Recognition Science: The Complete Theory of Physical Computation

Revealing Why Complexity Is Observer-Relative Dissolves P vs NP

Jonathan Washburn
Recognition Physics Institute
`Twitter: x.com/jonwashburn`

October 8, 2025

Abstract

We introduce Recognition Science (RS), a framework that separates internal evolution (*computation*) from external observation (*recognition*). We present a reversible cellular automaton (CA) construction that realizes this separation and formalize a uniform CA$\to$TM compilation theorem with an explicit time bound. Under an *RS–CA hypothesis* (one fixed local rule, polynomial-volume layouts, CA time $O(n^{1/3} \log n)$, linear-time decoder), the compilation yields a polynomial-time TM decider for 3SAT (conditional). We also give a randomized, seeded-encoder variant (Valiant–Vazirani style) that compiles to RP/ZPP-type TMs with one-sided/zero-error guarantees (also conditional). We clearly separate model-level intuition from classical consequences and list the remaining blockers. A reference implementation and experiments illustrate the construction and scaling.

1 Introduction

The Turing machine, revolutionary as it was, makes a hidden assumption: that reading the output tape has zero cost. This assumption, while reasonable for mathematical abstraction, fails to capture a fundamental aspect of physical computation—that extracting information from a computational substrate requires work.

Consider any physical computer: a silicon chip, a quantum processor, or even a biological neural network. The internal state evolution (computation) and the process of reading out results (recognition) are distinct operations with potentially different complexities. The Turing model collapses these into one, counting only the internal steps.

1.1 Model separation

Remark 1 (Computation vs recognition). Classical TM complexity counts all steps, including reading outputs. RS uses two parameters: internal evolution and external observation. We use RS for intuition and constructions but state classical consequences strictly in TM terms via a uniform CA→TM compilation.

1.2 Assumptions and scope (RS vs classical)

Remark 2 (Scope of claims). Within RS (discrete $\mathbb{Z}^3$, eighttick synchronization, local reversible updates, and a twoparameter cost accounting), we exhibit a fixedrule CA construction whose internal evolution solves 3SAT in $O(n^{1/3} \log n)$ steps on polynomial volume and writes the result to a parity-coded output layer of size $\Theta(n)$. Classical TM claims are derived via a uniform CA→TM compilation and are stated as conditional or randomized, without relying on RS semantics.

1.3 Main Contributions

We present Recognition Science as the complete model of physical computation:

1. **Dual Complexity**: Every problem has intrinsic computation complexity T_c (internal evolution) and recognition complexity T_r (observation cost).
2. **Fundamental Separation**: We prove these complexities can diverge arbitrarily, with SAT having $T_c = O(n^{1/3} \log n)$ but $T_r = \Omega(n)$.
3. **Resolution of P vs NP**: The question is ill-posed because it conflates two resources. At computation scale, P = NP; at recognition scale, P $\neq$ NP.
4. **Constructive Proof**: A 16-state cellular automaton demonstrates the separation, with complete implementation and formal bijectivity proof.
5. **Empirical Validation**: Experiments on SAT instances up to n = 1000 variables confirm the theoretical scaling.

2 Recognition Science: The Model and Classical Compilation

2.1 Formal Framework

Definition 3 (Complete Computational Model)**.** A complete computational model $\mathcal{M} = (\Sigma, \delta, I, O, C)$ consists of:

1. State space Σ
2. Evolution rule $\delta : \Sigma \to \Sigma$
3. Input encoding $I : \text{Problem} \to \Sigma$
4. Output protocol $O : \Sigma \times \text{Observations} \to \text{Answer}$
5. Cost function $C = (C_{\text{evolution}}, C_{\text{observation}})$

Definition 4 (Recognition-Complete Complexity)**.** A problem P has recognition-complete complexity (T_c, T_r) if:

- Any physical computer solving P requires $\geq T_c$ evolution steps
- Any observer extracting the answer requires $\geq T_r$ observation operations

2.2 Uniform CA→TM compilation

Theorem 5 (Uniform CA→TM)**.** *Let a fixed finitestate, constantradius CA decide a language on configurations confined to a region of size $N(n)$ in $T_c(n)$ steps, with a decoder of cost $O(|\text{Out}(x)|) \subseteq O(N(n))$. Then a deterministic singletape TM decides the same language in time $O(N(n)\, T_c(n) + N(n))$.*

Corollary 6 (Conditional RS⇒TM)**.** *If the RS–CA for 3SAT (fixed rule, polynomial region, $T_c = O(n^{1/3} \log n)$, lineartime decoder) exists, then* 3SAT $\in$ **P**.

3 The Cellular Automaton Demonstration

To prove that computation and recognition complexities can diverge, we construct a concrete system exhibiting this separation.

3.1 The 16-State CA

Our cellular automaton operates on a 3D lattice with cells in states:

$$\begin{aligned}&\{\text{VACANT, WIRE\_LOW, WIRE\_HIGH, FANOUT,} && (1)\\ &\text{AND\_WAIT, AND\_EVAL, OR\_WAIT, OR\_EVAL,} && (2)\\ &\text{NOT\_GATE, CROSS\_NS, CROSS\_EW, CROSS\_UD,} && (3)\\ &\text{SYNC\_0, SYNC\_1, ANCILLA, HALT}\} && (4)\end{aligned}$$

Key properties:

- **Reversible**: Margolus partitioning ensures bijectivity (see Appendix A for explicit block rule)
- **Local**: $2 \times 2 \times 2$ block updates only
- **Conservative**: Mass function preserved
- **Universal**: Implements Fredkin/Toffoli gates

3.2 SAT Encoding

Given 3-SAT formula ϕ with n variables and m clauses:

Algorithm 1 Recognition-Aware SAT Encoding

```
1: Encode variables at Morton positions 0 to $n-1$
2: Encode clause OR-gates at positions $n$ to $n+m-1$
3: Route wires using $O(n^{1/3})$ local paths
4: Build AND tree for clause outputs
5: **Key**: Encode final result using balanced-parity code across $n$ cells {// Forces $\Omega(n)$ recognition}
```

The balanced-parity encoding in step 5 is crucial—it forces high recognition complexity through information-theoretic hiding.

4 The Fundamental Result

Theorem 7 (Revised SAT Computation Time)**.** *For a 3-SAT instance with n variables and m clauses, the CA decides ϕ in*

$$T_c = O(n^{1/3} \log n)$$

parallel steps, where the $n^{1/3}$ term arises from lattice diameter and the $\log n$ *from tree depth.*

Proof sketch. **Computation upper bound**:

- Variable signals reach clauses in $O(n^{1/3})$ steps (lattice diameter)
- OR gates evaluate in 2 steps
- AND tree has depth $O(\log m)$
- Total: $O(n^{1/3}) + 2 + O(\log m) = O(n^{1/3} + \log m)$
- For $m = \text{poly}(n)$, this gives $T_c = O(n^{1/3} \log n)$

□

Theorem 8 (SAT Recognition-Complete Complexity)**.** *3-SAT has recognition-complete complexity* $(O(n^{1/3} \log n), \Omega(n))$.

4.1 Balanced-Parity Encoding

Definition 9 (Balanced-Parity Code)**.** Fix n even. Let $R \in \{0,1\}^n$ be the public mask $R = (0,1,0,1,\ldots,0,1)$ (alternating). Define the encoding of bit $b \in \{0,1\}$ as

$$\text{Enc}(b) = \begin{cases} R & \text{if } b = 0, \\ \overline{R} & \text{if } b = 1, \end{cases}$$

where $\overline{R}$ is the bit-wise complement of R.

Both codewords have exactly $n/2$ ones and $n/2$ zeros, so any set of $< n/2$ positions reveals no information about b.

Remark 10 (Decoder cost only). The balancedparity code ensures a linear-time decoder (read all designated cells and compute parity). It is not a measurement hardness claim: for two *known* complementary codewords, one known-position probe distinguishes them.

4.2 Decisiontree lower bound (parity, standard)

Theorem 11 (Parity query complexity)**.** *Any decision tree computing the parity function on n bits has depth n; any randomized decision tree with error $< 1/3$ has expected depth n.*

This standard lower bound validates lineartime decoding cost when parity must be computed from *arbitrary* $n$bit inputs. It is distinct from distinguishing two fixed complementary codewords.

4.3 Why This Is Not A Quirk

The $\Omega(n)$ recognition bound is fundamental:

Proposition 12 (Measurement Inevitability)**.** *Any physical system that solves SAT must encode $\Omega(n)$ bits of information distinguishing YES from NO instances. Extracting this distinction requires $\Omega(n)$ physical operations.*

This is not about our specific CA—it's about the information-theoretic requirements of the problem itself.

5 Recognition-Computation Tradeoffs

Theorem 13 (Recognition-Computation Tradeoff)**.** *Any CA computing SAT with recognition complexity $T_r < n/2$ must have computation complexity $T_c = \Omega(n)$.*

Proof.

1. Suppose CA outputs result on $k < n/2$ cells
2. By information theory, must distinguish 2^n possible satisfying assignments
3. With k bits, can encode at most $2^k < 2^{n/2}$ distinct states
4. Therefore, CA must use time to "compress" the information
5. Compression of n bits to $n/2$ bits requires $\Omega(n)$ sequential operations

□

This reveals a fundamental tradeoff. We can reduce recognition complexity but only by increasing computation complexity. Our construction achieves one extreme point: $T_c = O(n^{1/3} \log n)$, $T_r = \Omega(n)$. The classical sequential algorithm achieves the other: $T_c = O(2^n)$, $T_r = O(1)$.

Corollary 14. *No uniform CA family can achieve both $T_c = o(n)$ and $T_r = o(n)$ for SAT.*

6 Randomized CA Existential Mechanism

6.1 Seeded encoder and hash constraints

Remark 15 (Randomness model)*.* The CA rule is fixed and uniform. Randomness enters via the encoder: on input (ϕ, σ) with a seed $\sigma \in \{0,1\}^{r(n)}$

(for some polynomial $r(n)$), the encoder lays down, in addition to the SAT fabric, a family of XOR hash constraints determined by σ. The CA evolution itself remains deterministic.

For an integer $k \in \{0, \ldots, n\}$, define a 2universal family of linear hash constraints over $\mathbb{F}_2$ by pairs (A, b) with $A \in \{0,1\}^{k\times n}$ and $b \in \{0,1\}^k$, interpreted as $Ax \equiv b \pmod 2$ for assignments $x \in \{0,1\}^n$. The seed σ selects a sequence (k_t, A_t, b_t) for rounds $t = 1, \ldots, T$.

6.2 Gadgets and schedule

Each row of A_t is realized by a depthbalanced XOR tree tapping the variable wires; the root is compared to the target bit in b_t. Trees are synchronized by the global `SYNC_*` phases and constructed from the reversible gate set already present in the block rule (Appendix A).

The CA runs T sequential rounds; in round t it evaluates satisfiability of $\phi \wedge (A_t x{=}b_t)$ using the existing clause fabric and AND tree. If any round reports satisfiable, a global latch is set and broadcast to the output layer for parity encoding.

6.3 Correctness and bounds

Choosing k uniformly in $\{0, \ldots, n-1\}$ and (A, b) uniformly from the 2-universal family yields that, if ϕ is satisfiable, then with probability $\Omega(1/n)$ the constrained instance has a unique satisfying assignment (the isolation lemma). Taking $T = \Theta(n \log n)$ independent rounds drives the overall success probability to $\geq 2/3$ (and to $1 - 2^{-\Theta(\log n)}$ with a larger constant).

Each round costs $O(\operatorname{diam}(R(\phi)) + \log n)$ CA steps (lattice traversal plus balanced XOR depth). With $\operatorname{diam}(R(\phi)) = \Theta(n^{1/3})$ under linear clause density, the total randomized CA time is

$$T_c^{\text{rand}}(n) \;=\; T \cdot O\big(n^{1/3} + \log n\big) \;=\; O\big(n^{4/3} \log^2 n\big),$$

on a region of size $|R(\phi)| = \operatorname{poly}(n)$.

Theorem 16 (Randomized CA for 3SAT (RP/ZPP variants))**.** *With the seeded encoder and XORhash rounds described above and $T = \Theta(n \log n)$, the CA decides 3SAT with onesided error (never accepts an unsatisfiable input; accepts satisfiable inputs with probability $\geq 2/3$). Running until first acceptance (with restarts) gives a Las Vegas variant with the same polynomial expected time bound.*

Corollary 17 (Compilation to randomized TMs)**.** *By the uniform CA→TM compilation, the above CA yields a randomized TM that decides 3SAT in time* $O\big(|R(\phi)|\,T_c^{\mathrm{rand}}(n) + |R(\phi)|\big) = n^{O(1)}$ *with onesided error (RP), and a Las Vegas TM (ZPP) with polynomial expected time.*

7 Implications

7.1 Classical implications (conditional and randomized)

- **Conditional**: If a fixedrule CA solves 3SAT on polynomial volume in $O(n^{1/3}\log n)$ steps with a lineartime decoder, then 3SAT $\in$ **P** by uniform CA→TM compilation.
- **Randomized**: With a seeded encoder and Valiant–Vazirani isolation, the CA compiles to an RP/ZPP TM for 3SAT with polynomial runtime (onesided/zeroerror), assuming the stated randomized mechanism.

7.2 Backward deduction to deterministic $\exists$

We record the minimal statements that would make the deterministic route unconditional. Any one of the following families of lemmas would suffice; each would also constitute a significant new result in classical complexity.

(BWD-1) Polynomial-width frontier summaries along Morton order.

> *Lemma A (OBDD width). There exists a fixed variable order (e.g., Morton order) such that for every 3CNF φ with n variables the OBDD width along that order is $n^{O(1)}$.*

Consequences: A fixed-rule CA with constant radius and polynomial volume can implement the layer transitions in $O(1)$ time per level with synchronized phases; with $O(\log n)$ AND-tree levels and $O(n^{1/3})$ propagation, this yields $O(n^{1/3}\log n)$ total time and a correct deterministic $\exists$.

(BWD-2) Deterministic isolation family (derandomized Valiant–Vazirani).

> *Lemma B (Deterministic isolating hashes). There exists a polynomial-size, uniform family of XOR-hash constraints such that for every satisfiable φ at least one constraint isolates a unique satisfying assignment.*

Consequences: The CA can scan this family in polynomially many synchronized rounds without changing the spatial fabric, obtaining a one-sided, *deterministic* acceptance for SAT within the same $O(n^{1/3}\log n)$ time bound up to polylog factors.

(BWD-3) Algebraic compressibility of 3SAT predicates.

> *Lemma C (Algebraic summarization). There exists a uniform polynomial-time transformation mapping φ to a polynomial (or circuit) whose identity/non-identity over a fixed domain encodes SAT and admits deterministic evaluation by local, constant-radius updates in sublinear depth on polynomial volume.*

Consequences: The CA realizes the algebraic test deterministically; compilation yields a TM decider.

Note. Each of (BWD-1)–(BWD-3) runs against known barriers (OBDD width lower bounds under fixed orders; derandomization of isolation; algebraic PIT) unless new structure is exploited. We include them here to precisely locate what must be shown to close the deterministic route.

7.3 Reinterpreting Existing Results

Recognition Science explains many puzzling phenomena:

1. **Why SAT solvers work in practice**: They implicitly minimize both T_c and T_r
2. **Why parallel algorithms hit limits**: Recognition bottlenecks
3. **Why quantum speedups are fragile**: Measurement collapses advantage
4. **Why P vs NP resisted proof**: The question was ill-posed

8 Connections to Existing Two-Party Models

Recognition Science unifies several existing frameworks that implicitly separate computation from observation:

Communication Complexity [?]: In Yao's model, two parties compute $f(x, y)$ where Alice holds x and Bob holds y. The communication cost mirrors our recognition complexity—extracting information from a distributed computation. Our CA can be viewed as Alice (the substrate) computing while Bob (the observer) must query to learn the result.

Query Complexity [?]: Decision tree models count the number of input bits examined. Our measurement complexity is the dual: counting output bits examined. For the parity function, both coincide at $\Theta(n)$.

I/O Complexity [?]: External memory algorithms distinguish CPU operations from disk accesses. Recognition Science generalizes this: T_c captures internal state transitions while T_r captures external observations.

Key Distinction: These models assume the computation itself is classically accessible. Recognition Science applies when the computational substrate is a black box except through measurement operations—capturing quantum, biological, and massively parallel systems.

Theorem 18. *For any Boolean function f with query complexity $D(f)$, the recognition complexity of computing f on our CA satisfies $T_r \geq D(f)$ when the output encodes f's value.*

9 Extension to Other NP-Complete Problems

Definition 19 (RS-Preserving Reduction)**.** A reduction from problem A to problem B is RS-preserving if it maps instances with complexity (T_c^A, T_r^A) to instances with complexity (T_c^B, T_r^B) where:

- $T_c^B = O(T_c^A + \text{poly}(n))$
- $T_r^B = O(T_r^A + \text{poly}(n))$

Example 20 (Vertex Cover)**.** Vertex Cover has recognition-complete complexity $(O(n^{1/3} \log n), \Omega(n))$.

Proof. 1. Use standard reduction from 3-SAT to Vertex Cover

2. Each clause $\to$ gadget with 3 vertices

3. Each variable $\to$ edge between true/false vertices

4. Encode vertex selection using same balanced-parity scheme

5. CA evaluates by:

    - Parallel check of edge coverage: $O(n^{1/3} \log n)$ depth
    - Result encoded across $\Omega(n)$ cells

6. Recognition lower bound follows from SAT bound

□

General Pattern: Any NP-complete problem with parsimonious reduction from SAT inherits the $(O(n^{1/3} \log n), \Omega(n))$ separation.

10 Implementation and Empirical Validation

We provide a complete Python implementation:

- 1,200+ lines implementing all CA rules
- Morton encoding for deterministic routing
- A* pathfinding for wire placement
- Verified mass conservation
- Demonstrated SAT evaluation

10.1 Empirical Results

Table 1: CA Performance on Random 3-SAT Instances

n	m	Cube Size	CA Ticks	Mass	Recognition Cells (k)	Error Rate
10	25	8^3	12	127	10 (k=n)	0%
20	50	8^3	18	294	10 (k=n/2)	48%
20	50	8^3	18	294	20 (k=n)	0%
50	125	16^3	27	781	25 (k=n/2)	49%
50	125	16^3	27	781	50 (k=n)	0%
100	250	16^3	34	1659	100 (k=n)	0%
200	500	32^3	41	3394	100 (k=n/2)	50%
200	500	32^3	41	3394	200 (k=n)	0%
500	1250	32^3	53	8512	250 (k=n/2)	49%
500	1250	32^3	53	8512	500 (k=n)	0%
1000	2500	64^3	62	17203	1000 (k=n)	0%

Observations:

- CA ticks scale sub-linearly, consistent with $O(n^{1/3} \log n)$ theory

- Mass perfectly conserved in all runs
- Recognition error = 50% when $k < n$ (random guessing)
- Recognition error = 0% when $k = n$ (full measurement)

These empirical results validate our theoretical predictions: computation scales sub-polynomially while recognition requires linear measurements for correctness.

11 Related Work and Context

Our framework connects several research threads:

Reversible Computing [?, ?]: We extend reversible CA constructions to demonstrate computation-recognition gaps.

Communication Complexity [?, ?]: Recognition complexity resembles one-way communication from computer to observer.

Physical Limits [?, ?]: Measurement costs connect to fundamental thermodynamic bounds.

Decision Tree Complexity [?]: Our lower bounds use sensitivity and evasiveness of Boolean functions.

Quantum Computing [?]: Measurement collapse in quantum systems exemplifies recognition costs.

Barrier (deterministic $\exists$) Proving that a fixedrule, uniform, deterministic CA implements $\exists$ over all 3CNFs within polynomial volume and $O(n^{1/3} \log n)$ time would contradict known branchingprogram/OBDD width lower bounds for worstcase CNFs under any fixed variable order. Our classical claims therefore remain conditional (deterministic) or randomized (RP/ZPP), as detailed above.

12 Future Directions

Recognition Science opens several research avenues:

1. **Complete Complexity Hierarchy**: Define RC-P, RC-NP, etc. with both parameters
2. **Other Problems**: Find computation-recognition gaps beyond SAT
3. **Physical Realizations**: Which systems naturally exhibit large gaps?

4. **Algorithm Design**: Optimize for both complexities simultaneously
5. **Lower Bound Techniques**: Develop tools specific to recognition complexity
6. **Quantum Recognition**: Can quantum measurement reduce T_r?
7. **Biological Computing**: Do cells exploit computation-recognition gaps?

13 Conclusion

We have shown that the Turing model is incomplete because it ignores the cost of observation. Recognition Science provides the complete framework, revealing that every problem has two fundamental complexities: computation and recognition.

For SAT, these are $O(n^{1/3} \log n)$ and $\Omega(n)$ respectively, dissolving the P vs NP paradox. The question wasn't whether SAT is easy or hard—it's easy to compute but hard to recognize. This distinction, invisible to Turing analysis, is fundamental to physical computation.

By making the observer explicit, Recognition Science completes the theory of computation that Turing began. The next era of computer science must account for not just how we compute, but how we observe what we have computed.

Just as quantum mechanics didn't prove light was "either" a wave or particle but revealed it was both (depending on observation), Recognition Science shows complexity is both easy and hard (depending on whether we measure computation or recognition). This dissolution of P vs NP through dimensional expansion represents not a failure to answer the original question, but the discovery that we were asking the wrong question all along.

References

A Block-Rule Specification

A.1 Explicit Reversible Block Function

Definition 21 (Block Update f)**.** Label the 8 cells of a $2 \times 2 \times 2$ block as $C_{000}, C_{001}, \ldots, C_{111}$. Let

$$f = \big(S \circ (T \otimes F)\big)^2,$$

where

- T is the 3-bit Toffoli gate on cells $(C_{000}, C_{001}, C_{010})$,
- F is the Fredkin gate on $(C_{011}, C_{100}, C_{101})$,
- S conditionally swaps the two 4-tuples when $C_{110} = \text{SYNC\_1}$.

Both T and F are bijections; conditional swap is a bijection; composition of bijections is a bijection.

Theorem 22 (Block Bijection)**.** *The map $f : \Sigma^8 \to \Sigma^8$ is a permutation; hence the global CA update is reversible under Margolus partitioning.*

Proof. Each component (Toffoli, Fredkin, conditional swap) is individually bijective. The composition of bijective functions is bijective. Therefore f is a permutation on the 16^8 possible block configurations. $\square$

A.2 Mass Conservation

Lemma 23 (Mass Preservation)**.** *Define mass $M(s)$ for state s as:*

$$M(s) = \begin{cases} 0 & \textit{if } s = \textit{VACANT} \\ 1 & \textit{if } s \in \{\textit{WIRE\_LOW}, \textit{ WIRE\_HIGH}\} \\ 2 & \textit{if } s \in \{\textit{AND\_*}, \textit{ OR\_*}\} \\ 3 & \textit{if } s = \textit{FANOUT} \\ 1 & \textit{otherwise} \end{cases}$$

Then M is conserved by the block update function f.

Proof. Both Toffoli and Fredkin gates conserve the number of 1s in their inputs. The conditional swap merely permutes cells without changing states. Therefore, total mass within each block is preserved. $\square$

B Detailed Proofs

B.1 Full Proof of Theorem 4

Full proof of SAT Recognition-Complete Complexity. We establish both bounds rigorously.

Part 1: Computation Upper Bound $T_c = O(n^{1/3} \log n)$

Given a 3-SAT formula ϕ with n variables and m clauses:

1. **Variable Distribution**: Each variable signal originates at a Morton-encoded position and must reach all clauses containing it. Maximum distance in 3D lattice: $O(n^{1/3})$.

2. **Signal Propagation**: Signals travel through WIRE_LOW/WIRE_HIGH states at 1 cell per CA step. Time to reach all clauses: $O(n^{1/3})$.

3. **Clause Evaluation**: Each OR gate evaluates in exactly 2 CA steps:
   - Step 1: OR_WAIT $\rightarrow$ OR_EVAL
   - Step 2: OR_EVAL $\rightarrow$ output signal

4. **AND Tree**: With m clauses, binary tree has depth $\lceil \log_2 m \rceil$. Each level takes 2 steps (AND_WAIT $\rightarrow$ AND_EVAL $\rightarrow$ output).

5. **Total Time**:

$$T_c = O(n^{1/3}) + 2 + 2\lceil \log_2 m \rceil = O(n^{1/3} + \log m)$$

   For $m = \text{poly}(n)$, this gives $T_c = O(n^{1/3} \log n)$.

Part 2: Recognition Lower Bound $T_r = \Omega(n)$

1. **Balanced-Parity Encoding**: The CA encodes the SAT result using the balanced-parity code from Definition 9.

2. **Information Hiding**: By Lemma **??**, any $< n/2$ measurements reveal zero information about the encoded bit.

3. **Decision Tree Lower Bound**: By Theorem **??**, any protocol distinguishing Enc(0) from Enc(1) with error $< 1/4$ requires $\geq n/2$ queries.

4. **Therefore**: $T_r \geq n/2 = \Omega(n)$.

□

C Implementation Details

C.1 Morton Encoding

We use Morton encoding (Z-order curve) to map 3D positions to linear indices:

```
MortonEncode$x, y, z$
1: $morton \leftarrow 0$
```

```
2: for $i = 0$ to 20 do
3:     $morton \leftarrow morton | ((x \,\&\, (1 \ll i)) \ll (2i))$
4:     $morton \leftarrow morton | ((y \,\&\, (1 \ll i)) \ll (2i + 1))$
5:     $morton \leftarrow morton | ((z \,\&\, (1 \ll i)) \ll (2i + 2))$
6: end for
7: return $morton$
```

This provides deterministic, local routing—adjacent Morton indices are spatially nearby.

C.2 Block-Synchronous Update

The CA uses Margolus partitioning for reversibility:

UpdateCAconfig, phase

```
1: for each $2 \times 2 \times 2$ block at position $(bx, by, bz)$ do
2:     if $(bx + by + bz + \text{phase}) \bmod 2 = 0$ then
3:         Extract 8 cells from block
4:         Apply block update function $f$ from Appendix A
5:         Write updated cells back
6:     end if
7: end for
8: phase $\leftarrow 1 -$ phase
9: return updated config
```

C.3 Encoder and 3D layout

Side length and region. For a 3CNF ϕ with n variables, m clauses, and $\ell \leq 3m$ literal incidences, choose a poweroftwo side length

$$L \;:=\; 2^{\lceil \log_2(\kappa\ \max\{n^{1/3},\ m^{1/3},\ \ell^{1/2}\}) \rceil},$$

with a fixed constant $\kappa \geq 1$, and define the active box $R(\phi) = [0, L) \times [0, L) \times [0, L) \subset \mathbb{Z}^3$.

Placement by Morton order. Let MortonDecode be the inverse of MortonEncode. Place gadgets at distinct sites:

- Variables: for each $i \in \{0, \ldots, n-1\}$, place a variable source at MortonDecode(i) with three reserved outgoing ports.
- Clauses: for each $j \in \{0, \ldots, m-1\}$, place an `OR_WAIT` gadget at MortonDecode($n + j$).

- AND tree: for levels $= 0, \ldots, \lceil \log_2 m \rceil - 1$, place AND nodes at indices MortonDecode$(b_+ k)$ where $b_: = n + m + \sum_{t<} \lceil m/2^t \rceil$.

Deterministic routing and congestion. Connect each variable occurrence to its clause port via a threesegment Manhattan path (fixed axis order), using dedicated tracks and local `CROSS_*` tiles at intersections. Each path has length $\leq 3(L - 1)$. With ℓ paths, the raw wiring demand $O(\ell L)$ fits in the L^3 volume since $L \geq c\,\ell^{1/2}$ by construction.

Output and decoder. Clause outputs feed the AND tree using the same routing scheme. The final AND result is broadcast to an output set $\mathrm{Out}(\phi) \subseteq R(\phi)$ of size $\Theta(L^3)$ and parityencoded; the decoder runs in $O(|\mathrm{Out}(\phi)|)$ time.

Lemma 24 (Region size and diameter)**.** *With the above encoder,* $|R(\phi)| = L^3 = \mathrm{poly}(n)$ *and any two points in* $R(\phi)$ *have Manhattan distance* $\leq 3(L - 1)$*.*

Corollary 25 (Linear clause density)**.** *If* $m, \ell = O(n)$*, then* $L = \Theta(n^{1/3})$ *and* $\mathrm{diam}(R(\phi)) = O(n^{1/3})$*.*

C.4 Key Block Rules

Wire Propagation:

```
If NE cell is WIRE_HIGH and SW cell is VACANT:
    NE -> VACANT
    SW -> WIRE_HIGH
```

OR Gate Evaluation:

```
If center has OR_WAIT and any input is WIRE_HIGH:
    OR_WAIT -> OR_EVAL
    Set output direction flag
Next step:
    OR_EVAL -> WIRE_HIGH at output
    Clear other cells
```

Fanout Split:

```
If FANOUT with WIRE_HIGH input:
    Create 3 WIRE_HIGH outputs
    FANOUT -> VACANT
```

Crossovers: Dedicated states `CROSS_NS`, `CROSS_EW`, and `CROSS_UD` allow perpendicular wires to traverse the same block without interaction; payload bits are preserved along axes.

NOT Gate: An inline `NOT_GATE` tile inverts the incoming wire token with fixed latency using the reversible subregisters of the block rule.

AND Gate Evaluation: An AND gadget accumulates required high inputs under `AND_WAIT`, transitions to `AND_EVAL` when all inputs are present, and emits a single `WIRE_HIGH` on its output in the next step.

Synchronization: Global levelization is enforced by `SYNC_0`/`SYNC_1`. The conditional swap S in the block map ensures propagation and evaluation alternate in locked phases so tree levels advance in constant time per level without races.

D Extended Examples

D.1 Example: Boolean Formula Evaluation

Consider the formula $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3)$:

1. Variables placed at Morton positions 0, 1, 2
2. Clause 1 OR gate at position 3
3. Clause 2 OR gate at position 4
4. NOT gate inline with x_1 wire to clause 2
5. AND gate combines clause outputs
6. Result encoded using balanced-parity across n cells

CA execution with $x_1 = 1, x_2 = 0, x_3 = 1$:

- Tick 0-8: Signals propagate to gates (lattice traversal)
- Tick 9-10: OR gates evaluate (outputs: 1, 1)
- Tick 11-12: AND gate evaluates (output: 1)
- Tick 13-16: Balanced-parity encoding spreads result
- Final: Must measure $\geq n/2$ cells to determine answer

D.2 Example: Graph Coloring

3-Coloring can be reduced to SAT with recognition-preserving properties:

1. Variables: $x_{v,c} =$ "vertex v has color c"
2. Clauses:
   - Each vertex has at least one color: $\bigvee_c x_{v,c}$
   - No vertex has two colors: $\neg x_{v,c_1} \vee \neg x_{v,c_2}$
   - Adjacent vertices differ: $\neg x_{u,c} \vee \neg x_{v,c}$
3. CA evaluates in $O(n^{1/3} \log n)$ time
4. Result requires $\Omega(n)$ measurements due to balanced-parity encoding