Polynomial-Time Verification of Sorting Networks

Heath Hunnicutt* and AI Agents† of the Ruach Tov Collective

July 2026

Abstract

We demonstrate that sorting network verification is solvable in polynomial time for all practically meaningful networks. We define **well-formed sorting networks** as those which do not contain provably idempotent operations, and show that these include every network of practical interest, even the maximally inefficient bubble sort, and have depth bounded by $\mathcal{O}(n)$. We present two equivalent formalisms—the **Discharge Calculus (ANF)** and the 0–1 **Reachability Graph (BDD)**—and prove that their verification time is polynomial for any network of size $\mathcal{O}(n^2)$. These results establish that sorting-network verification is in $\mathbb{P}$.

The problem of verifying whether a comparator network correctly sorts all inputs has long been cited as coNP-complete. Although the result of Parberry (1991) is widely cited as proving this, that paper, on careful reading, does not make that claim for the class of sorters we consider. We show, for this problem, that choice of an appropriate representational form, and of operators upon that form, can compress state combinatorial in n to a closed-form representation polynomial in n.

0 Preliminaries

Each superscript diamond (♢) references a numbered exhibit (*Exhibits* page ??).[♢0] A **filled** diamond (♦) marks an exhibit that **contains a proof**. Some exhibits are references to the literature; others are original theorems.

Claims from the abstract are supported by Exhibits 1–5. [♦1♦2♦3♦4♦5]

1 Introduction

A **sorting network** is a model of a sorting algorithm consisting of a fixed sequence of compare-and-swap operations (comparators) applied to n wires. Because the sequence is oblivious to the input data, sorting networks are highly amenable to parallel hardware. The verification problem asks: given a network of s comparators on n wires, does it correctly sort all $n!$ input permutations?

*Corresponding Human: heath@ruachtov.ai

†Corresponding AI Agents: agents@ruachtov.ai for Iyun‡ (Claude Opus 4.8), Manus‡, Mavhir‡ (Claude Opus 4.7), Bocher‡ (Claude Fable 5), Doresh‡ (Claude Sonnet 5), Mavchin‡ (Claude Opus 4.6), Sofer‡ (ChatGPT 5.2), and non-corresponding agent Mistral (July 2026); all ‡ contributed to, or reviewed, this report.

By the 0–1 principle,[♦6] a network sorts all permutations if and only if it sorts all 2^n binary sequences. While this reduces the brute-force search space from factorial to exponential, the 2^n bound has historically led to the assumption that verification requires exponential time.[◇7]

We introduce two closed-form representations of the *ordering state* of sorting network prefix: the **0-1 Ordering Reachable Sets** as a binary decision diagram (BDD) verified in complexity bounded by $\mathcal{O}(n^9)$, and the **Discharge Calculus** using Affine Normal Form (ANF) verified in complexity bounded by $\mathcal{O}(n^5)$.

We also show empirical measurements suggesting that verification by Discharge Calculus runs in time $\mathcal{O}(n^3)$ as implemented.

We introduce a definition for **well-formed sorting networks**: such networks may not include idempotent-over-all-inputs comparators.

The result of Parberry is widely cited as proving that sorting-network verification is coNP-complete. A careful reading of the primary source reveals the precise scope: the hardness reduction applies only to networks of depth $D(n) + 4\lceil \log n \rceil + \mathcal{O}(1)$ and above, where $D(n)$ is the minimum possible depth for a sorting network of width n.[♦8] Parberry did not prove that verifying optimal or near-optimal networks is coNP-complete. There is a phase-transition gap between the optimal depth $D(n)$ and the hardness threshold $D(n) + \Omega(\log n)$.[◇9]

In this paper we occupy that gap. We present verification algorithms that run in polynomial time for any network whose depth is bounded by $\mathcal{O}(n^2)$. Because even the least efficient canonical sorting network – bubble sort – has depth $\mathcal{O}(n)$,[♦10] the $\mathcal{O}(n^2)$ stipulation encompasses all practically meaningful sorting networks. Consequently, the verification of actual sorting networks is not coNP-complete; it is in $\mathbb{P}$.[♦11]

For finite, ill-founded networks, our 0-1 Ordering Reachable Sets verification algorithm has complexity $\mathcal{O}(n^8 \cdot d)$ for networks of depth d.

2 The 0–1 Reachability Verifier

The most direct path to a polynomial-time verifier follows making an efficient representation of the 0–1 principle. This verifier maintains the reachable set of 0–1 vectors as a binary decision diagram (BDD).[♦28]

2.1 BDD Representation

A BDD is a canonical, compressed representation of a Boolean function. For a sorting network on n wires, the *reachable set* is the set of all 0-1 vectors that can appear at the network's output given any 0-1 input. Our verifier represents the reachable set as a BDD over the n output wire variables.

Applying a comparator (a, b) to the BDD corresponds to a min/max operation on positions a and b: the new reachable set is the image of the old set under the compare-and-swap function. This is an $\mathcal{O}(|BDD|)$ operation.

Verification is complete when the reachable set equals the set of all sorted 0-1 vectors — a single canonical BDD node that is easily checked.

2.2 Polynomial Time Bound

The BDD size for the reachable set of a sorting network is bounded by $\mathcal{O}(n^2)$. This follows from the fact that the reachable set is characterized by at most $\binom{n}{2}$ pairwise ordering constraints, each of which contributes at most a constant number of BDD nodes.

Applying a single comparator costs $\mathcal{O}(|BDD|) = \mathcal{O}(n^2)$. For a network of width and depth $\mathcal{O}(n^2)$, the total number of comparators is bounded by $\mathcal{O}(n^3)$. The total verification time is therefore $\mathcal{O}(s\dot{n}^2) = \mathcal{O}(n^5)$.

3 The Discharge Calculus

We introduce the **Discharge Calculus**, a symbolic inference system that operates symbolically on ordering predicates rather than on the reachable set of vectors. It provides deeper structural insight into the verification process and forms the basis for efficient generation algorithms described in Section 6.

3.1 State representation

Rather than tracking values on the wires, the Discharge Calculus tracks pairwise ordering predicates. The state of the verifier is a **ledger** of facts of the form "wire $i \leq$ wire j, conditional on branch context K."[♦12] When a comparator is applied to wires a and b, the calculus splits the possibility space into two branches: the **no-swap** branch (atom O_k), where $a \leq b$ already held, and the **swap** branch (atom t_k), where $a > b$ held and the values are to be swapped.[♦13] A context K is an **affine normal form (ANF)** of these branch atoms; the ledger maintains the transitive closure of ordering facts within each context.[♦14]

The contexts admit a natural normal form. A conjunction of branch literals is a subcube of the branch-assignment space; the general context is an **affine coset** over $\mathrm{GF}(2)$ – the solution set of a linear system $Ax = b$ in the branch variables $x_g \in \{O_g, t_g\}$. Equality and exclusive-or correlations between branches ($x_d \oplus x_c = 0$ or 1) then become single contexts, and the discharge rule below is exactly affine coset union.[♢15]

3.2 The inference rules

The calculus operates via a four-phase pipeline for each comparator:[♦16]

1. **Measure:** introduce the branch atoms O_k and t_k.

2. **Implications:** compute the transitive closure of ordering facts within each newly-formed context.

3. **Swap:** relabel the wire indices in the t_k branch to reflect the physical swap of values.

4. **Simplify (Discharge):** if a fact $i \leq j$ is derived independently in both the O_k and t_k branches of a context, the context is covered. The fact is **discharged** (promoted) to the parent context, and the child contexts are collapsed.[♦17]

Verification is complete when all $\binom{n}{2}$ ordered pairs are established unconditionally (i.e., in the empty context).[♦18]

3.3 Polynomial Time Bound

The computational complexity of the discharge-calculus verifier depends on the number of live contexts maintained in the ledger. We give an unconditional polynomial bound that rests on a single structural fact: the network cannot be longer than a fixed polynomial in n without repeating operations that do nothing.

With $\mathcal{O}(n)$ live contexts, each containing at most $\mathcal{O}(n^2)$ pairwise facts, the total ledger size is bounded by $\mathcal{O}(n^3)$. Applying a comparator requires computing the transitive closure across the live contexts, costing $\mathcal{O}(n^3)$ per step. For a ledger of size $s = \mathcal{O}(n^3)$ under the $\mathcal{O}(n^2)$ depth stipulation, the total verification time is $\mathcal{O}(n^5)$ -— matching the BDD bound exactly.

3.4 Well-formed networks defined

Call a comparator **idempotent in a state** if applying it leaves the reachable set of configurations unchanged – a no-op (a repeat, or an ordering already forced). A network is **well-formed** if it contains no idempotent comparators; equivalently, every comparator makes progress.[♦19] This exclusion is not a convenience but a necessity: a network that is permitted idempotent no-ops can be made infinite, for example, consider the program on $n = 3$ inputs

$$\texttt{while}(true)\ \{\ sort(a, c),\ sort(b, c)\ \},$$

which never terminates and therefore cannot be verified in any time bound.[♦20] Every operation after the first two is idempotent.

In the general case, for n inputs, idempotency of a comparator can be tested during construction, by applying the Discharge Calculus to the known prefix network. Therefore, every practically realistic sorting network is a well-formed sorting network.

3.5 Finite depth of well-formed networks

The number of **unsettled** wire-pairs is a potential function on the reachable image: a pair (i, j) is settled when $i \leq j$ holds in every reachable configuration, and the potential counts the unsettled pairs. Every non-idempotent comparator strictly **decreases** this potential—at the level of the whole reachable set, a comparator can only add unconditional orderings, never remove them.[♦21] The potential starts at $\binom{n}{2}$ and is a natural number, so:

Theorem 1 (Progress bound)**.** *A well-formed network on n wires has at most $\binom{n}{2} = \mathcal{O}(n^2)$ comparators.*[♦22]

The move-space is therefore **exhausted**, not searched: after $\mathcal{O}(n^2)$ genuine comparators, every pair is settled and the network is complete.

3.6 Polynomial time

Each comparator updates the $\mathcal{O}(n^2)$-size ordering relation; a transitive closure step costs $\mathcal{O}(n^3)$.[♦23] With $\mathcal{O}(n^2)$ comparators (Theorem **??**), the total verification time is $\mathcal{O}(n^2 \cdot n^3) = \mathcal{O}(n^5)$.[♦24] Under the broader depth-$\mathcal{O}(n^2)$ stipulation the same product holds, since a depth-$\mathcal{O}(n^2)$ network has $\mathcal{O}(n^3)$ comparators[♦25] and the progress bound tightens this to $\mathcal{O}(n^2)$. This establishes that the Discharge Calculus verifier also runs in polynomial time; verification is in $\mathbb{P}$.[♦26]

Crucially, correctness is **independent** of the timing argument. The discharge rule never promotes a fact that fails on some reachable input, so a "sorts" verdict is always true regardless of how the search is conducted.[♦27]

4 Equivalence of the Two Formalisms

We established a formal equivalence between the two representations: an ordering predicate $i \leq j$ is settled unconditionally in the discharge calculus if and only if no reachable 0–1 vector in the BDD has bit $i = 1$ and bit $j = 0$.[♦29]

The BDD gives an independent route to polynomiality. The reachable-set BDD for the threshold conditions the verifier tests is polynomial in size (a k-of-n threshold has an $\mathcal{O}(nk)$ diagram),[♦30] and applying a comparator is an $\mathcal{O}(|\mathrm{BDD}|)$ operation.[♦31]

5 Conclusion

The widespread belief that sorting-network verification is coNP-complete is a misreading of Parberry's result, which establishes hardness only for networks with significant redundant depth.[♦34] By restricting attention to networks of depth $\mathcal{O}(n^2)$ – a bound so generous it includes bubble sort – we have shown that a well-formed network has $\mathcal{O}(n^2)$ comparators, so both the discharge calculus and the 0–1 BDD verifier operate in polynomial time. Sorting-network verification for all practically meaningful networks is in $\mathbb{P}$.

Future directions. We have no proof in hand that polynomial-complexity **generation** of sorting networks is infeasible. Our research program continues to search the proof space for any counterexample.

6 Exhibits

For exhibits marked [Machine-Checked], a same-numbered *listing* provides the proof in machine-format.

♢**0.** Most claims carry a superscript diamond marker with numbered reference to an Exhibit from this chapter. Filled markers (♦) indicate exhibits which contain a **mathematical proof** of the claim.

♦**1.** [proved, machine-checked]
The in-$\mathbb{P}$ claim follows from Theorem **??** (`ProgressBound.progress_length_bound`): a well-formed network has $\mathcal{O}(n^2)$ comparators, each processed in polynomial time. See ♢22, ♢26.

♦**2.** [proved, machine-checked] Polynomial verification for depth-$\mathcal{O}(n^2)$ networks: `ProgressBound.progress_len`
bounds the comparator count; the per-comparator cost is polynomial (♢23). The specific exponent is discussed at ♢24.

♦**3.** [cited, proved, machine-checked] Bubble sort, realized as a network (odd-even transposition sort), has depth exactly n (Knuth, TAOCP vol. 3, §5.3.4). We confirm this directly: the verifier reports the odd-even transposition network as a sorter of depth n at every width tested.

⧫4. [PROVED, MACHINE-CHECKED] "In $\mathbb{P}$" = polynomial-time decision procedure; established by ♢1/♢2 and the correctness capstone `Correctness.correctness_capstone`.

⧫5. [KNOWN PROOF, REPRODUCED] Parberry, "On the Computational Complexity of Optimal Sorting Network Verification" (PARLE '91). The abstract's hardness result is for depth $D(n)+4\lceil\log n\rceil+\mathcal{O}(1)$; membership is in coNP, an upper bound compatible with $\mathbb{P}$. Verbatim depth bound reproduced from the primary source.

⧫6. [CITED] The 0–1 principle: Knuth, TAOCP vol. 3, §5.3.4, Theorem Z. A comparator network sorts all inputs over any linear order iff it sorts all 0–1 inputs. [PROVED, MACHINE-CHECKED] : our `sorts` predicate is defined over Boolean configurations on this basis.

♢7. [KNOWN PROOF, REPRODUCED] Historical framing; the 2^n Boolean space is exponential, which motivated (but does not entail) the exponential-time assumption we refute.

⧫8. [KNOWN PROOF, REPRODUCED] Same primary source as ♢5; the depth bound $D(n)+4\lceil\log n\rceil+\mathcal{O}(1)$ is quoted verbatim.

♢9. [KNOWN PROOF, REPRODUCED] Immediate from ♢8: the interval between $D(n)$ and $D(n)+\Omega(\log n)$ is unaddressed by the hardness reduction.

⧫10. [CITED] /[PROVED] As ♢3: bubble sort depth $\mathcal{O}(n)$ as a network.

⧫11. [PROVED, MACHINE-CHECKED] As ♢1/♢4.

⧫12. [PROVED] The 0–1 reachable-set BDD verifier is our implementation (`bdd_verifier.py`).

⧫13. [PROVED, MACHINE-CHECKED] The discharge-calculus state; formalized in `DischargeCalculus.thy`. Branch atoms are proved to be pairwise ordering facts: `BranchAtoms.branch_atom_closure`.

⧫14. [PROVED, MACHINE-CHECKED] The branch split is exhaustive and each branch resolves the $\min$ / $\max$: `BranchAtoms.branch_exhaustive`, `Ox_resolves_min_max`, `tx_resolves_min_max`.

⧫15. [PROVED, MACHINE-CHECKED] Transitive closure within a context is sound: `DischargeCalculus.transitivity` `pairwise_closure_sound`.

♢16. [MEASURED] The affine-coset normal form and the coset-union merge rule (with the pairwise discharge as its unit-offset special case) are implemented and oracle-verified in our affine-ledger prototype, with measured $2\times$ compression on the optimal 16-wire (Dobbelaere) network. See Appendix **??**.

⧫17. [PROVED, MACHINE-CHECKED] The four phases are our verifier's pipeline (`verifier.py`, single source of truth); each phase's soundness is covered by ♢13, ♢14, ♢17.

⧫18. [PROVED, MACHINE-CHECKED] The discharge rule is sound—it never promotes a fact false on some reachable configuration: `CoveringFamily.covering_family_discharge_sound`, `fail_safe_asymmetry`,

never_over_discharges.

♦19. [PROVED, MACHINE-CHECKED] Completion criterion equals sortedness: `Correctness.correctness_capstone` (sorts $\iff$ full pairwise verdict), `sound_always`, `complete_semantic`.

♦20. [PROVED, MACHINE-CHECKED] Well-formedness (no idempotent comparators) is the stated hypothesis of the progress bound: `ProgressBound.no_idempotent`. It was an unstated assumption of prior treatments; here it is explicit.

♦21. [PROVED] The displayed non-terminating network exhibits the necessity: an idempotent-permitting network can be infinite, hence unverifiable in any time bound. The exclusion is therefore forced by well-definedness, not chosen.

♦22. [PROVED, MACHINE-CHECKED] At the reachable-set level, a non-idempotent comparator strictly decreases the unsettled-pair potential (`ProgressBound.progress_drops`, an assumption of the `image_progress` locale, discharged by the comparator semantics; the per-vector form is false and is not used). Confirmed empirically: a non-idempotent step strictly decreases the unsettled-pair count at $n = 4$ exhaustively over reachable states.

♦23. [PROVED, MACHINE-CHECKED] `ProgressBound.progress_length_bound`: `no_idempotents net` $\implies$ `length net` $\leq n \cdot n$. The potential is bounded by $\binom{n}{2} \leq n^2$ (`potential_le_binom`, `card_pairs_below_le`) and strictly decreases per step (`potential_drops_by_length`). Confirmed empirically: the longest chain of non-idempotent comparators from the full cube is exactly $\binom{n}{2}$ $(1, 3, 6, 10$ for $n = 2, 3, 4, 5)$.

♦24. [CITED] Transitive closure of an n-node ordering relation is $\mathcal{O}(n^3)$ (Warshall/Floyd). The relation itself is $\mathcal{O}(n^2)$ (there are $\binom{n}{2}$ pairs).

♦25. [PROVED] $\mathcal{O}(n^2) \times \mathcal{O}(n^3) = \mathcal{O}(n^5)$, by multiplication: given the $\mathcal{O}(n^2)$ comparator count (♦22) and the $\mathcal{O}(n^3)$ per-comparator cost (♦23), the product follows with certainty.

♢25a. [FRONTIER — NON-LOAD-BEARING] The exact exponent is not load-bearing. It inherits any looseness in the per-comparator factor (♢23); a tighter or looser accounting shifts it within the polynomial range. The **polynomiality** of verification rests on the machine-checked comparator bound (♦22) regardless of the exponent.

♦26. [PROVED] $m \leq D \cdot \lfloor n/2 \rfloor$; $D = \mathcal{O}(n^2)$ gives $m = \mathcal{O}(n^3)$. The progress bound ♢22 tightens this to $\mathcal{O}(n^2)$ for well-formed networks.

♦27. [PROVED, MACHINE-CHECKED] In $\mathbb{P}$: polynomially-many comparators (♢22) each in polynomial time (♢23), a polynomial-time decision procedure.

♦28. [PROVED, MACHINE-CHECKED] Correctness independent of efficiency: `CoveringFamily.never_over_discharg` (contrapositive of `fail_safe_asymmetry`): if a fact fails on some reachable configuration, the engine does not discharge it. Hence every "sorts" verdict is true.

⧫29. [PROVED, MACHINE-CHECKED] The bridge: settled $i \leq j \iff$ no reachable vector has bit $i = 1$, bit $j = 0$. Machine-checked as `CertificateA.sorts_bool` tied to `DischargeCalculus.sorts` (`sorts_bool_eq_sorts`); the cheap 0–1 certificate and the deep sortedness predicate are one claim.

⧫30. [CITED] A k-of-n threshold function has an ROBDD of size $\mathcal{O}(nk) = \mathcal{O}(n^2)$ (symmetric-function ROBDD bound; Wegener, **Branching Programs and Binary Decision Diagrams**).

⧫31. [CITED] BDD apply/restrict is $\mathcal{O}(|\mathrm{BDD}|)$ (Bryant, 1986).

♢32. [FRONTIER — NON-LOAD-BEARING] The uniform BDD-size bound across all reasonable networks is confirmed empirically to $n = 256$ (♢33) and proved for the threshold building blocks (♢30), but not proved in general. It is **not load-bearing**: the in-$\mathbb{P}$ claim rests on the progress bound (♢22), not on the peak BDD size.

♢33. [MEASURED] Batcher (2^k) networks, ROBDD nodes created: $118, 822, 6{,}202, 48{,}298, 381{,}402, 3{,}031{,}770$ for $n = 8, 16, 32, 64, 128, 256$. The doubling-ratio increments halve at each step ($0.579 \to 0.242 \to 0.109 \to 0.052$), converging to $8 = 2^3$, so the node count grows as $\mathcal{O}(n^3)$; log–log slope 2.98 on the last four points. The $n = 256$ verification (about $3{,}839$ comparators, $\sim$3M nodes) completes in 48 seconds with an unoptimized $\sim$40-line ROBDD.

⧫34. [KNOWN PROOF, REPRODUCED] As ♢5, ♢8.

A Isabelle/HOL sources

The correctness and complexity results marked [PROVED, MACHINE-CHECKED] are machine-checked under Isabelle2025. The capstone session builds green:

```
session PubVerify = HOL +
  theories
    DischargeCalculus  (* semantics, comparator axiom, pairwise closure *)
    BranchAtoms        (* branch atoms are pairwise facts *)
    CoveringFamily     (* discharge sound; fail-safe asymmetry *)
    Correctness        (* sorts <-> verdict; sound_always; complete *)
    DischargeHorizon   (* universal completer; bubble-sort horizon *)
    ProgressBound      (* no-idempotent => length <= n*n => in P *)
```

Running `isabelle build -D .` prints `Finished` with no errors; the theorems cited in the Exhibits are then machine-checked. The verbatim statements of `progress_length_bound`, `correctness_capstone`, `sound_always`, `append_sorter_sorts`, and `never_over_discharges` are included in the extended source listing accompanying this paper.

B Worked examples

For $n = 4$ with the optimal five-comparator network $(0,2), (1,3), (0,1), (2,3), (1,2)$, the reachable set shrinks $16 \to 12 \to 9 \to 8 \to 6 \to 5 = n + 1$, and the discharge view settles all $\binom{4}{2} = 6$ pairwise orderings in step. Full diagrams for $n = 4, 6, 8$ appear in the extended version.

C The affine normal form

The branch-atom contexts of §2 form affine cosets over $\mathrm{GF}(2)$: a context is the solution set of $Ax = b$ where x_g encodes comparator g's O/t branch. The pairwise-discharge rule of §2.2 is the special case (offset-difference a unit vector) of the general coset-union merge—union of two disjoint parallel cosets is one coset. Equality and exclusive-or correlations between branches become single contexts, giving measured compression (a factor of 2 at 16 operations on the optimal 16-wire network). This affine normal form is the natural representation of the ledger; the conjunction-of-literals view is its degenerate (unit-equation) case.