L-DAG: An Algorithm for Deductive Reasoning in AI on Complex Logical Problem[*]

Wendy W. Xi[†]
Independent
xiw_98@yahoo.com

Abstract
Current LLMs primarily reason either by searching or heuristically mapping solution steps (*solution path space*) or by directly producing solutions through statistical mappings in their neural weights (*solution space*), followed by retroactive rationalization. Unlike these approaches, L-DAG (Logical Directed Acyclic Graph) reasons entirely within the *constraint space*. It dynamically constructs a solution path through an interactive iteration between deducting constraint search directions and searching for constraints; each cycle progressively restricts the solution space and converges efficiently to a solution. It is an efficient algorithm to solve DAG-structured complex deductive reasoning tasks. Its priority strategies and timely self-reward mechanism guide the constraint search process, while its topological order enforces dynamic construction under Global Dependency Management for tracking logical dependencies and consistency, and enables parallel processing. Two logical examples in the paper were tested on the leading AI systems. None of the tested systems produced a complete, correct solution using direct reasoning, Python, or MiniZinc.[2] A complex logical proposition proof (6 variables, 10 premises, 64-row truth table) demonstrated L-DAG's capacity for systematic exploration, clear derivation steps, and rapid convergence on a solution. L-DAG also provides conflict resolution support and integrative deductive reasoning to help humans discover novel insights and potential solution pathways.

1 Introduction and Motivation

Large Language Models (LLMs) excel at creative and associative reasoning but often struggle with strict, multi-step deductive logic. The L-DAG algorithm is introduced as a specialized process to address complex logical reasoning tasks.

Can World Models Lead to Deductive Reasoning and AGI?

Yann LeCun (2022) has actively promoted the idea of AI systems learning "world models". But can such models foster AI to develop deductive reasoning? Humans have lived within a world model since birth. Over hundreds of thousands of years, this model taught us basic causal reasoning

[†] Independent researcher, retired IT engineer, wusanxi@gmail.com.
[2] The last full testing on the following models was conducted on July 19, 2025. "x" indicates an incorrect solution, and "failed" means the attempt could not compile or run after multiple tries.

LLM (version)	**Example 2**			**Example 3 (3 Solutions)**		
	Reasoning	**Python**	**MiniZinc**	**Reasoning**	**Python**	**MiniZinc**
Gemini Pro 2.5 (2025-06-05)	x	x	failed	1	1	1
ChatGPT 4o** (2025-04-16)	x	x	failed	1	1	failed
DeepSeek r1 (2025-05-28)	x	x	x	1	2	1
Claude Sonnet 4 (2025-05-22)	x	x	x	x	1	1
Grok 3 (2025-02-17)	x	x	failed	x	x	1
LLaMA 4 (2025-04-05)	x	x	failed	x	x	x

* Example 1 is a complex logical proposition proof (6 variables, 10 premises, 64-row truth table).

essential for survival, such as tool-making, hunting, and avoiding danger. However, deductive reasoning (e.g., syllogisms) emerged only a few thousand years ago, allowing humans to uncover the principles governing the world. The later development of systematic logical thinking led to a profound re-interpretation of the same world model, catalyzing the Scientific Revolution (16th–18th centuries) and an unprecedented acceleration of discovery.

Cost Inefficiency: Combinatorial Trial and Error

Current AI systems incorporate Chain-of-Thought prompting, heuristic search, and pattern matching strategies to solve complex reasoning tasks. Although the approach is not purely brute-force, in nature it still relies on cost-inefficient combinatorial trial-and-error pathways and retrieval-based reasoning patterns, rather than systematic deduction,[3] and the reasoning process may not follow the systematic efficiency observed in human logical processing.[4] As Barez *et al.* (2025) note, "CoT often acts as a post-hoc rationalization rather than a transparent reflection of actual model processes." Many real-world problems are not linear chains but Directed Acyclic Graphs (DAGs).

Current Deduct Reasoning Task Solution Generation approaches

The primary approaches are summarized below.

#	Category	Description	Examples
1	Solution Space Search	Exhaustively searches a space of possible variable assignments to find one that satisfies all premises.	SAT Solvers (e.g., MiniSat, Glucose), Brute-force search algorithms
2	Solution Path Search	Incrementally constructs a reasoning pathway connecting premises to a conclusion, often using heuristics and pattern matching (e.g., forward/backward chaining).	LLM Prompting (Chain-of-Thought, Tree-of-Thought), Heuristic search planners (A*, STRIPS)
3	Constraint & Logic Systems	Systematically applies logical rules or pre-specified constraints to reduce the solution space, often without building an explicit search path.	Constraint Solvers: MiniZinc, Gecode Proof Assistants: Lean, Isabelle/HOL Logic Programming: Prolog

Interpretable and Controllable AI

Current LLMs are uninterpretable and uncontrollable. Moreover, the higher the intelligence of a black-box system, the more capable it is of deceiving humans. No one can guarantee that black-box AI systems can be aligned with human values before they reach uncontrollable levels.

L-DAG

[3] 2024 IMO, in 3 days, AlphaGeometry solved a geometry problem in 19 seconds, and AlphaProof solved 3 problems, while the two combinatorics problems remained unsolved. Under IMO rules, contestants have two 4.5-hour sessions, totaling 9 hours to solve the 6 problems (DeepMind AlphaProof and AlphaGeometry Teams, 2024).

[4] Trinh *et al*. (2024) proposed: "overall, we found that AlphaGeometry uses a much lower proof toolkit than humans, which limits the coverage of synthetic data, test time performance, and readability of proofs".

The paper introduces a new algorithm to enhance AI models' deductive and probabilistic reasoning abilities for solving complex problems. Unlike traditional approaches that reason in solution space or solution path space, L-DAG reasons entirely within the constraint space.
L-DAG is a dynamic solution path construction process driven by two iterative interactive steps:
Process i: reasoning about constraint search directions based on the currently active constraints, and
Process ii: applying new constraints that further narrow the scope of Process i.
Each cycle progressively restricts the solution space and incrementally builds an explicit, dynamically convergent solution path with high efficiency.

The logical directed acyclic graph in Figure 1 illustrates the scale and structural complexity of the target problem. It comprises 61 nodes and 89 deductive steps, with the longest reasoning chain spanning 17 steps. Despite this complexity, the problem is solvable through the sequential application of basic logical operations, as detailed in an introductory example in Section 2.3.

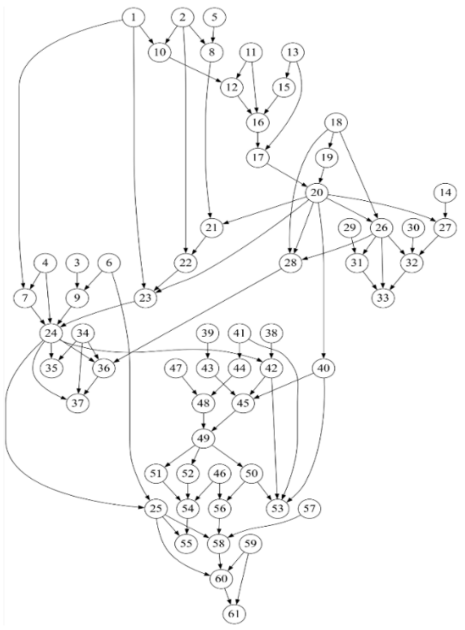


Figure 1 | L-DAG for Example 2 (the combination of Figure 6a and Figure 6b in Section 3.2)

2 Introduction to L-DAG

2.1 L-DAG Overview

- **Structure: Directed Acyclic Graph (DAG)** (Cormen et al., 2022):
  - **Directed**: The connections between elements follow a specific direction.
  - **Acyclic**: The graph contains no cycles or loops.

A DAG is a static structured representation of tasks and dependencies, although dynamic modifications, such as conditional branching or task creation, may occur during execution.

L-DAG is a dynamically constructed DAG through a stepwise logical process. Its structure guarantees logical consistency, based on two fundamental properties: directionality and acyclicity.

- **Core Element:**
    - **Node:** Represents Entities, their attributes, and the constraints or associations applied to them. There are two basic types of nodes: solution nodes and constraint nodes.
      Constraint nodes comprise:
        - all given premises, conditions, clues, and facts defined by the problem statement;
        - possibilities introduced during the reasoning process or initial assigned hypothetical values.

      An intermediate partial solution acts simultaneously as a solution node in the current context and as a constraint node for its descendant nodes.

    - **Edge:** Represents the relation between nodes
        - **Strong relation**: Parent-child logical dependencies.
        - **Weak relation**: Association.

    - **Task types:**
        - **Entity-Constraint-Task (ECT)**: Create a DAG for nodes that have constraint relations.
        - **Entity-Association-Task (EAT)**: Create an Undirected Graph (UG) for nodes that have association relations.

Throughout this document, the terms, *possibilities*, and *facts* are all considered constraints to a problem.

2.2 The Construction Process: Win-Rate and Search Prioritization
The L-DAG dynamic construction process is guided by Search Prioritization Strategies to progressively increase the Win-Rate.

- **Win-Rate** = k **DCD/P**
    - **Data Correlation Density (DCD)**: Higher data density provides more cross-reference information. In this paper, DCD is defined as the frequency with which an entity appears in the task domain, though it can be defined more generally or specifically for different tasks.
    - **Possibility (P):** The set of possibilities that satisfy all imposed constraints.
    - **Win-Rate is inversely related to the possibility and positively related to DCD.** However, these two factors may carry different weights. A coefficient *k* can be introduced to reflect their relative importance across different tasks. The nuanced application of this prioritization strategy is further discussed in the remarks on the L-DAG Construction Table in Section 3.2.4.

Remark
- The Win-Rate changes dynamically throughout the construction process because both Data Correlation Density and the number of Possibilities is updated.
- The Win-Rate can be pre-estimated before processing a node.
- Construction Process – A constraint-driven, iterative process: searching for and adding constraint nodes; generating possibility nodes based on those constraints; eliminating invalid

possibilities through logical operations; and adding intermediate solution nodes when all parent-node constraints are satisfied. This cycle repeats until a final solution is reached.

2.3 Introductory Examples

In this paper, L-DAG uses a topological sequence number (TID) to represent the processing order and dependencies between nodes. Consecutive numbering for TIDs is not required. The topological order only requires that the TID of a child node be greater than that of its parent because the parent node is constructed before its children. TIDs are indicated by bold numbers at the beginning of each node in L-DAG tables and figures. Possibility nodes in all figures are in dotted frame.

Example A: Logical Grid Puzzle

(1) L-DAG Dynamic Construction Process

Three men (German, English, and French) live in three houses arranged in a row (#1, #2, and #3). Each house has a unique color (green, blue, or yellow), and each man drinks a unique beverage (tea, water, or beer). The clues are as follows:

1) The German occupies house #1.
2) The Englishman drinks tea.
3) The water drinker is not associated with the green house.
4) The blue house is to the left of the yellow house.
5) The Frenchman's house is not adjacent to the beer drinker's house.

Question: Who lives in yellow house?

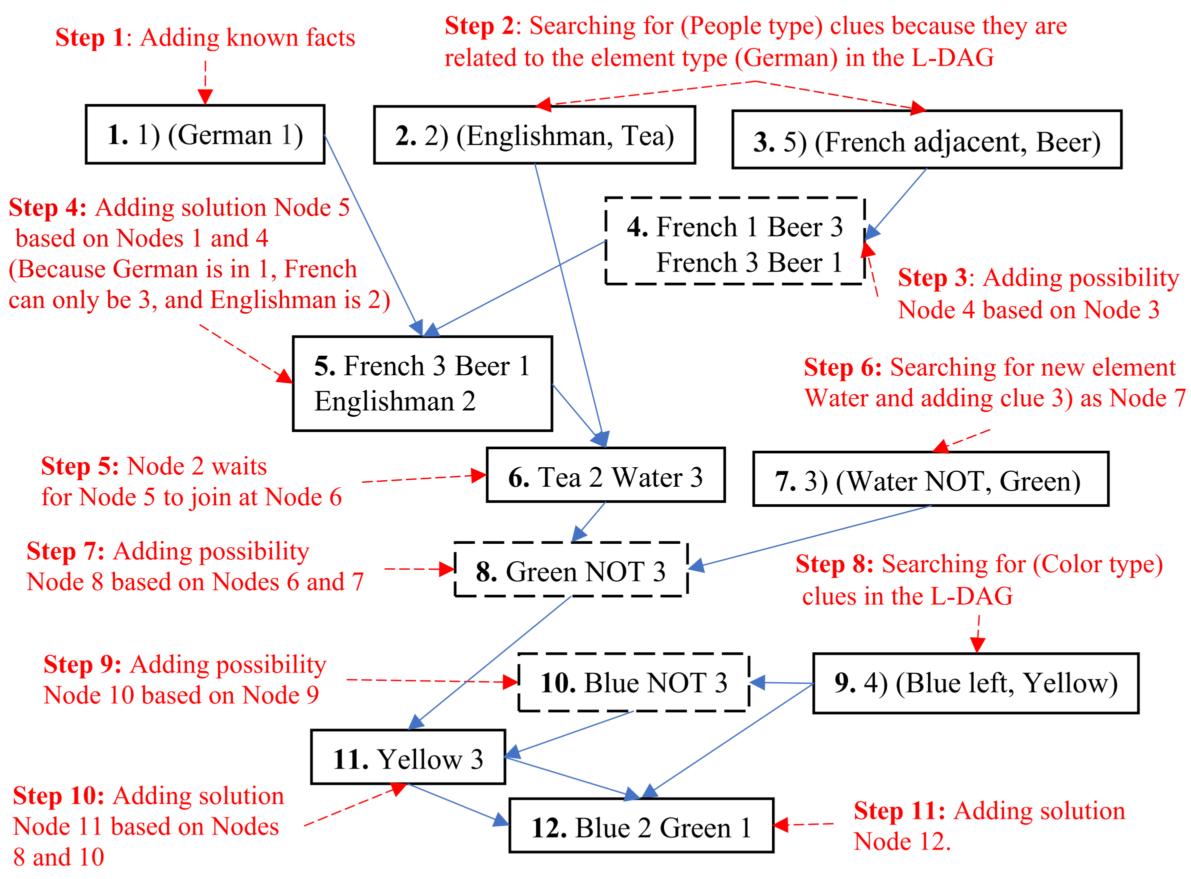

Step 1: Adding known facts
Step 2: Searching for (People type) clues because they are related to the element type (German) in the L-DAG
1. 1) (German 1)
2. 2) (Englishman, Tea)
3. 5) (French adjacent, Beer)
4. French 1 Beer 3 French 3 Beer 1
Step 3: Adding possibility Node 4 based on Node 3
Step 4: Adding solution Node 5 based on Nodes 1 and 4 (Because German is in 1, French can only be 3, and Englishman is 2)
5. French 3 Beer 1 Englishman 2
Step 6: Searching for new element Water and adding clue 3) as Node 7
Step 5: Node 2 waits for Node 5 to join at Node 6
6. Tea 2 Water 3
7. 3) (Water NOT, Green)
Step 7: Adding possibility Node 8 based on Nodes 6 and 7
8. Green NOT 3
Step 8: Searching for (Color type) clues in the L-DAG
Step 9: Adding possibility Node 10 based on Node 9
10. Blue NOT 3
9. 4) (Blue left, Yellow)
11. Yellow 3
Step 10: Adding solution Node 11 based on Nodes 8 and 10
12. Blue 2 Green 1
Step 11: Adding solution Node 12.

Figure 2a | L-DAG for Example A

Example A Solution table

Room	#1	#2	#3
Nationality	German	Englishman	French
Drink	Beer	Tea	Water
Color	Green	Blue	**Yellow**

Example B: Conflict Resolution Support

Example B is a variation of Example A by changing clue 3) (Water NOT, Green) to 3) (Water right, Green). Figure 2b shows the L-DAG for the changed part, illustrating the conflicting clues 3 and 4. Figure 2c (Example B1) presents Conflict Resolution Path 1, and Figure 2d (Example B2) presents Conflict Resolution Path 2.

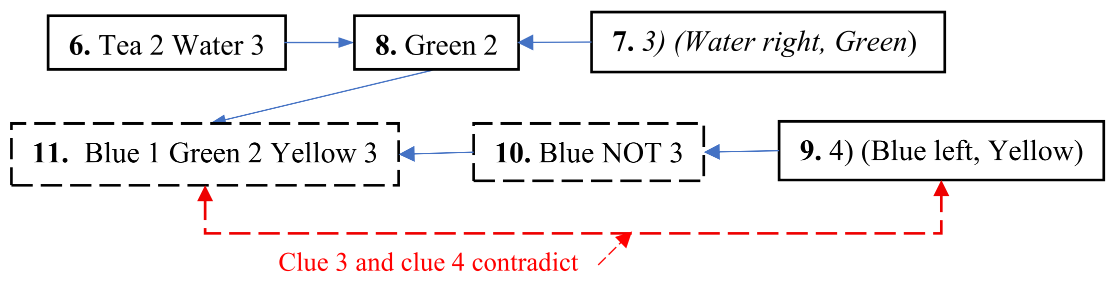

6. Tea 2 Water 3
8. Green 2
7. *3) (Water right, Green)*
11. Blue 1 Green 2 Yellow 3
10. Blue NOT 3
9. 4) (Blue left, Yellow)
Clue 3 and clue 4 contradict

Figure 2b | L-DAG for Introduction Example B (Conflict Clues: 3 and 4)

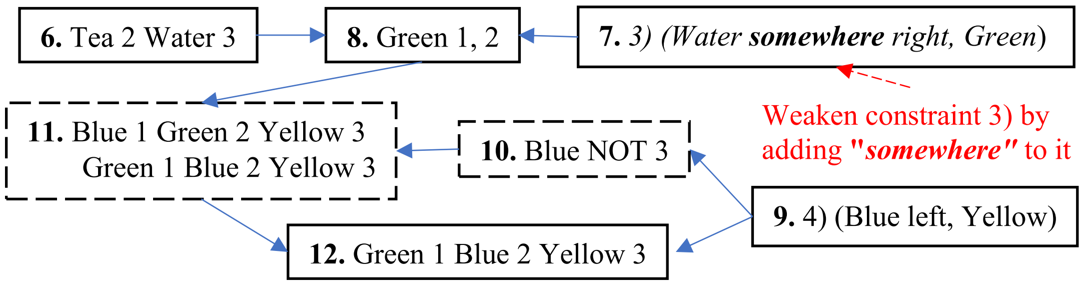

6. Tea 2 Water 3
8. Green 1, 2
7. 3) (Water somewhere right, Green)
Weaken constraint 3) by adding "somewhere" to it
11. Blue 1 Green 2 Yellow 3 Green 1 Blue 2 Yellow 3
10. Blue NOT 3
9. 4) (Blue left, Yellow)
12. Green 1 Blue 2 Yellow 3

Figure 2c | L-DAG for Example B1 (Resolution path 1)

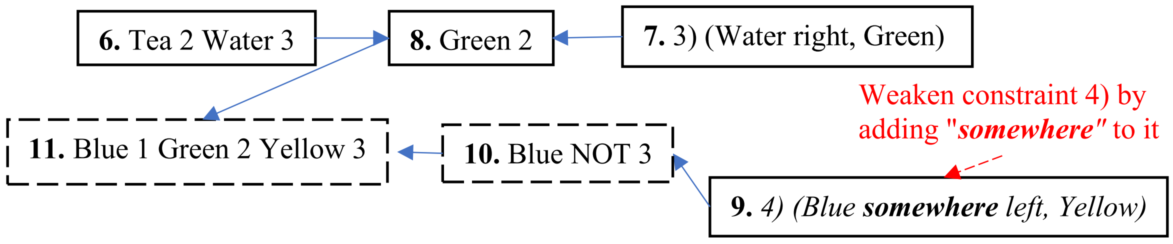

6. Tea 2 Water 3
8. Green 2
7. 3) (Water right, Green)
Weaken constraint 4) by adding "somewhere" to it
11. Blue 1 Green 2 Yellow 3
10. Blue NOT 3
9. 4) (Blue somewhere left, Yellow)

Figure 2d | L-DAG for Introduction Example B2 (Resolution path 2)

In real-world open problems, L-DAG can identify conflicts or other issues and provide possible resolution pathways corresponding to different solutions. These are presented to users, especially researchers, who will make decisions based on their domain knowledge.

3 L-DAG Examples

This section presents three examples: a. a Logical Proposition Proof, b. a complex Logic Grid Puzzle, c. a Multiple-Solution Logic Puzzle.

3.1 Example 1: Logical Proposition Proof (Exploration of Solution Pathways Using L-DAG)

3.1.1 Problem Instance and Process Overview[5]

Logical proposition proof consists in evaluating the conclusion under the condition that all premises are true.

Problem Definition:

Variables: {**A, B, C, D, E, G**}

Premises:

1) A and B differ if and only if C is false: $(A \oplus B) \leftrightarrow \neg C$
2) C is true if and only if B is false and E is true: $C \leftrightarrow (\neg B \land E)$
3) If not A and not E, then C.: $(\neg A \land \neg E) \rightarrow C$
4) If A is true or E is true, then D is false: $(A \lor E) \rightarrow \neg D$
5) If D is true, then both B and C are true: $D \rightarrow (B \land C)$
6) E is true if and only if B is true or D is false: $E \leftrightarrow (B \lor \neg D)$
7) If G is true, then A is true: $G \rightarrow A$
8) If G is true, then C is true and B is false: $G \rightarrow (C \land \neg B)$
9) If E is false and G is true, then D is true: $(\neg E \land G) \rightarrow D$

[5] The problem was created by ChatGPT-4o at my request and subsequently modified by Gemini Pro 2.5 to increase its complexity.

10) If A is true and G is true, then B is true: **(A ∧ G) → B**

Conclusion: B and D differ, and C and G are not both true: **(B⨁D)∧¬(C∧G)**

The problem consists of 6 Boolean variables and 10 premises. The 6 Boolean variables generate 2^6 = 64 possible combinations of variable assignments, corresponding to 64 rows in the truth table. The complete enumeration is omitted and available upon required.

The proof includes two phases.
Phase 1: Each row is evaluated to determine whether it satisfies all constraints. If none of the rows satisfies all constraints, then the conclusion is invalid.
Phase 2: Evaluation of the candidate solution rows.
a. If a counterexample is found, the conclusion is invalid.
b. Otherwise, the conclusion is considered valid.

The purpose of the L-DAG is to find a counterexample row with minimal effort by leveraging constraint search strategies and logical deduction.

3.1.2 L-DAG: Non-Graphic and Graphic Solutions

There are two ways to solve the problem: the non-graphic process and the graphic L-DAG. The non-graphic version is presented first because it also provides a step-by-step detailed explanation of the construction process for the graphic solution. In the proofs, True is represented by 1 and False by 0.

L-DAG construction table records the step-by-step process for the Non-Graphic Solution. In the table, deductive values are shown in bold.

Table 1. L-DAG Construction Process for No-Graphic Solution

Search strategies Applied **5**	**Node Type** **2**	**Parent Nodes** **3**	**TID Child nodes** **4**
Material Implication for **P→ Q:** If **P=0** (**false**), then **P → Q** is true for any **Q**. If **Q = 1** (**true**), then **P → Q** is true for any **P**.	Premise	**Root Seed**	1. 5) (¬E∧G)→D =1
			2. **D=0**
	Solution	1, 2	3. 4) (A ∨ E) → ¬D =1
	Premise	**Root Seed**	4. 7) G → A = 1
			5. **G=0**
			6. 8) G→(C∧¬B) =1
			7. 9) (¬E∧G)→D =1
	Solution	4, 5, 6, 7	8. 10) (A∧G)→B = 1
	Premise	3	9. 6) E ↔ (B ∨ ¬D) =1
	Solution	9	10. **E=1**
Search *E*-related premises	Premise	Root	11. 3) ((¬A ∧ ¬E) → C)
		10, 11	12. 3) (¬A ∧ 0 → C) =1
Material Implication: P = 0	Possibility	12	13. **C=1**
			14. **C=0**
	Premise	Root	15. 2) C ↔ (¬B ∧ E)

Search *C*-related premises		10, 15	16. 2) $(C \leftrightarrow \neg B) = 1$
	Possibility	13, 16	17. **B=0**
		14, 16	18. **B=1**
Search *B*-related premises	Premise	Root	19. 1) $(A \oplus B) \leftrightarrow \neg C)$
		19	20. 1) $(A \oplus B) \leftrightarrow 0\ ) = 1$
		14, 19	21. 1) $(A \oplus B) \leftrightarrow 1) = 1$
		21	22. $(A \oplus B) = 1$
		20	23. $(A \oplus B) = 0$
	Solution	17, 18, 22, 23	24. **A=0**
	Premise	1, 17	25.$(B \oplus D) = 0$
	Solution	25	26. $(B \oplus D) \wedge \neg(C \wedge G) = 0$
	Premise	1, 18	27. $(B \oplus D) = 1$
		2	28. $\neg\ (C \wedge G)$
		28	29. $\neg\ (C \wedge F) = 1$
	Solution	27, 29	30. $(B \oplus D) \wedge \neg(C \wedge G) = 1$

Solution table for Example 1

Variable & Premise	**A**	**B**	**C**	**D**	**E**	**G**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**All P**	**Conclusion**
Node 31	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	**1**	**1: valid**
Node 26	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	**1**	**0: invalid**

Answer: The argument is invalid because a counterexample exists. The row (Node 26) in the table demonstrates a case where all premises are true, but the conclusion is false.

Remark

i. Strategies for Phase 1

In order to make Phase 1 successful, we use search strategies that set the premises to true while giving other variables the maximum possibility to be configured as either 1 (true) or 0 (false). This approach aims to:

- Make other variables easier to satisfy the premises; and
- Once all premises are satisfied, retain more degrees of freedom to explore the solution space from the same root configuration

Consider the 64-row truth table: each row starts with the same initial values and varies the remaining variables to true or false, effectively branching from a common root.

ii. Application of Material Implication

In this example, we first use mathematical logic implication rules to set premises to true. Similar to Example A, where the fact "The German occupies house #1" was fixed as true, here we apply material implication for conditions of the form $P \rightarrow Q$. By setting *P = 0 (false)*, the premise becomes true regardless of *Q*. This provides two possibilities (0 or 1) for each variable in the expression *Q*, allowing more configurations to satisfy the remaining premises in downstream processing and to branch additional solution paths.

iii. Sequential Deduction of Remaining Premises

After exhausting all premises where material implication could be applied, three premises remain to be configured to true. Following the same approach as in Example A, we searched for premises related to each newly deduced value:

- After deducing $E = 1$ (Node 10), we searched all E-related premises.
- After deducing $C = 1$ (Node 13) or $C = 0$ (Node 14), we searched all C-related premises.
- After deducing $B = 0$ (Node 17) or $B = 1$ (Node 18), we searched all B-related premises, then deduced the final value $A = 0$ (Node 24).

iv. Exploration Outcome

This example successfully uses search strategies to identify the only two valid rows in the 64-row truth table:

- Node 26, a counterexample (showing the conclusion is invalid), and
- Node 30, a valid case.

v. Strategy Development for Other Conditions

In this example, we only require the material implication rule for one-directional conditions ($P \rightarrow Q$). For bi-directional conditions ($P \leftrightarrow Q$), additional strategies may be developed to make Phase 1 successful. For example, considering the formula $Win\text{-}Rate = k \cdot DCD/P$, one can use DCD metrics and domain expertise to design strategy rules to make exploration more efficient. This is further demonstrated in Section 3, where a much more complex logical Example 2 shows how strategies can be enriched using Win-Rate. The exploration process is part of the workflow summarized in Figure 5.

Figure 3 shows the L-DAG graphic solution. In Figure 3, all premises are highlighted in yellow, and the solutions are highlighted in blue. The final solutions (26 and 30) are shown in red font. All TIDs match those in Row 4 of the L-DAG construction table. From the table, it is clear that with these search strategies, L-DAG efficiently satisfies 7 of the 10 premises and easily deduces the remaining 3 values. This approach directly identified the only two rows in the 64-row truth table where all premises held true, without redundant exploration.

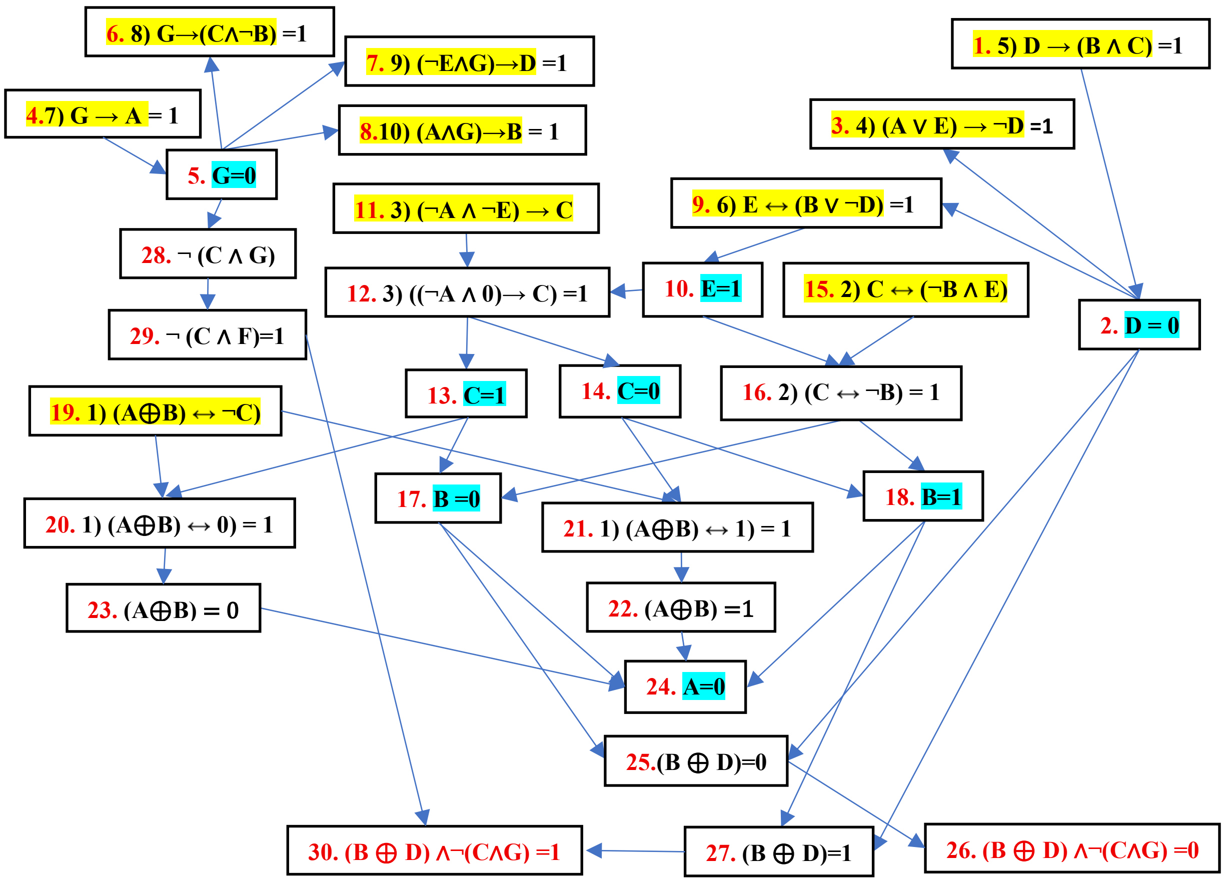


Figure 3. | L-DAG for Example 1

3.1.3 Exploration Implication

(1) Logical Proposition Proof

In this type of logical task, a counterexample often provides a more concise proof of invalidity. However, finding such a counterexample still requires systematic exploration of the logical space to identify a configuration that satisfies all premises while falsifying the conclusion. This exploration process cannot be omitted. L-DAG supports efficient global exploration by incrementally constructing reasoning paths through structurally guided, dependency-driven routing, without requiring exhaustive enumeration of all possible solutions.

(2) Broader Scientific Exploration

Example 1 demonstrates how L-DAG may support systematic exploration and discovery in scientific applications

- Each premise or constraint (e.g., $G \rightarrow A$, $(A \lor E) \rightarrow \neg D$, $(A \oplus B) \leftrightarrow \neg C$) can represent a definition, theorem, lemma, or inference in a scientific domain.

- Unlike standard proposition proof processes, which typically stop upon finding a counterexample, L-DAG continues exploring the solution space. All outcomes can be presented to humans for further evaluation and interpretation using domain knowledge.

- Counterexamples**:** A configuration that initially appears as a counterexample may be transformed into a valid solution by modifying or relaxing certain constraints if appropriate.
- Multiple Valid Solutions: When multiple valid solutions are identified, they may provide alternative pathways for product development, for example in biology or chemistry applications.
- Unique Solutions: If a unique solution is required, additional constraints can be applied to further narrow the solution space and eliminate ambiguity.

- Unlike the example, constraints may be defined by humans, generated automatically, or extended across an entire range of disciplines without a fixed limit on their number. The exploration can continue iteratively, pruning invalid branches and incorporating additional constraints to discover new insights and pathways for research.

3.2 Example 2: Problem with a Unique Solution

A Logic Grid Puzzle

There are five houses of different colors with different jobs; they love different books, sports, and instruments. The entities and constraints are as follows:

Color: {Blue, Green, Red, White, Yellow}
Name: {An, Kay, Mike, Nancy, Tom}
Job: {Artist, Doctor, Nurse, Teacher, Writer}
Sport: {Basketball, Football, Running, Swimming, Volleyball}
Book: {Fantasy, History, Mystery, Romance, Science}
Instrument: {Drums, Flute, Guitar, Piano, Violin}

1) The Doctor is next to the Teacher.
2) Tom is the Running lover.
3) The Romance lover is in house 1 or house 5.
4) The set of instruments is partitioned into two disjoint groups: string and non-string (S and ~S), such that ball sport enthusiasts (B) and non-ball sport enthusiasts (~B) are strictly assigned to different groups. This constitutes a Second-Order Logic (SOL) constraint.
5) Football lover is somewhere to the right of the Basketball lover.
6) The Writer is anywhere between the Nurse and the Doctor.
7) The Romance lover is a Swimming lover.
8) The Basketball lover lives in the Green house.
9) The Drums owner is anywhere to the right of the Piano owner.
10) For each house, add the number of letters in its color, job, book, sport, and instrument. The totals form the set S = {s1, s2, s3, s4, s5}. There exist at least two houses with the same total (this constitutes a Second-Order Logic (SOL) constraint).
11) The History lover is anywhere to the right of the Drums owner.
12) The Fantasy lover is next to the Science lover.
13) Even-letter names not adjacent
14) The Fantasy lover is at one of the ends.
15) The Red house is somewhere to the left of the Basketball lover.
16) The Writer is anywhere to the right of the Drums owner.

17) Nancy is anywhere between the Green house and Kay, in this sequence.
18) The Blue house is not next to the Volleyball lover.
19) The Piano owner is anywhere between the Doctor and the Guitar owner, in this sequence.
20) Tom is in house 1 or house 5.
21) The Yellow house is to the left of the White house.

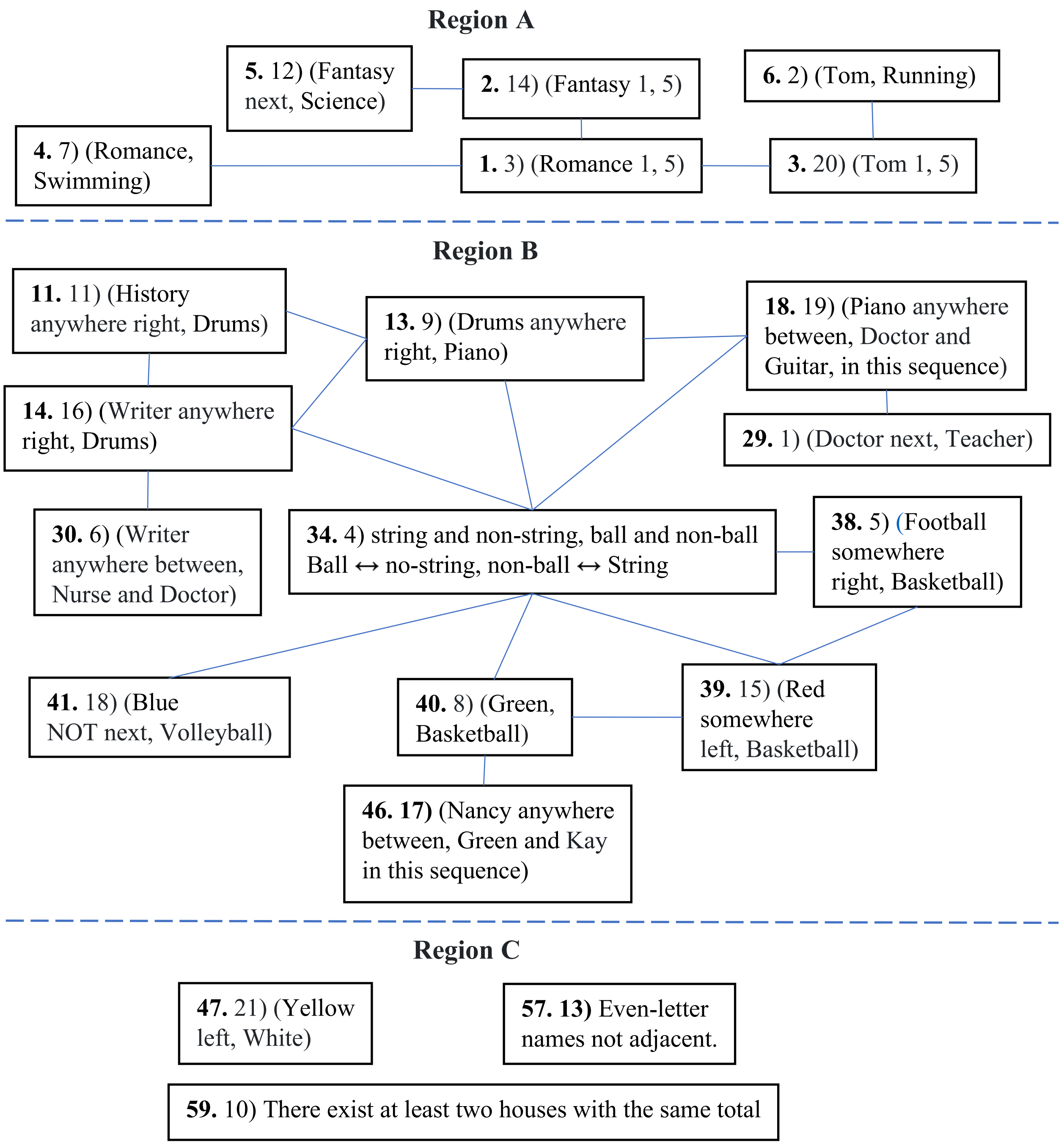
Question: who is to the left of Mystery lover?

3.2.1 Data Correlation Density in the Task Domain

For this type of task, the DCD is defined as the total number of times an entity appears within the task's constraints and clues. We can use an Entity–Association–Task (EAT) structure to generate an undirected graph representing the initial Data Correlation Density for Example 1. Figure 4 offers an intuitive visualization of why we term this metric "Data Correlation Density": entities with a higher count are involved in more relationships, creating a denser web of connections and providing more cross-reference information.

The DCD is dynamically updated whenever a new solution node is added. This is because a solution provides a new, factual constraint to other nodes it connects to, thereby creating new cross-reference data.

The final DCD is the completed solution table, because the clues are essentially static possibility nodes, functionally equivalent to the dynamically constructed possibility nodes. After removing the clues, the DAD becomes the solution table, which details all entities and their resolved relationships.



Region A
5. 12) (Fantasy next, Science)
2. 14) (Fantasy 1, 5)
6. 2) (Tom, Running)
4. 7) (Romance, Swimming)
1. 3) (Romance 1, 5)
3. 20) (Tom 1, 5)
Region B
11. 11) (History anywhere right, Drums)
13. 9) (Drums anywhere right, Piano)
18. 19) (Piano anywhere between, Doctor and Guitar, in this sequence)
14. 16) (Writer anywhere right, Drums)
29. 1) (Doctor next, Teacher)
30. 6) (Writer anywhere between, Nurse and Doctor)
34. 4) string and non-string, ball and non-ball Ball ↔ no-string, non-ball ↔ String
38. 5) (Football somewhere right, Basketball)
41. 18) (Blue NOT next, Volleyball)
40. 8) (Green, Basketball)
39. 15) (Red somewhere left, Basketball)
46. 17) (Nancy anywhere between, Green and Kay in this sequence)
Region C
47. 21) (Yellow left, White)
57. 13) Even-letter names not adjacent.
59. 10) There exist at least two houses with the same total

Figure 4 | Initial Data Correlation Density Graph for Example 2

3.2.2 Construction Workflow

Figure 5 is the L-DAG construction workflow, in which the Seed Identification Prioritization Strategy is one implementation of the Seed-seeking rule, which guides the construction process in the direction of increasing Win-Rate.

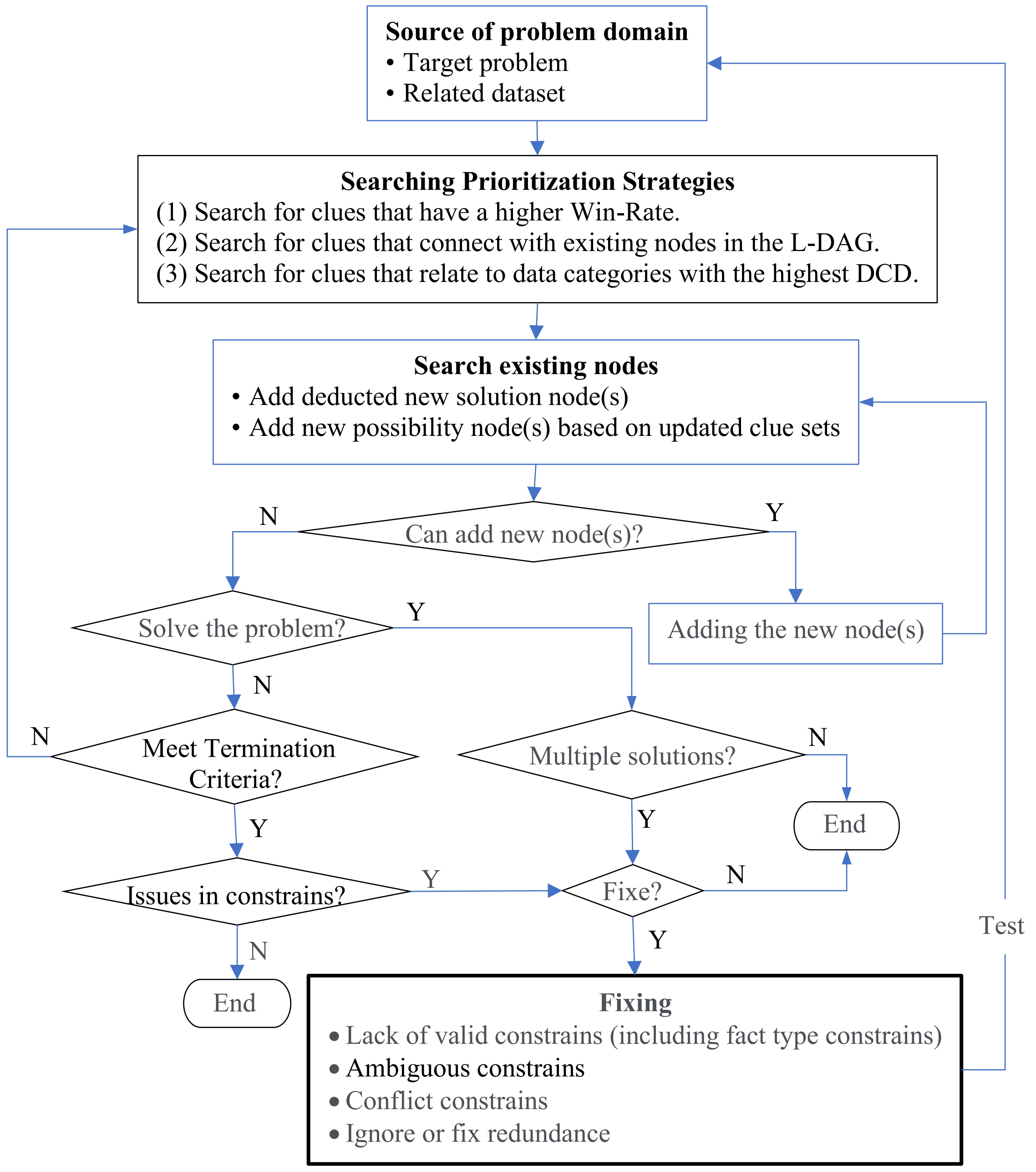

Source of problem domain
• Target problem
• Related dataset
Searching Prioritization Strategies
(1) Search for clues that have a higher Win-Rate.
(2) Search for clues that connect with existing nodes in the L-DAG.
(3) Search for clues that relate to data categories with the highest DCD.
Search existing nodes
• Add deducted new solution node(s)
• Add new possibility node(s) based on updated clue sets
Can add new node(s)?
N
Y
Solve the problem?
Y
Adding the new node(s)
N
N
Meet Termination Criteria?
Multiple solutions?
N
Y
Y
End
Issues in constrains?
Y
Fixe?
N
Y
N
Test
End
Fixing
• Lack of valid constrains (including fact type constrains)
• Ambiguous constrains
• Conflict constrains
• Ignore or fix redundance

Figure 5 | L-DAG Construction Workflow

Following the same workflow, there are two ways to solve the problem: the non-graphic process as shown in Section 3.2.3, and the graphic L-DAG shown in Section 3.2.4. The non-graphic version is presented first because it also provides a step-by-step detailed explanation of the construction process for the graphic solution.

3.2.3 Non-Graphic L-DAG

The columns are defined as follows:

i. Column 1:

Catalog DCD count: This is the total number of entities of each data type (catalog) that appear in the constructed L-DAG prior to this node. For example, "Color 4, Name 3" means there are 4 Color-type entities and 3 Name-type entities present in existing nodes.

Priorities applied (1), (2), and (3) are described in the Searching Prioritization Strategy in Figure 5. Priority (3) uses the maximum Catalog DCD count to search.

ii Column 2 – Node types: Root, Possibility, and Solution.
Node 50 has the type of both Possibility and Solution, because it is in a possibility path that branches from Node 49. After the other two possibility paths, 51 and 52, are eliminated, it becomes the solution.

Node 54 (Nancy 4, Kay 5) has one possibility, but it is not a solution node because it is in a possibility path.

iii Column 3 indicates the parent node TIDs of the child nodes in Column 4.

iv Column 5 lists the DCD for the entities in Column 4. For example, Node 30.6) (Writer anywhere between Nurse and Doctor): Writer, Nurse, and Doctor appear 2, 1, and 3 times, respectively. DCD is updated only when a new solution is added that provides a new Fact constraint.

v Column 6 is the list of possible positions of entities in the problem domain. In Node 59.10 (There exist at least two houses with the same total), P is an astronomical figure (the value calculated by o4-mini-high is 19,184,947,200).

vi Column 7 Win-Rate: Column 6 / Column 7. If there is more than one number in Column 6, the maximum DCD value is used to calculate the Win-Rate. The Win-Rate does not need to be updated because DCD is only updated when a new solution is reached, which implies the possibility is 1 (100%). In such cases, Win-Rates such as 3/1 or 4/1 provide no meaningful information and obscure the intended purpose of this column, which is to highlight the construction trend. The Win-Rate R for Node 59 is 1.6×10^{-10}.

A Note on Practical Application:
The detailed formalism of the L-DAG construction table, particularly the Win-Rate calculation, is designed to provide a systematic process for an AI. For a human user, the core strategy is simpler and more intuitive: typically, one can identify the next best clue to pursue by observing which entities appear most frequently across the constraints, as these offer the greatest potential for cross-referencing. The formal calculation translates this human intuition into a machine-executable algorithm.

L-DAG Construction table

Catalog DCD count & Priority applied 1	Node Type 2	Parent nodes 3	TID. Node 4	DCD 5	Possibi-lities 6	Win Rate 7 (5/6)

(1) Search More Win Rate entities	**Root**	**Seed**	**1.** 3) (Romance 1, 5)	2	2	1
			2. 14) (Fantasy 1, 5)	2	2	1
			3. 20) (Tom 1, 5)	2	2	1
(2) Search Romance Fantasy Tom	Root		**4.** 7) (Romance, Swimming)	2, 1	2	1
			5. 12) (Fantasy next, science)	2, 1	2	1
			6. 2) (Tom, Running)	2, 1	2	1
	Possibility	1, 4	**7.** Swimming 1, 5		2	
		2, 5	**8.** Science 2, 4		2	
		3, 6	**9.** Running 1, 5		2	
		2, 3	**10. History 2, 3, 4**		3	
(2) Search History	**Root**	**Seed**	**11.** 11) (History anywhere right Drums)	1, 3	10	0.3
	Possibility	10, 11	**12.** Drums 1, 2, 3		3	
(2) Search Drums	Root		**13.** 9) (Drums anywhere right Piano)	3, 2	10	0.3
			14. 16) (Writer anywhere right Drums)	2, 3	10	0.3
	Possibility	13	**15.** Drums NOT 1, 5		3	
		10, 12, 15	**16.** Drums 2, 3, History 3, 4		2	
		13, 16	**17.** Piano 1, 2		2	
(2) Search Piano	Root	17	**18.** 19) (Piano anywhere between doctor and Guitar, in this sequence)	2, 3, 1	10	0.3
	Possibility	18	**19.** Piano NOT 1, 5		3	
	Solution	16, 17, 19	**20.** Piano 2, Drums 3, History 4	3, 4, 5	1	
		8, 20	**21.** Science 2	2	1	
		2, 24	**22.** Fantasy 1	3	1	
		1, 20, 25	**23.** Romance 5, Mystery 3	3, 1	1	
		4, 7, 9, 26	**24.** Swimming 5 Running 1	2, 2	1	
		6, 27	**25.** Tom 1	3	1	
		18, 20	**26.** Doctor 1	4	1	
	Possibility	14, 20	**27.** Writer 4, 5		2	
		18, 20, 22	**28.** Guitar 4, 5		2	
(2) Search Doctor	Root		**29.** 1) (Doctor next Teacher)	3, 1	8	0.38
			30. 6) (Writer anywhere between Nurse and Doctor)	2, 1, 3	20	0.15
	Solution	26, 29	**31.** Teacher 2	2	1	
		26, 27, 30	**32.** Writer 4, Nurse 5	3, 2	1	
		26, 31, 32	**33.** Artist 3	1	1	
2) Search Instrument	Root		**34.** 4) String and Non-String, Ball and Non-Ball	8	12	0.67

			Ball ↔ No-String, Non-ball ↔ String			
	Solution	20, 24, 34	**35.** Flute 4	1	1	
		24, 28, 34	**36.** Guitar 5	2	1	
		24, 34, 36	**37.** Violin 1	1	1	
(2) Search Sport	Root		**38.** 5) Football somewhere right Basketball)	2, 3	10	0.3
			39. 15) (Red somewhere left Basketball)	1, 3	10	0.3
			40. 8) (Green, Basketball)	2, 3	5	0.6
			41. 18) (Blue NOT next Volleyball)	1, 1	12	0.17
	Possibility	34, 24, 38	**42.** Basketball 2, 3 Volleyball 3, 4 Football 4, 2		3	
		39, 42	**43.** Red 1, 2		2	
		41, 42	**44.** Blue 1, 2, 4, 5		4	
		40, 42, 43	**45.** Red 1 Green 2 Red 1 Green 3 Red 2 Green 3		3	
(2) Search Green	Root		**46.** 17) (Nancy anywhere between Green and Kay, in this sequence)	1, 2, 1	10	0.2
(Book, Job, Instrument, Sport filled) Color 4 Name 3 **(3) Search Color**	**Root**	**Seed**	**47.** 21) (Yellow left, White)	1, 1	10	0.1
	Possibility	44, 47	**48.** Yellow 1 White 2 Blue 4, 5 Yellow 2 White 3 Blue 4, 5, 1 Yellow 3 White 4 Blue 5, 1, 2 Yellow 4 White 5 Blue 1, 2		10	
	Possibility	45, 48	**49.** Red 1 Green 2 Yellow 3 White 4 Blue 5 Red 1 Blue 2 Green 3 Yellow 4 White 5 Red 2 Green 3 Yellow 4 White 5 Blue 1		3	
	Possibility &Solution	49	**50.** Red 1 Green 2 Yellow 3 White 4 Blue 5	2, 3, 2, 2, 2	1	
	Possibility	49	**51.** Red 1 Blue 2 Green 3 Yellow 4 White 5		1	
	Possibility	49	**52.** Red 2 Green 3 Yellow 4 White 5 Blue 1		1	
	Solution	40, 41, 42, 50	**53.** Basketball 2 Volleyball 3 Football 4	4, 2, 2	1	
	Possibility	46, 51, 52	**54.** Nancy 4 Kay 5		1	
		25, 54	**55.** An 2, 3 Mike 3, 2		2	
		50, 46	**56.** Nancy 3, 4 Kay 4, 5		2	
(2) Search Name	Root		**57.** 13) Even-letter names not adjacent	4	72	0.06
	Solution	25, 56, 57	**58.** Nancy 3 Key 5	2, 1	1	
(2) Search Rest	Root		**59.** 10) There exist at least two houses with the same total	30	**P**	**R**

	Possibility	All	**60.** An 2 Mike 4: Total (32, 36, 39, 35, 38) An 4 Mike 2: Total (32, 38, 39, 33, 33)			
	Solution	59, 60	**61.** Mike 2 An 4	1, 1	1	

Remark

v. Following the Searching Prioritization Strategies:
- Apply Priority 1: we find the starting seeds, the root nodes (clues: 3, 14, and 20), that have the highest Win-Rate.
- Apply Priority 2: we link the clues from two disconnected regions, Region A (which has the highest initial Win-Rate) and Region B (which has the highest DCD), in the DCD graph (Figure 4), using the new seed derived by constructing Possibility Node 10.
- Apply Priority 3: When no new clues can be found using any entities in the existing L-DAG (that is, after Priority (2) fails), we use the highest Catalog DCD count to find new seeds. For example, after Node 46, the Color catalog has a count of 6 and the Name catalog has a count of 4, so we search Color first, then Name. If a clue contains both Color and Name, it is selected as the first new seed.

vi. Figure 4 and the L-DAG construction table illustrate the following:
- Clues with higher Win-Rate (Max / Possibilities) are generally processed earlier, as frequently appearing entities tend to associate with more nodes and provide more cross-reference information. Clues with low Win-Rate are deferred until other branches reduce their possibilities. For example, Clue 10 (Node 59) had an astronomical number of possibilities, which dropped to 2 after joining another branch at Node 60.

- Second-Order Logical (SOL) clues, even with high Win-Rate, are processed later because their logical operations apply to sets, not individuals. These clues require the possibilities of individual entities to be reduced before set-level reasoning becomes feasible. For instance, Node 34 (Clue 4: "Ball ↔ No-String, Non-ball ↔ String") with a Win-Rate of 0.67 was processed only after related entities were partially resolved.

vii. Even when the possibility is theoretically computable, we estimate rather than explicitly calculate large values, as in the case of Clue 10. Since Clue 10 is a Second-Order Logical clue, it must be processed only after other processes have eliminated most combinations to reduce its possibilities.

Solution Table

House	#1	#2	#3	#4	#5
Color	Red	Green	Yellow	White	Blue
Name	Tom	Mike	Nancy	An	Kay
Job	Doctor	Teacher	Artist	Writer	Nurse
Sport	Running	Basketball	Volleyball	Football	Swimming
Book	Fantasy	Science	Mystery	History	Romance
Instrument	Violin	Piano	Drums	Flute	Guitar

The colors used in the table match those in the Graphic L-DAG nodes (Section 2.1.5) and are used to identify the paths leading to the solution.

3.2.4 Graphic L-DAG

Due to space constraints, Figure 6b continues from Figure 6a. Nodes 20, 24, 25, and 28 are duplicated from Figure 6a into Figure 6b.

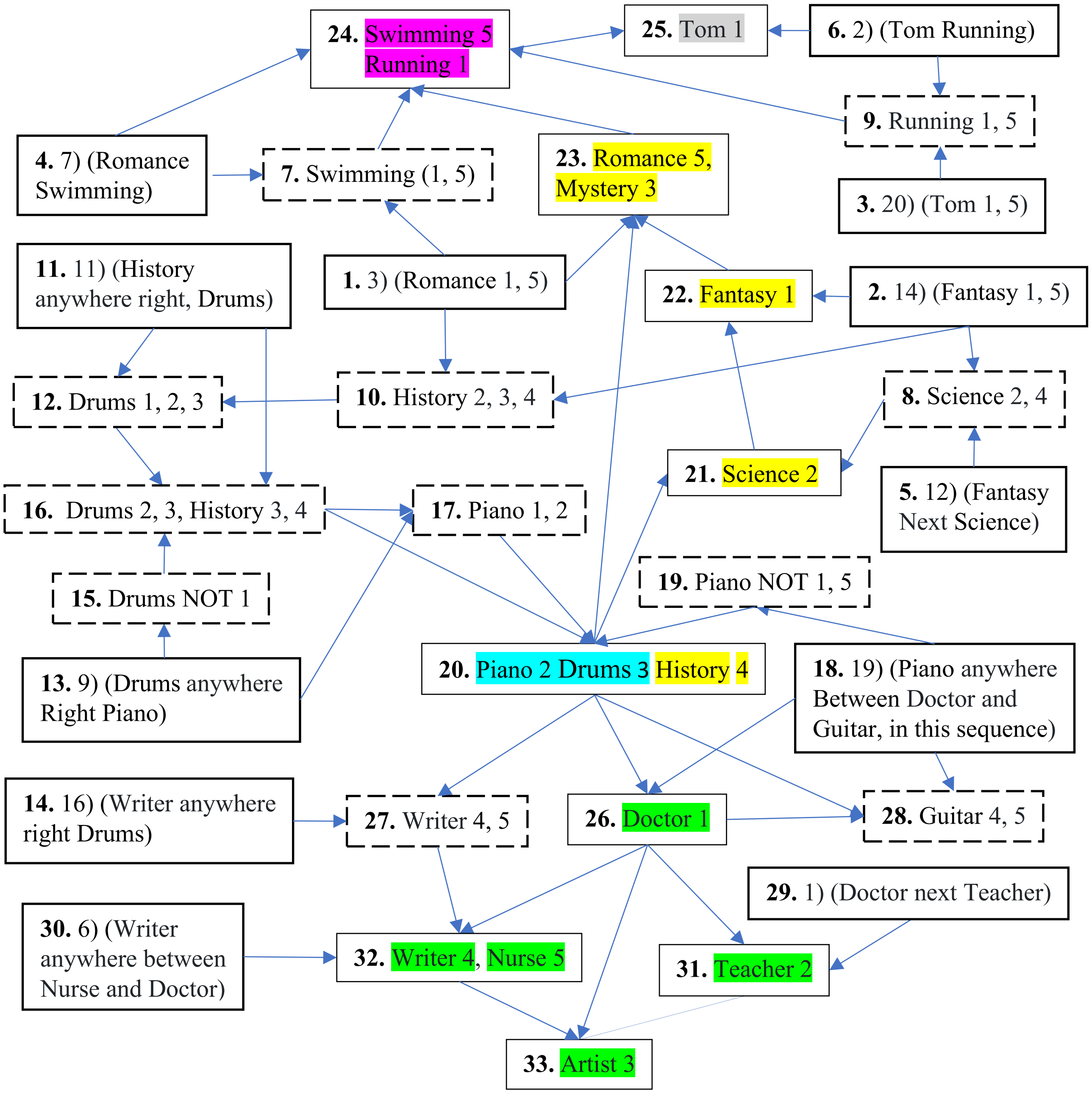


Figure 6a | L-DAG for Example 2

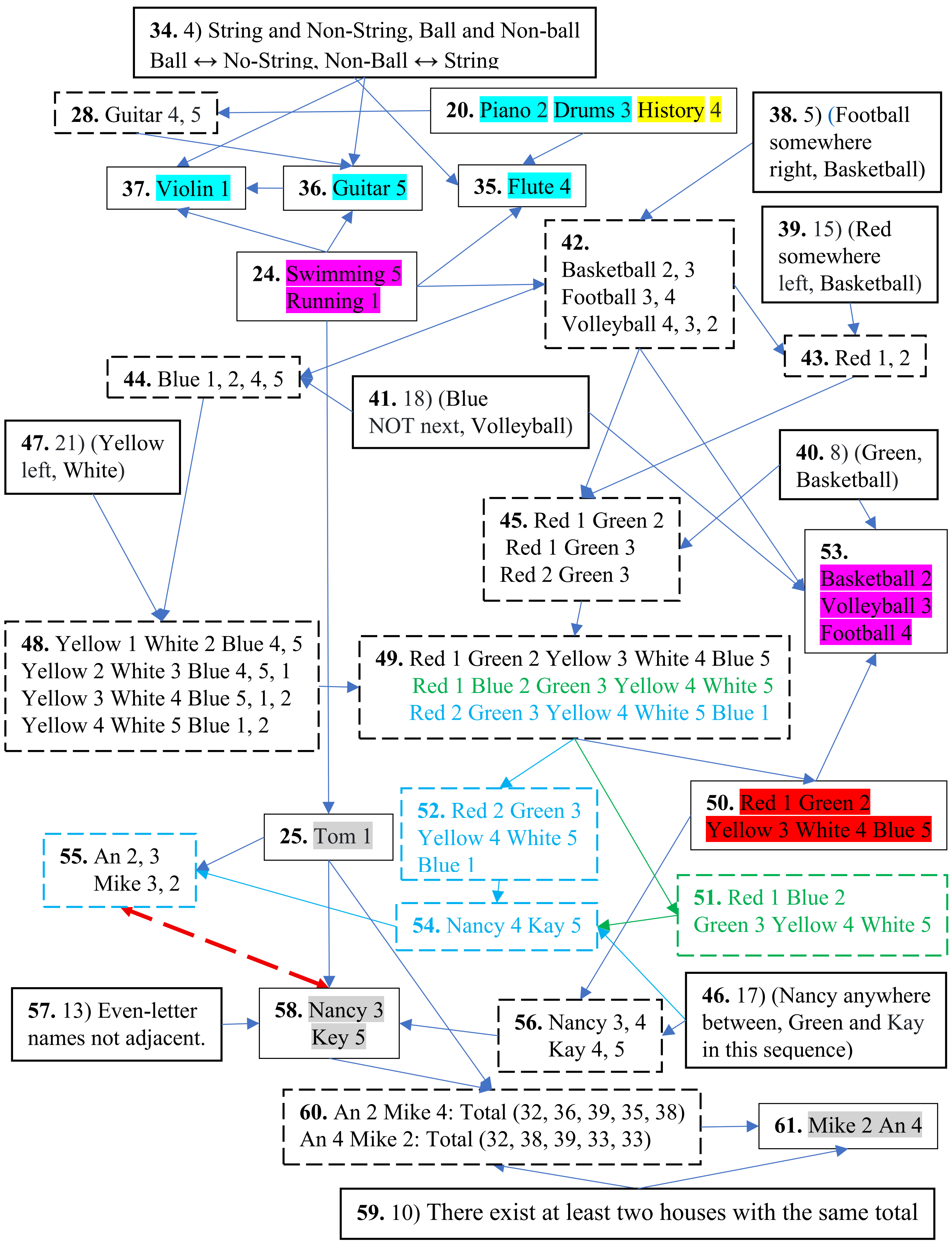

34. 4) String and Non-String, Ball and Non-ball
Ball ↔ No-String, Non-Ball ↔ String
28. Guitar 4, 5
20. Piano 2 Drums 3 History 4
38. 5) (Football somewhere right, Basketball)
37. Violin 1
36. Guitar 5
35. Flute 4
39. 15) (Red somewhere left, Basketball)
24. Swimming 5 Running 1
42. Basketball 2, 3 Football 3, 4 Volleyball 4, 3, 2
43. Red 1, 2
44. Blue 1, 2, 4, 5
41. 18) (Blue NOT next, Volleyball)
47. 21) (Yellow left, White)
40. 8) (Green, Basketball)
45. Red 1 Green 2 Red 1 Green 3 Red 2 Green 3
53. Basketball 2 Volleyball 3 Football 4
48. Yellow 1 White 2 Blue 4, 5 Yellow 2 White 3 Blue 4, 5, 1 Yellow 3 White 4 Blue 5, 1, 2 Yellow 4 White 5 Blue 1, 2
49. Red 1 Green 2 Yellow 3 White 4 Blue 5 Red 1 Blue 2 Green 3 Yellow 4 White 5 Red 2 Green 3 Yellow 4 White 5 Blue 1
52. Red 2 Green 3 Yellow 4 White 5 Blue 1
50. Red 1 Green 2 Yellow 3 White 4 Blue 5
25. Tom 1
55. An 2, 3 Mike 3, 2
54. Nancy 4 Kay 5
51. Red 1 Blue 2 Green 3 Yellow 4 White 5
57. 13) Even-letter names not adjacent.
58. Nancy 3 Key 5
56. Nancy 3, 4 Kay 4, 5
46. 17) (Nancy anywhere between, Green and Kay in this sequence)
60. An 2 Mike 4: Total (32, 36, 39, 35, 38) An 4 Mike 2: Total (32, 38, 39, 33, 33)
61. Mike 2 An 4
59. 10) There exist at least two houses with the same total

Figure 6b | L-DAG for Example 2 (Continued)

Remark

- At each step, all possibility nodes are constructed, regardless of whether they are ultimately used. For example, in Figure 6a, Nodes 7, 8, and 9 are created; Nodes 8 and 9 are used later, but Node 7 is unnecessary, as Node 24 can be directly deduced from Nodes 4 and 23 without it.

- In Figure 6b, Node 49 branches into three possible paths:
  – The Green Path (starting from Node 51) and the Blue Path (starting from Node 52) both terminate at Node 55, which conflicts with Node 58.
  – The Black Path (starting from Node 50), after eliminating the possible solutions of the Green and Blue Path branches, is the only valid path that satisfies all constraints.

- As shown in Figure A, after the three paths produce intermediate solution nodes 50 and 53, downstream reasoning can directly use these results. To ensure a clean task domain, backward parallel processing is triggered to prune the branches (38–45, 47–49, and 52–55), including the initial constraints and possibility nodes. In fact, you can create all such initial constraints nodes (38–41 and 47). If unremoved, these branches may be reconsidered during reasoning, disrupt the process, and consume unnecessary memory. Notice that after the pruning process, the TIDs are still in topological order with discontinued numbers. Reasoning and pruning should always proceed in parallel, step by step, to prevent uncontrolled growth. In some problems, if the intermediate results are lemmas or useful for case studies, they should be saved before pruning. Maintaining a clean and well-structured reasoning process is essential in the L-DAG process.

3.3 Example 3: L-DAG with Pre-Decomposition for Multiple Solutions

Problem. Four travelers from England, France, Japan, and Germany meet. Each speaks their own language and one of the other three languages. We know the following:

1) B cannot speak English, but can translate for A and C when they talk.
2) A is Japanese, and D cannot speak Japanese, but they can communicate without difficulty.
3) Only one language can be spoken by three of them.
4) No one among the four can speak both Japanese and French.

Note: The order of A, B, C, and D, and the order of England, France, Japan, and Germany, may not correspond.
Question: Which country is each of the travelers A, B, C and D from, and which foreign language does each of them speak? [6]

The data in the L-DAG are represented as follows:

People: A, B, C, D
Countries: Abbreviated as JP, FR, DE, UK
Foreign or Second Languages: English, German, French, Japanese

[6] It is a translation of a puzzle posted on a Chinese website. <https://zhidao.baidu.com/question/196769231.html?qbl=relate_question_0> The website is no longer exit. The original problem was "Four travelers from England, France, Japan, and Germany meet. Each speaks their own language and one of the other three languages. One language can be spoken by all three of them, but no language can be spoken by all four." I removed the redundant constraint, "but no language can be spoken by all four," because the problem already excludes the possibility of all four speaking the same language. The question is complex, and the poster's steps and answers were incorrect.
.

For example, A (JP/English) or (JP/German) indicates that A is from JP and his foreign language is either English or German.

3.3.1 L-DAG Workflow

Example 3 can be solved in two ways: (i) by applying L-DAG directly, as shown in Figure B, without detailed description since the process is identical to Example 2; or (ii) by using L-DAG with Pre-Decomposition.

Because the constraints in these examples are complex and composite, implicit conditions or hidden possibilities may be overlooked during reasoning. For example, none of the AI systems produced the correct answer. Therefore, we decompose the constraints into meta-constraints before applying the L-DAG process.

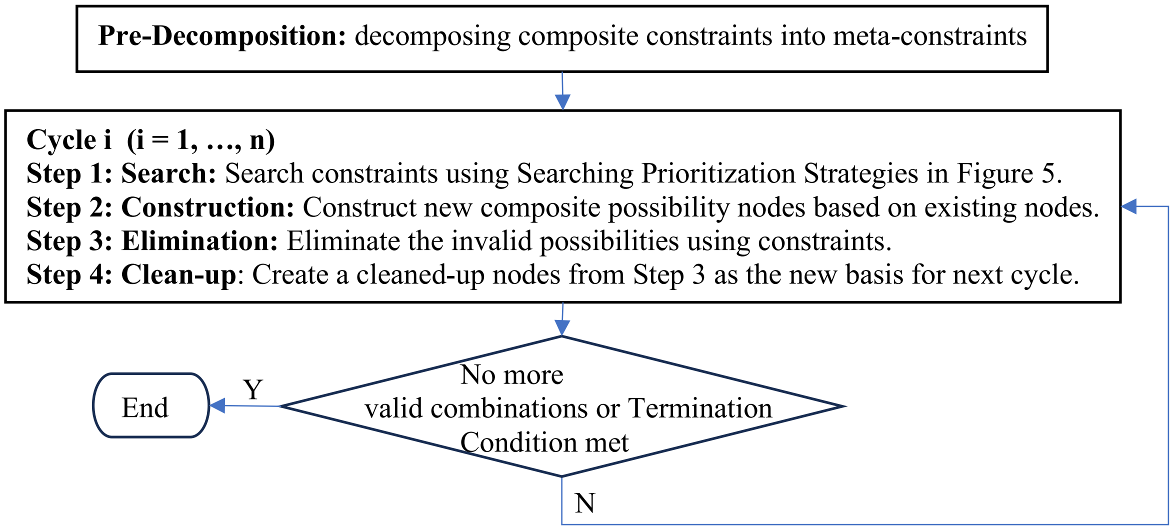

Pre-Decomposition: decomposing composite constraints into meta-constraints
Cycle i (i = 1, …, n)
Step 1: Search: Search constraints using Searching Prioritization Strategies in Figure 5.
Step 2: Construction: Construct new composite possibility nodes based on existing nodes.
Step 3: Elimination: Eliminate the invalid possibilities using constraints.
Step 4: Clean-up: Create a cleaned-up nodes from Step 3 as the new basis for next cycle.
No more valid combinations or Termination Condition met
Y
End
N

Figure 7 | Construction Flowchart: L-DAG with Pre-Decomposition

The main construction workflow in Figure 7 is fundamentally the same as the L-DAG process in Figure 5, with the primary difference being an initial pre-decomposition step. However, due to this pre-decomposition, the construction process becomes significantly simpler, almost a mechanical, procedural routing procedure.

3.3.2 Phase 1: Decompose Constraints into Meta-Constraints

Decompose the composite constraints/clues into multiple meta-clues. Due to space constraints, only short-form clues are shown in Figure 8, each represented by keywords. For some complex clues, a full version is provided in brackets as an example.

Clue 0 (from the problem statement)

0.1 NO two people from the same country.
0.2 NO language is spoken by all 4 people.
0.3 NO 1st language is the same as his 2nd language

Clue 1. B cannot speak English, but can translate for A and C when they talk.

1.1 B is NOT UK
1.2 B is NOT English (1.2f: B's 2nd language is NOT English).
1.3 C is NOT Japanese

1.4 A and B share a common language.
(Full version 1.4f:
A's 1st language is B's 2nd language OR
B's 1st language is A's 2nd language OR
A's 2nd language is B's 2nd language).
1.5 A and C share no common language.
1.6 B and C share a common language.

Clue 2. A is Japanese, and D cannot speak Japanese, but they can communicate without difficulty
2.1 A is JP
2.2 B is NOT JP
2.3 C is NOT JP
2.4 D is NOT JP
2.5 D is NOT Japanese.
2.6 A and D share a common language.

Clue 3. Only one language is spoken by 3 people

Clue 4. No one among the four can speak both Japanese and French
4.1 No JP/French
4.2 No FR/Japanese

3.3.3 Phase 2 Construct L-DAG

Figure 8 shows a five-cycle construction process. The Construction (Step 2) and Elimination (Step 3) steps are combined to create a single mark-up node: Step 2 first performs all combinations, and Step 3 then eliminates the invalid possibilities. Step 4 displays the clean version of this node after the cycle is complete.

Due to space constraints:

- Clues from the problem statement (0.1–0.3) are treated as defaults and are not shown in the L-DAG; they are implicitly applied during the construction process.

- All clues do not have a TID; instead, they are pre-added into the task domain and ready for connection.

- Some steps are also not displayed. For example, in Cycle 1, D's possibility nodes for country and second language are omitted.

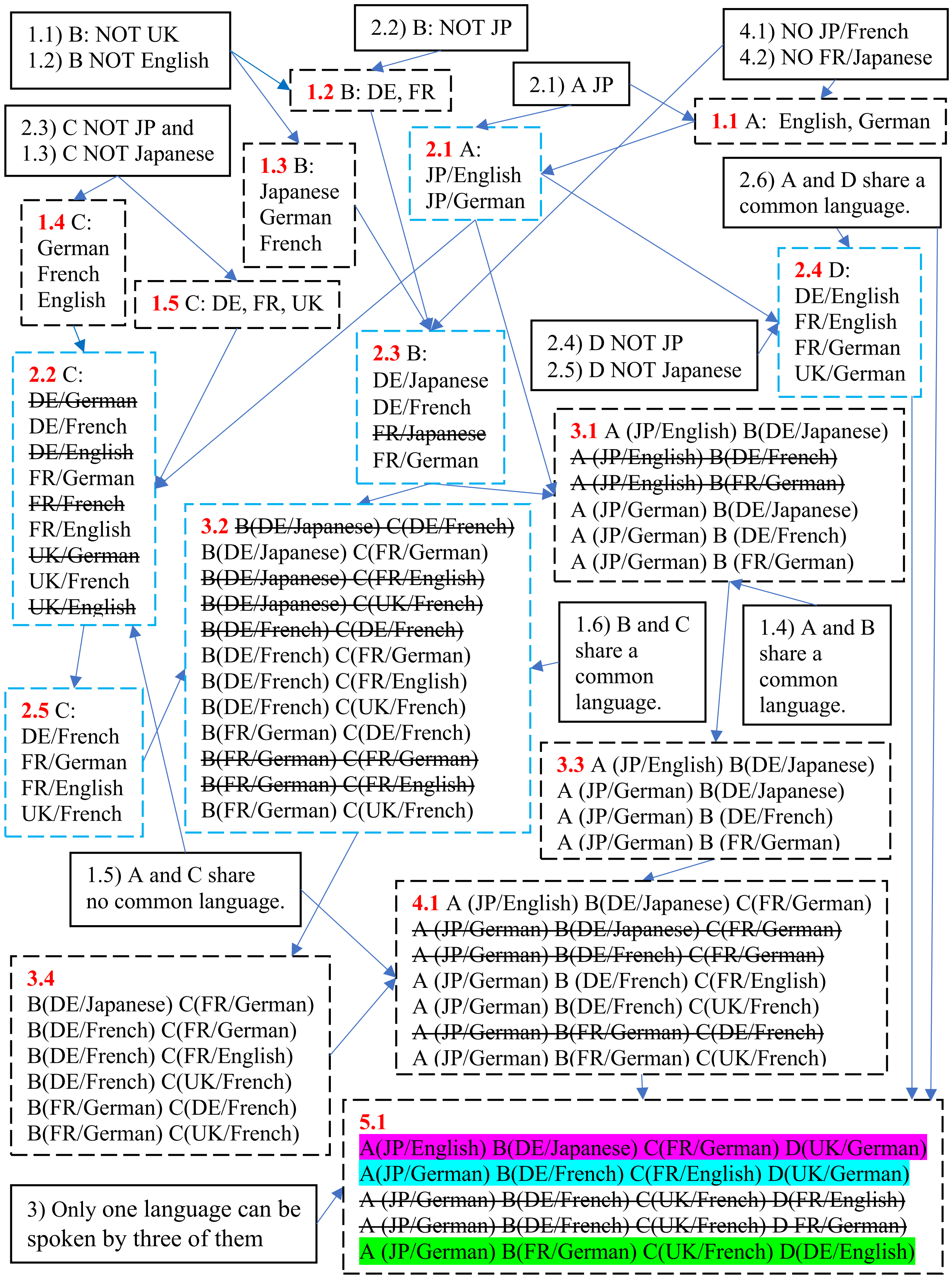

1.1) B: NOT UK
1.2) B NOT English
2.2) B: NOT JP
4.1) NO JP/French
4.2) NO FR/Japanese
1.2 B: DE, FR
2.1) A JP
1.1 A: English, German
2.3) C NOT JP and
1.3) C NOT Japanese
1.3 B: Japanese German French
2.1 A: JP/English JP/German
2.6) A and D share a common language.
1.4 C: German French English
1.5 C: DE, FR, UK
2.4 D: DE/English FR/English FR/German UK/German
2.2 C: ~~DE/German~~ DE/French ~~DE/English~~ FR/German ~~FR/French~~ FR/English ~~UK/German~~ UK/French ~~UK/English~~
2.3 B: DE/Japanese DE/French ~~FR/Japanese~~ FR/German
2.4) D NOT JP
2.5) D NOT Japanese
3.1 A (JP/English) B(DE/Japanese) ~~A (JP/English) B(DE/French)~~ ~~A (JP/English) B(FR/German)~~ A (JP/German) B(DE/Japanese) A (JP/German) B (DE/French) A (JP/German) B (FR/German)
3.2 ~~B(DE/Japanese) C(DE/French)~~ B(DE/Japanese) C(FR/German) ~~B(DE/Japanese) C(FR/English)~~ ~~B(DE/Japanese) C(UK/French)~~ ~~B(DE/French) C(DE/French)~~ B(DE/French) C(FR/German) B(DE/French) C(FR/English) B(DE/French) C(UK/French) B(FR/German) C(DE/French) ~~B(FR/German) C(FR/German)~~ ~~B(FR/German) C(FR/English)~~ B(FR/German) C(UK/French)
1.6) B and C share a common language.
1.4) A and B share a common language.
2.5 C: DE/French FR/German FR/English UK/French
3.3 A (JP/English) B(DE/Japanese) A (JP/German) B(DE/Japanese) A (JP/German) B (DE/French) A (JP/German) B (FR/German)
1.5) A and C share no common language.
4.1 A (JP/English) B(DE/Japanese) C(FR/German) ~~A (JP/German) B(DE/Japanese) C(FR/German)~~ ~~A (JP/German) B(DE/French) C(FR/German)~~ A (JP/German) B (DE/French) C(FR/English) A (JP/German) B(DE/French) C(UK/French) ~~A (JP/German) B(FR/German) C(DE/French)~~ A (JP/German) B(FR/German) C(UK/French)
3.4 B(DE/Japanese) C(FR/German) B(DE/French) C(FR/German) B(DE/French) C(FR/English) B(DE/French) C(UK/French) B(FR/German) C(DE/French) B(FR/German) C(UK/French)
5.1 A(JP/English) B(DE/Japanese) C(FR/German) D(UK/German) A(JP/German) B(DE/French) C(FR/English) D(UK/German) ~~A (JP/German) B(DE/French) C(UK/French) D(FR/English)~~ ~~A (JP/German) B(DE/French) C(UK/French) D FR/German)~~ A (JP/German) B(FR/German) C(UK/French) D(DE/English)
3) Only one language can be spoken by three of them

Figure 8 | L-DAG with Pre-Decomposition for Example 3

The following explanation walks through the five-cycle construction process shown in Figure 8, using Person C as the example and omitting Step 1 (Search). Person C is selected due to its higher complexity and larger number of constraints.

Cycle 1 (from 1.1 to 1.5)

Step 2: Construction

First construct basic possibility nodes for country and 2nd language. Node 1.4 is for C's possible countries and Node 1.5 is for C's possible 2nd languages.

Cycle 2 (from 2.1 to 2.5)

Step 2: Construction

Add Node 2.2 based on Nodes 1.4 and 1.5 for C.

Step 3: Elimination

- Using Clues 0.3, eliminate the possibilities in Node 2.2 where the 2nd language is the same as the 1st language (e.g., DE/German).
- Using Node 2.1 (A: JP/English & JP/German), eliminate the combination ~~DE/English~~ and ~~UK/German~~ in Node 2.2, because it violates Clue 1.5 (A and C share no common language).

Step 4: Clean-up

Create the cleaned Node 2.5 from marked-up Node 2.2.

Cycle 3 (from 3.1-3.4)

Step 2: Construction

Node 3.2 combines B from Node 2.3 with C from Node 2.5, yielding the combined possibilities of B and C; Node 3.1 combines A from Node 2.1 with B from Node 2.3, yielding the combined possibilities of A and B.

Step 3: Elimination

- Using Clue 0.5, eliminate combinations in Node 3.2 where two people are from the same country (e.g., ~~B(DE/Japanese) C(DE/French)~~).
- Using Clue 1.6, eliminate the combinations in Node 3.2 where B and C share no common language (e.g., ~~B(DE/Japanese) C(FR/English)~~).

Step 4: Clean-up

Create the cleaned Node 3.4 from marked-up node 3.2; and cleaned Node 3.3 from marked-up Node 3.1.

Cycle 4 (4.1)

Step 2: Construction

Add Node 4.1 by combining the possibilities for A and B in Node 3.3 with the possibilities for B and C in Node 3.4.

Step 3: Elimination

Using Clue 1.5 (A and C share no common language), eliminate combinations where A and C share a common language (e.g., ~~A (JP/German) B(DE/Japanese) C(FR/German~~)).

Step 4: Clean-up

The cleaned Node 4.2 is not displayed due to space constraints.

Cycle 5 (5.1)

Step 2: Construction

Add Node 5.1 by combining the possibilities for D in Node 2.4 with the combined possibilities for A, B and C in Node 4.1. Due to space limits, and using Clue 0.1 (No two people are from the same country), eliminate possibilities where D's country is the same as A's, B's, or C's country during the construction process.

Step 3: Elimination

- Using Clue 2.6 (A and D share a common language), eliminate the combination ~~A (JP/German) B(DE/French) C(UK/French) D(FR/English).~~
- Using Clue 3 (Only one language can be spoken by three of them), validate all combinations. Pink, Blue and Green pass the validation. ~~A (JP/German) B (DE/French) C (UK/French) D (FR/German)~~ is eliminated because two languages, French and German, are each spoken by three people.

The final three solutions are:

A(JP/English) B(DE/Japanese) C(FR/German) D(UK/German)

A(JP/German) B(DE/French) C(FR/English) D(UK/German)

A (JP/German) B(FR/German) C(UK/French) D(DE/English)

Step 4: Clean-up

The cleaned Node 5.2 is not displayed due to space constraints.

3.3.4 Features of L-DAG with Pre-decomposition

The process has the following features.

i. Less Deductive Reasoning in Pre-Decomposition than in L-DAG Construction

Although the pre-decomposition phase involves significantly less deep deductive reasoning than the L-DAG construction phase, it still requires decomposing clues into the finest possible meta-level granularity. For example, Meta Clue 1.4 in Figure 8 is shown in a simplified form ("A and B share a common language") due to space constraints. In the actual process, its full version should be used (Clue 1.4.f):

A's 1st language is B's 2nd language OR
B's 1st language is A's 2nd language OR
A's 2nd language is B's 2nd language.

If using its simplified meta constraint 1.4, we may make a mistake. For example, based on the meta constraint 4.1 (NO JP/French), we establish A's 2nd language options in Node 1.1 (A: English, German). If we apply the simplified meta constraint 1.4 (B and A share a common language) and meta constraint 1.2 (B NOT English), then we might erroneously eliminate English as A's 2nd language in Node 1.1. Additionally, the full version Clue 1.4f also clearly states that the constraint should only be applied after the Cycle 2 possibility nodes (Countries/Languages) are established which include all possible 1st and 2nd languages for both A and B.

Note: **Not all constraints can be decomposed into meta constraints.** For example, Clue 3, which has 48 possibilities (SOL), is not readily decomposed.

ii. Layered processing for avoiding premature constraint application caused incorrect possibilities construction and eliminating

- **Layered possibility construction: building from basic elements**

Possibility nodes must be constructed layer by layer, starting from the finest meta constraints to construct the most basic possibility nodes. For example, based on the lowest-level meta constraint 1.2 (B: NOT UK), we can construct B's 2nd language possibility in Node 1.3 as (B: Japanese, German, French), which includes all 3 possibilities. But if, at this stage, one were to also incorporate information from A's 2nd language possibility Node 1.1 (A: English, German) and the higher-level meta clue 1.4 (A and B share a common language), one may erroneously ignore Japanese as a possible 2nd language for B. Therefore, possibility nodes must be constructed layer by layer, beginning with the finest meta constraints to establish foundational possibility nodes without prematurely considering higher-level factors. The elements in possibility combinations may grow layer by layer.

- **Layered invalid possibility Elimination**
  - **Apply an N-component constraint only to possibility nodes in which all N components and their possibilities are fully extended.** This means a constraint involving specific entities or attributes should only be applied to possibility nodes in which the possibilities for all those entities/attributes have been fully extended under current conditions, as with the meta clue 1.4f example discussed in point (i) above. Another example: even though Clue 3 (Only one language is spoken by 3 people) is a set-based constraint with 48 possibilities, making it hard to decompose, in Cycle 5, after combinations involving all 4 individuals have been established, the number of relevant possibilities to check for Clue 3 is drastically reduced, from the initial 48 to just 5.

  - **Recheck constraints.** As with the direct L-DAG, some constraints may be applied multiple times. For example, Clue 1.5 is used twice (detailed in Cycle 2 Step 3 and Cycle 4 Step 3), because after more composite possibilities are established some previously valid possibilities may become invalid. To be safe, all constraints could potentially be rechecked in every cycle.

iii. Highly parallel routing processing

If strictly following the layered construction process with the finest granularized meta constraints, the process becomes almost a procedural routing task. Therefore, we can use more parallel processing than in the Direct L-DAG process. For example, we can eliminate the possibilities of two people being from the same country (e.g., A(JP/English) B(DE/Japanese) C(FR/German) D(DE/English)) during constructing Node 5.1. Alternatively, one might initially and somewhat procedural combine such invalid possibilities during the construction process, as with Node 3.2, and then eliminate them using additional parallel processing (e.g., ~~B(FR/German) C(FR/German)~~). Comparing Figure 8 with the direct L-DAG in Figure B shows that this approach is a much simpler routing process.

iv. Join Capability

In Figure 8, it can be seen that each cycle joins the nodes produced by the previous cycle:

- **Cycle 2** joins the people's country and their second language created in Cycle 1 for A, B, C, and D.
- **Cycle 3** joins entity relations: Node 3.1 is created by right joining Node 2.1 for A with Node 2.3 for B, and Node 3.2 results from right joining Node 2.3 for B with Node 2.5 for C.

- Cycle 4 joins A, B, and C: Node 4.1 is produced by right joining Node 3.3 for A and B with Node 3.4 for B and C.
- Cycle 5 performs the final join of A, B, C, and D: Node 5.1 results from right joining Node 4.1 for A, B, and C with Node 2.4 for D.

Each join has a strong logical connection and cannot be inconsistent.

4 L-DAG Applications

L-DAG can be used in the following three types of non-routing deductive reasoning tasks.

4.1 Complex DAG Structured Problem solving

Novel Type 1: Novel Form Problems

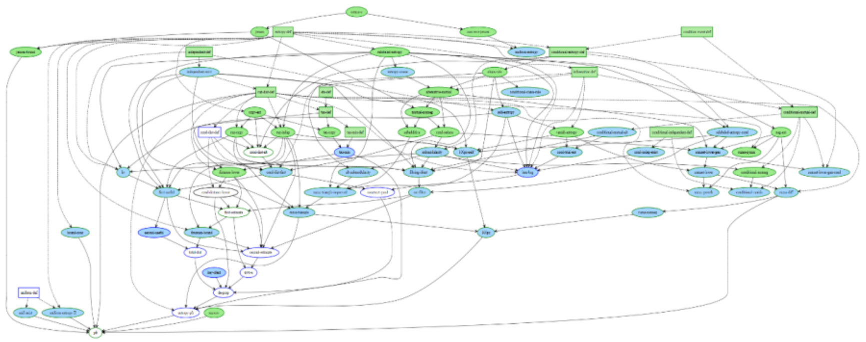
Example: Competition problems (e.g., IMO problems) and complex logic puzzles.

Novel Type 2: Novel (Open) Problems

These are problems with no known solution, demanding new research or theories. Example: Open problems in science and mathematics. L-DAG can assist researchers in exploring solution paths, identifying logical conflicts, and providing options for resolution.

Real-world research projects, such as the proof path of the Polynomial Freiman–Ruzsa conjecture, exhibit a clear directed acyclic graph (DAG) structure (Gowers et al., 2023; Tao, 2023). The proof's DAG consists of approximately 78 nodes: approximately 65 lemmas and 13 definitions.[7] For comparison, this paper's Example 1 forms a DAG of 61 nodes, where the solution is derived through 89 deductive steps (edges), reaching a maximum path depth of 18. Therefore, exploring a solution pathway for a real-world Novel Type 2 (Open) problem resembles solving a logical problem: deriving a solution from a network of logically connected constraints rather than following a linear Chain-of-Thought (CoT).

As shown in introductory Example A and Example 1, L-DAG can assist researchers in identifying critical bottlenecks, investigating issues, and exploring solution paths. This process has the potential to reduce years of exploration to a much shorter timeframe, thereby accelerating research and creating a "snowball effect" in addressing open problems.[8]



[7].

[8] For example, Zhang Yitang spent four years studying bounded gaps between primes. After three years, in July 2012, he had a critical insight that weakened a key condition, reducing the complexity of the proof and achieving a breakthrough. He described this as "fortunate to break through a distance as thin as a hair" (Zhang, 2014). In April 2013, he proved that there are infinitely many pairs of primes separated by at most 70 million (Beijing Spring, 2013). This breakthrough quickly led to further progress: in November 2013, James Maynard announced a new method that dramatically lowered the bound to 600 (Maynard, 2013), and in April 2014, the Polymath Project 8 reduced the bound

4.2 Exploration with Integrational Deductive Reasoning (basic AGI-0)

As scientific research becomes more complex and specialized, new discoveries now require either (a) highly specialized domain knowledge and expertise or (b) interdisciplinary integration. For individuals, possessing either attribute is already challenging, let alone possessing both.

L-DAG, with its Integrational Deductive Reasoning ability, can leverage AI's superhuman hardware capabilities to integrate vast and diverse data both within and across domains, and across modalities. It executes the entire reasoning process systematically and completely. It makes logical connections to uncover hidden clues, explore innovative insights, and propose potential solution pathways. This process assists humans in advancing science and technology.

- **Entity Association Task (EAT)**
  Examples:
  - **Create connections for innovation exploration.**
    For instance, in a chemical system with three entities A, B, and C, where each entity has a list of attributes:
    - Attribute 3 in list A is related to attribute 5 in list B
    - Attribute 4 in list B is related to attribute 2 in list C

    L-DAG can form a new chain (A3-B5-B4-C2) that links these related elements. This result may provide clues for applied innovations, particularly in biology and chemistry. It may also support exploratory discovery in broader scientific and technological contexts.

  - **Create the Data Correlation Density in its task domain for ECT**
    For instance, see Figure 4 in Example 2.

- **Entity Constraint Task (ECT)**

The vast majority of these nodes originate from the knowledge subnetwork (observations and facts) and the rule subnetwork (principles, rules, theorems, formulae, …) which is the underlying rules of the knowledges, reflecting that new discoveries are typically built upon established facts or principles.

As shown in Example 3, we can pre-decompose established elements in an AI rule system into smaller entities or meta-constraints. L-DAG then constructs logical connections (edges) among the nodes, as demonstrated in Examples 1, 2, and 3. It establishes parent-child logical dependencies among elements, discovers intermediate results (e.g., lemmas), and ultimately derives outcomes (e.g., new theorems) that satisfy the constraints imposed by their parent nodes.

4.3 L-DAG Features

Logical Directed Acyclic Graph (L-DAG), characterized by:

Global Dependency Management: Maintaining and tracking logical dependencies and consistency across network-structured levels.

to k≤246k (Polymath, 2014). This example illustrates how identifying precisely which constraint to weaken can trigger a rapid chain of advances—the kind of targeted exploration that L-DAG aims to assist.

Symbolic Logical Reasoning: Performing symbolic logical reasoning at each step using formal rules of inference to guarantee accuracy. Supports all forms of logical processes, including First-Order Logic (quantification over individuals), Second-Order Logic (quantification over sets, relations, or functions), and mathematical logic for proposition proving.

Topological Order–Enforced Iterative Dynamic Construction: Arriving at a final solution iteratively through the searching and adding constraint nodes → constructing possibility nodes → eliminating invalid possibilities process.

Searching Prioritization Strategies: Guiding the search for constraints and next seeds by prioritizing entities with high Win-Rate in the L-DAG.

Conflict Resolution Support: Identifying and providing options for resolving conflicts, ambiguities, redundancies, and insufficient constraints.

Timely Self-Reward Mechanism: Enhancing search and construction efficiency, and preventing combinatorial explosion by making function calls to calculate or estimate the possibility before processing a node.

Parallel Processing: Applying parallel processing in forward and backward (pruning of invalid branches) in both search and construction phases.

Integrational Deductive Reasoning: Enabling scientific exploration through Entity Constraint Task and Entity Relation Task processes.

5 Summary

L-DAG

L-DAG is a globally dependency-managed, symbolic logical reasoning parallel process, equipped with efficient search prioritization strategies and a timely self-reward mechanism. It enables AI systems to perform deductive reasoning to solve complex and novel reasoning tasks, and to apply Integrational Deductive Reasoning to explore novel insights and potential solution pathways. This assists human scientific and technological advancement.

Future Development

The examples in this article are just a few of the applications that are worth exploring. More applications, especially those involving real-world problems, are expected.

Capabilities and Integrative Use with Other Approaches

L-DAG is essentially a rule-based programming process, making it naturally suited to integrate with other approaches and tools in flexible ways. For example:

- Domain expertise can be configured as rules to guide exploration, as shown in Example 1, where established mathematical logic principles were applied.
- Formalization pipelines can be combined, such as those used in AlphaProof with Lean, to convert problems into machine-verifiable representations before deductive processing, as demonstrated in Example 3.

- Additional tools can be integrated as needed, including advanced constraint decomposers, automated verifiers, visualization modules, or new components developed to support complex deductive reasoning workflows.

By combining these capabilities, L-DAG can serve as a powerful deductive reasoning layer for tackling a wide range of tasks, especially complex challenges.

White-Box AI Controls Black-Box AI

L-DAG is a white-box component. A white-box AI is interpretable and human-controllable and can be used to control black-box AI systems and enforce their alignment with human values.

References

F. Barez, et al., "Chain-of-Thought Is Not Explainability, " https://www.alphaxiv.org/abs/2025.02.

Beijing Spring, "Zhang Yitang's breakthrough," 2013. Available at: http://beijingspring.com/bj2/2010/240/201369191603.htm.

T.H. Cormen, et al., *Introduction to Algorithms*. Fourth Edition. The MIT Press, 2022.

DeepMind AlphaProof and AlphaGeometry Teams. "AI Achieves Silver-Medal Standard Solving International Mathematical Olympiad Problems." DeepMind Blog, 2024. https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

W. T. Gowers, et al., "On a conjecture of Marton," *arXiv preprint arXiv:2311.05762*, Nov. 2023. https://arxiv.org/abs/2311.05762

J. Maynard, "Small gaps between primes," *arXiv preprint*, arXiv:1311.4600, 2013. Available at: https://arxiv.org/abs/1311.4600.

P. Mondorf and B. Plank, "Comparing Inferential Strategies of Humans and Large Language Models in Deductive Reasoning." *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9370–9402, August 11–16, 2024. ©2024 Association for Computational Linguistics.

A. Novikov, et al. "AlphaEvolve: A Coding Agent for Scientific and Algorithmic Discovery." *arXiv preprint*, 2025. https://arxiv.org/pdf/2506.13131

D. H. J. Polymath, "Variants of the Selberg sieve, and bounded gaps between primes," *Research in the Mathematical Sciences*, vol. 1, no. 1, pp. 1–83, 2014. DOI: 10.1186/s40687-014-0012-7.

D. Silver, et al., "Mastering the Game of Go Without Human Knowledge." *Nature*, vol. 550, 2017, pp. 354–359.

T. Tao, "Formalizing the proof of PFR in Lean4 using Blueprint: a short tour," *Terry Tao's Blog*, Nov. 2023. https://terrytao.wordpress.com/2023/11/18/formalizing-the-proof-of-pfr-in-lean4-using-blueprint-a-short-tour/

T. H. Trinh, et al., "Solving Olympiad Geometry Without Human Demonstrations." *Nature*, vol. 625, Feb. 2024, pp. 476–482.

J. Wei, et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." *arXiv preprint*, Jan. 2023.

Y. Zhang, et al., "Bounded gaps between primes," *Annals of Mathematics*, vol. 179, no. 3, pp. 1121–1174, 2014. DOI: 10.4007/annals.2014.179.3.7.

A
Pruning Process for Example 2

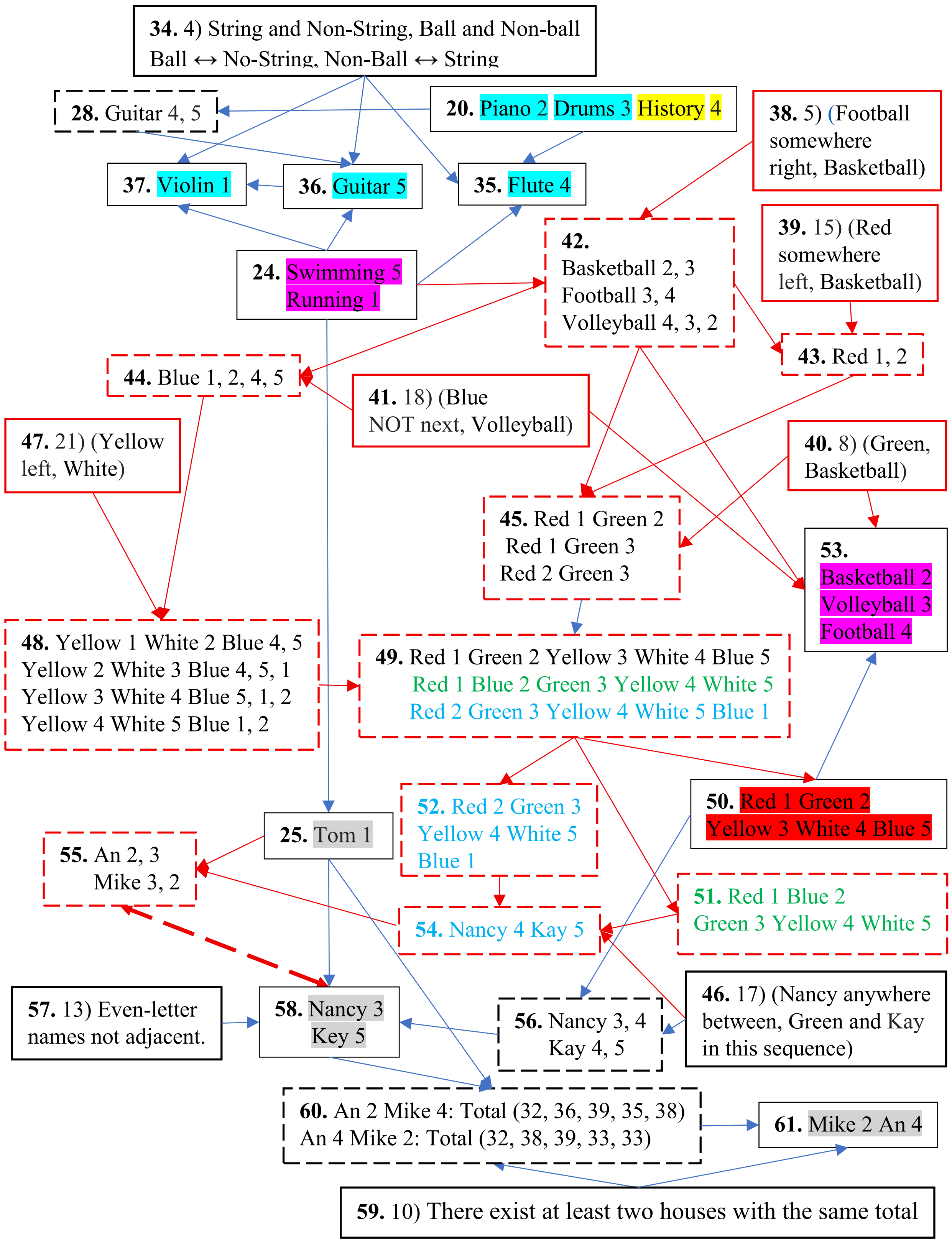
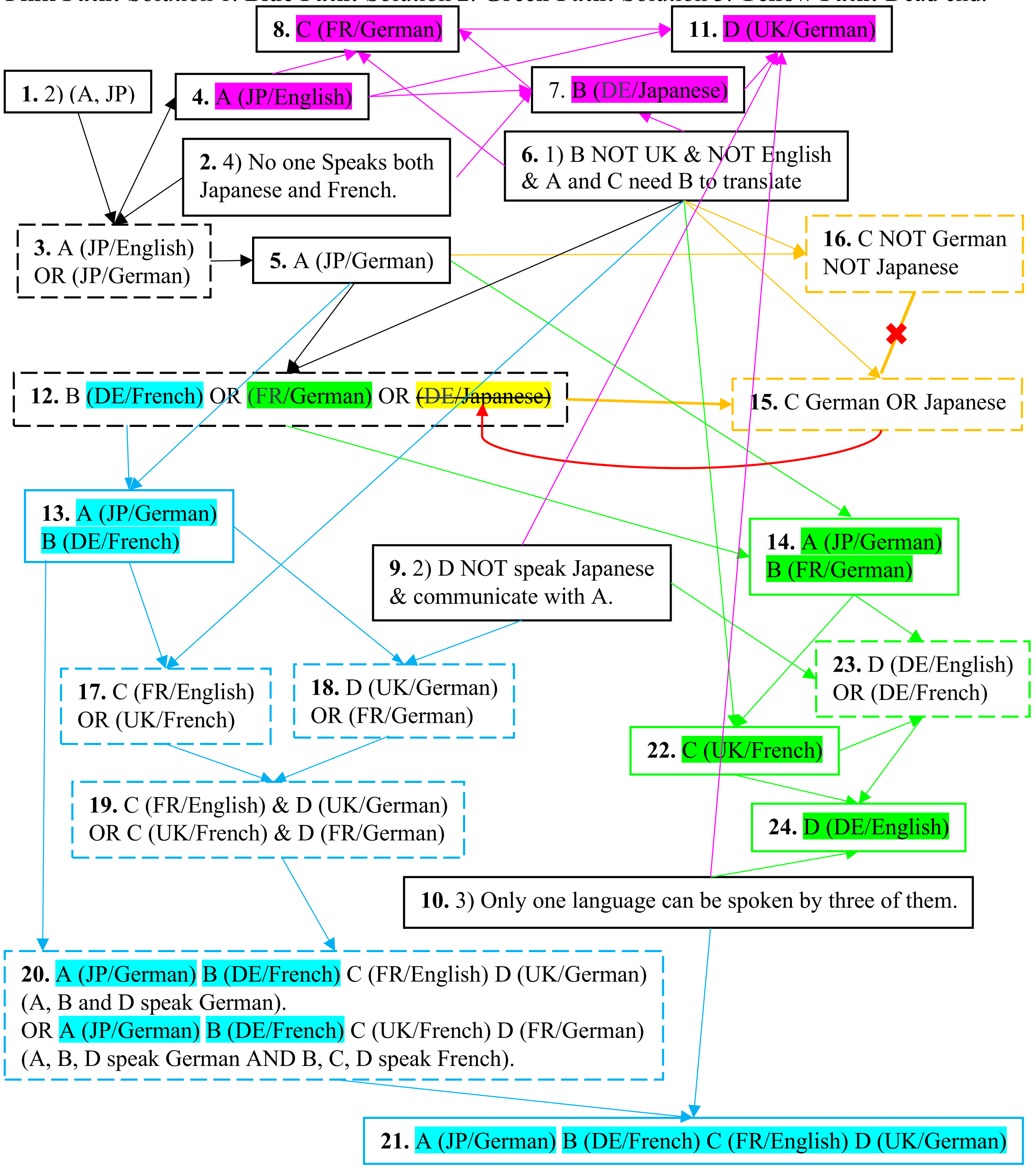

34. 4) String and Non-String, Ball and Non-ball
Ball ↔ No-String, Non-Ball ↔ String
28. Guitar 4, 5
20. Piano 2 Drums 3 History 4
38. 5) (Football somewhere right, Basketball)
37. Violin 1
36. Guitar 5
35. Flute 4
39. 15) (Red somewhere left, Basketball)
24. Swimming 5 Running 1
42. Basketball 2, 3 Football 3, 4 Volleyball 4, 3, 2
43. Red 1, 2
44. Blue 1, 2, 4, 5
41. 18) (Blue NOT next, Volleyball)
47. 21) (Yellow left, White)
40. 8) (Green, Basketball)
45. Red 1 Green 2 Red 1 Green 3 Red 2 Green 3
53. Basketball 2 Volleyball 3 Football 4
48. Yellow 1 White 2 Blue 4, 5
Yellow 2 White 3 Blue 4, 5, 1
Yellow 3 White 4 Blue 5, 1, 2
Yellow 4 White 5 Blue 1, 2
49. Red 1 Green 2 Yellow 3 White 4 Blue 5
Red 1 Blue 2 Green 3 Yellow 4 White 5
Red 2 Green 3 Yellow 4 White 5 Blue 1
25. Tom 1
52. Red 2 Green 3 Yellow 4 White 5 Blue 1
50. Red 1 Green 2 Yellow 3 White 4 Blue 5
55. An 2, 3 Mike 3, 2
54. Nancy 4 Kay 5
51. Red 1 Blue 2 Green 3 Yellow 4 White 5
57. 13) Even-letter names not adjacent.
58. Nancy 3 Key 5
56. Nancy 3, 4 Kay 4, 5
46. 17) (Nancy anywhere between, Green and Kay in this sequence)
60. An 2 Mike 4: Total (32, 36, 39, 35, 38)
An 4 Mike 2: Total (32, 38, 39, 33, 33)
61. Mike 2 An 4
59. 10) There exist at least two houses with the same total

Figure A | L-DAG Pruning Process for Example 2
(The pruned nodes and edges are highlighted in red)

B

L-DAG Application to Example 3 (Without Pre-Decomposition)

Pink Path: Solution 1. **Blue Path:** Solution 2. **Green Path:** Solution 3. **Yellow Path:** Dead end.



1. 2) (A, JP)
2. 4) No one Speaks both Japanese and French.
3. A (JP/English) OR (JP/German)
4. A (JP/English)
5. A (JP/German)
6. 1) B NOT UK & NOT English & A and C need B to translate
7. B (DE/Japanese)
8. C (FR/German)
9. 2) D NOT speak Japanese & communicate with A.
10. 3) Only one language can be spoken by three of them.
11. D (UK/German)
12. B (DE/French) OR (FR/German) OR ~~(DE/Japanese)~~
13. A (JP/German) B (DE/French)
14. A (JP/German) B (FR/German)
15. C German OR Japanese
16. C NOT German NOT Japanese
17. C (FR/English) OR (UK/French)
18. D (UK/German) OR (FR/German)
19. C (FR/English) & D (UK/German) OR C (UK/French) & D (FR/German)
20. A (JP/German) B (DE/French) C (FR/English) D (UK/German) (A, B and D speak German). OR A (JP/German) B (DE/French) C (UK/French) D (FR/German) (A, B, D speak German AND B, C, D speak French).
21. A (JP/German) B (DE/French) C (FR/English) D (UK/German)
22. C (UK/French)
23. D (DE/English) OR (DE/French)
24. D (DE/English)

Figure B | L-DAG for Example 3