12. Übungsblatt zu
Grundzüge der Theoretischen Informatik, WS 2013/14

Prof. Dr. Gert Smolka, Jonas Kaiser, M.Sc.
www.ps.uni-saarland.de/courses/ti-ws13/

Lesen Sie im Buch: Lecture 31 – Lecture 34

Programmiersysteme

Im Folgenden betrachten wir Akzeptierbarkeit und Entscheidbarkeit von Sprachen. Wir verwenden für diesen Zweck *Programmiersysteme*, welche genug Struktur liefern, um interessante Aussagen zu beweisen. Andererseits ist dieses Modell abstrakt und kann mit Turing-Maschinen instanziert werden.

Definition: Ein Programmiersystem ist ein Tripel $(\Sigma, \pi, [\,])$ mit den folgenden Eigenschaften:

1. Σ ist ein endliches Alphabet mit mindestens einem Zeichen.
2. $\pi \subseteq \Sigma^* \times \Sigma^*$ ist eine Akzeptanzrelation. Wir schreiben πxy für $(x, y) \in \pi$ und sagen „x akzeptiert y“.
3. $[\,] : \mathit{Exp} \to \Sigma^*$ ist eine Übersetzungsfunktion, wobei
   a) *Exp* die folgende Menge von Ausdrücken ist:

$$s, t \coloneqq x \mid s@t \mid \alpha \mid \lambda\alpha.s \mid T \mid R \mid \textbf{if } xst \qquad (x \in \Sigma^*)$$

   b) folgende Bedingungen erfüllt sind:

$$\begin{aligned} [x] &= x & \pi[s@t]y &\equiv \pi[s][t] \\ \pi[\alpha]y &\equiv \textbf{false} & \pi[\lambda\alpha.s]y &\equiv \pi[s^{\alpha}_{y}]\epsilon \\ \pi[T]y &\equiv \textbf{true} & \pi[\textbf{if } xst]y &\equiv (x = y \wedge \pi[s]y) \vee (x \neq y \wedge \pi[t]y) \\ \pi[R]y &\equiv \textbf{false} \end{aligned}$$

Wir definieren die von x akzeptierte Sprache wie folgt:

$$\mathcal{L}(x) \stackrel{\text{def}}{=} \{y \in \Sigma^* \mid \pi xy\}$$

Für $A \subseteq \Sigma^*$ definieren wir:

$$\begin{aligned} x \text{ akzeptiert } A &\stackrel{\text{def}}{\iff} \mathcal{L}(x) = A \\ A \text{ ist akzeptierbar} &\stackrel{\text{def}}{\iff} \exists x \in \Sigma^*.\mathcal{L}(x) = A \\ A \text{ ist entscheidbar} &\stackrel{\text{def}}{\iff} A \text{ ist akzeptierbar } \wedge\, \overline{A} \text{ ist akzeptierbar} \end{aligned}$$

Anmerkung zur Notation: Wir verwenden $\equiv$ für logische Äquivalenz, $\overline{A}$ für das Komplement von A und $\mathcal{L}(x) \coloneqq$ für die von x akzeptierte Sprache.

Entscheidbarkeit

Aufgabe 12.1 Seien die folgenden drei Mengen gegeben:

$$H \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \pi xx\}$$
$$H_y \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \pi xy\}$$
$$H_\forall \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \forall y \in \Sigma^*.\pi xy\}$$

Zeigen Sie die folgenden Aussagen:

(a) H ist akzeptierbar.
(b) $\overline{H}$ ist nicht akzeptierbar.
(c) H_y ist akzeptierbar.
(d) $\overline{H_y}$ ist nicht akzeptierbar.
(e) $\overline{H_\forall}$ ist nicht akzeptierbar.

Aufgabe 12.2 Zeigen Sie, dass die folgenden Sprachen nicht entscheidbar sind.

(a) $Fin = \{x \in \Sigma^* \mid \mathcal{L}(x) \text{ endlich}\}$
(b) $Kat = \{x \in \Sigma^* \mid \pi x(xx)\}$
(c) $D = \{x \in \Sigma^* \mid \mathcal{L}(x) = \emptyset\}$

Aufgabe 12.3 Sei $A \subseteq \Sigma^*$ endlich. Ist A entscheidbar? Beweisen Sie die Korrektheit Ihrer Antwort.

Programmiersysteme mit Schrittindizierung

Wir erweitern Programmiersysteme jetzt mit der Möglichkeit, die Anzahl der Ausführungsschritte eines Programms zu begrenzen. Ein derart eingeschränktes Programm terminiert immer, entweder regulär oder weil keine Schritte mehr erlaubt sind. Man spricht von *Schrittindizierung* (engl. step indexing).

Definition: Ein Programmiersystem ist ein Tripel $(\Sigma, \pi', [\,])$ mit den folgenden Eigenschaften:

1. Σ ist ein endliches Alphabet mit mindestens einem Zeichen.
2. $\pi' \subseteq \Sigma^* \times \Sigma^* \times \mathbb{N}$ ist eine *schrittindizierte* Akzeptanzrelation. Wir schreiben $\pi' xyn$ für $(x, y, n) \in \pi'$ und sagen „x akzeptiert y in (höchstens) n Schritten“.
3. π' ist monoton in der Schrittzahl: $\forall xymn.\pi' xym \to n \geq m \to \pi' xyn$
4. Sei $\pi xy \coloneqq \exists n.\pi' xyn$
5. Für π gelten weiterhin die bekannten Bedingungen und Definitionen.
6. Es gibt eine neue Ausdrucksform:

   $$s, t, u \coloneqq \ldots \mid \sigma s t_1 t_2$$

   Der neue Ausdruck muss die folgende Bedingung erfüllen:

   $$\pi[\sigma s t_1 t_2]y \equiv (\pi'[s]\epsilon|y| \wedge \pi[t_1]\epsilon) \vee (\neg\pi'[s]\epsilon|y| \wedge \pi[t_2]\epsilon)$$

Aufgabe 12.4 Zeigen Sie, dass $\overline{Fin}$ nicht akzeptierbar ist.

Turing-Maschinen

Aufgabe 12.5 Wie muss man die Transitionen von Turing-Maschinen einschränken, um Zweiwegautomaten zu erhalten?

Aufgabe 12.6 Geben Sie das Transitionsdiagramm für eine Turing-Maschine an, die die (markierte) Wiederholungssprache $W' = \{x\#x \mid x \in \{a,b\}^*\}$ akzeptiert. Nutzen Sie das Diagramm für $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ von letzter Woche als Vorlage.

Aufgabe 12.7 Implementieren Sie Ihre Turing-Maschine aus Aufgabe 12.6 in ML. Nutzen Sie folgendes Codefragment als Ausgangspunkt:

```
datatype Dir = L | R

datatype Gamma = TS | BLK | ...

datatype State = S | REJ | ACC | ...

fun trans S TS = SOME (S,TS,R)
  | ...
  | trans _ _ = NONE

fun run ...

fun exec input = case (run nil S (TS::input))
                 of ... => ...

val test0 = exec [M]
val test1 = exec [A,B,B,A,M,A,B,B,A]
val test2 = exec [A,B,M,B,A]
val test3 = exec [A,B,A]
val test4 = exec [A,B,M,A]
val test5 = exec [A,B,M,A,M]
```

Implementieren Sie die Transitionsrelation mit einer Funktion *trans* und den Lauf einer Maschine mit einer Funktion *run*. Die Funktionen sollten die folgenden Typen haben:

$$trans : State \to Gamma \to (State \times Gamma \times Dir)\ option$$
$$run : Gamma\ list \to State \to Gamma\ list \to (Gamma\ list \times State \times Gamma\ list)$$

Hinweis: Es vereinfacht die Implementierung deutlich, wenn die Funktion *run* die linke Bandhälfte der einzelnen Konfigurationen in umgekehrter Reihenfolge verwaltet.

Wenn die Prozedur *run* terminiert, liefert sie die Konfiguration in der die Maschine gehalten hat. Man kann den Zustand der Schlusskonfiguration dann mit $exec : Gamma\ list \to \alpha$ überprüfen, wobei α ein beliebiger Ausgabetyp ist (z.B. *string*).

Aufgabe 12.8 Skizzieren Sie eine Turing-Maschine, die die (nicht markierte) Wiederholungssprache $W = \{xx \mid x \in \{a,b\}^*\}$ akzeptiert. **Challenge:** Implementieren Sie Ihre Skizze in ML.