Dissertation Synopsis

While many aspects of the human brain remain a mystery, researchers have made many attempts to theoretically model the networks of neurons found within the brain. Some researchers try to model the brain as it physically exists in the human body, while others try to utilize particular processes within the brain, such as memory. Our research group studies a new form of computation inspired by the way the brain uses memory to process information. We are physicists who look at the brain for motivation, but we do not study it. Instead, we design and simulate special circuits known as digital memcomputing machines, where "mem" stands for memory (see Fig. 1). Research of these circuits is also of interest to many other academic fields, including computer science, engineering, and mathematics.

We do not actually build digital memcomputing machines. Instead, we use computer programs to simulate their behavior. The simulation is a very useful tool, because the final state of the simulated circuit is the solution to a very difficult type of problem. These problems, known as Boolean satisfiability problems, are very similar to the logic problems (true or false) many people learn in high school. An efficient solution method to these problems is sought after in real-world applications, for example, planning, scheduling, software design, and cryptography. Our research goal is to apply memcomputing techniques to Boolean satisfiability problems to solve them very quickly.

To understand our memcomputing circuits, let us first discuss a traditional logic circuit. Logic circuits are designed to interpret electrical signals (voltages) as true or false values. Those electrical signals are input into a logic gate that enforces some form of logical operation (AND, OR, etc.). At the top of Fig. 2 is an example of a 3-input OR gate with one output. Consider an example of its logic. There is a family of four individuals: Alice, Benny, and Charles are siblings (A, B, C), and Danielle is their mother (D). If Alice or Benny or Charles has a child, then their mother Danielle is a grandmother. There are seven possible ways for Danielle to be a grandmother, because the OR operators allows for more than one sibling to have a child. There is only one possible way for Danielle not to be a grandmother, and that occurs only if Alice, Benny, and Charles do not have children. In Fig. 2, we use this example to visualize the standard OR logic with a modified truth table.

Our digital memcomputing machine is imagined as a circuit of *self-organizing* OR gates. These special OR gates are called self-organizing because they no longer require the "one-way arrow" of the standard OR logic. This can be a bit confusing, so let's revisit the example above. We needed to know whether or not Alice, Benny, and Charles had children to determine if Danielle is a grandmother. This is the one-way arrow: we need information about Alice, Benny, and Charles to gain information about Danielle. What if we assumed information about Danielle? If we assume Danielle is a grandmother, then we know at least one of her siblings must have had children. (Remember there are seven possible ways for this to occur.) Our self-organizing OR gates have a "two-way arrow"

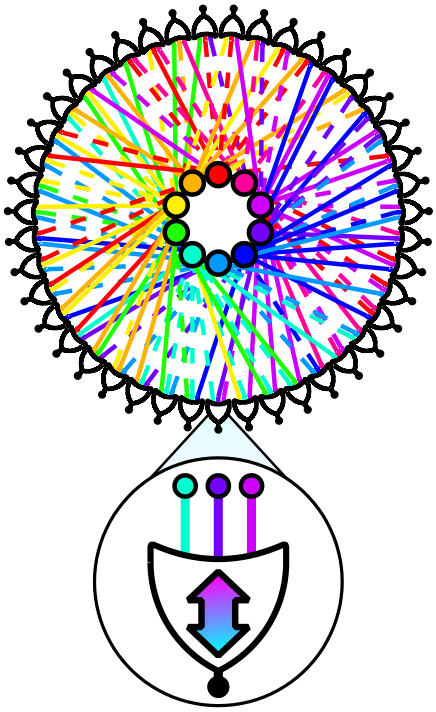


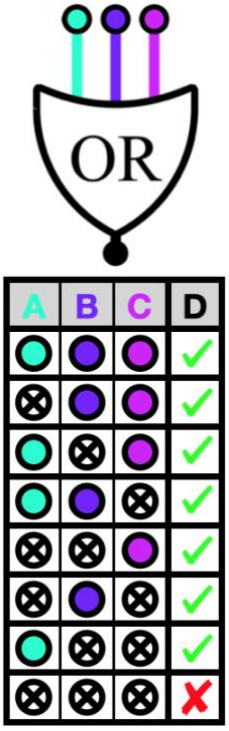
Figure 1: A digital memcomputing machine is a circuit (top) of self-organizing logic gates (bottom). On the outside of the circular circuit are 43 self-organizing OR gates. The 10 colorful circles on the inside of the circuit are where we read the electrical signals. Three electrical signal inputs are seen at top of the self-organizing OR gate (color circles), with an output signal below (black circle).

in their OR logic: We demand the output of the OR gate to be true, and the gate "self-organizes" its input signals to one of seven logically consistent states.

We connect many self-organizing logic gates that share similar input signals creating a self-organizing logic circuit that searches for a logically consistent state. When we design the circuit by connecting the input signals, we encode a Boolean satisfiability problem. Then, we simulate this circuit on a computer, and the logically consistent final state of the self-organizing logic circuit will encode the solution to the associated Boolean satisfiability problem.

Now that we understand the fundamental elements of digital memcomputing machines, let's review the results of my dissertation research. It was previously observed by our research group that the efficiency of digital memcomputing machines is caused by *collective behavior* in the circuits. Collective behavior is observed when multiple logic gates in the circuit *simultaneously* find a logically consistent state. We wondered if there was observable collective behavior at the more fundamental level of the self-organizing OR gates. Our simulations confirmed that there is indeed collective behavior within a single self-organizing OR gate.

In the next stage of research, we investigated the collective behavior within an entire circuit while solving Boolean satisfiability problems. We looked for what are known as avalanches in the voltage dynamics of the circuit. These avalanches borrow their name from the familiar avalanches seen on snowy mountain tops, where the snow at rest on top will suddenly slide down the mountainside. Analogous avalanche behavior was observed within the circuit, characterized by collective changes in the voltage dynamics (think multiple big piles of snow suddenly sliding down the mountainside). We predicted the size and frequency of the avalanches within the self-organizing logic circuits would indicate a special phenomenon. Indeed, we observed what is known as a *critical branching process* in our digital memcomputing machines. The presence of this process in the circuit results in avalanches of all sizes pervading the system and guiding it to equilibrium (solution). To use our snowy mountain analogy, consider this an event where many avalanches happen across the mountain until all snow has slid down the sides.



Once we confirmed the existence of critical branching processes in memcomputing machines, we wanted to apply our technique to very difficult instances of Boolean satisfiability problems. We compared the results to standard solution methods and found that the memcomputing simulations outperformed the best known methods. Strikingly, we found our memcomputing simulation could solve difficult problems in hours, where standard methods are predicted to take longer than the age of the Universe!

We intend to continue applying our memcomputing technique to many difficult problems of interest to multiple academic fields and industrial applications. One can research the efficiency of the memcomputing technique, or one could utilize the solution method to investigate an entirely unrelated research topic. An efficient solution method for Boolean satisfiability problems becomes a powerful research tool for investigating other topics in mathematics, physics, and computer science. Thus, memcomputing has the potential to further research in many scientific pursuits.

Figure 2: (Top) Standard OR gate, where the top three colored circles are inputs (A,B,C), and the bottom black circle is an output (D). (Bottom) A modified truth table, where a solid circle means a sibling (A,B,C) has children, and a circled X means the sibling has no children. Their mother (D) is a grandmother if a green checkmark appears in her column. In the bottom row we see if A and B and C have no children, then D is not a grandmother (red X). The self-organizing OR gates in Fig. 1 have the "D is not a mother" state removed from their truth table.