Turingmaschinen, Typ-0 und entscheidbare Sprachen

Im Beweis des Satzes von Kuroda sehen wir bei der Konstruktion der TM zu einer gegebenen Grammatik G, dass die Maschine im Fall $x \in L(G)$ auf mindestens einem Berechnungsweg akzeptiert. Im Fall $x \notin L(G)$ verlangen wir lediglich, dass es *keinen* akzeptierenden Berechnungsweg gibt.

Es wird ausdrücklich nicht verlangt, dass ein Berechnungsweg für sich allein feststellen kann, dass der Input nicht zu $L(G)$ gehört.

Daher sagen wir, dass die deterministische Turingmaschine M die Sprache L *akzeptiert,* wenn sie auf Eingaben aus L nach endlicher Zeit stoppt, bei Eingaben, die nicht zu L gehören, aber unendlich lange rechnet. Analog für nichtdeterministische Maschinen.

Dagegen *entscheidet* die deterministische Turingmaschine M die Sprache L, wenn sie in jedem Fall stoppt und JA ausgibt, wenn die Eingabe zu L gehört; NEIN, wenn die Eingabe nicht zu L gehört.

REC und r.e.

Nur mit den eben gegebenen Definitionen gilt also der Satz von Folie 37.1. Die Typ-0 Sprachen heißen auch *rekursiv aufzählbar*, auf englisch *recursively enumerable*, deshalb kürzt man die entsprechende Sprachklasse mit r.e. ab. Die Klasse der entscheidbaren Sprachen wird REC (für *recursive*) genannt.
Es gilt damit:

$$LBA = CSL = \text{Typ-1} \quad \subseteq \quad REC \quad \subseteq \quad r.e. = \text{Typ-0}$$

Wir möchten nun noch für die (unbeschränkten oder beschränkten) Turingmaschinen zwei Fragestellungen untersuchen, die sich quasi aufdrängen:

1. Macht es einen Unterschied, ob man deterministische oder nichtdeterministische Maschinen verwendet?
2. Sind die entsprechenden Klassen abgeschlossen gegen Komplementbildung?

Det. vs. Nichtdet. und Komplementabschluss

Für die allgemeinen (akzeptierenden) Turingmaschinen kann man die erste Frage beantworten: Jede nichtdeterministische TM kann so durch eine deterministische TM simuliert werden, dass die deterministische Maschine genau dann stoppt, wenn es bei der nichtdeterministischen Maschine mindestens einen akzeptierenden Berechnungsweg gibt. Trick: Breitensuche im Berechnungsbaum.

Für LBAs ist das eine offene Frage, genannt *LBA-Problem.* Mit der naheliegenden Bezeichnung *DLBA* ergibt sich also folgende offene Frage:

Gilt $LBA = DLBA$?

Und nun zum Komplementabschluss: Es ist eine bekannte Tatsache aus der Berechenbarkeitstheorie, dass Typ-0 NICHT abgeschlossen ist gegen Komplementbildung. Aber wie sieht es bei Typ-1 aus?

Satz von Immerman und Szelepcsényi

Vor fast 30 Jahren wurde die Frage nach der Abgeschlossenheit der Klasse CSL=Typ-1 gegen Komplementbildung von zwei Autoren gleichzeitig gelöst:

Satz von Immerman und Szelepcsényi:

Die Klasse der Typ-1 Sprachen ist abgeschlossen gegen Komplementbildung.

Tatsächlich haben beide das etwas allgemeinere Resultat erzielt, dass alle platzbeschränkten nichtdeterministischen Komplexitätsklassen, benannt mit $NSPACE(s)$ für geeignete Platzschranke s, komplementabgeschlossen sind.

Wo ist das Problem?

Warum ist es schwierig, mit einem LBA herauszufinden, dass ein Wort NICHT zu $L(G)$ für gegebene Typ-1 Grammatik G gehört?

Ein Gedankenspiel: Angenommen, man wüsste, wieviele Satzformen der Länge n mit unserer Grammatik G erzeugt werden können. Dann könnte man versuchen, diese eine nach der anderen zu erzeugen (nichtdeterministisch!). Wenn man so viele verschiedene Satzformen gefunden und erzeugt hat, wie es überhaupt nur gibt und die gesuchte war nicht dabei – dann ist das gesuchte Wort nicht erzeugbar...

Aber woher soll man wissen, wieviele Satzformen erzeugbar sind??

Die Situation ist ähnlich wie bei einem Bildersuchrätsel: Zwei Versionen sind gegeben (meist *Original und Fälschung* genannt). Ist die Aufgabe so gestellt: Suche die hier versteckten 8 Fehler, dann ist eine Lösung meist recht leicht – und wenn man acht Fehler identifiziert hat, kann man die Suche beenden. Aber wenn die Aufgabe lautet: Wieviele Fehler sind hier enthalten? Dann ist die Lösung erheblich schwieriger...

Beweis: Ein Plan

Wir haben eine Typ-1 Grammatik G und ein Wort $x \in \Sigma^*$, und die Frage ist, ob $x \in \overline{L(G)}$ gilt. Diese Frage werden wir beantworten, indem wir herausfinden, welche Satzformen der Länge $n = |x|$ in G erzeugbar sind. Die benutzte Technik wird häufig mit dem Ausdruck *induktives Zählen* beschrieben.

Dabei hangeln wir uns durch die benötigten Schrittzahlen wie folgt:

Die einzige in 0 Schritten erreichbare Satzform der Länge maximal n ist die Satzform S.

Wenn wir wissen wieviele Satzformen der Länge maximal n in k Schritten erreichbar sind, können wir für eine gegebene solche Satzform herausfinden, ob sie in $k+1$ Schritten erreichbar ist.

Damit können wir die Anzahl der in $k+1$ Schritten erreichbaren solchen Satzformen ermitteln. Details auf den nächsten Folien...

Zentraler Begriff: *Durchzykeln*

Es gibt nur eine beschränkte Zahl von Satzformen bis zur Länge n, je nach der (konstanten) Größe des Alphabets $V \cup \Sigma$ exponentiell viele – bezüglich n. Diese können wir uns der Reihe nach auf das Band schreiben. Dann entscheiden wir jeweils nichtdeterministisch, ob wir diese gerade betrachtete Satzform (in k Schritten) erreichen wollen. Wenn wir JA sagen, müssen wir das bestätigen, indem wir eine solche Ableitung in k Schritten (nichtdeterministisch) *erraten.* Wir zählen mit, wie oft wir JA gesagt haben.

Beachte: Bei falschem JA bricht dieser Berechnungszweig ab.

Aber: Bei falschem NEIN haben wir am Ende zu wenig Satzformen.

Wenn wir zu wenig Satzformen haben, brechen wir die Berechnung ab. Nun ist klar: Auf allen nicht abgebrochenen Berechnungswegen haben wir genau die richtigen Satzformen erraten und bestätigt.