TP 2 : Lexical Analysis

bogdan.pasca@ens-lyon.fr

21 September 2010

1 Lexical Analysis

1.1 Regular expressions and finite automata

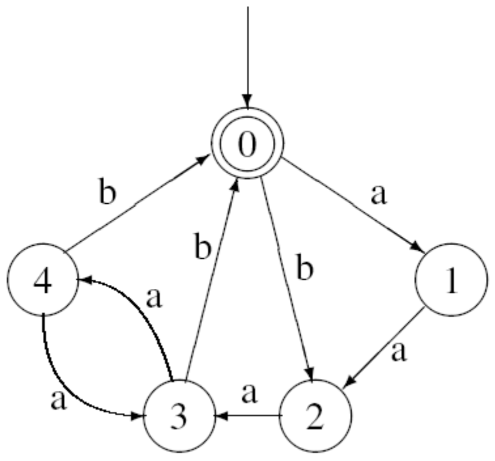
Exercise 1 Warm-up : Describe using regular expressions, the following :
a. the keyword `if`
b. a variable name in `C`
c. an integer number
d. a floating point number ; examples : 2.76, $-5.$, $.42$, $5e+4$, $11.22e-3$.
e. an integer number different from 42.
f. an integer number strictly greater than 42.
g. binary numbers n such that there exists an integer solution to $a^n + b^n = c^n$

Exercise 2 Convert the following regular expressions into nondeterministic finite automata using **Thompson's construction**. What are the drawbacks of this approach ?
a. $(if|then|else)$
b. $a((b|a^*c)x)^*|x^*a$
Convert these NFAs to a DFAs using the subset construction.

Exercise 3 Construct a DFA that recognizes balanced sequences of parenthesis with a maximal nesting depth of 3, e.g., ε, ()(), (()(())) or (()())()() but not (((()))) or (()(()(()))).

Exercise 4 Given that binary number strings are read with the most significant bit first and may have leading zeros, construct DFAs for each of the following languages :
a. Binary number strings that represent numbers that are multiples of 4, e.g., 0, 100 and 10100.
b. Binary number strings that represent numbers that are multiples of 5, e.g., 0, 101, 10100 and 11001.
c. Given a number n, what is the minimal number of states needed in a DFA that recognises binary numbers that are multiples of n ?

Exercise 5 Minimize the following DFA :



Exercise 6 Here is one algorithm for minimizing an automaton :

Step 1 Remove unreachable states.

Step 2 Mark the distinguishable pairs of states. To achieve this task, we first mark all pairs p, q, where $p \in F$ and $q \notin F$ as distinguishable. Then, we proceed as follows :

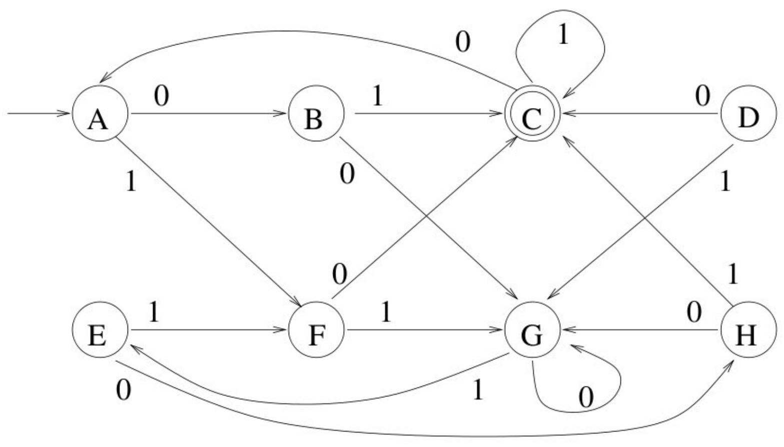
```
repeat
        for all non-marked pairs p, q do
                for each letter a do
                        if the pair $\delta(p,a)$,$\delta(q,a)$ is marked
                        then mark p, q
until no new pairs are marked
```

Step 3 Construct the reduced automaton A^*.

We first determine the equivalence classes of the indistinguishability relation. For each state q, the equivalence class of q consists of all states p for which the pair p, q is not marked in Step 2.

The states of A^* are the equivalence classes. The initial state $q0^*$ is this equivalence class that contains $q0$. The final states F^* are these equivalence classes that consist of final states of A. The transition function δ^* is defined as follows. To determine $\delta^*(X, a)$, for some equivalence class X, pick any $q \in X$, and set $\delta^*(X, a) = Y$, where Y is the equivalence class that contains $\delta(q, a)$.

Apply this minimization algorithm to the example below :



Exercise 7 Try to compress the transition table for the above automaton using the the **comb algorithm**(row displacement).

Exercise 8 Use Earley's method for creating a lexical analyzer automatically for detecting integer and floating-point numbers.

The full exercise

Exercise 9 If you have solved the exercises in the previous section, then you know how we can convert a language description written as a regular expression into an efficiently executable representation (a DFA). Now, we want to do something more : a program for lexical analysis, i.e. a lexer. It has to distinguish between several different types of tokens. For this exercise consider that the tokens can be : INTEGER, FLOAT, keyword IF, VARIABLE. Each of these are described by its own regular expression as in exercise 1. If there are several ways to split the input into legal tokens, the lexer has to decide which of these it should use. The simplest (dumb) approach would be to generate a DFA for each token definition and apply the DFAs one at a time to the input. Think about a smarter and faster way to generate a single DFA and test for all the tokens simultaneously. For this, use a principle similar to the previous section :

a. Create NFAs for each regular expression involved.
b. What are the accepting states of these NFAs ?
c. Think about a way to combine them in a single NFA.
d. Convert the combined NFA to a DFA
e. Can the same accepting state in the DFA accept several different token types ? If so, give a priority based scheme for solving this problem.
f. When we described minimisation of DFAs, we used two initial groups, one for accepting states and one for non-accepting states. How many initial groups do we have to use now ?
g. Compress the transition table.