CS536: Homework 6

Keith Funkhouser

November 5th, 2015

1

Suppose that students' grades are kept in a file with the following format:

- The file contains one or more student records.
- Each student record has one student's name, then their ID number, then zero or more grades, separated by commas.
- Each grade is an integer value followed by zero or more stars (to represent the number of days late).

Here's a CFG for the grade file format:

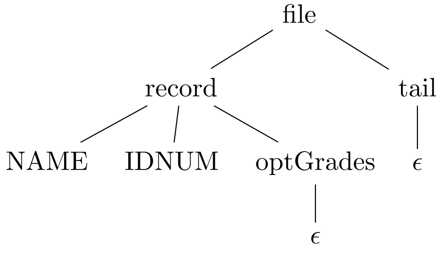
```
file → record tail
tail → file | ϵ
record → NAME IDNUM optGrades
optGrades → grades | ϵ
grades → oneGrade | oneGrade COMMA grades
oneGrade → INTLIT optLate
optLate → stars | ϵ
stars → STAR | stars STAR
```

1. Although the grade file CFG given above is not LL(1), some correct inputs can be parsed by a predictive parser. This is because those inputs never cause the parser to look at a table entry that contains two or more CFG rules.

   (a) Give the shortest such input (as a sequence of tokens, ending with `EOF`).
   Consider the following input, which is correct and can be parsed by a predictive parser:

   ```
   NAME IDNUM EOF
   ```

   (b) Draw the parse tree that the parser would build for the input you gave for Part (i). (Do not include the `EOF` token in the parse tree.)

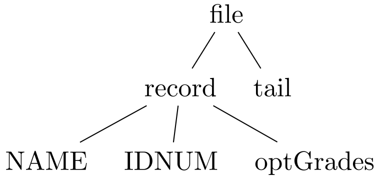


2. What is the shortest sequence of tokens that would “stump” a predictive parser trying to parse the language of this CFG (i.e., a sequence of tokens that is a prefix of a valid input, but for which the parser would not know how to continue to build the parse tree top-down because it looks at a table entry that contains two or more CFG rules)? To answer this question, give all of the following:

   (a) The sequence of tokens (a **prefix** of a valid input).

   ```
   NAME IDNUM INTLIT
   ```

   (b) The (partial) parse tree that the predictive parser would have built before being stumped.



   (c) The CFG rules that the predictive parser can’t choose between to continue to grow the parse tree.

   ```
   grades → oneGrade | oneGrade COMMA grades
   ```

2

This question concerns the following grammar (where non-terminals are in lower-case and terminals are in upper-case):

```
aa → A ee bb B

bb → cc bb
   | ϵ

cc → D C
   | F ee bb B dd

dd → E A ee bb B
   | ϵ

ee → ff ee
   | ϵ

ff → hh D gg

gg → C
   | H C

hh → G
   | J
```

Provide a table with the FIRST and FOLLOW sets for all of this grammar’s nonterminals. Provide a second table with the FIRST sets for all of the production right-hand sides.

FIRST sets for nonterminals:

x	FIRST(x)
hh	G, J
gg	C, H
ff	G, J
ee	G, J, ϵ
dd	E, ϵ
cc	D, F
bb	D, F, ϵ
aa	A

FOLLOW sets for non-terminals:

x	FOLLOW(x)
aa	EOF
bb	B
cc	D, F, B
dd	D, F, B
ee	D, F, B
ff	G, J, D, F, B
gg	G, J, D, F, B
hh	D

FIRST sets for production right-hand sides (note: only those productions which do not produce single terminals are shown below):

x	FIRST(x)
A ee bb B	A
cc bb	D, F
D C	D
F ee bb B dd	F
E A ee bb B	E
ff ee	G, J
hh D gg	G, J
H C	H
ϵ	ϵ