Programmierung 2 (SS 2021)
Universität des Saarlandes
Fakultät MI
Compiler Design Lab

Musterlösung 7
C0

Prof. Dr. Sebastian Hack
Julian Rosemann, B. Sc.
Thorsten Klößner, M. Sc.
Julia Wichlacz, M. Sc.
Maximilian Fickert, M. Sc.

Im Vorlesungskalender finden Sie Informationen über die Kapitel des Skripts, die parallel zur Vorlesung bearbeitet werden sollen bzw. dort besprochen werden. Die Übungsaufgaben dienen der Vertiefung des Wissens, das in der Vorlesung vermittelt wird und als Vorbereitung auf Minitests und Klausur.
Weitere Aufgaben zu den Themen finden Sie jeweils am Ende der Skriptkapitel.
Die Schwierigkeitsgrade sind durch Steine des 2048-Spiels gekennzeichnet, von 512 „leicht“ bis 2048 „schwer“. 4096 steht für Knobelaufgaben.

Aufgabe 7.0: C0 AST

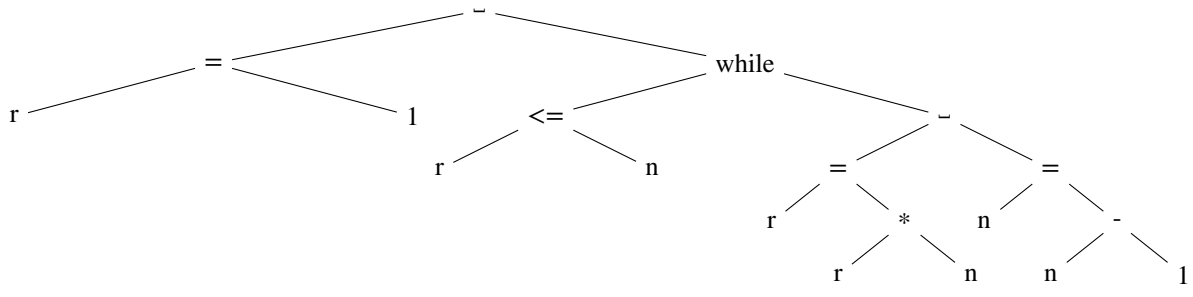
Zeichnen Sie den abstrakten Syntaxbaum zu jedem der folgenden C0-Programme:

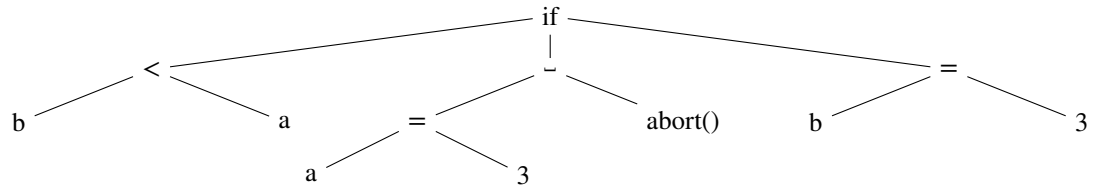
1.
```
r = 1;
while (r <= n) {
    r = r * n;
    n = n - 1;
}
```

2.
```
if (b < a) {
        a = 3;
        abort();
} else
        b = 3;
}
```

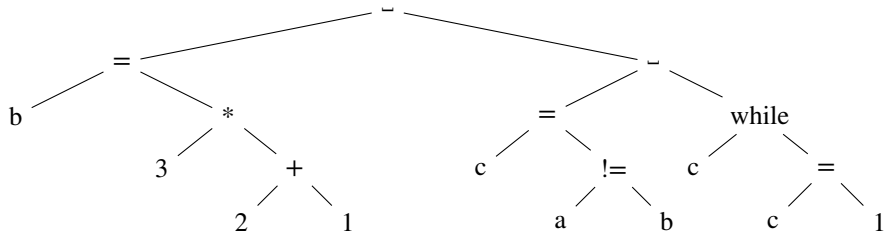
3.
```
b = 3 * (2 + 1);
c = a != b;
while (c)
        c = 1;
```

Lösung

1.


2.


3.



Aufgabe 7.1: Logische Formeln

Definieren Sie die abstrakte Syntax von aussagenlogischen Formeln. Diese bestehen aus den binären Operatoren $\wedge, \vee, \Rightarrow, \Leftrightarrow$, dem unären Operator $\neg$ und Bezeichnern.

Lösung

Kategorie		Form	Kommentar
$Var \ni v$	$::=$	a | b | c | $\cdots$	Bezeichner
$Op \ni o$	$::=$	$\wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow$	Binäre Operatoren
$Expr \ni e$	$::=$	v	Bezeichner
	$\mid$	$e_1\ o\ e_2$	Binärer Ausdruck
	$\mid$	$\neg\ e$	Unärer Ausdruck

Aufgabe 7.2: Zuweisung

Was passiert, wenn man das Programm

```
x = 1;
```

auf dem leeren Zustand ausführt?

Lösung

Da der Behälter zu der Variable `x` nicht existiert, bleibt dieses Programm stecken (`Assign` kann nicht ausgeführt werden).

Aufgabe 7.3: Syntaxfehler

In den folgenden C0-Statements haben sich Fehler eingeschlichen. Finden Sie diese.

1. `if (x < y) r = x else r = y;`
2. `while a < b r = r + 1; a = a + 1;`
3. `if (a = 4) b = 42; else b = x;`
4. `if (x > y) abort; else r = y;`
5. `x == 5; y = 4;; z = 0;`
6. `if (z < 0) abort() else x = x + y;`

Lösung

1. Semikolon der ersten Zuweisung fehlt

```
if (x < y) r = x; else r = y;
```

2. While-Bedingung muss geklammert sein

```
while (a < b) r = r + 1; a = a + 1;
```

3. Vergleich (auf Gleichheit) mit ==

```
if (a == 4) b = 42; else b = x;
```

4. Bei abort fehlen die Klammern

```
if (x > y) abort(); else r = y;
```

5. Zuweisung mit = nicht mit ==

```
x = 5; y = 4;; z = 0;
```

6. Semikolon nach abort() fehlt

```
if (z < 0) abort(); else x = x + y;
```

Aufgabe 7.4: Auswertung von Ausdrücken

Betrachten Sie die Ausdrucksauswertung in C0 auf einem Ausdruck e. Was ist der Unterschied zwischen folgenden Situationen: $[e]\sigma$ ist nicht definiert und $[e]\sigma = ?$.
Was passiert bei der Ausführung der Anweisung `x = e` wenn $[e]\sigma = ?$ gilt?

Lösung

Ist eine Adresse nicht im Urbildbereich der Speicherbelegung, dann existiert für diese Adresse kein Behälter. Der Wert ? ist der undefinierte Wert, der sich in dem Behälter befindet, bevor ein Wert in ihm abgelegt wurde.
Dies führt dazu, dass der uninitialisierte Wert ? gelesen wird, auf dem die Ausdrucksauswertung nicht definiert ist.

Aufgabe 7.5: $\sigma = (\rho, \mu)$

Betrachten Sie folgenden Zustand in Kurzschreibweise, die im Skript eingeführt wurde:

$$\sigma = \{\mathrm{x} \mapsto 1, \mathrm{y} \mapsto 2\}$$

Geben Sie ρ und μ an. Verwenden Sie $\triangle, \pentagon \ldots$, um Adressen darzustellen.

Lösung

$\sigma = (\rho, \mu)$ wobei $\rho = \{\mathrm{x} \mapsto \triangle, \mathrm{y} \mapsto \pentagon\}$, $\mu = \{\triangle \mapsto 1, \pentagon \mapsto 2\}$

Aufgabe 7.6: Moduloberechnung

a) Führen Sie folgendes Programm mit der Semantik von C0 auf dem Zustand

$$\rho := \{a \mapsto \triangle, b \mapsto \pentagon\}, \quad \mu := \{\triangle \mapsto 42, \pentagon \mapsto 17\}$$

aus:

```
while(a >= b) {
    a = a - b;
}
if(a==8) {
    abort();
} else {
    b=0;
}
```

b) Welche Form nimmt die Beendigung des Programmes aus a) an?

Wie sieht das Ganze aus, wenn wir stattdessen $\mu := \{\triangle \mapsto 40, \pentagon \mapsto 17\}$ wählen?

Auf welche Art wird das Programm mit $\mu := \{\triangle \mapsto 42, \pentagon \mapsto 0\}$ beendet?

Hinweis: Für Aufgabenteil b) reicht eine kurze Begründung

Lösung

a) Um die Lesbarkeit zu verbessern, kürzen wir den If-Block durch S ab. Lösungsvorschlag:

	⟨`while(a>=b) a=a-b; if(a==8) abort(); else b=0;` |	ρ,{$\triangle \mapsto 42, \pentagon \mapsto 17$}⟩	
→	⟨`if(a>=b){a=a-b;while(a>=b) a=a-b;}else ; S` |	ρ,{$\triangle \mapsto 42, \pentagon \mapsto 17$}⟩	`[While][Subst]`
→	⟨`a=a-b;while(a>=b) a=a-b; S` |	ρ,{$\triangle \mapsto 42, \pentagon \mapsto 17$}⟩	`[IfTrue][Subst]`
→	⟨`while(a>=b) a=a-b; S` |	ρ,{$\triangle \mapsto 25, \pentagon \mapsto 17$}⟩	`[Assign][Exec]`
→	⟨`if(a>=b){a=a-b;while(a>=b) a=a-b;}else ; S` |	ρ,{$\triangle \mapsto 25, \pentagon \mapsto 17$}⟩	`[While][Subst]`
→	⟨`a=a-b;while(a>=b) a=a-b; S` |	ρ,{$\triangle \mapsto 25, \pentagon \mapsto 17$}⟩	`[IfTrue][Subst]`
→	⟨`while(a>=b) a=a-b; S` |	ρ,{$\triangle \mapsto 8, \pentagon \mapsto 17$}⟩	`[Assign][Exec]`
→	⟨`if(a>=b){a=a-b;while(a>=b) a=a-b;}else ; S` |	ρ,{$\triangle \mapsto 8, \pentagon \mapsto 17$}⟩	`[While][Subst]`
→	⟨`; if(a==8) abort(); else b=0;` |	ρ,{$\triangle \mapsto 8, \pentagon \mapsto 17$}⟩	`[IfFalse][Subst]`
→	⟨`if(a==8) abort(); else b=0;` |	ρ,{$\triangle \mapsto 8, \pentagon \mapsto 17$}⟩	`[Empty][Exec]`
→	⟨`abort();` |	ρ,{$\triangle \mapsto 8, \pentagon \mapsto 17$}⟩	`[IfTrue]`
→	↯		`[Abort]`

b) Das Programm bricht ab.

Unter $\mu := \{\triangle \mapsto 40, \pentagon \mapsto 17\}$ terminiert das Programm in Zustand $\{\triangle \mapsto 6, \pentagon \mapsto 0\}$. Bis auf den Wert von $\triangle$ ist die Semantik der Ausführung identisch zu Aufgabenteil a).

Unter $\mu := \{\triangle \mapsto 42, \pentagon \mapsto 0\}$ divergiert das Programm (es findet keine Beendigung statt), da die Bedingung der while-Schleife stets erfüllt bleibt.

Aufgabe 7.7: Syntax vs Semantik

Es gibt syntaktisch richtige C0-Programme, die beim Ausführen stecken bleiben. Machen Sie sich den Unterschied zwischen Syntax und Semantik am folgenden Programm klar. Ist es syntaktisch bzw. semantisch richtig? Können Sie ein (anderes) Progrogramm finden, das syntaktisch, aber nicht semantisch bzw. semantisch aber nicht syntaktisch richtig ist finden?

```
{
    int x;
    x = 1337;
    x = x / 0;
}
```

Lösung

Die Syntax gibt an, wie gültige C0-Ausdruecke aufgeschrieben werden. Die Semantik beschreibt, wie diese auszuwerten sind. Das obige Programm ist syntaktisch richtig, bleibt aber stecken, da `x / 0` undefiniert ist. Es gibt keine Programme die semantisch richtig, aber syntaktisch falsch sind, da die Semantik nur auf korrekte Programme definiert ist. Ein anderes semantisch ungültige Programm ist:

```
{
  int x;
  x = 1337;
  x = y;
}
```

Aufgabe 7.8: Baumelnde Zeiger

Ein Zeiger heißt „baumelnd“ (engl. dangling pointer), wenn er auf eine Adresse referiert, zu der es keinen Behälter gibt. Konstruieren Sie ein C0pb-Programm, in dessen Ablauf ein baumelnder Zeiger vorkommt. Vergewissern Sie sich dessen, in dem Sie das Programm mit der formalen Semantik ausführen.

Lösung

```
{
  int x;
  int *p;
  {
    int y;
    p = &y;
  }
  x = *p;
}
```

Der Zeiger p zeigt nach der Ausführung des Blocks auf einen Behälter, den es nicht mehr gibt.

$\langle$`{int x; int *p; {int y; p = &y; } x = *p; }`$|\{\};\{\}\rangle$

$\rightarrow \langle$`{int y; p=&y; } x = *p;` $\blacksquare|\{\}, \{x \mapsto \triangle, p \mapsto \pentagon\}; \{\triangle \mapsto ?, \pentagon \mapsto ?\}\rangle$ [Block]

$\rightarrow \langle$`p=&y;` $\blacksquare$`x = *p;` $\blacksquare|\{\}, \{x \mapsto \triangle, p \mapsto \pentagon\}, \{y \mapsto \diamondsuit\}; \{\triangle \mapsto ?, \pentagon \mapsto ?, \diamondsuit \mapsto ?\}\rangle$ [Block][Subst]

$\rightarrow \langle$ $\blacksquare$`x = *p;` $\blacksquare|\{\}, \{x \mapsto \triangle, p \mapsto \pentagon\}, \{y \mapsto \diamondsuit\}; \{\triangle \mapsto ?, \pentagon \mapsto \diamondsuit, \diamondsuit \mapsto ?\}\rangle$ [Assign][Exec]

$\rightarrow \langle$`x = *p;` $\blacksquare|\{\}, \{x \mapsto \triangle, p \mapsto \pentagon\}; \{\triangle \mapsto ?, \pentagon \mapsto \diamondsuit\}\rangle$ [Leave][Exec]

$\rightarrow \langle \blacksquare|\{\}, \{x \mapsto \triangle, p \mapsto \pentagon\}; \{\triangle \mapsto ?, ...\}\rangle$?? [Exec]

$$
\begin{aligned}
[* p]\sigma &= \mu[* p]_L\sigma \\
&= \mu[p]\sigma \\
&= \mu(\mu([p]_L\sigma)) \\
&= \mu(\mu(\rho(p))) \\
&= \mu(\mu(\pentagon)) \\
&= \mu(\diamondsuit)
\end{aligned}
$$

Den Behälter $\diamondsuit$ gibt es nicht mehr, sodass auch die Regel `Assign` hier nicht anwendbar wäre.

Aufgabe 7.9: Do-while Schleife

Erweitern Sie die abstrakte Syntax und die Anweisungs-Semantik von C0 um die do-while Schleife (siehe Abschnitt Schleifen im C-Kapitel des Skripts).

2 12 512 16

Lösung

Erweiterung der Anweisungssprache:
do s **while**(e)
Erweiterung der operationalen Semantik:

[DoWhile] $\langle$**do** s **while**$(e)|\sigma\rangle \rightarrow \langle s$ **while**(e) $s|\sigma\rangle$

Aufgabe 7.10: For Schleife

Erweitern Sie die abstrakte Syntax und die Anweisungs-Semantik um eine eingeschränkte for-Schleife. Diese soll die (abstrakte) Syntax $\langle$**for** $(x = e_1; e_2; x{+}{+})$ $s\rangle$haben.

2 12 512 16

Lösung

Erweiterung der Anweisungssprache:
for $(x = e_1; e_2; x{+}{+})$ s
Erweiterung der operationalen Semantik:

[For] $\langle$**for** $(x = e_1; e_2; x{+}{+})$ $s|\sigma\rangle \rightarrow \langle\{x = e_1;$ **while**(e_2) s _ $x = x + 1;\}|\sigma\rangle$

Aufgabe 7.11: Pointer Arithmetik

Erweitern Sie die Sprache C0p um Zeigerarithmetik. Hierbei kann zu einem Zeiger ein int hinzuaddiert oder von ihm abgezogen werden.

256 25 1024 4

Lösung

Erweiterung der statischen Semantik der C0-Ausdruckssprache:

$$\text{PtrArith}\frac{\Gamma \vdash e : k* \quad \Gamma \vdash i : int \quad o \in \{+,-\}}{\Gamma \vdash e\ o\ i * sizeof(k) : k*}$$

Aufgabe 7.12: Unterprogramme

Fügen Sie der formalen Semantik von C0 Unterprogramme und Unterprogrammaufrufe hinzu. Hierzu gehen Sie wie folgt vor:

1. Definieren Sie eine neue Kategorie $Func$ für Unterprogramme in der abstrakten Syntax.
2. Ein Programm besteht nicht mehr nur aus einer Anweisung, sondern aus Unterprogramm-Definitionen.
3. Definieren Sie eine neue Komponente $\phi : Var \rightharpoonup Func$ im Zustand. Diese bildet Bezeichner auf den „Code“ der Unterprogramme ab.
4. Bei der Definition des Unterprogrammaufrufs können sie dann in ϕ den Code des Unterprogramms „nachschlagen“ und eine entsprechende Konfiguration erzeugen.

Lösung

1. $Func ::= t\ l\ (t_1\ v_1,\ ...,\ t_n\ v_n)\{s\}$
   $FuncCall ::= l(x_1,\ ...,\ x_n);$
   $funcCall \in Stmt$

2. $Prog ::= s \mid s\ p \mid func \mid func\ p, \quad s \in Stmt,\ p \in Prog,\ func \in Func$

3. $\Sigma ::= (Var \to Addr) \times (Addr \to Val \cup \{?\}) \times \Phi$
   $Phi := (Var \to Funct)$

4. $[Define] < t\ l\ (x_1,\ ...,\ x_n)\{s\}|\rho;\mu;\Phi > \to (\rho;\mu;\Phi[l- > t\ l\ (t_1\ v_1,\ ...,\ t_n\ v_n)\{s\}])$
   $[Call] < l\ (x_1,\ ...,\ x_n)|\rho;\mu;\Phi > \to < \{t_1\ v_1;...;t_n\ v_n; v_1 = x_1;...;v_n = x_n; s\}|\rho;\mu;\Phi >$
   Bemerkung: Die anderen Regeln müssen leicht abgeändert werden. Z.B müssen alle jetzt ein Tripel als Zustand akzeptieren und $[Exec]$, $[Subst]$ müssen auf $Prog$ statt $Stmt$ definiert werden.

Aufgabe 7.13: $C0p$ - Basic

Gegeben sei der Zustand

$$\sigma = (\rho, \mu)\ mit\ \rho := \{x \mapsto \triangle, y \mapsto \square, z \mapsto \Diamond\}\ \mu := \{\triangle \mapsto \square, \square \mapsto 3, \Diamond \mapsto \triangle\}$$

a) Erstellen Sie einen Ausdruck der unter σ zu 1 auswertet. Benutzen Sie dabei mindestens einen * und einen & Operator.

b) Berechnen Sie nun die R-Auswertung Ihres Ausdrucks anhand der Regeln von C0p.

Lösung

a)

$$* \; z == \&y$$

b)

$$[\![* \; z == \&y]\!]\sigma = [\![* \; z]\!]\sigma == [\![\&y]\!]\sigma$$

$$\begin{aligned}
[\![* \; z]\!]\sigma &= \mu[\![* \; z]\!]_L\sigma \\
&= \mu[\![z]\!]\sigma \\
&= \mu(\mu[\![z]\!]_L\sigma) \\
&= \mu(\mu(\rho \; z)) \\
&= \mu(\mu\diamondsuit) \\
&= \mu \; \triangle \\
&= \square
\end{aligned}$$

$$\begin{aligned}
[\![\&y]\!]\sigma &= [\![y]\!]_L\sigma \\
&= \rho \; y \\
&= \square
\end{aligned}$$

Der Ausdruck wertet also zu 1 aus.

Aufgabe 7.14: C0p - noch mehr Pointer

1. Betrachten Sie folgenden Ausschnitt aus einer Umgebung $\sigma = (\rho, \mu)$.

$$\rho := \{\mathtt{x} \mapsto \star, \mathtt{y} \mapsto \triangle, \mathtt{t} \mapsto \pentagon, \mathtt{z} \mapsto \female, \dots\}$$
$$\mu := \{\star \mapsto \diamond, \pentagon \mapsto 10, \triangle \mapsto \square, \diamond \mapsto 4, \square \mapsto 2, \female \mapsto ?, \dots\}$$

   Schreiben Sie nun ein Programm in C0p welches in `z` den Wert 42 ablegt, benutzen Sie dafür nur die oben benutzten Variablen, d.h. insbesondere keine Konstanten.

2. Überprüfen Sie ihr Programm, indem Sie die semantische Auswertung ausführen.

3. In den folgenden Ausdrücken haben sich möglicherweise Fehler eingeschlichen. Korrigieren Sie diese falls nötig und geben Sie eine Umgebung an, so dass das Programm eine Semantik bekommt.

   (a)

```
x = 0;
*p = &x+2;
```

   (b)

```
x = 0;
*p = &x;
**pP = &*p;
```

Lösung

1. `z = *x * t + *y;`

2. Auswertung

$$\begin{aligned}&\langle \mathtt{z = *x * t + *y;} \mid \{\mathtt{x} \mapsto \star, \mathtt{y} \mapsto \triangle, \mathtt{t} \mapsto \pentagon, \mathtt{z} \mapsto \female, \dots\};\\ &\qquad \{\star \mapsto \diamond, \pentagon \mapsto 10, \triangle \mapsto \square, \diamond \mapsto 4, \square \mapsto 2, \female \mapsto ?, \dots\}\rangle\\ &\rightarrow \{\mathtt{x} \mapsto \star, \mathtt{y} \mapsto \triangle, \mathtt{t} \mapsto \pentagon, \mathtt{z} \mapsto \female \dots\};\\ &\qquad \{\star \mapsto \diamond, \pentagon \mapsto 10, \triangle \mapsto \square, \diamond \mapsto 4, \square \mapsto 2, \female \mapsto 42, \dots\}\text{[Assign]}\end{aligned}$$

   wobei $[\![z]\!]_L = \female$ und

$$\begin{aligned}[\![\mathtt{*x * t + *y}]\!]\sigma &= [\![\mathtt{*x * t}]\!]\sigma + [\![\mathtt{*y}]\!]\sigma\\ &= ([\![\mathtt{*x}]\!]\sigma * [\![\mathtt{t}]\!]\sigma) + [\![\mathtt{*y}]\!]\sigma\\ &= (\mu[\![\mathtt{*x}]\!]_L\sigma * \mu[\![\mathtt{t}]\!]_L\sigma) + \mu[\![\mathtt{*y}]\!]_L\sigma\\ &= (\mu[\![\mathtt{x}]\!]\sigma * \mu(\rho\ \mathtt{t})) + \mu[\![\mathtt{y}]\!]\sigma\\ &= (\mu(\mu[\![\mathtt{x}]\!]_L\sigma) * \mu(\pentagon)) + \mu(\mu[\![\mathtt{y}]\!]_L\sigma)\\ &= (\mu(\mu(\rho\ \mathtt{x})) * \ 10) + \mu(\mu(\rho\ \mathtt{y}))\\ &= (\mu(\mu\ \star) * \ 10) + \mu(\mu\ \triangle)\\ &= (\mu(\diamond) * \ 10) + \mu(\square)\\ &= (4 \ * \ 10) + \ 2\\ &= 42\end{aligned}$$

3.

(a) Das `+2` kann man an dieser Stelle nicht verwenden, da wir in C0p keine Zeigerarithmetik haben. Eine mögliche Verbesserung:

```
x = 0;
*p = &x;
```

Umgebung:

$$\sigma := \{\mathtt{x} \mapsto \triangle, \mathtt{p} \mapsto \pentagon\}; \{\triangle \mapsto ?, \pentagon \mapsto \diamondsuit, \diamondsuit \mapsto ?\}$$

(b) Das Funktioniert!
Umgebung:

$$\sigma := \{\mathtt{x} \mapsto \triangle, \mathtt{p} \mapsto \pentagon, \mathtt{pP} \mapsto \square\}; \{\triangle \mapsto ?, \pentagon \mapsto \diamondsuit, \diamondsuit \mapsto ?, \square \mapsto \pentagon\}$$

Aufgabe 7.15: Ausführung und Zeiger

Führen Sie folgendes Programm mit der Semantik von C0p auf dem Zustand

$$\sigma = \{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon\}; \{\triangle \mapsto ?, \pentagon \mapsto ?\}$$

aus:

```
x = 3;
y = &x;
*y = *y + 1;
```

Geben Sie die Auswertung von $[\![\mathtt{\&x}]\!]$ und $[\![\mathtt{*y + 1}]\!]$ explizit an.

Lösung

$$\begin{aligned}
&\langle \mathtt{x = 3; \ y = \&x; \ *y = *y + 1;} \mid \{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon\}; \{\triangle \mapsto ?, \pentagon \mapsto ?\}\rangle \\
\to\ &\langle \mathtt{y = \&x; \ *y = *y + 1;} \mid \{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon\}; \{\triangle \mapsto 3, \pentagon \mapsto ?\}\rangle && \text{[Assign], [Exec]} \\
\to\ &\langle \mathtt{*y = *y + 1;} \mid \{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon\}; \{\triangle \mapsto 3, \pentagon \mapsto \triangle\}\rangle && \text{[Assign], [Exec]} \\
\to\ &\{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon\}; \{\triangle \mapsto 4, \pentagon \mapsto \triangle\} && \text{[Assign]}
\end{aligned}$$

$$\begin{aligned}
[\![\mathtt{\&x}]\!]\sigma &= [\![\mathtt{x}]\!]_L\sigma \\
&= \rho(x) \\
&= \triangle
\end{aligned}$$

$$\begin{aligned}
[\![\mathtt{*y + 1}]\!]\sigma &= [\![\mathtt{*y}]\!]\sigma + [\![\mathtt{1}]\!]\sigma \\
&= \mu[\![\mathtt{*y}]\!]_L\sigma + 1 \\
&= \mu[\![\mathtt{y}]\!]\sigma + 1 \\
&= \mu(\mu([\![\mathtt{y}]\!]_L\sigma)) + 1 \\
&= \mu(\mu(\rho(y))) + 1 \\
&= \mu(\mu(\pentagon)) + 1 \\
&= \mu(\triangle) + 1 \\
&= 3 + 1 \\
&= 4
\end{aligned}$$

Aufgabe 7.16: Ausführung und Zeiger

Führen Sie folgendes Programm mit der Semantik von C0p auf dem Zustand

$$\{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon, \mathtt{z} \mapsto \diamond, \mathtt{w} \mapsto \square\}; \{\triangle \mapsto ?, \pentagon \mapsto ?, \diamond \mapsto ?, \square \mapsto ?\}$$

aus:

```
x = 42;
y = &x;
z = &y;
w = **z;
```

Beobachten Sie vor allem, wie die Behälter „verzeigert“ sind.

Lösung

$$\begin{aligned}
&\langle \mathtt{x = 42; \ y = \&x; \ z = \&y; \ w = **z;} \mid \\
&\quad \{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon, \mathtt{z} \mapsto \diamond, \mathtt{w} \mapsto \square\}; \{\triangle \mapsto ?, \pentagon \mapsto ?, \diamond \mapsto ?, \square \mapsto ?\}\rangle \\
\rightarrow\ &\langle \mathtt{y = \&x; \ z = \&y; \ w = **z;} \mid \\
&\quad \{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon, \mathtt{z} \mapsto \diamond, \mathtt{w} \mapsto \square\}; \{\triangle \mapsto 42, \pentagon \mapsto ?, \diamond \mapsto ?, \square \mapsto ?\}\rangle && \text{[Assign], [Exec]} \\
\rightarrow\ &\langle \mathtt{z = \&y; \ w = **z;} \mid \\
&\quad \{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon, \mathtt{z} \mapsto \diamond, \mathtt{w} \mapsto \square\}; \{\triangle \mapsto 42, \pentagon \mapsto \triangle, \diamond \mapsto ?, \square \mapsto ?\}\rangle && \text{[Assign], [Exec]} \\
\rightarrow\ &\langle \mathtt{w = **z;} \mid \\
&\quad \{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon, \mathtt{z} \mapsto \diamond, \mathtt{w} \mapsto \square\}; \{\triangle \mapsto 42, \pentagon \mapsto \triangle, \diamond \mapsto \pentagon, \square \mapsto ?\}\rangle && \text{[Assign], [Exec]} \\
\rightarrow\ &\\
&\quad \{\mathtt{x} \mapsto \triangle, \mathtt{y} \mapsto \pentagon, \mathtt{z} \mapsto \diamond, \mathtt{w} \mapsto \square\}; \{\triangle \mapsto 42, \pentagon \mapsto \triangle, \diamond \mapsto \pentagon, \square \mapsto 42\} && \text{[Assign]}
\end{aligned}$$

Aufgabe 7.17: *C0p* - Reloaded

Führen Sie folgendes Programm mit der Semantik von C0p auf dem Zustand $\sigma := \rho; \mu$ mit
$\rho := \{a \mapsto \star, b \mapsto \pentagon, help \mapsto ♀, i \mapsto ♂, n \mapsto \diamond, c1 \mapsto \triangle, c2 \mapsto \square\}$,
$\mu := \{\triangle \mapsto 1, \square \mapsto 1, \star \mapsto \triangle, \pentagon \mapsto \square, ♀ \mapsto 0, ♂ \mapsto 0, \diamond \mapsto 3\}$
aus:

```
while(i < n - 2) {
  help = *a;
  *a = *b;
  *b = *a + help;
  i = i + 1;
}
*a = 0;
```

Welcher Wert steht nach Ende der Programmausführung in der Variable c2?
Können Sie eine allgemeine Formel für den Inhalt von c2 (für beliebiges $n \in \mathbb{N}\backslash\{0\}$) angeben?

Lösung

Siehe Anhang!

Aufgabe 7.18: ... ein Block

1. Ändern Sie folgendes Programm, sodass die Klammern des inneren Blocks wegfallen, die Anzahl der Variablendeklarationen und Zuweisungen gleich bleiben und trotzdem am Ende des äußeren Blocks der Wert in y wie vorher ist.

```
{
    int y;
    int x;
    x = 42;
    {
        int x;
        x = 7;
    }
    y = x;
}
```

2. Beschreiben Sie nun einen Algorithmus zum Entfernen geschachtelter Blöcke, ohne dass die statische Semantik des Programms sich ändert.

3. Verhält sich folgendes Programm nach Anwendung Ihres Algorithmus noch genau gleich?

```
{
    int x;
    int y;
    int i;
    x = 42;
    i = 0;
    while (i < 2) {
        int x;
        if (i == 0){
            x = 0;
        }
        y = x;
        i = i+1;
    }
}
```

Falls nein: Dürfte ein optimierender C-Compiler diese Umformung trotzdem anwenden?

Lösung

1.
```
{
    int y;
    int x;
    int x1;
    x = 42;
    x1 = 7;
    y = x;
}
```

2. (a) Beginne im innersten Block, der keine anderen Blöcke mehr enthält: Falls in einem Block eine Variable vereinbart wird, die bereits außerhalb des Blockes vereinbart wurde, gebe der Variable im Block einen noch nicht verwendeten Namen und benenne alle verwendenden Vorkommen um.
   (b) Bewege jetzt jede Variablenvereinbarung zu den Variablenvereinbarungen im obersten Block.
   (c) Lösche die Block-Klammern.

3. Dies ist mit unserem Algorithmus nicht der Fall. Das ursprüngliche Programm bleibt im zweiten Schleifendurchlauf stecken, da x hier undefiniert ist. Nach Anwendung des Algorithmus ist $x1$ jedoch bereits an 0 gebunden und es kommt somit nicht mehr zum Steckenbleiben des Programms.
   In C ist dieses Steckenbleiben undefiniertes Verhalten, daher wäre es legitim, in einem optimierendem C-Compiler eine solche Umformung vorzunehmen, da der Compiler davon ausgehen kann, dass undefiniertes Verhalten nicht eintrifft.

```
{
    int x;
    int y;
    int i;
    int x1;
    x = 42;
    i = 0;
    while (i < 2)
        if (i == 0)
            x1 = 0;
        y = x1;
        i = i+1;
}
```

Aufgabe 7.19: Blockausdrücke ableiten

Zum Üben: Leiten Sie die folgenden Programmanweisungen der Sprache *C0b* ab, wenn möglich! Beginnen Sie jeweils mit einem leeren Zustand.

1.
```
{int x; x = 42; {int x; x = 24; {int z; z = x * 7;}} x = x + 11;}
```

2.
```
{int x; int y; x = 5; y = 7;{int z; z = 5; y = 21;} x = x + y;}
```

3.
```
{{int x; x = 13; int y; y = x * 5;} int z; y = z + x;}
```

Lösung

Siehe Anhang!

Aufgabe 7.20: Blockausdruck ableiten

Leiten Sie die folgende Programmanweisung der Sprache *C0b* ab, wenn möglich! Beginnen Sie mit einem leeren Zustand.


```
{int a; int b; b = 15; a = b + 2;{ int b; b = a; a = b - 1;} b = b + 3; b = b + a;}
```

Lösung

$\langle$ {**int** a; **int** b; $b = 15$; $a = b + 2$; { **int** b; $b = a$; $a = b - 1$; } $b = b + 3$; $b = b + a$; } | {}; {} $\rangle$

$\langle$ $b = 15$; $a = b + 2$; { **int** b; $b = a$; $a = b - 1$; } $b = b + 3$; $b = b + a$; ■ | {}, {$a \to \triangle, b \to \square$}; {$\triangle \to ?, \square \to ?$} $\rangle$ [Block]

$\langle$ $a = b + 2$; { **int** b; $b = a$; $a = b - 1$; } $b = b + 3$; $b = b + a$; ■ | {}, {$a \to \triangle, b \to \square$}; {$\triangle \to ?, \square \to 15$} $\rangle$ [Assign][Exec]

$\langle$ { **int** b; $b = a$; $a = b - 1$; } $b = b + 3$; $b = b + a$; ■ | {}, {$a \to \triangle, b \to \square$}; {$\triangle \to 17, \square \to 15$} $\rangle$ [Assign][Exec]

$\langle$ $b = a$; $a = b - 1$; ■ $b = b + 3$; $b = b + a$; ■ | {}, {$a \to \triangle, b \to \square$}, {$b \to \bigcirc$}; {$\triangle \to 17, \square \to 15, \bigcirc \to ?$} $\rangle$ [Block][Subst]

$\langle$ $a = b - 1$; ■ $b = b + 3$; $b = b + a$; ■ | {}, {$a \to \triangle, b \to \square$}, {$b \to \bigcirc$}; {$\triangle \to 17, \square \to 15, \bigcirc \to 17$} $\rangle$ [Assign][Exec]

$\langle$ ■ $b = b + 3$; $b = b + a$; ■ | {}, {$a \to \triangle, b \to \square$}, {$b \to \bigcirc$}; {$\triangle \to 16, \square \to 15, \bigcirc \to 17$} $\rangle$ [Assign][Exec]

$\langle$ $b = b + 3$; $b = b + a$; ■ | {}, {$a \to \triangle, b \to \square$}; {$\triangle \to 16, \square \to 15$} $\rangle$ [Leave][Exec]

$\langle$ $b = b + a$; ■ | {}, {$a \to \triangle, b \to \square$}; {$\triangle \to 16, \square \to 18$} $\rangle$ [Assign][Exec]

$\langle$ ■ | {}, {$a \to \triangle, b \to \square$}; {$\triangle \to 16, \square \to 34$} $\rangle$ [Assign][Exec]

{}; {} [Leave]

Lösung Aufgabe 7.17

Wir kürzen `help = *a; *a = *b; *b = *a + help; i = i+1;` mit S ab.

$\langle$`while(i < n-2) S; *a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 1, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 0, \male \mapsto 0, \diamondsuit \mapsto 3\}\rangle$	
$\rightarrow\langle$`if(i < n-2) {S while(i < n-2) S} else; *a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 1, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 0, \male \mapsto 0, \diamondsuit \mapsto 3\}\rangle$	`[While][Subst]`
$\rightarrow\langle$`help = *a; *a = *b; *b = *a + help; i = i+1; while(i < n-2) S; *a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 1, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 0, \male \mapsto 0, \diamondsuit \mapsto 3\}\rangle$	`[IfTrue][Subst]`
$\rightarrow\langle$` *a = *b; *b = *a + help; i = i+1; while(i < n-2) S; *a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 1, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 1, \male \mapsto 0, \diamondsuit \mapsto 3\}\rangle$	`[Assign][Exec]`
$\rightarrow\langle$` *b = *a + help; i = i+1; while(i < n-2) S; *a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 1, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 1, \male \mapsto 0, \diamondsuit \mapsto 3\}\rangle$	`[Assign][Exec]`
$\rightarrow\langle$`i = i+1; while(i < n-2) S; *a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 2, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 1, \male \mapsto 0, \diamondsuit \mapsto 3\}\rangle$	`[Assign][Exec]`
$\rightarrow\langle$`while(i < n-2) S; *a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 2, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 1, \male \mapsto 1, \diamondsuit \mapsto 3\}\rangle$	`[Assign][Exec]`
$\rightarrow\langle$`if(i < n-2) {S while(i < n-2) S} else; *a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 2, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 1, \male \mapsto 1, \diamondsuit \mapsto 3\}\rangle$	`[While][Subst]`
$\rightarrow\langle$`;*a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 2, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 1, \male \mapsto 1, \diamondsuit \mapsto 3\}\rangle$	`[IfFalse][Subst]`
$\rightarrow\langle$`*a=0;`	$\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\}; \{\triangle \mapsto 1, \square \mapsto 2, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 1, \male \mapsto 1, \diamondsuit \mapsto 3\}\rangle$	`[Empty][Exec]`
$\rightarrow\{a \mapsto \star, b \mapsto \pentagon, help \mapsto \female, i \mapsto \male, n \mapsto \diamondsuit, c1 \mapsto \triangle, c2 \mapsto \square\};$ $\{\triangle \mapsto 0, \square \mapsto 2, \star \mapsto \triangle, \pentagon \mapsto \square, \female \mapsto 1, \male \mapsto 1, \diamondsuit \mapsto 3\}$		`[Assign]`

Hat initial n den Wert k, so hat c2 danach den Wert $fib(k)$, wobei $fib(i)$ die i. Fibonaccizahl ist ($fib(0) = 1, fib(1) = 2, fib(2) = fib(0) + fib(1) = 1 + 2 = 3, fib(3) = 5, \ldots$).

Lösung Aufgabe 7.19

1.

$\langle$ {**int** x; $x = 42$; { **int** x; $x = 24$; { **int** z; $z = x * 7$; } } $x = x + 11$; } |	{}; {} $\rangle$	
$\langle$ $x = 42$; { **int** x; $x = 24$; { **int** z; $z = x * 7$; } } $x = x + 11$; ■ |	{}, {$x \to \triangle$}; {$\triangle \to ?$} $\rangle$	[Block]
$\langle$ { **int** x; $x = 24$; { **int** z; $z = x * 7$; } } $x = x + 11$; ■ |	{}, {$x \to \triangle$}; {$\triangle \to 42$} $\rangle$	[Assign][Exec]
$\langle$ $x = 24$; { **int** z; $z = x * 7$; } ■ $x = x + 11$; ■ |	{}, {$x \to \triangle$}, {$x \to \bigcirc$}; {$\triangle \to 42, \bigcirc \to ?$} $\rangle$	[Block][Subst]
$\langle$ { **int** z; $z = x * 7$; } ■ $x = x + 11$; ■ |	{}, {$x \to \triangle$}, {$x \to \bigcirc$}; {$\triangle \to 42, \bigcirc \to 24$} $\rangle$	[Assign][Exec]
$\langle$ $z = x * 7$; ■ ■ $x = x + 11$; ■ |	{}, {$x \to \triangle$}, {$x \to \bigcirc$}, {$z \to \square$}; {$\triangle \to 42, \bigcirc \to 24, \square \to ?$} $\rangle$	[Block][Subst]
$\langle$ ■ ■ $x = x + 11$; ■ |	{}, {$x \to \triangle$}, {$x \to \bigcirc$}, {$z \to \square$}; {$\triangle \to 42, \bigcirc \to 24, \square \to 168$} $\rangle$	[Assign][Exec]
$\langle$ ■ $x = x + 11$; ■ |	{}, {$x \to \triangle$}, {$x \to \bigcirc$}; {$\triangle \to 42, \bigcirc \to 24$ $\rangle$	[Leave][Exec]
$\langle$ $x = x + 11$; ■ |	{}, {$x \to \triangle$}; {$\triangle \to 42$} $\rangle$	[Leave][Exec]
$\langle$ ■ |	{}, {$x \to \triangle$}; {$\triangle \to 53$} $\rangle$	[Assign][Exec]
	{}; {}	[Leave]

2.

$\langle$ {**int** x; **int** y; $x = 5$; $y = 7$; { **int** z; $z = 5$; $y = 21$; } $x = x + y$; } |	{}; {} $\rangle$	
$\langle$ $x = 5$; $y = 7$; { **int** z; $z = 5$; $y = 21$; } $x = x + y$; ■ |	{}, {$x \to \triangle, y \to \bigcirc$}; {$\triangle \to ?, \bigcirc \to ?$} $\rangle$	[Block]
$\langle$ $y = 7$; { **int** z; $z = 5$; $y = 21$; } $x = x + y$; ■ |	{}, {$x \to \triangle, y \to \bigcirc$}; {$\triangle \to 5, \bigcirc \to ?$} $\rangle$	[Assign][Exec]
$\langle$ { **int** z; $z = 5$; $y = 21$; } $x = x + y$; ■ |	{}, {$x \to \triangle, y \to \bigcirc$}; {$\triangle \to 5, \bigcirc \to 7$} $\rangle$	[Assign][Exec]
$\langle$ $z = 5$; $y = 21$; ■ $x = x + y$; ■ |	{}, {$x \to \triangle, y \to \bigcirc$}, {$z \to \square$}; {$\triangle \to 5, \bigcirc \to 7, \square \to ?$} $\rangle$	[Block][Subst]
$\langle$ $y = 21$; ■ $x = x + y$; ■ |	{}, {$x \to \triangle, y \to \bigcirc$}, {$z \to \square$}; {$\triangle \to 5, \bigcirc \to 7, \square \to 5$} $\rangle$	[Assign][Exec]
$\langle$ ■ $x = x + y$; ■ |	{}, {$x \to \triangle, y \to \bigcirc$}, {$z \to \square$}; {$\triangle \to 5, \bigcirc \to 21, \square \to 5$} $\rangle$	[Assign][Exec]
$\langle$ $x = x + y$; ■ |	{}, {$x \to \triangle, y \to \bigcirc$}; {$\triangle \to 5, \bigcirc \to 21$} $\rangle$	[Leave][Exec]
$\langle$ ■ |	{}, {$x \to \triangle, y \to \bigcirc$}; {$\triangle \to 26, \bigcirc \to 21$} $\rangle$	[Assign][Exec]
	{}; {}	[Leave]

3. Unsere Syntax für C0b erlaubt Deklarationen nur am Anfang eines Blockes, deshalb müsste man alle Deklarationen an den Anfang des jeweiligen Blockes schieben. Selbst wenn man die Deklarationen an den Anfang der Blöcke geschoben hat ist das Programm (semantisch) ungültigt. Der lesende Zugriff auf x und der schreibende Zugriff auf y außerhalb der jeweiligen Sichtbarkeitsbereiche sind ungültig. Entsprechend findet sich auch keine gültige Ableitung. Das Programm bleibt also stecken.