Learning to Reason: Intuition Formation in Language Models

Tian Ye

July 2025
CMU-ML-25-111

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee

Bhiksha Raj, Chair	*Carnegie Mellon University*
Sean Welleck	*Carnegie Mellon University*
Yuanzhi Li	*Meta*
Zeyuan Allen-Zhu	*Meta*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: machine learning, language modeling, feature learning

To all of the people that light up my life. ♡

Abstract

Large language models (LLMs) can now attain near-perfect accuracy on grade-school math benchmarks, yet their ability to perform genuine, structure-guided reasoning remains unclear. We study this question in a controlled setting by introducing PureIntersectionPoint, a corpus of 8 million pure intersection-point geometry theorems equipped with a graph language, four core tactics (`merge-by-cr`, `merge-by-def`, `cr-equal` and `cr-equiv`), and machine-checkable proofs. Each problem is a directed proof search over a state graph, making it possible to ask: *can an LLM learn to form and use "intuition"—i.e., subgraph-level heuristics that guide tactic choice?*

Training GPT-2-style models on PureIntersectionPoint reveals six core findings: (1) fully random index orders cripple learning; (2) presenting theorems in construction order makes them much easier; (3) an "active exploration" proof format boosts performance further; (4) clear signs of intuition emerge over training, as models shift from trivial to structure-driven subgoals; (5) models generalise beyond rote recall, producing unseen subgoals at test time; and (6) they even solve theorems our brute-force searcher cannot.

Together, these results show that data ordering and proof format act like curricula, that intuition is measurable and learnable, and that LLMs can discover proofs beyond exhaustive search.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Recent advances in *generative artificial intelligence* (AI), and in *large language models* (LLMs) in particular, have convincingly demonstrated that these systems can

1. produce fluent, high-quality natural language,
2. comprehend and generate source code, and
3. tackle a broad range of complex tasks.

Despite these successes, the extent to which LLMs possess genuine human-level reasoning and planning abilities remains an open question.[1] In this thesis, we investigate these capabilities through the lens of *mathematical problem solving*.

Over the past few years, increasingly *sophisticated* large-language models—e.g., GPT-4, Gemini-Ultra, Claude 3 Opus, and DeepSeek-R1—have achieved **near-perfect** accuracy on grade-school mathematics benchmarks such as GSM8K, and have dramatically narrowed the gap to human-expert performance on more demanding collections like the MATH dataset [Ope24; Gem25; Ant24; Guo+25; Cob+21; Hen+21].

Most strikingly, in July 2025 both Google DeepMind and OpenAI reported *gold-medal-level* performance on the International Mathematical Olympiad (IMO): DeepMind's Gemini Deep Think was *officially graded* by the IMO as solving five of the six problems within the 4.5-hour limit (35/42 points), while OpenAI announced an experimental model that achieved the same score under independent grading.[2] While top human contestants still outscored these systems, this milestone significantly updates the state of play and motivates a sharper inquiry:

Which hard mathematical problems can LLMs solve—and why?

Recent analyses of GSM-level tasks suggest that LLMs often succeed by performing *structured reasoning*: constructing internal dependency graphs, retrieving relevant facts,

[1] See, for example, [Bro+20; Cho+22; Ope+24] for empirical evidence and [Bub+23] for a critical discussion.

[2] DeepMind's official announcement (with confirmation from the IMO president): DeepMind Blog, July 21, 2025; see also *Nature* news report, July 2025 and *Reuters*, July 21, 2025. For OpenAI's result as publicly reported and analyzed, see Simon Willison's summary, July 19, 2025 and *Business Insider*, July 2025.

and executing back-tracking search over intermediate steps [Ye+24a; Ye+24b]. To understand why IMO-level challenges remain elusive, consider the following illustrative *warm-up* problem:

> **Problem.** Let $G = (V, E)$ be a *tournament*[3] on $n = |V| \geq 1$ vertices. Assume that every vertex has at least one outgoing edge, i.e.
>
> $$\forall\, u \in V\ \exists\, v \in V \setminus \{u\}\ :\ (u, v) \in E.$$
>
> Prove that there exist three distinct vertices $a, b, c \in V$ that form a directed triangle:
>
> $$(a, b),\ (b, c),\ (c, a)\ \in\ E.$$

Unlike grade-school arithmetic exercises, this combinatorial challenge does *not* yield to the dependency-graph heuristics or simple back-tracking search that have proved so effective on GSM-level tasks. Instead, a key *structural* insight is required.

> **Proof.** Choose a vertex $a \in V$ with *maximum in-degree*. Because every vertex has at least one successor, there exists some $b \in V \setminus \{a\}$ with $(a, b) \in E$. Let
>
> $$P\ :=\ \{\, x \in V : (x, a) \in E \,\} \quad \text{(the predecessors of } a\text{)}.$$
>
> By maximality of a's in-degree, we have $P \neq \varnothing$.
>
> 1. **Case 1:** $\exists\, c \in P$ such that $(b, c) \in E$. Then the edges $(a, b), (b, c), (c, a)$ form the desired directed triangle.
> 2. **Case 2:** $\forall\, x \in P$ we have $(x, b) \in E$. In this case every predecessor of a also points to b, so
>
> $$\deg^-(b)\ \geq\ \deg^-(a) + 1,$$
>
>   contradicting the choice of a as a vertex of *maximum* in-degree.
>
> Both cases cannot be avoided, so a directed triangle must exist in G. □

The preceding argument hinges on *choosing* the vertex a of *maximum in-degree*. A naive brute-force search over all vertices—and, worse, over all candidate sub-structures—quickly becomes intractable. What the human mathematician supplies almost instantly is a piece of *intuition*: a partially formed mental model that singles out a promising construction and filters away a vast space of irrelevant but logically admissible paths. Without that flash of insight, an LLM typically produces a profusion of correct *yet inconsequential* intermediate deductions, only to stall at a dead end.

Pushing this line of thought one step further, we ask:

Can large-language models *learn* to construct the right intuition ?

[3] A tournament is a directed graph in which, for every pair of distinct vertices $u, v \in V$, exactly one of (u, v) or (v, u) belongs to E.

1.1 Contributions

To probe this question empirically, we introduce the PureIntersectionPoint dataset: a corpus of eight million challenging Euclidean-geometry problems, each paired with a machine-checkable proof. Using this benchmark, we look at how different training and prompting choices affect an LLM's ability to spot the key ideas that lead to a correct solution. Our contributions are:

1. **PureIntersectionPoint: a controlled reasoning testbed.** We design a minimal graph language with four core tactics and release 8M theorem-proof pairs without isomorphic duplicates.
2. **Ordering as curriculum.** We show that random index/theorem order weakens learning, whereas construction order dramatically boosts accuracy.
3. **Active exploration format.** A "goal-after-evidence" proof format (version 2) further improves performance, letting models explore before committing to a subgoal.
4. **Tracking intuition as it forms.** By collapsing each first-line subgoal to a *maximally reduced* form, we observe the model move from rarely proposing any meaningful intermediate goal to consistently selecting correct, non-trivial ones. This shift provides a simple, intrinsic signal that the model is learning the structural features needed to guide its own reasoning.
5. **Generalisation beyond rote.** Under a subgoal-based split, models still produce unseen subgoals with high probability and maintain strong accuracy, indicating abstraction rather than memorisation.
6. **Beyond brute-force search.** The model solves a subset of theorems that a symbolic searcher cannot, demonstrating reasoning strategies that transcend exhaustive search.

Chapter 2

Preliminaries

To turn the idea of "building intuition" into something we can measure, we adopt a lightweight graph language inspired by LEAN. A *problem state* is a labelled directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$:

- Each vertex $v \in \mathcal{V}$ stands for an *object* (e.g. an integer or a set).
- Each edge $(u, v) \in \mathcal{E}$ records a *relation* between two objects (e.g. "$u \in v$" or "$\gcd(u, v) = 1$").
- The object type and the edge relation type are stored as labels on the vertices and edges, respectively.

2.1 Tactics as Graph Rewrites

Similar to LEAN, we prove the problem by *tactics*. A tactic is itself a small labelled graph $\mathcal{T} = (\mathcal{V}_T, \mathcal{E}_T)$ together with a list of *rewrite actions* (adding new vertices or edges). To *apply* a tactic we

1. find a sub-graph $\mathcal{G}' \subseteq \mathcal{G}$ that matches $\mathcal{T}$; and
2. perform the associated actions, producing an updated problem state $\mathcal{G}''$.

The entire proof is a sequence of such rewrites. A problem is *solved* once the target configuration (for example, a particular edge or sub-graph) appears in the current state.

2.2 Running Example

Recall the warm-up problem from Chapter 1:

> **Problem.** Let $G = (V, E)$ be a *tournament*[1] on $n = |V| \geq 1$ vertices. Assume each vertex has at least one outgoing edge,
>
> $$\forall\, u \in V\ \exists\, v \in V \setminus \{u\}\ :\ (u, v) \in E,$$

[1] A tournament is a directed graph where, for every pair of distinct vertices $u, v \in V$, exactly one of (u, v) or (v, u) is in E.

and prove that there are three distinct vertices $a, b, c \in V$ forming a directed triangle:

$$(a, b),\ (b, c),\ (c, a) \in E.$$

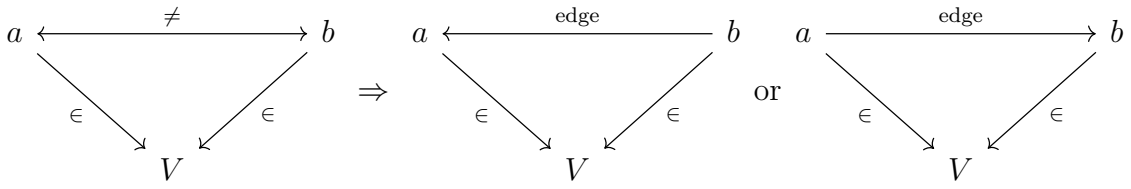
We will translate the natural-language proof into a sequence of tactics.

Background graph and basic tactics

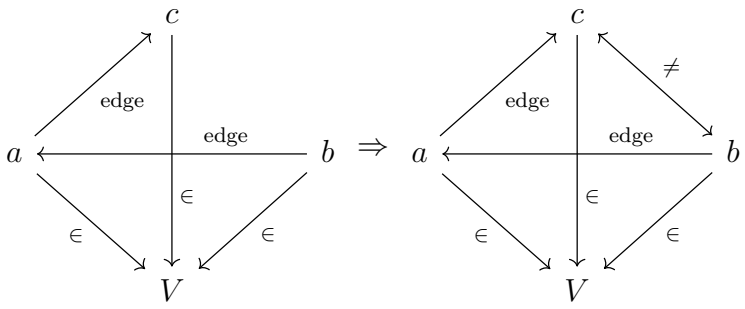
Background graph $\mathcal{G}$. Since G is a graph with at least one vertex, we can start with the following background graph $\mathcal{G}$:

$$v \xrightarrow{\ \in\ } V \tag{2.1}$$

Tournament property. Because G is a tournament, every unordered pair $\{a, b\} \subseteq V$ is joined by exactly one directed edge. The corresponding tactics are



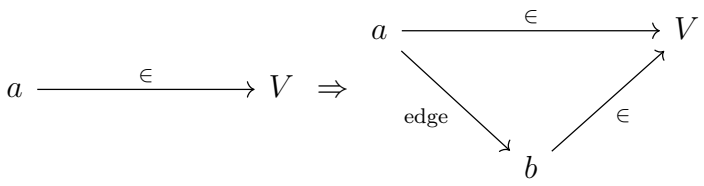
(2.2)



(2.3)

where the edge label edge represents the relationship between vertex a and its successor b.

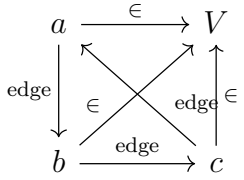
Out-degree ≥ 1. Each vertex has at least one outgoing edge.



(2.4)

Target configuration

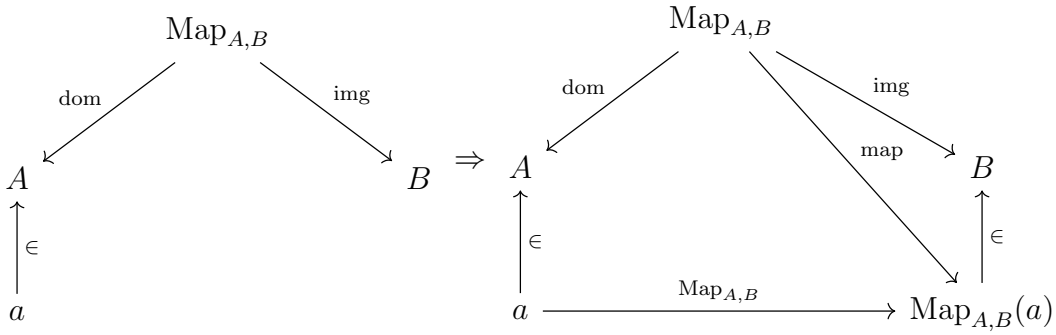
We seek a directed triangle in G, which is equivalent to finding the following subgraph in $\mathcal{G}$:



(2.5)

Basic tactics in set theory

We also rely on general "ground-truth" tactics. For instance, for a map $\mathrm{Map}_{A,B} : A \to B$, we have:

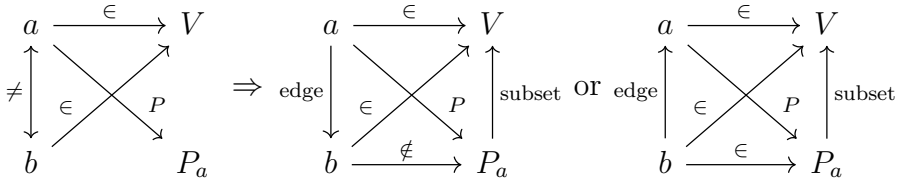


(2.6)

Proof in tactics

Now we can combine the tactics to prove the problem.

First we need to define the *predecessor set* of a vertex a in the tournament graph G. For any $a \in V$ let $P_a := \{\, x \in V : (x, a) \in E\}$[2]. The defining property of P_a can be encoded as the tactic



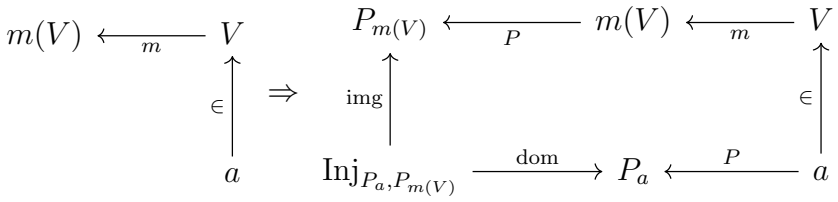
(2.7)

Here the label P records the link between a and its predecessor set P_a.

Because V is finite and non-empty, there exists at least one vertex with maximal in-degree; denote such a vertex by $m(V)$[3]. The maximality condition is captured by the tactic

[2]The existence of P_a required by (2.7) follows directly from the tournament tactic in (2.2). That tactic produces a Boolean-valued verifier on V, and the truth set of this verifier is P_a. We suppress the underlying set-theoretic details, but every step ultimately reduces to the basic tactics available in ZF set theory.

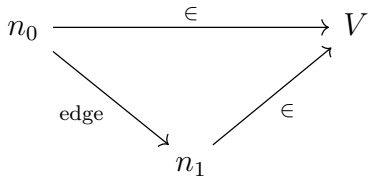
[3]A vertex of maximum in-degree exists since V is finite. A standard induction or well-ordering argument yields $m(V)$. The routine details are omitted.



(2.8)

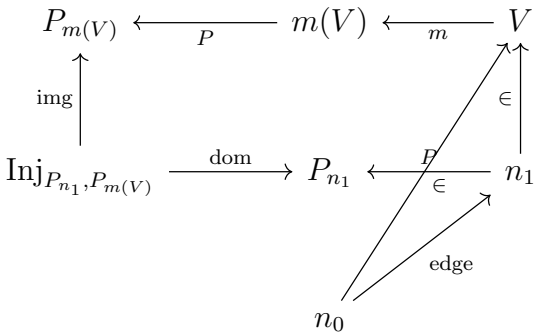
The object $\mathrm{Inj}_{P_a,P_{m(V)}}$ represents an injective map from P_a into $P_{m(V)}$, reflecting the fact that $m(V)$ has in-degree at least as large as that of a.

It is a good practice to prove that $P_{m(V)}$ is non-empty. In fact, by applying tactic (2.4) to the background graph (2.1), we obtained the updated graph:



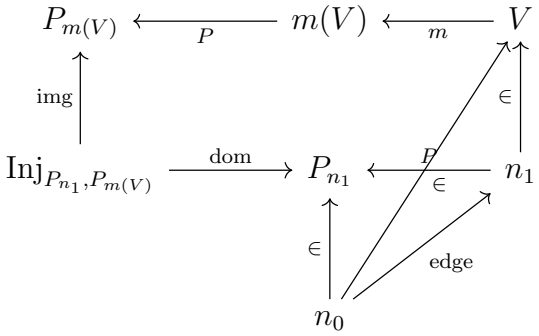
(2.9)

By the existence of $m(V)$ and applying tactic (2.8) to the subgraph $n_1 \xrightarrow{\in} V$, we can futher update the graph to



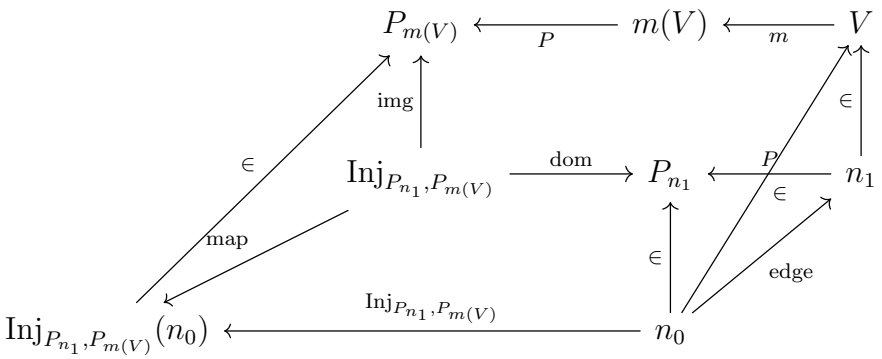
(2.10)

Further applying the tactic (2.7) yields



(2.11)

Finally, because $\mathrm{Inj}_{P_{n_1},P_{m(V)}}$ is also a map, we can apply the tactic (2.6) to obtain



(2.12)

This shows that $P_{m(V)}$ is non-empty, as required.

All remaining steps follow exactly the sequence of deductions in the informal argument: we apply each tactic in the same order, thereby transforming the background graph until the triangle (2.5) appears. The full, step-by-step derivation is recorded in Appendix A.1.

2.3 What We Mean by "Intuition"

In our framework, a proof is a *deterministic* sequence of tactic applications, each obtained by matching a pattern graph against the current state. As the state grows, the number of possible matches explodes, and the *order* in which tactics are applied becomes decisive. Humans cope with this combinatorial blow-up by relying on something we loosely call *intuition*. At a minimum, such intuition draws on two complementary sources:

1. **Prior experience.** Familiarity with similar problems guides our attention toward useful sub-structures. In the warm-up example, anyone who has met the notion of a *tournament* without directed 3-cycles will instantly recall that the resulting graph must be totally ordered, so the presence of a sink vertex contradicts the "out-degree ≥ 1" premise.

2. **Active exploration.** When experience runs out, we try multiple avenues and monitor "signals" that hint at progress. One may begin by

   (a) analysing small values of $|V|$, or

   (b) fixing an arbitrary edge $a \to b$ and following its consequences.

   Both lines of attack quickly reveal an ever-increasing in-degree sequence, and—if one knows the method of *infinite descent*—a contradiction is at hand, leading naturally to the proof sketched in Chapter 1.

In both settings above, an *intuition* is any *sub-graph of the current state that guides the next move*. It need not be a fully fledged lemma; rather, it can be *any* structural pattern whose appearance raises the likelihood of a successful proof by a non-trivial amount. Throughout this thesis we therefore use *intuition* to denote *any heuristic—learned or discovered—that nudges an LLM toward experience-based shortcuts or otherwise promising*

lines of attack. The next chapters refine this idea and show how it can be quantified on the PureIntersectionPoint dataset.

Chapter 3

The PureIntersectionPoint Dataset

Following the evaluation protocol of [Ye+24a; Ye+24b], we design a corpus that isolates an LLM's ability to acquire and exploit *intuition*. Two principles drive the construction:

1. **Minimal tactic set.** The language is limited to *four* primitive tactics, ensuring that success hinges on strategic planning rather than memorising a large rule book.
2. **High combinatorial difficulty.** Each instance is deliberately chaotic; brute-force search and shallow pattern matching break down long before a solution emerges.

3.1 Basic concepts

Problem family. Every item in the PureIntersectionPoint dataset is a *pure intersection-point theorem*. Starting from a small collection of free points or lines, we repeatedly

(a) connect two points to create a new line, or

(b) intersect two lines to create a new point.

The goal is always the same: prove that either *three points are collinear* or *three lines are concurrent*.

A classic example is **Pappus's theorem**:

> Given two distinct lines ℓ_1 and ℓ_2 with points $A, B, C \in \ell_1$ and $D, E, F \in \ell_2$, prove that the three intersection points $X := AE \cap BD$, $Y := BF \cap CE$, and $Z := CD \cap AF$ are collinear.

The shortest proof found online multiplies three carefully chosen applications of Menelaus's theorem; crucially, *which* triangle and *where* to apply the theorem matter. Despite the tiny tactic vocabulary, problems of this kind defeat naive enumeration and rote memorisation, making the PureIntersectionPoint dataset a clean test bed for investigating how large language models acquire geometric intuition.

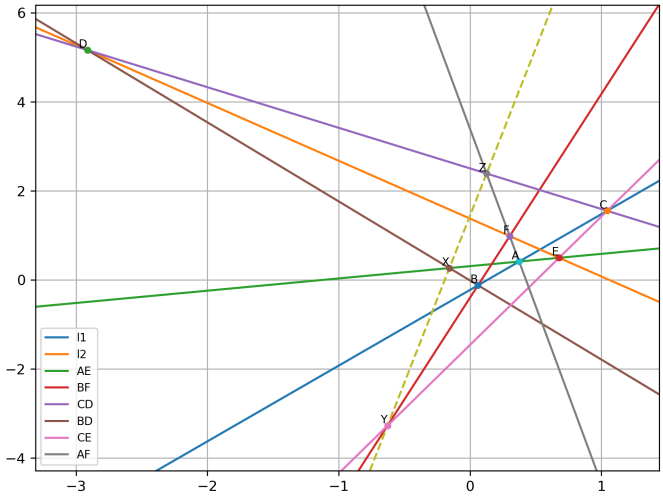


Figure 3.1: Illustration of Pappus's theorem: the points $X := AE \cap BD$, $Y := BF \cap CE$, and $Z := CD \cap AF$ lie on one line.

Proof technique.

All proofs in the PureIntersectionPoint dataset rely on a single numerical invariant, the *cross-ratio*. Given four distinct, collinear points or four distinct, concurrent lines, their cross-ratio is a real number that stays unchanged under projective transformations. In our formal language we do *not* manipulate this number directly; instead, we treat a cross-ratio as an *equivalence class of ordered quadruples*. To keep terminology clear we write

- *cross-ratio tuple* for the ordered quadruple (A, B, C, D) of points (or lines) on which the invariant is taken, and
- *cross-ratio value* for the equivalence class of all tuples that have the same numerical cross-ratio.

A full definition appears in Appendix A.2.1; here we state only the two properties that power every tactic in PureIntersectionPoint:

1. **Point-line duality.** If (A, B, C, D) is a cross-ratio tuple of collinear points and $(\ell_1, \ell_2, \ell_3, \ell_4)$ is a tuple of concurrent lines with $A \in \ell_1$, $B \in \ell_2$, $C \in \ell_3$, and $D \in \ell_4$, then the two tuples share the same cross-ratio value.
2. **Uniqueness of the fourth element.** Suppose (A, B, C, D) and (A, B, C, D') have identical cross-ratio values. Then the last elements must coincide: $D = D'$.

Illustration: Pappus's theorem. Below is a terse cross-ratio proof of the classical Pappus configuration; the detailed geometric set-up is shown in Figure 3.1. Using only

property 1, cross-ratios transform as follows:

$$\begin{aligned}\mathrm{cr}(ZF, ZC, ZE, ZY) &= \mathrm{cr}(ZF \cap CE,\ C,\ E,\ Y)\\ &= \mathrm{cr}(FA, FC, FE, FY)\\ &= \mathrm{cr}(A,\ C,\ AC \cap DF,\ B)\\ &= \mathrm{cr}(DA, DC, DF, DB)\\ &= \mathrm{cr}(A,\ DC \cap AE,\ E,\ X)\\ &= \mathrm{cr}(ZA, ZD, ZE, ZX).\end{aligned} \tag{3.1}$$

Because $ZF = ZA$ and $ZC = ZD$ by collinearity, and $ZE = ZE$ trivially, equality of the two cross-ratios forces $ZY = ZX$ by property 2. Hence the three points X, Y, and Z are collinear, which completes the proof.

Vertices and edges.

For every PureIntersectionPoint instance we maintain a *state graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with

$$\mathcal{V} \;=\; \mathcal{P} \;\cup\; \mathcal{L} \;\cup\; \mathcal{CP} \;\cup\; \mathcal{CL}, \tag{3.2}$$

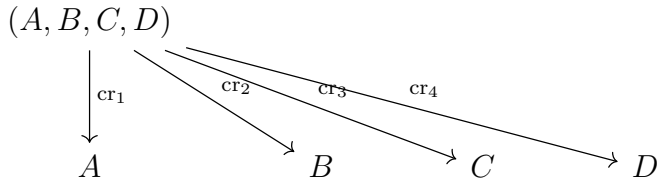
where $\mathcal{P}$ is the set of points created so far, $\mathcal{L}$ is the set of lines, $\mathcal{CP}$ is the set of cross-ratio tuples of points, and $\mathcal{CL}$ is the set of cross-ratio tuples of lines.

Edges are

Between points and lines. When a point A lies on a line ℓ, we add the edge

$$A \xleftrightarrow{\text{coincide}} \ell \tag{3.3}$$

Between cross-ratio tuples and points/lines. When a cross-ratio tuple (A, B, C, D) is formed, we add the edge



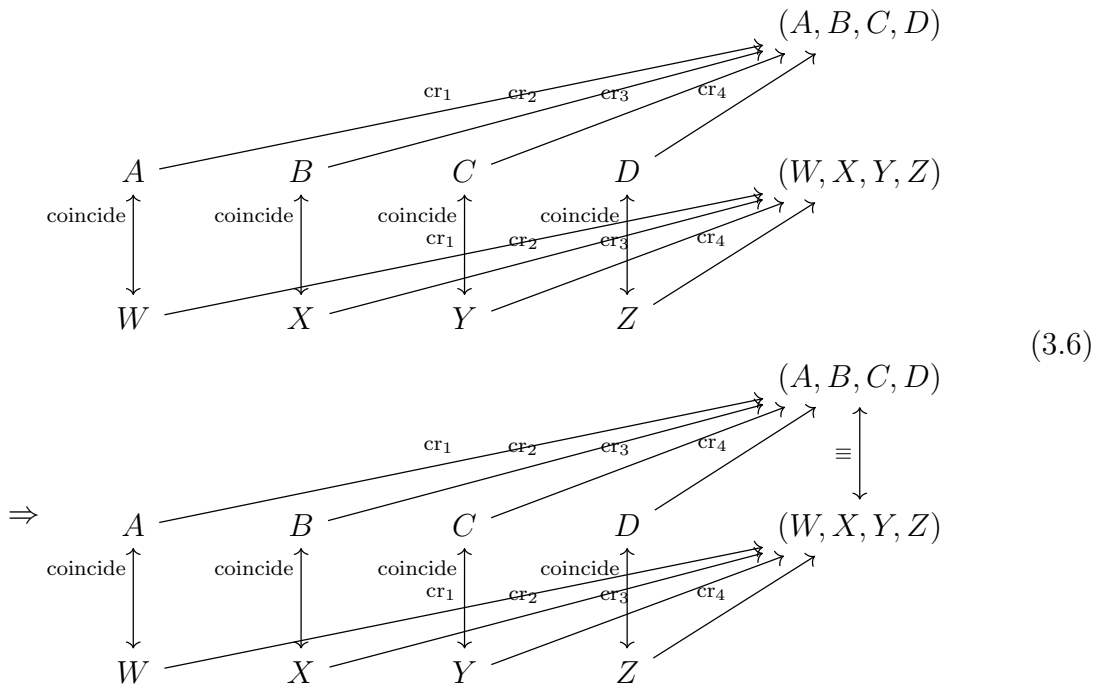
(3.4)

Between cross-ratio tuples. When two cross-ratio tuples share the same value, we add the edge

$$(A, B, C, D) \xleftrightarrow{\equiv} (W, X, Y, Z) \tag{3.5}$$

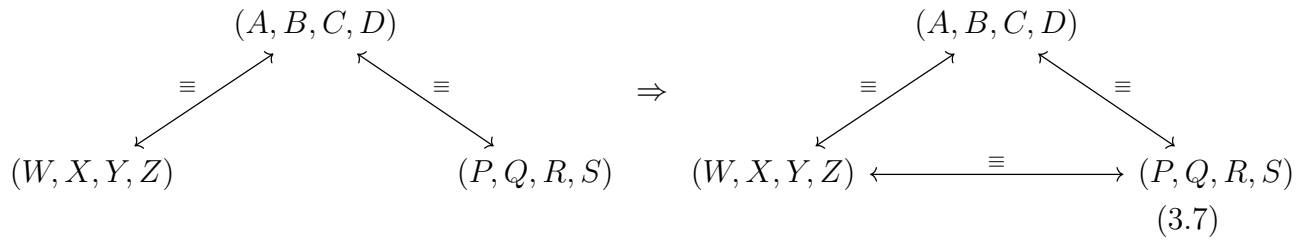
Tactics.

Equality of cross-ratio values (`cr-equal`). The duality of the cross ratio is captured by the tactic



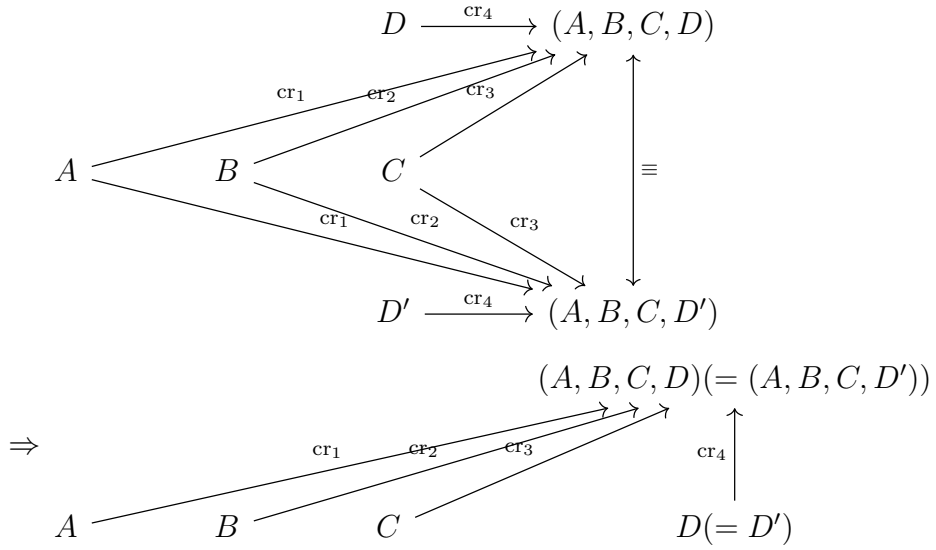
(3.6)

Transitivity of equal cross-ratios (`cr-equiv`). If two tuples each equal a third one, they equal each other:



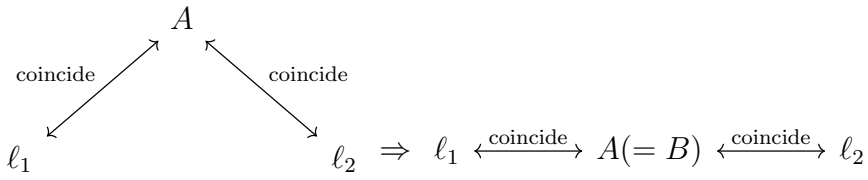
(3.7)

Merge by Cross-ratio (`merge-by-cr`). The uniqueness of the fourth element in a cross-ratio tuple can be expressed as a tactic



(3.8)

Merge by Definition (`merge-by-def`). If two distinct lines ℓ_1 and ℓ_2 meet a point twice, the points coincide:



(3.9)

B

Remark. Whenever we create a cross-ratio tuple (cf. (3.4)) or apply the merge-by-definition tactic (3.9), we must first *numerically* verify that the points or lines involved are distinct. Because these checks are performed outside the symbolic graph language, a derivation that passes the graph-level rules may still be invalid. This deliberate imperfection makes PureIntersectionPoint harder: a model that merely "knows the tactics" can still go wrong. To succeed, an LLM must develop enough *intuition* to avoid such pitfalls.

3.2 Dataset construction

Definition 1 (PureIntersectionPoint theorem statement)**.** *A PureIntersectionPoint theorem statement is a finite ordered list of* constructions *together with a single* goal*. Each construction is exactly one of:*

1. ***Create a free object:*** *introduce a new free point or a new free line.*

2. ***Incidence choice:*** *create a new free point on an existing line, or draw a new free line through an existing point.*
3. ***Derived object:***
   (a) *intersect two distinct lines to create a new point; or*
   (b) *connect two distinct points to create a new line (by point-line duality we also say the points "intersect").*

The goal always has the form "point P lies on line L" or equivalently "line L passes through point P" (i.e. $P \in L$).

Example (Pappus). Below is one admissible PureIntersectionPoint encoding of Pappus's theorem. Object names (shown here as $p*$ for points and $l*$ for lines) are arbitrary strings; renaming does not change the theorem. By point-line duality we reuse the verb "intersect" both for two lines meeting in a point and two points determining a line. The only forbidden operation is "intersect a point with a line"—ruling this out ensures the theorem graph (Definition 3) remains bipartite.

Sample PureIntersectionPoint Statement: Pappus

Create $l0$
Create $l1$
From $l0$ draw $p2$
From $l0$ draw $p3$
From $l0$ draw $p4$
From $l1$ draw $p5$
From $l1$ draw $p6$
From $l1$ draw $p7$
$p2$ and $p6$ intersect at $l8$
$p3$ and $p7$ intersect at $l9$
$p4$ and $p5$ intersect at $l10$
$p3$ and $p5$ intersect at $l11$
$p4$ and $p6$ intersect at $l12$
$p2$ and $p7$ intersect at $l13$
$l8$ and $l11$ intersect at $p14$
$l9$ and $l12$ intersect at $p15$
$l10$ and $l13$ intersect at $p16$
$p14$ and $p16$ intersect at $l17$
Goal: $p15$ is on $l17$

Definition 2 (Valid PureIntersectionPoint theorem (informal))**.** *A PureIntersectionPoint theorem statement is* valid *if, for* almost all *geometric realisations of its free points and lines, the goal relation (point P lies on line L) holds after carrying out the stated constructions.*[1]

[1]See Definition 12 for the formal definition of "almost all".

Definition 3 (Theorem graph)**.** *A* theorem graph *is a connected undirected bipartite graph. Given a pure intersection-point theorem, its theorem graph is $G = (V, E)$, where V is the set of all points and lines appearing in the statement and E is the set of point-line incidence edges defined in (3.3).*

The goal edge itself is *not* added unless the constructions already included it; in that trivial case the theorem is considered proved and is excluded from this thesis.

The bipartition of G naturally labels vertices as "points" or "lines"[2]. Thus a theorem graph records only coincidence relations, independent of the order or manner in which the objects were created.

Definition 4 (Equivalent theorems)**.** *Let each pure intersection-point theorem T be given by its theorem graph $G = (V, E)$ together with a designated* goal pair *(u, v): this encodes either "point u lies on line v" (if u is a point, v a line) or, by duality, "point v lies on line u" (if v is a point, u a line). The goal edge itself is not in E. Two theorems*

$$T_1 = (G_1 = (V_1, E_1), (u_1, v_1)) \qquad \text{and} \qquad T_2 = (G_2 = (V_2, E_2), (u_2, v_2))$$

are equivalent*, written $T_1 \cong T_2$, if there exists a bijection $\phi : V_1 \to V_2$ such that:*

1. *$\phi(\{u_1, v_1\}) = \{u_2, v_2\}$ (the unordered goal pair is preserved; ϕ may swap the two elements), and*
2. *for every $u, v \in V_1$,*
$$(u, v) \in E_1 \iff \big(\phi(u), \phi(v)\big) \in E_2.$$

Points need not map to points under ϕ; consequently, every theorem is equivalent to its *dual* obtained by swapping points and lines.

Remark. Equivalence does *not* preserve validity in general (Appendix A.2.2). However, if the theorem graph is non-degenerate (Definition 5), then either every statement with that graph is valid or none is.

Definition 5 (Non-degenerate theorem graph)**.** *A theorem graph $G = (V, E)$ is* non-degenerate *if it admits at least one geometric realisation in which any two distinct vertices in the same part correspond to distinct points (resp. distinct lines).*

Throughout this thesis we consider only non-degenerate theorem graphs.

Definition 6 (Constructive theorem graph)**.** *A theorem graph is* constructive *if it can be reduced to the empty graph by repeatedly deleting any vertex whose degree is < 3 together with all of its incident edges.*

An ordering $(v_1, v_2, \ldots, v_n)$ of V is called a constructive order *if deleting the vertices in the* reverse *order $v_n, v_{n-1}, \ldots, v_1$ is a valid deletion sequence; that is, for each $k \in [n]$ the vertex v_k has degree < 3 in the (sub)graph induced by $\{v_1, \ldots, v_k\}$.*

At the first glance, this definition seems to depend on the order in which vertices are deleted. However, we can show that the notion of a constructive theorem graph is well defined, and that it does not depend on the order in which vertices are deleted.

Theorem 1. *The notion of a constructive theorem graph is well defined; that is, it does not depend on the order in which vertices are deleted.*

[2]Which side is taken as points is irrelevant up to duality.

Proof of Theorem 1. Assume, for contradiction, that a constructive theorem graph $G = (V, E)$ admits two deletion sequences whose results differ. Let $c_1, \ldots, c_n$ delete G completely, and let $d_1, \ldots, d_{n'}$ stop at a non-empty sub-graph G' in which every vertex has degree ≥ 3. Let i be the smallest index such that $c_i \notin \{d_1, \ldots, d_{n'}\}$. Because $c_1, \ldots, c_{i-1}$ have already been removed, c_i has degree < 3 in G', contradicting the definition of G'. □

Theorem 2. *(i) Every pure intersection-point theorem yields a constructive theorem graph.*

*(ii) (*Order-realisation*) Let $G = (V, E)$ be a constructive theorem graph and let $\sigma = (v_1, v_2, \ldots, v_n)$ be any* constructive order *of G. Then there exists a PureIntersectionPoint theorem statement whose objects are introduced* exactly *in the order σ and whose theorem graph is G (up to isomorphism preserving vertex types).*

Proof of Theorem 2. **(i)** Given a valid PureIntersectionPoint statement list the introduced objects in creation order $v_1, \ldots, v_n$. Each object is formed using at most two earlier objects (free: 0; incidence choice: 1; derived: 2). Hence, when deleting in reverse, v_k has degree < 3 in the then-current subgraph. Iterating deletes all vertices, so the resulting theorem graph is constructive.

(ii) Fix a constructive order $\sigma = (v_1, \ldots, v_n)$ of a constructive graph G. We *synthesize* a theorem statement whose k-th construction introduces v_k.

Because G is bipartite we fix once and for all its partition into *points* and *lines*.

Inductively for $k = 1$ to n let G_k be the subgraph induced by $\{v_1, \ldots, v_k\}$. Because σ is constructive, v_k has degree < 3 in G_k. Introduce v_k according to this degree:

- **Degree** 0**:** Declare v_k a new free object of its predetermined type (point or line).
- **Degree** 1**:** Its unique neighbour w is already introduced. If w is a line, create a point on w; if w is a point, create a line through w.
- **Degree** 2**:** Its two neighbours are of the same type (bipartite). If v_k is a point, introduce it as the intersection of those two lines; if v_k is a line, introduce it as the line through those two points.

This realises G (up to renaming) because every edge incident to v_k in G_k is created exactly when v_k is introduced; no extra edges are added. Thus the statement's construction order is precisely σ.

Therefore *every constructive order* of G is realisable as a valid PureIntersectionPoint construction sequence, establishing (ii). □

Corollary 1. *Up to the equivalence in Definition 4, PureIntersectionPoint theorem statements are in bijection with constructive theorem graphs together with their goal pairs.*

Some constraints in G can be removed without affecting the theorem's validity, yet the resulting theorem is no longer equivalent to the original. To keep the dataset compact, we retain only the maximally constrained form of each theorem.

Definition 7 (Theorem reduction)**.** *Let a PureIntersectionPoint theorem statement be $T = \big(G = (V, E), (u, v)\big)$, where G is non-degenerate and (u, v) is the (unordered) goal pair. Form the augmented graph $\widehat{G}$ by adding the temporary goal edge between u and v.*

Repeatedly delete any vertex w of degree < 3 in the current *graph together with its incident edges until no such vertex remains.*[3] *By the argument used in the proof of Theorem 2, the resulting graph does not depend on the order of deletions. Denote the final graph by $\widehat{G}_{\text{core}}$. Remove the temporary goal edge (if still present) to obtain G', and define the* reduction *of T to be $T' = \big(G', (u, v)\big)$.*

Theorem 3. *If a PureIntersectionPoint theorem statement T is valid, then its reduction T' is also valid.*

Proof. Let $T = \big(G = (V, E), (u, v)\big)$ and $T' = \big(G' = (V', E'), (u, v)\big)$ be as in the reduction definition, with $V' = V \setminus \{v_{n'+1}, \ldots, v_n\}$ the vertices that survive the deletion process. Write

$$\sigma \;=\; (v_1, \ldots, v_{n'}) \quad \text{and} \quad (v_{n'+1}, \ldots, v_n)$$

for the construction order of G' and the list of deleted vertices, respectively.

Step 1: extending a realisation. Take an arbitrary realisation $\mathcal{R}'$ of T'. Re-insert the deleted vertices *in reverse deletion order* $v_n, v_{n-1}, \ldots, v_{n'+1}$, choosing their geometric positions as follows:

- Whenever v_k had degree 0 at deletion, place it as a new free point (or line, according to its type) in general position.
- If v_k had degree 1, attach it generically to its unique neighbour (point on a given line or line through a given point).
- If v_k had degree 2, it is a point incident with two distinct lines or vice-versa.

Because every v_k had degree < 3 at the moment of deletion, one of the three cases always applies. The result is a realisation $\mathcal{R}$ of the *full* statement T that extends $\mathcal{R}'$.

Step 2: invoking validity of T. Since T is valid, the goal relation $P \in L$ (with the correct interpretation of the unordered pair (u, v)) holds in *almost all* realisations ($\mathcal{R}$) of T. In particular, it holds in *almost all* realisations $\mathcal{R}'$ as step 1 only adds new objects without changing the existing ones[4].

Step 3: conclusion. We have shown that *almost all* realisations of T' satisfies the goal. Therefore T' is valid. □

Base-theorem generation

A custom Python script performs a random search that starts with a small set of free points and lines and iteratively

1. intersects two lines to create a new point, or
2. joins two points to create a new line.

[3]The goal vertices u and v *may* be deleted during this process. If either is removed, the reduced object no longer represents a valid PureIntersectionPoint theorem with the original goal pair.

[4]If one of u or v were deleted in the reduction process, then $\widehat{G}$ would be constructive, contradicting the validity of T.

The search halts when either

- the newly created point numerically lies on a line that does not share an edge with, or
- the newly created line numerically passes through a point that does not share an edge with, or
- a preset step limit is reached.

After each halt we discard superfluous constructions, reduce the theorem graph, and keep the resulting theorem. This yields approximately 10,000 distinct base theorems.

Theorem augmentation

To enlarge the corpus we apply three augmentation rules to every base theorem:

1. **Add redundant coincidences.** For example, if points A, B, and C seem unrelated, we postulate A, B, C are collinear and check whether the theorem still holds[5].
2. **Swap condition and goal.** If the original statement assumes $A \in \ell$, we instead assume the original goal and set $A \notin \ell$ as the *new* goal.
3. **Merge two theorems.** Given T_1 and T_2, we replace a hypothesis of T_2 with the conclusion of T_1, forcing any proof of T_2 to establish T_1 first.

We filter out invalid augmented theorems, apply theorem reduction, and finally obtain more than 8 million distinct pure intersection-point theorems.

Proof generation

To serve as training data, every theorem in the corpus must be accompanied by a machine-checkable proof. Our strategy is to run an exhaustive—but carefully pruned—search on each candidate theorem; theorems for which no proof is found within a time limit are discarded.

Overall proof schema. Starting from the initial theorem graph, we repeatedly

1. apply `merge-by-cr` (3.8) to *merge* exactly two vertices, and
2. immediately apply `merge-by-def` (3.9) *exhaustively*, collapsing every point-point or line-line pair that now coincides by definition.

After each merge cycle we check whether the goal edge is present; if so, the current sequence of tactic applications constitutes a valid proof and the search terminates successfully.

Finding a `merge-by-cr` opportunity. The main bottleneck is locating a pair of cross-ratio tuples to which `merge-by-cr` can be applied. In practice this requires first generating a large supply of *equivalent* cross-ratios via

[5]Concretely, introduce a new vertex l and add the edges (A, l), (B, l), and (C, l). If the resulting theorem graph, together with the original goal pair, is still valid, the augmented theorem is kept.

- `cr-equal` (3.6), which equates two tuples when their corresponding points or lines coincide, and
- `cr-equiv` (3.7), which propagates equality transitively.

Search routine. Our proof-search pipeline proceeds as follows:

1. **Augment.** Add auxiliary points and lines by repeatedly intersecting existing lines or joining existing points.
2. **Saturate.** Apply `cr-equal` and `cr-equiv` exhaustively, enlarging the equality graph of cross-ratios.
3. **Select a merge.** If at least one instance of `merge-by-cr` is now enabled, choose one[6] and perform the merge; then execute the automatic `merge-by-def`-phase.
4. **Check goal.** If the goal incidence appears, record the entire tactic sequence as a proof and stop; otherwise, return to Step 1. Abort if a global step or time limit is exceeded.

The dilemma of construction depth. Let V be the initial vertex set (level 0). Define recursively

$$\begin{aligned} V_k &:= \bigcup_{i=0}^{k-1} V_i \times V_{k-1-i}, \\ V' \times V'' &:= \{\, v' \wedge v'' \mid v' \in V',\ v'' \in V'',\ v', v'' \text{ have the } \textit{same} \text{ type and are distinct} \}, \end{aligned} \tag{3.10}$$

where $v' \wedge v''$ denotes the line through two points or the intersection of two lines. The union $\bigcup_{k\geq 0} V_k$ contains *all* constructible objects, but $|V_k|$ grows exponentially in k.

Dilemma. Choosing a small depth k (e.g. 1 or 2) keeps the search space small but may omit the constructions needed for some proofs. Choosing a large k (≥ 3) explodes the search space and floods the routine with merges that lead into infinite "dead-end" construction paths.

Harmonic cross-ratios to the rescue.

Definition 8 (Harmonic cross-ratio)**.** *A cross-ratio tuple $cr = (A, B, C, D)$ is* harmonic *if it is equivalent to a tuple obtained by swapping exactly two elements (e.g. (A, D, C, B)).*

Theorem 4. *Any two harmonic cross-ratio tuples are equivalent under a suitable permutation.*

(The proof is technical; see Appendix A.2.1.)

With this tool, we extend Step 2:

2. **Saturate with harmonic cross-ratios.** Apply `cr-equal`, `cr-equiv`, and the harmonic rule from Theorem 4 exhaustively, greatly enlarging the pool of equivalent tuples.

Three mechanisms now certify that two cross-ratios are equivalent:

[6] Implemented via a heuristic width-bounded BFS to minimise the search space.

1. **Chain transitivity.** A path of `cr-equal` links, closed by `cr-equiv`.
2. **Cycle transitivity.** If a tuple can be returned to itself by swapping exactly two of its elements—and the same holds for a second tuple—then both tuples are *harmonic*, and Theorem 4 gives their equality under the corresponding permutation.
3. **Mixed transitivity.** A chain leading to a permutation of the target tuple, followed by harmonic equivalence, closes the loop.

Outcome. Applying this procedure to the > 8 million augmented theorems, we discarded those without a proof within the budget. The survivors yield a corpus of roughly **eight million** valid PureIntersectionPoint theorems, each paired with at least one machine-checkable proof.

3.3 Dataset Encoding Schemes

Every theorem in the dataset is represented as a pair $T = (G = (V, E), (u, v))$, and every proof is a sequence of tactics that grows a sub-graph of the working graph $\mathcal{G}$ until the goal edge appears. Because both theorems and proofs are graphs or graph transformations, there are many plausible tokenisations. For example, one could emit a token stream that lists each tactic followed by the vertices to which it is applied. In practice, however, that "flat" format makes it hard for a model to learn *why* a tactic is chosen—intuitive sub-goals (such as "merge these two vertices") are posponded to the future therefore invisible now.

To study how models acquire intuition, we experiment with the following configurations.

Vertex labelling. All points are labelled $p*$ and all lines $l*$, where $* \in \{0, \ldots, 499\}$.[7] We consider two schemes:

- **Random labelling.** Every time a new vertex appears—whether in the statement or in the proof—we randomly assign it an *unused* label.
- **Ascending labelling.** We first choose two random starting indices, one for points and one for lines. Subsequent vertices of each type receive the next unused label in ascending order, taken modulo 500.[8]

Problem order. A theorem statement can be serialized in either of two ways:

1. **Random order.** We randomly shuffle the edges of the theorem graph G and list them as

$$(u_1, v_1),\ (u_2, v_2),\ \ldots,\ (u_n, v_n), \quad \textbf{goal: } (u, v), \tag{3.11}$$

   where (u_i, v_i) is the i-th edge in the shuffled sequence.

[7] Our corpus never exceeds 500 points or 500 lines per theorem.

[8] Using a *fixed* start such as p_0/l_0 would cap the largest index ever seen during training; labels beyond that cap would remain untrained, hindering generalisation to larger theorems. Randomising the start avoids this issue.

2. **Construction order.** We use the same format (3.11), but the edges now appear *exactly* in the order in which they are produced during the proof.

Concretely, in construction order we add an edge to the statement when:

- a vertex v is introduced by an *incidence choice*; we list the single edge (u, v), where u is the pre-existing point or line on/through which v is created;
- a vertex v is introduced as a *derived object*; we list both edges (u_1, v) and (u_2, v), where u_1, u_2 are the two objects used to create v.

Creations of *free* points or lines need not be included, because each such object eventually participates in a subsequent construction that records its first incidence.

A token stream written in construction order can be translated back into a constructive theorem statement in a unique way, because the ordered, oriented edge list fully specifies the topological order for construction.

Proof format. A naive way to serialize a proof is to list the tactics strictly in the order they are applied, each written as `tactic-name(subgraph)`. While concise, such a stream is nearly unreadable: when you encounter a tactic you have no clue *why* it was invoked or how its result will be used. In practice a proof is easier to follow when every tactic is preceded by the *intuition*—a mini-goal—that motivates it.

In our setting the structure is naturally two-tiered:

- **Outer tier: `merge-by-cr`.** We first state *which two vertices we intend to merge* with `merge-by-cr`, then supply the subgraph required by the rule.
- **Inner tier: cross-ratio equality.** To enable `merge-by-cr` we must show that two cross-ratio tuples are equal. Section 3.2 gives three ways to prove such an equality. Here we have two ordering options:
  - i) *Version 1 (goal-first).* State the conjectured equality of the two tuples, then present a proof using any of the three methods.
  - ii) *Version 2 (evidence-first).* Provide a chain of equalities using the three methods, and only afterwards declare the tuple equality as a consequence.

Unless stated otherwise, "format version 1" and "format version 2" refer to these two choices for the inner hierarchy.

Example. The next block shows a single theorem and its proof serialized with the options {`index order`: *ascending*, `thm order`: *construction*, `proof order`: *version 1*}.

Sample PureIntersectionPoint Theorem and Proof

begin_of_statement
(p106 l413) (p106 l414) (l414 p107) (l415 p107) (p107 l416) (p108 l416)
(l415 p109) (l413 p109) (p106 l417) (p108 l417) (l414 p110) (l417 p111)
(l415 p111) (p109 l418) (p108 l418) (p110 l419) (p108 l420) (p110 l420)
(l419 p112) (l418 p112) (l413 p113) (l420 p113) (l419 p114) (l416 p114)
(p112 l421) (p113 l421) (l421 p115) (l417 p115) (p111 l422) (p110 l422)
(p115 l423) (p114 l423) (l422 p116) (l423 p116) (p116 l424) (p113 l424)
(p110 l425) (p115 l425) (l417 p117) (l424 p117) (l423 p118) (l415 p118)
(p117 l426) (p118 l426) (p111 l427) (p112 l427) (l427 p119) (l425 p119)
(p119 l428) (p116 l428) (l426 p120) (l427 p120) (l425 p121) (l424 p121)
(p120 l429) (p121 l429) (l426 p122) (l428 p122) (p122 l430) (p121 l430)
(l427 p123) (l430 p123) (l428 p124) (l429 p124) (p124 l431) (p123 l431)
(l431 p125) (l426 p125)
thm_goal p125 l422

Answer:
merge_by_cr [p122 [l415 l424] p126] l432 l430
– p122 (l426 l428 [p113 p122] l433 l432) !!= p122 (l426 l428 l433 l430)
– these_two_crs_are_equal p122 (l426 l428 l433 l432) ?= p122 (l426 l428 l433 l430)
— p122 (l426 l428 l433 l432) = l424 (p117 p116 p113 p126)
— l424 (p117 p116 p113 p126) = p111 (l417 l422 [p111 p113] l434 l415)
— p111 (l417 l422 l434 l415) = l420 (p108 p110 p113 [l415 l420] p127)
— l420 (p108 p110 p113 p127) = p109 (l418 [p110 p109] l435 l413 l415)
— p109 (l418 l435 l413 l415) = l414 ([l414 l418] p128 p110 p106 p107)
— l414 (p128 p110 p106 p107) = p108 (l418 l420 l417 l416)
— p108 (l418 l420 l417 l416) = l419 (p112 p110 [l417 l419] p129 p114)
— l419 (p112 p110 p129 p114) = p115 (l421 l425 l417 l423)
— p115 (l421 l425 l417 l423) = l424 (p113 p121 p117 p116)
— l424 (p113 p121 p117 p116) = p122 (l433 l430 l426 l428)
— p122 (l433 l430 l426 l428) = p122 (l426 l428 l433 l430)
merge_by_def p126 p121
– they_share_those_two_neighbours l424 l432
merge_by_cr [l429 [p119 [l426 l422] p130] l436] p131 [l429 [p119 p125] l437] p132
– l429 (p124 p126 p131 p120) !!= l429 (p124 p126 p132 p120)
– this_cr_is_harmonic l429 (p124 p126 p131 p120)
— l429 (p124 p126 p131 p120) = p119 (l428 l425 l436 l427)
— p119 (l428 l425 l436 l427) = l422 (p116 p110 p130 p111)
— l422 (p116 p110 p130 p111) = p117 (l424 [p110 p117] l438 l426 l417)
— p117 (l424 l438 l426 l417) = l425 (p126 p110 [l425 l426] p133 p115)
— l425 (p126 p110 p133 p115) = p118 (l415 [p118 p110] l439 l426 l423)
— p118 (l415 l439 l426 l423) = l422 (p111 p110 p130 p116)
– this_cr_is_harmonic l429 (p124 p126 p132 p120)
— l429 (p124 p126 p132 p120) = p119 (l428 l425 l437 l427)
— p119 (l428 l425 l437 l427) = l432 (p122 p126 [l437 l432] p134 p123)

— l432 (p122 p126 p134 p123) = p125 (l426 [p126 p125] l440 l437 l431)
— p125 (l426 l440 l437 l431) = l429 (p120 p126 p132 p124)
merge_by_def l436 l437
– they_share_those_two_neighbours p119 p131
merge_by_def p130 p125
– they_share_those_two_neighbours l426 l436
end_of_statement

Chapter 4

Baseline Results under Default Pre-training

4.1 Experimental Setup

The 8-million-theorem PureIntersectionPoint corpus contains no isomorphic duplicates, so a straightforward train-test split suffices. We sort the theorems by the number of vertices in their statements and assign even-indexed items to the training set and odd-indexed items to the test set, resulting in 4 M theorems for each split.

Models. We evaluate two GPT-2 backbones—**small** (12 layers, 12 heads, 768 hidden units) and **medium** (20 layers, 16 heads, 1 024 hidden units)—each augmented with Canon layers [All25].

Training. All models are trained from scratch for 200 k steps with mini-batches of 128 sequences, each 1 024 tokens long. We use the AdamW optimiser and grid-search the learning rate in $\{2 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}\}$ and the weight decay in $\{0.0125, 0.025, 0.05\}$; the configuration that maximises test accuracy is reported.

Purpose. These baselines allow us to isolate the effect of alternative data formats and to analyse how models acquire geometric-reasoning skills during pre-training.

4.2 Result 1: Random Index Order Is Hard to Learn

When the model is trained under the fully shuffled configuration {`index order`: *random*, `thm order`: *random*, `proof order`: *version 1*}, its test accuracy stays below 1 % for both the *small* and *medium* backbones. In other words, the model fails to learn the task at all.

Failure mode. Inspection shows that the decoder frequently assigns *duplicate* labels to fresh vertices; the proof stream then violates the no-collision constraint and is rejected immediately.

Model	Random theorem order	Construction theorem order
GPT-2 small	17.2%	50.8%
GPT-2 medium	31.3%	70.3%

Table 4.1: Test accuracy on the PureIntersectionPoint dataset under different theorem orders.

Why duplicates are inevitable. Large language models struggle to generate long sequences of non-repeating tokens—even when explicitly trained to do so:

- Sampling **100 unique labels** from a pool of 500: in more than 99 % of trials the first duplicate appears within the first *six* tokens.
- Sampling **100 unique ordered pairs** (a, b) with $a, b \in \{1, \ldots, 32\}$: again, a duplicate pair is emitted in the first six outputs with probability above 99 %.

Take-away. The poor performance is therefore *not* due to insufficient data or model capacity; it is an inherent consequence of presenting the underlying graph in a completely random index order. Subsequent sections show that accuracy improves dramatically once the model can rely on a deterministic—or at least partially ordered—labeling scheme.

Therefore, in the rest of this section, we focus on the **ascending index order** configuration, which is the default setting for the PureIntersectionPoint dataset.

4.3 Result 2: Ordered Theorems Are Easier to Learn

We compare two configurations:

- **random theorem order**: {`index order`: *ascending*, `thm order`: *random*, `proof order`: *version 1*}
- **construction theorem order**: {`index order`: *ascending*, `thm order`: *construction*, `proof order`: *version 1*}

Results. A key difference from Section 4.2 is that the model no longer assigns duplicate labels to fresh vertices. This frees it to focus on learning the underlying reasoning instead of merely satisfying the no-collision constraint.

Table 4.1 shows non-trivial accuracy in both cases, but the construction-order setting is markedly easier: the *small* backbone reaches 50.8% accuracy, and the *medium* backbone reaches 70.3%.

A thorough analysis of *why* construction order helps appears in Section 4.5.

4.4 Result 3: Active Exploration Improves Learning

Does presenting edges in construction order already push accuracy to its ceiling? Not quite. We can further boost performance by changing the *proof order*.

Model	Format version 1	Format version 2
GPT-2 small	50.8%	61.2%
GPT-2 medium	70.3%	74.0%

Table 4.2: Test accuracy on the PureIntersectionPoint dataset under different proof orders.

Observed failure mode. Even with construction order, the model often **posits a plausible inner-tier subgoal but fails to prove it**. For example, it may assert that "cross-ratio$_1$ = cross-ratio$_2$", then wander through an ever-longer chain of equalities before devolving into nonsense tokens.

Active exploration (format version 2). To address this, we adopt *format version 2*, which we call **active exploration**: while extending a chain of cross-ratio equalities, the model *actively chooses* which mini-goal to aim for. It can change course mid-chain—spotting a promising pattern and steering the next step so that the chain eventually lands on a valid subgoal.

Setups compared. We therefore compare:

- **Format version 1 (goal-first):**
  {`index order`: *ascending*, `thm order`: *construction*, `proof order`: *version 1*}
- **Format version 2 (active exploration):**
  {`index order`: *ascending*, `thm order`: *construction*, `proof order`: *version 2*}

Results. Table 4.2 shows a clear gain from active exploration, especially for the *small* backbone (50.8% → 61.2%). The *medium* backbone also improves (70.3% → 74.0%).

Here is an example generated by GPT-2 medium showing how the model explores the proof space *without losing sight of the goal*:

Active exploration example

begin_of_statement
(p422 l97) (p422 l98) (p423 l98) (p423 l99) (l99 p424) (l97 p424)
(l97 p425) (l99 p426) (p426 l100) (p425 l100) (p422 l101) (p426 l101)
(p423 l102) (p425 l102) (l102 p427) (l101 p427) (l98 p428) (l100 p428)
(p428 l103) (p427 l103) (p424 l104) (p428 l104) (l103 p429) (l97 p429)
(l104 p430) (l101 p430) (p429 l105) (p430 l105) (l105 p431) (l99 p431)
(p431 l106) (p425 l106) (l100 p432) (l105 p432) (l103 p433) (l99 p433)
(l106 p434) (l98 p434) (p432 l107) (p427 l107) (l107 p435) (l106 p435)
(p433 l108) (p435 l108) (l97 p436) (l108 p436) (p436 l109) (p434 l109)
thm_goal p430 l109

Answer:
merge_by_cr [[p434 p430] l110 l97] p437 p436
– l97 (p429 p425 p424 p437) !!= l97 (p429 p425 p424 p436)
— l97 (p429 p425 p424 p437) = p430 (l105 [p425 p430] l111 l104 l110)
— p430 (l105 l111 l104 l110) = l106 (p431 p425 [l104 l106] p438 p434)
— l106 (p431 p425 p438 p434) = p428 ([p431 p428] l112 l100 l104 l98)
— p428 (l112 l100 l104 l98) = l99 (p431 p426 p424 p423)
— l99 (p431 p426 p424 p423) = p430 (l105 l101 l104 [p423 p430] l113)
— p430 (l105 l101 l104 l113) = l98 ([l105 l98] p439 p422 p428 p423)
— l98 (p439 p422 p428 p423) = p429 (l105 l97 l103 [p429 p423] l114)
— p429 (l105 l97 l103 l114) = l102 ([l102 l105] p440 p425 p427 p423)
— l102 (p440 p425 p427 p423) = p431 (l105 l106 [p427 p431] l115 l99)
— p431 (l105 l106 l115 l99) = l107 (p432 p435 p427 [l107 l99] p441)
— l107 (p432 p435 p427 p441) = p433 ([p433 p432] l116 l108 l103 l99)
— p433 (l116 l108 l103 l99) = l105 (p432 [l105 l108] p442 p429 p431)
— l105 (p432 p442 p429 p431) = p435 (l107 l108 [p429 p435] l117 l106)
— p435 (l107 l108 l117 l106) = l97 ([l107 l97] p443 p436 p429 p425)
— l97 (p443 p436 p429 p425) = p427 (l107 [p427 p436] l118 l103 l102)
— p427 (l107 l118 l103 l102) = l108 (p435 p436 p433 [l102 l108] p444)
— l108 (p435 p436 p433 p444) = p425 (l106 l97 [p425 p433] l119 l102)
— p425 (l106 l97 l119 l102) = l99 (p431 p424 p433 p423)
— l99 (p431 p424 p433 p423) = p429 (l105 l97 l103 l114)
— p429 (l105 l97 l103 l114) = l98 (p439 p422 p428 p423)
— l98 (p439 p422 p428 p423) = p430 (l105 l101 l104 l113)
— p430 (l105 l101 l104 l113) = l99 (p431 p426 p424 p423)
— l99 (p431 p426 p424 p423) = p425 (l106 l100 l97 l102)
— p425 (l106 l100 l97 l102) = l105 (p431 p432 p429 p440)
— l105 (p431 p432 p429 p440) = p427 (l115 l107 l103 l102)
— p427 (l115 l107 l103 l102) = l106 (p431 p435 [l106 l103] p445 p425)
— l106 (p431 p435 p445 p425) = p433 (l99 l108 l103 l119)
— p433 (l99 l108 l103 l119) = l97 (p424 p436 p429 p425)
— l97 (p424 p436 p429 p425) = l97 (p429 p425 p424 p436)
– these_two_crs_are_equal l97 (p429 p425 p424 p437) ?= l97 (p429 p425 p424 p436)

merge_by_def l110 l109
– they_share_those_two_neighbours p434 p437
end_of_statement

Active exploration example

begin_of_statement
(l53 p339) (l53 p340) (l53 p341) (l54 p342) (l54 p343) (l54 p344)
(p339 l55) (p343 l55) (p340 l56) (p344 l56) (p341 l57) (p342 l57)
(p340 l58) (p342 l58) (p341 l59) (p343 l59) (p339 l60) (p344 l60)
(l55 p345) (l58 p345) (l56 p346) (l59 p346) (l57 p347) (l60 p347)
(p345 l61) (p347 l61)
thm_goal p346 l61
Answer:
merge_by_cr [p346 p345] l62 [p346 p347] l63
– p346 ([p342 p346] l64 l62 l59 l56) !!= p346 (l64 l63 l59 l56)
– these_two_crs_are_equal p346 (l64 l62 l59 l56) ?= p346 (l64 l63 l59 l56)
— p346 (l64 l62 l59 l56) = l58 (p342 p345 [l59 l58] p348 p340)
— l58 (p342 p345 p348 p340) = p343 (l54 l55 l59 [p343 p340] l65)
— p343 (l54 l55 l59 l65) = l53 ([l54 l53] p349 p339 p341 p340)
— l53 (p349 p339 p341 p340) = p344 (l54 l60 [p344 p341] l66 l56)
— p344 (l54 l60 l66 l56) = l57 (p342 p347 p341 [l56 l57] p350)
— l57 (p342 p347 p341 p350) = p346 (l64 l63 l59 l56)
merge_by_def l61 l62
– they_share_those_two_neighbours p345 p347
end_of_statement

In this run the model builds a chain of length 29—substantially longer than any chain seen during training. A closer look reveals that it revisits the cross-ratio `p429 ( l105 l97 l103 l114 )` *twice*. Such deliberate re-use of a previously established cross-ratio never appears in the training data.

These behaviours suggest that the model (1) learns from earlier failed attempts and adjusts its strategy on the fly, and (2) abstracts the "chain/circle" pattern from training and applies it to unseen cases—evidence of genuine generalisation rather than rote memorisation.

4.5 Result 4: Intuitions Emerge During Training

Returning to the configuration {`index order`: *ascending*, `thm order`: *construction*, `proof order`: *version 1*}, one might ask: why can the model often predict *which two vertices to merge* but fail to predict *what is required* by the `merge-by-cr` tactic? More concretely, if the model reliably identifies a merge pair, it must "know" the corresponding subgraph of the theorem graph. If it really knows that subgraph, it should have seen similar instances

Partitions	Evaluated segment	10k	50k	100k	150k	200k
Trivial	**First line**	60.2%	39.8%	37.5%	35.9%	36.7%
Non-trivial	**First line**	12.5%	41.4%	52.3%	57.8%	59.4%
Trivial	**Whole step**	7.0%	13.2%	16.4%	19.5%	21.1%
Non-trivial	**Whole step**	7.0%	31.25%	43.8%	56.3%	56.3%

Table 4.3: The proportion of trivial and non-trivial first-line theorems in the solutions generated by the model at different training steps.

during training—including how to carry out `merge-by-cr`.

We consider two explanations:

- The model learns features that *hint* a subgoal is likely, but do not guarantee it. Thus, even without capturing the entire subgraph, it can still guess a merge pair with non-trivial probability. This vague intuition prevents it from always applying `merge-by-cr` perfectly.
- The model is genuinely familiar with *some* subgraphs, but elsewhere relies on "tricks" to guess the merge pair. When it uses a known subgraph it succeeds at `merge-by-cr` with high probability; when it relies on the trick, it typically fails.

We cannot directly rule out the first explanation, but we do find evidence supporting the second.

Definition 9 (First-line theorem)**.** *Given a theorem T together with its (not necessarily correct) proof, the first line of the proof is always the goal of a* ***`merge-by-cr`*** *tactic in our setting. Replace the original goal of T by this first-line (sub)goal to obtain a new theorem T'. We call T' a* ***first-line theorem*** *of the proof.*

Definition 10 (First-line trivial theorem)**.** *A first-line theorem is* ***trivial*** *if it can be derived by repeatedly applying* ***`merge-by-def`*** *once we additionally assume the original goal holds.*

"Trivial" here does *not* mean the theorem is easy to prove; it means that the subgoal is easy to *predict* given the original goal. For language models, proposing trivial subgoals can be seen as a "trick" for guessing which two vertices to merge.

Therefore, we can split the set of solutions generated by the model into three partitions: (1) the one whose first-line theorem is not valid, (2) the one with trivial first-line theorem, and (3) the one with non-trivial trivial first-line theorem.

Table 4.3 shows:

- **Trivial subgoals are easy to pick up early.** At 10k steps the model proposes trivial first-line theorems in 60.2% of cases, indicating that this "shortcut" is quickly learned.
- **Non-trivial subgoals are learned gradually.** Their share rises from 12.5% at 10k to 59.4% at 200k steps.
- **Non-trivial subgoals are executed more reliably.** At 200k, the success rate of completing the entire first step is $56.3\%/59.4\% \approx 94.8\%$ for non-trivial subgoals versus $21.1\%/36.7\% \approx 57.5\%$ for trivial ones, suggesting the model's predictions are more faithful when it relies on genuine structure rather than a trivial trick.

Partitions	Evaluated segment	10k	50k	100k	150k	200k
Trivial	**First line**	74.2%	63.3%	60.9%	53.9%	54.7%
Non-trivial	**First line**	3.1%	7.0%	18.8%	24.2%	32.0%
Trivial	**Whole step**	4.7%	7.0%	14.0%	17.2%	17.2%
Non-trivial	**Whole step**	0.8%	2.3%	7.8%	14.8%	21.0%

Table 4.4: The proportion of trivial and non-trivial first-line theorems in the solutions generated by the model at different training steps (random theorem order setting).

This pattern also helps explain the weak performance in the *random theorem order* setting. Table 4.4 reports the same breakdown: We see that the model rarely proposes non-trivial subgoals here: only 32.0% at 200k steps, versus 59.4% under construction order. A plausible explanation is that subgraph isomorphism is NP-hard in general, but geometric theorems expressed in construction order expose structural cues (e.g., incremental incidence patterns) that make the matching problem far easier for the model to learn.

Take-away. The model learns to propose subgoals *progressively*: it first picks easy (trivial) ones, then shifts toward harder, structure-driven (non-trivial) ones. This supports the second explanation above: early on, the model leans on cheap tricks; later, it actually recognises and exploits meaningful subgraphs. Once a non-trivial subgoal is chosen, the follow-up tactics succeed almost always, showing that the "right intuition" largely determines success.

What this implies.

- *Intuition emerges during training.* The rise of non-trivial first-line theorems is a direct signal that useful heuristics are being formed.
- *Order matters.* Presenting edges in construction order acts like a curriculum: it lowers the search burden for the matching subgraph and lets intuition form earlier.
- *Measureable signal.* The fraction of non-trivial first-line theorems (and their completion rate) is a simple, task-intrinsic proxy for "how much intuition" the model has learned.

These observations motivate the next chapter, where we will intervene on which subgraphs the model masters, test whether it can develop unseen skills, and explore how to boost performance on the PureIntersectionPoint dataset.

Chapter 5

How LM Master Intuition-A Deeper Analysis

Last chapter we observed three key phenomena: (1) ordering the input (indices and theorem edges) dramatically eases learning; (2) allowing *active exploration* (format version 2) further boosts accuracy; and (3) the fraction of *non-trivial* first-line subgoals rises steadily during training, suggesting that "intuition" is something the model *acquires*, not something it has at initialization. In this chapter we ask: **What intuitions does the model exactly learn, and how?**

To answer this, we need to introduce some new notations.

5.1 Maximally-Reduced First-Line Theorem

A raw first-line theorem T (Definition 9) produced by the model can be quite messy:

- It may "reconstruct" an object that already exists. For example, if $l_1 \cap l_2 = p_1$ and $l_1 \cap l_3 = p_2$, the model might connect p_1 and p_2 to form a "new" line that is in fact identical to l_2, yielding a degenerate situation (Definition 5).
- It may introduce an object that is *geometrically identical* to an existing one in a way that is trivial *given the original theorem goal*. For example, suppose the original goal is $p_1 \in l_1$ and, in the original theorem, l_1 was defined as the line through p_2 and p_3. A frequent first-line subgoal is to "prove" that the line through p_1 and p_3 (call it l_2) coincides with l_1. Once $p_1 \in l_1$ is assumed (or once `merge-by-def` applies), this coincidence follows immediately and is therefore *trivial*. Such patterns can generate arbitrarily many trivial first-line theorems at negligible cost.

Counting all such variants separately would let the model inflate its "diversity" of subgoals almost for free—making "intuition" cheap. We therefore collapse each raw first-line theorem to a canonical core.

Definition 11 (Compactly reduced theorem)**.** *Let a first-line theorem be given as an ordered statement with construction order $\sigma = (v_1, \ldots, v_m)$, where $v_1, \ldots, v_n$ come from the original theorem statement and $v_{n+1}, \ldots, v_m$ are introduced by the model.*

Process the sequence once, in order $i = 1$ to m:

- *If v_i is* syntactically identical *to an earlier vertex $v_i' \in \{v_1, \ldots, v_{i-1}\}$, ignore v_i and replace every future occurrence of v_i by v_i'.*
- *If v_i is* geometrically identical *to an earlier vertex v_i', stop and set the new goal to "v_i coincides with v_i'".*

The statement obtained at termination is the compactly reduced *version of the first-line theorem.*

Because the original theorem is non-degenerate, the outcome of this reduction does not depend on the internal order of constructions that belonged to the original statement; only the model's additions affect the result.

For a first-line theorem T, we can compactly reduce it to a *compactly reduced first-line theorem* T', which is a non-degenerate theorem without any extra coincidence.

Because T' is non-degenerate, we can further reduce T' by the theorem reduction technique (Definition 7) to obtain a further reduced theorem T''. According to Theorem 3, T'' valid if T' is.

Can we reduce T'' further? The answer is yes.

Algorithm 1 Maximally reduced theorem

Input: A non-degenerate, valid theorem $T = (G = (V, E), (u, v))$.
Output: A maxiamlly reduced theorem T.
while True **do**
    reduced $\leftarrow$ False
    for $e \in E$ **do**
        $G' \leftarrow (V, E \setminus \{e\})$
        reduced theorem $\hat{T} \leftarrow$ reduce theorem $(G', (u, v))$
        if $\hat{T}$ is valid **then**
            $T \leftarrow \hat{T}$
            reduced $\leftarrow$ True
            break
        end if
    end for
    if not reduced **then**
        break
    end if
end while
return T

Different from theorem reduction in Definition 7, the maximally reduce theorem algorithm may result in different maximally reduced theorems. Such "ambiguise theorem" is about 5% among all theorems in PureIntersectionPoint. To be rigorous, we consider the union of all possible maximally reduced theorems as the maximally reduced theorems of T.

Model	Random index split	Subgoal-based split
GPT-2 small	50.8%	49.2%
GPT-2 medium	70.3%	50.8%

Table 5.1: Test accuracy on PureIntersectionPoint under the ordinary (index) split versus the subgoal-based split described above.

5.2 Result 5: Model never learn by rote

To probe whether the model's "intuition" is simply rote recall, we performed a targeted split based on the *maximally reduced* first-line theorems (Algorithm 1).

Dataset construction. We extracted every maximally reduced first-line theorem in the full PureIntersectionPoint corpus—about 800 000 in total—and ranked them by frequency. Even-ranked items were labelled *training*, odd-ranked items *testing*. A theorem then falls into one of three buckets:

- **Training set:** all of its standard solution's first-line theorems are in the training half.
- **Testing set:** all belong to the testing half.
- **Mixed set:** it contains both training and testing items.

Protocol. We pre-trained models solely on the *training* set and evaluated on the *testing* set. If success depended on memorising specific subgoals, accuracy should collapse on this split.

Results. Although accuracy drops on the subgoal split (Table 5.1), the models still achieve respectable performance—e.g. 50.8 % for the medium backbone—despite never seeing those subgoals during training.

Even more striking: among solutions whose *first line is correct and non-trivial*,

- on the training split, > 96 % of generated subgoals belong to the training half;
- on the testing split, > 62 % belong to the unseen testing half.

Thus the model produces unfamiliar subgoals with high probability—a strong sign it has learned abstract structural cues rather than memorising concrete instances.

Take-away. The model's intuition generalises: it recognises and re-creates patterns it never encountered in training, indicating genuine reasoning rather than rote recall.

5.3 Result 6: The Model Solves Theorems the Search Machine Cannot

The previous section showed that the model's intuition is not mere rote recall. Can it also solve problems that our theorem-search machine cannot? To test this, we collected 16 hard

PureIntersectionPoint theorems that the search machine fails to prove (Section 3.2). For each theorem we gave the model 16 chances: in each trial we randomly fixed a constructive order and asked it to produce a proof.[1]

Astonishingly, the model solved 4 out of 16 theorems—a **25%** success rate—despite (i) the search machine failing on all of them and (ii) the model never seeing these theorems during training.

Example. Below is one such theorem (and its proof) that the model solves but the search machine does not:

Sample PureIntersectionPoint Theorem and Proof

begin_of_statement
(l384 p457) (l384 p458) (p459 l385) (l385 p460) (p458 l386) (l384 p461)
(l385 p461) (p459 l387) (p458 l387) (p462 l388) (p460 l389) (p457 l389)
(p462 l390) (p461 l390) (p460 l391) (p462 l391) (l391 p463) (l387 p463)
(p457 l392) (p463 l392) (l390 p464) (l387 p464) (l389 p465) (l386 p465)
(p459 l393) (p465 l393) (l388 p466) (l386 p466) (l390 p467) (l392 p467)
(p466 l394) (p464 l394) (l388 p468) (l389 p468) (p468 l395) (p467 l395)
(l393 p469) (l394 p469)
thm_goal l395 p469

Answer:
merge_by_cr [l395 l394] p470 p469
– l394 (p464 p466 p470 [l389 l394] p471) !!= l394 (p464 p466 p469 p471)
– these_two_crs_are_equal l394 (p464 p466 p470 p471) ?= l394 (p464 p466 p469 p471)
— l394 (p464 p466 p470 p471) = p468 ([p464 p468] l396 l388 l395 l389)
— p468 (l396 l388 l395 l389) = l390 (p464 p462 p467 [l389 l390] p472)
— l390 (p464 p462 p467 p472) = p457 ([p464 p457] l397 [p462 p457] l398 l392 l389)
— p457 (l397 l398 l392 l389) = l391 ([l391 l397] p473 p462 p463 p460)
— l391 (p473 p462 p463 p460) = p464 (l397 l390 l387 [p460 p464] l399)
— p464 (l397 l390 l387 l399) = l385 ([l397 l385] p474 p461 p459 p460)
— l385 (p474 p461 p459 p460) = p457 (l397 l384 [p459 p457] l400 l389)
— p457 (l397 l384 l400 l389) = l387 (p464 p458 p459 [l387 l389] p475)
— l387 (p464 p458 p459 p475) = p465 ([p464 p465] l401 l386 l393 l389)
— p465 (l401 l386 l393 l389) = l394 (p464 p466 p469 p471)
end_of_statement

Why the machine fails but the model succeeds. The search machine faces the dilemma from Section 3.2: restricting auxiliary constructions to depth ≤ 2 misses solutions, whereas allowing depth ≥ 3 explodes the search space and floods it with merges that lead to infinite dead-end branches. The language model, by contrast, *actively explores* and prunes on the fly, steering itself toward a productive chain of equalities and a successful `merge-by-cr`.

[1]The statement order does not affect the search machine.

Chapter 6

Conclusions and Future Directions

This thesis investigated how large language models acquire *intuition*—simple yet powerful heuristics that guide long-horizon reasoning—within a strictly defined geometric proof domain. The main contributions are:

1. **PureIntersectionPoint dataset.** An 8-million-theorem corpus with a minimal tactic set, machine-checkable proofs, and flexible ordering schemes.
2. **Ordering matters.** Construction order acts as a curriculum, lifting accuracy from near-zero (random order) to over 70 %.
3. **Active exploration.** A proof format that lets the model "think about" subgoals delivers an additional double-digit improvement.
4. **Measuring intuition.** The notion of *maximally reduced first-line theorems* provides a simple, intrinsic proxy for how much structure the model has internalised.
5. **Generalisation beyond rote.** Even when test subgoals never occur in training, the model attains competitive accuracy and solves theorems that defeat a brute-force search engine.

Limitations

- *Domain specificity.* All experiments are confined to intersection-point geometry; it is unclear how readily the findings transfer to richer theorem spaces or other reasoning tasks.
- *Model scale.* Only GPT-2 small/medium backbones were tested. Larger models might exhibit qualitatively different behaviour.
- *Search baseline.* Our symbolic prover uses simple width-bounded search; stronger engines could close the gap highlighted in Result 6.

Future directions

1. **Self improving RL.** Fine tune the model with reinforcement learning from generated proofs, aiming for higher accuracy and harder theorems.

2. **Pattern probing via LoRA.** Pick classical patterns—e.g. Pappus, Desargues—and apply low rank adaptation [Hu+22] to test whether the pretrained model has already internalised these patterns.
3. **Cross domain transfer.** Build analogous corpora for number theory, algebra, and combinatorics to see whether the same intuition formation signals emerge, and whether skills learned in PureIntersectionPoint transfer.
4. **Incremental tactic sets.** Develop methods for injecting *new* tactics efficiently— e.g. via curriculum design or contrastive fine tuning—without catastrophic forgetting of earlier skills.

Closing Remarks The experiments show that with the right curriculum and representation, even small-sized LLMs develop non-trivial geometric intuition and can outperform brute-force search. These insights suggest a path toward hybrid systems where symbolic structure and neural flexibility reinforce one another—pushing automated reasoning closer to human-level ingenuity.

Bibliography

This bibliography contains 14 references.

[All25] Zeyuan Allen-Zhu. "Physics of Language Models: Part 4.1, Architecture Design and the Magic of Canon Layers". In: *SSRN Electronic Journal* (May 2025). https://ssrn.com/abstract=5240330. DOI: 10.2139/ssrn.5240330.

[Ant24] Anthropic. *The Claude 3 Model Family: Opus, Sonnet, Haiku*. Technical report. PapersWithCode lists 95% accuracy on GSM8K for Claude 3 Opus. 2024. URL: https://assets.anthropic.com/Claude-3-Model-Card.pdf.

[Bro+20] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[Bub+23] Sébastien Bubeck et al. *Sparks of Artificial General Intelligence: Early Experiments with GPT-4*. 2023. arXiv: 2303.12712 [cs.CL]. URL: https://arxiv.org/abs/2303.12712.

[Cho+22] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. arXiv: 2204.02311 [cs.CL]. URL: https://arxiv.org/abs/2204.02311.

[Cob+21] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. "Training verifiers to solve math word problems". In: *arXiv preprint arXiv:2110.14168* (2021).

[Gem25] Gemini Team, Google DeepMind. *Gemini: A Family of Highly Capable Multimodal Models*. Section 5 reports 94.4% accuracy on GSM8K. 2025. arXiv: 2312.11805 [cs.CL].

[Guo+25] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning". In: *arXiv preprint arXiv:2501.12948* (2025).

[Hen+21] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. "Measuring mathematical problem solving with the math dataset". In: *arXiv preprint arXiv:2103.03874* (2021).

[Hu+22] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. "Lora: Low-rank adaptation of large language models." In: *ICLR* 1.2 (2022), p. 3.

[Ope+24] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[Ope24] OpenAI. *GPT-4 Technical Report*. Table 2 reports 92.0% accuracy on GSM8K. 2024. arXiv: 2303.08774 [cs.CL].

[Ye+24a] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. "Physics of language models: Part 2.1, grade-school math and the hidden reasoning process". In: *arXiv preprint arXiv:2407.20311* (2024).

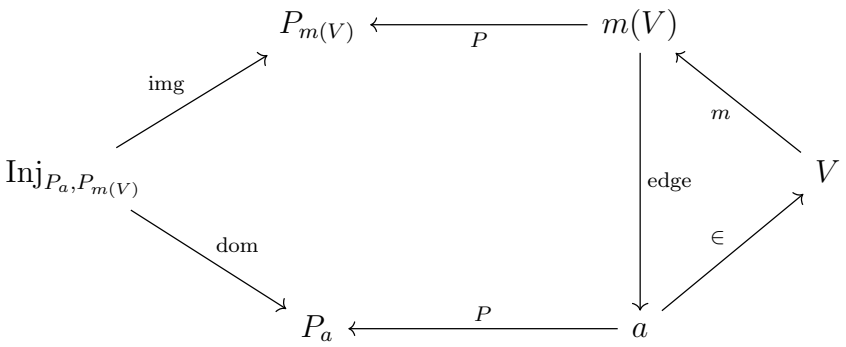
[Ye+24b] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. "Physics of language models: Part 2.2, how to learn from mistakes on grade-school math problems". In: *arXiv preprint arXiv:2408.16293* (2024).

Appendix A

Appendix

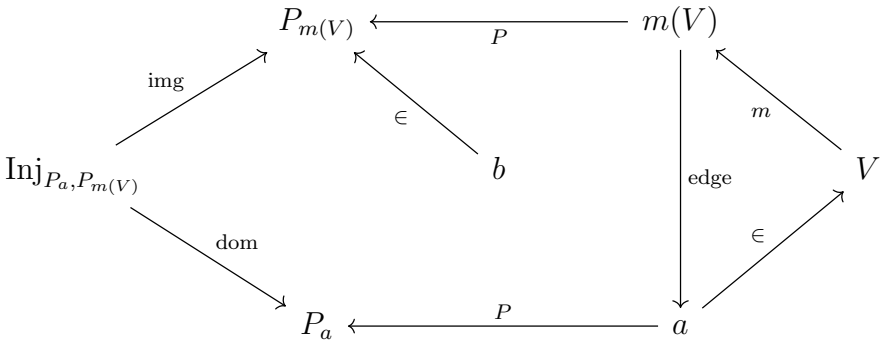
A.1 Proof of the Warm-Up Problem

Let's turn back to the proof of the warm-up problem. We can now apply tactics (2.4) and (2.7) to the subgraph $m(V) \xrightarrow{\in} V$, which yields



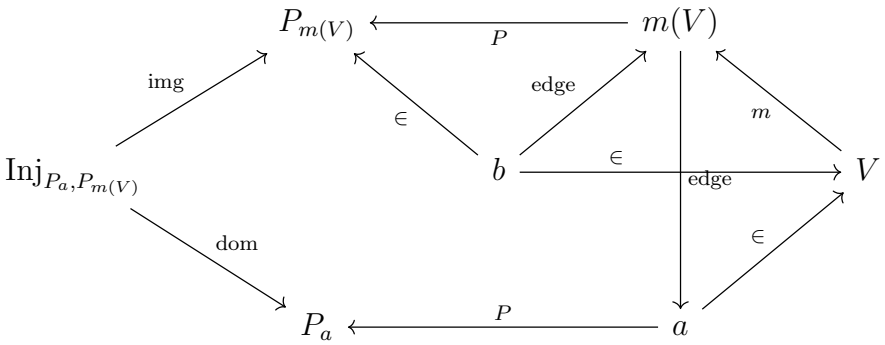
(A.1)

Next, we synthesize a custom tactic, starting from the following subgraph:



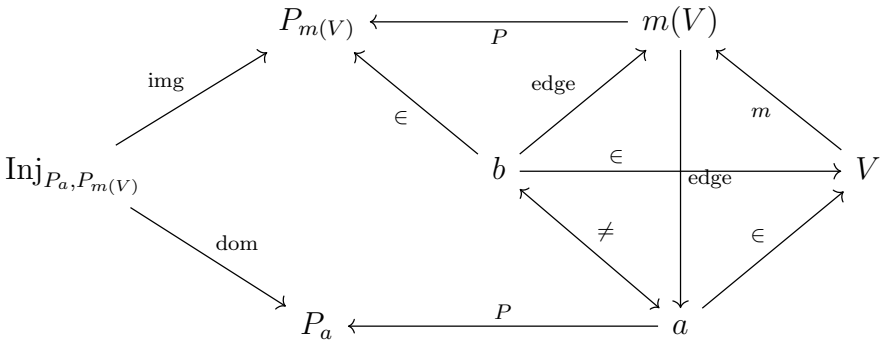
(A.2)

By tactic 2.7, this derives the following subgraph:



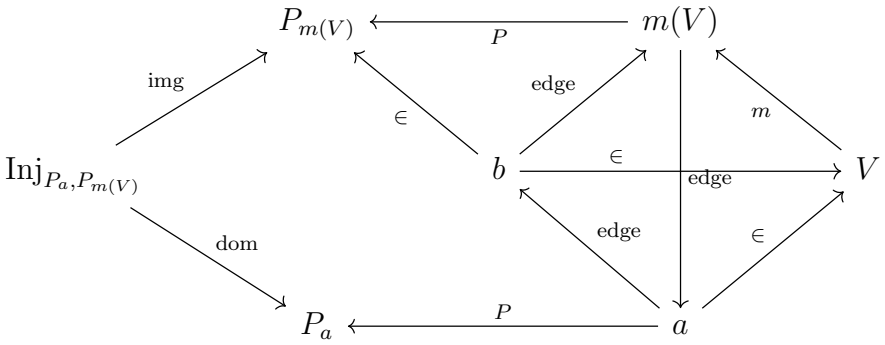
(A.3)

Applying tactic 2.3, we obtain:



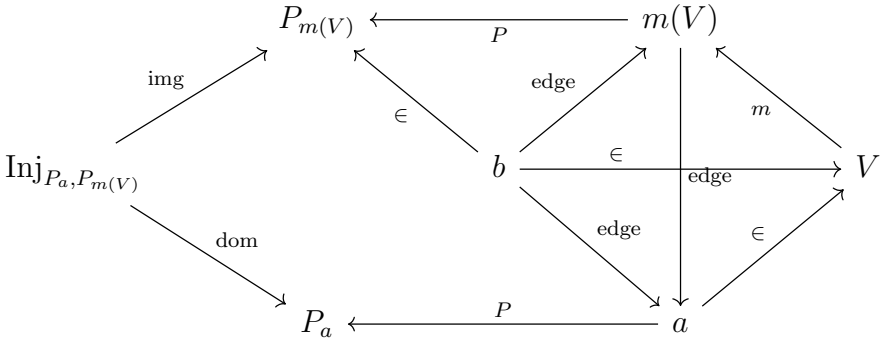
(A.4)

By tactic 2.2, this splits into the two cases:



(A.5)

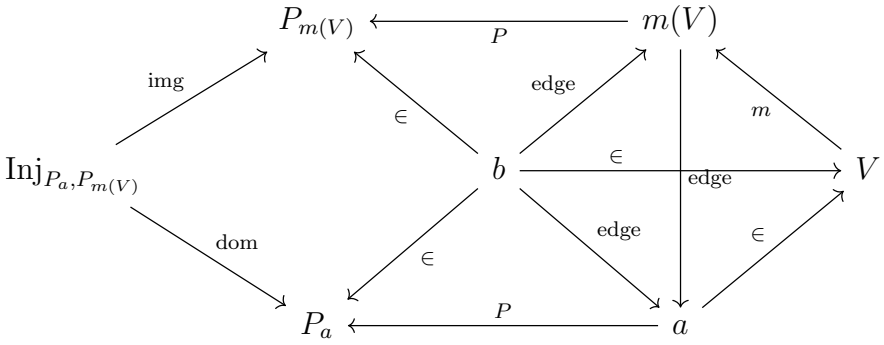
or



(A.6)

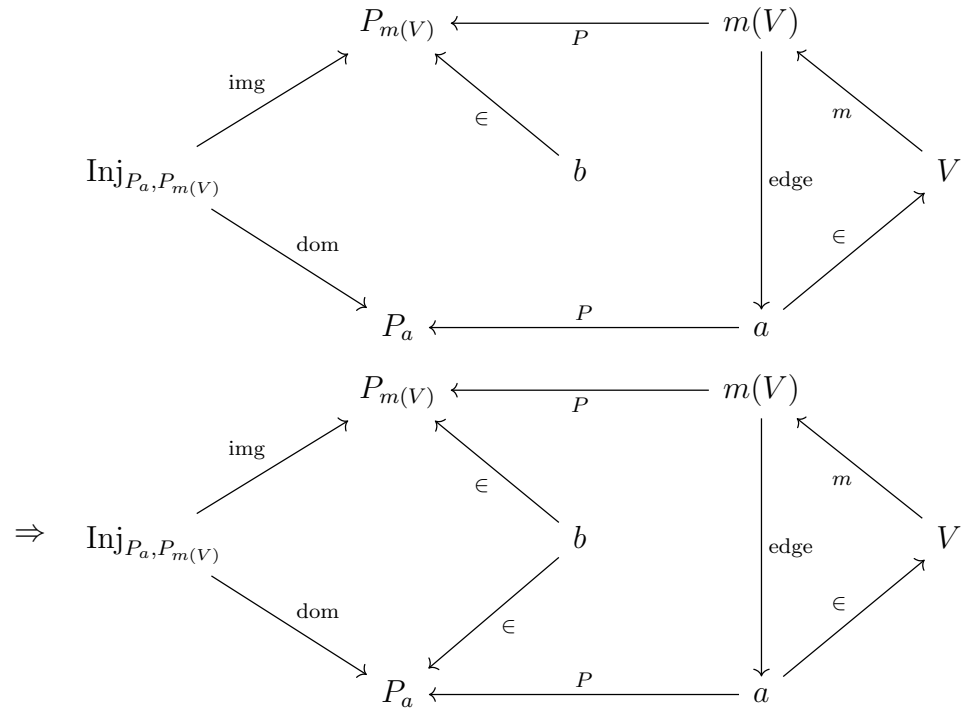
In subgraph (A.5), we already find a subgraph isomorphic to the goal graph (2.5), so only case (A.6) remains.

Again applying tactic 2.7, we derive:



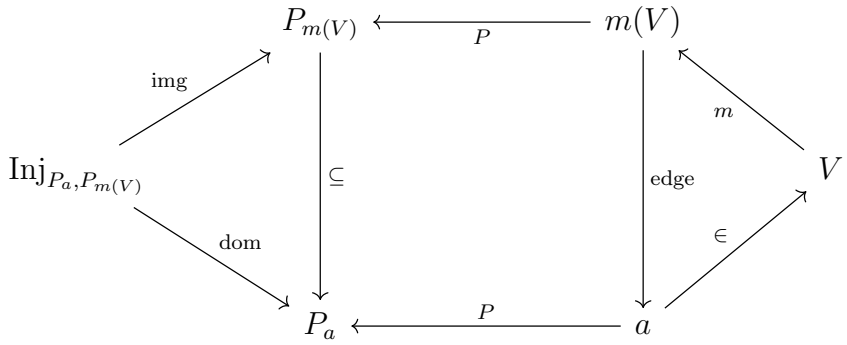
(A.7)

In summary, we have established the following tactic:



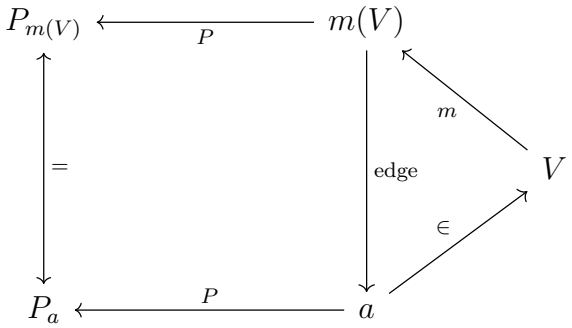
(A.8)

which can be summarized as the following status graph via the basic set-theoretic tactic:



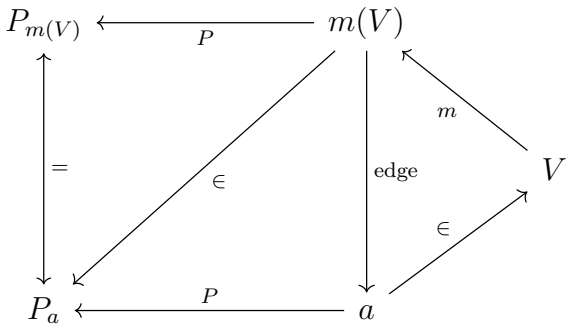
(A.9)

Applying the set-theoretic tactic once more, we obtain the status graph:



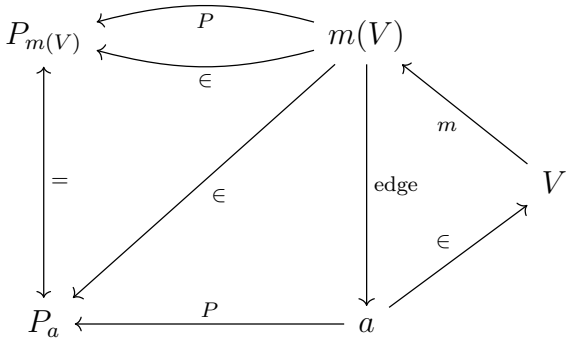
(A.10)

However, by tactic 2.7, we can further derive the following status graph:



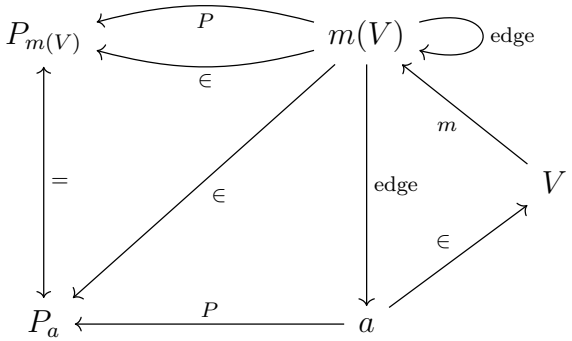
(A.11)

and then



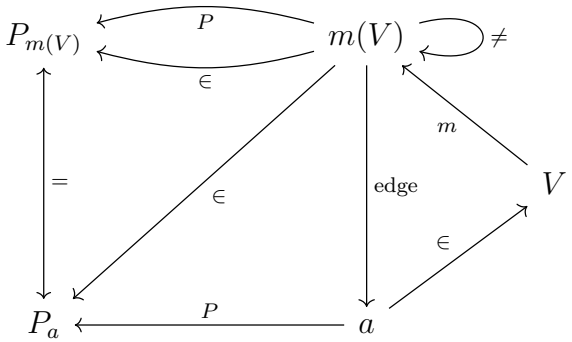
(A.12)

Again, by tactic 2.7, we have the following status graph:



(A.13)

By tactic 2.3, this yields the status graph:



(A.14)

which is a contradiction, since a vertex cannot be unequal to itself. Therefore, every valid branch must contain the goal graph (2.5) as a subgraph, and the proof is complete.

A.2 Introduction to PureIntersectionPoint Dataset

A.2.1 Introduction to the Cross-Ratio

The real projective plane and point–line duality

The real projective plane $\mathbb{RP}^2$ is the space of lines through the origin in $\mathbb{R}^3$. A *point* $v \in \mathbb{RP}^2$ is represented by any nonzero vector $\mathbf{v} \in \mathbb{R}^3$ up to nonzero scaling; we write $v = [\mathbf{v}]$. A *line* $\ell \subset \mathbb{RP}^2$ is represented by a two-dimensional linear subspace (a plane through the origin) in $\mathbb{R}^3$; equivalently, by a nonzero covector $\boldsymbol{\lambda} \in (\mathbb{R}^3)^*$ up to scale, with $\ell = \{[\mathbf{x}] : \boldsymbol{\lambda}(\mathbf{x}) = 0\}$.

Fix once and for all a Euclidean inner product $\langle \cdot, \cdot \rangle$ on $\mathbb{R}^3$ and use it to identify $(\mathbb{R}^3)^* \cong \mathbb{R}^3$. Under this identification we may represent lines also by nonzero vectors $\boldsymbol{\ell} \in \mathbb{R}^3$ up to scale, and the *incidence pairing*

$$\langle v, \ell \rangle \;:=\; \langle \mathbf{v}, \boldsymbol{\ell} \rangle$$

is well defined up to a common nonzero scalar: $\langle v, \ell \rangle = 0$ iff the point v lies on the line ℓ.

For $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3$, we will use the *scalar triple product*

$$[\mathbf{a}, \mathbf{b}, \mathbf{c}] \;:=\; \det(\mathbf{a}, \mathbf{b}, \mathbf{c}) \;=\; \langle \mathbf{a},\ \mathbf{b} \times \mathbf{c} \rangle.$$

It is multilinear, alternating, and scales homogeneously with each argument.

Joins and meets. If $v = [\mathbf{v}]$ and $w = [\mathbf{w}]$ are two distinct points, their join (the unique line through them) is represented by the vector

$$\boldsymbol{\ell}_{vw} \;=\; \mathbf{v} \times \mathbf{w}.$$

Dually, if $\ell = [\boldsymbol{\ell}]$ and $m = [\boldsymbol{m}]$ are two distinct lines, their meet (the unique intersection point) is represented by

$$\mathbf{p}_{\ell m} \;=\; \boldsymbol{\ell} \times \boldsymbol{m}.$$

With our fixed identification, these are just the usual cross products in $\mathbb{R}^3$.

Cross-ratio on collinear points and concurrent lines

Let v_1, v_2, v_3, v_4 be four distinct collinear points in $\mathbb{RP}^2$, represented by nonzero vectors $\mathbf{v}_i \in \mathbb{R}^3$. Define their *cross-ratio* by

$$(v_1, v_2; v_3, v_4) \;:=\; \frac{\langle \mathbf{v}_1 \times \mathbf{v}_2 \,,\, \mathbf{v}_3 \times \mathbf{v}_4 \rangle}{\langle \mathbf{v}_2 \times \mathbf{v}_3 \,,\, \mathbf{v}_4 \times \mathbf{v}_1 \rangle}. \tag{A.15}$$

The numerator and denominator are homogeneous of the same degree in each $\mathbf{v}_i$, so the ratio is independent of the particular representatives chosen in $\mathbb{R}^3$ and depends only on the projective points v_i.

Dually, if $\ell_1, \ell_2, \ell_3, \ell_4$ are four distinct concurrent lines (all passing through a common point) represented by nonzero vectors $\boldsymbol{\ell}_i \in \mathbb{R}^3$, define

$$(\ell_1, \ell_2; \ell_3, \ell_4) \;:=\; \frac{\langle \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \,,\, \boldsymbol{\ell}_3 \times \boldsymbol{\ell}_4 \rangle}{\langle \boldsymbol{\ell}_2 \times \boldsymbol{\ell}_3 \,,\, \boldsymbol{\ell}_4 \times \boldsymbol{\ell}_1 \rangle}. \tag{A.16}$$

Again the ratio is representation-invariant.

Proposition 1 (An equivalent bracket formula)**.** *If v_1, v_2, v_3, v_4 are collinear and $\mathbf{p} \in \mathbb{R}^3$ represents any point $p \notin \overline{v_1 v_2}$, then*

$$(v_1, v_2; v_3, v_4) \;=\; \frac{[\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}]\,[\mathbf{v}_3, \mathbf{v}_4, \mathbf{p}]}{[\mathbf{v}_2, \mathbf{v}_3, \mathbf{p}]\,[\mathbf{v}_4, \mathbf{v}_1, \mathbf{p}]}. \tag{A.17}$$

Similarly, if $\ell_1, \ell_2, \ell_3, \ell_4$ are concurrent and $\mathbf{q} \in \mathbb{R}^3$ represents any line $\ell_1 \cap \ell_2 \notin q$, then

$$(\ell_1, \ell_2; \ell_3, \ell_4) \;=\; \frac{[\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \mathbf{q}]\,[\boldsymbol{\ell}_3, \boldsymbol{\ell}_4, \mathbf{q}]}{[\boldsymbol{\ell}_2, \boldsymbol{\ell}_3, \mathbf{q}]\,[\boldsymbol{\ell}_4, \boldsymbol{\ell}_1, \mathbf{q}]}. \tag{A.18}$$

Proof of Proposition 1. Let L be the common line of v_i, and pick a nonzero normal $\mathbf{n}$ to the plane representing L so that $\mathbf{v}_i \times \mathbf{v}_j = t_{ij}\,\mathbf{n}$ for scalars t_{ij}. For any $\mathbf{p}$ not on L we have

$$[\mathbf{v}_i, \mathbf{v}_j, \mathbf{p}] = \langle \mathbf{v}_i,\, \mathbf{v}_j \times \mathbf{p} \rangle = \langle \mathbf{p},\, \mathbf{v}_i \times \mathbf{v}_j \rangle = t_{ij}\,\langle \mathbf{p}, \mathbf{n} \rangle.$$

Hence

$$\langle \mathbf{v}_i \times \mathbf{v}_j,\, \mathbf{v}_k \times \mathbf{v}_\ell \rangle = t_{ij} t_{k\ell}\,\langle \mathbf{n}, \mathbf{n} \rangle = \frac{[\mathbf{v}_i, \mathbf{v}_j, \mathbf{p}]\,[\mathbf{v}_k, \mathbf{v}_\ell, \mathbf{p}]}{\langle \mathbf{p}, \mathbf{n} \rangle^2}\,\langle \mathbf{n}, \mathbf{n} \rangle.$$

The common factor cancels in the ratio (A.15), giving (A.17).

For concurrent lines, (A.18) follows by the same argument. □

Two basic facts

We now establish the two statements requested in the introduction.

Proposition 2 (Point–line dual invariance of cross-ratio)**.** *Let v_1, v_2, v_3, v_4 be four distinct collinear points and $\ell_1, \ell_2, \ell_3, \ell_4$ four distinct concurrent lines such that $\langle v_i, \ell_i \rangle = 0$ for $i = 1, 2, 3, 4$ (i.e. $v_i \in \ell_i$). Then*

$$(v_1, v_2; v_3, v_4) \;=\; (\ell_1, \ell_2; \ell_3, \ell_4).$$

Proof. Let p be their common concurrency point and choose a representative $\mathbf{p} \in \mathbb{R}^3$. By incidence and concurrency, each ℓ_i is exactly the line through v_i and p, hence $\boldsymbol{\ell}_i = \lambda_i(\mathbf{v}_i \times \mathbf{p})$ for some nonzero scalars λ_i. By Prop 1 we have

$$(v_1, v_2; v_3, v_4) = \frac{[\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}]\,[\mathbf{v}_3, \mathbf{v}_4, \mathbf{p}]}{[\mathbf{v}_2, \mathbf{v}_3, \mathbf{p}]\,[\mathbf{v}_4, \mathbf{v}_1, \mathbf{p}]}.$$

Using the fact that $(\mathbf{a} \times \mathbf{p}) \times (\mathbf{b} \times \mathbf{p}) = [\mathbf{a}, \mathbf{b}, \mathbf{p}]\,\mathbf{p}$, we compute

$$\langle \boldsymbol{\ell}_i \times \boldsymbol{\ell}_j,\, \boldsymbol{\ell}_k \times \boldsymbol{\ell}_\ell \rangle = \lambda_i\lambda_j\lambda_k\lambda_\ell\,[\mathbf{v}_i, \mathbf{v}_j, \mathbf{p}]\,[\mathbf{v}_k, \mathbf{v}_\ell, \mathbf{p}]\,\langle \mathbf{p}, \mathbf{p} \rangle,$$

so all λ's and $\langle \mathbf{p}, \mathbf{p} \rangle$ cancel in the ratio (A.16), yielding the same bracket expression as above. Hence the two cross-ratios are equal. □

Proposition 3 (Uniqueness of the fourth point)**.** *Let v_1, v_2, v_3, v_4, v_4' be five collinear points in $\mathbb{RP}^2$ with v_1, v_2, v_3 distinct. If*

$$(v_1, v_2; v_3, v_4) = (v_1, v_2; v_3, v_4'),$$

then $v_4 = v_4'$.

Proof. Let $L \subset \mathbb{RP}^2$ be the projective line containing v_1, v_2, v_3. Choose nonzero representatives $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^3$ that span the 2-dimensional subspace $U \subset \mathbb{R}^3$ projecting to L. For any nonzero $\mathbf{v} \in U$ representing $v = [\mathbf{v}] \in L$, set

$$N(\mathbf{v}) := \big\langle \mathbf{v}_1 \times \mathbf{v}_2\,,\, \mathbf{v}_3 \times \mathbf{v} \big\rangle, \qquad D(\mathbf{v}) := \big\langle \mathbf{v}_2 \times \mathbf{v}_3\,,\, \mathbf{v} \times \mathbf{v}_1 \big\rangle.$$

Define

$$f : L \to \mathbb{RP}^1, \qquad f([\mathbf{v}]) := [\,N(\mathbf{v}) : D(\mathbf{v})\,]. \tag{A.19}$$

Step 1: f is well-defined and projective-linear. First, N and D are *linear* in $\mathbf{v}$. Indeed, using the standard identity

$$(\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{v}) = (\mathbf{a} \cdot \mathbf{c})(\mathbf{b} \cdot \mathbf{v}) - (\mathbf{a} \cdot \mathbf{v})(\mathbf{b} \cdot \mathbf{c}) = \big\langle (\mathbf{a} \cdot \mathbf{c})\,\mathbf{b} - (\mathbf{b} \cdot \mathbf{c})\,\mathbf{a},\, \mathbf{v} \big\rangle, \tag{A.20}$$

we get

$$N(\mathbf{v}) = \big\langle (\mathbf{v}_1 \cdot \mathbf{v}_3)\,\mathbf{v}_2 - (\mathbf{v}_2 \cdot \mathbf{v}_3)\,\mathbf{v}_1,\ \mathbf{v} \big\rangle, \quad D(\mathbf{v}) = \big\langle (\mathbf{v}_3 \cdot \mathbf{v}_1)\,\mathbf{v}_2 - (\mathbf{v}_2 \cdot \mathbf{v}_1)\,\mathbf{v}_3,\ \mathbf{v} \big\rangle,$$

hence both are linear functionals on U. Consequently, for any $\lambda \neq 0$, $N(\lambda\mathbf{v}) = \lambda N(\mathbf{v})$ and $D(\lambda\mathbf{v}) = \lambda D(\mathbf{v})$, so the homogeneous pair $[N(\mathbf{v}) : D(\mathbf{v})]$ depends only on $[\mathbf{v}] \in L$, not on the choice of representative.

Next, f does not depend on the particular homogeneous representatives of v_1, v_2, v_3: if we replace $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ by $(\alpha\mathbf{v}_1, \beta\mathbf{v}_2, \gamma\mathbf{v}_3)$ with $\alpha, \beta, \gamma \neq 0$, then N and D both get multiplied by the common nonzero factor $\alpha\beta\gamma$ (by bilinearity of $\times$ and $\langle \cdot, \cdot \rangle$), hence $[N : D]$ is unchanged.

Finally, the linear map

$$T : U \longrightarrow \mathbb{R}^2, \qquad \mathbf{v} \longmapsto \big(N(\mathbf{v}), D(\mathbf{v})\big)$$

has rank 2. Indeed,

$$T(\mathbf{v}_1) = \big(N(\mathbf{v}_1), D(\mathbf{v}_1)\big) = \big(\langle \mathbf{v}_1 \times \mathbf{v}_2,\ \mathbf{v}_3 \times \mathbf{v}_1 \rangle,\ 0\big),$$

and the first component is nonzero because $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are distinct and collinear, so $\mathbf{v}_1 \times \mathbf{v}_2$ and $\mathbf{v}_3 \times \mathbf{v}_1$ are nonzero parallel vectors in $U^\perp$. Similarly,

$$T(\mathbf{v}_3) = \big(0, \langle \mathbf{v}_2 \times \mathbf{v}_3,\ \mathbf{v}_3 \times \mathbf{v}_1 \rangle\big)$$

with the second component nonzero for the same reason. Thus $T(\mathbf{v}_1)$ and $T(\mathbf{v}_3)$ are linearly independent in $\mathbb{R}^2$, so $\operatorname{rank} T = 2$. In particular, $T(\mathbf{v}) \neq (0, 0)$ for every nonzero $\mathbf{v} \in U$.

Since T is invertible, it induces a projective-linear isomorphism

$$\mathbb{P}(T) : \ \mathbb{P}(U) \xrightarrow{\cong} \mathbb{P}(\mathbb{R}^2) \cong \mathbb{RP}^1,$$

and (A.19) is exactly this map. Hence f is bijective (in particular, injective).

Step 2: Relating f to the cross-ratio. By definition of the cross-ratio used in this section,

$$(v_1, v_2; v_3, v) \ = \ \frac{N(\mathbf{v})}{D(\mathbf{v})}$$

whenever $D(\mathbf{v}) \neq 0$. If $D(\mathbf{v}) = 0$ (which happens precisely at $v = v_1$), we interpret $(v_1, v_2; v_3, v) = \infty$, which corresponds to $f(v) = [1 : 0]$. Thus, the statement

$$(v_1, v_2; v_3, v_4) = (v_1, v_2; v_3, v_4')$$

is equivalent to

$$[\, N(\mathbf{v}_4) : D(\mathbf{v}_4) \,] = [\, N(\mathbf{v}_4') : D(\mathbf{v}_4') \,],$$

i.e., $f(v_4) = f(v_4')$ in $\mathbb{RP}^1$ (covering both the finite and the ∞ case uniformly).

Step 3: Conclusion. Since f is injective, $f(v_4) = f(v_4')$ implies $v_4 = v_4'$, as desired. □

Summary

We defined the cross-ratio on $\mathbb{RP}^2$ using homogeneous coordinates and the fixed inner-product identification between points and lines. We proved:

(i) Under the natural incidence correspondence, the cross-ratio of four collinear points equals the cross-ratio of the four concurrent lines through a fixed point (Prop. 2).

(ii) Given three fixed distinct collinear points, the cross-ratio determines the fourth point uniquely (Prop. 3).

Both results are standard manifestations of the projective invariance and the duality between points and lines in $\mathbb{RP}^2$.

A.2.2 Validity of The PureIntersectionPoint Theorem

The PureIntersectionPoint theorem

Recall from Definition 1 that a PureIntersectionPoint theorem is specified by a finite sequence of elementary constructions, each producing either a point or a line in $\mathbb{RP}^2$:

1. **Free object:** introduce a new free point or a new free line.
2. **Incidence choice:** choose a point on a previously constructed line, or choose a line through a previously constructed point.
3. **Derived object:**
   (a) intersect two distinct lines to create a new point; or
   (b) join two distinct points to create a new line (by point–line duality, we may also say the points "intersect").

Parametric viewpoint. A convenient way to formalize a PureIntersectionPoint statement is to regard it as a *parametrized construction*.

- A *free object* is specified by a parameter in the projective plane:

$$\text{"free point/line"} \quad \Longleftrightarrow \quad \mathbb{RP}^2 \to \mathbb{RP}^2, \;\; u \mapsto u,$$

  i.e., by the identity map on $\mathbb{RP}^2$.

- An *incidence choice* is specified by a parameter on a projective line, which is canonically isomorphic to $\mathbb{RP}^1$:

$$\text{"point on a given line"} \quad \Longleftrightarrow \quad \mathbb{RP}^1 \to \mathbb{RP}^2,$$

  and similarly for "line through a given point".[1]

Thus, if a statement uses n_2 free choices in $\mathbb{RP}^2$ and n_1 incidence choices in $\mathbb{RP}^1$, its *parameter space* is the product

$$\mathcal{P} \;=\; (\mathbb{RP}^2)^{n_2} \times (\mathbb{RP}^1)^{n_1}.$$

Derived objects via cross products. We work in homogeneous coordinates: represent points and lines by nonzero vectors in $\mathbb{R}^3$ modulo scaling, using a fixed identification of the dual so that incidence is captured by the Euclidean pairing $\langle \cdot, \cdot \rangle$. For nonzero vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$:

$$\text{join/meet} \quad = \quad \mathbf{a} \times \mathbf{b}.$$

Concretely, if $v = [\mathbf{v}]$ and $w = [\mathbf{w}]$ are distinct points, then the line vw is $[\mathbf{v} \times \mathbf{w}]$; if $\ell = [\boldsymbol{\ell}]$ and $m = [\boldsymbol{m}]$ are distinct lines, then $\ell \cap m = [\boldsymbol{\ell} \times \boldsymbol{m}]$. The cross product is bilinear and alternating in the chosen representatives, hence each derived construction is given by

[1] Fix any two distinct points on the line (or two distinct lines through the point) to obtain homogeneous coordinates $[s : t] \in \mathbb{RP}^1$. In any affine chart this is a linear—hence polynomial—parameterization.

homogeneous polynomial expressions in the parameters of previously constructed objects (polynomial on any affine chart, homogeneous polynomial in projective coordinates).

To make every construction step a *total* map (defined for all parameter values), we adjoin a cemetery element for degeneracies. Set

$$\overline{\mathbb{RP}}^2 \;:=\; \mathbb{RP}^2 \sqcup \{\texttt{NaN}\},$$

where `NaN` is a formal symbol corresponding to the zero vector $[0:0:0]$ (not a valid projective point/line). Extend the cross product by

$$u \times u = \texttt{NaN}, \qquad u \times \texttt{NaN} = \texttt{NaN} = \texttt{NaN} \times u, \qquad \forall\, u \in \overline{\mathbb{RP}}^2.$$

Thus, coincident arguments (e.g., "intersecting" a line with itself or "joining" a point to itself) produce `NaN` and propagate forward, while for nondegenerate inputs the usual projective cross product is recovered.

The global construction map. List the m objects produced by the statement in construction order. By composing the elementary maps step by step (free choices, incidence choices, and cross products), we obtain a well-defined total map

$$F :\; \mathcal{P} \longrightarrow (\overline{\mathbb{RP}}^2)^m,$$

whose j-th component records the j-th constructed point or line (with value `NaN` on degenerate parameter tuples). In homogeneous coordinates the components of F are given by homogeneous polynomial expressions in the parameters; in any affine chart they are polynomial maps. Consequently, a PureIntersectionPoint theorem may be viewed as a polynomially parametrized projective construction with a canonical totalization at degenerate loci.

Validity of the PureIntersectionPoint theorem

Since $\mathbb{RP}^2$ and $\mathbb{RP}^1$ carry their standard Hausdorff, second-countable topologies, the parameter space $\mathcal{P} = (\mathbb{RP}^2)^{n_2} \times (\mathbb{RP}^1)^{n_1}$ (cf. §A.2.2) is endowed with the product topology.

Definition 12 (PureIntersectionPoint theorem validity)**.** *A PureIntersectionPoint theorem is* valid *if there exists a dense open set $U \subset \mathcal{P}$ such that for every $u \in U$ the m-tuple $F(u)$ has no `NaN` components and the two designated goal objects $u_\star, v_\star \in \overline{\mathbb{RP}}^2$ constructed by F are incident, i.e. $\langle u_\star, v_\star \rangle = 0$.*

We show that, for a fixed non-degenerate, constructive theorem graph, validity does not depend on the chosen constructive realization. The proof explicitly avoids appeal to the Fundamental Theorem of Projective Geometry.

Theorem 5. *If a theorem graph (Definition 3) is non-degenerate (Definition 5) and constructive (Definition 6), then validity is invariant across constructive realizations: if one constructive realization is valid, then every constructive realization is valid.*

Proof. Let o_1, o_2 be two constructive realizations of the same non-degenerate theorem graph. Write $\mathcal{P}_i = (\mathbb{RP}^2)^{n_2^{(i)}} \times (\mathbb{RP}^1)^{n_1^{(i)}}$ for their parameter spaces, and $F_i : \mathcal{P}_i \to (\overline{\mathbb{RP}}^2)^m$ for the corresponding total construction maps (§A.2.2). Assume o_1 is valid.

The realization space. Let $\mathcal{A} \subset (\overline{\mathbb{RP}}^2)^m$ be the open locus where all m objects are genuine (projective) points/lines and pairwise distinct within their type (no `NaN`, no coincidence/parallelism). Let $\mathcal{S} \subset \mathcal{A}$ be the set of m-tuples that satisfy *all* primitive relations of the theorem graph: each "incidence choice" object lies on its parent line (or passes through its parent point), and each "derived" object equals the appropriate cross product of its parents in homogeneous coordinates. In affine charts these are polynomial equations/inequalities; hence $\mathcal{S}$ is a (semi-)algebraic subset of $\mathcal{A}$.

Set

$$\mathcal{S}^{\mathrm{nd}} := \mathcal{S} \qquad \text{(non-degenerate realization space)}.$$

By Definition 5, $\mathcal{S}^{\mathrm{nd}} \neq \varnothing$.

Step 1: Non-emptiness and surjectivity onto $\mathcal{S}^{\mathrm{nd}}$. For each realization o_i and for each $s \in \mathcal{S}^{\mathrm{nd}}$, there exists $u_i \in \mathcal{P}_i$ with $F_i(u_i) = s$. Indeed, fix s and build u_i along the construction order of o_i: *free* objects get their parameters equal to the corresponding entries of s; for an *incidence choice* (point on a known line, or line through a known point), read off its homogeneous $\mathbb{RP}^1$ coordinate from s (uniquely well-defined since s is non-degenerate); *derived* objects introduce no parameters. Thus the assignment produces u_i with $F_i(u_i) = s$. Consequently the non-degenerate parameter locus

$$\mathcal{P}_i^{\mathrm{nd}} := F_i^{-1}\big(\mathcal{S}^{\mathrm{nd}}\big)$$

is nonempty for each i, and $F_i\big(\mathcal{P}_i^{\mathrm{nd}}\big) = \mathcal{S}^{\mathrm{nd}}$.

Moreover, degeneracy conditions ("`NaN`", coincidences, parallels) are given by polynomial equalities in the parameters (determinants or minors vanishing); hence $\mathcal{P}_i^{\mathrm{nd}}$ is open and dense in $\mathcal{P}_i$.

Step 2: Local diffeomorphism (open mapping). Fix $u \in \mathcal{P}_i^{\mathrm{nd}}$ and write $s = F_i(u) \in \mathcal{S}^{\mathrm{nd}}$. Work in affine charts for $\mathbb{RP}^2$ and $\mathbb{RP}^1$ that contain the relevant objects. In those charts F_i is polynomial. Order the parameters according to the construction sequence and order the outputs likewise. The Jacobian of F_i at u has a block lower-triangular structure whose diagonal blocks are:

- for each *free* object: a 2×2 identity block (the created object equals its parameter);
- for each *incidence choice*: a 2×1 block given by a nonzero linear map $\mathbb{R} \to \mathbb{R}^2$ (the local homogeneous coordinate on the carrier line/pencil);
- for each *derived* object: no new parameter column.

Non-degeneracy ensures these diagonal blocks are full rank and occur at distinct output positions. Therefore $\operatorname{rank} D(F_i)(u) = k := 2V - E$ (the degree of freedom of the graph; cf. §A.2.2), and by the inverse function theorem there exist neighborhoods $V_u \subset \mathcal{P}_i^{\mathrm{nd}}$ and $W_s \subset \mathcal{S}^{\mathrm{nd}}$ such that

$$F_i\big|_{V_u} \;:\; V_u \xrightarrow{\cong} W_s$$

is a diffeomorphism. In particular, F_i is an *open map* at every non-degenerate point, and the family $\{W_s : s \in \mathcal{S}^{\mathrm{nd}}\}$ forms an open cover of $\mathcal{S}^{\mathrm{nd}}$ by coordinate charts coming from *every* realization o_i.

Step 3: From one valid realization to an open goal locus in $\mathcal{S}^{\mathrm{nd}}$. Let $U_1 \subset \mathcal{P}_1$ be a dense open set given by the validity of o_1 (Definition 12). Intersect with the open dense non-degenerate locus to get $U_1' := U_1 \cap \mathcal{P}_1^{\mathrm{nd}}$, which is open and dense in $\mathcal{P}_1$. Fix any $u \in U_1'$ and let $s := F_1(u) \in \mathcal{S}^{\mathrm{nd}}$. By Step 2 there is a chart $V_u \xrightarrow{\cong} W_s$ with $u \in V_u$. Since U_1' is open, $U_1' \cap V_u$ is a nonempty open subset of V_u, and its image $F_1(U_1' \cap V_u)$ is a nonempty *open* subset of W_s.

Define the *goal-incidence* function $G : \mathcal{S}^{\mathrm{nd}} \to \mathbb{R}$ by $G(x) := \langle u_\star(x), v_\star(x) \rangle$, where $u_\star(x), v_\star(x)$ are the two designated goal objects read off from the tuple x. In affine charts G is a polynomial (hence real-analytic) function of x. By validity of o_1, G vanishes on $F_1(U_1')$, and in particular on the open set $F_1(U_1' \cap V_u) \subset W_s$. Since zeros of a nontrivial real-analytic function have empty interior, it follows that

$$G \equiv 0 \quad \text{on } W_s.$$

Because the charts W_s with $s \in \mathcal{S}^{\mathrm{nd}}$ cover $\mathcal{S}^{\mathrm{nd}}$ (Step 2), we conclude

$$G \equiv 0 \quad \text{on the entire } \mathcal{S}^{\mathrm{nd}}. \tag{A.21}$$

Step 4: Transporting validity to any realization. For realization o_2, set

$$U_2 := \mathcal{P}_2^{\mathrm{nd}} = F_2^{-1}(\mathcal{S}^{\mathrm{nd}}).$$

As noted in Step 1, U_2 is open and dense in $\mathcal{P}_2$; moreover $F_2(U_2) = \mathcal{S}^{\mathrm{nd}}$. Combining this with (A.21) gives that for every $u_2 \in U_2$, the tuple $F_2(u_2)$ has no `NaN`, is non-degenerate, and satisfies the goal incidence $G\big(F_2(u_2)\big) = 0$. Thus o_2 is valid.

This proves that validity is invariant across constructive realizations. □

Failure of validity invariance without non-degeneracy

The hypothesis of Theorem 5 is necessary: the following pair of PureIntersectionPoint statements share the same theorem graph, yet only the first is valid while the second is not.

The key point is that, given a point p_0, if we draw two **distinct** lines l_1, l_2 through it, then their intersection $p_3 = l_1 \cap l_2$ is necessarily p_0 (the only degenerate case is $l_1 = l_2$). By contrast, if we change the construction order—first choose p_3 freely, then draw l_1 and l_2 as the lines through p_0 and p_3—the construction remains legal, but there is no longer any constraint forcing $p_3 = p_0$.

Counterexample — Realization 1 (valid)

Create $p52$
Create $p58$
From $p52$ draw $l264$
Create $p55$
Create $p54$
$p58$ and $p55$ intersect at $l260$
$l260$ and $l264$ intersect at $p56$
$p55$ and $p54$ intersect at $l258$
$p52$ and $p55$ intersect at $l259$
$p54$ and $p56$ intersect at $l257$
Create $l265$
$p52$ and $p54$ intersect at $l256$
From $p58$ draw $l261$
$l265$ and $l256$ intersect at $p51$
$l261$ and $l259$ intersect at $p57$
$p57$ and $p51$ intersect at $l253$
$l264$ and $l265$ intersect at $p63$
$l253$ and $l258$ intersect at $p50$
$l261$ and $l257$ intersect at $p62$
$l265$ and $l260$ intersect at $p64$
$p50$ and $p58$ intersect at $l254$
$p63$ and $p62$ intersect at $l263$
$l257$ and $l254$ intersect at $p53$
$l254$ and $l263$ intersect at $p61$
$p51$ and $p53$ intersect at $l255$
$l255$ and $l261$ intersect at $p60$
$p60$ and $p61$ intersect at $l262$
$l264$ and $l261$ intersect at $p65$
$p65$ and $p64$ intersect at $l267$
$l264$ and $l262$ intersect at $p66$
Goal: $l267$ is on $p66$

Counterexample — Realization 2 (invalid)

Create $p54$
From $p54$ draw $l256$
Create $p57$
From $p57$ draw $l259$
From $l256$ draw $p51$
From $l259$ draw $p55$
Create $p56$
$p54$ and $p55$ intersect at $l258$
$l259$ and $l256$ intersect at $p52$
$p55$ and $p56$ intersect at $l260$
$p56$ and $p54$ intersect at $l257$
From $l257$ draw $p53$
$p51$ and $p57$ intersect at $l253$
$l253$ and $l258$ intersect at $p50$
$p53$ and $p50$ intersect at $l254$
$l254$ and $l260$ intersect at $p58$
$p52$ and $p56$ intersect at $l264$
$p53$ and $p51$ intersect at $l255$
From $l264$ draw $p66$
$p57$ and $p58$ intersect at $l261$
$l255$ and $l261$ intersect at $p60$
$p60$ and $p66$ intersect at $l262$
$l254$ and $l262$ intersect at $p61$
$l257$ and $l261$ intersect at $p62$
$p61$ and $p62$ intersect at $l263$
$l263$ and $l264$ intersect at $p63$
$p51$ and $p63$ intersect at $l265$
$l265$ and $l260$ intersect at $p64$
$l261$ and $l264$ intersect at $p65$
$p64$ and $p65$ intersect at $l267$
Goal: $l267$ is on $p66$

A.2.3 Equivalence between Harmonic Cross-Ratios

In this section, we use Propositions 2 and 3 to show that any two harmonic cross-ratios are equivalent.

We start with the following lemma.

Lemma 6. *Suppose* $p0, p1, p2, p3$ *are four distinct collinear points and* $p0, p4, p5, p6$ *are four distinct collinear points. Assume they lie on lines* $l0$ *and* $l1$*, respectively. If the cross-ratio* $(p0, p1, p2, p3)$ *is equivalent to* $(p0, p3, p2, p1)$ *and the cross-ratio* $(p0, p4, p5, p6)$ *is equivalent to* $(p0, p6, p5, p4)$*, then* $(p0, p1, p2, p3)$ *is equivalent to* $(p0, p4, p5, p6)$*.*

The proof is provided in our tokenized format.

Equivalence Proof

Answer:
merge_by_cr p2 [l0 [[[p1 p4] l2 [p3 p6] l3] p7 [l1 [p2 [[p1 p6] l4 [p3 p4] l5] p8] l6] p9] l7] p10
– l0 (p0 p1 p2 p3) !!= l0 (p0 p1 p10 p3)
– these_two_crs_are_equal l0 (p0 p1 p2 p3) ?= l0 (p0 p1 p10 p3)
— l0 (p0 p1 p2 p3) = l0 (p0 p3 p2 p1)
— l0 (p0 p3 p2 p1) = p8 ([p0 p8] l8 l5 l6 l4)
— p8 (l8 l5 l6 l4) = l1 (p0 p4 p9 p6)
— l1 (p0 p4 p9 p6) = p7 ([p0 p7] l9 l2 l7 l3)
— p7 (l9 l2 l7 l3) = l0 (p0 p1 p10 p3)
merge_by_def l6 l7
– they_share_those_two_neighbours p2 p9
merge_by_cr p5 [l1 [p7 [l0 [p5 p8] l10] p11] l11] p12
– l1 (p0 p4 p5 p6) !!= l1 (p0 p4 p12 p6)
– these_two_crs_are_equal l1 (p0 p4 p5 p6) ?= l1 (p0 p4 p12 p6)
— l1 (p0 p4 p5 p6) = l1 (p0 p6 p5 p4)
— l1 (p0 p6 p5 p4) = p8 (l8 l4 l10 l5)
— p8 (l8 l4 l10 l5) = l0 (p0 p1 p11 p3)
— l0 (p0 p1 p11 p3) = p7 (l9 l2 l11 l3)
— p7 (l9 l2 l11 l3) = l1 (p0 p4 p12 p6)
merge_by_def l10 l11
– they_share_those_two_neighbours p5 p11
merge_by_def l6 l10
– they_share_those_two_neighbours p7 p8

Finally we can prove the equivalence
– l0 (p0 p1 p2 p3) !!= l1 (p0 p4 p5 p6)
– these_two_crs_are_equal l0 (p0 p1 p2 p3) ?= l1 (p0 p4 p5 p6)
— l0 (p0 p1 p2 p3) = p7 (l9 l2 l6 l3)
— p7 (l9 l2 l6 l3) = l1 (p0 p4 p5 p6)

With this lemma, one can prove the equivalence of any two harmonic cross-ratios by introducing a third cross-ratio that *shares* one object with each of the two cross-ratios, and then applying the lemma twice; the proof follows.