Minitest 3

Minh Khue Pham (2579036)

01.06.2021 13:50:37 - 14:02:08

Multiple Choice (5 Punkte)

Beantworten Sie die folgenden Fragen mit **wahr** oder **falsch**. Für jede richtige Antwort bekommen Sie 1 Punkt. Für falsche Antworten bekommen Sie 0 Punkte. Wenn Sie keine Antwort geben bekommen Sie 0.5 Punkte.
English instruction: Answer the following questions with **true** or **false**. For every correct answer you obtain 1 point. For incorrect answers you get 0 points. If you do not give an answer you receive 0.5 points.

1. Die Funktion der R-Auswertung ist auf allen Ausdrücken und Zuständen defininiert.
1. The R-evaluation function is defined for all expressions and states.

Response:

- ☒ wahr / true
- ☐ falsch / false

2. Die L-Auswertung liefert die Adresse eines Behälters.
2. The L-evaluation of a container evaluates to its address.

Response:

- ☒ wahr / true
- ☐ falsch / false

3. Nur syntaktisch inkorrekte C-Programme können steckenbleiben.
3. Only syntactically incorrect C programs can get stuck.

Response:

- ☐ wahr / true
- ☒ falsch / false

4. Der Ausdruck y = &&x is in C0p syntaktisch korrekt.
4. The expression y = &&x is a syntactically correct C0p expression.

Response:

- ☒ wahr / true
- ☐ falsch / false

5. Das Programm `x = x / y;` bleibt für den Anfangszustand $\sigma = \{y \mapsto \square\}; \{\square \mapsto 42\}$ stecken.
5. The program `x = x / y;` will get stuck for the initial state $\sigma = \{y \mapsto \square\}; \{\square \mapsto 42\}$.

Response:

- ☐ wahr / true
- ☒ falsch / false

Operationelle Semantik (9 Punkte)

Führen Sie folgende Konfiguration gemäß der operationalen Semantik von C0 für höchstens 4 Schritte aus, insofern das Programm nicht vor Schritt 4 terminiert, abbricht oder steckenbleibt. Der erste Schritt ist bereits vorgegeben. Geben Sie die Regeln der operationellen Semantik die verwendet wurden in jedem Schritt an.
English instruction: Execute the following configuration with respect to the operational semantics of C0. Stop after at most 4 steps, unless the program terminates, aborts or gets stuck beforehand. The first step is already given. State the rules of the operational semantics you used for each step.

$\langle$ `x = 1; while (y > 0) x = x * 2; y = y - 1;` $| \{x \mapsto \square, y \mapsto \Diamond\}; \{\square \mapsto ?, \Diamond \mapsto 3\}\rangle$

$\langle$ `while (y > 0) x = x * 2; y = y - 1;` $| \{x \mapsto \square, y \mapsto \Diamond\}; \{\square \mapsto 1, \Diamond \mapsto 3\}\rangle$ [Assign, Exec]

$\langle$ `if (y>0) {x = x*2; y = y-1; while (y > 0) x = x * 2; y = y - 1; } else ; y=y-1;` $| \{x \mapsto \square, y \mapsto \Diamond\}; \{\square \mapsto$ 1 $, \Diamond \mapsto$ 3 $\}\rangle$ [While] Subst (?) (0.5)

$\langle$ `x = x*2; y = y-1; while (y > 0) x = x * 2; y = y - 1; y = y-1;` $| \{x \mapsto \square, y \mapsto \Diamond\}; \{\square \mapsto$ 1 $, \Diamond \mapsto$ 3 $\}\rangle$ [IfTrue] Subst (0.5)

$\langle$ `y = y-1; while (y > 0) x = x * 2; y = y - 1; y = y-1;` $| \{x \mapsto \square, y \mapsto \Diamond\}; \{\square \mapsto$ 2 $, \Diamond \mapsto$ 3 $\}\rangle$ [Assign, Exec]

"y=y-1" gehört entweder zum Schleifenrumpf oder nicht (beide Interpretationen waren hier zulässig). Beides gleichzeitig ist jedoch nicht möglich.

Zustände (6 Punkte)

Schauen Sie sich folgendes Programm an
English instruction: Take a look at the following program

Da hinter dem if(..) ; else; noch eine weitere Instruktion steht, muss die Subst-Regel angewandt werden.

(bei Fragen -> OH)

Bei dem While kommt es darauf an, ob "y=y-1" nun zum Rumpf gehört oder nicht - in letzterem Fall ist auch Subst nötig.

```
while(a == b){
    a = a / b;
}
```

Geben Sie einen Zustand an, sodass die Ausführung des Programms

1. Terminiert
2. Stecken bleibt
3. Divergiert

Hinweis: Schreiben Sie die Zustände als Paar $\sigma = (\rho; \mu)$. Verwenden Sie nicht die Kurzschreibweise aus den Übungsblättern. Verwenden sie Großbuchstaben (A, B, C...) für Adressen.
English instruction: Give a program state for which this program

1. terminates
2. gets stuck
3. diverges

Hint: Write down the states explicitly as a pair $\sigma = (\rho; \mu)$. Do not use the shorthand from the exercise sheets. Please use captial letters (A, B, C...) for addresses.

Answer 1. {a->A, b->B};{A->2, B->2} ✓

Answer 2. {a->A, b->B};{A->0, B->0} ✓

Answer 3. {a->A, b->B};{A->1, B->1} ✓

Kann dieses Programm abbrechen (im Sinne der C0-Semantik)? Begründen Sie mit einem Satz.
English instruction: Can this program abort (with respect to the C0 semantics)? Justify your answer with one sentence.

nein, da es kein abort() in while-loop gibt ✓

Semantik_Referenz

Anweisungssprache

$Stmt \ni s$	::=	$l = e;$	Zuweisung
	|	$s_1 \_ s_2$	Hintereinander-Ausführung
	|	`if` (e) s_1 `else` s_2	Fallunterscheidung
	|	`while` (e) s	Schleife
	|	`abort();`	Abbruch des Programms
	|	`;`	Die leere Anweisung

Ausdruckssprache

$LExpr \ni l$	::=	x	Variable
	|	$*e$	Indirektion
$Expr \ni e$	::=	l	L-Wert
	|	c	Konstante
	|	$e_1 \, o \, e_2$	Binärer Ausdruck
	|	$\&e$	Adresse von
$Op \ni o$	::=	$r \mid m$	Operator
$AOp \ni r$	::=	$+ \mid - \mid \cdots$	Arithmetischer Operator
$COp \ni m$	::=	$== \mid != \mid < \mid \cdots$	Vergleichsoperator

Operationale Semantik

[Empty] $\langle ; \mid \sigma \rangle \to \sigma$

[Assign] $\langle l = e; \mid (\rho, \mu) \rangle \to (\rho, \mu[a \mapsto v])$ falls $[\![e]\!]\sigma = v \in Val$ und $[\![l]\!]_L \sigma = a \in Addr$

[IfTrue] $\langle \texttt{if } (e) \ s_1 \texttt{ else } s_2 \mid \sigma \rangle \to \langle s_1 \mid \sigma \rangle$ falls $[\![e]\!]\,\sigma \neq 0$

[IfFalse] $\langle \texttt{if } (e) \ s_1 \texttt{ else } s_2 \mid \sigma \rangle \to \langle s_2 \mid \sigma \rangle$ falls $[\![e]\!]\,\sigma = 0$

[While] $\langle \texttt{while } (e) \ s \mid \sigma \rangle \to \langle \texttt{if } (e) \ \{s \texttt{ while } (e) \ s\} \texttt{ else ;} \mid \sigma \rangle$

[Abort] $\langle \texttt{abort();} \mid \sigma \rangle \to \lightning$

$$[\text{Exec}]\frac{\langle s_1 \mid \sigma \rangle \to \sigma'}{\langle s_1 \, s_2 \mid \sigma \rangle \to \langle s_2 \mid \sigma' \rangle} \qquad [\text{Crash}]\frac{\langle s_1 \mid \sigma \rangle \to \lightning}{\langle s_1 \, s_2 \mid \sigma \rangle \to \lightning}$$

$$[\text{Subst}]\frac{\langle s_1 \mid \sigma \rangle \to \langle s_1' \mid \sigma' \rangle}{\langle s_1 \, s_2 \mid \sigma \rangle \to \langle s_1' \, s_2 \mid \sigma' \rangle}$$

Ausdrucksauswertung

$$[\![\cdot]\!] \cdot : Expr \to \Sigma \rightharpoonup Val \qquad [\![\cdot]\!]_L \cdot : LExpr \to \Sigma \rightharpoonup Addr$$

$$\begin{aligned} [\![c]\!]\sigma &= c \\ [\![e_1 \, o \, e_2]\!]\sigma &= [\![e_1]\!]\sigma \, o \, [\![e_2]\!]\sigma \\ [\![l \,]\!] \rho; \mu &= \mu\left([\![l \,]\!]_L \rho; \mu\right) \qquad [\![\mathrm{x}]\!]_L \, \rho; \mu = \rho \, \mathrm{x} \end{aligned}$$

$$\begin{aligned} [\![\&l \,]\!]\sigma &:= [\![l \,]\!]_L \sigma \\ [\![*e \,]\!]_L \sigma &:= [\![e]\!]\sigma \quad \text{für } [\![e]\!]\sigma \in Addr \end{aligned}$$