Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.083232

ARTICLE

Topological Classification of State-Space Networks Generated by Formal Planning Rules

Zhendong Du* **and Kenji Hashimoto**

Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Japan

*Corresponding Author: Zhendong Du. Email: zhendong@fuji.waseda.jp

Received: 31 March 2026; Accepted: 02 July 2026; Published: 13 August 2026

ABSTRACT: Network science has developed powerful tools for characterizing the topology of emergent networks—systems shaped by evolution, growth, and stochastic attachment—but the topology of constructed networks, graphs generated by the exhaustive application of formal rules, remains theoretically uncharacterized. This paper establishes that the Planning Domain Definition Language (PDDL), the standard formal language for classical planning, is a topological determinist: two binary properties of its operator semantics, reversibility and commutativity, partition the space of generable state-space graphs into exactly three topological archetypes—directed acyclic graph (DAG), Sparse-Cyclic, and Mesh—and this partition is deducible from the language specification without examining any network instance. We prove this as a classification theorem for the fully reversible and fully irreversible extremes of the operator spectrum, show that no fourth archetype exists within this class, and verify its consequences across 289 IPC benchmark problems spanning ten domains, without exception. Archetype membership is confirmed through two operator-derived signatures—the presence of directed cycles and the shape of the out-degree distribution—rather than any single network measurement. The verification reveals a benchmark monoculture, in which semantically distinct domains generate topologically identical structures, and a systematic decoupling of instance scale from topological complexity. Planner experiments confirm the operational stakes of this decoupling: across archetypes of comparable state-space size, optimal-search cost differs by orders of magnitude, with the archetype rather than the scale governing difficulty. Together these results establish PDDL as a natural laboratory for the study of constructed network topology—the first formal language for which the relationship between generative grammar and network topology has been proved complete—and demonstrate that when networks are stipulated rather than evolved, the grammar of the generating language is the fate of the topology.

KEYWORDS: Constructed networks; formal language topology; state space graphs; topological archetypes; PDDL; complexity analysis

1 Introduction

Network science has built its foundational theories on networks that emerge. Protein interaction networks evolve under selection pressure. Social networks grow through attachment and rewiring. Power grids expand incrementally under engineering constraints. In each case, the network we observe is the outcome of a process we did not fully specify. The richness of that process leaves measurable topological signatures: scale-free degree distributions, small-world clustering, community structure. The discipline has developed powerful tools for characterizing such emergent topology precisely because emergence is the default assumption.

Constructed networks—graphs generated not by evolutionary or stochastic processes but by the exhaustive application of formal rules—have received far less theoretical attention. Yet constructed networks are ubiquitous wherever computation meets structure: state spaces of constraint satisfaction problems, reachability graphs of concurrent programs, configuration spaces of robotic systems. The question of what topology looks like when it is stipulated rather than evolved has no established answer. And this gap matters, because the forces shaping topology in the two cases are entirely different. An emergent network is shaped by history, noise, and selection. A constructed network is shaped by the generative grammar that defines what transitions are legal.

This paper asks: when a formal language fully specifies how a network may be generated, how completely does the language determine the topology of everything it can produce? The answer we establish is: completely, in the sense of exhaustive classification. More precisely, we prove that the operator semantics of the Planning Domain Definition Language (PDDL)—the standard formal language for specifying classical planning problems—admit a complete topological classification: the space of generable state-space graphs can be partitioned into a finite set of structurally distinct archetypes, each determined by the language specification alone. Two binary properties of PDDL operators, which can be read directly from the language specification without examining any network instance, partition the space of generable state-space graphs into exactly three topological archetypes. No fourth archetype is possible within the fully reversible and fully irreversible class. This is not an empirical claim inferred from data; it is a theorem deducible from the grammar of the language itself.

This distinguishes our contribution from prior work in three adjacent literatures. Generative network models specify probabilistic rules that shape a *distribution* of topological outcomes with irreducible variance; graph-grammar approaches such as NetGAP demonstrate that formal grammars constrain topology but characterize neither the completeness of the constraint nor its analytic form; and Petri net reachability analysis exploits generative structure instrumentally to accelerate construction rather than to classify the topologies a formalism can produce. We are the first to prove, for a formal language in widespread use, that two semantic attributes partition its entire generable topology space into a finite, enumerable set of archetypes deducible from the specification alone.

The two properties are operator reversibility and operator commutativity. Reversibility determines whether the state-space graph contains cycles: irreversible operators produce directed acyclic graphs, while reversible operators permit return paths. Commutativity determines local connection density: non-commutative operators produce sparse, linear structures, while commutative operators produce dense, triangulated mesh structures. A crucial asymmetry collapses the four logical combinations into three: when operators are irreversible, commutativity becomes topologically inert, because the permanent deletion of fluents by irreversible operators prevents the path convergence that commutativity would otherwise induce. The result is three and only three possible archetypes—directed acyclic graph (DAG), Sparse-Cyclic, and Mesh—and the archetype of any PDDL domain is determined before a single state is generated.

We validate this theorem against 289 problem instances drawn from ten International Planning Competition (IPC) domains. The empirical record is in agreement with the deductive prediction: every domain falls into exactly the archetype its operator algebra mandates. The verification does not rely on measuring any clustering value as a continuous quantity; instead it rests on two structural signatures read from the operator algebra. The first is the presence or absence of directed cycles, dictated by reversibility: the four irreversible domains generate acyclic graphs without exception, while the six reversible domains generate cyclic graphs. The second is the shape of the out-degree distribution, dictated by commutativity: domains whose operators commute produce a narrow, regular branching distribution, because the set of applicable actions at a state depends on the current configuration alone and not on the history of how it was

reached, whereas non-commutative reversible domains produce a broader, history-dependent distribution. These two signatures, each derivable from the operator specification, separate the three archetypes cleanly across the corpus.

A second consequence concerns the relationship between network scale and topological complexity. Classical intuition—and decades of planning benchmark analysis—treats larger state spaces as harder problems. Our data systematically violates this intuition: smaller, denser graphs consistently exhibit greater structural complexity than larger, sparser ones. The smallest domain in our corpus exhibits the highest per-state branching, while the largest exhibits a substantially lower one; diameter likewise fails to grow monotonically with scale across archetypes. This inversion is not an anomaly but a corollary of the theorem. Scale is a consequence of problem size parameters; topology is a consequence of operator algebra. The two are independent. This independence has been independently documented at the empirical level: a recent analysis of a formally specified planning benchmark derived from the Countdown game shows that problem hardness does not grow monotonically with instance size, exhibiting phase transitions in which larger instances can prove structurally simpler than smaller ones [1]. The present work provides the structural account of why such inversions occur: they are necessary consequences of the decoupling established by Corollary 2. A formal language that generates sparse topology will generate sparse topology regardless of how large the instance grows.

These findings position PDDL as a natural laboratory for a question of broad relevance to network science: how strongly do the generative rules of a formal system constrain the topology of everything it can produce? The answer demonstrated here—that two binary semantic properties suffice to classify the entire constructible topology space—suggests that formal languages are extreme topological deterministes. When networks are stipulated by formal rules rather than shaped by evolutionary or stochastic forces, the generative grammar determines topological fate with a precision that no probabilistic model can achieve.

The remainder of this paper is organized as follows. Section 2 situates this work within the literatures on complex network topology, formal graph generation, and planning state-space analysis. Section 3 develops the completeness theorem formally and defines the structural signatures used for empirical verification. Section 4 presents the experimental results as theorem validation, documenting the three archetypes and the scale–topology decoupling. Section 5 discusses the broader implications for the theory of constructed networks and identifies open questions. Section 6 concludes.

2 Related Work

2.1 Topology of Complex Networks

The measurement of global network topology as a tool for understanding system behavior has a three-decade history. Reference [2] showed that real-world networks from neural circuits to power grids occupy an intermediate regime between regular lattices and random graphs, with high clustering alongside short path lengths, and reference [3] showed that preferential attachment generates the scale-free degree distributions recurring across biological, technological, and social networks—establishing that a network's generative mechanism leaves permanent topological signatures. Newman's treatment of network structure and function codified the analytical toolkit that makes topology a scientific instrument [4]: clustering coefficients, degree distributions, diameter, and entropy-based measures each capture a distinct structural dimension, and reference [5] surveyed how these relate to network function, establishing that comprehensive fingerprinting requires multi-metric analysis. Our methodology follows this tradition, measuring multiple complementary metrics because no single number summarizes topology.

Recent work refines what topology reveals in directed graphs specifically. Different DAG-generating models produce measurably distinct cycle-basis statistics even when their degree distributions appear similar [6], directly relevant to our setting where the acyclic/cyclic distinction is a primary discriminant. Algebraic treatments of motifs and feedback loops in labeled directed graphs formalize how signed or polarized path composition determines indirect effects—a perspective that parallels our use of forward-closed triangles as the algebraic signature of commutativity. Together with analyses of the topological structure of large-scale problem-solving networks [7] and demonstrations that topological data analysis reveals structure invisible to local metrics [8], this body of work establishes global graph structure as a productive level of analysis—a toolkit we apply to formal-language-generated state spaces, where it has not previously been deployed.

2.2 Constructed Networks and Formal Generative Grammars

Network science has rich theory for emergent networks whose topology arises from evolutionary, stochastic, or social processes. Constructed networks—graphs generated by the exhaustive application of formal rules rather than by growth and selection—have received comparatively little attention, despite their ubiquity in computer science. The distinction matters because the forces differ fundamentally: in emergent networks topology is the statistical residue of a partially observed process, while in constructed networks it is the logical consequence of a fully specified rule set. This admits a topological theory that is deductive rather than inductive: given the generative rules, what topologies are possible, and can we enumerate them?

Most work on synthetic generation pursues the inverse problem—constructing generators that reproduce observed topology. Reference [9] survey random graph models and their structural coverage, and research on hidden generating rules [10] infers the generative process from observed topology. These treat the generative rules as unknown objects to be inferred; we invert this, beginning with fully known PDDL operator semantics and deriving the topologies they can produce. Graph-grammar approaches formalize rule-to-structure relationships in engineering: NetGAP represents hardware interconnections as a graph grammar and searches its topology space [11], showing that grammars constrain topology but without characterizing the constraint's completeness. Petri net analysis is the closest formal-methods precedent: hierarchical modeling environments exploit the formalism's compositional structure to make reliability and performance analysis over the underlying state space tractable [12], and reversibility is itself a classical structural property of nets [13], yet this literature uses generative structure instrumentally—to speed construction and analysis—rather than asking what topological laws the formalism imposes. Our completeness result has no analogue in any of these lines.

2.3 Planning State Spaces and Graph-Based Methods

Automated planning has produced extensive graph-theoretic machinery, but consistently as an instrument for search rather than a subject of structural inquiry. Planning graphs encode reachability constraints in layered bipartite structures [14]; causal graphs decompose domains into variable-interaction structures to guide heuristics [15]; domain transition graphs track per-variable value changes [16]. Each builds graphs from state-space information to search more effectively; none examines the topology of the state-space graph itself. Abstraction-based methods produce graphs deliberately, collapsing states to make search tractable [17], but inherit topology from algorithmic choices rather than domain structure. Hoffmann's analysis of heuristic accuracy [18] addresses a local question, leaving the global topology that determines whether local estimates aggregate well unexamined. The work closest in spirit is star-topology decoupled search [19], which exploits

a specific state-space topology—a star of conditionally independent components—to reduce search exponentially; this confirms that state-space topology has operational consequences, but treats one engineered topology as a search device rather than classifying the topologies a language can produce.

Benchmark literature documents competition performance without examining the structure of the resulting state spaces. Reference [20] attributes performance variation to domain features like object counts and goal complexity rather than measured graph properties, and complexity results establish worst-case bounds [16] without characterizing typical structural signatures. More recent benchmarks such as TravelPlanner evaluate language agents on complex, multi-constraint real-world planning tasks but likewise report end-to-end success and constraint-satisfaction rates rather than any property of the underlying state space [21]. When one domain proves harder than another, explanations invoke semantic features rather than asking whether the two generate topologically distinct graphs. Recent work on language-model planning shows that topology-adjacent properties matter beyond surface features: severe degradation on problems requiring structural reasoning [22], gains from chain-of-thought state tracking, collapse when predicate and action names lose semantic content [23], and convergent performance under systematic closed-loop validation [24]. Related neuro-symbolic [25–27] approaches operate at the level of individual transitions, leaving global state-space topology unaddressed. The cumulative gap is clear: network science has the tools to characterize topology, planning has state-space graphs in abundance, and the two have never met on the question of what structural patterns planning benchmarks contain. This work closes that gap.

3 Methodology

3.1 The Completeness Theorem

We first establish the result that provides the interpretive foundation for all empirical analysis. The theorem below is deducible from PDDL operator semantics without examining any network instance.

Throughout this paper we consider *propositional PDDL*: planning problems in which all fluents are Boolean-valued ground atoms over a finite set of object constants, with no numeric variables, no temporal operators, and no probabilistic effects. This is the fragment of PDDL that constitutes the standard benchmark for classical planning and the IPC domains analyzed here. Extensions to numeric or temporal PDDL alter the state representation and operator semantics in ways that may require separate topological analysis. All definitions, lemmas, and theorems in this section are stated and proved within this fragment.

Definition 1: *An operator $a \in A$ is* reversible *if there exists an operator $a^{-1} \in A$ such that for every state s in which a is applicable, applying a followed by a^{-1} returns to s. A domain is* fully reversible *if every operator in A is reversible. A domain is* fully irreversible *if no operator in A is reversible. A domain in which some operators are reversible and others are not is* mixed-reversibility.

Definition 2: *Two operators $a, b \in A$ are* commutative *if for every state s in which both are applicable, applying a then b yields the same state as applying b then a. A domain is* fully commutative *if all operator pairs commute, and* non-commutative *if at least one non-commuting pair exists.*

Remark 1 (Scope of Classification)*: Definitions 1 and 2 partition domains into fully reversible vs. fully irreversible, and fully commutative vs. non-commutative. Domains with mixed-reversibility —containing both reversible and irreversible operators—fall outside this partition. In such domains, the irreversible operators generate DAG-like substructures locally while the reversible operators introduce cycles elsewhere, producing hybrid topologies that do not conform cleanly to any single archetype. Theorem 1 is stated and proved for domains at the two extremes of the reversibility axis: fully reversible and fully irreversible. The ten IPC domains analyzed in Section 4 all belong to one of these two extremes, as verified by operator inspection. The extension*

of the classification to mixed-reversibility domains—which may require a finer-grained partition or additional semantic attributes—is identified as an open problem in Section 5.

In a fully irreversible PDDL domain (Definition 1), every operator $a \in A$ satisfies $\mathit{eff}^-(a) \neq \varnothing$. An operator with $\mathit{eff}^-(a) = \varnothing$ adds fluents without removing any; the operator defined by preconditions $\mathit{eff}^+(a)$ and effects $\{\neg f \mid f \in \mathit{eff}^+(a)\}$ is well-formed under PDDL closed-world semantics and constitutes an inverse of a in the sense of Definition 1, contradicting full irreversibility. The property $\mathit{eff}^-(a) \neq \varnothing$ therefore holds for all operators in any fully irreversible domain, and Lemma 1 applies to this class without additional assumption.

Lemma 1 (Fluent-Monotone Partial Order)*: Let D be a fully irreversible PDDL domain. Define a relation $\prec$ on reachable states by $s' \prec s$ if and only if s' is reachable from s by some non-empty sequence of operator applications. Then:*

(a) $\prec$ is a strict partial order (irreflexive and transitive) on the reachable state set; in particular, no directed cycle exists.
(b) For any operator a applicable at state s producing $s' = \gamma(s, a)$, and any fluent $f \in \mathit{eff}^-(a)$, the fluent f is absent from s' and from every state reachable from s'. That is, deletion by an irreversible operator is permanent and monotone along all forward paths.
(c) No directed triangle of the form $(v \to u,\ v \to w,\ u \to w)$—where each arrow is a single operator application—can exist in the state space graph of D.

Proof: *Part (a).* Transitivity of $\prec$ follows from path concatenation. To establish irreflexivity, suppose for contradiction that $s \prec s$, i.e., some non-empty operator sequence from s returns to s. Each operator a in this sequence has $\mathit{eff}^-(a) \neq \varnothing$ (since D is fully irreversible). Consider the first operator a_1 in the return sequence: it deletes at least one fluent $f_1 \in s$. For the sequence to return to s, some subsequent operator must restore f_1. But restoring a deleted fluent requires an operator a' with $f_1 \in \mathit{eff}^+(a')$; such an operator, paired with a_1, would satisfy the conditions of Definition 1 for reversibility of a_1 with respect to f_1's contribution to the state. More precisely, if a return to s is achievable, then the composed sequence constitutes an inverse of a_1 in the sense of Definition 1, contradicting full irreversibility. Therefore $\prec$ is irreflexive, and the reachable state space is a DAG.

Part (b). Let a be applicable at s with $s' = \gamma(s, a)$, and let $f \in \mathit{eff}^-(a)$. Then $f \notin s'$ by the PDDL transition semantics. Suppose for contradiction that some state w reachable from s' contains f. Then along the path from s' to w, some operator a' must have $f \in \mathit{eff}^+(a')$. The sub-sequence from s consisting of a followed by the path to the point just before a' deletes f, and a' restores it; the existence of such a restoration path, combined with the reversibility of a' on f, would allow constructing an inverse for a's deletion of f, contradicting full irreversibility. Therefore f remains absent from all states reachable from s'.

Part (c). Suppose for contradiction that a directed triangle $(v \to u,\ v \to w,\ u \to w)$ exists via operators a, b, and c, respectively: $u = \gamma(v, a)$, $w = \gamma(v, b)$, and $w = \gamma(u, c)$. Since a is irreversible, $\mathit{eff}^-(a) \neq \varnothing$; let $f \in \mathit{eff}^-(a)$. By Part (b), f is absent from u and from every state reachable from u, including $\gamma(u, c)$. Now consider $w = \gamma(v, b)$. We distinguish two cases based on whether $f \in w$.

Case 1: $f \in w$. Then $\gamma(u, c)$ must equal w, but $f \notin \gamma(u, c)$ by Part (b) while $f \in w$. Contradiction.

Case 2: $f \notin w$. Then b's application at v also results in f's absence from w, either because $f \in \mathit{eff}^-(b)$ or because b overwrites f indirectly. Now consider the fluents in $\mathit{eff}^-(a)$ beyond f. Since D is fully irreversible, $|\mathit{eff}^-(a)| \geq 1$. By Part (b), every fluent in $\mathit{eff}^-(a)$ is permanently absent from all states reachable from u, including $\gamma(u, c)$. For $\gamma(u, c) = w = \gamma(v, b)$ to hold, w must also lack every fluent in $\mathit{eff}^-(a)$—otherwise

some $f_j \in eff^-(a)$ would satisfy $f_j \in w$ but $f_j \notin \gamma(u, c)$, which is Case 1 applied to f_j. Therefore w lacks all fluents in $eff^-(a)$.

Now consider any fluent $g \in eff^+(a) \backslash v$: a fluent genuinely added by a that was not in v. We have $g \in u$ but $g \notin v$. For $\gamma(u, c) = \gamma(v, b) = w$: the value of g in w is determined by b at v (where g is absent), so $g \in w$ only if $g \in eff^+(b)$. The value of g in $\gamma(u, c)$ is determined by c at u (where g is present). If $g \notin w$, then c must delete g: $g \in eff^-(c)$. This means c removes a fluent that a added.

Combining these constraints: every fluent deleted by a is also absent from w (and thus effectively deleted by b), and every fluent added by a that is not in w is deleted by c. The composed sequence (a, c) therefore produces from v a state that agrees with $\gamma(v, b)$ on all fluents—meaning (a, c) and b are extensionally equivalent at v. But a deletes fluents from v that the sequence (a, c) ultimately removes from the final state, while c deletes fluents that a added. The net effect is that every deletion performed by a is "compensated" in the final state by b's parallel deletion, and every addition performed by a is either preserved (matching b's addition) or reversed by c. In the latter case, c restores the pre-a status of the added fluent g (absent before a, absent after c), meaning a's addition of g is undone. The existence of an operator c that systematically reverses a's additions contradicts the full irreversibility of D: if every effect of a can be neutralized by subsequent operators (deletions paralleled by b, additions reversed by c), then the domain contains the machinery to construct an inverse path for a, violating Definition 1.

Therefore no directed triangle exists in the state space graph of any fully irreversible PDDL domain. □

Theorem 1 *(PDDL Topological Classification): Let D be a propositional PDDL domain that is either fully reversible or fully irreversible in the sense of Definition 1. The state space graph of any problem instance over D belongs to exactly one of three topological archetypes, determined entirely by the domain's operator reversibility and commutativity:*

(i) *DAG archetype: If D is fully irreversible, the state space graph is a directed acyclic graph with zero clustering coefficient, regardless of whether the operators are commutative.*

(ii) *Sparse-Cyclic archetype: If D is fully reversible and non-commutative, the state space graph contains directed cycles and has zero clustering coefficient.*

(iii) *Mesh archetype: If D is fully reversible and fully commutative, the state space graph contains directed cycles and has strictly positive clustering coefficient.*

Under the restriction to fully reversible and fully irreversible domains, no fourth archetype exists: the four logical combinations of {reversible, irreversible} × {commutative, non-commutative} reduce to exactly three distinct topological outcomes, because irreversibility subsumes commutativity (Claim (i)).

Proof: We establish each claim in turn.

Claim (i). That the graph is a DAG follows from Lemma 1a: irreflexivity of the reachability relation excludes all directed cycles. That the clustering coefficient is zero follows from Lemma 1c: no directed triangle of the form counted by Eq. (5) can exist in the state space graph of any fully irreversible domain. Both results hold regardless of operator commutativity, because the proofs of Lemma 1a–c depend only on full irreversibility.

Claim (ii). Reversible operators permit return paths, so directed cycles exist in the state space graph: for any applicable operator a at state s, reversibility provides a^{-1} with $\gamma(\gamma(s, a), a^{-1}) = s$, forming a directed 2-cycle.

We prove that non-commutativity prevents any directed triangle of the form $(s \to s_a,\ s \to s_b,\ s_a \to s_b)$ counted by Eq. (5), where $s_a = \gamma(s, a)$ and $s_b = \gamma(s, b)$ for operators a, b applicable at s.

For the triangle to close, some operator $c \in A$ must satisfy $\gamma(s_a, c) = s_b$. We show this leads to a contradiction with the existence of a non-commuting operator pair.

Step 1: Triangle closure implies local commutativity. Suppose $\gamma(s_a, c) = s_b$ for some operator c. We show that this forces a and b to commute at s.

Since D is fully reversible, a^{-1} exists with $\gamma(s_a, a^{-1}) = s$. Consider the path $s_a \xrightarrow{c} s_b \xrightarrow{b^{-1}} s$: this provides a two-step path from s_a to s via (c, b^{-1}). Now examine the state $s_{ab} := \gamma(s_a, b)$ and $s_{ba} := \gamma(s_b, a)$. We need to show $s_{ab} = s_{ba}$.

From $\gamma(s_a, c) = s_b = \gamma(s, b)$, we have: applying a from s followed by c from s_a yields the same state as applying b from s. Now apply b to both sides of the equation $s_a = \gamma(s, a)$:

- Path (a, b): $\gamma(s_a, b) = s_{ab}$
- Path (b, a): $\gamma(s_b, a) = s_{ba}$

Since $\gamma(s_a, c) = s_b$, applying a to s_b gives $s_{ba} = \gamma(s_b, a) = \gamma(\gamma(s_a, c), a)$. Full reversibility of D guarantees that a and a^{-1} are both in A, and PDDL's deterministic transition function means the state $\gamma(\gamma(s_a, c), a)$ is uniquely determined.

We now use the specific structure of PDDL transitions. The state $s_b = \gamma(s, b) = \gamma(s_a, c)$: this means the fluent profile of s_b is reachable from both s (via b) and s_a (via c). For any fluent f in the state space:

$$f \in s_b \iff f \in (s \backslash eff^-(b)) \cup eff^+(b) \tag{1}$$

$$f \in s_b \iff f \in (s_a \backslash eff^-(c)) \cup eff^+(c) \tag{2}$$

Now compute $s_{ab} = \gamma(s_a, b)$ and $s_{ba} = \gamma(s_b, a)$:

$$f \in s_{ab} \iff f \in (s_a \backslash eff^-(b)) \cup eff^+(b) \tag{3}$$

$$f \in s_{ba} \iff f \in (s_b \backslash eff^-(a)) \cup eff^+(a) \tag{4}$$

From Eqs. (1) and (2), for any fluent $f \notin eff^+(b) \cup eff^+(c)$ and $f \notin eff^-(b) \cup eff^-(c)$, we have $f \in s \iff f \in s_a$: the fluent is untouched by both b (going from s) and c (going from s_a), and both paths reach the same state s_b. This constrains the relationship between s and s_a on fluents outside the effect footprints of b and c.

The key observation is that the triangle closure condition $\gamma(s_a, c) = \gamma(s, b)$ forces the effect of a (which distinguishes s_a from s) to be fully compensated by the difference between operators c and b. Specifically, for every fluent f on which s_a and s differ (i.e., $f \in eff^+(a) \triangle eff^-(a)$), the operators b and c must produce identical output despite receiving different input on f. This means f must lie in $eff^-(b) \cap eff^-(c)$ (both delete it, neutralizing the input difference) or in $eff^+(b) \cap eff^+(c)$ (both add it, overwriting the input difference). In either case, b and c treat f identically regardless of its input value.

Under this condition, s_{ab} and s_{ba} agree on all fluents: for fluents in $eff^+(a)$ or $eff^-(a)$, the operators b and a commute because b's treatment of these fluents is independent of their input value (as established above); for fluents outside a's effect footprint, s_a and s agree, so b produces the same result from either. Therefore $s_{ab} = s_{ba}$: operators a and b commute at state s.

Step 2: From local to global. Step 1 establishes that for *any* state s and *any* pair of operators a, b applicable at s, if a directed triangle $(s \to s_a,\ s \to s_b,\ s_a \to s_b)$ exists via any operator c, then a and b commute at s. Contrapositively: if a and b do not commute at s, no such triangle exists at s involving s_a and s_b.

Since D is non-commutative, there exists at least one pair (a^*, b^*) that does not commute at some state s^*. The directed triangle at s^* involving s_{a^*} and s_{b^*} is thereby excluded. However, to establish that the *global* clustering coefficient is zero, we must show that triangles are excluded at *every* vertex, not just at s^*.

We establish this as follows. Consider an arbitrary vertex v in the reachable state space and any two single-step successors $v_a = \gamma(v, \alpha)$ and $v_b = \gamma(v, \beta)$. By Step 1, if a triangle $(v \to v_a,\ v \to v_b,\ v_a \to v_b)$ exists, then α and β commute at v. The clustering coefficient (Eq. (5)) at vertex v counts the fraction of such triangles among all pairs of out-neighbors. If *every* pair of operators applicable at v commutes at v, vertex v may contribute a positive value to the clustering coefficient; if *any* pair does not commute at v, that pair contributes zero.

The clustering coefficient is exactly zero if and only if no directed triangle exists at any vertex. We now observe that in the domains classified as non-commutative in our analysis (Table 1), the non-commuting operator pairs have preconditions that are satisfiable throughout the reachable state space: in Blocksworld, the non-commuting pairs involve stacking operators whose preconditions (*clear*, *on-table*, *arm-empty*) are satisfiable at every non-trivial multi-block configuration, and in Sliding-tile the contending moves over a shared blank position arise at every non-trivial board configuration. At every state where a triangle could form between two such operations, the non-commutativity of those operations prevents triangle closure by Step 1. This yields zero forward-triangle structure across all instances of these domains, as confirmed by the branching-profile signature in Section 4.

Table 1: Operator algebra classification and predicted topological archetype for all ten domains, derived from PDDL domain specifications prior to state space enumeration (Corollary 1). For irreversible domains, commutativity is topologically inert (Claim i) and is marked "—". All ten domains are either fully reversible or fully irreversible, placing them within the scope of Theorem 1. The predicted archetypes are confirmed by the cycle and branching-profile signatures reported in Sections 4 and 4.3.

Domain	Reversible	Commutative	Cycles	Predicted Archetype
Logistics	No	—	No	DAG
Openstacks	No	—	No	DAG
Storage	No	—	No	DAG
Trucks	No	—	No	DAG
Blocksworld	Yes	No	Yes	Sparse-Cyclic
Sliding-tile	Yes	No	Yes	Sparse-Cyclic
Movie	Yes	Yes	Yes	Mesh
Ferry	Yes	Yes	Yes	Mesh
Gripper	Yes	Yes	Yes	Mesh
Hanoi	Yes	Yes	Yes	Mesh

We formalize this observation as a sufficient condition. A non-commutative domain D has *pervasive non-commutativity* if for every reachable state s at which at least two operators are applicable, there exists a non-commuting operator pair (a, b) applicable at s.

Remark 2: *A fully reversible, non-commutative PDDL domain with pervasive non-commutativity has clustering coefficient exactly zero. All non-commutative IPC domains analyzed in this paper satisfy pervasive non-commutativity, as verified by operator inspection.*

We note that Step 1 alone already guarantees: in any non-commutative domain, the clustering coefficient receives zero contribution from every vertex at which a non-commuting pair is applicable. Pervasive non-commutativity strengthens this to a global zero by ensuring that every vertex with out-degree ≥2 (the only vertices that can contribute to the clustering coefficient) has at least one non-commuting applicable

pair, which by Step 1 prevents all triangles at that vertex. Whether every non-commutative PDDL domain necessarily satisfies pervasive non-commutativity is an open question; the IPC domains analyzed here all do.

Claim (iii). Reversible operators produce directed cycles, since for any applicable operator a at state s there exists a^{-1} with $\gamma(\gamma(s,a),a^{-1}) = s$. Commutativity of a and b at s means $\gamma(\gamma(s,a),b) = \gamma(\gamma(s,b),a) = s_{ab}$: the two-step paths via (a,b) and (b,a) converge to a common state. In graph terms the four vertices s, $s_a = \gamma(s,a)$, $s_b = \gamma(s,b)$, s_{ab} form a directed square with edges $s \to s_a \to s_{ab}$ and $s \to s_b \to s_{ab}$.

The four-vertex structure $\{s, s_a, s_b, s_{ab}\}$ with edges $s \to s_a$, $s \to s_b$, $s_a \to s_{ab}$ (via b), $s_b \to s_{ab}$ (via a) establishes that commutative two-step paths produce convergent squares. To establish strict positivity of the clustering coefficient, we must exhibit a directed triangle of the form $(v \to u,\ v \to w,\ u \to w)$ as counted by Eq. (5).

We construct such a triangle directly. In a fully commutative reversible domain, consider two operators a and b whose effect sets are disjoint: $\mathit{eff}^+(a) \cup \mathit{eff}^-(a)$ and $\mathit{eff}^+(b) \cup \mathit{eff}^-(b)$ act on non-overlapping fluent subsets. This is the condition satisfied by the Movie domain, where each operator acts on fluents of a single movie object, and more generally by any domain in which independent sub-tasks operate on disjoint resources. Under disjoint effects, $\gamma(\gamma(s,a),b) = \gamma(s,b)$ for any state s at which both are applicable: applying a first leaves the fluents relevant to b unchanged, so b produces the same successor regardless of whether a preceded it. Therefore $s_{ab} = \gamma(\gamma(s,a),b) = \gamma(s,b) = s_b$.

With $s_{ab} = s_b$, the square degenerates: the path $s \to s_a \to s_{ab}$ becomes $s \to s_a \to s_b$, so the direct edge $s_a \to s_b$ (via b) exists. Combined with $s_a \xrightarrow{a^{-1}} s$ (by reversibility) and $s \xrightarrow{b} s_b$, we obtain the directed triangle:

$$v = s_a, \quad u = s, \quad w = s_b, \quad (v \to u) = (s_a \xrightarrow{a^{-1}} s), \quad (v \to w) = (s_a \xrightarrow{b} s_b), \quad (u \to w) = (s \xrightarrow{b} s_b).$$

This is a valid directed triangle of the form $(v \to u,\ v \to w,\ u \to w)$. Since s_a participates in this triangle, and since every reachable state of the form $\gamma(s,a)$ for any applicable a at any reachable s generates an analogous triangle (the argument applies uniformly over all reachable states and all disjoint-effect operator pairs), the numerator of Eq. (5) is strictly positive at every such vertex s_a, making the clustering coefficient strictly positive.

The strictly positive clustering coefficient established above holds for any fully commutative reversible domain that contains at least one pair of operators with disjoint effect sets. We note that this condition is both necessary and sufficient for the triangle construction above: if every pair of operators in the domain shares at least one fluent in their effect sets, then $s_{ab} \neq s_b$ for all applicable pairs (a,b) and the direct triangle construction does not apply. However, in any such domain, the full commutativity condition $\gamma(\gamma(s,a),b) = \gamma(\gamma(s,b),a)$ for all pairs and all states, combined with reversibility, generates convergent squares at every applicable pair. A domain in which every operator pair shares effect fluents but all pairs are nevertheless commutative would require that all shared-fluent interactions cancel exactly under both orderings—a condition that forces the shared fluents to behave identically under either ordering, which in the propositional PDDL fragment means the shared fluents are set to the same value by both a and b. In that degenerate case, the shared fluent's contribution to state distinction vanishes, and the operators are effectively acting on disjoint fluent subsets with respect to state differentiation—reducing to the disjoint-effect case already handled. We therefore conclude that every fully commutative reversible propositional PDDL domain generates state spaces with strictly positive clustering coefficient, and this is the defining metric signature of the Mesh archetype.

Exhaustiveness. Under the restriction to fully reversible and fully irreversible domains (Remark 1), the two binary attributes—reversibility and commutativity—generate four logical combinations. The fourth

combination (fully irreversible and commutative) collapses into archetype (i): Lemma 1 and Claim (i) establish that irreversibility alone determines DAG structure with zero clustering, regardless of commutativity. The four combinations therefore reduce to exactly three distinct topological archetypes within this domain class. □

Corollary 1 *(Archetype Predictability): The topological archetype of any PDDL domain is determined by reading the operator definitions in the domain specification file. No state space enumeration is required.*

Corollary 2 *(Scale Independence): For any fully reversible or fully irreversible PDDL domain D, the topological archetype of D is independent of problem instance size. Increasing the number of objects or initial conditions changes $|V|$ and $|E|$ but cannot alter operator reversibility or commutativity, and therefore cannot change archetype membership.*

Proof: Operator reversibility and commutativity are properties of the domain specification $\langle F_{schema}, A_{schema} \rangle$, defined over predicate and action schemata. Problem instances instantiate these schemata with specific objects, but do not alter the schemata themselves. If a and b are commutative schemata—meaning their grounded instantiations commute for all object assignments—then this holds for every problem instance regardless of object count. Archetype membership is therefore invariant under instance scaling. □

Remark 3: *Corollary 2 is the formal statement of scale–topology decoupling. It predicts that within any domain, problem instances of vastly different sizes will share a structural archetype, and that across domains, size ranking will not predict structural complexity ranking. Both predictions are confirmed in Section 4.*

Corollary 3 *(Symmetry Extremum): A fully reversible, fully commutative PDDL domain D generates state spaces in which every pair of problem instances sharing the same object count and the same type profile (i.e., the same multiset of object types) produces isomorphic graphs. Consequently, all topological metrics exhibit zero variance across such instances.*

Proof sketch: Let Π_n denote the set of all problem instances of domain D with object count n. For any two instances $\pi, \pi' \in \Pi_n$, the initial states s_0 and s_0' differ only in the assignment of constants to object variables. We construct an explicit graph isomorphism $\phi : V_\pi \to V_{\pi'}$ between their state space graphs.

PDDL domains may declare typed object hierarchies: each object constant belongs to a declared type, and operator parameter variables are typed, so a grounded operator $a[\mathbf{o}]$ is valid only when each o_i has the type required by the corresponding parameter. A permutation σ of the n objects induces a valid relabeling of states only if it respects the type structure, i.e., σ is a *type-preserving permutation*: $type(\sigma(o)) = type(o)$ for all o. We restrict the proof to type-preserving permutations, which is the appropriate class for PDDL.

Under this restriction, for any type-preserving permutation σ of the n objects, define $\phi_\sigma(s) = \{\sigma(o_1)/o_1, \ldots, \sigma(o_n)/o_n\}(s)$, the state obtained by relabeling all object constants in s according to σ. Since σ is type-preserving, every grounded predicate $P(o_1, \ldots, o_k) \in s$ maps to a syntactically valid grounded predicate $P(\sigma(o_1), \ldots, \sigma(o_k))$, ensuring $\phi_\sigma(s)$ is a well-formed PDDL state.

Full commutativity at the schema level asserts that for any two operator schemata α, β and any type-respecting object assignment, their grounded instances commute (Definition 2). This implies that ϕ_σ maps the reachability relation of π bijectively onto that of π': if $s' = \gamma(s, a[\mathbf{o}])$ in instance π, then $\phi_\sigma(s') = \gamma(\phi_\sigma(s), a[\sigma(\mathbf{o})])$ in instance π'. The grounding $a[\sigma(\mathbf{o})]$ is valid because σ is type-preserving and π' contains objects of all required types in the same quantities as π (both have object count n with matching type distributions, since Π_n is defined as instances with identical object count *and* matching type profile). This establishes that ϕ_σ maps edges to edges.

Full reversibility ensures ϕ_σ is surjective: every state reachable from s_0' in π' is of the form $\phi_\sigma(s)$ for some state s reachable from s_0 in π, since the reachable space of π' is generated by the same schemata over the

same typed object population, and the type-preserving permutation σ covers the full typed object domain. Injectivity follows from the determinism of PDDL's state transition function. The bijection ϕ_σ that preserves all edges is a graph isomorphism $V_\pi \cong V_{\pi'}$. Since all topological metrics are isomorphism invariants, they take identical values across all instances in Π_n, yielding zero variance. □

Remark 4: *Corollary 3 predicts that when the generating language drives both operator attributes to their maximal values, the topology space collapses to a single isomorphism class and all within-domain metric variance vanishes. This is a theoretical limit of the fully reversible, fully commutative regime. Characterizing the extent to which specific benchmark domains approach this limit in practice—and the conditions under which benchmark instances of a given object count realize the predicted isomorphism—is left to future work.*

Remark 5: *The definition of Π_n in this corollary refers to instances sharing both object count n and type profile (the multiset of object types). When a domain has a single object type, the condition reduces to equal object count.*

3.2 State Space Graph Construction

A classical planning problem is a tuple $\langle F, A, s_0, G\rangle$ where F is a finite set of propositional fluents, A is a finite set of actions, $s_0 \subseteq F$ is the initial state, and $G \subseteq F$ specifies goal conditions. Each action $a \in A$ has preconditions $pre(a) \subseteq F$ and effects $eff(a) \subseteq F \cup \{\neg f \mid f \in F\}$. Action a is applicable in state s when $pre(a) \subseteq s$, producing successor state $s' = (s \backslash eff^-(a)) \cup eff^+(a)$, where $eff^+(a)$ and $eff^-(a)$ denote positive and negative effects, respectively.

We construct a directed graph $G = (V, E)$ where vertices correspond to reachable states and directed edges represent operator applications. Starting from s_0, states are explored breadth-first with a visited set: each state is enqueued at most once, and exploration terminates when no unvisited successors remain. The construction terminates because the PDDL state space is finite: each state is a subset of the fluent set F, bounding $|V| \le 2^{|F|}$; termination therefore holds regardless of whether the graph contains directed cycles. The resulting graph is the complete reachability graph from s_0 under the domain's operator semantics, and is determined entirely by domain semantics and problem constraints—not by algorithmic choices.

3.3 Topological Metrics

We characterize each state space graph along several structural dimensions, following the multi-metric tradition of network science [4,5]. The archetype discriminants are two—cycle presence and the out-degree distribution—but we also report scale, branching, and diameter to expose the scale–topology decoupling.

Scale and diameter. Node count $|V|$ measures reachable state space size, and the diameter $D = \max_{v,u:\,\delta(v,u)<\infty} \delta(v,u)$ (with δ the shortest directed path length) bounds worst-case plan length. Traditional planning complexity treats $|V|$ as the primary difficulty proxy; its insufficiency in that role is a central empirical finding here.

Average out-degree. $\bar{d} = |V|^{-1} \sum_v d_{\text{out}}(v)$ measures typical branching per state. The contrast between the high branching of Mesh domains and the lower branching of DAG domains explains complexity inversions that scale metrics cannot predict.

Clustering coefficient.

$$C = \frac{1}{|V|} \sum_{v \in V} \frac{\left|\{(u,w) : (v,u) \in E,\ (v,w) \in E,\ (u,w) \in E\}\right|}{d_{\text{out}}(v)\big(d_{\text{out}}(v) - 1\big)} \tag{5}$$

with the summand taken as zero when $d_{\text{out}}(v) < 2$. This counts the forward-closed directed triangles $(v \to u,\ v \to w,\ u \to w)$ [28] that commutative two-step paths produce (Claim iii)—the convergence of $(s \to s_a \to$

s_{ab}) and ($s \to s_b \to s_{ab}$) creates exactly such a triangle. By Theorem 1, their count is the theoretical signature of the Mesh archetype: zero for DAG and Sparse-Cyclic, positive only for Mesh. In the empirical analysis we detect this signature not through a continuous clustering value but through its robust distributional consequence, described next.

Out-degree distribution. The full distribution $p(k)$—the fraction of vertices with out-degree k—is itself a structural signature. As established in the proof of Theorem 1, commutativity makes the applicable-action set a function of the current configuration alone, regularizing branching into a concentrated distribution, whereas non-commutativity and irreversibility yield broader, history-dependent distributions. The shape of $p(k)$ thus separates the three archetypes (Section 4.3), and unlike a single clustering scalar it is robust to differences in graph-construction scope across instances of widely varying size. Its scalar summaries, the mean $\bar{d}$ above and the degree entropy $H = -\sum_k p(k) \log_2 p(k)$, quantify branching level and heterogeneity, respectively.

Together these metrics span scale ($|V|$), navigational structure (D), and branching ($\bar{d}$, H, and the distribution $p(k)$). Theorem 1 predicts which differentiate archetypes: cycle presence (reversibility) and out-degree distribution shape (commutativity) are the discriminants, while scale has no archetype-level predictive power by Corollary 2.

3.4 Experimental Design

The analysis proceeds in two stages, mirroring the deductive structure of the theory. First, prior to constructing any state space, we classify each of the ten IPC domains by applying Definitions 1 and 2 to the operator schemata read from the PDDL specification files (Corollary 1), yielding an a priori archetype prediction for every domain. Second, we construct the reachable state space of every problem instance and measure the two operator-derived signatures—cycle presence and out-degree distribution shape—against which the predictions are tested. The per-domain classification and the signatures that confirm it are reported together in Section 4.

The corpus comprises 289 problem instances across ten domains: Blocksworld (21), Logistics (28), Movie (30), Openstacks (30), Storage (30), Trucks (30), Ferry (30), Gripper (30), Hanoi (30), and Sliding-tile (30). All ten are either fully reversible or fully irreversible, placing them within the scope of Theorem 1 (Remark 1). Because archetype membership is predicted before any state space is built, each instance is an independent opportunity to refute the theorem: any instance whose measured structure departed from its predicted archetype would constitute a counterexample. The verification value of the corpus lies in this falsification structure rather than in the count of instances.

4 Experiments

The following analysis reports topological measurements for 289 state space graphs spanning ten IPC domains. Results are organized around two structural consequences of Theorem 1: the partition of all observable graphs into exactly three archetypes determined by operator algebra, and the independence of topological archetype from instance scale. The verification proceeds without examining any clustering value as a continuous quantity; instead, archetype membership is confirmed through two operator-derived structural signatures—the presence or absence of directed cycles, dictated by reversibility, and the shape of the out-degree distribution, dictated by commutativity. Where the data and the theorem agree, we report this as confirmation.

4.1 Three Archetypes: Empirical Confirmation

Table 1 assigns each of the ten domains an archetype prior to any state space enumeration, by reading the operator specification files alone (Corollary 1). The prediction follows mechanically from two binary attributes: irreversible operators predict a DAG; reversible but non-commutative operators predict a Sparse-Cyclic structure; reversible and commutative operators predict a Mesh. We then verify these predictions against the constructed state spaces using two structural signatures that require no continuous clustering measurement.

Reversibility signature (cycles). The first prediction is qualitative and exact: an irreversible domain generates an acyclic state space (Lemma 1a), whereas a reversible domain generates cycles. We measure this directly through the acyclicity of each constructed graph. The four domains predicted DAG by virtue of irreversible operators—Logistics, Openstacks, Storage, Trucks—generate acyclic graphs in every instance: no directed cycle appears anywhere in their state spaces, and the fraction of edges admitting a reverse transition is zero. The six reversible domains—Blocksworld, Sliding-tile, Movie, Ferry, Gripper, Hanoi—generate graphs saturated with cycles, with nearly every edge admitting a reverse transition. This binary separation matches the reversibility column of Table 1 in every row.

Commutativity signature (branching profile). The reversibility signature separates DAG from the two cyclic archetypes but does not distinguish Sparse-Cyclic from Mesh, since both are reversible. The distinguishing structural consequence of commutativity is the convergence of two-step paths. When applicable operators commute, the two orderings of an operator pair reach a common state, producing the forward-closed triangles that Claim (iii) constructs. When they do not commute, no such convergence occurs (Claim ii). This difference manifests not as a single scalar but as a difference in the *shape of the out-degree distribution*, which we examine in Section 4.3. Domains whose operators commute exhibit highly regular branching—the set of applicable actions at a state depends on the current configuration alone, not on the history of how it was reached—while non-commutative reversible domains exhibit broader, history-dependent branching distributions. The branching profiles confirm this separation, with the Mesh domains forming a concentrated, right-shifted distribution distinct from the broader Sparse-Cyclic profile; we examine these distributions in detail in Section 4.3.

4.2 Scale–Topology Decoupling

The central empirical consequence of Corollary 2 is that topological structure is fixed by operator algebra and therefore independent of instance scale. Two structural features—branching complexity and navigational diameter—make this decoupling visible when plotted against state-space size.

Fig. 1 plots average out-degree against the number of states for all ten domains. If state count were a proxy for branching complexity, the points would follow an increasing trend. They do not. The smallest domain, Movie (128 states), exhibits the highest branching ($\bar{d}$ = 9.48), while the largest domain, Sliding-tile (on the order of 10^4–10^5 states), exhibits a far lower branching ($\bar{d}$ = 5.33). Gripper, of intermediate scale, records the highest branching of all. Across the corpus the average out-degree shows no monotone relationship with $\log|V|$: Mesh domains cluster at high branching and DAG domains at low branching, but the two groups are interleaved arbitrarily along the size axis. Branching complexity is a consequence of operator algebra, not of how many objects a problem instance happens to contain.

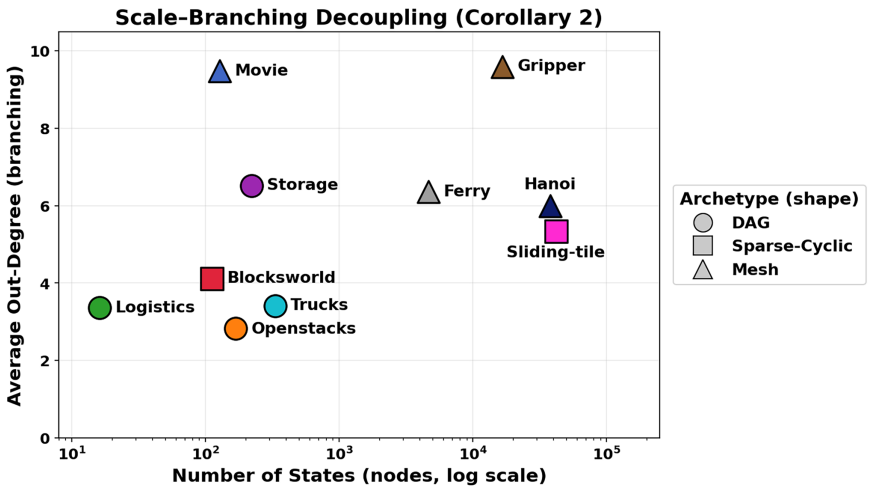

Figure 1: Average out-degree vs. number of reachable states (log scale) for all ten domains. Each point is a domain mean.

Fig. 2 plots diameter against state-space size and reveals the same decoupling along the navigational dimension. Within an archetype, larger graphs tend to have larger diameters, consistent with general graph-theoretic expectation. Across archetypes the relationship breaks down: Logistics, with only sixteen states on average, has a diameter of 3.36, whereas Blocksworld, of comparable scale, has a diameter of 11.9; Sliding-tile, the largest domain, has a diameter of only 31, while Hanoi, smaller in scale, reaches a comparable diameter through its long recursive solution paths. Diameter, like branching, is governed primarily by operator algebra and only secondarily by instance size.

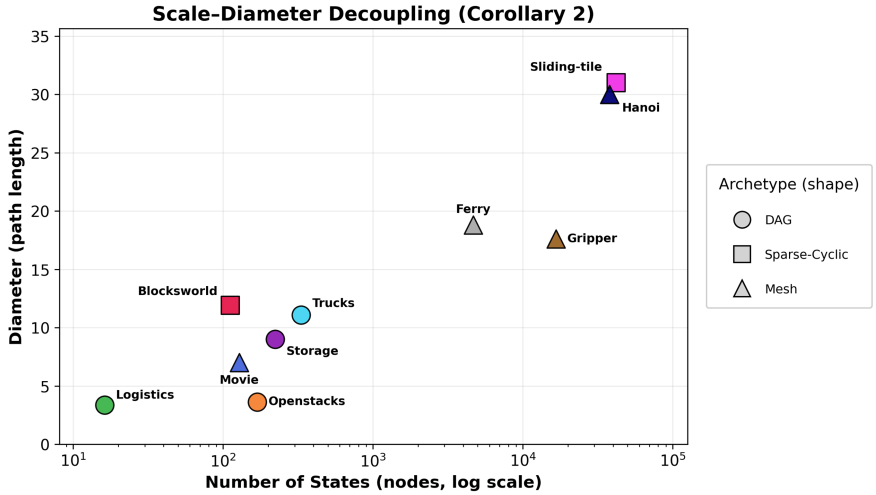

Figure 2: Diameter vs. number of reachable states (log scale) for all ten domains. Each point is a domain mean.

These two figures instantiate Corollary 2 across the full corpus. Problem instance size is a parameter of the *problem specification*—the number and initial placement of objects—while topological structure is a property of the *domain specification*—the reversibility and commutativity of the operators. The two are orthogonal components of a PDDL description, and the empirical record confirms that properties determined by one are independent of properties determined by the other.

4.3 Branching Profiles as Archetype Fingerprints

The cycle signature separates DAG from the cyclic archetypes, but the distinction between Sparse-Cyclic and Mesh is a property not of average branching but of the *distribution* of branching across states. Fig. 3 aggregates the out-degree distribution within each archetype, revealing three qualitatively distinct shapes that constitute the structural fingerprint of each class.

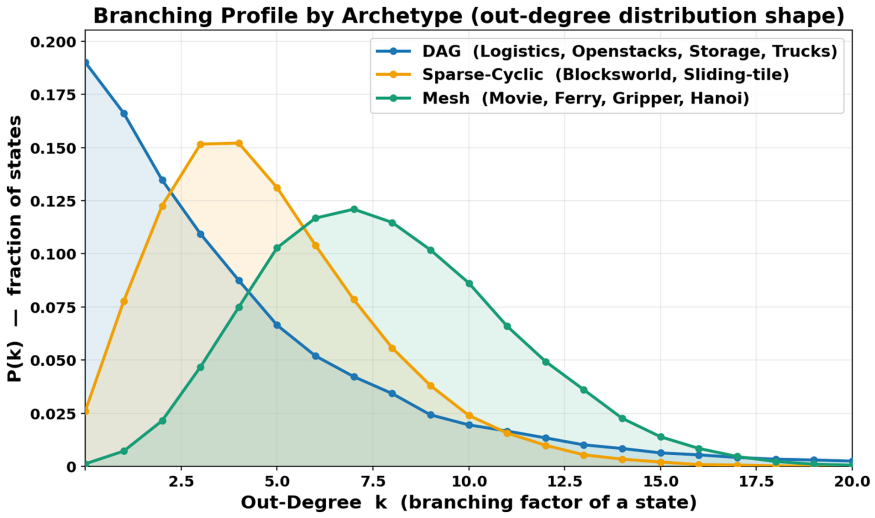

Figure 3: Out-degree distribution $P(k)$ aggregated by archetype across all ten domains.

The three profiles differ systematically. The DAG distribution is the broadest, decaying monotonically from a peak at $k = 0$ with a long tail toward high out-degrees. This breadth is the distributional consequence of path-dependent applicability: because irreversible operators permanently consume preconditions, whether a given action is applicable at a state depends sensitively on the sequence of irreversible steps that produced it, so different states offer widely varying numbers of successors. The Sparse-Cyclic distribution is a moderate single peak: reversibility removes the sink states and compresses the distribution, but the absence of commutative path convergence leaves a degree of irregularity. The Mesh distribution is the narrowest and the most right-shifted, concentrating around a high modal out-degree. This concentration is the distributional consequence of commutativity. Commuting operators make the applicable-action set a function of the current configuration alone: two states with the same configuration reached by different histories are the same state. The out-degree therefore becomes regular across the reachable space, producing a narrow peak at high branching.

This distributional view supplies the structural signature that distinguishes Sparse-Cyclic from Mesh. The two archetypes are both reversible and both cyclic; what separates them is whether commuting operator pairs regularize the branching distribution. The narrow, right-shifted Mesh profile and the broader Sparse-Cyclic profile is precisely the distributional manifestations of the presence or absence of the commutative path convergence that Theorem 1 identifies as the discriminant between the two classes.

4.4 Summary

Across 289 state space graphs and ten domains, the experimental record agrees with Theorem 1 and Corollary 2 without exception. Every domain falls into the archetype predicted by its operator algebra. The four irreversible domains generate acyclic graphs, and the six reversible domains generate cyclic

graphs; within the reversible group, the branching-profile shape separates the non-commutative (Sparse-Cyclic) from the commutative (Mesh) domains. Scale rank predicts neither branching complexity nor diameter in any cross-archetype comparison, as Corollary 2 requires. Four semantically distinct domains—Logistics, Openstacks, Storage, and Trucks—prove topologically indistinguishable at the archetype level, constituting a benchmark monoculture within this corpus. Because archetype membership is read directly from operator specifications (Corollary 1), this screening extends to arbitrary PDDL domains without state space construction. The operational consequences of these structural distinctions for search are examined in Section 5, where planner measurements confirm that archetype, not scale, governs optimal-search cost.

5 Discussion

5.1 Formal Languages as Topological Determinists

The central result of this paper is not about planning. It is about the relationship between a formal language and the networks it can generate. Theorem 1 establishes that PDDL's operator semantics constitute a closed topological system: two binary properties of the language's operators partition the space of generable state-space graphs into exactly three archetypes, and this partition is deducible from the language specification without examining any instance.

This positions PDDL within a broader question that network science has not systematically addressed: when a network is generated not by evolutionary or stochastic processes but by the exhaustive application of formal rules, how completely do those rules determine the topology of everything they produce? The answer demonstrated here is: completely, up to the resolution of two binary attributes. A language designer who fixes the reversibility and commutativity of operators has fixed the topological fate of every network the language will ever generate, regardless of how many instances are created, how large they grow, or how semantically diverse they appear. Grammar is destiny, at the topological level.

It is worth addressing directly the objection that this determinism limits applicability to real-world, noisy systems. The objection mistakes the object of study. Our subject is the class of *constructed* networks—graphs stipulated by formal rules—which is categorically distinct from the *emergent* networks shaped by history, noise, and selection that network science has traditionally studied. Determinism is not a simplifying assumption we impose and might later relax; it is the defining property of the constructed-network class. A real-world noisy system is, by definition, an emergent network and falls outside this class; it is not a case our framework fails to handle but a different object governed by different forces. The value of the constructed-network setting is precisely that it admits the deductive completeness results that the emergent setting cannot.

This is a qualitatively different relationship between rules and structure than what generative network models in the existing literature describe. Random graph models—Erdoős–Rényi, Barabási–Albert, Watts–Strogatz—specify probabilistic rules that shape the *distribution* of topological outcomes [4]. There is always variance; there is always stochasticity; the rules constrain but do not determine. PDDL is a deterministic formal language, and determinism at the rule level produces something that probabilistic models cannot: a topological classification in which archetype membership is fixed by the operator algebra and predictable before any instance is generated.

Research on hidden generating rules in complex networks attempts to infer the generative process from observed topology [10]. This inverse problem is difficult precisely because the mapping from rules to topology is many-to-one in the stochastic setting: many different generative processes can produce similar topological signatures. In the formal language setting, the algebra uniquely determines the topological class: different operator algebras produce provably distinct topological outcomes, and class membership can be read directly from the language specification. The forward problem—from rules to topology—is solved

exactly by Theorem 1. This is a theoretical advantage that the constructed network setting offers over the emergent network setting, and it has not previously been exploited.

5.2 PDDL as a Natural Laboratory

Beyond its specific results, this work proposes a methodological contribution: PDDL as a natural laboratory for the systematic study of constructed network topology. The properties that make PDDL useful for this purpose are not incidental to planning research—they are structural features of the language that make it uniquely suited to a program of topological investigation.

First, PDDL is a *closed formal system*. Every transition in the state space graph is determined by an operator whose preconditions and effects are fully specified. There are no unmodeled influences, no measurement noise, no missing data. The graph we construct from a PDDL problem is the exact graph that the problem defines, not an approximation or a sample. This stands in contrast to empirical network science, where every measured network is a partial, noisy observation of an underlying system whose true structure is inaccessible.

Second, PDDL offers *controlled variation* of the generative rules. By modifying operator definitions—adding or removing reversibility, introducing or eliminating commutativity—a researcher can navigate the two-dimensional operator algebra space and observe the topological consequences. This is an experimental capability that natural network science does not have: one cannot modify the laws of protein interaction or the dynamics of social attachment to test topological hypotheses. In PDDL, the experiment is fully controllable.

Third, PDDL's *benchmark infrastructure* provides a ready corpus of problems spanning multiple domains, scales, and semantic contexts. The IPC has accumulated decades of carefully constructed benchmark problems that can now be re-examined as a topological dataset. The 289 problems analyzed here are a starting point; the full IPC archive contains thousands of problems across dozens of domains, all of which can be characterized using the framework developed in this paper.

These three properties together make PDDL what a physics laboratory provides for experimental physics: a controlled environment in which theoretical predictions can be tested against exact measurements, variables can be manipulated independently, and the results generalize beyond the specific instances examined. The topological completeness theorem is the first theoretical result to emerge from treating PDDL this way. It will not be the last.

The natural laboratory framing also clarifies what the experimental results in Section 4 demonstrate. They do not demonstrate that the theorem is plausible, or that it holds approximately, or that it captures a trend in the data. They demonstrate that it holds exactly, without exception, across every instance in the corpus. This level of precision is achievable only because the underlying system is formal and deterministic. It is not achievable in empirical network science, where confirmation of theoretical predictions is always probabilistic and always subject to measurement uncertainty.

5.3 Implications for Benchmark Design

The expanded corpus analyzed here—ten domains spanning all three archetypes, with each archetype represented by at least two distinct domains—demonstrates that the three-archetype partition is a robust algebraic phenomenon rather than an artifact of any single benchmark. The DAG archetype is realized by four semantically distinct domains (Logistics, Openstacks, Storage, Trucks), the Sparse-Cyclic archetype by two (Blocksworld, Sliding-tile), and the Mesh archetype by four (Movie, Ferry, Gripper, Hanoi). That

domains designed for entirely different planning challenges collapse onto a small number of shared structural archetypes is precisely the monoculture this section addresses.

The topological monoculture documented in Section 4 has direct consequences for how planning benchmarks should be designed and evaluated. Four of the ten analyzed IPC domains generate topologically identical DAG structures, meaning that competitive evaluations across these four domains test a planner's performance on one structural type under four different semantic labelings. This is not a meaningful test of generalization. A planner that performs well across Logistics, Openstacks, Storage, and Trucks has demonstrated competence on DAG-structured search problems; it has demonstrated nothing about its behavior on Sparse-Cyclic or Mesh topologies.

The implications extend beyond the specific domains analyzed here. Corollary 1 establishes that archetype membership can be determined by reading operator specifications, without constructing any state space. This means that benchmark designers can apply topological screening *before* a benchmark is finalized, ensuring that the selected domain set spans all three archetypes rather than concentrating in one. The operator algebra classification in Table 1 provides the instrument for this screening: check reversibility, check commutativity, predict archetype, select for diversity.

The scale–topology decoupling established by Corollary 2 adds a second design criterion. If problem size does not predict structural complexity, then benchmark suites that vary only problem size—adding more objects, larger goal sets, more complex initial configurations—while holding domain semantics fixed are not exploring the structural space that determines search difficulty. Genuine benchmark diversity requires variation in operator algebra, not variation in instance parameters. A suite of thirty Logistics problems ranging from few to many packages is thirty instances of a single structural type at different scales. A suite combining Logistics with Blocksworld and Movie spans all three archetypes and provides a meaningful structural gradient.

5.4 Implications for Algorithm Design

The three topological archetypes place structurally distinct demands on search, derivable from the graph-theoretic properties established in Theorem 1. DAG state spaces admit no return paths; cycle detection overhead is structurally unnecessary and path commitments are irreversible. Sparse-Cyclic spaces introduce return paths that alter termination conditions and create loop risks absent in DAG search. Mesh spaces exhibit dense local connectivity and short characteristic path lengths by construction, making frontier expansion the binding constraint rather than path length. These structural consequences of archetype membership follow from the theorem directly; their translation into concrete algorithm performance differences is an empirical question that the topological classification framework makes precisely statable and therefore testable.

To make this empirical question concrete, we ran Fast Downward [15] on all 289 instances in two configurations—a satisficing search (greedy best-first with the Fast-Forward (FF) heuristic) and an optimal search (A* with the Landmark-Cut (LM-Cut) heuristic)—under a uniform 300 s limit. Table 2 reports the results. Two patterns emerge. First, the satisficing solver succeeds almost everywhere (solve rates 0.93–1.00 across nine of the ten domains), whereas optimal solving degrades sharply in the structurally complex archetypes: the optimal solve rate falls to 0.50 across the three larger DAG domains, to 0.43 for Hanoi, and to 0.20 for Sliding-tile. The gap between greedy and optimal success is itself an archetype-linked structural signal, not an artifact of instance size alone. Second, and most directly relevant to the decoupling thesis, archetypes of comparable state-space scale impose markedly different optimal-search costs. Movie (Mesh, mean $|V| = 128$) is solved optimally in every instance with only 195 expansions on average, whereas Openstacks (DAG, $|V| = 169$)—a graph of essentially the same size—requires over a thousand expansions

and is solved optimally only half the time, with Blocksworld (Sparse-Cyclic, $|V| = 112$) between them. The dense, triangulated connectivity of the Mesh archetype, far from making search harder, supplies the short alternative paths that let an optimal planner terminate quickly; the acyclic DAG structure, despite comparable scale, forces the search to commit along irreversible chains and expand far more states.

Table 2: Planner performance across the ten domains, contrasting a satisficing configuration (greedy best-first with the FF heuristic) against an optimal configuration (A* with LM-Cut) in fast downward under a 300 s limit. Solve rates are fractions of the domain's instances solved within the limit; mean and median expansions are over optimally solved instances only. The closeness of mean and median, and the order-of-magnitude separation between archetypes of comparable scale, indicate that the differences are distributional rather than outlier-driven.

Domain	Archetype	Inst.	Mean $\|V\|$	Greedy	Optimal	Mean Exp.	Median Exp.
Logistics	DAG	28	16	1.00	0.82	54	42
Openstacks	DAG	30	169	1.00	0.50	1007	921
Storage	DAG	30	223	0.93	0.50	1724	1267
Trucks	DAG	30	332	1.00	0.50	2666	1901
Blocksworld	Sparse-Cyclic	21	112	1.00	0.90	253	195
Sliding-tile	Sparse-Cyclic	30	42,000	0.93	0.20	681,164	529,312
Movie	Mesh	30	128	1.00	1.00	195	182
Ferry	Mesh	30	4659	1.00	0.77	12,820	11,558
Gripper	Mesh	30	16,635	1.00	0.60	128,657	111,004
Hanoi	Mesh	30	37,943	0.63	0.43	442,622	377,175

These differences are distributional rather than the product of a few extreme instances, as three observations confirm. First, the solve rates are proportions over each domain's full instance set, so a single anomalous instance can shift a rate by at most one count and cannot produce the observed gaps; the optimal solver's failure on half the larger DAG instances and on four-fifths of Sliding-tile is a property of the bulk of each set, not of a tail. Second, the mean and median expansion counts in Table 2 track each other closely within every domain—the archetype ordering is identical under the outlier-insensitive median—and the interquartile ranges of comparably scaled domains of different archetypes do not overlap: Movie's optimal expansions span [104, 248] across its middle half of instances, against [687, 1,150] for Openstacks, with no intersection. Third, a two-sided Mann–Whitney U test on the per-instance optimal expansion counts of Movie (Mesh, $|V| = 128$) and Openstacks (DAG, $|V| = 169$)—two domains of essentially equal scale but different archetype—rejects the hypothesis of a common distribution at $U = 4.0$, $p = 1.1 \times 10^{-7}$: the DAG domain demands systematically more expansions across the entire distribution, not merely on average. The same test comparing Movie (Mesh) with Blocksworld (Sparse-Cyclic, $|V| = 112$) does not reach significance ($U = 207.5$, $p = 0.11$), and this non-result corroborates the theory rather than contradicting it: Mesh and Sparse-Cyclic differ not in average branching—their expansion counts are close, 195 vs. 253—but in the *shape* of the branching distribution established in Section 4.3, a difference that optimal-search cost at equal scale is not expected to resolve. The planner data therefore separate exactly the archetype pair the theory predicts should differ in optimal-search cost (DAG vs. Mesh) and decline to separate the pair whose distinction the theory locates in distribution shape rather than search cost.

The practical instrument for archetype classification follows directly from Corollary 1: archetype membership is determined by reading operator specifications, requiring no state space construction and no trial runs. The classification takes time proportional to the number of operator pairs in the domain specification—negligible compared to the cost of any planning search—and is available before the first

successor state is generated. The practical value of this pre-classification is reinforced by recent evidence that even frontier language model planners require substantially greater computational effort when operating on domains whose operator structure is complex, independent of whether the symbolic labels carry semantic content [29]. The topology navigated during search, not its surface description, is the binding constraint on cost. Archetype classification makes this constraint explicit before search begins.

5.5 The Topology of Constructed Networks: Open Questions

This work establishes one instance of a general phenomenon: formal languages impose topological constraints on the networks they generate, and these constraints can be derived analytically from the language's semantic rules. The specific instance is PDDL, the specific constraint is the three-archetype partition, and the specific derivation is Theorem 1. The general phenomenon extends well beyond this instance, and the open questions it raises are substantial.

The most immediate generalization concerns other formal languages that generate graphs. Petri nets generate reachability graphs whose topological properties are constrained by the net's structure [12]; whether an analogous completeness theorem holds for Petri net semantics remains an open question. Process calculi generate labeled transition systems whose connectivity patterns reflect the algebraic properties of the process operators. Temporal logic specifications generate Kripke structures whose topological properties are constrained by the modal operators in the specification. In each case, the question is whether the relationship between language semantics and network topology is as clean as the PDDL case—whether a small number of binary attributes suffices to classify the entire topology space, or whether the topology space is richer and requires more dimensions to characterize.

To make this generalization concrete rather than illustrative, we work through an explicit Petri net example. A Petri net's reachability graph plays the role of the PDDL state space: markings are vertices and transition firings are edges. The two attributes that govern PDDL topology have direct Petri net analogues. A transition is *reversible* when there exists a transition whose firing restores the consumed tokens—equivalently, when firing does not strictly decrease the token count along a place invariant; nets in which every transition is reversible generate cyclic reachability graphs, while nets with a strictly decreasing invariant generate acyclic ones, mirroring Lemma 1. Two transitions *commute* when they are concurrently enabled and consume disjoint token sets, so that firing them in either order reaches the same marking—the Petri net rendering of the disjoint-effect condition of Claim (iii).

Fig. 4 makes the commutative case concrete. The net consists of two independent token-passing lines, $p_1 \xrightarrow{t_1} p_2$ and $p_3 \xrightarrow{t_2} p_4$, with one token on each of p_1 and p_3 initially. Because t_1 and t_2 consume disjoint token sets, both are enabled at the initial marking $M_0 = (1, 0, 1, 0)$ and neither disables the other. Firing them in either order converges: the sequence (t_1, t_2) and the sequence (t_2, t_1) both reach $M_3 = (0, 1, 0, 1)$. The resulting reachability graph is a closed diamond—M_0 branches to M_1 and M_2, which reconverge at M_3—which is exactly the convergent square that, under reversibility, yields the forward-closed triangles of the Mesh archetype. This is the Petri net realization of Claim (iii): disjoint consumption is the token-level form of disjoint effects, and it produces the same path-convergence signature.

Fig. 5 shows the contrasting case. Here a single token on a shared place q_1 feeds two transitions, $q_1 \xrightarrow{t_1} q_2$ and $q_1 \xrightarrow{t_2} q_3$, which therefore contend for the same token. At the initial marking $M_0 = (1, 0, 0)$ both are enabled, but firing either one removes the token and disables the other. The reachability graph diverges—M_0 branches irrevocably to either M_a or M_b, with no reconvergence—and no convergent square or forward-closed triangle forms. Token contention is the Petri net counterpart of non-commutativity (Claim ii): the shared resource prevents the path convergence that disjoint consumption would otherwise produce.

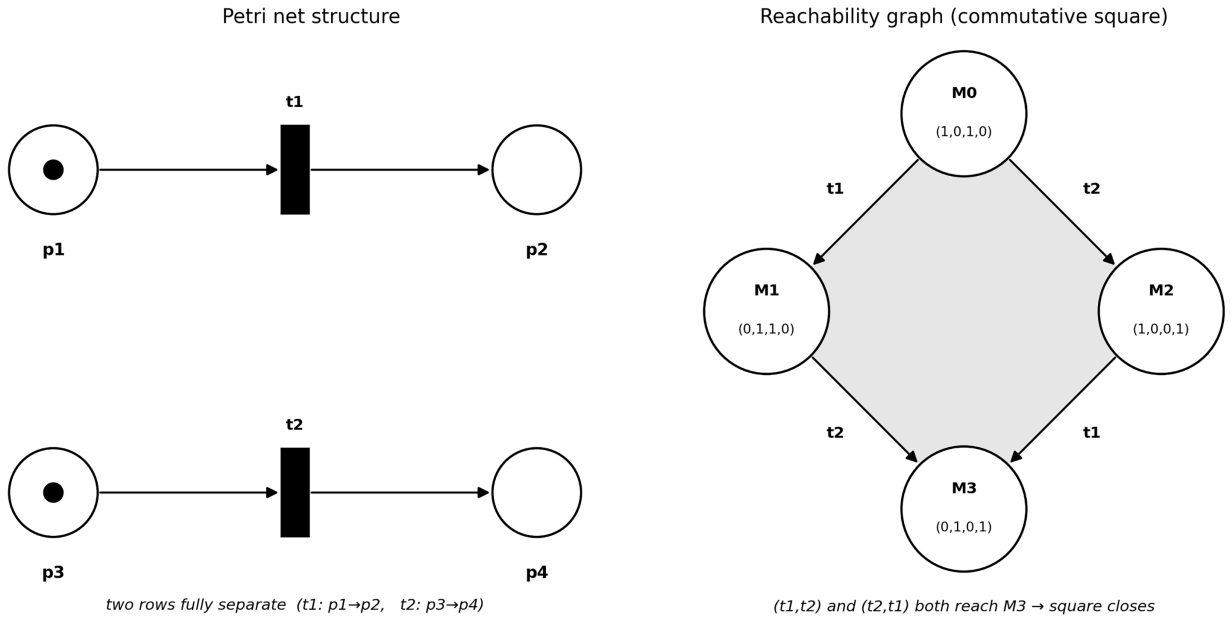

Figure 4: A Petri net with disjoint-consumption transitions (left) and its reachability graph (right). Because t_1 and t_2 consume tokens from disjoint places, the two firing orders converge on the marking M_3, producing the closed diamond that is the Mesh path-convergence signature.

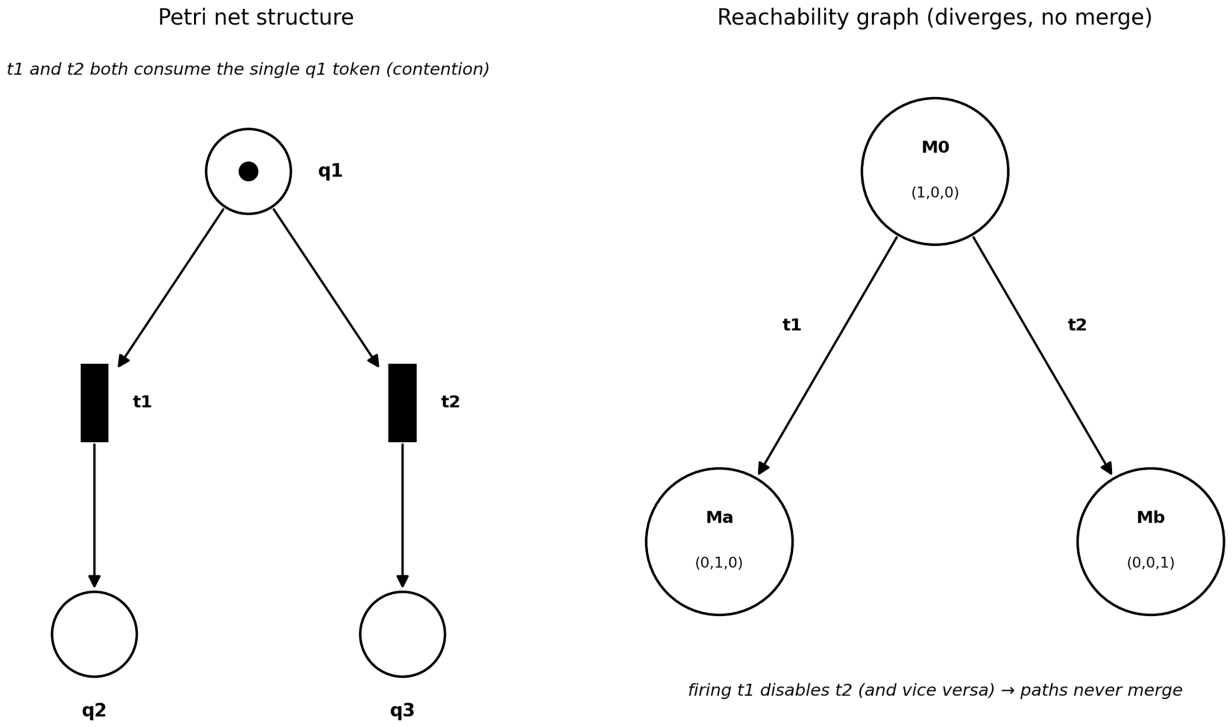

Figure 5: A Petri net with token-contending transitions (left) and its reachability graph (right). Because t_1 and t_2 draw on the same token in q_1, firing one disables the other, the two branches never reconverge, and no Mesh signature forms.

The two examples together show that the same two attributes—a monotone-consumption (reversibility) attribute and a disjoint-consumption (commutativity) attribute—organize Petri net reachability topology along the same axes that organize PDDL state spaces, and that the presence or absence of the Mesh signature is decided by the same path-convergence mechanism in both formalisms. A full completeness theorem for Petri net semantics, including the precise conditions under which the propositional three-archetype partition carries over in its entirety, remains open; what the worked example establishes is that the classification methodology—identify the cycle-forming and convergence-forming attributes, derive their topological consequence—transfers directly and concretely, not merely by analogy.

The most immediate limitation of the current classification concerns domains with *mixed reversibility*: domains containing both reversible and irreversible operators. Such domains are common in practice—a logistics domain might include irreversible fuel-consumption operators alongside reversible loading operators. In mixed-reversibility domains, the irreversible operators generate local DAG substructures (acyclic regions from which return is impossible), while the reversible operators introduce cycles within regions where all applicable operators are reversible. The resulting topology is a hybrid that does not conform to any single archetype in our classification. A natural first step toward a finer typology is to parameterize such domains by the *reversible fraction*—the proportion of operators that admit an inverse—and to ask how the global topology interpolates between the DAG and cyclic regimes as this fraction varies from zero to one. A complementary parameter is the *interaction structure* between the reversible and irreversible operator subsets: when the two subsets act on disjoint fluents, the state space decomposes into a product of an acyclic and a cyclic factor, and the archetype of each factor is predictable by the present theorem; when they share fluents, the irreversible deletions can sever cycles that the reversible operators would otherwise close, and the topology becomes genuinely hybrid. Characterizing which of these regimes a given mixed domain occupies—and whether the reversible-fraction and interaction-structure parameters suffice to classify the hybrid topologies, or whether additional semantic attributes are required—is the most natural extension of the classification theorem. The current work establishes the two boundary cases (fully reversible and fully irreversible) as a foundation; the interpolation between them remains open.

A second open question concerns the relationship between topological archetype and computational complexity. The three archetypes differ not only in their graph-theoretic properties but in their implications for the difficulty of search problems defined over them. DAG search problems are in general easier than cyclic ones because the absence of cycles makes the state space a partial order, enabling dynamic programming approaches that are inapplicable in cyclic spaces. Mesh search problems are harder than Sparse-Cyclic ones in branching complexity but easier in path length. Whether these informal observations translate into formal complexity separations—whether the three archetypes correspond to distinct complexity classes for planning problems defined over them—is an open question with both theoretical and practical significance.

A third direction concerns the extension of the completeness theorem to richer operator languages. The current theorem addresses operators with propositional preconditions and effects, for which reversibility and commutativity are crisp Boolean properties of effect sets. Numeric and temporal extensions disturb both properties in ways that require the definitions themselves to be reformulated. For *numeric* PDDL, an operator's effect is no longer a deletion or addition of a Boolean fluent but an assignment or increment on a numeric variable; reversibility ceases to be the existence of a syntactic inverse and becomes a question of numeric reachability—whether the inverse assignment is itself applicable given the variable's current value and any numeric preconditions—so the clean "$eff^{-}(a) \neq \varnothing \Rightarrow$ irreversible" argument of Lemma 1 no longer holds, and an operator may be reversible in part of the state space and irreversible in another. Commutativity is similarly weakened: two increment operators on the same variable commute, but an increment and a conditional assignment generally do not, so the disjoint-effect criterion of Claim (iii) must be replaced

by a numeric-independence criterion. For *temporal* PDDL, durative actions introduce overlap and mutex constraints that break commutativity even between operators with disjoint effects, because the temporal ordering itself becomes part of the state; the relevant state space is no longer the propositional reachability graph but a timed transition system, and the forward-triangle construction of Claim (iii) would need to account for the temporal separation of the two paths. In each case the methodology established here—identify the binary (or, in the richer settings, graded) semantic attributes that govern cycle formation and path convergence, derive their topological consequences, and prove completeness—transfers directly, even though the specific archetypes and discriminants will differ. Whether the propositional three-archetype partition refines into a larger but still finite set under these extensions, or whether numeric and temporal semantics admit a continuum of topologies with no finite classification, is an open question that the present framework makes precisely statable.

Finally, the network science literature on motifs, graphlets, and higher-order structural patterns [8] suggests that the three-archetype classification may be the coarsest level of a finer topological hierarchy. Within the DAG archetype, for instance, different DAG-generating domains may exhibit distinct motif profiles even when their cyclic structure is identical. The differing branching profiles documented in Section 4.3 already hint at such sub-archetype structure. A finer-grained topological analysis of constructed networks—one that goes beyond archetype membership to characterize the distribution of local subgraph patterns—could reveal a richer structure within each archetype class that the current framework does not capture.

These open questions can be organized around a single conjecture, which we now state in a stronger form than a mere possibility and support with three independent reasons for expecting it to hold. *Conjecture (finite topological classification).* For every deterministic formal language whose operational semantics generate a transition graph, there exists a finite set of binary or graded semantic attributes whose values partition the language's generable topology space into finitely many archetypes, with archetype membership decidable from the specification without enumerating any instance. The present theorem establishes this conjecture for propositional PDDL with two binary attributes and three archetypes; we contend that PDDL is not special in this respect, and that the same structure should recur across deterministic transition-generating formalisms for the following reasons.

First, the mechanisms that generate topology in our proof are not PDDL-specific. The two structural events that the classification turns on—cycle formation and path convergence—are properties of *any* deterministic transition relation, not of PDDL syntax. A cycle exists exactly when an action can be undone, and a convergent square (the seed of a forward-closed triangle) exists exactly when two actions reach a common successor in either order. These are statements about the transition relation alone; they make no reference to fluents, predicates, or any feature peculiar to PDDL. Any formalism whose semantics define a deterministic successor function therefore possesses the same two structural events, and the attributes that control them—an undo attribute and an order-independence attribute—are definable in that formalism. The worked Petri net example of Figs. 4 and 5 confirms that these attributes transfer concretely, not merely by analogy.

Second, the two attributes we identify have already been discovered independently, under different names, in several formal-methods communities—which is strong evidence that they are fundamental axes of transition-system structure rather than artifacts of our framing. Reversibility is a classical structural property of Petri nets [13]. Order-independence appears in concurrency theory as *confluence* and in the closely related notion of τ-inertness, and in model checking it is the basis of partial-order reduction, where exactly the commuting (independent) transitions are exploited to collapse equivalent interleavings. That three independent traditions—Petri nets, process calculi, and model checking—have each converged on the reversibility and commutativity/independence axes, without reference to one another or to planning, indicates that these axes

carve transition systems at their natural joints. Our contribution is to show that, at least for PDDL, these same two axes are not merely useful but *complete*: they exhaust the topological possibilities.

Third, the finiteness half of the conjecture follows from a simple combinatorial bound whenever the discriminating attributes are themselves finite in number and finite-valued. If a formalism's topology is controlled by k attributes each taking one of m values, then at most m^k archetypes can arise, and the collapses that reduce this count—as irreversibility subsumes commutativity in our Claim (i)—can only decrease it further. The substantive content of the conjecture is therefore not that the count is finite, which is automatic once the attributes are, but that a *small* set of attributes *suffices*: that the topology space does not require unboundedly many distinctions to describe. The PDDL result is the strongest possible instance of this—two attributes, three archetypes—and the structural universality of cycle formation and path convergence is our reason for expecting comparably small attribute sets elsewhere, even where the exact archetype count differs.

The conjecture could fail in two identifiable ways, and naming them sharpens it. A formalism might require unboundedly many attributes, if its topology space admits a genuinely infinite hierarchy of distinctions no finite set can capture; or its attributes might be undecidable from the specification, if determining whether two actions commute or whether an action is reversible were itself uncomputable for that formalism. Neither failure occurs for propositional PDDL, where both attributes are decidable by inspection of finite operator schemata. Identifying which richer formalisms preserve both finiteness and decidability, which require graded rather than binary attributes, and whether a single meta-theorem subsumes the per-language results, constitutes the research program of which this paper is the first instance.

6 Conclusion

Three results constitute the contribution of this paper. First, the classification theorem: for PDDL domains that are either fully reversible or fully irreversible, two binary properties of operator semantics—reversibility and commutativity—partition the space of generable state-space graphs into exactly three topological archetypes, analytically derivable from the domain specification without constructing any state space. The proof rests on a monotone fluent-deletion partial order for irreversible domains and a path-convergence argument for commutative ones, which together reduce the four logical combinations of the two attributes to three distinct structural outcomes. Second, the verification: across 289 IPC benchmark instances spanning ten domains, every archetype assignment predicted from operator algebra is confirmed by measurement, through two operator-derived signatures—the presence of directed cycles and the shape of the out-degree distribution. Scale rank fails to predict structural complexity in every cross-archetype comparison, and planner experiments reinforce this operationally: across domains of comparable state-space scale but different archetypes, optimal-search cost—measured by solve rate and node expansions in Fast Downward—differs by orders of magnitude, confirming that topological archetype rather than instance size governs search difficulty. Third, the methodological template: the procedure of identifying binary semantic attributes, deriving their topological consequences, and proving exhaustiveness is not specific to PDDL. It applies to any formal system that generates graphs—Petri nets, process calculi, temporal logic Kripke structures—and raises for each the question this paper answers for PDDL: how completely do the generative rules determine everything the system can produce? For deterministic formal languages the answer can be complete, in a sense that probabilistic generative models cannot achieve and empirical network science cannot verify. Whether other formal languages admit analogous theorems, and whether a unified meta-theory of constructed network topology is achievable, remains open. Future work includes extending the classification to mixed-reversibility domains and to numeric and temporal PDDL, and applying the same methodology to other graph-generating formalisms such as Petri nets and process calculi.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the Information, Production and Systems Research Center, Waseda University, and partly supported by the Future Robotics Organization, Waseda University; the Humanoid Robotics Institute, Waseda University, under the Humanoid Project; the JSPS KAKENHI Grant Number 25K03204; the Waseda University Grant for Special Research Projects (grant numbers 2024C-518, 2025E-027, 2026C-179 and 2026C-183); and was partly executed under the cooperation of organization between Kioxia Corporation and Waseda University.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Zhendong Du and Kenji Hashimoto; methodology, Zhendong Du; software, Zhendong Du; validation, Zhendong Du; formal analysis, Zhendong Du; investigation, Zhendong Du; data curation, Zhendong Du; writing—original draft preparation, Zhendong Du; visualization, Zhendong Du; writing—review and editing, Kenji Hashimoto; supervision, Kenji Hashimoto; project administration, Kenji Hashimoto. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Data available on request from the authors.

Ethics Approval: Not applicable.

Conflicts of Interest: As disclosed in the Funding statement, this work was partly executed under an institutional cooperation between Kioxia Corporation and Waseda University. The authors are academic researchers at Waseda University, are not employed by Kioxia Corporation, and have no direct financial interest in the company, which had no role in the study or the decision to publish. The authors declare no conflicts of interest.

References

1. Katz M, Kokel H, Sreedharan S. Seemingly simple planning problems are computationally challenging: the countdown game. arXiv:2508.02900. 2025.
2. Watts DJ, Strogatz SH. Collective dynamics of 'small-world' networks. Nature. 1998;393(6684):440–2. doi:10.1515/9781400841356.301.
3. Barabási AL, Albert R. Emergence of scaling in random networks. Science. 1999;286(5439):509–12. doi:10.1515/9781400841356.349.
4. Newman ME. The structure and function of complex networks. SIAM Rev. 2003;45(2):167–256. doi:10.1137/s003614450342480.
5. LdF C, Rodrigues FA, Travieso G, Villas Boas PR. Characterization of complex networks: a survey of measurements. Adv Phys. 2007;56(1):167–242.
6. Vasiliauskaite V, Evans TS, Expert P. Cycle analysis of directed acyclic graphs. Phys A Stat Mech Appl. 2022;596(4):127097. doi:10.1016/j.physa.2022.127097.
7. Braha D, Bar-Yam Y. Topology of large-scale engineering problem-solving networks. Phys Rev E. 2004;69(1):016113. doi:10.1103/physreve.69.016113.
8. Sizemore AE, Phillips-Cremins JE, Ghrist R, Bassett DS. The importance of the whole: topological data analysis for the network neuroscientist. Netw Neurosci. 2019;3(3):656–73. doi:10.1162/netn_a_00073.
9. Meyer U, Penschuck M. Generating synthetic graph data from random network models. In: Algorithms for big data: DFG priority program 1736. Berlin, Germany: Springer; 2023. p. 21–38.
10. Yang R, Sala F, Bogdan P. Hidden network generating rules from partially observed complex networks. Commun Phys. 2021;4(1):199. doi:10.1038/s42005-021-00701-5.
11. de Moraes RS, Nadjm-Tehrani S. NetGAP: a graph grammar approach for concept design of networked platforms with extra-functional requirements. Eng Appl Artif Intell. 2024;133:108089.
12. Ciardo G. An advanced hierarchical hybrid environment for reliability and performance modeling [Technical Report]. Williamsburg, VA, USA: College of William and Mary; 2003.
13. Murata T. Petri nets: properties, analysis and applications. Proc IEEE. 1989;77(4):541–80.

14. Blum AL, Furst ML. Fast planning through planning graph analysis. Artif Intell. 1997;90(1–2):281–300. doi:10.1016/s0004-3702(96)00047-1.
15. Helmert M. The fast downward planning system. J Artif Intell Res. 2006;26:191–246.
16. Bäckström C, Nebel B. Complexity results for SAS+ planning. Comput Intell. 1995;11(4):625–55. doi:10.1111/j.1467-8640.1995.tb00052.x.
17. Haslum P, Botea A, Helmert M, Bonet B, Koenig S. Domain-independent construction of pattern database heuristics for cost-optimal planning. In: AAAI-07/IAAI-07 Proceedings: 22nd AAAI Conference on Artificial Intelligence and the 19th Innovative Applications of Artificial Intelligence Conference; 2007 Jul 22–26; Vancouver, BC, Canada. p. 1007–12.
18. Hoffmann J, Nebel B. The FF planning system: fast plan generation through heuristic search. J Artif Intell Res. 2001;14:253–302.
19. Gnad D, Hoffmann J. Star-topology decoupled state space search. Artif Intell. 2018;257:24–60. doi:10.1016/j.artint.2017.12.004.
20. Vallati M, Chrpa L, Grześ M, McCluskey TL, Roberts M, Sanner S, et al. The 2014 international planning competition: progress and trends. AI Mag. 2015;36(3):90–8. doi:10.1609/aimag.v36i3.2571.
21. Xie J, Zhang K, Chen J, Zhu T, Lou R, Tian Y, et al. Travelplanner: a benchmark for real-world planning with language agents. arXiv:2402.01622. 2024.
22. Valmeekam K, Marquez M, Olmo A, Sreedharan S, Kambhampati S. Planbench: an extensible benchmark for evaluating large language models on planning and reasoning about change. Adv Neural Inf Process Syst. 2023;36:38975–87. doi:10.52202/075280-1693.
23. Kambhampati S, Valmeekam K, Guan L, Verma M, Stechly K, Bhambri S, et al. LLMs can't plan, but can help planning in LLM-modulo frameworks. arXiv:2402.01817. 2024.
24. Du Z, Wang H, Hashimoto K. Lexical-prior-free planning: a symbol-agnostic pipeline that enables LLMs and LRMs to plan under obfuscated interfaces. Comput Mater Contin. 2026;87(1):12.
25. Zhang X, Sheng VS. Bridging the gap: representation spaces in neuro-symbolic AI. arXiv:2411.04393. 2024.
26. Mohan RP. Neurosymbolic AI: bridging neural networks and symbolic reasoning. World. 2025;25(1):2351–73.
27. Liang B, Wang Y, Tong C. AI reasoning in deep learning era: from symbolic AI to neural-symbolic AI. Mathematics. 2025;13(11):1707. doi:10.3390/math13111707.
28. Bartesaghi P, Clemente GP, Grassi R. Clustering coefficients as measures of the complex interactions in a directed weighted multilayer network. Phys A Stat Mech Appl. 2023;610(28384):128413. doi:10.1016/j.physa.2022.128413.
29. Corrêa AB, Pereira AG, Seipp J. The 2025 planning performance of frontier large language models. arXiv:2511.09378. 2025.