q_0 & & q_0 & & q_0 \end{array}$$

9:05

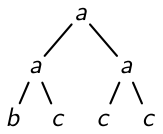
Fragen: Welche Runs gibt es? Was ist die erkannte Sprache?

Beispiel 1

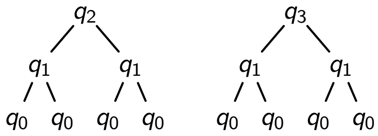
- Sei $\Sigma = \{a/2, b/0, c/0\}$ und $\mathcal{A} = (\{q_0, \ldots, q_3\}, \Sigma, \Delta, \{q_2\})$

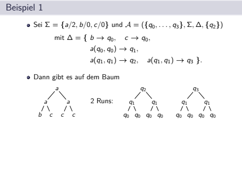
$$\text{mit } \Delta = \{\ b \to q_0, \quad c \to q_0,$$
$$a(q_0, q_0) \to q_1,$$
$$a(q_1, q_1) \to q_2, \quad a(q_1, q_1) \to q_3\ \}.$$

- Dann gibt es auf dem Baum



2 Runs:





9:05

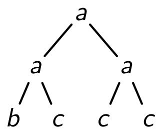
Fragen: Welche Runs gibt es? Was ist die erkannte Sprache?

Beispiel 1

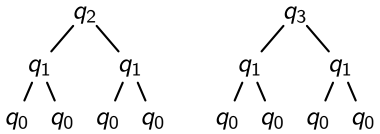
- Sei $\Sigma = \{a/2, b/0, c/0\}$ und $\mathcal{A} = (\{q_0, \dots, q_3\}, \Sigma, \Delta, \{q_2\})$

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

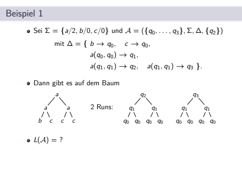
- Dann gibt es auf dem Baum



2 Runs:



- $L(\mathcal{A}) = ?$



9:05

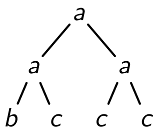
Fragen: Welche Runs gibt es? Was ist die erkannte Sprache?

Beispiel 1

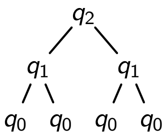
- Sei $\Sigma = \{a/2, b/0, c/0\}$ und $\mathcal{A} = (\{q_0, \dots, q_3\}, \Sigma, \Delta, \{q_2\})$

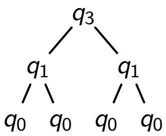
$$\text{mit } \Delta = \{\ b \to q_0, \quad c \to q_0, \quad a(q_0, q_0) \to q_1, \quad a(q_1, q_1) \to q_2, \quad a(q_1, q_1) \to q_3\ \}.$$

- Dann gibt es auf dem Baum

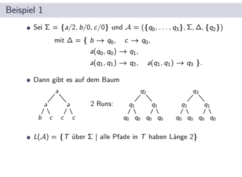


2 Runs:





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



9:05

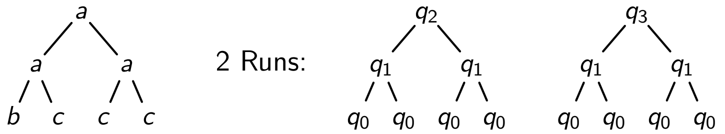
Fragen: Welche Runs gibt es? Was ist die erkannte Sprache?

Beispiel 1

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

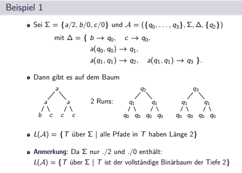
  mit $\Delta = \{\ b \to q_0, \quad c \to q_0,$
  $a(q_0, q_0) \to q_1,$
  $a(q_1, q_1) \to q_2, \quad a(q_1, q_1) \to q_3\ \}.$

- 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 Σ nur ./2 und ./0 enthält:
  $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid T \text{ ist der vollständige Binärbaum der Tiefe } 2\}$



9:05

Fragen: Welche Runs gibt es? Was ist die erkannte Sprache?

Beispiel 2

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

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

9:08 bis 9:15

Gemeinsam machen, Zeit nehmen!

Beispiel 2

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

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

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

Beispiel 2

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

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

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

9:08 bis 9:15

Gemeinsam machen, Zeit nehmen!

Beispiel 2

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

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

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

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

Beispiel 2
Sei $\Sigma = \{a/2,\ b/1,\ c/0,\ d/0\}$. Welcher NEBA erkennt $\{T$ über $\Sigma \mid$ jedes c-Blatt hat ein rechtes d-Geschwister$\}$?
$\mathcal{A} = (\{q_c, q_d, q_f\}, \Sigma, \Delta, \{q_f\})$ mit
$\Delta = \{ c \to q_c, d \to q_d, d \to q_f, a(q_c, q_d) \to q_f, a(q_f, q_f) \to q_f, b(q_f) \to q_f \}$
Übergang $a(q_d, q_d) \to q_f$ ist überflüssig: $d \to q_f$ und $a(q_f, q_f) \to q_f$.

9:08 bis 9:15

Gemeinsam machen, Zeit nehmen!

Beispiel 2

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

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

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

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**

Beispiel 2
Sei $\Sigma = \{a/2,\ b/1,\ c/0,\ d/0\}$. Welcher NEBA erkennt $\{T$ über $\Sigma \mid$ jedes c-Blatt hat ein rechtes d-Geschwister$\}$?
$\mathcal{A} = (\{q_c, q_d, q_f\}, \Sigma, \Delta, \{q_f\})$ mit
$\Delta = \{ c \to q_c,\ d \to q_d,\ d \to q_f,\ a(q_c, q_d) \to q_f,\ a(q_f, q_f) \to q_f,\ b(q_f) \to q_f \}$
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

9:08 bis 9:15

Gemeinsam machen, Zeit nehmen!

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 $L(\mathcal{A}) = L$.

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 $L(\mathcal{A}) = L$.

9:15, 5 min Pause bis 9:20

Determinismus

Definition 2.5

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

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

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

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

Determinismus

Definition 2.5
Sei $\mathcal{A} = (Q, \Sigma, \Delta, F)$ ein NEBA.
Enthält Δ für jedes jedes $a \in \Sigma_m$ und alle $(q_1, \ldots, q_m) \in Q^m$ höchstens eine[1] Regel $a(q_1, \ldots, q_m) \to q$
dann ist $\mathcal{A}$ ein deterministischer endlicher Baumautomat (DEBA).

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

9:20

Determinismus, „höchstens eine Regel …":
Papierkorbzustand q^- ist hier umständlicher als bei NEAs:

- mehr Kombis $a(q_1, \ldots, q_m) \to q^-$
- man brauchte bei m-stelligen Zeichen a lange Regel $a(q^-, \ldots, q^-) \to q^-$

Beispiele: **beide** Aut. sind NEBAs.

Frage stellen!

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).

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

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

9:20

Determinismus, „höchstens eine Regel …“:
Papierkorbzustand q^- ist hier umständlicher als bei NEAs:
- mehr Kombis $a(q_1, \dots, q_m) \to q^-$
- man brauchte bei m-stelligen Zeichen a lange Regel $a(q^-, \dots, q^-) \to q^-$

Beispiele: **beide** Aut. sind NEBAs.

Frage stellen!

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)**.

$\rightsquigarrow$ 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

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).
$\rightsquigarrow$ 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

9:20

Determinismus, „höchstens eine Regel …“:
Papierkorbzustand q^- ist hier umständlicher als bei NEAs:
- mehr Kombis $a(q_1, \dots, q_m) \to q^-$
- man brauchte bei m-stelligen Zeichen a lange Regel $a(q^-, \dots, q^-) \to q^-$

Beispiele: **beide** Aut. sind NEBAs.

Frage stellen!

Potenzmengenkonstruktion

Antwort: Ja!

Satz 2.6

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

Potenzmengenkonstruktion
Antwort: Ja!
Satz 2.6
Für jeden NEBA $\mathcal{A}$ gibt es einen DEBA $\mathcal{A}^d$ mit $L(\mathcal{A}^d) = L(\mathcal{A})$.

9:23 bis 9:50? $\to$ 10 min Reserve

“ist DEBA”: sogar genau 1 Folgezustand statt “höchstens 1”

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.**
  $S = \{q \mid \exists q_1 \in S_1, \dots, \exists q_m \in S_m : a(q_1, \dots, q_m) \to q \in \Delta\}$

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. $S = \{q \mid \exists q_1 \in S_1, \dots, \exists q_m \in S_m : a(q_1, \dots, q_m) \to q \in \Delta\}$

9:23 bis 9:50? $\longrightarrow$ 10 min Reserve

“ist DEBA”: sogar genau 1 Folgezustand statt “höchstens 1”

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** □

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

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

9:23 bis 9:50? $\to$ 10 min Reserve

"ist DEBA": sogar genau 1 Folgezustand statt "höchstens 1"

Und nun ...

2019-02-01

Teil 2: endliche Bäume
└─Charakterisierungen erkennbarer Baumsprachen
└─Und nun ...

Und nun ...
1. Motivation: semistrukturierte Daten
2. Grundbegriffe
3. Charakterisierungen erkennbarer Baumsprachen
4. Top-down-Baumautomaten
5. Abschlusseigenschaften
6. Entscheidungsprobleme
7. Anwendung: XML-Schemasprachen

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

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

$\leadsto\ L(\mathcal{A}) =$

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
$\Delta = \{\ c \to q_0,\ b(q_0) \to q_1,\ a(q_0, q_0) \to q_1,\ b(q_1) \to q_0,\ a(q_1, q_1) \to q_0\ \}.$
$\leadsto\ L(\mathcal{A}) =$

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

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

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

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
$\Delta = \{\ c \to q_0,\ b(q_0) \to q_1,\ a(q_0, q_0) \to q_1,\ b(q_1) \to q_0,\ a(q_1, q_1) \to q_0\ \}$.
$\rightsquigarrow$ $L(\mathcal{A}) = \{T \mid \text{alle Wurzel-Blatt-Pfade in } T \text{ haben gerade Länge}\}$.

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

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

$\rightsquigarrow$ $L(\mathcal{A}) = \{T \mid \text{alle Wurzel-Blatt-Pfade in } T \text{ haben gerade Länge}\}$.
$\neq \{T \mid T \text{ hat gerade Höhe}\}$ **T 2.4**

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
$\Delta = \{\ c \to q_0,\ b(q_0) \to q_1,\ a(q_0, q_0) \to q_1,\ b(q_1) \to q_0,\ a(q_1, q_1) \to q_0\ \}.$
$\rightsquigarrow$ $L(\mathcal{A}) = \{T \mid \text{alle Wurzel-Blatt-Pfade in } T \text{ haben gerade Länge}\}$.
$\neq \{T \mid T \text{ hat gerade Höhe}\}$ T 2.4

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

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

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

2019-02-01

Teil 2: endliche Bäume
└─Charakterisierungen erkennbarer Baumsprachen
└─Nichterkennbare Baumsprachen: Intuitionen

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
$\Delta = \{\ c \to q_0,\ b(q_0) \to q_1,\ a(q_0, q_0) \to q_1,\ b(q_1) \to q_0,\ a(q_1, q_1) \to q_0\ \}$.
$\rightsquigarrow$ $L(\mathcal{A}) = \{T \mid \text{alle Wurzel-Blatt-Pfade in } T \text{ haben gerade Länge}\}$. $\neq \{T \mid T \text{ hat gerade Höhe}\}$ 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.

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

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

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

2019-02-01

Teil 2: endliche Bäume
└Charakterisierungen erkennbarer Baumsprachen
└Nichterkennbare Baumsprachen: Intuitionen

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
$\Delta = \{\ c \to q_0,\ b(q_0) \to q_1,\ a(q_0,q_0) \to q_1,\ b(q_1) \to q_0,\ a(q_1,q_1) \to q_0\ \}.$
$\rightsquigarrow$ $L(\mathcal{A}) = \{T \mid \text{alle Wurzel-Blatt-Pfade in } T \text{ haben gerade Länge}\}.$ $\neq \{T \mid T \text{ hat gerade Höhe}\}$ 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)

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)

Wir müssen Bäume ineinander “stecken” können.
Dafür brauchen wir eine Markierung für die “Andockstelle” (Variable) und den Begriff des Kontexts (Baum mit Andockstelle).

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

Wir müssen Bäume ineinander “stecken” können.
Dafür brauchen wir eine Markierung für die “Andockstelle” (Variable) und den Begriff des Kontexts (Baum mit Andockstelle).

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 $i \in \mathbb{N}$.

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 $i \in \mathbb{N}$.

8:30

Beweis des Pumping-Lemmas

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

8:32

Beweis des Pumping-Lemmas

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

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 $k = |Q|$.

8:32

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 $k = |Q|$.

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 erfolgreicher Run von $\mathcal{A}$ auf T.

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 erfolgreicher Run von $\mathcal{A}$ auf T.

8:32

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 erfolgreicher 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 $q \in Q$.

8:32

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 erfolgreicher 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 $q \in Q$.

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 erfolgreicher 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
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 erfolgreicher 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

8:32

Beweis des Pumping-Lemmas

Seien nun:

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

Beweis des Pumping-Lemmas
Seien nun:
$U = T_{p_1}$
$C =$ derjenige Kontext mit $C[U] = T$
$V = T_{p_2}$
$D =$ derjenige Kontext mit $U = D[V]$

8:38

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.

Beweis des Pumping-Lemmas

Seien nun:

$U = T_{p_1}$
C = derjenige Kontext mit $C[U] = T$
$V = T_{p_2}$
D = derjenige Kontext mit $U = D[V]$

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

8:38

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.**

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.

8:38

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

Seien nun:

$U = T_{p_1}$
C = derjenige Kontext mit $C[U] = T$
$V = T_{p_2}$
D = derjenige Kontext mit $U = D[V]$

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 $i \geq 0$.

8:38

Beweis des Pumping-Lemmas

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

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

8:42

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.

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:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \textit{ kein } \text{Nachfolger von } p_1 \text{ ist} \\ r(p_2 p') & \text{falls } p = p_1 p' \end{cases} \qquad (*)$$

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:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \textit{ kein } \text{Nachfolger von } p_1 \text{ ist} \\ r(p_2 p') & \text{falls } p = p_1 p' \end{cases} \qquad (*)$$

8:42

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:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \text{ \textit{kein} Nachfolger von } p_1 \text{ ist} \\ r(p_2 p') & \text{falls } p = p_1 p' \end{cases} \qquad (*)$$

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

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \text{ \textit{kein} Nachfolger von } p_1 \text{ ist} \\ r(p_2 p') & \text{falls } p = p_1 p' \end{cases} \qquad (*)$$

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

8:42

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:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \textit{ kein } \text{Nachfolger von } p_1 \text{ ist} \\ r(p_2 p') & \text{falls } p = p_1 p' \end{cases} \qquad (*)$$

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

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

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:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \textit{ kein } \text{Nachfolger von } p_1 \text{ ist} \\ r(p_2 p') & \text{falls } p = p_1 p' \end{cases} \qquad (*)$$

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

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

8:42

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 $T_0 \in L$.

8:42

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

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.

8:47

Fallunterscheidung **nicht vorlesen**, sondern anzeichnen!

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:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \textit{ kein} \text{ Nachfolger von } p_1 \text{ ist} \\ r(p_1 p') & \text{falls } p = p_1 p_3^j p',\ \ p' \textit{ kein} \text{ NF von } p_3 \text{ und} \\ & \qquad\qquad\qquad\qquad p \textit{ kein} \text{ NF von } p_1 p_3^i \\ r(p_2 p') & \text{falls } p = p_1 p_3^i p' \end{cases}$$

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:

8:47

Fallunterscheidung **nicht vorlesen**, sondern anzeichnen!

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:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \text{ \textit{kein} Nachfolger von } p_1 \text{ ist} \\ r(p_1 p') & \text{falls } p = p_1 p_3^j p',\ p' \text{ \textit{kein} NF von } p_3 \text{ und} \\ & \qquad\qquad\qquad p \text{ \textit{kein} NF von } p_1 p_3^i \\ r(p_2 p') & \text{falls } p = p_1 p_3^i p' \end{cases}$$

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

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:
$r_0(p) = \begin{cases} r(p) & \text{falls } p \text{ kein Nachfolger von } p_1 \text{ ist} \\ r(p_1 p') & \text{falls } p = p_1 p_3^j p',\ p' \text{ kein NF von } p_3 \text{ und } p \text{ kein NF von } p_1 p_3^i \\ r(p_2 p') & \text{falls } p = p_1 p_3^i p' \end{cases}$
Wie im 1. Fall: r_i ist erfolgreicher Run von $\mathcal{A}$ auf T_i.

8:47

Fallunterscheidung **nicht vorlesen**, sondern anzeichnen!

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:

$$r_0(p) = \begin{cases} r(p) & \text{falls } p \textit{ kein } \text{Nachfolger von } p_1 \text{ ist} \\ r(p_1 p') & \text{falls } p = p_1 p_3^j p',\ \ p' \textit{ kein } \text{NF von } p_3 \text{ und} \\ & \qquad\qquad\qquad p \textit{ kein } \text{NF von } p_1 p_3^i \\ r(p_2 p') & \text{falls } p = p_1 p_3^i p' \end{cases}$$

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

Also $T_i \in L$. $\square$

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

8:47

Fallunterscheidung **nicht vorlesen**, sondern anzeichnen!

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

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

8:52

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:

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

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:
$C[T_1] \in L$ genau dann, wenn $C[T_2] \in L$

9:14

Satz ohne Beweis. An Tafel nur Bsp.

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

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:
$C[T_1] \in L$ genau dann, wenn $C[T_2] \in L$
Satz 2.9
$L \subseteq \Sigma^*$ is NEBA-erkennbar gdw. $\sim_L$ endlichen Index hat.
T 2.8

9:14

Satz ohne Beweis. An Tafel nur Bsp.

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

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:
$C[T_1] \in L$ genau dann, wenn $C[T_2] \in L$
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

9:14

Satz ohne Beweis. An Tafel nur Bsp.

Und nun ...

Und nun ...
1. Motivation: semistrukturierte Daten
2. Grundbegriffe
3. Charakterisierungen erkennbarer Baumsprachen
4. Top-down-Baumautomaten
5. Abschlusseigenschaften
6. Entscheidungsprobleme
7. Anwendung: XML-Schemasprachen

Drehen wir jetzt alles um? ☺

$\overset{?}{\rightsquigarrow}$

2019-02-01

Teil 2: endliche Bäume
└Top-down-Baumautomaten
└Drehen wir jetzt alles um? ☺

16:00

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

16:00

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.

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

Berechnungen, Akzeptanz und erkannte Sprache
Definition 2.11
Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA und $T = (P, t)$ ein Σ-Baum.
Run von $\mathcal{A}$ auf T ist eine Fkt. $r : P \to Q$ mit:
$r(\varepsilon) \in I$

16:03

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.11

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

- **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 $r(p1) = q_1, \quad \ldots, \quad r(pm) = q_m,$

16:03

Berechnungen, Akzeptanz und erkannte Sprache
Definition 2.11
Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA und $T = (P, t)$ ein Σ-Baum.
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 $r(p1) = q_1, \ldots, r(pm) = q_m,$

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.11

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

- **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 $(a, q) \to (q_1, \dots, q_m) \in \Delta$.

Berechnungen, Akzeptanz und erkannte Sprache
Definition 2.11
Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA und $T = (P, t)$ ein Σ-Baum.
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 $r(p1) = q_1, \dots, r(pm) = q_m$,
dann gibt es eine Regel $(a, q) \to (q_1, \dots, q_m) \in \Delta$.

16:03

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.11

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

- **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 $(a, q) \to () \in \Delta$.

Berechnungen, Akzeptanz und erkannte Sprache
Definition 2.11
Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA und $T = (P, t)$ ein Σ-Baum.
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 $r(p1) = q_1, \dots, r(pm) = q_m$, 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 $(a, q) \to () \in \Delta$.

16:03

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.11

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

- **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**.

Berechnungen, Akzeptanz und erkannte Sprache
Definition 2.11
Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA und $T = (P, t)$ ein Σ-Baum.
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 $r(p1) = q_1, \dots, r(pm) = q_m$, 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.

16:03

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.11

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

- **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 $r(p1) = q_1, \quad \ldots, \quad r(pm) = q_m$,
    dann gibt es eine Regel $(a, q) \to (q_1, \ldots, q_m) \in \Delta$.
  - 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
  $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$.

Berechnungen, Akzeptanz und erkannte Sprache
Definition 2.11
Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA und $T = (P, t)$ ein Σ-Baum.
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 $r(p1) = q_1, \ldots, r(pm) = q_m$, dann gibt es eine Regel $(a, q) \to (q_1, \ldots, q_m) \in \Delta$.
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 $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$.

16:03

Berechnungen, Akzeptanz und erkannte Sprache

Definition 2.11

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

- **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 $r(p1) = q_1, \quad \ldots, \quad r(pm) = q_m$, dann gibt es eine Regel $(a, q) \to (q_1, \ldots, q_m) \in \Delta$.
  - 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".

Berechnungen, Akzeptanz und erkannte Sprache
Definition 2.11
Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA und $T = (P, t)$ ein Σ-Baum.
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 $r(p1) = q_1, \ldots, r(pm) = q_m$, dann gibt es eine Regel $(a, q) \to (q_1, \ldots, q_m) \in \Delta$.
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".

16:03

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

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

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

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

16:07

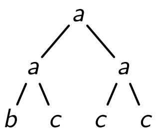
Beispiel 1

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

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

$$\begin{aligned}\Delta = \{\ & (a, q_0) \to (q_1, q_1), \quad (b, q_2) \to (), \\ & (a, q_1) \to (q_2, q_2), \quad (c, q_2) \to (), \\ & (a, q_0) \to (q_x, q_x) \qquad\qquad\qquad \}\end{aligned}$$

- Dann gibt es auf dem Baum



nur 1 Run:

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
Dann gibt es auf dem Baum
nur 1 Run:

16:07

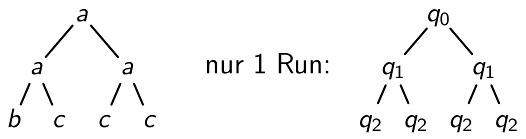
Beispiel 1

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

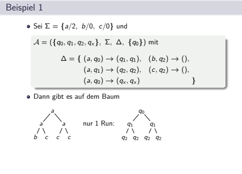
$\mathcal{A} = (\{q_0, q_1, q_2, q_x\},\ \Sigma,\ \Delta,\ \{q_0\})$ mit

$$\begin{aligned}\Delta = \{\ & (a, q_0) \to (q_1, q_1), \quad (b, q_2) \to (), \\ & (a, q_1) \to (q_2, q_2), \quad (c, q_2) \to (), \\ & (a, q_0) \to (q_x, q_x) \qquad\qquad\qquad \}\end{aligned}$$

- Dann gibt es auf dem Baum



2019-02-01



16:07

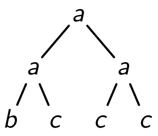
Beispiel 1

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

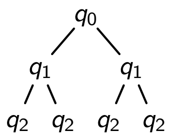
$\mathcal{A} = (\{q_0, q_1, q_2, q_x\},\ \Sigma,\ \Delta,\ \{q_0\})$ mit

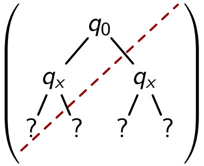
$$\begin{aligned}\Delta = \{\ & (a, q_0) \to (q_1, q_1), \quad && (b, q_2) \to (),\\ & (a, q_1) \to (q_2, q_2), && (c, q_2) \to (),\\ & (a, q_0) \to (q_x, q_x) && \qquad\qquad \}\end{aligned}$$

- Dann gibt es auf dem Baum



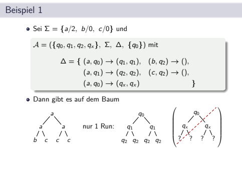
nur 1 Run:





2019-02-01

Teil 2: endliche Bäume
└Top-down-Baumautomaten
└Beispiel 1



16:07

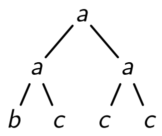
Beispiel 1

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

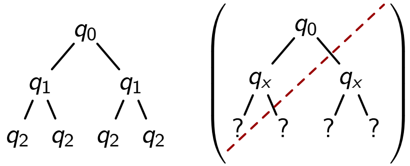
$$\mathcal{A} = (\{q_0, q_1, q_2, q_x\},\ \Sigma,\ \Delta,\ \{q_0\}) \text{ mit}$$

$$\begin{aligned}\Delta = \{\ & (a, q_0) \to (q_1, q_1), && (b, q_2) \to (), \\ & (a, q_1) \to (q_2, q_2), && (c, q_2) \to (), \\ & (a, q_0) \to (q_x, q_x) && \}\end{aligned}$$

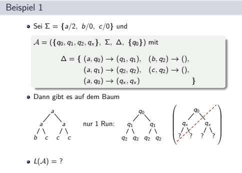
- Dann gibt es auf dem Baum



nur 1 Run:



- $L(\mathcal{A}) = ?$



16:07

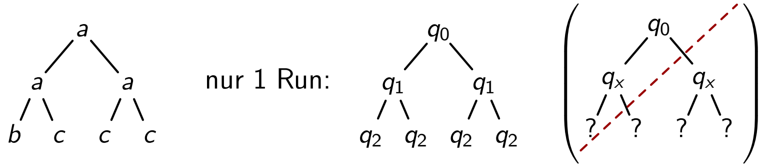
Beispiel 1

- 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



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

2019-02-01

Teil 2: endliche Bäume
└Top-down-Baumautomaten
└Beispiel 1

16:07

Beispiel 2

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

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

16:11

Beispiel 2

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

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

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

Beispiel 2
Sei $\Sigma = \{a/2,\ b/1,\ c/0,\ d/0\}$. Welcher NETDBA erkennt
$L_{\text{cd}} = \{T \text{ über } \Sigma \mid$ jedes c-Blatt hat ein rechtes d-Geschwister$\}$?
NETDBA $\mathcal{A} = (\{q_0, q_c, q_d\}, \Sigma, \Delta, \{q_0\})$ mit
$\Delta = \{ (a, q_0) \to (q_0, q_0), b(q_0) \to q_0, (c, q_c) \to (), (a, q_0) \to (q_c, q_d), (d, q_d) \to (), (d, q_0) \to () \}$

16:11

Beispiel 2

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

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

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

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

$$\begin{aligned}\Delta = \{\ & a(q_f, q_f) \to q_f, \quad b(q_f) \to q_f, \quad c \to q_c,\\ & a(q_c, q_d) \to q_f, \qquad\qquad\qquad d \to q_d,\\ & \qquad\qquad\qquad\qquad\qquad\qquad d \to q_f\ \}\end{aligned}$$

Beispiel 2
Sei $\Sigma = \{a/2,\ b/1,\ c/0,\ d/0\}$. Welcher NETDBA erkennt
$L_{\text{cd}} = \{T \text{ über } \Sigma \mid \text{ jedes } c\text{-Blatt hat ein rechtes } d\text{-Geschwister}\}$?
NETDBA $\mathcal{A} = (\{q_0, q_c, q_d\}, \Sigma, \Delta, \{q_0\})$ mit
Vergleiche mit dem NEBA $\mathcal{A} = (\{q_c, q_d, q_f\}, \Sigma, \Delta, \{q_f\})$ mit

16:11

Beispiel 2

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

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

$$\begin{array}{llll} \Delta = \{ & (a, q_0) \to (q_0, q_0), & b(q_0) \to q_0, & (c, q_c) \to (), \\ & (a, q_0) \to (q_c, q_d), & & (d, q_d) \to (), \\ & & & (d, q_0) \to () \ \} \end{array}$$

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

$$\begin{array}{llll} \Delta = \{ & a(q_f, q_f) \to q_f, & b(q_f) \to q_f, & c \to q_c, \\ & a(q_c, q_d) \to q_f, & & d \to q_d, \\ & & & d \to q_f \ \ \} \end{array}$$

Was sagt uns das über das Verhältnis NETDBAs : NEBAs?

Beispiel 2
Sei $\Sigma = \{a/2,\ b/1,\ c/0,\ d/0\}$. Welcher NETDBA erkennt $L_{\text{cd}} = \{T$ über $\Sigma \mid$ jedes c-Blatt hat ein rechtes d-Geschwister$\}$?
NETDBA $\mathcal{A} = (\{q_0, q_c, q_d\}, \Sigma, \Delta, \{q_0\})$ mit
$\Delta = \{ (a, q_0) \to (q_0, q_0),\ b(q_0) \to q_0,\ (c, q_c) \to (),\ (a, q_0) \to (q_c, q_d),\ (d, q_d) \to (),\ (d, q_0) \to () \}$
Vergleiche mit dem NEBA $\mathcal{A} = (\{q_c, q_d, q_f\}, \Sigma, \Delta, \{q_f\})$ mit
$\Delta = \{ a(q_f, q_f) \to q_f,\ b(q_f) \to q_f,\ c \to q_c,\ a(q_c, q_d) \to q_f,\ d \to q_d,\ d \to q_f \}$
Was sagt uns das über das Verhältnis NETDBAs : NEBAs?

16:11

NETDBAs vs. NEBAs

NETDBAs und NEBAs sind gleichmächtig!

Satz 2.12

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

NETDBAs vs. NEBAs
NETDBAs und NEBAs sind gleichmächtig!
Satz 2.12
$\{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NETDBA}\} = \{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NEBA}\}.$

16:15 → bis 16:20

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 NETDBA.
Konstruieren NEBA $\mathcal{A}^\uparrow = (Q, \Sigma, \Delta^\uparrow, F^\uparrow)$ mit:

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

NETDBAs vs. NEBAs
NETDBAs und NEBAs sind gleichmächtig!
Satz 2.12
$\{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NETDBA}\} = \{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NEBA}\}.$
Beweis. Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA.
Konstruieren NEBA $\mathcal{A}^\uparrow = (Q, \Sigma, \Delta^\uparrow, F^\uparrow)$ mit:
$\Delta^\uparrow = \{a(q_1, \dots, q_m) \to q) \mid (a, q) \to (q_1, \dots, q_m) \in \Delta\}$
$F^\uparrow = I$

16:15 → bis 16:20

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 NETDBA.
Konstruieren NEBA $\mathcal{A}^\uparrow = (Q, \Sigma, \Delta^\uparrow, F^\uparrow)$ mit:

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

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

NETDBAs vs. NEBAs
NETDBAs und NEBAs sind gleichmächtig!
Satz 2.12
{L(A) | A ist ein NETDBA} = {L(A) | A ist ein NEBA}.
Beweis. Sei A = (Q, Σ, Δ, I) ein NETDBA.
Konstruieren NEBA A↑ = (Q, Σ, Δ↑, F↑) mit:
Δ↑ = {a(q1, ..., qm) → q) | (a, q) → (q1, ..., qm) ∈ Δ}
F↑ = I
Dann ist jeder Run von A auf einem Σ-Baum T
auch ein erfolgreicher Run von A↑ auf T
und umgekehrt.

16:15 $\to$ bis 16:20

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 NETDBA.
Konstruieren NEBA $\mathcal{A}^\uparrow = (Q, \Sigma, \Delta^\uparrow, F^\uparrow)$ mit:

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

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

NETDBAs vs. NEBAs
NETDBAs und NEBAs sind gleichmächtig!
Satz 2.12
$\{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NETDBA}\} = \{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NEBA}\}.$
Beweis. Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA.
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 $L(\mathcal{A}^\uparrow) = L(\mathcal{A})$.

16:15 $\to$ bis 16:20

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 NETDBA.
Konstruieren NEBA $\mathcal{A}^\uparrow = (Q, \Sigma, \Delta^\uparrow, F^\uparrow)$ mit:

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

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. □

NETDBAs vs. NEBAs
NETDBAs und NEBAs sind gleichmächtig!
Satz 2.12
$\{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NETDBA}\} = \{L(\mathcal{A}) \mid \mathcal{A} \text{ ist ein NEBA}\}.$
Beweis. Sei $\mathcal{A} = (Q, \Sigma, \Delta, I)$ ein NETDBA.
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. □

16:15 $\to$ bis 16:20

Determinisierung von NETDBAs

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

Determinisierung von NETDBAs

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

16:20

Schauen wir uns auch hier noch die deterministische Variante von Top-down-Baumautomaten an.

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

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

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

16:20

Schauen wir uns auch hier noch die deterministische Variante von Top-down-Baumautomaten an.

Determinisierung von NETDBAs

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

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

▲ Nichtdeterminismus!

Determinisierung von NETDBAs
Erinnerung an Beispiel 2: Sei Σ = {a/2, b/1, c/0, d/0} und L_cd = {T über Σ | jedes c-Blatt hat ein rechtes d-Geschwister}.
NETDBA A = ({q0, qc, qd}, Σ, Δ, {q0}) mit
Δ = { (a, q0) → (q0, q0), b(q0) → q0, (c, qc) → (), (a, q0) → (qc, qd), (d, qd) → (), (d, q0) → () }
▲ Nichtdeterminismus!

16:20

Schauen wir uns auch hier noch die deterministische Variante von Top-down-Baumautomaten an.

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

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

▲ Nichtdeterminismus!

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

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

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

▲ Nichtdeterminismus!

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

16:20

Schauen wir uns auch hier noch die deterministische Variante von Top-down-Baumautomaten an.

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?

Determinisierung von NETDBAs?
Betrachte
• Σ = {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?

16:23 bis 16:30

Konsequenz: DETDBAs sind schwächer als DEBAs (und NE(TD)BAs); NEBAs bleiben das Standardmodell!

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**

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

16:23 bis 16:30

Konsequenz: DETDBAs sind schwächer als DEBAs (und NE(TD)BAs); NEBAs bleiben das Standardmodell!

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.

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.

16:23 bis 16:30

Konsequenz: DETDBAs sind schwächer als DEBAs (und NE(TD)BAs);
NEBAs bleiben das Standardmodell!

Und nun ...

2019-02-01

Teil 2: endliche Bäume
└Abschlusseigenschaften
└Und nun ...

Und nun ...

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

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 $L_1 \cap L_2$.

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 $L_1 \cap L_2$.

16:30

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?

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?

16:30

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?

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?

16:30

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?

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?

16:30

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? ✓

16:30

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.

16:33

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

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

16:34 bis 16:38

Letzte Zeile:
Zeige: jeder Run von $\mathcal{A}$ auf T entspricht einem Run von $\mathcal{A}_1$ oder $\mathcal{A}_2$, und umgekehrt.

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.

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.

16:34 bis 16:38

Letzte Zeile:
Zeige: jeder Run von $\mathcal{A}$ auf T entspricht einem Run von $\mathcal{A}_1$ oder $\mathcal{A}_2$, und umgekehrt.

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.

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

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

16:34 bis 16:38

Letzte Zeile:
Zeige: jeder Run von $\mathcal{A}$ auf T entspricht einem Run von $\mathcal{A}_1$ oder $\mathcal{A}_2$, und umgekehrt.

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

16:34 bis 16:38

Letzte Zeile:
Zeige: jeder Run von $\mathcal{A}$ auf T entspricht einem Run von $\mathcal{A}_1$ oder $\mathcal{A}_2$, und umgekehrt.

Abgeschlossenheit unter Komplement

Lemma 2.17

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

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

16:38 bis 16:40

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

16:38 bis 16:40

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

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 DEBA $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, F^d)$ mit $L(\mathcal{A}^d) = L(\mathcal{A})$ und **genau einem** Run pro Eingabebaum.

16:38 bis 16:40

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 DEBA $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, F^d)$ mit $L(\mathcal{A}^d) = L(\mathcal{A})$ und genau einem Run pro Eingabebaum.

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 DEBA $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, F^d)$ 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 $\overline{L(\mathcal{A})}$. $\square$

16:38 bis 16:40

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 DEBA $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, F^d)$ 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 $\overline{L(\mathcal{A})}$. $\square$

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 $\overline{}$:

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

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

16:40

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 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")

16:40

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.

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

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

Abgeschlossenheit unter Verkettungsoperationen
Randbemerkung:
Man kann Analoga zu · 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.
$L_1L_2 = \{C[T] \mid T \in L_1,\ C \in \text{Con}(L_2)\}$
$L^* = \{C_1[C_2[\dots[C_n[T]]\dots]] \mid T \in L,\ C_1,\dots,C_n \in \text{Con}(L),\ n \geqslant 0\}$
Abgeschlossenheit unter $\cdot, *$ kann man dann wie für NEAs zeigen, aber mit mehr technischem Aufwand (Eliminierung ε-Kanten ...)

16:41 bis 16:44; 5min Pause, 1min Puffer $\longrightarrow$ 16:50

Beachte: Bäume sind nach Def. nichtleer.
Deshalb schließt die Def. von L^* den leeren Baum auch aus.

Und nun . . .

Und nun . . .
1. Motivation: semistrukturierte Daten
2. Grundbegriffe
3. Charakterisierungen erkennbarer Baumsprachen
4. Top-down-Baumautomaten
5. Abschlusseigenschaften
6. Entscheidungsprobleme
7. Anwendung: XML-Schemasprachen

Das Leerheitsproblem

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

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

d. h. $\mathbf{LP}_{\mathsf{NEBA}} = \{\mathcal{A} \mid \mathcal{A} \text{ NEBA}, L(\mathcal{A}) = \emptyset\}$ (analog für DEBAs)

2019-02-01

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

16:50

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.

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.

16:50

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

$\mathsf{LP}_{\mathsf{NEBA}}$ und $\mathsf{LP}_{\mathsf{DEBA}}$ sind entscheidbar und **P**-vollständig.

Beweis.

- Entscheidbarkeit in Polyzeit analog zu NEAs:
  prüfe, ob ein akz. Zustand erreichbar ist (nächste Folie)

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
$\mathsf{LP}_{\mathsf{NEBA}}$ und $\mathsf{LP}_{\mathsf{DEBA}}$ sind entscheidbar und P-vollständig.

Beweis.
- Entscheidbarkeit in Polyzeit analog zu NEAs:
  prüfe, ob ein akz. Zustand erreichbar ist (nächste Folie)

16:50

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“
  (≈ Erreichbarkeit in Hypergraphen mit ternärer Kantenrelation),
  siehe [Comon et al. 2008, Exercise 1.19]

Das Leerheitsproblem
Eingabe: NEBA (oder DEBA) $\mathcal{A}$
Frage: Ist $L(\mathcal{A}) = \emptyset$?
d. h. $\mathrm{LP}_{\mathrm{NEBA}} = \{\mathcal{A} \mid \mathcal{A} \text{ NEBA}, L(\mathcal{A}) = \emptyset\}$ (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“ (≈ Erreichbarkeit in Hypergraphen mit ternärer Kantenrelation), siehe [Comon et al. 2008, Exercise 1.19]

16:50

Das Leerheitsproblem

Polynomialzeitalgorithmus:

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

Das Leerheitsproblem
Polynomialzeitalgorithmus:
- Berechne Menge der erreichbaren Zustände
- Prüfe, ob ein akzeptierender Zustand darin ist.

16:53 bis 17:18

„in Polyzeit:“ weil R monoton wachsend und durch Q beschränkt!

Letzte Äquivalenz: erkannte Sprache leer gdw. kein akz. Zust. erreichbar

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:

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

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

16:53 bis 17:18

„in Polyzeit:“ weil R monoton wachsend und durch Q beschränkt!

Letzte Äquivalenz: erkannte Sprache leer gdw. kein akz. Zust. erreichbar

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:

- $R := \{q \mid a \to q \in \Delta$ für ein $a \in \Sigma_0\}$
- Wenn es $q_1, \ldots, q_m \in R$ und $a \in \Sigma_m$ gibt mit $a(q_1, \ldots, q_m) \to q \in \Delta$ 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.

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:
- $R := \{q \mid a \to q \in \Delta$ für ein $a \in \Sigma_0\}$
- Wenn es $q_1, \ldots, q_m \in R$ und $a \in \Sigma_m$ gibt mit $a(q_1, \ldots, q_m) \to q \in \Delta$ 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.

16:53 bis 17:18

„in Polyzeit:“ weil R monoton wachsend und durch Q beschränkt!

Letzte Äquivalenz: erkannte Sprache leer gdw. kein akz. Zust. erreichbar

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** $\square$

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 $\square$

16:53 bis 17:18

„in Polyzeit:“ weil R monoton wachsend und durch Q beschränkt!

Letzte Äquivalenz: erkannte Sprache leer gdw. kein akz. Zust. erreichbar

Das Wortproblem alias Zugehörigkeitsproblem

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

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

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

Das Wortproblem alias Zugehörigkeitsproblem
Eingabe: NEBA (oder DEBA) $\mathcal{A}$, Baum T über Σ
Frage: Ist $T \in L(\mathcal{A})$?
d. h. $\mathbf{WP_{NEBA}} = \{(\mathcal{A}, T) \mid \mathcal{A} \text{ NEBA}, \ T \in L(\mathcal{A})\}$ (analog f. DEBAs)

17:18 bis 17:22

„alias“: Bäume sind eben keine Wörter. Werde trotzdem weiter „WP“ benutzen.

LOGCFL:
Probleme, die sich in logspace aufs Wortproblem für kfS reduzieren lassen

LOGDCFL:
dito, aber Red. auf *determinist.* kfS (d. h. durch DPDA erkennbar)

$$\mathrm{L} \subseteq \begin{matrix}\mathrm{LogDCFL} \\ \mathrm{NL}\end{matrix} \subseteq \mathrm{LogCFL} \subseteq \mathrm{P}$$

Das Wortproblem alias Zugehörigkeitsproblem

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

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

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

Satz 2.20

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

Das Wortproblem alias Zugehörigkeitsproblem
Eingabe: NEBA (oder DEBA) $\mathcal{A}$, Baum T über Σ
Frage: Ist $T \in L(\mathcal{A})$?
d. h. $\mathbf{WP}_{\mathbf{NEBA}} = \{(\mathcal{A}, T) \mid \mathcal{A} \text{ NEBA}, \ T \in L(\mathcal{A})\}$ (analog f. DEBAs)
Satz 2.20
WP_{NEBA} und WP_{DEBA} sind entscheidbar und in P.

17:18 bis 17:22

„alias“: Bäume sind eben keine Wörter. Werde trotzdem weiter „WP“ benutzen.

LOGCFL:
Probleme, die sich in logspace aufs Wortproblem für kfS reduzieren lassen

LOGDCFL:
dito, aber Red. auf *determinist.* kfS (d. h. durch DPDA erkennbar)

$$\text{L} \subseteq \begin{matrix}\text{LogDCFL}\\ \text{NL}\end{matrix} \subseteq \text{LogCFL} \subseteq \text{P}$$

Das Wortproblem alias Zugehörigkeitsproblem

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

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

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

Satz 2.20

$\mathrm{WP}_{\mathrm{NEBA}}$ und $\mathrm{WP}_{\mathrm{DEBA}}$ 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!) □

Das Wortproblem alias Zugehörigkeitsproblem
Eingabe: NEBA (oder DEBA) $\mathcal{A}$, Baum T über Σ
Frage: Ist $T \in L(\mathcal{A})$?
d. h. $\mathbf{WP}_{\mathbf{NEBA}} = \{(\mathcal{A}, T) \mid \mathcal{A} \text{ NEBA}, T \in L(\mathcal{A})\}$ (analog f. DEBAs)
Satz 2.20
$\mathrm{WP}_{\mathrm{NEBA}}$ und $\mathrm{WP}_{\mathrm{DEBA}}$ sind entscheidbar und in P.
Beweis. analog zu NEAs, per Reduktion zum LP:
$T \in L(\mathcal{A})$ 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!) □

17:18 bis 17:22

„alias“: Bäume sind eben keine Wörter. Werde trotzdem weiter „WP“ benutzen.

LOGCFL:
Probleme, die sich in logspace aufs Wortproblem für kfS reduzieren lassen

LOGDCFL:
dito, aber Red. auf *determinist.* kfS (d. h. durch DPDA erkennbar)

$$\mathrm{L} \subseteq \begin{matrix}\mathrm{LogDCFL} \\ \mathrm{NL}\end{matrix} \subseteq \mathrm{LogCFL} \subseteq \mathrm{P}$$

Das Wortproblem alias Zugehörigkeitsproblem

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

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

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

Satz 2.20

$\mathrm{WP}_{\mathrm{NEBA}}$ und $\mathrm{WP}_{\mathrm{DEBA}}$ 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!) □

$\mathrm{WP}_{\mathrm{NEBA}}$ ist LOGCFL-vollständig. (zwischen **NL** und **P**)
$\mathrm{WP}_{\mathrm{DEBA}}$ ist in LOG**D**CFL. (Genaue Komplexität ist offen!)
$\mathrm{WP}_{\mathrm{DETDBA}}$ ist **L**-vollständig.

Das Wortproblem alias Zugehörigkeitsproblem
Eingabe: NEBA (oder DEBA) $\mathcal{A}$, Baum T über Σ
Frage: Ist $T \in L(\mathcal{A})$?
d. h. $\mathrm{WP}_{\mathrm{NEBA}} = \{(\mathcal{A}, T) \mid \mathcal{A} \text{ NEBA}, T \in L(\mathcal{A})\}$ (analog f. DEBAs)
Satz 2.20
$\mathrm{WP}_{\mathrm{NEBA}}$ und $\mathrm{WP}_{\mathrm{DEBA}}$ sind entscheidbar und in P.
Beweis. analog zu NEAs, per Reduktion zum LP:
$T \in L(\mathcal{A})$ 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!) □
$\mathrm{WP}_{\mathrm{NEBA}}$ ist LOGCFL-vollständig. (zwischen NL und P)
$\mathrm{WP}_{\mathrm{DEBA}}$ ist in LOGDCFL. (Genaue Komplexität ist offen!)
$\mathrm{WP}_{\mathrm{DETDBA}}$ ist L-vollständig.

17:18 bis 17:22

„alias“: Bäume sind eben keine Wörter. Werde trotzdem weiter „WP“ benutzen.

LOGCFL:
Probleme, die sich in logspace aufs Wortproblem für kfS reduzieren lassen

LOGDCFL:
dito, aber Red. auf *determinist.* kfS (d. h. durch DPDA erkennbar)

$$\mathrm{L} \subseteq \begin{matrix}\mathrm{LogDCFL} \\ \mathrm{NL}\end{matrix} \subseteq \mathrm{LogCFL} \subseteq \mathrm{P}$$

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.

Das Äquivalenzproblem
Eingabe: NEBAs (oder DEBAs) $\mathcal{A}_1, \mathcal{A}_2$
Frage: Ist $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?
d. h. $\text{ÄP}_{\text{NEBA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \text{ NEBAs}, L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$ etc.

17:22 bis 17:25

$\triangle$ = symmetrische Differenz zweier Mengen;
ausdrückbar mittels $\cup, \cap, \bar{\cdot}$ $\rightsquigarrow$ Abschlusseigenschaften!

Untere Schranke für DEBAs: weiß nicht …

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**.

Das Äquivalenzproblem
Eingabe: NEBAs (oder DEBAs) $\mathcal{A}_1, \mathcal{A}_2$
Frage: Ist $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?
d. h. $\text{ÄP}_{\text{NEBA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \text{ NEBAs}, L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$ etc.
Satz 2.21
ÄP_{NEBA} und ÄP_{DEBA} sind entscheidbar.
ÄP_{NEBA} ist ExpTime-vollständig; ÄP_{DEBA} ist in P.

17:22 bis 17:25

$\triangle$ = symmetrische Differenz zweier Mengen;
ausdrückbar mittels $\cup, \cap, \bar{\cdot}$ $\rightsquigarrow$ Abschlusseigenschaften!

Untere Schranke für DEBAs: weiß nicht ...

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:

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

Das Äquivalenzproblem
Eingabe: NEBAs (oder DEBAs) $\mathcal{A}_1, \mathcal{A}_2$
Frage: Ist $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?
d. h. $\text{ÄP}_{\text{NEBA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \text{ NEBAs}, L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$ 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:
$L(\mathcal{A}_1) = L(\mathcal{A}_2)$ gdw. $L(\mathcal{A}_1) \vartriangle L(\mathcal{A}_2) = \emptyset$

17:22 bis 17:25

$\vartriangle$ = symmetrische Differenz zweier Mengen;
ausdrückbar mittels $\cup, \cap, \bar{\cdot}$ $\rightsquigarrow$ Abschlusseigenschaften!

Untere Schranke für DEBAs: weiß nicht ...

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:

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

- obere Schranken: Automat für $L(\mathcal{A}_1) \vartriangle L(\mathcal{A}_2)$ ist exponentiell in der Größe der Eingabe-NEBAs / polynomiell für DEBAs

Das Äquivalenzproblem
Eingabe: NEBAs (oder DEBAs) $\mathcal{A}_1, \mathcal{A}_2$
Frage: Ist $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?
d. h. $\text{ÄP}_{\text{NEBA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \text{ NEBAs}, L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$ 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:
$L(\mathcal{A}_1) = L(\mathcal{A}_2)$ gdw. $L(\mathcal{A}_1) \vartriangle L(\mathcal{A}_2) = \emptyset$
obere Schranken: Automat für $L(\mathcal{A}_1) \vartriangle L(\mathcal{A}_2)$ ist exponentiell in der Größe der Eingabe-NEBAs / polynomiell für DEBAs

17:22 bis 17:25

$\vartriangle$ = symmetrische Differenz zweier Mengen;
ausdrückbar mittels $\cup, \cap, \bar{\cdot}$ $\rightsquigarrow$ Abschlusseigenschaften!

Untere Schranke für DEBAs: weiß nicht …

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:

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

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

Das Äquivalenzproblem
Eingabe: NEBAs (oder DEBAs) $\mathcal{A}_1, \mathcal{A}_2$
Frage: Ist $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?
d. h. $\text{ÄP}_{\text{NEBA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \text{ NEBAs}, L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$ 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:
$L(\mathcal{A}_1) = L(\mathcal{A}_2)$ gdw. $L(\mathcal{A}_1) \vartriangle L(\mathcal{A}_2) = \emptyset$
obere Schranken: Automat für $L(\mathcal{A}_1) \vartriangle L(\mathcal{A}_2)$ ist exponentiell in der Größe der Eingabe-NEBAs / polynomiell für DEBAs
ExpTime-Härte: Reduktion vom Universalitätsproblem (F. 59) □

17:22 bis 17:25

$\vartriangle$ = symmetrische Differenz zweier Mengen;
ausdrückbar mittels $\cup, \cap, \bar{\cdot}$ $\rightsquigarrow$ Abschlusseigenschaften!

Untere Schranke für DEBAs: weiß nicht ...

Das Universalitätsproblem

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

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

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

Das Universalitätsproblem
Eingabe: NEBA (oder DEBA) $\mathcal{A}$
Frage: Ist $L(\mathcal{A}) = \mathcal{T}(\Sigma)$? $(\mathcal{T}(\Sigma) = \{T \mid T \text{ ist ein } \Sigma\text{-Baum}\})$
d. h. $\mathbf{UP}_{\mathbf{NEBA}} = \{\mathcal{A} \mid \mathcal{A} \text{ NEBA}, L(\mathcal{A}) = \mathcal{T}(\Sigma)\}$ (analog f. DEBAs)

17:25 bis 17:27

Das Universalitätsproblem

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

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

d. h. $\mathbf{UP_{NEBA}} = \{\mathcal{A} \mid \mathcal{A}$ NEBA, $L(\mathcal{A}) = \mathcal{T}(\Sigma)\}$ (analog f. DEBAs)

Satz 2.22

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

Das Universalitätsproblem
Eingabe: NEBA (oder DEBA) $\mathcal{A}$
Frage: Ist $L(\mathcal{A}) = \mathcal{T}(\Sigma)$? ($\mathcal{T}(\Sigma)$ $= \{T \mid T$ ist ein Σ-Baum$\})$
d. h. $\mathrm{UP_{NEBA}} = \{\mathcal{A} \mid \mathcal{A}$ NEBA, $L(\mathcal{A}) = \mathcal{T}(\Sigma)\}$ (analog f. DEBAs)
Satz 2.22
$\mathrm{UP_{NEBA}}$ und $\mathrm{UP_{DEBA}}$ sind entscheidbar.
$\mathrm{UP_{NEBA}}$ ist ExpTime-vollständig; $\mathrm{UP_{DEBA}}$ ist in P.

17:25 bis 17:27

Das Universalitätsproblem

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

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

d. h. $\mathbf{UP_{NEBA}} = \{\mathcal{A} \mid \mathcal{A} \text{ NEBA}, L(\mathcal{A}) = \mathcal{T}(\Sigma)\}$ (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!)

2019-02-01

Teil 2: endliche Bäume
└─Entscheidungsprobleme
 └─Das Universalitätsproblem

17:25 bis 17:27

Das Universalitätsproblem

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

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

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

Satz 2.22

$\mathrm{UP_{NEBA}}$ und $\mathrm{UP_{DEBA}}$ sind entscheidbar.
$\mathrm{UP_{NEBA}}$ ist **ExpTime**-vollständig; $\mathrm{UP_{DEBA}}$ 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]) $\square$

2019-02-01

Teil 2: endliche Bäume
└Entscheidungsprobleme
└Das Universalitätsproblem

Das Universalitätsproblem
Eingabe: NEBA (oder DEBA) $\mathcal{A}$
Frage: Ist $L(\mathcal{A}) = \mathcal{T}(\Sigma)$? $(\mathcal{T}(\Sigma) = \{T \mid T \text{ ist ein } \Sigma\text{-Baum}\})$
d. h. $\mathrm{UP_{NEBA}} = \{\mathcal{A} \mid \mathcal{A} \text{ NEBA, } L(\mathcal{A}) = \mathcal{T}(\Sigma)\}$ (analog f. DEBAs)
Satz 2.22
$\mathrm{UP_{NEBA}}$ und $\mathrm{UP_{DEBA}}$ sind entscheidbar.
$\mathrm{UP_{NEBA}}$ ist ExpTime-vollständig; $\mathrm{UP_{DEBA}}$ ist in P.
Beweis:
Entscheidbarkeit & obere Schranken per Red. zum ÄP: $L(\mathcal{A}) = \mathcal{T}(\Sigma)$ 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]) $\square$

17:25 bis 17:27

Überblick Entscheidungsprobleme für NEBAs/DEBAs

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

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

Überblick Entscheidungsprobleme für NEBAs/DEBAs

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

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

17:27 bis 17:29 $\leadsto$ 1 min Reserve
Hier nochmal für Euch als Überblick

Nächstes Mal: XML, Kapitel zuende

Zum **Literaturverzeichnis** blättern

Und nun ...

Und nun ...

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

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

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

8:30

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:

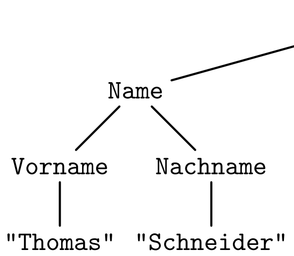
```
Person: {Name: {VN: "Thomas", NN: "Schneider"},
         Telnr: 64432,
         Telnr: 43776243,
         Email: "ts@informatik..."}
```

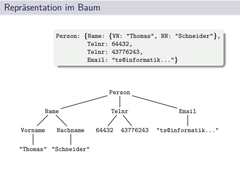

8:30

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..."}
```





8:32

Größeres Bsp.: XML-Dokument für Konferenzprogramm

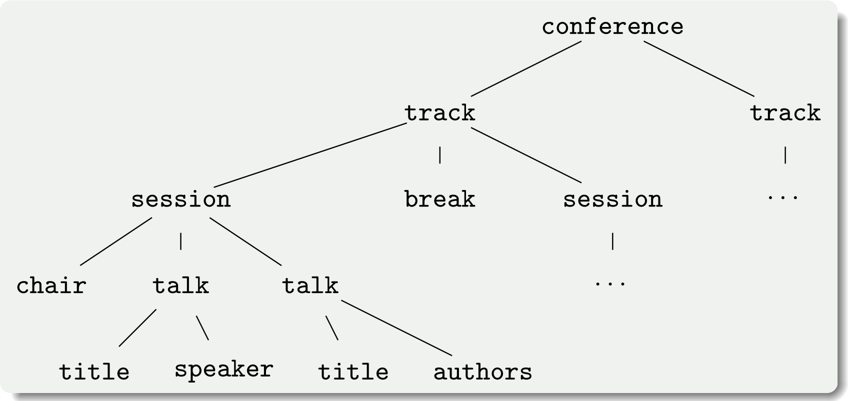
```
<conference>
  <track>
    <session>
       <chair> F. Angorn </chair>
       <talk>
         <title> The Pushdown Hierarchy </title>
         <speaker> D.J. Gaugal </speaker>
       </talk>
       <talk>
         <title> Trees Everywhere </title>
         <authors> B. Aum, T. Rees  </authors>
 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

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

Größeres Bsp.: XML-Dokument für Konferenzprogramm

8:33

Zugehöriger Baum



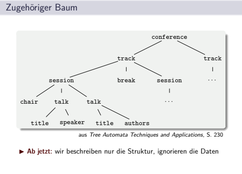
aus *Tree Automata Techniques and Applications*, S. 230

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

2019-02-01

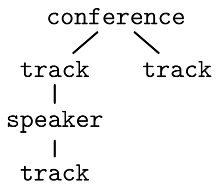
Teil 2: endliche Bäume
└*Anwendung: XML-Schemasprachen*

└Zugehöriger Baum



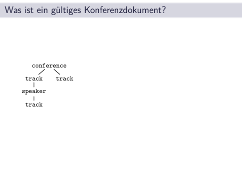
8:35

Was ist ein gültiges Konferenzdokument?



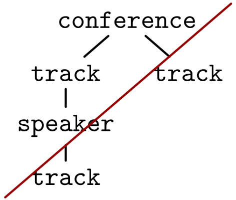
2019-02-01

Teil 2: endliche Bäume
└ *Anwendung: XML-Schemasprachen*
└ Was ist ein gültiges Konferenzdokument?



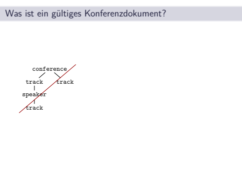
8:37

Was ist ein gültiges Konferenzdokument?



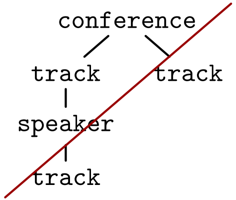
2019-02-01

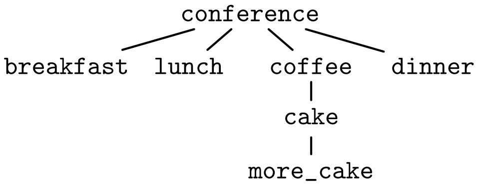
Teil 2: endliche Bäume
└*Anwendung: XML-Schemasprachen*
└Was ist ein gültiges Konferenzdokument?


Was ist ein gültiges Konferenzdokument?

8:37

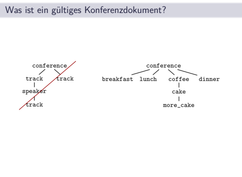
Was ist ein gültiges Konferenzdokument?





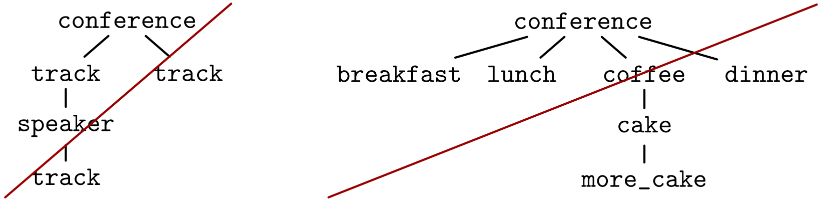
2019-02-01

Teil 2: endliche Bäume
└*Anwendung: XML-Schemasprachen*
└Was ist ein gültiges Konferenzdokument?



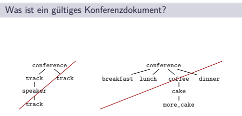
8:37

Was ist ein gültiges Konferenzdokument?



2019-02-01

Teil 2: endliche Bäume
└*Anwendung: XML-Schemasprachen*
└Was ist ein gültiges Konferenzdokument?



8:37

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.

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.

8:38

Gültige Dokumente als Baumsprachen!

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

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

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

Gültige Dokumente als Baumsprachen!

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

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

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

8:40

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**

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**

8:40

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?)

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?)

8:42

Zum ÄP, „Vereinfachung von Schemata“:
Ist das Resultat der Vereinfachung äquivalent zum ursprünglichen Schema?

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"?)

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"?)

8:42

Zum ÄP, „Vereinfachung von Schemata":
Ist das Resultat der Vereinfachung äquivalent zum ursprünglichen Schema?

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)

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)

8:42

Zum ÄP, „Vereinfachung von Schemata":
Ist das Resultat der Vereinfachung äquivalent zum ursprünglichen Schema?

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

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

8:44

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)>
]>
```

, $\hat{=}$ Verkettung

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)>
]>
, ≙ Verkettung

8:45

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)>
]>
```

, ≙ 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

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)>
]>
```

, ≙ 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

8:45

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)>
]>
```

, $\hat{=}$ 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

$\leadsto$ **beliebige Stelligkeit!**

Beispiel-DTD für Konferenzdokumente
<!ELEMENT conference (track+|(session,break?)+)> … , ≙ 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 ⇝ beliebige Stelligkeit!

8:45

Wir brauchen Symbole mit beliebiger Stelligkeit!

Erweitern unser r-Alphabet:

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

Wir brauchen Symbole mit beliebiger Stelligkeit!
Erweitern unser r-Alphabet:
- U: Menge von Symbolen ohne Stelligkeit
- $\Sigma = U \cup \bigcup_{i \geqslant 0} \Sigma_i$

8:48

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 $\{j \mid pj \in P\} = \{1, \dots, m\}$.
  2. Wenn $t(p) \in U$, dann $\{j \mid pj \in P\} = \{1, \dots, k\}$ für ein $k \geqslant 0$.

Wir brauchen Symbole mit beliebiger Stelligkeit!
Erweitern unser r-Alphabet:
U: Menge von Symbolen ohne Stelligkeit
Σ = U ∪ ⋃i⩾0 Σi
Endlicher geordneter Baum T = (P, t) über Σ:
P ⊆ N+* nichtleere endl. präfix-abgeschlossene Menge
t : P → Σ Funktion mit
Wenn t(p) ∈ Σm, dann {j | pj ∈ P} = {1, . . . , m}.
Wenn t(p) ∈ U, dann {j | pj ∈ P} = {1, . . . , k} für ein k ⩾ 0.

8:48

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.): $\Sigma = U$

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
Wenn $t(p) \in \Sigma_m$, dann $\{j \mid pj \in P\} = \{1, \ldots, m\}$.
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.): $\Sigma = U$

8:48

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
- $\boldsymbol{H(\Sigma)}$: Menge aller Hecken über Σ

8:50

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

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
- $\boldsymbol{H(\Sigma)}$: 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 $a(\varepsilon)$

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
- $\boldsymbol{H(\Sigma)}$: 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 $a(\varepsilon)$

8:50

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**

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

8:53 bis 8:58

Beispiele für Hecken und Bäume

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

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

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

2019-02-01

Teil 2: endliche Bäume
└ *Anwendung: XML-Schemasprachen*
└ Beispiele für Hecken und Bäume

Beispiele für Hecken und Bäume
Sei $\Sigma = \{a, b, c\}$.
ε ist eine Hecke
$\leadsto$ $a = a()$ ist ein Baum
$\leadsto$ aa ist eine Hecke
$\leadsto$ $b(aa)$ ist ein Baum
$\leadsto$ $ab(aa)c$ ist eine Hecke
$\leadsto$ $a(ab(aa)c)$ ist ein Baum T 2.11
$(a(c(b)cb(ab))$ ist ein Baum T 2.11 Forts.

8:53 bis 8:58

Heckenautomaten

…sind Bottom-up-Automaten auf Bäumen ohne Stelligkeit

Können sie analog zu NEBAs definiert werden?

Heckenautomaten
…sind Bottom-up-Automaten auf Bäumen ohne Stelligkeit
Können sie analog zu NEBAs definiert werden?

8:58 bis 9:06

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 $a(q_1, \dots, q_m) \to q$ **pro** $m \geqslant 0$. **T 2.12**

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 $a(q_1, \dots, q_m) \to q$ pro $m \geqslant 0$. T 2.12

8:58 bis 9:06

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 $a(q_1, \dots, q_m) \to q$ **pro** $\boldsymbol{m \geqslant 0}$. **T 2.12**

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

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 $a(q_1, \dots, q_m) \to q$ **pro** $\boldsymbol{m \geqslant 0}$. **T 2.12**
Abhilfe: Nutzen **reguläre Ausdrücke** über Q in linken Regelseiten

8:58 bis 9:06

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 $a(q_1, \dots, q_m) \to q$ **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.

8:58 bis 9:06

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 $a(q_1, \dots, q_m) \to q$ 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.

- **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 $r(p1) \cdots r(pm) \in R$.

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.
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 $r(p1) \cdots r(pm) \in R$.

9:06

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.

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

9:06

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.

- **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
  $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$.

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.
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 $L(\mathcal{A}) = \{T \text{ über } \Sigma \mid \mathcal{A} \text{ akzeptiert } T\}$.

9:08

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**

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**

9:08 bis 9:15

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**

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

T 2.13 Forts.

9:08 bis 9:15

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
$L = \{T = (P, t) \mid \text{es gibt } p_1, p_2 \in P \text{ mit } t(p_1) = t(p_2) = b \text{ und } t(\mathrm{tgV}(p_1, p_2) = c)\}$
T 2.13 Forts.

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**

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

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.

9:08 bis 9:15

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
$L = \{T = (P, t) \mid$ es gibt $p_1, p_2 \in P$ mit $t(p_1) = t(p_2) = b$ und $t(\mathrm{tgV}(p_1, p_2) = c)\}$
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

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

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

Beispiel
$L = \{T = (P,t) \mid$ es gibt $p_1, p_2 \in P$ mit $t(p_1) = t(p_2) = b$ und $t(\mathrm{tgV}(p_1,p_2) = c)\}$
$\mathcal{A} = (Q, \Sigma, \Delta, F)$ mit $Q = \{q_0, q_b, q_c\}$, $F = \{q_c\}$ und

9:15 bis 9:21; 5min Pause $\rightarrow$ bis 9:26

(TODO: einarbeiten) $\mathcal{A}$ ist natürlich stark nichtdeterministisch: mehrere Übergänge mit demselben Buchstaben und Zustandsfolge, z. B. $a(q_b) \rightarrow q_0$ und $a(q_b) \rightarrow q_b$.

Das kann man beheben, indem man die Ausdrücke, die mit demselben Zeichen auftreten, paarweise disjunkt macht, z. B. hier:

1. Spalte: $a(q_0^*) \rightarrow q_0$, dito für c, $b((q_0 + q_c)^*) \rightarrow q_b$
2. Spalte: $a(q_0^* q_b q_0^*) \rightarrow q_b$, dito für c, $c((q_0 + q_c)^* q_b (q_0 + q_c)^* q_b (q_0 + q_c)^*) \rightarrow q_c$
3. Spalte: bleibt so

Beispiel

$$L = \{T = (P,t) \mid \text{es gibt } p_1, p_2 \in P \text{ mit } t(p_1) = t(p_2) = b \text{ und } t(\text{tgV}(p_1,p_2) = c)\}$$

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

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

T 2.13 Forts.

9:15 bis 9:21; 5min Pause → bis 9:26

(TODO: einarbeiten) $\mathcal{A}$ ist natürlich stark nichtdeterministisch: mehrere Übergänge mit demselben Buchstaben und Zustandsfolge, z. B. $a(q_b) \to q_0$ und $a(q_b) \to q_b$.

Das kann man beheben, indem man die Ausdrücke, die mit demselben Zeichen auftreten, paarweise disjunkt macht, z. B. hier:

1. Spalte: $a(q_0^*) \to q_0$, dito für c, $b((q_0 + q_c)^*) \to q_b$
2. Spalte: $a(q_0^* q_b q_0^*) \to q_b$, dito für c, $c((q_0 + q_c)^* q_b (q_0 + q_c)^* q_b (q_0 + q_c)^*) \to q_c$
3. Spalte: bleibt so

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)>
]>
```


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)>
]>

9:26

Wollen jetzt sehen, wie man DTDs in NEHAs umwandelt,
um Automatentechniken auf Validierung usw. anwenden zu können.

1. Schritt: erweiterte kfG, die der DTD entspricht (hier am Bsp.)
(„erweitert“: mit regulären Ausdrücken auf rechten Regelseiten)

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:

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

Umwandlung der Beispiel-DTD in einen NEHA (1)

9:26

Wollen jetzt sehen, wie man DTDs in NEHAs umwandelt,
um Automatentechniken auf Validierung usw. anwenden zu können.

1. Schritt: erweiterte kfG, die der DTD entspricht (hier am Bsp.)
(„erweitert“: mit regulären Ausdrücken auf rechten Regelseiten)

Umwandlung der Beispiel-DTD in einen NEHA (2)

Erweiterte kontextfreie Grammatik (Wdhlg.):

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

Umwandlung der Beispiel-DTD in einen NEHA (2)
Erweiterte kontextfreie Grammatik (Wdhlg.):

9:29 bis 9:35

Wie funktionieren eigentlich **erweiterte** kfGs?
Hier keine formale Def., sondern Vorgehen bei einzelnem Ableitungsschritt

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

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
...
title → DATA
Startsymbol: hier conference
Ableitungsschritt:
Wähle mit ℓ beschriftetes Blatt, ℓ ∈ Σ
Wähle Regel ℓ → R (R: reg. Sprache über Σ, Inhaltsmodell)
Wähle a1 ··· an ∈ R und füge Kinder a1, ..., an zu ℓ hinzu

9:29 bis 9:35

Wie funktionieren eigentlich **erweiterte** kfGs?
Hier keine formale Def., sondern Vorgehen bei einzelnem Ableitungsschritt

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 (2)
Erweiterte kontextfreie Grammatik (Wdhlg.):
Startsymbol: hier conference
Ableitungsschritt:
Wähle mit ℓ beschriftetes Blatt, ℓ ∈ Σ
Wähle Regel ℓ → R (R: reg. Sprache über Σ, Inhaltsmodell)
Wähle a1 ⋯ an ∈ R und füge Kinder a1, …, an zu ℓ hinzu
Beispielableitung: siehe Tafel T 2.14

9:29 bis 9:35

Wie funktionieren eigentlich **erweiterte** kfGs?
Hier keine formale Def., sondern Vorgehen bei einzelnem Ableitungsschritt

Umwandlung der Beispiel-DTD in einen NEHA (3)

Erweiterte kontextfreie Grammatik (Wdhlg.):

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

Umwandlung der Beispiel-DTD in einen NEHA (3)
Erweiterte kontextfreie Grammatik (Wdhlg.):
conference → track⁺ + (session (break + ε))⁺
track → (session (break + ε))⁺
title → DATA

9:35 bis 9:38

Schritt 2: NEHA aus Grammatik bauen

- Zustände = Alphabet
- Δ: aus den Produktionen (Zustände kursiv, Zeichen aufrecht)

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

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

Umwandlung der Beispiel-DTD in einen NEHA (3)

9:35 bis 9:38

Schritt 2: NEHA aus Grammatik bauen

- Zustände = Alphabet
- Δ: aus den Produktionen (Zustände kursiv, Zeichen aufrecht)

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.)

9:38

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.)

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

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

2019-02-01

Teil 2: endliche Bäume
└Anwendung: XML-Schemasprachen

└Präzise Definition DTD & zugehöriger NEHA

9:38

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.

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.

8:30, nur Wdhlg. 9:41 bis 9:52

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.?

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.?

8:30, nur Wdhlg. 9:41 bis 9:52

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?)

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?)

8:30, nur Wdhlg. 9:41 bis 9:52

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 . . .

8:30, nur Wdhlg. 9:41 bis 9:52

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 . . .

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“

Lokale Sprachen
Definition 2.26
Die von einer DTD D erzeugte Sprache ist L(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“

8:30, nur Wdhlg. 9:41 bis 9:52

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“)

2019-02-01

Teil 2: endliche Bäume
└ *Anwendung: XML-Schemasprachen*
└ Lokale Sprachen

Lokale Sprachen
Definition 2.26
Die von einer DTD D erzeugte Sprache ist L(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 ⩾ 5 Vortragende“)

8:30, nur Wdhlg. 9:41 bis 9:52

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

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

8:31, nur Wdhlg. 9:52

DTDs werden aber laut W3C-Standard nochmal eingeschränkt, nämlich wie folgt.

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.

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.

8:31, nur Wdhlg. 9:52

DTDs werden aber laut W3C-Standard nochmal eingeschränkt, nämlich wie folgt.

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.

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

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

8:31, nur Wdhlg. 9:52

DTDs werden aber laut W3C-Standard nochmal eingeschränkt, nämlich wie folgt.

Deterministische Inhaltsmodelle – Beispiel

Betrachte die Zeile

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

und die zugehörige Regel

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

Deterministische Inhaltsmodelle – Beispiel
Betrachte die Zeile
<!ELEMENT talk ((title,authors)|(title,speaker))>
und die zugehörige Regel
talk → (title authors) + (title speaker)

8:32, nur Wdhlg. 9:54

Deterministische Inhaltsmodelle – Beispiel

Betrachte die Zeile

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

und die zugehörige Regel

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

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

Deterministische Inhaltsmodelle – Beispiel

Betrachte die Zeile
<!ELEMENT talk ((title,authors)|(title,speaker))>
und die zugehörige Regel
talk → (title authors) + (title speaker)
Für Wörter über Σ, die mit dem Buchstaben title beginnen, ist nicht klar, welchem Vorkommen von title im Inhaltsmodell dieser Buchstabe entspricht!

8:32, nur Wdhlg. 9:54

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 $i \neq j$ enthält.

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 $i \neq j$ enthält.

8:33, nur Wdhlg. 9:55 bis 10:00 $\to$ Ende? (Rest dauert noch $\approx$5 min)

TODO: Klarmachen, dass RAe über $+, \cdot, ^*$ definiert sind und kein $\cdot^+$ erlaubt ist!

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 $i \neq j$ enthält.

Etwas Notation:

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

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.: $(a+b)^*b(ab)^* \rightsquigarrow (a_1+b_1)^*b_2(a_2b_3)^* =: r'$.
r ist deterministisch, wenn $L(r')$ keine zwei Wörter ua_iv und ua_jw mit $i \neq j$ enthält.
Etwas Notation:
RA r über $\Sigma \rightsquigarrow$ markierter RA r' über Σ'
wie üblich: $L(r) \subseteq \Sigma^*$ und $L(r') \subseteq \Sigma'^*$

8:33, nur Wdhlg. 9:55 bis 10:00 $\longrightarrow$ Ende? (Rest dauert noch $\approx$5 min)

TODO: Klarmachen, dass RAe über $+, \cdot, ^*$ definiert sind und kein $\cdot^+$ erlaubt ist!

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 $i \neq j$ enthält.

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**

8:33, nur Wdhlg. 9:55 bis 10:00 $\rightarrow$ Ende? (Rest dauert noch $\approx$5 min)

TODO: Klarmachen, dass RAe über $+, \cdot, ^*$ definiert sind und kein $\cdot^+$ erlaubt ist!

Was nützen uns nun DRAs?

2019-02-01

Teil 2: endliche Bäume
└─*Anwendung: XML-Schemasprachen*

└─Was nützen uns nun DRAs?

Was nützen uns nun DRAs?

8:34 (10:00 bis 10:01)

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.)

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.)

8:34 **(10:00 bis 10:01)**

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 $a(R) \to q \in \Delta$ R als DEA gegeben ist.

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 $a(R) \to q \in \Delta$ R als DEA gegeben ist.

8:34 (10:00 bis 10:01)

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 $a(R) \to q \in \Delta$ R als DEA gegeben ist.

Und dieses Resultat garantiert nun . . . ?

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 $a(R) \to q \in \Delta$ R als DEA gegeben ist.
Und dieses Resultat garantiert nun . . . ?

8:34 (10:00 bis 10:01)

Deterministische DTDs sind effizient!

Satz 2.30

Für deterministische DTDs sind in Polynomialzeit lösbar:

- das Zugehörigkeitsproblem
- das Leerheitsproblem
- das Äquivalenzproblem

(Ohne Beweis.)

Deterministische DTDs sind effizient!
Satz 2.30
Für deterministische DTDs sind in Polynomialzeit lösbar:
- das Zugehörigkeitsproblem
- das Leerheitsproblem
- das Äquivalenzproblem
(Ohne Beweis.)

8:35 (10:01 bis 10:02)

„Ohne Beweis“: eins davon voraussichtlich auf Ü-Blatt 3

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?

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?

8:35 (10:01 bis 10:02)

„Ohne Beweis“: eins davon voraussichtlich auf Ü-Blatt 3

Sind deterministische DTDs schwächer als allgemeine?

Sind deterministische DTDs schwächer als allgemeine?

8:36 (10:02 bis 10:03)

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:

8:36 (10:02 bis 10:03)

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:

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.

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.

8:36 **(10:02 bis 10:03)**

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**.

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.

8:37 (10:03 bis 10:04)

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

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

8:38 **(10:04 bis 10:05)**

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

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

8:38 (10:04 bis 10:05)

Damit sind wir am Ende dieses Kapitels.

Vielen Dank.

2019-02-01

Teil 2: endliche Bäume

└Damit sind wir am Ende dieses Kapitels.

8:39 → nochmal zur Literatur, bis 8:40 (10:05) Ende → nochmal zur Literatur

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 (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.

2019-02-01
Teil 2: endliche Bäume
└─Literatur für diesen Teil (weiterführend)

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.

2019-02-01

Teil 2: endliche Bäume