Assignment 1

Joshua Dunfield

due: Friday, 25 January 2019

Name: **Estimated time spent:**

1 Grammars

Suppose we have the following grammar.

$$
\begin{array}{llcl}
\text{integers} & n & & \\
\text{variables} & x, y, z & & \\
\text{statements} & s & ::= & \mathsf{new}\ x\ \mathsf{init}\ n\ \mathsf{in}\ s \\
 & & & \mid \mathsf{print}\ x \\
 & & & \mid \mathsf{set}\ x\ \mathsf{to}\ n \\
 & & & \mid s\ ;\ s
\end{array}
$$

Part (a). According to the above grammar, which of the following strings can be produced from the nonterminal (meta-variable) s? (List their numbers, or put a checkmark next to those that can be produced.)

1. $\mathsf{print}\ 3$
2. $\mathsf{print}\ x$
3. $\mathsf{print}\ z$
4. 4
5. $\mathsf{new}\ z\ \mathsf{init}\ 3\ \mathsf{in}\ \mathsf{print}\ z$
6. $\mathsf{new}\ z\ \mathsf{init}\ 3\ \mathsf{in}\ \mathsf{print}\ z$
7. $\mathsf{new}\ z\ \mathsf{init}\ 3\ \mathsf{in}\ \mathsf{print}\ y$
8. $\mathsf{new}\ y, z\ \mathsf{init}\ 3, 4\ \mathsf{in}\ \mathsf{print}\ y$
9. $\mathsf{set}\ y\ \mathsf{to}\ 5\ ;\ \mathsf{print}\ z\ ;\ \mathsf{print}\ z$

Part (b). Below, I have drawn one of two possible parse trees for the statement

$$\mathsf{print}\ x\ ; \mathsf{print}\ y\ ;\ \mathsf{print}\ z$$

Draw the *other* possible parse tree.

```
         ;
        / \
       /   \
   print    ;
     |     / \
     x    /   \
       print  print
         |      |
         y      z
```

Part (c). Lisp-style syntax, like `(+ 1 (+ 2 3))`, eliminates the possibility of multiple parse trees for the same string. Rewrite the above grammar of s to use Lisp-style syntax: every production should look like `(word ...)`, where `word` is unique to that production (e.g. don't write two productions that both start with `set`) and ... contains only meta-variables (n, x, s).

I have given you the first production; for the last production, my intent was that s ; s represents a sequence of statements, so the word `seq` would be a reasonable choice.

integers	n		
variables	x, y, z		
statements	s	::=	(new x n s)
		|	
		|	
		|	

2 Grammars as inductive definitions

Following the example at the beginning of lec2 §4 for arithmetic expressions, write an inductive definition that corresponds to the original grammar for statements (*not* your grammar with Lisp-like syntax). I have given you the first part.

(a) If x is a variable and n is an integer and s is a statement, then new x init n in s is a statement.

(b)

(c)

(d)

Now, for the Lisp-style grammar, write the part of the inductive definition that corresponds to the first production, (new x n s).

(a)

3 "In logic, there are no morals."

Since recognizing when things are wrong is just as important as recognizing when things are right, let's consider some "joke" semantics. We define a judgment, $e \Downarrow_0 v$, that resembles our big-step semantics but "likes zero".

$\boxed{e \Downarrow_0 v}$ expression e evaluates to value v, but likes zero

$$\frac{}{n \Downarrow_0 0}\ \text{evalzero-const} \qquad\qquad \frac{e_1 \Downarrow_0 n_1 \qquad e_2 \Downarrow_0 n_2}{(+\ e_1\ e_2) \Downarrow_0 n_1 + n_2}\ \text{evalzero-add}$$

Prove the following conjecture. You may choose whether to induct on the expression or the derivation, and whether to do case analysis on the expression or the derivation; for this proof, either approach should work.

Write out every step in detail. Use additional pages if necessary.

Conjecture 3.1 (Always zero)**.**
For all expressions e and derivations $\mathcal{D}$ such that $\mathcal{D}$ derives $e \Downarrow_0 v$, it is the case that $v = 0$.

Proof. By structural induction on

Induction hypothesis:

Consider cases of

- Case

□

Now consider yet another judgment, which also likes zero: to evaluate an integer expression, it must be zero.

$\boxed{e\ 0{\Downarrow_0}\ v}$ expression e evaluates to value v, but only works for zero

$$\frac{}{0\ 0{\Downarrow_0}\ 0}\ \text{zeroevalzero-const} \qquad \frac{e_1\ 0{\Downarrow_0}\ n_1 \qquad e_2\ 0{\Downarrow_0}\ n_2}{(+\ e_1\ e_2)\ 0{\Downarrow_0}\ n_1 + n_2}\ \text{zeroevalzero-add}$$

We now have three different definitions of big-step evaluation: the "real" definition $e \Downarrow v$, and two "joke" definitions, $e \Downarrow_0 v$ and $e\ 0{\Downarrow_0}\ v$. Let's compare these definitions in the framework of soundness and completeness. Since we have (I hope) some faith in the validity of $e \Downarrow v$, we will consider that to be our "ground truth", and compare the joke definitions with respect to $e \Downarrow v$.

$\boxed{e \Downarrow v}$ expression e evaluates to value v

$$\frac{}{n \Downarrow n}\ \text{eval-const} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{(+\ e_1\ e_2) \Downarrow n_1 + n_2}\ \text{eval-add}$$

(a) The theory $e \Downarrow_0 v$ is *not* sound with respect to $e \Downarrow v$. Give a counterexample:

$e =$ $\qquad\qquad$ $v =$

(b) The theory $e \Downarrow_0 v$ is *not* complete with respect to $e \Downarrow v$. Again, give a counterexample.

$e =$ $\qquad\qquad$ $v =$

(c) Is $e\ 0{\Downarrow_0}\ v$ sound with respect to $e \Downarrow v$? To approach this question, you may want to spend a little time looking for a counterexample—if you find one, you're done; if not, doing so should help you understand how the $0{\Downarrow_0}$ rules work. If you don't find a counterexample, state and prove that $e\ 0{\Downarrow_0}$ *is* sound with respect to $e \Downarrow v$! There's room on the next page.

(d) The theory $e\ 0{\Downarrow_0}\ v$ is *not* complete with respect to $e \Downarrow v$. Give a counterexample.

$e =$ $\qquad\qquad$ $v =$

Your answers should give you a picture of the soundness/completeness connections (if any) that our two joke systems have to $e \Downarrow v$. Now let's consider their relationships with *each other*. (Yes, this is a very long question.)

(e) Is $e\ 0{\Downarrow_0}\ v$ sound *with respect to* $e \Downarrow_0 v$? (**Note:** *not* with respect to $e \Downarrow v$!) Start by following the approach of part (c), but if you can't find a counterexample, explain in 1 or 2 sentences whether you think it is sound (or unsound) and *why* you think it is sound (or unsound).

(f) Is $e\ 0{\Downarrow_0}\ v$ complete *with respect to* $e \Downarrow_0 v$? Give a counterexample.

(g) After completing part (f), argue *in one sentence* why $e \Downarrow_0 v$ is *not* sound with respect to $e\ 0{\Downarrow_0}\ v$.

Finally, consider the following rule, which is exactly the same as zeroevalzero-const, except that it derives the "good" big-step judgment rather than the joke judgment:

$$\frac{}{0 \Downarrow 0}\ \text{proposed-eval-const-zero}$$

(h) Is proposed-eval-const-zero an *admissible* rule? That is, if proposed-eval-const-zero can derive a judgment, could we also derive that judgment using our two existing rules for $\Downarrow$? Briefly explain.

Conjecture 3.2 (Proof of soundness for part (c)—if you don't find a counterexample)**.**

For all

such that

it is the case that

Proof. By structural induction on

Consider cases of

□