Assignment 1 Sample Solution

Joshua Dunfield

January 29, 2019

1 Grammars

Suppose we have the following grammar.

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

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. $\mathsf{print}\ 3$ ✗[3 is not a variable]
2. $\mathsf{print}\ x$ ✓
3. $\mathsf{print}\ z$ ✓
4. 4 ✗[an n by itself is not a production]
5. $\mathsf{new}\ z\ \mathsf{init}\ 3\ \mathsf{in}\ \mathsf{print}\ z$ ✓
6. $\mathsf{new}\ z\ \mathsf{init}\ 3\ \mathsf{in}\ \mathsf{print}\ z$ ✓
(I erroneously repeated this, but no one complained)
7. $\mathsf{new}\ z\ \mathsf{init}\ 3\ \mathsf{in}\ \mathsf{print}\ y$ ✓
[y doesn't look like it's in scope, but that isn't enforced by grammars]
8. $\mathsf{new}\ y, z\ \mathsf{init}\ 3, 4\ \mathsf{in}\ \mathsf{print}\ y$ ✗
[grammar doesn't allow lists like y, z]
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    ;                             ;    print
    |     / \                           / \      |
    x    /   \                         /   \     z
      print  print                  print  print
        |      |                      |      |
        y      z                      x      y
```

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.

$$
\begin{array}{llcl}
\text{integers} & n & & \\
\text{variables} & x, y, z & & \\
\text{statements} & s & ::= & (\texttt{new}\ x\ n\ s) \\
& & \mid & (\texttt{print}\ x) \\
& & \mid & (\texttt{set}\ x\ n) \\
& & \mid & (\texttt{seq}\ s\ s)
\end{array}
$$

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) If x is a variable then `print` x is a statement.

(c) If x is a variable and n is an integer, then `set` x `to` n is a statement.

(d) If s_1 is a statement and s_2 is a statement, then s_1 ; s_2 is a statement.

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

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

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 e. [Also possible to use structural induction on $\mathcal{D}$.]

[**Note:** This proof looks *much* longer than what was required, because I have added a lot of explanation. Only the highlighted parts were required.]

Induction hypothesis: [To obtain the IH, *always* follow this process:

1. Copy the statement of the conjecture. [Statement of conjecture:
   For all expressions e and derivations $\mathcal{D}$ such that $\mathcal{D}$ derives $e \Downarrow_0 v$, it is the case that $v = 0$.]

2. Rename *all* the meta-variables. Since the current meta-variables (e, $\mathcal{D}$, v) don't have prime marks, we can systematically rename by adding prime marks.[1]

3. Add a restriction that the renamed meta-variable of the thing we're inducting on is smaller than the thing we're inducting on. In this case, we're inducting on e and the renamed e is called e', so the restriction is $e' \prec e$.

Every step is required. Step 1 ensures the IH has the same structure as the conjecture. Step 2 ensures that the IH is sufficiently general, in case we need that generality, and makes Step 3 possible. Step 3 ensures that we don't fall into circular reasoning.

Step 2 produces:

For all expressions e' and derivations $\mathcal{D}'$ such that $\mathcal{D}'$ derives $e' \Downarrow_0 v'$, it is the case that $v' = 0$.

Step 3 produces the IH.]

For all expressions e' and derivations $\mathcal{D}'$ such that $e' \prec e$ and $\mathcal{D}'$ derives $e' \Downarrow_0 v'$, it is the case that $v' = 0$.

Consider cases of e.

- Case $e = n$:

[1] We can sometimes complete an inductive proof without renaming every meta-variable in the IH. However, we *must* rename the meta-variable of the thing we're induction on—e in this example; if we don't rename it, we can't add the restriction in step 2.

$e = n$	Given [assumption within this case]
$e \Downarrow_0 v$	Given
$n \Downarrow_0 v$	By above equation [$e = n$]
$v = 0$	By inversion on rule evalzero-const

[Our goal was $v = 0$, so we're done with this case.]

- Case $e = (+\ e_1\ e_2)$:

$e = (+\ e_1\ e_2)$	Given [assumption within this case]
$e \Downarrow_0 v$	Given
$(+\ e_1\ e_2) \Downarrow_0 v$	By above equation [$e = (+\ e_1\ e_2)$]
$v = n_1 + n_2$	By inversion on rule evalzero-add
$e_1 \Downarrow_0 n_1$	"
$e_2 \Downarrow_0 n_2$	"
$e_1 \prec e$	By above equation [$e = (+\ e_1\ e_2)$]
$e_1 \Downarrow_0 n_1$	Above
$n_1 = 0$	By IH [with e_1 as e' and n_1 as v']
$e_2 \prec e$	By above equation [$e = (+\ e_1\ e_2)$]
$e_2 \Downarrow_0 n_2$	Above
$n_2 = 0$	By IH [with e_2 as e' and n_2 as v']
$v = n_1 + n_2$	Above
$v = 0 + 0$	By above equations [$n_1 = 0$ and $n_2 = 0$]
$v = 0$	By arithmetic

[Our goal was $v = 0$, so we're done with this case.]

[Some of the above steps can be omitted, such as the lines justified by "Above": these cases are not that long, so the reader can quickly find the fact marked "Above".]

[Uses of the IH can *never* be omitted.]

[Writing something like "Hence proved" is okay, but not required.] □

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:
[This question was unclear: I didn’t specify whether I wanted the v that we get from $\Downarrow_0$ or the v that we get from $\Downarrow$. I intended the v from $\Downarrow_0$, but I accepted both.]

$$e = 1 \qquad v = 0$$

[We can derive $1 \Downarrow_0 0$, but not $1 \Downarrow 0$.]

(b) The theory $e \Downarrow_0 v$ is *not* complete with respect to $e \Downarrow v$. Again, give a counterexample. [This question was unclear: I didn’t specify whether I wanted the v that we get from $\Downarrow_0$ or the v that we get from $\Downarrow$. I intended the v from $\Downarrow$, but I accepted both.]

$$e = 1 \qquad v = 1$$

[We can derive $1 \Downarrow 1$, but not $1 \Downarrow_0 1$.]

(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.

$e \; 0{\Downarrow}_0 \; v$ is sound w.r.t. $\Downarrow$. See the sample proof below.

(d) The theory $e \; 0{\Downarrow}_0 \; v$ is *not* complete with respect to $e \Downarrow v$. Give a counterexample.

$$e = 1 \qquad v = 1$$

[We can derive $1 \Downarrow 1$, but not $1 \; 0{\Downarrow}_0 \; 1$.]

[These are the shortest answers to (a), (b), and (d). The expression $e = (+ \; 1 \; 1)$ works too.]

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).

I think it's sound, because $0{\Downarrow_0}$ only works (gives a value) when all the numbers in e are 0, either because $e = 0$ or $e = \texttt{(+ (+ 0 0) 0)}$, etc. In those situations, $\Downarrow_0$ also gives 0, so $0{\Downarrow_0}$ is sound w.r.t. $\Downarrow_0$.

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

$e = 1 \qquad v = 0$

[We can derive $1 \Downarrow_0 0$, but not $1 \; 0{\Downarrow_0} \; 0$.]

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

Completeness is soundness in reverse: $0{\Downarrow_0}$ isn't complete w.r.t. $\Downarrow_0$, so $\Downarrow_0$ isn't sound w.r.t. $0{\Downarrow_0}$.

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.

Rule proposed-eval-const-zero can derive only one judgment: $0 \Downarrow 0$. We can derive $0 \Downarrow 0$ using the existing rule eval-const. So proposed-eval-const-zero is admissible.

Conjecture 3.2 (Proof of soundness for part (c)—if you don't find a counterexample)**.**
For all e *and* v
such that $e \; 0{\Downarrow}_0 \; v$ *[system whose soundness is being examined]*
it is the case that $e \Downarrow v$. *["ground truth" system]*

Proof. By structural induction on e.

[I didn't leave room to write the IH here, but:
IH: For all e' and v' such that $e' \prec e$ and $e' \; 0{\Downarrow}_0 \; v'$, it is the case that $e' \Downarrow v'$.]

Consider cases of e.

- Case $e = n$:

$e = n$	Given [assumption within this case]
$e \; 0{\Downarrow}_0 \; v$	Given
$n \; 0{\Downarrow}_0 \; v$	By above equation [$e = n$]
$n = 0$	By inversion on rule zeroevalzero-const
$v = 0$	"
$0 \Downarrow 0$	By rule eval-const
$e \Downarrow v$	By above equations [$e = n$ and $n = 0$ and $v = 0$]

[Our goal was $e \Downarrow v$, so we're done with this case.]

- Case $e = (+ \; e_1 \; e_2)$:

$e = (+ \; e_1 \; e_2)$	Given [assumption within this case]
$e \; 0{\Downarrow}_0 \; v$	Given
$(+ \; e_1 \; e_2) \; 0{\Downarrow}_0 \; v$	By above equation [$e = (+ \; e_1 \; e_2)$]
$v = n_1 + n_2$	By inversion on rule zeroevalzero-add
$e_1 \; 0{\Downarrow}_0 \; n_1$	"
$e_2 \; 0{\Downarrow}_0 \; n_2$	"
$e_1 \prec e$	By above equation [$e = (+ \; e_1 \; e_2)$]
$e_1 \; 0{\Downarrow}_0 \; n_1$	Above
$e_1 \Downarrow n_1$	By IH [with e_1 as e' and n_1 as v']
$e_2 \prec e$	By above equation [$e = (+ \; e_1 \; e_2)$]
$e_2 \; 0{\Downarrow}_0 \; n_2$	Above
$e_2 \Downarrow n_2$	By IH [with e_2 as e' and n_2 as v']
$(+ \; e_1 \; e_2) \Downarrow n_1 + n_2$	By rule eval-add
$e \Downarrow v$	By above equations [$e = (+ \; e_1 \; e_2)$ and $v = n_1 + n_2$

[Our goal was $e \Downarrow v$, so we're done with this case.] □