Review Set 1

Kevin Leach

January 26, 2018

1 Cool Syntax

1. Consider the Cool program below

```
class Main inherits IO {
    x : Int <- 5;
    main () : Object {
        let newa : A <- new A, newb : A <- new B, newc : A <- new C in
        {
            out_string(newa.m(1));
            out_string(newb.m(1));
            out_string(newc.m(1));
            out_string(newc@A.m(2));
        }
    };
};

class A {
    v : Int;
    m ( x : Int ) : String {
        if x < v then
            "Hello\n"
        else
            "Goodbye\n"
        fi
    };
    setV (newv : Int) {
        v <- newv
    };
};

class B inherits A {
    m ( x : Int ) : String {
        if x > v then
            "Hola\n"
        else
            "Adios\n"
        fi
    };
    setV (newv : Int) {
        v <- newv + 1
    };
};
```

```
class C inherits A {
    m ( x : Int ) : String {
        if x = v then
            "Nihao\n"
        else
            "Zaijian\n"
        fi
    };
};
```

What is the output of this program?

2 Regular Expressions

1. Write a regular expression over the alphabet $\Sigma = \{a, b\}$ for the language of strings that have an odd (and non-zero) number of occurrences of a.

2. Draw a **NFA** that accepts the language from the above problem.

3. ALWAYS, SOMETIMES, NEVER. Given a regular expression r, there exists a DFA d such that $L(r) = L(d)$.

4. ALWAYS, SOMETIMES, NEVER. Given a regular expression r, there exists an LL(1) grammar g such that $L(r) = L(g)$.

5. ALWAYS, SOMETIMES, NEVER. Given a context-free grammar g, there exists a regular expression r such that $L(g) = L(r)$.

3 Context-Free Grammars

1. Consider the following grammar with terminals x, y.

$$\begin{array}{lcl} S & \rightarrow & Ax \\ & | & Ay \\ A & \rightarrow & Bxx \\ B & \rightarrow & By \\ & | & \epsilon \end{array}$$

   (a) Give at least 3 strings that are in this language.

   (b) Is this grammar ambiguous? Why or why not?

   (c) Is this language recursive? Why or why not?

   (d) Left-factor the production rule for non-terminal S.

2. Consider the following grammar with four terminals: =, +, *, `int`

$$\begin{array}{lcl} S & \rightarrow & B + B \\ A & \rightarrow & * \\ B & \rightarrow & \epsilon \\ B & \rightarrow & \texttt{int}BB \\ B & \rightarrow & A = \end{array}$$

   (a) Fill in the table with the First and Follow sets for the non-terminals

	FIRST	FOLLOW
S		
A		
B		

   (b) Fill in the LL(1) parsing table

	=	+	*	$\widehat{\text{int}}$	$
S					
A					
B					

   (c) Is this grammar LL(1)? Why or why not?