Bridge of formal language theory and automata theory

Jiaxuan Chen

CSE 645: Seminar in languages, Department of Computer Science, Stony Brook University

Work of Lyman P. Hurd, Department of Mathematics, Princeton University

Hurd, Lyman P. *Formal language characterizations of cellular automaton limit sets*. Complex systems 1.1 (1987): 69-80.
https://content.wolfram.com/sites/13/2018/02/01-1-6.pdf

September 24, 2026



Outline

1. Introduction
2. Cellular Automaton Limit Sets
3. Limit Languages
4. Formal Language Background
5. Determining Language Complexity
6. Cellular automation Limit Languages
7. Discussion

Fun facts

This article came from Lyman Hurd's PhD thesis at Princeton University, under supervision of Jack Milnor.
I met Lyman this summer at the Wolfram Summer Research Program held by the Wolfram Institute.



Figure: I and Lyman, July 2026

Introduction

- Some intuition:
- Dynamical system: a set with some "rule" on it which "rewrites" elements of the set to the others.
- Formal language: an alphabet (finite) and some formation rules, giving rise to the family of "words" that obey our "grammar"; equivalently, set of outputs of a Turing machine.
- Cellular automata (CA): a dynamical system on a discrete lattice of sites taking value in some alphabet with a "local" rule of rewriting.
- CA limit set: the "limiting" sets of lattices of site values reached by some CA after indefinitely many times of evolutions. The limit set is in fact a language.
- Our goal: Characterize CA limit sets in terms of formal language.

Cellular automation

- Given S which is finite alphabet, $|S| = k$. We take this to be the *site values* of our CA.
- Intuitively, a *cellular automaton* on S is an evolution on the infinite tape $S^{\mathbb{Z}}$ that is given by successive action of some "local" rewriting rule, in a sense that the transformation of a value at a site only depends on sites up to r steps away.
- Formally, a *cellular automation* is an ordered pair (S, ϕ, r) where S is a finite set of *site values*, and ϕ is a local transformation acting on some $r-$neighborhood of each site value

$$\phi : S^{2r+1} \to S.$$

- This would give rise to a unique *global transition rule* on $S^{\mathbb{Z}}$

$$\Phi : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$$

where

$$\Phi(s)_n = \phi(s_{n-r}, ..., s_0, ..., s_{n+r}).$$

Consequently, $(S^{\mathbb{Z}}, \Phi)$ is a dynamical system.

- These Φ give rise to all continuous map on $S^{\mathbb{Z}}$ into itself (endowed with the discrete product topology) which are invariant under translation, i.e. the continuous maps $f : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ such that for the shift map

$$\sigma : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$$

where

$$\sigma(s_i) = s_{i+1}$$

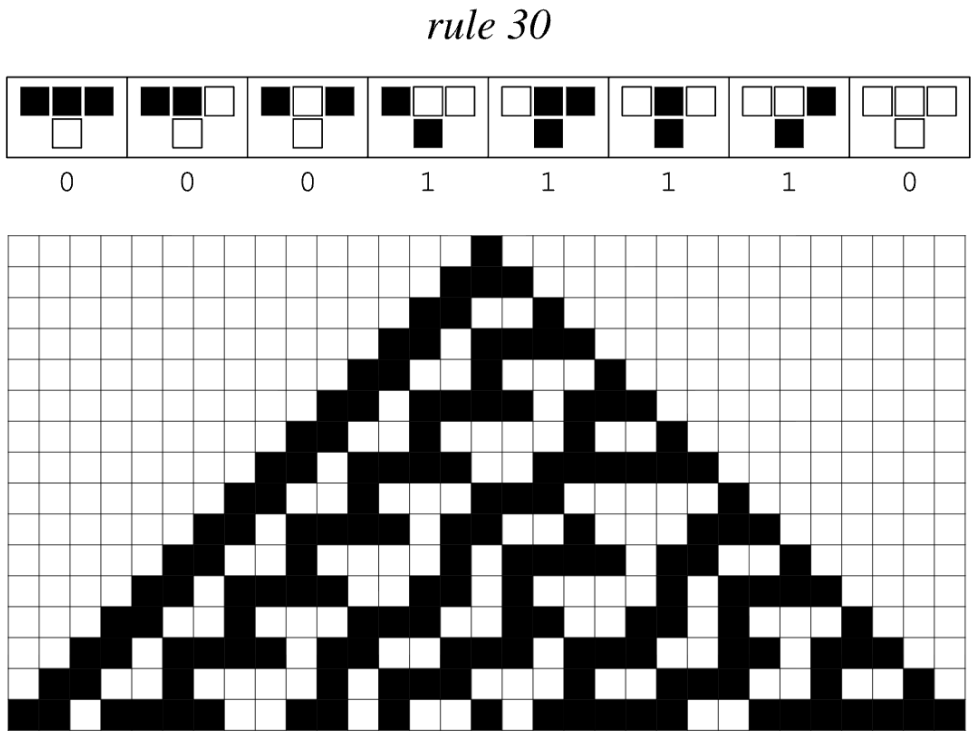
we have

$$f \circ \sigma = \sigma \circ f.$$

Example

(Rule 30, source: Wolfram MathWorld)

$$S = \{B, W\}, r = 1$$



Limit Sets

- Given rule Φ: let $\Omega_\Phi^{(0)} = S^{\mathbb{Z}}$, $\Omega_\Phi^{(i)} = \Phi\left(\Omega_\Phi^{(i-1)}\right)$.
- Note that $\Omega_\Phi^{(i)} \supseteq \Omega_\Phi^{(j)}$ for $i \leq j$, for if $c \in \Omega_\Phi^{(j)}$, then for some $s, \Phi(s) \in S^{\mathbb{Z}}$,

$$c = \Phi^{j-1}(\phi(s))$$

and the claim follows by induction on j. Then, arbitrary intersection of a nested sequence of sets is well-defined, and let

$$\Omega_\Phi = \bigcap_{i=0}^{\infty} \Omega_\Phi^{(i)}$$

be the **limit set** of Φ. Intuitively, this is the set of configurations with arbitrarily long past histories starting with some initial configuration in $(S^{\mathbb{Z}}, \Phi)$.

- $c \in \Omega_\Phi$ iff there exist a countably infinite set of configurations c^{-i}, $i = 0, 1, 2, \ldots$, with $c^0 = c$ and $\Phi(c^{-i}) = c^{-i+1}$, i.e. $c = c^0 = \Phi(c^{-1}) = \Phi^2(c^{-2}) = \ldots$

Limit language characterizes limit set of CA

Definition 1

For $\Omega \subseteq S^{\mathbb{Z}}$, the set of all finite connected substrings of configurations in Ω is the **associated language** $\mathsf{L}[\Omega]$. For a CA (S, ϕ, r), $\mathsf{L}[\Omega_\Phi]$ is the **limit language** of Φ.

Theorem 1

If Δ_1, Δ_2 are closed, translation-invariant subsets of $S^{\mathbb{Z}}$, then

$$\mathsf{L}[\Delta_1] = \mathsf{L}[\Delta_2] \implies \Delta_1 = \Delta_2.$$

Corollary 1. On a CA with global transformation rules Φ_1, Φ_2, we have $\mathsf{L}[\Omega_{\Phi_1}] = \mathsf{L}[\Omega_{\Phi_2}] \Rightarrow \Omega_{\Phi_1} = \Omega_{\Phi_2}$: the language *determines* the limit set.

Formal Languages from Machines and Grammars

- Formal language theory is the study of sets of string from a finite alphabet.
- Two equivalent definitions of formal languages: output of some class of machines; or end product of some typographcial substitution system (grammars). For more details, see M.Minsky *Finite and Infinite Machines*.
- We will restrict our attention to those languages that are given by *generative grammars*, or, equivalently, by *Turing machines*

Turing machine

Definition (Turing machine)

Formally, a *Turing machine* is a quintuple $T(Q, A, F, G, D)$ with a finite set of *states* $Q = \{q_1, ..., q_n\}$, *finite alphabet of symbols* $A = \{s_1, ..., s_n\}$, *output function* $F : Q \times A \to A$, *state transition function* $G : Q \times A \to Q$, *head direction function* $D : Q \times A \to \{-1, 1\}$.

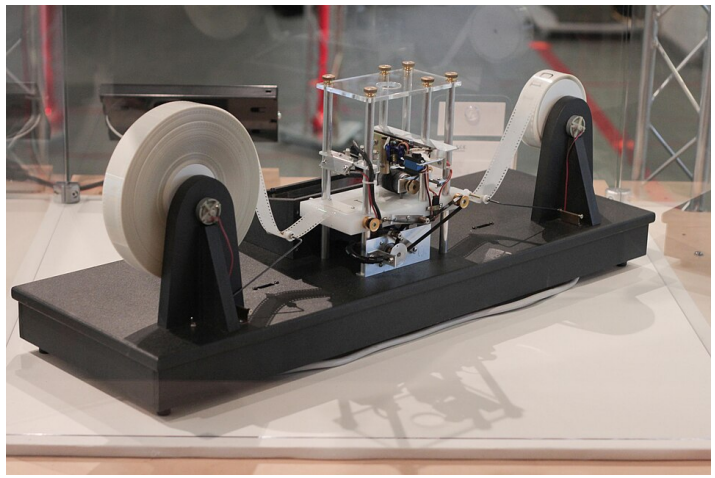


Figure: A physical model of a Turing machine (wikipedia)

Cellular automation from Turing machine

A Turing machine gives rise to a cellular automation rule Ψ_T which emulates the Turing machine T. Intuitively, Ψ_T acts on an infinite tape divided into squares, where each square is occupied by some tape symbol $s_i \in A$, and a head with a state moves on the tape, rewriting the current square's symbol according to F, and then changes its state by G, and determine the next move by D.

Formally, $((Q \cup \{l, r\}) \times A, \Psi_T, r = 1)$ is a cellular automaton, where we restrict $(Q \times A)^{\mathbb{Z}}$ to all those sequences $(q_i, s_i)_{i \in \mathbb{Z}} \in ((Q \cup \{l, r\}) \times A)^{\mathbb{Z}}$ that satisfies the property that there exists a unique i such that $(q_i) = (..., l, l, l, q_i, r, r, r, ...)$. The action of Ψ_T on such (q_i, s_i) is canonically given by the intuitive description above.

This *corresponding language* of a Turing machine T is defined as all possible tape value outputs of T, i.e. all element of $\Omega_{\Psi_T}^{(n)}$ for all $n \in \mathbb{Z}_+$ where we take the string induced by each element (omit the state) to be a sentence in our language.

Generative grammar

Definition (generative grammar)

A *generative grammar* is a tuple $G(\Sigma, T, N, \phi)$ with start symbol Σ, set of terminal symbols T, set of non-terminal symbols N and a set of $\Phi = \{\phi_i\}$ of typographical substitution rules ϕ_i

The corresponding language of a generative grammar is the set of all terminal symbols derived from Σ by successive typographical substitutions.

The Chomsky Hierarchy of r.e. languages

Language	Grammar	Machine
r.e. languages	generative	Turing machine
context-sensitive	string $\rightarrow$ longer string	linear bounded automaton
context-free	symbol $\rightarrow$ string	push-down automaton
regular	right-linear	finite-state machine

- Classifies all recursively enumerable (r.e.) languages according to the complexity of the generating grammar, or equivalently, the complexity of the Turing machine of recognition.
- Position of a language in the Chomsky hierarchy suggests about what propositions on this language are computable/decidable, i.e. can be computed by some Turing machine in finite time. Otherwise, we say this proposition is undecidable.

Recursively enumerable languages

r.e. Language

We say a language is *recursively enumerable* if it is the language corresponding to some generative grammar, i.e. corresponding to some Turing machine.

For example, the language $a^n b^n$ is generated by the generative grammar $T = \{a, b\}, N = \{\Sigma, A\}$ with typographical substitution rules $\Sigma \to A$, $A \to \epsilon$ and $A \to aAb$.

Context-sensitive languages

Context sensitive Language

A r.e. language is *context sensitive* if it can be given by a generative grammar such that every production rule in the grammar increases the length of the string. Equivalently, those are the languages which can be recognized by *linear bounded automata*, i.e. Turing machines with memory bounded linearly by the length of the word to be recognized.

For example the language $x^n y^n z^n$ with alphabet $\{x, y, z\}$ is context-free.

Context-free languages

context-free Language

A r.e. language is *context-free* if it has grammar such that all strings on the left side of transformation rules are of length one, i.e. sends only symbols to strings. Equivalently, those are the languages of the push down automata, i.e. Turings machines on a stack (tape with first-in-last-out principle). Note that, all context-free languages are context-sensitive.

For example, the language of balanced parentheses (the bracketings) is context-free.

Regular languages

regular Language

A r.e. language is *regular* if it can be recognized by a *finite state machine*, i.e. Turing machines with finite memory. Equivalently, these are the languages generated by right linear grammars, i.e. grammas such that any non-terminal is sent to either: a non-terminal followed by a terminal; or a terminal.

Equivalent characterizations:

- Walks through a graph where edge labels are from the given alphabet.
- as "regular expressions", defined by union, disjunction and Kleene closure inductively.

Examples of regular languages

The set of all strings in the alphabet $\{0, 1\}$ that does not contain two consecutive ones is a regular language.

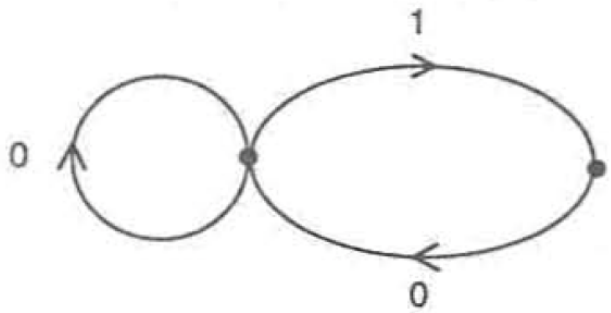


Figure 1: Graph representing a regular language containing all strings in the alphabet $\{0, 1\}$ with no two consecutive ones.

Figure: Source: Lyman's paper

Pumping Lemmas

Motivation: for an arbitrary generative grammar, it is undecidable which part of the Chomsky hierarchy it belongs to. However, the pumping lemmas can guarantee that a given language is not in a class of the Chomsky hierarchy – they give necessary conditions for a language to be in a class.

Regular pumping lemma

For regular R, $\exists\, n$ such that every string of length $\geq n$ factors as abc with $ab^i c \in R$ for all i.

Context-free pumping lemma

For context-free F, $\exists\, n$ such that every string of length $\geq n$ factors as $abcde$ with $ab^i cd^i e \in F$ for all i.

- Consequence: $a^n b^n$ is **not regular**; $a^n b^n c^n$ is **not context-free**.

Existence of limit languages of each class

- Given that it is undecidable to classify a given language in the Chomsky hierarchy – we consider the weaker existence questions instead.
- Known: there are many CA rules with (strictly) regular limit languages.
- Natural follow-up open question: are there rules with limit languages that are strictly context-free or strictly context-sensitive?
- Wolfram (1984): $\mathrm{L}\left[\Omega_{\Phi}^{(n)}\right]$ is always **regular** for finite n.
- Open question answered by Lyman: is $\mathrm{L}[\Omega_{\Phi}]$ (the full limit language) always regular?

The answer is negative (and even stronger)

Lyman exhibits CA rules whose limit languages are, in turn:

1. not regular,
2. not context-free,
3. not recursively enumerable.

Rule Λ: A Non-Regular Limit Language

$S = \{r, I, W, o\}$, $r = 2$, with transformation rules

Pattern	$\rightarrow$	Result
$.rox.$	$\rightarrow$	r
$..yoI$	$\rightarrow$	I
$..rWI$	$\rightarrow$	I
$rWI..$	$\rightarrow$	r
otherwise: identity		

- x: any symbol except I; y: any symbol except r; .: any symbol.
- Simulates two markers (r, I) walking toward a wall W and annihilating symmetrically.

Some finite section of Λ

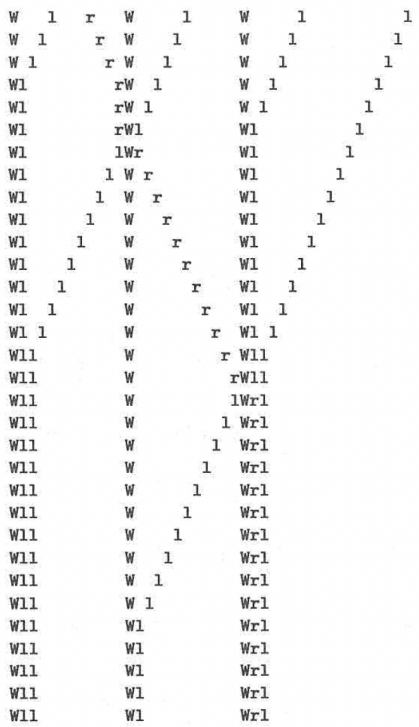


Figure 2: The evolution of cellular automaton rule Λ from a sample initial state. Blanks denote the symbol *o*.

Figure: Source: Lyman's paper

Theorem 2: $\mathrm{L}[\Omega_\Lambda]$ is Not Regular

Proof idea

Intersect with a regular language

$$F = \mathrm{L}[\Omega_\Lambda] \cap \{oolo^* Wo^* roo\}$$

would still give a regular language.

- $F \neq \emptyset$: strings $oolo^n Wo^n roo$ have predecessors $o^* ro^i Wo^i lo^*$.
- If $c = oolo^n Wo^m roo \in F$ with $n, m > 0$, its predecessor forces $n = m$.
- So $F = \{oolo^n Wo^n roo\}$, which **violates the regular pumping lemma**, then $\mathrm{L}[\Omega_\Lambda]$ itself cannot be regular. ■

Rule Λ': A Non-Context-Free Limit Language

$S = \{r, R, l, L, W, o\}$, $r_{\text{radius}} = 4$ — same strategy, one more "layer" of markers (r/R and l/L).

Theorem 3

$\mathrm{L}[\Omega_{\Lambda'}]$ is not context-free.

- Intersect with regular $\{ooooLo^*lo^*Wo^*ro^*Roooo\}$ to get

$$F' = \mathrm{L}[\Omega_{\Lambda'}] \cap \{ooooLo^*lo^*Wo^*ro^*Roooo\}.$$

- Then $F' = \{ooooLo^{n+2}lo^nWo^nro^{n+2}Roooo\}$.
- This set **violates the context-free pumping lemma**, so $\mathrm{L}[\Omega_{\Lambda'}]$ is not context-free. ■

Finite sections of Λ'

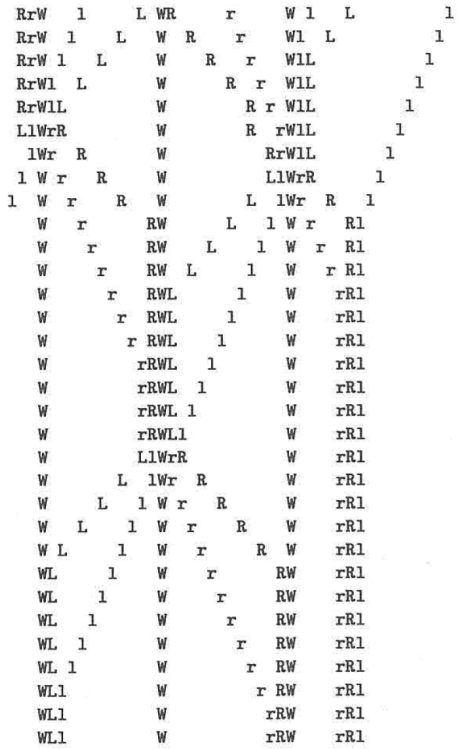


Figure 3: The evolution of cellular automaton rule Λ' from a sample initial state. Blanks denote the symbol o.

Figure: Source: Lyman's paper

Rule Ψ_T: Emulating a Turing Machine

Given $T = (Q, A, F, G, D)$, we construct ψ_T as before.

Theorem 4

For a Turing machine T and state q_h: whether (q_h, s_i) lies in the limit language of Ψ_T is **undecidable** in general.

f this is decidable, then the proposition addressed in the following lemma is decidable, yielding contradiction.

Lemma 1.

Given Turing machine T and a sate $q_h \in Q$, it is undecidable whether T can reach state q_h after some n steps for $n \in \mathbb{N}$. Proof is given in [7].

The proof sketch of this lemma is given in section 7 of Lyman's paper. It essentially reduces this problem to the problem of a machine halting on a blank tape, which is known to be undecidable. For a full proof, consult [8].

Non-Recursive Limit Languages

Theorem 5

The complement of $L[\Omega_\Psi]$ is r.e. for *every* CA rule Ψ.

- A string s excluded from the limit language is excluded after finitely many steps n; strings excluded by step n are computable from the n^{th} iterate of Φ.

Corollary 2

There exists a rule Φ such that $L[\Omega_\Phi]$ is **not recursively enumerable**.

- Definition: A language is recursive if it is undecidable whether a string lies in its limit language.
  Fact: a language is recursive iff it *and* its complement are r.e.
- Theorem 4 gives a rule Ψ_T whose limit language is not recursive, by definition. By Theorem 5, this rule's language has a r.e. complement. Then, it is necessary that this language is not r.e., proving Corollary 2.

Discussion & Open Questions

- Very few limit languages have been computed explicitly.
- By **Rice's Theorem**, every non-trivial (i.e. not globally true/false) property of r.e. languages is undecidable in general — an analogous undecidability is conjectured for CA limit languages.
- Conjectured undecidable propositions: is a given rule's limit language regular? Does it consist of more than one string? Do two rules share a limit set?

More open problems

- Characterize exactly which languages *can* occur as CA limit languages.
- Even simpler: which *regular* languages occur as the one-step image $\Omega_\Phi^{(1)}$ of a CA?

Summary

Rule	Limit language	Class boundary crossed
Λ	$\mathsf{L}[\Omega_\Lambda]$	not regular
Λ'	$\mathsf{L}[\Omega_{\Lambda'}]$	not context-free
Ψ_T	$\mathsf{L}[\Omega_{\Psi_T}]$	not recursive / not r.e.

- Formally describe CA limit sets using formal language
- Use examples to show existence of CA limit languages in each layer of the Chomsky hierarchy, and strictly beyond each layer of the Chomsky hierarchy.

References

[1] S. Wolfram, "Computation theory of cellular automata," *Comm. Math. Phys.* **96** (1984) 15–57.

[2] G. A. Hedlund, "Endomorphisms and automorphisms of the shift dynamical system," *Math. Systems Theory* **3** (1969) 320.

[4] J. E. Hopcroft, J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[5] M. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, 1967.

[7–8] U. Golze, on differences between 1- and 2-dimensional cell spaces; dissertation, TU Hannover.

Thank you

Any questions?