

Design and Selection of Programming Languages

13 September 2005

Exercise 1.1 — Finite-State Machines

- Given two non-deterministic finite-state machines M_1 and M_2 , construct (mathematically) a non-deterministic finite-state machines M_\cup that accepts exactly the **intersection** of the languages accepted by M_1 and M_2 .
- Given two non-deterministic finite-state machines M_1 and M_2 , construct (mathematically) a non-deterministic finite-state machines M_\cup that accepts exactly the **union** of the languages accepted by M_1 and M_2 .
- What changes with the move to **deterministic** finite-state machines?

Exercise 1.2 — *TokenStream* (Textbook Ex. 2.3)

Starting from the skeleton `TokenStream.java` that can be found among the student materials on the textbook's web site, complete the implementation of the method `nextToken` and all its auxiliary methods.

Test the completed `TokenStream` class using the example Jay program `program.jay` (Fig. 2.19 in the textbook) to certify that it finds each token in the input stream and correctly classifies each token type.

Exercise 1.3 — Expression Representation in Java

The following expression classes are based on those of last year's project assignment in SE2S; only **variables** have been added.

```
class Operator {
    private char _op;
    public Operator (char c) { _op = c; }
}

abstract class Expression {} // Expression = Value + Variable + Binary

class Value extends Expression { // Value = int
    int intValue;
}
class Variable extends Expression { // Variable = String
    String name;
}
class Binary extends Expression { // Binary = Expression × Operator × Expression
    Operator op;
    Expression term1, term2;
}
```

- (a) Add a `toString0` method to these classes such that `e.toString0()` produces an infix representation of any *Expression* `e` with **at least** (see c) the necessary parentheses for being able to apply higher-priority operators like `*` to argument expressions constructed from lower-priority operators like `+`.
- (b) Make the fields private and add constructors that ensure that only expressions with reasonable string representation can be constructed.

Document your definition of “reasonable string representation” in each case!

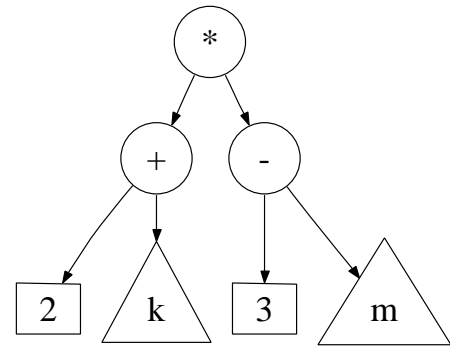
- (c) Add a `toString` method to these classes such that `e.toString()` produces an infix representation of any *Expression* `e` with the **minimally necessary parentheses** for being able to apply higher-priority operators like `*` to argument expressions constructed from lower-priority operators like `+`.
- (d) The `graphviz` tool suite from AT&T includes, among others, the graph layout tools
 - `dot` for layout of directed, typically acyclic graphs — the main principle of the algorithm is to arrange nodes into “levels”, and then reduce edge crossings, and
 - `neato` for layout of undirected graphs using a “spring embedding” algorithm that understands nodes as electrically charged and therefore repelling each other, and edges as springs of equal spring constants and lengths, and therefore produces always drawings with straight-line edges.

To the right is the the drawing that `dot` produces when invoked with

```
dot -Tps -o E1.ps E1.dot
```

on the following input file corresponding to the expression $(2 + k) * (3 - m)$:

```
digraph MT1a {
  node [fontsize="30"];
  edge [labeldistance="1",fontsize="30"];
  "+" [shape="circle"];
  "*" [shape="circle"];
  "-" [shape="circle"];
  "2" [shape="box"];
  "3" [shape="box"];
  "k" [shape="triangle"];
  "m" [shape="triangle"];
  "*" --> "+";
  "*" --> "-";
  "+" --> "k";
  "+" --> "2";
  "-" --> "3";
  "-" --> "m";
}
```



Extend the expression class with a `writeDotGraph` method that produces such a `dot` file.

Note: What should the interface of this method be? How do you distribute its task among the sub-classes?

Document and justify your design decisions! Test your output with `dot`!

Advanced: Add a method `showDotGraph` that writes the `dot` file, invokes `dot` on it, and invokes a program to display the output of `dot` (if necessary in a different format).