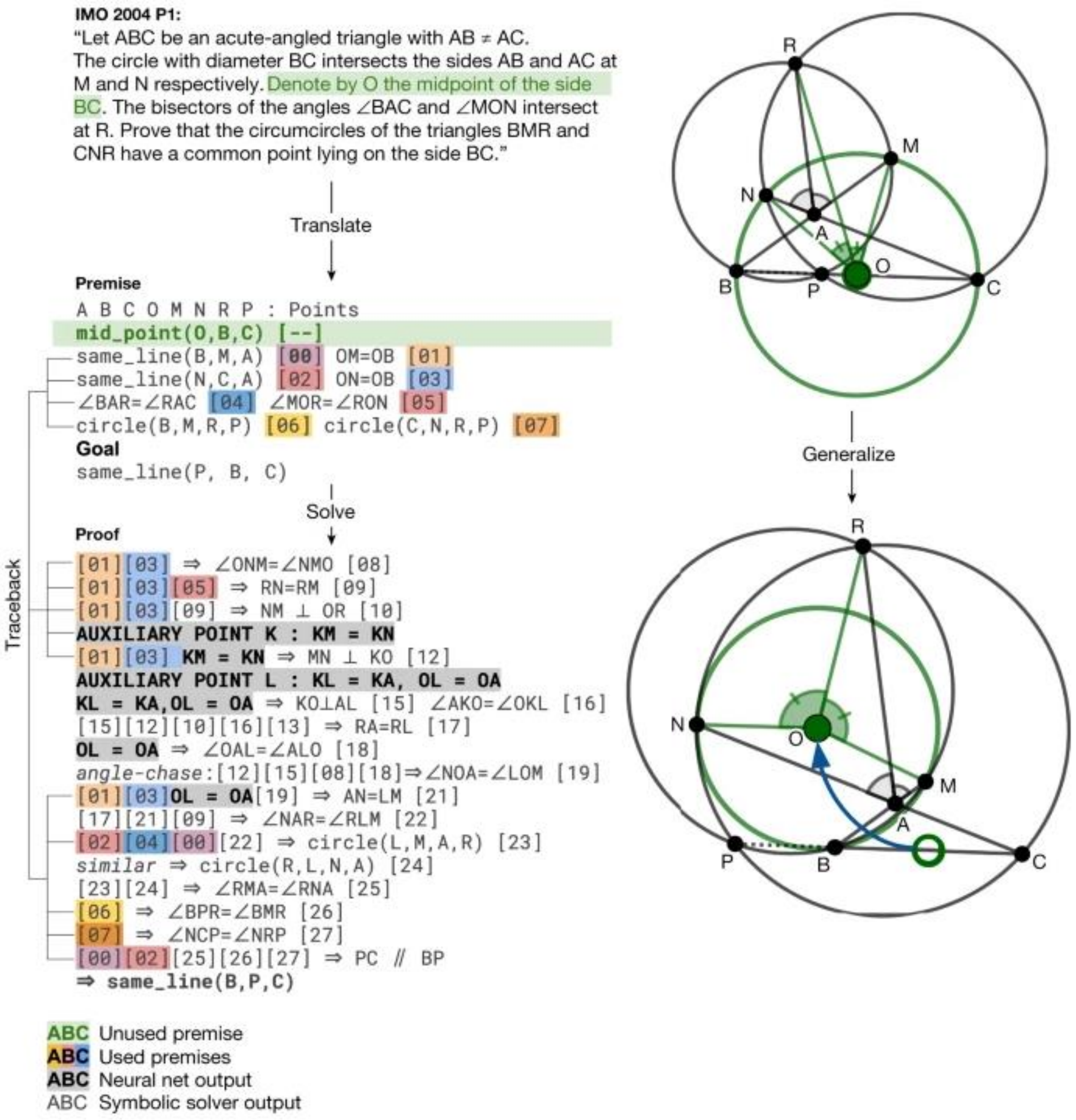
AlphaGeometry: Solving olympiad geometry without human demonstrations

February 8, 2024

AlphaGeometry: Solving olympiad geometry without human demonstrations

Trieu H. Trinh et al., Google Deepmind

https://www.nature.com/articles/s41586-023-06747-5



AlphaGeometry overview

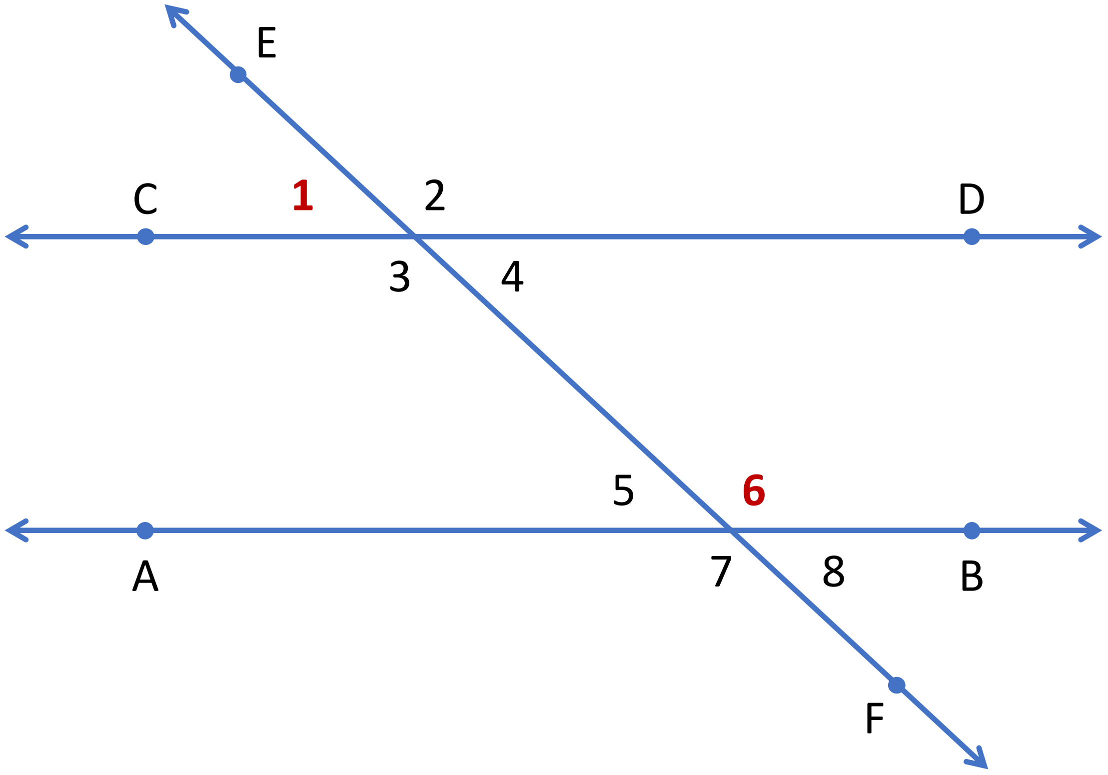
- Solves Euclidean plane geometry proofs from International Mathematical Olympiad contest problems, achieving SOTA
- Combines off the shelf symbolic deduction engine with an LLM
  - Symbolic engine expands list of facts by matching thousands of theorem premises with existing facts, calculating the closure
  - LLM suggests new objects to construct, allowing further deduction
  - The space of auxiliary construction is huge, so LLM must reason well
- They solved the training data problem by generating synthetic proofs
- LLM was pretrained on full proofs, then fine-tuned on subset requiring auxiliary construction

Symbolic deduction engine

- Symbolic deduction engine has a database of geometric rules
  - They added algebraic rules
- Rules are Horn clauses of the form $Q(x) \leftarrow P_1(x_1), P_2(x_2), \ldots, P_k(x_k)$
  - Each P_i or Q is a predicate such as “equal segments” or “collinear”
  - Each x is a set of one or more point objects
- If all of the needed P’s are matched, then Q(x) is added to the list of known facts
- The symbolic engine alone will solve simpler proofs that don’t require constructing additional objects, but it won’t construct new objects
- Existing techniques and code generally ran in seconds

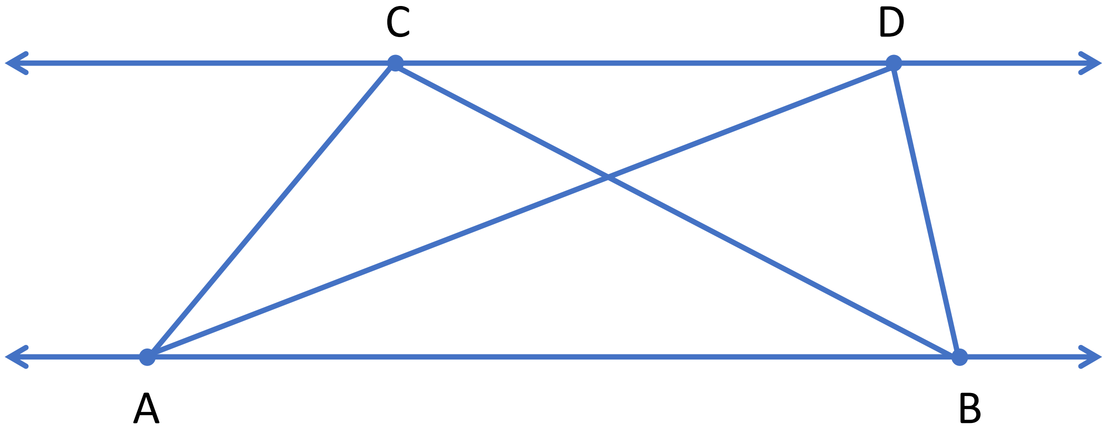
Deduction example

- We want to prove that angle 1 and angle 6 are supplementary, given parallel lines AB and CD
- First, symbolic engine might prove that angle 1 equals angles 4, 5, and 8
- Then, it might prove that angle 6 and angle 8 are supplementary
- Combining the above two, we have shown that angle 1 and angle 6 are supplementary



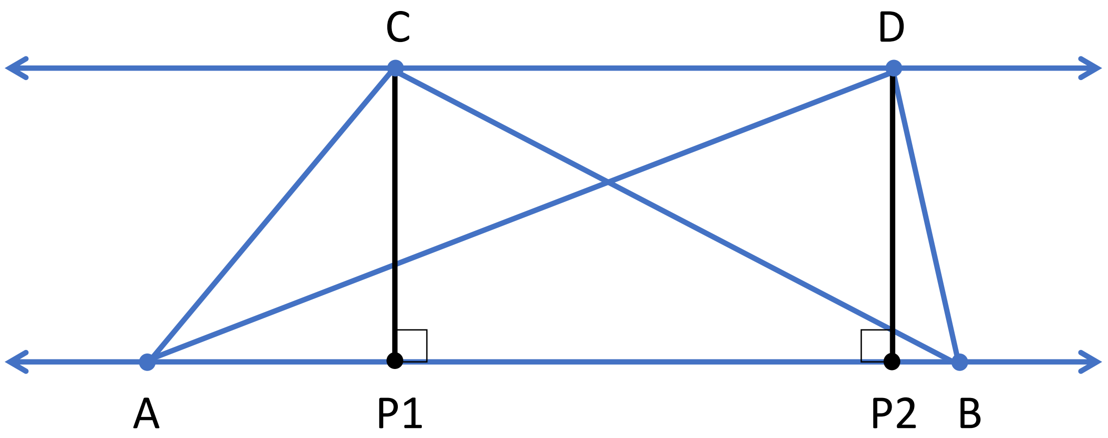
Deduction insufficiency example [1]

- We want to prove that triangle ABC and triangle ABD have the same area, given parallel lines AB and CD



Deduction insufficiency example [2]

- We want to prove that triangle ABC and triangle ABD have the same area, given parallel lines AB and CD
- Adding two points P1 and P2 allows us to show the heights of both triangles are the same
- Since both share the same base, the areas are the same



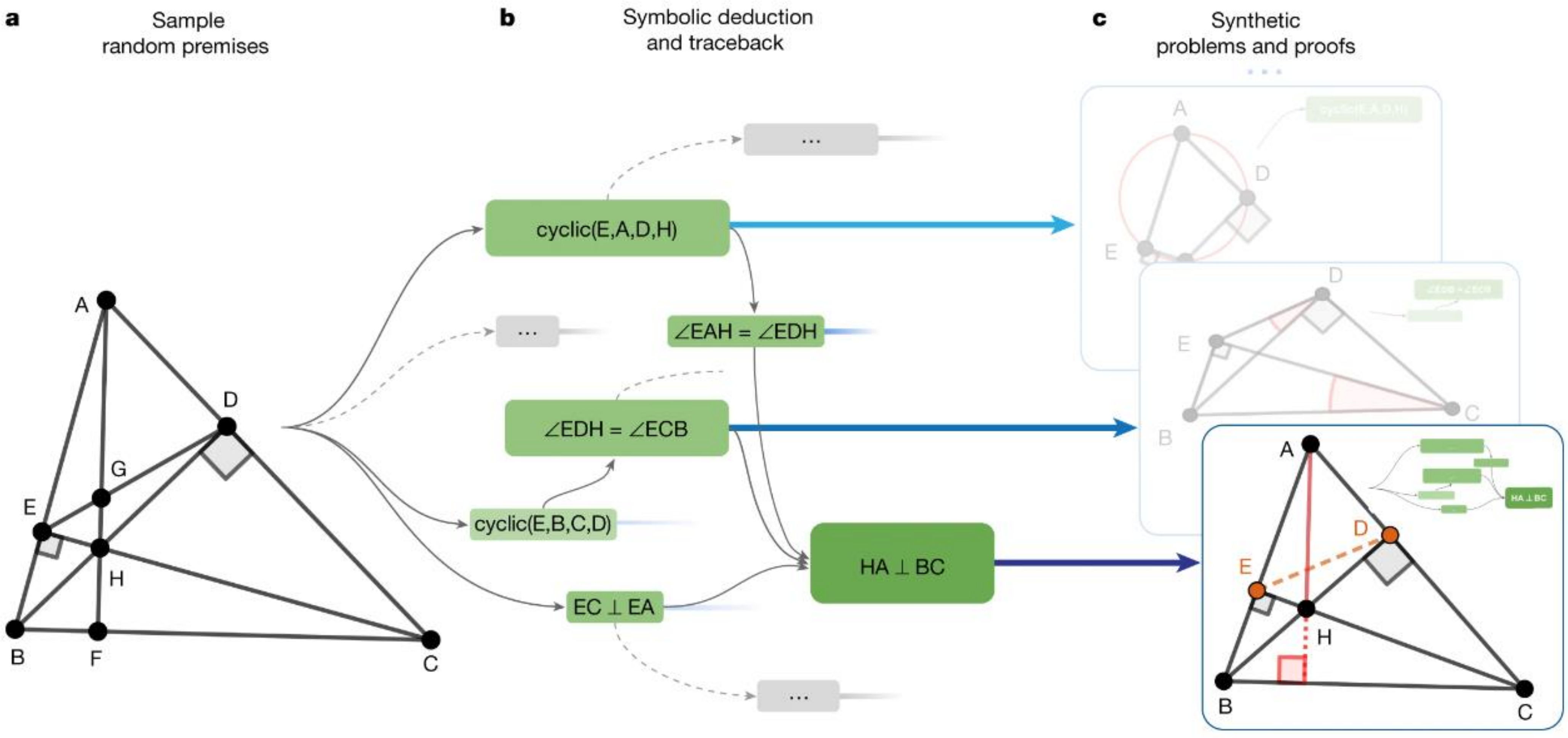
Training data

- Arguably the biggest problem was needing a very large set of training data
- Very few geometry proofs available to train on
  - Geometry can be difficult to translate into general purpose mathematical languages such as Lean
  - Geometry-specific languages exist, but each can only express a subset of concepts needed for proofs
- Decided against using any human proofs
- Instead, created their own set of synthetic theorems and proofs

Synthetic data

- Start by sampling random theorem premises
- Use symbolic deduction engine to expand to deduction closure
  - Track each deduction step, forming a DAG
- For each node N, trace backward along DAG for minimal set of premises P and deductions in a subgraph G(N)
- This forms a training example:
  - Premises P
  - Conclusion N
  - Proof G(N)

Synthetic data process



Synthetic auxiliary construction

- Start with a proof:
  - Premises P
  - Conclusion N
  - Proof G(N)
- The subset of P that N is independent of can be removed from P and added to the proof as auxiliary construction steps
  - Consider our points P1 and P2 from the triangle area example
- We now have harder proof(s) that require auxiliary construction
  - Footnote: they perform exhaustive testing with all subsets of auxiliary points to ensure all of them are really needed

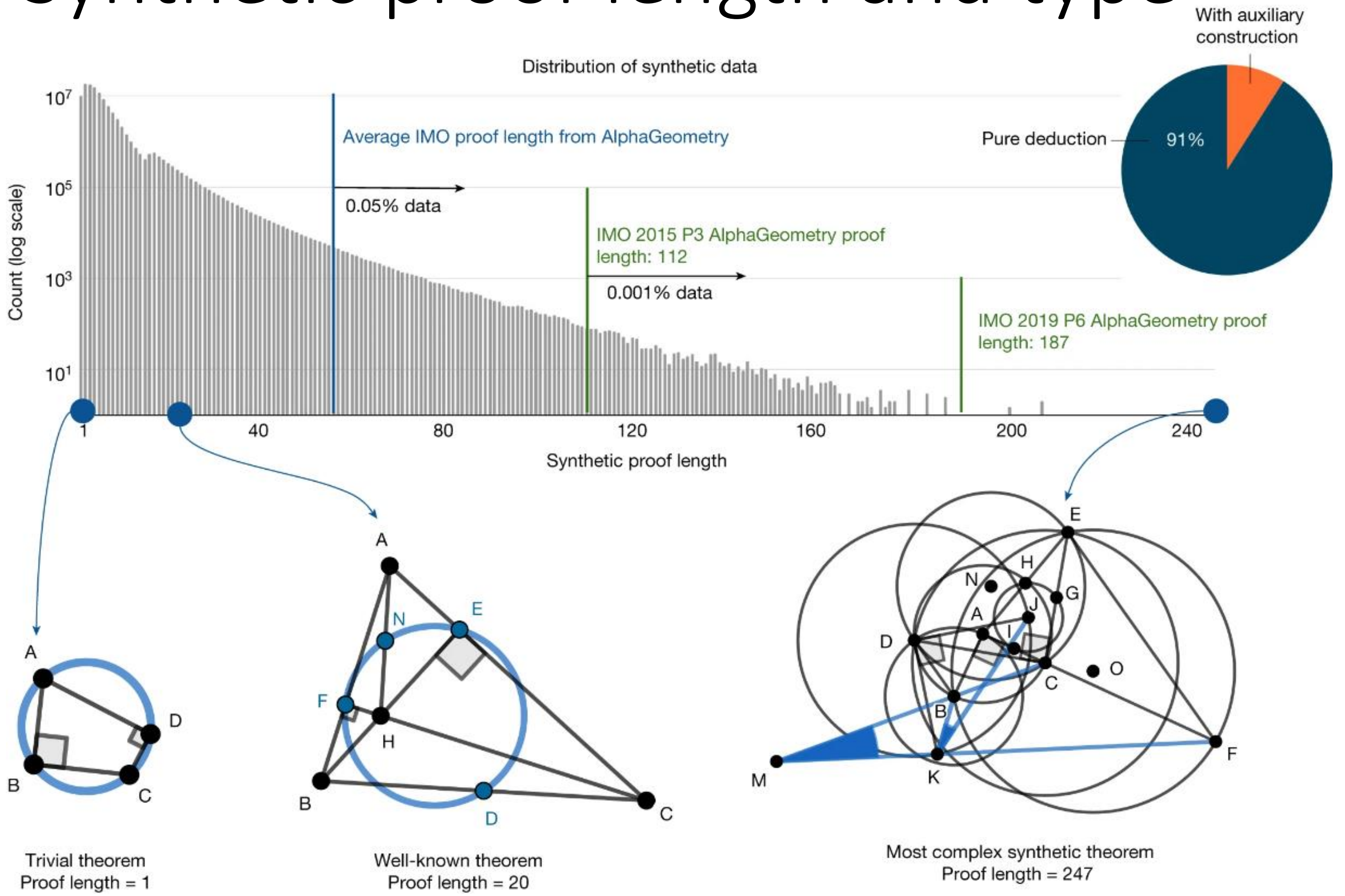
List of auxiliary construction forms [1]

Construction	Description
X = angle bisector(A, B, C)	Construct a point X on the angle bisector of ∠ABC
X = angle mirror(A, B, C)	Construct a point X such that BC is the bisector of ∠ABX
X = circle(A, B, C)	Construct point X as the circumcenter of A, B, C
A, B, C, D = eq_quadrilateral()	Construct quadrilateral ABCD with AD = BC
A, B, C, D = eq_trapezoid()	Construct trapezoid ABCD with AD = BC
X = eqtriangle(B, C)	Construct X such that XBC is an equilateral triangle
X = eqangle2(A, B, C)	Construct X such that ∠BAX = ∠XCB
A,B,C,D = eqdia_equadrilateral()	Construct quadrilateral ABCD with AC = BD
X = eqdistance(A, B, C)	Construct X such that XA = BC
X = foot(A, B, C)	Construct X as the foot of A on BC
X = free	Construct a free point X
X = incenter(A, B, C)	Construct X as the incenter of ABC
X,Y,Z,I = incenter2(A, B, C)	Construct I as the incenter of ABC with touchpoints X, Y, Z
X = excenter(A, B, C)	Construct X as the excenter of ABC
X,Y,Z,I = excenter2(A,B,C)	Construct X as the excenter of ABC with touchpoints X,Y,Z
X = centroid(A,B,C)	Construct X as the centroid of ABC
X,Y,Z,I = midpointcircle(A,B,C)	Construct X, Y, Z as the midpoints of triangle ABC, and I as the circumcenter of XYZ
A,B,C = isos()	Construct A, B, C such that AB = AC
X = tangent(O, A)	Construct X such that OA is perpendicular to AX
X = midpoint(A, B)	Construct X as the midpoint of AB
X = mirror(A, B)	Construct X such that B is the midpoint of AX
X = rotate90(A, B)	Construct X such that AXB is a right isosceles triangle
X = on_aline(A, B, C, D, E)	Construct X such that ∠XAB = ∠CDE
X = on_bline(X, A, B)	Construct X on the perpendicular bisector of AB
X = on_circle(O, A)	Construct X such that OA = OX
X = on_line(A, B)	Construct X on line AB
X = on_pline(A, B, C)	Construct X such that XA is parallel to BC
X = on_tline(A, B, C)	Construct X such that XA is perpendicular to BC
X = orthocenter(A, B, C)	Construct X as the orthocenter of ABC

List of auxiliary construction forms [2]

Construction	Description
X = orthocenter(A, B, C)	Construct X as the orthocenter of ABC
X = parallelogram(A, B, C)	Construct X such that ABCX is a parallelogram
A, B, C, D, E = pentagon()	Construct pentagon ABCDE
A, B, C, D = quadrilateral()	Construct quadrilateral ABCD
A, B, C, D = trapezoid()	Construct right trapezoid ABCD
A, B, C = r_triangle()	Construct right triangle ABC
A, B, C, D = rectangle()	Construct rectangle ABCD
X = reflect(A, B, C)	Construct X as the reflection of A about BC
A, B, C = risos()	Construct right isosceles triangle ABC
X = angle(A, B, α)	Construct X such that $\angle ABX = \alpha$
A, B = segment()	Construct two distinct points A, B
X = shift(B, C, D)	Construct point X such that XB=CD and XC=BD
X Y = square(A, B)	Construct X, Y such that XYAB is a square
A, B, C, D = init_square()	Construct square ABCD
A, B, C, D = trapezoid()	Construct trapezoid ABCD
A, B, C = triangle()	Construct triangle ABC
A, B, C = triangle12()	Construct trianglel ABC with AB:AC = 1:2
X,Y,Z,I = 2L1C(A, B, C, O)	Construct circle center I that touches line AC and line BC and circle (O, A) at X, Y, Z
X, Y, Z = 3PEQ(A, B, C)	Construct X, Y, Z on three sides of triangle ABC such that Y is the midpoint of XZ
X, Y = trisect(A, B, C)	Construct X, Y on AC such that BX and BY trisect $\angle ABC$
X, Y = trisegment(A, B)	Construct X, Y on segment AB such that AX=XY=YB
X = on_dia(A, B)	Construct point X such that AX is perpendicular to BX
A, B, C = ieqtriangle()	Construct equilateral triangle ABC
X, Y, Z, T = cc_tangent(O, A, W, B)	Construct common tangents of circles (O, A) and (W, B) with touchpoints X, Y for one tangent and Z, T for the other.
X = eqangle3(A, B, D, E, F)	Construct point X such that $\angle AXB = \angle EDF$
X, Y = tangent(A, O, B)	Construct points X, Y as the tangent touch points from A to circle (O, B)
X = intersect(f, g)	Construct point X as the intersection of two functions f() and g(), where f() and g() is any of the above functions that returns more than one possible construction.

Synthetic proof length and type



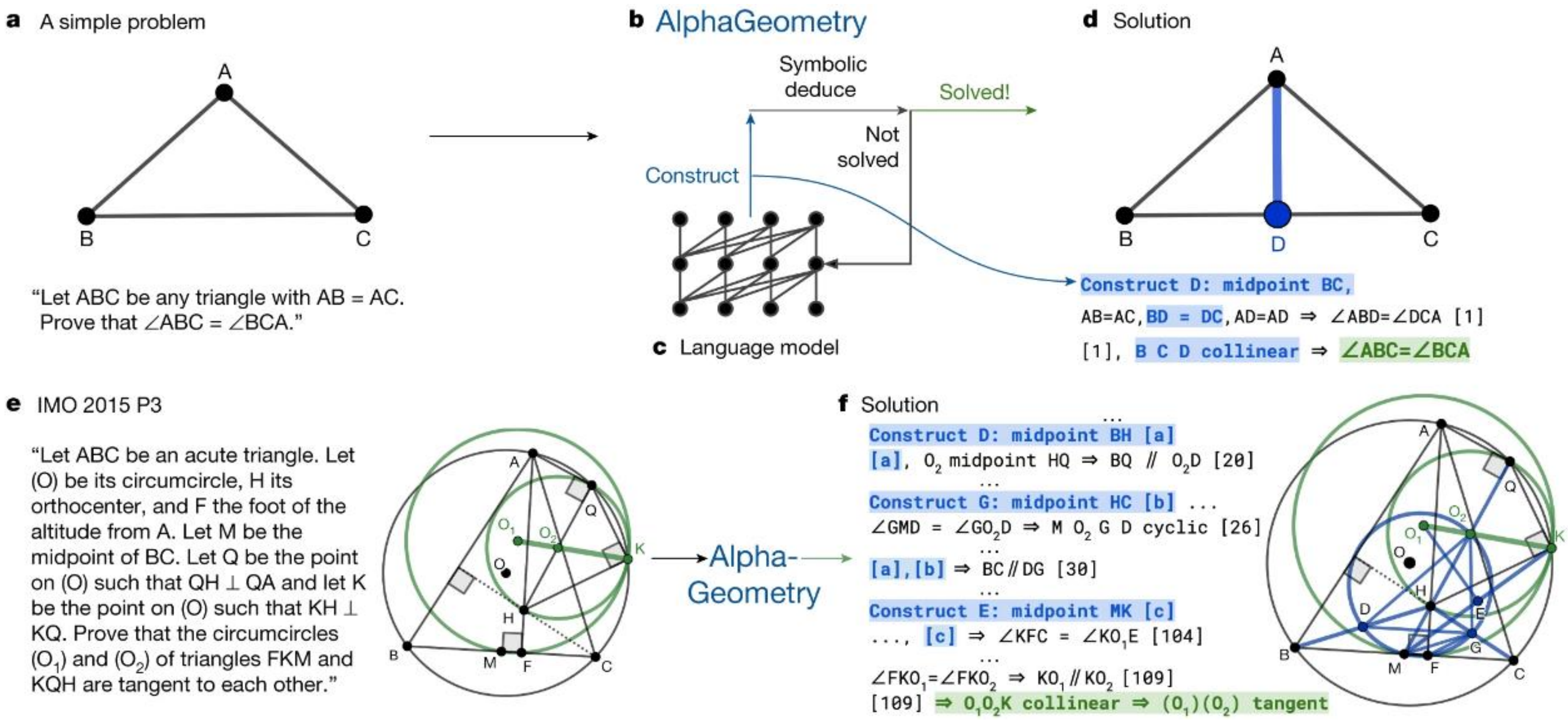
LLM and Training

- GPT style decoder only LLM
  - 12 layers, model dimension 1024, 8 attention heads, 4x MLP expansion, ReLU
  - Max sequence length 1024
  - 151 million parameters
  - T5-style relative position embeddings
  - Custom tokenizer, vocab size 757
- Convert each proof into a text string of the form "<premises><conclusion><proof>"
- Pretraining usual next token prediction with cross-entropy loss
  - Over 100 million synthetic proofs in training set
- Fine-tune on just the 9 million proofs requiring auxiliary construction

Proof Solving with LLM Inference

- Alternate between the symbolic engine and LLM
- Start the LLM with the prompt "<premises><conclusion>" and have it generate one auxiliary construction such as "construct point X so that ABCX is a parallelogram"
  - Use beam search keeping top *k* completions (default was *k*=512, max depth=16)
- Each symbolic engine turn, expand the deduction closure for all *k* new constructions
  - If the conclusion is reached, the proof is complete
- On subsequent LLM turns, append prior constructions to the prompt and have the LLM generate one additional auxiliary construction
  - Note that the LLM doesn't get to see the symbolic engine's deduction closure

Proof solving workflow



Results

- 30 IMO proofs since 2000 were this kind of geometry
- AlphaGeometry beat existing computer algebra and search methods, including their attempts to boost DD and GPT-4 with techniques used here

Method		**Problems solved (out of 30)**
Computer algebra	Wu’s method (previous state of the art)	10
	Gröbner basis	4
Search (human-like)	GPT-4	0
	Full-angle method	2
	Deductive database (DD)	7
	DD + human-designed heuristics	9
	DD + AR (ours)	14
	DD + AR + GPT-4 auxiliary constructions	15
	DD + AR + human-designed heuristics	18
	AlphaGeometry	**25**
	• Without pretraining	21
	• Without fine-tuning	23
	• Only 20% of training data	21
	• Beam search *k*=8	21

AlphaGeometry conclusion

- AlphaGeometry solves Euclidean plane geometry problems better than any computer baseline, and at the level of top IMO competitors
    - AlphaGeometry solved 25/30 and silver medalist avg. 22.9; gold avg. 25.9
- Data-scarcity was solved by synthetically generating training data
    - Key factor was generating proofs with auxiliary construction
- Final solution uses symbolic engine for deduction and LLM for suggestions for auxiliary construction
    - They call this combination a “neuro-symbolic” system
- These search-style proofs can be made human-readable, unlike the computer algebra techniques which just output True

References

- AlphaGeometry blog post: https://deepmind.google/discover/blog/alphageometry-an-olympiad-level-ai-system-for-geometry/
- AlphaGeometry GitHub repo: https://github.com/google-deepmind/alphageometry
- Generative language modeling for automated theorem proving
  Polu, S. and Sutskever, I. (2020)
  https://arxiv.org/abs/2009.03393
- Llemma: An Open Language Model For Mathematics
  Azerbayev, Z. et al. (2023)
  https://arxiv.org/abs/2310.10631

References [2]

- AlphaGeometry supplementary material with GPT-4 prompts and solutions to IMO proofs: https://static-content.springer.com/esm/art%3A10.1038%2Fs41586-023-06747-5/MediaObjects/41586_2023_6747_MOESM1_ESM.pdf