# A refutation of MathDB 377269, PAPER v1

Myeongjin Kang

AI-assisted research preprint. Independent checks are internal AI reviews, not external journal peer review. No worldwide-priority certification is asserted.

2026-11-15. This is a complete ordinary refutation and a standalone Lean source package, with a bounded kernel-reduced subset recurrence. All inherited and current process defects remain UNWAIVED.

## Literal problem and version boundary

For every integer k >= 3 and every family m_i >= 2, X is the directed Cartesian product of the directed cycles of lengths m_i. For any vertices u,v, the claim is

HamiltonianPath_X(u,v) iff d_X(u,v) = -1 modulo gcd(m_i).

Here d is shortest directed distance. We negate this exact universal iff and also the universal with the additional hypothesis u != v. We do not change its direction, domain, cycle convention or quantifiers. The owned raw MathDB record is 377269. The owned primary source is David Austin, Heather Gavlas and Dave Witte, *Hamiltonian Paths in Cartesian Powers of Directed Cycles*, arXiv:math/0110073v1 (5 October 2001), Conjecture 1.4, page 2. Its Definition 2.2, page 3, supplies the directed Cartesian product; Notation 2.5 supplies vertex-sequence paths; Proposition 2.6, page 4, identifies the standard-generator Cayley model. Source URLs: https://arxiv.org/abs/math/0110073 and https://mathdb.com/p/377269/hamiltonian-path-distance-congruence-conjecture-for-cartesia . No new request to either was made here. A live released statement field is supplied evidence, not a new live verification by this implementation.

Only this arXiv version and owned MathDB field are asserted. The broader historical manuscript, unrelated public works, registry proof bodies and author code are not imported as axioms. Independent ordinary/source/raw-live PASS and root bounded SUBSTANTIAL adoption are supplied evidence; none occurs in a Lean theorem premise. No fresh prior-art clearance or significance adjudication is claimed here.

## Source graph, paths and distance

Represent a cycle of length m by Fin m. Its sole forward successor has value (x+1) mod m. A product vertex is the dependent tuple (i : Fin k) -> Fin(m_i). An arc u->v means there is one index i for which v_i=(u_i+1) mod m_i and every other coordinate agrees. For m_i>=2 this step changes coordinate i, so it cannot be a loop. In length two, +1 modulo two supplies both directed arcs. Nothing reverses an arc in the length-three factor.

This tuple definition is exactly the iterated directed Cartesian product: the binary product definition says either the first coordinate advances and the second is fixed or conversely. Induction on the number of factors gives precisely one changing coordinate. Conversely a chosen changing coordinate selects the corresponding binary-product branch at each preceding factor. This establishes both arc directions for arbitrary k, rather than appealing to undirected adjacency.

A walk of n arcs is a function f : Fin(n+1)->Vertex with f(0)=u, f(n)=v, and an actual product arc f(t)->f(t+1) for each t<n. The ordered sequence f(0),...,f(n) is the corresponding finite vertex list. Conversely, a nonempty ordered vertex list gives this function by its indices. Injectivity of f means no repeated vertex. Surjectivity means every source vertex occurs. Thus Hamiltonian is existence of such an injective and surjective directed sequence. No certificate predicate substitutes for this definition; all of these requirements occur explicitly in the endpoint audit. A bijective f implies n+1 equals the cardinality of the actual vertex type.

The Lean distance is the least n for which such a directed walk exists, using Nat.find, with a zero default only for an unreachable pair. distance_minimum proves that when a walk exists, this number has a walk and is <= every walk

mathdb:2611.00108v1 17 Nov 2026

length. Directed cycle products with positive lengths are strongly connected: advance coordinate i by the residue of v_i-u_i modulo m_i, one coordinate at a time. This is also an ordinary construction of a finite walk. Deleting a repeated segment from a minimum walk gives a shorter walk with the same endpoints, so a minimum walk is a path. For this refutation no general connectivity or loop-deletion hypothesis is assumed: the witness has a directly proved one-arc walk, and the proved zero-walk iff equal-endpoints theorem excludes length zero. It therefore has shortest directed path distance exactly one. This is the permitted actual minimum-walk-distance route; no unproved coordinate-distance formula is used.

The modulus is the fold of Nat.gcd over the actual factor list, with terminal zero. Integer divisibility g | (d+1) is exactly d congruent to -1 modulo g. Standard associativity of gcd identifies the fold with gcd of all factors; for the witness its exact value is kernel-reduced to one.

## Ordinary refutation

Take m=(2,2,3), u=(0,0,0), v=(0,0,1). The factors satisfy all hypotheses. These endpoints are distinct and there is a direct vertical arc u->v. A zero-length walk has equal endpoints, so d(u,v)=1. The gcd is one, and the required congruence holds.

Suppose a Hamiltonian path exists. There are twelve product vertices, each visited exactly once, hence eleven arcs. Write A,B,C for the numbers of advances in coordinates one, two, three. Each arc advances exactly one coordinate, so A+B+C=11. The first two endpoints agree in their respective length-two factors, so A and B are even. Thus C is odd. The third endpoint is one above the starting height modulo three, so C=1 modulo three. With 0<=C<=11, these two requirements give C=1 or C=7. If C=1, the height sequence only reaches heights zero and one; every height-two vertex is missed. Consequently C=7 and A+B=4.

Project vertices onto their first two coordinates, called columns. Partition the path at its horizontal arcs into maximal consecutive intervals of constant column. Every interval is nonempty, every vertical arc lies inside one interval, and every horizontal arc is the boundary between consecutive intervals. Four horizontal arcs give five intervals I_0,...,I_4 with columns q_0,...,q_4. Consecutive columns are adjacent in the square of two bits. The endpoints imply q_0=q_4=00. Coverage of all twelve vertices requires all four columns. Therefore the four positions q_0,q_1,q_2,q_3 contain all four different columns: if two agreed, the five positions, with the repeated final q_0, would cover at most three columns. In particular, each intermediate column q_1,q_2,q_3 is visited in exactly one interval.

For each j=1,2,3, the sole interval I_j must cover all three heights of its column. No vertex repeats, so it has exactly three vertices and two internal vertical arcs. Four horizontal arcs change no height. The segment from the last vertex of I_0 through the first vertex of I_4 therefore has precisely six vertical advances, the two in each of I_1,I_2,I_3. Its first and last columns are both 00, and its height change is six, which is zero modulo three. Those two positions have the same actual vertex. They are different positions, separated by the three nonempty intermediate intervals. This contradicts nonrepetition. This argument covers both possible placements of the remaining vertical arc: the internal vertical counts in I_0,I_4 are nonnegative and sum to 7-6=1.

Hence no Hamiltonian path u->v exists, while the distance congruence holds. Instantiating the asserted universal iff at this valid product gives a contradiction. Since u!=v, the additionally distinct-endpoint universal also fails. No computational observation is a premise of this ordinary proof.

## Exact formal recurrence and semantic bridges

The formal proof uses the allowed subset-reachability fallback. It uses the actual dependent-product type V=Vertex lengths throughout, not a Fin12 assignment encoding. Define step(u,i) by incrementing coordinate i modulo its actual length. arc_step proves Arc(u,v) iff some step(u,i)=v, in both directions.

The state search(n,u,v,S) has the following recurrence:

- search(0,u,v,S) iff u=v and S is empty.
- search(n+1,u,v,S) iff for some i in Fin3, a=step(u,i) belongs to S and search(n,a,v,S minus {a}).

mathdb:2611.00108v1 17 Nov 2026

S is the set of vertices still to visit, not a guessed scalar count. sequence_search proves by induction on n that an injective actual directed sequence f with remaining set exactly its positive-index image yields the recurrence state. In the successor step, remove f(1), shift f by one index, and use injectivity to show that erasing f(1) removes exactly the first tail vertex. The n=0 case forces the remaining set empty.

search_sequence proves the converse, under u notin S. The zero state constructs the singleton sequence. In the successor case the induction hypothesis constructs the tail g starting at a, with remaining set S minus {a}. Prepend u using Fin.cases. Every g vertex lies in S; since u notin S this cannot repeat u. Injectivity of g handles pairs of tail positions. Arc(u,a) comes from arc_step; the other arcs are those of g. Membership in S is exactly being the initial tail vertex a or a positive-index tail vertex. This proves exact remaining-set coverage, endpoints and nonrepetition, not merely an overapproximation.

For S=univ minus {start}, hamiltonian_search first uses the bijection f to prove n=11 from the actual source cardinality 12. It proves S is exactly the tail image using f's surjectivity and injectivity, then applies sequence_search. Conversely search_sequence reconstructs the bijective source sequence by adding its start to the covered remaining set. recurrence_iff_hamiltonian is therefore the exact equivalence. negative_search proves the single required initial state false by ordinary decide; Lean's kernel reduces the finite recurrence. There is no native_decide, executable oracle, acquired validator or external table. Each branch chooses one of three directed successors and erases it before recursion. The depth is eleven; the complete ternary recursion has at most (3^12-1)/2=265720 nodes before membership pruning. This is a proved subset recurrence, not a 12! permutation sweep. No scientific generation/validation run was used (0/2; 0/180 scientific CPU seconds).

no_hamiltonian applies the negative recurrence theorem to hamiltonian_search. refutation and distinct_refutation discharge the two universal negations without premises. The endpoint audit expands the actual graph and path quantifiers, preserving arbitrary k,m,u,v in the negated statement. Its only remaining distance name is printed with its Walk and minimum theorem; these definitions are literal in Appendix A.

For the ordinary coordinate description, coordinateEquiv is a proved bijection between V and Fin2 x Fin2 x Fin3. adjacency_both_directions proves that a source arc corresponds exactly to one of the three explicit modular coordinate advances in TripleArc. source_cardinality and triple_cardinality give twelve. start_transport and finish_transport give 000/001. path_transport transports directed consecutive arcs, endpoints, injection and surjection in both directions. The finite recurrence does not depend on this alternative representation; it has no further Fin12/certificate transport boundary to assume.

## Unconditional endpoint, providers and axioms

Production dependency order: SourceSemantics, WitnessSearch, ProductBridge, EndpointAudit. The final two endpoints are Torus377269.expanded_unconditional and Torus377269.expanded_distinct_unconditional. Their types are negations of full universal assertions; no nonexistence, recurrence, graph isomorphism, source review or distance correctness premise is present. The conditional reductions in SourceSemantics are visibly intermediate helpers, discharged by the production no_hamiltonian theorem.

All printed production endpoint, recurrence-equivalence, sequence-completeness/soundness, coordinate-equivalence, adjacency, path-transport and witness-distance axiom lists are exactly [propext, Classical.choice, Quot.sound]. Failed-draft elaboration may print sorryAx after an error; it is not a production theorem and is retained as failure evidence. There is no sorry/admit/new axiom/unsafe/native_decide in the literal production sources. #eval Lean.versionString in EndpointAudit is a counted version query only; no mathematical theorem uses executable evaluation. It reports 4.34.1.

Each production module is separately compiled and separately leanchecker-checked through the frozen 32GiB wrappers. LEAN_PATH contains only this lane's build output and the eight pinned STANDARD package build roots listed in PINS.json. Direct external imports are specific modules, never import Mathlib. No other project scientific source, olean or cache is used. The wrapper verifies the Lean and leanchecker binary hashes on every invocation.

mathdb:2611.00108v1 17 Nov 2026

The STANDARD8 oleans are trusted as authorized; neither their entire origin history nor all transitive olean source bytes are independently attested by this lane. This provider limitation is explicit, not an extra mathematical premise. Actual frozen helpers and PINS remain unchanged.

## Extraction, commands and verification boundary

Appendix A contains every production Lean byte, including every import and all constants. There are no tables or external mathematical data. The markers include SHA256 of UTF-8 bytes. extract_paper.py extracts exactly the four dependency-ordered modules, verifies hashes and byte equality, and stages them in extracted/src. For the self-rebuild it writes those verified extracted bytes to src through the owned writer; frozen wrappers archive any prior own olean and rebuild. The recorded self-replay uses four fresh compiles and four separate kernel checks. It is not the required later independent replay.

Compile/check commands, from the assigned lane, are exactly:

taskset -c 4,5 bash ./check32.sh SourceSemantics LABEL
taskset -c 4,5 bash ./checker32.sh SourceSemantics LABEL
taskset -c 4,5 bash ./check32.sh WitnessSearch LABEL
taskset -c 4,5 bash ./checker32.sh WitnessSearch LABEL
taskset -c 4,5 bash ./check32.sh ProductBridge LABEL
taskset -c 4,5 bash ./checker32.sh ProductBridge LABEL
taskset -c 4,5 bash ./check32.sh EndpointAudit LABEL
taskset -c 4,5 bash ./checker32.sh EndpointAudit LABEL

For an independent fresh directory, first recover the named helper files and PINS from Appendix B and the extractor from Appendix C, create the build and cache directories, and place PAPER-v1.md beside them. The extractor accepts a missing src file, verifies the appendix digest, and writes the extracted source through the included writer; when src exists, it additionally verifies byte equality. Pinned STANDARD8/toolchain paths must be available as stated; changing pins requires separate authorization and verification. No missing source is silently treated as a preexisting comparison.

LABEL is unique per invocation, with actual chronological labels in the receipt table. Source files are src/MODULE.lean and own results build/MODULE.olean. Frozen check32/checker32, formal_command.py, run_lean_32g.py and PINS are included literally below for rebuild review. Runtime caps: affinity4,5; j1; M3500; warningsAsErrors; AS32GiB only for formal operations; sampled RSS6GiB; CPU115 seconds and wall130 seconds per formal child. Documentary work uses the own 6GiB runner. Every query, failure and checker counts against 72. No budget file was edited.

## Hostile self-review and process qualifications

The counterexample does not exploit u=v. Digons are expressly within the source domain; length-three reversal is forbidden. Hamiltonian means actual twelve-vertex coverage without repetition, not a residue condition. The recurrence's completeness and soundness are proved, the negative state is kernel reduced, and the source-sequence bridge is unconditional. The distance-one theorem has an explicit real arc and excludes length zero. The source's proved equal-factor theorem does not resolve the unequal (2,2,3) instance. A supplied PASS/SUBSTANTIAL report is not an axiom, a new priority proof or acceptance authority.

Mandatory TASK and override were personally read in bounded slices, followed by bootstrap and standalone frozen-before (48/48 matched). Root/batch/campaign AGENTS were read; the final campaign tail was read with the model controls. Model and target reservation, resource budgets, helper interfaces, source identity and actual standard interfaces used were examined. Large control JSONs were recursively screened over keys and scalar values for the target ID/arXiv ID/title and author aliases, with no matched adverse work restriction. Origins were not followed. Actual primary text reading is the first 10000 characters of directed.txt (through the relevant graph definitions and Proposition2.6 context), title/author metadata, the raw statement field, and source pages2-3

mathdb:2611.00108v1 17 Nov 2026

rendered and visually inspected. The independent report's ordinary proof and process section were read. Other portions of the independent report, INVESTIGATION, full registry/acceptance fields, unrelated papers, and the remainder of the primary manuscript are not represented as personally read. Full frozen copies remain available to the independent reviewer. No GET/search/install/git/account/message/publication/upload or child was used.

UNWAIVED current defects: the initial run.py interface read printed 2487 documentary characters, exceeding the 2400 payload limit; early interfaces were read directly before the runner interface had been fully learned. The agents-campaign-8 call attempted too much ownership text, exceeded bounded logical/display policy and truncated it; it is not claimed as full ownership reading. An early restrictions screen printed generic-host/unrelated matches; later target-specific recursive screening corrected the scope but does not erase the earlier defect. The first formal01 receipt was printed at excessive response length before formal output filtering was corrected. At least one functions JavaScript parse failure occurred before a command ran. Final document generation also failed on a receipt-label KeyError (formal1 versus formal01), was corrected, and its failure receipt and source are retained. Some standard import paths/interfaces were guessed before the actual pinned source read; the missing Mathlib.Tactic.Omega import failed and was corrected to the installed standard Lean tactic, and actual used interfaces were subsequently read. The first final-preseal documentary audit failed because it counted an extractor regex literal as an additional appendix marker; the corrected audit anchors markers at line starts. This failure is preserved and UNWAIVED. All compiler failures and failed-draft sorryAx diagnostics are retained. Successful later builds do not waive these defects.

Inherited UNWAIVED reviewer defects include initial oversized/truncated TASK display, failed literal taskset spelling, oversized generic control diagnostics, late auxiliary normalization, a serialization failure, uninstrumented redirect-hop accounting and an exact diagnostic that checked the110-second bound after completion instead of installing a watchdog. Producer defects additionally include oversized TASK display, prebootstrap/order and direct-interface deviations, control dump/truncation/timing and redirect accounting. Their complete unchanged source reports remain frozen inputs. Correct mathematics does not launder any of these into clean-process PASS.

This lane makes no credit/acceptance/ledger edits. After all writes, standalone frozen-after must run, followed by an exhaustive physical SHA seal excluding exactly the manifest itself and its generating receipt/stdout. No command or write may follow that seal. Final frozen-after and seal files are authoritative for those final outcomes. Fresh independent extraction/rebuild/source/kernel/axiom review remains mandatory afterward.

## Formal invocation receipts (all failures included)

|Receipt|exit|child CPU seconds|wall seconds|sampled group RSS bytes|
|---|---:|---:|---:|---:|
|formal01.receipt.json|1|0.231734|0.233120|247840768|
|formal02.receipt.json|1|0.763957|0.796241|1652080640|
|formal03.receipt.json|0|0.797612|0.800846|1652240384|
|formal04.receipt.json|0|1.208780|1.214648|1395040256|
|formal05.receipt.json|1|0.755995|0.764401|1686794240|
|formal06.receipt.json|1|0.757801|0.762016|1686900736|
|formal07.receipt.json|0|0.776754|0.780797|1686900736|
|formal08.receipt.json|0|15.600789|15.677538|2925473792|
|formal09.receipt.json|1|15.574431|15.679806|2928218112|
|formal10.receipt.json|0|14.807125|14.896284|2928386048|
|formal11.receipt.json|0|9.052821|9.219387|2649837568|
|formal12.receipt.json|1|0.764675|0.860606|1678602240|
|formal13.receipt.json|1|0.994304|0.998334|1737965568|
|formal14.receipt.json|0|1.043753|1.048149|1741004800|
|formal15.receipt.json|0|1.307431|1.403693|1476706304|
|formal16.receipt.json|1|0.743933|0.816366|1689915392|

mathdb:2611.00108v1 17 Nov 2026

|formal17.receipt.json|1|0.750247|0.753411|1695498240|
|formal18.receipt.json|0|0.763560|0.767075|1695604736|
|formal19.receipt.json|0|1.191097|1.196999|1438535680|
|formal20.receipt.json|0|0.797996|0.801375|1656156160|
|formal21.receipt.json|0|1.209968|1.215208|1394954240|
|formal22.receipt.json|0|16.085017|16.355183|2928328704|
|formal23.receipt.json|0|9.507849|9.774928|2662309888|
|formal24.receipt.json|0|1.023840|1.028087|1741295616|
|formal25.receipt.json|0|1.301529|1.306748|1478651904|
|formal26.receipt.json|0|0.768040|0.772206|1695617024|
|formal27.receipt.json|0|1.198750|1.203893|1438269440|

At document generation: 27/72 formal invocations; total recorded formal child CPU 99.779788s; maximum recorded formal sampled RSS 2928386048 bytes. No formal killed receipt is omitted. Full stdout, stderr, command and caps are in each receipt/output pair.

## Frozen identity and dependency hashes

- TASK.txt: 73a54daebc86fb02bdf02f2de69ace006a164aa7e2f322136a023ac133e74c7f
- frozen-inputs.json: ce3cc6a5f05e91f0a37f3b13db1b56533d0dbac65567317fe8e4ddc31ab13ee9
- inputs/1324-inversions-material-prior-root-park-20261115-v1.json:
ac6612464585eea138d3d6084ac904044b5c66851f48cd410c05fd5161ce1aec
- inputs/349311-corrected-classification-root-eligibility-stop-20261115-v1.json:
f12780dda0c907e2f6c6247d0ff612c6e61226d2f130741e7e99e5d15a0a5b2d
- inputs/377269-independent-pass-formal-go-root-intake-20261115-v1.json:
18dd30b283e4bed38eb2a13b7968bb7145f007869e9725fffd7a78f6bd73fd8c
- inputs/377269-owned-target-binding-provenance-20261115-v1.json:
bd676875be8bb1346d16f1914b850e1a444e4e2835c5aa79fd5251992dd31ac6
- inputs/DISPLAY-AND-AFFINITY-OVERRIDE.json:
ab9758ec873f73219260af1926048c7fc2fc0727ff50bbca997a8b53deeb21f7
- inputs/INDEPENDENT-PROOF-SOURCE-CONTRIBUTION-SELF-HANDOFF-v1.md:
b74a34e3023f2eb1b71a3821b1d101d42215662396a57961c779ad39eda22d33
- inputs/INVESTIGATION.md: 2caa00f642792d8e32b73c8d6624afbfb1c5cead02c856bb433b278165306880
- inputs/LEAN-CAP-POLICY.json: 4e11b1a2cb675ff168d8b2c3c679ec90147aedd52311116f28e8e38374aab6a4
- inputs/PDF-SKILL.md: a6e20f4ae66364a06d577dd51e0ce66ee2ef955be129cbe195e0a27f403daaf8
- inputs/SOURCE-IDENTITY-CARD.json: c65f45c533fce7b24cc1660e086a5f7fde40d1d7a0c909c88c0f5a54aa648d5c
- inputs/acceptance-registry-head-refresh-20261115-v4.json:
c3246f53de269590828f41b0bef960d995e1f372ddefd44f9199442ee2f1ad8d
- inputs/campaign-AGENTS-current.md: d643a8fc94d1529d852fb50f91570689c7934b778848fb9ccfb97fb189667df6
- inputs/candidate-metadata.json: 28d0474d4220df718048a3c0479b5c6773eedcd9a19214325ea4be064c6383c9
- inputs/current-statement-catalogue.json:
fd41a43180c7bfa7bf1935cda4013ceea222a8bf69d904a394afecd82fa13d01
- inputs/cyclic-bibd-material-spectrum-root-park-20261115-v1.json:
e722142ebe2b2dbd5b25d62c4365ebd705fa8b350642d2d124b0ced6e9330e11
- inputs/directed-abs.html: 985cb79f710f1add3dd0e15d8c7a4449e1be0c364a975846cb0e2d185f6310ba
- inputs/directed-page2.png: 0225ef4733d6f71810d9b73d674a2082b5c996fc74f21a58d170336098bd2ea7
- inputs/directed.pdf: 3ff46e6951fc5dad0001e9e441b4190d1640f676b37556b267f815be5e6bc612
- inputs/directed.txt: 4eeb7d982244120c0d58a85ed6c70ecc91b56c6393caa418f06025df8aa5f334
- inputs/field-release-user-authorization-20261114-v1.json:
c320027b03de2aba0bd527e4171892bbb4f5708887953a4a4aa4c070bfb9c833

mathdb:2611.00108v1 17 Nov 2026

- inputs/imset-14-endpoint-field-release-20261114-v1.json:
0f532fab5ea81180f8a8684efe62747babf6fe352759ae056b84e97129b97cc7
- inputs/live377269-statement-field-release-20261115-v1.json:
fe797c13061ceaaa1cdf9e0415b97ebb72d2305c587d35b2439e831ca7c3023e
- inputs/mathdb-377269.raw.jsonl: c6376ea03a28dcf12e37b395e3606e5fa86f9624674bc0643aecadcf7d45e2d3
- inputs/pinned-standard-library-text-path-release-20261115-v1.json:
00d18b4e59efeb00f5f8eca3d1bc2825486ae1ecfe467e45a696173c5796f332
- inputs/public-work-restrictions-latest-20261114-v22.json:
7a8844e2e26b1482ad846384abb4ceba0f79356371da0bcf2c40ecec93a69ca3
- inputs/registry-coverage-provenance.json:
f53cda08c7fdb0bcedf280a758dda0e68e559a3be76f698185f108f49ad10e06
- inputs/sanitized-acceptance-release-110-v9-20261115.json:
b4e876b4c2b494e62436ed82880fd225706b1d319e312ca4d0c9b321ffe8d885

## Appendix A: complete literal production Lean

<!-- LEANFILE src/SourceSemantics.lean
sha256=e596edb4d3c05f87657020bf96cb576bc4021abfa27a4663f116d3ec4db7dffb -->
Literal Lean source appendix: publication/lean-appendix-18.md in the accompanying proof-sources.json / supporting.zip. All proof source bytes are supplied separately.
<!-- ENDLEANFILE -->

<!-- LEANFILE src/WitnessSearch.lean
sha256=446c09beae00f78789f1e459b6b7095a784c36339fa65ab83726503d1572bf34 -->
Literal Lean source appendix: publication/lean-appendix-19.md in the accompanying proof-sources.json / supporting.zip. All proof source bytes are supplied separately.
<!-- ENDLEANFILE -->

<!-- LEANFILE src/ProductBridge.lean
sha256=2c3f05a4068438eee3f176416b308a677dd05bcbe3a0ab0fb3464e4d852d33ae -->
lean
import WitnessSearch
import Mathlib.Data.Fintype.Prod

namespace Torus377269
abbrev Triple := Fin 2 × Fin 2 × Fin 3

def coordinates (u : V) : Triple := (u 0, u 1, u 2)
def fromTriple (t : Triple) : V := fun j =>
⟨(![t.1.val, t.2.1.val, t.2.2.val]) j, by
fin_cases j
· exact t.1.isLt
· exact t.2.1.isLt
· exact t.2.2.isLt⟩

def coordinateEquiv : V ≃ Triple where
toFun := coordinates
invFun := fromTriple
left_inv := by intro u; funext j; fin_cases j <;> rfl
right_inv := by rintro ⟨a, b, c⟩; rfl

mathdb:2611.00108v1 17 Nov 2026

```
theorem from_coordinates (u : V) : fromTriple (coordinates u) = u := coordinateEquiv.left_inv u
theorem coordinates_from (t : Triple) : coordinates (fromTriple t) = t := coordinateEquiv.right_inv t

def TripleArc (a b : Triple) : Prop :=
((b.1.val = (a.1.val + 1) % 2) ∧ b.2 = a.2) ∨
(b.1 = a.1 ∧ b.2.1.val = (a.2.1.val + 1) % 2 ∧ b.2.2 = a.2.2) ∨
(b.1 = a.1 ∧ b.2.1 = a.2.1 ∧ b.2.2.val = (a.2.2.val + 1) % 3)

theorem adjacency_both_directions : ∀ u v : V,
Arc u v ↔ TripleArc (coordinates u) (coordinates v) := by unfold Arc TripleArc; decide

theorem source_cardinality : Fintype.card V = 12 := by decide
theorem triple_cardinality : Fintype.card Triple = 12 := by decide
theorem start_transport : coordinates start = (0, 0, 0) := rfl
theorem finish_transport : coordinates finish = (0, 0, 1) := rfl

-- Exact transport of a spanning simple directed sequence, in both directions.
def TripleHamiltonian (u v : Triple) : Prop :=
∃ n, ∃ f : Fin (n + 1) → Triple,
f 0 = u ∧ f (Fin.last n) = v ∧
(∀ i : Fin n, TripleArc (f i.castSucc) (f i.succ)) ∧
Function.Injective f ∧ Function.Surjective f

theorem path_transport (u v : V) :
Hamiltonian u v ↔ TripleHamiltonian (coordinates u) (coordinates v) := by
constructor
· rintro ⟨n, f, h0, hl, ha, hi, hs⟩
refine ⟨n, coordinates ∘ f, congrArg coordinates h0, congrArg coordinates hl, ?_,
coordinateEquiv.injective.comp hi, coordinateEquiv.surjective.comp hs⟩
intro i
exact (adjacency_both_directions _ _).mp (ha i)
· rintro ⟨n, f, h0, hl, ha, hi, hs⟩
refine ⟨n, fromTriple ∘ f, ?_, ?_, ?_, coordinateEquiv.symm.injective.comp hi,
coordinateEquiv.symm.surjective.comp hs⟩
· simpa only [Function.comp_apply, from_coordinates] using congrArg fromTriple h0
· simpa only [Function.comp_apply, from_coordinates] using congrArg fromTriple hl
· intro i
apply (adjacency_both_directions _ _).mpr
simpa only [Function.comp_apply, coordinates_from] using ha i

#print axioms adjacency_both_directions
#print axioms path_transport
end Torus377269


<!-- ENDLEANFILE -->

<!-- LEANFILE src/EndpointAudit.lean
sha256=a19ed5580c1b7b5aaf47c015f8082b888c95c1d302be4fb8d747c3a483854709 -->
lean
import ProductBridge

namespace Torus377269
-- No certificate, graph, distance or nonexistence hypothesis is a premise.
```

mathdb:2611.00108v1 17 Nov 2026

```
theorem expanded_unconditional :
¬ (∀ (k : ℕ), 3 ≤ k → ∀ (m : Fin k → ℕ), (∀ i, 2 ≤ m i) →
∀ (u v : (i : Fin k) → Fin (m i)),
(∃ n, ∃ f : Fin (n + 1) → ((i : Fin k) → Fin (m i)),
f 0 = u ∧ f (Fin.last n) = v ∧
(∀ t : Fin n, ∃ i : Fin k,
(f t.succ i).val = ((f t.castSucc i).val + 1) % m i ∧
∀ j, j ≠ i → f t.succ j = f t.castSucc j) ∧
Function.Injective f ∧ Function.Surjective f) ↔
(((List.ofFn m).foldr Nat.gcd 0 : ℕ) : ℤ) ∣ ((distance u v : ℤ) + 1)) :=
refutation

theorem expanded_distinct_unconditional :
¬ (∀ (k : ℕ), 3 ≤ k → ∀ (m : Fin k → ℕ), (∀ i, 2 ≤ m i) →
∀ (u v : (i : Fin k) → Fin (m i)), u ≠ v →
((∃ n, ∃ f : Fin (n + 1) → ((i : Fin k) → Fin (m i)),
f 0 = u ∧ f (Fin.last n) = v ∧
(∀ t : Fin n, ∃ i : Fin k,
(f t.succ i).val = ((f t.castSucc i).val + 1) % m i ∧
∀ j, j ≠ i → f t.succ j = f t.castSucc j) ∧
Function.Injective f ∧ Function.Surjective f) ↔
(((List.ofFn m).foldr Nat.gcd 0 : ℕ) : ℤ) ∣ ((distance u v : ℤ) + 1))) :=
distinct_refutation

#print expanded_unconditional
#print expanded_distinct_unconditional
#print distance
#print distance_minimum
#print Walk
#print Arc
#print factorGCD
#print search
#print search_sequence
#print sequence_search
#print recurrence_iff_hamiltonian
#print coordinateEquiv
#print TripleArc
#print adjacency_both_directions
#print path_transport
#eval Lean.versionString
#print axioms expanded_unconditional
#print axioms expanded_distinct_unconditional
#print axioms recurrence_iff_hamiltonian
#print axioms search_sequence
#print axioms sequence_search
#print axioms coordinateEquiv
#print axioms adjacency_both_directions
#print axioms path_transport
#print axioms witness_distance
```

mathdb:2611.00108v1 17 Nov 2026

end Torus377269

<!-- ENDLEANFILE -->

## Appendix B: exact frozen pins and formal wrappers

### PINS.json SHA256 f7f3d6f49f2ed48dd7a181d9c692a1bf95c14810fe1f8778c9298ab8e352c089

text

{
"lean_version": "4.34.1",
"mathlib_revision": "d13f23b723b8a846827a245b89c10fc7d3f11612",
"lean_bin": "/home/mj-kang/Documents/Dev/math/dlp-research/.tools/lean-4.34.1-linux/bin/lean",
"lean_sha256": "e8baaa71855a616dc351028f3ad2200051b0671f423a1696a100e809302d5550",
"leanchecker_bin": "/home/mj-kang/Documents/Dev/math/dlp-research/.tools/lean-4.34.1-linux/bin/leanchecker",
"leanchecker_sha256": "ccef1bfe9bfeecea65137d94a69033c1a89f360711f50fa3178e7ba3e696fcbe",
"packages_root": "/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages",
"upstream_lean_paths": [
"/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages/mathlib/.lake/build/lib/lean",
"/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages/batteries/.lake/build/lib/lean",
"/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages/aesop/.lake/build/lib/lean",
"/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages/Qq/.lake/build/lib/lean",
"/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages/proofwidgets/.lake/build/lib/lean",
"/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages/importGraph/.lake/build/lib/lean",
"/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages/LeanSearchClient/.lake/build/lib/lean",
"/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages/plausible/.lake/build/lib/lean"
],
"mathlib_source": "/home/mj-kang/Documents/Dev/math/openai-proof-research/frontier/campaigns/2026-10-14/formal/.lake/packages/mathlib",
"mathlib_metadata_sha256": {
"Mathlib.lean": "61d699d8ace6240934a751f1b1951339090d6e380b505041632227dd7eb73a29",
"lake-manifest.json": "8f67b2cf24143ac091cdb425e886c2164b2fc2d6fbccd6db9454b697f9541a67"
},
"pin_limits": "Root hashed binaries and available upstream source metadata, did NOT execute Lean/git or attest all upstream olean origins. Worker verifies runtime version/pin metadata before compilation; accepted pinned upstream oleans trusted. Never overwrite PINS.json; record discovered verification limits separately."
}


### check32.sh SHA256 4706b93c28ab6885f257d458726e87b36e5ca56192575593b07529cc18a56d7c

text
#!/bin/bash

mathdb:2611.00108v1 17 Nov 2026

```
set -eu
export FORMAL_OPERATION=compile
export FORMAL_MODULE="${1:?module name required}"
export FORMAL_ROOT="$(cd "$(dirname "$0")" && pwd)"
exec timeout 140 python3 -I -B "$FORMAL_ROOT/run_lean_32g.py" "${2:?unique receipt label required}" <<'PY'
import os,runpy
runpy.run_path(os.path.join(os.environ["FORMAL_ROOT"],"formal_command.py"),run_name="__main__")
PY
```

```
### checker32.sh SHA256 b6366a53972ea147ec8a8dfffea2d156b21fb2fec71aa708742881e6ec465f76
```

```
text
#!/bin/bash
set -eu
export FORMAL_OPERATION=check
export FORMAL_MODULE="${1:?module name required}"
export FORMAL_ROOT="$(cd "$(dirname "$0")" && pwd)"
exec timeout 140 python3 -I -B "$FORMAL_ROOT/run_lean_32g.py" "${2:?unique receipt label required}" <<'PY'
import os,runpy
runpy.run_path(os.path.join(os.environ["FORMAL_ROOT"],"formal_command.py"),run_name="__main__")
PY
```

```
### formal_command.py SHA256 f0af4db06cefea766d2702f50b9513256d2f0233d62e87f8093d46e500e66059
```

```
text
import os,re,json,hashlib,subprocess,sys,shutil
from pathlib import Path
p=Path(__file__).resolve().parent
pins=json.loads((p/'PINS.json').read_text())
mode=os.environ['FORMAL_OPERATION'];module=os.environ['FORMAL_MODULE']
assert mode in ['compile','check'] and re.fullmatch(r'[A-Za-z][A-Za-z0-9_]*',module)
def sha(q):return hashlib.sha256(q.read_bytes()).hexdigest()
assert sha(Path(pins['lean_bin']))==pins['lean_sha256'] and
sha(Path(pins['leanchecker_bin']))==pins['leanchecker_sha256']
source=p/'src'/(module+'.lean');olean=p/'build'/(module+'.olean')
assert source.is_file();before=sha(source)
env=dict(os.environ);env['PATH']=str(Path(pins['lean_bin']).parent)+os.pathsep+env.get('PATH','');env['LEAN_PATH']=o
s.pathsep.join([str(p/'build')]+pins['upstream_lean_paths']);env['LEAN_NUM_THREADS']='1'
prior=sha(olean) if olean.exists() else None
if mode=='compile':
if olean.exists():
archive=p/'cache'/(module+'.'+prior+'.olean')
if not archive.exists():shutil.copyfile(olean,archive)
olean.unlink()
args=[pins['lean_bin'],'-j1','-M3500','-DwarningAsError=true','-o',str(olean),str(source)]
else:
assert olean.is_file(),'Kernel checker requires actual production olean'
args=[pins['leanchecker_bin'],module]
r=subprocess.run(args,cwd=p,env=env,text=True,capture_output=True)
sys.stdout.write(r.stdout);sys.stderr.write(r.stderr)
```

mathdb:2611.00108v1 17 Nov 2026

mathdb:2611.00108v1 17 Nov 2026

```
after=sha(source);out=sha(olean) if olean.exists() else None
print(json.dumps(dict(stage=mode,module=module,argv=args,exit=r.returncode,source_sha256_before=before,sour
ce_sha256_after=after,olean_sha256_before=prior,olean_sha256_after=out,LEAN_PATH=env['LEAN_PATH'],LEAN_NU
M_THREADS=1)))
assert before==after,'Source changed during compile/check'
if r.returncode==0:
assert out is not None,'No fresh olean produced'
if mode=='check':assert prior==out,'Checker changed olean'
raise SystemExit(r.returncode)
```

### run_lean_32g.py SHA256 dd826a8e5bcdc01f7dd46022a5a1a06dda1507e56a0dcf82624790c9befc7388

```
text
import os,resource
os.sched_setaffinity(0,sorted(os.sched_getaffinity(0))[:2])
resource.setrlimit(resource.RLIMIT_CPU,(120,120)); resource.setrlimit(resource.RLIMIT_AS,(32*10243,32*10243))
import sys,subprocess,time,json,pathlib,datetime,signal
lane=pathlib.Path(__file__).resolve().parent
assert os.environ.get('FORMAL_OPERATION') in ('compile','check') and os.environ.get('FORMAL_MODULE'),'This
runner is only for Lean compile and kernel-check runs'
name=sys.argv[1]; code=sys.stdin.read(); start=time.monotonic();
utc=datetime.datetime.now(datetime.timezone.utc).isoformat()
env=dict(os.environ,MALLOC_ARENA_MAX='2')
prefix='import os,resource; os.sched_setaffinity(0,sorted(os.sched_getaffinity(0))[:2]);
resource.setrlimit(resource.RLIMIT_CPU,(115,115)); resource.setrlimit(resource.RLIMIT_AS,(32*10243,32*10243));\n'
p=subprocess.Popen([sys.executable,'-I','-B','-c',prefix+code],cwd=lane,env=env,stdout=subprocess.PIPE,stderr=subp
rocess.PIPE,start_new_session=True)
peak=0; samples=0; killed=None; vm=0
while p.poll() is None:
rss=0
for ent in pathlib.Path('/proc').iterdir():
if not ent.name.isdigit(): continue
try:
s=(ent/'stat').read_text().split(') ',1)[1].split()
if int(s[2])==p.pid:
rss+=int((ent/'statm').read_text().split()[1])*os.sysconf('SC_PAGE_SIZE')
vm=max(vm,max([int(l.split()[1])*1024 for l in (ent/'status').read_text().splitlines() if l.startswith('VmPeak:')] or [0]))
except (OSError,ValueError,IndexError): pass
peak=max(peak,rss); samples+=1
if rss>6*10243 or time.monotonic()-start>130:
killed='rss' if rss>6*10243 else 'deadline'; os.killpg(p.pid,signal.SIGKILL); break
try: out,err=p.communicate(timeout=.08); break
except subprocess.TimeoutExpired: pass
out,err=p.communicate(); wall=time.monotonic()-start
r={'utc':utc,'name':name,'cwd':str(lane),'command':[sys.executable,'-I','-B','-c',prefix+code],'env_overrides':{'MALLOC_
ARENA_MAX':'2'},'affinity':sorted(os.sched_getaffinity(0)),'cpu_limit_s':115,'as_limit_bytes':32*10243,'group_rss_guard
_bytes':6*10243,'deadline_s':130,'wall_s':wall,'sampled_group_rss_peak_bytes':peak,'sampled_max_process_vmpeak_
bytes':vm,'samples':samples,'exit':p.returncode,'killed':killed,'child_cpu_cumulative_s':resource.getrusage(resource.R
USAGE_CHILDREN).ru_utime+resource.getrusage(resource.RUSAGE_CHILDREN).ru_stime,'stderr':err.decode(errors='
```

mathdb:2611.00108v1 17 Nov 2026

```
replace')}
def write(n,s):
z=subprocess.run([sys.executable,'-I','-B',str(lane/'write_lane.py'),n],input=s,text=True,capture_output=True,env=env,
timeout=10); assert z.returncode==0,z.stderr
write(name+'.receipt.json',json.dumps(r,indent=2)); write(name+'.output.txt',out.decode(errors='replace'))
print(out.decode(errors='replace')); print(json.dumps(r)); sys.exit(p.returncode)

### write_lane.py SHA256 b45b49e99f75e73c33b03852587f95e61b502476adaa69d9a10a6619827fe636

text
import os,resource
os.sched_setaffinity(0,sorted(os.sched_getaffinity(0))[:2])
resource.setrlimit(resource.RLIMIT_CPU,(10,10))
resource.setrlimit(resource.RLIMIT_AS,(512*10242,512*10242))
import pathlib,sys
lane=pathlib.Path(__file__).resolve().parent
rel=pathlib.PurePath(sys.argv[1])
assert not rel.is_absolute() and ".." not in rel.parts
target=(lane/rel).resolve()
assert target.is_relative_to(lane) and target!=lane and not target.is_symlink()
text=sys.stdin.read()
target.parent.mkdir(parents=True,exist_ok=True)

if len(sys.argv)>2 and sys.argv[2]=='--base64':
import base64
target.write_bytes(base64.b64decode(text))
else:
target.write_text(text)
print(str(target))

### run.py SHA256 d8f5ab3b62612d972ec34bb1c90dd661a96d058d6b131c76f8b0de9217e50264

text
import os,resource
os.sched_setaffinity(0,sorted(os.sched_getaffinity(0))[:2])
resource.setrlimit(resource.RLIMIT_CPU,(120,120)); resource.setrlimit(resource.RLIMIT_AS,(6*10243,6*10243))
import sys,subprocess,time,json,pathlib,datetime,signal
lane=pathlib.Path(__file__).resolve().parent
name=sys.argv[1]; code=sys.stdin.read(); start=time.monotonic();
utc=datetime.datetime.now(datetime.timezone.utc).isoformat()
env=dict(os.environ,MALLOC_ARENA_MAX='2')
prefix='import os,resource; os.sched_setaffinity(0,sorted(os.sched_getaffinity(0))[:2]);
resource.setrlimit(resource.RLIMIT_CPU,(115,115)); resource.setrlimit(resource.RLIMIT_AS,(6*10243,6*10243));\n'
p=subprocess.Popen([sys.executable,'-I','-B','-c',prefix+code],cwd=lane,env=env,stdout=subprocess.PIPE,stderr=subp
rocess.PIPE,start_new_session=True)
peak=0; samples=0; killed=None
while p.poll() is None:
rss=0
for ent in pathlib.Path('/proc').iterdir():
if not ent.name.isdigit(): continue
try:
s=(ent/'stat').read_text().split(') ',1)[1].split()
```

mathdb:2611.00108v1 17 Nov 2026

```
if int(s[2])==p.pid:
rss+=int((ent/'statm').read_text().split()[1])*os.sysconf('SC_PAGE_SIZE')
except (OSError,ValueError,IndexError): pass
peak=max(peak,rss); samples+=1
if rss>6*10243 or time.monotonic()-start>130:
killed='rss' if rss>6*10243 else 'deadline'; os.killpg(p.pid,signal.SIGKILL); break
try: out,err=p.communicate(timeout=.08); break
except subprocess.TimeoutExpired: pass
out,err=p.communicate(); wall=time.monotonic()-start
r={'utc':utc,'name':name,'cwd':str(lane),'command':[sys.executable,'-I','-B','-c',prefix+code],'env_overrides':{'MALLOC_
ARENA_MAX':'2'},'affinity':sorted(os.sched_getaffinity(0)),'cpu_limit_s':115,'as_limit_bytes':6*10243,'group_rss_guard_
bytes':6*10243,'deadline_s':130,'wall_s':wall,'sampled_group_rss_peak_bytes':peak,'samples':samples,'exit':p.returnc
ode,'killed':killed,'child_cpu_cumulative_s':resource.getrusage(resource.RUSAGE_CHILDREN).ru_utime+resource.getr
usage(resource.RUSAGE_CHILDREN).ru_stime,'stderr':err.decode(errors='replace')}
def write(n,s):
z=subprocess.run([sys.executable,'-I','-B',str(lane/'write_lane.py'),n],input=s,text=True,capture_output=True,env=env,
timeout=10); assert z.returncode==0,z.stderr
write(name+'.receipt.json',json.dumps(r,indent=2)); write(name+'.output.txt',out.decode(errors='replace'))
display=out.decode(errors='replace')
print(display[:10000])
import hashlib
summary={k:v for k,v in r.items() if k not in {'command','stderr'}}
summary.update({'full_receipt':name+'.receipt.json','full_stdout':name+'.output.txt','stdout_characters':len(display),'s
tdout_display_truncated':len(display)>10000,'command_sha256':hashlib.sha256(json.dumps(r['command']).encode())
.hexdigest(),'stderr_bytes':len(err),'display_policy':'Full stdout/command/stderr preserved on disk; bounded display is
not full personal reading.'})
encoded=json.dumps(summary)
assert len(encoded)<4000
print(encoded)
sys.exit(p.returncode)
```

## Final self-replay outcome

All eight extracted-source operations formal20 through formal27 returned zero: each of the four modules was freshly compiled and separately checked by leanchecker. Sources are byte-identical to Appendix A and the paper-extraction report. This is SELF replay only. The expanded final audit in formal26.output.txt has no endpoint premises and only the three permitted axioms. All nine earlier failed invocations remain in the receipt table and their source drafts remain available.

## Appendix C: exact self-extractor

```
python
from pathlib import Path
import re,hashlib,subprocess,json
root=Path(__file__).resolve().parent
s=(root/'PAPER-v1.md').read_text()
pat=r'<!-- LEANFILE (src/[A-Za-z]+\.lean) sha256=([a-f0-9]{64}) -->\nlean\n(.*?)\n<!-- ENDLEANFILE -->'
entries=re.findall(pat,s,re.S)
expected=['src/'+m+'.lean' for m in ['SourceSemantics','WitnessSearch','ProductBridge','EndpointAudit']]
assert [n for n,_,_ in entries]==expected
```

```
report=[]
for n,h,t in entries:
b=t.encode();actual=hashlib.sha256(b).hexdigest()
assert actual==h,(n,'appendix digest')
previous_match = b==(root/n).read_bytes() if (root/n).exists() else None
assert previous_match is not False,(n,'production equality')
for dest in ['extracted/'+n,n]:
p=subprocess.run(['python3','-I','-B','write_lane.py',dest],input=t,text=True,capture_output=True)
assert p.returncode==0,p.stderr
report.append({'path':n,'sha256':actual,'bytes':len(b),'matches_production':previous_match})
out=json.dumps({'status':'PASS_BYTE_IDENTICAL_SELF_EXTRACTION','modules':report,'independent':False},indent=2)
p=subprocess.run(['python3','-I','-B','write_lane.py','paper-extraction-v1.json'],input=out,text=True,capture_output=T
rue)
assert p.returncode==0
print(out)
```

mathdb:2611.00108v1 17 Nov 2026