The computational complexity of small universal Turing machines

Damien Woods                  Turlough Neary

Caltech, United States        Univ. Zurich & ETH Zurich, Switzerland

Isaac Newton Institute for Mathematical Sciences
Cambridge, 2012
Motivations

Some surprisingly simple processes are computationally powerful

Aim 1: To find the border between decidability & undecidability of problems related to program size
  - For example, by finding small universal Turing machines or showing that none exist

Aim 2: To categorise trade-offs between universal-program size versus time, space and encoding complexity
  - Application computational complexity tools to simple models of computation such as small universal Turing machines, tag systems and cellular automata
“An interesting unsolved problem is to find the minimum possible state-symbol product for a universal Turing machine”

Claude Shannon, 1956
Small universal Turing machines

- E.F. Moore ('52) & C. Shannon ('56)
- Ikeno ('58), Watanabe ('60, '61, '72), ...
- Minsky ('60, '61, '62), Rogozhin ('79/'82, '96), Kudelk, Baiocchi, Robinson gave small universal Turing machines
  - Simulated 2-tag systems
- 2-tag systems proven universal by Cocke & Minsky ('63/'64)
- However, their simulation was exponentially slow

Cocke/Minsky

\[ \text{Turing machine} \leftrightarrow^\text{exp} \text{2-tag system} \leftrightarrow \text{small UTM} \]

- Hasenjaeger’s physical Turing machine: 1960’s

Figure
What is going on here?

ONE DAY I STARTED LAYING DOWN ROWS OF ROCKS.

SURE, IT’S ROCKS INSTEAD OF ELECTRICITY, BUT IT’S THE SAME THING, JUST SLOWER.

BUT I HAVE INFINITE TIME AND SPACE.

THE EONS BLUR PAST AS I WALK DOWN A SINGLE ROW.

SO IF YOU SEE A MOTE OF DUST VANISH FROM YOUR VISION IN A LITTLE FLASH OR SOMETHING

EACH NEW ROW FOLLOWED FROM THE LAST IN A SIMPLE PATTERN.

AFTER A WHILE, I PROGRAMMED IT TO BE A PHYSICS SIMULATOR.

THE ROWS BLUR PAST TO COMPUTE A SINGLE STEP.

SO I DECIDED TO SIMULATE A UNIVERSE.

AND IN THE SIMULATION

I’M SORRY. I MUST HAVE MISPLACED A ROCK

WITH EVERY PIECE OF INFORMATION ABOUT A PARTICLE, IT WAS ENCODED AS A STRING OF BITS WRITTEN IN THE STONES.

WITH ENOUGH TIME AND SPACE, I COULD FULLY SIMULATE TWO PARTICLES INTERACTING.

WITH THE RIGHT SET OF RULES AND ENOUGH SPACE, I WAS ABLE TO BUILD A COMPUTER.

EACH NEW ROW OF STONES IS THE NEXT ITERATION OF THE COMPUTATION.

Randall Munroe. xkcd, # 505.
Cyclic tag systems were used by Cook (2004) to show that rule 110 is universal:

- Turing machine $\mapsto_{\exp}^\exp$ 2-tag system
- $\mapsto$ cyclic tag system $\mapsto$ rule 110
Result: Cyclic tag systems are efficiently universal

Theorem (Neary, Woods. ICALP 2006)

Let $M$ be a single-tape deterministic Turing machine that computes in time $t$. Then there is a cyclic tag system $C_M$ that simulates the computation of $M$ in time $O(t^3 \log t)$.

Thus rule 110 efficiently simulates TMs:

$$
\text{TM} \leftrightarrow \text{cyclic tag system} \leftrightarrow \text{rule 110}
$$
What is going on here?

Efficient computation. Prediction is P-complete (presumably requires explicit simulation).
Rule 110 - Not so slow!

One day I started laying down rows of rocks.

Each new row followed from the last in a simple pattern.

With the right set of rules and enough space,
I was able to build a computer.

But I have infinite time and space.

The eons blur past as I walk down a single row.

The rows blur past to compute a single step.

And in the simulation

I'm sorry, I must have misplaced a rock

Sure, it's rocks instead of electricity, but it's the same thing, just slower.

After a while, I programmed it to be a physics simulator.

Every piece of information about a particle was encoded as a string of bits written in the stones.

With enough time and space, I could fully simulate two particles interacting.

So I decided to simulate a universe.

But if you see a mote of dust vanish from your vision in a little flash or something

Sometime in the last few billions and billions of millennia.
Efficient computation (improves on an xkcd comic)
Prediction is P-complete
Open: Is Rule 110 intrinsically universal?
Gave some evidence that extremely \textit{simple} universal programs can compute efficiently
We gave small UTMs that simulate TMs with an $O(t^2)$ overhead [NW, TCS 2006]

- Direct simulation
- Not the smallest UTMs, but the fastest so far

Figure
What’s next?

Are the smallest universal machines \textit{fast}?
UTMs run in $O(2^{2t} t^2)$ time via Cocke/Minsky

Turing machine $\leftrightarrow^{exp}$ 2-tag system $\leftrightarrow$ small UTM
2-tag system

Σ, a finite alphabet of symbols

\( R : \Sigma \rightarrow \Sigma^* \), a finite set of rules

Configuration/dataword:

\[ \sigma_1 \sigma_2 \sigma_3 \ldots \sigma_\ell \in \Sigma^* \]

Computation step:

\[ \sigma_1 \sigma_2 \sigma_3 \ldots \sigma_\ell \]

\[ \vdash \sigma_3 \ldots \sigma_\ell \sigma_{\ell+1} \ldots \sigma_{\ell+c} \]

if there is a rule of the form \( \sigma_1 \rightarrow \sigma_{\ell+1} \ldots \sigma_{\ell+c} \)
2-tag system

\[ \Sigma, \text{ a finite alphabet of symbols} \]
\[ R : \Sigma \rightarrow \Sigma^*, \text{ a finite set of rules} \]

Configuration/dataword:

\[ \sigma_1\sigma_2\sigma_3 \ldots \sigma_\ell \in \Sigma^* \]

Computation step:

\[ \sigma_1\sigma_2\sigma_3 \ldots \sigma_\ell \]
\[ \vdash \sigma_3 \ldots \sigma_\ell \sigma_{\ell+1} \ldots \sigma_{\ell+c} \]

if there is a rule of the form \( \sigma_1 \rightarrow \sigma_{\ell+1} \ldots \sigma_{\ell+c} \)
2-tag system: example

Example

\[ \Sigma = \{x, \dot{x}, \bar{x}, \dot{\bar{x}}, y, O, E\} \]

\[ R = \{\bar{x} \rightarrow yO, x \rightarrow \varepsilon, y \rightarrow E\} \]

\[ \begin{array}{l}
\bar{x}\dot{x} x\dot{x} x\dot{x} x\dot{x} x\dot{x} \\
\vdash x\dot{x} x\dot{x} x\dot{x} x\dot{x} yO \\
\vdash x\dot{x} x\dot{x} x\dot{x} yO \\
\vdash x\dot{x} x\dot{x} yO \\
\vdash x\dot{x} yO \\
\vdash yO \\
\vdash E
\end{array} \]

even length \Rightarrow \text{output } E \quad \text{odd length } \Rightarrow \text{output } O

We can detect odd/even length without destroying the dataword.
2-tag system: example

Example

Σ = \{x, \dot{x}, \bar{x}, \dot{\bar{x}}, x, \dot{x}, \bar{x}, \dot{\bar{x}}, x_1, \dot{x}_1, \bar{x}_1, \dot{\bar{x}}_1\} \quad R = \{\bar{x} \rightarrow \bar{x}_1 \dot{x}_1, x \rightarrow x_1 \dot{x}_1\}

\[
\begin{align*}
\bar{x} \dot{x} x \dot{x} x \dot{x} x \dot{x} x \\
\vdash x \dot{x} x \dot{x} x \dot{x} x \dot{x} \dot{x}_1 \\
\vdash x \dot{x} x \dot{x} x \dot{x} \dot{x}_1 \dot{x}_1 \\
\vdash x \dot{x} x \dot{x} \dot{x} \dot{x}_1 \dot{x}_1 \dot{x}_1 \\
\vdash x \dot{x} \dot{x} \dot{x}_1 \dot{x}_1 \dot{x}_1 \dot{x}_1 \dot{x}_1 \\
\vdash \bar{x} \dot{x} x \dot{x} x \dot{x} x \dot{x} x \dot{x} x
\end{align*}
\]

even length ⇒ halt reading \( \bar{x}_1 \) \quad odd length ⇒ halt reading \( \dot{x}_1 \)

We can detect odd/even length without destroying the dataword. Cocke & Minsky used this, with a unary encoding and modular arithmetic, to prove universality.
Fast *and* small?

**Before:** $O(2^{2t} t^2)$ via Cocke/Minsky

- Turing machine $\mapsto^{\exp} 2$-tag system $\mapsto$ small UTM

2-tag systems are efficiently universal [WN, FOCS 2006]:

**After:** $O(t^8 \log^4 t)$ for small UTMs

- Turing machine $\mapsto$ cyclic tag system $\mapsto$ 2-tag system $\mapsto$ small UTM

Further improvement (Neary, PhD thesis):

$O(t^4 \log^2 t)$ for small UTMs

- Turing machine $\mapsto$ 2-tag system $\mapsto$ small UTM

Figure
Fast *and* small?

**Before:** $O(2^{2t}t^2)$ via Cocke/Minsky

Turing machine $\xrightarrow{\text{exp}}$ 2-tag system $\xrightarrow{}$ small UTM

2-tag systems are efficiently universal [WN, FOCS 2006]:

**After:** $O(t^8 \log^4 t)$ for small UTMs

Turing machine $\xrightarrow{}$ cyclic tag system

$\xrightarrow{}$ 2-tag system $\xrightarrow{}$ small UTM

Further improvement (Neary, PhD thesis):

$O(t^4 \log^2 t)$ for small UTMs

Turing machine $\xrightarrow{}$ 2-tag system $\xrightarrow{}$ small UTM

Figure
Apart from Turing machines, rule 110, 2-tag systems, cyclic tag systems, we show that the following models are efficiently universal (exp. speedup):

- Rothemund’s DNA Turing machine & tag-system (’96)
- Siegelmann and Margenstern’s small neural network (’99)
- Rogozhin and Verlan’s small tissue P system (’05)
- Harju and Margenstern’s H-system (’05)
- Lindgren and Nordahl’s cellular automata (’90)
- Margenstern's non-erasing machines (’93, ’95)
- Robinson’s tiling (’71)

...
Hasenjaeger’s machine

On display here at the Isaac Newton Institute for Mathematical Sciences, Cambridge
Universal machine with 4 states, 2 symbols, 3 tapes
A “non-erasing” Turing machine
Reverse engineered by Rainer Glaschick
Hasenjaeger’s machine: is it fast or slow?

Universal machine with 4 states, 2 symbols, 3 tapes
A “non-erasing” Turing machine

- Hasenjaeger simulates Hao Wang’s model (in linear time)
  - ←, →, mark tape, Jump on mark
- Wang proved his model universal in 1957; via an exponentially slow simulation of Turing machines
- Here we showed Wang’s model can efficiently simulate non-erasing Turing machines, which in turn can efficiently simulate Turing machines
- The proof did not go via our previous speed-up techniques (tag systems)
- Hasenjaeger machine simulates Turing machines in $O(t^3)$ time, $O(t^2)$ space
Hasenjaeger’s machine: is it fast or slow?

Universal machine with 4 states, 2 symbols, 3 tapes
A “non-erasing” Turing machine

- Hasenjaeger simulates Hao Wang’s model (in linear time)
  - ←, →, mark tape, Jump on mark
- Wang proved his model universal in 1957; via an exponentially slow simulation of Turing machines
- Here we showed Wang’s model can efficiently simulate non-erasing Turing machines, which in turn can efficiently simulate Turing machines
- The proof did not go via our previous speed-up techniques (tag systems)
- Hasenjaeger machine simulates Turing machines in $O(t^3)$ time, $O(t^2)$ space
Hasenjaeger’s machine: is it fast or slow?

Universal machine with 4 states, 2 symbols, 3 tapes
A “non-erasing” Turing machine

- Hasenjaeger simulates Hao Wang’s model (in linear time)
  - ←, →, mark tape, Jump on mark
- Wang proved his model universal in 1957; via an exponentially slow simulation of Turing machines
- Here we showed Wang’s model can efficiently simulate non-erasing Turing machines, which in turn can efficiently simulate Turing machines
- The proof did not go via our previous speed-up techniques (tag systems)
- Hasenjaeger machine simulates Turing machines in $O(t^3)$ time, $O(t^2)$ space
Conclusions: Computational complexity

- “Simple” universal systems (UTMs, rule 110, 2-tag systems) are not necessarily exponentially slow, in fact most are efficient.
- Exponential improvement on the simulation overhead for other universal models of computation.
- Predicting these simple systems is P-complete.
- Polynomial trade-off? (Tight upper and lower bounds?)
- Is Rule 110 intrinsically universal?
- Explore encoding and decoding issues in a formal way?
- The small UTM field was initially concerned with universality/decidability questions, however by peeking through the lens of complexity theory we find a more fine grained structure and new ideas of independent interest.
Acknowledgements

- Rainer Glaschick, Niall Murphy (UCM-UPM)
- Matthew Cook (ETH), Erik Winfree (Caltech), Shuki Bruck (Caltech), and their research groups
- Science Foundation Ireland
- Irish Research Council for Science, Engineering & Technology
- Junta de Andalucía, Spain
- The Molecular Programming Project at Caltech, National Science Foundation, United States