《软件分析与验证》第四次书面作业讲解

朱俸民

清华大学

2020 年 5 月

Contents

Contents

1-1
参考解答

The new command has the syntax havoc x, where x is a variable. The effect of executing havoc x is to assign an arbitrary integer to the variable x, nondeterministically.

Question

Based upon the big-step operational semantics of IMP, we need to add *ONE* inference rule for havoc. Please find it out.

1-1
参考解答

The new command has the syntax havoc x, where x is a variable. The effect of executing havoc x is to assign an arbitrary integer to the variable x, nondeterministically.

Question

Based upon the big-step operational semantics of IMP, we need to add *ONE* inference rule for havoc. Please find it out.

$$(\text{Havoc})\ \frac{}{\langle \sigma, \text{havoc } x \rangle \Downarrow \sigma[x \mapsto n]}$$

1-1

错误解答 1

$$(\text{Havoc})\ \frac{\mathcal{A}[\![x]\!]_\sigma = n(n\ is\ random)}{\langle \sigma, \text{havoc } x\rangle \Downarrow \sigma[x \to n]}$$

$$(\text{Havoc})\ \frac{\mathcal{A}[\![n]\!]_\sigma = n}{\langle \sigma, havoc\ x\rangle \Downarrow \sigma[x \mapsto n]}$$

1-1

错误解答 1

$$(\text{Havoc})\ \frac{\mathcal{A}[\![x]\!]_\sigma = n(n\ is\ random)}{\langle \sigma, \text{havoc } x\rangle \Downarrow \sigma[x \to n]}$$

$$(\text{Havoc})\ \frac{\mathcal{A}[\![n]\!]_\sigma = n}{\langle \sigma, havoc\ x\rangle \Downarrow \sigma[x \mapsto n]}$$

σ 是程序执行前的状态，它应该是任意的。

1-1
错误解答 2

$$(\text{Havoc})\ \frac{\langle \sigma, \text{havoc } x \rangle \Downarrow \sigma' \quad \mathcal{A}[\![x]\!]_{\sigma'} = n}{\langle \sigma, \text{havoc } x \rangle \Downarrow \sigma[x \mapsto n]}$$

错误解答 2

$$(\text{Havoc})\ \frac{\langle \sigma, \text{havoc } x\rangle \Downarrow \sigma' \quad \mathcal{A}[\![x]\!]_{\sigma'} = n}{\langle \sigma, \text{havoc } x\rangle \Downarrow \sigma[x \mapsto n]}$$

σ 循环定义：$\langle \sigma, \text{havoc } x\rangle$ 同时出现在前提和结论上。

1-1
错误解答 3

$$(\text{Havoc})\ \frac{}{\langle \sigma, \text{havoc } x \rangle \Downarrow \varnothing[x \mapsto n]}$$

错误解答 3

$$(\text{Havoc})\ \frac{}{\langle \sigma, \text{havoc } x \rangle \Downarrow \varnothing[x \mapsto n]}$$

执行 havoc 会导致状态清空（$\varnothing$）？

1-1
不规范解答

字体问题：

$$(\text{Havoc})\ \frac{n\ is\ a\ Ingeter}{\langle \sigma, \text{havoc } a\rangle \Downarrow \sigma[a \mapsto n]}$$

$$(\text{Havoc})\ \frac{}{\langle \sigma, \text{havoc x}\rangle \Downarrow \sigma[x \mapsto n]}$$

$$(\text{Havoc})\ \frac{n \in Z}{\langle \sigma, \text{havoc } x\rangle \Downarrow \sigma[x \mapsto n]}$$

$$(\text{havoc})\ \frac{}{\langle \sigma, havocX\rangle \Downarrow \sigma[X \mapsto n]}$$

1-1
不规范解答

Typos:

$$(\text{Havoc})\ \frac{}{\langle \sigma, \textsf{havoc}\ x \rangle \Downarrow \bigvee_{n \in \mathbb{Z}} \sigma[x \mapsto n]}$$

$$(\text{Havoc})\ \frac{}{\langle \sigma, \textsf{havoc} \rangle \Downarrow \sigma[x \to n]}$$

$$(\text{Havoc})\ \frac{}{\langle \varnothing, \textsf{havoc}\ x \rangle \Downarrow \varnothing[x \mapsto somevar]}$$

1-2
参考解答

Question

Show that the following evaluation relation can hold:

$$\langle \varnothing, \text{skip}; \text{havoc } Z \rangle \Downarrow \varnothing[Z \mapsto 42].$$

1-2
参考解答

Question

Show that the following evaluation relation can hold:

$$\langle \varnothing, \mathsf{skip}; \mathsf{havoc}\ Z \rangle \Downarrow \varnothing[Z \mapsto 42].$$

$$\text{(Seq)}\ \frac{\text{(Skip)}\ \dfrac{}{\langle \varnothing, \mathsf{skip} \rangle \Downarrow \varnothing} \qquad \text{(Havoc)}\ \dfrac{}{\langle \sigma, \mathsf{havoc}\ Z \rangle \Downarrow \varnothing[Z \mapsto 42]}}{\langle \varnothing, \mathsf{skip}; \mathsf{havoc}\ Z \rangle \Downarrow \varnothing[Z \mapsto 42]}$$

Contents

2-1
参考解答

Question

Based on the above description, write a *complete* definition (i.e. inference rules) of the evaluation relation "$\langle \sigma, c \rangle \Downarrow \langle \sigma', s \rangle$". Or, you may read this problem as: "translate the above natural language into formal language".

2-1

参考解答

Question

Based on the above description, write a *complete* definition (i.e. inference rules) of the evaluation relation "$\langle \sigma, c \rangle \Downarrow \langle \sigma', s \rangle$". Or, you may read this problem as: "translate the above natural language into formal language".

$$\text{(Skip)}\ \frac{}{\langle \sigma, \mathsf{skip} \rangle \Downarrow \langle \sigma, \triangleright \rangle}$$

$$\text{(Break)}\ \frac{}{\langle \sigma, \mathsf{break} \rangle \Downarrow \langle \sigma, \not\triangleright \rangle}$$

$$\text{(Ass)}\ \frac{\mathcal{A}[\![a]\!]_\sigma = n}{\langle \sigma, x := a \rangle \Downarrow \langle \sigma[x \mapsto n], \triangleright \rangle}$$

$$\text{(SeqCont)}\ \frac{\langle \sigma, c_1 \rangle \Downarrow \langle \sigma', \triangleright \rangle \qquad \langle \sigma', c_2 \rangle \Downarrow \langle \sigma'', s \rangle}{\langle \sigma, c_1; c_2 \rangle \Downarrow \langle \sigma'', s \rangle}$$

$$\text{(SeqBreak)}\ \frac{\langle \sigma, c_1 \rangle \Downarrow \langle \sigma', \not\triangleright \rangle}{\langle \sigma, c_1; c_2 \rangle \Downarrow \langle \sigma', \not\triangleright \rangle}$$

$$\text{(IfTrue)}\ \frac{\mathcal{B}[\![b]\!]_\sigma = \top \qquad \langle \sigma, c_1 \rangle \Downarrow \langle \sigma', s \rangle}{\langle \sigma, \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2\ \mathsf{fi} \rangle \Downarrow \langle \sigma', s \rangle}$$

$$\text{(IfFalse)}\ \frac{\mathcal{B}[\![b]\!]_\sigma = \bot \qquad \langle \sigma, c_2 \rangle \Downarrow \langle \sigma', s \rangle}{\langle \sigma, \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2\ \mathsf{fi} \rangle \Downarrow \langle \sigma', s \rangle}$$

$$\text{(WhileFalse)}\ \frac{\mathcal{B}[\![b]\!]_\sigma = \bot}{\langle \sigma, \mathsf{while}\ b\ \mathsf{do}\ c\ \mathsf{end} \rangle \Downarrow \langle \sigma, \triangleright \rangle}$$

$$\text{(WhileTrueCont)}\ \frac{\mathcal{B}[\![b]\!]_\sigma = \top \qquad \langle \sigma, c \rangle \Downarrow \langle \sigma', \triangleright \rangle \qquad \langle \sigma', \mathsf{while}\ b\ \mathsf{do}\ c\ \mathsf{end} \rangle \Downarrow \langle \sigma'', s \rangle}{\langle \sigma, \mathsf{while}\ b\ \mathsf{do}\ c\ \mathsf{end} \rangle \Downarrow \langle \sigma'', s \rangle}$$

$$\text{(WhileTrueBreak)}\ \frac{\mathcal{B}[\![b]\!]_\sigma = \top \qquad \langle \sigma, c \rangle \Downarrow \langle \sigma', \not\triangleright \rangle}{\langle \sigma, \mathsf{while}\ b\ \mathsf{do}\ c\ \mathsf{end} \rangle \Downarrow \langle \sigma', \triangleright \rangle}$$

2-1
参考解答（另）

$$\text{(WhileTrue-1)}\ \frac{\mathcal{B}[\![b]\!]_\sigma = \top \qquad \langle \sigma, c\rangle \Downarrow \langle \sigma', \not\triangleright\rangle}{\langle \sigma, \textsf{while } b \textsf{ do } c \textsf{ end}\rangle \Downarrow \langle \sigma', \triangleright\rangle}$$

题干已经指出：循环的返回信号总是 continue，因为 break只能跳出最里层的循环。

2-1
不规范解答

$$\text{(Seq)}\ \frac{\langle \sigma, c_1 \rangle \Downarrow \langle \sigma', \triangleright \rangle \qquad \langle \sigma', c_2 \rangle \Downarrow \langle \sigma'', \triangleright \rangle}{\langle \sigma, c_1; c_2 \rangle \Downarrow \langle \sigma'', \triangleright \rangle}$$

$$\text{(Seq)}\ \frac{\langle \sigma, c_1 \rangle \Downarrow \langle \sigma', \triangleright \rangle \qquad \langle \sigma', c_2 \rangle \Downarrow \langle \sigma'', \not\triangleright \rangle}{\langle \sigma, c_1; c_2 \rangle \Downarrow \langle \sigma'', \not\triangleright \rangle}$$

问题：

- 忘记括号
- 不同规则的名称一样
- 有些冗余（但是不算错误）

2-2

错误解答 1

Question

Show that for every command c, states σ and σ', and signal s, if $\langle \sigma, \text{break}; c \rangle \Downarrow \langle \sigma', s \rangle$, then $\sigma = \sigma'$.

2-2
错误解答 1

Question

Show that for every command c, states σ and σ', and signal s, if $\langle \sigma, \text{break}; c\rangle \Downarrow \langle \sigma', s\rangle$, then $\sigma = \sigma'$.

From Break rule above, we get $\langle \sigma, \text{break}\rangle \Downarrow \langle \sigma, \not\triangleright\rangle$ for all state σ.
And then by SeqBreak rule above, we get $\langle \sigma, \text{break}; c\rangle \Downarrow \langle \sigma, \not\triangleright\rangle$ for all command c, state σ.
Finally, since $\langle \sigma, \text{break}; c\rangle \Downarrow \langle \sigma, \not\triangleright\rangle$ and $\langle \sigma, \text{break}; c\rangle \Downarrow \langle \sigma', s\rangle$, by determinacy of natural semantics, we conclude $\sigma = \sigma'$.
Therefore, for every command c, states σ and σ', and signal s, if $\langle \sigma, \text{break}; c\rangle \Downarrow \langle \sigma', s\rangle$, then $\sigma = \sigma'$. □

because big-step has certainty,

$$\langle \sigma, \text{break}; c\rangle \Downarrow \langle \sigma', s\rangle \wedge \langle \sigma, \text{break}; c\rangle \Downarrow \langle \sigma, \not\triangleright\rangle \Rightarrow \sigma' = \sigma$$

大步语义总是具有确定性吗？

反例

在 IMP 的算术表达式中引入特别的语句 foo x。定义语义为：

$$\text{(foo-0)}\ \frac{}{\langle \sigma, \text{foo } x \rangle \Downarrow \sigma[x \mapsto 0]}$$

$$\text{(foo-1)}\ \frac{}{\langle \sigma, \text{foo } x \rangle \Downarrow \sigma[x \mapsto 1]}$$

大步语义不总是具有确定性，需要作为系统的元性质加以证明。

2-2
错误解答 2

证明. 由 Break，有 $\langle \sigma, \mathsf{break} \rangle \Downarrow \langle \sigma, \triangleright \rangle$；又由 Seq1，有 $(\text{Seq1}) \dfrac{\langle \sigma, c_1 \rangle \Downarrow \langle \sigma', \triangleright \rangle}{\langle \sigma, c_1; c_2 \rangle \Downarrow \sigma'' \triangleright}$，故有 $\langle \sigma, \mathsf{break}; c \rangle \Downarrow \langle \sigma', s \rangle$，故得证 ？ □

-0.5

正向推理后，能直接得出结论吗？

逻辑错误

以下演绎推理不成立：

$$\frac{R \implies P \quad R \implies Q}{P \implies Q}$$

不能按照上述格式证明 $P \implies Q$!

2-2
参考解答 1

Proof. The derivation tree must be in the form of the following one:

$$\text{(SeqBreak)}\ \frac{\text{(Break)}\ \dfrac{}{\langle \sigma, \mathsf{break} \rangle \Downarrow \langle \sigma, \not\blacktriangleright \rangle}}{\langle \sigma, \mathsf{break}; c \rangle \Downarrow \langle \sigma', s \rangle}$$

By the rule (Seq), we can know that $\sigma = \sigma'$.

反向推理，说明此推导是唯一的！

2-2
参考解答 2

Proof. Because $\langle \sigma, break \rangle \Downarrow \langle \sigma, \not\triangleright \rangle$, so $\langle \sigma, \text{break}; c \rangle \Downarrow \langle \sigma, \not\triangleright \rangle$
因为加入了break的自然语义依旧保持确定性，所以$\sigma = \sigma'$

下面说明加入了break的自然语义依旧保持确定性：对于2-1写的10条推理规则，没有两条规则对于一条语句，是可以同时适用的：

对于IfTrue和IfFalse，条件$\mathcal{B}[\![b]\!]_\sigma$不同。

对于Seq1和Seq2，条件$\langle \sigma, c_1 \rangle \Downarrow \langle \sigma', \not\triangleright \rangle$和$\langle \sigma, c_1 \rangle \Downarrow \langle \sigma', \triangleright \rangle$不同。

对于While语句：条件$\mathcal{B}[\![b]\!]_\sigma$不同，条件$\langle \sigma, c \rangle \Downarrow \langle \sigma', \not\triangleright \rangle$和$\langle \sigma, c \rangle \Downarrow \langle \sigma', \triangleright \rangle$不同。

可知没有两条规则对于一条语句，是可以同时适用的。所以加入了break后，自然语义仍然保持确定性。 □

说明这里的语义满足确定性！

2-3

Question

Show that for every command c, states σ and σ', and boolean expression b, if $\langle \sigma, \textcolor{blue}{\text{while } b \text{ do } c \text{ end}} \rangle \Downarrow \langle \sigma', \triangleright \rangle$ and $\mathcal{B}[\![\sigma']\!]_b = \top$, then there exists a state σ'' s.t. $\langle \sigma'', c \rangle \Downarrow \langle \sigma', \not\triangleright \rangle$.

2-3

Question

Show that for every command c, states σ and σ', and boolean expression b, if $\langle \sigma, \text{while } b \text{ do } c \text{ end} \rangle \Downarrow \langle \sigma', \triangleright \rangle$ and $\mathcal{B}[\![\sigma']\!]_b = \top$, then there exists a state σ'' s.t. $\langle \sigma'', c \rangle \Downarrow \langle \sigma', \not\triangleright \rangle$.

直观：循环跳出后循环条件还为真，说明只可能是从 break 跳出来的。
证明方法：按照证明步数（证明树高度）归纳。

2-3
参考解答

证明. Suppose we can prove $\langle \sigma, \textsf{while } b \textsf{ do } c \textsf{ end} \rangle \Downarrow \langle \sigma, \triangleright \rangle$ where $\mathcal{B}[\![b]\!]_{\sigma'} = \bot$ with n steps.

1. When $n = 1$, we use inference rule $\dfrac{\mathcal{B}[\![b]\!]_{\sigma} = \top \quad \langle \sigma, c \rangle \Downarrow \langle \sigma', \not\triangleright \rangle}{\langle \sigma, \textsf{while } b \textsf{ do } c \textsf{ end} \rangle \Downarrow \langle \sigma', \triangleright \rangle}$, otherwise we use inference rule $WhileFalse$, indicating that $\mathcal{B}[\![b]\!]_{\sigma'} = \mathcal{B}[\![b]\!]_{\sigma} = \bot$, contradiction. So $\langle \sigma, c \rangle \Downarrow \langle \sigma', \not\triangleright \rangle$ must hold, in which $\sigma'' = \sigma$.
2. Suppose for all states σ' have $\mathcal{B}[\![b]\!]_{\sigma'} = \top$, if $\langle \sigma, \textsf{while } b \textsf{ do } c \textsf{ end} \rangle \Downarrow \langle \sigma', \triangleright \rangle$ can be proved by k steps, there exists σ'' that $\langle \sigma'', c \rangle \Downarrow \langle \sigma', \not\triangleright \rangle$.
   For all states that $\langle \sigma, \textsf{while } b \textsf{ do } c \textsf{ end} \rangle \Downarrow \langle \sigma', \triangleright \rangle$ can be proved by $k+1$ steps, there must be some σ'', $\langle \sigma, c \rangle \Downarrow \langle \sigma'', \triangleright \rangle$ and $\langle \sigma'', \textsf{while } b \textsf{ do } c \textsf{ end} \rangle \Downarrow \langle \sigma', \triangleright \rangle$ hold. By induction htpothesis, there exists σ''', $\langle \sigma''', c \rangle \Downarrow \langle \sigma', \not\triangleright \rangle$ holds.

Therefore, $\forall c, \sigma, \sigma', (\langle \sigma, \textsf{while } b \textsf{ do } c \textsf{ end} \rangle \Downarrow \langle \sigma', \triangleright \rangle \wedge \mathcal{B}[\![b]\!]_{\sigma'} = \top) \mapsto \exists \sigma'', \langle \sigma'', c \rangle \Downarrow \langle \sigma', \not\triangleright \rangle$ □

2-3
不规范解答

证明.

We proceed by induction on n. Suppose we can prove $\langle \sigma, \text{while } b \text{ do } c \text{ end} \rangle \Downarrow \langle \sigma', \triangleright \rangle$ with $\mathcal{B}[\![b]\!]_{\sigma'} = \bot$ by n steps.

先定义，再使用!

Contents

3-1
参考解答

Question

Give a different derivation trace for the above program. Remember to mention the name of the rule you applied, like we have done above.

3-1
参考解答

Question

Give a different derivation trace for the above program. Remember to mention the name of the rule you applied, like we have done above.

Solution

$\langle \sigma, \text{par (if true then } X := 1 \text{ else } X := 0 \text{ fi) with (if true then } Y := 0 \text{ else } Y := 1 \text{ fi) end} \rangle$

$\to \langle \sigma, \text{par (if true then } X := 1 \text{ else } X := 0 \text{ fi) with } (Y := 0) \text{ end} \rangle$ by (Par2)

$\to \langle \sigma[Y \mapsto 0], \text{par (if true then } X := 1 \text{ else } X := 0 \text{ fi) with skip end} \rangle$ by (Par2)

$\to \langle \sigma[Y \mapsto 0], \text{par } (X := 1) \text{with skip end} \rangle$ by (Par1)

$\to \langle \sigma[Y \mapsto 0][X \mapsto 1], \text{par skip with skip end} \rangle$ by (Par1)

$\to \langle \sigma[Y \mapsto 0][X \mapsto 1], \text{skip} \rangle$ by (ParDone)

3-2
参考解答

Question

Let's now consider a more interesting parallel program involving a loop:

$$P: \text{par } (Y := 1) \text{ with } (\text{while } Y = 0 \text{ do } X := X + 1 \text{ end}) \text{ end}$$

Find a property that the above program holds.

3-2
参考解答

Question

Let's now consider a more interesting parallel program involving a loop:

$$P: \text{par } (Y := 1) \text{ with } (\text{while } Y = 0 \text{ do } X := X + 1 \text{ end}) \text{ end}$$

Find a property that the above program holds.

$$\langle \varnothing, P \rangle \to_* \langle \varnothing[Y \mapsto 1], \text{skip} \rangle$$

Solution Use Hoare triple to describe:

$$\{true\} P \{Y = 1\} \blacksquare$$

Solution For every state σ that satisfies $\langle \varnothing, P \rangle \to_* \langle \sigma, \text{skip} \rangle$, we have $\sigma(X) \geq 0$ and $\sigma(Y) = 1$. $\blacksquare$

3-3
参考解答

Question

We said that big-step operational semantics cannot work in this parallel case. Why? Briefly explain the reason in *ONE or TWO* sentences.

3-3
参考解答

Question

We said that big-step operational semantics cannot work in this parallel case. Why? Briefly explain the reason in *ONE or TWO* sentences.

Solution 大步语义描述如何得到语句执行终止的最终状态，因此c_1, c_2中需要将其中一个语句执行到底，才可以执行另一个语句。但是par 语句支持我们执行部分的$c_1(c_2)$语句，然后执行部分$c_2(c_1)$，以此类推，因此不能用大步语义表达。 ■

Solution In big-step operational semantics, we must execute c_1 or c_2 entirely before executing another. However, the $Par1$ and $Par2$ inference rules requires executing some steps of c_1 or c_2 alternatively. ■

Solution 大步语义的规则只能以c_1, c_2为整体进行推导，无法描述c_1, c_2交替执行的情况。 ■

3-3
错误解答

Question

We said that big-step operational semantics cannot work in this parallel case. Why? Briefly explain the reason in *ONE or TWO* sentences.

3-3
错误解答

Question

We said that big-step operational semantics cannot work in this parallel case. Why? Briefly explain the reason in *ONE or TWO* sentences.

Solution For the big-step operational semantics, the certainty is violated, that is:

$$\forall \sigma, \sigma_1, \sigma_2, c.(\langle c, \sigma \rangle \Downarrow \sigma_1 \wedge \langle c, \sigma \rangle \Downarrow \sigma_2) \rightarrow (\sigma_1 = \sigma_2)$$

is not valid. The final state σ could not be unique. ■

Solution The order in which subcommands are executed is uncertain, so the big-step operational semantics may have more than one result. ■

Solution 对于大步语义中的公理：相同环境下执行相同语句得出的结果相同。在多线程环境下不再适用，理由是由于执行的顺序不一样，可能会导出不同的结果。 ■

大步语义不要求有确定性！

3-3
无效解答

Solution Because in this parallel case, the statement $\langle c, \sigma\rangle \Downarrow \sigma'$ is ill-defined. For some c, σ, starting from the environment of σ, c may either terminate or loop forever (as shown in the last question), which violates the definition of $\langle c, \sigma\rangle \Downarrow \sigma'$ that starting from σ, c will definitely terminate. ■

大步语义的记号虽然隐含了“终止”条件，但是大步语义也只关心那些能终止的情况下，程序的语义。虽然 IMP 包括了死循环，但这不妨碍我们用大步语义描述 IMP 的语义。

其他

区分中英文字体:

Read the instructions below carefully before you start working on the assignment:

- ☐ Please typeset your answers in the attached LaTeX source file, compile it to a PDF, and finally hand the PDF to Tsinghua Web Learning before the due date.
- ☐ Make sure you fill in your name and Tsinghua ID, and replace all "TODO"s with your solutions.
- ☐ Any kind of dishonesty is strictly prohibited in the full semester. If you refer to any material that is not provided by us, you must cite it.
- ☐ Unlike previous assignments, in this one, you will do more reading (and also thinking) than writing.

没有扣分不意味着做全对!