Compiler Construction/Fall 2015/Homework 4

Eva Rose *evarose@cs.nyu.edu*

Kristoffer Rose *krisrose@cs.nyu.edu*

Assigned Thursday 10/1/2015, due Thursday 10/8/2015 at 8am

Reading Assignments

- Lecture on 10/1/2015 (this homework): Dragon book 2.3, 5.4 (21p); HACS §5+§10.
- Lecture on 10/8/2015: Dragon book 5.1–5.3 + 6.3, 6.5 (40p).
- *Project Milestone 1 due 10/12/2015*.

Homework Assignments

The following assignments should be submitted for a maximum of 20 points and 5 bonus points.

1 Recursive Schemes

Consider the following grammar for nested lists over numbers.

$$L \to \textbf{num}\ L \mid \{\ L\ \}\ L \mid \epsilon$$

Question 1.1 (3 points)**.** Write a syntax-directed recursive scheme that computes the product of all numbers in a list.

Question 1.2 (3 points)**.** Write a syntax-directed recursive scheme that generates a flattened copy of a nested list with all the same numbers in the same order but with no {}s.

Question 1.3 (4 points)**.** Write a syntax-directed recursive scheme that computes the reverse of a nested list and all the contained nested lists.

Question 1.4 (5 bonus points)**.** Implement the previous schemes in HACS and demonstrate how to run them. (You can use the "Computed" sort or your own homebrew "Add" dummy construction to represent addition.)

2 HACS

In this exercise, we will study and extend the HACS script in Figure 1.

Question 2.1 (5 points)**.** Use HACS to compile *Bool.hx* and run HACS to show the output of parsing the term `((T)&!(F|T))`. Explain why the output is different from the input.

Question 2.2 (5 points)**.** Replace the two lines marked “MISSING” in the code with proper HACS rules to complete the system to also handle disjunction. Show that you can then run your system as follows:

```
$ hacs Bool.hx
$ ./Bool.run --scheme=Evaluate --term='((T)&!(F|T))'
F
```

```
module edu.nyu.cs.cc.Bool {

  // Syntax.
  sort B
  | ⟦ T ⟧@4| ⟦ F ⟧@4                  // constants
  | sugar ⟦ ( ⟨B#⟩ ) ⟧@4→B#           // parenthesis
  | ⟦ ⟨B@2⟩ | ⟨B@1⟩ ⟧@1              // disjunction
  | ⟦ ⟨B@3⟩ & ⟨B@2⟩ ⟧@2               // conjunction
  | ⟦ ! ⟨B@3⟩ ⟧@3                     // negation
  ;

  // Semantic operations.
  sort B;
  | scheme And(B,B) ;
  And(⟦T⟧, #2) →#2 ;
  And(⟦F⟧, #2) →⟦F⟧ ;

  | scheme Or(B,B) ;
  //MISSING: rules for Or

  | scheme Not(B) ;
  Not(⟦F⟧) →⟦T⟧ ;
  Not(⟦T⟧) →⟦F⟧ ;

  // Recursive Evaluation scheme
  sort B;
  | scheme Evaluate(B) ;
  Evaluate(⟦ T ⟧) →⟦ T ⟧ ;
  Evaluate(⟦ F ⟧) →⟦ F ⟧ ;
  // MISSING: evaluation of disjunction
  Evaluate(⟦ ⟨B#1⟩ & ⟨B#2⟩ ⟧) →And(Evaluate(#1), Evaluate(#2)) ;
  Evaluate(⟦ ! ⟨B#⟩ ⟧) →Not(Evaluate(#)) ;
}
```

Figure 1: *Bool.hx* HACS script.