10. Übung Programmierung

Dominic Deckert

3. Juli 2017

Previously on ...

- Hoare-Kalkül
- Schleifeninvariante

2

$SI =$

2

$$SI = (b = a^{n-i}) \wedge (i \geq 0)$$

2

$$SI = (b = a^{n-i}) \wedge (i \geq 0)$$
$$A = SI,\ B = SI \wedge \neg(i > 0)$$
$$C = SI,\ D = SI \wedge (i > 0)$$
$$E = \{b = b * a; i = i - 1;\}$$

1

$SI =$

1

$$SI = (z = i^3) \wedge (i \leq n)$$

1

$SI = (z = i^3) \wedge (i \leq n)$
$A = (z = 0) \wedge (n \geq 0) \wedge (i = 0)$, $B = C = SI$
$C = SI$, $D = SI \wedge \neg(i < n)$
$E = D = SI \wedge \neg(i < n)$, $F = (z = n^3)$
$G = SI \wedge (i < n)$, $H = SI$

H_0

H_0: Fragment von Haskell
Variablen: "xi"
tailrekursiv (nur der letzte Auswertungsschritt ist ein Funktionsaufruf)

H_0

H_0: Fragment von Haskell
Variablen: "x*i*"
tailrekursiv (nur der letzte Auswertungsschritt ist ein Funktionsaufruf)

Bsp:

f x1 x2 = x1 + x2
f x1 x2 = **if** x1 == 0 **then** g x2 **else** f x1 x1
f x1 x2 = g x1 + g x2
f x1 x2 = h(g x1, f x2 x2)

H_0

H_0: Fragment von Haskell
Variablen: "x*i*"
tailrekursiv (nur der letzte Auswertungsschritt ist ein Funktionsaufruf)

Bsp:	f x1 x2 = x1 + x2	ja
	f x1 x2 = **if** x1 == 0 **then** g x2 **else** f x1 x1	ja
	f x1 x2 = g x1 + g x2	nein
	f x1 x2 = h(g x1, f x2 x2)	nein

$H_0 \rightarrow AM_0$

- baumstrukturierte Adressen starten mit Funktionsname
- Variable “xi” in Register i
- Funktionsaufruf: berechne alle Parameter, speichere danach ab
- Ausgabe nur aus Speicherzelle 1

a)

```
(main)        READ 1; READ 2;
              LOAD 2; LIT 3; LOAD 1; STORE 3; STORE 2; STORE 1;
(f)      f.2  LOAD 1; LIT 0; EQ; JMC f.1;
              LOAD 3; STORE 1; WRITE 1; JMP 0;
         f.1  LOAD 1; LIT 1; SUB; LOAD 3;
              LOAD 2; LOAD 3; MULT; LOAD 3; ADD;
              STORE 3; STORE 2; STORE 1; JMP f.2;
```

b)

Ansatz: Nutze zusätzliche Parameter als Zwischenspeicher

b)

Ansatz: Nutze zusätzliche Parameter als Zwischenspeicher

```
f :: Int -> Int -> Int-> Int
f x1 x2 x3 = if x3 <= 2 then x1
  else f (x1*x1 + x2*x2) x1 (x3-1)

main = do x1 <- readLn
  print (f 1 1 x1)
```

3 c)

$C_0 \rightarrow H_0$:
- Funktionsname stellt baumstrukturierte Adresse dar
- Parameter stellen alle lokalen Variablen dar

3 c)

$C_0 \rightarrow H_0$:
- Funktionsname stellt baumstrukturierte Adresse dar
- Parameter stellen alle lokalen Variablen dar

C_0:
if (x1%2 == 0) then x1 = x1 / 2;
 else x1 = x1 -1;
x1 = 2 * x1;

3 a)

```
f x1 x2 x3 x4 = if(x4 == 0) then (0)
  else (if (x3 == x4) then (x1+x2)
  else (if (x1+x2) ((x3+1)*x2) (x3+1) x4))

main = do x4 <- readLn
  print(f 0 1 1 x4)
```

3 b)

```
f:      LOAD 1; LIT 42; LT; JC f.1;
        LOAD 1; STORE 1; WRITE 1; JMP 0;
f.1:    LOAD 1; LIT 42; GT; JC f.1.1;
        LOAD 1; LIT 2; DIV; STORE 1; JMP f;
f.1.1:  LIT 42; STORE 1; WRITE 1; JMP 0;
```