ITT8040 — Cellular Automata
Lecture 7

Silvio Capobianco

Institute of Cybernetics at TUT

April 24, 2013

Turing machines

A (deterministic) Turing machine is specified by:

- a finite set of states Q;
- three special states $q_I, q_A, q_R \in Q$ called the initial, accepting, and rejecting state;
- a finite tape alphabet Γ;
- a finite input alphabet $\Sigma \subset \Gamma$;
- a blank tape symbol $b \in \Gamma \setminus \Sigma$; and
- a transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$.

The transition function specifies the behavior of a read-write head on a bi-infinite tape, in the following way:

if you are in current state and read current symbol,
then take next state, write next symbol, and move by one block

We require $\delta(q_A, \gamma) = (q_A, \gamma, +1)$ and $\delta(q_R, \gamma) = (q_R, \gamma, +1)$.

Turing machines (cont.)

An instantaneous description of a configuration of a Turing machine is a triple $(q, i, t) \in Q \times \mathbb{Z} \times \Gamma^{\mathbb{Z}}$ where:

- q is the current state
- i is the current position of the head
- t is the global state of the tape

The next configuration (q', i', t') is defined straightforwardly:

$$\text{if } \delta(q, t(i)) = (q', \gamma, x)$$
$$\text{then } i' = i + x,\ t'(i) = \gamma,\ t'(j) = t(j) \text{ for } i \neq j$$

If t_w is the tape with w in positions 1 to $|w|$ and blank elsewhere, then the machine accepts w if $(q_I, 1, t_w) \to^* (q_A, i, t)$.
It rejects w if either $(q_I, 1, t_w) \to^* (q_R, i, t)$ or never halts.

Turing machines as a formal model of semi-algorithms

We say that a Turing machine T and a semi-algorithm S are equivalent if they recognize the same language, that is, for every input x:

- T accepts x if and only if S returns "yes" on x; and
- T rejects x if and only if S returns "no" on x or does not halt on x.

Then there is an algorithm that, given an arbitrary Turing machine T, constructs an equivalent semi-algorithm S.

Theorem: There exists an algorithm that, given an arbitrary semi-algorithm S, constructs an equivalent Turing machine T.

Turing's halting problem

Consider the following problem:

- given an arbitrary Turing machine T,
- determine whether or not T halts on the empty tape.

Theorem:
Turing's halting problem is semi-decidable but not decidable.

- Given A and w, construct the semi-algorithm:
$$\mathrm{B(u) = A(w)}$$
- Construct a Turing machine T equivalent to B.
- Then T halts on the empty tape if and only if A halts on w.
- This is a reduction of semi-algorithm halting to Turing's halting problem.
  As the former is undecidable, so is the latter.

Undecidable problems about tilings

Tiling problem (co-semi-decidable)

- given a finite tile set $\mathcal{T} = (T, N, R)$,
- determine whether or not $\mathcal{T}$ admits a valid tiling t.

Seeded tiling problem (co-semi-decidable)

- given $\mathcal{T}$ and a special seed tile $s \in T$,
- determine if there is a valid tiling t with $t(0,0) = s$.

Finite tiling problem (semi-decidable)

- given $\mathcal{T}$ and a special blank tile $B \in T$,
- determine if $\mathcal{T}$ admits a finite nontrivial valid tiling.

NW-deterministic tiling problem (co-semi-decidable)

- given a NW-deterministic tile set $\mathcal{T} = (T, N, R)$,
- determine whether or not $\mathcal{T}$ admits a valid tiling t.

Periodic tiling problem (semi-decidable)

- determine whether or not $\mathcal{T}$ admits a valid periodic tiling t.

Nilpotent cellular automata

A cellular automaton $A = (S, d, N, f)$ with quiescent state q and global function G is nilpotent if every configuration ultimately evolves into the quiescent configuration.

This is the same as satisfiying the following condition:
There exists $n \geq 1$ such that the n-th iteration G^n sends every configuration into the quiescent configuration.

- Let c be a rich configuration containing every possible pattern.
- Then G^n makes every configuration quiescent if and only if it makes q quiescent.

Nilpotency is thus semi-decidable.

Nilpotency of 2D cellular automata is undecidable

Let T be a finite set of Wang tiles.

- Set $S = T \sqcup \{q\}$.
- Let f leave the middle tile unchanged if the tiling is correct, and turn it to q otherwise.
- This CA is nilpotent if and only if T does not admit a valid tiling.

We have reduced the tiling problem to (non-)nilpotency of 2D CA. As the former is undecidable, so is the latter.

Nilpotency of 1D cellular automata is undecidable

Let T be a NW-deterministic finite set of Wang tiles.

- Set $S = T \sqcup \{q\}$ and

$$f(x, y) = \begin{cases} z & \text{if } \begin{array}{|c|c|} \hline & y \\ \hline x & z \\ \hline \end{array} \text{ is a match}, \\ q & \text{otherwise}. \end{cases}$$

- This CA is nilpotent if and only if T does not admit a valid tiling.

We have reduced the NW-deterministic tiling problem to (non-)nilpotency of 1D CA.
As the former is undecidable, so is the latter.

Reversibility of 2D cellular automata is undecidable

Let T be a finite tile set.
Let D be a finite tile set with the plane filling property.

- Let $S = T \times D \times \{0, 1\}$.
- Define the local update rule as follows:
  - if both tiling components are valid then XOR the bit component with that of the follower;
  - otherwise do nothing
- The resulting CA is reversible if and only if T does not admit a valid tiling.

This reduces the tiling problem to 2D CA (non-)reversibility.

A special tile set

Consider the tile set in Figure 30.
The only finite paths that are portions of valid tilings, are the rectangular loops such as the one in Figure 31.

- Suppose t contains such a path.
- Then it must contain at least one of the nontrivial tiles.
- But whatever that one tile is, the only way to have a finite valid tiling containing that tile, is to build a rectangular loop with a "cross" in the middle.

Surjectivity of 2D cellular automata is undecidable

Let $\mathcal{T} = (T, N, R)$ be a finite Wang tile set with blank symbol B. Consider the tile set D in Figure 30.

- Let S be the set of triples $(d, t, x) \in D \times T \times \{0, 1\}$ such that:
  - if $d = c$, the cross tile, then $t \neq B$; and
  - if $d = b$, the blank tile, or d contains a label, then $t = B$.
- Define the local update rule as follows:
  - if both tiling components are valid then XOR the bit component with that of the follower;
  - otherwise do nothing
- The resulting CA is surjective if and only if T does not admit a valid finite tiling.

This reduces the finite tiling problem to 2D CA (non-)surjectivity.

R.e.-completeness

A problem C is r.e.-complete if

- it is semi-decidable, and
- every semi-decidable problem P is many-to-one reducible to C.

Semi-algorithm halting is r.e.-complete.

- Let P be a semi-decidable problem.
- As P is semi-decidable, there exists a semi-algorithm S that halts and returns "yes" for every "yes"-instance of P.
- But S can easily be modified so that it only halts on "yes"-instances.
- Then x is a "yes"-instance of P if and only if S halts on x.

R.e.-complete problems

The following problems are r.e.-complete:

- Turing's halting problem
- Finite tiling problem
- Periodic tiling problem
- CA nilpotency
- 2D CA reversibility

The complements of the following problems are r.e.-complete:

- Tiling problem
- Seeded tiling problem
- NW-deterministic tiling problem
- 2D CA surjectivity

Abstract systems

We call system a pair (X, F) where

- X is a space, and
- $F : X \to X$ is a function.

Usually, additional requirements are made, for example:

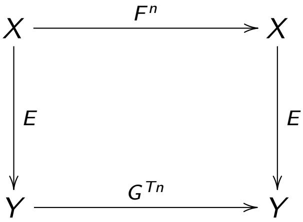
- X may be a topological space with specific properties (metric, compact, separable, etc.)
- F may show some kind of regularity (continuous, differentiable, computable, etc.)

Examples:

- Turing machines:
  X: tape configurations; F: update function.
- Cellular automata:
  X: d-dimensional configurations; F: CA global function.

Simulations

Let (X, F) and (Y, G) be two systems.
A (many-to-one) simulation in linear time $T \geq 1$ of (X, F) by (Y, G) is a function $E : X \to Y$ such that, for every $n \geq 0$, the following diagram commutes:



For $T = 1$ we say that (Y, G) simulates (X, F) in real time.

Simulating Turing machines with cellular automata

Suppose we are given a Turing machine M with:

- set of states Q,
- initial, accepting, and rejecting states $q_I, q_A, q_R \in Q$,
- tape alphabet Γ,
- input alphabet Σ,
- blank symbol b, and
- transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$.

Let F be the transformation of tapes determined by M.

Can we simulate the behavior of M with a cellular automaton?

Simulating Turing machines with cellular automata (cont.)

Let us construct a 1D CA as follows:

- The set of states is $S = (Q \sqcup \{0\}) \times \Gamma$, where 0 is a new state.
- The neighborhood is the radius-1 von Neumann neighborhood.
- The local update rule is defined as such:
  - If $y = (q, a)$ and $\delta(q, a) = (q', a', d')$, then $f(x, y, z) = (0, a')$.
  - If $x = (q, a)$, $y = (0, p)$ and $\delta(q, a) = (q', a', +1)$, then $f(x, y, z) = (q', p)$.
  - If $z = (q, a)$, $y = (0, p)$ and $\delta(q, a) = (q', a', -1)$, then $f(x, y, z) = (q', p)$.
  - In all other cases, $f(x, y, z) = y$.

Let G be the global function of this cellular automaton.

Simulating Turing machines with cellular automata (final)

Consider the systems (X, F) and (Y, G) where:

- X is the set of instantaneous descriptions of configurations of the Turing machine,
- F is the update rule of the Turing machine M,
- $Y = S^{\mathbb{Z}}$ with $S = (Q \sqcup \{0\}) \times \Gamma$, and
- G is the global function of the cellular automaton A constructed in the previous slide from the Turing machine M.

Define $E : X \to Y$ as follows:

$$E((q, i, t)) = c \;\; \text{with} \;\; c(i) = (q, t(i)) \;\; \text{and} \;\; c(j) = (0, t(j)) \; \forall j \neq i$$

Then E is a simulation, in real time, of the Turing machine M by the cellular automaton A.

A universal cellular automaton

There exists a 1D CA such that, for a given subset $F \subset S$ of states, the following problem is r.e.-complete:

$$\text{given a configuration } c, \\ \text{determine whether } G^n(c)(i) \in F \text{ for some } i \in \mathbb{Z},\ n \geq 0$$

- Let M be a Turing machine with a r.e.-complete language.
- Construct a CA that simulates M, as described in the previous slides.
- Set $F = \{(q_A, x) \mid x \in \Gamma\}$.

Other r.e.-complete problems for CA

For each of the following problems there exists a 1D CA for which the problem is r.e.-complete:

1. Given a finite configuration c,
   determine if c ultimately becomes quiescent.
2. Given a spreading state s and a finite configuration c,
   determine if s ultimately appears in c.
3. Given a finite configuration c,
   determine if it evolves into a fixed point.
4. Given two finite configurations c and e,
   determine whether c evolves into e.