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

Julian Dörfler

Aufgabe P5.1 (Sprachen revisited)
Sei $k \in \mathbb{N}$ beliebig aber fest. Zeigen Sie für die folgenden Sprachen aus Aufgabe P4.1 mithilfe des Myhill-Nerode-Theorems, dass diese nicht regulär sind.

(a) $E_k = \{0^n 1^n \mid n \geq k\}$

(b) $F = \{0^n 1^m \mid n \neq m\}$

(c) $J = \{0^n 1^m \mid 2^n \neq m\}$

Lösung P5.1 (Sprachen revisited)

(a) E_k ist nicht regulär. Wir zeigen dies indem wir unendlich viele Myhill-Nerode-Äquivalenzklassen nachweisen. Hierzu betrachten wir 0^{i+k} und 0^{j+k} für $i \neq j$. Dann ist $0^{i+k}1^{i+k} \in E_k$, aber $0^{j+k}1^{i+k} \notin E_k$. Somit ist $0^{i+k} \not\sim_{E_k} 0^{j+k}$ und E_k hat unendlich viele Myhill-Nerode-Äquivalenzklassen, da $i, j \in \mathbb{N}$ beliebig waren. Damit ist E_k nicht regulär.

(b) F ist nicht regulär. Wir zeigen dies indem wir unendlich viele Myhill-Nerode-Äquivalenzklassen nachweisen. Hierzu betrachten wir 0^i und 0^j für $i \neq j$. Dann ist $0^i 1^i \notin F$, aber $0^j 1^i \in F$. Somit ist $0^i \not\sim_F 0^j$ und F hat unendlich viele Myhill-Nerode-Äquivalenzklassen, da $i, j \in \mathbb{N}$ beliebig waren. Damit ist F nicht regulär.

(c) J ist nicht regulär. Wir verwenden das Myhill-Nerode-Theorem: Seien $i \neq j$ natürliche Zahlen. Dann sind $0^i \not\sim_J 0^j$, da $0^i 1^{2^i} \notin J$ ist, aber $0^j 1^{2^i} \in J$. Da i und j beliebig aus den natürlichen Zahlen gewählt waren, hat J also unendlich viele Myhill-Nerode-Äquivalenzklassen und ist somit nicht regulär.

Aufgabe P5.2 (Syntaktischer Zucker)
Damit wir uns Schreibarbeit sparen können, führen wir syntaktischen Zucker ein. Simulieren Sie hierfür die folgenden Befehle durch Befehle der in der Vorlesung definierten `FOR`-Sprache:

(a) $x_i := x_j^{x_k}$

(b) if $x_i = 0$ then P_1 else P_2 fi

(c) $x_i := x_j$ div x_k (wobei div die ganzzahlige Division bezeichnet)

(d) $x_i := x_j$ mod x_k (wobei mod die Modulo-Funktion bezeichnet)

(e) Überlegen Sie sich weiterhin, wie Sie die Bedingungen aus Teilaufgabe (b) für weitere Operationen verallgemeinern können. Simulieren Sie hierzu folgende Verallgemeinerung für $\circ \in \{<, \leq, =, \geq, >, \neq\}$:

if $x_i \circ x_j$ then P_1 else P_2 fi

Lösung P5.2 (Syntaktischer Zucker)

(a) Wir benutzen hierfür den Code für die Multiplikation, der im Skript in Kapitel 8 eingeführt wurde.

$x_i := 1;$
for x_k **do**
    $x_i := x_i \cdot x_j$
od

(b) Wir simulieren das Konditional durch drei For-Schleifen. Zunächst setzen wir x_{zero} und $x_{nonzero}$ auf 0, bzw 1, je nachdem, ob x_i auf 0 gesetzt ist, oder nicht. Ist x_{zero} beim Erreichen der zweiten Schleife dann 1, so wird P_1 ausgeführt. Ist $x_i > 0$, so wird die erste Schleife betreten, mindest einmal ausgeführt, x_{zero} auf 0 gesetzt, $x_{nonzero}$ auf 1 gesetzt und somit nur P_2 ausgeführt. Ist $x_i = 0$, so wird nur P_1 ausgeführt.

$x_{zero} := 1;$
$x_{nonzero} := 0;$
for x_i **do**
    $x_{zero} := 0;$
    $x_{nonzero} := 1$
od;
for x_{zero} **do**
    P_1
od;
for $x_{nonzero}$ **do**
    P_2
od

(c) Wir ziehen den Wert x_k wiederholt vom ursprünglich auf x_j initialisierten x_{acc} ab, bis wir ein $x_{acc} < x_k$, also $x_{comp} = x_{acc} + 1 - x_k \leq 0$ erhalten. Die Anzahl der Subtraktionen führen wir in x_{count} mit. Sobald x_{comp} zum ersten Mal $x_{comp} \leq 0$ erfüllt, schreiben wir x_{count} in x_i und berühren die Variable x_i im weiteren Programmfluss nicht mehr. Somit gilt dann nach der Ausführung $x_i = x_j$ div x_k.

$x_i := 0;$
$x_{set} := 0;$
$x_{acc} := x_j;$
$x_{count} := 0;$
for x_j **do**
    $x_{count} := x_{count} + 1;$
    $x_{acc} := x_{acc} - x_k;$
    $x_{comp} := x_{acc} + 1;$

$x_{comp} := x_{comp} - x_k$;
if $x_{comp} = 0$ **then**
if $x_{set} = 0$ **then**
$x_i := x_{count}$;
$x_{set} := 1$
else
fi
else
fi
od

(d) Wegen $a = (a \text{ div } b) \cdot b + a \bmod b$ erhalten wir $a \bmod b = a - (a \text{ div } b) \cdot b$. Im Nachfolgenden seien x_1 und x_2 zwei vorher noch nicht benutzte Variablen.

$x_1 := x_j \text{ div } x_k$;
$x_2 := x_1 \cdot x_k$;
$x_i := x_j - x_2$

(e) Wir zeigen, wie man die anderen Vergleichsoperationen auf einen Vergleich der Form $x_{comp} = 0$ reduzieren kann.

$x_i \leq x_j$: Die Bedingung $x_i \leq x_j$ ist äquivalent zu $x_i - x_j \leq 0$. In `WHILE` entspricht dies einem Vergleich mit 0, da `WHILE`-Programme keine negativen Zahlen speichern.

$x_{comp} := x_i - x_j$;

$x_i \geq x_j$: Analog zu $x_i \leq x_j$ mit vertauschten Variablen.

$x_i < x_j$: Da x_i und x_j natürliche Zahlen sind ist $x_i < x_j$ äquivalent zu $x_i + 1 \leq x_j$.

$x_{comp} := x_i + 1 - x_j$;

$x_i > x_j$: Analog zu $x_i > x_j$ mit vertauschten Variablen.

$x_i = x_j$: Hier benutzen wir, dass $x_i = x_j$ äquivalent ist zu $x_i \leq x_j \wedge x_i \geq x_j$. Die Konjunktion $x_{\leq} = 0 \wedge x_{\geq} = 0$ können wir durch eine gemeinsame Bedingung $x_{\leq} + x_{\geq} = 0$ austauschen. Sobald eine der Variablen $x_{\leq}, x_{\geq}$ ungleich 0 ist, so kann die Summe durch das Fehlen negativer Zahlen niemals zu 0 auswerten.

$x_{\leq} := x_i - x_j$;
$x_{\geq} := x_j - x_i$;
$x_{comp} := x_{\leq} + x_{\geq}$;

$x_i \neq x_j$: Um Ungleichheit zu simulieren vertauschen wir in

if $x_i \neq x_j$ then P_1 else P_2 fi

die Programme P_1 und P_2 und erhalten

if $x_i = x_j$ then P_2 else P_1 fi.

Aufgabe P5.3 (Benutzen Sie syntaktischen Zucker)
Versuchen Sie folgendes `FOR`-Programm zu verstehen und vereinfachen Sie dieses mithilfe von syntaktischem Zucker möglichst weit.

$x_1 := 0;$
$x_2 := 1;$
$x_3 := 0;$
$x_4 := 1;$
for x_0 **do**
    $x_5 := x_0 + x_3;$
    $x_6 := 0;$
    for x_4 **do**
        $x_6 := x_6 + x_4$
    od;
    $x_7 := x_5 - x_6;$
    $x_8 := x_6 - x_5;$
    $x_9 := x_7 + x_8;$
    $x_{10} := 1;$
    $x_{11} := 0;$
    for x_9 **do**
        $x_{10} := 0;$
        $x_{11} := 1$
    od;
    for x_{10} **do**
        $x_1 := 1$
    od;
    for x_{11} **do**
        $x_4 := x_4 + x_2$
    od
od;
$x_0 = x_1 + x_3$

Hinweis: Vermeiden Sie es, Ihren Tutoren Code in diesem Stil abzugeben und verwenden Sie stattdessen syntaktischen Zucker; insbesondere eine sinnvolle Benennung von Variablen. Wir behalten uns vor, auf besonders unübersichtliche Abgaben weniger Punkte zu vergeben.

Lösung P5.3 (Benutzen Sie syntaktischen Zucker)
Der Code prüft, ob die Eingabe eine Quadratzahl > 0 ist. Der vereinfachte Code ist der folgende (x_{in} ist hierbei die Eingabe und $x_{issquare}$ ist hierbei die Ausgabe):

$x_{issquare} := 0;$
$x_{root} := 1;$
for x_{in} **do**
    if $x_{in} = x_{root}^2$ **then**
        $x_{issquare} := 1$
    else

$x_{root}++$

fi

od