CIS 500 Software Foundations

Homework Assignment 3

Induction; Operational Semantics

Due: Monday, September 22, 2002, by noon

Submission instructions:

You may submit your solutions either electronically (in ascii, postscript, or PDF format) or in handwritten hardcopy form.

- Hand-written solutions should be submitted to Jennifer Finley in Levine 302.
- Electronic solutions should be submitted following the same instructions as last time; these can be found at `http://www.seas.upenn.edu/~cis500/homework.html`.

1 Exercise Explain the flaw in the following "proof."

> **Theorem(?!):** All horses are the same color.
>
> **Proof:** Let $P(n)$ be the predicate "in all non-empty collections of n horses, all the horses are the same color." We show that $P(n)$ holds for all n by induction on n (using 1 as the base case).
>
> **Base case**: Clearly, $P(1)$ holds.
>
> **Induction case**: Given $P(n)$, we must show $P(n+1)$.
>
> Consider an arbitrary collection of $n+1$ horses. Remove one horse temporarily. Now we have n horses and hence, by the induction hypothesis, these n horses are all the same color. Now call the exiled horse back and send a different horse away. Again, we have a collection of n horses, which, by the induction hypothesis, are all the same color. Moreover, these n horses are the same color as the first collection. Thus, the horse we brought back was the same color as the second horse we sent away, and all the $n+1$ horses are the same color.

2 Exercise How about this one?

> **Theorem(?!):** $n^2 + n$ is odd for every $n \geq 1$.
>
> **Proof:** By induction on n (again starting from 1). For the base case, observe that 1 is odd by definition. For the induction step, assume that n^2+n is odd; we then show that $(n+1)^2+(n+1)$ is odd as follows. $(n+1)^2 + (n+1) = n^2 + 2n + 1 + n + 1 = (n^2 + n) + (2n + 2)$. But $n^2 + n$ is odd by the induction hypothesis, and $2n + 2$ is clearly even. Thus, $(n^2 + n) + (2n + 2)$ is the sum of an odd number and an even number, hence odd.

3 Exercise [Optional] Make up your own "false proof" in which an incorrect use of the induction hypothesis leads to a surprising conclusion.

4 Exercise 3.5.13 in TAPL.

5 Exercise 3.5.17 in TAPL.

6 Exercise 4.2.2 in TAPL.

To complete this exercise, you will need to download your own copy of the files for the `arith` implementation. These files can be found on the TAPL web site

`http://www.cis.upenn.edu/~bcpierce/tapl/checkers/arith.tar.gz` (as a single bundle)
`http://www.cis.upenn.edu/~bcpierce/tapl/checkers/arith` (as separate files)

Instructions for compiling these files into a running implementation can be found here:

`http://www.cis.upenn.edu/~bcpierce/tapl/resources.html#checkers`

Chapter 10 of Jason Hickey's notes has more information on OCaml compilation.
The only changes you will need to make for this problem are to the `eval` function in the file `core.ml`. Please include your version of *just this file* in the solutions you hand in.

7 Debriefing

1. Are you in a study group?
2. If so, who else is in it?
3. How many hours did you spend on this assignment?
4. Would you rate it as easy, moderate, or difficult?
5. Did you work on it mostly alone, or mostly with your study group?
6. 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 directly to `bcpierce@cis.upenn.edu` to make sure they don't get lost in a pile of homeworks!

Solutions

The solutions not given here can be found in the back of the book.

1. The problem with this proof is that the inductive step does not work for all n, namely for $n = 1$. According to our proof, to show that $n + 1 = 2$ horses are the same color, we consider horse #1 with color C_1 and then we separately consider horse #2 with color C_2. The two groups of horses have no overlapping members, and thus we have no reason to assert that these two horses have the same color. The argument works fine when $n > 1$, but the $n = 1$ case invalidates the induction.

2. The base case is confused — it does not show what it needs to show.

6. The definitions of `NoRuleApplies`, `isnumericval`, and `isval` are the same as in the original; `eval1` is deleted; `eval` is rewritten as follows:

```
let rec eval t = match t with
    v when isval v -> v
  | TmIf(_,t1,t2,t3) ->
      begin
        match eval t1 with
          TmTrue _ -> eval t2
        | TmFalse _ -> eval t3
        | _ -> raise NoRuleApplies
      end
  | TmSucc(fi,t1) ->
      begin
        match eval t1 with
          nv1 when isnumericval nv1 -> TmSucc (dummyinfo, nv1)
        | _ -> raise NoRuleApplies
      end
  | TmPred(fi,t1) ->
      begin
        match eval t1 with
          TmZero _ -> TmTrue(dummyinfo)
        | TmSucc(_, nv1) -> nv1
        | _ -> raise NoRuleApplies
      end
  | TmIsZero(fi,t1) ->
      begin
        match eval t1 with
          TmZero _ -> TmTrue(dummyinfo)
        | TmSucc(_, _) -> TmFalse(dummyinfo)
        | _ -> raise NoRuleApplies
      end
  | _ ->
      raise NoRuleApplies
```