Lehrstuhl für Informatik 1
Prof. Dr. Berthold Vöcking
Alexander Skopalik
Matthias Englert

WS 2007/08
3.12.2007

Übung zur Vorlesung
Berechenbarkeit und Komplexität

Lösung Blatt 7

Aufgabe 7.1: (**10**)

(a) Wir reduzieren H auf H_{EQ}. Wir konstruieren eine berechenbare Funktion $f : \Sigma^* \to \Sigma^*$, so dass $x \in H \Leftrightarrow f(x) \in H_{\text{EQ}}$ gilt.

Sei x das Argument von f. Zunächst wird geprüft, ob x von der Form $x = \langle M \rangle w$ ist. Falls nicht, so ist $f(x) = \langle M_0 \rangle \langle M_1 \rangle$ wobei M_0 nie anhält und M_1 immer anhält und akzeptiert.

Ansonsten ist $f(x) = \langle M' \rangle \langle M^\dagger \rangle$. Die TM M' arbeitet wie folgt: M' löscht zunächst die Eingabe, und schreibt dann w auf das Eingabeband. Dann verhält sich M' genau wie M. Unabhängig von der tatsächlichen Eingabe simuliert M' also immer M auf w. Sei $M^\dagger$ eine TM, die auf allen Eingaben hält.

Hat x nicht die Form $x = \langle M \rangle w$, so gilt $x \notin H$ und offensichtlich $f(x) \notin H_{\text{EQ}}$. Sei nun $x = \langle M \rangle w$. Es gelten die folgenden Implikationen:

$x \in H \Rightarrow M$ hält auf $w \Rightarrow M'$ akzeptiert jede Eingabe $\Rightarrow f(x) = \langle M' \rangle \langle M^\dagger \rangle \in H_{\text{EQ}}$.

$x \notin H \Rightarrow M$ hält nicht auf $w \Rightarrow M'$ hält auf keiner Eingabe $\Rightarrow f(x) = \langle M' \rangle \langle M^\dagger \rangle \notin H_{\text{EQ}}$.

(b) Wir reduzieren H_{all} auf $H_\emptyset$. Wir konstruieren eine berechenbare Funktion $f : \Sigma^* \to \Sigma^*$, so dass $x \in H_{\text{all}} \Leftrightarrow f(x) \in H_\emptyset$ gilt.

Sei x das Argument von f. Zunächst wird geprüft, ob x von der Form $x = \langle M \rangle$ ist. Falls nicht, so ist $f(x) = \langle M_0 \rangle$ wobei M_0 jedes Wort verwirft.

Ansonsten ist $f(x) = \langle M' \rangle$. Die TM M' arbeitet wie folgt: M' verhält sich M' wie M. Wenn M hält, dann akzeptiert M'.

Hat x nicht die Form $x = \langle M \rangle$, so gilt $x \notin H_{\text{all}}$ und offensichtlich $f(x) \notin H_\emptyset$. Sei nun $x = \langle M \rangle$. Es gelten die folgenden Implikationen:

$x \in H_{\text{all}} \Rightarrow M$ hält auf jeder Eingabe $\Rightarrow M'$ akzeptiert jede Eingabe und hält immer $\Rightarrow f(x) = \langle M' \rangle \in H_\emptyset$.

$x \notin H_{\text{all}} \Rightarrow M$ hält nicht auf jeder Eingabe $\Rightarrow M'$ hält nicht auf jeder Eingabe $\Rightarrow f(x) = \langle M' \rangle \notin H_\emptyset$.

c) Wir nutzen die Transitivität der Reduktion aus. Wir reduzieren $H_\emptyset$ auf H_{Σ^*}. Falls x nicht von der Form $x = \langle M \rangle$ ist, so ist $f(x) = x$ wobei M_0 jedes Wort akzeptiert.

Ansonsten $f(x) = \langle M' \rangle$ wobei M' sich verhält wie M und verwirft wenn M akzetiert und akzeptiert, wenn M verwirft.

Aufgabe 7.2: (**10**)

Bemerkung: Im Folgenden ersetzen wir Anweisungen der Art $x_j := x_{k+2} + c$, $c \in \{-1, 0, 1\}$, durch die vereinfachte Darstellung $x_j := c$.

Bevor wir eine Lösung für die eigentliche Aufgabenstellung präsentieren, stellen wir eine Lösung für das folgende Problem vor.

Falls $x_l = 0$ ist, dann führe P aus.

Die Lösung für dieses Problem werden wir anschließend zur Lösung der Aufgabenstellung verwenden.

$x_{k+3} := x_l + 0;$
$x_{k+4} := 1;$
WHILE $x_{k+3} \neq 0$ DO
    $x_{k+3} := 0;$
    $x_{k+4} := 0;$
END
WHILE $x_{k+4} \neq 0$ DO
    $P;$
    $x_{k+4} := 0;$
END

Mit Hilfe einer WHILE-Anweisung können wir nur Bedingungen der Art $x_l \neq 0$ verwenden. Mit Hilfe des obigen Programmes können wir nun auch dessen Negation $x_l = 0$ verwenden. Die Korrektheit des Programmes können wir formal mit einer Fallunterscheidung ($x_{k+3} \neq 0$ und $x_{k+3} = 0$) beweisen. Die formale Ausführung bleibt dem Leser überlassen.

Wir stellen nun ein WHILE-Programm für das in der Aufgabenstellung genannte Problem vor.

Falls $x_{k+1} = i$ dann führe P_i aus.

Zunächst betrachten wir den Fall $i = 1$.

$x_{k+3} := x_{k+1};$
WHILE $x_{k+3} \neq 0$ DO
    $x_{k+3} := x_{k+3} - 1;$
    $x_{k+4} := 1;$
    WHILE $x_{k+3} \neq 0$ DO
        $x_{k+3} := 0;$
        $x_{k+4} := 0;$
    END
    WHILE $x_{k+4} \neq 0$ DO
        $P_1;$
        $x_{k+4} := 0;$
    END
    $x_{k+3} = 0;$
END

Die äußere WHILE-Anweisung garantiert, dass die Variable x_{k+1} (der Befehlszähler) ungleich Null ist. Falls sie ungleich Null ist, dann ziehen wir Eins von ihr ab und überprüfen, ob sie nun gleich Null ist. Falls dies der Fall ist, dann führen wir das Programm P_1 aus, anderenfalls nicht.

Das Programm P_1' können wir nun wie folgt verallgemeinern. Dabei müssen wir berücksichtigen, dass Variablen nicht negativ werden dürfen.

```
x_{k+3} = x_{k+1};
WHILE x_{k+3} ≠ 0 DO
    x_{k+3} := x_{k+1} − 1;                 1. Subtraktion
    x_{k+4} := 0;
    WHILE x_{k+3} ≠ 0 DO
        x_{k+3} := x_{k+3} − 1;             2. Subtraktion
        ...
            WHILE x_{k+3} ≠ 0 DO
                x_{k+3} := x_{k+3} − 1;     i-te Subtraktion
                x_{k+4} := 1;               Teste, ob x_{k+3} jetzt gleich 0 ist.
                WHILE x_{k+3} ≠ 0 DO        Falls dies der Fall ist, dann ändere x_{k+4} nicht.
                    x_{k+3} := 0;           Anderenfalls setze x_{k+4} zurück auf Null.
                    x_{k+4} := 0;           ⋮
                END                         Test beendet.
            END
        ...
    END
    WHILE x_{k+4} ≠ 0 DO
        P_i;
        x_{k+4} := 0;
    END
    x_{k+3} = 0;
END
```

Das Programm versucht i-Mal 1 von x_{k+3} abzuziehen. Falls dies möglich ist, überprüft es, ob x_{k+3} gleich 0 ist. Nur in diesem Fall wird die Variable x_{k+4} gleich 1 gesetzt. Dies wiederum bewirkt, dass P_i ausgeführt wird. Falls x_{k+4} initial ungleich i ist, dann ist x_{k+4} zu Beginn der letzten WHILE-Anweisung immer noch gleich 0, P_i wird also nicht ausgeführt.

Die Länge des Programms P_i' beträgt: $2 \cdot i + 12$.

Aufgabe 7.3: (**10**)

Das folgende Programm berechnet $x_0 \cdot x_1$.

```
x_{k+3} := x_0 + 0;
x_0 := 0;
WHILE x_1 ≠ 0 DO
    x_{k+4} := x_{k+3};
    WHILE x_{k+4} ≠ 0 DO
        x_0 := x_0 + 1;
        x_{k+4} := x_{k+4} − 1;
    END
    x_1 = x_1 − 1;
END
```

Die äußere Schleife wird genau $x_1 - 1$ mal durchlaufen. In der inneren Schleife wir zu x_0 der Wert von x_3, welches dem Wert von x_0 bei Programmbeginn entspricht, addiert.