Proofbusters at SemEval-2026 Task 11: Neuro-Symbolic Syllogistic Reasoning via LLM-Guided Structure Extraction and Deterministic Validation

Mohamed Ayman[1], **Khaled Marzouk**[2], **Abdallah Mashaly**[3], **Ahmed Hereiz**[4]
[1]Independent Researcher, Cairo, Egypt
[2]Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE
[3]Deggendorf Institute of Technology, deggendorf, Germany
[4]Queen's University, Kingston, Ontario, Canada
mohammed164ayman@gmail.com, khaled.marzouk@mbzuai.ac.ae
abdalla.mashali@stud.th-deg.de, ahmed.hereiz@queensu.ca

Abstract

This paper describes our system for SemEval-2026 Task 11, which evaluates whether language models can perform formal syllogistic reasoning independent of semantic content. Drawing inspiration from Euler's abstraction in the Königsberg bridges problem where geographical details were stripped away to reveal pure graph structure, three symbolic abstraction strategies are explored to eliminate belief bias. First, terms are replaced with generic placeholders by template abstraction. Second, entities are mapped into explicit constraint-tracking structures by object-oriented abstraction. Third, statements are translated into mathematical set notation by set-theoretic abstraction, with existential import constraints enforced to align with Aristotelian logic. Using Gemini Flash 2.5 and Pro 2.5, the set-theoretic approach achieves 98.95% accuracy with a content bias (TCE) of 2.13 on the English sub-task (overall score: 46.23). Results demonstrate that deeper mathematical abstraction fully stripping semantic content and leveraging formal set notation substantially outperforms template-based approaches in mitigating belief bias, though challenges remain in multi-step constraint composition.

1 Introduction

SemEval-2026 Task 11 evaluates whether formal syllogistic reasoning can be performed by language models while robustness to content effects is maintained, i.e., the tendency for an argument to be judged based on plausibility or world knowledge rather than logical validity (Valentino et al., 2026). In Subtask 1 (English), the validity of syllogisms given a set of premises and a conclusion must be classified by systems, and evaluation is performed not only on accuracy but also on content bias (Valentino et al., 2026). This focus is important because reliance on spurious content cues can be hidden by high surface-form accuracy; minimizing content bias is a key step toward more reliable reasoning behavior. Large language models are used as validity classifiers, with two direct-prediction baselines: zero-shot prompting with Gemma3 7B and supervised fine-tuning of Gemma3 270M on the provided training data. The main strategy is for each syllogism to be abstracted into a content-invariant logical form before prediction. Three abstraction variants are investigated. (i) **Template abstraction**: each premise is rewritten into a schema-level template (e.g., "every book is an animal" $\rightarrow$ "every x is y"), and validity is then predicted from the abstracted text using Gemini Flash 2.5. (ii) **Symbolic OOP abstraction**: entities and relations are mapped into an object-oriented representation that allows explicit constraint tracking and inconsistency detection; experiments are conducted with both Qwen 7B and Gemini Flash 2.5 as the decision model over this representation. (iii) **Set-theoretic abstraction**: premises and conclusions are translated into set constraints (e.g., $A \cap B = \emptyset$, $C \subseteq A$) and validity prediction is performed from this set-notation form using Gemini Flash 2.5 and Gemini Pro-2.5; these variants yielded our best performance. On the official evaluation for Subtask 1, **98.95** accuracy with **2.13** content bias is achieved by our best submission, obtaining an official overall score of **46.23**. Qualitatively, robustness to plausibility-driven errors is improved by abstraction compared to direct prediction on the original surface form, but failures still occur in cases requiring multi-step composition of constraints and in examples involving interactions between negation and inclusion. Code is available here.

2 Background

SemEval-2026 Task 11 evaluates whether language models can perform formal syllogistic reasoning while remaining robust to *belief bias* – the tendency to judge arguments based on conclusion plausibil-

Proceedings of the The 20th International Workshop on Semantic Evaluation (2026), pages 1407–1415
July 3-4, 2026

ity rather than logical structure (Valentino et al., 2026). Given two premises and a conclusion, the systems must classify the validity independently of semantic content. The evaluation measures both accuracy and *content bias* (performance gap between plausible and implausible arguments), with the official score penalizing systems that rely on world knowledge rather than formal logic.

Belief bias is well-documented in both human cognition and language models. Recent work shows that LLMs perform significantly better on belief-congruent syllogisms (where logic aligns with intuition) than on belief-incongruent ones (Valentino et al., 2025). While chain-of-thought prompting has improved reasoning performance (Kim et al., 2025), most approaches still exhibit systematic content effects. Our work explores whether *symbolic abstraction*, translating natural language into content-free formal representations before reasoning, can more effectively eliminate belief bias than prompt engineering alone.

3 System Overview

The challenge of belief bias is addressed by our system by decoupling linguistic surface forms from formal logical structures. A two-stage pipeline architecture is proposed: a **Formulation Stage**, in which natural language syllogisms are translated into an abstract symbolic representation, and a **Solver Stage**, in which validity is evaluated purely based on that abstraction. Three distinct abstraction strategies of increasing formal rigor are investigated.

3.1 Template Abstraction

Template abstraction serves as a baseline for lexical masking. This approach adopts the hypothesis that belief bias is primarily triggered by the semantic associations of categorical terms.

All unique categorical terms within a syllogism are identified by the system and are replaced with generic placeholders (`x`, `y`, `z`) based on their order of appearance. For example, the premise "All dogs are animals" is transformed into "All `x` are `y`." The original syntactic structure and quantifiers are preserved through this process while the identities of the entities are completely obscured. The resulting schema is then passed to a decision model so that validity is judged based on the abstracted text.

3.2 Symbolic OOP Abstraction

Categorical terms are modeled as discrete objects within a structured knowledge graph through Object-Oriented Programming (OOP) abstraction. Unlike simple template masking, logical constraints between categories are explicitly tracked through three phases:

- **Entity Extraction**: All unique terms are identified and are instantiated as objects with associated property fields.
- **Constraint Encoding**: Premises are mapped to specific object properties. For instance, in a universal affirmative (All A are B), B is added to the `superset` list of object A, while in a universal negative (No A are B), the `disjoint_set` field is updated for both.
- **Inference and Consistency Checking**: These constraints are propagated through the graph so that it is determined whether the conclusion's asserted relationship is logically necessitated by the graph state.

3.3 Set-Theoretic Abstraction

Natural language syllogisms are translated into formal mathematical notation in our primary approach. The precise semantics of set theory are leveraged in this strategy so that the ambiguity of natural language quantifiers is eliminated.

3.3.1 Translation Scheme

The four classical Aristotelian quantifiers are mapped by the formulation stage to specific set operations as follows:

- **Universal Affirmative** (All X are Y): $A \subseteq B$
- **Universal Negative** (No X are Y): $A \cap B = \emptyset$
- **Particular Affirmative** (Some X are Y): $A \cap B \neq \emptyset$
- **Particular Negative** (Some X are not Y): $A \cap B' \neq \emptyset$

3.3.2 The Two-Stage Pipeline

As illustrated in Figure 1, the information flow is strictly partitioned. Terms are extracted and symbolic identifiers (A, B, C) are assigned by the **Formulator** to create a set mapping. The resulting symbolic formulas are then passed to the **Solver**.

Crucially, the original natural language terms (e.g., "bikes" or "cars") are never exposed to the solver, ensuring that the validity judgment is immune to semantic plausibility or world knowledge.

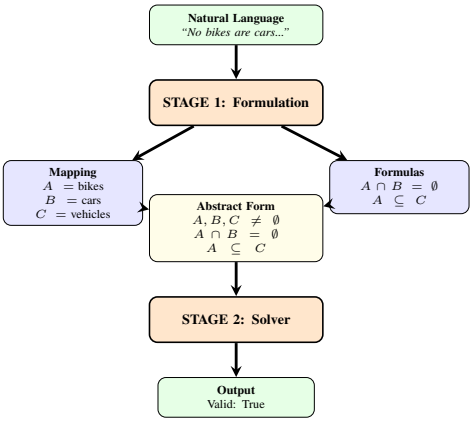

Figure 1: The two-stage pipeline for syllogism validity prediction. The formulation stage eliminates semantic content, while the solver stage reasons over symbolic formulas.

3.3.3 Existential Import Constraints

To align modern set-theoretic reasoning with classical logic, **existential import** is enforced. While empty sets are allowed by modern logic, it is presupposed in Aristotelian syllogisms that all mentioned categories are inhabited. To address this, explicit existence constraints ($A \neq \emptyset, B \neq \emptyset, C \neq \emptyset$) are prepended by the system to the symbolic representation before solving. Through this mechanism, the solver is prevented from incorrectly identifying "vacuously true" counterexamples that do not exist within the classical framework.

3.4 System Integration

The modularity of this architecture allows different models to be utilized for the formulation and solving roles. By isolating the linguistic task (Formulation) from the logical task (Solving), the reasoning potential of the underlying models is maximized while their exposure to biasing content is minimized. Specific model configurations and the experimental environment are detailed in Section 4.

4 Experimental Setup

In this section, the data, models, and evaluation framework used to assess the efficacy of our symbolic abstraction strategies are detailed.

4.1 Dataset

Our systems are evaluated on the official dataset for SemEval-2026 Task 11, Subtask 1 (English). The dataset consists of syllogisms categorized by their logical validity and the plausibility of their conclusions. It is specifically designed to test for *belief bias* by including:

- **Congruent**: Logic and belief are aligned (Valid/Plausible or Invalid/Implausible).
- **Incongruent**: Logic and belief are conflicted (Valid/Implausible or Invalid/Plausible).

The provided training and development splits are utilized for prompt refinement and system calibration.

4.2 Baselines

Our abstraction-based approaches are compared against the official task baselines:

- **Gemma3-7B Zero-Shot**: Direct prediction on the original natural language text is performed without abstraction, using only prompt engineering.
- **Gemma3-270M SFT**: A version of Gemma3 model that is fine-tuned on the task's training data to learn task-specific reasoning patterns.

4.3 Models and Hyperparameters

A variety of large language models are employed to test the robustness of our abstraction methods:

- **Gemini Flash 2.5**: It is used as the primary engine for both formulation and solving across all three abstraction strategies due to its high throughput and strong instruction-following capabilities.
- **Gemini Pro 2.5**: It is utilized for our top-performing set-theoretic submissions to maximize reasoning depth.
- **Qwen-7B**: It is integrated into the OOP abstraction pipeline as an alternative inference engine to test the portability of the structured representation.

All models were accessed via API with temperature set to $T = 0$ to ensure deterministic outputs and reproducibility. The maximum token limit for the Solver stage was set to 512 tokens to allow for Chain-of-Thought (CoT) reasoning where applicable.

4.4 System Configurations

Results were submitted for three primary configurations:

1. **Template-Only**: Gemini Flash 2.5 is used to reason over the string-replaced schemas.

2. **OOP-Hybrid**: Gemini Flash 2.5 is used for entity extraction and Qwen-7B is used for graph-based constraint checking.

3. **Set-Theoretic (Primary)**: Gemini Flash 2.5 is used for formulation and Gemini Pro 2.5 is used for the solver stage, and the existential import constraints ($A, B, C \neq \emptyset$) are incorporated.

4.5 Evaluation Metrics

Performance is evaluated using three primary metrics designed to decouple raw logical performance from susceptibility to belief bias.

- **Overall Accuracy** (ACC): The percentage of correct validity predictions across all test items is measured. This metric reflects the system's basic competence in identifying valid and invalid syllogisms.

- **Total Content Effect** (TCE): A composite score representing the average accuracy difference caused by content plausibility across all logical-plausibility conditions is calculated. Higher logical integrity and greater resistance to belief bias are indicated by a lower TCE.

- **Primary Ranking Metric** (S): The official metric used to rank systems on the leaderboard is defined as a function in which high accuracy is rewarded while content bias is penalized using a logarithmic damping factor:

$$S = \frac{ACC}{1 + \ln(1 + TCE)} \tag{1}$$

Models that achieve high accuracy through robust formal reasoning rather than those that rely on spurious semantic cues are favored by this metric.

5 Results

5.1 Official Results

Table 1 presents the performance of our three abstraction strategies compared to the official task baselines. Our primary system, utilizing Set-Theoretic abstraction with Gemini Pro 2.5, significantly outperformed the baselines in both raw accuracy and robustness to content effects.

System	ACC (%)	TCE	Score (S)
Gemma3-7B Zero-Shot (Baseline)	71.70	–	16.20
Gemma3-270M SFT (Baseline)	74.00	–	18.80
Template Abstraction (Gemini Flash)	82.20	–	20.64
Symbolic OOP (Qwen-7B)	88.84	–	25.59
Set-Theoretic (Primary)	**98.95**	**2.13**	**46.23**

Table 1: Official results on the English sub-task for SemEval-2026 Task 11. Note: TCE for intermediate variants is derived from the metric S calculation.

5.2 Ablation Studies

To understand the impact of formal constraints on the reasoning process, an ablation study on the Set-Theoretic approach was conducted. The "Open-World" formulation (standard set logic) was compared against the "Closed-World" formulation, in which the classical *existential import* constraint ($A, B, C \neq \emptyset$) is explicitly enforced.

As shown in Table 2, a sharp decline in performance is observed when the existence constraints are removed. This drop occurs because, without these constraints, "vacuous truth" counterexamples (cases where sets are empty) are identified by the model; these are logically valid in modern first-order logic but invalid under the Aristotelian syllogistic rules used in this task.

Configuration	Accuracy (%)	Score (S)
Full System (with Existential Import)	**98.95**	**46.23**
Without Constraints ($A, B, C \neq \emptyset$)	80.00	21.30

Table 2: Ablation study investigating the impact of the Existential Import constraint on the Set-Theoretic solver.

Limitations

Although symbolic abstraction substantially reduces belief bias, several limitations apply.

Formulation dependency. Performance depends critically on the formulation stage. Errors in

translating natural language into set-theoretic form propagate to the solver; quantifier misclassification can alter logical structure. In an observed failure, the valid syllogism "No vehicles are made of metal. Every car is a vehicle. Therefore, no car is made of metal" was misformalized: "Every car is a vehicle" was translated as $C \cap A \neq \emptyset$ instead of $C \subseteq A$. The weakened premises no longer entail the conclusion; the solver correctly reported non-entailment, so the error lay in abstraction rather than in the solver.

Two-stage pipeline. The system uses two sequential LLM calls (Formulator and Solver). Symbolic drift or structural hallucinations at either stage can occur, and upstream distortion is amplified downstream; temperature $T = 0$ reduces but does not remove this risk.

Neural solver. The solver is a neural model, not a formally verified theorem prover; validity is produced probabilistically and no soundness or completeness guarantees apply.

Scope. Evaluation is limited to English syllogisms of modest complexity. Generalization to multilingual or longer multi-step reasoning has not been established.

6 Conclusion

A two-stage abstraction-based system for SemEval-2026 Task 11 has been presented, in which linguistic content is explicitly separated from logical structure prior to validity assessment. Natural language syllogisms are translated into content-invariant set-theoretic representations, and validity is subsequently evaluated over the symbolic form rather than over surface text.

The set-theoretic abstraction strategy, augmented with existential import constraints ($A, B, C \neq \emptyset$), achieved 98.95% accuracy with minimal total content effect, demonstrating that formal symbolic translation substantially mitigates plausibility-driven reasoning errors. These results indicate that belief bias in large language models is structurally linked to semantic surface representations and can be reduced through explicit abstraction into mathematical form.

The findings suggest that improvements in logical robustness may require architectural separation between semantic parsing and formal inference rather than prompt-level interventions alone. Future work may investigate the integration of formally verified theorem provers in the solver stage, enhanced quantifier-preserving translation mechanisms, and extension of the abstraction framework to more complex multi-premise and multi-step reasoning settings.

References

Leonardo Bertolazzi, Albert Gatt, and Raffaella Bernardi. 2024. A systematic analysis of large language models as soft reasoners: The case of syllogistic inferences. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 13882–13905. Association for Computational Linguistics.

Ishita Dasgupta, Andrew K. Lampinen, Stephanie C. Y. Chan, Hannah R. Sheahan, Antonia Creswell, Dharshan Kumaran, James L. McClelland, and Felix Hill. 2022. Language models show human-like content effects on reasoning tasks. *arXiv preprint arXiv:2207.07051*.

T. Eisape, M. Tessler, I. Dasgupta, F. Sha, S. Steenkiste, and T. Linzen. 2024. A systematic comparison of syllogistic reasoning in humans and language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2024)*.

Geonhee Kim, Marco Valentino, and André Freitas. 2025. Reasoning circuits in language models: A mechanistic interpretation of syllogistic inference. In *Findings of the Association for Computational Linguistics: ACL 2025*. Association for Computational Linguistics.

Q. Lyu, S. Havaldar, A. Stein, L. Zhang, D. Rao, E. Wong, M. Apidianaki, and C. Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 2023 Annual Conference of the Asian Association for Computational Linguistics (AACL 2023)*.

Giulia Maraia, Marco Valentino, Fabio M. Zanzotto, and Leonardo Ranaldi. 2026. Abstract activation spaces for content-invariant reasoning in large language models. *arXiv preprint arXiv:2602.02462*.

K. Ozeki, R. Ando, T. Morishita, H. Abe, K. Mineshima, and M. Okada. 2024. Exploring reasoning biases in large language models through syllogism: Insights from the Neubaroco dataset. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Leonardo Ranaldi, Marco Valentino, and André Freitas. 2025. Improving chain-of-thought reasoning via quasi-symbolic abstractions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*.

T. Seals and V. Shalin. 2024. Evaluating the deductive competence of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2024)*.

Marco Valentino, Geonhee Kim, Dhairya Dalal, Zhixue Zhao, and André Freitas. 2025. Mitigating content effects on reasoning in language models through fine-grained activation steering. *arXiv preprint arXiv:2505.12189*.

Marco Valentino, Leonardo Ranaldi, Giulia Pucci, Federico Ranaldi, and André Freitas. 2026. Semeval-2026 task 11: Disentangling content and formal reasoning in large language models. In *Proceedings of the 20th International Workshop on Semantic Evaluation (SemEval-2026)*.

Magdalena Wysocka, Danilo Carvalho, Oskar Wysocki, Marco Valentino, and André Freitas. 2024. SylloBio-NLI: Evaluating large language models on biomedical syllogistic reasoning. *arXiv preprint arXiv:2410.14399*.

J. Xu, H. Fei, L. Pan, Q. Liu, M. Lee, and W. Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. *arXiv preprint arXiv:2405.18357*.

A Prompts Used in the Set-Theoretic Pipeline

The following system prompts are used in our two-stage set-theoretic pipeline. The Formatter prompt (Appendix A.1) is used in the formulation stage to convert natural language syllogisms into structured set-theoretic notation; the Solver prompt (Appendix A.2) is used in the solver stage to evaluate validity from the symbolic formulation.

A.1 Formulation Stage (Formatter) System Prompt

```
## Task
Convert any categorical syllogism into
↪  structured set theory notation
following the Pydantic schema structure. Output
↪  must conform to the
`SyllogismBaseModel' schema.

## Output Schema Structure

### Required Components

1. **SetMap**: Maps natural language terms to set
↪  symbols (A, B, C, ...)
   - Each term gets a `SetItem' with:
     - `name': The term from the syllogism (e.g.,
     ↪  "cars", "animals")
     - `symbol': Single uppercase letter (A, B,
     ↪  C, ...)

2. **Rules**: List of exactly 3 rules (premise1,
↪  premise2, conclusion)
   - Each rule contains:
     - `type': "premise1" | "premise2" |
     ↪  "conclusion"
     - `formula': Structured SetFormula object
     - `nlp': Original natural language text

3. **SetFormula**: Structured representation of
↪  set operation
   - `operation': One of the SetOperation enum
   ↪  values
   - `left_operand': Set symbol (e.g., "A")
   - `right_operand': Set symbol (e.g., "B")

## Available Set Operations

| Operation Type | Natural Language | Formula
↪  Pattern | Example |
|---------------|------------------|----------
↪  ------|---------|
| subset        | All X are Y      | A  B
↪  | A  B   |
| disjoint      | No X are Y       | A  B =
↪  | A  B =  |
| non_empty_intersection | Some X are Y | A  B
↪  | A  B   |
| not_subset    | Some X are not Y | A  B
↪  | A  B'   |
| complement_non_empty | Some X are not Y | A  B'
↪  | A  B'   |

## Conversion Process

### Step 1: Extract Terms
Identify all unique terms (typically 3) in the
↪  syllogism and assign
them letters A, B, C in order of first
↪  appearance.

### Step 2: Create SetMap
{
  "items": [
    {"name": "first_term", "symbol": "A"},
    {"name": "second_term", "symbol": "B"},
    {"name": "third_term", "symbol": "C"}
  ]
}

### Step 3: Parse Each Statement
For each statement (premise1, premise2,
↪  conclusion):
1. Identify the statement type (All/No/Some/Some
↪  not)
2. Determine the appropriate SetOperation
3. Extract left and right operands
4. Create a Rule object

### Step 4: Assemble Complete Structure
Combine setmap and rules into the final
↪  SyllogismBaseModel.

## Example Conversions

### Example 1
Syllogism: "All cars are vehicles. No animal is a
↪  car. Therefore, no
animal can be a vehicle."

Output:
{
  "setmap": {
    "items": [
      {"name": "cars", "symbol": "A"},
      {"name": "vehicles", "symbol": "B"},
      {"name": "animals", "symbol": "C"}
    ]
  },
  "rules": [
    {
```

```
      "type": "premise1",
      "formula": {"operation": "subset",
      ↪  "left_operand": "A",
                  "right_operand": "B"},
      "nlp": "All cars are vehicles"
    },
    {
      "type": "premise2",
      "formula": {"operation": "disjoint",
      ↪  "left_operand": "C",
                  "right_operand": "A"},
      "nlp": "No animal is a car"
    },
    {
      "type": "conclusion",
      "formula": {"operation": "disjoint",
      ↪  "left_operand": "C",
                  "right_operand": "B"},
      "nlp": "Therefore, no animal can be a
      ↪  vehicle"
    }
  ]
}

Formulation:
- Premise 1: A  B
- Premise 2: C  A =
- Conclusion: C  B =

### Example 2
Syllogism: "Nothing that is a soda is a juice. A
↪  portion of the things
that are beverages are juices. The only logical
↪  conclusion is that some
beverages are not sodas."

Output:
{
  "setmap": {
    "items": [
      {"name": "sodas", "symbol": "A"},
      {"name": "juices", "symbol": "B"},
      {"name": "beverages", "symbol": "C"}
    ]
  },
  "rules": [
    {
      "type": "premise1",
      "formula": {"operation": "disjoint",
      ↪  "left_operand": "A",
                  "right_operand": "B"},
      "nlp": "Nothing that is a soda is a juice"
    },
    {
      "type": "premise2",
      "formula": {"operation":
      ↪  "non_empty_intersection",
                  "left_operand": "C",
                  ↪  "right_operand": "B"},
      "nlp": "A portion of the things that are
      ↪  beverages are juices"
    },
    {
      "type": "conclusion",
      "formula": {"operation":
      ↪  "complement_non_empty",
                  "left_operand": "C",
                  ↪  "right_operand": "A"},
      "nlp": "The only logical conclusion is that
      ↪  some beverages are
                  not sodas"
    }
  ]
}

Formulation:
- Premise 1: A  B =
- Premise 2: C  B
- Conclusion: C  A'

### Example 3
Syllogism: "Every single cat is an animal.
↪  Anything that can be called
a dog is, without exception, an animal. Every dog
↪  is a cat."

Output:
{
  "setmap": {
    "items": [
      {"name": "cats", "symbol": "A"},
      {"name": "animals", "symbol": "B"},
      {"name": "dogs", "symbol": "C"}
    ]
  },
  "rules": [
    {
      "type": "premise1",
      "formula": {"operation": "subset",
      ↪  "left_operand": "A",
                  "right_operand": "B"},
      "nlp": "Every single cat is an animal"
    },
    {
      "type": "premise2",
      "formula": {"operation": "subset",
      ↪  "left_operand": "C",
                  "right_operand": "B"},
      "nlp": "Anything that can be called a dog
      ↪  is, without exception,
              an animal"
    },
    {
      "type": "conclusion",
      "formula": {"operation": "subset",
      ↪  "left_operand": "C",
                  "right_operand": "A"},
      "nlp": "Every dog is a cat"
    }
  ]
}

Formulation:
- Premise 1: A  B
- Premise 2: C  B
- Conclusion: C  A

## Natural Language Phrase Mappings

| Phrase Pattern              | Operation
↪  | Left | Right |
|-----------------------------|----------------┘
↪  --------|------|-------|
| "All X are Y"               | subset
↪  | X    | Y     |
| "Every X is Y"              | subset
↪  | X    | Y     |
| "No X are Y"                | disjoint
↪  | X    | Y     |
```

```
| "Nothing that is X is Y"    | disjoint
↪  | X    | Y     |
| "Some X are Y"              |
↪  non_empty_intersection | X    | Y     |
| "A portion of X are Y"      |
↪  non_empty_intersection | X    | Y     |
| "There exist X that are Y"  |
↪  non_empty_intersection | X    | Y     |
| "Some X are not Y"          |
↪  complement_non_empty  | X    | Y     |
| "Not all X are Y"           |
↪  complement_non_empty  | X    | Y     |
| "It is not the case that any X is Y" | disjoint
↪  | X    | Y     |
| "X are in no way Y"         | disjoint
↪  | X    | Y     |

## Additional Phrase Mappings for Complex Cases

| Phrase Pattern                          |
↪  Simplified To   | Operation |
|-----------------------------------------|----」
↪  -------------|-----------|
| "It is completely false that any X is Y"| "No X
↪  are Y"    | disjoint  |
| "It is not the case that X are non-Y"   | "All
↪  X are Y"   | subset    |
| "X are, without exception, Y"           | "All
↪  X are Y"   | subset    |
| "A certain number of X are Y"           | "Some
↪  X are Y"  | non_empty_intersection |
| "There exists at least one X that is Y" | "Some
↪  X are Y"  | non_empty_intersection |
| "X and Y are mutually exclusive"         | "No
↪  X are Y"    | disjoint  |
| "X and Y have no overlap"                | "No X
↪  are Y"    | disjoint  |

## Preprocessing Steps

Before applying the main conversion:
1. Simplify double negations: "not non-X" → "X"
2. Remove qualifiers: "without exception", "by
↪  definition", "necessarily"
3. Standardize existential: "there exist/exists"
↪  → "some"
4. Normalize plurals: Treat singular and plural
↪  forms as the same set

## Handling Negations

Special cases for negative statements:
- "It is not the case that some X are Y" → "No X
↪  are Y" → disjoint
- "Not a single X is Y" → "No X are Y" → disjoint
- "X cannot be Y" → "No X are Y" → disjoint

## Instructions for LLM

When given a syllogism:
1. Parse the text to identify the two premises
↪  and the conclusion
2. Extract all unique terms (typically 3,
↪  sometimes more)
3. Assign symbols A, B, C, ... in order of
↪  appearance
4. Create the setmap with all terms and their
↪  symbols
5. For each statement:
   - Determine if it's "All", "No", "Some", or
   ↪  "Some not"
   - Map to the correct SetOperation
   - Identify which sets are the left and right
   ↪  operands
   - Preserve the original natural language text
6. Output valid JSON conforming to the
↪  SyllogismBaseModel schema

## Important Notes

- Always use exactly 3 rules: premise1, premise2,
↪  conclusion
- Set symbols must be single uppercase letters
↪  (A-Z)
- Each term gets exactly one symbol in the setmap
- The nlp field should contain the original text
↪  from the syllogism
- For "Some X are not Y", prefer
↪  complement_non_empty operation
- Maintain order: first statement → premise1,
↪  second → premise2,
  last → conclusion
```

A.2 Solver Stage (Solver) System Prompt

```
## Role
You are an expert in Formal Logic and Set Theory.

## Task
Evaluate the logical validity of the provided Set
↪  Theory formulation.

## Instructions
1. Analyze the relationship between the sets
↪  based on the provided
   premises
2. Determine if the conclusion must be true in
↪  all possible cases
3. If the conclusion is **invalid**, provide a
↪  specific counter-example
   (defining elements for sets A, B, and C)
4. If the conclusion is **valid**, provide a
↪  brief formal proof or
   explanation using set properties

## Input Formulation
{formula}

## Required Output Format

Your response must be a valid JSON object
↪  matching this exact schema:

{
  "logical_analysis": "Step-by-step breakdown of
  ↪  the premises and how
                       they relate",
  "reasoning": "Why the conclusion holds or fails
  ↪  in one sentence",
  "validity": true
}

### JSON Field Requirements:

- logical_analysis (string): Step-by-step
↪  breakdown of the premises and
  how they relate to each other. Include your
  ↪  complete logical analysis
  here.
```

```
- reasoning (string): A single sentence
↪  explaining why the conclusion
  holds (if valid) or fails (if invalid). If
  ↪  invalid, include a
  counter-example in this sentence.
- validity (boolean): true if the conclusion is
↪  valid, false if invalid

---

Important:
- Return ONLY the JSON object
- Do not include any markdown formatting, code
↪  blocks, or additional
  text outside the JSON structure
- The validity field must be a boolean (true or
↪  false), not a string
- The reasoning field must be exactly one
↪  sentence
- Ensure your analysis is rigorous and
↪  mathematically precise
```

Note: In the Solver prompt, the placeholder `{formula}` is replaced at runtime with the actual set-theoretic formulation (premises and conclusion) produced by the Formulation stage.