Abschlusseig.

Grundbegriffe

Reguläre Ausdrücke

Charakterisierungen

Wintersemester 2018/19 Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)

http://tinyurl.com/ws1819-autom

Teil 1: endliche Wörter

Automatentheorie und ihre Anwendungen

Wintersemester 2018/19 Thomas Schneide http://tinyurl.com/ws1819-autom

Vorlesungsübersicht

Kapitel 1: endliche Automaten auf endlichen Wörtern

Kapitel 2: endliche Automaten auf endlichen Bäumen

Kapitel 3: endliche Automaten auf unendlichen Wörtern

Kapitel 4: endliche Automaten auf unendlichen Bäumen

Teil 1: endliche Wörter

└─Vorlesungsübersicht

Kapitel 1: endliche Automaten auf endlichen Wörtern Kapitel 2: endliche Automaten auf endlichen Bäumen Kapitel 3: endliche Automaten auf unendlichen Wörtern Kapitel 4: endliche Automaten auf unendlichen Bäumen

Vorlesungsübersicht

8:55

Hier nochmal die 4 Kapitel (kurz!)

Ziel dieses Kapitels

- Wiederholung der Definitionen & Resultate zu endlichen Automaten aus "Theoretische Informatik 1"
- Kennenlernen zweier Anwendungen endlicher Automaten



8:55

Das wird ein kurzer, leichter Teil. Ca. 2 Sitzungen.

- (1) Wdhlg. mache ich knapp; wenn Euch auffällt, was Ihr nicht mehr parat habt, dann schlagt es im Skript nach.
- (2) Die Anwendungen sind (hoffentlich) neu für Euch.

Überblick

Grundbegriffe

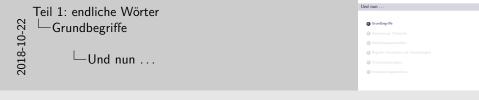
- Grundbegriffe
- 2 Anwendung: Textsuche
- 3 Abschlusseigenschaften
- 4 Reguläre Ausdrücke und Anwendungen
- 5 Charakterisierungen
- 6 Entscheidungsprobleme

Teil 1: endliche Wörter

Office of the standard of Testandra
Office of the standard of Testandra
Office of the standard of Testandra
Office of the standard of the standard

Und nun ...

- Grundbegriffe
- 2 Anwendung: Textsuche
- 3 Abschlusseigenschaften
- Reguläre Ausdrücke und Anwendungen
- 5 Charakterisierunger
- 6 Entscheidungsprobleme



Grundbegriffe Textsuche Abschlusseig. Reguläre Ausdrücke Charakterisierungen

Wörter, Sprachen, ...

• Symbole *a*, *b*, . . .

- Alphabet Σ: endliche nichtleere Menge von Symbolen
- (endliches) Wort w über Σ : endliche Folge $w = a_1 a_2 \dots a_n$ von Symbolen $a_i \in \Sigma$
- ullet leeres Wort arepsilon
- Wortlänge $|a_1 a_2 \dots a_n| = n$, $|\varepsilon| = 0$
- Menge aller Wörter über Σ : Σ^*
- Sprache L über Σ : Teilmenge $L \subseteq \Sigma^*$ von Wörtern
- Sprachklasse L: Menge von Sprachen

8:56

2018-10-22

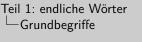
Entscheidungsprobleme

Endliche Automaten

Definition 1.1

Ein nichtdeterministischer endlicher Automat (NEA) über einem Alphabet Σ ist ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, wobei

- Q eine endliche nichtleere Zustandsmenge ist,
- Σ ein Alphabet ist,
- $\Delta \subseteq Q \times \Sigma \times Q$ die Überführungsrelation ist, (*)
- $I \subseteq Q$ die Menge der Anfangszustände ist,
- $F \subset Q$ die Menge der akzeptierenden Zustände ist.



Delention II. I is middeterministischer endicher Automat (NEA) über einer Alphabetz ist ein STupul $A = (Q \times A, I, F)$, webei Q eine endliche nichtleser Zustandienunge sist, Q eine endliche nichtleser Zustandienunge sist, Q eine Alphabet ist, Q ex Q

Endliche Automaten

Endliche Automaten

8:57

2018-

Einziger Unterschied zur Def. aus Theorie 1: mehrere Anfangszustände (lassen sich aber immer auf 1 reduzieren)

Außerdem steht Δ vor I – das ist aber nur Festlegungssache.

Akz. Zustände werden oft Endzustände genannt (en: final states $\leadsto F$). Das kann aber für Verwirrung sorgen, denn die Berechnung muss beim Erreichen eines solchen Zustandes noch nicht enden.

Deshalb benutze ich "akz. Zustände".

Endliche Automaten

Definition 1.1

Ein nichtdeterministischer endlicher Automat (NEA) über einem Alphabet Σ ist ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, wobei

- Q eine endliche nichtleere Zustandsmenge ist,
- Σ ein Alphabet ist,
- $\Delta \subset Q \times \Sigma \times Q$ die Überführungsrelation ist, (*)
- $I \subseteq Q$ die Menge der Anfangszustände ist,
- $F \subseteq Q$ die Menge der akzeptierenden Zustände ist.
- (*) bedeutet: Δ besteht aus Tripeln (q, a, q') mit $q, q' \in Q$ und $a \in \Sigma$
- $(q, a, q') \in \Delta$ bedeutet intuitiv: ist A in Zustand q und liest ein a, geht er in Zustand q' über.

Teil 1: endliche Wörter -Grundbegriffe

-Endliche Automaten

Alphabet Σ ist ein 5-Tupel $A = (Q, \Sigma, \Delta, I, F)$, wobe O eine endliche nichtleere Zustandsmenge ist Δ ⊂ Q × Σ × Q die Überf

ührungsrelation ist,

u I ⊂ Q die Menge der Anfangszustände ist u F ⊂ Q die Menge der akzeptierenden Zustände is

Endliche Automaten

 Δ besteht aus Tripeln (q, a, q') mit $q, q' \in Q$ und $a \in \Sigma$

 $u(q, a, q') \in \Delta$ bedeutet intuitiv: ist A in Zustand q und liest ein a, geht er in Zustand q' über

8:57

2018-1

Einziger Unterschied zur Def. aus Theorie 1: mehrere Anfangszustände (lassen sich aber immer auf 1 reduzieren)

Außerdem steht Δ vor I – das ist aber nur Festlegungssache.

Akz. Zustände werden oft Endzustände genannt (en: final states $\rightsquigarrow F$). Das kann aber für Verwirrung sorgen, denn die Berechnung muss beim Erreichen eines solchen Zustandes noch nicht enden. Deshalb benutze ich "akz. Zustände".

Betrachte $\mathcal{A} = \{(q_0, q_1\}, \{a, b\}, \{(q_0, a, q_0), (q_0, b, q_1), (q_1, b, q_1)\}, \{q_0\}, \{q_1\}\}$

- Zustände: q_0, q_1
- Alphabet $\{a, b\}$
- Übergänge: von q_0 mittels a zu q_0, \ldots
- Anfangszustand q₀
- einziger akzeptierender Zustand q_1

Teil 1: endliche Wörter Grundbegriffe

Beispiel und graphische Repräsentation von NEAs

Beispiel und graphische Repräsentation von NEAs

 $\begin{array}{l} \text{Betrachte } \mathcal{A} = \\ (\{q_0,q_1\}, \ \{a,b\}, \ \{(q_0,a,q_0), (q_0,b,q_1), (q_1,b,q_1)\}, \ \{q_0\}, \{q_1\}) \end{array} \right]$

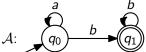
- \mathbf{z} Zustände: q_0,q_1
- Alphabet {a, b}
 Übergänge: von q₀ mittels a zu q₀, . .
- Anfangszustand q₀
 einziger akzeptierender Zustand o

9:00

2018-1

Betrachte A = $(\{q_0, q_1\}, \{a, b\}, \{(q_0, a, q_0), (q_0, b, q_1), (q_1, b, q_1)\}, \{q_0\}, \{q_1\})$

- Zustände: q_0, q_1
- Alphabet {a, b}
- Übergänge: von q_0 mittels a zu q_0, \ldots
- Anfangszustand q₀
- einziger akzeptierender Zustand q₁

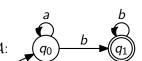












Teil 1: endliche Wörter -Grundbegriffe

> -Beispiel und graphische Repräsentation von NEAs

Beispiel und graphische Repräsentation von NEAs $\{q_0, q_1\}, \{a, b\}, \{(q_0, a, q_0), (q_0, b, q_1), (q_1, b, q_1)\}, \{q_0\}, \{q_1\})$ ■ Zustände: q₀, q₁ · Alphabet {a,b} Obergänge: von q₀ mittels a zu q₀. a einziger akzeptierender Zustand ø

Definition 1.2

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

• Ein Run von \mathcal{A} auf $w = a_1 a_2 \dots a_n$ ist eine Folge

$$r = q_0 q_1 q_2 \dots q_n$$

so dass für alle i = 0, ..., n - 1 gilt: $(q_i, a_{i+1}, q_{i+1}) \in \Delta$. Man sagt/schreibt: w überführt q_0 in q_n , $q_0 \vdash_{\perp}^{w} q_n$

Teil 1: endliche Wörter Grundbegriffe Berechnungen und Akzeptanz

Sei $A = (Q, \Sigma, \Delta, I, F)$ ein NEA. Ein Run von A auf w = a.a. . . . a. ist eine Folge $r = q_0q_1q_2 \dots q_n$ so dass für alle $i=0,\ldots,n-1$ gilt: $(q_i,a_{i+1},q_{i+1})\in\Delta$ Man saet/schreibt: w überführt on in g., on H o.

Berechnungen und Akzeptanz

9:02

2018-10-22

 \vdash_{A}^{w} war in Theorie 1 \Longrightarrow_{A}

Definition 1.2

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

• Ein Run von \mathcal{A} auf $w = a_1 a_2 \dots a_n$ ist eine Folge

$$r = q_0 q_1 q_2 \dots q_n$$

so dass für alle $i = 0, \ldots, n-1$ gilt: $(q_i, a_{i+1}, q_{i+1}) \in \Delta$. Man sagt/schreibt: w überführt q_0 in q_n , $q_0 \vdash_{\perp}^{w} q_n$

• Ein Run $r = q_0 q_1 q_2 \dots q_n$ von \mathcal{A} auf w ist erfolgreich, wenn $q_0 \in I$ und $q_n \in F$.

Teil 1: endliche Wörter -Grundbegriffe Berechnungen und Akzeptanz

Sei $A = (Q, \Sigma, \Delta, I, F)$ ein NEA. Ein Run von A auf w = a.a. . . . a. ist eine Folge $r = q_0q_1q_2 \dots q_n$ so dass für alle $i=0,\ldots,n-1$ gilt: $(q_i,a_{i+1},q_{i+1})\in\Delta$ Man sagt/schreibt: w überführt q₀ in q_n, q₀ ⊢_A q₀ w Ein Run $r = g_1g_1g_2...g_n$ von A auf w ist erfolgreich.

Berechnungen und Akzeptanz

9:02

2018-10-22

 \vdash_A^w war in Theorie 1 \Longrightarrow_A

Berechnungen und Akzeptanz

Abschlusseig.

Definition 1.2

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

• Ein Run von \mathcal{A} auf $w = a_1 a_2 \dots a_n$ ist eine Folge

$$r = q_0 q_1 q_2 \dots q_n$$

so dass für alle $i = 0, \ldots, n-1$ gilt: $(q_i, a_{i+1}, q_{i+1}) \in \Delta$. Man sagt/schreibt: w überführt q_0 in q_n , $q_0 \vdash_{\perp}^w q_n$

- Ein Run $r = q_0 q_1 q_2 \dots q_n$ von \mathcal{A} auf w ist erfolgreich, wenn $q_0 \in I$ und $q_n \in F$.
- A akzeptiert w, wenn es einen erfolgreichen Run von A auf w gibt.

Teil 1: endliche Wörter -Grundbegriffe

Berechnungen und Akzeptanz

Sei $A = (Q, \Sigma, \Delta, I, F)$ ein NEA. Ein Run von A auf w = a.a. . . . a. ist eine Folgs so dass für alle i = 0, ..., n-1 gilt: $(q_i, a_{i+1}, q_{i+1}) \in \Delta$ Man sagt/schreibt: w überführt q₀ in q_n, q₀ ⊢_A q₀ Ein Run r = g₁g₁g₂...g_n von A auf w ist erfolgreich. · A akzeptiert w, wenn es einen erfolgreichen Run von A auf w

Berechnungen und Akzeptanz

9:02

2018-10

 \vdash_{Δ}^{w} war in Theorie 1 \Longrightarrow_{A}

Berechnungen und Akzeptanz

Definition 1.2

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

• Ein Run von A auf $w = a_1 a_2 \dots a_n$ ist eine Folge

$$r = q_0 q_1 q_2 \dots q_n$$

so dass für alle $i = 0, \ldots, n-1$ gilt: $(q_i, a_{i+1}, q_{i+1}) \in \Delta$. Man sagt/schreibt: w überführt q_0 in q_n , $q_0 \vdash_{\perp}^w q_n$

- Ein Run $r = q_0 q_1 q_2 \dots q_n$ von \mathcal{A} auf w ist erfolgreich, wenn $q_0 \in I$ und $q_n \in F$.
- A akzeptiert w, wenn es einen erfolgreichen Run von A auf w gibt.
- Die von A erkannte Sprache ist $L(A) = \{ w \in \Sigma^* \mid A \text{ akzeptiert } w \}.$

Teil 1: endliche Wörter -Grundbegriffe

Berechnungen und Akzeptanz

Sei $A = (Q, \Sigma, \Delta, I, F)$ ein NEA. Ein Run von A auf w = a₁a₂ . . . a₂ ist eine Folge so dass für alle i = 0, ..., n-1 gilt: $(q_i, a_{i+1}, q_{i+1}) \in \Delta$ Man sagt/schreibt: w überführt q_0 in q_a , $q_0 \vdash_A^w q_a$ Ein Run r = g₁g₁g₂...g_n von A auf w ist erfolgreich. · A akzeptiert w, wenn es einen erfolgreichen Run von A auf Die von A erkannte Sprache ist

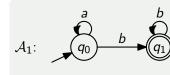
 $L(A) = \{w \in \Sigma^* \mid A \text{ akzeptiert } w\}.$

9:02

2018-1

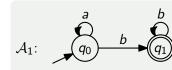
 \vdash_A^w war in Theorie 1 \Longrightarrow_A

Grundbegriffe



$$L(A_1) =$$





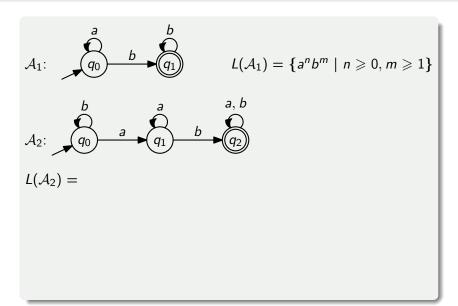
$$L(\mathcal{A}_1) = \{a^n b^m \mid n \geqslant 0, m \geqslant 1\}$$

Teil 1: endliche Wörter
C7-01-8
Grundbegriffe

— Beispiele



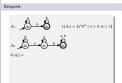
Grundbegriffe

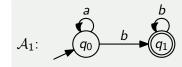


Teil 1: endliche Wörter

7-01

Beispiele



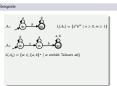


$$L(\mathcal{A}_1) = \{a^n b^m \mid n \geqslant 0, m \geqslant 1\}$$

$$A_2$$
: q_0 q_1 q_2

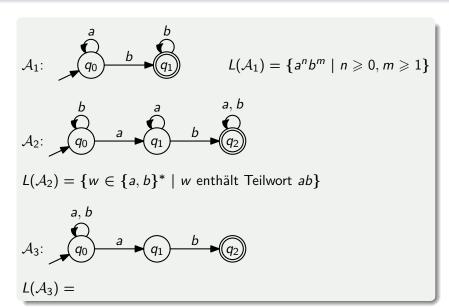
$$L(A_2) = \{ w \in \{a, b\}^* \mid w \text{ enthält Teilwort } ab \}$$

Teil 1: endliche Wörter -Grundbegriffe Beispiele



9:04

2018-10-22



Teil 1: endliche Wörter

Grundbegriffe

Beispiele

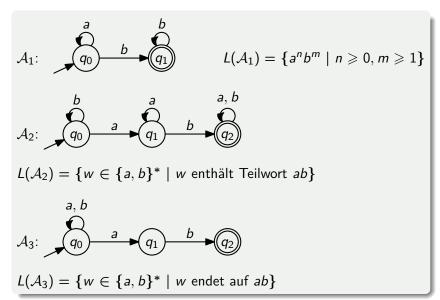
Beispiele

L(A) = (*** (a, b) + | w enthlis Talkeur ab)

L(A) = (*** (a, b) + | w enthlis Talkeur ab)

9:04

2018-10-22



Teil 1: endliche Wörter

Grundbegriffe

As Describe

U(A) = (*P** | section Televort ab)

U(A) = (**e* (A.b)** | se contain Televort ab)

U(A) = (**e* (A.b)** | se contain Televort ab)

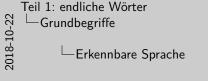
2018-10-22

Erkennbare Sprache

Grundbegriffe

Definition 1.3

Eine Sprache $L \subseteq \Sigma^*$ ist (NEA-)erkennbar, wenn es einen NEA \mathcal{A} gibt mit $L = L(\mathcal{A})$.



Definition 1.3 Eine Sprache $L \subseteq \Sigma^*$ ist (NEA-)erkensbar, wenn es einen NEA A gibt mit L = L(A).

Erkennbare Sprache

Determinismus

Definition 1.4

Sei $A = (Q, \Sigma, \Delta, I, F)$ ein NEA.

Enthält Δ für jedes $q \in Q$ u. jedes $a \in \Sigma$ genau 1 Tripel (q, a, q') und enthält I genau 1 Zustand,

dann ist \mathcal{A} ein deterministischer endlicher Automat (DEA).

Teil 1: endliche Wörter Grundbegriffe

—Determinismus

9:08

Tafelanschrieb: nur Bsp. ganz kurz

Determinismus

Sei $A = (Q, \Sigma, \Delta, I, F)$ ein NEA. Enthält Δ für jedes $q \in Q$ u. jedes $a \in \Sigma$ genau 1 Tripel (q, a, q')und enthält I genau 1 Zustand, dann ist A ein deterministischer endlicher Automat (DEA).

Determinismus

Definition 1.4

Sei $A = (Q, \Sigma, \Delta, I, F)$ ein NEA.

Enthält Δ für jedes $q \in Q$ u. jedes $a \in \Sigma$ genau 1 Tripel (q, a, q') und enthält I genau 1 Zustand,

dann ist A ein deterministischer endlicher Automat (DEA).

 \rightarrow Nachfolgezustand für jedes Paar (q, a) eindeutig bestimmt

Teil 1: endliche Wörter Grundbegriffe

—Determinismus

9:08

Tafelanschrieb: nur Bsp. ganz kurz

Determinismus

Sei $A = \{Q, \Sigma, \Delta, l, F\}$ ein NEA. Enthilt Δ für jedes $q \in Q$ u. jedes $a \in \Sigma$ genau 1 Tripel $\{q, a, q'\}$ und enthilt I genau 1 Zustand, dans ist A ein deterministischer endlicher Automat (DEA).

→ Nachfolgezustand f
ür jedes Paar (q, a) eindeutig bestimm

Determinismus

Definition 1.4

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

Enthält Δ für jedes $q \in Q$ u. jedes $a \in \Sigma$ genau 1 Tripel (q, a, q') und enthält I genau 1 Zustand,

dann ist A ein deterministischer endlicher Automat (DEA).

- \rightarrow Nachfolgezustand für jedes Paar (q, a) eindeutig bestimmt
 - Jeder DEA ist ein NEA,
 aber nicht umgekehrt (z. B. A₁, A₃ auf Folie 10).

Teil 1: endliche Wörter Grundbegriffe

—Determinismus

inismus

• Jeder DEA ist ein NEA,
aber nicht umgekehrt (z. B. .A., .A.) auf Felin 10).

Determinismus

Sei $A = (Q, \Sigma, \Delta, I, F)$ ein NEA. Enthält Δ für jedes $q \in Q$ u. jedes $a \in \Sigma$ genau 1 Tripel (q, a, q^n) und enthält I genau 1 Zustand, dann ist A ein deterministischer endlicher Automat (DEA).

→ Nachfolgezustand f
ür jedes Paar (q, a) eindeutig bestimm

9:08

2018-1

Tafelanschrieb: nur Bsp. ganz kurz

Determinismus

Definition 1.4

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

Abschlusseig.

Enthält Δ für jedes $q \in Q$ u. jedes $a \in \Sigma$ genau 1 Tripel (q, a, q')und enthält / genau 1 Zustand,

dann ist A ein deterministischer endlicher Automat (DEA).

- \rightarrow Nachfolgezustand für jedes Paar (q, a) eindeutig bestimmt
 - Jeder DEA ist ein NEA. aber nicht umgekehrt (z. B. A_1 , A_3 auf Folie 10).
 - Auf Folie 10 ist nur A_2 ein DEA; A_1 kann mittels *Papierkorbzustand* zum DEA werden; T 1.1 bei A_3 genügt auch das nicht.

Teil 1: endliche Wörter Grundbegriffe

-Determinismus

9:08

2018-1

Tafelanschrieb: nur Bsp. ganz kurz

Sei $A = (Q, \Sigma, \Delta, I, F)$ ein NEA. Enthält Δ für jedes $q \in Q$ u. jedes $a \in \Sigma$ genau 1 Tripel (q, a, q')dann ist A ein deterministischer endlicher Automat (DEA)

- → Nachfolgezustand f
 ür jedes Paar (q, a) eindeutig bestimm

 - aber nicht umgekehrt (z.B. A., A. auf Folie 10).
 - w Auf Folie 10 ist nur As ein DEA A₁ kann mittels Papierkorbzustand zum DEA werden; T1.1 bei As genügt auch das nicht.

Potenzmengenkonstruktion

Frage: Sind DEAs und NEAs gleichmächtig?



9:11

Klären: was heißt "gleichmächtig"? (L DEA-erkennbar gdw. L NEA-erkennbar. Hinrichtung trivial.)

Hier nur kurz die Konstruktion und ein Beispiel.

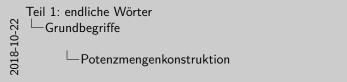
Vollständiger Beweis siehe Theorie 1.

(Wir werden fast dieselbe Konstruktion später an Baumautomaten genauer ausführen.)

Potenzmengenkonstruktion

Frage: Sind DEAs und NEAs gleichmächtig?

Antwort: Ja!



9:11

Klären: was heißt "gleichmächtig"? (L DEA-erkennbar gdw. L NEA-erkennbar. Hinrichtung trivial.)

Hier nur kurz die Konstruktion und ein Beispiel.

Vollständiger Beweis siehe Theorie 1.

(Wir werden fast dieselbe Konstruktion später an Baumautomaten genauer ausführen.)

Potenzmengenkonstruktion

Potenzmengenkonstruktion

Frage: Sind DEAs und NEAs gleichmächtig?

Antwort: Ja!

Satz 1.5 (Rabin, Scott 1959)

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

Teil 1: endliche Wörter

Grundbegriffe

Grundbegriffe

Potenzmengenkonstruktion

Potenzmengenkonstruktion

9:11

Klären: was heißt "gleichmächtig"? (L DEA-erkennbar gdw. L NEA-erkennbar. Hinrichtung trivial.)

Hier nur kurz die Konstruktion und ein Beispiel.

Vollständiger Beweis siehe Theorie 1.

(Wir werden fast dieselbe Konstruktion später an Baumautomaten genauer ausführen.)

Potenzmengenkonstruktion

Frage: Sind DEAs und NEAs gleichmächtig?

Antwort: Ja!

Satz 1.5 (Rabin, Scott 1959)

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

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

Wir konstruieren $A^d = (Q^d, \Sigma, \Delta^d, I^d, F^d)$ wie folgt.

- $Q^d = 2^Q$ (Potenzmenge der Zustandsmenge)
- $I^d = \{I\}$
- $(S, a, S') \in \Delta^d$ gdw. $S' = \{q' \mid \exists q \in S : (q, a, q') \in \Delta\}$
- $F^d = \{S \subset Q \mid S \cap F \neq \emptyset\}$ T 1.2

Teil 1: endliche Wörter

└─Potenzmengenkonstruktion

-Grundbegriffe

ür isden NEA A gibt es einen DEA A^d mit $L(A^d) = L(A)$. Vir konstruieren $A^d = (Q^d, \Sigma, \Delta^d, I^d, F^d)$ wie folgt $v(S, a, S') \in \Delta^d$ gdw. $S' = \{q' \mid \exists q \in S : (q, a, q') \in \Delta\}$

Potenzmengenkonstruktion

9:11

Klären: was heißt "gleichmächtig"? (L DEA-erkennbar gdw. L NEA-erkennbar. Hinrichtung trivial.)

Hier nur kurz die Konstruktion und ein Beispiel.

Vollständiger Beweis siehe Theorie 1.

(Wir werden fast dieselbe Konstruktion später an Baumautomaten genauer ausführen.)

Potenzmengenkonstruktion

Frage: Sind DEAs und NEAs gleichmächtig?

Antwort: Ja!

Satz 1.5 (Rabin, Scott 1959)

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

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

Wir konstruieren $A^d = (Q^d, \Sigma, \Delta^d, I^d, F^d)$ wie folgt.

- $Q^d = 2^Q$ (Potenzmenge der Zustandsmenge)
- $I^d = \{I\}$
- $(S, a, S') \in \Delta^d$ gdw. $S' = \{q' \mid \exists q \in S : (q, a, q') \in \Delta\}$
- $F^d = \{S \subset Q \mid S \cap F \neq \emptyset\}$

T 1.2

Im schlimmsten Fall kann \mathcal{A}^d im Vergleich zu \mathcal{A} exponentiell viele Zustände haben (s. Hopcroft et al. 2001, S. 65). Teil 1: endliche Wörter -Grundbegriffe

└─Potenzmengenkonstruktion

Für ieden NEA A gibt es einen DEA A^d mit $L(A^d) = L(A)$. Nir konstruieren $A^d = (Q^d, \Sigma, \Delta^d, I^d, F^d)$ wie folgt • $(S, a, S') \in \Delta^d$ gdw. $S' = \{q' \mid \exists q \in S : (q, a, q') \in \Delta\}$ Im schlimmsten Fall kann Ad im Vergleich zu A xponentiell viele Zustände haben (s. Hopcroft et al. 2001, S. 65

Potenzmengenkonstruktion

9:11

Klären: was heißt "gleichmächtig"? (L DEA-erkennbar gdw. L NEA-erkennbar. Hinrichtung trivial.)

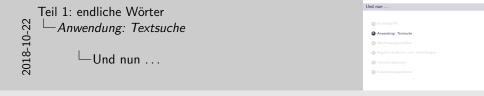
Hier nur kurz die Konstruktion und ein Beispiel.

Vollständiger Beweis siehe Theorie 1.

(Wir werden fast dieselbe Konstruktion später an Baumautomaten genauer ausführen.)

Und nun ...

- Grundbegriffe
- 2 Anwendung: Textsuche
- 3 Abschlusseigenschaften
- 4 Reguläre Ausdrücke und Anwendungen
- 5 Charakterisierungen
- 6 Entscheidungsprobleme



Stichwortsuche

Entscheidungsprobleme

-Stichwortsuche

eine Liste aller Dokumente Dr. die wr enthalten invertierte Indizes sind zeitaufwändig zu ersteller und setzen voraus, dass die D. sich nur lanesam ändern

9:15

2018-1

Typisches Problem aus dem Internetzeitalter

Gegeben sind Stichwörter $w_1, \ldots, w_n \in \Sigma^*$ und **Dokument**e $D_1, \ldots, D_M \in \Sigma^*$.

Finde alle j, so dass D_i mindestens ein (alle) w_i als Teilwort hat.

- relevant z.B. für Suchmaschinen
- übliche Technologie: invertierter Index speichert für jedes im Internet auftretende wi eine Liste aller Dokumente D_i , die w_i enthalten
- invertierte Indizes sind zeitaufwändig zu erstellen und setzen voraus, dass die D_i sich nur langsam ändern

Grundbegriffe Reguläre Ausdrücke Charakterisierungen Entscheidungsprobleme

Stichwortsuche ohne invertierte Indizes?

Invertierte Indizes versagen, wenn

- die (relevanten) Dokumente sich schnell ändern:
 - Suche in tagesaktuellen Nachrichtenartikeln
 - Einkaufshelfer sucht nach bestimmten Artikeln in aktuellen Seiten von Online-Shops
- die Dokumente nicht katalogisiert werden können:
 - Online-Shops wie Amazon generieren oft Seiten für ihre Artikel nur auf Anfragen hin.
- → Wie kann man dennoch Stichwortsuche implementieren?

Teil 1: endliche Wörter Anwendung: Textsuche Eighnofehalfer make mark bestimmten Autikal in aktuellen Seiten von Online-Shoos a die Dokumente nicht katalogisiert werden könner -Stichwortsuche ohne invertierte Indizes? · Online-Shops wie Amazon generieren oft Seiten für ihre Artike - Wie kann man dennoch Stichwortsuche implementieren

Stichwortsuche ohne invertierte Indizes?

9:17

2018-

Gegeben sind Stichwörter $w_1, \ldots, w_n \in \Sigma^*$ und Dokumente $D_1, \ldots, D_M \in \Sigma^*$.

Finde alle j, so dass D_i mindestens ein w_i als Teilwort hat.

Ziel: konstruiere NEA \mathcal{A} , der

- ein D_i zeichenweise liest und
- in einen Endzustand geht gdw. er eins der w_i findet

Der Einfachheit halber legen wir fest, dass A ein Wort w akzeptiert, wenn A bereits nach Lesen eines Teilworts einen akz. Zustand erreicht.

Teil 1: endliche Wörter -Anwendung: Textsuche

∟Ein Fall für endliche Automaten!

Finde alle i, so dass D: mindestens ein w: als Teilwort hat Ziel: konstruiere NEA ,4, der · ein D: zeichenweise liest und

Fin Fall für endliche Automaten!

und Dokumente $D_1, \dots, D_M \in \Sigma^*$.

Geeeben sind Stichwörter $w_1, \dots, w_n \in \Sigma^n$

Gegeben sind Stichwörter $w_1, \ldots, w_n \in \Sigma^*$ und Dokumente $D_1, \ldots, D_M \in \Sigma^*$.

Finde alle j, so dass D_i mindestens ein w_i als Teilwort hat.

Ziel: konstruiere NEA \mathcal{A} , der

- ein D_i zeichenweise liest und
- in einen Endzustand geht gdw. er eins der w_i findet

Der Einfachheit halber legen wir fest, dass A ein Wort w akzeptiert, wenn A bereits nach Lesen eines *Teilworts* einen akz. Zustand erreicht.

Beispiel 1.6

 $w_1 = \text{web und } w_2 = \text{ebay}$

T 1.3

Teil 1: endliche Wörter -Anwendung: Textsuche

∟Ein Fall für endliche Automaten!

Geeeben sind Stichwörter $w_1, \dots, w_n \in \Sigma^n$ und Dokumente $D_1, \dots, D_M \in \Sigma^*$. Finde alle i, so dass D: mindestens ein w: als Teilwort hat Ziel: konstruiere NEA ,4, der · ein D: zeichenweise liest und Day Finfarkhait halber legen wir fest, dass A ein Wort w akzeptiert $w_1 = web \text{ und } w_2 = ebay$

Fin Fall für endliche Automaten!

Implementation des NEAs ${\cal A}$

Eine Möglichkeit:

- Determinisierung (Potenzmengenkonstruktion)
- 2 Simulation des resultierenden DEA \mathcal{A}^d

Wird \mathcal{A}^d nicht zu groß? ($2^{27} > 134 \,\text{Mio. Zustände bei Stichw. "Binomialkoeffizient", "Polynom")}$



9:22 bis 9:30

Implementation des NEAs ${\cal A}$

Eine Möglichkeit:

- Determinisierung (Potenzmengenkonstruktion)
- 2 Simulation des resultierenden DEA A^d

Wird \mathcal{A}^d nicht zu groß?

 $(2^{27}>134\,\mathrm{Mio.}\ \mathrm{Zust\ddot{a}nde}\ \mathrm{bei}\ \mathrm{Stichw.}\ \mathrm{"Binomialkoeffizient"},\ \mathrm{"Polynom"})$

Nein,

- mit der leicht geänderten Definition von Akzeptanz
- und unserer Variante der Potenzmengenkonstruktion

wird \mathcal{A}^d genauso viele Zustände haben wie \mathcal{A} !

T 1.4

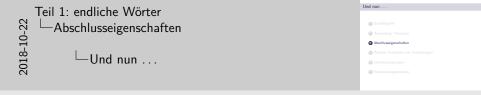
wird A^d genauso viele Zustände haben wie A!

Implementation des NEAs A

9:22 bis 9:30

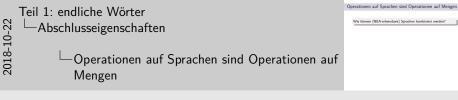
Und nun ...

- Grundbegriffe
- 2 Anwendung: Textsuche
- 3 Abschlusseigenschaften
- 4 Reguläre Ausdrücke und Anwendungen
- 5 Charakterisierunger
- 6 Entscheidungsprobleme



Operationen auf Sprachen sind Operationen auf Mengen

Wie können (NEA-erkennbare) Sprachen kombiniert werden?



9:30

Grundbegriffe

Reguläre Ausdrücke

Wie können (NEA-erkennbare) Sprachen kombiniert werden?

 Boolesche Operationen Vereinigung

$$L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ oder } w \in L_2 \}$$

Schnitt
$$L_1 \cap L_2 = \{ w \mid w \in L_1 \text{ und } w \in L_2 \}$$

Komplement
$$\overline{L} = \{ w \in \Sigma^* \mid w \notin L \}$$

Teil 1: endliche Wörter -Abschlusseigenschaften

> -Operationen auf Sprachen sind Operationen auf Mengen

Operationen auf Sprachen sind Operationen auf Mengen Wie können (NEA-erkennbare) Sprachen kombiniert werden?

9:30

Wie können (NEA-erkennbare) Sprachen kombiniert werden?

Boolesche Operationen

Vereinigung
$$L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ oder } w \in L_2 \}$$

Schnitt
$$L_1 \cap L_2 = \{ w \mid w \in L_1 \text{ und } w \in L_2 \}$$

Komplement
$$\overline{L} = \{ w \in \Sigma^* \mid w \notin L \}$$

Wortoperationen

Konkatenation
$$L_1 \cdot L_2 = \{vw \mid v \in L_1 \text{ und } w \in L_2\}$$

Kleene-Hülle
$$L^* = \bigcup_{i>0} L^i$$
,

wobei
$$L^0 = \{\varepsilon\}$$
 und $L^{i+1} = L^i \cdot L$ für alle $i \geqslant 0$

Teil 1: endliche Wörter -Abschlusseigenschaften

> -Operationen auf Sprachen sind Operationen auf Mengen

Operationen auf Sprachen sind Operationen auf Menger Wie können (NEA-erkennbare) Sprachen kombiniert werden? $L_1 \cup L_2 = \{w \mid w \in L_1 \text{ oder } w \in L_2\}$ Konkatenation $L_1 \cdot L_2 = \{vw \mid v \in L_1 \text{ und } w \in L_2\}$ Kleene-Hille $L^* = \bigcup L^i$, wobei $L^0 = \{\varepsilon\}$ und $L^{i+1} = L^i \cdot L$ für alle $i \ge 0$

9:30

Abgeschlossenheit

Die Menge der erkennbaren Sprachen heißt abgeschlossen unter . . .

- Vereiniqung, 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 $L_1 \cap L_2$.
- Konkatenation, falls gilt: Falls L_1 , L_2 erkennbar, so auch $L_1 \cdot L_2$.
- Kleene-Stern, falls gilt: Falls L erkennbar, so auch L^* .

Teil 1: endliche Wörter -Abschlusseigenschaften -Abgeschlossenheit

Abgeschlossenheit

- Die Menge der erkennbaren Sprachen heißt abgeschlossen unter · Vereinigung, falls gilt:
- Falls L1. L2 erkennbar, so auch L1 U L2 w Komplement, falls eilt:
- Falls L erkennbar, so auch \overline{L} Schnitt, falls eilt:
- Falls L_1, L_2 erkennbar, so auch $L_1 \cap L$ a Konkatenation, falls gilt:
- Falls L_1, L_2 erkennbar, so auch $L_1 \cdot L_2$ u Kleene-Stern, falls gilt: Falls L erkennbar, so auch L*

9:32

2018-10-22

Fragen: Wer weiß es noch?

Gemeinsam durchgehen & rekapitulieren:

- Vereinigungsautomat
- Produktautomat
- Det.+Vertauschen aZ/nicht-aZ
- Hintereinanderhängen
- Schleife aZ→AZ

Textsuche Abschlusseig. Reguläre Ausdrücke Charakterisierungen Entscheidungsprobleme

Abgeschlossenheit

Die Menge der erkennbaren Sprachen heißt abgeschlossen unter ...

- Vereiniqung, 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 $L_1 \cap L_2$.
- Konkatenation, falls gilt: Falls L_1 , L_2 erkennbar, so auch $L_1 \cdot L_2$.
- Kleene-Stern, falls gilt: Falls L erkennbar, so auch L^* .

Unter welchen Op. sind die NEA-erkennbaren Sprachen abgeschlossen?

Teil 1: endliche Wörter -Abschlusseigenschaften -Abgeschlossenheit

Abgeschlossenheit

Die Menge der erkennbaren Sprachen heißt abgeschlossen unter · Vereinigung, falls gilt:

Unter welchen Op, sind die NEA-erkennbaren Sprachen abgeschlossen?

- Falls L1. L2 erkennbar, so auch L1 U L2 w Komplement, falls eilt: Falls L erkennbar, so auch I
- Schnitt, falls eilt:
- Falls L_1, L_2 erkennbar, so auch $L_1 \cap L$ a Konkatenation, falls gilt:
- Falls L1. L2 erkennbar, so auch L1 L2.
- u Kleene-Stern, falls gilt: Falls L erkennbar, so auch L*

9:32

2018-1

Fragen: Wer weiß es noch?

Gemeinsam durchgehen & rekapitulieren:

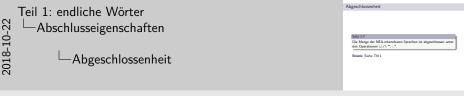
- Vereinigungsautomat
- Produktautomat
- Det.+Vertauschen aZ/nicht-aZ
- Hintereinanderhängen
- Schleife aZ→AZ

Abgeschlossenheit

Satz 1.7

Die Menge der NEA-erkennbaren Sprachen ist abgeschlossen unter den Operationen \cup , \cap , $\overline{\ }$, \cdot , *.

Beweis: Siehe Thl 1.



9:36

Und nun ...

Grundbegriffe

- Grundbegriffe
- 2 Anwendung: Textsuche
- 3 Abschlusseigenschaften
- 4 Reguläre Ausdrücke und Anwendungen
- 5 Charakterisierungen
- 6 Entscheidungsprobleme

Teil 1: endliche Wörter

Reguläre Ausdrücke und Anwendungen

Und nun ...

Reguläre Ausdrücke und Anwendungen

Reguläre Ausdrücke sind . . .

- bequeme Charakterisierung NEA-erkennbarer Sprachen
- besonders praktisch für Anwendungen



9:37

Grundbegriffe Textsuche Abschlusseig. Reguläre Ausdrücke Charakterisierungen

Reguläre Sprachen

Definition 1.8

Eine Sprache $L \subseteq \Sigma^*$ ist regulär, falls gilt:

- $L = \emptyset$ oder
- $L = \{\varepsilon\}$ oder
- $L = \{a\}, a \in \Sigma$, oder
- L lässt sich durch (endlichmaliges) Anwenden der Operatoren \cup , \cdot , * aus den vorangehenden Fällen konstruieren.

Teil 1: endliche Wörter

Reguläre Ausdrücke und *Anwendungen*Reguläre Sprachen

Reguläre Sprachen

Eine Sprache $L \subseteq \Sigma^*$ ist regulär, falls gilt: $L = \emptyset$ oder $L = \{\varepsilon\}$ oder $L = \{a\}, a \in \Sigma$, oder

L lässt sich durch (endlichmaliges) Anwenden der Operatoren
 U., * aus den vorangehenden Fällen konstruieren.

9:37

Reguläre Sprachen

Definition 1.8

Eine Sprache $L \subseteq \Sigma^*$ ist regulär, falls gilt:

- $L = \emptyset$ oder
- $L = \{\varepsilon\}$ oder
- $L = \{a\}, a \in \Sigma$, oder
- L lässt sich durch (endlichmaliges) Anwenden der Operatoren ∪, •, * aus den vorangehenden Fällen konstruieren.

Beispiele:

$$(\{a\} \cup \{b\})^* \cdot \{a\} \cdot \{b\}$$
 (siehe A_3 auf Folie 10)
 $\{b\}^* \cdot \{a\} \cdot \{a\}^* \cdot \{b\} \cdot (\{a\} \cup \{b\})^*$ (s. A_2 auf Folie 10)

Reguläre Sprachen Teil 1: endliche Wörter Reguläre Ausdrücke und Anwendungen Eine Sprache $L \subseteq \Sigma^*$ ist regulär, falls gilt: v L = 0 oder • $L = \{\varepsilon\}$ oder $a \ L = \{a\}, \ a \in \Sigma, \text{ oder }$ u L lässt sich durch (endlichmaliges) Anwenden der Operatoren U. . . * aus den vorangehenden Fällen konstruierer Reguläre Sprachen

 $(\{a\} \cup \{b\})^* \cdot \{a\} \cdot \{b\}$ (siehe A_3 auf Folie 10)

 $\{b\}^* \cdot \{a\} \cdot \{a\}^* \cdot \{b\} \cdot (\{a\} \cup \{b\})^*$ (s. As auf Folie 10)

9:37

Definition 1.9

Ein regulärer Ausdruck (RA) r über Σ und die zugehörige Sprache $L(r) \subset \Sigma^*$ werden induktiv wie folgt definiert.

 \bullet $r = \emptyset$

ist ein RA mit $L(r) = \emptyset$

 \bullet $r = \varepsilon$

- ist ein RA mit $L(r) = \{\varepsilon\}$
- r = a, für $a \in \Sigma$, ist ein RA mit $L(r) = \{a\}$
- $r = (r_1 + r_2)$
- ist ein RA mit $L(r) = L(r_1) \cup L(r_2)$
- $r = (r_1 r_2)$
- ist ein RA mit $L(r) = L(r_1) \cdot L(r_2)$

• $r = (r_1)^*$

ist ein RA mit $L(r^*) = (L(r))^*$

Teil 1: endliche Wörter Reguläre Ausdrücke und Anwendungen

Reguläre Ausdrücke

9:39

Reguläre Ausdrücke ist ein RA mit $L(r) = L(r_1) \cdot L(r_2)$ ist ein RA mit $L(r^*) = (L(r))^*$

Definition 1.9

Ein regulärer Ausdruck (RA) r über Σ und die zugehörige Sprache $L(r) \subset \Sigma^*$ werden induktiv wie folgt definiert.

$$\bullet$$
 $r = \emptyset$

ist ein RA mit
$$L(r) = \emptyset$$

$$\bullet$$
 $r = \varepsilon$

ist ein RA mit
$$L(r) = \{\varepsilon\}$$

•
$$r = a$$
, für $a \in \Sigma$, ist ein RA mit $L(r) = \{a\}$

•
$$r = (r_1 + r_2)$$
 ist ein RA r

ist ein RA mit
$$L(r) = L(r_1) \cup L(r_2)$$

•
$$r = (r_1 r_2)$$
 ist ein RA mi

ist ein RA mit
$$L(r) = L(r_1) \cdot L(r_2)$$

•
$$r = (r_1)^*$$

ist ein RA mit
$$L(r^*) = (L(r))^*$$

Beispiele: (wir lassen Klammern weg soweit eindeutig)

$$(a + b)^*ab$$
 (siehe A_3 auf Folie 10)

$$b^*aa^*b(a+b)^*$$
 (siehe A_2 auf Folie 10)

Teil 1: endliche Wörter Reguläre Ausdrücke und Anwendungen Reguläre Ausdrücke

Ein regulärer Ausdruck (RA) r über Σ und die zugehörige Sprache ist ein RA mit $L(r) = \{\varepsilon\}$ ist ein RA mit $L(r) = L(r_1) \cdot L(r_2)$ ist ein RA mit $L(r^*) = (L(r))^*$ assen Klammern weg soweit eindeutig)

(a + b)*ab (siehe A1 auf Folie 10)

 $b^*aa^*b(a+b)^*$ (siehe A_2 auf Folie 10)

Reguläre Ausdrücke

9:39

Reguläre und NEA-erkennbare Sprachen

Satz 1.10 (Kleene 1956)

Sei $L \subseteq \Sigma^*$ eine Sprache.

- **1** L ist regulär gdw. es einen RA r gibt mit L = L(r).
- ② L ist regulär gdw. L NEA-erkennbar ist.

Teil 1: endliche Wörter

Reguläre Ausdrücke und Anwendungen

Reguläre und NEA-erkennbare Sprachen

Sei $L \subseteq \Sigma^*$ eine Sprache. \bullet L ist regulär gdw. es einen RA r gibt mit L = L(r). \bullet L ist regulär gdw. L NEA-erkennbar ist.

Reguläre und NEA-erkennbare Sprachen

9:41

Reguläre und NEA-erkennbare Sprachen

Satz 1.10 (Kleene 1956)

Sei $L \subseteq \Sigma^*$ eine Sprache.

- **1** L ist regulär gdw. es einen RA r gibt mit L = L(r).
- ② L ist regulär gdw. L NEA-erkennbar ist.

Beweis.

- Folgt offensichtlich aus Def. 1.8, 1.9.
- Benutze Punkt 1.

": Induktion über Aufbau von r.

IA: gib Automaten an, die \emptyset , $\{\varepsilon\}$, $\{a\}$ erkennen. IS: benutze Abschlusseigenschaften (Satz 1.7)

..←": siehe Theoretische Informatik 1.

Teil 1: endliche Wörter

Reguläre Ausdrücke und Anwendungen

Reguläre und NEA-erkennbare Sprachen

Regulare und NEA-erkennbare Sprachen

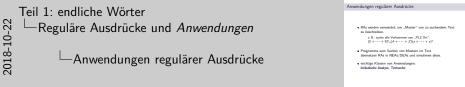
| Sizi 1.12 (Neuva Spachs) |
| Sizi 1.2 ⊆ Y view Spachs |
| ◆ L in regular gibe ... It NEA-erkenbar int. l
| ◆ L in regular gibe ... It NEA-erkenbar int. l
| Brown. |
| Brown |
| We will be a bound |
| We will be a boun

9:41

• RAs werden verwendet, um "Muster" von zu suchendem Text zu beschreiben.

z. B.: suche alle Vorkommen von "PLZ Ort":
$$(0 + \cdots + 9)^5 \sqcup (A + \cdots + Z)(a + \cdots + z)^*$$

- Programme zum Suchen von Mustern im Text übersetzen RAs in NEAs/DEAs und simulieren diese.
- wichtige Klassen von Anwendungen: lexikalische Analyse, Textsuche



```
8:30 9:43 bis 9:44, 1 min Reserve ↑ Zeitplanung ist Quatsch; die Sitzung war von 12:15 bis 13:45.
```

Ankündigen: Terminfindung (Mo. 12–14 klappt bei 2 TN nicht). In Pause.

Grundbegriffe Reguläre Ausdrücke Charakterisierungen Entscheidungsprobleme Abschlusseig.

Komfortablere Syntax regulärer Ausdrücke

- UNIX und andere Anwendungen erweitern Syntax von RAs
- Hier: nur "syntaktischer Zucker" die Erweiterungen, die nicht aus den regulären Sprachen herausführen

Teil 1: endliche Wörter Reguläre Ausdrücke und Anwendungen

Komfortablere Syntax regulärer Ausdrücke

8:32

2018-1

u Hier: nur "syntaktischer Zucker" - die Erweiterungen

Reguläre Ausdrücke

Charakterisierungen

Entscheidungsprobleme

Komfortablere Syntax regulärer Ausdrücke

- UNIX und andere Anwendungen erweitern Syntax von RAs
- Hier: nur "syntaktischer Zucker" die Erweiterungen, die nicht aus den regulären Sprachen herausführen
- Alphabet Σ: alle ASCII-Zeichen
- RA . mit $L(.) = \Sigma$
- RA $[a_1 a_2 \dots a_k]$, Abkürzung für $a_1 + a_2 + \dots + a_k$
- RAs für Bereiche: z.B. [a-z0-9]. Abkü. für [ab...z01...9]
- Operator | anstelle +
- Operator ?: r? steht für $\varepsilon + r$
- Operator +: r+ steht für rr*
- Operator {n}: r{5} steht für rrrrr
- Klammern und * wie gehabt

Teil 1: endliche Wörter Reguläre Ausdrücke und Anwendungen

Komfortablere Syntax regulärer Ausdrücke

Komfortablere Syntax regulärer Ausdrücke u Hier: nur "syntaktischer Zucker" - die Erweiterunge die nicht aus den regulären Sprachen berausführen

UNIX und andere Anwendungen erweitern Syntax von RAs

« Alphabet Σ: alle ASCII-Zeichen

 \mathbf{v} RA . mit $L(.) = \Sigma$

• RA $[a_1a_2...a_k]$, Abkürzung für $a_1 + a_2 + \cdots + a_k$

Operator ?: r? steht für ε + r

 Operator +: r+ steht für rr* u Operator (n): r(5) steht für mmr

8:32

Reguläre Ausdrücke

Charakterisierungen

Entscheidungsprobleme

Komfortablere Syntax regulärer Ausdrücke

- UNIX und andere Anwendungen erweitern Syntax von RAs
- Hier: nur "syntaktischer Zucker" die Erweiterungen, die nicht aus den regulären Sprachen herausführen
- Alphabet Σ: alle ASCII-Zeichen
- RA . mit $L(.) = \Sigma$
- RA $[a_1 a_2 \dots a_k]$, Abkürzung für $a_1 + a_2 + \dots + a_k$
- RAs für Bereiche: z.B. [a-z0-9]. Abkü. für [ab...z01...9]
- Operator | anstelle +
- Operator ?: r? steht für $\varepsilon + r$
- Operator +: r+ steht für rr*
- Operator {n}: r{5} steht für rrrrr

Klammern und * wie gehabt

PLZ-Ort-Beispiel: $[0-9]{5}_{\sqcup}[A-Z][a-z]*$

Komfortablere Syntax regulärer Ausdrücke

Teil 1: endliche Wörter Reguläre Ausdrücke und Anwendungen

Komfortablere Syntax regulärer Ausdrücke

u Alphabet Σ: alle ASCII-Zeichen • RA $[a_1a_2...a_k]$, Abkürzung für $a_1 + a_2 + \cdots + a_k$

Operator ?: r? steht für ε + r Operator +: r+ steht für rr* u Operator (n): r(5) steht für mm

UNIX und andere Anwendungen erweitern Syntax von RA

u Hier: nur "syntaktischer Zucker" - die Erweiterunge

die nicht aus den regulären Sprachen berausführen

8:32

Anwendung: lexikalische Analyse

- Lexer (auch: Tokenizer) durchsucht Quellcode nach Token: zusammengehörende Zeichenfolgen, z.B. Kennwörter, Bezeichner
- Ausgabe des Lexers: Token-Liste, wird an Parser weitergegeben
- Mit RAs: Lexer leicht programmier- und modifizierbar



8:35

Lexer: kurz für "lexikalischer Scanner", auch Tokenizer

Lex: a computer program that generates lexical analyzers" [Wikipedia]

Flex: "fast lexical analyzer generator", "a free and open-source software alternative to lex" [Wikipedia]

- Lexer (auch: Tokenizer) durchsucht Quellcode nach Token: zusammengehörende Zeichenfolgen, z.B. Kennwörter, Bezeichner
- Ausgabe des Lexers: Token-Liste, wird an Parser weitergegeben
- Mit RAs: Lexer leicht programmier- und modifizierbar
- UNIX-Kommandos lex und flex generieren Lexer
 - Eingabe: Liste von Einträgen RA + Code
 - Code beschreibt Ausgabe des Lexers für das jeweilige Token
 - generierter Lexer wandelt alle RAs in einen DEA um, um Vorkommen der Tokens zu finden (siehe Folie 17)
 - anhand des Zustands des DEAs lässt sich bestimmen. welches Token gefunden wurde

-Anwendung: lexikalische Analyse

Anwendung: lexikalische Analyse

welches Token gefunden wurde

- UNIX-Kommandos lex und flex generieren Lexe Eingabe: Liste von Einträgen RA + Code
- Code beschreibt Ausgabe des Lexers für das jeweilige Token . generierter Lexer wandelt alle RAs in einen DEA un
- um Vorkommen der Tokens zu finden (siehe Folie 17) anhand des Zustands des DEAs lässt sich bestimmen.

8:35

2018-1

kurz für "lexikalischer Scanner", auch Tokenizer

Lex: a computer program that generates lexical analyzers [Wikipedia]

Flex: "fast lexical analyzer generator", "a free and open-source software alternative to lex" [Wikipedia]

Beispieleingabe für lex

```
else
  {return(ELSE);}
[A-Za-z][A-za-z0-9]*
  {<Trage gefundenen Bezeichner in Symboltabelle ein>;
  return(ID);}
>=
  {return(GE):}
  {return(EQ);}
```

(Lexer-Generator muss Prioritäten beachten: else wird auch vom 2. RA erkannt, ist aber reserviert)

Teil 1: endliche Wörter

Reguläre Ausdrücke und Anwendungen

Beispieleingabe für 1ex

Beispieleingabe für 1ex

(extra(EE))
(crtage gefandenen bezeichner in Synbolitabelle sitt)
(crtage gefandenen bezeichner in Synbolitabelle sitt)
(crtage (EE))
(crtage (EE))
(crtage (EE))
(crtage (EE))
(crtage (EE))
(crtage (EE))

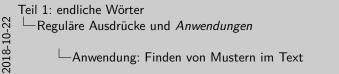
8:37

ZF: Der Lexer-Generator dient dazu, Lexer zu erzeugen. Diese Tabelle hier gibt an, wie das passiert. Der Vorteil ist die leichte Änderbarkeit, wenn sich mal ein Token oder dessen Beschreibung (RA!) ändert.

Anwendung: Finden von Mustern im Text

Beispiel: Suchen von Adressen (Str. + Hausnr.) in Webseiten Solche Angaben sollen gefunden werden:

Parkstraße 5 Enrique-Schmidt-Straße 12a Breitenweg 24A Knochenhauergasse 30-32



Anwendung: Finden von Mustern im Text

Solche Angaben sollen gefunden werden

Enrique-Schmidt-Straße 12a Breitenweg 24A Knochenhauergasse 30-32

8:40

Solche Angaben sollen gefunden werden:

Parkstraße 5 Enrique-Schmidt-Straße 12a Breitenweg 24A Knochenhauergasse 30-32

aber auch solche:

```
Straße des 17. Juni 17 ...boulevard, ...allee, ...platz, ...
Postfach 330 440
```

Am Wall 8

Teil 1: endliche Wörter
Reguläre Ausdrücke und *Anwendungen*Anwendung: Finden von Mustern im Text

Anwendung: Finden von Mustern im Text

Solche Angaben sollen gefunden werden

Straße dez 17. Juni 17

Postfach 330 440 Am Wall 8

.boulevard ... allee ... platz

Enrique-Schmidt-Straße 12a Breitenweg 24A Knochenhauergasse 30-32

8:40

Anwendung: Finden von Mustern im Text

Beispiel: Suchen von Adressen (Str. + Hausnr.) in Webseiten Solche Angaben sollen gefunden werden:

Parkstraße 5 Enrique-Schmidt-Straße 12a Breitenweg 24A Knochenhauergasse 30-32

aber auch solche:

Straße des 17. Juni 17 ...boulevard, ...allee, ...platz, ... Postfach 330 440

Am Wall 8

- → Ausmaß der Variationen erst während der Suche deutlich.
- → Gesucht: einfach modifizierbare Beschreibung der Muster

Teil 1: endliche Wörter Solche Angaben sollen gefunden werden Reguläre Ausdrücke und Anwendungen Breitenweg 24A Knochenhauergasse 30-32 aber auch solche 2018-1 -Anwendung: Finden von Mustern im Text -- Ausmaß der Variationen erst während der Suche deutlich

~ Gesucht: einfach modifizierbare Beschreibung der Muste

Anwendung: Finden von Mustern im Text

Enrique-Schmidt-Straße 12a

Straße dez 17. Juni 17

Am Wall 8

.boulevard. ... allee. ... platz.

8:40

Mustersuche mit regulären Ausdrücken

Mögliches Vorgehen:

- (1) Beschreibung des Musters mit einem einfachen RA
- (2) Umwandlung des RA in einen NEA
- (3) Implementation des DEA wie auf Folie 17+18
- (4) Test
- (5) Wenn nötig, RA erweitern/ändern und Sprung zu Schritt 2



8:42

Dies ist ein ganz banales, heuristisches Vorgehen. Keine tiefgründigen Techniken. Dient nur zur Demonstration der einfachen Erweiterbarkeit, wenn man RAs benutzt.

So kann sich der RA entwickeln:

Grundbegriffe

• Vorkommen von "straße" etc.:1 straße|str\.|weg|gasse

- Teil 1: endliche Wörter Reguläre Ausdrücke und Anwendungen
 - Adresssuche mit regulären Ausdrücken

Adresssuche mit regulären Ausdrücken

¹Weil der UNIX-RA . für Σ reserviert ist, steht \. für {.}

8:43 bis 8:46

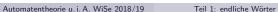
2018-10-22

So kann sich der RA entwickeln:

Grundbegriffe

- Vorkommen von ..straße" etc.:1 straße|str\.|weg|gasse
- Plus Name der Straße und Hausnummer:





¹Weil der UNIX-RA . für Σ reserviert ist, steht \. für $\{.\}$

Adresssuche mit regulären Ausdrücken

¹Weil der UNIX-RA . für Σ reserviert ist, steht \. für {.}

8:43 bis 8:46

2018-1

Teil 1: endliche Wörter

-Reguläre Ausdrücke und Anwendungen

Adresssuche mit regulären Ausdrücken

So kann sich der RA entwickeln:

- Vorkommen von "straße" etc.:¹ straße|str\.|weg|gasse
- Plus Name der Straße und Hausnummer:

 $[A-Z][a-z]*(straße|str\.|weg|gasse) [0-9]*$

 1 Weil der UNIX-RA . für Σ reserviert ist, steht \. für {.}

Adresssuche mit regulären Ausdrücken

 Vorkommen von "straße" etc.:¹ straße|str\.|weg|gasse

• Plus Name der Straße und Hausnummer: [A-Z] [a-z]*(straße|str\.|weg|gasse) [0-9]*

¹Weil der UNIX-RA . für Σ reserviert ist, steht \. für {.}

8:43 bis 8:46

2018-1

Teil 1: endliche Wörter

Reguläre Ausdrücke und Anwendungen

-Adresssuche mit regulären Ausdrücken

Automatentheorie u. i. A. WiSe 2018/19

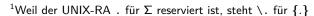
Teil 1: endliche Wörter

Reguläre Ausdrücke Abschlusseig. Charakterisierungen Entscheidungsprobleme

Adresssuche mit regulären Ausdrücken

So kann sich der RA entwickeln:

- Vorkommen von "straße" etc.:¹ straße|str\.|weg|gasse
- Plus Name der Straße und Hausnummer: $[A-Z][a-z]*(straße|str\.|weg|gasse) [0-9]*$
- Hausnummern mit Buchstaben (12a), -bereiche (30–32):



Teil 1: endliche Wörter Reguläre Ausdrücke und Anwendungen

-Adresssuche mit regulären Ausdrücken

[A-Z] [a-z]*(strafe|str\.|weg|gasse) [0-9]* u Hausnummern mit Buchstaben (12a), -bereiche (30-32)

Adresssuche mit regulären Ausdrücken

¹Weil der UNIX-RA . für Σ reserviert ist, steht \. für {.}

8:43 bis 8:46

Abschlusseig. Reguläre Ausdrücke Charakterisierungen Entscheidungsprobleme

Adresssuche mit regulären Ausdrücken

So kann sich der RA entwickeln:

Grundbegriffe

- Vorkommen von "straße" etc.:¹ straße|str\.|weg|gasse
- Plus Name der Straße und Hausnummer: $[A-Z][a-z]*(straße|str\.|weg|gasse) [0-9]*$
- Hausnummern mit Buchstaben (12a), -bereiche (30–32): [A-Z] [a-z] * (straße | str\. | weg | gasse) ([0-9]*[A-Za-z]?-)?[0-9]*[A-Za-z]?

¹Weil der UNIX-RA . für Σ reserviert ist, steht \. für $\{.\}$

Adresssuche mit regulären Ausdrücken

- [A-Z] [a-z]*(strafe|str\.|weg|gasse) [0-9]*
 - u Hausnummern mit Buchstaben (12a), -bereiche (30-32 ([0-9]*[A-Za-z]?-)?[0-9]*[A-Za-z]1

¹Weil der UNIX-RA . für Σ reserviert ist, steht \. für {.}

8:43 bis 8:46

2018-1

Teil 1: endliche Wörter

Reguläre Ausdrücke und Anwendungen

-Adresssuche mit regulären Ausdrücken

So kann sich der RA entwickeln:

- Vorkommen von "straße" etc.:1 straße|str\.|weg|gasse
- Plus Name der Straße und Hausnummer:
 [A-Z] [a-z]*(straße|str\.|weg|gasse) [0-9]*
- Hausnummern mit Buchstaben (12a), -bereiche (30–32):

 [A-Z] [a-z]*(straße|str\.|weg|gasse)

 ([0-9]*[A-Za-z]?-)?[0-9]*[A-Za-z]?
- und mehr:

Grundbegriffe

- Straßennamen mit Bindestrichen
- "Straße" etc. am Anfang
- Plätze, Boulevards, Alleen etc.
- Postfächer
- . . .

 $^1\mbox{Weil}$ der UNIX-RA . für Σ reserviert ist, steht \ . für $\{.\}$

.

Teil 1: endliche Wörter
Reguläre Ausdrücke und Anwendungen

–Adresssuche mit regulären Ausdrücken

Adversanche mit regulation Ausdrücken

• Vorkenmen von "stable" (st.2.

• Extra leitert, Vollegtunse

• The Name der Straße und destamment

[La2 (La-2) / Lage (Lage ausgrand) (D-2)

• Hausenmens mit Beschäusen (Da), Senicke (D-2)

[La2 (La-2) / La-2)

• Lage (La-2) / La-2 / Lage ausgrand

• Lage (La-2) / Lage ausgrand

• Lage (La-2) / Lage ausgrand

• Lage (La-2) / Lage ausgrand

• Lage ausgran

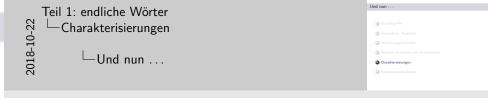
¹Weil der UNIX-RA . für Σ reserviert ist, steht \. für {.}

8:43 bis 8:46

Und nun ...

Grundbegriffe

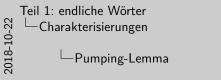
- Grundbegriffe
- 2 Anwendung: Textsuche
- 3 Abschlusseigenschaften
- Reguläre Ausdrücke und Anwendungen
- 5 Charakterisierungen
- 6 Entscheidungsprobleme



Pumping-Lemma

Grundbegriffe

Wie zeigt man, dass L nicht NEA-erkennbar (regulär) ist?



Wie zeigt man, dass L nicht NEA-erkennbar (regulär) ist?

Pumping-Lemma

8:46

Fragen: Wer kann sich an die 2 Werkzeuge aus Thl 1 erinnern?

Am Ende: Fragebogen F. 2a

Pumping-Lemma

Wie zeigt man, dass L nicht NEA-erkennbar (regulär) ist?

Satz 1.11 (Pumping-Lemma)

Sei *L* eine NEA-erkennbare Sprache.

Dann gibt es eine Konstante $p \ge 0$, so dass für alle Wörter $w \in L$ mit $|w| \ge p$ gilt:

Es gibt eine Zerlegung w = xyz mit $y \neq \varepsilon$ und $|xy| \leqslant p$, so dass $xy^iz \in L$ für alle $i \geqslant 0$.

Beweis: siehe Thl 1.

Teil 1: endliche Wörter

We nig man, das £ didt NEA orbandur (ngslår) ist?

We nig man, das £ didt NEA orbandur (ngslår) ist?

Bitt 1 (Pemping Lemma)

Set å ein NEA detenision Synoth.
Dang file an ein Nematik sy ja // ja // ji /

8:46

2018-1

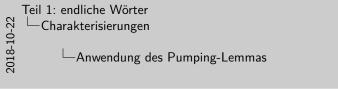
Fragen: Wer kann sich an die 2 Werkzeuge aus Thl 1 erinnern?

Am Ende: Fragebogen F. 2a

Benutzen Kontraposition:

Wenn es für alle Konstanten $p \ge 0$ ein Wort $w \in L$ mit $|w| \ge p$ gibt, so dass es **für alle** Zerlegungen w = xyz mit $y \neq \varepsilon$ und $|xy| \leqslant p$ ein $i \ge 0$ qibt mit $xy^iz \notin L$, dann ist L keine NEA-erkennbare Sprache.

T 1.5



ein Wort $w \in L$ mit $|w| \ge p$ gibt, so dass es ein $i \ge 0$ gibt mit $xy^iz \notin L$, dann ist L keine NEA-erkennbare Sprache

Anwendung des Pumping-Lemmas

8:48 bis 8:55

Bemerkungen zum Pumping-Lemma

Die Bedingung in Satz 1.11 ist ...

- notwendig dafür, dass L NEA-erkennbar ist
- nicht hinreichend Bsp.: $\{a^n b^k c^k \mid n, k \ge 1\} \cup \{b^n c^k \mid n, k \ge 0\}$

→ Pumping-L. nur zum Widerlegen von Erkennbarkeit verwendbar, nicht zum Beweisen, dass L regulär ist

(Notwendige und hinreichende Variante: Jaffes Pumping-Lemma)

Bemerkungen zum Pumping-Lemma Teil 1: endliche Wörter Charakterisierungen Die Bedingung in Satz 1.11 ist . Bsp.: $\{a^{\alpha}b^{k}c^{k} \mid n, k \ge 1\} \cup \{b^{\alpha}c^{k} \mid n, k \ge 0\}$ ~ Pumping-L. nur zum Widerlegen von Erkennbarkeit verwendbar, -Bemerkungen zum Pumping-Lemma (Notwendige und hinreichende Variante: Jaffes Pumping-Lemma)

8:55

2018-1

Grundbegriffe

Reguläre Ausdrücke Abschlusseig.

Charakterisierungen

Der Satz von Myhill-Nerode

Ziel: notwendige und hinreichende Bedingung für Erkennbarkeit

Definition 1.12

Sei $L \subseteq \Sigma^*$ eine Sprache.

Zwei Wörter $u, v \in \Sigma^*$ sind L-äquivalent (Schreibweise: $u \sim_I v$), wenn für alle $w \in \Sigma^*$ gilt:

 $uw \in L$ genau dann, wenn $vw \in L$

2018-1

Teil 1: endliche Wörter Charakterisierungen

☐ Der Satz von Myhill-Nerode

Der Satz von Myhill-Nerode Sei $L \subseteq \Sigma^*$ eine Sprache. Zwei Wörter $u, v \in \Sigma^*$ sind L-Jaulyalent (Schreibweise: $u \sim_I v$). wenn für alle $w \in \Sigma^*$ gilt: $uw \in L$ genau dann, wenn $vw \in L$

8:57

Satz von Myhill-Nerode liefert eine "echte" Charakterisierung der erkennbaren Sprachen (und eine weitere nützliche Information)

Reguläre Ausdrücke

Charakterisierungen

Der Satz von Myhill-Nerode

Abschlusseig.

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

Definition 1.12

Sei $L \subseteq \Sigma^*$ eine Sprache.

Zwei Wörter $u, v \in \Sigma^*$ sind *L*-äquivalent (Schreibweise: $u \sim_L v$), wenn für alle $w \in \Sigma^*$ gilt:

 $uw \in L$ genau dann, wenn $vw \in L$

~, heißt Nerode-Rechtskongruenz und ist Äguivalenzrelation (Reflexivität, Symmetrie, Transitivität sind offensichtlich)

Index von \sim_I : Anzahl der Äguivalenzklassen

2018-1

Teil 1: endliche Wörter Charakterisierungen

☐ Der Satz von Myhill-Nerode



8:57

Satz von Myhill-Nerode liefert eine "echte" Charakterisierung der erkennbaren Sprachen (und eine weitere nützliche Information)

Der Satz von Myhill-Nerode

Satz 1.13 (Myhill-Nerode)

 $L \subset \Sigma^*$ is NEA-erkennbar gdw. \sim_L endlichen Index hat.

Beweis: siehe Thl 1. T1.6

Teil 1: endliche Wörter
Charakterisierungen
Der Satz von Myhill-Nerode

Der Satz vom Myhill-Nerode

Satz 121 (Myhi-Nerode)

L. C. T. in NEA-standar glis — e melichen Index hat.

Brank: siehe TM 1.

T1.6

8:59

Fragebogen F. 2b

13 min Pause: Terminfindung \sim 9:15 (Ist aber eigentlich zu zeitig für Pause)

Grundbegriffe Textsuche Abschlusseig.

Reguläre Ausdrücke

Charakterisierungen

Der Satz von Myhill-Nerode

Satz 1.13 (Myhill-Nerode)

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

Beweis: siehe Thl 1. T 1.6

Interessantes "Nebenprodukt" des Beweises:

Endlicher Index n von \sim_I

= minimale Anzahl von Zuständen in einem DEA, der L erkennt.

Teil 1: endliche Wörter Charakterisierungen 2018-1 ☐ Der Satz von Myhill-Nerode Der Satz von Myhill-Nerode

8:59

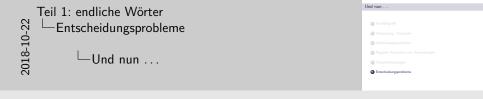
Fragebogen F. 2b

13 min Pause: Terminfindung \rightarrow 9:15 (Ist aber eigentlich zu zeitig für Pause)

Und nun ...

Grundbegriffe

- Grundbegriffe
- 2 Anwendung: Textsuche
- Abschlusseigenschaften
- 4 Reguläre Ausdrücke und Anwendungen
- 5 Charakterisierunger
- 6 Entscheidungsprobleme



Entscheidbarkeit

Grundbegriffe

(Entscheidungs-)Problem

- ... ist eine Teilmenge $X \subseteq M$
 - Eingabe: $m \in M$; Frage: $m \in X$?
 - $m \in X$: Ja-Instanzen; $m \in M \setminus X$: Nein-Instanzen



9:15

Entscheidungsprobleme sind zentral für diese Vorlesung, denn wir werden sehen, dass algorithmische Probleme in Anwendungen auf Entscheidungsprobleme gewisser Automatenmodelle zurückzuführen sind.

Z.B.: Validierung eines XML-Dokuments entspricht dem Wortproblem für gewisse Baumautomaten.

Durch das Studium dieser Entscheidungsprobleme bekommen wir also einen prinzipiellen Ansatz zur Lösung der Anwendungsprobleme.

Zunächst ein kurzer Abriss von Entscheidbarkeit und Komplexität; mehr dazu im ThI2-Skript.

"Algorithmus": Prog.sprache & Rechnermodell sind relativ unerheblich ((erweiterte) Church-Turing-These)

Entscheidbarkeit

(Entscheidungs-)Problem

- ... ist eine Teilmenge $X \subseteq M$
 - Eingabe: $m \in M$; Frage: $m \in X$?
 - $m \in X$: Ja-Instanzen; $m \in M \setminus X$: Nein-Instanzen
- Beispiele:
 - $X = \text{Menge aller Primzahlen}, M = \mathbb{N}$
 - $X = \text{Menge aller NEAs } \mathcal{A} \text{ mit } L(\mathcal{A}) \neq \emptyset$, M = Menge aller NEAs



9:15

Entscheidungsprobleme sind zentral für diese Vorlesung, denn wir werden sehen, dass algorithmische Probleme in Anwendungen auf Entscheidungsprobleme gewisser Automatenmodelle zurückzuführen sind.

Z. B.: Validierung eines XML-Dokuments entspricht dem Wortproblem für gewisse Baumautomaten.

Durch das Studium dieser Entscheidungsprobleme bekommen wir also einen prinzipiellen Ansatz zur Lösung der Anwendungsprobleme.

Zunächst ein kurzer Abriss von Entscheidbarkeit und Komplexität; mehr dazu im ThI2-Skript.

"Algorithmus": Prog.sprache & Rechnermodell sind relativ unerheblich ((erweiterte) Church-Turing-These)

(Entscheidungs-)Problem

- ... ist eine Teilmenge $X \subseteq M$
 - Eingabe: $m \in M$; Frage: $m \in X$?
 - $m \in X$: Ja-Instanzen; $m \in M \setminus X$: Nein-Instanzen
- Beispiele:
 - $X = \text{Menge aller Primzahlen}, M = \mathbb{N}$
 - $X = \text{Menge aller NEAs } A \text{ mit } L(A) \neq \emptyset$. M = Menge aller NEAs
- man stelle sich eine Blackbox vor:

$$m \in M \xrightarrow{\mathsf{Eingabe}} m \in X ? \xrightarrow{\mathsf{Ausgabe}} \sqrt{\text{"ja"}} \Rightarrow m \in X$$



9:15

Entscheidungsprobleme sind zentral für diese Vorlesung, denn wir werden sehen, dass algorithmische Probleme in Anwendungen auf Entscheidungsprobleme gewisser Automatenmodelle zurückzuführen sind.

Z. B.: Validierung eines XML-Dokuments entspricht dem Wortproblem für gewisse Baumautomaten.

Durch das Studium dieser Entscheidungsprobleme bekommen wir also einen prinzipiellen Ansatz zur Lösung der Anwendungsprobleme.

Zunächst ein kurzer Abriss von Entscheidbarkeit und Komplexität; mehr dazu im ThI2-Skript.

"Algorithmus": Prog.sprache & Rechnermodell sind relativ unerheblich ((erweiterte) Church-Turing-These)

2018-1

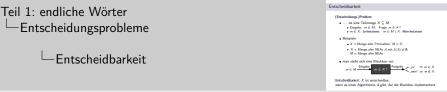
Grundbegriffe

(Entscheidungs-)Problem

- ... ist eine Teilmenge $X \subseteq M$
 - Eingabe: $m \in M$; Frage: $m \in X$?
 - $m \in X$: Ja-Instanzen; $m \in M \setminus X$: Nein-Instanzen
- Beispiele:
 - $X = \text{Menge aller Primzahlen}, M = \mathbb{N}$
 - $X = \text{Menge aller NEAs } A \text{ mit } L(A) \neq \emptyset$. M = Menge aller NEAs
- man stelle sich eine Blackbox vor:

$$m \in M \xrightarrow{\mathsf{Eingabe}} m \in X ? \xrightarrow{\mathsf{Ausgabe}} \sqrt{\text{"ja"}} \Rightarrow m \in X$$
 $m \in M \xrightarrow{\mathsf{min"}} m \notin X$

Entscheidbarkeit: X ist entscheidbar, wenn es einen Algorithmus A gibt, der die Blackbox implementiert.



9:15

Entscheidungsprobleme sind zentral für diese Vorlesung, denn wir werden sehen, dass algorithmische Probleme in Anwendungen auf Entscheidungsprobleme gewisser Automatenmodelle zurückzuführen sind.

Z. B.: Validierung eines XML-Dokuments entspricht dem Wortproblem für gewisse Baumautomaten.

Durch das Studium dieser Entscheidungsprobleme bekommen wir also einen prinzipiellen Ansatz zur Lösung der Anwendungsprobleme.

Zunächst ein kurzer Abriss von Entscheidbarkeit und Komplexität; mehr dazu im ThI2-Skript.

"Algorithmus": Prog.sprache & Rechnermodell sind relativ unerheblich ((erweiterte) Church-Turing-These)

Komplexität



Komplexität:

zusätzliche Anforderungen an Zeit-/Speicherplatzbedarf von A



9:18

Komplexität



Komplexität:

zusätzliche Anforderungen an Zeit-/Speicherplatzbedarf von A

• Polynomialzeit: Anzahl Rechenschritte von A ist $\leq |m|^k$, |m|: Länge der Eingabe; k: beliebige Konstante



9:18

Komplexität



Komplexität:

zusätzliche Anforderungen an Zeit-/Speicherplatzbedarf von A

- Polynomialzeit: Anzahl Rechenschritte von A ist $\leq |m|^k$, |m|: Länge der Eingabe; k: beliebige Konstante
- Polynomieller Platz: von A benötigter Speicherplatz $\leq |m|^k$

9:18

2018-10-22

Komplexität



Komplexität:

zusätzliche Anforderungen an Zeit-/Speicherplatzbedarf von A

- Polynomialzeit: Anzahl Rechenschritte von A ist $\leq |m|^k$, |m|: Länge der Eingabe; k: beliebige Konstante
- Polynomieller Platz: von A benötigter Speicherplatz $\leq |m|^k$
- Exponentialzeit: Anzahl Rechenschritte von A ist $\leq 2^{|m|^k}$

Teil 1: endliche Wörter
Entscheidungsprobleme
Komplexität

 $m \in M$ $\stackrel{\text{Eingabe}}{=} m \in X$ $\stackrel{\text{Aungabe}}{=} (m \in X)$ $\stackrel{\text{Aungabe}}$

Polynomialzeit: Anzahl Rechenschritte von A ist ≤ |m|^k,
 |m| : L\(\text{linge der Eingabe}\); \(k : \text{beliebige Konstante}\)

 \mathbf{v} Polynomieller Platz: von A benötigter Speicherplatz $\leq |m|^2$

a Exponentialzeit: Anzahl Rechenschritte von A ist $\leqslant 2^{|w|}$

Komplexităt

9:18

2018-10-22

Entscheidungsprobleme

Francostialzeit: Anzahl Rachenschritte von 4 ist < 200

Komplexität

Ausgabe \langle ",ja" $\Rightarrow m \in X$ ",nein" $\Rightarrow m \notin X$

Komplexität:

zusätzliche Anforderungen an Zeit-/Speicherplatzbedarf von A

- Polynomialzeit: Anzahl Rechenschritte von A ist $\leq |m|^k$, |m|: Länge der Eingabe; k: beliebige Konstante
- Polynomieller Platz: von A benötigter Speicherplatz $\leq |m|^k$
- Exponentialzeit: Anzahl Rechenschritte von A ist $\leq 2^{|m|^k}$
- . . .

9:18

2018-10-22

Teil 1: endliche Wörter

Entscheidungsprobleme

-Komplexität

Einige übliche Komplexitätsklassen

Name	Bedeutung	Beispiel-Problem Erreichbarkeit, ungerichtete Graphen Erreichbarkeit, gerichtete Graphen Primzahlen	
L NL P	logarithm. Speicherplatz nichtdetermin. log. Platz Polynomialzeit		
NP PSpace	nichtdeterminist. Polyzeit polynom. Speicherplatz	Erfüllbarkeit Aussagenlogik Erfüllbarkeit QBF	
ExpTime NExpTime ExpSpace :	Exponentialzeit nichtdet. Exponentialzeit exponentieller Platz :	Gewinnstrategie $n \times n$ -Schach Clique f. schaltkreiscodierte Graphen Äquiv. regulärer Ausdrücke mit " \cdot^2 "	
	unentscheidbar	Erfüllbarkeit Prädikatenlogik	

Komplementklassen: coNL, coNP etc.

Teil 1: endliche Wörter
Entscheidungsprobleme

Name Bedautung Behatung Behatu

Einige übliche Komplexitätsklassen

Einige übliche Komplexitätsklassen

9:21

Klassen so vorlesen: "... ist die Menge aller Probleme, die sich mit einer ... TM in ... Zeit/Platz lösen lassen"

Beispiele nur am Rande erwähnen

Reduktion

$$m \in M$$
 Eingabe $m \in X$? Ausgabe $m \in X$, $m \in X$, $m \in X$

(Polynomielle) Reduktion von $X\subseteq M$ nach $X'\subseteq M'$ ist eine (in Polyzeit) berechenbare Funktion $\pi:M\to M'$ mit

$$m \in X$$
 gdw. $\pi(m) \in X'$

Teil 1: endliche Wörter Entscheidungsprobleme

$$\begin{split} & \text{Reduktion} \\ & m \in M \xrightarrow{\text{Engploy}} \mathbf{m} \in X \mathbb{Z} \xrightarrow{\text{Anogabe}} \sum_{m \in X} \mathbf{m} & = n \in X \\ & \text{(Polymoriscide) Roduktion von } X \subseteq M \text{ such } X' \subseteq M \\ & \text{ist sine (in Polymor) benechmber Funktion } x : M = M' = m \\ & m \in X \qquad \text{gdw.} \qquad x(n) \in X' \end{split}$$

Reduktion

9:26 bis 9:38

2018-10-22

Reduktion: wichtiges Hilfsmittel zum genauen Bestimmen der Komplexität

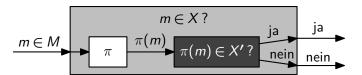
Intuition: Wenn $X \leq X'$, dann ist X höchstens so schwer wie X'.

Reduktion



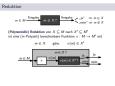
(Polynomielle) Reduktion von $X\subseteq M$ nach $X'\subseteq M'$ ist eine (in Polyzeit) berechenbare Funktion $\pi:M\to M'$ mit

$$m \in X$$
 gdw. $\pi(m) \in X'$



Teil 1: endliche Wörter Entscheidungsprobleme

Reduktion



9:26 bis 9:38

2018-10-22

Reduktion: wichtiges Hilfsmittel zum genauen Bestimmen der Komplexität

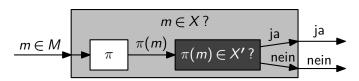
Intuition: Wenn $X \leq X'$, dann ist X höchstens so schwer wie X'.

Reduktion

$$m \in M \xrightarrow{\mathsf{Eingabe}} m \in X$$
? Ausgabe $m \in X$, $m \in X$, $m \in X$

(Polynomielle) Reduktion von $X\subseteq M$ nach $X'\subseteq M'$ ist eine (in Polyzeit) berechenbare Funktion $\pi:M\to M'$ mit

$$m \in X$$
 gdw. $\pi(m) \in X'$



Schreibweise: X < X' bzw. $X <_P X'$ (X auf X' reduzierbar) T 1.7

Teil 1: endliche Wörter Entscheidungsprobleme

 $m \in M \xrightarrow{\text{output}} m \in X^{r} \text{ for } M$ $(Polysoneiche) Rodaktin von <math>X \subseteq M$ ist eine (in Polysot) breverbnhare Fi $m \in X \qquad \text{gdw.}$ $m \in M \qquad \text{gfw.}$ $Schreibevise: <math>X \leq X'$ bzw. $X \leq p X$

Reduktion

☐ Reduktion

9:26 bis 9:38

2018-10-22

Reduktion: wichtiges Hilfsmittel zum genauen Bestimmen der Komplexität

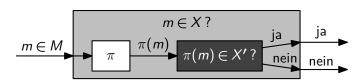
Intuition: Wenn $X \leq X'$, dann ist X höchstens so schwer wie X'.

Reduktion

$$m \in M \xrightarrow{\mathsf{Eingabe}} m \in X$$
? Ausgabe $m \in X$, $m \in X$, $m \in X$

(Polynomielle) Reduktion von $X\subseteq M$ nach $X'\subseteq M'$ ist eine (in Polyzeit) berechenbare Funktion $\pi:M\to M'$ mit

$$m \in X$$
 gdw. $\pi(m) \in X'$

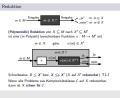


Schreibweise: X < X' bzw. $X <_P X'$ (X auf X' reduzierbar) T 1.7

Wenn alle Probleme aus Komplexitätsklasse $\mathcal C$ auf X reduzierbar, dann ist X schwer für $\mathcal C$.

Teil 1: endliche Wörter
Entscheidungsprobleme

Reduktion



9:26 bis 9:38

2018-10-22

Reduktion: wichtiges Hilfsmittel zum genauen Bestimmen der Komplexität

Intuition: Wenn X < X', dann ist X höchstens so schwer wie X'.

Normalerweise zeigt man, dass ein Problem $X \subseteq M \dots$

- \bullet in einer Komplexitätsklasse $\mathcal C$ liegt, indem man
 - einen Algorithmus A findet, der X löst
 - zeigt, dass A korrekt ist (ja/nein-Antworten) und terminiert
 - zeigt, dass A für jedes $m \in M$ höchstens die C-Ressourcen braucht

... A kann z. B. eine Reduktion zu einem Problem aus $\mathcal C$ sein

Teil 1: endliche Wörter Entscheidungsprobleme

-Bestimmung der Komplexität

Bestimmung der Komplexität

zeiet, dass A korrekt ist (ia/nein-Antworten) und terminiert

... A kann z. B. eine Reduktion zu einem Problem aus C sei

9:38

2018-1

Fragebogen F. $1 \rightsquigarrow 9:02$

Bestimmung der Komplexität

Normalerweise zeigt man, dass ein Problem $X \subseteq M \dots$

- \bullet in einer Komplexitätsklasse \mathcal{C} liegt, indem man
 - einen Algorithmus A findet, der X löst
 - zeigt, dass A korrekt ist (ja/nein-Antworten) und terminiert
 - zeigt, dass A für jedes $m \in M$ höchstens die C-Ressourcen braucht
 - ... A kann z. B. eine Reduktion zu einem Problem aus $\mathcal C$ sein
- \bullet schwer (hard) für \mathcal{C} ist, indem man
 - ein Problem $X' \subset M'$ findet, dass schwer für C ist
 - und eine Reduktion von X' nach X angibt

Teil 1: endliche Wörter Entscheidungsprobleme

-Bestimmung der Komplexität

 zeiet, dass A korrekt ist (ia/nein-Antworten) und terminieri A know v. R. eine Reduktion zu einem Problem zus /* sei

ein Problem X' ⊂ M' findet, dass schwer f
ür C ist · und eine Reduktion von X' nach X angibt

Bestimmung der Komplexität

9:38

2018-1

Fragebogen F. $1 \rightsquigarrow 9:02$

Normalerweise zeigt man, dass ein Problem $X \subseteq M \dots$

- \bullet in einer Komplexitätsklasse \mathcal{C} liegt, indem man
 - einen Algorithmus A findet, der X löst
 - zeigt, dass A korrekt ist (ja/nein-Antworten) und terminiert
 - zeigt, dass A für jedes $m \in M$ höchstens die C-Ressourcen braucht

... A kann z. B. eine Reduktion zu einem Problem aus $\mathcal C$ sein

- \bullet schwer (hard) für \mathcal{C} ist, indem man
 - ein Problem $X' \subset M'$ findet, dass schwer für C ist
 - und eine Reduktion von X' nach X angibt
- \bullet vollständig für C ist, indem man zeigt, dass es
 - \bullet in $\mathcal C$ liegt und
 - schwer für C ist

Teil 1: endliche Wörter Entscheidungsprobleme

-Bestimmung der Komplexität

 zeiet, dass A korrekt ist (ia/nein-Antworten) und terminieri A know v. R. eine Reduktion zu einem Problem aus (* se \bullet ein Problem $X'\subseteq M'$ findet, dass schwer für $\mathcal C$ ist · und eine Reduktion von X' nach X angibt vollständig für C ist, indem man zeigt, dass er

Bestimmung der Komplexität

schwer für C ist

9:38

2018-1

Fragebogen F. $1 \rightsquigarrow 9:02$

Reguläre Ausdrücke

Charakterisierungen

Entscheidungsprobleme

Entscheidungsprobleme für endliche Automaten

- Betrachten wesentliche Eigenschaften von Sprachen (Sprachen repräsentiert durch NEAs oder reguläre Ausdr.)
 - Ist eine gegebene Sprache leer?
 - Ist ein gegebenes Wort w in einer Sprache L?
 - Beschreiben zwei Repräsentationen einer Sprache tatsächlich dieselbe Sprache?

Teil 1: endliche Wörter Entscheidungsprobleme

-Entscheidungsprobleme für endliche Automaten

Entscheidungsprobleme für endliche Automaten

u Betrachten wesentliche Eigenschaften von Sprachen (Sprachen repräsentiert durch NEAs oder reguläre Ausdr.) . Ist eine gegebene Sprache leer?

• lst ein gegebenes Wort w in einer Sprache L? Beschreiben zwei Repräsentationen einer Sorache tatsächlich dieselbe Sprache?

9:42

2018-

Umwandlung . . .

- DEA → NEA: konstante Zeit ●
- NEA → DEA: Exponentialzeit ②
- reg. Ausdr. → NEA: Polynomialzeit
- NEA → reg. Ausdr.: Exponentialzeit ©
- NEA ↔ Typ-3-Gramm.: Polynomialzeit ●

- Betrachten wesentliche Eigenschaften von Sprachen (Sprachen repräsentiert durch NEAs oder reguläre Ausdr.)
 - Ist eine gegebene Sprache leer?
 - Ist ein gegebenes Wort w in einer Sprache L?
 - Beschreiben zwei Repräsentationen einer Sprache tatsächlich dieselbe Sprache?
- Wichtig für Anwendungen (siehe Einführung)

Teil 1: endliche Wörter

Entscheidungsprobleme

Entscheidungsprobleme für endliche Automaten

Entscheidungsprobleme für endliche Automaten

 Betrachten wesentliche Eigenschaften von Sprachen (Sprachen repr\u00e4sentiert durch NEAs oder regul\u00e4re Ausdr.)
 Int eine gegebene Sprache Ieer?
 Int ein gegebene Wort w in einer Sprache L?

Beschreiben zwei Repräsentationen einer Sprache tatsächlich dieselbe Sprache?

 Wichtig für Anwendungen (siehe Einführung)

9:42

2018-

Umwandlung ...

- DEA → NEA: konstante Zeit ●
- NEA → DEA: Exponentialzeit ☺
- reg. Ausdr. \rightarrow NEA: Polynomialzeit \odot
- NEA \rightarrow reg. Ausdr.: Exponentialzeit \odot
- NEA ↔ Typ-3-Gramm.: Polynomialzeit ●

Reguläre Ausdrücke

Charakterisierungen

Entscheidungsprobleme für endliche Automaten

- Betrachten wesentliche Eigenschaften von Sprachen (Sprachen repräsentiert durch NEAs oder reguläre Ausdr.)
 - Ist eine gegebene Sprache leer?
 - Ist ein gegebenes Wort w in einer Sprache L?
 - Beschreiben zwei Repräsentationen einer Sprache tatsächlich dieselbe Sprache?
- Wichtig für Anwendungen (siehe Einführung)
- Art der Repräsentation spielt manchmal eine Rolle: NEA, DEA, regulärer Ausdruck, Typ-3-Grammatik etc. Wir betrachten im Folgenden NEAs und DEAs.

Teil 1: endliche Wörter Entscheidungsprobleme

-Entscheidungsprobleme für endliche Automaten

Entscheidungsprobleme für endliche Automaten

u Betrachten wesentliche Eigenschaften von Sprachen (Sprachen repräsentiert durch NEAs oder reguläre Ausdr.) . Ist eine gegebene Sprache leer?

Ist ein gegebenes Wort w in einer Sprache L? . Beschreiben zwei Repräsentationen einer Sprache tatsächlich dieselbe Sprache?

Wichtig für Anwendungen (siehe Einführung) Art der Repräsentation spielt manchmal eine Rolle: NEA. DEA. regulärer Ausdruck. Tvp-3-Grammatik etc. Wir betrachten im Folgenden NEAs und DEAs

9:42

2018-

Umwandlung . . .

- DEA → NEA: konstante Zeit ●
- NEA → DEA: Exponentialzeit ②
- reg. Ausdr. → NEA: Polynomialzeit
- NEA → reg. Ausdr.: Exponentialzeit ©
- NEA ↔ Typ-3-Gramm.: Polynomialzeit ⊕

2018-1

Eingabe: NEA (oder DEA) \mathcal{A}

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

d.h.
$$LP_{NEA} = \{ \mathcal{A} \mid \mathcal{A} \text{ NEA}, L(\mathcal{A}) = \emptyset \},$$

 $LP_{DEA} = \{ \mathcal{A} \mid \mathcal{A} \text{ DEA}, L(\mathcal{A}) = \emptyset \}$

☐ Das Leerheitsproblem

9:44 Def. der Probleme: Eingabe ist üblicherweise ein Wort, also braucht man eine geeignete Kodierung von NEAs/DEAs (s. ThI 2).

Überprüfung, ob Eingabe wohlgeformt ist, ist üblicherweise billig, macht also keinen Unterschied, ob sie mit in Komplexität zählt oder nicht.

coNL, weil das Komplement (Nichtleerheit) NL-vollst. ist.

coNL-Härte: bei DEAs aufpassen – für jede ausgehende Kante e. Knotens braucht man ein neues Zeichen, also Alphabetgröße = max. Ausgangsgrad. Außerdem Papierkorbzustände einbauen.

(Nicht-)Leerheit von NEAs/DEAs ist also nichts anderes als Wegsuche in gerichteten Graphen.

Fragebogen F. 2: Tabelle, während der nächsten Folien vervollständigen

Eingabe: NEA (oder DEA) \mathcal{A}

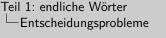
Frage: Ist $L(A) = \emptyset$?

d. h.
$$LP_{NEA} = \{ A \mid A \text{ NEA}, L(A) = \emptyset \},$$

 $LP_{DEA} = \{ A \mid A \text{ DEA}, L(A) = \emptyset \}$

Satz 1.14

LP_{NFA} und LP_{DFA} sind entscheidbar und coNL-vollständig.



d.h. $LP_{NEA} = \{A \mid A \text{ NEA}, L(A) = \emptyset\},$ $LP_{DEA} = \{A \mid A \text{ DEA}, L(A) = \emptyset\}$ Para und LPnra sind entscheidbar und coNL-vollständig

Das Leerheitsproblem

☐ Das Leerheitsproblem

9:44 Def. der Probleme: Eingabe ist üblicherweise ein Wort, also braucht man eine geeignete Kodierung von NEAs/DEAs (s. ThI 2).

Überprüfung, ob Eingabe wohlgeformt ist, ist üblicherweise billig, macht also keinen Unterschied, ob sie mit in Komplexität zählt oder nicht.

coNL, weil das Komplement (Nichtleerheit) NL-vollst. ist.

coNL-Härte: bei DEAs aufpassen – für jede ausgehende Kante e. Knotens braucht man ein neues Zeichen, also Alphabetgröße = max. Ausgangsgrad. Außerdem Papierkorbzustände einbauen.

(Nicht-)Leerheit von NEAs/DEAs ist also nichts anderes als Wegsuche in gerichteten Graphen.

Fragebogen F. 2: Tabelle, während der nächsten Folien vervollständigen

Das Leerheitsproblem

Eingabe: NEA (oder DEA) \mathcal{A}

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

d. h.
$$\mathsf{LP}_{\mathsf{NEA}} = \{ \mathcal{A} \mid \mathcal{A} \; \mathsf{NEA}, \; \mathcal{L}(\mathcal{A}) = \emptyset \},$$

 $\mathsf{LP}_{\mathsf{DEA}} = \{ \mathcal{A} \mid \mathcal{A} \; \mathsf{DEA}, \; \mathcal{L}(\mathcal{A}) = \emptyset \}$

Satz 1.14

LP_{NFA} und LP_{DFA} sind entscheidbar und co**NL**-vollständig.

Beweis.

- Entscheidbarkeit (in Polyzeit): siehe Thl 1
- **coNL**-Zugehörigkeit: Reduktion zu Erreichbarkeit in gerichteten Graphen, siehe T1.7
- coNL-Härte: Reduktion von Erreichbarkeit, analog

Teil 1: endliche Wörter Entscheidungsprobleme

☐ Das Leerheitsproblem

Das Leerheitsproblem d.h. $LP_{NEA} = \{A \mid A \text{ NEA, } L(A) = \emptyset\},$ $LP_{DEA} = \{A \mid A \text{ DEA, } L(A) = \emptyset\}.$ Para und LPnra sind entscheidbar und coNL-vollständig a Entscheidbarkeit (in Polyzeit): siehe Thl 1 Reduktion zu Erreichbarkeit in gerichteten Graphen. siehe T1.

9:44 Def. der Probleme: Eingabe ist üblicherweise ein Wort, also braucht man eine geeignete Kodierung von NEAs/DEAs (s. ThI 2).

Überprüfung, ob Eingabe wohlgeformt ist, ist üblicherweise billig, macht also keinen Unterschied, ob sie mit in Komplexität zählt oder nicht.

coNL, weil das Komplement (Nichtleerheit) NL-vollst. ist.

coNL-Härte: bei DEAs aufpassen – für jede ausgehende Kante e. Knotens braucht man ein neues Zeichen, also Alphabetgröße = max. Ausgangsgrad.

Außerdem Papierkorbzustände einbauen.

(Nicht-)Leerheit von NEAs/DEAs ist also nichts anderes als Wegsuche in gerichteten Graphen.

Fragebogen F. 2: Tabelle, während der nächsten Folien vervollständigen

Entscheidungsprobleme

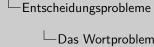
Das Wortproblem

Eingabe: NEA (oder DEA) \mathcal{A} , Wort $w \in \Sigma^*$

Frage: Ist $w \in L(A)$?

d.h.
$$WP_{NEA} = \{(A, w) \mid A \text{ NEA}, w \in L(A)\},\$$

 $WP_{DEA} = \{(A, w) \mid A \text{ DEA}, w \in L(A)\}$



☐ Das Wortproblem

Für DEAs muss nicht geraten werden.

9:48 Reduktion zum I P:

• liefert nur ..in P"

Teil 1: endliche Wörter

- A_w ist ein N/DEA, der nur w akzeptiert (einfach zu bauen)
- nutzt Abgeschlossenheit unter Schnitt (Konstr. Produktautomat)

obere Schranke: Rate Weg der Länge < |Q| ab q_0 , so dass im i-ten Schritt das i-te Zeichen von w gelesen wird. Akzeptiere, wenn am Ende ein akz. Zustand erreicht ist.

NL-Härte: Wandle gegebenen G = (V, E) in NEA, so dass alle Kanten mit a beschriftet sind. Anfangs-/akz. Zustand: s, t. Zusätzlich a-Schleife an t. Dann frage nach $w = a^{|V|}$.

L-Härte: Führt hier zu weit; braucht speziellen Reduktionsbegriff (weak red., siehe auch Holzer & Kutrib, Inf & Comp. 2011)

☐ Das Wortproblem

Für DEAs muss nicht geraten werden.

Das Wortproblem

Das Wortproblem

Eingabe: NEA (oder DEA) \mathcal{A} , Wort $w \in \Sigma^*$

Frage: Ist $w \in L(A)$?

d.h.
$$WP_{NEA} = \{(A, w) \mid A \text{ NEA, } w \in L(A)\},\ WP_{DEA} = \{(A, w) \mid A \text{ DEA, } w \in L(A)\}$$

Satz 1.15

WP_{NFA} und WP_{DFA} sind entscheidbar.

WP_{NFA} ist **NL**-vollständig; WP_{DFA} ist **L**-vollständig.

9:48 Reduktion zum I P:

• liefert nur ..in P"

- A_w ist ein N/DEA, der nur w akzeptiert (einfach zu bauen)
- nutzt Abgeschlossenheit unter Schnitt (Konstr. Produktautomat)

obere Schranke: Rate Weg der Länge < |Q| ab q_0 , so dass im i-ten Schritt das i-te Zeichen von w gelesen wird.

Akzeptiere, wenn am Ende ein akz. Zustand erreicht ist.

NL-Härte: Wandle gegebenen G = (V, E) in NEA, so dass alle Kanten mit a beschriftet sind. Anfangs-/akz. Zustand: s, t. Zusätzlich a-Schleife an t. Dann frage nach $w = a^{|V|}$.

L-Härte: Führt hier zu weit; braucht speziellen Reduktionsbegriff (weak red., siehe auch Holzer & Kutrib, Inf & Comp. 2011)

Das Wortproblem

Eingabe: NEA (oder DEA) \mathcal{A} , Wort $w \in \Sigma^*$

Frage: Ist $w \in L(A)$?

d.h.
$$WP_{NEA} = \{(A, w) \mid A \text{ NEA, } w \in L(A)\},\$$

 $WP_{DEA} = \{(A, w) \mid A \text{ DEA, } w \in L(A)\}$

Satz 1.15

WP_{NFA} und WP_{DFA} sind entscheidbar.

WP_{NFA} ist **NL**-vollständig; WP_{DFA} ist **L**-vollständig.

Beweis.

- Entscheidbarkeit (in Polyzeit): siehe ThI 1 (Reduktion zum LP: $w \in L(A)$ gdw. $L(A) \cap L(A_w) \neq \emptyset$)
- NL-Vollst.: ≈ Erreichbarkeit in gerichteten Graphen

Teil 1: endliche Wörter Eingabe: NEA (oder DEA) A, Wort $w \in \Sigma^*$ Entscheidungsprobleme d.h. $WP_{MEA} = \{(A, w) \mid A \text{ NEA, } w \in L(A)\},$ $WP_{DEA} = \{(A, w) \mid A \text{ DEA, } w \in L(A)\}.$ ☐ Das Wortproblem (Reduktion zum LP: $w \in L(A)$ edw. $L(A) \cap L(A_{-}) \neq \emptyset$) w NL-Vollst.: ≈ Erreichbarkeit in gerichteten Graphen

9:48 Reduktion zum I P:

• liefert nur ..in P"

- A_w ist ein N/DEA, der nur w akzeptiert (einfach zu bauen)
- nutzt Abgeschlossenheit unter Schnitt (Konstr. Produktautomat)

obere Schranke: Rate Weg der Länge < |Q| ab q_0 , so dass im i-ten Schritt das i-te Zeichen von w gelesen wird.

Akzeptiere, wenn am Ende ein akz. Zustand erreicht ist. Für DEAs muss nicht geraten werden.

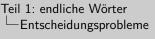
NL-Härte: Wandle gegebenen G = (V, E) in NEA, so dass alle Kanten mit a beschriftet sind. Anfangs-/akz. Zustand: s, t. Zusätzlich a-Schleife an t. Dann frage nach $w = a^{|V|}$.

L-Härte: Führt hier zu weit; braucht speziellen Reduktionsbegriff (weak red., siehe auch Holzer & Kutrib, Inf & Comp. 2011)

Eingabe: NEAs (oder DEAs) A_1, A_2

Frage: Ist $L(A_1) = L(A_2)$?

d. h.
$$\ddot{\mathsf{A}}\mathsf{P}_{\mathsf{NEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{NEAs}, \; L(\mathcal{A}_1) = L(\mathcal{A}_2)\}, \\ \ddot{\mathsf{A}}\mathsf{P}_{\mathsf{DEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{DEAs}, \; L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$$



□ Das Äguivalenzproblem

9:54

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

Eingabe: NEAs (oder DEAs) A1, A2

d.h. $\mathsf{AP}_{\mathsf{NEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{NEAs}, \; \mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)\}$ $\mathsf{AP}_{\mathsf{DEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{DEAs}, \; \mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)\}$

Eingabe: NEAs (oder DEAs) A_1, A_2

Frage: Ist $L(A_1) = L(A_2)$?

d. h.
$$\ddot{\mathsf{A}}\mathsf{P}_{\mathsf{NEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{NEAs}, \; L(\mathcal{A}_1) = L(\mathcal{A}_2)\}, \\ \ddot{\mathsf{A}}\mathsf{P}_{\mathsf{DEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{DEAs}, \; L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$$

Satz 1.16

ÄP_{NFA} und ÄP_{DFA} sind entscheidbar.

ÄP_{NFA} ist **PSpace**-vollständig; ÄP_{DFA} ist **NL**-vollständig.

Teil 1: endliche Wörter Entscheidungsprobleme Das Äguivalenzproblem Eingabe: NEAs (oder DEAs) A1, A2 d.h. $\mathsf{AP}_{\mathsf{NEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{NEAs}, \; \mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)\}$ $\mathsf{AP}_{\mathsf{DEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{DEAs}, \; \mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)\}$ APNEA und APNEA sind entscheidbar APNEA ist PSpace-vollständig: APDEA ist NL-vollständig

□ Das Äguivalenzproblem

9:54

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

Das Äguivalenzproblem

Eingabe: NEAs (oder DEAs) A_1, A_2

Frage: Ist $L(A_1) = L(A_2)$?

d. h.
$$\ddot{\mathsf{A}}\mathsf{P}_{\mathsf{NEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{NEAs}, \; L(\mathcal{A}_1) = L(\mathcal{A}_2)\}, \\ \ddot{\mathsf{A}}\mathsf{P}_{\mathsf{DEA}} = \{(\mathcal{A}_1, \mathcal{A}_2) \mid \mathcal{A}_1, \mathcal{A}_2 \; \mathsf{DEAs}, \; L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$$

Satz 1.16

 AP_{NEA} und AP_{DEA} sind entscheidbar.

ÄP_{NEA} ist **PSpace**-vollständig; ÄP_{DEA} ist **NL**-vollständig.

Beweis.

- Entscheidbarkeit: siehe Thl 1 (Red. zum LP: $L(A_1) = L(A_2)$ gdw. $L(A_1) \triangle L(A_2) = \emptyset$)
- Komplexität: Automat für $L(A_1) \triangle L(A_2)$ ist exponentiell in der Größe der Eingabe-NEAs; polynomiell im Fall von DEAs Details: siehe [Holzer & Kutrib 2011]

Teil 1: endliche Wörter Entscheidungsprobleme

d.h. $AP_{NEA} = \{(A_1, A_2) \mid A_1, A_2 \text{ NEAs}, L(A_1) = L(A_2)\}$ $AP_{NEA} = \{(A_1, A_2) \mid A_1, A_2 \text{ DEAs}, L(A_1) = L(A_2)\}$ APNEA und APnea sind entscheidbar APNEA ist PSpace-vollständig: APDEA ist NL-vollständig (Red. zum LP: $L(A_1) = L(A_2)$ edw. $L(A_1) \wedge L(A_2) = \emptyset$

Komplexität: Automat für L(A₁) △ L(A₂) ist exponentiell in

der Größe der Eingabe-NEAs; polynomiell im Fall von DEAs Details: siehe [Holzer & Kutrib 2011]

Das Äguivalenzproblem

Frage: lst $L(A_1) = L(A_2)$?

Eingabe: NEAs (oder DEAs) A1, A2

—Das Äguivalenzproblem

9:54

 \triangle = symmetrische Differenz zweier Mengen; ausdrückbar mittels \cup , \cap , $\bar{\cdot}$ \rightarrow Abschlusseigenschaften! **Eingabe**: NEA (oder DEA) \mathcal{A}

Frage: Ist
$$L(A) = \Sigma^*$$
?

d. h.
$$UP_{NEA} = \{ \mathcal{A} \mid \mathcal{A} \text{ NEA}, \ \mathcal{L}(\mathcal{A}) = \Sigma^* \},$$

 $UP_{DEA} = \{ \mathcal{A} \mid \mathcal{A} \text{ DEA}, \ \mathcal{L}(\mathcal{A}) = \Sigma^* \}$

Das Universalitätsproblem

2018-1

□ Das Universalitätsproblem

9:57

Reduktion vom/zum Komplement LP:

$$L(A) = \Sigma^* \text{ gdw. } \overline{L(A)} = \emptyset$$

Für DEAs NL-v., analog zu Wegsuche: ist ein nicht-akz. Zustand erreichbar?

Für NEAs PSPACE-v., Wegsuche im Potenz-DEA on-the-fly

Eingabe: NEA (oder DEA) \mathcal{A}

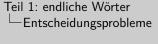
Frage: Ist
$$L(A) = \Sigma^*$$
?

d.h.
$$UP_{NEA} = \{ \mathcal{A} \mid \mathcal{A} \text{ NEA}, \ \mathcal{L}(\mathcal{A}) = \Sigma^* \},\ UP_{DEA} = \{ \mathcal{A} \mid \mathcal{A} \text{ DEA}, \ \mathcal{L}(\mathcal{A}) = \Sigma^* \}$$

Satz 1.17

UP_{NFA} und UP_{DFA} sind entscheidbar.

Beweis: Übungsaufgabe



d.h. $UP_{NEA} = \{A \mid A \text{ NEA, } L(A) = \Sigma^*\}$ $UP_{DEA} = \{A \mid A \text{ DEA, } L(A) = \Sigma^*\}$

Das Universalitätsproblem

Beweis: Obungsaufgabe

□ Das Universalitätsproblem

9:57

2018-1

Reduktion vom/zum Komplement LP:

$$L(A) = \Sigma^*$$
 gdw. $\overline{L(A)} = \emptyset$

Für DEAs NL-v., analog zu Wegsuche: ist ein nicht-akz. Zustand erreichbar?

Für NEAs PSPACE-v., Wegsuche im Potenz-DEA on-the-fly

Grundbegriffe Textsuche Abschlusseig. Reguläre Ausdrücke

		für DEAs	für NEAs
Problem	entscheidbar?	effizient lösbar?	effizient lösbar?
LP	✓	✓	✓
WP	✓	✓	\checkmark
ÄP	✓	✓	X *
UP	✓	✓	X *

NEAs/DEAs

└─Überblick Entscheidungsprobleme für

9:58 bis 10:00, Ende

Hier nochmal für Euch als Überblick

Zum Literaturverzeichnis blättern

^{*} unter den üblichen komplexitätstheoretischen Annahmen (z. B. PSpace ≠ P)

Literatur für diesen Teil (Basis)



John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman.

Introduction to Automata Theory, Languages and Computation.

2. Auflage, Addison-Wesley, 2001.

Kapitel 1,2.

Verfügbar in SUUB (verschiedene Auflagen, auch auf Deutsch)



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

Teil 1: endliche Wörter

Literatur für diesen Teil (Basis)

à John E. Hopendh, Rijaw Montani, Juffrey D. Ulman.
Introduction to Automata Theory, Languages and Computation.
2. Autogra, Automos Workey, 2011.
Equal 12.

Mary 1. Sept. 1.

Literatur für diesen Teil (Basis)

automata/automata-notes.pdf

"Basis": um die Inhalte der Vorlesung nachzulesen

Literatur für diesen Teil (weiterführend)



Markus Holzer, Martin Kutrib.

Descriptional and computational complexity of finite automata – A survey.

Information and Computation 209:456-470, 2011.

Kapitel 3: sehr umfassender Überblick über Entscheidungsprobleme für endliche Automaten und deren Komplexität; viel Literatur

Verfügbar in SUUB (elektronisch)

https://doi.org/10.1016/j.ic.2010.11.013

Teil 1: endliche Wörter

Literatur für diesen Teil (weiterführend)

Markus Holzer, Martin Kutrib.

Descriptional and computational complexity of firste automata – A survey.

Information and Computation 209:456–470, 2011.

Rapitel 3: sehr umtassender Überblick über Entscheidungs für endliche Automaten und deren Komplexität; viel Litera Verfügbar in SUUB (elektronisch)

Literatur für diesen Teil (weiterführend

"weiterführend": um bestimmte Details zu vertiefen, die in der Vorlesung nur am Rande angesprochen werden

