Formalized Cellular Automata Theory in Lean 4

This project formalizes key results about cellular automata that recognize languages, focusing on **real-time** recognition, **one-way (left-independent) CAs**, and **advice mechanisms**. All results compile with Lean 4 and Mathlib4; proofs are axiom-verified (only `Quot.sound`, `Classical.choice`, and `propext` are used).

Setup and Non-Standard Definitions

Cellular Automaton

A CA is a one-dimensional cellular automaton with radius-1 neighborhood, given as a tuple $C = (Q, \Sigma, \Gamma, \delta, \mathsf{embed}, \mathsf{project})$ with state set Q, input alphabet Σ, output alphabet Γ, local transition $\delta : Q^3 \to Q$, and maps $\mathsf{embed} : \Sigma \to Q$, $\mathsf{project} : Q \to \Gamma$. The split into input/output types lets CAs act as transducers ($\Sigma \to \Gamma$), not just language recognizers. Acceptance becomes a special case where $\Gamma = \mathsf{Bool}$.

A **configuration** is a map $c : \mathbb{Z} \to Q$. One step: $\mathsf{next}(c)_p = \delta(c_{p-1}, c_p, c_{p+1})$. We write $\Delta_C^t(c)$ for the t-fold iterate, and $\mathsf{comp}_C(c, t, i) = \mathsf{project}(\Delta_C^t(\mathsf{embed} \circ c)_i)$.

Relation to the standard definition. In the literature (e.g. Kutrib, Malcher, Worsch), a language-recognizing CA is typically $C = (Q, \Sigma, \#, \delta, F_+)$ with $\Sigma \subset Q$ (the input alphabet is a subset of the state set), a quiescent border $\#$ with $\delta(\#, \#, \#) = \#$, and an accepting set $F_+ \subset Q$. Our formalization differs as follows:

1. **Separate input/output types** via embed and $\mathsf{project}$, enabling the transducer viewpoint. The standard definition is recovered by setting $\mathsf{embed} = \mathrm{id}$, $\Gamma = \mathsf{Bool}$, and $\mathsf{project} = 1_{F_+}$.
2. **No border constraints.** The border state $\mathsf{embed}(\#)$ is not assumed quiescent or dead. Results 4 and 5 below show this is without loss of generality.
3. **Timed acceptance.** Instead of a fixed F_+, a timed CA specifies functions $t(n), p(n)$ and accepts via $\mathsf{project}\left(\Delta_C^{t(n)}(\langle w \rangle)_{p(n)}\right) = \mathsf{true}$. Real-time is the special case $t(n) = n - 1$, $p(n) = 0$.

Word Embedding (0-indexed)

Words are embedded into configurations with **0-based indexing**: a word w of length n occupies positions $0, 1, ..., n - 1$, with all other positions set to the border symbol $\#$. Formally:

$$\langle w \rangle(p) = \begin{cases} w_p & \text{if } 0 \le p < |w| \\ \# & \text{otherwise} \end{cases}$$

For language-recognizing CAs the input alphabet is $\Sigma_\# = \Sigma \cup \{\#\}$, so $\mathsf{embed}(\#)$ gives the border state.

Trace

The **trace** of C on configuration c is the temporal output sequence at position 0:

$$\mathsf{trace}_C(c) : \mathbb{N} \to \Gamma, \quad t \mapsto \mathsf{comp}_C(c, t, 0)$$

Real-Time Trace

The **real-time trace** is the word-to-word transduction where position i reads out time i:

$$\mathsf{trace\_rt}_C(w) = (\mathsf{trace}_C(\langle w \rangle)(0), \mathsf{trace}_C(\langle w \rangle)(1), ..., \mathsf{trace}_C(\langle w \rangle)(n - 1))$$

This is the central notion for composing CA transducers: $\mathsf{trace\_rt}_C : \Sigma^* \to \Gamma^*$ is a length-preserving map.

Left-Independent (One-Way) CA
A CA is **left-independent** if δ ignores its left argument:

$$\forall a, a', b, c : \quad \delta(a, b, c) = \delta(a', b, c)$$

These correspond to **one-way CAs (OCA)**. The **left-independent light cone** at position p and time t for a word of length n is $\{p \ \mathtt{mid} - t \leq p < n\}$.

Real-Time Language Class $\mathcal{L}(\mathsf{CA}_{\mathsf{rt}})$
A CA **accepts** a word w of length n by reading a designated cell at a designated time. A **timed CA** specifies functions $t(n)$ (time) and $p(n)$ (position) and accepts w iff $\mathsf{comp}_C(\langle w \rangle, t(|w|), p(|w|)) = \mathsf{true}$.

For the standard classes:
- CA: read position 0, i.e. $p(n) = 0$.
- $\mathsf{CA}_{\mathsf{rt}}$: read position 0 at time $n - 1$ (real-time).
- OCA: left-independent CA reading at position 0.
- $\mathsf{OCA}_{\mathsf{rt}}$: left-independent, real-time.

The class $\mathcal{L}(\mathsf{CA}_{\mathsf{rt}})$ is the set of languages recognized by real-time CAs. Note the 0-indexed embedding: a word of length n occupies positions $0, ..., n - 1$, and at time $n - 1$ the information from the rightmost cell has just reached position 0.

Advice Functions
An **advice** is a length-preserving map $f : \Sigma^* \to \Gamma^*$ with $|f(w)| = |w|$.

- **RT-closed:** f is RT-closed if $\mathcal{L}(\mathsf{CA}_{\mathsf{rt}}(\Sigma \times \Gamma)/f) = \mathcal{L}(\mathsf{CA}_{\mathsf{rt}}(\Sigma))$, i.e. the advice does not increase the power of real-time CAs.
- **Causal (prefix-stable):** $f\left(w_{[0..i)}\right) = f(w)_{[0..i)}$ for all w, i.
- **Two-stage:** f factors as $f = M \circ \mathsf{trace\_rt}_C$, where C is a CA real-time transducer and M is a finite-state transducer scanning right-to-left.

Part I: Classical Constructions (existing literature, sorry-free)

The following results are well-known in the literature (see e.g. Kutrib, Malcher et al.). The proofs here sometimes differ from the classical ones, as certain constructions were adapted to be more amenable to formal verification in Lean 4. All proofs are **completely sorry-free**.

Result 1: Left-Independent ↔ Regular Simulation
Given a left-independent CA C, construct a regular CA C' such that:

$$\Delta^t_{C'}(c)_i = \Delta^{2t}_C(c)_{i-t}$$

Conversely, given any CA C, construct a left-independent C' with $Q' = Q \cup (Q \times Q)$ such that:

$$\Delta^{2t}_{C'}(c)_i = \Delta^t_C(c)_{i+t}$$

This establishes the equivalence of OCA and CA up to a constant factor of 2 in time.

Lean: `result_left_indep_to_regular`, `result_regular_to_left_indep`

Result 2: k-Step Left-Independent Speedup
Given a left-independent CA $C = (Q, \delta)$ and $k \geq 2$, construct a left-independent $C' = \left(Q^k, \delta'\right)$ compressing k consecutive diagonal cells into one tuple. Define coordinate maps:

$$\psi(i, j) = ki + j, \quad \varphi(t, i, j) = t - (k-1)i - j$$

Then for $i < 0$ and $0 \leq j < k$:

$$\mathsf{comp}_{C'}(w, t, i)_j = \mathsf{comp}_C(w, \varphi(t, i, j), \psi(i, j))$$

The proof proceeds by outer induction on t and inner descending induction on j within each time step.

Lean: `result_left_indep_speedup`

Result 3: General k-Step RT Speedup
For any CA C and constant k, construct C' such that:

$$\mathsf{trace}_{C'}(w)(i) = \mathsf{trace}_C(w)(i + k)$$

This achieves a constant additive speedup by chaining QuiescentBorder and DeadBorder constructions.

Lean: `SpeedupKSteps.spec`

Result 4: Quiescent Border for Left-Independent CAs
Given a left-independent CA C, construct C' whose border is **quiescent** ($\delta(\#, \#, \#) = \#$), while $\mathsf{comp}_{C'} = \mathsf{comp}_C$ inside the left-independent light cone. Together with Result 5, this shows that the unconstrained border in our formalization is without loss of generality.

Lean: `result_quiescent_border_left_indep`

Result 5: Dead Border
Given any CA C, construct C' whose border state $\#$ is **dead** (absorbing: $\delta(\cdot, \#, \cdot) = \#$), while preserving the trace: $\mathsf{trace}_{C'}(w)(t) = \mathsf{trace}_C(w)(t)$ for all $t < c \cdot |w|$, where c is a constant depending on C'. In particular, the trace is preserved for any linear-time computation. Uses a zigzag folding of cells into lanes.

Lean: `result_dead_border`

Result 6: Exponential Word Length is RT-Recognizable
The language $\{w \texttt{ mid } |w| = 2^n \text{ for some } n\}$ is in $\mathcal{L}(\mathsf{CA}_{\mathsf{rt}})$. The construction uses a signal-bouncing technique: a signal is sent from the left border, bounces off the right border, and its return time encodes the word length.

Lean: `exp_word_length_rt`

Part II: Advice Theory (likely novel, sorry-free)

The following results are likely **novel** and form the core contribution of this project. They develop a structural theory of *advice* for cellular automata, establishing closure properties of RT transducers and two-stage advice, and classifying causal RT-closed advice as RT transducers.

Result 7: RT transducers are closed under composition *sorry-free*

Given CA transducers $C_1 : \Sigma \to \Gamma_1$ and $C_2 : \Gamma_1 \to \Gamma_2$, there exists a CA C with $\mathsf{trace\_rt}_C = \mathsf{trace\_rt}_{C_2} \circ \mathsf{trace\_rt}_{C_1}$. This is the most technically challenging result in the project, requiring the full machinery of dead border, passive border, k-step speedup, and left-independent $\leftrightarrow$ regular simulation. The proof uses a multi-stage pipeline:

$$\mathsf{AddBorder} \to \mathsf{CompressToDiag} \to \mathsf{SimFrom}\ \Lambda \to \mathsf{DecompressTriple} \to \mathsf{SpeedupKSteps}$$

Lean: `result_rt_transducers_closed_under_composition`

Result 8: Two-stage advice is RT-closed *sorry-free*

If f is two-stage, then $\mathcal{L}(\mathsf{CA}_{\mathsf{rt}}(\Sigma \times \Gamma)/f) = \mathcal{L}(\mathsf{CA}_{\mathsf{rt}}(\Sigma))$. This follows from Result 7: the CA component is RT-closed, and the right-to-left FST can be absorbed into the receiving CA.

Lean: `result_two_stage_is_rt_closed`

Result 9: Prefix-membership advice is two-stage *sorry-free*

For any $L \in \mathcal{L}(\mathsf{CA}_{\mathsf{rt}})$, the advice f_L defined by

$$f_L(w)_i = \left[w_{[0..i+1)} \in L\right]$$

is itself a two-stage advice (and hence an RT transducer): $f_L = \mathsf{trace\_rt}_C$ for a suitable CA C that runs the recognizer for L and outputs the acceptance bit at each step.

Lean: `result_advice_prefix_mem_is_two_stage_advice`

Result 10: RT-closed $\wedge$ causal $\Longrightarrow$ CArt advice *sorry-free*

If an advice f is both RT-closed and causal (prefix-stable), then f is a CArt advice, i.e., computable by a single CA RT transducer. The proof constructs C by observing that causality lets one reduce f to a product of prefix-membership advices (one for each output symbol $c \in \Gamma$, via the language $L_c = \left\{w \ \mathtt{mid}\ f(w)_{|w|} = c\right\}$), each of which is an RT transducer by Result 9.

Lean: `result_is_cart_advice_of_rt_closed_and_causal`

Result 11: Two-stage advice is closed under composition *sorry-free*

Given two-stage advices $f_1 : \Sigma^* \to \Gamma_1^*$ and $f_2 : \Gamma_1^* \to \Gamma_2^*$, the composition $f_2 \circ f_1$ is again two-stage. The proof works by commuting the FST of f_1 past the CA of f_2 using a “backwards FSM” construction that absorbs the FST into the CA’s state space.

Lean: `result_two_stage_closed_under_composition`

Result 12: Middle advice is *not* two-stage *sorry-free*

The advice f_{mid} that marks position $\lfloor n/2 \rfloor$ (i.e., $f_{\text{mid}}(w)_i = [i = \lfloor |w|/2 \rfloor]$) cannot be expressed as a two-stage advice. The proof uses a bottleneck argument: the FST has finitely many states, but the middle position requires information about the full word length, which the CA’s real-time trace at the midpoint cannot encode in bounded state.

Lean: `result_middle_not_two_stage_advice`

Incomplete Results (with sorry)

Exponential-Middle Advice is Two-Stage *4 sorry remaining*

The advice that marks the largest power-of-2 position $\leq n/2$ is conjectured to be two-stage. The two-stage decomposition (a CA transducer marking powers of 2, composed with an FST selecting the last "true") is fully constructed. The 4 remaining `sorry`s are in combinatorial counting lemmas.

Lean: `exp_middle_two_stage_advice`

Unproven Conjectures *8 sorry in results_unproven.lean*

The following are stated but unproven:

- **Constant speedup:** $\mathcal{L}(\{C \in \mathsf{CA}\ \texttt{mid}\ t(n) = n + k - 1\}) = \mathcal{L}(\mathsf{CA}_{\mathsf{rt}})$
- **CA linear time = 2n:** $\mathcal{L}(\mathsf{CA}_{\mathsf{lt}}) = \mathcal{L}(\mathsf{CA}_{2n})$
- **OCA linear time = 2n:** $\mathcal{L}(\mathsf{OCA}_{\mathsf{lt}}) = \mathcal{L}(\mathsf{OCA}_{2n})$
- **OCAr linear time = CA rt:** $\mathcal{L}(\mathsf{OCA}^{r}_{\mathsf{lt}}) = \mathcal{L}(\mathsf{CA}_{\mathsf{rt}})$
- **Reversal closure implies lt = rt:** $\mathcal{L}(\mathsf{CA}) = \mathcal{L}(\mathsf{CA}^{r}) \implies \mathcal{L}(\mathsf{CA}) = \mathcal{L}(\mathsf{CA}_{\mathsf{lt}})$
- **Advice shift-left preserves two-stage**
- **CartTraceFstAdvice classification**

Open Question

Is every RT-closed advice two-stage, without the causality assumption? We conjecture that no such counterexample exists. However, if a non-two-stage RT-closed advice did exist, its RT-closedness proof would probably require a fundamentally different, non-geometric simulation construction — and it could be promising to investigate whether such a construction can be shown to be generally uncomputable.

Project Statistics

Category	**Files**	**Sorry-free**	**With sorry**
Core proofs & utilities	7	7	0
Main theorems	5	4	**1**
Basic CA constructions	10	10	0
Speedup constructions	3	3	0
Direction conversions	2	2	0
Composition pipeline	8	8	0
Framework & scripts	4	4	0
Total	**39**	**38**	**1**

Total `sorry` count: **4** in proof files (`exp_middle_two_stage.lean`) + **8** in `results_unproven.lean` (conjectured theorems). The 10 results in `results.lean` are **completely sorry-free**.