Cheat Sheet für Berechenbarkeit und Formale Sprachen

15. Februar 2021

Verifizierer

Verifizierer verifizieren Sprachen und entscheiden sie nicht!
Ein Verifizierer braucht zusätzlich zu seiner Eingabe einen Beweis. Um daraus einen Entscheider zu generieren könnte folgendermaßen vorgegangen werden: Es können alle möglichen Beweise ausprobiert werden und jeweils mit dem Verifizierer getestet werden, ob der aktuelle Beweis gültig für die gegebene Eingabe ist. Falls das für **einen** Beweis gilt, so wird die Eingabe akzeptiert, falls es für **keinen** Beweis gilt, so wird die Eingabe verworfen. Bei diesem Vorgehen muss jedoch sicher gestellt werden, dass die Anzahl der möglichen Beweise endlich ist. Für Entscheider in Polynomzeit muss sowohl der Verifizierer polynomiell zeitbeschränkt sein, als auch die Anzahl der möglichen Beweise dürfen nur polynomiell viele sein.

Nichtdeterminismus

Aus nichtdeterministischen Turingmaschinen kann man nur einen (deterministischen) Entscheider für die von der NTM akzeptierten Sprache erzeugen, falls die NTM beschränkte Laufzeit hat. Ansonsten akzeptiert die entsprechende deterministische Turingmaschine die entsprechende Sprache nur (vergleiche mit Sprache DioGl $= \{\langle p \rangle \mid p$ ist Polynom (auch über mehreren variablen) mit ganzzahligen Koeffizienten und ganzzahligen Nullstellen$\}$).

Nichtdeterministische Turingmaschinen (NTMs) / Nichtdeterministische Automaten (NFAs) / oder nichtdeterministische Kellerautomaten (NPDAs) entscheiden keine Sprachen. Sie akzeptieren sie nur. Nur deterministische Automaten oder Turingmaschinen *können* Sprachen entscheiden.

Reduktionen

Sprachen können entschieden werden - Sprachen können aber i.A. nicht berechnet werden.
Funktionen können berechnet werden - Funktionen können aber nicht entschieden werden.

Gödelnummern von Turingmaschinen können nicht gestartet werden, sondern nur die Turingmaschinen die sie repräsentieren. Turingmaschinen können (in einer Funktion) nicht berechnet werden, sondern nur deren Gödelnummern.

NOGOs	Richtig	Akzeptabel
$f(x)$ hält	$f(x) = \langle FM... \rangle$ und $FM...$ gestartet mit beliebiger Eingabe hält	siehe „Richtig“
M hält immer/M hält	M gestartet mit beliebiger Eingabe hält	M hält für jede Eingabe
M hält nie/M hält nicht	M gestartet mit beliebiger Eingabe hält nicht	M hält für keine Eingabe
M mit w hält	M gestartet mit Eingabe w hält	M gestartet mit w hält
$FM...$ bleibt in Zeile 3 hängen	$FM...$ gestartet mit beliebiger Eingabe kehrt nicht aus Zeile 3 zurück	siehe „Richtig“

Syntaxanalyse ist nicht das Allheilmittel für Berechenbarkeit der Reduktionsfunktion - insbesondere wenn in der Fallunterscheidung steht $x \in SAT$ oder eine andere Sprache.

Die Datentypen der Reduktionsfunktion müssen stimmen. Wenn vom Halteproblem H aus reduziert wird ($H \leq L$) ist davon auszugehen, dass in der Fallunterscheidung geprüft werden muss ob $x = \langle M \rangle w$ also ob x von der Form $\langle M \rangle w$ ist (das geht mit Syntaxanalyse). Der Datentyp des Funktionswertes sollte im Positivfall auf jeden Fall mit dem Datentyp der Zielsprache übereinstimmen. Wenn zum Beispiel eine Gödelnummer samt Eingabe gefordert ist sollte auch die Eingabe nicht vergessen werden.

Wenn man schreibt $x \in \langle M \rangle w$ so ist das falsch, denn „$\in$“ bei einem String-Vergleich ist sinnlos - „$\in$“ kann nur benutzt werden wenn rechts eine Menge steht.

Sollten $\langle M \rangle$ und/oder w in beiden Sprachen (Bild- und Urbildsprache) vorkommen so sollte man dafür unterschiedliche Bezeichnungen benutzen. Es ist insgesamt fatal Variablen in einer Aufgabe mit unterschiedlichen Bedeutungen zu versehen (z.B. i bei Pumpeigenschaft).

n muss nicht immer die Eingabelänge sein. $\Rightarrow$ Das heißt man muss ggf. mit $O(poly(|x|))$ anstelle von $O(poly(n))$ argumentieren.

$O(n!)$ ist nicht polynomiell.

Wenn eine Sprache $L \in P$ ist und man dies zeigen will (z.B. $L = \{bin(a)\#bin(b) \mid a, b \in \mathbb{N}\}$) muss bei Turingmaschinen und auch bei Registermaschinenprogrammen die L entscheiden sollen erst ein Syntaxcheck durchgeführt werden, da ansonsten $bin(a)$ nicht mit $bin(b)$ verglichen werden kann weil es möglicherweise gar nicht da ist.

Automaten

Für einen deterministischen Automaten gilt immer, dass es keine ε-Übergänge gibt und jeder Zustand pro Zeichen maximal einen Übergang haben kann.

Beim durch das Potenzmengenverfahren erzeugten deterministischen Automaten gibt es je Zustand *immer genau* einen Übergang je Zeichen in Σ. Dies kann dazu führen, dass $\emptyset$ ein Zustand des neuen Automaten darstellt (dieser wird sehr gerne vergessen) und auch dort je Zeichen aus Σ ein Übergang existieren muss. Es gilt $\{\emptyset\}$ ist **nicht** die leere Menge sondern $\emptyset$ oder $\{\}$.

Pump-Eigenschaft

n_L ist immer *echt* größer als das kürzeste Wort in L. Wenn man behauptet dass irgendetwas (z. B. z) beliebig aber fest sein soll, dann darf man es nicht danach fest setzen (z.B. bei z durch feste Belegung von u, v, w, welche nicht zu jedem beliebigen z passt). Warum $uv^iw \in L$ oder $uv^iw \notin L$ muss begründet werden (auch wenn es trivial scheint). Bei Negation der Pump-Eigenschaft muss das festgelegte gewählte Wort immer in Abhängigkeit von n_L sein - sonst kann das nicht funktionieren. Bei $i = 1$ kann es nicht sein, dass $uv^iw \notin L$.

Allgemeines

Wenn beispielsweise eine Grammatik angegeben werden soll so muss das komplette Tupel also $G = (V, \Sigma, P, S)$ angegeben werden und falls noch nicht definiert auch jedes Element in diesem Tupel. Selbiges gilt für formale Beschreibungen von Turingmaschinen, Automaten, Kellerautomaten, ... Einzige Ausnahme bildet die graphische Darstellung von Automaten - dabei muss jedoch Startzustand sowie Endzustände und übrige Teile gemäß Notation der Vorlesung dargestellt sein.

Wenn ein Verfahren nach oder aus der Vorlesung umgesetzt werden muss, so muss das Verfahren auch erkennbar sein. Z.B. beim Potenzmengenverfahren sollten die Zustände des Potenzmengenautomaten Teilmengen

der ursprünglichen Zustandsmenge sein, beim Entfernen von ε-Übergängen sollte mindestens die Menge der ε-Erzeuger angegeben werden und dann die vollständige Grammatik (also Tupel usw.) auch wenn sich nur die Produktionen ändern, usw. .

Es sollte hingeschrieben werden was gemacht wird. Z.B. bei Reduktionen was auf was reduziert wird. Am Ende sollte eine entsprechende Folgerung stehen.