

Possible CDM Projects

- Universal Register Machine

Construct a complete URM, without using the macros from lecture. You might want to write a small compiler that translates the version using macros into a real register machine.

Try to make the resulting machine as small as possible and demonstrate its correctness by simulating a few small register machines.

- One Instruction Language

Universal computers can be obtained in many ways. One possible approach is a “one instruction language (OIL).” Show that the classical OIL is computationally complete. Write an interpreter for OIL programs and construct some interesting examples.

<http://www.cs.cmu.edu/~cdm/Projects/oneil.pdf>

- A Small “Universal” Turing Machine

There is a claim that a 2-state/3-symbols Turing machine can already be universal, see http://en.wikipedia.org/wiki/Wolfram%27s_2-state_3-symbol_Turing_machine, <http://www.wolframscience.com/prizes/tm23/index.html> and <http://www.cs.cmu.edu/~cdm/Projects/TM23Proof.pdf>. The alleged proof by Alex Smith has lots and lots of holes, see for example the objections raised at FOM linked at the wiki site.

Determine whether Smith’s argument holds water.

- Divisibility and Minimal DFAs

Determine the size of the minimal DFA recognizing all multiples of a fixed modulus m in various numeration systems. This has been done for normal radix notation and reverse radix.

<http://www.cs.cmu.edu/~cdm/Projects/divisibility.pdf>

- Regular Polyhedra

We have used the Dihedral groups D_n to describe motions of regular n -gons in the plane. In 3-dimensional Euclidean space there are only 5 regular polyhedra and their corresponding symmetry groups. Describe these symmetry groups in as intuitive a way as ever possible – in this case, pictures are better than algebra.

- Polya-Redfield Extensions

We have seen that Polya-Redfield can be used to count patterns produced by groups acting on a space of objects. However, the theory does not directly address the issue of generating patterns (one representative for each). Describe a corresponding generalization of the Polya-Redfield machinery and implement the method. Interesting would be in particular the problems alluded to by Halbersma.

<http://www.cs.cmu.edu/~cdm/Projects/Halbersma.txt>

- Cycle Finding Algorithms

Floyd's algorithm is just one example of a memoryless cycle finding algorithm. Implement a number of these algorithms and compare their performance. A good test case are elementary cellular automata, see <http://www.cs.cmu.edu/~cdm/notes.html#CA>: try to compute transients and periods for n -bit configurations for n as large as possible.

- Testing for k -Cycles

We have seen that first order properties of structures given “by a finite state machine” can be decided automatically. Consider the concrete case of elementary cellular automata operating on bit vectors of length n (assuming periodic boundary conditions). Implement an algorithm that, given the ECA number r , $0 \leq r < 256$ and $k \geq 1$, determines for which n the ECA has a k cycle.

For ECAs see <http://www.cs.cmu.edu/~cdm/Projects/ca-introduction.pdf> Note that k -cycle means that there are in fact k distinct configurations on the cycle.

- Implementing Safra

Write a decent implementation of Safra's determinization algorithm in C/C++. The file

<http://www.cs.cmu.edu/~cdm/Projects/BuechiExpl.tgz>

contains a few test cases (the format is self-explanatory). Note that the “monster” machines get quite large. Efficiency is quite an issue here, think carefully about the underlying data structures before you start to hack.

- Ehrenfeucht-Mycielsky Sequence

Describe the sequence and some of its apparent properties. Explain in particular the mechanisms that make it likely that the sequence is balanced.

<http://www.cs.cmu.edu/~cdm/Projects/em-sequence.pdf>

- Primality Testing in Polynomial Time

A celebrated result by Agarwal-Kayal-Saxena shows that one can check in polynomial time whether a number is prime. The algorithm uses arithmetic in a polynomial ring. Explain the algorithm and its running time.

There is a lot of information on the web, just google.

Some of these projects can be handled essentially without recourse to the literature, others involve some reading. I can supply some material but you will also have to use the web/library.

For those projects involving programming, it is crucial that you produce well-documented, maintainable code. Spaghetti and hacks are not acceptable.

There is no need to prove new results – though of course it's fantastic if you do come up with something new. You are expected to demonstrate a clear understanding of the issues and the ability to describe the problem and its solution.