Readability

Sasha and Marco

May 17, 2017

Problem Statement: How can one decompose a system that may be given as a spaghetti mess, into something readable, e.g., a simple procedural part with declarative modules.

Related question: Can one mine a readable system from an event log?

Issues: One conceptual difficulty is formalising "readable". Let's take as working hypothesis that "readable" should be equated with "succinct as possible" (in frameworks where the constructs are understandable).

Example 1. *Suppose we are interested in finite-state systems, i.e., regular languages. Here are increasingly succinct models: DFA, NFA, AFA, Concurrent AFA. Each is exponentially more succinct than the previous. Thus, concurrent AFAs are 3exp more succinct than DFAs [1].*

Question 1. *Are automata later in this list more readable then equivalent automata earlier in the list?*

Here are some examples which should shed light on this question.

Example 2 (nondeterminism can improve readability)**.** *Consider the language L_n consisting of all binary strings whose nth last bit is 0. The smallest DFA for this language has about 2^n states, while the smallest NFA has about n states. Clearly, to me, the usual NFA for this language is more readable than any equivalent DFA.*

Example 3 (concurrency can improve readability)**.** *Consider the regular language $L_n = \{u\#v : u = v \in \{0,1\}^n\}$. It has an $O(n^2)$-state concurrent-DFA (the jth component stores the jth bit of u and compares this with the jth bit of v). But every NFA (and hence every DFA) for L_n requires about 2^n states (basically the NFA needs to store the word u in its state).*

Example 4 (concurrency need not improve readability)**.** *Consider the regular language $L_n = \{u : |u| = n\}$. The smallest DFA for this language requires n states and is fairly easy to analyse. However, there is a $\log^2 n$ concurrent DFA for this language that is harder to analyse, i.e., it uses $2\log n$ states and the conditions on the edges can be of length $\log n$. This is based on Figure 1.*

Example 5 (concurrent NFAs can be more readable than DFAs)**.** *Consider the regular language $M_n = \{\{0,1,\#\}\#w\#\{0,1,\#\}\$w : w \in \{0,1\}^n\}$. The smallest*

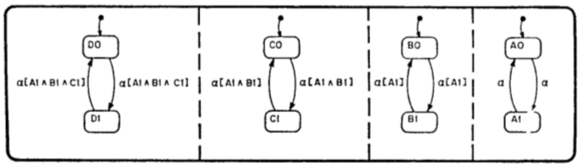

FIG. 6. A four-bit statechart counting the occurrences of α.

Figure 1: Taken from [1]

DFA for M_n requires about 2^{2^n} states (basically, it needs to remember all substrings of length n that occur before the \$, and there are doubly-exponentially many such sets). And thus the smallest NFA for M_n requires about 2^n states. However, there is a concurrent AFA of size about $\log^2 n$ for M_n. Basically, the AFA nondeterministically guesses the start of w; it then branches universally: on the one hand it checks that there the next # sign is n characters away; on the other, it reads the next bit (until a # is reached), remembers this bit, and starts a counter that freezes when # is reached, and resumes when \$ is reached, and when the counter reaches n it checks that the bit being read was the one remembered. By the previous example, one can implement a counter up to n with $\log^2 n$ states:

Question 2. *Does mixing alternation and concurrency improve readability?*

Finally, a rough/wild idea: using partial observability of k nondeterministic players one can probably save a tower of exponents of height k (this would be based on ideas of [2]).

References

[1] Doron Drusinsky and David Harel. On the power of bounded concurrency I: finite automata. *J. ACM*, 41(3):517–539, 1994.

[2] Gary L. Peterson and John H. Reif. Multiple-person alternation. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 348–363, 1979.