GTI Übungsblatt 8
Tutor: Marko Schmellenkamp
ID: MS1
Übung: Mi 16-18

Max Springenberg, 177792

8.1

8.1.1

(i)

$f(1) = \perp$
x_4 ist bei $x_1 = 1$ nach Ablauf des ersten WHILE-Programma (3-11) mit:
$x_4 := x_4 \dot{-} 1 = 1, \qquad x_1 := x_1 \dot{-} 1 = 0$
echt größer 0, damit endet das Folgende WHILE-Programm (12-14) nie.

$f(2) = 4$

(ii)

Da das WHILE-Programm (12-14) nur dann endet, bzw. umgangen wird, wenn zuvor x_4 auf 0 gesetzt wurde. Muss das WHILE-Programm (3-11) mindestens zwei mal durchlaufen werden, da x_4 initial auf 2 gesetzt und in diesem dekrementiert wird.
Die Funktion f_P ergibt sich dadurch zu:

$$f_P(n) = \begin{cases} 2^n, n \geq 2 \\ \perp, \text{sonst} \end{cases}$$

Werte- und Definitionsbereich sind wie folgt definiert.
$D(f) = \{n \in \mathbb{N}_0 | f(n) \neq \perp\}$
$W(f) = \{n \in \mathbb{N}_0 | \exists m \in \mathbb{N}_0 : f(m) = n\}$
Aus den Definitionen und der Funktion f_P ergeben sich $D(f_P), W(f_P)$ zu:
$D(f_P) = \{n \in \mathbb{N}_0 | f_P(n) \neq \perp\} = \{n \in \mathbb{N}_0 | n \geq 2\}$
$W(f_P) = \{n \in \mathbb{N}_0 | \exists m \in \mathbb{N}_0 : f_P(m) = n\} = \{2^n | n \in \mathbb{N}_0 \wedge n \geq 2\} \cup \{\perp\}$

8.1.2

Konstruktionsidee:

1. rechne $x_{diff} = x_2 \dot{-} x_1$
2. wenn $x_{diff} = 0$ gebe x1 aus
3. sonst gebe x2 aus
4. halte an.

Dieses Vorgehen gibt immer den maximalen Wert aus, da für den Fall $x_1 \geq$ die Differenz x_{diff} mit $\dot{-}$ gleich 0 ist und x_1 ausgegeben wird und für den Fall $x_1 < x_2$ die Differenz größer 0 ist und x_2 ausgegeben wird.

Da nun Diese Operationen nicht genau so mit unserer Syntax für GOTO-Programme kompatibel sind müssen wir Anpassungen vornehmen.
So brauchen wir:
Eine Variable x_3 die x_1 kopiert und über die das Subtrahieren von x_1 simuliert werden kann.
Eine Variable x_4 die die Funktion von x_{diff} übernimmt und initial gleich x_2 ist.
Ferner benötigen wir eine Variable x_5 für Schleifensprünge.

Daraus ergibt sich das folgende GOTO-Programm P:

1 : $x_3 := x_1$;
2 : $x_4 := x_2$;
3 : $x_5 := 1$;

4 : IF $x_3 = 0$ THEN GOTO 8;
5 : $x_3 := x_3 \dot{-} 1$;
6 : $x_4 := x_4 \dot{-} 1$;
7 : IF $x_5 = 1$ THEN GOTO 4;

8 : IF $x_4 = 0$ THEN GOTO 10;
9 : $x_1 := x_2$;

10 : $HALT$

(4-6) simulieren $x_{diff} = x_2 - x_1$ und der Ausgabewert x_1 wird nur dann auf x_2 gesetzt, wenn $x_{diff} > 0$ gilt.

8.2

8.2.1

Betrachte:
$w \stackrel{\text{def}}{=} bab$

Ersten 10 Konfigurationen:
$(q_b, (\epsilon, \triangleright, bab)) \vdash (q_b, (\triangleright, b, ab))$
$\vdash (q_r, (\triangleright\underline{b}, a, b))$
$\vdash (q_r, (\triangleright\underline{b}a, b, \epsilon))$
$\vdash (q_r, (\triangleright\underline{b}ab, \sqcup, \epsilon))$
$\vdash (q_l, (\triangleright\underline{b}a, b, \underline{b}))$
$\vdash (q_l, (\triangleright\underline{b}, a, b\underline{b}))$
$\vdash (q_l, (\triangleright, \underline{b}, ab\underline{b}))$
$\vdash (q_l, (\epsilon, \triangleright, \underline{b}ab\underline{b}))$
$\vdash (q_b, (\triangleright, \underline{b}, ab\underline{b}))$

Erste Konfiguration in q_c:
$(q_c, (\triangleright\underline{b}a\underline{b}b, \underline{b}, \sqcup))$

Haltekonfiguration für Eingabe w
$(h, (\epsilon, \triangleright, babbb))$

8.2.2

Bedeutung der Zustände:
q_b

q_b ließt den gesamten String von links nach rechts, bis er endet, bzw. ein $\sqcup$ auftritt, oder ein b gelesen wird.
Wenn ein b gelesen wird, so wird dieses durch ein markiertes b ($\underline{b}$) ersetzt und in den Zustand q_l gewechselt.
Wenn ein $\sqcup$ gelesen wird, hat der String keine unmarkierten b mehr und es wird in den Zustand q_c gewechselt.

q_c

q_c liest den gesamten String von links nach rechts und ersetzt alle markierten b mit unmarkierten, bis das Startsymbol erreicht wird.
In diesem Fall wird aufgehört den Lesekopf zu bewegen und in den Zustand h gewechselt.

q_r

q_r verschiebt den Lesekopf solange nach rechts, bis der String endet, bzw. ein $\sqcup$ gelesen wird.
Dann wird dieses durch ein markiertes b ersetzt, bzw. ein markiertes b an den String hinten ange-

hangen.

q_l

q_l verschiebt den Lesekopf solange nach links, bis das Startsymbol gelesen wird.
Dann wird in den Zustand q_b gewechselt.

h

Im Zustand h hält die Touringmaschine.

Aus der Bedeutung der Zustände geht hervor, dass:
Die Touringmaschine jedes b markiert und anschließend ein markiertes b an den String hinten anhängt, bis es keine unmarkierten b mehr gibt.
Abschließend werden alle markierten b durch normale b ersetzt und die TM hält.
Da nach endlichen Operationen bei endlichen Eingaben alle b markiert worden sind hält die TM auch für alle Eingaben aus $\{a, b\}^*$ und weisst damit keine Definitionslücken auf.

Für die Funktion f_M der TM bedeutet das, dass die Touring maschine aus einem Wort $w \in \{a, b\}^*$ ein Wort $v \in \{a, b\}$, das aus w konkateniert mit $n = \#_b(w)$ b besteht , also der Form $v = wb^{n=\#_b(w)}$ macht.
Dadurch ergibt sich dann auch die Funktion f_M zu:

$$f_M(w) = wb^{n=\#_b(w)}$$

8.3

8.3.1

Eingabewort der Form: $u\$v\$$, mit $|u| = |v|$

Konstruktionsidee:

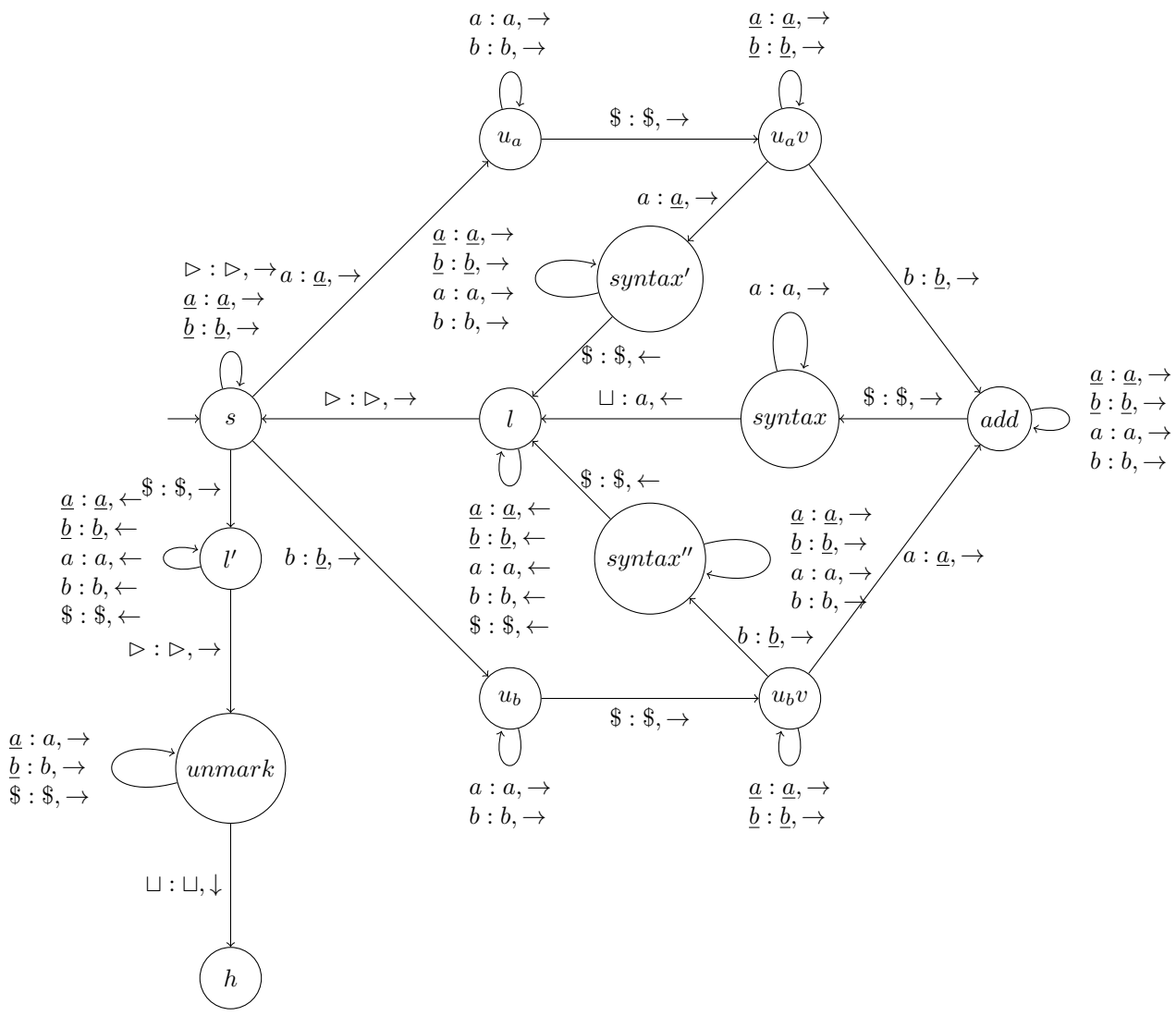
1. betrachte das erste unmarkierte Zeichen aus u und gleiche ab, ob das erste unmarkierte Zeichen aus v gleich diesem ist. Markiere die beiden Zeichen

2. falls nein hänge an das Wort ein a

3. gehe zurück zum Anfang des Wortes und mache weiter bei 1, bis alles Markiert wurde. Dann mache die markierungen rüeckgängig und halte an.

Das Markieren und überlesen der markierten Zustände erfolgt in s, dann wird über u_σ ein $\sigma \in \{a, b\}$ gemerkt und nachdem v in $u_\sigma v$ erreicht wurde alle markierten Zeichen überlesen werden und verglichen wird, ob das erste unmarkierte Zeichen gleich σ ist.
Falls ja wird zurück gegangen. Sonst wird ein a angehangen und dann zurück gegangen. Das anhängen eines a erfolgt über add, das Zurücklaufen über l.
Das rückgängigmachen der Markierung erfolgt über die Zustände $\{l', unmark\}$, nachdem alles markiert wurde.
Das Halten darauf über den Zustand h.
Des weiteren muss die Syntax der eingabe nach \$ gecheckt werden, von $|u| = |v|$ kann nach Aufgabe ausgegangen werden.
Dieses Syntax-checking, ob das eingegebene Wort auch mit \$ endet führt zu den Zuständen $\{syntax, syntax', syntax''\}$

Eine mögliche Lösung wäre die TM $M = (Q, \Gamma, \delta, s)$, mit:
$Q = \{, s, u_a, u_b, u_a v, u_b v, add, l, l', unmark, syntax, syntax', syntax'', h\}$
$\Gamma = \{a, b, \$, \triangleright, \sqcup, \underline{a}, \underline{b}\}$
δ nach Aufgabenstellung nicht formal spezifiziert.
$s = \triangleright$
, mit dem zugehörigem Diagramm:



Alle nicht angegebenen Zustände führen zum halten, bzw. verwerfen der Eingabe.

8.3.2

Es müsste jedes Zeichen gleich sein und ferner keine a angehangen werden.
Der Zustand *add* und alle Transitionen inzident zu diesem können entfernt werden.
Ferner führen nun alle Transitionen, die zuvor nach *add* führten nun nach *nein* und statt markieren bleibt der Lesekopf stehen, da bereits ein ungleiches Zeichen ausreicht, damit das Wort nicht in der Sprache ist.
Zudem muss nun lediglich akzeptiert und nichtmehr demakiert werden, also können $\{l', unmark, h\}$ und ihre inzidenten Transitionen auch gelöscht und durch einen Zustand ja ersetzt werden.

Die resultierende TM hätte damit das zugehörige Diagramm: