Assignment 1
Language Definitions and Inference Rules

CS 7400: Intensive Principles of Programming Languages
Chris Martens

Due Friday, October 4, 11:59pm
80 pts

This assignment is due on the above date and must be submitted electronically on Gradescope. Please use the LaTeX template on the website to typeset your assignment and make sure to include your full name and NU ID. For the written problems, you may (alternatively) submit handwritten answers that have been scanned and are **easily legible** (if you're not sure, check with a classmate).

You should submit *one* file, `hw01.pdf`, with your written solutions to the questions.

1 Transition Relations

Consider defining a state transition relation on some set of states $s \in \mathcal{S}$. We write $s \mapsto s'$ to mean "s steps to s'" is a valid transition, and we could define this relation many different ways depending on the system of interest (such as dynamic semantics of a programming languages).

Regardless of the specific definition of $\mapsto$, it is often useful to define the *reflexive-transitive closure* of $\mapsto$, i.e., the relation $\mapsto^*$ that adds reflexivity and transitivity to $\mapsto$:

$$\frac{}{s \mapsto^* s}\ \mathsf{step}^*/\mathsf{refl} \qquad \frac{s_1 \mapsto s_2 \quad s_2 \mapsto^* s_3}{s_1 \mapsto^* s_3}\ \mathsf{step}^*/\mathsf{trans}$$

Note that the left premise of the $step^*/trans$ rule is $\mapsto$ without the star symbol.

Alternatively, we could define the relation $\Rightarrow$ as follows:

$$\frac{}{s \Rightarrow s}\ \mathsf{steps}/\mathsf{refl} \qquad \frac{s \mapsto s'}{s \Rightarrow s'}\ \mathsf{steps}/\mathsf{step} \qquad \frac{s_1 \Rightarrow s_2 \quad s_2 \Rightarrow s_3}{s_1 \Rightarrow s_3}\ \mathsf{steps}/\mathsf{trans}$$

Task 1 (20 pts) *Prove by rule induction that these two definitions are equivalent, in the sense that $s \mapsto^* s'$ if and only if $s \Rightarrow s'$.*

Note 1: Proving an "if and only if" claim means writing two proofs, one for each direction.
Note 2: You might need to state and prove a lemma separately from the main proof.

2 Yet Another Small Language

For this problem, we will define yet another small language (YASL) similar to Hutton's Razor + bool + let. It will include internally-defined natural numbers and a case-analysis construct over them.

The syntax is:

Expressions	$e ::= \mathsf{z} \mid \mathsf{s}(e) \mid \mathsf{ncase}(e, e_1, x.e_2) \mid \mathsf{true} \mid \mathsf{false} \mid \mathsf{ite}(e_1, e_2, e_3) \mid x$
Types	$\tau ::= \mathsf{Nat} \mid \mathsf{Bool}$

Type system:

$$\frac{}{\Gamma \vdash \mathsf{z} : \mathsf{Nat}}\ \mathsf{ty/z} \qquad \frac{\Gamma \vdash e : \mathsf{Nat}}{\Gamma \vdash \mathsf{s}(e) : \mathsf{Nat}}\ \mathsf{ty/s} \qquad \frac{\Gamma \vdash e : \mathsf{Nat} \quad \Gamma \vdash e_1 : \tau \quad \Gamma, x : \mathsf{Nat} \vdash e_2 : \tau}{\Gamma \vdash \mathsf{ncase}(e, e_1, x.\ e_2) : \tau}\ \mathsf{ty/ncase}$$

$$\frac{}{\Gamma \vdash \mathsf{true} : \mathsf{Bool}}\ \mathsf{ty/true} \qquad \frac{}{\Gamma \vdash \mathsf{false} : \mathsf{Bool}}\ \mathsf{ty/false} \qquad \frac{\Gamma \vdash e_1 : \mathsf{Bool} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \mathsf{ite}(e_1, e_2, e_3) : \tau}\ \mathsf{ty/ite}$$

$$\frac{}{\Gamma, x{:}\tau \vdash x : \tau}\ \mathsf{ty/var}$$

Evaluation semantics:

$$\frac{}{\mathsf{z\ value}}\ \mathsf{value/z} \qquad \frac{e\ \mathsf{value}}{\mathsf{s}(e)\ \mathsf{value}}\ \mathsf{value/s} \qquad \frac{}{\mathsf{true\ value}}\ \mathsf{value/true} \qquad \frac{}{\mathsf{false\ value}}\ \mathsf{value/false}$$

$$\frac{v\ \mathsf{value}}{v \Downarrow v}\ \mathsf{eval/value} \qquad \frac{e \Downarrow \mathsf{true} \quad e_1 \Downarrow v}{\mathsf{ite}(e, e_1, e_2) \Downarrow v}\ \mathsf{eval/if/t} \qquad \frac{e \Downarrow \mathsf{false} \quad e_2 \Downarrow v}{\mathsf{ite}(e, e_1, e_2) \Downarrow v}\ \mathsf{eval/if/f}$$

$$\frac{e \Downarrow v}{\mathsf{s}(e) \Downarrow \mathsf{s}(v)}\ \mathsf{eval/s} \qquad \frac{e_1 \Downarrow \mathsf{z} \quad e_2 \Downarrow v_2}{\mathsf{ncase}(e_1, e_2, x.\ e_3) \Downarrow v_2}\ \mathsf{eval/ncase/z} \qquad \frac{e_1 \Downarrow \mathsf{s}(v_1) \quad [v_1/x]e_3 \Downarrow v_3}{\mathsf{ncase}(e_1, e_2, x.\ e_3) \Downarrow v_3}\ \mathsf{eval/ncase/s}$$

Task 2 (10 pts) *Write two distinct (up to α-equivalence) well-typed expressions that use the* ncase() *construct. Give their typing and evaluation derivations.*

Task 3 (10 pts) *Write a variation on the rule* ty/ncase *called* ty/ncase/bad *that causes type soundness* not *to hold, and explain why not. (That is, give an example of a well-typed expression using that rule that evaluates to something whose type differs from the original expression.)*

Since we define values here not as a special syntactic class, but by inference rules, we would like to know that everything in the second place of the $e \Downarrow e'$ judgment is in fact a value.

Theorem 1 (Evaluation gives values) *If $e \Downarrow v$, then v* value.

Task 4 (20 pts) *Prove Theorem 1. Remember to state what you do induction on and explicitly justify each line of your proof.*

We can also give a type soundness theorem that says, if e is well-typed *and* it evaluates, then its value has the same type.

Theorem 2 (Type Soundness) *If $\cdot \vdash e : \tau$ and $e \Downarrow v$, then $v : \tau$.*

To prove this, we need the following *substitution lemma*:

Lemma 3 (Substitution Typing) *If $\Gamma, x{:}\tau_1 \vdash e_2 : \tau_2$, and $\Gamma \vdash e_1 : \tau_1$, then $\Gamma \vdash [e_1/x]e_2 : \tau_2$.*

You may use this lemma without proof in the next task.

Task 5 (20 points) *Prove Theorem 2.*