Karlsruher Institut für Technologie

Tutorium 42, #6

Max Göckel– *uzkns@student.kit.edu*

Institut für Theoretische Informatik - Grundbegriffe der Informatik

Übungsblatt: Häufige Fehler

- 1.a): $G = (((\neg A) \vee B) \rightarrow B) \rightarrow$ gesamten Audruck umklammern!
- 2) "Beweisen sie induktiv folgende Aussage" $\rightarrow$ Bis zum Ende rechnen und Folgerung aufschreiben!
- 4) Auf den folgenden Folien
- 6.abcd) Begründung $\neq$ Beweis!
- Induktiv definieren ist nicht gleich induktivem Beweisen. man braucht zwar auch einen IA und eine Art IS aber es gilt nicht irgendeine Aussage zu beweisen.

Beweisbarkeit von Aussagen

Drei Axiome:

- $Ax_{AL1} = \{(G \to (H \to G)) | G, H \in For_{AL}\}$
- $Ax_{AL2} = \{(G \to (H \to K)) \to ((G \to H) \to (G \to K)) | G, H, K, \in For_{AL}\}$
- $Ax_{AL3} = \{(\neg H \to \neg G) \to ((\neg H \to G) \to H) | G, H \in For_{AL}\}$

Modus Ponens: $MP = \{(G \to H, G, H) | G, H \in For_{AL}\}$

Beweisbarkeit von Aussagen: Aufgabe aus dem ÜB

Mit den Aximon und ggf. Prämissen kann man jede Aussage G "ableiten".

Prämissen: $\{A \to B, A \to \neg B\}$, Formel: $F = \neg A$

1. $A \to B$, Prämisse P_1
2. $A \to \neg B$, Prämisse P_2
3. $(A \to B) \to ((A \to \neg B) \to \neg A)$, Ax_{AL3} $(G = \neg B, H = \neg A)$
4. $(A \to \neg B) \to \neg A$, MP(3.,1.)
5. $\neg A$, MP(4.,2.)

Induktive Definition

Alphabet A, formale Sprache $L = \{w | w \in A^*, |w| = 2^n, n \in \mathbb{N}_0\}$.

Startwert: $L_1 = A$ Induktiv also: $\forall n \in \mathbb{N}^+ : L_{n+1} = A \cdot L_n$
oder: $\forall n \in \mathbb{N}^+ : L_{n+1} = L_n \cdot L_n$.
Definiert uns L_n für alle $n \in \mathbb{N}^+$

Hire aufhören reicht nicht, wir haben zwar alle L_n definiert aber nicht L insgesamt.

Speicher: Aufgabe

Wie sieht m_3 nach der folgenden verketteten Operation auf Tabelle m_2 aus?

- $m_3 = memwrite(m_2, 11, memread(memwrite(m_2, 10, 000), 00))$

m_2:

Stelle	Wert
00	010
01	000
10	111
11	100

Speicher: Lösung

Wie sieht m_3 nach der folgenden verketteten Operation auf Tabelle m_2 aus?

- $m_3 = memwrite(m_2, 11, memread(memwrite(m_2, 10, 000), 00))$

m_3:

Stelle	Wert
00	010
01	000
10	111
11	010

Die MIMA: Aufgabe

Was tut das folgende Programm?

```
LDC 1
STV 011
LDV 010
NOT
ADD 011
ADD 001
JMN here
LDC 1
STV 111 ;result
JMP end
here: LDC 0
STV 111 ;result
end: HALT
```

Die MIMA: Aufgabe

Beschreibt die folgenden Befehle.

JMP a
LDV a
HALT
ADD a

Die MIMA: Aufgabe

Beschreibt die folgenden Befehle.

JMP a: Springt zum Befehl an Stelle a (oder in GBI: Mit Markierung a)
LDV a: Lädt den Wert an Adresse a in den Akku
HALT: Stoppt die MIMA
ADD a: Addiert den Wert an Adresse a auf den Akku drauf (Ergebnis landet wieder im Akku)

Die MIMA: Aufgabe

Schreibe ein Programm, dass den Wert an Adresse 001 verdreifacht.

Die MIMA: Lösung

Schreibe ein Programm, dass den Wert an Adresse 001 verdreifacht.

```
LDV 001
ADD 001
ADD 001
STV 001
HALT.
```

Die MIMA: Aufgabe

Was tut das folgende Programm?

```
LDV adr1
NOT
STV adr2
LDC 1
ADD adr2
STV adr3
HALT
```

Die MIMA: Aufgabe

Was tut das folgende Programm?

Es bildet das Zweierkomplement von adr1 und speichert es an adr3.

Die MIMA: Aufgaben

1) Schreibe ein Programm dass für ein $x \geq 0$ an Adresse adr *x div 2* ausführt und und das Ergebnis wieder in adr abspeichert.
2) Schreibe ein Programm, das für den Wert x an der Adresse *001* x mod 2 berechnet und das Ergebnis wieder in x speichert.
3) An Speicherstelle adr1 steht ein Wert $x \geq 1$.
Schreibe ein Programm, dass den Wert an adr2 mit x multipliziert (adr $\cdot$ x)

Die Lösungen stehen im ILIAS.

Die MIMA: Befehle

LDC c - lädt Konstante c in den Akku
LDV a - Lädt Wert an Stelle a in den Akku
STV a - speicher Wert vom Akku in Stelle a
ADD a - Addiert Wert an Stelle a auf den Wert im Akku drauf (Ergebn. $\rightarrow$ Akku)
AND a - VerUNDet Wert v.St. a und Akku $\rightarrow$ Akku
OR a - VerODERt Wert v.St. a und Akku $\rightarrow$ Akku
XOR a - VerXORt Wert v.St. a und Akku $\rightarrow$ Akku
EQL a - Falls Akku = Wert a.St. a $\rightarrow$ Akku -1 in Akku, sonst 0
JMP a - Springt zum Befehl mit Marker a
JMN a - Springt zum Befehl a wenn Akku < 0
HALT - Endet die Auführung der MIMA
NOT - Invertiert den Akku (0 $\rightarrow$ 1, 1 $\rightarrow$ 0)
RAR - Rotiert den Akku um eins nach rechts

Der Wert des Akku wird immer überschreiben wenn man neue Werte lädt, also Werte die man mehrfach braucht zwischenspeichern (STV).