Info III Tutorium

Thomas Pajor & Julian Hörst

01. März 2006

Don't Panic!

Frage

Wie kann man aus einem DEA einen RegExp konstruieren?

Ein bisschen Vorbereitung...

Betrachte einen DEA $\mathcal{A} := (\{q_1, \ldots, q_n\}, \Sigma, \delta, q_1, F)$

Ein bisschen Vorbereitung...

Betrachte einen DEA $\mathcal{A} := (\{q_1, \ldots, q_n\}, \Sigma, \delta, q_1, F)$

Definiere:

- $L_f := \{w \in \Sigma^* \mid \mathcal{A}$ endet nach Abarbeitung von w in $f\}$

Ein bisschen Vorbereitung...

Betrachte einen DEA $\mathcal{A} := (\{q_1, \ldots, q_n\}, \Sigma, \delta, q_1, F)$

Definiere:

- $L_f := \{w \in \Sigma^* \mid \mathcal{A}$ endet nach Abarbeitung von w in $f\}$
- $L^m_{q_i,q_j} := \{w \in \Sigma^* \mid$ Abarbeitung von w aus q_i nach q_j hat nur Zwischenzustände $\{q_1, \ldots, q_m\}\}$

Ein bisschen Vorbereitung...

Betrachte einen DEA $\mathcal{A} := (\{q_1, \ldots, q_n\}, \Sigma, \delta, q_1, F)$

Definiere:

- $L_f := \{w \in \Sigma^* \mid \mathcal{A} \text{ endet nach Abarbeitung von } w \text{ in } f\}$
- $L^m_{q_i,q_j} := \{w \in \Sigma^* \mid \text{Abarbeitung von } w \text{ aus } q_i \text{ nach } q_j \text{ hat nur Zwischenzustände } \{q_1, \ldots, q_m\}\}$

Es gilt:

$$L(\mathcal{A}) = \bigcup_{f \in F} L_f = \bigcup_{f \in F} L^n_{q_1,f}$$

Das Induktive Verfahren

Wir definieren die Sprache $L(A)$ induktiv nach m:

- Fall $m = 0$ und $q_j = q_i$:

$$L^0_{q_i,q_i} := \{a \in \Sigma \mid \delta(q_i, a) = q_i\} \cup \{\varepsilon\}$$

- Fall $m = 0$ und $q_j \neq q_i$:

$$L^0_{q_i,q_j} := \{a \in \Sigma \mid \delta(q_i, a) = q_j\}$$

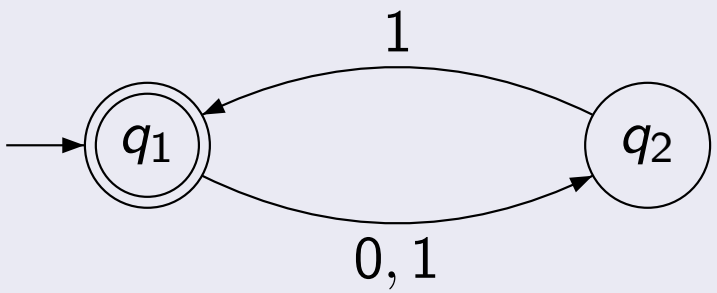
- Sonst $(m > 0)$:

$$L^{m+1}_{q_i,q_j} := L^m_{q_i,q_j} \cup \left(L^m_{q_i,q_{m+1}} (L^m_{q_{m+1},q_{m+1}})^* L^m_{q_{m+1},q_j} \right)$$

Wunschaufgabe

Aufgabe

Gegeben sei ein DEA $\mathcal{A}$ über $\Sigma := \{0, 1\}$, der durch folgendes Übergangsdiagramm definiert wird:



Bestimmen Sie durch *systematische Konstruktion* einen regulären Ausdruck R derart, dass gilt $L(R) = L(\mathcal{A})$.

Frage

Was sind die Unterschiede zwischen Entscheidbar/Semi–Entscheidbar/Unentscheidbar?

Entscheidbarkeit

Eine Sprache L heißt semi–entscheidbar falls es eine Turing Maschine $\mathcal{T}$ gibt, so dass für alle Wörter $w \in L$ gilt: $\mathcal{T}$ akzeptiert w.

Entscheidbarkeit

Eine Sprache L heißt semi–entscheidbar falls es eine Turing Maschine $\mathcal{T}$ gibt, so dass für alle Wörter $w \in L$ gilt: $\mathcal{T}$ akzeptiert w.

Eine Sprache L heißt entscheidbar falls es eine Turing Maschine $\mathcal{T}$ gibt, so dass für alle $w \in L$ gilt: $\mathcal{T}$ akzeptiert w und für alle $w \in \Sigma^*$ gilt: $\mathcal{T}$ *hält* bei Eingabe w.

Entscheidbarkeit

Eine Sprache L heißt semi–entscheidbar falls es eine Turing Maschine $\mathcal{T}$ gibt, so dass für alle Wörter $w \in L$ gilt: $\mathcal{T}$ akzeptiert w.

Eine Sprache L heißt entscheidbar falls es eine Turing Maschine $\mathcal{T}$ gibt, so dass für alle $w \in L$ gilt: $\mathcal{T}$ akzeptiert w und für alle $w \in \Sigma^*$ gilt: $\mathcal{T}$ *hält* bei Eingabe w.

Eine Sprache L heißt „unentscheidbar“ falls es keine Turingmaschine $\mathcal{T}$ gibt, die alle Wörter $w \in L$ akzeptiert.

Some facts...

Ist L eine semi–entscheidbare Sprache die aber nicht entscheidbar ist, so ist ihr Komplement unentscheidbar.

Some facts...

Ist L eine semi–entscheidbare Sprache die aber nicht entscheidbar ist, so ist ihr Komplement unentscheidbar.

Ist L eine semi–entscheidbare Sprache deren Komplement auch semi–entscheidbar ist, so ist L entscheidbar.

Some facts...

Ist L eine semi–entscheidbare Sprache die aber nicht entscheidbar ist, so ist ihr Komplement unentscheidbar.

Ist L eine semi–entscheidbare Sprache deren Komplement auch semi–entscheidbar ist, so ist L entscheidbar.

Sei L_1 entscheidbar und L_2 semi–entscheidbar. Ist $L_1 \cap L_2$ im Allgemeinen entscheidbar?

Some facts...

Ist L eine semi–entscheidbare Sprache die aber nicht entscheidbar ist, so ist ihr Komplement unentscheidbar.

Ist L eine semi–entscheidbare Sprache deren Komplement auch semi–entscheidbar ist, so ist L entscheidbar.

Sei L_1 entscheidbar und L_2 semi–entscheidbar. Ist $L_1 \cap L_2$ im Allgemeinen entscheidbar?
Nein! Betrachte $L_1 := \Sigma^*$. Dann ist $L_1 \cap L_2 = L_2$ natürlich weiterhin semi–entscheidbar.

Frage

Warum ist die Diagonalsprache L_d nicht entscheidbar?

Diagonalsprache

Definition

Sei w_i eine Binärzahl dann ist $\mathcal{T}_i$ die Turingmaschine die durch w_i (Gödel–)kodiert ist.

$$L_d := \{w_i \in \Sigma^* \mid \mathcal{T}_i \text{ akzeptiert } w_i \text{ nicht } \}$$

Diagonalsprache

Die Diagonalsprache

$L_d := \{w_i \in \Sigma^* \mid \mathcal{T}_i \text{ akzeptiert } w_i \text{ nicht } \}$

Diagonalsprache

Die Diagonalsprache

$L_d := \{w_i \in \Sigma^* \mid \mathcal{T}_i \text{ akzeptiert } w_i \text{ nicht } \}$

Beweis

Annahme: L_d ist entscheidbar.

Diagonalsprache

Die Diagonalsprache

$L_d := \{w_i \in \Sigma^* \mid \mathcal{T}_i \text{ akzeptiert } w_i \text{ nicht } \}$

Beweis

Annahme: L_d ist entscheidbar. Dann existiert eine Turingmaschine $\mathcal{M}_i$ die alle $w \in L_d$ akzeptiert.

Diagonalsprache

Die Diagonalsprache

$L_d := \{w_i \in \Sigma^* \mid \mathcal{T}_i \text{ akzeptiert } w_i \text{ nicht } \}$

Beweis

Annahme: L_d ist entscheidbar. Dann existiert eine Turingmaschine $\mathcal{M}_i$ die alle $w \in L_d$ akzeptiert.
Betrachte $w_i = \langle \mathcal{M}_i \rangle$. Ist $w_i \in L_d$?

- $w_i \in L_d$. Dann akzeptiert $\mathcal{M}_i$ das Wort w_i. Nach Def ist dann aber $w_i \notin L_d$

Diagonalsprache

Die Diagonalsprache

$L_d := \{w_i \in \Sigma^* \mid \mathcal{T}_i \text{ akzeptiert } w_i \text{ nicht } \}$

Beweis

Annahme: L_d ist entscheidbar. Dann existiert eine Turingmaschine $\mathcal{M}_i$ die alle $w \in L_d$ akzeptiert.
Betrachte $w_i = \langle \mathcal{M}_i \rangle$. Ist $w_i \in L_d$?

- $w_i \in L_d$. Dann akzeptiert $\mathcal{M}_i$ das Wort w_i. Nach Def ist dann aber $w_i \notin L_d$
- $w_i \notin L_d$. Dann akzeptiert $\mathcal{M}_i$ das Wort w_i nicht. Damit ist aber $w_i \in L_d$

Diagonalsprache

Die Diagonalsprache

$L_d := \{w_i \in \Sigma^* \mid \mathcal{T}_i \text{ akzeptiert } w_i \text{ nicht } \}$

Beweis

Annahme: L_d ist entscheidbar. Dann existiert eine Turingmaschine $\mathcal{M}_i$ die alle $w \in L_d$ akzeptiert.
Betrachte $w_i = \langle \mathcal{M}_i \rangle$. Ist $w_i \in L_d$?

- $w_i \in L_d$. Dann akzeptiert $\mathcal{M}_i$ das Wort w_i. Nach Def ist dann aber $w_i \notin L_d$
- $w_i \notin L_d$. Dann akzeptiert $\mathcal{M}_i$ das Wort w_i nicht. Damit ist aber $w_i \in L_d$

Widerspruch! □

Frage

Was hat es mit den Komplexitätsklassen auf sich?

„Deterministisch Polynomiell“:

$$\mathcal{P} := \{L \subseteq \Sigma^* \mid \exists \text{ DTM } \mathcal{T}_L : \forall w \in \Sigma^* : \text{time}_{\mathcal{T}}(w) < p(|w|)\}$$

„Deterministisch Polynomiell“:

$$\mathcal{P} := \{L \subseteq \Sigma^* \mid \exists \text{ DTM } \mathcal{T}_L : \forall w \in \Sigma^* : \text{time}_{\mathcal{T}}(w) < p(|w|)\}$$

„Nichtdeterministisch Polynomiell“:

$$\mathcal{NP} := \{L \subseteq \Sigma^* \mid \exists \text{ NTM } \mathcal{T}_L : \forall w \in L : \text{ntime}_{\mathcal{T}}(w) < p(|w|)\}$$

„Deterministisch Polynomiell“:

$$\mathcal{P} := \{L \subseteq \Sigma^* \mid \exists \text{ DTM } \mathcal{T}_L : \forall w \in \Sigma^* : \text{time}_{\mathcal{T}}(w) < p(|w|)\}$$

„Nichtdeterministisch Polynomiell“:

$$\mathcal{NP} := \{L \subseteq \Sigma^* \mid \exists \text{ NTM } \mathcal{T}_L : \forall w \in L : \text{ntime}_{\mathcal{T}}(w) < p(|w|)\}$$

„$\mathcal{NP}$–hart“:

$$\mathcal{NP}\# := \{L \subseteq \Sigma^* \mid \forall L' \in \mathcal{NP} : L' \leq_p L\}$$

„Deterministisch Polynomiell“:

$$\mathcal{P} := \{L \subseteq \Sigma^* \mid \exists \text{ DTM } \mathcal{T}_L : \forall w \in \Sigma^* : \text{time}_{\mathcal{T}}(w) < p(|w|)\}$$

„Nichtdeterministisch Polynomiell“:

$$\mathcal{NP} := \{L \subseteq \Sigma^* \mid \exists \text{ NTM } \mathcal{T}_L : \forall w \in L : \text{ntime}_{\mathcal{T}}(w) < p(|w|)\}$$

„$\mathcal{NP}$–hart“:

$$\mathcal{NP}\# := \{L \subseteq \Sigma^* \mid \forall L' \in \mathcal{NP} : L' \leq_p L\}$$

„$\mathcal{NP}$–vollständig“:

$$\mathcal{NPV} := \{L \subseteq \Sigma^* \mid L \in \mathcal{NP} \text{ und } L \in \mathcal{NP}\#\}$$

Frage

Wie zeige ich dass ein Problem $\mathcal{NP}$–vollständig ist?

Reduktionen...

Lösungsweg...

Reduktionen...

Lösungsweg...

- Zeige $L \in \mathcal{NP}$

Reduktionen...

Lösungsweg...

- Zeige $L \in \mathcal{NP}$
- Zeige L ist $\mathcal{NP}$–hart.

Reduktionen...

Lösungsweg...

- Zeige $L \in \mathcal{NP}$
- Zeige L ist $\mathcal{NP}$–hart.
  Benutze ein Problem L', von dem bekannt ist, dass es $\mathcal{NP}$–*vollständig* ist, und reduziere es auf L. Dazu ist zu zeigen:

Reduktionen...

Lösungsweg...

- Zeige $L \in \mathcal{NP}$
- Zeige L ist $\mathcal{NP}$–hart.
  Benutze ein Problem L', von dem bekannt ist, dass es $\mathcal{NP}$–*vollständig* ist, und reduziere es auf L. Dazu ist zu zeigen:
  - Die Reduktion ist polynomiell

Reduktionen...

Lösungsweg...

- Zeige $L \in \mathcal{NP}$
- Zeige L ist $\mathcal{NP}$–hart.
  Benutze ein Problem L', von dem bekannt ist, dass es $\mathcal{NP}$–*vollständig* ist, und reduziere es auf L. Dazu ist zu zeigen:
  - Die Reduktion ist polynomiell
  - Die Reduktion funktioniert:

$$I \in L \text{ hat Lösung } \Leftrightarrow I' \in L' \text{ hat Lösung}$$

Wunschaufgabe

Das Problem Hitting Set

Gegeben: Eine Menge $\mathcal{U}$ und eine Familie $M := \{M_1, \ldots, M_n\}$ von Teilmengen von $\mathcal{U}$ sowie ein Parameter $K \in \mathbb{N}$.
Frage: Gibt es eine Teilmenge $H \subseteq \mathcal{U}$ mit $|H| = K$ derart, dass

$$\forall M_i \in M : |H \cap M_i| \geq 1?$$

Wunschaufgabe

Das Problem Hitting Set

Gegeben: Eine Menge $\mathcal{U}$ und eine Familie $M := \{M_1, \ldots, M_n\}$ von Teilmengen von $\mathcal{U}$ sowie ein Parameter $K \in \mathbb{N}$.
Frage: Gibt es eine Teilmenge $H \subseteq \mathcal{U}$ mit $|H| = K$ derart, dass

$$\forall M_i \in M : |H \cap M_i| \geq 1?$$

Aufgabe

Zeigen Sie: Hitting Set ist $\mathcal{NP}$–vollständig.

Frage

Gibt es Probleme die $\mathcal{NP}$–hart aber nicht $\mathcal{NP}$–vollständig sind?

Wunschaufgabe

Aufgabe

Zeigen Sie: Das Halteproblem ist $\mathcal{NP}$–hart. Warum ist es nicht $\mathcal{NP}$–vollständig?

Frage

Kannst du eine Aufgabe zu co–$\mathcal{NP}$ machen?

Aufgabe

Ich hatte eigentlich dazu eine Aufgabe im letzten Tutorium am 13.02.2006. Aufgabe und Lösung gibt es unter
www.logn.de/tut/material.php

Frage

Kannst du eine Aufgabe zu Approximationsalgorithmen machen?

Approximationsalgorithmen

Approximationsalgorithmen

Sei L ein Optimierungsproblem, und $I \in L$ eine Instanz von L.

Approximationsalgorithmen

Sei L ein Optimierungsproblem, und $I \in L$ eine Instanz von L.

- OPT(I) – die Optimallösung von I.

Approximationsalgorithmen

Sei L ein Optimierungsproblem, und $I \in L$ eine Instanz von L.

- $\mathrm{OPT}(I)$ – die Optimallösung von I.
- $\mathcal{A}(I)$ – die Lösung die ein Algorithmus $\mathcal{A}$ zu I liefert.

Approximationsalgorithmen

Sei L ein Optimierungsproblem, und $I \in L$ eine Instanz von L.

- $\mathrm{OPT}(I)$ – die Optimallösung von I.
- $\mathcal{A}(I)$ – die Lösung die ein Algorithmus $\mathcal{A}$ zu I liefert.

Zu einem Minimierungsproblem L ist ein Algorithmus $\mathcal{A}$ ein *Approximationsalgorithmus mit relativer Gütegarantie* ρ, falls gilt

$$\frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} \leq \rho \qquad \forall I \in L$$

Wunschaufgabe

Das Problem Bin Packing mit Bins der Größe 1

Gegeben: Eine Menge $M := \{a_1, \ldots, a_n\}$ von Elementen, sowie eine Gewichtsfunktion $w : M \to (0, 1]$.
Frage: Wieviele Bins B_j der Größe 1 sind mindestens nötig um die Elemente alle zu verpacken?

Wunschaufgabe

Das Problem Bin Packing mit Bins der Größe 1

Gegeben: Eine Menge $M := \{a_1, \ldots, a_n\}$ von Elementen, sowie eine Gewichtsfunktion $w : M \to (0, 1]$.
Frage: Wieviele Bins B_j der Größe 1 sind mindestens nötig um die Elemente alle zu verpacken?

Aufgabe

Zeigen Sie, dass der Algorithmus Next–Fit (nächste Folie) eine Faktor–2–Approximation liefert.

Next–Fit

Der Algorithmus Next–Fit

$K := 1$
$B_K := \{a_1\}$
for $i = 2, \ldots, n$
    if $w(a_i) > 1 - \sum_{a \in B_K} w(a)$
        $K := K + 1$
    $B_K := B_K \cup \{a_i\}$
return K

Frage

Ist das Pumping–Lemma für kontextfreie Sprachen klausurrelevant?

Eher unwahrscheinlich, da nichtmal Übungsblattaufgaben dazu da waren. Ausschließen möchte ich es aber nicht.

Noch Fragen?

Noch Fragen?

Viel Erfolg in der Klausur!