Grundzüge der Theoretischen Informatik, WS 21/22: Musterlösung zum 5. Übungsblatt

Julian Dörfler

Aufgabe A5.1 (Wiederholung: Reguläre Sprachen) *(4 Punkte)*
Welche der folgenden Sprachen sind regulär. Beweisen Sie Ihre Antworten.

(a) Die Sprache A aller Variablen in `WHILE` über dem Alphabet $\Sigma = \{x, 0, 1, \ldots, 9\}$. Hierzu lesen wir z.B. die Variable x_{42} als das Wort $x42$. Beachten Sie, dass Variablenbezeichner keine führenden Nullen enthalten mit der Ausnahme von x_0.

(b) Die Sprache B aller syntaktisch gültigen `WHILE`-Programme die wir auf die Variablen $x_0, \ldots, x_{10}$ und Konstanten $0, 1, \ldots, 10$ beschränken[1].

Lösung A5.1 (Wiederholung: Reguläre Sprachen)

(a) Diese Sprache ist regulär und wird vom regulären Ausdruck $x0 + x(dig)(dig + 0)^\star$ erkannt, wobei $dig = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9$.

(b) Sei $n \in \mathbb{N} \setminus \{0\}$ beliebig. Wähle $uvw = (\textbf{while } x_0 \neq 0 \textbf{ do})^n\ x_0 := 0\ \textbf{od}^n \in B$ mit

$$u = (\textbf{while } x_0 \neq 0 \textbf{ do})^n\ x_0 := 0, v = \textbf{od}^n, w = \varepsilon.$$

Sei $xyz = v$ eine beliebige Unterteilung von v mit $|y| > 0$. Dann ist

$$t = uxy^0zw = (\textbf{while } x_0 \neq 0 \textbf{ do})^n\ x_0 := 0\ \textbf{od}^{n-|y|} \notin B,$$

da $|y| > 0$. Also lässt sich B nicht pumpen , woraus $B \notin \mathsf{REG}$ folgt.

Aufgabe A5.2 (Umkehrung der Paarfunktion) *(4 Punkte)*
Zeigen Sie, dass die Funktionen π_1 und π_2 aus der Vorlesung `FOR`-berechenbar sind. *Hinweis: Sie dürfen und sollten den syntaktischen Zucker aus der Vorlesung hier verwenden.*

Lösung A5.2 (Umkehrung der Paarfunktion)
Wir geben beide `WHILE`-Programme als ein gemeinsames Program an, das mit $x_1 = \pi_1(p)$ und $x_2 = \pi_2(p)$ beide Umkehrfunktionen zeitgleich berechnet.

Input: p
  $z := 0$; ▷ finding the largest z
  for p **do**
    $x_{tmp} := \frac{1}{2}(z + 1)(z + 2)$;
    if $x_{tmp} \leq p$ **then**

[1] Dies ist notwendig, da wir formal `WHILE`-Programme über einem unendlichen Alphabet definiert hatten, aber reguläre Sprache über endlichen Alphabeten.

```
        z++
    fi
od;                                                    ▷ z = x1 + x2
x2 := p − 1/2 z(z + 1);
x1 := z − x2
```

Aufgabe A5.3 (FOR-Programme) *(4 Punkte)*

(a) Zeigen Sie, wie sich in FOR-Programmen die Anweisungen $x_i := x_j + x_k$, $x_i := x_j - x_k$ und $x_i := c$ durch die Anweisungen $x_i{+}{+}$, $x_i{-}{-}$ und $x_i := 0$ ersetzen lassen.

(b) Ersetzen Sie $x_i{-}{-}$ durch andere Anweisungen, wobei Sie natürlich weder $x_i := c$ für $c \neq 0$ noch $x_i := x_j \pm x_k$ verwenden dürfen. Erlaubt sind nur FOR-Schleifen, $x_i := 0$ und $x_i{+}{+}$ und Konkatenation.

Wichtig: In dieser Aufgabe ist außer Umbenennen der Variablen und Verwendung der Assoziativität der Konkatenation kein syntaktischer Zucker erlaubt.

Lösung A5.3 (FOR-Programme)

(a) Wir nehmen allgemein an, dass $i \notin \{j, k\}$. Ansonsten ersetzen wir x_i in der Anweisung zuerst durch eine frische Variable x_ℓ und kopiere anschliessend x_ℓ nach x_i mit dem folgenden Program:

```
xi := 0;
for xℓ do
    xi++
od
```

Anweisung „$x_i := x_j + x_k$“:

```
xi := 0;
for xj do
    xi++
od;
for xk do
    xi++
od
```

Anweisung „$x_i := x_j - x_k$“: Ähnlich wie oben, man ersetze lediglich in der zweiten FOR-Schleife das „$x_i{+}{+}$“ durch „$x_i{-}{-}$“:

```
xi := 0;
for xj do
    xi++
od;
for xk do
    xi−−
```

od

Anweisung „$x_i = c$“:

$x_i := 0;$
$\left.\begin{array}{l} x_i{+}{+}; \\ \vdots \\ x_i{+}{+} \end{array}\right\} c$ mal

(b)

Anweisung „$x_i - -$“:

$x_k := 0;$
for x_i **do**
  $x_i := 0;$
  for x_k **do**
    $x_i{+}{+}$
  od;
  $x_k{+}{+}$
od

In diesem FOR-Programm wird eine frische Variable x_k von 0 bis zum Eingabewert hochgezählt. x_i wird dabei immer vor der Inkrementierung auf x_k gesetzt. Dadurch ist der Wert von x_i am Ende um eins niedriger.

Aufgabe A5.4 (FOR-Programme terminieren) *(4 Punkte)*
Beweisen Sie, dass FOR-Programme immer terminieren. Zeigen Sie dafür per struktureller Induktion, dass für jedes FOR-Programm P die Funktion Φ_P total ist.

Lösung A5.4 (FOR-Programme terminieren)
Wir zeigen pre struktureller Induktion über den Aufbau von P, dass Φ_P immer total ist.

P ist ein simple Statement : Für simple Statements ist Φ_P nach Definition immer total.

P ist “$P_1; P_2$” : P_1 und P_2 sind kleinere FOR-Programme. Φ_{P_1} und Φ_{P_2} sind also nach Induktionsvorraussetzung total. Nun gilt

$$\Phi_P(S) = \Phi_{P_2}(\Phi_{P_1}(S))$$

Da Φ_{P_1} und Φ_{P_2} beide total sind, ist auch Φ_P total.

*P ist “**for** x_i **do** P_1 **od**”* : P_1 ist ein kleineres FOR-Programm. Φ_{P_1} ist also nach Induktionsvorraussetzung total.

Wir zeigen nun, dass $\Phi_{P_1}^{(n)}(S)$ für alle $n \in \mathbb{N}$ und S definiert ist per Induktion über n.

I.A. $n = 0$ Es gilt $\Phi_{P_1}^{(0)}(S) = S$ und somit ist $\Phi_{P_1}^{(0)}(S)$ trivial definiert.

I.V. Sei $\Phi_{P_1}^{(n)}(S)$ für ein festes, aber beliebiges $n \in \mathbb{N}$ definiert.

I.S. $n \to n+1$ Es gilt $\Phi_{P_1}^{(n+1)}(S) = \Phi_{P_1}(\Phi_{P_1}^{(n)}(S))$. Nach Induktionsvorraussetzung ist $\Phi_{P_1}^{(n)}(S)$ definiert. Zusätzlich ist Φ_{P_1} total. Somit ist $\Phi_{P_1}^{(n+1)}(S)$ definiert.

Nach Definition ist

$$\Phi_P(S) = \Phi_{P_1}^{(\sigma_i)}(S)$$

Da $\sigma_i \in \mathbb{N}$ gilt aber, dass $\Phi_P(S)$ für alle S definiert ist. Φ_P ist also total.