Prof. Dr. Christoph Karg
Studiengang Informatik
Hochschule Aalen

Klausur Berechenbarkeit & Komplexität (Wintersemester 2010/2011)

Lösungshinweise

(Alle Angaben ohne Gewähr[1])

Aufgabe 1. Auswertung der Formel

$$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_1 \vee x_3)$$

mit dem 2-SAT Algorithmus

first	x_1	x_2	x_3	$x_1 \vee x_2$	$\neg x_1 \vee \neg x_2$	$\neg x_1 \vee \neg x_3$	$x_2 \vee x_3$	$x_1 \vee x_3$
true	1			✓	$x_2 \to 0$			
true	1	0		✓	$x_2 \to 0$	$x_3 \to 0$		
true	1	0		✓	$x_2 \to 0$	$x_3 \to 0$	$\lightning$	
false	0			$x_2 \to 1$				
false	0	1		✓	✓	✓	✓	$x_3 = 1$
false	0	1	1	✓	✓	✓	✓	✓
true	(0)	(1)	(1)	(✓)	(✓)	(✓)	(✓)	(✓)

Eine erfüllende Belegung für F ist $x_1 = 0$, $x_2 = 1$, $x_3 = 1$.

Aufgabe 2. Ziel ist die Konstruktion einer deterministische Einband Turing Maschine für die Sprache

$$L = \{x \in \{a,b\}^* \mid x \text{ enthält mehr } a\text{'s als } b\text{'s}\}.$$

a) Arbeitsweise des Algorithmus:

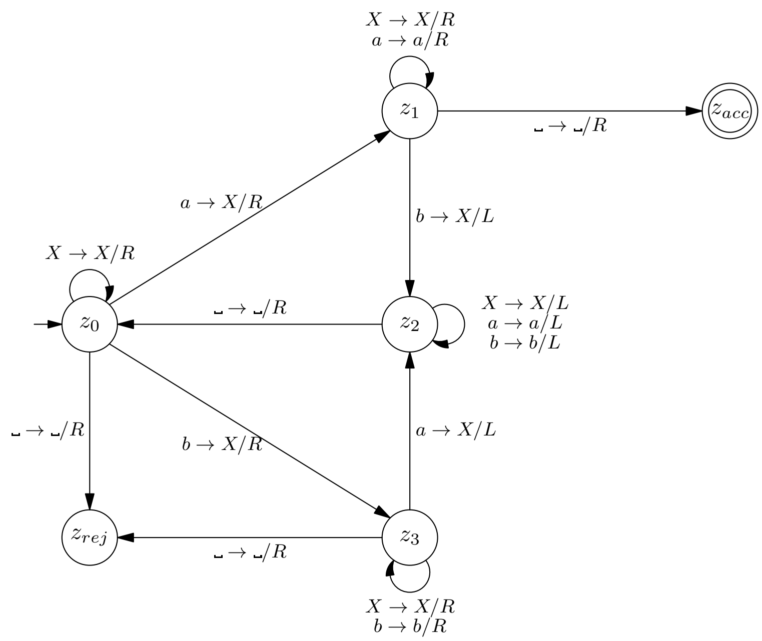
(1) Lese die Eingabe von links nach rechts und suche das erste Vorkommen von a oder b. Wird keiner dieser Buchstaben gefunden, dann verwerfe.

(2a) Wird ein a gefunden, dann ersetze es durch X und suche rechts von der Position des L/S-Kopfs nach einem b. Wird kein b gefunden, dann akzeptiere. Ansonsten ersetze b durch X und bewege den L/S-Kopf an das linke Ende des beschriebenen Teils des Arbeitsbands und springe zu (1).

(2b) Wird ein b gefunden, dann ersetze es durch X und suche rechts von der Position des L/S-Kopfs nach einem a. Wird kein a gefunden, dann verwerfe. Ansonsten ersetze a durch X und bewege den L/S-Kopf an das linke Ende des beschriebenen Teils des Arbeitsbands und springe zu (1).

[1] Sachdienliche Hinweise zur Fehlerbekämpfung senden Sie bitte an `christoph.karg@htw-aalen.de`

b) Zustandsdiagramm:



Aufgabe 3. Algorithmus zur Berechnung von $f(n) = 8n^4 + 3n^2 + 5$:

COMPUTEF(1^n)
Input: Natürliche Zahl n in Unärdarstellung
Output: $f(n) = 8n^4 + 3n^2 + 5$ in Binärdarstellung

$s := 0$
Scanne das Eingabeband von links nach rechts und erhöhe für jede gelesene 1 *den Wert von* s *um* 1
// Ab hier gilt: $(s)_2 = n$ und $|s| = \log_2 n$
$t := 0$
<u>for</u> $i := 1$ **<u>to</u>** s **<u>do</u>**
  $t := t + s$
// Ab hier gilt: $t = n^2$
$q := 0$
<u>for</u> $i := 1$ **<u>to</u>** t **<u>do</u>**
  $q := q + t$
// Ab hier gilt: $q = n^4$
$r := 5$
<u>for</u> $i := 1$ **<u>to</u>** 8 **<u>do</u>**
  $r := r + q$
<u>for</u> $i := 1$ **<u>to</u>** 3 **<u>do</u>**

```
    r := r + t
// Ab hier gilt: r = 8n^4 + 3n^2 + 5
return r
```

Analyse: Offensichtlich berechnet der Algorithmus die obige Funktion $f(n)$ und ist somit korrekt. Die Laufzeit setzt sich zusammen aus:

- Einlesen von 1^n und berechnen der Binärzahl s: $O(n \log_2 n)$
- Berechnung von $t = n^2$: $O(n \log_2 n)$
- Berechnung von $q = n^4$: $O(n^2 \log_2 n)$
- Berechnung von $r = 8n^4 + 3n^2 + 5$: $O(\log_2 n)$

Dies ergibt eine Gesamtlaufzeit von $O(n^2 \log_2 n)$.

Aufgabe 4. Da A entscheidbar ist, gibt es einen Algorithmus WordInA(x) mit

$$\textsc{WordInA}(x) = \texttt{true} \Leftrightarrow x \in A,$$

der auf allen Eingaben stoppt. Da auch B entscheidbar ist, existiert ein entsprechender Algorithmus WordInB(x) für B.

Betrachte den folgenden Algorithmus:

```
Check(x)
Input: Wort x = a_1 ... a_n, |x| = n
Output: true genau dann, wenn x ∈ A ∘ B
if WordInA(x) = true and WordInB(ε) = true then
    return true
if WordInA(ε) = true and WordInB(x) = true then
    return true
for i := 1 to n − 1 do
    if WordInA(a_1 ... a_i) = true and WordInB(a_{i+1} ... a_n) = true then
        return true
return true
```

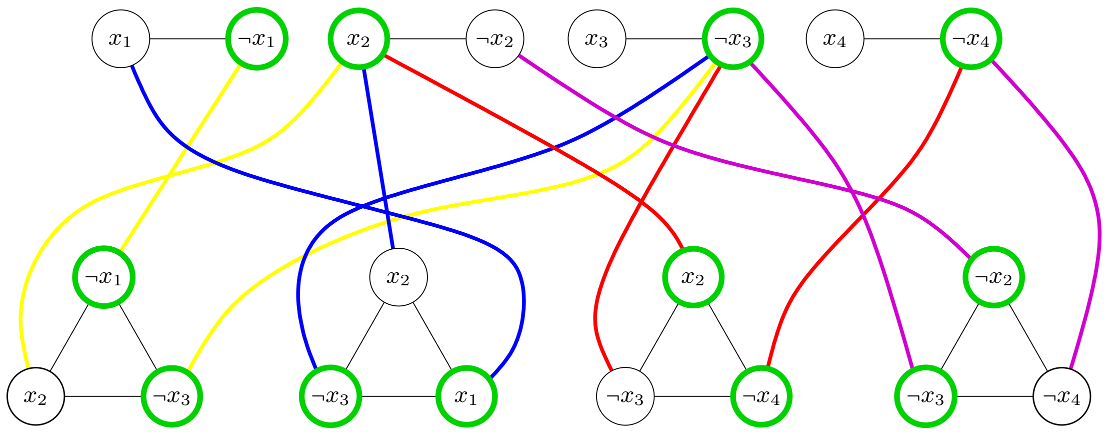
Da WordInA(x) und WordInB(x) auf allen Eingaben stoppen, stoppt auch Check(x) auf allen Eingaben.

Das Wort x ist in $A \circ B$ genau dann, wenn man x in zwei Teile y und z zerlegen kann mit $x = yz$ und $y \in A$ und $z \in B$. Genau dies wird in obigem Algorithmus überprüft. Also ist Check = `true` genau dann, wenn $x \in A \circ B$.

Aufgabe 5. Reduktion der Formel

$$(\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3 \vee x_1) \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4).$$

auf eine Problemstellung für Vertex Cover, wobei $k = 12$:



Die erfüllende Belegung $x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 0$ korrespondiert zu der in den Graphen eingezeichneten Lösung für Vertex Cover.

Aufgabe 6. Um zu beweisen, dass SI $\in$ NP, wird ein Polynomialzeit Verifikationsalgorithmus konstruiert:

CheckSI$((a_1, b_1), \ldots, (a_n, b_n), x)$
Input: Paare von natürlichen Zahlen $(a_1, b_1), \ldots, (a_n, b_n)$ mit $a_i \leq b_i$ für alle $i = 1, \ldots, n$, natürliche Zahl x.
Output: `true` falls $x \not\equiv a_i \pmod{b_i}$ für alle $i = 1, \ldots, n$, `false`, sonst.

```
for i := 1 to n do
   if x mod b_i = a_i then
   return false
return true
```

Aufgabe 7. Die entsprechende Formel in 3-KNF ist:

$$(x_1 \vee \neg x_2 \vee z_1) \wedge (\neg z_1 \vee x_3 \vee z_2) \wedge (\neg z_2 \vee x_4 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_3 \vee \neg x_3 \vee \neg x_3)$$

Die Belegung $x_1 = 1, x_2 = 0, x_3 = 0, z_1 = 0, z_2 = 0$ ist eine erfüllende Belegung für diese Formel. Die Belegung der x-Variablen ist eine erfüllende Belegung für F.