Notizen zur Vorlesung

Theoretische Informatik und Logik

Zusammenfassung

Notizen zur Vorlesung https://iccl.inf.tu-dresden.de/web/Theoretische_Informatik_und_Logik_(SS2017).

Ich versuche möglichst ohne formelle Symbole und Definitionen zu arbeiten, daher verweisen die Markierungen jeweils auf die Vorlesungsnummer in FS bzw. TIL. Obwohl der Schwerpunkt auf TheoLog liegt, habe ich ein paar Definitionen aus Formale Systeme mit einbezogen, da TheoLog diese weiterverwendet.

Einige Formulierungen habe ich aus den hervorragenden Folien von Prof. Krötzsch geliehen. Quellen dieser Folien sind auf Github zu finden unter https://github.com/mkroetzsch und sind unter der Lizenz CC BY 3.0 DE verwendbar. Für diese gilt: „(C) Markus Krötzsch, https://iccl.inf.tu-dresden.de/web/TheoLog2017, CC BY 3.0 DE".

Inhaltsverzeichnis

Autor	Dominik Pataky
Dozent	Prof. Krötzsch
Ort	Fakultät Informatik, TU Dresden
Zeit	Sommersemester 2017
Letztes Update	2. Mai 2017
Lizenz	CC BY-SA 4.0

1 Formale Systeme

1.1 Sprachen und Automaten

(formale) Sprache Menge von Wörtern/Symbolen/Tokens, z.B. Programmiercode oder natürliche Sprache. Zusätzliche Begriffe: Konkatenation, Präfix/Suffix/Infix, leeres Wort FS 1

Symbol Token der Sprache, z.B. if/else, +/-, True/False, "Hello World"-String

Alphabet nichtleere, endliche Menge von Symbolen

Wort endliche Sequenz von Symbolen

Grammatik formelle Spezifikation einer Sprache. Aus einer Grammatik kann man widerum eine Sprache erzeugen FS 2

Rechenoperationen Vereinigung, Schnitt, Komplement, Produkt, Potenz, Kleene-Abschluss

Abschlusseigenschaft Beispiel: Wenn Sprache A und Sprache B regulär sind, wäre dann auch der Schnitt der beiden Sprachen wieder regulär? FS 5

Automat Beginnt von einem Startzustand und folgt je nach Eingabe seinen Übergängen in die jeweiligen Zustände. Akzeptiert, wenn letzter Zustand ein akzeptierter Endzustand.

Deterministischer endlicher Automat (DFA) erkennen reguläre Sprachen FS 3

nichtdeterministischer endlicher Automat (NFA) „rät" die richtigen Übergänge, arbeitet parallel. Nichtdeterminismus sinnvoll? Kompaktere Darstellungen, Start für Entwicklung DFA, kann bei Untersuchung Komplexität/Berechenbarkeit helfen FS 4

Kellerautomat (PDA) PDA erweitern endliche Automaten um einen unbeschränkt großen Speicher, der aber nur nach dem LIFO-Prinzip verwendet werden kann. PDAs erkennen genau die kontextfreien Sprachen. FS 15

Turingmaschine (TM) liefert allgemeines Modell der Berechnung. Liest und schreibt in einem Schritt, hat unendlichen Speicher, kann beliebig auf Speicher zugreifen (im Gegensatz zu LIFO bei PDA). Kann ein Band oder mehrere Bänder haben. Kann deterministisch (DTM) oder nichtdeterministisch (NTM) sein. Alle Varianten der TM können die selben Funktionen berechnen - einzig der Aufwand ist unterschiedlich (NTM kann DTM darstellen, NTM kann durch DTM simuliert werden etc.). Siehe auch Church-Turing-These. FS 18

Kardinalität Unterscheidung abzählbar (mit natürlichen Zahlen) und überabzählbar

Chomsky-Hierarchie Kategorische Einteilung von Sprachen je nach Komplexität ihrer Grammatik. Hierarchie 0 > 1 > 2 > 3. These: „Die meisten Sprachen können nicht mit Grammatiken beschrieben werden (abzählbar viele Grammatiken vs. überabzählbar viele Sprachen)". FS 2

Typ 0 beliebige Grammatiken (Turingmaschinen)

Typ 1 kontextsensitive Grammatiken

Typ 2 kontextfreie Grammatiken (CYK, Kellerautomaten)

Typ 3 reguläre Grammatiken (DFA, NFA, Pumping Lemma)

Probleme Probleme formulieren Berechnungsfragen.

Wortproblem Wortproblem für eine Sprache über einem Alphabet ist die Berechnung der Ausgabe „ja, Wort ist in Sprache" oder „nein, Wort ist nicht in Sprache", für die Eingabe eines Wortes gebildet aus dem Alphabet FS 3

Leerheitsproblem (DFA, NFA) Entscheidung für „ja, Automat erzeugt Sprache" oder „nein, durch den Automaten erzeugte Sprache ist leer" (es wird nie ein Endzustand erreicht). FS 10

Inklusionsproblem (DFA, NFA) Entscheidung für „ja, Sprache A eines Automaten ist eine Teilmenge der Sprache B eines anderen Automaten" oder „nein, Sprache A ist keine Teilmenge der Sprache B". FS 10

Äquivalenzproblem (DFA, NFA) Entscheidung für „ja, Sprache A eines Automaten ist gleich der Sprache B eines anderen Automaten" oder „nein, Sprache A unterscheidet sich von Sprache B". FS 10

Endlichkeitsproblem (DFA, NFA) Entscheidung für „ja, erzeugte Sprache eines Automaten ist endlich" oder „nein, erzeugte Sprache ist nicht endlich" (z.B., wenn Zyklen auf dem Pfad von Start- zu Endzustand existieren). **FS 10**

Universalitätsproblem (DFA, NFA) Entscheidung für „ja, erzeugte Sprache eines Automaten ist Σ^*" oder „nein, erzeugte Sprache ist nicht Σ^*" (heißt, Komplement der erzeugten Sprache ist leer). **FS 10**

Halteproblem (TM) Entscheidet, ob eine Turingmaschine für eine Eingabe jemals hält oder nicht. Unentscheidbar. **FS 19**

Church-Turing-These Die Turingmaschine kann alle Funktionen berechnen, die intuitiv berechenbar sind. Auch: „Eine Funktion ist genau dann im intuitiven Sinne berechenbar, wenn es eine Turingmaschine gibt, die für jede mögliche Eingabe den Wert der Funktion auf das Band schreibt und anschließend hält." **FS 18**

Entscheidbarkeit Eine Sprache L ist entscheidbar / berechenbar / rekursiv, wenn es eine Turingmaschine gibt, die das Wortproblem der Sprache L entscheidet. D.h. die Turingmaschine ist Entscheider und die Sprache L ist gleich der durch die Turingmaschine erzeugten Sprache. Andernfalls heißt die Sprache L unentscheidbar.
Die Sprache L ist semi-entscheidbar / Turing-erkennbar / rekursiv aufzählbar, wenn es eine Turingmaschine gibt, deren erzeugte Sprache zwar L ist, jedoch die Turingmaschine kein Entscheider ist. **FS 19**

Satz von Rice Sei E eine Eigenschaft von Sprachen, die für manche Turing-erkennbare Sprachen gilt und für manche Turing-erkennbare Sprachen nicht gilt (= „nicht-triviale Eigenschaft").
Dann ist das folgende Problem unentscheidbar: die Eingabe besteht aus einer Turingmaschine. Anhand der Ausgabe wollen wir prüfen, ob die durch diese Turingmaschine erzeugte Sprache die Eigenschaft E besitzt. **FS 20**
Der Beweis für die Unentscheidbarkeit dieses Problems ist eine Reduktion auf das Halteproblem.

1.2 Aussagenlogik

Die Aussagenlogik untersucht logische Verknüpfungen von atomaren Aussagen. FS 21

Atomare Aussage Behauptungen, die wahr oder falsch sein können.
Auch: aussagenlogische Variablen, Propositionen, Atome

Operatoren, Junktoren Verknüpfung von atomaren Aussagen.
Negation, Konjunktion, Disjunktion, Implikation, Äquivalenz.
Können auch äquivalent durch andere Junktoren ausgedrückt werden (de Morgan). FS 22
Eine Disjunktion von Literalen nennt man **Klausel**.
Eine Konjunktion von Literalen nennt man **Monom**.

Formel Jedes Atom ist eine Formel, jede durch Junktoren verknüpfte Formeln sind wieder Formeln. Diese zusammengesetzten Formeln bestehen dann wieder aus Teilformeln (auch: Unterformeln, $Sub(F)$). Eine Formel, die nur aus einem Atom besteht, nennt man auch **Literal**.

unerfüllbar Formel hat keine Modelle

erfüllbar Formel hat mindestens ein Modell

allgemeingültig alle Interpretationen sind Modelle für Formel. Auch: **Tautologie**, $\models F$

widerlegbar Formel ist nicht allgemeingültig

Syntax Sprache einer Logik (Formeln mit logischen Operatoren). Wichtig: Klammerung.

Semantik Definition der Bedeutung. Wertzuweisung von Wahrheitswerten zu Atomen mit Hilfe der Interpretation. „Die Bedeutung einer Formel besteht darin, dass sie uns Informationen darüber liefert, welche Wertzuweisungen möglich sind, wenn die Formel wahr sein soll.“

Interpretation eine Funktion w, die von einer Menge Atome auf die Menge {0, 1} abbildet.

Wahrheitstabelle Schrittweise Auflösung einer Formel durch Lösen ihrer Teilformeln.

Modell eine Interpretation, dessen Abbildung eine Formel nach 1 löst.

Logische Konsequenz eine Formel G ist eine logische Konsequenz einer Formel F ($F \models G$), wenn jedes Modell von F auch ein Modell für G ist.

Logische Äquivalenz zwei Formeln F und G sind semantisch äquivalent ($F \equiv G$), wenn sie genau dieselben Modelle haben FS 22

Normalform jede Formel lässt sich in eine äquivalente Formel in Normalform umformen.
Für die Umformungen gibt es Algorithmen, siehe FS 22

Negationsnormalform (NNF) enthält nur UND, ODER und Negation, wobei Negation nur direkt vor Atomen vorkommt.

Konjunktive Normalform (KNF) Formel ist eine Konjunktion von Disjunktionen von Literalen.

Disjunktive Normalform (DNF) Disjunktion von Konjunktionen von Literalen.

1.3 Komplexität

Turingmaschinen sind begrenzt durch die Anzahl ihrer Speicherzellen (Speicher) und der Anzahl möglicher Berechnungsschritte (Zeit). Schranken sind Funktionen gerichtet nach der Länge der Angabe. FS 24

O-Notation charakterisiert Funktionen nach ihrem Verhalten und versteckt lineare Faktoren.

$O(f)$-zeitbeschränkt es gibt eine Funktion, so dass eine DTM bei beliebiger Eingabe nach einer maximalen Anzahl Schritte anhält

$O(f)$-speicherbeschränkt es gibt eine Funktion, os dass eine DTM bei beliebiger Eingabe nur eine maximale Anzahl Speicherzellen verwendet

Sprachklassen Einteilung von Sprachen nach der Möglichkeit der Entscheidbarkeit.
„Klasse aller Sprachen, welche…“

- **DTIME**$f(n)$ …durch eine $O(f)$-zeitbeschränkte DTM entschieden werden können
- **DSPACE**$f(n)$ …durch eine $O(f)$-speicherbeschränkte DTM entschieden werden können
- **NTIME**$f(n)$ …durch eine $O(f)$-zeitbeschränkte NTM entschieden werden können
- **NSPACE**$f(n)$ …durch eine $O(f)$-speicherbeschränkte NTM entschieden werden können

Komplexitätsklassen erfassen Sprachklassen je nach ihrer Komplexität. Stehen untereinander in Beziehung und bilden quasi Hierarchie. FS 24

- **PTime (P)** deterministisch, polynomielle Zeit
- **ExpTime (Exp)** deterministisch, exponentielle Zeit
- **LogSpace (L)** deterministisch, logarithmischer Speicher
- **PSpace** deterministisch, polynomieller Speicher
- **NPTime (NP)** nichtdeterministisch, polynomielle Zeit
- **NEexpTime (NExp)** nichtdeterministisch, exponentielle Zeit
- **NLogSpace (NL)** nichtdeterministisch, logarithmischer Speicher
- **NPSpace** nichtdeterministisch, polynomieller Speicher

SAT Boolean Satisfiability Problem. Problem, welches nach einem Modell für eine Formel sucht. In NP. Interessant für Untersuchung, da SAT ein Problem darstellt, für welches es schwierig ist eine Lösung zu finden, jedoch sehr einfach ist eine Lösung auf Korrektheit zu prüfen. FS 25

Reduktion Rückführung eines Problems auf ein anderes. Beispiel Drei-Farben-Problem ist auf SAT reduzierbar, da sich die Farb-Zustände als Formeln ausdrücken kodieren lassen. „Alle Probleme in NP können polynomiell auf SAT reduziert werden“ (**Cook, Levin**)

Härte und Vollständigkeit für P und NP FS 25

- **NP-hart** Sprache ist NP-hart, wenn jede Sprache in NP polynomiell darauf reduzierbar ist (Beispiel Halteproblem und jedes weitere unentscheidbare Problem).
- **NP-vollständig** Sprache ist NP-hart und liegt selbst in NP (Beispiel SAT).
- **P-hart** Sprache ist P-hart, wenn jede Sprache in P mit logarithmischem Speicherbedarf auf diese reduzierbar ist.
- **P-vollständig** Sprache ist P-hart und liegt selbst in P (Beispiel HornSAT).

Zusammenfassung aller Themenkomplexe, Hierarchien und Zusammenhänge in FS 26.

2 Theoretische Informatik

Turingmaschine DTM und NTM, besteht aus Tupel M mit (endlicher Menge von Zuständen, Eingabealphabet, Arbeitsalphabet, Übergangsfunktion, Startzustand und Menge von akzeptierenden Endzuständen). TIL 1

- **Konfiguration** der „Gesamtzustand" einer TM, bestehend aus Zustand, Bandinhalt und Position des Lese-/Schreibkopfs; geschrieben als Wort (Bandinhalt), in dem der Zustand vor der Position des Kopfes eingefügt ist
- **Übergangsrelation** Beziehung zwischen zwei Konfigurationen wenn die TM von der ersten in die zweite übergehen kann (deterministisch oder nichtdeterministisch)
- **Lauf** mögliche Abfolge von Konfigurationen einer TM, beginnend mit der Startkonfiguration; kann endlich oder unendlich sein
- **Halten** Ende der Abarbeitung, wenn die TM in einer Konfiguration keinen Übergang mehr zur Verfügung hat