TECHNISCHE UNIVERSITÄT DRESDEN

PROGRAMMIERUNG

Übung 10: C_0 und AM_0

Eric Kunze
eric.kunze@mailbox.tu-dresden.de

TU Dresden, 21. Juni 2019

C_0 und AM_0

- **Ziel:** Implementierung einer einfachen Programmiersprache C_1
- **Hier:** zunächste Einschränkung auf C_0

C_0 und AM_0

- **Ziel:** Implementierung einer einfachen Programmiersprache C_1
- **Hier:** zunächste Einschränkung auf C_0
  - genau eine main-Funktion
  - Zugriff auf `stdio` durch `# include`
  - einzig zugelassende Datenstruktur: `int`, Konstanten
  - Kontrollstrukturen: Ein-/Ausgabebefehle, Zuweisungen, Sequenzen, Verzweigungen, bedingte Schleifen

C_0 und AM_0

- **Ziel:** Implementierung einer einfachen Programmiersprache C_1
- **Hier:** zunächste Einschränkung auf C_0
  - genau eine main-Funktion
  - Zugriff auf `stdio` durch `# include`
  - einzig zugelassende Datenstruktur: `int`, Konstanten
  - Kontrollstrukturen: Ein-/Ausgabebefehle, Zuweisungen, Sequenzen, Verzweigungen, bedingte Schleifen
- **Implementierung** durch
  - Syntax von C_0
  - Befehle und Semantik einer abstrakten Maschine AM_0
  - Übersetzer $C_0 \leftrightarrow AM_0$

Übersetzung von `if - then - else`

$$
\begin{aligned}
sttrans(\texttt{if (}exp\texttt{)}\ stat_1\ \texttt{else}\ stat_2, tab, a) := \\
&\quad boolexptrans(exp, tab) \\
&\quad \texttt{JMC}\ a.1; \\
&\quad sttrans(stat_1, tab, a.2) \\
&\quad \texttt{JMP}\ a.3; \\
a.1 : &\quad sttrans(stat_2, tab, a.4) \\
a.3 : &
\end{aligned}
$$

für alle $exp \in \mathrm{W}(\langle \text{BoolExpression} \rangle)$, $stat_1, stat_2 \in \mathrm{W}(\langle \text{Statement} \rangle)$, $tab \in \text{Tab}$ und $a \in \mathbb{N}^*$.

Aufgabe 1

```
1   #include <stdio.h>
2   int main( ) {
3      int a,b,max;
4      scanf("%i", &a);
5      scanf("%i", &a);
6      if (a > b) max = a;
7      else max = b;
8      printf("%d", max);
9      return 0;
10  }
```

Aufgabe 1 – Teil (a)

Baumstrukturierte Adressen:

```
        READ 1;
        READ 2;
        LOAD 1;
        LOAD 2;
        GT;
        JMC 1.3.1;
        LOAD 1;
        STORE 3;
        JMP 1.3.3;
1.3.1:  LOAD 2;
        STORE 3;
1.3.3:  WRITE 3;
```

Aufgabe 1 – Teil (a)

Baumstrukturierte Adressen:

```
        READ 1;
        READ 2;
        LOAD 1;
        LOAD 2;
        GT;
        JMC 1.3.1;
        LOAD 1;
        STORE 3;
        JMP 1.3.3;
1.3.1:  LOAD 2;
        STORE 3;
1.3.3:  WRITE 3;
```

Linearisierte Adressen:

```
 1:  READ 1;
 2:  READ 2;
 3:  LOAD 1;
 4:  LOAD 2;
 5:  GT;
 6:  JMC 10;
 7:  LOAD 1;
 8:  STORE 3;
 9:  JMP 12;
10:  LOAD 2;
11:  STORE 3;
12:  WRITE 3;
```

Aufgabe 1 – Teil (b)

Ablauf der abstrakten Maschine:

	BZ	,	DK	,	HS	,	Inp	,	Out	
(	1	,	ε	,	[ ]	,	5:7	,	ε	)
(	2	,	ε	,	[1/5]	,	7	,	ε	)
(	3	,	ε	,	[1/5, 2/7]	,	ε	,	ε	)
(	4	,	5	,	[1/5, 2/7]	,	ε	,	ε	)
(	5	,	7:5	,	[1/5, 2/7]	,	ε	,	ε	)
(	6	,	0	,	[1/5, 2/7]	,	ε	,	ε	)
(	10	,	ε	,	[1/5, 2/7]	,	ε	,	ε	)
(	11	,	7	,	[1/5, 2/7]	,	ε	,	ε	)
(	12	,	ε	,	[1/5, 2/7, 3/7]	,	ε	,	ε	)
(	13	,	ε	,	[1/5, 2/7, 3/7]	,	ε	,	7	)

Aufgabe 1 – Teil (b)

Ablauf der abstrakten Maschine:

	BZ	,	DK	,	HS	,	Inp	,	Out	
(	1	,	ε	,	[ ]	,	5:7	,	ε	)
(	2	,	ε	,	[1/5]	,	7	,	ε	)
(	3	,	ε	,	[1/5, 2/7]	,	ε	,	ε	)
(	4	,	5	,	[1/5, 2/7]	,	ε	,	ε	)
(	5	,	7:5	,	[1/5, 2/7]	,	ε	,	ε	)
(	6	,	0	,	[1/5, 2/7]	,	ε	,	ε	)
(	10	,	ε	,	[1/5, 2/7]	,	ε	,	ε	)
(	11	,	7	,	[1/5, 2/7]	,	ε	,	ε	)
(	12	,	ε	,	[1/5, 2/7, 3/7]	,	ε	,	ε	)
(	13	,	ε	,	[1/5, 2/7, 3/7]	,	ε	,	7	)

$$\mathcal{P}[\![Max_0]\!](5:7) = proj_5^{(5)}\Big(\mathcal{I}[\![Max_0]\!](1, \varepsilon, [], 5:7, \varepsilon)\Big) = 7$$

Aufgabe 2

```
1   #include <stdio.h>
2
3   int main( ) {
4      int x,y,a;
5      scanf("%i", &y);
6      scanf("%i", &a);
7   x = 0;
8      while (x < a) {
9         x = x + 1;
10        y = y * y;
11     }
12     printf("%d", y);
13     return 0;
14  }
```

Aufgabe 2 – Teil (a)

```
1: READ 2;
2: READ 3;
3: LIT 0;
4: STORE 1;
5: LOAD 1;
6: LOAD 3;
7: LT;
8: JMC 18;
9: LOAD 1;
10: LIT 1;
11: ADD;
12: STORE 1;
13: LOAD 2;
14: LOAD 2;
15: MUL;
16: STORE 2;
17: JMP 5;
18: WRITE 2;
```

Aufgabe 2

```
1: READ 1;
2: READ 2;
3: LOAD 1;
4: LOAD 2;
5: LIT 0;
6: SUB;
7: JMC 9;
8: JMP 5;
9: WRITE 2;
```

Aufgabe 2 – Teil (b)

	BZ	,	DK	,	HS	,	Inp	,	Out	
(	1	,	ε	,	[ ]	,	0:1	,	ε	)
(	2	,	ε	,	[1/0]	,	1	,	ε	)
(	3	,	ε	,	[1/0, 2/1]	,	ε	,	ε	)
(	4	,	0	,	[1/0, 2/1]	,	ε	,	ε	)
(	5	,	1:0	,	[1/0, 2/1]	,	ε	,	ε	)
(	6	,	0:1:0	,	[1/0, 2/1]	,	ε	,	ε	)
(	7	,	1:0	,	[1/0, 2/1]	,	ε	,	ε	)
(	8	,	0	,	[1/0, 2/1]	,	ε	,	ε	)
(	5	,	0	,	[1/0, 2/1]	,	ε	,	ε	)
(	6	,	0:0	,	[1/0, 2/1]	,	ε	,	ε	)
(	7	,	0	,	[1/0, 2/1]	,	ε	,	ε	)
(	9	,	ε	,	[1/0, 2/1]	,	ε	,	ε	)
(	10	,	ε	,	[1/0, 2/1]	,	ε	,	1	)

Aufgabe 2 – Teil (b)

	BZ	,	DK	,	HS	,	Inp	,	Out	
(	1	,	ε	,	[ ]	,	0:1	,	ε	)
(	2	,	ε	,	[1/0]	,	1	,	ε	)
(	3	,	ε	,	[1/0, 2/1]	,	ε	,	ε	)
(	4	,	0	,	[1/0, 2/1]	,	ε	,	ε	)
(	5	,	1:0	,	[1/0, 2/1]	,	ε	,	ε	)
(	6	,	0:1:0	,	[1/0, 2/1]	,	ε	,	ε	)
(	7	,	1:0	,	[1/0, 2/1]	,	ε	,	ε	)
(	8	,	0	,	[1/0, 2/1]	,	ε	,	ε	)
(	5	,	0	,	[1/0, 2/1]	,	ε	,	ε	)
(	6	,	0:0	,	[1/0, 2/1]	,	ε	,	ε	)
(	7	,	0	,	[1/0, 2/1]	,	ε	,	ε	)
(	9	,	ε	,	[1/0, 2/1]	,	ε	,	ε	)
(	10	,	ε	,	[1/0, 2/1]	,	ε	,	1	)