Homework 1: Reasoning about Languages

15-814: Types and Programming Languages
Fall 2017
Instructor: Karl Crary
TA: Yong Kiam Tan

Out: Sep 11, 2017 (04 pm)
Due: Sep 25, 2017 (11 pm)

Notes:

- Welcome to 15-814's first homework assignment!
- Please email your work as a PDF file to `yongkiat@cs.cmu.edu` titled "15-814 Homework 1". Your PDF should be named "<your-name>-hw1-sol.pdf".
- We will be using Piazza to answer questions as well as make clarifications, corrections and announcements. Please sign up for the class on Piazza to make sure you don't miss anything: `piazza.com/cmu/fall2017/15814`
- Although office hours haven't been posted yet, feel free to contact the instructor or the TA to set up a meeting for questions.

1 Arithmetic

We will start with a simple expression language. There are two types, natural numbers `nat` and booleans `bool`. The language consists of literals, a few arithmetic and comparison operators, and a conditional expression.

$$\begin{array}{rcl} \tau & ::= & \texttt{nat} \mid \texttt{bool} \\ e & ::= & x \mid \overline{n} \mid \overline{\texttt{tt}} \mid \overline{\texttt{ff}} \mid e + e \mid e * e \mid e \leq e \mid \texttt{if}(e, e, e) \end{array}$$

The typing judgment, $\Gamma \vdash e : \tau$, and operational semantics, e val and $e \mapsto e'$, are defined in Appendix A (N.B. $\overline{n}$ denotes the term representation of the natural number n in our simple language). Type contexts Γ are defined with the following grammar:

$$\Gamma \quad ::= \quad \cdot \mid \Gamma, x : \tau$$

Before we add anything new, let us check some properties to make sure you are comfortable with rule induction. For the proofs in this assignment, you may condense similar cases in arguments by induction, but be clear and rigorous in your reasoning.

Task 1 *Prove the following* inversion lemma*:*

(If Inversion) If $\Gamma \vdash \texttt{if}(e_1, e_2, e_3) : \tau$*, then* $\Gamma \vdash e_1 : \texttt{bool}$*,* $\Gamma \vdash e_2 : \tau$*, and* $\Gamma \vdash e_3 : \tau$*.*

This seems immediate, but really follows from the induction principle for the typing judgment. (Tip: Prove that for all Γ*,* e*,* τ *such that* $\Gamma \vdash e : \tau$*, if* $e = \texttt{if}(e_1, e_2, e_3)$ *for some* e_1*,* e_2*,* e_3*, then* $\Gamma \vdash e_1 : \texttt{bool}$*,* $\Gamma \vdash e_2 : \tau$*, and* $\Gamma \vdash e_3 : \tau$*.)*

In general, an inversion lemma is one which recovers the premises of a rule from its conclusion. In the rest of the assignment, you may use these without proof, but be sure to note explicitly when you apply them and check for yourself that they hold.

Task 2 *Prove unicity of typing for this language.*

(Unicity of Typing) *For any* Γ, e, τ, τ' *such that* $\Gamma \vdash e : \tau$ *and* $\Gamma \vdash e : \tau'$, *we have* $\tau = \tau'$. *You may assume that any variable appears at most once in a given context.*

2 Days of the Week

We will now extend the expression language with a new type `day` representing days of the week. For the purposes of this assignment, we shall assume that Mondays to Thursdays do not exist, i.e.:

$$d \quad ::= \quad \texttt{Fri} \mid \texttt{Sat} \mid \texttt{Sun}$$

The language is extended with the following constructs:

$$\begin{array}{rcl} \tau & ::= & \cdots \mid \texttt{day} \\ e & ::= & \cdots \mid \overline{d} \mid \overline{\texttt{nextn}}(e, e) \end{array}$$

Here, $\overline{d}$ is the term representation of the day d, and $\overline{\texttt{nextn}}(e_1, e_2)$ represents the e_1^{th} day after e_2.

Before we define the static and dynamic semantics of these constructs, we will need some simple helper definitions:

Task 3 *Define the* $\texttt{next}(d)$ `is` d' *judgement which takes a day* d *and returns the next day,* d'. *You should assume that the next day from Sunday is Friday.*

Task 4 *Define the* $\texttt{nextn}(n, d)$ `is` d' *judgement which takes a natural number* n, *a day* d, *and returns the* n^{th} *day after* d. *You should make use of the inductive definition of* `nat`.

Task 5 *Using your answer to Task 4, extend the static and dynamic semantics of* e *with the cases for* $\overline{d}$ *and* $\overline{\texttt{nextn}}(e_1, e_2)$. *Your definition should satisfy progress and type preservation, which you will need to prove below.*

3 Type Safety

We will now show type safety for the language, **including your extension in Task 5**, by proving progress and type preservation.

Task 6 *Carefully state a canonical forms lemma for your extended semantics. You do not have to prove the lemma, and you may assume it for the rest of your proof.*

Task 7 *Prove progress for your extended semantics, i.e.*

(Progress) *If* $\vdash e : \tau$, *then either* e val *or there exists* e' *such that* $e \mapsto e'$.

Task 8 *Prove preservation for your extended semantics, i.e.*

(Preservation) *If* $\vdash e : \tau$ *and* $e \mapsto e'$, *then* $\vdash e' : \tau$.

A Base Expression Language

A.1 Statics

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ (\textsc{Hyp}) \qquad \frac{}{\Gamma \vdash \overline{n} : \texttt{nat}}\ (\textsc{Num})$$

$$\frac{}{\Gamma \vdash \overline{\mathtt{tt}} : \mathtt{bool}}\ (\textsc{True}) \qquad \frac{}{\Gamma \vdash \overline{\mathtt{ff}} : \mathtt{bool}}\ (\textsc{False})$$

$$\frac{\Gamma \vdash e_1 : \mathtt{nat} \quad \Gamma \vdash e_2 : \mathtt{nat}}{\Gamma \vdash e_1 + e_2 : \mathtt{nat}}\ (\textsc{Plus}) \qquad \frac{\Gamma \vdash e_1 : \mathtt{nat} \quad \Gamma \vdash e_2 : \mathtt{nat}}{\Gamma \vdash e_1 * e_2 : \mathtt{nat}}\ (\textsc{Times})$$

$$\frac{\Gamma \vdash e_1 : \mathtt{nat} \quad \Gamma \vdash e_2 : \mathtt{nat}}{\Gamma \vdash e_1 \leq e_2 : \mathtt{bool}}\ (\textsc{Leq}) \qquad \frac{\Gamma \vdash e_1 : \mathtt{bool} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \mathtt{if}(e_1, e_2, e_3) : \tau}\ (\textsc{If})$$

A.2 Dynamics

$$\frac{}{\overline{n}\ \mathsf{val}}\ (\textsc{Num-V}) \qquad \frac{}{\overline{\mathtt{tt}}\ \mathsf{val}}\ (\textsc{True-V}) \qquad \frac{}{\overline{\mathtt{ff}}\ \mathsf{val}}\ (\textsc{False-V})$$

$$\frac{e_1 \mapsto e_1'}{e_1 + e_2 \mapsto e_1' + e_2}\ (\textsc{Plus-S1}) \qquad \frac{e_1\ \mathsf{val} \quad e_2 \mapsto e_2'}{e_1 + e_2 \mapsto e_1 + e_2'}\ (\textsc{Plus-S2})$$

$$\frac{}{\overline{m} + \overline{n} \mapsto \overline{m+n}}\ (\textsc{Plus-I})$$

$$\frac{e_1 \mapsto e_1'}{e_1 * e_2 \mapsto e_1' * e_2}\ (\textsc{Times-S1}) \qquad \frac{e_1\ \mathsf{val} \quad e_2 \mapsto e_2'}{e_1 * e_2 \mapsto e_1 * e_2'}\ (\textsc{Times-S2})$$

$$\frac{}{\overline{m} * \overline{n} \mapsto \overline{m*n}}\ (\textsc{Times-I})$$

$$\frac{e_1 \mapsto e_1'}{e_1 \leq e_2 \mapsto e_1' \leq e_2}\ (\textsc{Leq-S1}) \qquad \frac{e_1\ \mathsf{val} \quad e_2 \mapsto e_2'}{e_1 \leq e_2 \mapsto e_1 \leq e_2'}\ (\textsc{Leq-S2})$$

$$\frac{m \leq n}{\overline{m} \leq \overline{n} \mapsto \overline{\mathtt{tt}}}\ (\textsc{Leq-I1}) \qquad \frac{m > n}{\overline{m} \leq \overline{n} \mapsto \overline{\mathtt{ff}}}\ (\textsc{Leq-I2})$$

$$\frac{e_1 \mapsto e_1'}{\mathtt{if}(e_1, e_2, e_3) \mapsto \mathtt{if}(e_1', e_2, e_3)}\ (\textsc{If-S})$$

$$\frac{}{\mathtt{if}(\overline{\mathtt{tt}}, e_2, e_3) \mapsto e_2}\ (\textsc{If-I1}) \qquad \frac{}{\mathtt{if}(\overline{\mathtt{ff}}, e_2, e_3) \mapsto e_3}\ (\textsc{If-I2})$$