CS241 Final Review

Yang Tian Zi, Edward Tan

April 7, 2017

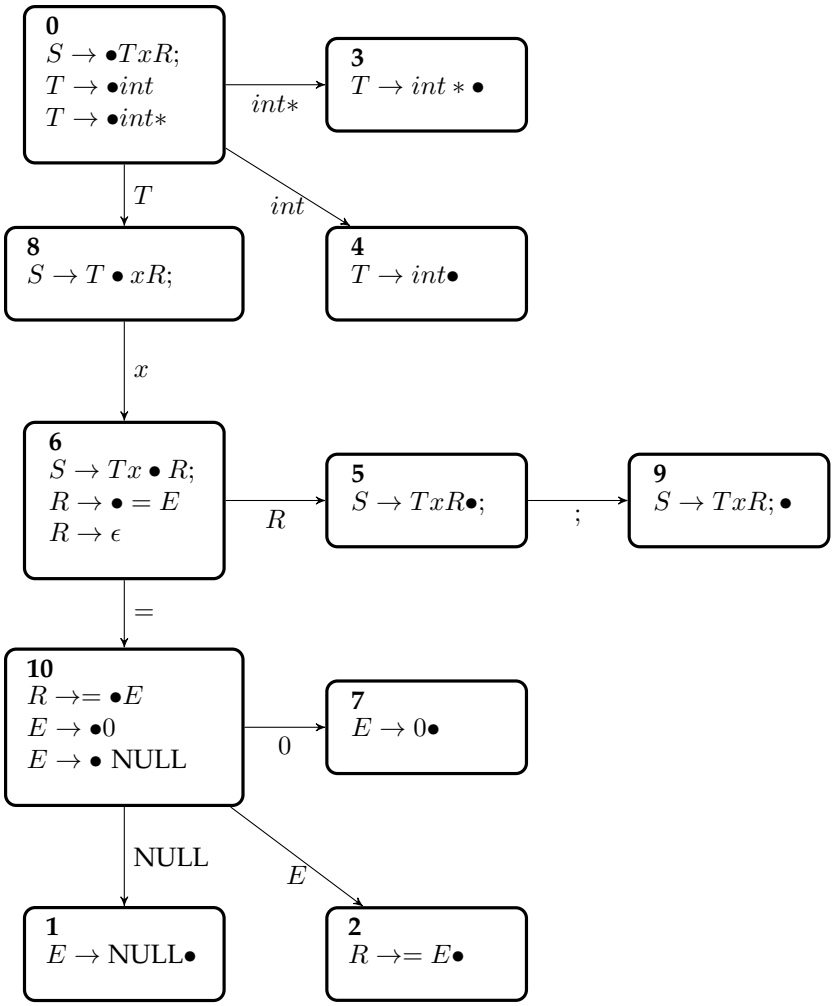
1 Scanning and Parsing

1. For the set of tokens $T = \{\text{ggg}, \text{ty}, \text{gg}, \text{b}, \text{gl}, \text{hf}, ?\}$, tokenize the following string using maximal munch and simplified maximal munch. Indicate the tokens parsed, and if an error occur, indicate when it happens.
   (a) tybggbggbggg
   (b) glhf???
   (c) gtyggggggg
2. Consider the following CFG with E being the start state.

$$E \to E + T$$
$$E \to T$$
$$T \to F * T$$
$$T \to F$$
$$F \to a|b|c$$

   (a) Construct the first, follow and nullable sets.
   (b) Construct the predictor table.
   (c) Is this grammar LL(1)? Why? If not, modify the grammar to make it LL(1), if possible.
3. Consider the following CFG with start state S and the SLR(1) DFA:

$$S \to T \text{ x } R;$$
$$T \to int$$
$$T \to int*$$
$$R \to= E$$
$$R \to \epsilon$$
$$E \to \text{NULL}$$
$$E \to 0$$



perform LR parsing with the following strings:

(a) `int* x = 0;`

(b) `int x;`

(c) `int* x = NULL;`

(d) `int x = 1;`

2 Semantic Analysis and Code Generation

1. For the following WLP4 programs, find and circle the error (if there are any) and indicate which of the following stage will report the error: Scanning, Parsing, Semantic Analysis, Code Generation, Program Execution.

(a)
```
int main ( int a, int b ){
    int c = 0;
    return a+b+c;
}
```

(b)
```
int wain ( int *a, int b ){
    return a + *a + b - *b;
}
```

(c)
```
int wain ( int a, int b ){
    int *x = NULL;
    x = new int [a+b];
    a = *&*&(a) + *x;
    return a:
}
```

(d)
```
int wain ( int *a, int b ) {
  if (*a < b){
     b = *a;
  }
  return b;
}
```

(e)
```
int wain ( int a, int b ){
    println( &a - &b / 2 );
    return a;
}
```

2. For the machine-generated MIPS program below, give an equivalent program in WLP4. The symbol table is provided to help you resolve variable names and type.

Name	Type	Offset
a	int	0
b	int	4

```
.import init
.import new
.import delete

lis $4
.word 4

sw $31, -4($30)
sub $30, $30, $4

sub $29, $30, $4
sw $1, 0($29)
sw $2, -4($29)
lis $3
.word 8
sub $30, $29, $3

add $2, $0, $0
lis $20
.word init
jalr $20
lw $3, 0($29)
add $1, $3, $0
lis $21
.word new
jalr $21
sw $3, 0($30)
sub $30, $30, $4

lw $3, -4($29)
add $1, $3, $0
jalr $21
add $30, $30, $4
lw $5, 0($30)
sub $3, $5, $3
div $3, $4
mflo $3

add $30, $29, $4
add $30, $30, $4
lw $31, -4($30)
jr $31
```

3. Many modern languages support **Ternary (Conditional) Operator**. In C++, we can use ternary operator with the following example:

```
int a = 5;
a = ( a >= 5 ) ?  a + 2 : a + 1;
```

The code above is equivalent to the following:

$$a = \begin{cases} \text{a} + 2 \text{, if a } \geq 5 \\ \text{a} + 1 \text{, otherwise} \end{cases}$$

For this question, you will add this feature to the WLP4 language.

(a) List the additional tokens required by the WLP4 scanner. Give a DFA that recognize these tokens.

(b) Extend the WLP4 grammar to support ternary operators. You may assume that all tenary expressions are derived from non-terminal symbol *expr*. For our version of ternary operator, you only have to support at most one comparison in the condition, and all conditions are enclosed by parenthesis.

(c) Describe the new rules we have to check during semantic analysis.

(d) Write the pseudocode to generates MIPS code for these new rules.

3 Memory Management

1. What does fragmentation mean? What is the difference between internal fragmentation and external fragmentation?
2. List one advantage and one disadvantage of using fixed size blocks when allocating memory.
3. What is the 50% rule?
4. In this question, you are given a heap with N units of memory and 2 functions:

   ```
   void *new (int n); // allocate n units of memory in arena
   void delete (void *p); // reclaim the memory
   ```

   For each scenario below, give a sequence of the new/delete function calls that will fails to allocate memory from the heap. You are not allow to allocate memory more than what the heap currently has.

   (a) N = 30, it use First Fit to allocate memory.

   (b) N = 37, it uses Worst Fit to allocate memory.

   (c) N = 66, it uses Best Fit to allocate memory.

4 Additional Practice Problems

1. **Boolean type** is a common data type used in many programming languages. In this question, we will extend WLP4 to support boolean type with three common operators : && , || and !.

   (a) The following C++ code segment uses boolean expression to write an if statement. Give an equivalent WLP4 code, assume that all variables are type int and declared. (Note: The precedency for the operators are Not $>$ And $>$ Or.)

   ```
   if ( a > 10 && a < 20 || !(b == 3)){
       println(a);
   }else{
       println(b);
   }
   ```

   (b) To handle boolean type, we need to first recognize boolean expressions. Extend the WLP4 grammar to support all possible boolean expressions. You may assume that all boolean expressions are derived from non-terminal symbol *expr*. The keyword "true" is represented with terminal symbol *TRUE*, "false" with *FALSE*, and "boolean" with *BOOLEAN*.

   (c) Extend WLP4 grammar to support boolean type declarations and assignments. For simplicity, a boolean variable can only be true or false when initializing. An example of a boolean type variable declarations and assignments will be the following:

   ```
   boolean a = true;
   boolean b = false;

   b = a || b;
   a = a && b;
   a = !(2 == 2);
   ```

   (d) Modify the WLP4 grammar to allow boolean type variables and boolean expression used in if and while statements.

(e) Describe all new rules for semantic analysis. You should not assume any numeric values for false and true, and you should not make implicit conversion between integer and boolean.

(f) i. Give the pseudocode that generates MIPS code for all rules with boolean expressions.

ii. Give the pseudocode that generates MIPS code for all rules with boolean variable declaration and assignments.

iii. Give the pseudocode that generates MIPS code for all rules with IF and WHILE statement with the addition of boolean type variables and boolean expressions.

2. Can an ambiguous grammar be LL(1) or SLR(1)? Why?