Assignment 1

Jana Dunfield

due: Thursday, 2022–07–09, 11:59 pm

You may **not** work in a group on this assignment.

ID number(s): 20311734

Name(s): Zhimin Zhao

1 Grammars

Suppose we have the following grammar.

$$\begin{array}{lrcl} \text{integers} & n & \in \mathbb{Z} & \\ \text{variables} & x, y, z & & \\ \text{systems} & s & ::= & \mathsf{start}\ z\ \mathsf{in}\ s \\ & & & \mid \mathsf{stop} \\ & & & \mid \mathsf{get}\ y\ z \\ & & & \mid \mathsf{set}\ x\ \mathsf{to}\ y \\ & & & \mid s\ ;\ s \end{array}$$

Part (a). According to the above grammar, which of the following strings can be produced from the nonterminal (meta-variable) s?

1. $\mathsf{stop}\ ;\ \mathsf{stop}\ ;\ \mathsf{stop}$
2. $\mathsf{set}\ x, y\ \mathsf{to}\ z\ ;\ \mathsf{get}\ z\ x\ ;\ \mathsf{stop}$
3. $\mathsf{get}\ 5\ z$
4. $\mathsf{start}\ y\ \mathsf{in}\ \mathsf{get}\ z\ \mathsf{stop}$
5. $\mathsf{start}\ z\ \mathsf{in}\ \mathsf{start}\ z\ \mathsf{in}\ \mathsf{get}\ z\ z$
6. $\mathsf{set}\ x\ \mathsf{to}\ \mathsf{stop}\ ;\ \mathsf{get}\ x\ y$

List their numbers, or put a checkmark next to those that can be produced:

1,5

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

$$\mathsf{start}\ z\ \mathsf{in}\ \mathsf{set}\ y\ \mathsf{to}\ z\ ;\ \mathsf{get}\ y\ x$$

Draw the *other* possible parse tree.

```
start_in_
     /   \
    z     ;
         / \
   set_to_  get
    / \      / \
   y   z    y   x
```

```
           ;
       /       \
 start_in_     get
  /   \        /  \
 z  set_to_  y     x
      /   \
     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 `get`) and "···" contains only meta-variables (n, x, y, z, s).

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

$$
\begin{array}{lrcl}
\text{integers} & n & \in \mathbb{Z} & \\
\text{variables} & x, y, z & & \\
\text{systems} & s & ::= & (\mathtt{start}\ x\ s) \\
 & & & \mid \mathtt{stop} \\
 & & & \mid (\mathtt{get}\ y\ z) \\
 & & & \mid (\mathtt{set}\ x\ y) \\
 & & & \mid (\mathtt{seq}\ s\ s)
\end{array}
$$

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 of systems (*not* your grammar with Lisp-like syntax). I have given you the first part.

Hint: The grammar allows the string `get` x y ; `stop`. Make sure that your definition also allows that string.

(a) If x is a variable and s is a system, then `start` x `in` s is a system.

(b) If s is a system and s' is a system, then s ; s' is a system.

(c) If x is a variable and y is a variable, then `get` x y is a system.

(d) If x is a variable and y is a variable, then `set` x `to` y is a system.

(e) `stop` is a system.

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

(a) If x is a variable and s is a system, then `(start` x s`)` is a system.

3 "In logic, there are no morals."

Recognizing when things are wrong is just as important as recognizing when things are right.

We define a judgment, $e \Downarrow_{\text{odd}} n$, that resembles our big-step semantics but "likes odd numbers".

$\boxed{e \Downarrow_{\text{odd}} n}$ expression e evaluates to n, but likes odd numbers

$$\frac{}{n \Downarrow_{\text{odd}} 7}\ \text{evalodd-const} \qquad \frac{e_1 \Downarrow_{\text{odd}} n_1 \qquad e_2 \Downarrow_{\text{odd}} n_2}{(\texttt{+}\ e_1\ e_2) \Downarrow_{\text{odd}} 2n_1 + 2n_2 + 1}\ \text{evalodd-add}$$

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 can work.

After "By structural induction on", state what you are inducting on.

After "Induction hypothesis:", write the appropriate induction hypothesis.

You will need to "case-split" on something. **After "Consider cases of :", write what you are considering cases of** (for example: the form of the expression e, or the rule concluding $\mathcal{D}$, or something else of your choosing).

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

Proof of "Always odd"

Conjecture 3.1 (Always odd)**.**
For all expressions e and derivations $\mathcal{D}$ such that $\mathcal{D}$ derives $e \Downarrow_{\text{odd}} n$,
there exists $m \in \mathbb{Z}$ such that $n = 2m + 1$.

Proof. By structural induction on e

Induction hypothesis: For all expressions $e' \prec e$ and derivations $\mathcal{D}'$ such that $\mathcal{D}'$ derives $e' \Downarrow_{\text{odd}} n$, there exists $m \in \mathbb{Z}$ such that $n = 2m + 1$.

Consider cases of e

- Case: $e = n$.
  $n = 7$ By inversion
  $n = 2 * 3 + 1$ By arithmetic
  Let $m = 3$.
  $n = 2m + 1$

- Case: $e = (\texttt{+}\ e_1\ e_2)$.
  For all expressions $e_1, e_2 \prec e$ and derivations $\mathcal{D}_1$, $\mathcal{D}_2$ such that $\mathcal{D}_1$ derives $e_1 \Downarrow_{\text{odd}} n_1$, $\mathcal{D}_2$ derives $e_2 \Downarrow_{\text{odd}} n_2$, there exists $m_1, m_2 \in \mathbb{Z}$ such that $n_1 = 2m_1 + 1$, $n_2 = 2m_2 + 1$.
  $(\texttt{+}\ e_1\ e_2) \Downarrow_{\text{odd}} 2n_1 + 2n_2 + 1$ By evalodd-add
  $n = 2n_1 + 2n_2 + 1$ By inversion
  Let $m = n_1 + n_2$.
  $n = 2m + 1$

□

Soundness and completeness

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

$\boxed{e\ {}_{\text{odd}}\!\Downarrow v}$ expression e evaluates to value v, but only works for odd numbers

$$\frac{}{2n+1\ {}_{\text{odd}}\!\Downarrow 2n+1}\ \text{oddeval-const} \qquad \frac{e_1\ {}_{\text{odd}}\!\Downarrow n_1 \qquad e_2\ {}_{\text{odd}}\!\Downarrow n_2}{(+\ e_1\ e_2)\ {}_{\text{odd}}\!\Downarrow n_1+n_2}\ \text{oddeval-add}$$

We now have three different definitions of big-step evaluation: the "real" definition $e \Downarrow v$, and two "joke" definitions, $e \Downarrow_{\text{odd}} v$ and $e\ {}_{\text{odd}}\!\Downarrow v$. 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 $e \Downarrow v$. We repeat those rules here:

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

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

Each of the following questions asks you to either state *and prove* the appropriate theorem, or give a counterexample.

(a) Is the theory $e \Downarrow_{\text{odd}} v$ sound with respect to $e \Downarrow v$?

If it is sound, state *and prove* the appropriate theorem.

✓If it is not sound, give a counterexample (e and v such that $e \Downarrow_{\text{odd}} v$ is derivable, but $e \Downarrow v$ is not derivable).

Let $e = 10$.

$e \Downarrow_{\text{odd}} 7$	By evalodd-const
$e \Downarrow 10$	By eval-const
$7 \neq 10$	By arithmetic

Thus it is not sound.

(b) Is the theory $e \Downarrow_{\text{odd}} v$ complete with respect to $e \Downarrow v$?

If it is complete, state *and prove* the appropriate theorem.

✓If it is not complete, give a counterexample (e and v such that $e \Downarrow v$ is derivable, but $e \Downarrow_{\text{odd}} v$ is not derivable).

Let $e = 10$.

$e \Downarrow 10$	By eval-const
$e \Downarrow_{\text{odd}} 7$	By evalodd-const
$10 \neq 7$	By arithmetic

Thus it is not sound.

(c) Is the theory $e \;{}_{\mathsf{odd}}\!\Downarrow v$ sound with respect to $e \Downarrow v$?

✓If it is sound, state *and prove* the appropriate theorem.

If it is not sound, give a counterexample (e and v such that $e \;{}_{\mathsf{odd}}\!\Downarrow v$ is derivable, but $e \Downarrow v$ is not derivable).

Problem Statement: Theory $e \;{}_{\mathsf{odd}}\!\Downarrow v$ is sound with respect to $e \Downarrow v$

Proof: By structural induction on e

Induction hypothesis: For all $e' \prec e$ expressions and derivations $\mathcal{D}'$ such that $\mathcal{D}'$ derives $e' \;{}_{\mathsf{odd}}\!\Downarrow v'$, then $e' \Downarrow v'$.

Consider cases of e

(a) Case: $e = 2n + 1$.

$e \;{}_{\mathsf{odd}}\!\Downarrow 2n+1$	By oddeval-const
$e \Downarrow 2n+1$	By eval-const

(b) Case: $e = (\texttt{+}\ e_1\ e_2)$.

$e_1 \;{}_{\mathsf{odd}}\!\Downarrow n_1$	By assumption
$e_1 \Downarrow n_1$	By IH
$e_2 \;{}_{\mathsf{odd}}\!\Downarrow n_2$	By assumption
$e_2 \Downarrow n_2$	By IH
$e \;{}_{\mathsf{odd}}\!\Downarrow n_1 + n_2$	By oddeval-add
$e \Downarrow n_1 + n_2$	By eval-add

A counterexample fits on a line, and a proof doesn't, so I can't give you an appropriate amount of space without giving away parts of the answers. Therefore, please write your answers on the next page.

If you need to write one or more proofs, I suggest following the language of Conjecture 3.1: "For all . . . , . . . " and the form of that proof: say what you are inducting on, state the induction hypothesis, and give all necessary detail.