CISC-499 Projects 2016–17

Kai Salomaa
School of Computing, Queen's University
`ksalomaa [AT] cs [DOT] queensu [DOT] ca`

1. **Converting machines to regular expressions**

   The well known state elimination algorithm converts a finite state machine to a regular expression. This algorithm tends to generate very large and redundant regular expressions, partly because it does no simplification. In this project you will develop and implement heuristics for simplifying regular expressions and apply these heuristics to the problem of converting finite state machines to regular expressions. The tasks can include the following:

   - investigate some of the current literature on simplification of regular expressions
   - study the expressions that tend to be generated in conversion of finite state machines, and characterize the types of simplification that would be useful
   - develop and implement a simplification heuristics for regular expressions
   - test the efficacy of your algorithms by using them in the conversion of finite state machines to regular expressions
   - (optional, if time permits) evaluate the cost of the simplification heuristics and develop a metric for deciding when to use them

   Grail is a symbolic computation environment for finite state automata, regular expressions and other language objects. Grail's `fmtore` function converts finite state machines to regular expressions using the state elimination algorithm. Alternatively, you can base the expreriments on some larger software library, such as Vaucanson `http://vaucanson-project.org/?eng`

   The project requires an understanding of the basics of finite automata and regular expressions. The amount of programming required is not large, but you should expect to run a significant number of simulations and other experiments.

2. **Transforming regular expressions to finite-state machines**

   Given a regular expression of length n, what is the worst-case size of the minimimal deterministic finite automaton (DFA) for the language? An exponential upper bound is known but average regular expressions can be implemented more efficiently.

   The main goal of this project is to generate libraries of "random" regular expressions and determine their state complexity. The regular expression–to–DFA transformations, as well as, the minimization of the DFAs have been automated using in the *Grail* environment. Larger software libraries such as Vaucanson `http://vaucanson-project.org/?eng` include much more functionality. The software libraries provide a large collection of operations to convert finite-state machines to regular expressions or vice versa and for minimizing finite state machines.

The second goal in the project can be to find different types of “bad” examples: regular expressions where the equivalent minimized DFA is large.

3. **Finite automata on trees: implementing basic operations**

   Tree automata are an extension of finite automata that operate on trees instead of strings. Many of the well known constructions and algorithms for finite automata extend in a natural way to tree automata that process inputs from the leaves to the root. General information on tree automata can be found at `http://tata.gforge.inria.fr/` (For the purposes of this project we need only basic definitions from section 1.1 of the TATA book project.)

   The goal of this project is to implement some basic operations for tree automata: (for example) the cross-product construction for union and intersection and the determinization of nondeterministic automata. One should design a suitable symbolic (Grail like) representation for the tree automata.

4. **Other projects**

   I have available some additional topics – please come to see me in my office or send me email. If you have your own idea for a project related to my research, please come to talk with me about it.