Particle Logic 2.0

Aaron T. Becker

***Abstract*— This introduces the first *pizza problem*, an challenge of unknown complexity that is accessible to all lab members. A successful completion of one part of this problem entitles the solver and a guest to a free tutorial and meal at Prof. Becker's home – learn how to make hand-tossed pizza!**

In previous work [1], [2], [3], [4], our lab showed the computational universality of swarms of micro- or nano-scale robots in complex environments, controlled not by individual navigation, but by a uniform global, external force. Consider a 2D grid world, in which all obstacles and robots are unit squares, and for each actuation, robots move maximally until they collide with an obstacle or another robot.

We demonstrated components of *particle computation* in this world, designing obstacle configurations that implement AND and OR logic gates: by using dual-rail logic, we designed NOT, NOR, NAND, XOR, XNOR logic gates. With the help of 2×1 robots we designed FAN-OUT gates, which is necessary for simulating the full range of complex interactions that are present in arbitrary digital circuits. Using these gates we are able to establish the full range of computational universality as presented by complex digital circuits. As an example we connect our logic elements to produce a 3-bit counter. We also demonstrated how to implement a data storage element. These techniques were implemented on the large prototype in Fig. 2. MATLAB code for generating and actuating logic gates and other particle-computation gadgets are found at github.com/aabecker/particleComputation/.../gadgets.m.

The following are open projects for our lab, which we describe as Particle Computation 2.0:

1) The current memory element, shown in Fig. 3 is not good enough for a journal submission. It is not conservative. In a *conservative* gate, particles are neither created nor destroyed, and so the same number of particles enter and leave each gate. This gate uses a particle to set the gate to true. When the gate is cleared, if the gate was true this particle exits, but if it was false, no particle exits. The new version should represent state (true or false) by the position of a 2×1 slider. This new gate may require more than 1 CW cycle.
2) Currently, all gates function after one clock-

Fig. 1. This could be you! Just complete a problem segment.

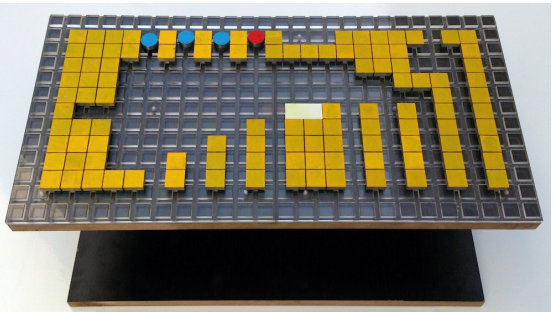

Fig. 2. Gravity-fed hardware implementation of particle computation. The reconfigurable prototype is setup as a FAN-OUT gate using a 2×1 robot (white). This paper proves that such a gate is impossible using only 1×1 robots. See the demonstrations in the video attachment http://youtu.be/EJSv8ny31r8.

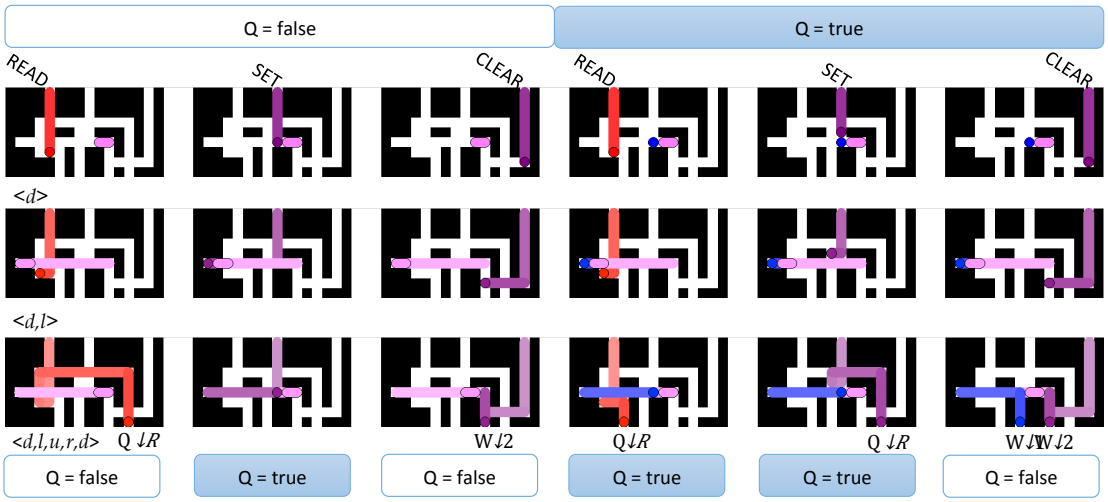

Fig. 3. A flip-flop memory in [4]. This device has three inputs, *Read*, *Set*, *Clear*, a state variable (shown in blue), and a 2×1 slider. Depending on which input is active, the *clockwise* control sequence $\langle d, l, u, r \rangle$ will read, set, or clear the memory. This example runs by uncommenting line 89 `[G.obstacle_pos,RobotPts] = memorycw; %memory`, located at github.com/.../matlab/gadgets.m We want to make a similar gate that is conservative.

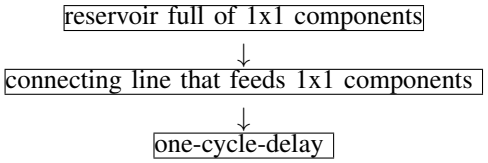

Fig. 4. a collection of obstacles and 2x1 sliders that work as a one-cycle delay

wise cycle $\langle d, l, u, r \rangle$. To use these gadgets as an automated factory a collection of obstacles and 2x1 sliders that work as a one-cycle delay:

OUTPUT: every other CW cycle a 1x1 block comes out. Currently we get a new component every CW cycle. We'd like to get one every other cycle, or concatenate these to get a 1x1 output every n cycles.

3) *Preferential Assembly:* Add tiny magnets to some of the edges of the sliders so that sliders preferentially assemble if two sliders with mating arrangements hit each other.
   - how many *species* or *slider varieties* can we construct?
   - our round sliders rotate, and don't have preferential orientations. We could weight one side of the sliders. Could octagonal sliders work in our present system? Would they rotate?
4) Develop theory showing that we can use a CW cycle and deliver a polyomino part to a destination, and then with another CW cycle hit this part at an arbitrary location with an additional 1x1 part. I want to show that we can build arbitrary shaped polyominoes, or prove that certain shapes cannot be constructed. For instance, using our sliders with magnets, can we construct an 'o'-shaped arrangement using eight 1x1 sliders?
   What is the minimum number of species required? Can we accomplish generalized assembly with just North-pole-out and South-pole-out sliders?
5) read about Sándor's aTAM theory and design components that could be assembled using this technique.

References

[1] A. Becker, E. Demaine, S. Fekete, G. Habibi, and J. McLurkin, "Reconfiguring massive particle swarms with

limited, global control," in *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, Sep. 2013.

[2] A. Becker, E. Demaine, S. Fekete, and J. McLurkin, "Particle computation: Controlling robot swarms with only global signals," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[3] A. Becker, E. D. Demaine, S. P. Fekete, G. Habibi, and J. McLurkin, "Reconfiguring massive particle swarms with limited, global control," in *Algorithms for Sensor Systems*, ser. Lecture Notes in Computer Science, P. Flocchini, J. Gao, E. Kranakis, and F. Meyer auf der Heide, Eds. Springer Berlin Heidelberg, 2014, pp. 51–66. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45346-5_5

[4] H. M. Shad, R. Morris-Wright, E. D. Demaine, S. P. Fekete, and A. T. Becker, "Particle computation: Device fan-out and binary memory," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015 (under review). [Online]. Available: https://github.com/aabecker/particleComputation/blob/master/fanout/FanOutgatesAndBinaryCounters.pdf