COMP3161/COMP9164
Semantics Exercises

Liam O'Connor

September 29, 2019

1. [⋆⋆] **Jankenbon:**

   Below is a language specified to compute the results of rock paper scissors tournaments:

   $$L \quad ::= \quad \text{✊} \mid \text{✋} \mid \text{✌} \mid (\texttt{Play}\ L\ L)$$

   We assume the existence of an operation **match** which computes the result of an individual game.

   $$\begin{aligned}
   \textbf{match}(\text{✊}, \text{✋}) &= \text{✋}\\
   \textbf{match}(\text{✌}, \text{✋}) &= \text{✌}\\
   \textbf{match}(\text{✋}, \text{✋}) &= \text{✋}\\
   \textbf{match}(\text{✊}, \text{✌}) &= \text{✊}\\
   \textbf{match}(\text{✌}, \text{✌}) &= \text{✌}\\
   \textbf{match}(\text{✋}, \text{✌}) &= \text{✌}\\
   \textbf{match}(\text{✊}, \text{✊}) &= \text{✊}\\
   \textbf{match}(\text{✌}, \text{✊}) &= \text{✊}\\
   \textbf{match}(\text{✋}, \text{✊}) &= \text{✋}
   \end{aligned}$$

   Here are some small step semantics to compute the tournament result:

   $$\frac{e_1 \mapsto e_1'}{(\texttt{Play}\ e_1\ e_2) \mapsto (\texttt{Play}\ e_1'\ e_2)}$$

   $$\frac{v_1 \in \{\text{✌}, \text{✋}, \text{✊}\} \qquad e_2 \mapsto e_2'}{(\texttt{Play}\ v_1\ e_2) \mapsto (\texttt{Play}\ v_1\ e_2')}$$

   $$\frac{v_1 \in \{\text{✌}, \text{✋}, \text{✊}\} \qquad v_2 \in \{\text{✌}, \text{✋}, \text{✊}\}}{(\texttt{Play}\ v_1\ v_2) \mapsto \textbf{match}(v_1, v_2)}$$

   Determine a suitable big-step equivalent semantics for this language.

2. **Logical Formulae:** Imagine we have a simple propositional expression language[1]:

   $$\frac{}{\top\ \textsc{Prop}} \qquad \frac{}{\bot\ \textsc{Prop}} \qquad \frac{x\ \textsc{Prop} \quad y\ \textsc{Prop}}{x \wedge y\ \textsc{Prop}} \qquad \frac{x\ \textsc{Prop}}{\neg x\ \textsc{Prop}}$$

   (a) Here are the big-step evaluation semantics for this language:

   $$\frac{}{\top \Downarrow \texttt{True}}\textsc{True} \qquad \frac{}{\bot \Downarrow \texttt{False}}\textsc{False}$$

   $$\frac{x \Downarrow \texttt{True}}{\neg x \Downarrow \texttt{False}}\textsc{Not}_1 \quad \frac{x \Downarrow \texttt{False}}{\neg x \Downarrow \texttt{True}}\textsc{Not}_2 \quad \frac{x \Downarrow \texttt{False}}{x \wedge y \Downarrow \texttt{False}}\textsc{And}_1 \quad \frac{x \Downarrow \texttt{True} \quad y \Downarrow v}{x \wedge y \Downarrow v}\textsc{And}_2$$

   Determine a small step (SOS) semantics for the language Prop.

   i. [⋆] Identify the set of states Σ, the set of initial states I, and the set of final states F.

   ii. [⋆⋆] Provide inference rules for a relation $\mapsto \subseteq \Sigma \times \Sigma$, which performs one step only of the expression evaluation.

[1] Yes, the grammar is ambiguous, but assume it's just a symbolic representation of abstract syntax.

(b) We shall now prove that the *reflexive, transitive closure* of $\mapsto$, $\overset{\star}{\mapsto}$, is implied by the big-step semantics above. $\overset{\star}{\mapsto}$ is defined by the following rules:

$$\frac{}{x \overset{\star}{\mapsto} x}\textsc{Refl}^* \qquad \frac{x \mapsto y \quad y \overset{\star}{\mapsto} z}{x \overset{\star}{\mapsto} z}\textsc{Trans}^*$$

i. [⋆⋆] First prove the following transitivity lemma:

$$\frac{p \overset{\star}{\mapsto} q \quad q \overset{\star}{\mapsto} r}{p \overset{\star}{\mapsto} r}\textsc{Transitive}$$

ii. [⋆⋆] Now prove the following two lemmas about Not:

$$\frac{x \overset{\star}{\mapsto} \top}{\neg x \overset{\star}{\mapsto} \bot}\textsc{Lemma-Not}_1 \qquad \frac{x \overset{\star}{\mapsto} \bot}{\neg x \overset{\star}{\mapsto} \top}\textsc{Lemma-Not}_2$$

iii. [⋆⋆] Now prove the following lemmas about And:

$$\frac{x \overset{\star}{\mapsto} \bot}{x \wedge y \overset{\star}{\mapsto} \bot}\textsc{Lemma-And}_1 \qquad \frac{x \overset{\star}{\mapsto} \top}{x \wedge y \overset{\star}{\mapsto} y}\textsc{Lemma-And}_2$$

iv. [⋆⋆⋆] Using these lemmas or otherwise, show that $E \Downarrow V$ implies $\sigma_E \overset{\star}{\mapsto} \sigma_V$, where σ_E is the state corresponding to the expression E and σ_V is the final state corresponding to the value V.

(c) Now we will prove the other direction of the equivalence.

i. [⋆⋆⋆⋆] Prove the following lemma by rule induction on the small-step premise:

$$\frac{e \mapsto e' \quad e' \Downarrow v}{e \Downarrow v}$$

Hint: It might help to keep v arbitrary in the induction, and thus prove for each case:

$$\forall v.\ \frac{e' \Downarrow v}{e \Downarrow v}$$

ii. [⋆] Now prove that every completed small step trace has a correspoding big step evaluation:

$$\frac{e \overset{\star}{\mapsto} \sigma_v}{e \Downarrow v}$$

(d) [⋆⋆] Suppose we wanted to add quantifiers and variables to our logic language:

$$\frac{x\ \textsc{Prop}}{\exists v.\ x\ \textsc{Prop}} \qquad \frac{x\ \textsc{Prop}}{\forall v.\ x\ \textsc{Prop}} \qquad \frac{v \text{ is a variable name}}{v\ \textsc{Prop}}$$

Write a static semantics judgement for this language, written $\vdash e\ \mathit{Ok}$ (with whatever context you like before the $\vdash$), that ensures that there are *no free variables* in a given logical formula. Remember that a *free variable* is a variable that is not bound by a quantifier or lambda.

3. **Bizarro-Poland:** Imagine we have a reverse Polish notation calculator language. Reverse Polish notation is an old calculator format that does not require the use of parenthetical expressions. To achieve this, it moves all operators to post-fix, rather than in-fix order. E.g $1 + 2$ becomes $1\ 2\ +$, or $1 - (3 + 2)$ becomes $1\ 3\ 2\ +\ -$. These calculators evaluated these expressions by pushing symbols onto a stack until an operator was encountered, when two symbols would be popped off and the result of the operation pushed on. The grammar is easily defined:

$$\frac{x \in \mathbb{N}}{x \text{ Symbol}} \qquad \frac{x \in \{+, -, /, *\}}{x \text{ Symbol}}$$

$$\frac{}{\epsilon \text{ RPN}} \qquad \frac{x \text{ Symbol} \quad xs \text{ RPN}}{x\ xs \text{ RPN}}$$

(a) The issue is that this grammar allows for invalid programs (such as $1 + 2$ or $-\ +\ *$).

i. [$\star\star$] Write some static semantics inference rules for a judgement $\vdash e\ \mathit{Ok}$ to ensure that programs are well formed.

ii. [$\star$] Show that $\vdash 1\ 3\ 2 + -\ \mathit{Ok}$.

(b) We will now define some big-step evaluation semantics for this calculator. It may be helpful to read the program from right-to-left rather than left-to-right.

i. [$\star$] Identify the set of evaluable expressions E, and the set of result values V. It may be helpful to use a *stack*, defined as follows:

$$\frac{}{\circ\ \text{Stack}} \qquad \frac{x \in \mathbb{N} \quad xs\ \text{Stack}}{x \rhd xs\ \text{Stack}}$$

ii. [$\star\star$] Define a relation $\Downarrow\ \subseteq E \times V$ which evaluates RPN programs.

(c) [$\star\star$] Now we will try small-step semantics.

Our states Σ will be of the form $s \vdash p$ where s is a stack of natural numbers and p is an RPN program.

Initial states are all states of the form $\circ \vdash p$.

Final states are all states of the form $n \rhd \circ \vdash \epsilon$.

Define a relation $\mapsto\ \subseteq \Sigma \times \Sigma$, which evaluates one step of the calculation.

(d) [$\star\star\star\star$] Now we will prove the easier direction of the equivalence proof. Show using rule induction on $\Downarrow$ that, for all expressions e, if $e \Downarrow v$ then $\sigma_e \overset{\star}{\mapsto} \sigma_v$ (where σ_e and σ_v are initial and final states respectively corresponding to e and v).

Hint: You may find it helpful to assume the following lemma (A proof of it is provided in the solutions):

$$\frac{\circ \vdash xs \overset{\star}{\mapsto} v \vdash \epsilon}{\circ \vdash xs\ x \overset{\star}{\mapsto} v \vdash x}\ \textsc{Append}$$

It is worth noting that the lemma TRANSITIVE from Question 1 applies here also.

(e) [$\star\star\star\star$] Show that your static semantics defined in (a) ensure that the program will not reach a stuck state. That is, show that $\vdash e\ \mathit{Ok} \implies \circ \vdash e \overset{\star}{\mapsto} s \vdash \epsilon$, for some s. You may find it helpful to generalise your proof goal before beginning induction.