Prof. Javier Esparza
Philipp Czerner, Martin Helfrich

Technische Universität München
Lehrstuhl für Theoretische Informatik

Einführung in die Theoretische Informatik

Sommersemester 2021 – Hausaufgabenblatt 6

Abgabe: 31.05.2021, 12:00 CEST

- Diese Woche werden alle Aufgaben korrigiert.
- Wenn Sie einen Beweis aufstellen, von dem Sie wissen, das einzelne Schritte problematisch oder unvollständig sind, merken Sie dies bitte in Ihrer Lösung an, damit wir das bei der Korrektur positiv berücksichtigen können.

Aufgabe H6.1. (*Xuby*) 1+1+1+1 Punkte

Sei $\Sigma := \{a, b\}$, und sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik mit den Produktionen

$$\begin{aligned} S &\to XUb \mid Y & Y &\to ab \mid YY \\ X &\to UbY \mid UU & U &\to aU \mid \varepsilon \end{aligned}$$

Konvertieren Sie G mit dem aus der Vorlesung bekannten Verfahren schrittweise zur Chomsky-Normalform, indem Sie G so anpassen, dass für jede Produktion $\alpha \to \beta$ folgende zusätzlichen Einschränkungen gelten:

(a) $\beta \in \Sigma \cup V^*$ (b) $|\beta| \leq 2$ (c) $\beta \neq \varepsilon$ (d) $\beta \notin V$

Sie sollen also eine Grammatik G' angeben, die (a) und $L(G') = L(G)$ erfüllt, eine Grammatik G'', die (a), (b) und $L(G'') = L(G)$ erfüllt, usw.

Aufgabe H6.2. (*Matriririkelnununummern*) 2+2 Punkte

Eines Morgens erhalten Sie einen panischen Anruf von einem alten Bekannten[1], dem Rektor der Technischen Hochschule Estlingen-Oberfeld. Nachdem eine Horde wütender Studierender das Verwaltungsgebäude stürmte und die Wiedereinführung der traditionellen Estlinger Matrikelnummern verlangte, hat sich der Rektor nun im Keller verbarrikadiert und sucht verzweifelt nach Lösungen, um diese automatisch verarbeiten zu können. In einem alten Kabinett hat er einen Bausatz für Automaten gefunden, mit denen man anscheinend beliebige kontextfreie Sprachen akzeptieren kann. Sind diese „Kellerautomaten“ vielleicht die Lösung?

(a) Sei $\Sigma := \{0, ..., 9\}$. Zeigen Sie, dass die Sprache $L := \{\#\}\{w \in \Sigma^* : \sum_{i \geq 1} w_i \text{ prim}\}$ nicht kontextfrei ist.

Nachdem sich seine letzte Hoffnung in Luft aufgelöst hat, beschließt der Rektor, durch einen der vielen unterirdischen Tunnel zu fliehen. Diese werden leider von der Bergbaufakultät kontrolliert, und können nur mit bestimmten Klopfzeichen betreten werden, die einem komplizierten System folgen, von dem der Rektor schlicht überfordert ist. Aber vielleicht kann er sich ja einen Kellerautomaten dafür bauen...

(b) Zeigen oder widerlegen Sie: $L := \{a^i b^j c^k : i, j, k \in \mathbb{N} \text{ und } j^2 \leq ik\}$ ist kontextfrei.

[1] Siehe H3.2

Aufgabe H6.3. (((()(((())()))))) 3+1 Punkte

S-Expressions sind eine Notation für verschachtelte Listen. Für unsere Zwecke verwenden wir das Alphabet $\Sigma := \{a, ..., z, 0, ..., 9\}$, und ein Atom ist ein Wort in Σ^*. Eine *S-Expression* ist entweder ein Atom, oder eine Liste $(w_1\ w_2\ ...\ w_n)$, wobei $w_1, ..., w_n$ S-Expressions sind. Beispiele:

```
theo
(theo 2021)
((1) (2 3) (4 5) () (6))
```

Wir definieren uns nun eine Programmiersprache, die wir theoLISP nennen, aus folgenden Komponenten.

- Arithmetische Operationen:

  `(add` x y`)` $\to x + y$
  `(sub` x y`)` $\to x - y$
  `(mul` x y`)` $\to xy$

- Bedingungen und Schleifen:

  `(if` x y z`)` Wenn $x > 0$, wird y ausgeführt, sonst z
  `(while` x y`)` Solange $x > 0$, wird y ausgeführt.

- Variablenzuweisungen:

  `(set` x y`)` Setzt die Variable mit Namen x auf den Wert von y

Ein Programm ist eine Liste an Befehlen, die Ausgabe eines Programms ist der Wert der Variablen `result`. Alle Variablen sind mit 0 initialisiert, und können Werte in $\mathbb{Z}$ annehmen.

Beispielprogramme mit Ihren Ausgaben:

```
((set x 42)
 (set result (sub x 2021)))
```

$\to -1979$

```
((set x 1)
 (while (sub 16 i) (
     (set x (mul x 2))
     (set i (add i 1))))
 (set result x))
```

$\to 65536$

```
((set a 247) (set b 299)
 (while a (
     (set a (sub b a))
     (if (sub a b) (
         (set t a) (set a b) (set b t))
         () )))
 (set result b))
```

$\rightarrow 13$

Folgende sechs Programme sind nicht gültig:

```
((print 42))
((set 5 7))
(set result 42)
((if (while x ()) () ()))
((set result (if x 16 42)))
((if ((add 20 21)) () ()))
```

Insbesondere erwartet z.B. `set` als erstes Argument einen Variablennamen, keine Zahl; und `if` als erstes Argument eine Variable, Zahl, oder arithmetische Operation, und keine Anweisung oder Liste an Anweisungen.

(a) Konstruieren Sie eine kontextfreie Grammatik, die genau die theoLISP Programme erzeugt. Es genügt, wenn Sie nur die Programme erzeugen, die keine unnötigen Leerzeichen oder Zeilenumbrüche enthalten. Das erste Beispielprogramm wäre also

```
((set x 42) (set result (sub x 2021))
```

Sie können Leerzeichen mit `.` notieren.

(b) Geben Sie einen Syntaxbaum Ihrer Grammatik für folgendes Programm an:

```
((while x ()) (set x (add 42 7)))
```

Bonusaufgabe H6.4. (*S-Parser*) 1+1+1 Bonuspunkte

Bei Ausgrabungen im antiken Rom findet der Archäologe Jasper Vazarie eine kuriose Marmortafel mit folgender Beschriftung:

```
((set n 339) (while (sub 1 result) ((set nn (sub n 1)) (while (
sub 1 flag) ((set dd 0) (while (sub nn 1) ((set dd (add dd 1)) (
set nn (sub nn 2)))) (if nn ((set flag 1)) ((set nn dd) (set d dd
) (set r (add r 1)))))) (set k 4) (while k ((set a (mul (add a 1)
89)) (while ((sub a 112)) ((set a (sub a 113)))) (set x 1) (set
set dd d) (while dd ((set x (mul x a)) (while (sub x (sub n 1)) (
(set x (sub x n)))) (set dd (sub dd 1)))) (set xx (mul (sub x 1)
(sub x (sub n 1)))) (set flag 0) (if (mul xx xx) ((set rr (sub r
1)) (while rr ((set x (mul x x)) (while (sub x (sub n 1)) ((set x
(sub x n)))) (if (sub (sub n 1) x) () ((set flag 1) (set rr 0)))
(set rr (sub rr 1))))) ((set flag 1))) (if (sub 1 flag) ((set k 0
)) ()) (set k (sub k 1)))) (if (add k 1) ((set result n)) ((set n
(add n 2))))))))
```

Jasper vermutet, dass es sich hierbei um ein theoLISP Programm handelt, da dies eine beliebte Programmiersprache in der damaligen hispanischen Provinz Gallaecia war. Die Tafel stammt aus dem Jahre 339 n. Chr. und wurde, so die Überlieferung, dazu verwendet, die Geburt des nächsten großen Kaisers vorherzusagen. Leider litt auch die Marmortafel unter dem Zahn der Zeit und es befinden sich einige Fehler im Programm.

Helfen Sie Jasper, und korrigieren das Programm! Implementieren Sie hierzu einen Recogniser, der entscheidet, ob ein Wort ein syntaktisch korrektes theoLISP Programm ist, wie in Aufgabe H6.3 definiert. (Sie dürfen annehmen, dass das Programm keine unnötigen Leerzeichen und Zeilenumbrüche enthält.)

Hinweise: Verwenden Sie zur Lösung dieser Aufgabe einen Computer. Beschreiben Sie bitte Ihren Ansatz in *natürlicher Sprache* und illustrieren die wesentlichen Schritte Ihrer Lösung mit geeigneten Codefragmenten. Sie können (müssen aber nicht), Ihrer Lösung Ihren vollständigen Programmcode beifügen, jedoch steht es den Korrektoren frei, diesen zu ignorieren. Beschränken Sie sich in Ihrer Implementierung auf grundlegende Funktionalitäten verbreiteter Programmiersprachen. Insbesondere ist die Verwendung von Library-Funktionen für reguläre Ausdrücke oder andere Parser-Generatoren nicht gestattet.

Obiges Programm hat genau drei Fehler. Um Ihr Programm zu implementieren, mag es sinnvoll sein, Ihre Lösung zu Aufgabe H6.3(a) zu verwenden, und für jede Variable V eine Funktion zu schreiben, die (rekursiv) ein maximales Präfix aus $L_G(V)$ einliest. Zum Finden der Syntaxfehler bietet es sich an, dass Ihr Recogniser zusätzliche Informationen ausgibt, wenn er einen Fehler feststellt.

Die Beispielprogramme auf diesem Blatt finden Sie unter diesem Link. Im Ordner `nospace` finden Sie jeweils Versionen ohne unnötige Leerzeichen und Zeilenumbrüche. Die syntaktisch korrekten Beispiele beginnen mit einem `a`, die inkorrekten Beispiele mit einem `b`, und obiges Programm heißt `c1`.