

CDM

Cellular Automata and FSMs

Klaus Sutner

Carnegie Mellon University

www.cs.cmu.edu/~sutner

Fall 2003

Battleplan

- Reversibility
- The Language of a CA
- Surjectivity

Reversibility and Coding

Recall from last time that two-dimensional CA might be useful to encode images. For example, if we could try to use a reversible 2-dim CA for encoding, and its inverse for decoding.

Question: How hard is it to check if a CA is reversible?

More precisely, we want an algorithm to determine whether the global map

$$G_\rho : \mathcal{C} \rightarrow \mathcal{C}$$

is injective. In the infinite case \mathcal{C}_∞ the input is just the local map, in the finite case \mathcal{C}_n (or $\mathcal{C}_{n,m}$) we also have to know the size of the grid and the boundary conditions (always cyclic for us).

The infinite case is a bit easier, so let's start with that.

Gardens-of-Eden

A concept closely related to reversibility in the study of cellular automata is that of a *Garden-of-Eden (GoE)*:

A configuration X is a Garden-of-Eden if there is no predecessor configuration Y such that $G_\rho(Y) = X$.

In other words, there is a Garden-of-Eden if and only if the global map

$$G_\rho : \mathcal{C} \rightarrow \mathcal{C}$$

fails to be surjective.

The Infinite Case, Dimension 1

GoEs and reversibility would seem to be unrelated, but that is not so. We skip the proof of the next theorem.

Theorem 1. *A reversible cellular automaton has no Gardens-of-Eden.*

I.e., if the global map of a cellular automaton is injective then it is also surjective.

So we could test surjectivity first to filter out cases that cannot be reversible.

Dimension 1

We can think of a configuration of a one-dimensional infinite CA as a bi-infinite word $X : \mathbb{Z} \rightarrow \Sigma$.

But it suffices to consider finite blocks of these bi-infinite words.

Definition 1. The **cover** of a bi-infinite word X is the set of all finite factors. The cover of a set of words is the union of the covers of the individual words. In symbols: $\text{cov}(X)$.

The **language** of ρ is

$$\mathcal{L}(\rho) = \bigcup \{ \text{cov}(G_\rho(X)) \mid X \in \mathcal{C}_\infty \}.$$

Covers

More precisely, let us write

$$X[i, j] = X(i)X(i+1) \dots X(j-1)X(j)$$

for the block from i to j (where $i \leq j$, otherwise just ε) in bi-infinite word X .

Then

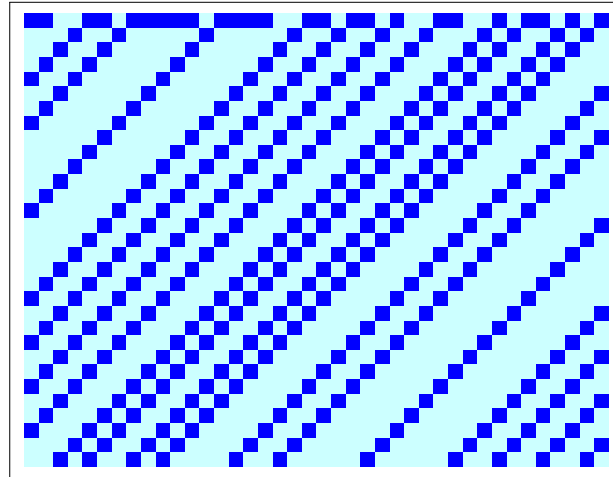
$$\mathcal{L}(\rho) = \{ G_\rho(X)[i, j] \mid X \in \Sigma^\infty \} \subseteq \Sigma^*$$

Covers are just sets of words, so finite state machines might help to study them.

ECA 34

ECA number 34 has table

000	→ 0	100	→ 0
001	→ 1	101	→ 1
010	→ 0	110	→ 0
011	→ 0	111	→ 0



$\mathcal{L}(\rho_{34})$: all words not containing 11. Note that only time $t = 1$ counts here.

The Language of a CA

What information can we get out of $\mathcal{L}(\rho)$?

Lemma 1. *The global map is surjective if and only if $\mathcal{L}(\rho) = \Sigma^*$.*

Proof.

Implication \rightarrow is clear. For the opposite direction let $Y \in \Sigma^\infty$.

Every finite block $Y[-n, n]$ is in $\mathcal{L}(\rho)$, so there is some X_n such that $\rho(X_n)[-n, n] = Y[-n, n]$.

Here is the crucial fact: there is a subsequence X_{n_i} such that $X = \lim_{i \rightarrow \infty} X_{n_i}$ exists.

It is not hard to see that $\rho(X) = Y$.

□

König's Lemma

But why should the limit exist? What does limit even mean?

Definition 2. A sequence X_n **converges** to X , $X = \lim_{n \rightarrow \infty} X_n$, if

$$\forall m \exists n_0 \forall n > n_0 (X[-m, m] = X_n[-m, m])$$

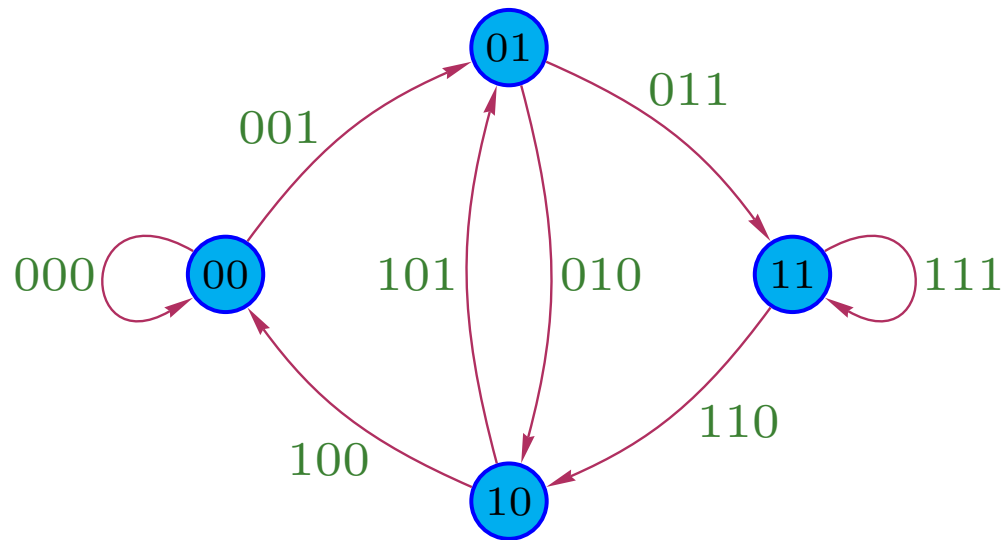
To show every sequence in Σ^∞ has a convergent subsequence one can use König's Lemma.

Lemma 2. *Every infinite but finitely branching tree has an infinite branch.*

De Bruijn Automata

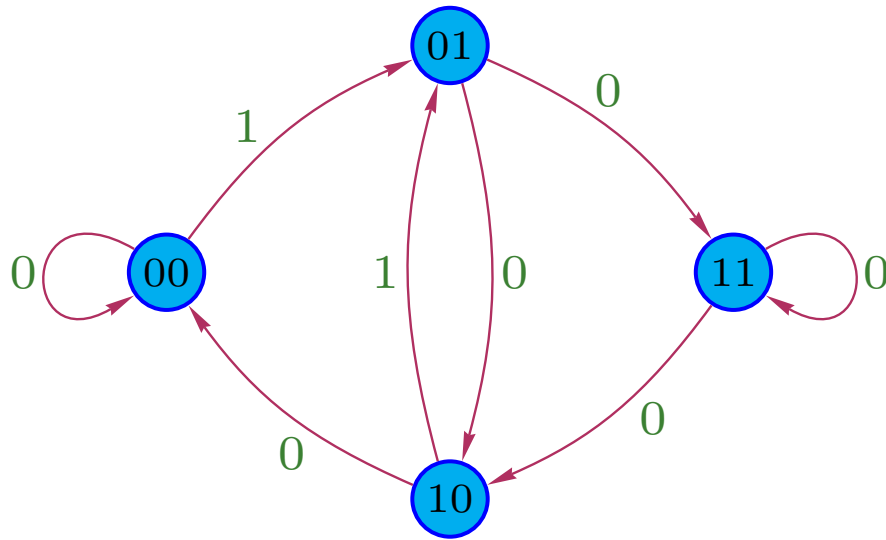
So we only have to check if $\mathcal{L}(\rho) = \Sigma^*$ to test surjectivity.

Can be build a finite state machine for $\mathcal{L}(\rho)$?



Relabel

Replace the edge labels xyz by $\rho(xyz)$.



Nondeterministic Automata

The resulting “automaton” is different from the DFAs we previously considered: it is *nondeterministic*.

- There are nondeterministic transitions (p, a, p') and (p, a, p'') .
- There are multiple initial states.

Definition 3. A **nondeterministic automaton (NFA)** is a directed graph whose edges are labeled by symbols in some alphabet Σ . Some nodes are declared initial and final.

A **semiautomaton** is a nondeterministic automaton where every state is considered initial and final.

Note that one edge may be labeled by multiple symbols.

Acceptance of NFAs

More precisely, a nondeterministic finite automaton is a structure

$$M = \langle Q, \Sigma, \tau, I, F \rangle$$

where Q is a finite set of states, Σ an alphabet, $I, F \subseteq Q$ are the initial and final states, respectively. $\tau \subseteq Q \times \Sigma \times Q$ is the transition relation.

Recall that DFAs were the embodiment of fast algorithms. How about NFAs?

Proposition 1. *One can check acceptance of a word x by an NFA in time $O(|x| |Q|^2)$.*

Relationship To DFAs

Theorem 2. Rabin-Scott

For every nondeterministic automaton there is an equivalent DFA: the DFA accepts the same language as the NFA.

Proof. Given a nondeterministic machine on states Q define the new machine on state set $Q' = \mathfrak{P}(Q)$.

The new transitions are given by

$$\delta'(P, a) = \{ q \mid (p, a, q) \in \tau, p \in P \}$$

Also $q'_0 = I$ and

$$F' = \{ P \subseteq Q \mid P \cap F \neq \emptyset \}$$

□

Application: Reversal Closure

Let

$$L^r = \{ x^r \mid x \in L \}$$

be the *reversal* of a language.

Theorem 3. *For any regular language L , the reversal L^r is also regular.*

Proof.

Let $M = \langle Q, \Sigma, \tau, I, F \rangle$ be an NFA for L .

Then

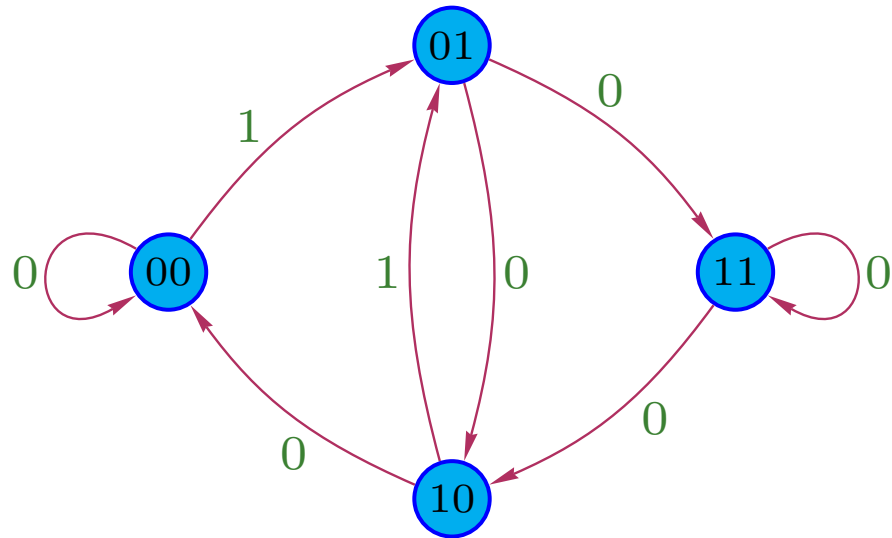
$$M^r = \langle Q, \Sigma, \tau^r, F, I \rangle$$

□

Note: some texts insist that in an NFA $|I| = 1$. Clearly, this is a bad idea.

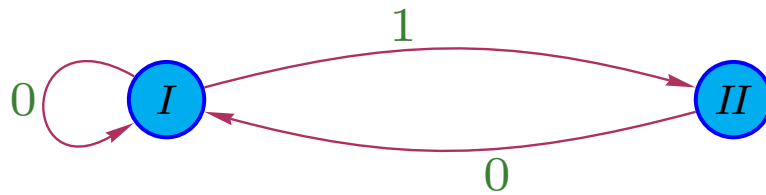
Back To De Bruijn Automata

Here is the semiautomaton for ECA 34 again.



Simplify

The last machine is clearly equivalent to semiautomaton



Acceptance language is all words without a factor 11.

Surjectivity Algorithm

So now we have an algorithm to test surjectivity of an infinite CA:

- Construct the de Bruijn automaton for ρ .
- Convert to a DFA.
- Minimize the DFA, and check if result has only 1 state.

Is correct, but no good: there may be exponential blow-up during conversion to DFA.

Surprisingly, there is a simple polynomial time algorithm based on the de Bruijn semiautomaton.

Ambiguity

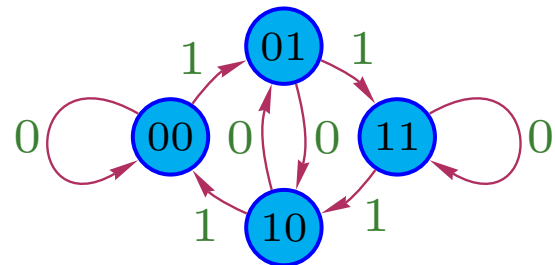
Definition 4. A semiautomaton is **ambiguous** if it has states p and q such that there are two distinct paths from p to q labeled by the same word.

Example:

The de Bruijn automaton for ECA 34 is ambiguous.

However, the simplified automaton is unambiguous.

The de Bruijn automaton for ECA 90 is unambiguous.



Surjectivity and Ambiguity

We won't go into the proof of the next result.

Lemma 3. *A one-dimensional infinite cellular automaton is surjective iff its de Bruijn automaton is unambiguous.*

How does one test ambiguity? Efficiently?

We need to keep track of pairs of paths in a semiautomaton \mathcal{A} .

Use a *product automaton* \mathcal{A}^2 (really just a product graph with edge labels).

$$(p, q) \xrightarrow{a} (p', q') \iff p \xrightarrow{a} p' \wedge q \xrightarrow{a} q'$$

Paths

Any path in \mathcal{A}^2 clearly corresponds to two paths with the same labels in \mathcal{A} .

How do we make sure the two paths start and end at the same point, but are not identical?

Start and end: must go from (p, p) to (q, q) .

But how about not identical?

We are interested only in de Bruijn automata: they are all strongly connected.

Path must leave the SCC containing the diagonal (p, p) , and return.

Surjectivity and SCCs

Proposition 2. *A de Bruijn automaton is ambiguous iff the strongly connected component of the product automaton containing the diagonal is larger than the diagonal.*

Can use Tarjan's algorithm to compute the SCCs in linear time.

Hence whole algorithm is quadratic in the size of the de Bruijn automaton, which is essentially the same as the size of the rule table for ρ .

Hence we can test surjectivity in quadratic time.

Injectivity

With a bit more work one can show that

Proposition 3. *A cellular automata ρ is injective iff the product automaton of the de Bruijn automaton for ρ contains no non-trivial strongly connected components other than the diagonal.*

As before, this result holds only for one-dimensional infinite CAs.

But we can test injectivity in quadratic time.

In fact, we can surjectivity and injectivity together in quadratic time: the only thing that changes is the check for the SCCs.

Higher Dimensions

The use of finite state machines to deal with dimension one does not carry over in any obvious way to higher dimensions (what is a 2-D word?).

Theorem 4. *Surjectivity and injectivity of two-dimensional infinite cellular automata is undecidable.*

Of course the same holds for higher dimensions.

So there is a huge gap in the complexity of one versus two-dimensional CAs.

The Finite Case

As one might suspect, the infinite case casts its shadows down into the finite world. Note, though, that since the configuration space is finite, injectivity and surjectivity are equivalent.

Theorem 5. *Surjectivity of two-dimensional finite cellular automata is NP -hard.*

This is exceedingly bad news for any encoding algorithm trying to use reversible CAs: we would have to construct the encoding CA specifically (rather than picking it at random) to make sure it is reversible – but that raises security issues.

It's also not all that easy.

Finite, Dimension One

Here things are more interesting.

We will only deal with cyclic boundary conditions. Needless to say, surjectivity of

$$\rho : \mathcal{C}_n \rightarrow \mathcal{C}_n$$

in general depends very much on n .

Example:

ECA 90 is never surjective on \mathcal{C}_n .

ECA 45 is surjective on \mathcal{C}_n iff n is odd.

ECA 150 is surjective on \mathcal{C}_n iff $n \not\equiv 0 \pmod{3}$.

ECA 51 is always surjective on \mathcal{C}_n .

Reducing the Search Space

Proposition 4. *ρ is surjective on \mathcal{C}_n for infinitely many n implies that ρ is surjective on \mathcal{C}_∞ .*

Proof.

Consider $Y \in \Sigma^\infty$.

For any n such that ρ is surjective on \mathcal{C}_n there is an $X_n \in \mathcal{C}_n$ such that $\rho(X_n) = Y[0, n - 1]$ (with cyclic boundary conditions).

Let $X = \lim X_n$ (rather: pick a converging subsequence).

Then $\rho(X) = Y$.

□

Warning: The converse is false, ECA 90 is a counterexample.

FSMs to the Rescue

Let \mathcal{A} be the de Bruijn semiautomaton for ρ .

Let \mathcal{A}_p be the NFA obtained from \mathcal{A} with p as initial and final state.

Set

$$\mathcal{B} = \bigoplus_p \mathcal{A}_p$$

But then $\Sigma^n \subseteq \mathcal{L}(\mathcal{B})$ iff ρ is surjective on \mathcal{C}_n .

Any word $Y = Y_1Y_2 \dots Y_n \in \mathcal{L}(\mathcal{B})$ must come from a path in \mathcal{A} labeled Y and tracing a cycle (not necessarily simple) in \mathcal{A} .

Deciding Inclusion

Lemma 4. *For regular languages $L, K \subseteq \Sigma^*$ one can decide whether $L \subseteq K$ (and thus also whether $L = K$).*

Proof.

$L \subseteq K$ iff $L - K = \emptyset$.

But given FSMs for L and K we can construct a FSM for $L - K$.

Testing whether this machine has empty acceptance language is easy.

□

It's a good exercise to come with an algorithm that tests whether a FSM accepts only finitely many words.

Back To CA

Note that subset testing is guaranteed to be polynomial time only if the given machines are DFAs, otherwise there may be exponential blow-up.

Needless to say, the de Bruijn automata we are interested here are usually nondeterministic, so we don't have a really good method yet to check if ρ is surjective on \mathcal{C}_n .

Also note that we have only tackled the decision problem: Given ρ and n , is ρ surjective on \mathcal{C}_n ?

What we really would like is a more general description of the set of n for which ρ is surjective on \mathcal{C}_n .

Summary

- Two-dimensional cellular automata are much more complicated than their one-dimensional counter parts.
- Some properties that are decidable for one-dimensional CA become undecidable for two-dimensional CA.
- Potentially interesting for applications if hardware support is available.
- Used implicitly in image processing.
- Usefulness in cryptography unclear at this point.