Submission instructions:

This assignment should be done together with your study group. Only one assignment should be submitted per group, and it does not matter which group member submits the assignment. However, make sure that all group members’ names are listed on the submission. These members will be in your group for the entire semester, unless there are special circumstances. If you receive help from someone outside of your study group, excluding the course staff, you must also acknowledge them on your assignment.

Solutions must be submitted electronically (in ascii, postscript, or PDF format). Follow the instructions at [http://www.seas.upenn.edu/~cis500/homework.html](http://www.seas.upenn.edu/~cis500/homework.html).

1 Exercise Consider this simple grammar:

\[
t ::= \text{One} \\
    \text{Zero} \\
    \text{OneAnd} \ t \\
    \text{ZeroAnd} \ t
\]

Terms \( t \) can be thought of as representing binary integers abstractly. For example, the representation of 5 (i.e., 101 in binary) is \( \text{OneAnd} \ (\text{ZeroAnd} \ \text{One}) \).

State the structural induction principle for terms \( t \).

2 Exercise Using the big-step operational semantics for the Arith language (listed at the end of the assignment), we can show that

\[
\text{if} \ (\text{if} \ (\text{iszero} \ (\text{succ} \ 0)) \ \text{then} \ \text{false} \ \text{else} \ \text{true}) \ \text{then} \ \text{succ} \ (\text{if} \ \text{true} \ \text{then} \ 0 \ \text{else} \ (\text{succ} \ 0)) \ \text{else} \ 0 \downarrow \ \text{succ} \ 0
\]

Give a derivation witnessing this fact, noting the rule applied at each step. (Hint: you might want to start by drawing the parse tree for the term on the left-hand side first.)

3 Exercise 3.5.13 in TAPL.

4 Exercise 3.5.17 in TAPL.

5 Exercise The operational semantics of the Arith language (listed at the end of the assignment) is not total. In other words, the following property does not hold: For all \( t \), there exists a \( t' \) such that \( t \downarrow t' \).

In this problem, we will fix that by introducing a new expression to be the result of evaluating terms like \( \text{succ} \ \text{false} \). To do so, we add \( \text{wrong} \) to the syntax of the terms of Arith and introduce two new syntactic categories.

\[
t ::= \ldots \\
\text{wrong}
\]

badnat ::= non-numeric normal forms
bv boolean values
wrong run-time error
badbool ::= non-boolean normal forms
nv numeric values
wrong run-time error

We also augment the large-step evaluation relation with the following new rules:
Show that the operational semantics of the language with these additions is total. Although many of the parts of this proof may be similar to each other, for this first assignment, write each case out in full.

6 Exercise  Recall the abstract syntax of “binary” numbers, from the first exercise.

t ::= One
    Zero
    OneAnd t
    ZeroAnd t

Consider the following inductive definition of a function that returns the natural number that a term represents:

\[
\begin{align*}
\text{number}(\text{Zero}) &= 0 \\
\text{number}(\text{One}) &= 1 \\
\text{number}(\text{ZeroAnd } t) &= 2 \cdot \text{number}(t) \\
\text{number}(\text{OneAnd } t) &= 2 \cdot \text{number}(t) + 1
\end{align*}
\]

This function can be viewed as a relation \( \text{number} \subseteq (\text{Terms}, \mathbb{N}) \), between terms and natural numbers. Here’s an equivalent definition of the function, using inference rules:

\[
(\text{Zero}, 0) \in \text{number}
\]

\[
(\text{One}, 1) \in \text{number}
\]

\[
(t, i) \in \text{number} \\
(\text{ZeroAnd } t, 2 \cdot i) \in \text{number}
\]

\[
(t, i) \in \text{number} \\
(\text{OneAnd } t, 2 \cdot i + 1) \in \text{number}
\]

1. Define a function \( \text{length}(.) \) that returns the length of a given term when viewed as a bitstring. For example it must be that:
As with \( \text{number}(.) \), present \( \text{length}(.) \) in two ways, with both the inductive definition and with inference rules.

2. Define a function \( \text{zeros}(.) \) that returns the number of zeros that a given term contains when viewed as a bitstring. For example:

\[
\text{zeros(ZeroAnd (OneAnd ZeroAnd One))} = 2
\]

Similarly, provide both the inductive definition of the function as well as inference rules.

3. Using inference rules, define a relation \( \# \subseteq (\text{Terms}; \text{Terms}) \) such that:

\[
\text{zeros}(t) = \text{length}(t') \Rightarrow t \# t'
\]

There are many such relations – e.g. the empty relation is one of them! – but try to define the biggest relation you can. Prove that:

\[
\text{zeros}(t) = \text{length}(t') \Rightarrow t \# t'
\]

The following is not to be turned in. Depending on your definition of the relation above, you may or may not be able to prove that

\[
\text{zeros}(t) = \text{length}(t') \Rightarrow t \# t'
\]

You might want to try to prove this for the relation you came up with before, using induction on the pair of terms \( (t, t') \). More details will be provided in the homework solutions.

7 Debriefing

1. How many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. Did everyone in your study group participate?
4. How deeply do you feel you understand the material it covers (0%–100%)?

If you have any other comments, we would like to hear them; please send them cis500@cis.upenn.edu.
The Arith Language

Syntax

\[
\begin{align*}
t & ::= \\
& \text{true} \\
& \text{false} \\
& 0 \\
& \text{succt} \\
& \text{pred t} \\
& \text{iszero t} \\
& \text{if t then t else t}
\end{align*}
\]

Big-step operational semantics

\[
\begin{align*}
v & \downarrow v & \text{B-Value} \\
\text{v} & \downarrow \text{true} & \text{B-IfTrue} \\
\text{v} & \downarrow \text{false} & \text{B-IfFalse} \\
\text{succ t} & \downarrow \text{succt} & \text{B-Succ} \\
\text{pred t} & \downarrow \text{nv} & \text{B-Pred} \\
\text{iszero t} & \downarrow \text{true} & \text{B-IsZeroZero} \\
\text{iszero t} & \downarrow \text{false} & \text{B-IsZeroSucc}
\end{align*}
\]

Solutions

Some solutions can be found in the back of the book. You should write out your answers to the assignment before looking. Note, however, that the solution to 3.5.13 is incorrect there – see http://www.cis.upenn.edu/~bcpierce/tapl/errata.txt.