程序设计语言的形式语义

2022-2023第一学期期末试卷（A卷）　　闭卷考试　　教师：梁红瑾

学号：_____　　姓名：_____　　成绩：_____（满分100分）

系（专业）：_____　　年级：_____　　班级：_____

题号	1	2	3	4	5	6	7
分数							

Fan is designing a programming language for randomized programs without loops. In this exam, you will help Fan develop the semantics.

$$\begin{array}{rrcl} (\mathit{IntExp}) & e & ::= & n \mid x \mid e + e \mid e - e \\ (\mathit{BoolExp}) & b & ::= & \mathbf{true} \mid \mathbf{false} \mid e = e \mid e < e \mid e \leq e \mid \neg b \mid b \wedge b \mid b \vee b \\ (\mathit{Comm}) & c & ::= & \mathbf{skip} \mid x := e \mid c; c \mid \mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{else}\ c \mid c \oplus_p c \end{array}$$

Figure 1: Syntax of the programming language

Figure 1 shows the syntax of Fan's programming language for randomized programs without loops. Here n ranges over $0, 1, 2, \ldots$. For simplicity, we write $0, 1, 2, \ldots$ for syntactic numerals as well as natural numbers, and write $+, -, <, \ldots$ as the language syntax as well as mathematical operations (i.e., addition, subtraction, comparison, ...) in the real life. It is similar for booleans.

We write $x, y, z, \ldots$ for integer program variables, and PVar is the set of all program variables. A store is a total function from program variable to integers (this is the same as for the simple programming language you learned in the class):

$$(\mathit{Store}) \quad s \in \mathit{PVar} \to \mathit{Int}$$

$[\![e]\!]_{\mathit{intexp}}\, s$ and $[\![b]\!]_{\mathit{boolexp}}\, s$ represent the big-step evaluation of integer expressions and boolean expressions respectively. Their definitions are the same as in the simple programming language you learned in the class. Here we only show the definition for $[\![e]\!]_{\mathit{intexp}}\, s$ below.

$$\begin{array}{rcl} [\![n]\!]_{\mathit{intexp}}\, s & \stackrel{\text{def}}{=} & n \\ [\![x]\!]_{\mathit{intexp}}\, s & \stackrel{\text{def}}{=} & s(x) \\ [\![e_1 + e_2]\!]_{\mathit{intexp}}\, s & \stackrel{\text{def}}{=} & [\![e_1]\!]_{\mathit{intexp}}\, s + [\![e_2]\!]_{\mathit{intexp}}\, s \\ [\![e_1 - e_2]\!]_{\mathit{intexp}}\, s & \stackrel{\text{def}}{=} & [\![e_1]\!]_{\mathit{intexp}}\, s - [\![e_2]\!]_{\mathit{intexp}}\, s \end{array}$$

We write p for a real number such that $0 \leq p \leq 1$. It is used to represent probability. The new command $c_1 \oplus_p c_2$ is called a probabilistic choice. It steps to c_1 with probability p, and it steps to c_2 with probability $1 - p$.

Fan defines the small-step operational semantics for commands in Figure 2. As usual, a configuration is a pair (c, s). A command's one-step execution has the form $(c, s) \xrightarrow{p} (c', s')$, which says that the configuration (c, s) steps to (c', s') with probability p.

$$\frac{}{(\mathbf{skip}, s) \xrightarrow{1} (\mathbf{skip}, s)} \qquad \frac{[\![e]\!]_{intexp}\, s = n}{(x := e, s) \xrightarrow{1} (\mathbf{skip}, s\{x \rightsquigarrow n\})}$$

$$\frac{c_1 \neq \mathbf{skip} \qquad (c_1, s) \xrightarrow{p} (c_1', s')}{(c_1; c_2, s) \xrightarrow{p} (c_1'; c_2, s')} \qquad \frac{}{(\mathbf{skip}; c_2, s) \xrightarrow{1} (c_2, s)}$$

$$\frac{[\![b]\!]_{boolexp}\, s = \mathbf{true}}{(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, s) \xrightarrow{1} (c_1, s)} \qquad \frac{[\![b]\!]_{boolexp}\, s = \mathbf{false}}{(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, s) \xrightarrow{1} (c_2, s)}$$

$$\frac{}{(c_1 \oplus_p c_2, s) \xrightarrow{p} (c_1, s)} \qquad \frac{}{(c_1 \oplus_p c_2, s) \xrightarrow{1-p} (c_2, s)}$$

Figure 2: Small-step operational semantics rules

1. (10 points in total; you get: _____)

   Please formalize each of the following two properties of Fan's programming language. (Please do not use natural language in your formalization.)

   (a) The small-step operational semantics is not deterministic. As usual, by "deterministic", we mean that every configuration has at most one next step. So, by "not deterministic", we mean that some configuration can step to different configurations or with different probabilities.

   (b) The small-step operational semantics is total. That is, every configuration has at least one next step. (This property implies that every configuration has an infinite execution path.)

2. (20 points in total; you get: _____)

   Please write down all the execution paths for the following program c_0 from the initial store $s_0 = \{x \rightsquigarrow 0, y \rightsquigarrow 0\}$.

$$x := 0 \oplus_{0.3} x := 1;$$
$$\textbf{if } x = 0 \textbf{ then } y := 2 \textbf{ else } (x := 0; y := 2)$$

   Each execution path you write down should be ended when it reaches **skip** for the first time.

3. (10 points in total; you get: _____)

Fan defines n-step transitions below. Informally, if there is only one n-step execution path from (c, s) to (c', s'), the probability p in $(c, s) \xrightarrow{p}{}^{n} (c', s')$ is the product of the probability of every step on the path. If there are more than one such execution paths, we have to sum up the probabilities of all the paths to get p. Here $p_1 \cdot p_2$ computes the product (i.e. multiplication) of p_1 and p_2. The sum in the second rule requires us to sum up all the products $p_1 \cdot p_2$ for any p_1, p_2, c' and s' satisfying that $(c, s) \xrightarrow{p_1} (c', s') \wedge (c', s') \xrightarrow{p_2}{}^{n} (c'', s'')$.

$$\frac{}{(c, s) \xrightarrow{1}{}^{0} (c, s)} \qquad \frac{p = \sum_{c', s'} \{p_1 \cdot p_2 \mid (c, s) \xrightarrow{p_1} (c', s') \wedge (c', s') \xrightarrow{p_2}{}^{n} (c'', s'')\}}{(c, s) \xrightarrow{p}{}^{n+1} (c'', s'')}$$

(a) Consider the program c_0 and the initial store s_0 defined in Problem 2. Please write down specific p, c' and s' such that $(c_0, s_0) \xrightarrow{p}{}^{7} (c', s')$ holds. (You only need to write down one group of p, c' and s', if you have multiple answers.)

(b) Does each of the following properties holds? Write down yes or no.

$$\forall c, s.\ \exists p, n, s'.\ (c, s) \xrightarrow{p}{}^{n} (\mathbf{skip}, s')$$

Yes or no: _____

$$\forall c, s, p, n, s', m.\ (c, s) \xrightarrow{p}{}^{n} (\mathbf{skip}, s') \wedge n \leq m \implies (c, s) \xrightarrow{p}{}^{m} (\mathbf{skip}, s')$$

Yes or no: _____

$$\frac{}{(\mathbf{skip}, s) \Downarrow \delta(s)} \qquad \frac{[\![e]\!]_{intexp}\, s = n}{(x := e, s) \Downarrow \delta(s\{x \rightsquigarrow n\})}$$

$$\frac{(c_1, s) \Downarrow \mu_1 \qquad [\![c_2]\!]_{comm}\, \mu_1 = \mu}{(c_1; c_2, s) \Downarrow \mu}$$

$$\frac{[\![b]\!]_{boolexp}\, s = \mathbf{true} \qquad (c_1, s) \Downarrow \mu}{(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, s) \Downarrow \mu} \qquad \frac{[\![b]\!]_{boolexp}\, s = \mathbf{false} \qquad (c_2, s) \Downarrow \mu}{(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, s) \Downarrow \mu}$$

$$\frac{(c_1, s) \Downarrow \mu_1 \qquad (c_2, s) \Downarrow \mu_2}{(c_1 \oplus_p c_2, s) \Downarrow \mu_1 \oplus_p \mu_2}$$

Auxiliary definitions:

$$\delta(s) \stackrel{\text{def}}{=} \lambda s_1.\ \begin{cases} 1, & \text{if } s_1 = s \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_1 \oplus_p \mu_2 \stackrel{\text{def}}{=} \lambda s.\ p \cdot \mu_1(s) + (1-p) \cdot \mu_2(s)$$

$$[\![c]\!]_{comm}\, \mu \stackrel{\text{def}}{=} \lambda s.\ \textstyle\sum_{s_1} \{\mu(s_1) \cdot \mu'(s) \mid (c, s_1) \Downarrow \mu'\}$$

Figure 3: Big-step operational semantics rules

4. (10 points in total; you get: _____)

   Fan is not satisfied with the small-step operational semantics. He thinks that a randomized program transforms an initial state into a final *state distribution*. A state distribution μ is a total function from states to probabilities (i.e. real numbers between 0 and 1) such that the sum of all the probabilities is 1 (formally, $\sum_s \mu(s) = 1$). Fan defines the big-step operational semantics in Figure 3. Please formalize each of the following two properties.

   (a) The big-step operational semantics is deterministic. That is, every configuration has at most one final state distribution.

   (b) The big-step operational semantics is total. That is, every configuration has at least one final state distribution.

5. (20 points in total; you get: _____)

To help understand his big-step operational semantics, Fan proves the following two properties:

(a) $\forall c, s, \mu.\ (c, s) \Downarrow \mu \implies [\![c]\!]_{comm}\,(\delta(s)) = \mu$

(b) $\forall c, s_1, s_2, \mu.\ (c, s_1) \Downarrow \mu \wedge (c, s_2) \Downarrow \mu \implies \forall p.\ [\![c]\!]_{comm}\,(\delta(s_1) \oplus_p \delta(s_2)) = \mu$

For the program c_0 and the initial store s_0 defined in Problem 2, please show the computation of the final state distribution using the big-step operational semantics. (In your computation, you can apply the above two properties.) For your convenience, c_0 and s_0 are copied here: s_0 is $\{x \rightsquigarrow 0, y \rightsquigarrow 0\}$, and c_0 is $x := 0 \oplus_{0.3} x := 1;$ **if** $x = 0$ **then** $y := 2$ **else** $(x := 0; y := 2)$.

6. (10 points in total; you get: _____)

Fan believes that his small-step operational semantics in Figure 2 and his big-step operational semantics in Figure 3 are equivalent. You ask him, "What's your definition of equivalence?" Fan defines the equivalence in two steps. First, he defines $(c, s) \downarrow (s', p)$ to say that in the small-step operational semantics, the execution of (c, s) finally reaches **skip** at the final store s' with probability p. Here is his formal definition:

$$(c, s) \downarrow (s', p) \;\text{ iff }\; \exists n.\, \forall m.\, m \geq n \implies (c, s) \xrightarrow{p}{}^{m} (\mathbf{skip}, s')$$

Then, he formalizes the equivalence property as follows:

$$\forall c, s, \mu.\, ((c, s) \Downarrow \mu) \iff (\forall s'.\, (c, s) \downarrow (s', \mu(s')))$$

You tell Fan, "Your equivalence property does not hold for your semantics!"

(a) Please write down a counterexample (that is, instantiate c, s and μ so that $((c, s) \Downarrow \mu) \iff (\forall s'.\, (c, s) \downarrow (s', \mu(s')))$ does not hold), and explain why Fan's equivalence property does not hold at your example.

(b) Please correct Fan's formalization (that is, define an equivalence property which holds for the semantics).

7. (20 points in total; you get: _____)

Fan wants to develop a program logic to prove $\vdash \{P\}c\{Q\}$. Here the assertions P and Q describe properties over state distributions. He writes $\mu \models P$ to mean that the state distribution μ satisfies P. Informally the judgment semantics $\models \{P\}c\{Q\}$ says, starting from any state distribution satisfying P, if the execution of c terminates, the final state distribution satisfies Q.

(a) Please formalize the definition of $\models \{P\}c\{Q\}$. (Hint: you can use the big-step operational semantics in Figure 3.)

(b) Fan hopes that his program logic is sound. Please formalize the soundness theorem that he needs to prove.