Grundbegriffe der Informatik Crashkurs WS 2017/18

fuks e.V. - Fachübergreifende Unternehmensberatung Karlsruher Studenten

Miguel Santos Correa

miguelsantoscorrea@gmail.com

https://github.com/miguel-sc/GBI-Crashkurs

Gliederung

Prädikatenlogik

Prädikatenlogik

Drei Schritte:

- Definiert Terme, die aus Konstanten, Variablen und Funktionssymbolen zusammengesetzt sind.
- Mit Hilfe von Relationssymbolen und Termen konstruiert man dann atomare Formeln.
- Mit Hilfe von zwei Quantoren werden allgemeine prädikatenlogische Formeln konstruiert.

Beispiel für Terme

- c
- y
- $g(x)$
- $f(x, g(z))$
- $f(c, g(g(z)))$

Atomare Formeln

- Relationssymbole mit Alphabet Rel_{PL}
- kurz R,S,..
- $\doteq$ Relation immer dabei

Atomare Formeln

korrekte Formeln:

- $R(y, c, g(x))$
- $f(x, y) \doteq g(z)$
- $S(c)$

syntaktisch falsche Formeln:

- $S(x) \doteq S(x)$
- $f(R(x, c))$
- $R(S(x), c, y)$
- $x \doteq y \doteq z$

Quantoren

- Allquantor $\forall$
- Existenzquantor $\exists$
- Klammerregel: Quantoren binden am stärksten
- $(\exists x F)$ statt $(\neg(\forall x(\neg F)))$
- $A_{For} = A_{Rel} \cup \{\neg, \wedge, \vee, \rightarrow, \forall, \exists\}$
- z.B. $\forall x R(x, y) \wedge S(x)$

Interpretation

Es seien Alphabete $Const_{PL}$, Fun_{PL} und Rel_{PL} gegeben.
Sind eine Interpretation (D,I) und eine Variablenbelegung β festgelegt, so kann man

- jedem Term einen Wert aus D und
- jeder Formel einen Wahrheitswert zuordnen.

Interpretation von Termen

$$val_{D,I,\beta}(t) = \begin{cases} \beta(x_i), \text{ falls } t = x_i \in Var_{PL} \\ I(c_i), \text{ falls } t = c_i \in Const_{PL} \\ I(f_i)(val_{D,I,\beta}(t_1), .., val_{D,I,\beta}(t_k)), \text{ falls } t = f_i(t_1, ..., t_k) \end{cases}$$

Interpretation von atomaren Formeln

$$val_{D,I,\beta}(R_i(t_1, ..., t_k)) = \begin{cases} w, \text{ falls } (val_{D,I,\beta}(t_1), ..., val_{D,I,\beta}(t_k)) \in I(R_i)) \\ f, \text{ falls } (val_{D,I,\beta}(t_1), ..., val_{D,I,\beta}(t_k)) \notin I(R_i)) \end{cases}$$

$$val_{D,I,\beta}(t_1 \doteq t_2) = \begin{cases} w, \text{ falls } val_{D,I,\beta}(t_1) = val_{D,I,\beta}(t_2) \\ f, \text{ falls } val_{D,I,\beta}(t_1) \neq val_{D,I,\beta}(t_2) \end{cases}$$

Quantoren

$$\beta_{x_i}^{d} : Var_{PL} \rightarrow D : x_j \rightarrow \begin{cases} \beta(x_j), \text{ falls } i \neq j \\ d, \text{ falls } i = j \end{cases}$$

$$val_{D,I,\beta}(\forall x_i F) = \begin{cases} w, \text{ falls für jedes } d \in D \text{ und } \beta' = \beta_{x_i}^{d} : val_{D,I,\beta'}(F) = w \\ f, \text{ sonst} \end{cases}$$

$$val_{D,I,\beta}(\exists x_i F) = \begin{cases} w, \text{ falls für mind. } d \in D \text{ und } \beta' = \beta_{x_i}^{d} : val_{D,I,\beta'}(F) = w \\ f, \text{ sonst} \end{cases}$$

Beispiele

- $D = \mathbb{N}_0$, $I(c) = 0$, $I(f)$ Addition, $I(R)$ kleiner oder gleich
- $\beta(x) = 3$, $\beta(y) = 42$
- $val_{D,I,\beta}(R(y, c)) =?$

Beispiele

- $D = \mathbb{N}_0$, $I(c) = 0$, $I(f)$ Addition, $I(R)$ kleiner oder gleich
- $\beta(x) = 3$, $\beta(y) = 42$
- $val_{D,I,\beta}(R(y,c)) = f$
- weil $\beta(y) > I(c)$

Beispiele

- $D = \{a, b\}^+$, $I(c) = bb$, $I(f)$ Konkatenation
- $I(R)$ hat gleich viele a's
- $\beta(x) = a$, $\beta(y) = abba$
- $val_{D,I,\beta}(f(f(x, c), y) =?$
- $val_{D,I,\beta}(R(f(y, x), c)) =?$

Beispiele

- $D = \{a, b\}^+$, $I(c) = bb$, $I(f)$ Konkatenation
- $I(R)$ hat gleich viele a's
- $\beta(x) = a$, $\beta(y) = abba$
- $val_{D,I,\beta}(f(f(x, c), y) = abbabba$
- $val_{D,I,\beta}(R(f(y, x), c)) = f$

Allgemeingültigkeit

Eine prädikatenlogische Formel heißt allgemeingültig, wenn (D, I) und jede passende Variablenbelegung β gilt: $val_{D,I,\beta}(F) = w$.
Bsp.

$$(\forall x_i(G \rightarrow H)) \rightarrow ((\forall x_i G) \rightarrow (\forall x_i H))$$

freie und gebundene Vorkommen

Wenn in einer prädikatenlogischen Formel G in einem Term eine Variable x steht, dann spricht man auch von einem Vorkommen der Variablen x in G. (Die Anwesenheit einer Variablen unmittelbar hinter einem Quantor zählt nicht als Vorkommen.)

freie und gebundene Vorkommen

- Für jede Formel G, die atomar ist, sind alle Vorkommen von Variablen frei
- Für jede Formel G der Form $(\forall x_i H)$ oder $(\exists x_i H)$ ist ist jedes Vorkommen von x in H gebunden
- Beispiel: $\forall x(R(x,y) \wedge \exists y R(x,y))$

Substitutionen

Es ist möglich Variablen einer prädikatenlogischen Formel durch Terme zu ersetzen. Eine Substitution ist eine Abbildung $\sigma : Var_{PL} \rightarrow L_{Ter}$
$\sigma_{\{x/c,y/f(x)\}}$:

$$\sigma(x) = c$$
$$\sigma(y) = f(x)$$
$$\sigma(z) = z, z \notin \{x, y\}$$

Kollisionsfrei

Eine Substitution σ heiße kollisionsfrei für eine Formel G, wenn für jede Variable x_i, die durch σ verändert wird (also $\sigma(x_i) \neq x_i$) und jede Stelle eines freien Vorkommens von x_i in G gilt: Diese Stelle liegt nicht im Wirkungsbereich eines Quantors $\forall x_j$ oder $\exists x_j$, wenn x_j eine Variable ist, die in $\sigma(x_i)$ vorkommt.

Beispiel

Kollisionsfrei:

- $G = S(x) \wedge \forall x(R(x, y))$
- $\sigma_{\{x/f(y), y/c\}}(G) = S(f(y)) \wedge \forall x(R(x, c))$

nicht Kollisionsfrei:

- $G = \exists y(R(y, c) \wedge R(x, c))$
- $\sigma_{\{x/f(y), y/c\}}(G) = \exists y(R(y, c) \wedge R(f(y), c))$

Klausuraufgabe: WS 2015/16 A2

Beantworten Sie für jede der folgenden prädikatenlogischen Formeln die Frage: „Ist die Formel allgemeingültig?“

(i) $(\neg\exists x : P(x)) \leftrightarrow (\forall x : \neg P(x))$

(ii) $(\forall x \exists y \exists z : Q(x, y, z)) \rightarrow (\exists y \forall x \exists z : Q(x, y, z))$

(iii) $(\exists z \exists y \forall x : Q(x, y, z)) \rightarrow (\forall x \exists y \exists z : Q(x, y, z))$

Dabei ist P ein einstelliges Relationssymbol und Q ein dreistelliges Relationssymbol.

Klausuraufgabe: WS 2015/16 A2

Beantworten Sie für jede der folgenden prädikatenlogischen Formeln die Frage: „Ist die Formel allgemeingültig?“

(i) $(\neg\exists x : P(x)) \leftrightarrow (\forall x : \neg P(x))$

(ii) $(\forall x \exists y \exists z : Q(x,y,z)) \rightarrow (\exists y \forall x \exists z : Q(x,y,z))$

(iii) $(\exists z \exists y \forall x : Q(x,y,z)) \rightarrow (\forall x \exists y \exists z : Q(x,y,z))$

Dabei ist P ein einstelliges Relationssymbol und Q ein dreistelliges Relationssymbol.

(i) Ja

Klausuraufgabe: WS 2015/16 A2

Beantworten Sie für jede der folgenden prädikatenlogischen Formeln die Frage: „Ist die Formel allgemeingültig?“

(i) $(\neg\exists x : P(x)) \leftrightarrow (\forall x : \neg P(x))$

(ii) $(\forall x \exists y \exists z : Q(x,y,z)) \rightarrow (\exists y \forall x \exists z : Q(x,y,z))$

(iii) $(\exists z \exists y \forall x : Q(x,y,z)) \rightarrow (\forall x \exists y \exists z : Q(x,y,z))$

Dabei ist P ein einstelliges Relationssymbol und Q ein dreistelliges Relationssymbol.

(i) Ja

(ii) Nein

Klausuraufgabe: WS 2015/16 A2

Beantworten Sie für jede der folgenden prädikatenlogischen Formeln die Frage: „Ist die Formel allgemeingültig?“

(i) $(\neg\exists x : P(x)) \leftrightarrow (\forall x : \neg P(x))$

(ii) $(\forall x \exists y \exists z : Q(x, y, z)) \rightarrow (\exists y \forall x \exists z : Q(x, y, z))$

(iii) $(\exists z \exists y \forall x : Q(x, y, z)) \rightarrow (\forall x \exists y \exists z : Q(x, y, z))$

Dabei ist P ein einstelliges Relationssymbol und Q ein dreistelliges Relationssymbol.

(i) Ja

(ii) Nein

(iii) Ja

Klausuraufgabe: WS 2015/16 A2

Formulieren Sie die folgenden Aussagen als prädikatenlogische Formeln über dem Universum aller Menschen:

(i) Jeder Student außer Tom lächelt.

(ii) Jeder mag jeden, der sich nicht selbst mag.

Klausuraufgabe: WS 2015/16 A2

Formulieren Sie die folgenden Aussagen als prädikatenlogische Formeln über dem Universum aller Menschen:

(i) Jeder Student außer Tom lächelt.

(ii) Jeder mag jeden, der sich nicht selbst mag.

(i) $\forall x(\mathit{student}(x) \rightarrow (\neg \mathit{Tom}(x) \leftrightarrow \mathit{l\ddot{a}chelt}(x)))$

Klausuraufgabe: WS 2015/16 A2

Formulieren Sie die folgenden Aussagen als prädikatenlogische Formeln über dem Universum aller Menschen:

(i) Jeder Student außer Tom lächelt.

(ii) Jeder mag jeden, der sich nicht selbst mag.

(i) $\forall x(\mathit{student}(x) \rightarrow (\neg \mathit{Tom}(x) \leftrightarrow \mathit{lächelt}(x)))$

(ii) $\forall x \forall y(\neg \mathit{mag}(y, y) \rightarrow \mathit{mag}(x, y))$

Klausuraufgabe: WS 2015/16 A2

Gegeben sei die prädikatenlogische Formel

$$\forall x \forall y (R(x,y) \rightarrow R(f(x), f(y)))$$

und eine Interpretation (D, I) dafür, wobei das Universum D die Menge $\{a, b\}$ sei und die Interpretationsabbildung I gegeben sei durch $I(f)(a) = b$, $I(f)(b) = a$ und $I(R) = \{(a,a), (a,b)\}$.

(i) Geben Sie den Wahrheitswert der Formel in der Interpretation an.

(ii) Erläutern sie kurz Ihre Antwort aus Teil (i):

Klausuraufgabe: WS 2015/16 A2

Gegeben sei die prädikatenlogische Formel

$$\forall x \forall y (R(x, y) \to R(f(x), f(y)))$$

und eine Interpretation (D, I) dafür, wobei das Universum D die Menge $\{a, b\}$ sei und die Interpretationsabbildung I gegeben sei durch $I(f)(a) = b$, $I(f)(b) = a$ und $I(R) = \{(a, a), (a, b)\}$.

(i) Geben Sie den Wahrheitswert der Formel in der Interpretation an.

(ii) Erläutern sie kurz Ihre Antwort aus Teil (i):

Wahrheitswert: Falsch.

Speicher

Menge aller Abbildungen

Die Menge aller Abbildungen von A nach B ist definiert als B^A.

- $B^A = \{f : A \to B\}$
- $|B^A| = |B|^{|A|}$
- Für jedes $a \in A$ gibt es $|B|$ Möglichkeiten einen Funktionswert aus B zu finden.
- Es gibt $|A|$ Elemente in A, also ergibt sich für die Anzahl aller Kombinationen: $\underbrace{|B| \cdot |B| \cdots |B|}_{|A| \text{ mal}}$

Funktionen als Argument einer Funktion

Beispiel:

- Es sei M eine Menge
- Jede Teilmenge $L \subseteq M$ kann man bijektiv mit einer Abbildung $f : M \to \{0, 1\}$, also einem $f \in \{0, 1\}^M$ in Beziehung setzen.
- z.B. $L = \{2\} \subseteq M = \{1, 2, 3\}$ also $f(1) = 0, f(2) = 1, f(3) = 0$
- Dann kann man die Vereinigung als Abbildung $V : \{0, 1\}^M \times \{0, 1\}^M \to \{0, 1\}^M$ auffassen.

Funktionen als Argument einer Funktion

Beispielbild für $L_1 = \{a, c, d\}$ und $L_2 = \{b, c\}$

	L_1	L_2	$L_1 \cup L_2$
x	$f_1(x)$	$f_2(x)$	$V(f_1, f_2)$
a	1	0	1
b	0	1	1
c	1	1	1
d	1	0	1
e	0	0	0

Wie definiert man $V(f_1, f_2)$?

Funktionen als Argument einer Funktion

Beispielbild für $L_1 = \{a, c, d\}$ und $L_2 = \{b, c\}$

	L_1	L_2	$L_1 \cup L_2$
x	$f_1(x)$	$f_2(x)$	$V(f_1, f_2)$
a	1	0	1
b	0	1	1
c	1	1	1
d	1	0	1
e	0	0	0

Wie definiert man $V(f_1, f_2)$?

Zum Beispiel so: $V(f_1, f_2)(x) = \max(f_1(x), f_2(x))$

Speicher

Ein Speicher kann als Abbildung von einer Menge aus Adressen nach einer Menge an Speicherwerten aufgefasst werden.
$m : \text{Adr} \rightarrow \text{Val}$

Adresse 1	Wert 1
Adresse 2	Wert 2
Adresse 3	Wert 3
Adresse 4	Wert 4
⋮	⋮
Adresse n	Wert n

000	10110101
001	10101101
010	10011101
011	01110110
100	00111110
101	10101101
110	00101011
111	10101001

Speicher

Das Auslesen eines Wertes ist realisiert durch die memread Funktion

$$\begin{aligned} \mathsf{memread} : \mathsf{Val}^{\mathsf{Adr}} \times \mathsf{Adr} &\to \mathsf{Val} \\ (m, a) &\mapsto m(a) \end{aligned}$$

Speicher

memwrite speichert den Wert v in Adresse a:

$$\begin{aligned}\text{memwrite} : \text{Val}^{\text{Adr}} \times \text{Adr} \times \text{Val} &\to \text{Val}^{\text{Adr}} \\ (m, a, v) &\mapsto m'\end{aligned}$$

Speicher

Eigenschaften von memread und memwrite:

- $\text{memread}(\text{memwrite}(m, a, v), a) = v$
- $\text{memread}(\text{memwrite}(m, a', v'), a) = \text{memread}(m, a)$
- Auslesen einer Speicherstelle ist unabhängig davon, was vorher an einer anderen Adresse gespeichert war.

Speicher

Die Tabelle sei unser Speicher im Zustand $m \in Mem = \mathrm{Val}^{\mathrm{Adr}}$

000	10110101
001	10101101
010	10011101
011	01110110
100	00111110
101	10101101
110	00101011
111	10101001

- $\mathrm{memread}(m, 100) = ?$
- $\mathrm{memwrite}(m, 110, 00011000) = ?$
- $\mathrm{memread}(\mathrm{memwrite}(m, 001, 00110011), 001) = ?$
- $m(101) = ?$

Speicher

Die Tabelle sei unser Speicher im Zustand $m \in Mem = \mathrm{Val}^{\mathrm{Adr}}$

000	10110101
001	10101101
010	10011101
011	01110110
100	00111110
101	10101101
110	00101011
111	10101001

- $\mathrm{memread}(m, 100) = 00111110$
- $\mathrm{memwrite}(m, 110, 00011000) = m'$
- $\mathrm{memread}(\mathrm{memwrite}(m, 001, 00110011), 001) = 00110011$
- $m(101) = 10101101$

MIMA

MIMA

- Adressen: 20 bit
- Speicherwerte: 24 bit
- (die meisten) Befehle: 4 bit + 20 bit
- z.B. LDC 10

Beispiel: Multiplikation

```
        LDC 0
        STV prod
start : LDC 0
        NOT
        ADD b
        STV b
        JMN end
        LDV prod
        ADD a
        STV prod
        JMP start
end :   HALT
```

Befehle

- **LDC const** Lädt eine 20 bit Zahl in den Akkumulator.

Befehle

- **LDC const** Lädt eine 20 bit Zahl in den Akkumulator.
- **LDV adr** Lädt den Speicherwert von adr in den Akkumulator.

Befehle

- **LDC const** Lädt eine 20 bit Zahl in den Akkumulator.
- **LDV adr** Lädt den Speicherwert von adr in den Akkumulator.
- **STV adr** Speichert den Wert vom Akkumulator in adr.

Befehle

- **LDC const** Lädt eine 20 bit Zahl in den Akkumulator.
- **LDV adr** Lädt den Speicherwert von adr in den Akkumulator.
- **STV adr** Speichert den Wert vom Akkumulator in adr.
- **LDIV adr** Lädt den Speicherwert vom Speicherwert von adr $M(M(adr))$ in den Akkumulator.

Befehle

- **LDC const** Lädt eine 20 bit Zahl in den Akkumulator.
- **LDV adr** Lädt den Speicherwert von adr in den Akkumulator.
- **STV adr** Speichert den Wert vom Akkumulator in adr.
- **LDIV adr** Lädt den Speicherwert vom Speicherwert von adr $M(M(adr))$ in den Akkumulator.
- **STIV adr** Speichert den Wert vom Akkumulator in $M(M(adr))$.

Befehle

- **LDC const** Lädt eine 20 bit Zahl in den Akkumulator.
- **LDV adr** Lädt den Speicherwert von adr in den Akkumulator.
- **STV adr** Speichert den Wert vom Akkumulator in adr.
- **LDIV adr** Lädt den Speicherwert vom Speicherwert von adr $M(M(adr))$ in den Akkumulator.
- **STIV adr** Speichert den Wert vom Akkumulator in $M(M(adr))$.
- **ADD adr** Addiert den Speicherwert von adr auf den Akkumulator und speichert das Ergebnis im Akkumulator.

Befehle

- **AND adr** Bitweise AND vom Speicherwert von adr mit dem Akkumulator. Ergebnis wird im Akkumulator gespeichert.

Befehle

- **AND adr** Bitweise AND vom Speicherwert von adr mit dem Akkumulator. Ergebnis wird im Akkumulator gespeichert.
- **OR adr** Bitweise OR.

Befehle

- **AND adr** Bitweise AND vom Speicherwert von adr mit dem Akkumulator. Ergebnis wird im Akkumulator gespeichert.
- **OR adr** Bitweise OR.
- **XOR adr** Bitweise XOR.

Befehle

- **AND adr** Bitweise AND vom Speicherwert von adr mit dem Akkumulator. Ergebnis wird im Akkumulator gespeichert.
- **OR adr** Bitweise OR.
- **XOR adr** Bitweise XOR.
- **NOT** Invertiert die Bits des Akkumulators.

Befehle

- **RAR** Rotation der Akku-Bits nach rechts. Beispiel:
  $101100 \rightarrow 010110$

Befehle

- **RAR** Rotation der Akku-Bits nach rechts. Beispiel: $101100 \rightarrow 010110$
- **EQL adr** Vergleicht den Speicherwert von adr mit dem Akkumulator. Wenn die Zahlen gleich sind wird die Zweierkomplementdarstellung von -1 im Akkumulator gespeichert, wenn nicht dann wird die Zahl 0 im Akkumulator gespeichert.

Befehle

- **RAR** Rotation der Akku-Bits nach rechts. Beispiel: $101100 \rightarrow 010110$
- **EQL adr** Vergleicht den Speicherwert von adr mit dem Akkumulator. Wenn die Zahlen gleich sind wird die Zweierkomplementdarstellung von -1 im Akkumulator gespeichert, wenn nicht dann wird die Zahl 0 im Akkumulator gespeichert.
- **JMP adr** Programm springt an die Adresse adr und setzt mit dem Befehl in adr fort.

Befehle

- **RAR** Rotation der Akku-Bits nach rechts. Beispiel: $101100 \rightarrow 010110$
- **EQL adr** Vergleicht den Speicherwert von adr mit dem Akkumulator. Wenn die Zahlen gleich sind wird die Zweierkomplementdarstellung von -1 im Akkumulator gespeichert, wenn nicht dann wird die Zahl 0 im Akkumulator gespeichert.
- **JMP adr** Programm springt an die Adresse adr und setzt mit dem Befehl in adr fort.
- **JMN adr** Programm springt an die Adresse adr, falls der Akkumulator negativ ist.

Zweierkomplement

- positive Zahlen bleiben unverändert
- $\mathsf{Zkpl}_5(6) = 00110, \mathsf{Zkpl}_5(15) = 01111$
- negative Zahlen: Alle Bits umdrehen und 1 addieren
- $\mathsf{Zkpl}_5(-6) = 11010, \mathsf{Zkpl}_5(-15) = 10001$
- $01111 + 10001 = (1)00000$

Wichtige Blöcke

Lade -1:

$$LDC\ 0$$
$$NOT$$

Wichtige Blöcke

Addiere Konstante c:

$$LDC\ c$$
$$ADD\ a$$
$$STV\ a$$

Wichtige Blöcke

Addiere negative Konstante $-c$:

$$
\begin{aligned}
&LDC\ c-1\\
&NOT\\
&ADD\ a\\
&STV\ a
\end{aligned}
$$

Wichtige Blöcke

Addiere negative Konstante -5:

$$\begin{aligned}&LDC\ 4\\&NOT\\&ADD\ a\\&STV\ a\end{aligned}$$

Wichtige Blöcke

Verwandle $a \rightarrow -a$:

LDV a
NOT
STV a
LDC 1
ADD a
STV a

Wichtige Blöcke

n Schleifendurchläufe:

$$\begin{aligned} start :& LDC\ 0 \\ & NOT \\ & ADD\ n \\ & STV\ n \\ & JMN\ end \\ & Block \\ & JMP\ start \end{aligned}$$

Beispiel: Division

```
        LDC 0
        STV div
        LDV b
        NOT
        STV b
        LDC 1
        ADD b
        STV b

start : LDV a
        ADD b
        STV a
        JMN end
        LDC 1
        ADD div
        STV div
        JMP start
  end : HALT
```

Aufgabenblatt: WS 2015/16 A5.2

Es seien a_1 und a_2 zwei verschiedene 20bit Adressen. Im Speicher stehe in Adresse a_1 die Zweierkomplementdarstellung einer nicht-negativen ganzen Zahl x, für die 2^x mit 24bit in Zweierkomplementdarstellung darstellbar ist. Ergänzen Sie die fehlenden Konstanten und Adressen im unvollständigen Minimalmaschinenprogramm derart, dass nach dessen Ausführung 2^x in Zweierkomplementdarstellung im Speicher bei Adresse a_2 steht. Beachten Sie, dass alle arithmetischen Ausdrücke, in denen x vorkommt, keine Konstanten sind, und, dass $2^0 = 1$ gilt.

Aufgabenblatt: WS 2015/16 A5.2

$$
\begin{aligned}
& LDC \\
& STV \\
while :& LDC \\
& NOT \\
& ADD \\
& STV \\
& JMN\ end \\
& LDV \\
& ADD \\
& STV \\
& JMP\ while \\
end :& HALT
\end{aligned}
$$

Aufgabenblatt: WS 2015/16 A5.2

$$
\begin{aligned}
& LDC\ 1 \\
& STV\ a_2 \\
while:\ & LDC \\
& NOT \\
& ADD \\
& STV \\
& JMN\ end \\
& LDV \\
& ADD \\
& STV \\
& JMP\ while \\
end:\ & HALT
\end{aligned}
$$

Aufgabenblatt: WS 2015/16 A5.2

$$
\begin{aligned}
& LDC\ 1 \\
& STV\ a_2 \\
while:\ & LDC\ 0 \\
& NOT \\
& ADD\ a_1 \\
& STV\ a_1 \\
& JMN\ end \\
& LDV \\
& ADD \\
& STV \\
& JMP\ while \\
end:\ & HALT
\end{aligned}
$$

Aufgabenblatt: WS 2015/16 A5.2

$$
\begin{aligned}
& LDC\ 1 \\
& STV\ a_2 \\
while :& LDC\ 0 \\
& NOT \\
& ADD\ a_1 \\
& STV\ a_1 \\
& JMN\ end \\
& LDV\ a_2 \\
& ADD\ a_2 \\
& STV\ a_2 \\
& JMP\ while \\
end :& HALT
\end{aligned}
$$

Aufgabenblatt: WS 2014/15 A5.2

Es seien a_1 und a_2 zwei verschiedene Adressen. Welche Zahlen in Zweierkomplementdarstellung stehen nach Ausführung des Programms in den Adressen a_1 und a_2 im Speicher?

$$LDV\ a_1$$
$$XOR\ a_2$$
$$STV\ a_1$$
$$LDV\ a_2$$
$$XOR\ a_1$$
$$STV\ a_2$$
$$LDV\ a_1$$
$$XOR\ a_2$$
$$STV\ a_1$$

Aufgabenblatt: WS 2014/15 A5.2

Beispiel mit zwei 8 bit Zahlen:

a_1	a_2
11101001	00100111
11001110	00100111
11001110	11101001
00100111	11101001

Aufgabenblatt: WS 2014/15 A5.2

Beispiel mit zwei 8 bit Zahlen:

a_1	a_2
11101001	00100111
11001110	00100111
11001110	11101001
00100111	11101001

Die Speicherwerte der Adressen a_1 und a_2 werden vertauscht.

Aufgaben

Aufgabenblatt: WS 2014/15 A6.4

$\{z = a(0) \wedge x = 1\}$

while $x \leqslant n - 1$

do

  if $a(x) \leqslant z$ **then**

    $z \leftarrow a(x)$

  else

    $z \leftarrow z$

  fi

  $x \leftarrow x + 1$

od

$\{z = \min\limits_{i \in \mathbb{Z}_n} a(i)\}$

Beispiel: While-Schleife

$\{I\}$

while B

do

$\quad\{I \wedge B\}$

$\quad S$

$\quad\{I\}$

od

$\{I \wedge \neg B\}$

$\{z = a(0) \wedge x = 1\}$

while $x \leqslant n - 1$

do

  if $a(x) \leqslant z$ **then**

    $z \leftarrow a(x)$

  else

    $z \leftarrow z$

  fi

  $x \leftarrow x + 1$

  $\{z = \min_{i \in \mathbb{Z}_x} a(i)\}$

od

$\{z = \min_{i \in \mathbb{Z}_n} a(i)\}$

$\{z = a(0) \wedge x = 1\}$

while $x \leqslant n - 1$

do

  if $a(x) \leqslant z$ **then**

    $z \leftarrow a(x)$

  else

    $z \leftarrow z$

  fi

  $x \leftarrow x + 1$

  $\{z = \min_{i \in \mathbb{Z}_x} a(i)\}$

od

$\{z = \min_{i \in \mathbb{Z}_x} a(i) \wedge \neg(x \leqslant n - 1)\}$

$\{z = \min_{i \in \mathbb{Z}_n} a(i)\}$

$\{z = a(0) \wedge x = 1\}$

while $x \leqslant n - 1$

do

  if $a(x) \leqslant z$ **then**

    $z \leftarrow a(x)$

  else

    $z \leftarrow z$

  fi

  $x \leftarrow x + 1$

  $\{z = \min_{i \in \mathbb{Z}_x} a(i) \wedge x \leqslant n\}$

od

$\{z = \min_{i \in \mathbb{Z}_x} a(i) \wedge x \leqslant n \wedge \neg(x \leqslant n - 1)\}$

$\{z = \min_{i \in \mathbb{Z}_n} a(i)\}$

Master-Theorem

$T(n) = a \cdot T(\frac{n}{b}) + f(n)$ mit $a \geqslant 1$ und $b > 1$

- Fall 1: Wenn $f \in O(n^{\log_b a - \epsilon})$ für ein $\epsilon > 0$ ist, dann ist $T \in \Theta(n^{\log_b a})$.
- Fall 2: Wenn $f \in \Theta(n^{\log_b a})$ ist, dann ist $T \in \Theta(n^{\log_b a} \log n)$.
- Fall 3: Wenn $f \in \Omega(n^{\log_b a + \epsilon})$ für ein $\epsilon > 0$ ist, und wenn es eine Konstante d gibt mit $0 < d < 1$, so dass für alle hinreichend großen n gilt $af(n/b) \leqslant df$, dann ist $T \in \Theta(f)$.

Master-Theorem

Beispiel:

- $T(n) = 4T(\frac{n}{2}) + n \log n$
- $n^{\log_b a} = n^{\log_2 4} = n^2$ und $f(n) = n \log n$
- $n \log n \in O(n^{2-\epsilon})$
- $\Rightarrow T(n) \in \Theta(n^2)$

Allgemein:

- f(n) und $n^{\log_b a}$ vergleichen
- den passenden Fall benutzen

Master-Theorem

Schätzen Sie T(n) mit Hilfe des Master-Theorems ab, falls das Master-Theorem anwendbar ist.

- $T(n) = 9T(\frac{n}{3}) + n^2 + 2n + 1$
- $T(n) = \sqrt{3}T(\frac{n}{2}) + \log n$
- $T(n) = 2^n T(\frac{n}{2}) + n$

Master-Theorem

Schätzen Sie T(n) mit Hilfe des Master-Theorems ab, falls das Master-Theorem anwendbar ist.

- $T(n) = 9T(\frac{n}{3}) + n^2 + 2n + 1$
  Lösung: $n^{\log_3 9} = n^2,\ T(n) \in \Theta(n^2 \log n)$

Master-Theorem

Schätzen Sie T(n) mit Hilfe des Master-Theorems ab, falls das Master-Theorem anwendbar ist.

- $T(n) = 9T(\frac{n}{3}) + n^2 + 2n + 1$
  Lösung: $n^{\log_3 9} = n^2,\ T(n) \in \Theta(n^2 \log n)$
- $T(n) = \sqrt{3}T(\frac{n}{2}) + \log n$
  Lösung: $n^{\log_2 \sqrt{3}} = n^{0,79},\ T(n) \in \Theta(n^{\log_2 \sqrt{3}})$

Master-Theorem

Schätzen Sie T(n) mit Hilfe des Master-Theorems ab, falls das Master-Theorem anwendbar ist.

- $T(n) = 9T(\frac{n}{3}) + n^2 + 2n + 1$
  Lösung: $n^{\log_3 9} = n^2$, $T(n) \in \Theta(n^2 \log n)$
- $T(n) = \sqrt{3}T(\frac{n}{2}) + \log n$
  Lösung: $n^{\log_2 \sqrt{3}} = n^{0,79}$, $T(n) \in \Theta(n^{\log_2 \sqrt{3}})$
- $T(n) = 2^n T(\frac{n}{2}) + n$
  Lösung: Nicht anwendbar, $a = 2^n$ nicht konstant

Induktion über Wortlänge

Es sei $f : A^* \to \mathbb{N}$ mit $A = \{a, b\}$

- $f(\epsilon) = 0$
- $\forall x \in A$ und $\forall w \in A^* : f(xw) = u(x) + f(w)$
- $u(a) = 1$ $u(b) = 0$

Beweisen Sie durch vollständige Induktion, dass $f(w) = N_a(w)$ gilt.
A_n: die Aussage $f(w) = N_a(w)$ mit $|w| = n$

Induktion über Wortlänge

Induktion über die Wortlänge $n = |w|$
Induktionsanfang $n = 0$:
$\rightarrow w = \epsilon$
$f(\epsilon) = 0$
$N_a(\epsilon) = 0$

Induktion über Wortlänge

Induktionsvorraussetzung:
Für ein beliebiges aber festes n gilt $f(w) = N_a(w)$ für alle $w \in A^*$ mit $|w| = n$
Neu: A_n ist wahr, also $f(w) = N_a(w)$ für alle $w \in A^*$ mit $|w| = n$

Induktion über Wortlänge

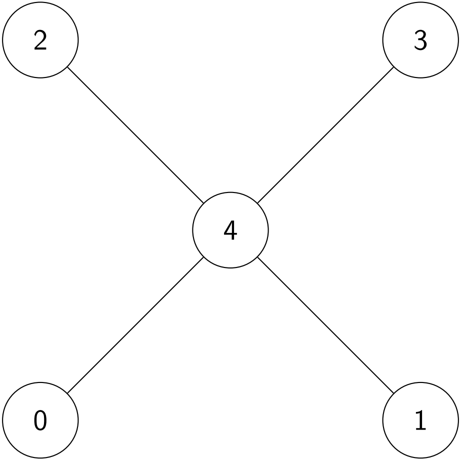
Induktionsschritt:
Es sei $w = xw'$ mit $|w| = n + 1$
$f(w) = f(xw') = u(x) + f(w')$
$f(w) = u(x) + N_a(w')$ (I.V.)
$f(w) = N_a(x) + N_a(w')$
$f(w) = N_a(xw') = N_a(w)$

Klausuraufgabe: WS 2016/17 A4

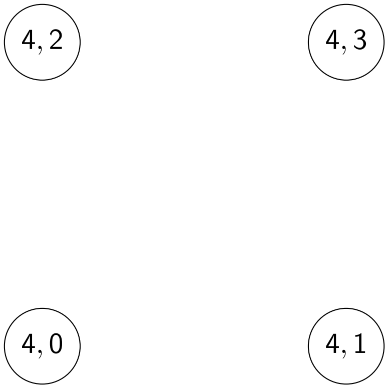
Für jeden ungerichteten Graphen $G = (V, E)$ ist der sogenannte Kantengraph $L(G) = (V', E')$ wie folgt definiert:

$$V' = E$$
$$E' = \{\{e_1, e_2\} | e_1, e_2 \in E \wedge e_1 \cap e_2 \notin \emptyset\}$$

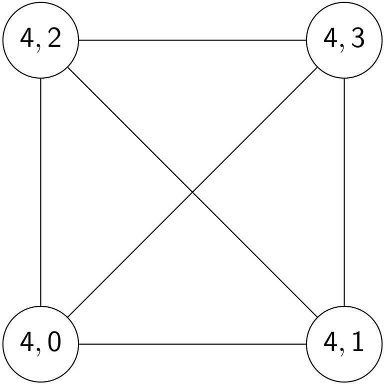
Klausuraufgabe: WS 2016/17 A4



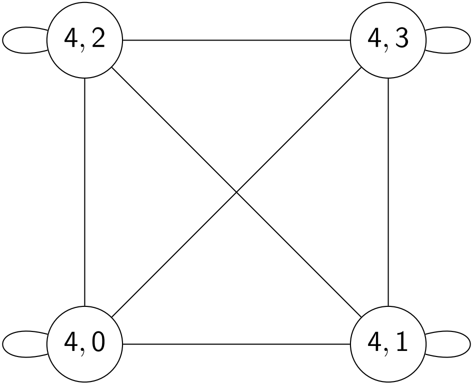
Klausuraufgabe: WS 2016/17 A4



Klausuraufgabe: WS 2016/17 A4



Klausuraufgabe: WS 2016/17 A4



Klausuraufgabe: SS 2009 A1

$$L_1 = \{a^k b^m | k, m \in \mathbb{N}_0 \wedge k \bmod 2 = 0 \wedge m \bmod 3 = 1\}$$
$$L_2 = \{b^k a^m | k, m \in \mathbb{N}_0 \wedge k \bmod 2 = 1 \wedge m \bmod 3 = 0\}$$

Geben Sie für jede der folgenden formalen Sprachen L je einen regulären Ausdruck R_L an mit $\langle R_L \rangle = L$.

- $L = L_1$

Klausuraufgabe: SS 2009 A1

$$L_1 = \{a^k b^m | k, m \in \mathbb{N}_0 \wedge k \bmod 2 = 0 \wedge m \bmod 3 = 1\}$$
$$L_2 = \{b^k a^m | k, m \in \mathbb{N}_0 \wedge k \bmod 2 = 1 \wedge m \bmod 3 = 0\}$$

Geben Sie für jede der folgenden formalen Sprachen L je einen regulären Ausdruck R_L an mit $\langle R_L \rangle = L$.

- $L = L_1$
  $R_L = (aa) * b(bbb)*$

Klausuraufgabe: SS 2009 A1

$$L_1 = \{a^k b^m | k, m \in \mathbb{N}_0 \wedge k \bmod 2 = 0 \wedge m \bmod 3 = 1\}$$
$$L_2 = \{b^k a^m | k, m \in \mathbb{N}_0 \wedge k \bmod 2 = 1 \wedge m \bmod 3 = 0\}$$

Geben Sie für jede der folgenden formalen Sprachen L je einen regulären Ausdruck R_L an mit $\langle R_L \rangle = L$.

- $L = L_1$
  $R_L = (aa) * b(bbb)*$
- $L = L_1 \cdot L_2$

Klausuraufgabe: SS 2009 A1

$$L_1 = \{a^k b^m | k, m \in \mathbb{N}_0 \land k \bmod 2 = 0 \land m \bmod 3 = 1\}$$
$$L_2 = \{b^k a^m | k, m \in \mathbb{N}_0 \land k \bmod 2 = 1 \land m \bmod 3 = 0\}$$

Geben Sie für jede der folgenden formalen Sprachen L je einen regulären Ausdruck R_L an mit $\langle R_L \rangle = L$.

- $L = L_1$
  $R_L = (aa) * b(bbb)*$
- $L = L_1 \cdot L_2$
  $R_L = (aa) * b(bbb) * b(bb) * (aaa)*$

Klausuraufgabe: SS 2009 A1

$$L_1 = \{a^k b^m | k, m \in \mathbb{N}_0 \land k \text{ mod } 2 = 0 \land m \text{ mod } 3 = 1\}$$
$$L_2 = \{b^k a^m | k, m \in \mathbb{N}_0 \land k \text{ mod } 2 = 1 \land m \text{ mod } 3 = 0\}$$

Geben Sie für jede der folgenden formalen Sprachen L je einen regulären Ausdruck R_L an mit $\langle R_L \rangle = L$.

- $L = L_1$
  $R_L = (aa) * b(bbb)*$
- $L = L_1 \cdot L_2$
  $R_L = (aa) * b(bbb) * b(bb) * (aaa)*$
- $L = L_1 \cap L_2$

Klausuraufgabe: SS 2009 A1

$$L_1 = \{a^k b^m | k, m \in \mathbb{N}_0 \wedge k \bmod 2 = 0 \wedge m \bmod 3 = 1\}$$
$$L_2 = \{b^k a^m | k, m \in \mathbb{N}_0 \wedge k \bmod 2 = 1 \wedge m \bmod 3 = 0\}$$

Geben Sie für jede der folgenden formalen Sprachen L je einen regulären Ausdruck R_L an mit $\langle R_L \rangle = L$.

- $L = L_1$
  $R_L = (aa) * b(bbb)*$
- $L = L_1 \cdot L_2$
  $R_L = (aa) * b(bbb) * b(bb) * (aaa)*$
- $L = L_1 \cap L_2$
  $R_L = b(bbbbbb)*$

Turingmaschine entwerfen

Wir wollen eine Turingmaschine, die prüft, ob das Eingabewort $w \in \{a, b\}^*$ genauso viele **a**s wie **b**s

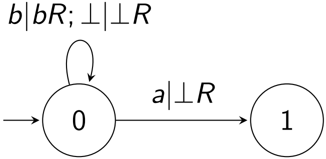
Idee: Wir gehen das Wort durch und löschen pro **a** ein **b**. Bleiben dann keine Zeichen mehr übrig, wird das Wort akzeptiert. Ansonsten wird es abgelehnt.

Turingmaschine entwerfen

- Wir laufen so lange nach rechts, bis wir ein **a** gefunden haben und löschen es. ($\perp$ soll ein gelöschtes Zeichen darstellen)

Turingmaschine entwerfen

- Wir laufen so lange nach rechts, bis wir ein **a** gefunden haben und löschen es. ($\perp$ soll ein gelöschtes Zeichen darstellen)



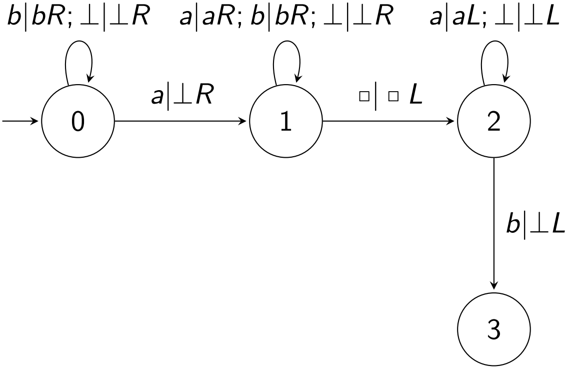
	0				
␣	a	b	b	a	␣
		1			
␣	$\perp$	b	b	a	␣

Turingmaschine entwerfen

- Jetzt laufen wir bis zum Ende des Wortes und suchen von rechts ein **b**, welches wir löschen.

Turingmaschine entwerfen

- Jetzt laufen wir bis zum Ende des Wortes und suchen von rechts ein **b**, welches wir löschen.



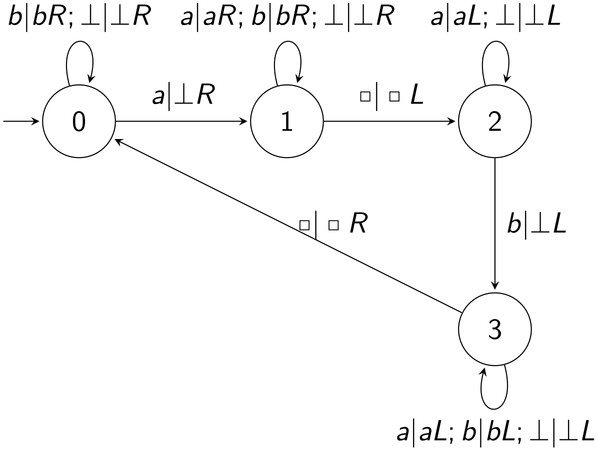
		1			
␣	⊥	b	b	a	␣
					2
␣	⊥	b	b	a	␣
		3			
␣	⊥	b	⊥	a	␣

Turingmaschine entwerfen

- Ist das **b** gelöscht, laufen wir wieder ganz nach links und wiederholen alle Schritte.

Turingmaschine entwerfen

- Ist das **b** gelöscht, laufen wir wieder ganz nach links und wiederholen alle Schritte.



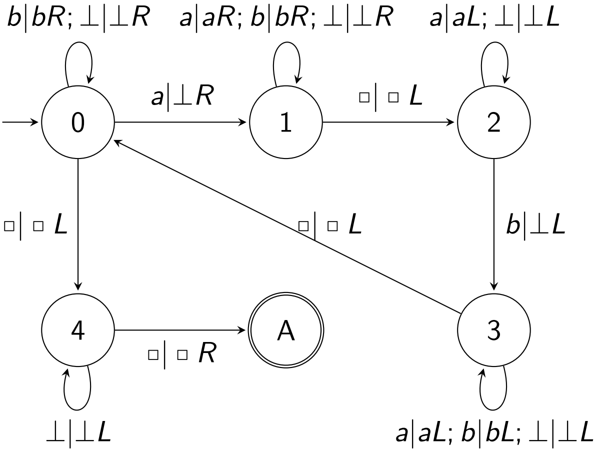
		3			
␣	$\perp$	b	$\perp$	a	␣
	3				
␣	$\perp$	b	$\perp$	a	␣
3					
␣	$\perp$	b	$\perp$	a	␣
	0				
␣	$\perp$	b	$\perp$	a	␣
	0				
␣	$\perp$	$\perp$	$\perp$	$\perp$	␣

Turingmaschine entwerfen

- Findet man nun kein **a** mehr, wird geprüft, ob noch ein **b** auf dem Band steht. Wenn nein, wird akzeptiert.

Turingmaschine entwerfen

- Findet man nun kein **a** mehr, wird geprüft, ob noch ein **b** auf dem Band steht. Wenn nein, wird akzeptiert.



	0				
␣	⊥	⊥	⊥	⊥	␣
					0
␣	⊥	⊥	⊥	⊥	␣
				4	
␣	⊥	⊥	⊥	⊥	␣
4					
␣	⊥	⊥	⊥	⊥	␣
	A				
␣	⊥	⊥	⊥	⊥	␣