TECHNISCHE UNIVERSITÄT DRESDEN

FORMALE SYSTEME

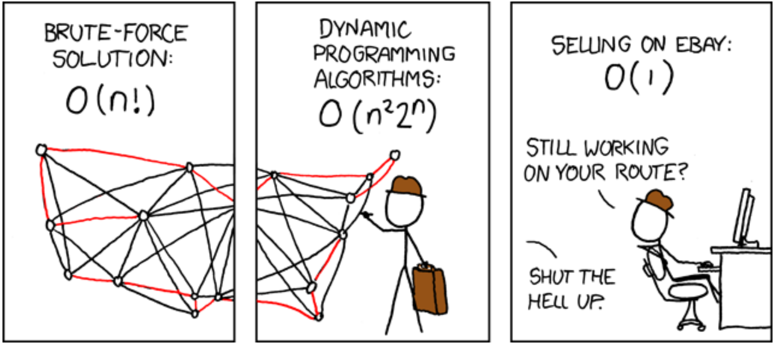
26. Vorlesung: Zusammenfassung und Ausblick

Markus Krötzsch

Professur für Wissensbasierte Systeme

TU Dresden, 25. Januar 2024

Zusammenfassung



Randall Munroe, `http://xkcd.com/399/`, CC-BY-NC 2.5

Sprachen und Berechnung

- Formale Wörter als allgemeine Abstraktion aller Daten, die in Computern verarbeitet werden können
- Formale Sprachen als Mengen von Ein- oder Ausgaben
- Worterkennung als allgemeine Berechnungsaufgabe

Beispiel: Auch die Berechnung von Funktionen kann als Wortproblem ausgedrückt werden. Anstatt zu fragen, „Was ergibt $n + m$?“ kann man fragen „Ist $n + m = r$?“

Daraus ergibt sich das Kernthema diese Vorlesung:

Sprachen zu klassifizieren heißt Rechenaufgaben klassifizieren

Zwei Hierarchien

Wir haben zwei Hierarchien von Sprachklassen kennengelernt

(1) Chomsky-Hierarchie:

$$\text{Typ 3} \subset \text{det. kontextfrei} \subset \text{Typ 2} \subset \text{Typ 1} \subset \text{Typ 0}$$

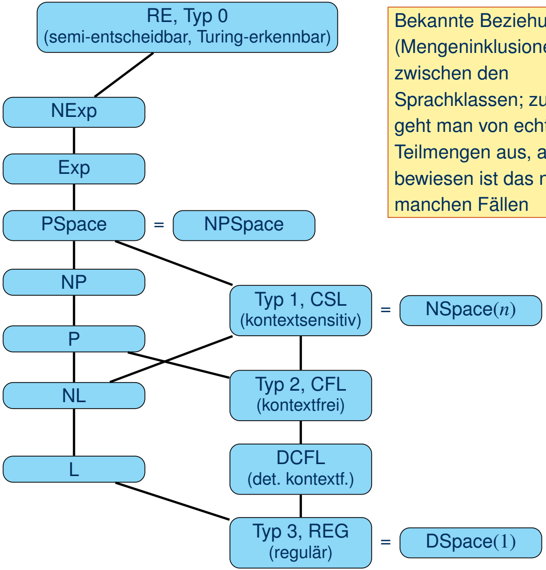
Ansatz: natürliche Definition über Grammatiken

(2) Hierarchie der Komplexitätsklassen:

$$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSpace} \subseteq \text{Exp} \subseteq \text{NExp}$$

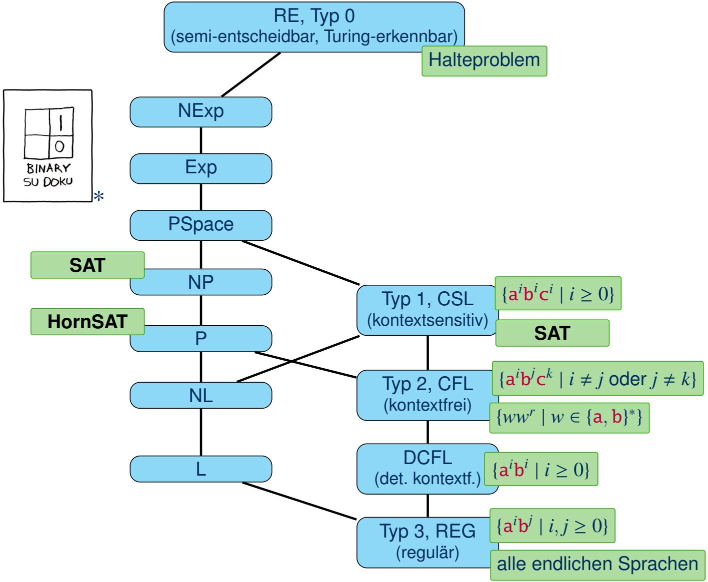
Ansatz: natürliche und robuste Definition durch Beschränkung von Turingmaschinen

Eine Hierarchie?

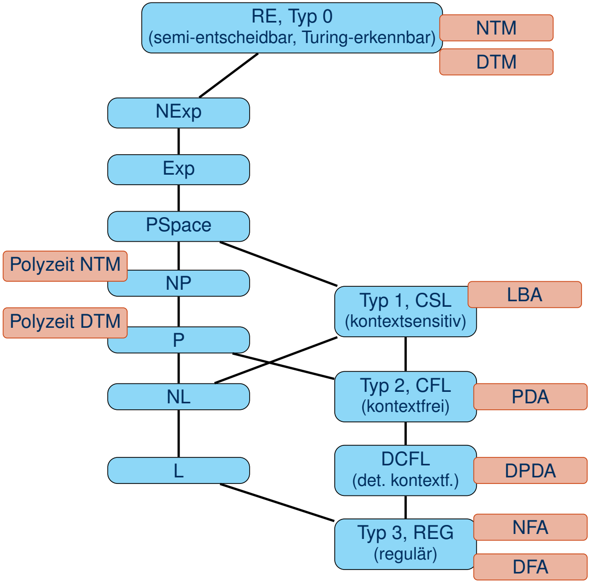


Bekannte Beziehungen (Mengeninklusionen) zwischen den Sprachklassen; zumeist geht man von echten Teilmengen aus, aber bewiesen ist das nur in manchen Fällen

Typische Beispiel-Sprachen



Berechnungsmodelle



Trennung der Sprachklassen

Die Chomsky-Hierarchie ist echt. Methoden, um Nichtenthaltensein einer Sprache in einer bestimmten Hierarchieebene zu zeigen:

- Typ 3: reguläres Pumping-Lemma, Myhill-Nerode-Index, Abschlusseigenschaften (V10)
- Typ 2: kontextfreies Pumping-Lemma (V13), Abschlusseigenschaften (V14)
- det. Typ 2: Abschlusseigenschaften (V16)
- Typ 1: Entscheidbarkeit, Abschlusseigenschaften (V19/V20)
- Typ 0: Semi-Entscheidbarkeit (V19/V20)

Bei den Komplexitätsklassen sind bisher weitaus weniger Unterschiede bewiesen. Exponentielle Ressourcenzugaben erzeugen echt größere Klassen (z.B. P $\subset$ Exp).

Aus einem analogen Ergebnis für Speicher folgt auch, dass ExpSpace-harte Sprachen nicht kontextsensitiv sind, auch wenn sie entscheidbar sind

Übersicht Abschlusseigenschaften

Sprache	Abschluss unter … $\cap$	$\cup$	$\overline{}$	$\circ$	$*$	Automat
Typ 0	✓	✓	×	✓	✓	TM (DTM/NTM)
Typ 1	✓	✓	✓	✓	✓	LBA ($\stackrel{?}{=}$ det. LBA)
Typ 2	×	✓	×	✓	✓	PDA
Det. Typ 2	×	×	✓	×	×	DPDA
Typ 3	✓	✓	✓	✓	✓	DFA/NFA

Übersicht Abschlusseigenschaften

Sprache	Ergebnis von Typ … $\cap$	$\cup$	$\overline{}$	$\circ$	$*$	Automat
Typ 0	0	0	–	0	0	TM (DTM/NTM)
Typ 1	1	1	1	1	1	LBA ($\stackrel{?}{=}$ det. LBA)
Typ 2	1	2	1	2	2	PDA
Det. Typ 2	1	2	D2	2	2	DPDA
Typ 3	3	3	3	3	3	DFA/NFA

Übersicht Probleme

Die Entscheidbarkeit verschiedener relevanter Probleme ist je nach Sprachklasse unterschiedlich:

Sprache	Wortproblem	Leerheit	Äquivalenz	Regularität	Inklusion	Schnitt
Typ 0	×	×	×	×	×	×
Typ 1	✓	×	×	×	×	×
Typ 2	✓	✓	×	×	×	×
Det. Typ 2	✓	✓	✓	✓	×	×
Typ 3	✓	✓	✓	(✓)	✓	✓

Wortprobleme lösen

Wie schwer ist es, dass Wortproblem zu lösen, wenn die Eingabe eine (geeignet kodierte) Sprache und ein zu testendes Wort ist?

Sprache	Zeitkomplexität bzgl. Wortlänge n
Typ 3	$O(n)$ (Abarbeitung DFA)
Det. Typ 2	$O(n)$ (Abarbeitung DPDA)
Typ 2	$O(n^3)$ (CYK-Algorithmus)
P	polynomiell (z.B. Hyperresolution für Hornlogik)
NP	exponentiell (z.B. Resolution allgemein)
Typ 1	$O(n \cdot \|\Gamma\|^n)$ (z.B. über LBA-Konfigurationsgraph)
Typ 0	unentscheidbar

Das Wortproblem für NP und Typ 1 ist NP-vollständig bzw. PSpace-vollständig. Es ist nicht bewiesen aber wahrscheinlich, dass es keine subexponentiellen Algorithmen gibt.

Nichtdeterminismus

Nichtdeterministische Akzeptanzbedingung:
Gibt es mindestens einen Lauf, der akzeptiert?

Det. Automatenmodell	Nichtdet. Automatenmodell	äquivalent?
DFA	NFA	✓
DPDA	PDA	×
DLBA	LBA	?
DTM	NTM	✓

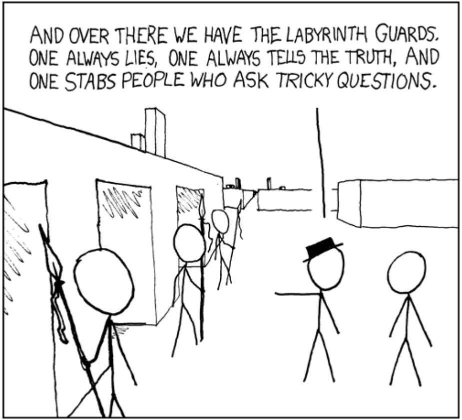
- Oft ist es schwierig, Nichtdeterminismus unter Ressourcenbeschränkungen aufzulösen, nicht nur bei LBAs
  z.B. ist auch $P \neq NP$ offen
- Wegen der Asymmetrie hat jede nichtdeterministische Klasse eine Komplementärklasse, die oft (vermutlich) unterschiedlich ist (z.B. coNP vs. NP)

Ein einfacher Beweis für P = NP ;-)

Wir wissen:	$\mathbf{L} \in P$	impliziert	$\mathbf{L} \in NP$
Daher gilt:	$\mathbf{L} \notin NP$	impliziert	$\mathbf{L} \notin P$
Anders gesagt:	$\mathbf{L} \in coNP$	impliziert	$\mathbf{L} \in coP$
Das heißt:	$coNP \subseteq coP$		
Wegen $coP = P$ gilt:	$coNP \subseteq P$		
Somit gilt:	$NP \subseteq P$		
D.h., wegen $P \subseteq NP$ gilt:	$NP = P$		

q.e.d.?

Ausblick und Anwendungen



Formale Sprachen

Formale Sprachen in der Praxis:

- Typ 3: extrem weit verbreitet in Form von regulären Ausdrücken; im Kompilerbau als Lexer; noch einfachere Sprachen bei Anfrage/Auswahlmechanismen z.B. CSS-Selektoren
- det. Typ 2: besonders relevant im Kompilerbau (LR(k)-Grammatiken)
- nichtdet. Typ 2: in der Sprachverarbeitung; in dieser Anwendung teils auch etwas stärkere Sprachklassen (z.B. Tree-Adjoining Grammars)

Typ-1-Sprachen haben kaum praktische Anwendungen, Typ-0-Sprachen fallen mit allgemeinen TMs zusammen

Automatentheorie

Es gibt viele Automatenmodelle jenseits der hier vorgestellten:

- Baumautomaten arbeiten auf Baumstrukturen, die sie von oben oder unten her lesen
- Automaten für unendliche Strukturen verwenden andere Akzeptanzbedingungen, die für unendliche Abarbeitungen Sinn ergeben
- Hybride Automaten modellieren komplexe dynamische Systeme mithilfe von Differentialgleichungen
- Eingeschränkte Automatenmodelle z.B. partiell geordnete Automaten, erkennen spezielle reguläre Sprachen
- …

Wesentliche Anwendungen von Automaten:

- Definition „interessanter“ Sprachklassen
- Lösung algorithmischer Probleme (z.B. Inklusionstest von Sprachen)

Logik

Aussagenlogik ist nur der Anfang . . .

- Prädikatenlogik/Logik erster Stufe erweitert die Struktur atomarer Aussagen (Prädikate, Terme, Variablen, . . .); Quantoren $\forall$ und $\exists$ ermöglichen es, sich auf viele Aussagen zu beziehen ohne alle einzeln zu nennen
- Logik zweiter Stufe führt zudem Variablen für Prädikate und entsprechende Quantoren ein

$\leadsto$ Ausgangspunkt vieler anwendungsspezifischer Logiken

Wesentliche Anwendungen:

- Wissensrepräsentation
- Logikprogrammierung
- Constraint-Erfüllungsprobleme
- Verifikation

Logisches Schließen

Großes Fachgebiet; sehr stark anwendungsspezifisch

Nennenswerte Klassen stark optimierter Logiktools:

- SAT-Solver: aussagenlogisches Schließen
- Theorembeweiser: Entwickeln formaler Beweise in sehr ausdrucksstarken Logiken
- Model-Checker: effiziente Verifikation von formalen Aussagen bzgl. abstrakter Programmmodelle
- Logikprogramm-Systeme: Berechnung der Ergebnisse logischer Programme verschiedenster Form
- Ontologie-Reasoner: Anfragebeantwortung über logischen Wissensbasen und Datenbanken

Komplexitätstheorie

Großes Gebiet in der theoretischen Informatik, mit zwei wesentlichen Bedeutungen:

1. Eigenständiges Forschungsgebiet, das sich vielen grundlegenden Fragen widmet (einschl. $P \neq NP?$); Theorie der Kryptographie; Quantenkomplexität
2. Methoden für andere Forschungsfelder, welche die komplexitätstheoretische Analyse von Problemen in vielen Fachgebieten ermöglichen; Themen wie parametrisierte Komplexität oder Ausgabekomplexität sind aus Anwendungen motiviert

Weiteres Fachgebiet: Berechenbarkeitstheorie (Klassifikation unentscheidbarer Probleme, alternative Berechnungsmodelle)

TheoLog

Im nächsten Semester gibt es „Theoretische Informatik und Logik“ (Pflicht für einige, offen für alle)

Hören Sie die großen Fragen der Mathematik – und die Antworten der Informatik!

- Berechenbarkeit
  - Fleißige Biber und manch Unentscheidbares
  - Von Turingmaschinen zu Programmen
- Komplexität
  - NP: Spiele für eine Person
  - PSpace: einfache Spiele für zwei
- Prädikatenlogik
  - Die Sprache der Mathematik
  - Resolution reloaded
  - Endliche Modelle (besser bekannt als “Datenbanken”)
- Mathematiker:innen als Programmierer:innen
  - Die Grenzen der Mathematik
  - Gödel, Turing und der ganze Rest

Zusammenfassung

Formale Sprachen sind die Grundlage zahlreicher Forschungs- und Anwendungsfelder der Informatik.

Berechnungsmodelle erlauben uns, allgemeine Aussagen über die Schwere und Lösbarkeit von Berechnungsaufgaben zu treffen

Formale Logik wird als Spezifikationssprache für (zumeist anspruchsvolle) Probleme in vielen Gebieten verwendet

Offene Fragen:

- Haben Sie noch inhaltliche Fragen? ($\leadsto$ Konsultationen und zusätzlichen Übungstermin nutzen)
- Haben Sie sich ausreichend auf die Prüfung vorbereitet?