Komplexität und natürliche Sprache

Algorithmische Komplexität: Einordnung natürlicher Sprache

Timm Lichte & Christian Wurm

Heinrich-Heine-Universität Düsseldorf

Sommersemester 2017, 14.06.2017

SFB 991

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Outline

1. Überblick
2. Einordnung natürlicher Sprache
3. Real-Time-Turingmaschinen (RTTM)

Outline

1 Überblick

2 Einordnung natürlicher Sprache

3 Real-Time-Turingmaschinen (RTTM)

Überblick

- Einordnung natürlicher Sprache: Was für eine TM ist das?
- Real-Time-Turingmaschinen (RTTM)

Outline

1. Überblick

2. Einordnung natürlicher Sprache

3. Real-Time-Turingmaschinen (RTTM)

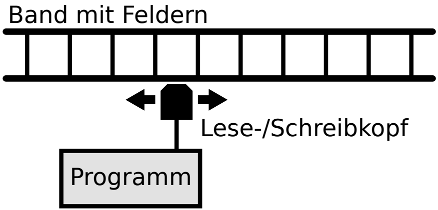
Algorithmische Komplexität

Intuition

Je komplexer desto aufwändiger ist das Wortproblem zu berechnen.

Wortproblem: $w \in L_G$?

Modell: Turingmaschine (Turing (1936))



(aus Wikipedia)

- Wie viele Schritte werden benötigt? (Zeitkomplexität)
- Wie viele Zellen werden benötigt? (Raumkomplexität)
- Hält die Turingmaschine? (Halteproblem)

Algorithmische Komplexität

semi-entscheidbar		
entscheidbar		
2-EXPTIME	$\mathcal{O}(2^{2^{poly(n)}})$	
factorial time	$\mathcal{O}(n!)$	
EXPSPACE		
EXPTIME	$\mathcal{O}(2^{poly(n)})$	
NPSPACE		
PSPACE		"nicht effizient"
NLINSPACE		
NP(TIME)	$\mathcal{O}(n^c)$	
P(TIME)	$\mathcal{O}(n^c)$	
LOGSPACE		
LINTIME	$\mathcal{O}(n)$	"effizient"
konstante Zeit	$\mathcal{O}(c)$	

Extensionale und algorithmische Komplexität

type 0: recursively enumerable

HPSG, TG, TM — $a^{f(n)}$ — semi-entscheidbar

type 1: context-sensitive

LFG, LBA — a^{2^n}, $\overline{w}^n$

NP

P

$a^n b^n c^n \ldots$, $\overline{w}^k$

LMG, RCG = PTIME

mildly context-sensitive

TAG, EPDA — $a^n b^m c^n d^m$, $\overline{ww}$ — $\mathcal{O}(n^6)$

type 2: context-free

CFG, PDA — $a^n b^m c^m d^n$, $\overline{ww}^R$ — $\mathcal{O}(n^3)$

type 3: regular

FSA — $a^n b^m c^k d^l$ — $\mathcal{O}(n)$

Natürliche Sprache ist schwach kontextsensitiv?

Joshi (1985), Joshi, Shanker & Weir (1990):

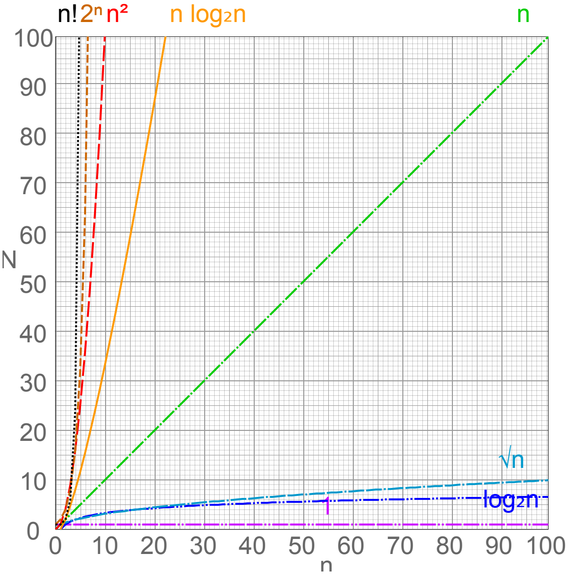
- $\subseteq$ PTIME, $\supset$ CFL
- kreuzende Abhängigkeiten, d.h. $\overline{ww}$
- semi-linear

Wir wissen aus eigener Erfahrung:

- Sprache fällt uns leicht.
- Sprache wird (strikt) inkrementell verarbeitet.

Welche Zeitfunktion hat Sprache wohl?
(Problem: Durchschnitt versus Worst Case)

Verschiedene Funktionen im Vergleich



aus Wikipedia

Die inkrementelle Verarbeitung der Kopiersprache

Ein Experiment:

- Sei $L = \{\overline{w}\overline{w} | \overline{w} \in \{a, b\}^*$ und gegeben $\overline{x} \in \{a, b\}^*$.
- Sie müssen das Wortproblem $\overline{x} \in L$? im Kopf lösen, d.h. ohne Papier und Bleistift.
- Die Buchstaben aus $\overline{x}$ werden nacheinander genau einmal genannt.
- Die Antwort (akzeptiert/nicht akzeptiert) muss innerhalb von 1 Sekunde gegeben werden.

Die inkrementelle Verarbeitung der Kopiersprache

Ein Experiment:

- Sei $L = \{\overline{w}\overline{w} | \overline{w} \in \{a, b\}^*$ und gegeben $\overline{x} \in \{a, b\}^*$.
- Sie müssen das Wortproblem $\overline{x} \in L$? im Kopf lösen, d.h. ohne Papier und Bleistift.
- Die Buchstaben aus $\overline{x}$ werden nacheinander genau einmal genannt.
- Die Antwort (akzeptiert/nicht akzeptiert) muss innerhalb von 1 Sekunde gegeben werden.

Ergebnisse:

- Schon relative kurze Worte sind schwer zu erkennen.
- Kein Kontrast zwischen Kopier- und Palindromsprache
- Kopiersprachen sind wesentlich schwieriger zu erkennen als Zählsprachen:
  - $\overline{w}\overline{w}$ versus $a^n b^n c^n d^n$
  - $\overline{w}\overline{w}^R$ versus $a^n b^n$

Outline

1. Überblick

2. Einordnung natürlicher Sprache

3. Real-Time-Turingmaschinen (RTTM)

Real-Time-Turingmaschinen (RTTM)

- “entdeckt” in den 60ern [2; 5; 6; 8]
- Turingmaschine mit einem Eingabeband (nur ein Lesekopf) und k Arbeitsbändern (mit je einem Lese-/Schreibkopf).
- Genau ein Übergang pro Eingabe-Buchstaben!
- Vergleich der umgebenden Zeitkomplexitätsklassen, wobei $n = |\overline{w}|$ und c ist eine Konstante:

LINTIME	nc
REALTIME	n
CONSTTIME	c

- **Wichtige Anpassung hier:** nur ein Arbeitsband mit einem Schreib-/Lesekopf und $k - 1$ Leseköpfen. Wir nennen diese Variante RTTM_k. Außerdem erlauben wir konstante c weitere Übergänge, um mehr Flexibilität beim Entwurf der Maschinen zu haben.

RTTM$_k$: Definition

Definition (Deterministische Turingmaschine mit k Köpfen (TM$_k$))

Eine determinische Turingmaschine mit k Köpfen auf dem Arbeitsband, oder TM$_k$, ist ein Tupel $\langle Q, E, q_0, \Sigma_I, \Sigma_W, \Box, \delta\rangle$:

- Q ist eine endliche Menge von Zuständen, $E \subseteq Q$ ist eine Menge von Endzuständen, and $q_0 \in Q$ ist der Startzustand.
- Σ_I ist das Eingabealphabet und Σ_W ist das Arbeitsalphabet. $\Box$ ist das Leerzeichen und $\Box \in \Sigma_I$ und $\Box \in \Sigma_W$.
- $\delta : (Q \times \Sigma_I \times [\Sigma_W]^k) \to (Q \times \Sigma_W \times \{L, R, N\} \times [\Sigma_W]^{k-1})$ ist die Übergangsfunktion.

Real-time TM$_k$ (RTTM$_k$)

Eine Real-Time-TM$_k$, oder RTTM$_k$, ist eine TM$_k$ die einen Inputstring der Länge n mit höchstens $c + n$ Übergängen akzeptiert, wobei c eine Konstante ist.

RTTM$_1$ für $a^n b^n c^n$

$$\mathcal{M} = \langle \{q_0, q_1, q_2, q_3, q_4, q_e\}, \{q_e\}, q_o, \{a, b, \Box\}, \{a, b, \Box\}, \Box, \delta \rangle$$

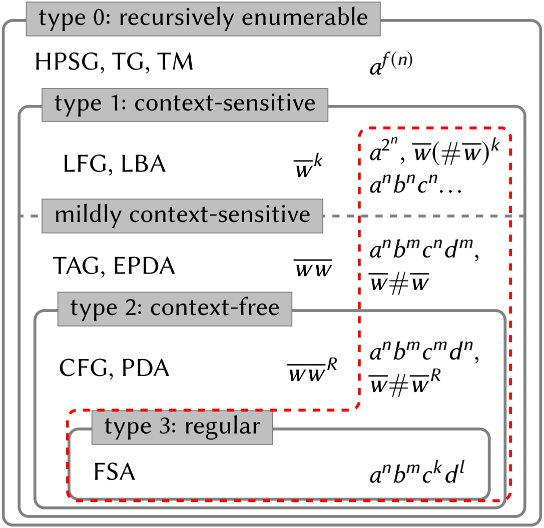
$$\begin{array}{rcl}
\delta(q_0, a, \Box) & = & (q_1, a, R) \\
\delta(q_0, \Box, \Box) & = & (q_e, \Box, N) \\[1ex]
\delta(q_1, a, \Box) & = & (q_1, a, R) \\
\delta(q_1, b, \Box) & = & (q_2, b, L) \\[1ex]
\delta(q_2, b, b) & = & (q_2, b, L) \\
\delta(q_2, c, a) & = & (q_3, c, R) \\[1ex]
\delta(q_3, c, b) & = & (q_3, c, R) \\
\delta(q_3, \Box, b) & = & (q_4, \Box, R) \\[1ex]
\delta(q_4, \Box, \Box) & = & (q_e, \Box, N)
\end{array}$$

RTTM: wichtige Lemmas

Die offene Frage: Ist eine $RTTM_k$ mit k Köpfen äquivalent mit einer RTTM mit k Arbeitsbändern?

- RTTML mit k Bändern $\subset$ RTTML mit $k+1$ Bändern[1]
- *RTTML* $\subset$ *CSL*
  - RTTM haben einen linear beschränkten Speicher ($\subseteq$ LINSPACE)
  - RTTM können nicht die Sprache $\overline{ww}$ erkennen.
- RTTML $\not\subset$ | $\not\supset$ deterministic CFL, DPDA[6]
- RTTML $\not\subset$ | $\not\supset$ CFL (wegen $\overline{ww}^R$)
- RL $\subset$ RTTML

RTTM$_k$ und die Chomsky-Hierarchie



RTTM: Abschlusseigenschaften

Rosenberg (1967): für RTTM mit mehreren Arbeitsbändern

- abgeschlossen unter Komplementation, Schnitt, und Vereinigung (Boole'sche Algebra)
- nicht abgeschlossen unter Konkatenation, Kleene'schen Operatoren, Umkehrung

RTTM$_2$ für $\overline{w}\#\overline{w}$

$$\mathcal{M} = \langle \{q_0, q_1, q_2\}, \{q_e\}, q_o, \{a, b, \Box\}, \{a, b, \Box\}, \Box, \delta \rangle$$

$$\begin{array}{rcl}
\delta(q_0, a, \Box, \Box) & = & (q_1, a, R, N) \\
\delta(q_0, b, \Box, \Box) & = & (q_1, b, R, N) \\
\delta(q_0, \#, \Box, \Box) & = & (q_e, \Box, N, N) \\
\\
\delta(q_1, a, \Box, _) & = & (q_1, a, R, N) \\
\delta(q_1, b, \Box, _) & = & (q_1, b, R, N) \\
\delta(q_1, \#, \Box, _) & = & (q_2, \Box, N, N) \\
\\
\delta(q_2, a, \Box, a) & = & (q_2, \Box, N, R) \\
\delta(q_2, b, \Box, b) & = & (q_2, \Box, N, R) \\
\delta(q_2, \Box, \Box, \Box) & = & (q_e, \Box, N, N)
\end{array}$$

LTTM_k und PTTM_k

Definition (Linear-Time-TM_k (LTTM_k))

Eine Linear-Time-TM_k, or LTTM_k, ist eine TM_k die einen Eingabestring der Länge n mit höchstens cn akzeptiert, wobei c eine Konstante ist.

Definition (Polynomial-Time-TM_k (PTTM_k))

Eine Polynomial-Time-TM_k, or PTTM_k, ist eine TM_k die einen Eingabestring der Länge n mit höchstens n^c Übergängen akzeptiert, wobei c eine Konstante ist.

Idee: Für die Vor- und Nachverarbeitung steht abhängig von der Wortlänge mehr Zeit zur Verfügung.

LTTM$_2$ für $\overline{ww}^R$

$$\mathcal{M} = \langle \{q_0, q_1, q_2, q_3, q_e\}, \{q_e\}, q_o, \{a, b, \Box\}, \{a, b, \underline{a}, \underline{b}, \Box\}, \Box, \delta \rangle$$

$$\begin{array}{rcl}
\delta(q_0, a, \Box, \Box) & = & (q_1, a, R, N) \\
\delta(q_0, b, \Box, \Box) & = & (q_1, b, R, N) \\
\delta(q_0, \Box, \Box, \Box) & = & (q_e, \Box, N, N) \\
\\
\delta(q_1, a, \Box, _) & = & (q_1, a, R, N) \\
\delta(q_1, b, \Box, _) & = & (q_1, b, R, N) \\
\delta(q_1, \Box, \Box, _) & = & (q_2, \Box, L, N) \\
\\
\delta(q_2, \Box, a, a) & = & (q_3, \underline{a}, N, R) \\
\delta(q_2, \Box, b, b) & = & (q_3, \underline{b}, N, R) \\
\\
\delta(q_3, \Box, a, a|b) & = & (q_2, a, L, N) \\
\delta(q_3, \Box, b, a|b) & = & (q_2, b, L, N) \\
\delta(q_3, \Box, a, \underline{a}) & = & (q_e, a, N, N) \\
\delta(q_3, \Box, b, \underline{b}) & = & (q_e, b, N, N)
\end{array}$$

$a^n b^n$ leichter als $\overline{w}\#\overline{w}^R$?

Idee: Zählen ist weniger aufwändig als das Merken von Teilstrings

Definition (RTTM$_k$ mit einem unveränderlichen, semi-unendlichen Arbeitsband über $\mathbb{N}$ ($\mathbb{N}$-RTTM$_k$))

Sei $\mathcal{M} = \langle Q, E, q_0, \Sigma_I, \Sigma_W, \Box, \delta \rangle$ eine RTTM$_k$. Eine RTTM$_k$ mit einem unveränderlichen, semi-unendlichen Arbeitsband $\mathbb{N}$ ($\mathbb{N}$-RTTM$_k$) kann entsprechend definiert werden als Tupel $\langle Q, E, q_0, \Sigma_I, \mathbb{N}, \Box, \delta \rangle$, mit $\delta : (Q \times \Sigma_I \times \mathbb{N}) \to (Q \times [\{L, R, N\}]^k)$.

Man kann sagen: Bei einer $\mathbb{N}$-RTTM$_k$ ist die Speicherlast konstant, nämlich k. Man muss sich nur die Position der Köpfe merken.

$\mathbb{N}$-RTTM$_1$ für $a^n b^n$

$$\mathcal{M} = \langle \{q_0, q_1, q_2, q_e\}, \{q_e\}, q_o, \{a, b, \square\}, \mathbb{N}, \square, \delta \rangle$$

$$\begin{array}{rcl} \delta(q_0, a, 0) & = & (q_1, R) \\ \delta(q_0, \square, 0) & = & (q_e, N) \\[1em] \delta(q_1, a, \_) & = & (q_1, R) \\ \delta(q_1, b, \_) & = & (q_2, L) \\[1em] \delta(q_2, b, \_) & = & (q_2, L) \\ \delta(q_2, \square, 0) & = & (q_e, N) \end{array}$$

Speicherbezogene RTTM$_k$-Komplexitätsklassen

CLASS	STRING LANGUAGE	MEMORY LOAD
LMem	$a^n b^m$, $\{a, b\}^*$	0 (regular, TM_0)
	$a^n b^n$	1 ($\mathbb{N}$-$RTTM_1$)
	$a^n b^n c^n$	2 ($\mathbb{N}$-$RTTM_2$)
	a^{2^n}	2 ($\mathbb{N}$-$RTTM_2$)
	$\{\overline{w} : \lvert a\rvert = \lvert b\rvert = \lvert c\rvert\}$ (MIX)	3 ($\mathbb{N}$-$RTTM_3$)
UMem	$\overline{w}\#\overline{w}^R$	$\lvert\overline{w}\rvert + 1$ ($RTTM_1$)
real-time	$\overline{w}\#\overline{w}$	$\lvert\overline{w}\rvert + 2$ ($RTTM_2$)
linear-time	$\overline{ww}^R$	$\lvert\overline{ww}\rvert + 2$ ($LTTM_2$)
	$\overline{ww}$	$\lvert\overline{ww}\rvert + 2$ ($LTTM_2$)
polyn.-time	$\overline{w'}\,\overline{w}\,\overline{w''}\#\overline{w}^R$	$\lvert\overline{w'}\,\overline{w}\,\overline{w''}\#\overline{w}^R\rvert + 2$ ($PTTM_2$?)

RTTM$_k$: Zusammenfassung

- Zeitkomplexität fixiert
- Komplexitätsunterschiede über Speicherauslastung (Raumkomplexität) und Anzahl der Köpfe
- quer zur Chomsky-Hierarchie
- Eingrenzung auf Sprachen aus CFL und CSL, die **zum Teil** (z.B. nicht a^{2^n}) einen direkteren Bezug zu natürlicher Sprache haben.
- weitere Einschränkungsmöglichkeiten: Gleichzeitigkeit und Richtung der Kopfbewegung (z.B. nur R), …
- vergleichbare Grammatikklasse?

Literaturverzeichnis

[1] Aanderaa, Stål O. 1974. On k-tape versus $(k-1)$-tape real time computation. In *Siam-ams proceedings. complexity of computation*, vol. 7, 75–96.

[2] Hartmanis, Juris & Richard E. Stearns. 1965. On the computational complexity of algorithms. *Transactions of the American Mathematical Society* 117. 285–306. http://www.jstor.org/stable/1994208.

[3] Joshi, Aravind K. 1985. Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions. In David Dowty, Lauri Karttunen & Arnold Zwicky (eds.), *Natural language parsing*, 206–250. Cambridge University Press.

[4] Joshi, Aravind K., K. Vijay Shanker & David Weir. 1990. *The convergence of mildly context-sensitive grammar formalisms.* CIS Technical Report MS-CIS-90-01. Philadelphia, PA: Department of Computer & Information Science, University of Pennsylvania. http://repository.upenn.edu/cis_reports/539/.

[5] Rabin, Michael O. 1963. Real time computation. *Israel Journal of Mathematics* 1. 203–211.

[6] Rosenberg, Arnold L. 1967. Real-time definable languages. *Journal of the Association for Computing Machinery* 14(4). 645–662.

[7] Turing, Alan M. 1936. On computable numbers, with an application to the Entscheidungsproblem. *Journal of Mathematics* 58(345-363). 5.

[8] Yamada, Hisao. 1962. Real-time computation and recursive functions not real-time computable. *IRE Transactions on Electronic Computers* 11(6). 753–760.