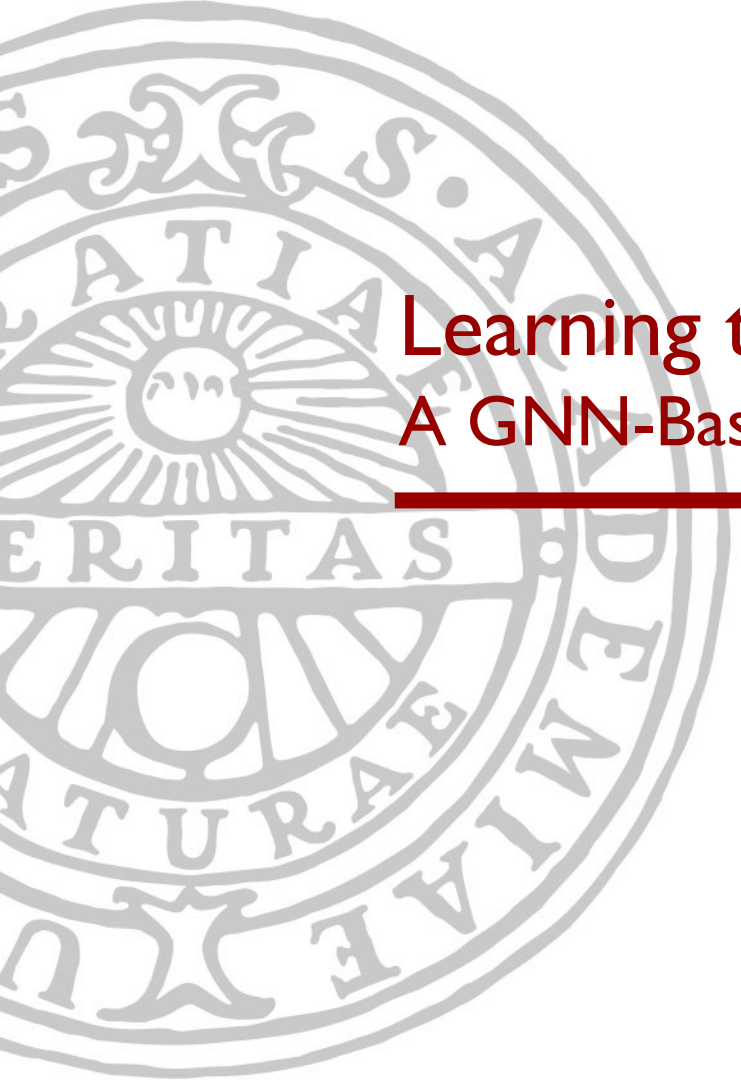


Learning to Guide Automated Reasoning:
A GNN-Based Framework

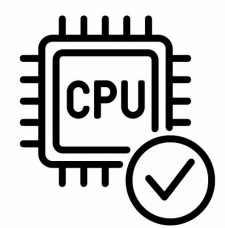
6 May 2025

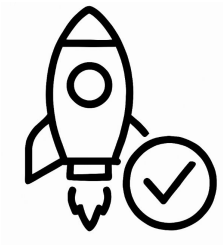
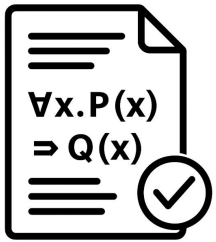
Doctoral Student	Chencheng Liang	
Supervisors	Philipp Rümmer, Parosh Aziz Abdulla, Yi Wang	
Opponent	Stephan Schulz	DHBW Stuttgart
Jury Members	Konstantin Korovin	University of Manchester
	Mihaela Sighireanu	ENS Paris-Saclay
	Christian Rohner	Uppsala University
	Tobias Wrigstad	Uppsala University

Symbolic Methods

- Background
  - Applications
    - Verify complex systems.
    - Theorem proving.
    - Constraint solving in optimization.

  - Symbolic expressions effectively encode large sets of states and transitions.



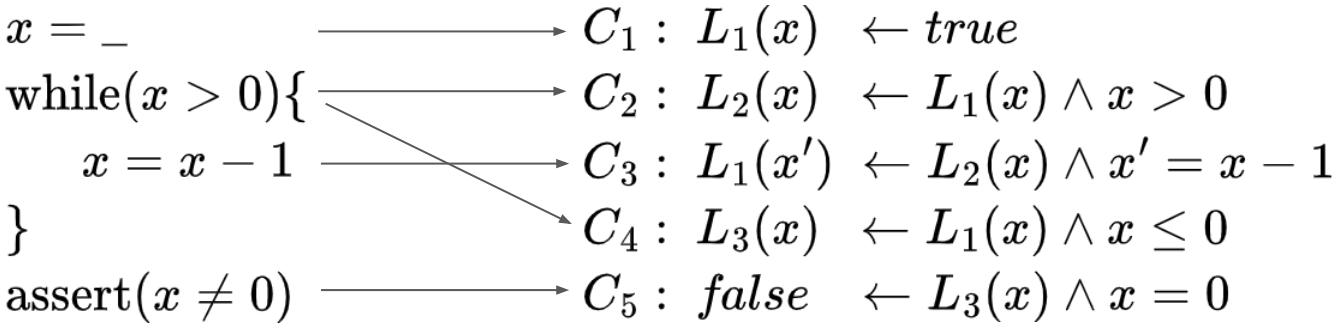



Symbolic Methods

- Fundamental techniques in automated reasoning.
  - Deductive reasoning, term rewriting, constraint solving, etc.

- Approaches
  - Automated Theorem Provers (ATPs).
  - Boolean Satisfiability (SAT)/Satisfiability Modulo Theories (SMT) solvers.
  - Constrained Horn Clause (CHC) solvers.

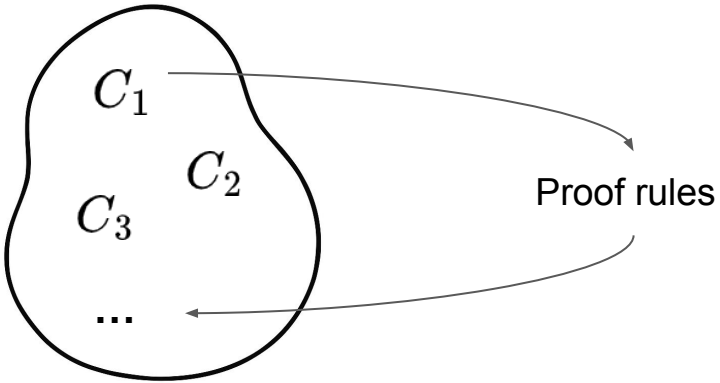
Motivating Examples (Program Verification)



Motivating Examples (Program Verification)

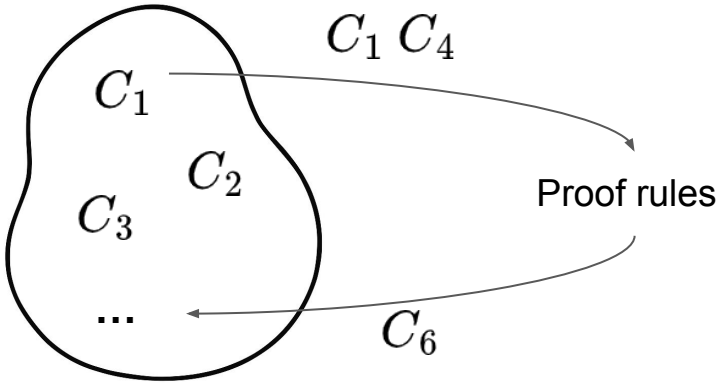
$x = \_$
$\text{while}(x > 0)\{$
$\quad x = x - 1$
$\}$
$\text{assert}(x \neq 0)$

$$
\begin{array}{lll}
C_1: & L_1(x) & \leftarrow true \\
C_2: & L_2(x) & \leftarrow L_1(x) \wedge x > 0 \\
C_3: & L_1(x') & \leftarrow L_2(x) \wedge x' = x - 1 \\
C_4: & L_3(x) & \leftarrow L_1(x) \wedge x \leq 0 \\
C_5: & false & \leftarrow L_3(x) \wedge x = 0
\end{array}
$$



Motivating Examples

$$
\begin{aligned}
C_1 &: L_1(x) \leftarrow true \\
C_2 &: L_2(x) \leftarrow L_1(x) \wedge x > 0 \\
C_3 &: L_1(x') \leftarrow L_2(x) \wedge x' = x - 1 \\
C_4 &: L_3(x) \leftarrow L_1(x) \wedge x \leq 0 \\
C_5 &: false \leftarrow L_3(x) \wedge x = 0
\end{aligned}
\qquad \longrightarrow \quad C_6 : L_3 \leftarrow true \wedge x \leq 0
$$



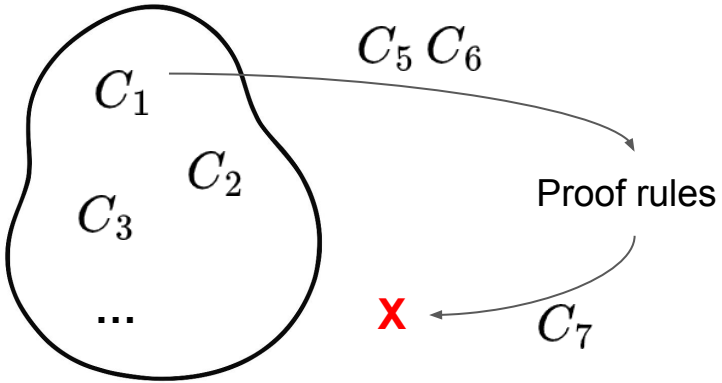
Motivating Examples

$$C_1 : L_1(x) \leftarrow true$$
$$C_2 : L_2(x) \leftarrow L_1(x) \wedge x > 0$$
$$C_3 : L_1(x') \leftarrow L_2(x) \wedge x' = x - 1$$
$$C_4 : L_3(x) \leftarrow L_1(x) \wedge x \leq 0$$
$$C_5 : false \leftarrow L_3(x) \wedge x = 0$$

$C_6 : L_3 \leftarrow true \wedge x \leq 0$

$C_7 : false \leftarrow x \leq 0 \wedge x = 0$

$false \leftarrow true$



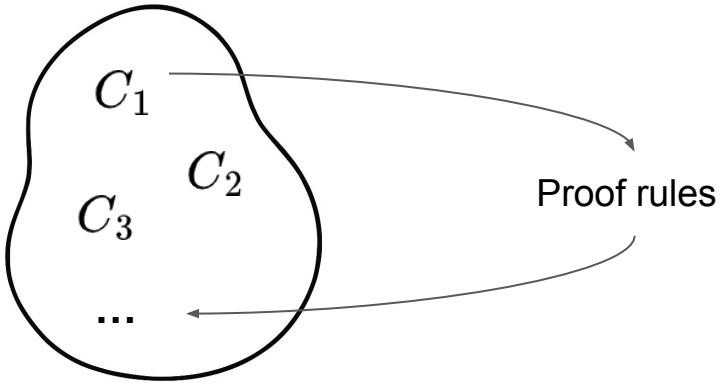
Motivating Examples

$$C_1 : L_1(x) \leftarrow true$$
$$C_2 : L_2(x) \leftarrow L_1(x) \wedge x > 0$$
$$C_3 : L_1(x') \leftarrow L_2(x) \wedge x' = x - 1$$
$$C_4 : L_3(x) \leftarrow L_1(x) \wedge x \leq 0$$
$$C_5 : false \leftarrow L_3(x) \wedge x = 0$$

$$C_6 : L_3 \leftarrow true \wedge x \leq 0$$

$$C_7 : false \leftarrow x \leq 0 \wedge x = 0$$

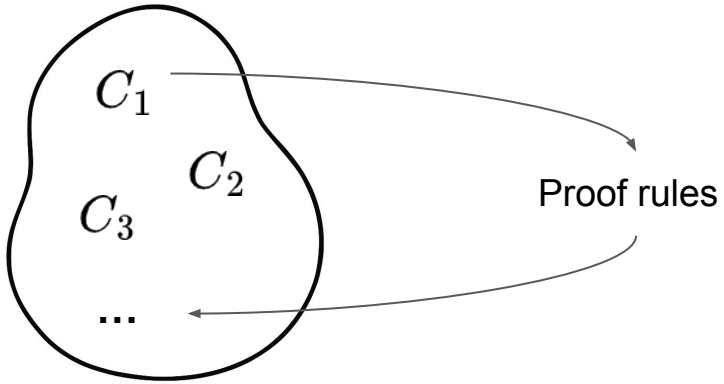
$$false \leftarrow true$$



Key challenge: pick correct clauses

Motivating Examples (Theorem Proving)

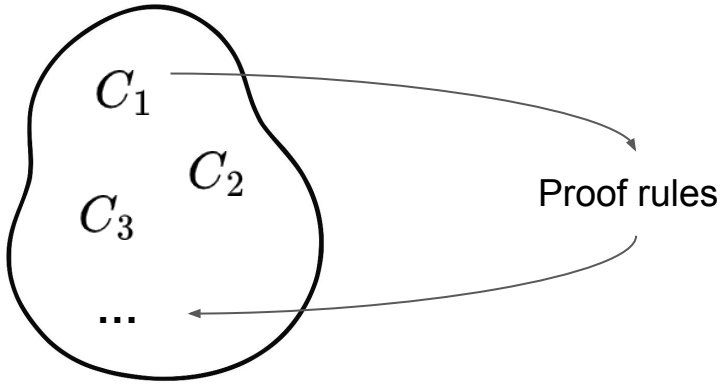
- Need heuristics to pick clauses
  - Predefined features [1]:
    - Clause age.
    - Clause size.
    - Relevance to the proof goal.



[1] S. Schulz and M. Möhrmann, “Performance of clause selection heuristics for saturation-based theorem proving,”2016.

Motivating Examples (Theorem Proving)

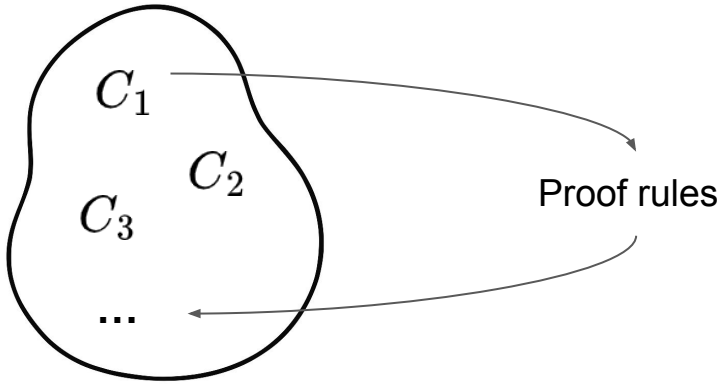
- Need heuristics to pick clauses
  - Predefined features [1]:
    - Clause age.
    - Clause size.
    - Relevance to the proof goal.
  - Learned abstract features:
    - Usually non-linear functions.
    - Data-driven.



[1] S. Schulz and M. Möhrmann, “Performance of clause selection heuristics for saturation-based theorem proving,”2016.

Motivating Examples (Theorem Proving)

- Need heuristics to pick clauses
  - Predefined features [1]:
    - Clause age.
    - Clause size.
    - Relevance to the proof goal.
  - Learned abstract features:
    - Usually non-linear functions.
    - Data-driven.

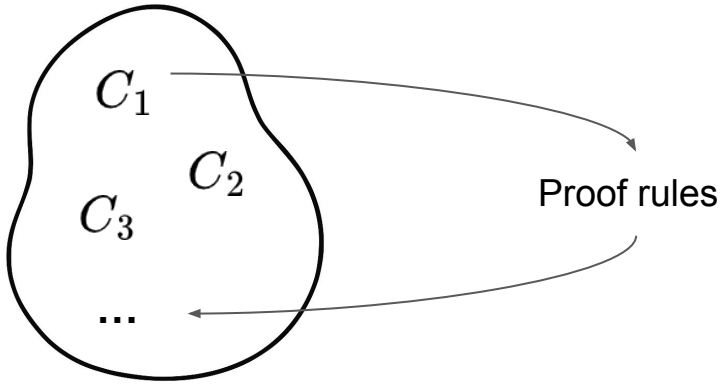


- Deepire [2] uses recursive neural networks to classify the clauses based on their derivation history.

[2] M. Suda, “Vampire with a brain is a good ITP hammer,” 2021.

Motivating Examples (Theorem Proving)

- Need heuristics to pick clauses
  - Predefined features [1]:
    - Clause age.
    - Clause size.
    - Relevance to the proof goal.
  - Learned abstract features:
    - Usually non-linear functions.
    - Data-driven.



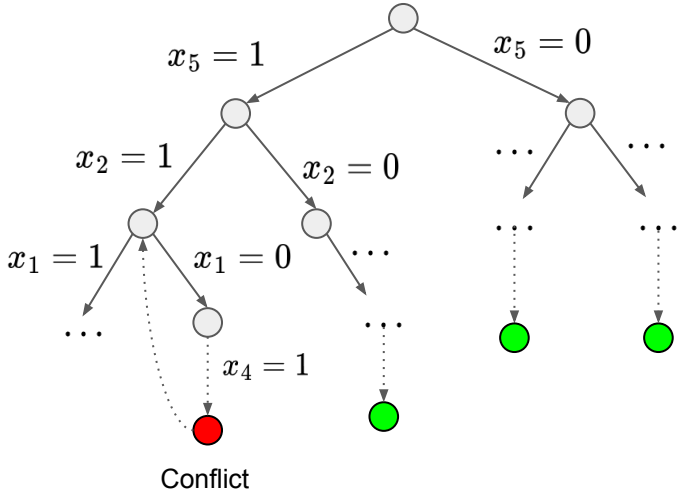
- ENIGMA [3] uses both Gradient Boosting Decision Trees (GBDTs) and Graph Neural Networks (GNNs) to select the clauses.

[3] Jakub˚uv, K. Chvalovský, M. Olšák, B. Piotrowski, M. Suda, and J. Urban, "ENIGMA anonymous: Symbol-independent inference guiding machine (system description)," 2020.

Motivating Examples (SAT Solving)

- Clauses

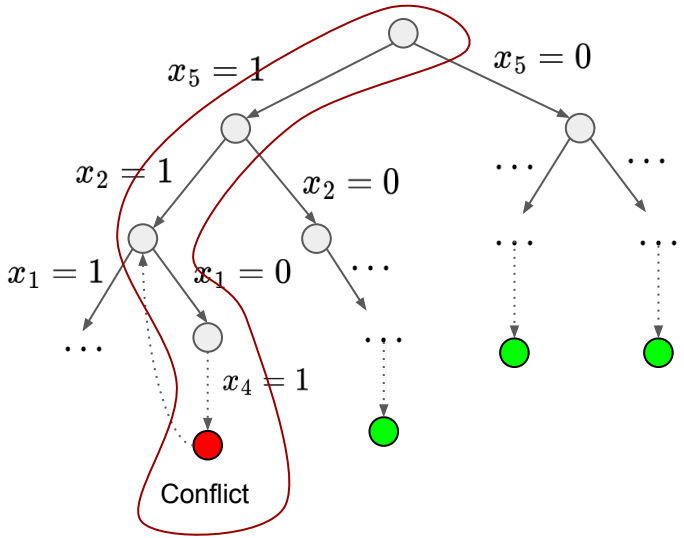
$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$$
$$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4)$$



Motivating Examples (SAT Solving)

- Clauses

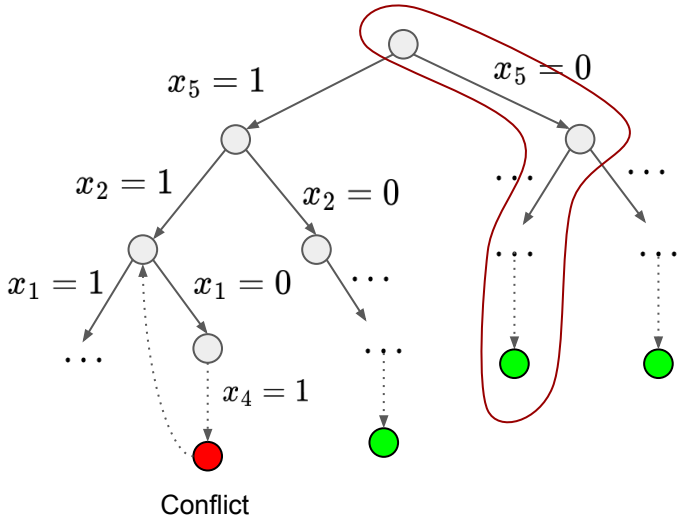
$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$$
$$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4)$$



Motivating Examples (SAT Solving)

- Clauses

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$$
$$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4)$$

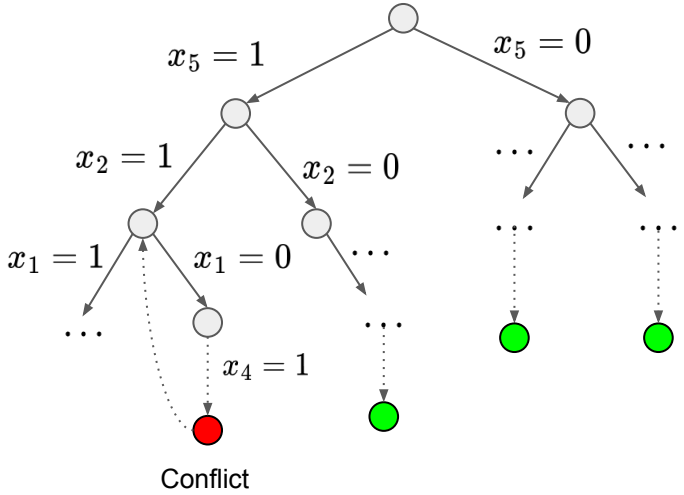


Motivating Examples (SAT Solving)

- Clauses

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$$
$$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4)$$

- SAT Solver
  - Selecting an variable for branching.
  - Which branch to go.
  - Deciding when to restart.

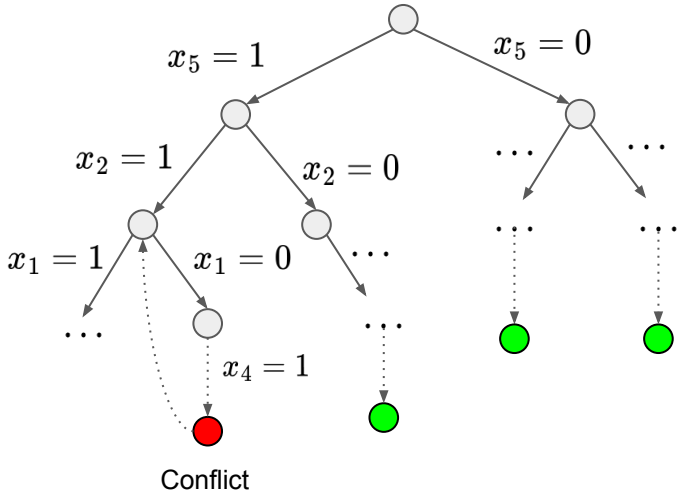


Key challenge: find the correct path

Motivating Examples (SAT Solving)

- Conflict-Driven Clause Learning (CDCL)-based SAT solving
  - Selecting an variable for branching.
  - Which branch to go.
  - Deciding when to restart.

- NeuroSAT [4] periodically resets Exponential Variable State-Independent Decaying Sum (EVSIDS) scores based on the predictions of a message-passing neural network.

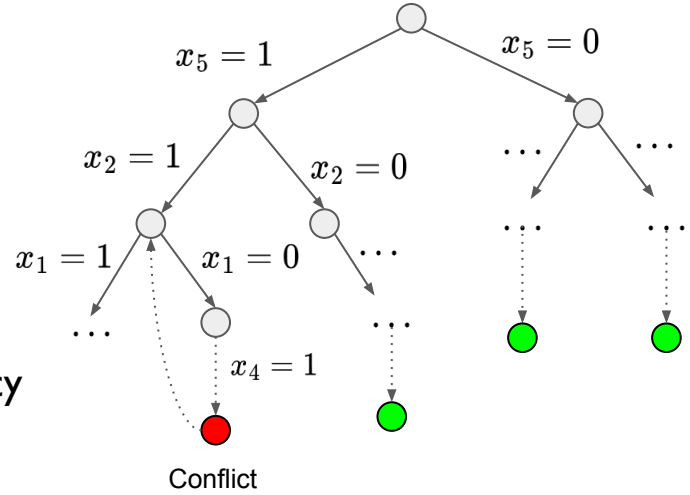


[4] D. Selsam and N. Bjørner, “Guiding high-performance SAT solvers with unsat-core predictions,” 2019.

Motivating Examples (SAT Solving)

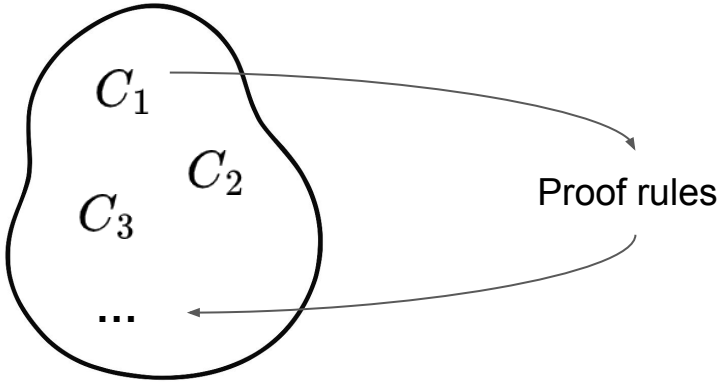
- Conflict-Driven Clause Learning (CDCL)-based SAT solving
  - Selecting an variable for branching.
  - Which branch to go.
  - Deciding when to restart.

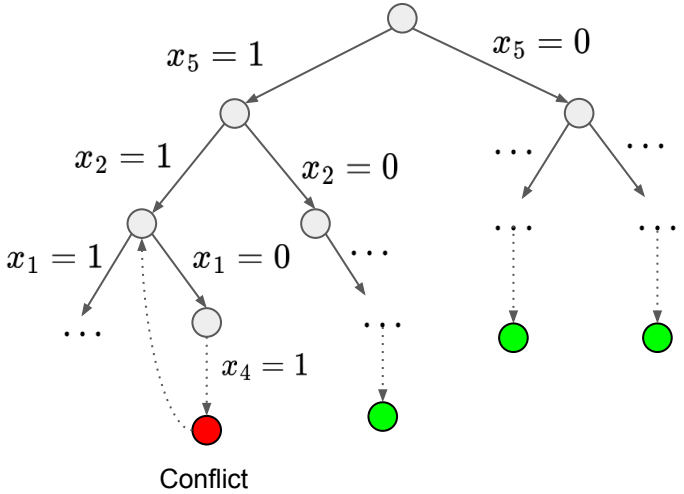
- Authors in [5] propose a data-driving-based restart policy by analyzing the history of previously learned clauses.



[5] J. H. Liang, C. Oh, M. Mathew, C. Thomas, C. Li, and V. Ganesh, “Machine learning-based restart policy for CDCL SAT solvers,” 2018.

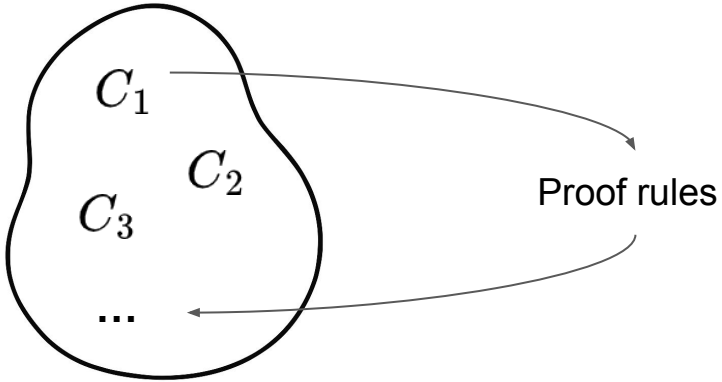
Summary of Motivating Examples

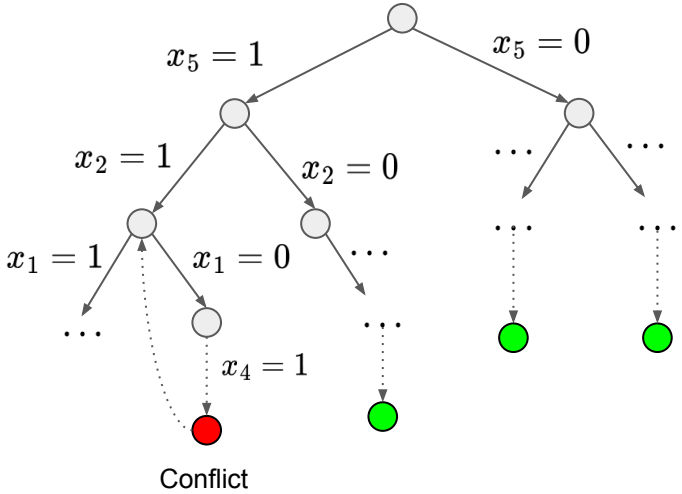




- Nearly all approaches of symbolic methods incorporate multiple heuristic-driven decision processes.

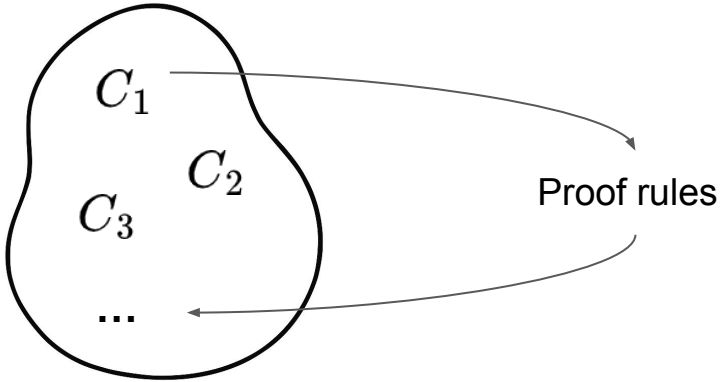
Summary of Motivating Examples

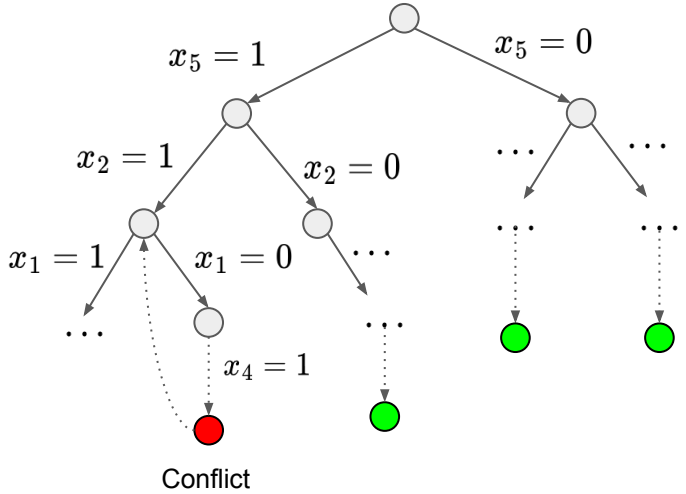




- Nearly all approaches of symbolic methods incorporate multiple heuristic-driven decision processes.
- Deep-learning guided heuristics show promising results.

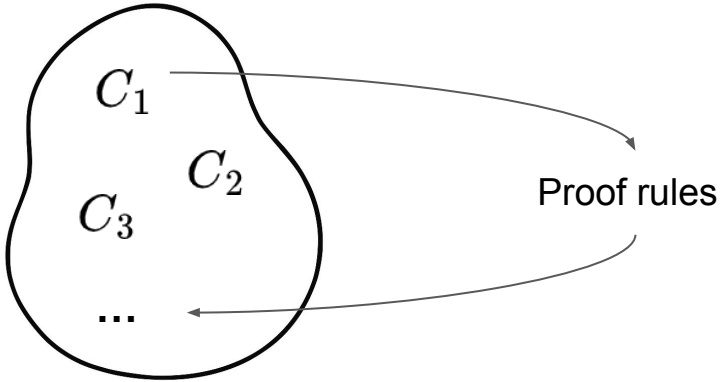
Summary of Motivating Examples

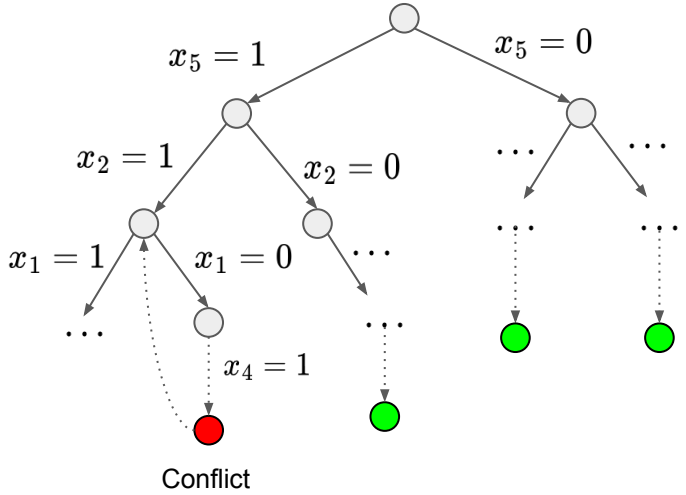




- Nearly all approaches of symbolic methods incorporate multiple heuristic-driven decision processes.
- Deep-learning guided heuristics show promising results.
- Each deep-learning guided heuristic is highly customized.

Summary of Motivating Examples





Can we generalize the deep-learning based guiding systematically, so that the decision processes can be improved within a unified framework?

List of papers

I **Exploring Representation of Horn Clauses Using GNNs.** Chencheng Liang, Philipp Rümmer, Marc Brockschmidt. *In Proceedings of 8th Workshop on Practical Aspects of Automated Reasoning (PAAR)*, 2022.

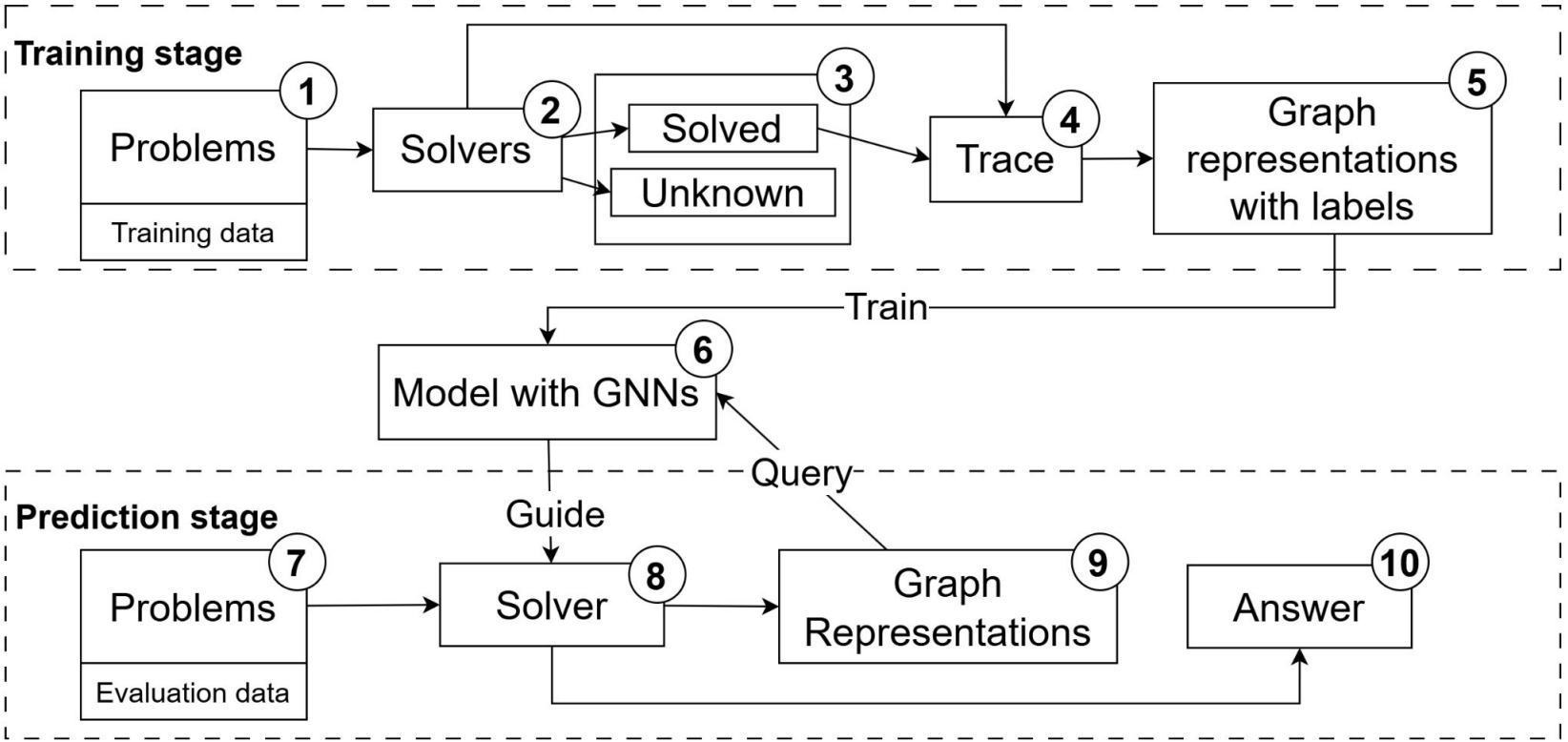
II **Boosting Constrained Horn Solving by Unsat Core Learning.** Parosh Aziz Abdulla, Chencheng Liang, Philipp Rümmer. *In Proceedings of 25th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, 2024.

III **Guiding Word Equation Solving Using Graph Neural Networks.** Parosh Aziz Abdulla, Mohamed Faouzi Atig, Julie Cailler, Chencheng Liang, Philipp Rümmer. *In Proceedings of 22nd International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2024.

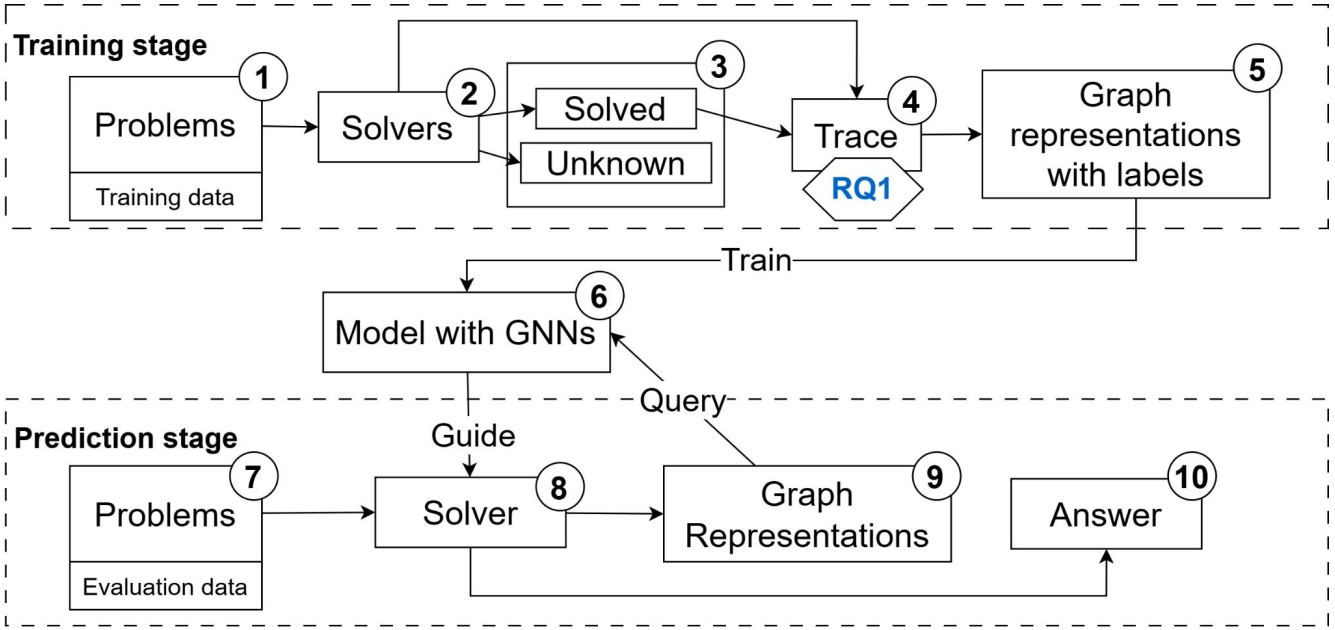
IV **When GNNs Met a Word Equations Solver: Learning to Rank Equations.** Parosh Aziz Abdulla, Mohamed Faouzi Atig, Julie Cailler, Chencheng Liang, Philipp Rümmer. *Under Submission*, 2025.

- Deep-learning based framework
- Instance 1: Constrained Horn clauses (CHCs) solving (paper I and II)
- Instance 2: Word equation solving (paper III and IV)
- Conclusion and future works

Deep Learning-Based Framework (Syntactic Structure)

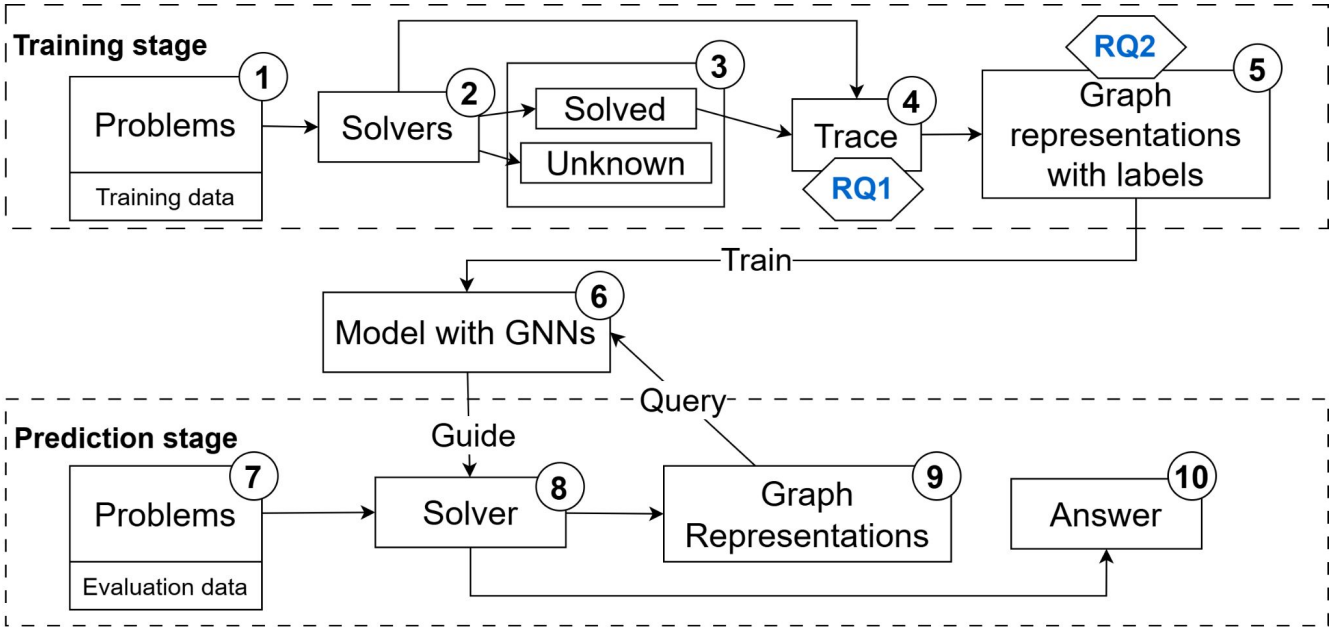


Deep Learning-Based Framework



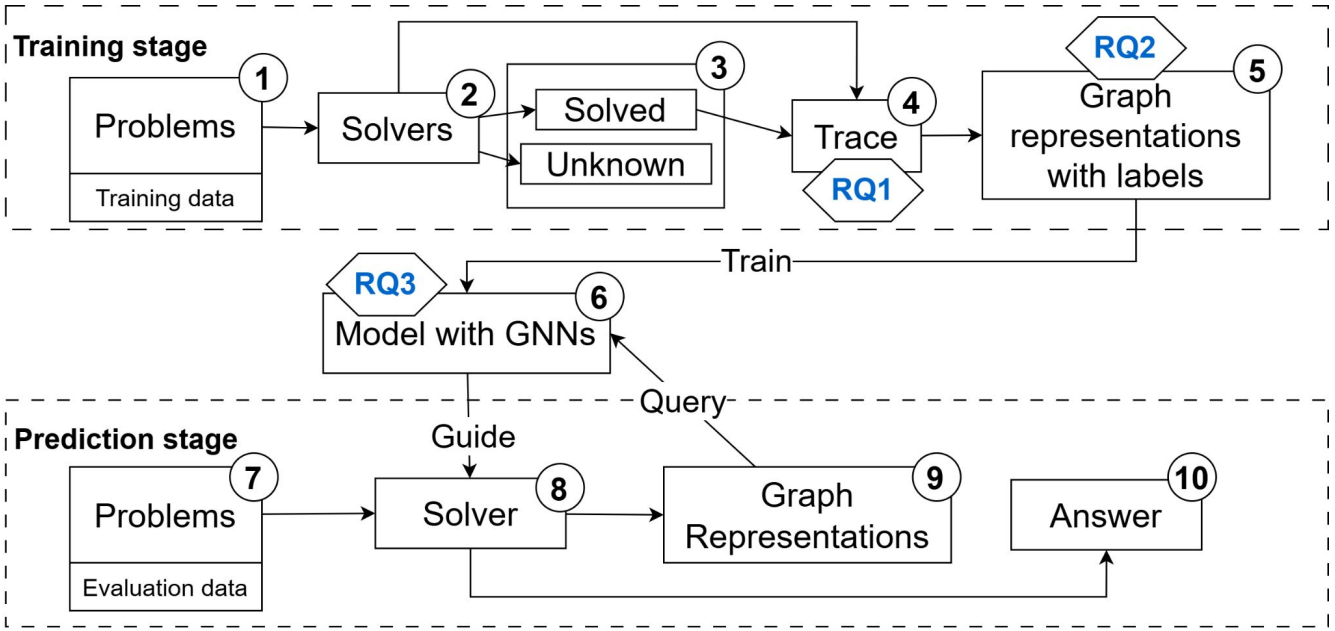
- RQ1: What are good encodings of symbolic decision processes as training tasks?

Deep Learning-Based Framework



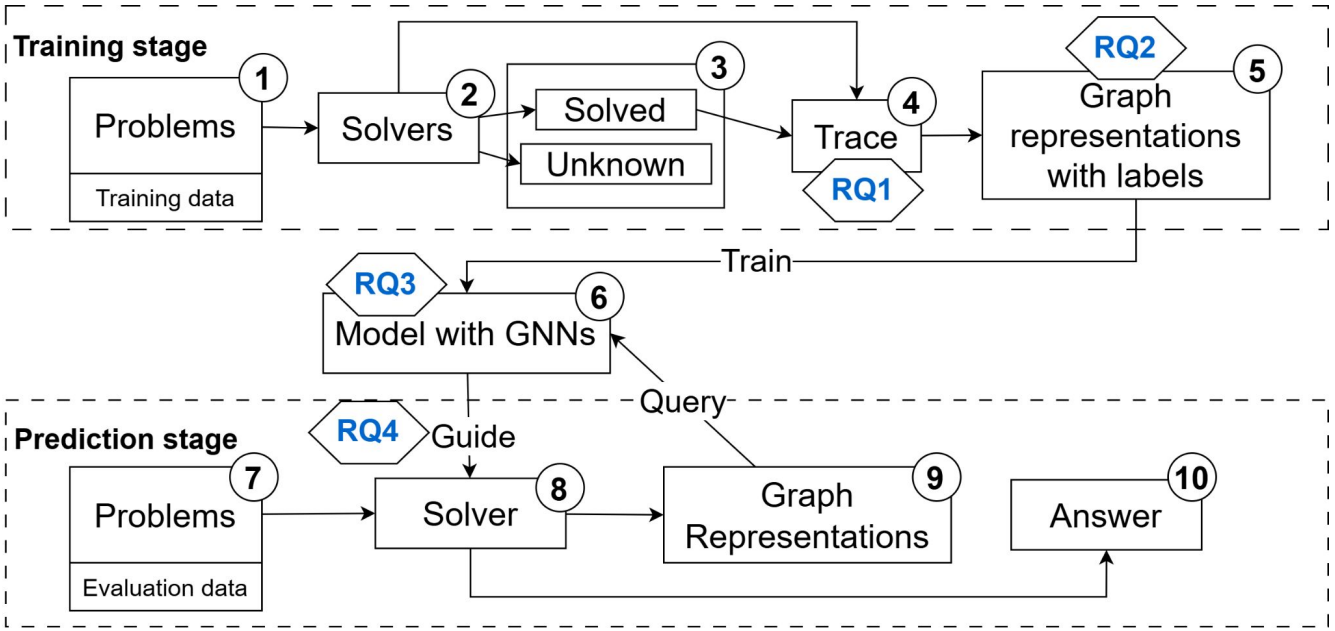
- RQ1: What are good encodings of symbolic decision processes as training tasks?
- RQ2: What is the most effective format for representing formulas in deep learning?

Deep Learning-Based Framework



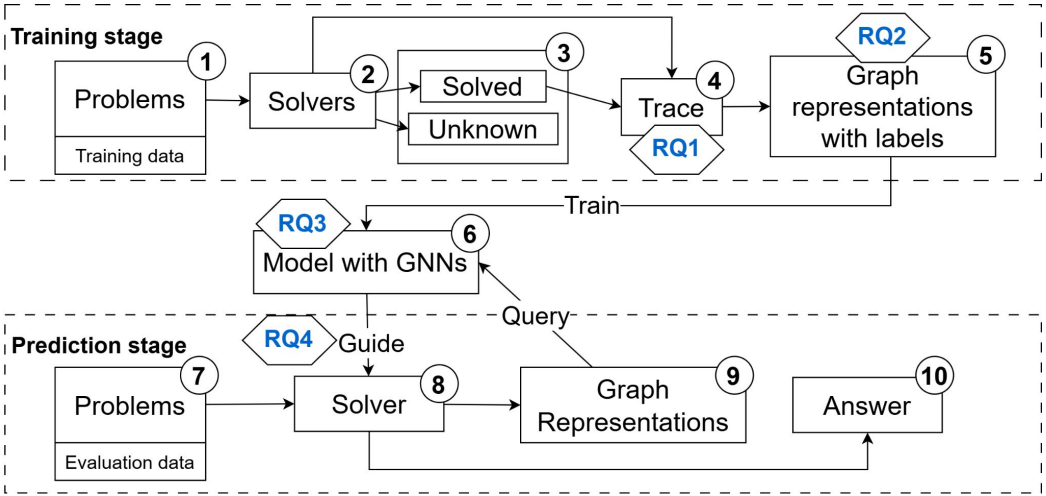
- RQ1: What are good encodings of symbolic decision processes as training tasks?
- RQ2: What is the most effective format for representing formulas in deep learning?
- RQ3: Which deep learning technique is best suited for feature extraction from formulas?

Deep Learning-Based Framework



- RQ1: What are good encodings of symbolic decision processes as training tasks?
- RQ2: What is the most effective format for representing formulas in deep learning?
- RQ3: Which deep learning technique is best suited for feature extraction from formulas?
- RQ4: What are the methods for integrating the trained model into algorithms?

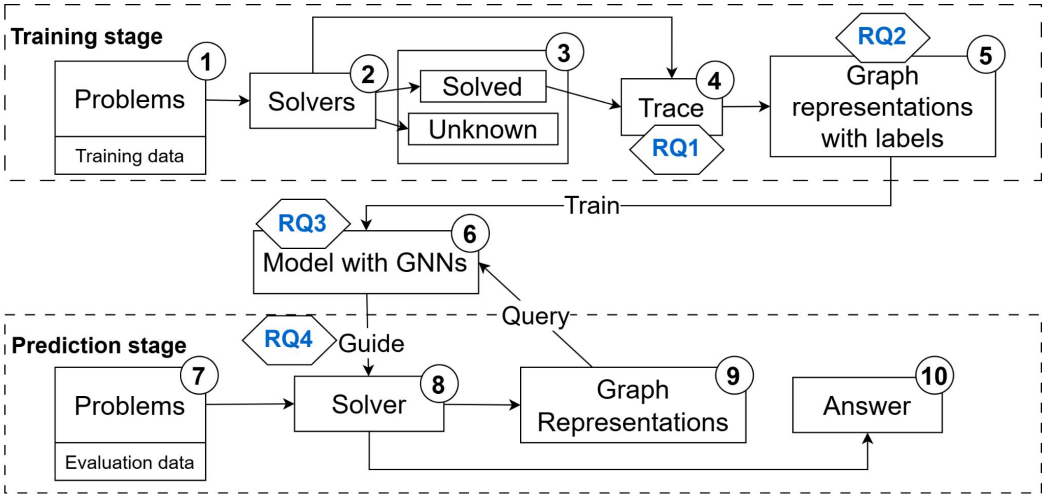
Deep Learning-Based Framework (Instances)



- Guide Constrained Horn Clauses (CHCs) solving (Paper I and II)

$$
\begin{aligned}
F(0,0) &\leftarrow true \\
F(1,1) &\leftarrow true \\
F(x, y_1 + y_2) &\leftarrow F(x-1, y_1) \wedge F(x-2, y_2) \wedge x > 1 \\
false &\leftarrow F(x, y) \wedge y < 0
\end{aligned}
$$

Deep Learning-Based Framework (Instances)



- Guide Constrained Horn Clauses (CHCs) solving (Paper I and II)
- Guide word equation solving (Paper III and IV)

$$XaY = YbX \wedge XabY = YbaX$$

Constrained Horn Clauses (CHCs)

- A CHC is a formula in the format

$$L(\bar{t}) \leftarrow L_1(\bar{t}_1) \wedge \ldots \wedge L_n(\bar{t}_n) \wedge \varphi$$

where $\bar{t}_i$ are terms,
$L, L_1, \ldots, L_n$ are relation symbols,
$L_i(\bar{t}_i)$ is an atom,
φ is a constraint in the background theory T.

A CHC system is satisfiable if there exists a interpretation such that every clause in the system evaluates to true.

CHCs (Example)

- Fibonacci function logic expression

$$\begin{aligned}&\text{fib}(0) = 0\\&\text{fib}(1) = 1\\&\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2),\ \text{for } n > 1\end{aligned}$$

- Its CHC encoding:

$$\begin{aligned}F(0,0) &\leftarrow \mathit{true}\\F(1,1) &\leftarrow \mathit{true}\\F(x, y_1 + y_2) &\leftarrow F(x-1, y_1) \wedge F(x-2, y_2) \wedge x > 1\end{aligned}$$

CHCs (Example)

- An assertions such as "the Fibonacci function does not return negative numbers" can be encoded as:

$$false \leftarrow F(x, y) \wedge y < 0$$

Where *false* is a predicate representing an assertion violation.

CHCs (Example)

- An assertions such as "the Fibonacci function does not return negative numbers" can be encoded as:

$$false \leftarrow F(x, y) \wedge y < 0$$

Where *false* is a predicate representing an assertion violation.

$$\begin{aligned} F(0,0) &\leftarrow true \\ F(1,1) &\leftarrow true \\ F(x, y_1 + y_2) &\leftarrow F(x-1, y_1) \wedge F(x-2, y_2) \wedge x > 1 \\ false &\leftarrow F(x, y) \wedge y < 0 \end{aligned}$$

Satisfiability of CHCs (Example)

- CHC encoding of Fibonacci function with an assertion:

$$
\begin{aligned}
F(0,0) &\leftarrow true \\
F(1,1) &\leftarrow true \\
F(x, y_1 + y_2) &\leftarrow F(x-1, y_1) \wedge F(x-2, y_2) \wedge x > 1 \\
false &\leftarrow F(x, y) \wedge y < 0
\end{aligned}
$$

- Consider a model:

$$F(x, y) \equiv x \geq 0 \wedge y \geq 0$$

Replacing *F* with this formula will make all clauses valid which means the system is satisfiable (SAT).

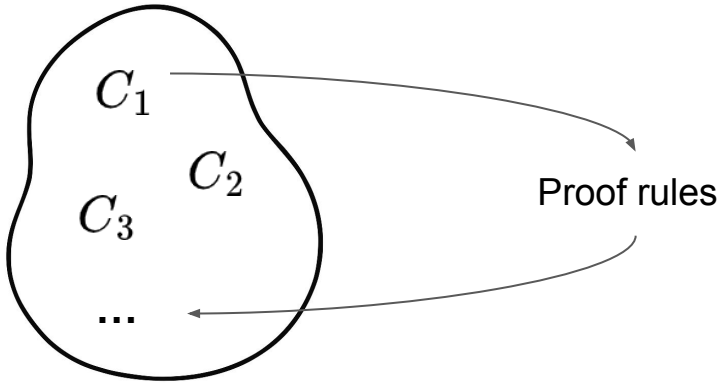
CHC Solver

- Algorithms implemented in CHC solvers
  - Counterexample-Guided Abstraction Refinement (CEGAR).
  - Symbolic execution.

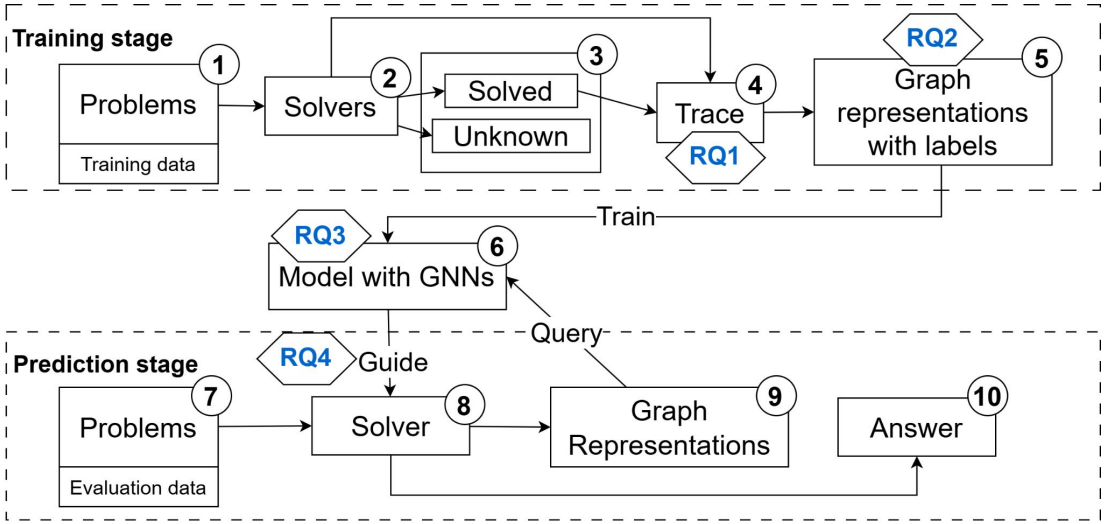
CHC Solver

- Algorithms implemented in CHC solvers
  - Counterexample-Guided Abstraction Refinement (CEGAR).
  - Symbolic execution.

- Learning to tank clauses before solving

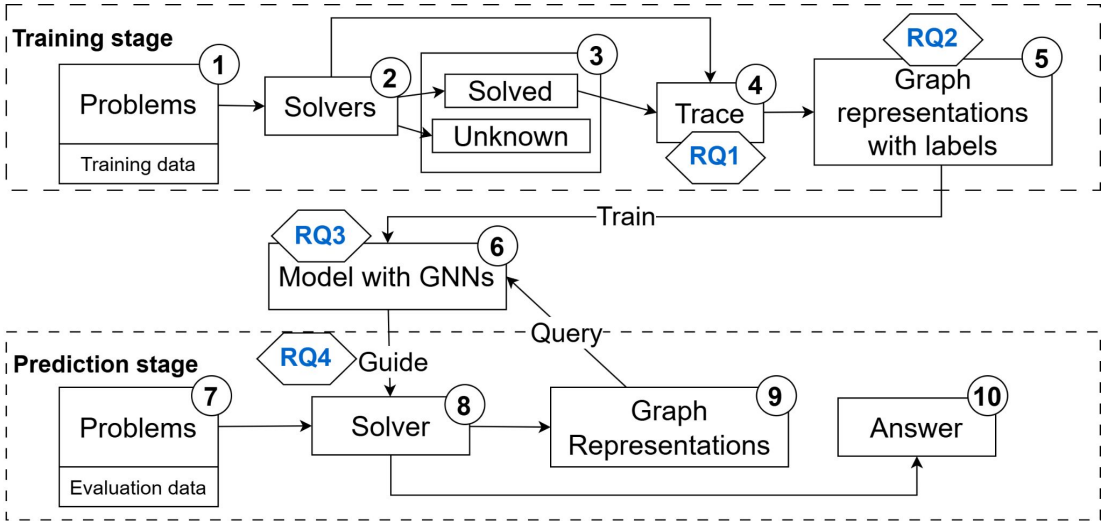


Rank CHCs to Guide the Solving



- RQ1: Train task.
- RQ2: Graph representation.
- RQ3: GNN models.
- RQ4: Integrating methods.

Rank CHCs to Guide the Solving

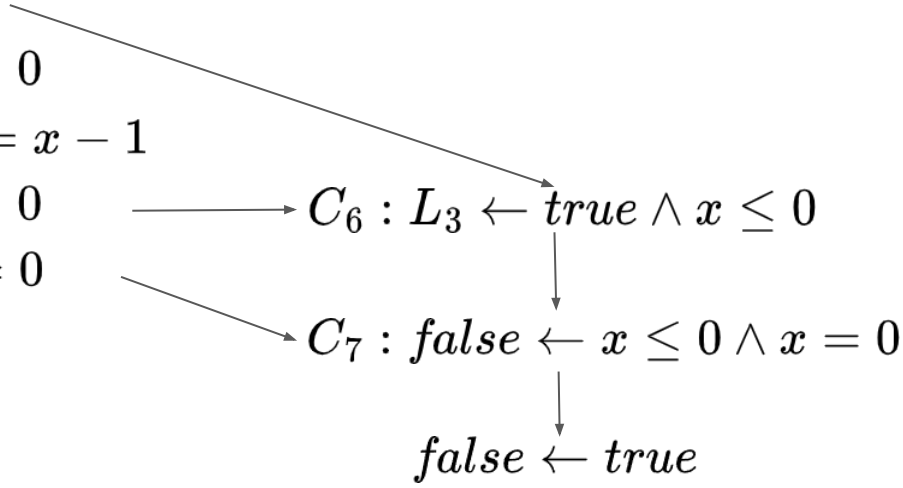


- Answer to RQ1 (train task):
  - Learn from Minimal Unsatisfiable Subset (MUS).

Rank CHCs to Guide the Solving

- Example of MUS

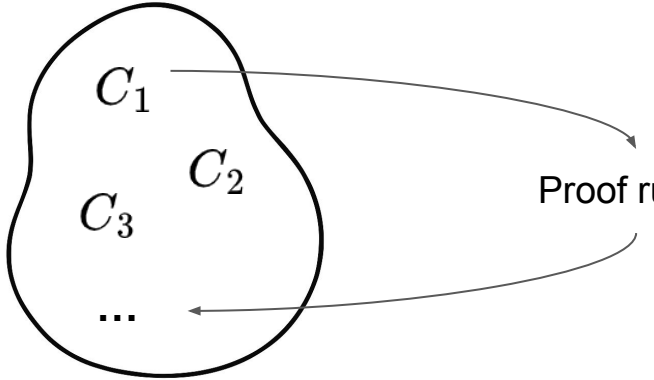
$$
\begin{aligned}
C_1 &: L_1(x) \leftarrow true \\
C_2 &: L_2(x) \leftarrow L_1(x) \wedge x > 0 \\
C_3 &: L_1(x') \leftarrow L_2(x) \wedge x' = x - 1 \\
C_4 &: L_3(x) \leftarrow L_1(x) \wedge x \leq 0 \\
C_5 &: false \leftarrow L_3(x) \wedge x = 0
\end{aligned}
$$



Rank CHCs to Guide the Solving

- Example of MUS

$$
\begin{aligned}
C_1 &: L_1(x) \leftarrow true \\
C_2 &: L_2(x) \leftarrow L_1(x) \wedge x > 0 \\
C_3 &: L_1(x') \leftarrow L_2(x) \wedge x' = x - 1 \\
C_4 &: L_3(x) \leftarrow L_1(x) \wedge x \leq 0 \\
C_5 &: false \leftarrow L_3(x) \wedge x = 0
\end{aligned}
$$



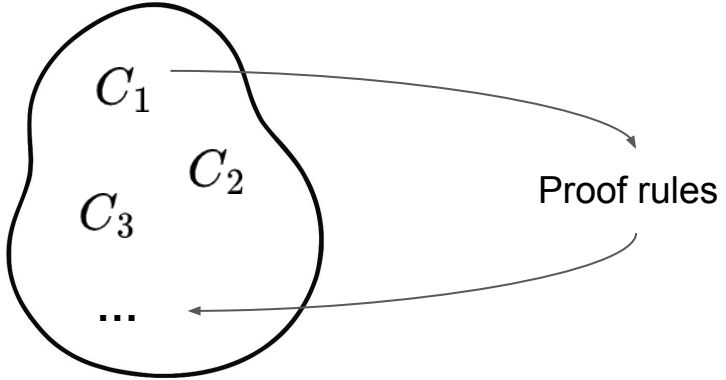
$$C_6 : L_3 \leftarrow true \wedge x \leq 0$$

$$C_7 : false \leftarrow x \leq 0 \wedge x = 0$$

$$false \leftarrow true$$

Rank CHCs to Guide the Solving

- Example of MUS



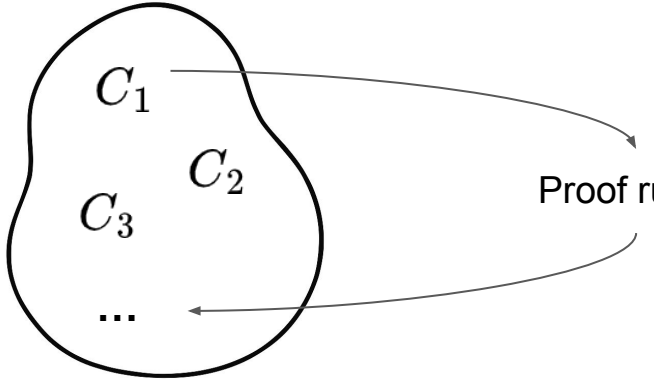
$$C_1 : L_1(x) \leftarrow true$$
$$\sout{C_2 : L_2(x) \leftarrow L_1(x) \wedge x > 0}$$
$$\sout{C_3 : L_1(x') \leftarrow L_2(x) \wedge x' = x - 1}$$
$$C_4 : L_3(x) \leftarrow L_1(x) \wedge x \leq 0$$
$$C_5 : false \leftarrow L_3(x) \wedge x = 0$$

$\{C1, C4, C5\}$ is a MUS.

$$C_6 : L_3 \leftarrow true \wedge x \leq 0$$
$$C_7 : false \leftarrow x \leq 0 \wedge x = 0$$
$$false \leftarrow true$$

Rank CHCs to Guide the Solving

- Example of MUS



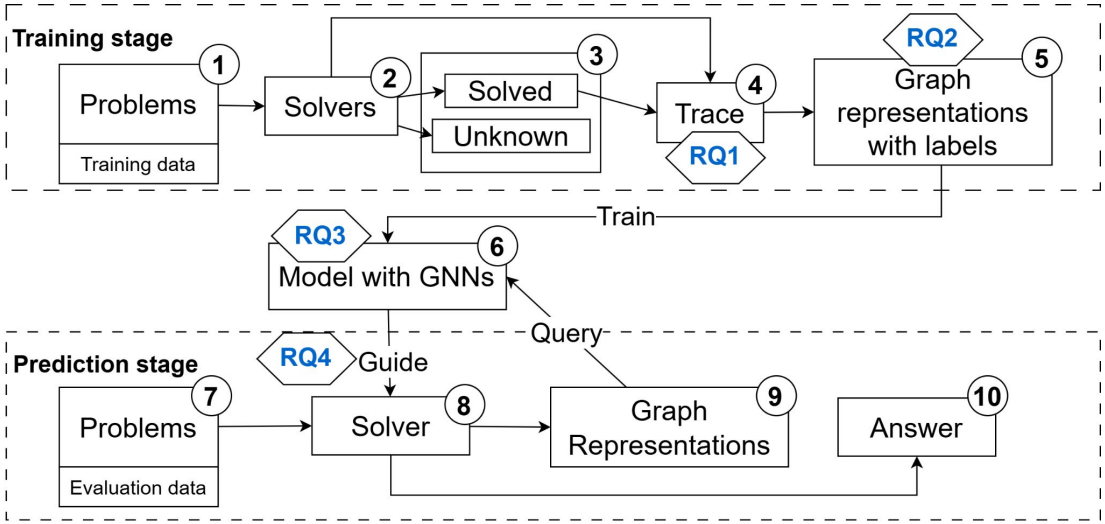
$$C_1 : L_1(x) \leftarrow true$$
$$\sout{C_2 : L_2(x) \leftarrow L_1(x) \wedge x > 0}$$
$$\sout{C_3 : L_1(x') \leftarrow L_2(x) \wedge x' = x - 1}$$
$$C_4 : L_3(x) \leftarrow L_1(x) \wedge x \leq 0$$
$$C_5 : false \leftarrow L_3(x) \wedge x = 0$$

$\{C1, C4, C5\}$ is a MUS.

$$C_6 : L_3 \leftarrow true \wedge x \leq 0$$
$$C_7 : false \leftarrow x \leq 0 \wedge x = 0$$
$$false \leftarrow true$$

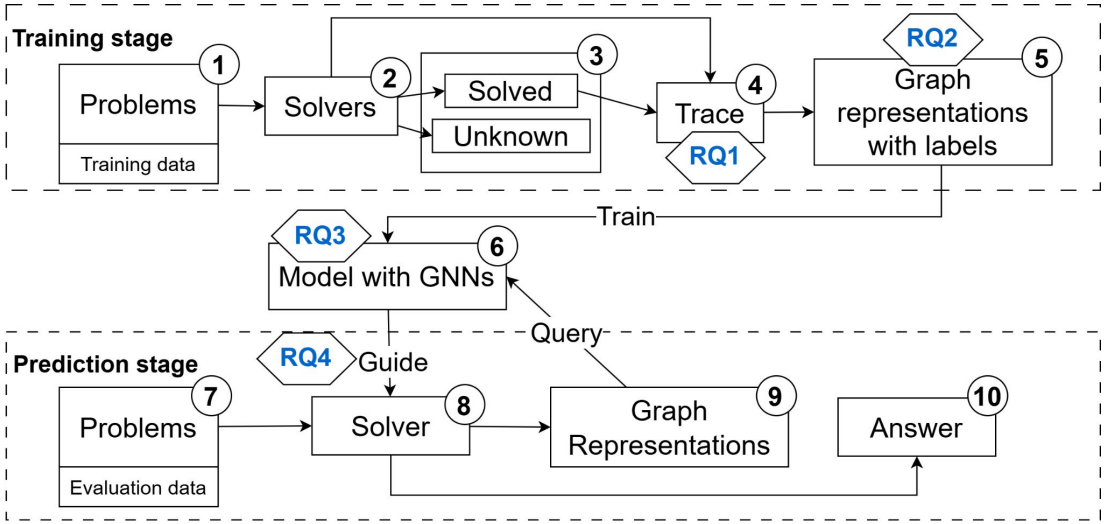
Let ϕ be a set of formulas such that ϕ is unsatisfiable. A subset $\phi' \subseteq \phi$ is called a MUS if ϕ' is unsatisfiable and for all proper subset $\phi'' \subseteq \phi'$, ϕ'' is satisfiable.

Rank CHCs to Guide the Solving



- Answer to RQ1 (train task):
  - Belongs to MUS or not.

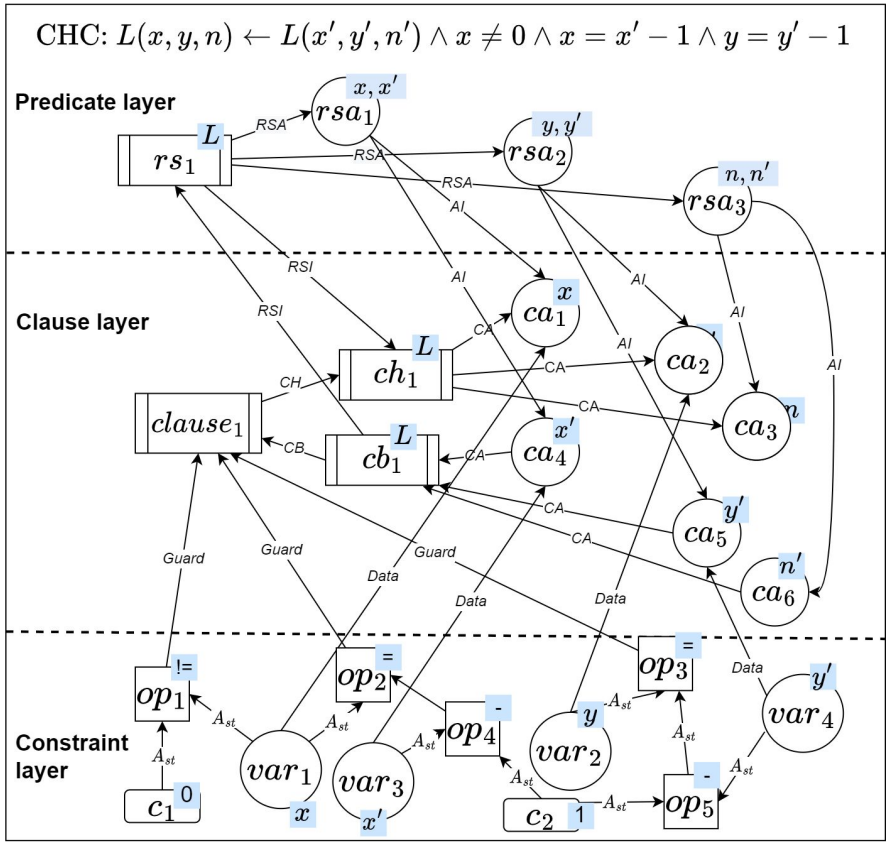
Rank CHCs to Guide the Solving



- Answer to RQ2 (graph representation):
  - Syntactic structure.
  - Control-flow and data-flow.

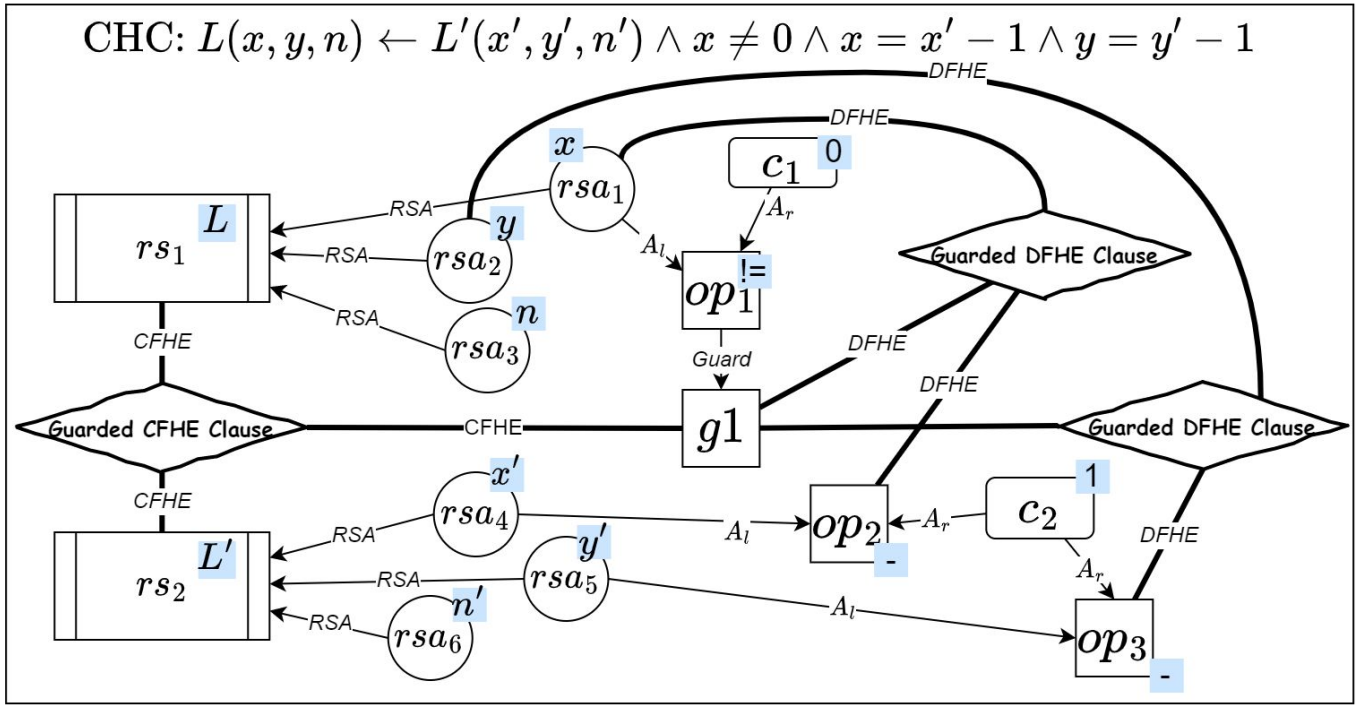
Rank CHCs to Guide the Solving (Graph Representation)

Constraint graph (CG)

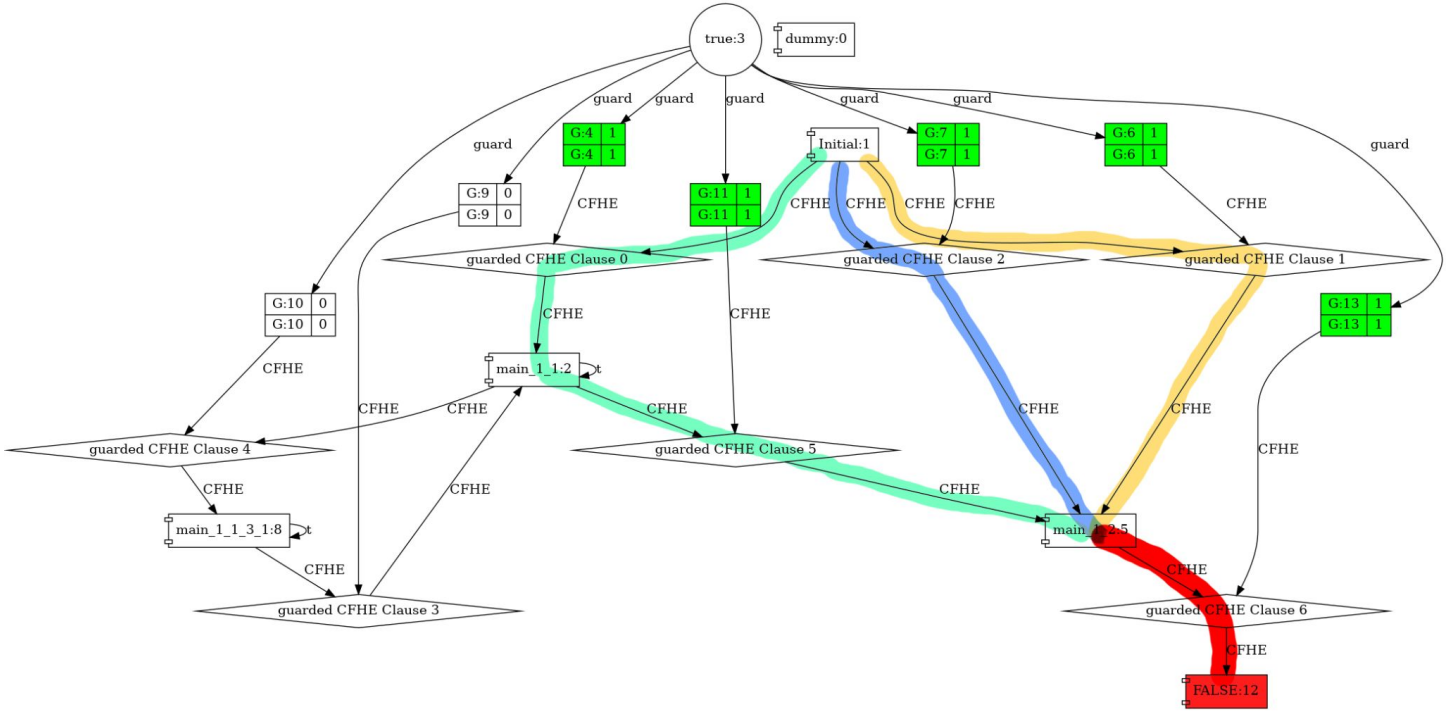


Rank CHCs to Guide the Solving (Graph Representation)

Control- and data- flow hypergraph (CDHG)

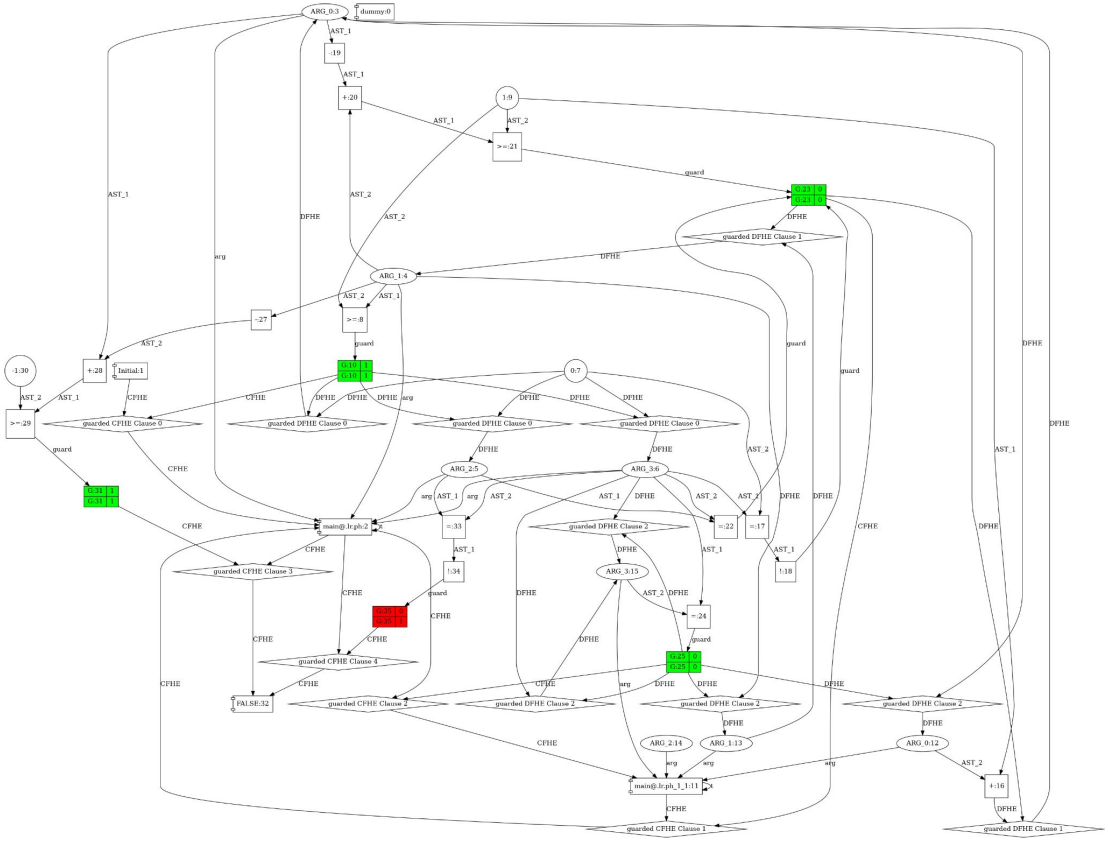


Control- and Data-Flow Hypergraph (CDHG)

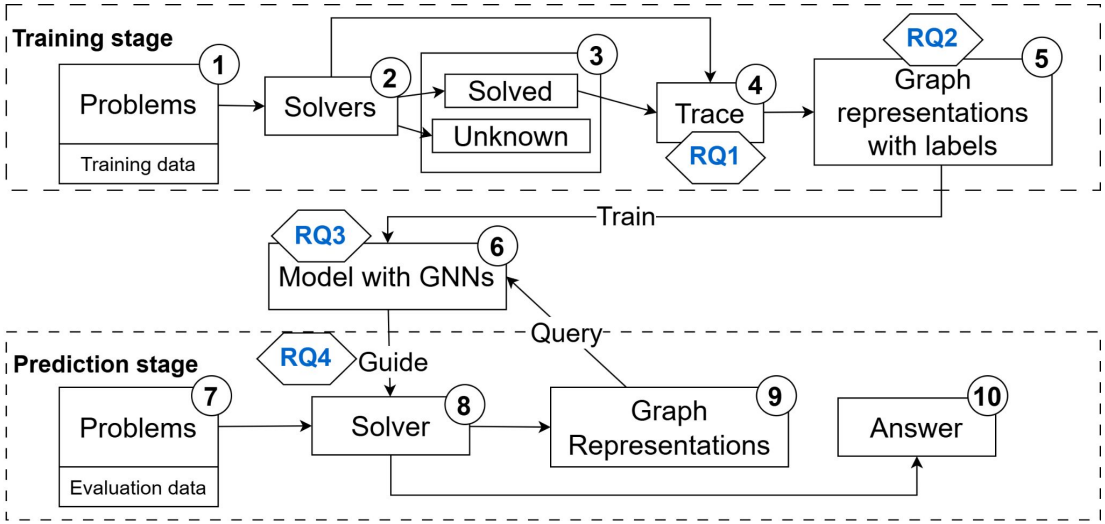


Paper I and II

Control- and Data-Flow Hypergraph (CDHG)



Rank CHCs to Guide the Solving



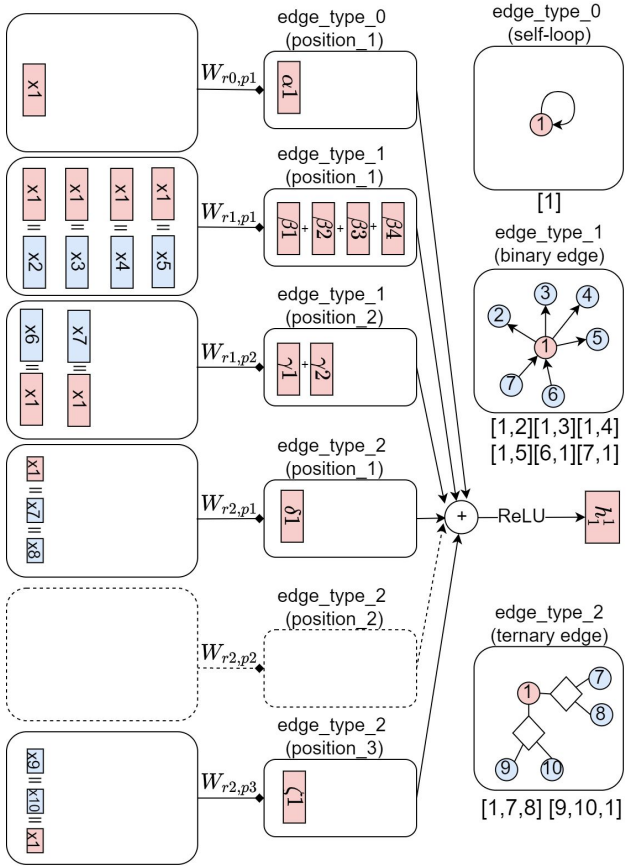
- Answer to RQ3 (GNN model):
  - Relational Hyper-Graph Neural Network (R-HyGNN).

Relational Hyper-Graph Neural Network (R-HyGNN)

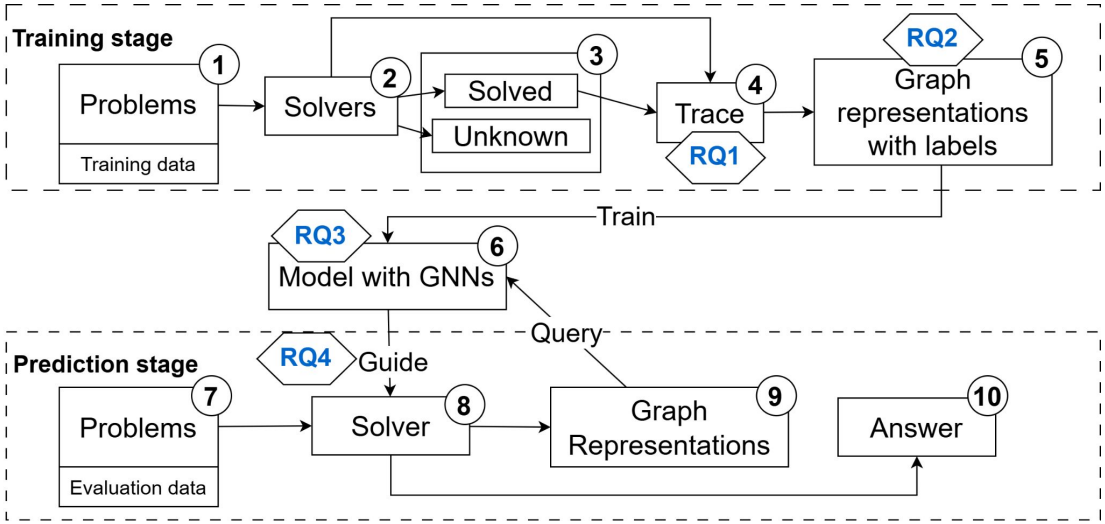
The updating rule for node representation in time step t:

$$h_v^t = \mathrm{ReLU}\left(\sum_{r\in R}\sum_{p\in P_r}\sum_{e\in E_v^{r,p}} W_{r,p}^t\right)\cdot ||\{h_u^{t-1}|u\in e\},$$

where $||\{\cdot\}$ denotes concatenation of elements in a set,
R is the set of edge types,
P_r is the set of node position under edge type r,
$W_{r,p}^t$ denotes learnable parameters,
$E_v^{r,p}$ is the set of hyperedges.



Rank CHCs to Guide the Solving



- Answer to RQ4 (integrating methods):
  - Use prediction alone.
  - Combine with existing heuristics.
  - Combine with random clause selection.

Experimental Results (Improved Percentage)

- Evaluated in CHC solver Eldarica [6].
- Using CHC-COMP dataset.

Benchmark Algorithm	MUS data set (best count)	Best ranking function (improvement in %)						
		Number of Solved Problems			Average Time			
		Total	SAT	UNSAT	All	Common	SAT	UNSAT
Linear CEGAR	Union (0)	R-Plus (1.4%)	R-Plus (2.4%)	R-Minus (1.0%)	R-Plus (1.3%)	S-Plus (19.1%)	S-Minus (46.5%)	Rank (31.1%)
	Single (3)	Rank (3.6%)	**R-Plus (4.0%)**	Rank (8.2%)	R-Plus (1.9%)	S-Plus (26.6%)	**R-Minus (57.9%)**	**Rank (36.3%)**
	Intersection (4)	**R-Plus (4.1%)**	S-Plus (0.8%)	**R-Plus (9.3%)**	**R-Plus (3.1%)**	**S-Plus (27.6%)**	R-Minus (45.0%)	S-Plus (0.0%)
Linear SymEx	Union (4)	**Two-Q (1.0%)**	**S-Plus* (0.0%)**	**Random (2.0%)**	Two-Q (0.9%)	R-Minus (12.7%)	**R-Minus (30.2%)**	S-Plus (26.5%)
	Single (3)	S-Minus* (0.5%)	**S-Plus* (0.0%)**	**Random (2.0%)**	Random (0.8%)	**S-Plus (12.9%)**	Random (28.4%)	S-Plus (17.6%)
	Intersection (5)	**S-Plus* (1.0%)**	**S-Plus* (0.0%)**	**S-Plus* (2.0%)**	**S-Plus (1.3%)**	Score (9.5%)	Random (28.4%)	**R-Plus (35.8%)**

[6] H. Hojjat and P. Ruemmer, “The ELDARICA Horn solver,” 2018.

Word Equations (example)

A word equation:

$$XabY = YbaX$$

where *a* and *b* are letters,
X, Y, and Z are variables ranging over strings of these letters.

Satisfiability of a Word Equation

A word equation:

$$XabY = YbaX$$

where *a* and *b* are letters,
X, Y, and Z are variables ranging over strings of these letters.

If there exists a substitution of the variables with strings over the letters that makes the equation hold, then the equation is satisfiable.

Satisfiability of a Word Equation

A word equation:

$$XabY = YbaX$$

where *a* and *b* are letters,
X, Y, and Z are variables ranging over strings of these letters.

If there exists a substitution of the variables with strings over the letters that makes the equation hold, then the equation is satisfiable.

SAT or UNSAT?

Satisfiability of a Word Equation

A word equation:

$$XabY = YbaX$$

where *a* and *b* are letters,
X, Y, and Z are variables ranging over strings of these letters.

If there exists a substitution of the variables with strings over the letters that makes the equation hold, then the equation is satisfiable.

$$X = b, Y = \epsilon$$

$$bab = bab$$

Word Equations

- Important in modeling string constraints in verification tasks .
  - E.g., Validate user inputs, ensuring correct string manipulations.

Word Equations

- Important in modeling string constraints in verification tasks.
  - E.g., Validate user inputs, ensuring correct string manipulations.

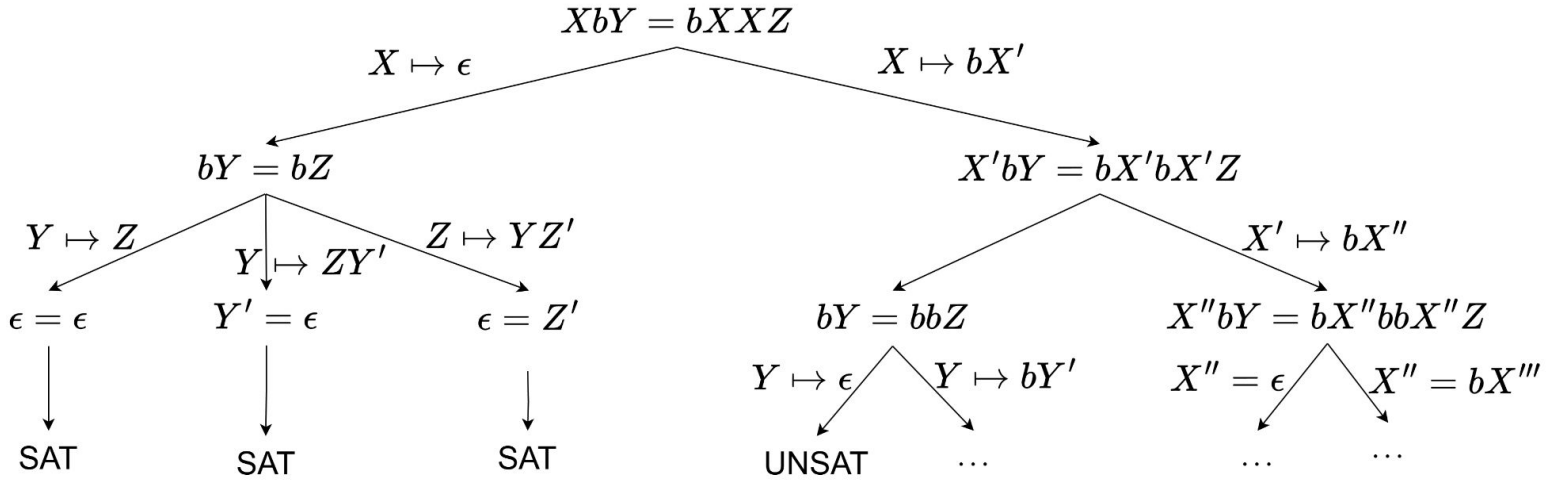
- Difficult to solve.
  - Decision procedures such as [6] and [7] have no implementation.
  - Practical algorithms are incomplete.

[6] Makanin, G.S.: The problem of solvability of equations in a free semigroup. 1977
[7] Plandowski, W.: Satisfiability of word equations with constants is in pspace. 1999.

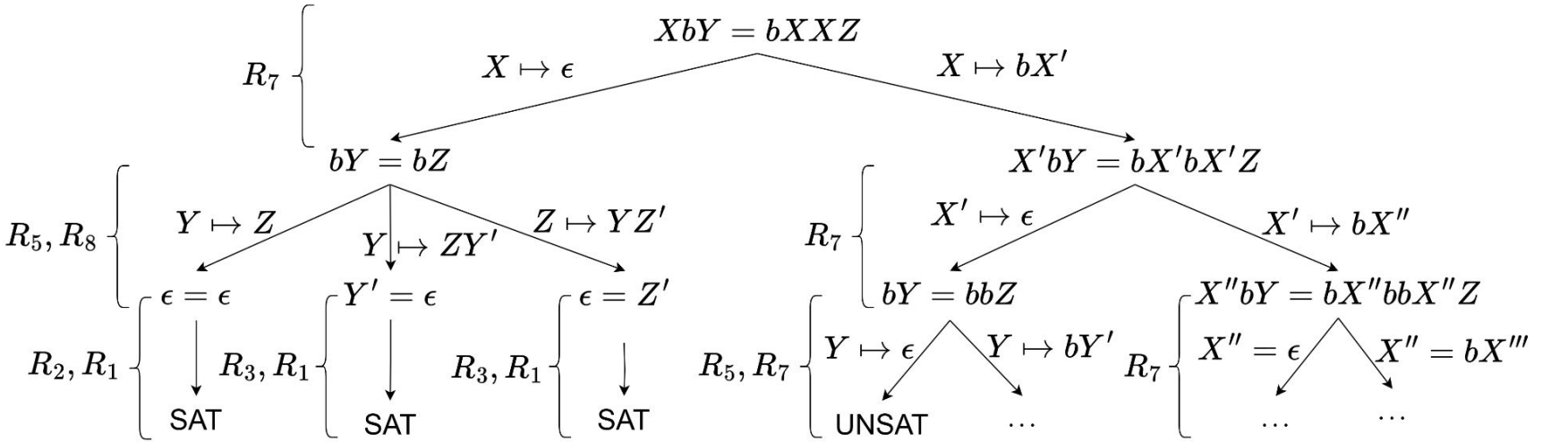
Solving a Word Equation System

- Split Algorithm
  - Based on Levi's lemma (Nielsen transformation in group theory).

Split Algorithm (Branch Process)



Split Algorithm (Branch Process)



$$R_7 \; \frac{X \cdot u = a \cdot v \wedge \phi}{\begin{array}{c|c} [X \mapsto \epsilon] & [X \mapsto a \cdot X'] \\ u = a \cdot v \wedge \phi & X' \cdot u = v \wedge \phi \end{array}}$$

$$R_8 \; \frac{X \cdot u = Y \cdot v \wedge \phi}{\begin{array}{c|c|c} [X \mapsto Y] & [X \mapsto Y \cdot Y'] & [Y \mapsto X \cdot X'] \\ u = v \wedge \phi & Y' \cdot u = v \wedge \phi & u = X' \cdot v \wedge \phi \end{array}}$$

Calculus

$$R_1\ \frac{true}{\text{SAT}} \qquad R_2\ \frac{\epsilon = \epsilon \wedge \phi}{\phi} \qquad R_3\ \frac{X = \epsilon \wedge \phi}{\begin{array}{c}[X \mapsto \epsilon] \\ \phi\end{array}} \qquad R_4\ \frac{a \cdot u = \epsilon \wedge \phi}{\text{UNSAT}}$$

with $X \in \Gamma$ and $a \in \Sigma$.

(a) Simplification rules

$$R_5\ \frac{a \cdot u = a \cdot v \wedge \phi}{u = v \wedge \phi} \qquad R_6\ \frac{a \cdot u = b \cdot v \wedge \phi}{\text{UNSAT}}$$

with a, b two different letters from Σ.

(b) Letter-letter rules

$$R_7\ \frac{X \cdot u = a \cdot v \wedge \phi}{\begin{array}{c|c}[X \mapsto \epsilon] & [X \mapsto a \cdot X'] \\ u = a \cdot v \wedge \phi & X' \cdot u = v \wedge \phi\end{array}}$$

with X' a *fresh* element of Γ.

(c) Variable-letter rules

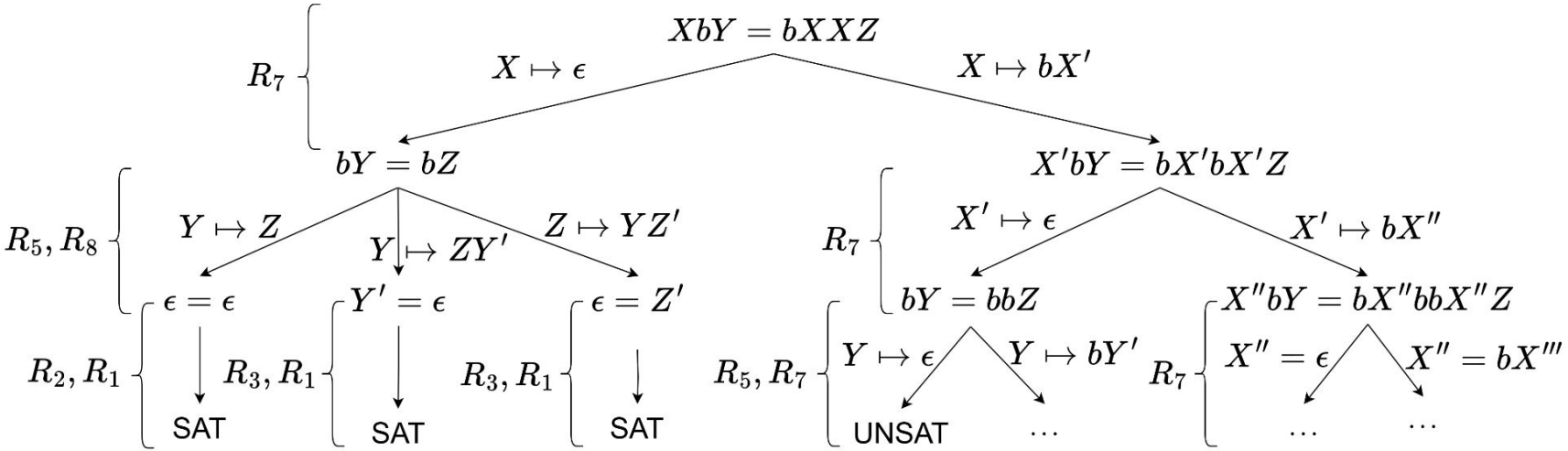
$$R_8\ \frac{X \cdot u = Y \cdot v \wedge \phi}{\begin{array}{c|c|c}[X \mapsto Y] & [X \mapsto Y \cdot Y'] & [Y \mapsto X \cdot X'] \\ u = v \wedge \phi & Y' \cdot u = v \wedge \phi & u = X' \cdot v \wedge \phi\end{array}}$$

with $X \neq Y$ and X', Y' *fresh* elements of Γ.

$$R_9\ \frac{X \cdot u = X \cdot v \wedge \phi}{u = v \wedge \phi}$$

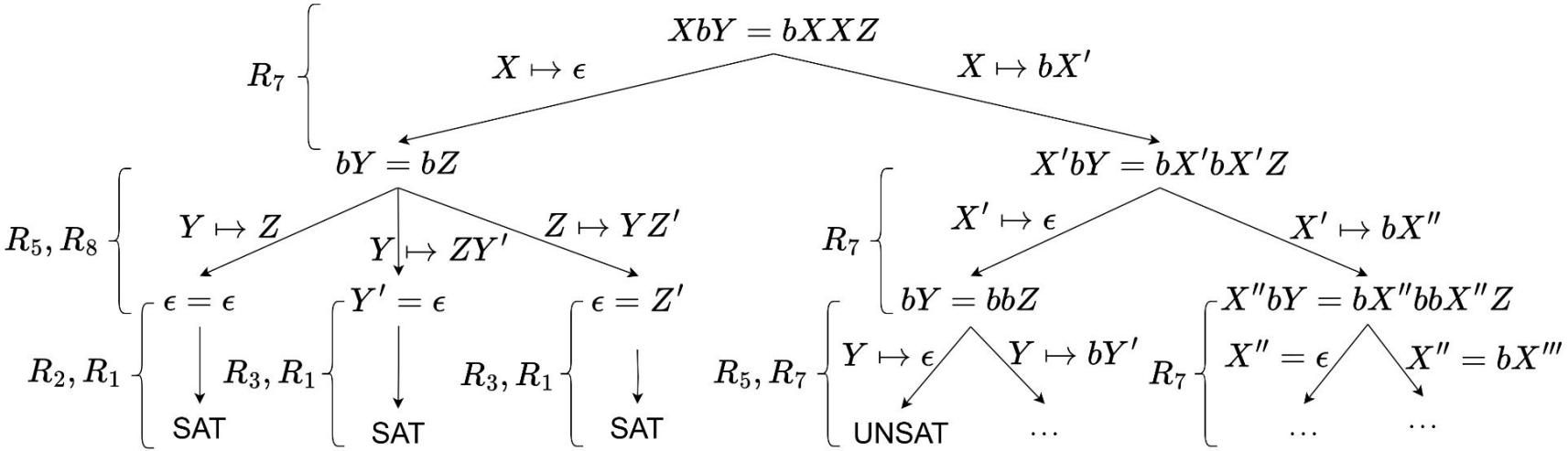
(d) Variable-variable rules

Split Algorithm (Branch Process)



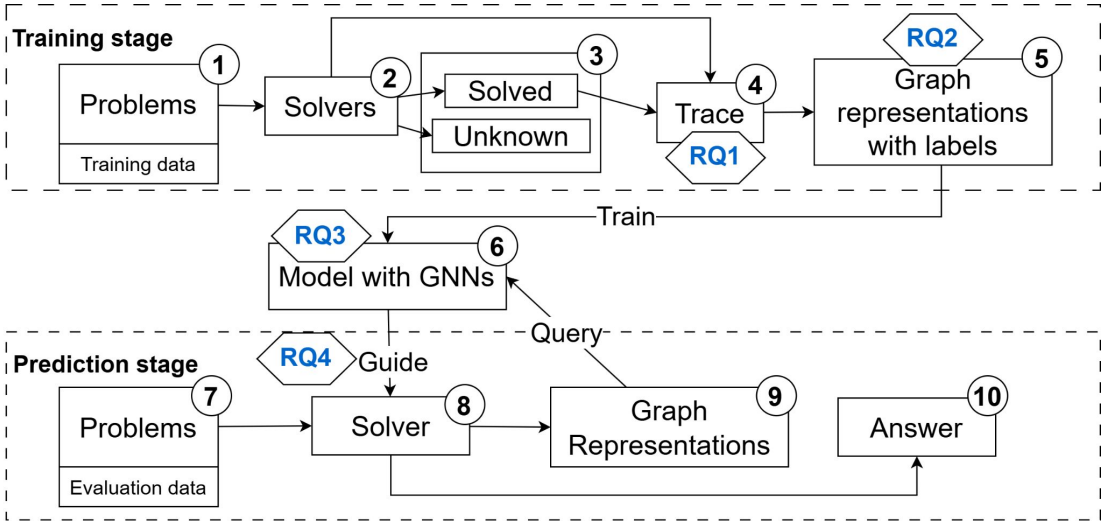
- If there is one branch SAT, then the word equation is SAT.
- If all branches are UNSAT, then the word equation is UNSAT.

Split Algorithm (Branch Process)



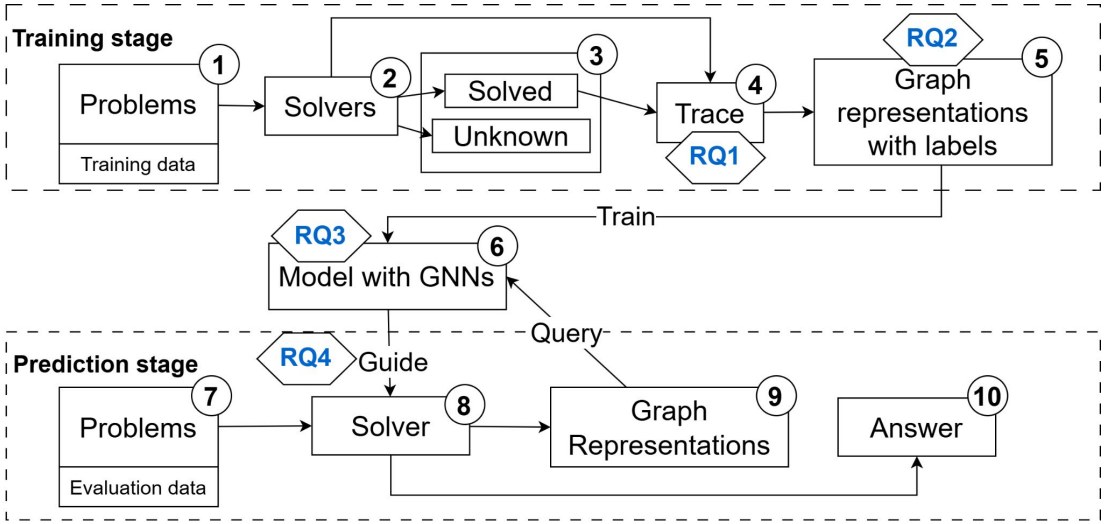
- Branching significantly affects the performance.

Select Branches to Guide the Solving



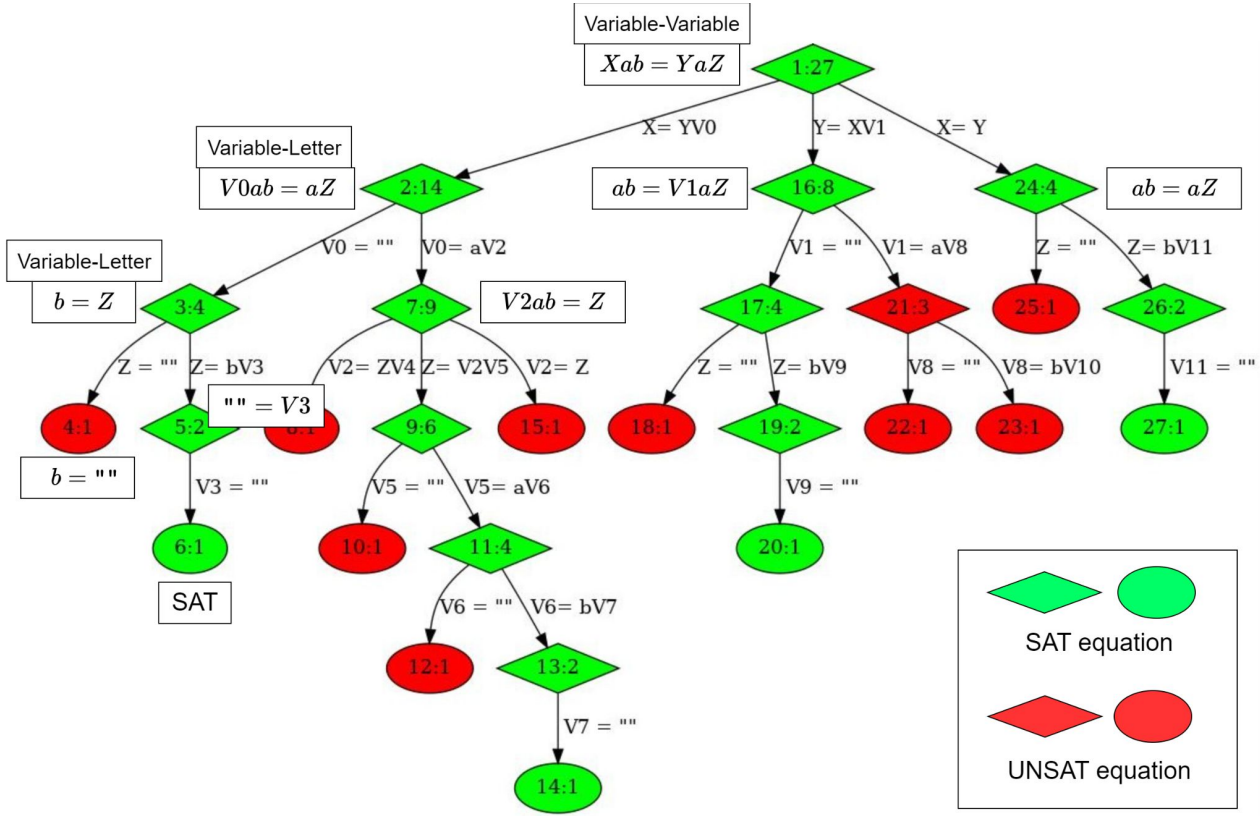
- RQ1: Train task.
- RQ2: Graph representation.
- RQ3: GNN models.
- RQ4: Integrating methods.

Select Branches to Guide the Solving

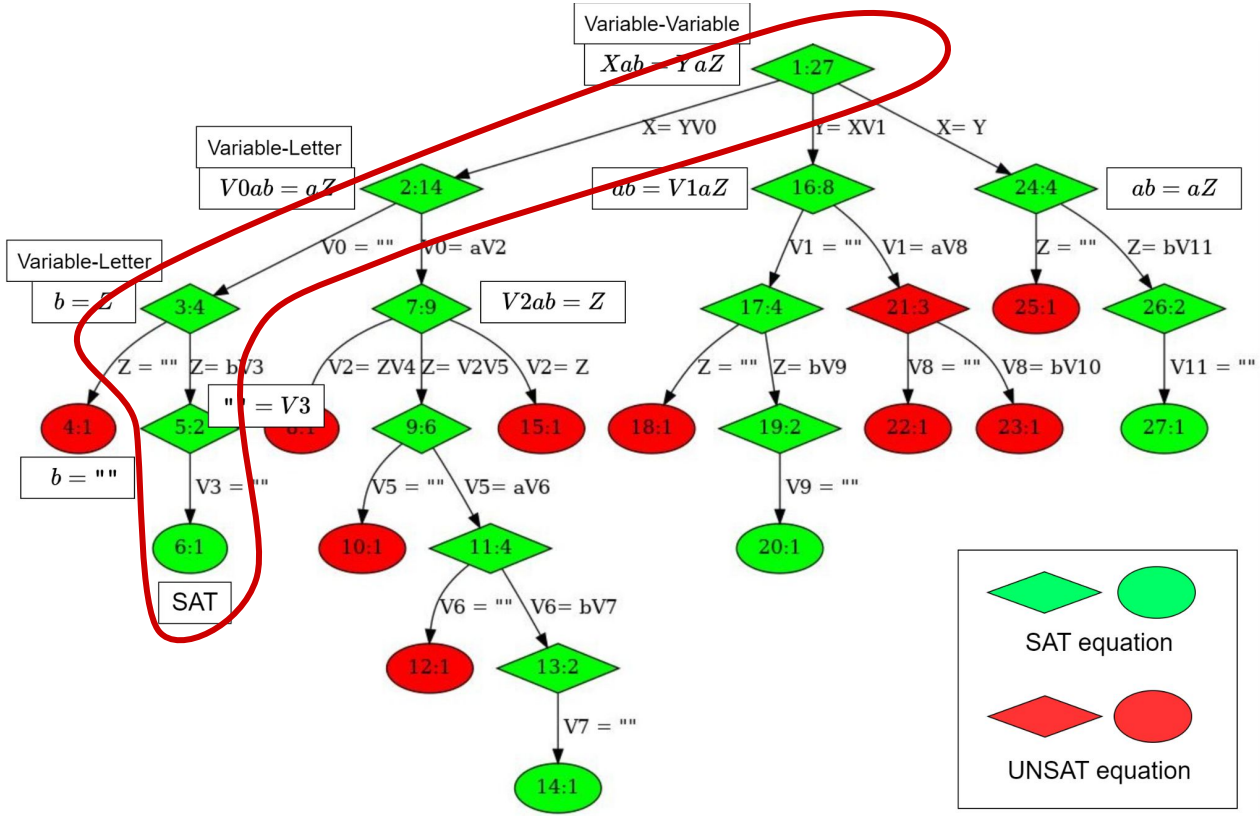


- Answer to RQ1 (train task):
  - Learn from shortest path to SAT.

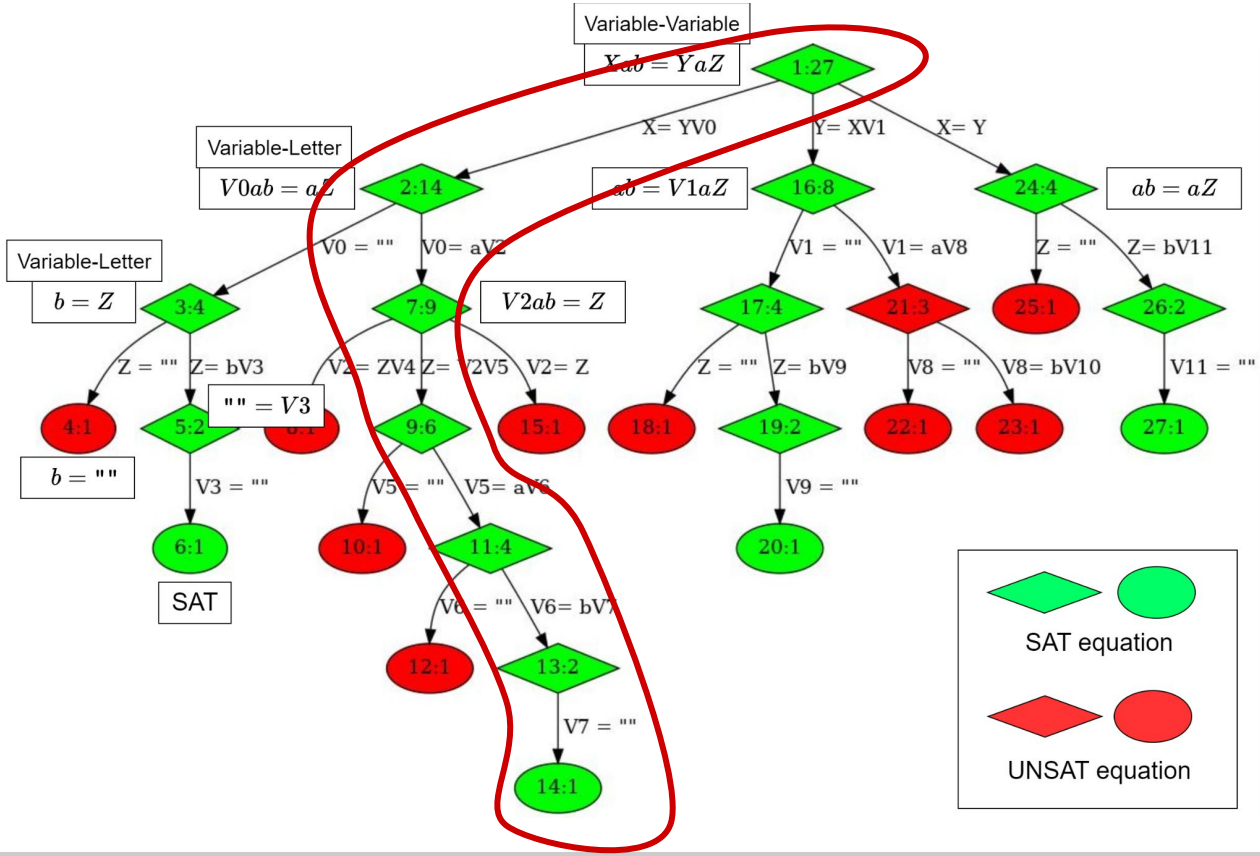
Select Branches to Guide the Solving (Train Data Collection)



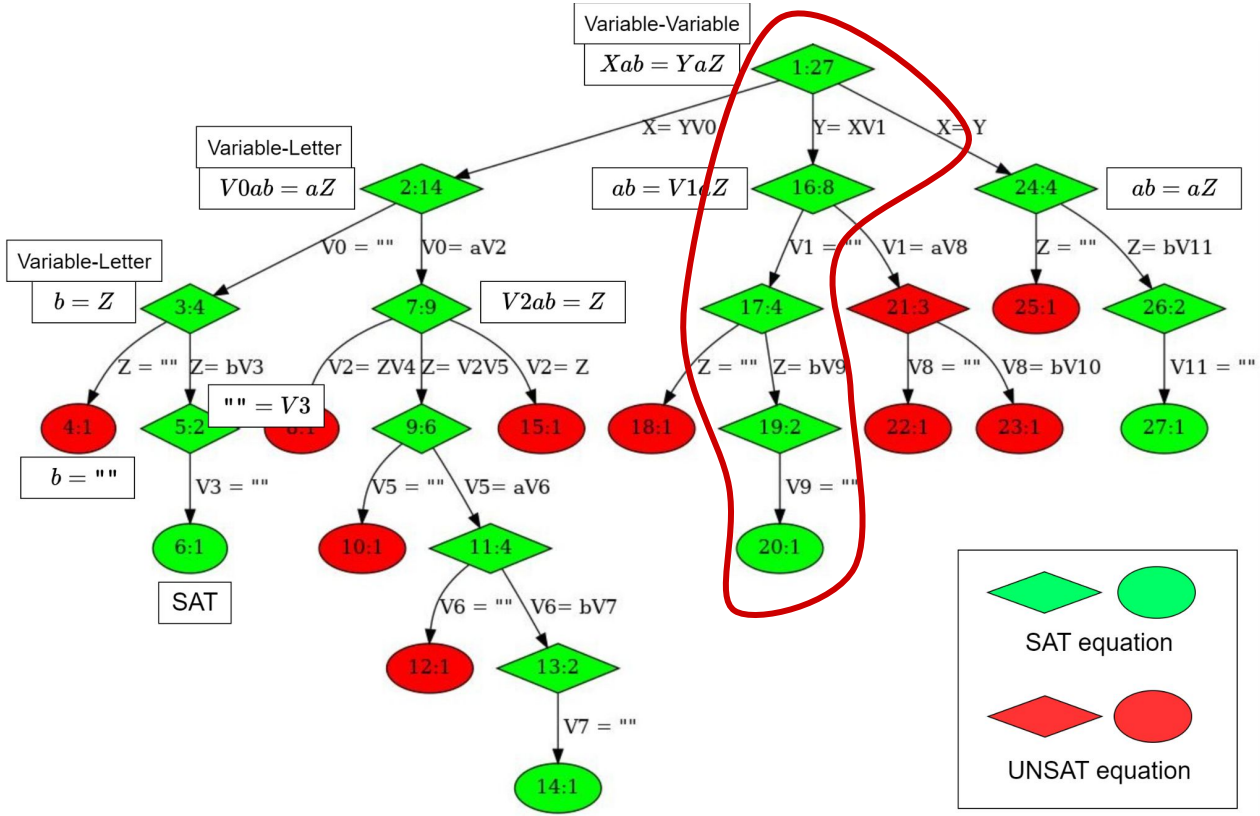
Select Branches to Guide the Solving (Train Data Collection)



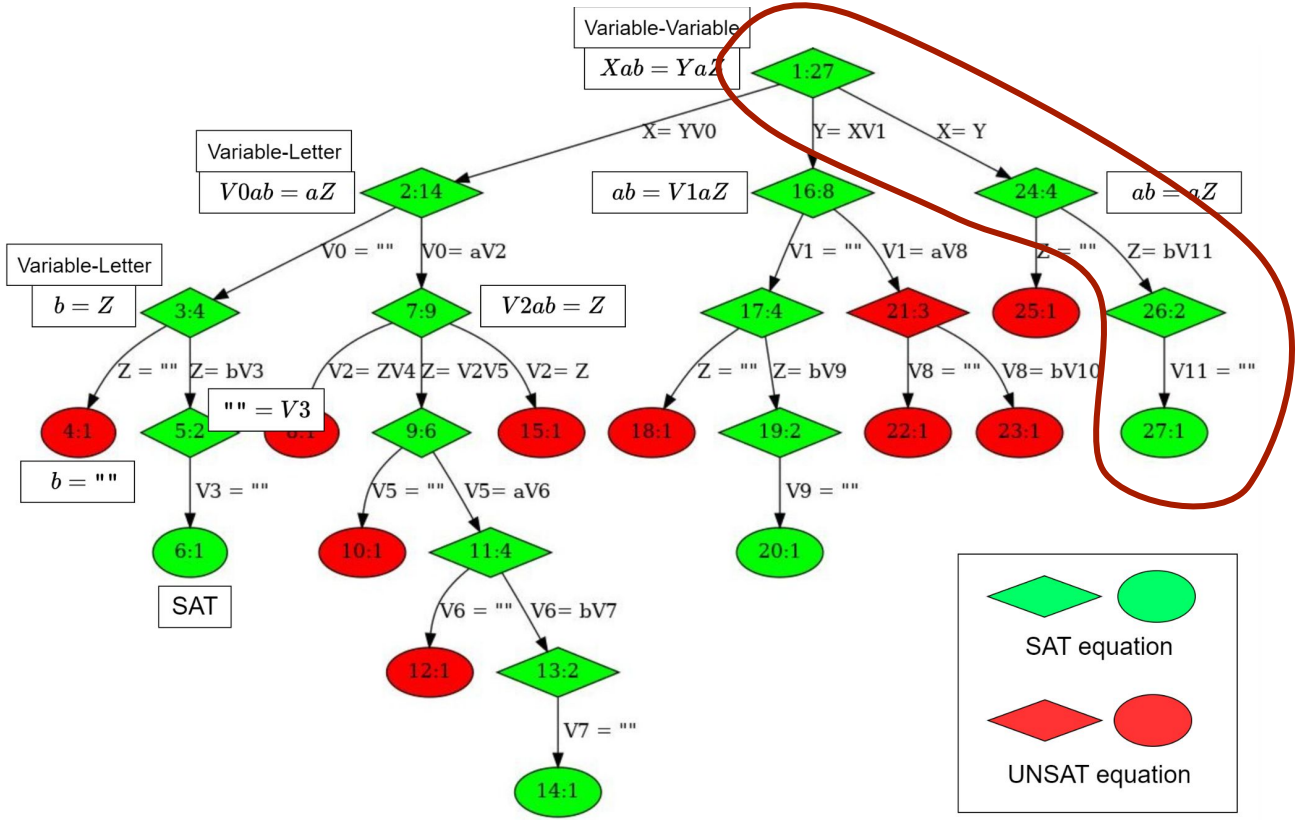
Select Branches to Guide the Solving (Train Data Collection)



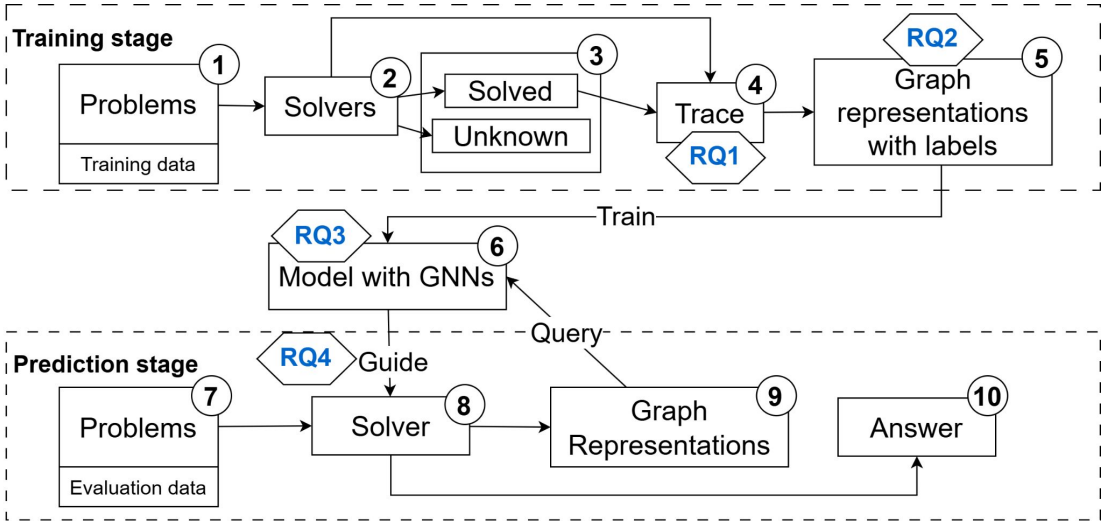
Select Branches to Guide the Solving (Train Data Collection)



Select Branches to Guide the Solving (Train Data Collection)



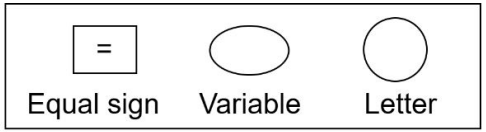
Select Branches to Guide the Solving

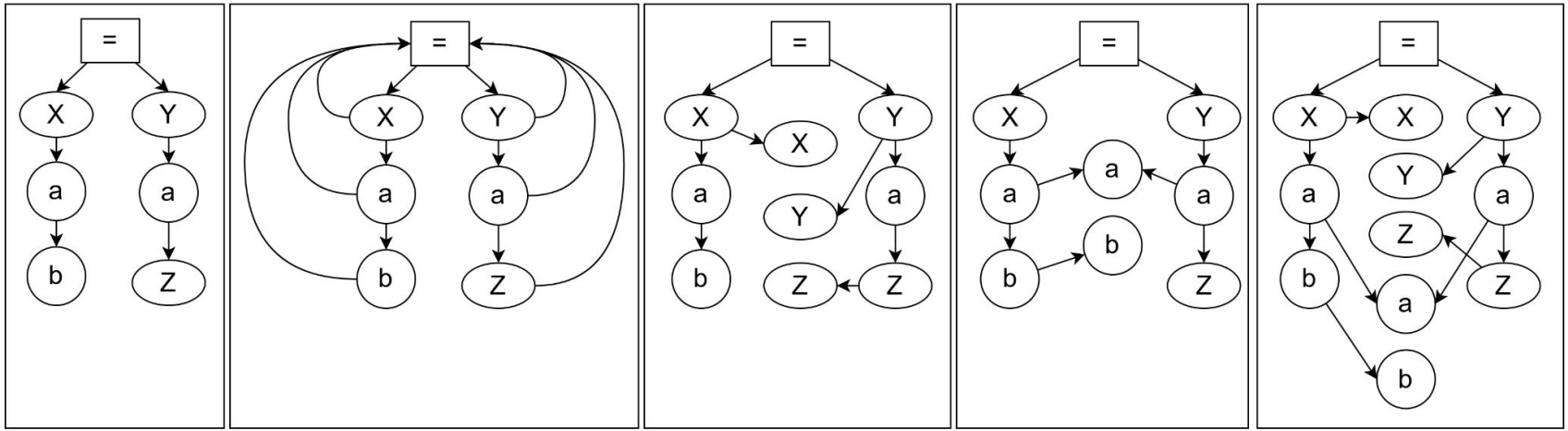


- Answer to RQ2 (graph representation).

Select Branches to Guide the Solving (Graph Representation)

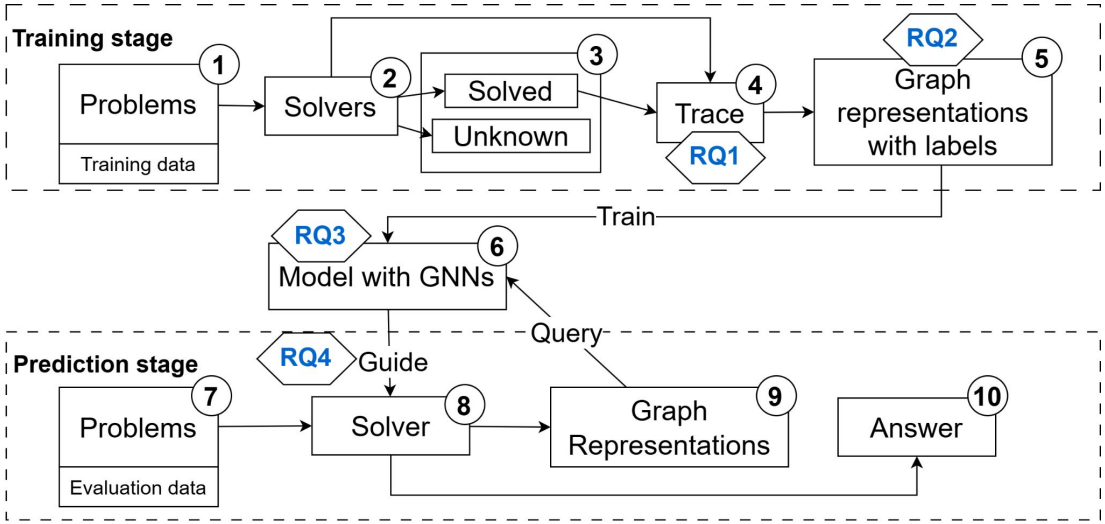
$$Xab = YaZ$$





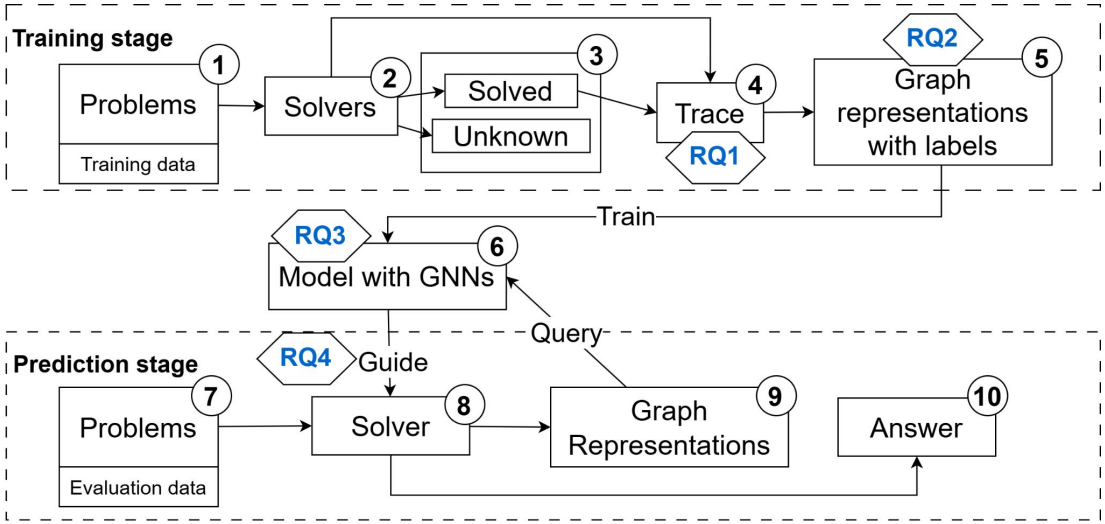
Graph 1 Graph 2 Graph 3 Graph 4 Graph 5

Select Branches to Guide the Solving



- Answer to RQ3 (GNN model):
  - Graph Convolutional Network (GCN).
  - Graph Attention Network (GAT).
  - Graph Isomorphism Network (GIN).

Select Branches to Guide the Solving



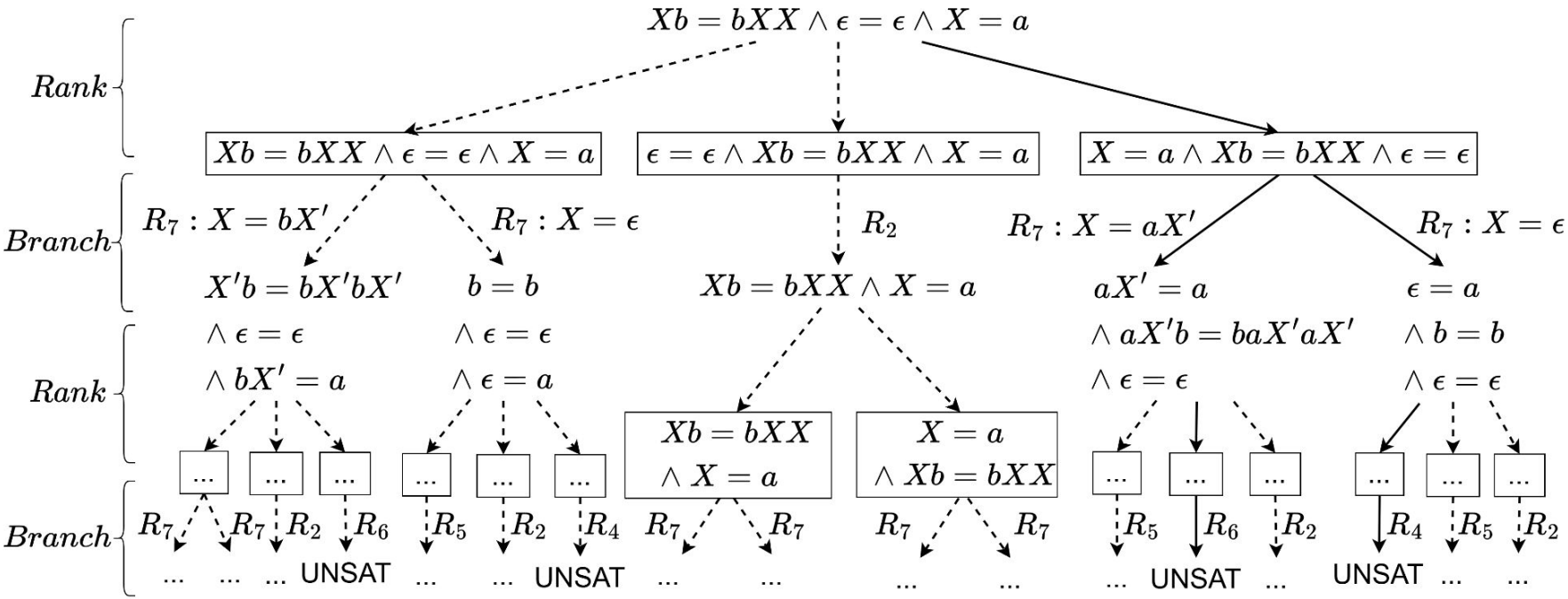
- Answer to RQ4 (integrating methods):
  - Use prediction alone.
  - Combine with random branch selection.

Experimental Results

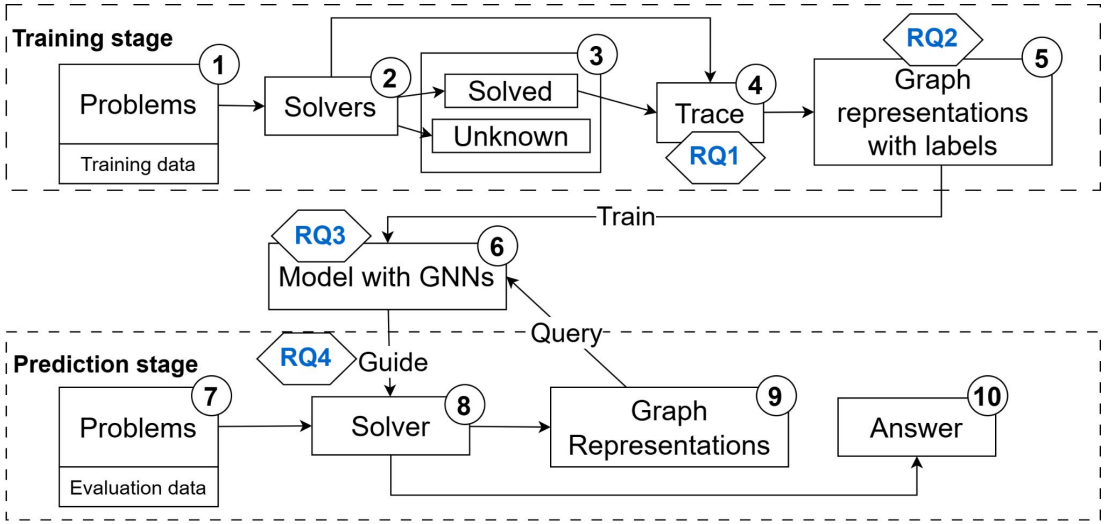
- Evaluated in our word equation solver, Z3, cvc5, etc.

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
2 (1000 in total)	Fixed	33	0	10	1	0	13.2 (1115.2)	- (-)	4.8 (7)	- (-)
	Random	41	0	6			11.7 (3879.5)	- (-)	4.2 (60)	- (-)
	GNN	**71**	0	27			46.0 (1813.5)	- (-)	5.1 (5)	- (-)
	cvc5	4	2	4			2.0 (-)	0.1 (-)	0.1 (-)	- (-)
	Ostrich	14	**43**	44			40.7 (-)	31.8 (-)	2.5 (-)	- (-)
	Woorpje	23	0	2			38.3 (-)	- (-)	0.1 (-)	- (-)
	Z3	6	0	2			0.1 (-)	- (-)	4.2 (-)	- (-)
	Z3-Noodler	19	0	0			45.8 (-)	- (-)	4.2 (-)	- (-)

Solving Word Equation System

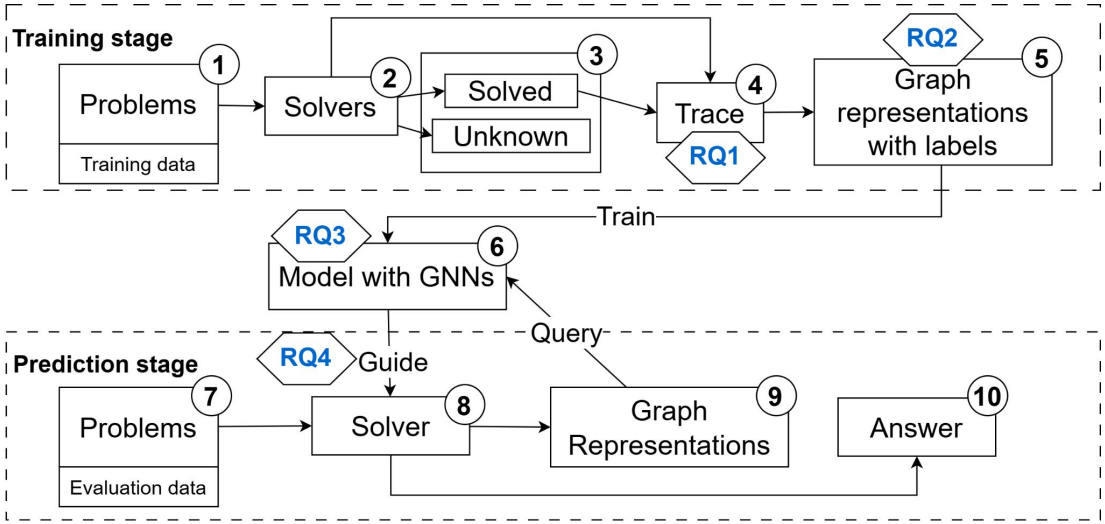


Rank Word Equations to Guide the Solving



- RQ1: Train task.
- RQ2: Graph representation.
- RQ3: GNN models.
- RQ4: Integrating methods.

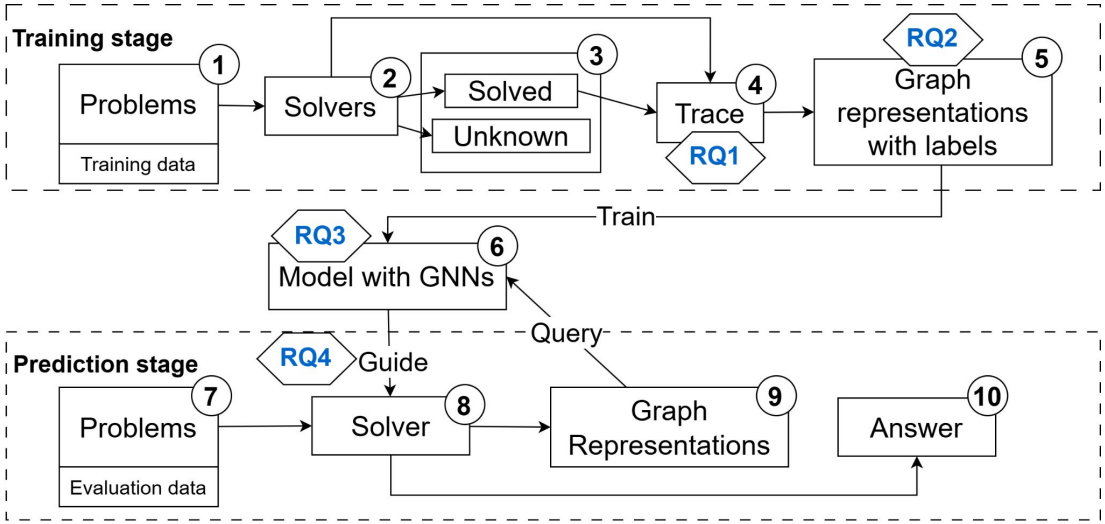
Rank Word Equations to Guide the Solving



- Answer to RQ1 (train task):

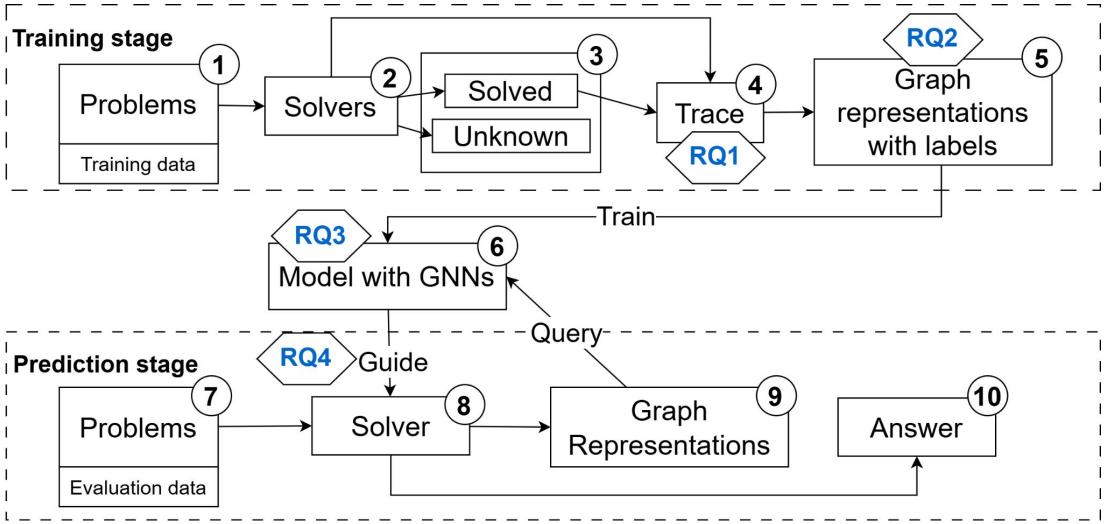
$$XaY = YbX \wedge XabY = YbaX$$

Rank Word Equations to Guide the Solving



- Answer to RQ1 (train task):
  - Learn from MUSes given from other solvers.
  - Learn from shortest path from split algorithm.

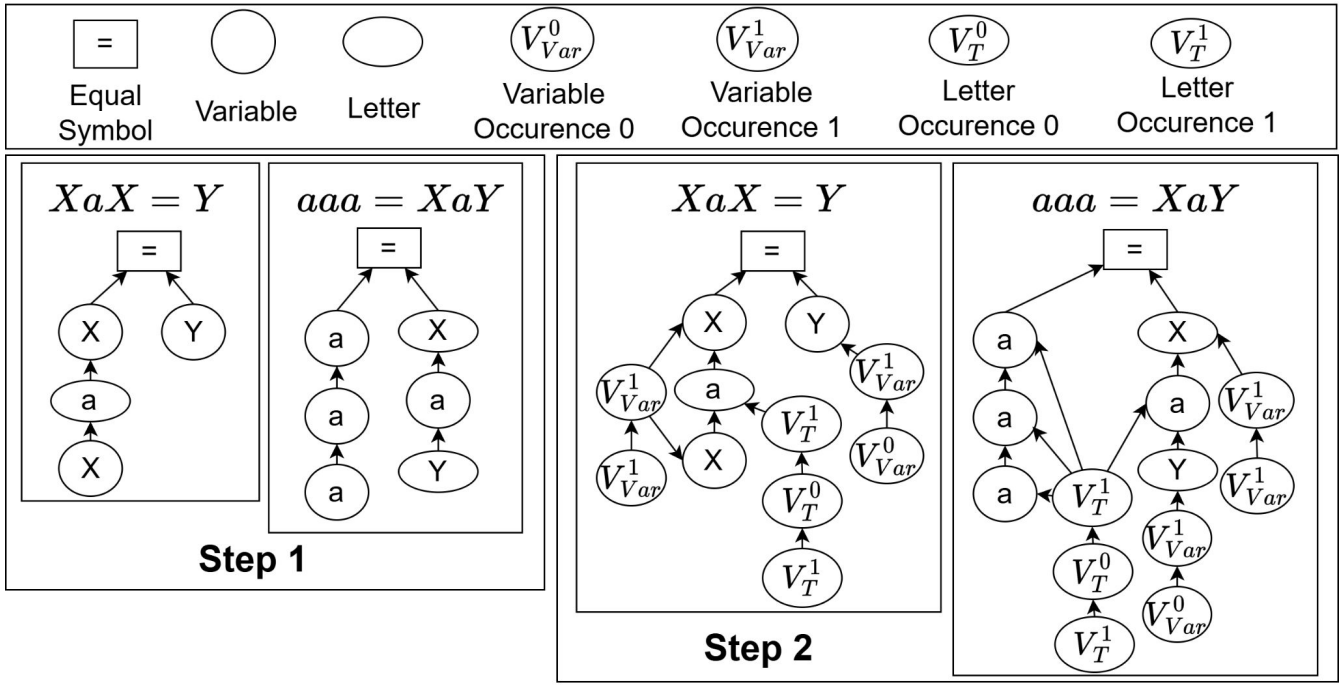
Rank Word Equations to Guide the Solving



- Answer to RQ2 (graph representation):
  - Global information.

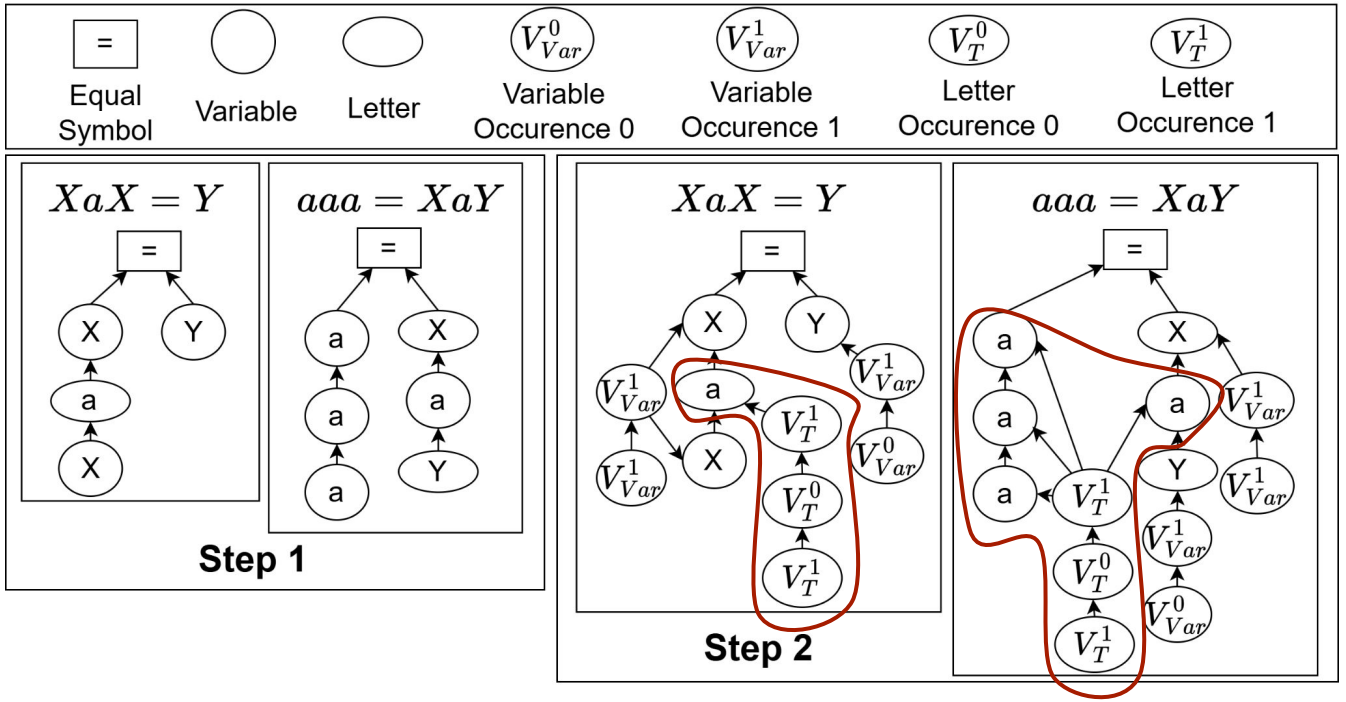
Rank Word Equations (Graph Representation)

$$XaX = Y \wedge aaa = XaY$$

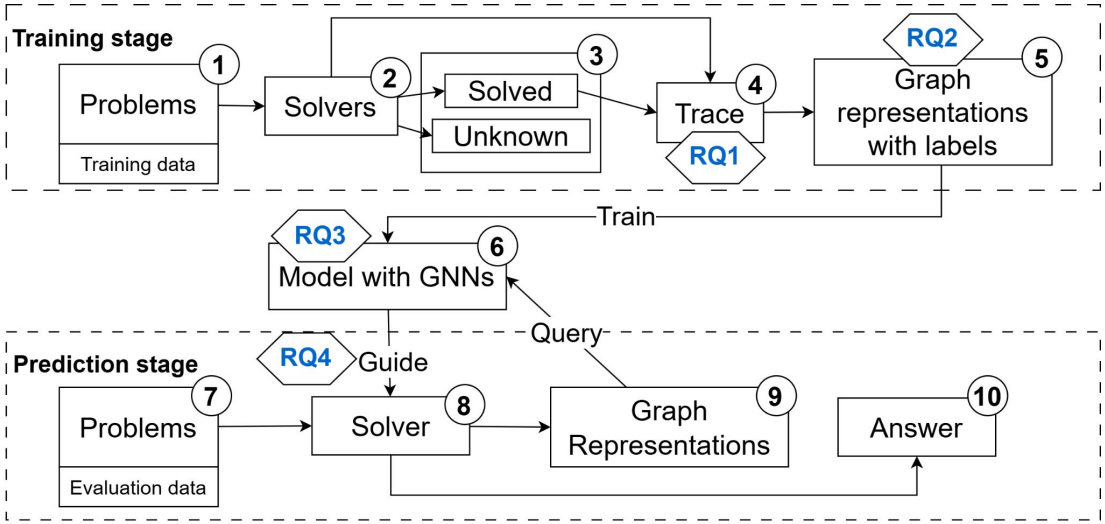


Rank Word Equations (Graph Representation)

$$XaX = Y \wedge aaa = XaY$$

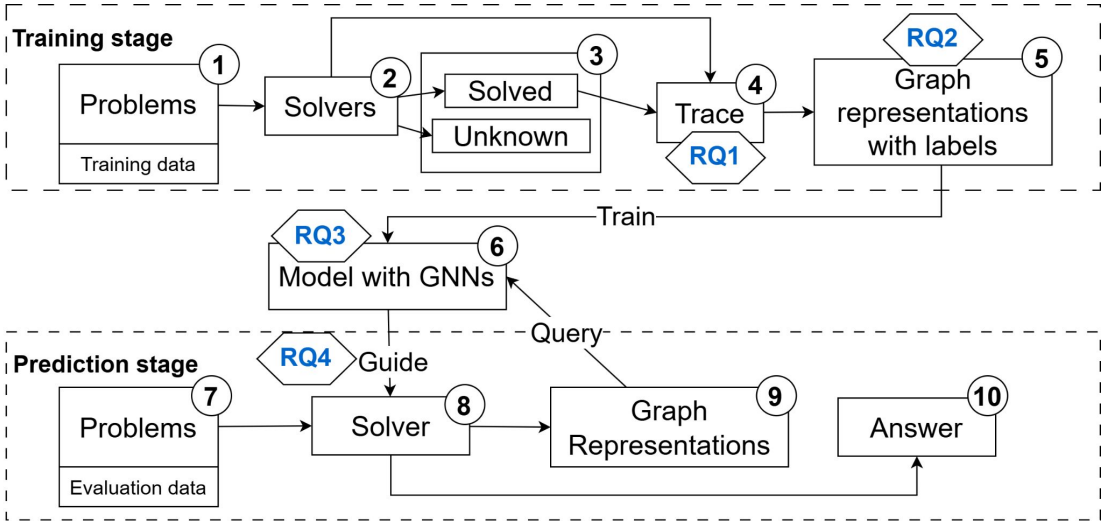


Rank Word Equations to Guide the Solving



- Answer to RQ3 (GNN model):
  - GCN.
  - GCN+GIN.
  - GNN filters.

Rank Word Equations to Guide the Solving



- Answer to RQ4 (integrating method):
  - Different frequency (e.g., one-short).
  - Combine with random ranking strategy.
  - Combine with manually designed heuristics.

Experimental Results

- Evaluated in our word equation solver, Z3, cvc5, etc.

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNSAT	UNI	CS	CU	SAT	UNSAT	CS	CU
A1	DragonLi	24	955	0	13	678	5.6 (244.8)	6.5 (1085.3)	5.0 (94.4)	5.7 (126.3)
	Random-DragonLi	22	944	0			5.6 (198.8)	6.3 (932.6)	5.6 (137.6)	5.7 (180.5)
	GNN-DragonLi	**24**	**961**	0			6.1 (164.7)	7.5 (1974.8)	6.1 (96.4)	6.3 (**60.5**)
	cvc5	**24**	952	1			0.5	0.6	0.1	0.3
	Z3	17	960	0			8.7	0.4	1.1	0.1
	Z3-Noodler	22	939	2			5.7	0.3	4.8	0.1
	Ostrich	17	931	0			15.0	5.5	8.0	4.7
	Woorpje	23	744	0			3.0	12.5	0.1	12.2

Answers to Research Questions

- RQ1: What are good encodings of symbolic decision processes as training tasks?
  - Encode the problems to classification task.
  - Collect train data from multiple sources.

Answers to Research Questions

- RQ1: What are good encodings of symbolic decision processes as training tasks?
  - Encode the problem to classification task.
  - Collect train data from multiple sources.
- RQ2: What is the most effective format for representing formulas in deep learning?
  - The graph representation must include all syntactic elements.
  - Use compact graph encoding (e.g., merge identical nodes) .

Answers to Research Questions

- RQ1: What are good encodings of symbolic decision processes as training tasks?
  - Encode the problem to classification task.
  - Collect train data from multiple sources.
- RQ2: What is the most effective format for representing formulas in deep learning?
  - The graph representation must include all syntactic elements.
  - Use compact graph encoding (e.g., merge identical nodes) .
- RQ3: Which deep learning technique is best suited for feature extraction from formulas?
  - GCN serves as baseline.
  - GAT, GIN, R-HyGNN, etc.

Answers to Research Questions

- RQ1: What are good encodings of symbolic decision processes as training tasks?
  - Encode the problem to classification task.
  - Collect train data from multiple sources.
- RQ2: What is the most effective format for representing formulas in deep learning?
  - The graph representation must include all syntactic elements.
  - Use compact graph encoding (e.g., merge identical nodes) .
- RQ3: Which deep learning technique is best suited for feature extraction from formulas?
  - GCN serves as baseline.
  - GAT, GIN, R-HyGNN, etc.
- RQ4: What are the methods for integrating the trained model into algorithms?
  - Cache embeddings.
  - Combine with predefined heuristics did not yield the best performance.

Conclusions

- A deep learning-based framework for decision problems in symbolic methods.

- Two instances of the framework.
  - CHC solver.
  - Word equation solver.

Conclusions

- A deep learning-based framework for decision problems in symbolic methods.
- Two instances of the framework
  - CHC solver.
  - Word equation solver.
- Our framework can easily adapt to other decision problems in symbolic methods.
- GNN is still the best option for extracting structural information in symbolic expression.

Future Directions

- Training tasks
  - Reinforcement learning.
  - Sequential model.
  - Generative model.
- Extend to new problem domains.
  - Regular expression in word equations.
  - New theories in CHCs.
- Simultaneously guide multiple decision processes.

Thank you for listening

Word Equation System

A word equation:

$$XabY = YbaX$$

where *a* and *b* are letters,
X, Y, and Z are variables ranging over strings of these letters.

A word equation system (conjunctive word equations):

$$\phi = e_1 \wedge \ldots \wedge e_n,$$

where e_i is a word equation.

$$XaY = YbX \wedge XabY = YbaX$$

Motivating Examples

Prove:

$$\forall x.\, L_1(x) \leftarrow true \wedge$$
$$\forall x.\, L_2(x) \leftarrow \wedge x > 0 \wedge$$
$$\forall x, x'.\, L_1(x') \leftarrow L_2(x) \wedge x' = x - 1 \wedge$$
$$\forall x.\, L_3(x) \leftarrow L_1(x) \wedge x \leq 0 \wedge$$
$$\forall x.\, false \leftarrow L_3(x) \wedge x = 0$$

For Simplicity:

$$C1:\ L_1(x) \leftarrow true$$
$$C2:\ L_2(x) \leftarrow L_1(x) \wedge x > 0$$
$$C3:\ L_1(x') \leftarrow L_2(x) \wedge x' = x - 1$$
$$C4:\ L_3(x) \leftarrow L_1(x) \wedge x \leq 0$$
$$C5:\ false \leftarrow L_3(x) \wedge x = 0$$

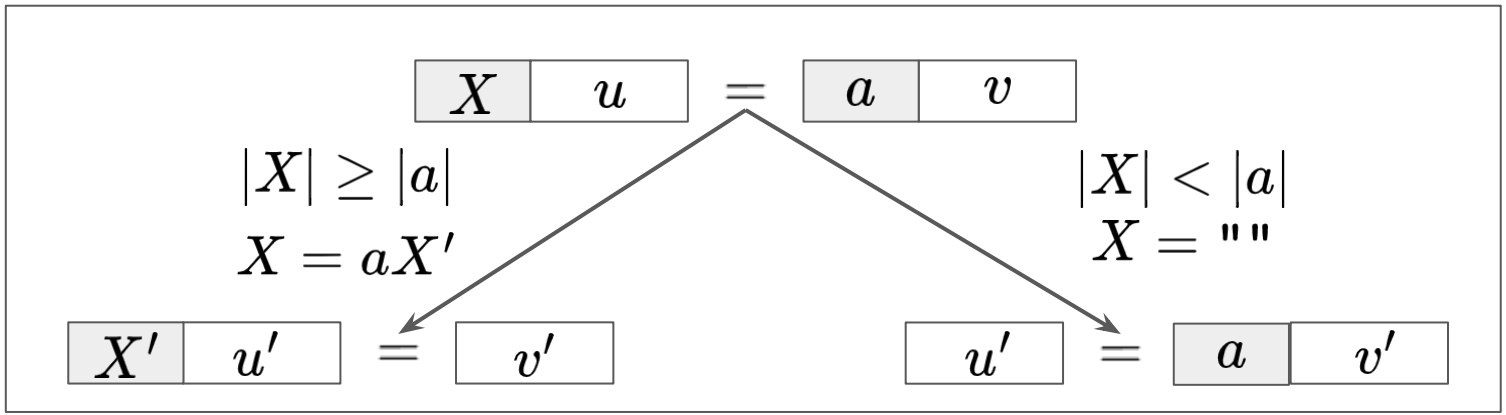
Motivating Examples (Program Verification)

Prove:

$$\forall x.\, L_1(x) \leftarrow true \wedge$$
$$\forall x.\, L_2(x) \leftarrow \wedge x > 0 \wedge$$
$$\forall x, x'.\, L_1(x') \leftarrow L_2(x) \wedge x' = x - 1 \wedge$$
$$\forall x.\, L_3(x) \leftarrow L_1(x) \wedge x \leq 0 \wedge$$
$$\forall x.\, false \leftarrow L_3(x) \wedge x = 0$$

Solving a Word Equation System

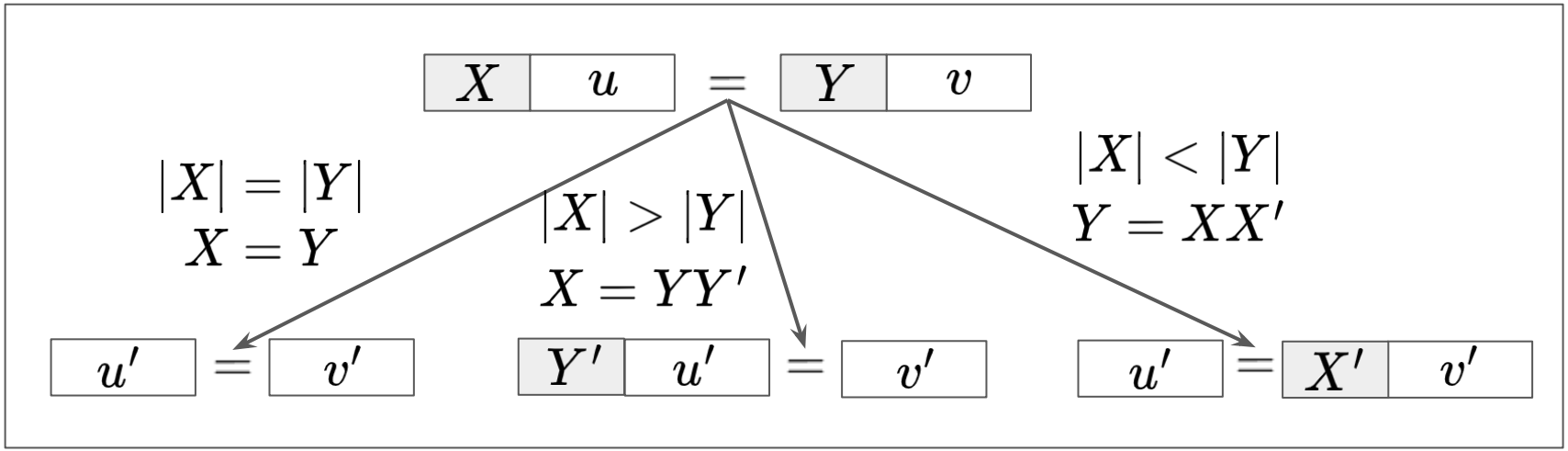
- Split Algorithm (branching example)
  - First terms are variable and terminal



u, v, u', v' are terms, a is a letter, and X, X' are variables

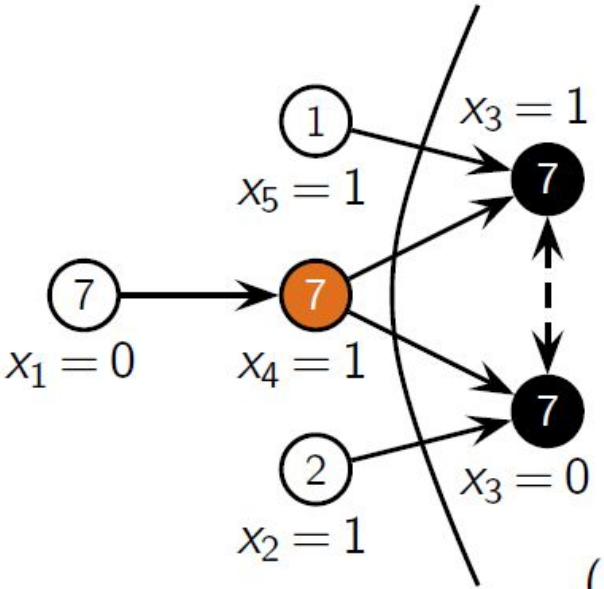
Solving a Word Equation System

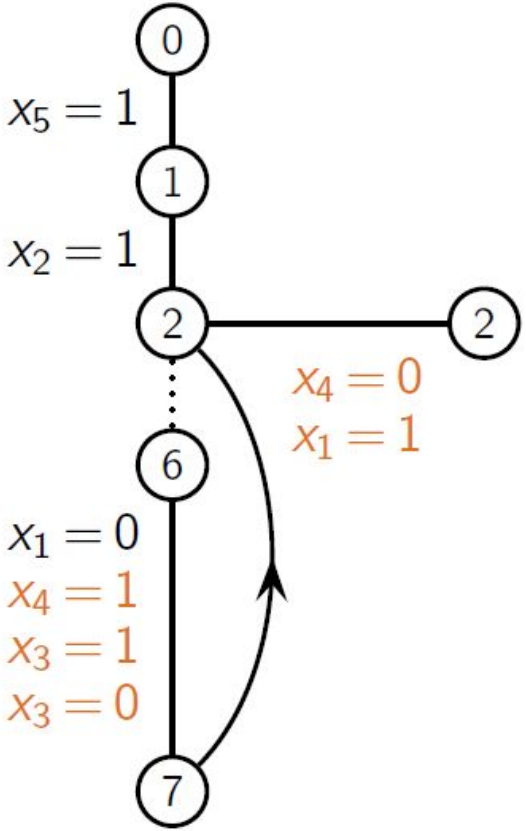
- Split Algorithm (branching example)
  - First terms are variables



u, v, u', v' are terms, and X, Y, X', Y' are variables

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$





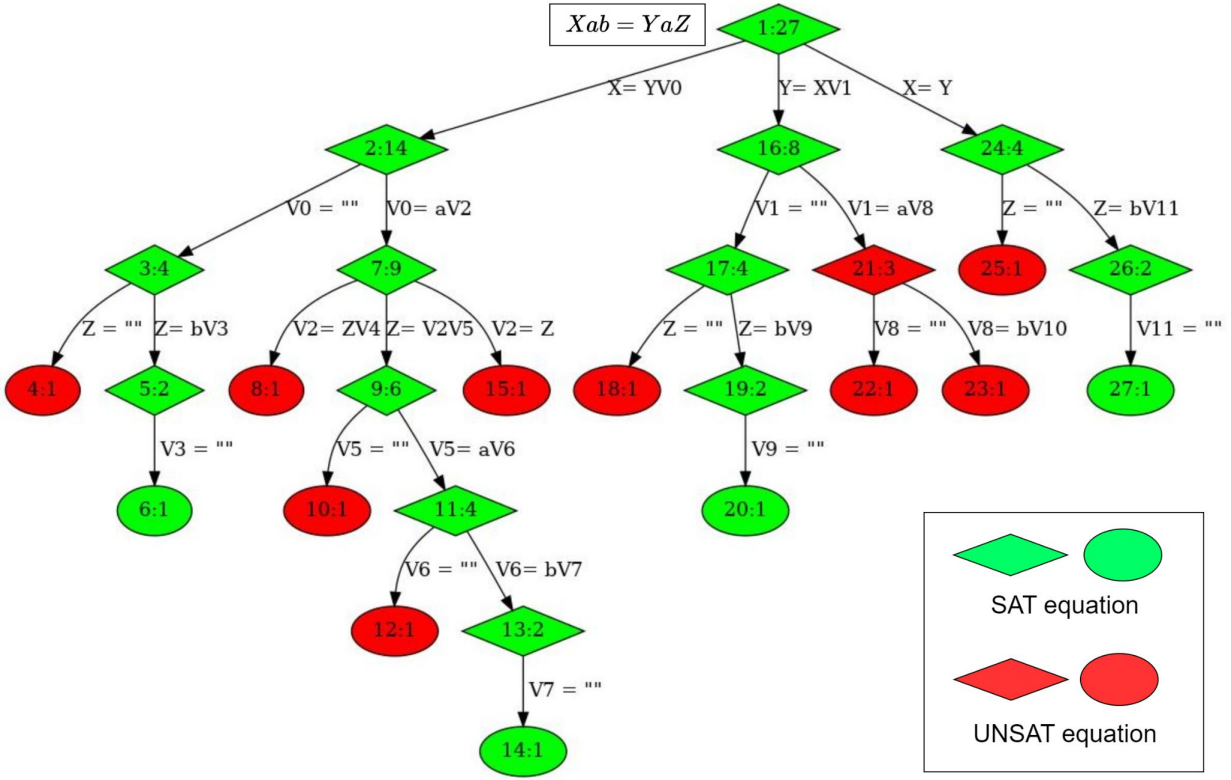
$(\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5)$

Symbolic Methods

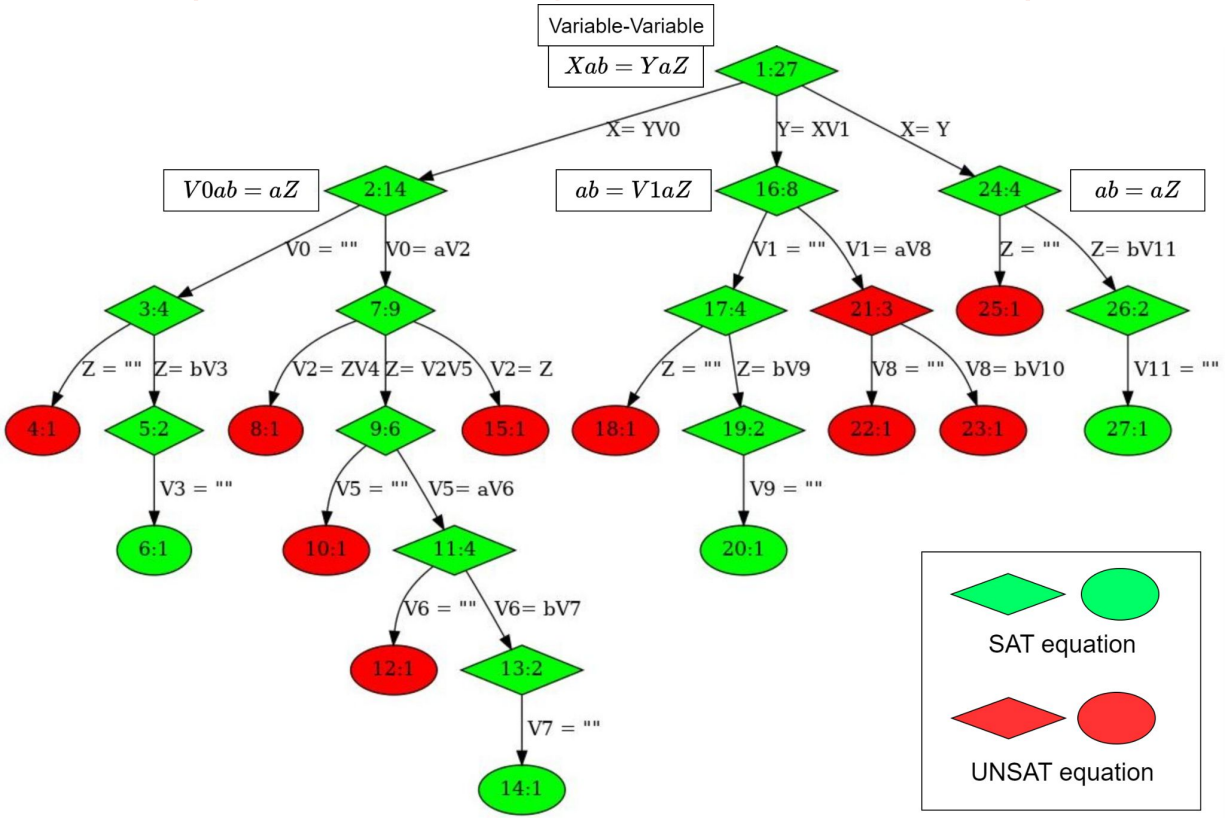
- Approaches
  - Automatic Theorem Provers (ATPs)
    - An example to show why this is interesting
  - Boolean Satisfiability (SAT)/Satisfiability Modulo Theories (SMT) solvers
  - Constrained Horn Clause (CHC) solvers

- Challenges
  - Complex representation
  - Theory handling
  - **Scalability**

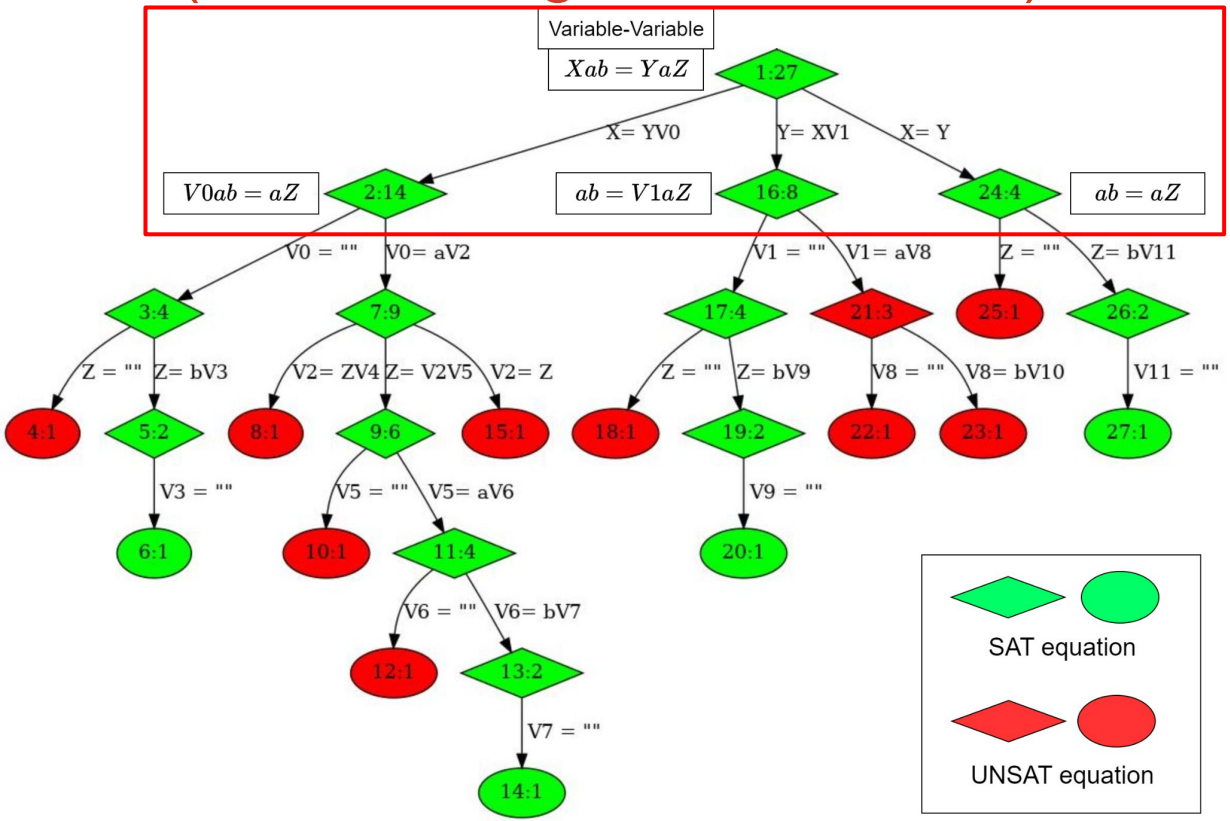
Split Algorithm (Constructing the Proof Tree)



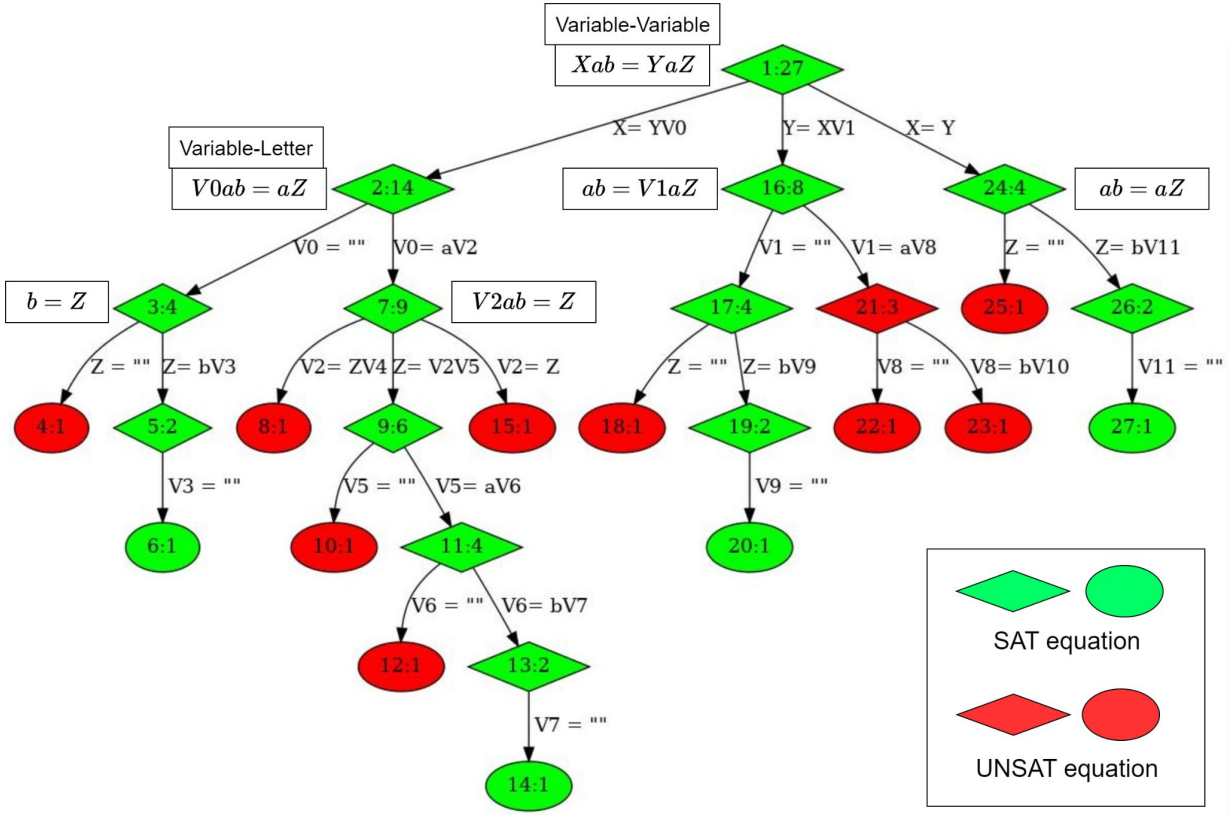
Split Algorithm (Constructing the Proof Tree)



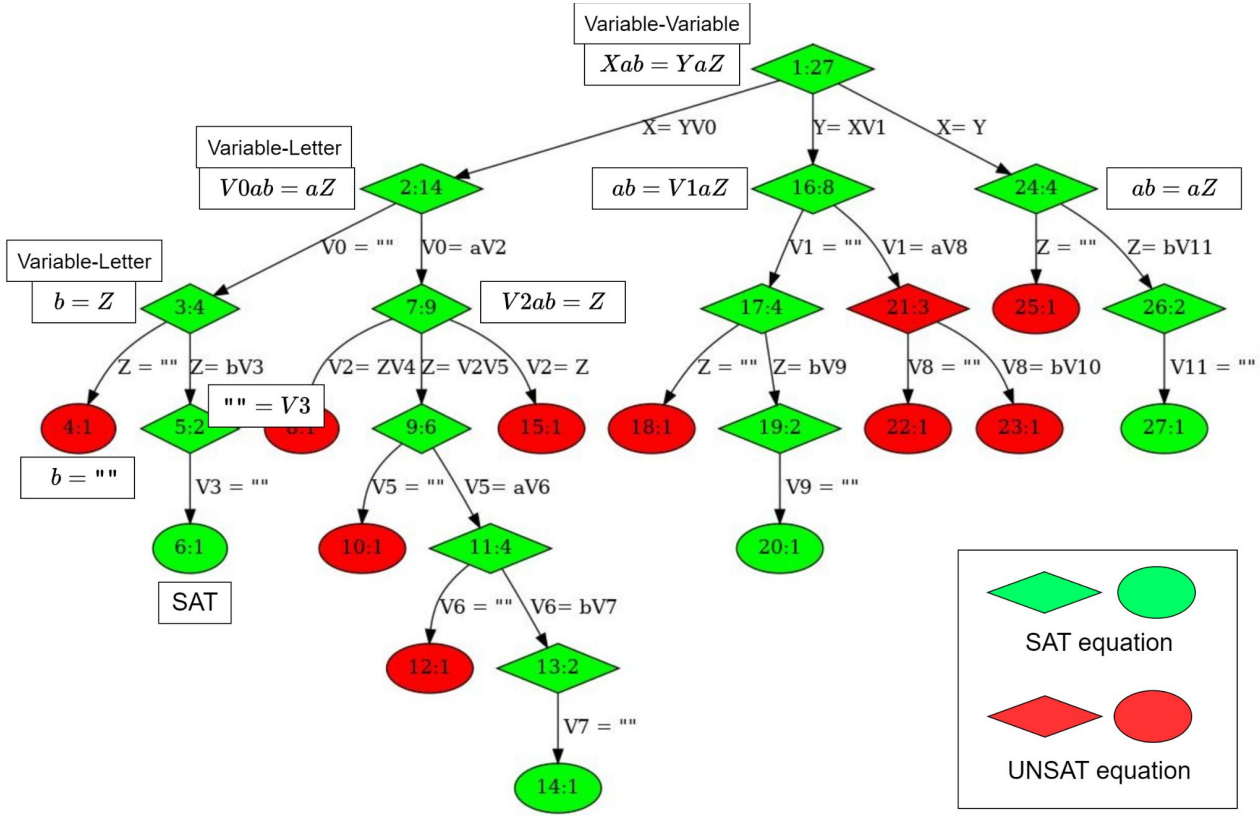
Split Algorithm (Constructing the Proof Tree)



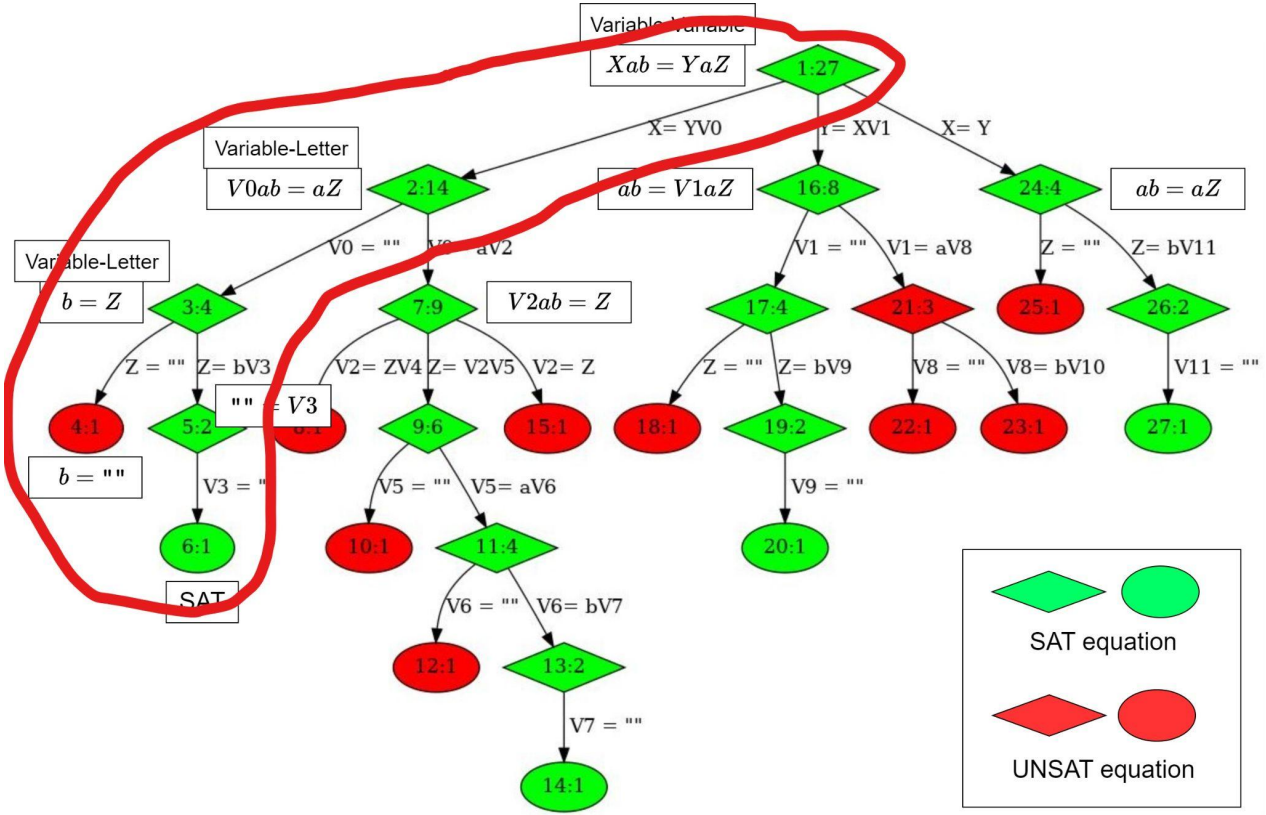
Split Algorithm (Constructing the Proof Tree)



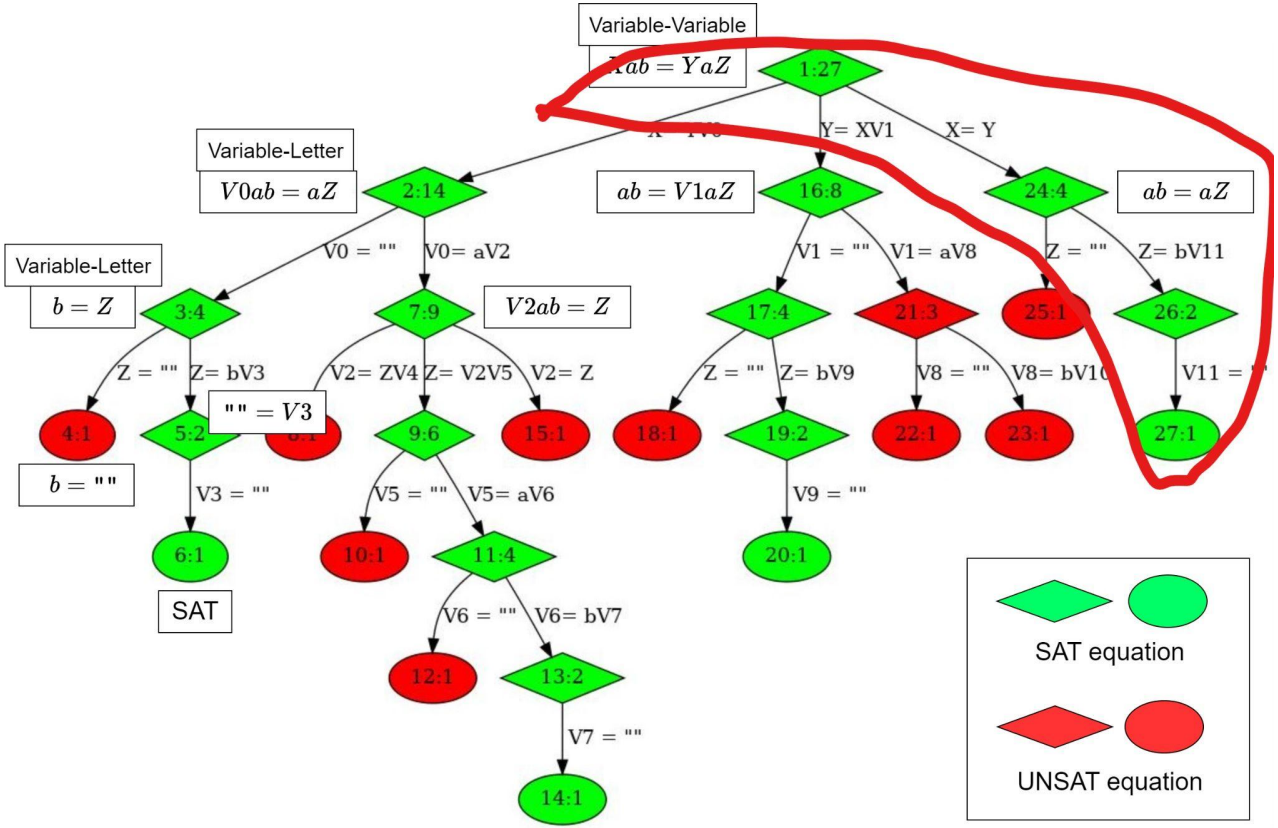
Split Algorithm (Constructing the Proof Tree)



Split Algorithm (A Path Leading to SAT)

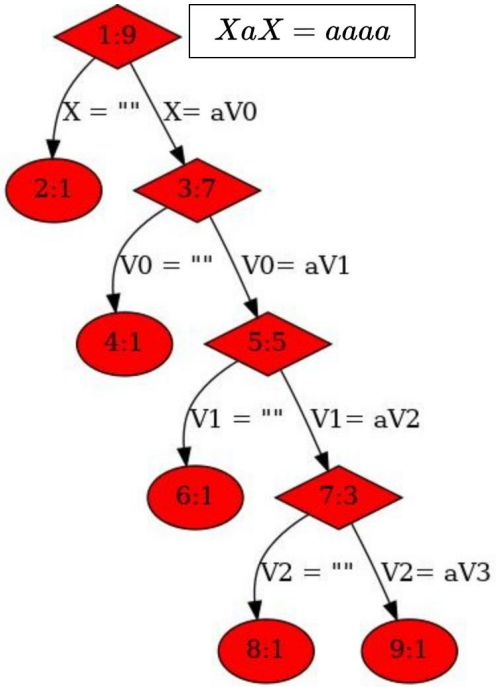


Split Algorithm (A Shortest Path Leading to SAT)

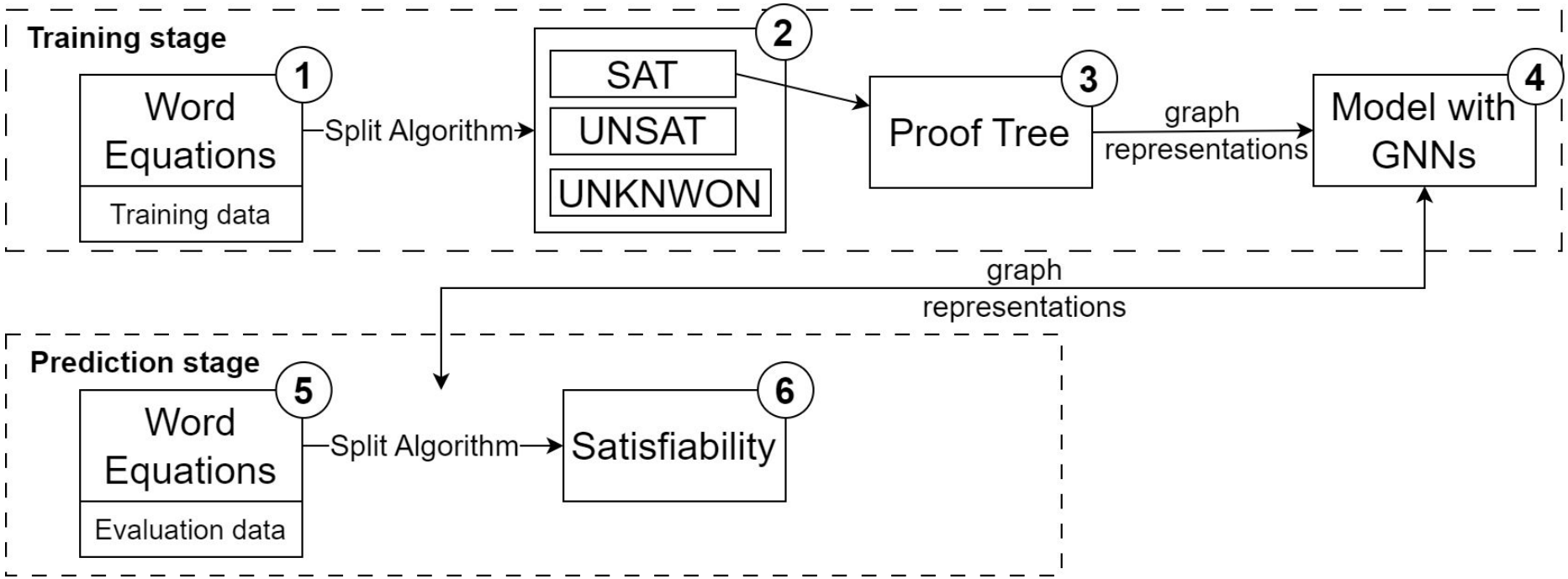


Split Algorithm (Proof Tree for UNSAT Problem)

- Need explore all paths to conclude UNSAT



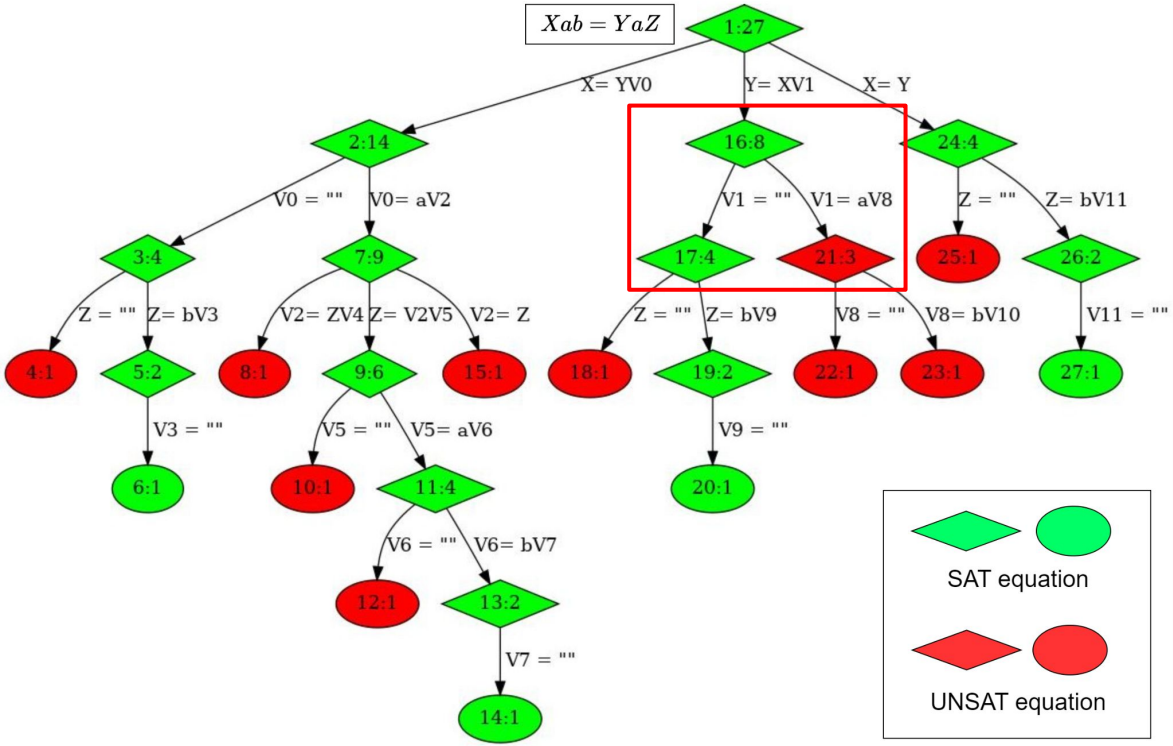
Working Pipeline (Data-driven Based Heuristic)



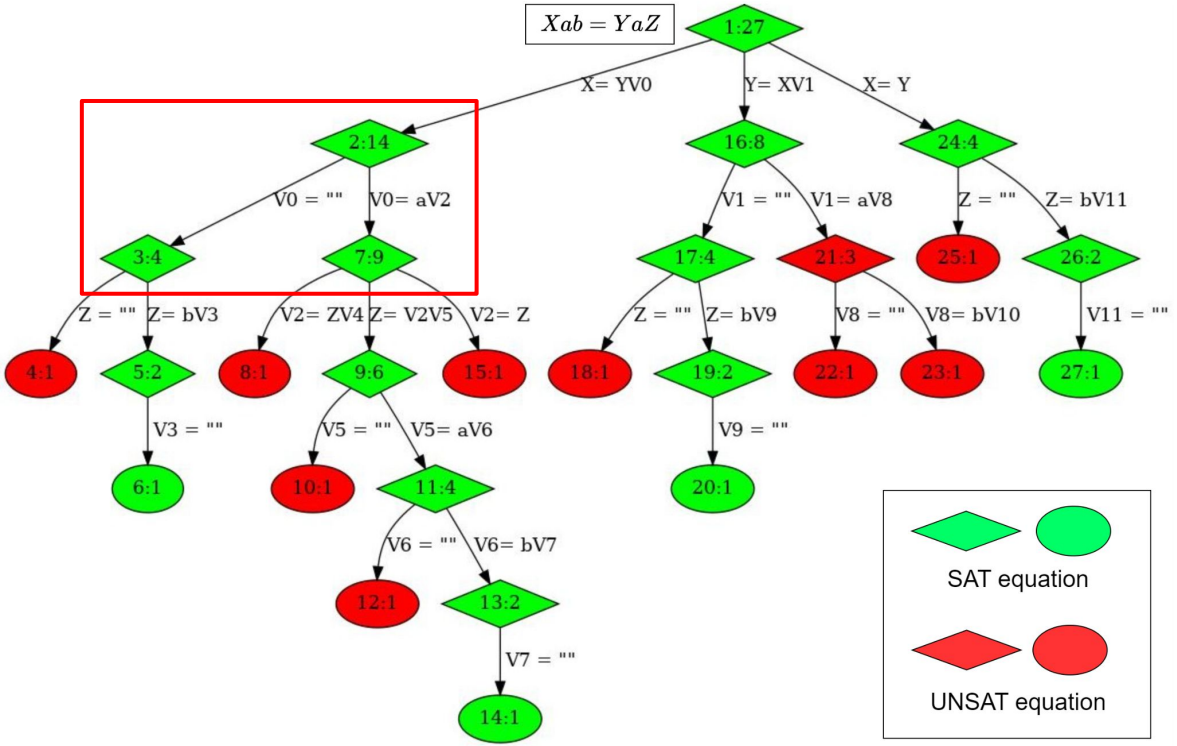
Graph Neural Networks (GNNs)

- A set of fully connected neural networks
- Take graph as input, output node and graph representations
- Can capture the structural features of graph

Label the Training Data at a Branch Point

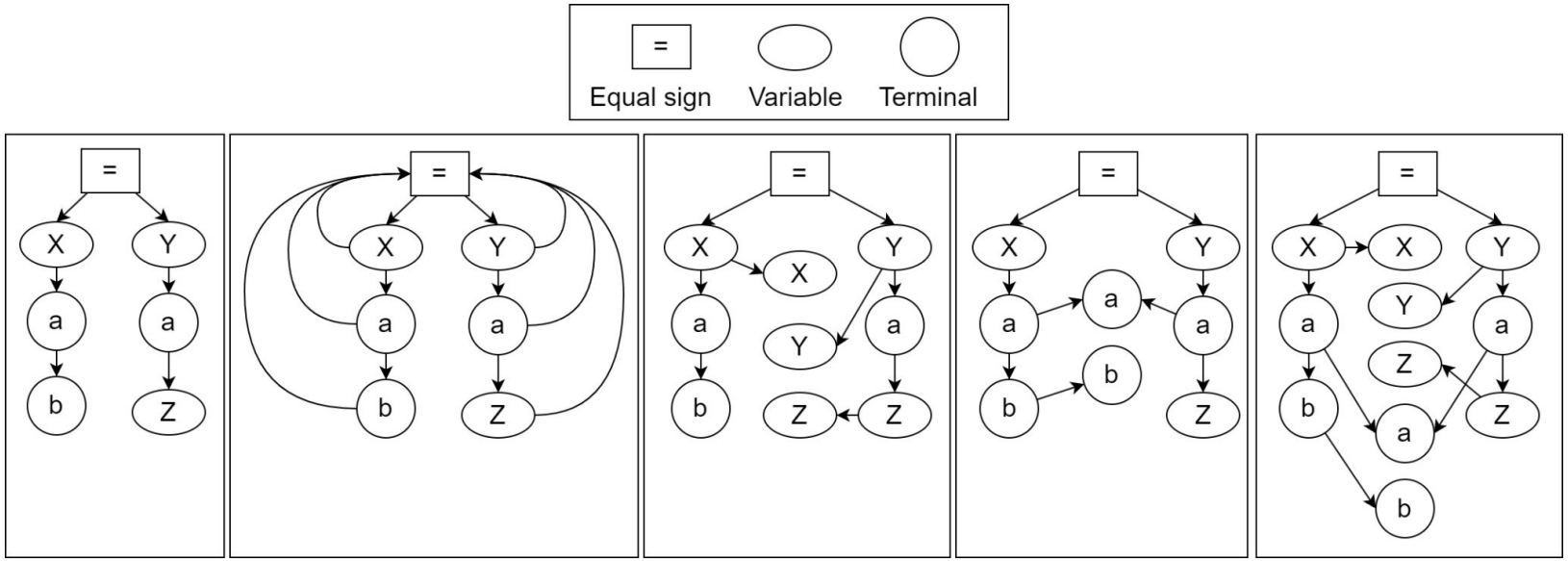


Label the Training Data at a Branch Point

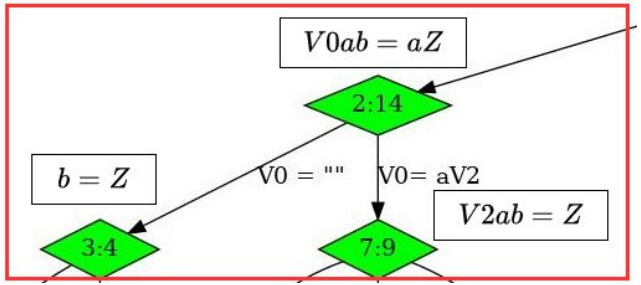


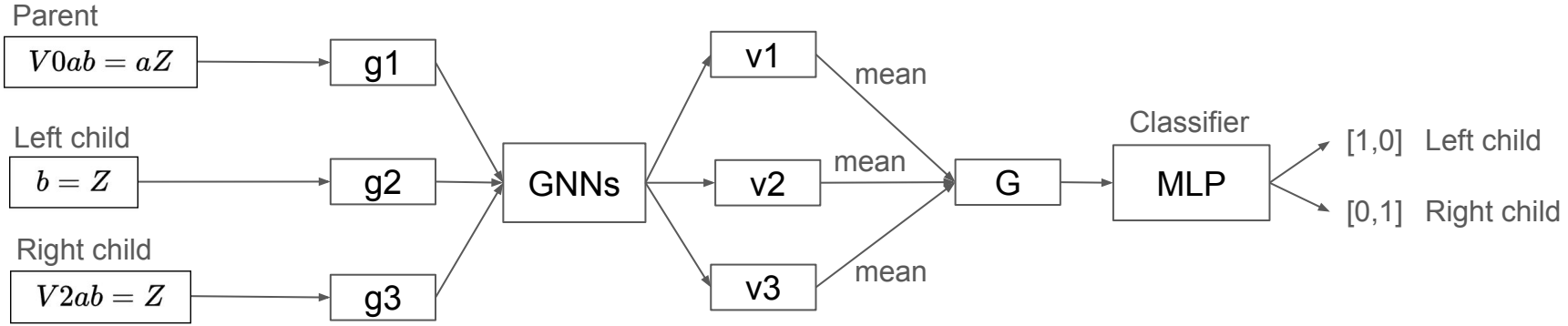
Graph Representations of Word Equation

Equation: $Xab = YaZ$



Model Structure





Benchmarks 1-4

1. Randomly generated SAT word equation

$$aksCwqeuafhkajshfweeta = aksfjdfbabDeuafhkajshBCDta$$

Benchmarks 1-4

1. Randomly generated SAT word equation
2. Word equation with a particular pattern [5]

$$X_n a X_n b X_{n-1} b X_{n-2} \cdots b X_1 = a X_n X_{n-1} X_{n-1} b X_{n-2} b \cdots b X_1 X_1 baa$$

[5] Day, J.D., Ehlers, T., Kulczynski, M., Manea, F., Nowotka, D., Poulsen, D.B.: On solving word equations using SAT. Reachability Problems, 93–106. (2019)

Benchmarks 1-4

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1

$$kSY = WHXGk$$
$$\wedge\, vZX = vtntssemtvnetm$$
$$\wedge\, yffyyfFWVff = yffyyfyXLGZfU$$
$$\wedge\, NRQxgGI = xxgggFJTK$$

Benchmarks 1-4

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

$$AabcdBCefDEghF = BciabADGCFHE$$

$$abAaBabcdCdcDE = AbeFCcdbaBDcdGfg$$

Benchmarks 1-4 (Overview)

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

Table 1: Number of SAT (✓), UNSAT (×), UNKNOWN (∞), and evaluation (Eval) problems in the four benchmarks

Benchmark 1 Total: 3000				Benchmark 2 Total: 21000				Benchmark 3 Total: 41000				Benchmark 4 Total: 2310			
2000			Eval	20000			Eval	40000			Eval	1855			Eval
✓	×	∞		✓	×	∞		✓	×	∞		✓	×	∞	
1997	0	3	1000	1293	0	18707	1000	1449	1137	37414	1000	1673	16	166	455

Benchmarks 1-4 (Overview)

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

Table 1: Number of SAT (✓), UNSAT (×), UNKNOWN (∞), and evaluation (Eval) problems in the four benchmarks

Benchmark 1 Total: 3000				Benchmark 2 Total: 21000				Benchmark 3 Total: 41000				Benchmark 4 Total: 2310			
2000			Eval	20000			Eval	40000			Eval	1855			Eval
✓	×	∞		✓	×	∞		✓	×	∞		✓	×	∞	
1997	0	3	1000	1293	0	18707	1000	1449	1137	37414	1000	1673	16	166	455

Benchmarks 1-4 (Overview)

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

Table 1: Number of SAT (✓), UNSAT (×), UNKNOWN (∞), and evaluation (Eval) problems in the four benchmarks

Benchmark 1 Total: 3000				Benchmark 2 Total: 21000				Benchmark 3 Total: 41000				Benchmark 4 Total: 2310			
2000			Eval	20000			Eval	40000			Eval	1855			Eval
✓	×	∞		✓	×	∞		✓	×	∞		✓	×	∞	
1997	0	3	1000	1293	0	18707	1000	1449	1137	37414	1000	1673	16	166	455

Benchmarks 1-4 (Overview)

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

Table 1: Number of SAT (✓), UNSAT (×), UNKNOWN (∞), and evaluation (Eval) problems in the four benchmarks

Benchmark 1 Total: 3000				Benchmark 2 Total: 21000				Benchmark 3 Total: 41000				Benchmark 4 Total: 2310			
2000			Eval	20000			Eval	40000			Eval	1855			Eval
✓	×	∞		✓	×	∞		✓	×	∞		✓	×	∞	
1997	0	3	1000	1293	0	18707	1000	1449	1137	37414	1000	1673	16	166	455

Experimental Results (Benchmark 1)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
1 (1000 SAT)	Fixed	999	-	0	777	0	4.1 (182.0)	- (-)	4.0 (169.0)	- (-)
	Random	996	-	0			4.2 (349.6)	- (-)	4.1 (269.8)	- (-)
	GNN	995	-	0			7.6 (215.7)	- (-)	7.0 (162.3)	- (-)
	cvc5	**1000**	-	0			0.1 (-)	- (-)	0.1 (-)	- (-)
	Ostrich	918	-	0			20.4 (-)	- (-)	19.6 (-)	- (-)
	Woorpje	967	-	0			1.6 (-)	- (-)	0.5 (-)	- (-)
	Z3	902	-	0			3.4 (-)	- (-)	2.4 (-)	- (-)
	Z3-Noodler	935	-	0			1.9 (-)	- (-)	1.1 (-)	- (-)

Experimental Results (Benchmark 1)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
1 (1000 SAT)	Fixed	999	-	0	777	0	4.1 (182.0)	- (-)	4.0 (169.0)	- (-)
	Random	996	-	0			4.2 (349.6)	- (-)	4.1 (269.8)	- (-)
	GNN	995	-	0			7.6 (215.7)	- (-)	7.0 (162.3)	- (-)
	cvc5	**1000**	-	0			0.1 (-)	- (-)	0.1 (-)	- (-)
	Ostrich	918	-	0			20.4 (-)	- (-)	19.6 (-)	- (-)
	Woorpje	967	-	0			1.6 (-)	- (-)	0.5 (-)	- (-)
	Z3	902	-	0			3.4 (-)	- (-)	2.4 (-)	- (-)
	Z3-Noodler	935	-	0			1.9 (-)	- (-)	1.1 (-)	- (-)

Experimental Results (Benchmark 1)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
1 (1000 SAT)	Fixed	999	-	0	777	0	4.1 (182.0)	- (-)	4.0 (169.0)	- (-)
	Random	996	-	0			4.2 (349.6)	- (-)	4.1 (269.8)	- (-)
	GNN	995	-	0			7.6 (215.7)	- (-)	7.0 (162.3)	- (-)
	cvc5	**1000**	-	0			0.1 (-)	- (-)	0.1 (-)	- (-)
	Ostrich	918	-	0			20.4 (-)	- (-)	19.6 (-)	- (-)
	Woorpje	967	-	0			1.6 (-)	- (-)	0.5 (-)	- (-)
	Z3	902	-	0			3.4 (-)	- (-)	2.4 (-)	- (-)
	Z3-Noodler	935	-	0			1.9 (-)	- (-)	1.1 (-)	- (-)

Experimental Results (Benchmark 1)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
1 (1000 SAT)	Fixed	999	-	0	777	0	4.1 (182.0)	- (-)	4.0 (169.0)	- (-)
	Random	996	-	0			4.2 (349.6)	- (-)	4.1 (269.8)	- (-)
	GNN	995	-	0			7.6 (215.7)	- (-)	7.0 (162.3)	- (-)
	cvc5	**1000**	-	0			0.1 (-)	- (-)	0.1 (-)	- (-)
	Ostrich	918	-	0			20.4 (-)	- (-)	19.6 (-)	- (-)
	Woorpje	967	-	0			1.6 (-)	- (-)	0.5 (-)	- (-)
	Z3	902	-	0			3.4 (-)	- (-)	2.4 (-)	- (-)
	Z3-Noodler	935	-	0			1.9 (-)	- (-)	1.1 (-)	- (-)

Experimental Results (Benchmark 2)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
2 (1000 in total)	Fixed	33	0	10	1	0	13.2 (1115.2)	- (-)	4.8 (7)	- (-)
	Random	41	0	6			11.7 (3879.5)	- (-)	4.2 (60)	- (-)
	GNN	**71**	0	27			46.0 (1813.5)	- (-)	5.1 (5)	- (-)
	cvc5	4	2	4			2.0 (-)	0.1 (-)	0.1 (-)	- (-)
	Ostrich	14	**43**	44			40.7 (-)	31.8 (-)	2.5 (-)	- (-)
	Woorpje	23	0	2			38.3 (-)	- (-)	0.1 (-)	- (-)
	Z3	6	0	2			0.1 (-)	- (-)	4.2 (-)	- (-)
	Z3-Noodler	19	0	0			45.8 (-)	- (-)	4.2 (-)	- (-)

Experimental Results (Benchmark 3)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
3 (1000 in total)	Fixed	32	79	0	23	50	5.2 (1946.2)	65.8 (4227.0)	3.6 (57.0)	38.3 (796.6)
	Random	32	79	0			9.5 (3861.8)	65.0 (4227.0)	3.8 (61.7)	38.5 (796.6)
	GNN	32	65	0			214.3 (1471.2)	1471.2 (1471.2)	4.6 (63.7)	84.0 (796.6)
	cvc5	32	943	2			0.1 (-)	0.3 (-)	0.1 (-)	0.3 (-)
	Ostrich	27	926	0			5.8 (-)	4.7 (-)	4.6 (-)	4.5 (-)
	Woorpje	**34**	723	1			12.4 (-)	12.3 (-)	0.1 (-)	23.2 (-)
	Z3	26	**953**	10			5.6 (-)	0.5 (-)	4.7 (-)	0.1 (-)
	Z3-Noodler	28	926	0			22.7 (-)	0.3 (-)	8.9 (-)	0.1 (-)

Experimental Results (Benchmark 4)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
4 (455 in total)	Fixed	416	6	0	403	2	5.1 (105.5)	17.7 (17119.5)	5.1 (51.0)	5.0 (246)
	Random	415	6	0			4.9 (61.3)	17.9 (17119.5)	4.9 (38.1)	4.4 (246)
	GNN	418	5	0			5.5 (118.3)	31.8 (5019.6)	5.3 (49.0)	8.8 (246)
	cvc5	406	34	0			0.1 (-)	0.1 (-)	0.1 (-)	0.1 (-)
	Ostrich	406	6	0			1.4 (-)	1.2 (-)	1.4 (-)	1.2 (-)
	Woorpje	**420**	2	0			0.2 (-)	3.6 (-)	0.2 (-)	3.6 (-)
	Z3	**420**	10	0			0.1 (-)	0.1 (-)	0.1 (-)	0.1 (-)
	Z3-Noodler	**420**	**35**	1			0.1 (-)	0.1 (-)	0.1 (-)	0.1 (-)

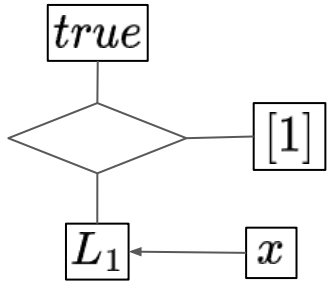
Extract train data

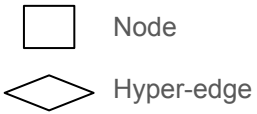
- Binary classification label
  - Union
  - Intersection
  - Single

Label	Clauses	
1	[1] $L_1(x)$	$\leftarrow true$
0	[2] $L_2(x)$	$\leftarrow L_1(x) \wedge x > 0$
0	[3] $L_1(x')$	$\leftarrow L_2(x) \wedge x' = x - 1$
1	[4] $L_3(x)$	$\leftarrow L_1(x) \wedge x \leq 0$
1	[5] $false$	$\leftarrow L_3(x) \wedge x \neq 0$

Represent CHCs by graphs

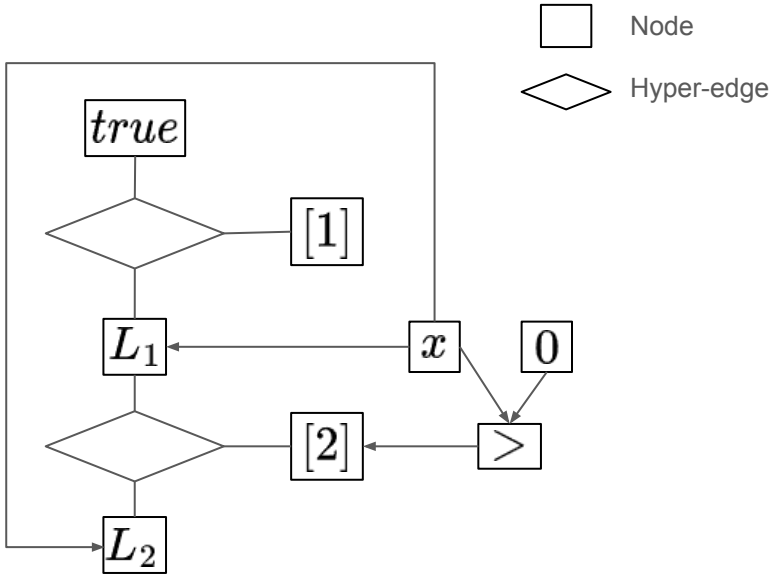
Label	Clauses	
1	[1] $L_1(x)$	$\leftarrow true$
0	[2] $L_2(x)$	$\leftarrow L_1(x) \wedge x > 0$
0	[3] $L_1(x')$	$\leftarrow L_2(x) \wedge x' = x - 1$
1	[4] $L_3(x)$	$\leftarrow L_1(x) \wedge x \leq 0$
1	[5] $false$	$\leftarrow L_3(x) \wedge x \neq 0$





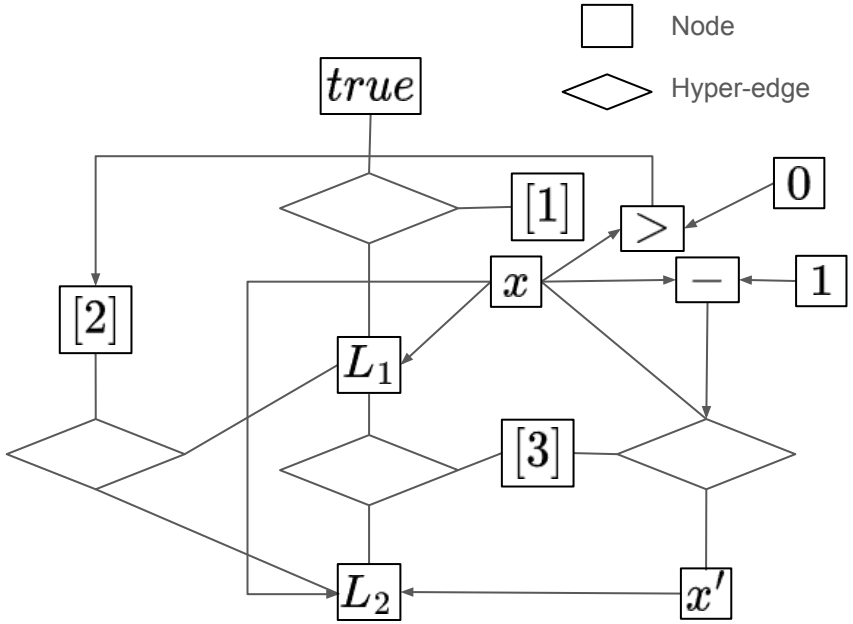
Represent CHCs by graphs

Label		Clauses
1	[1] $L_1(x)$	$\leftarrow true$
0	[2] $L_2(x)$	$\leftarrow L_1(x) \wedge x > 0$
0	[3] $L_1(x')$	$\leftarrow L_2(x) \wedge x' = x - 1$
1	[4] $L_3(x)$	$\leftarrow L_1(x) \wedge x \leq 0$
1	[5] $false$	$\leftarrow L_3(x) \wedge x \neq 0$



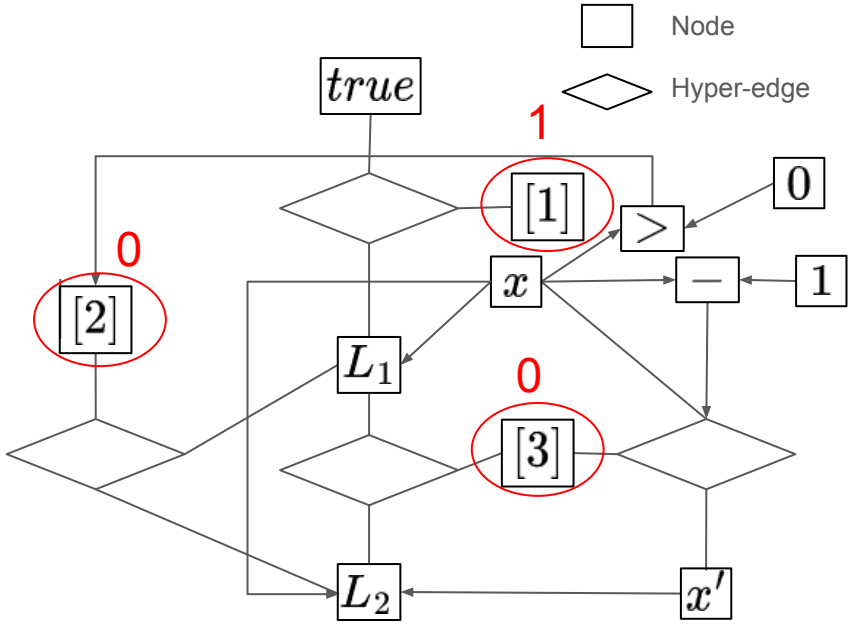
Represent CHCs by graphs

Label		Clauses
1	[1] $L_1(x)$	$\leftarrow true$
0	[2] $L_2(x)$	$\leftarrow L_1(x) \wedge x > 0$
0	[3] $L_1(x')$	$\leftarrow L_2(x) \wedge x' = x - 1$
1	[4] $L_3(x)$	$\leftarrow L_1(x) \wedge x \leq 0$
1	[5] $false$	$\leftarrow L_3(x) \wedge x \neq 0$

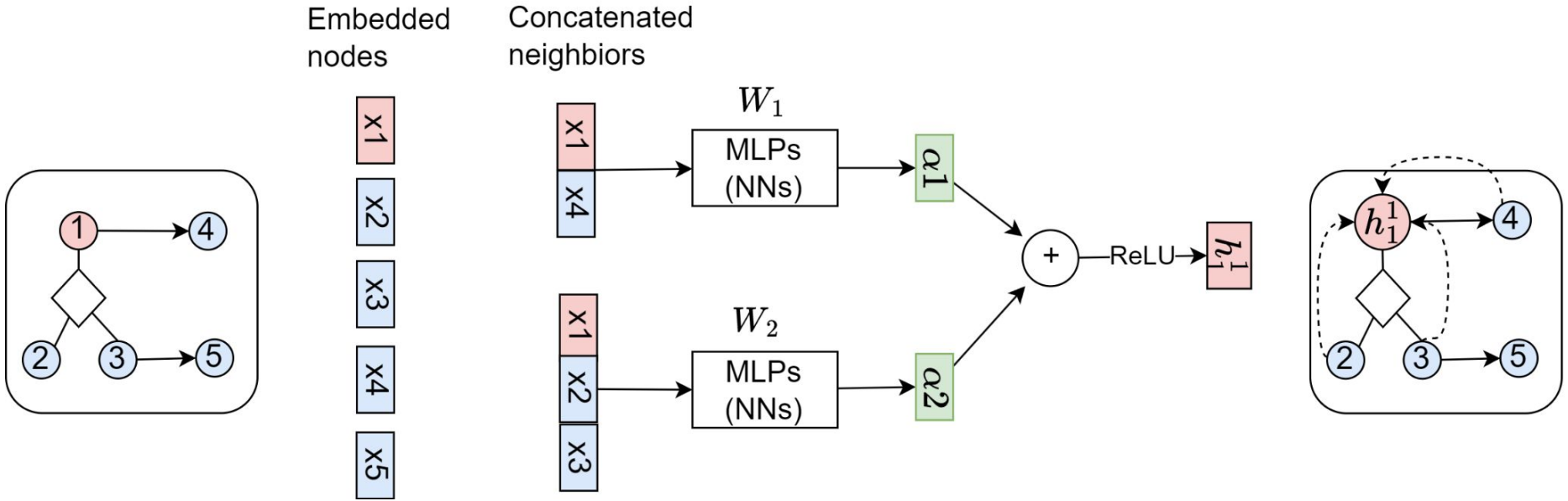


Represent CHCs by graphs

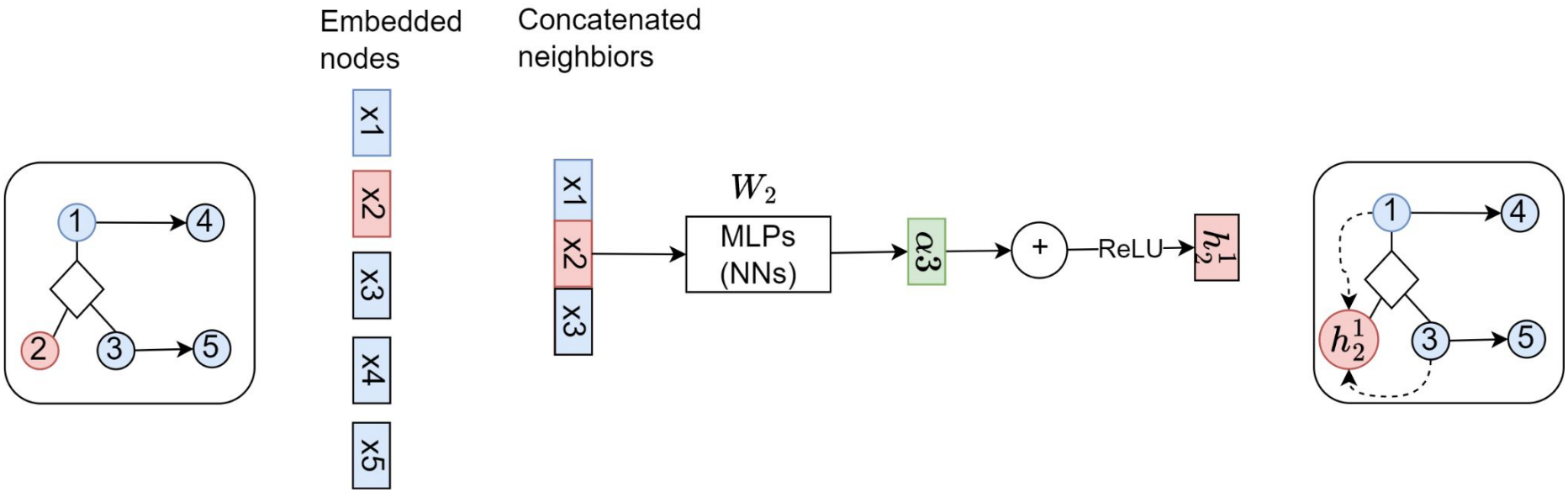
Label		Clauses
1	[1] $L_1(x)$	$\leftarrow true$
0	[2] $L_2(x)$	$\leftarrow L_1(x) \wedge x > 0$
0	[3] $L_1(x')$	$\leftarrow L_2(x) \wedge x' = x - 1$
1	[4] $L_3(x)$	$\leftarrow L_1(x) \wedge x \leq 0$
1	[5] $false$	$\leftarrow L_3(x) \wedge x \neq 0$



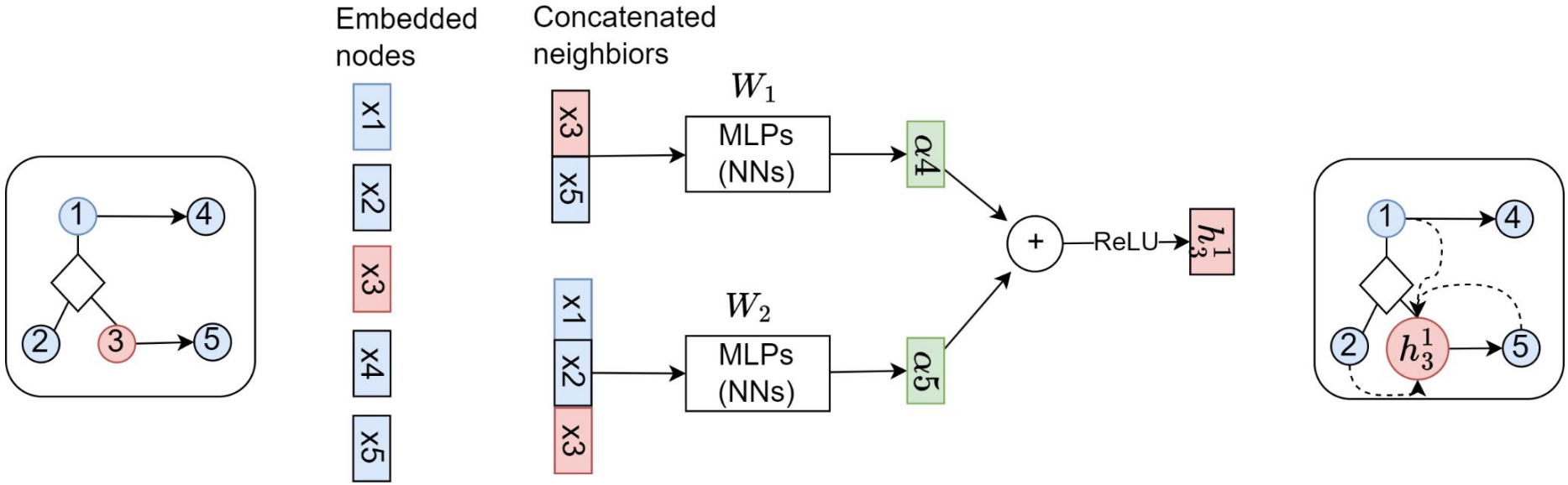
MUSHyperNet Framework (GNN model):



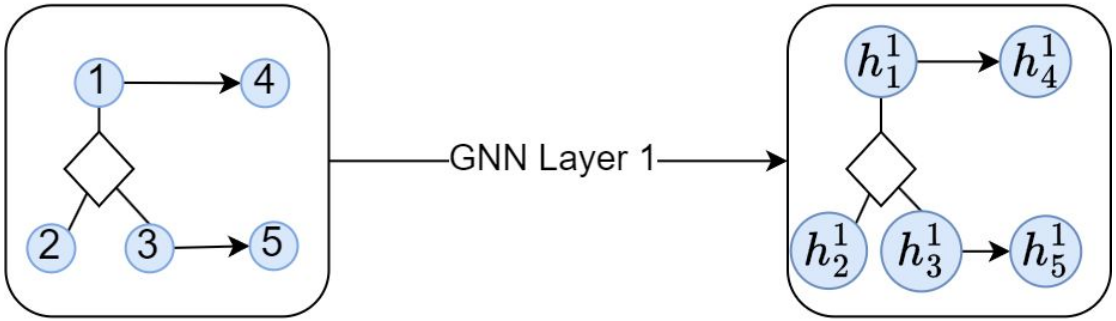
MUSHyperNet Framework (GNN model):



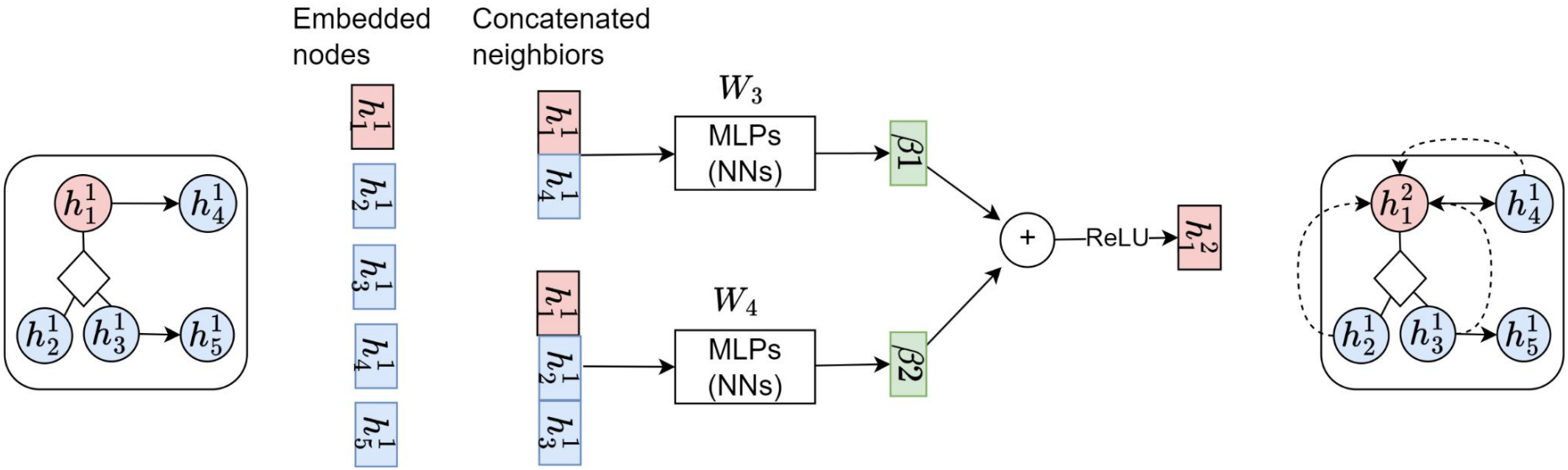
MUSHyperNet Framework (GNN model):



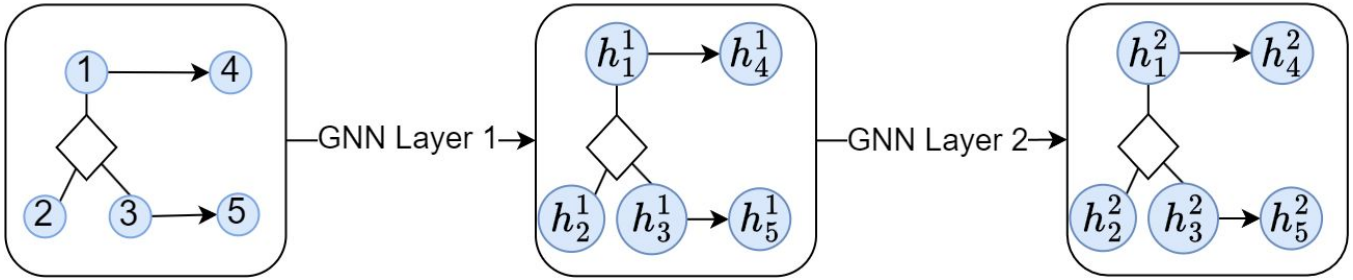
MUSHyperNet Framework (GNN model):



MUSHyperNet Framework (GNN model):



MUSHyperNet Framework (GNN model):



Use predicted MUSes to guide the algorithms

- Prioritize CHCs by using predicted scores of CHCs
  - Use scores alone
  - Combine with original prioritizing scores
    - Add/subtract normalized or ranked scores with coefficient
    - Randomly shift to MUS and original score

Algorithm	Name
CEGAR	Fixed
	Random
	Score
	Rank
	R-Plus
	S-Plus
	R-Minus
	S-Minus
SymEx	Fixed
	Random
	Score
	Rank
	R-Plus
	S-Plus
	R-Minus
	S-Minus
	Two-queue

Experimental results

- Benchmarks from CHC-COMP

Linear LIA problems						Non-linear LIA problems					
8705						8425					
Benchmarks for training				Holdout set		Benchmarks for training				Holdout set	
7834 (90%)				871 (10%)		7579 (90%)				846 (10%)	
UNSAT		SAT	T/O	Eval.	N/A	UNSAT		SAT	T/O	Eval.	N/A
1585		4004	2245	383	488	3315		4010	254	488	358
Train	Valid	N/A				Train	Valid	N/A			
782	87	716				1617	180	1518			

Experimental results (Improved percentage)

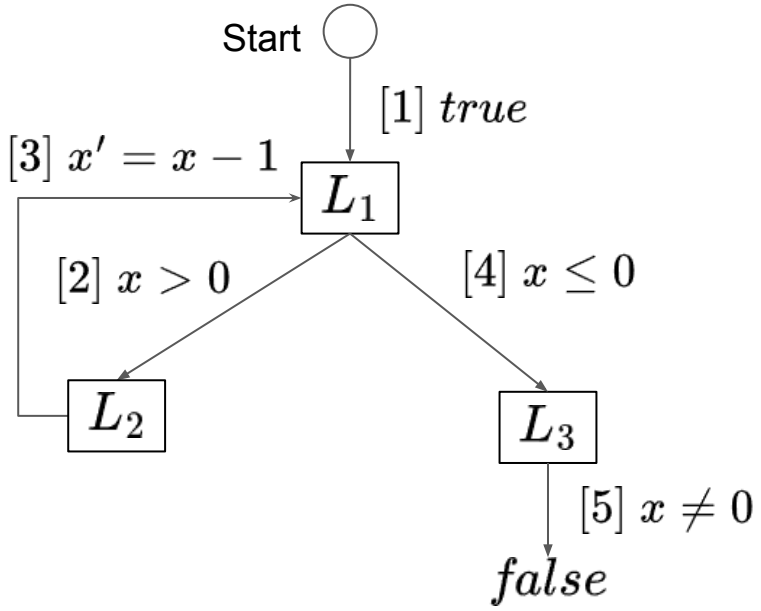
Benchmark Algorithm	MUS data set (best count)	Best ranking function (improvement in %)						
		Number of Solved Problems			Average Time			
		Total	SAT	UNSAT	All	Common	SAT	UNSAT
Linear CEGAR	Union (0)	R-Plus (1.4%)	R-Plus (2.4%)	R-Minus (1.0%)	R-Plus (1.3%)	S-Plus (19.1%)	S-Minus (46.5%)	Rank (31.1%)
	Single (3)	Rank (3.6%)	**R-Plus (4.0%)**	Rank (8.2%)	R-Plus (1.9%)	S-Plus (26.6%)	**R-Minus (57.9%)**	**Rank (36.3%)**
	Intersection (4)	**R-Plus (4.1%)**	S-Plus (0.8%)	**R-Plus (9.3%)**	**R-Plus (3.1%)**	**S-Plus (27.6%)**	R-Minus (45.0%)	S-Plus (0.0%)
Linear SymEx	Union (4)	**Two-Q (1.0%)**	**S-Plus* (0.0%)**	**Random (2.0%)**	Two-Q (0.9%)	R-Minus (12.7%)	**R-Minus (30.2%)**	S-Plus (26.5%)
	Single (3)	S-Minus* (0.5%)	**S-Plus* (0.0%)**	**Random (2.0%)**	Random (0.8%)	**S-Plus (12.9%)**	Random (28.4%)	S-Plus (17.6%)
	Intersection (5)	**S-Plus* (1.0%)**	**S-Plus* (0.0%)**	**S-Plus* (2.0%)**	**S-Plus (1.3%)**	Score (9.5%)	Random (28.4%)	**R-Plus (35.8%)**

Experimental results (Improved percentage)

Benchmark Algorithm	MUS data set (best count)	Best ranking function (improvement in %)						
		Number of Solved Problems			Average Time			
		Total	SAT	UNSAT	All	Common	SAT	UNSAT
Non-Linear CEGAR	Union (7)	**S-Plus (0.5%)**	**S-Plus (0.8%)**	**S-Plus* (0.0%)**	**S-Plus (7.1%)**	**R-Minus (20.8%)**	**Rank (53.5%)**	**S-Plus (19.4%)**
	Single (1)	R-Plus (0.2%)	R-Plus (0.4%)	**R-Plus* (0.0%)**	R-Plus (6.6%)	S-Plus (18.4%)	R-Minus (52.8%)	R-Minus (14.2%)
	Intersection (1)	R-Plus* (0.0%)	S-Plus (0.5%)	**S-Plus* (0.0%)**	R-Plus (5.9%)	R-Plus (20.3%)	Rank (45.8%)	S-Plus (16.8%)
Non-Linear SymEx	Union (6)	**Two-Q (6.1%)**	**S-Minus* (1.6%)**	Random (12.3%)	**Two-Q (13.3%)**	**R-Minus (7.3%)**	**Score (5.1%)**	**R-Plus (19.9%)**
	Single (3)	**Two-Q (6.1%)**	**Score (1.6%)**	**Two-Q (12.9%)**	Two-Q (12.4%)	Rank (-2.2%)	R-Minus (0.2%)	Two-Q (11.2%)
	Intersection (3)	**Two-Q (6.1%)**	**S-Plus (1.6%)**	**Two-Q (12.9%)**	Two-Q (12.7%)	S-Minus (0.6%)	Two-Q (1.7%)	S-Plus (5.4%)

Visualize CHCs with dependency graph

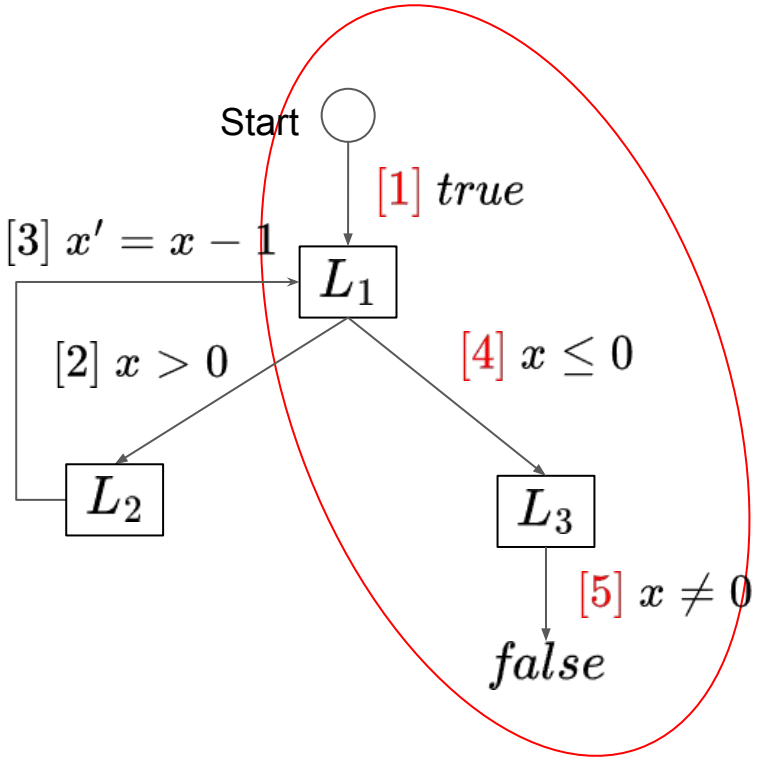
$$
\begin{array}{ll}
[1]\ L_1(x) & \leftarrow true \\
[2]\ L_2(x) & \leftarrow L_1(x) \wedge x > 0 \\
[3]\ L_1(x') & \leftarrow L_2(x) \wedge x' = x - 1 \\
[4]\ L_3(x) & \leftarrow L_1(x) \wedge x \leq 0 \\
[5]\ false & \leftarrow L_3(x) \wedge x \neq 0
\end{array}
$$



MUSes of CHCs

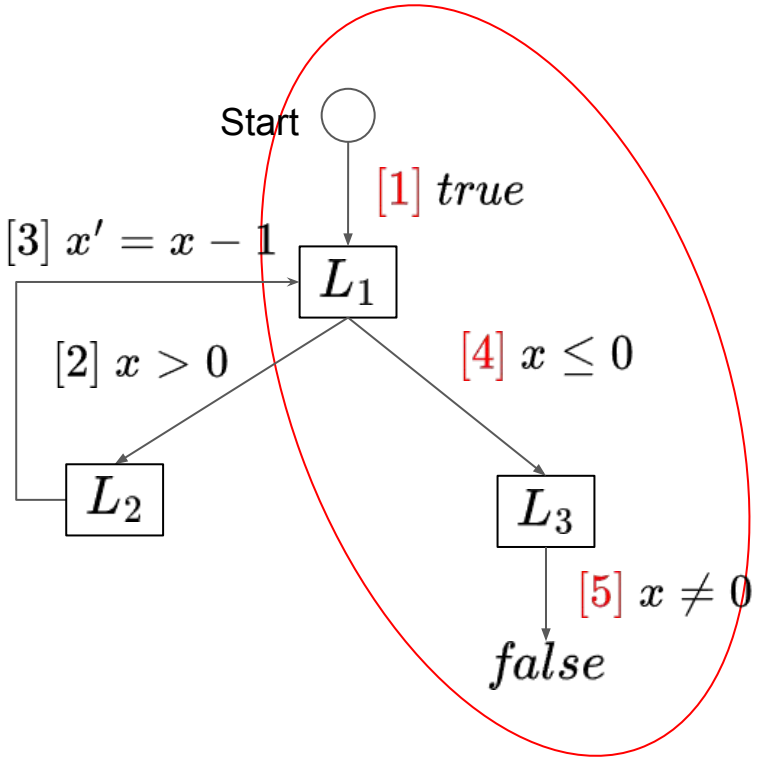
$$
\begin{array}{ll}
[1]\ L_1(x) & \leftarrow true \\
[2]\ L_2(x) & \leftarrow L_1(x) \wedge x > 0 \\
[3]\ L_1(x') & \leftarrow L_2(x) \wedge x' = x - 1 \\
[4]\ L_3(x) & \leftarrow L_1(x) \wedge x \leq 0 \\
[5]\ false & \leftarrow L_3(x) \wedge x \neq 0
\end{array}
$$

{[1], [4], [5]} is the only MUSes



MUSes of CHCs

$$
\begin{array}{lll}
[1]\ L_1(x) & \leftarrow & true \\
[2]\ L_2(x) & \leftarrow & L_1(x) \wedge x > 0 \\
[3]\ L_1(x') & \leftarrow & L_2(x) \wedge x' = x - 1 \\
[4]\ L_3(x) & \leftarrow & L_1(x) \wedge x \leq 0 \\
[5]\ false & \leftarrow & L_3(x) \wedge x \neq 0
\end{array}
$$



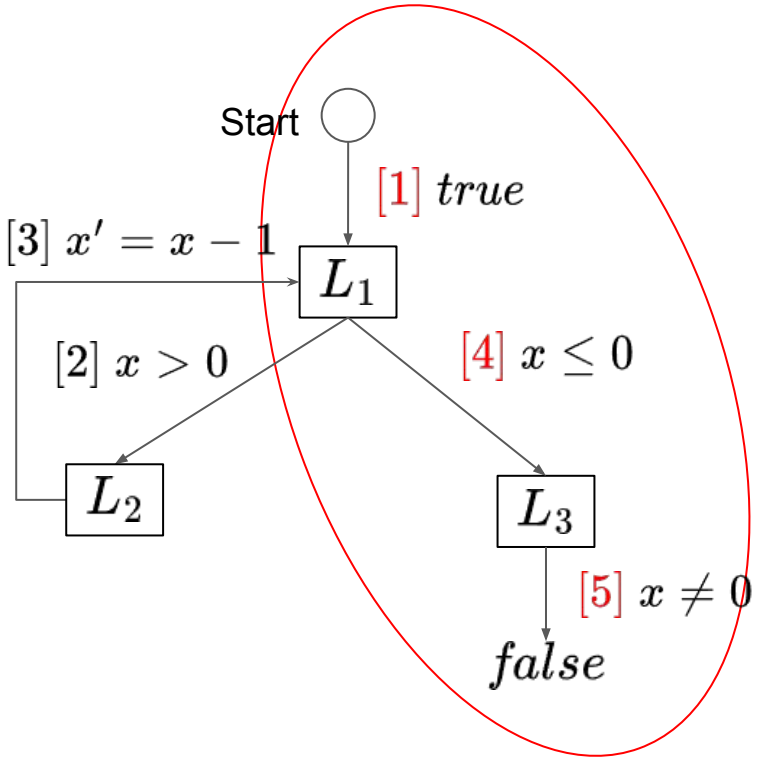
- Algorithms
  - Counterexample-guided abstraction refinement (CEGAR)
  - Symbolic execution (Symex)

MUSes of CHCs

Score

0.8	$[1]\ L_1(x)$	$\leftarrow true$
0.2	$[2]\ L_2(x)$	$\leftarrow L_1(x) \wedge x > 0$
0.1	$[3]\ L_1(x')$	$\leftarrow L_2(x) \wedge x' = x - 1$
0.75	$[4]\ L_3(x)$	$\leftarrow L_1(x) \wedge x \leq 0$
0.6	$[5]\ false$	$\leftarrow L_3(x) \wedge x \neq 0$

- Algorithms
  - Counterexample-guided abstraction refinement (CEGAR)
  - Symbex execution (Symex)

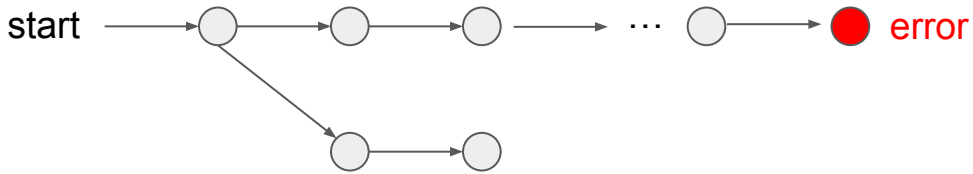


Framework overview

- Extract program features by graph neural networks from CHC's graph representation

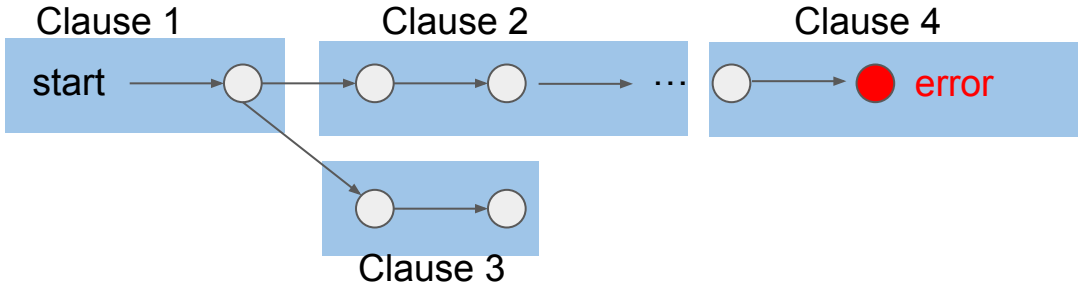
Five proxy tasks from simple to difficult

1. Predict if an graph node is an argument of relation symbol
2. Predict the number of occurrence of the relation symbols in all clauses
3. Predict if a relation symbol is in a cycle
4. Predict the existence of argument bound
5. Predict the clause membership in all/some minimal unsat cores



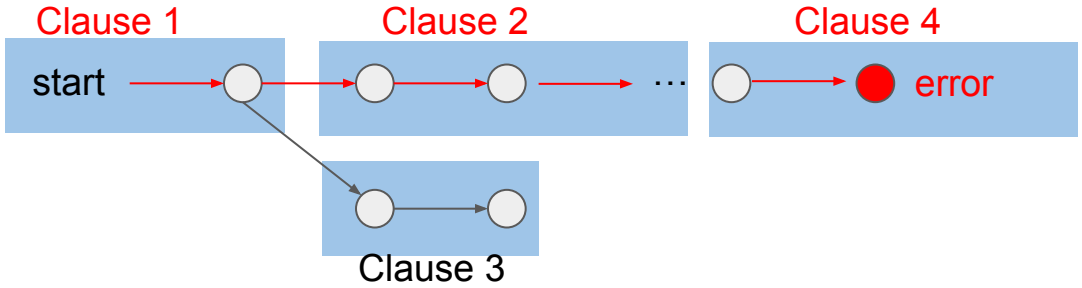
Five proxy tasks from simple to difficult

1. Predict if an graph node is an argument of relation symbol
2. Predict the number of occurrence of the relation symbols in all clauses
3. Predict if a relation symbol is in a cycle
4. Predict the existence of argument bound
5. Predict the clause membership in all/some minimal unsat cores



Five proxy tasks from simple to difficult

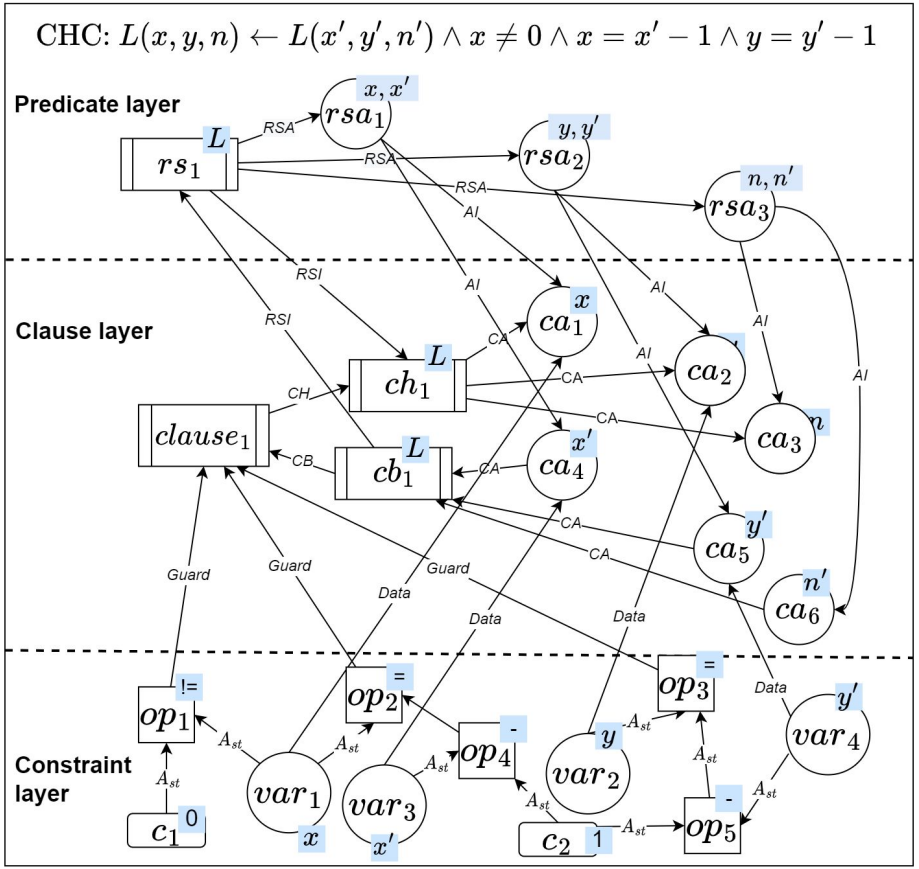
1. Predict if an graph node is an argument of relation symbol
2. Predict the number of occurrence of the relation symbols in all clauses
3. Predict if a relation symbol is in a cycle
4. Predict the existence of argument bound
5. Predict the clause membership in all/some minimal unsat cores



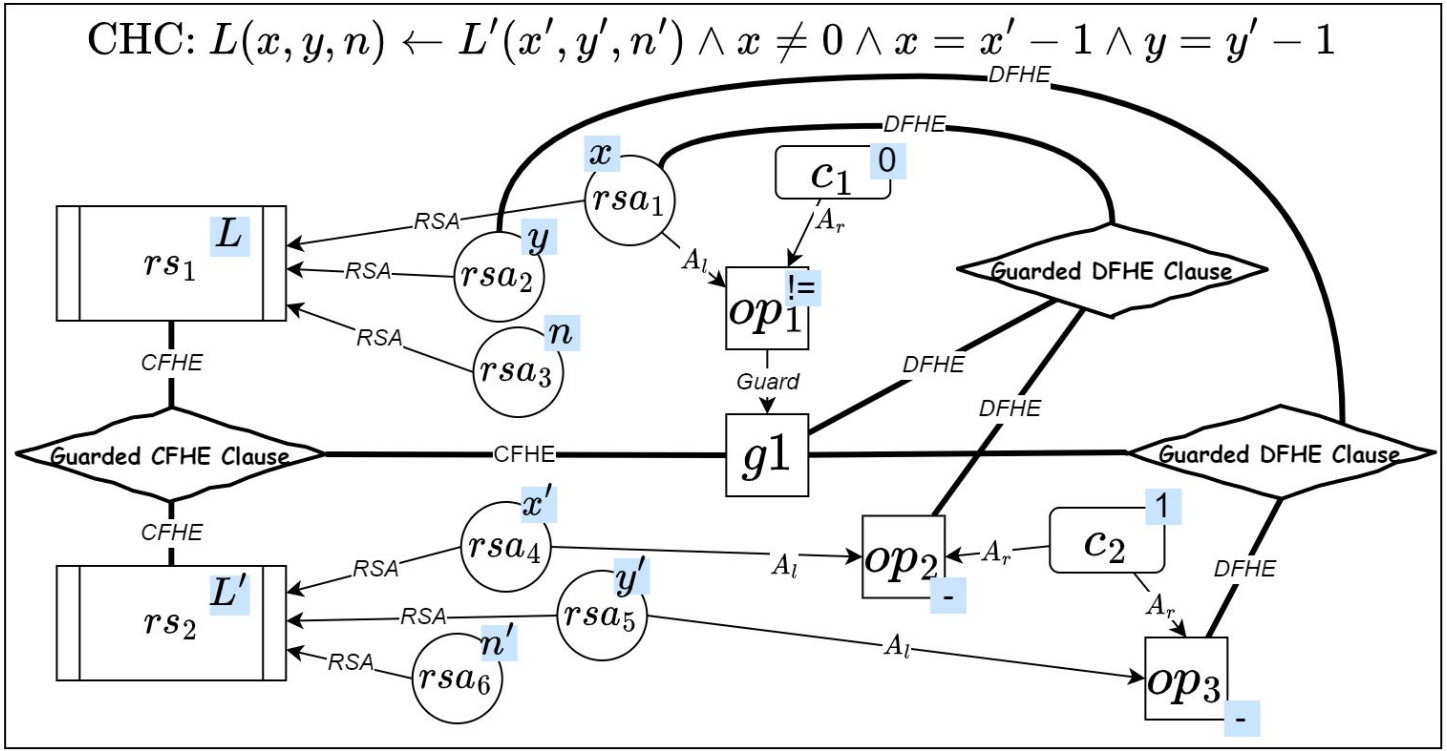
Framework overview

- Two graph representations for CHCs

Constraint graph (CG)

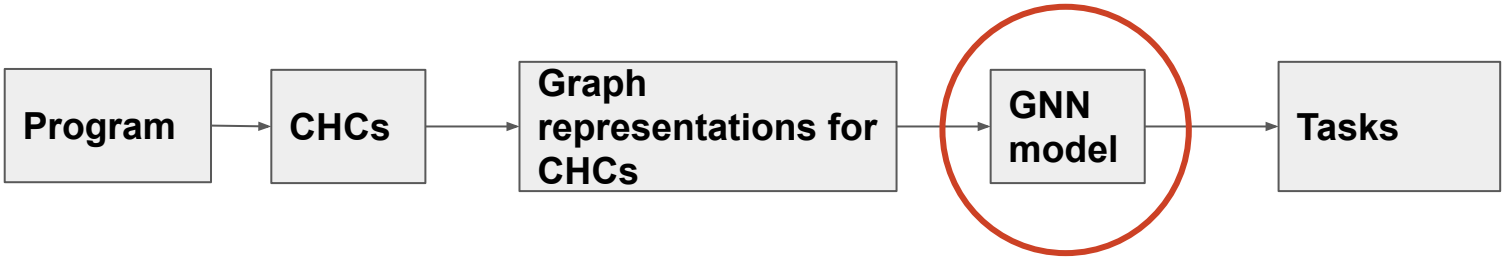


Control- and data-flow hypergraph (CDHG)

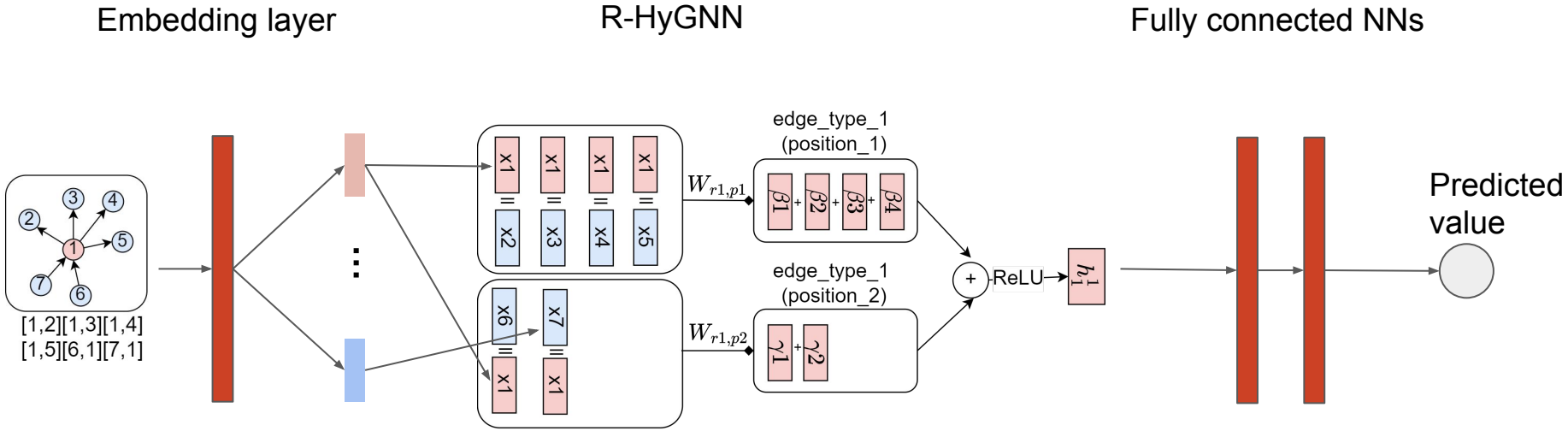


Framework overview

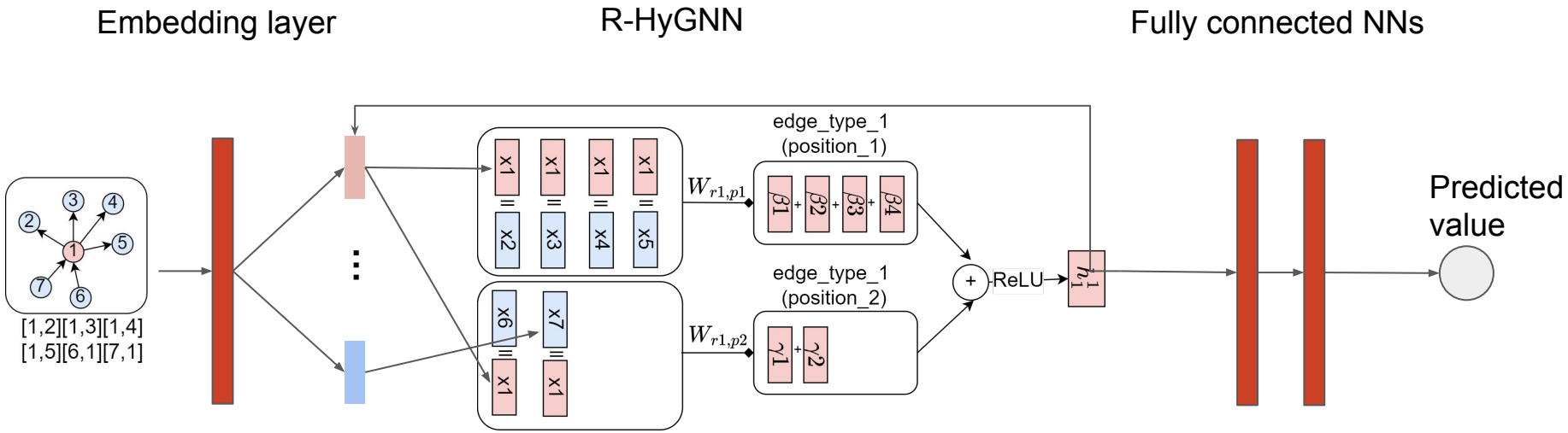
- Relational Hypergraph Neural Network (R-HyGNN)



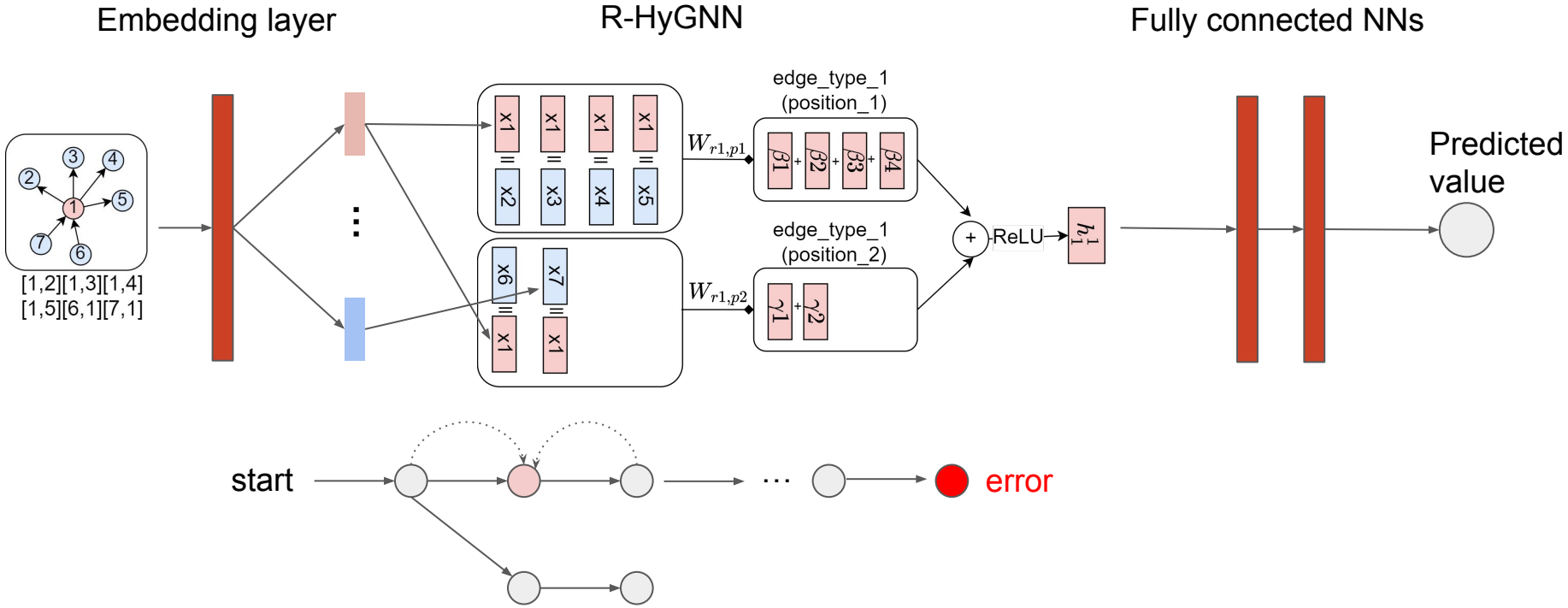
Training model



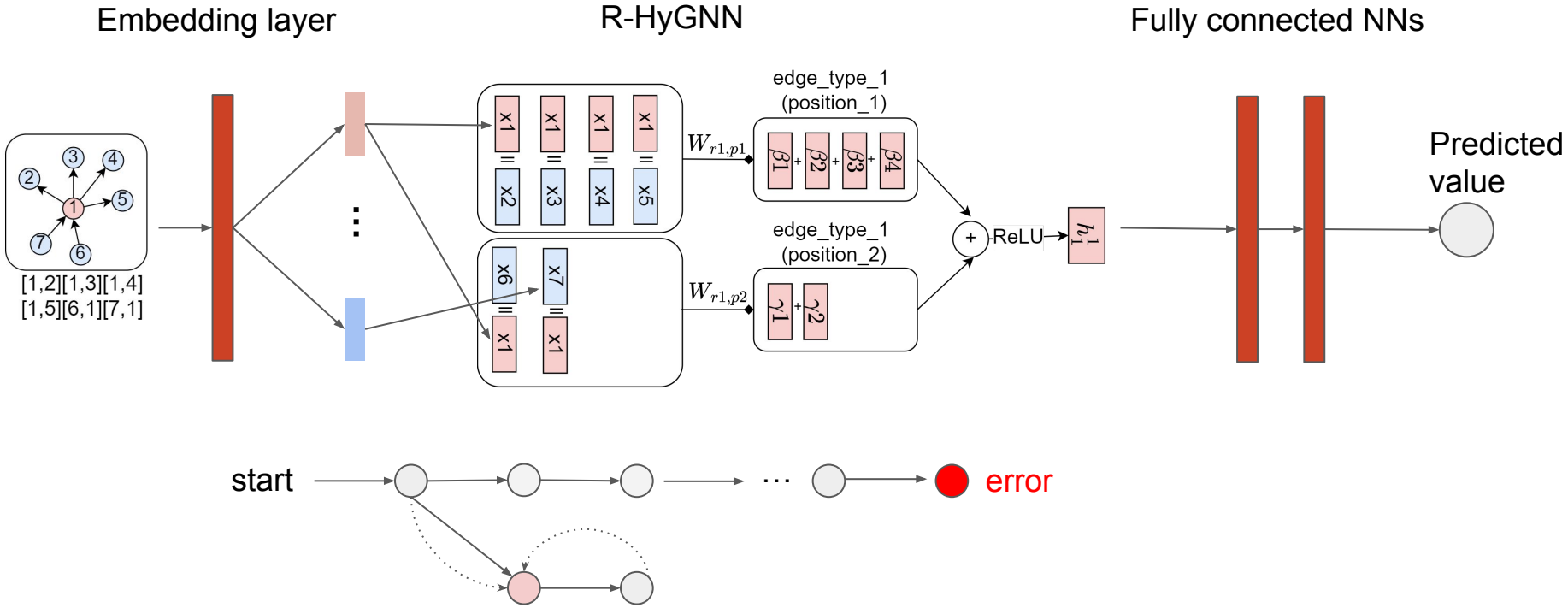
Training model



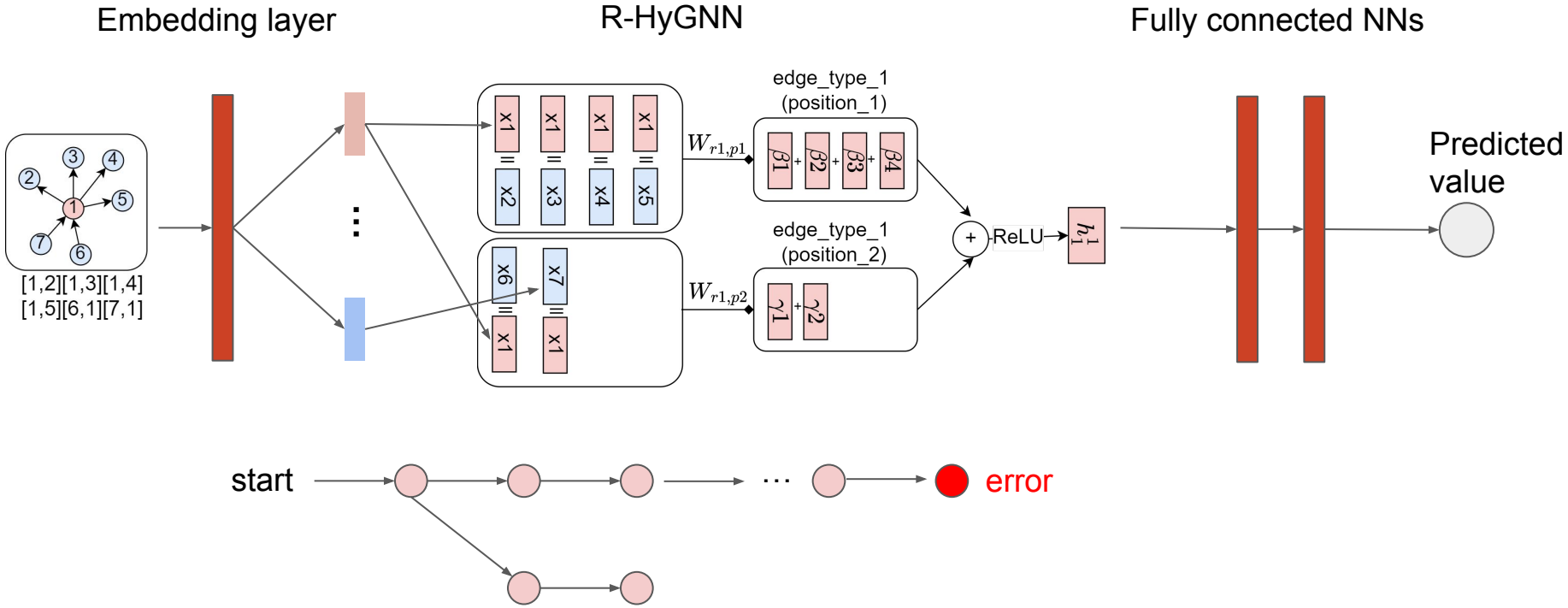
Training model



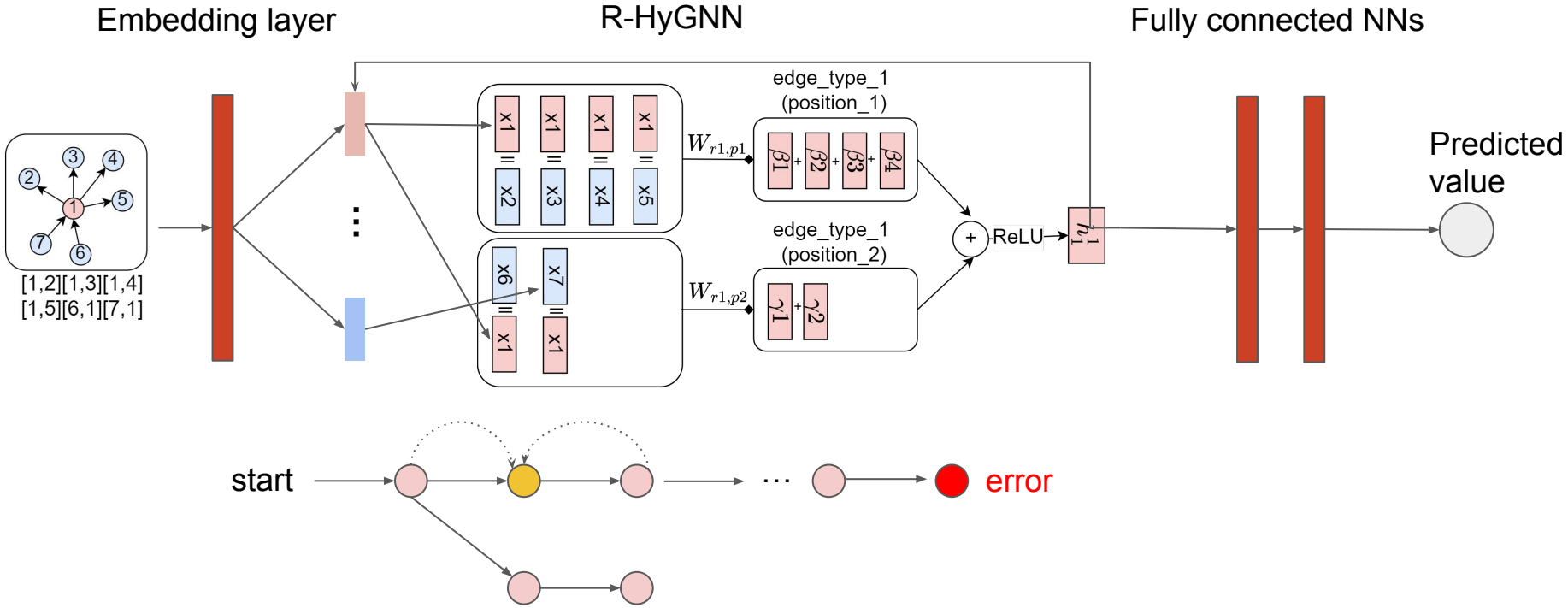
Training model



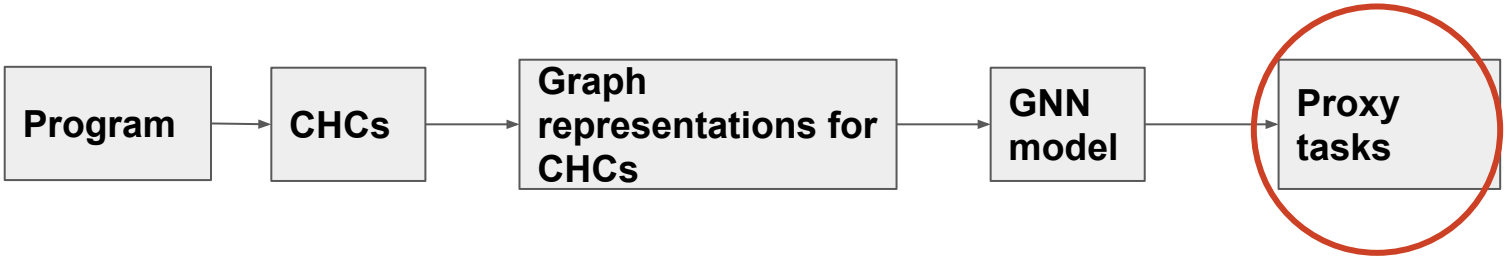
Training model



Training model



Framework overview



Evaluation on five proxy tasks

- Dominate distribution

			CG				CDHG			
Task	Files	P / T	+	-	Acc.	Dom.	+	-	Acc.	Dom.
1	1121	+	93863	0	100%	95.1%	142598	0	99.9%	72.8%
		-	0	1835971			10	381445		
3	1115	+	3204	133	96.1%	70.1%	8262	58	99.6%	50.7%
		-	301	7493			15	8523		
4 (a)	1028	+	13685	5264	91.2%	79.7%	30845	4557	94.3%	75.2%
		-	2928	71986			3566	103630		
4 (b)		+	18888	4792	91.4%	74.8%	41539	4360	94.3%	67.8%
		-	3291	66892			3715	92984		
5 (a)	386	+	1048	281	95.0%	84.7%	1230	206	96.9%	86.4%
		-	154	7163			121	9036		
5 (b)	383	+	3030	558	84.6%	53.1%	3383	481	90.6%	54.8%
		-	622	3428			323	4361		

Evaluation on five proxy tasks

- Accuracy is higher than dominate distribution

			CG				CDHG			
Task	Files	P / T	+	-	Acc.	Dom.	+	-	Acc.	Dom.
1	1121	+	93863	0	100%	95.1%	142598	0	99.9%	72.8%
		-	0	1835971			10	381445		
3	1115	+	3204	133	96.1%	70.1%	8262	58	99.6%	50.7%
		-	301	7493			15	8523		
4 (a)	1028	+	13685	5264	91.2%	79.7%	30845	4557	94.3%	75.2%
		-	2928	71986			3566	103630		
4 (b)		+	18888	4792	91.4%	74.8%	41539	4360	94.3%	67.8%
		-	3291	66892			3715	92984		
5 (a)	386	+	1048	281	95.0%	84.7%	1230	206	96.9%	86.4%
		-	154	7163			121	9036		
5 (b)	383	+	3030	558	84.6%	53.1%	3383	481	90.6%	54.8%
		-	622	3428			323	4361		

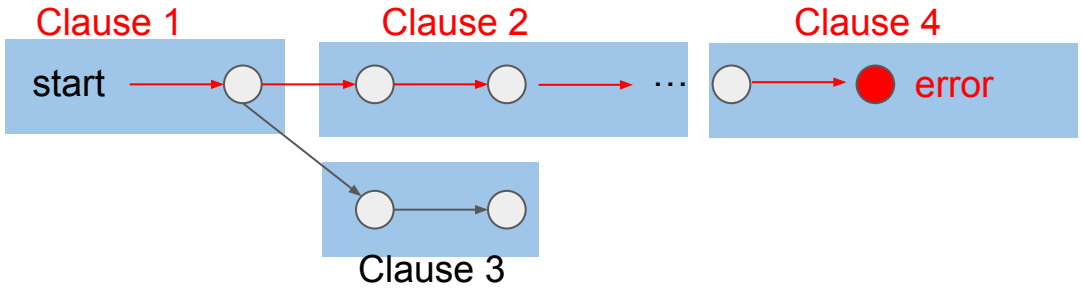
Evaluation on five proxy tasks (Task 5 results)

- Task 5: Predict the clause membership in (a) some and (b) all of the minimal unsat cores
- Task 5 (a) 1155, 386, 386 problems for train, valid, and test respectively

			CG				CDHG			
Task	Files	P / T	+	-	Acc.	Dom.	+	-	Acc.	Dom.
1	1121	+	93863	0	100%	95.1%	142598	0	99.9%	72.8%
		-	0	1835971			10	381445		
3	1115	+	3204	133	96.1%	70.1%	8262	58	99.6%	50.7%
		-	301	7493			15	8523		
4 (a)	1028	+	13685	5264	91.2%	79.7%	30845	4557	94.3%	75.2%
		-	2928	71986			3566	103630		
4 (b)		+	18888	4792	91.4%	74.8%	41539	4360	94.3%	67.8%
		-	3291	66892			3715	92984		
5 (a)	386	+	1048	281	95.0%	84.7%	1230	206	96.9%	86.4%
		-	154	7163			121	9036		
5 (b)	383	+	3030	558	84.6%	53.1%	3383	481	90.6%	54.8%
		-	622	3428			323	4361		

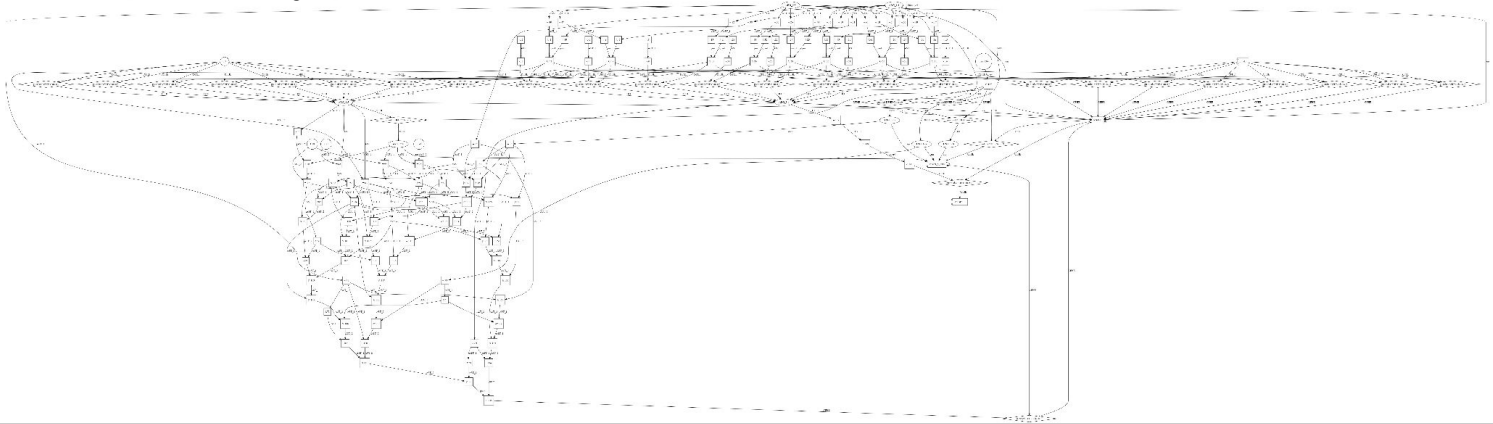
Predict the clause membership in minimal unsat cores

- Simple patterns
  - Clauses close to the assertions are likely in the minimal unsat cores
- Intricate patterns
  - Perfectly predict the clause membership of minimal unsat cores in the case that contain 290 clauses.

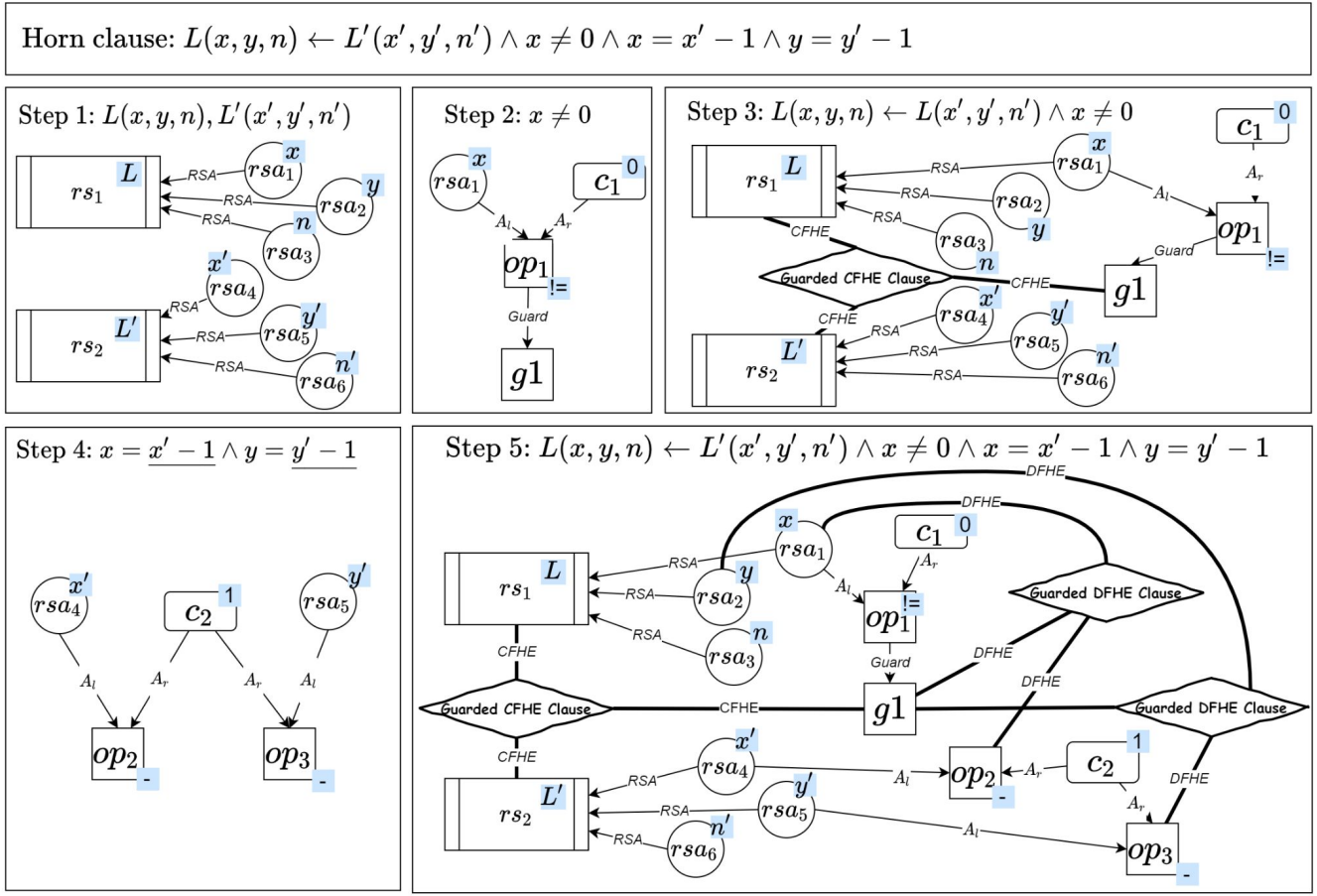


Predict the minimal unsatisfiable cores

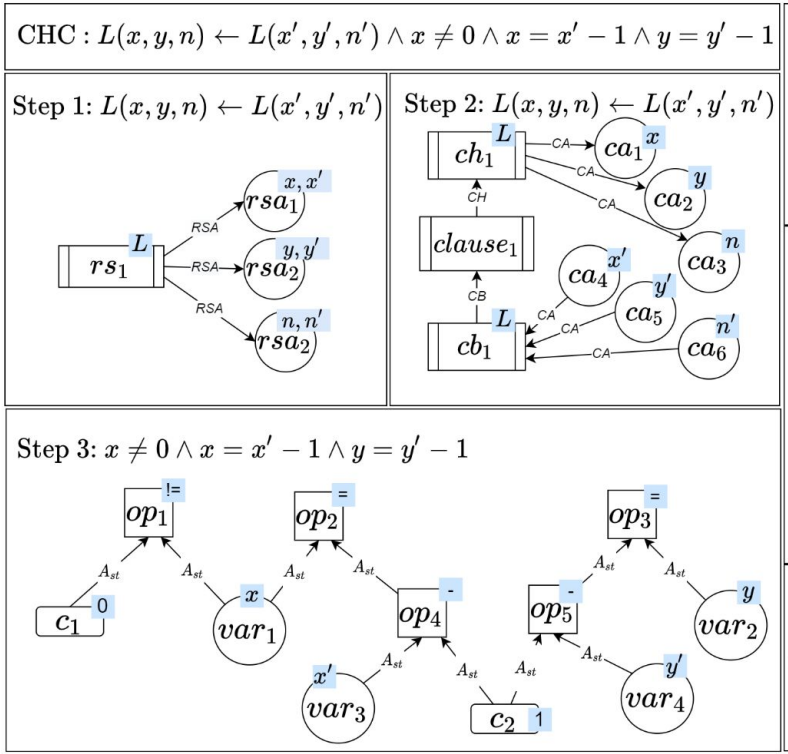
- Simple patterns
  - Clauses close to the assertions are likely in the minimal unsatisfiable cores
- Intricate patterns
  - In the verification problem that contain 290 clauses, the model can perfectly predict the clause membership in minimal unsat cores.



Control- and data-flow hypergraph (CDHG)

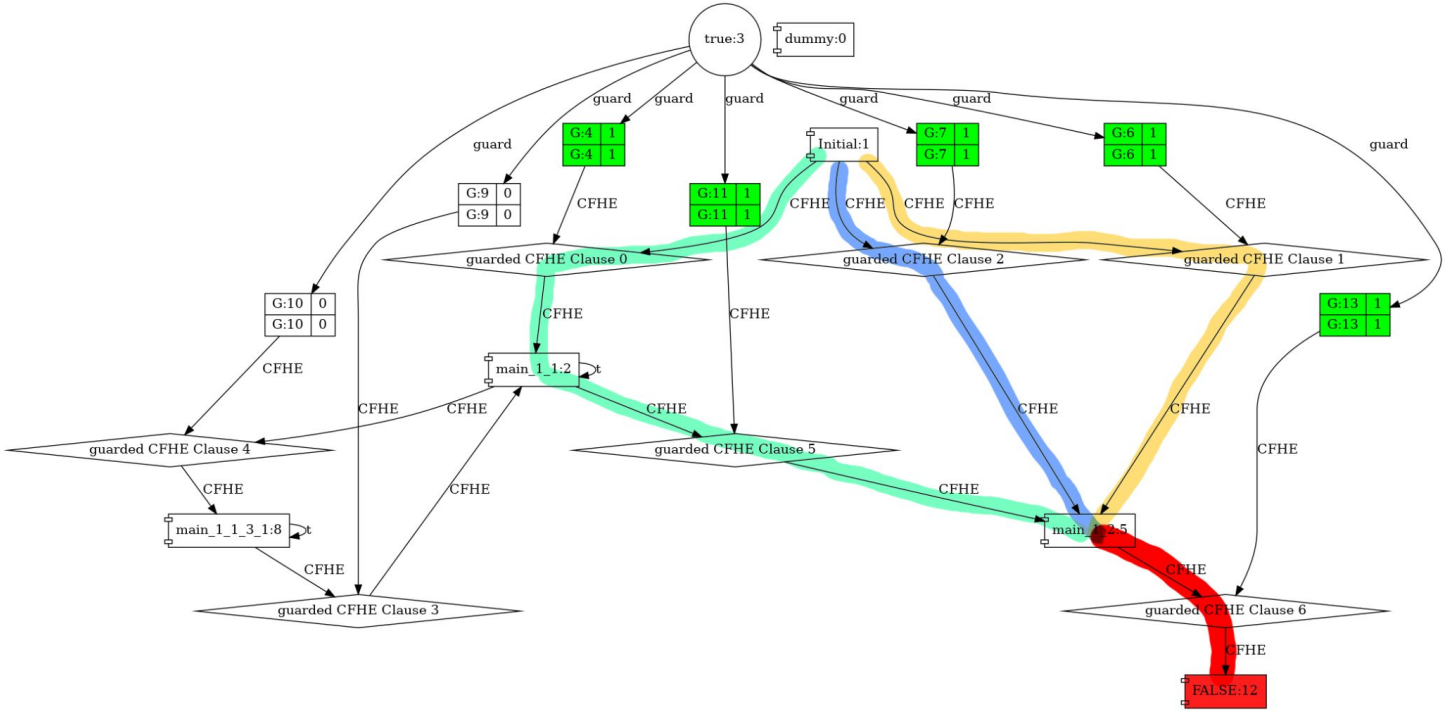


Constraint graph (CG)

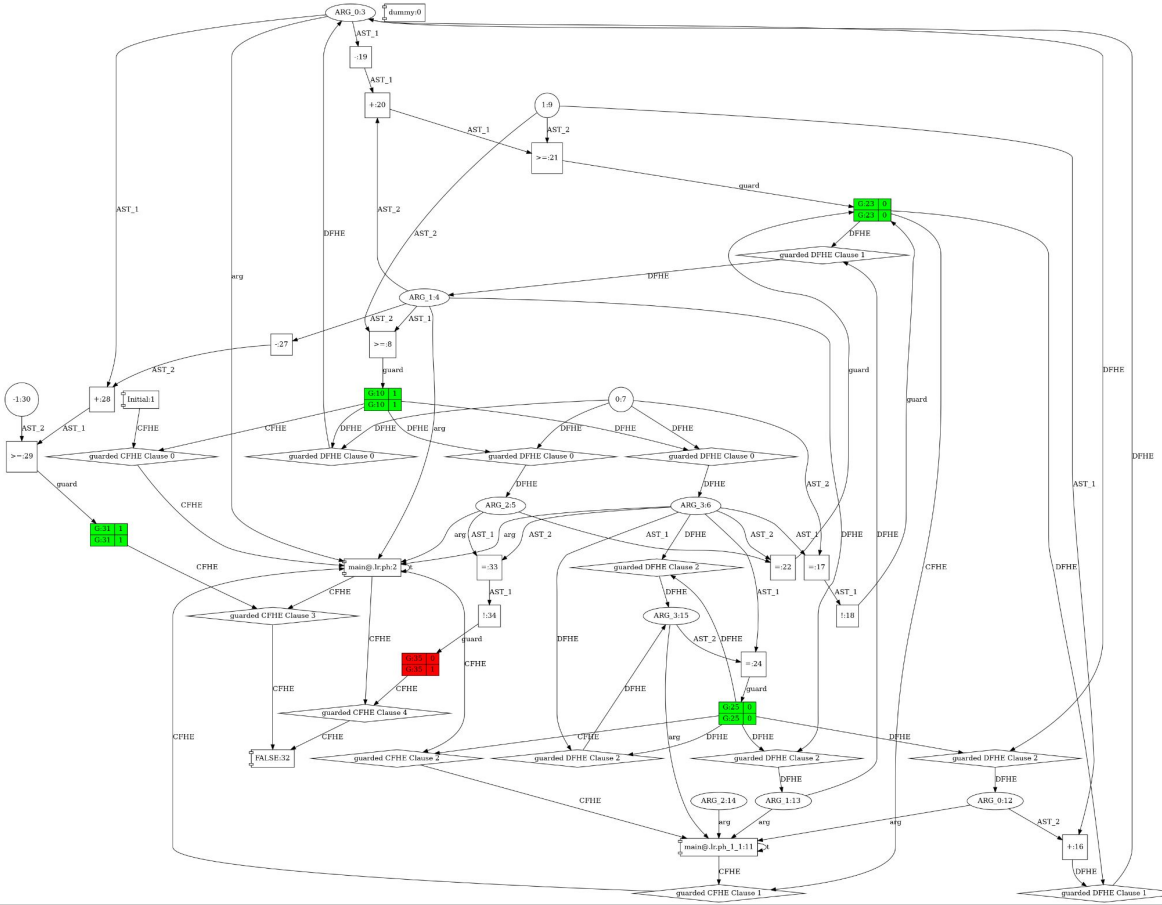


Predict the minimum unsatisfiable cores

- Three minimum unsatisfiable cores



Technique details

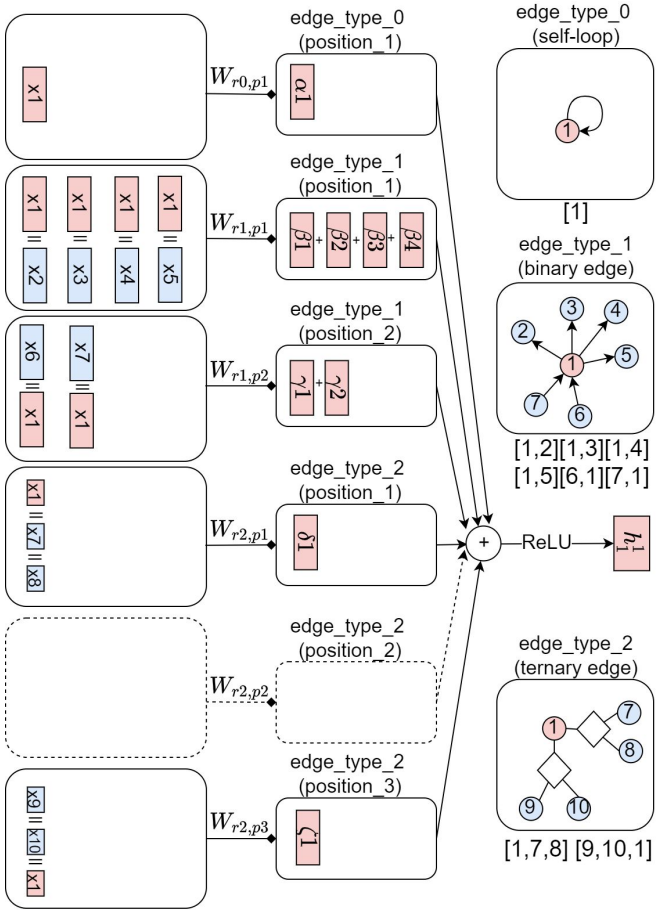


R-HyGNN

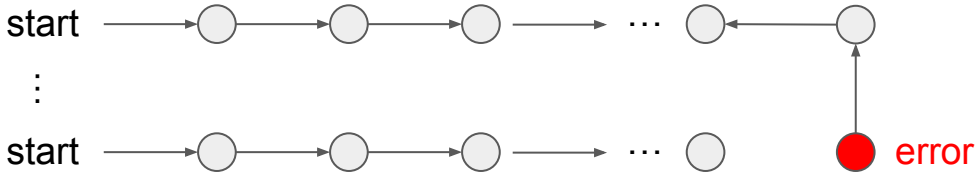
The updating rule for node representation in time step t :

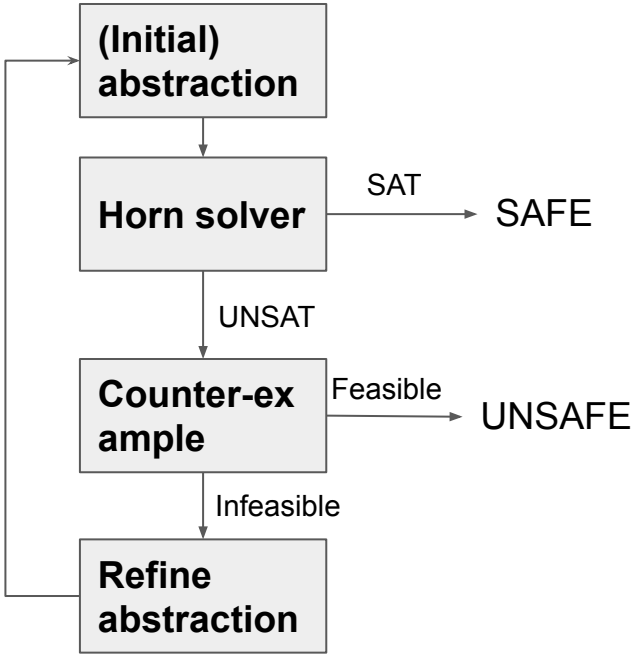
$$h_v^t = \mathrm{ReLU}(\sum_{r \in R} \sum_{p \in P_r} \sum_{e \in E_v^{r,p}} W_{r,p}^t \cdot \|[h_u^{t-1} \mid u \in e]),$$

where$\|\{\cdot\}$ denotes concatenation of all elements in a set, $r \in R = \{r_i \mid i \in \mathbb{N}\}$ is the set of edge types (relations), $p \in P_r = \{p_j \mid j \in \mathbb{N}\}$ is the set of node positions under edge type $r, W_{r,p}^t$ denotes learnable parameters when the node is in the$p$th position with edge type r, and $e \in E_v^{r,p}$ is the set of hyperedges of type r in the graph in which node v appears in position p, where$e$is a list of nodes.

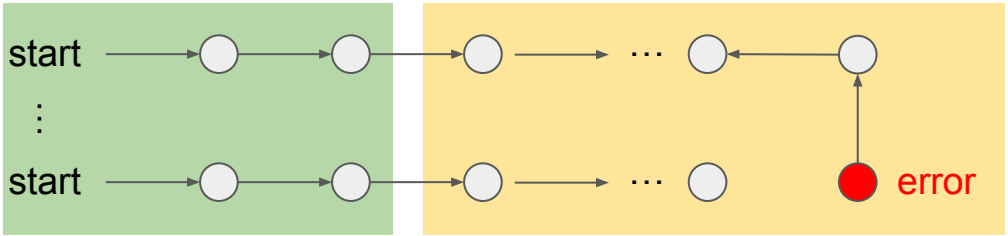


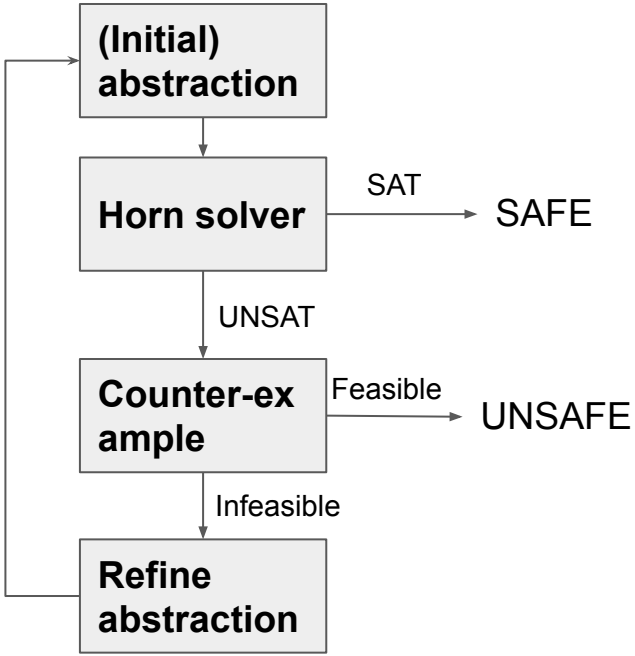
Solving CHCs by counter-example guided abstraction refinement (CEGAR) based model checking



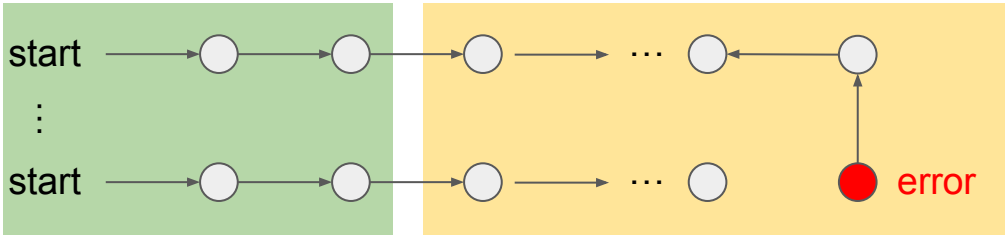


Solving CHCs by counter-example guided abstraction refinement (CEGAR) based model checking



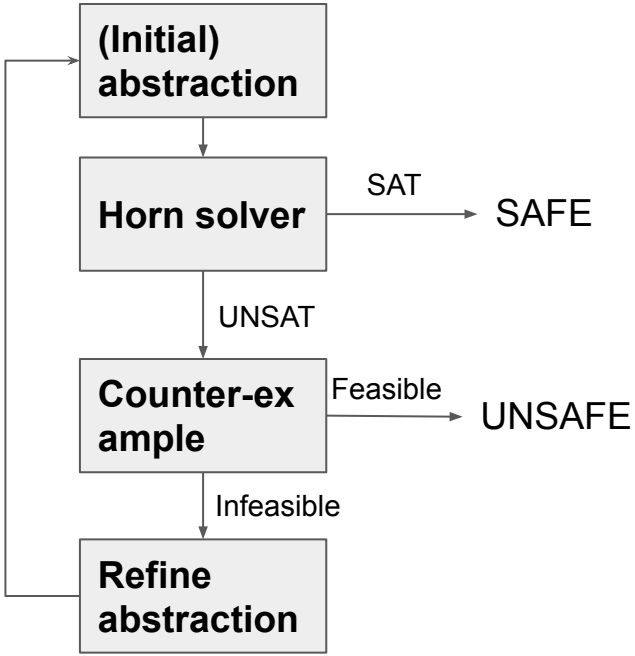


Solving CHCs by counter-example guided abstraction refinement (CEGAR) based model checking

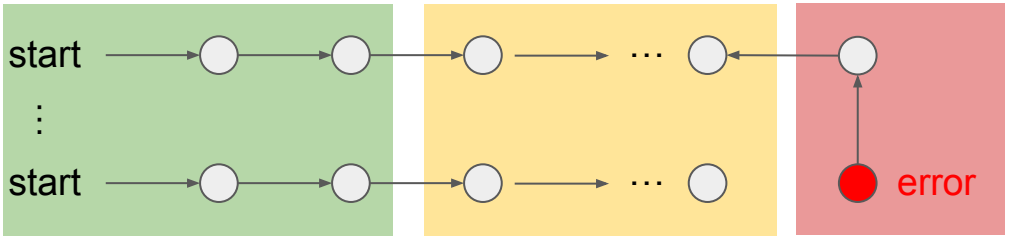


Abstraction

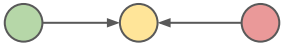
UNSAT with a infeasible counter-example



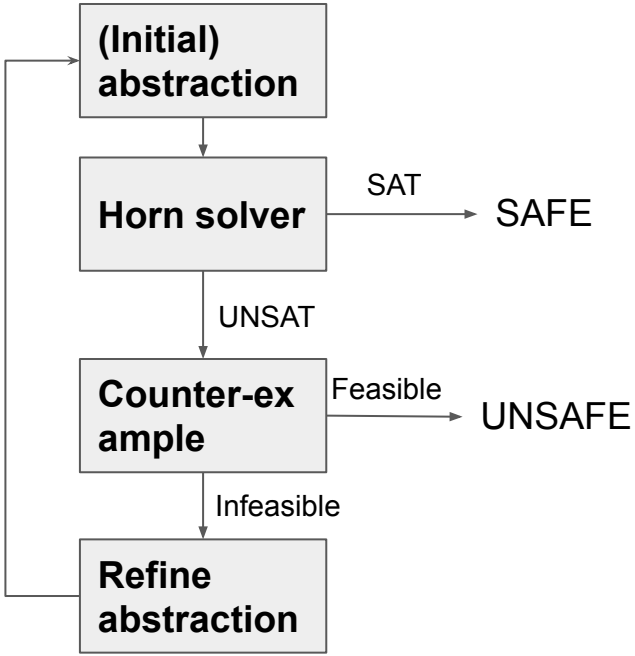
Solving CHCs by counter-example guided abstraction refinement (CEGAR) based model checking



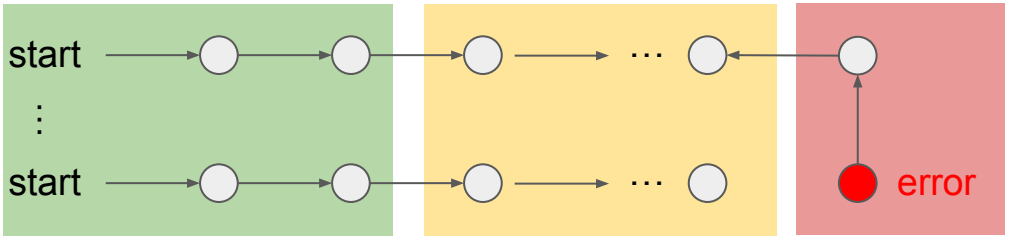
New abstraction



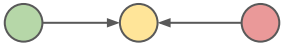
SAT with new refinement



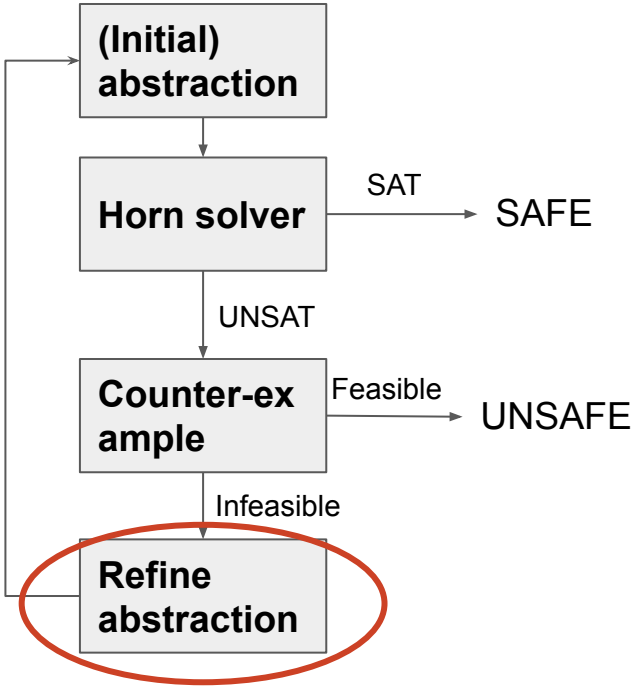
Solving CHCs by counter-example guided abstraction refinement (CEGAR) based model checking



New abstraction



SAT with new refinement



Motivating Examples (Theorem Proving)

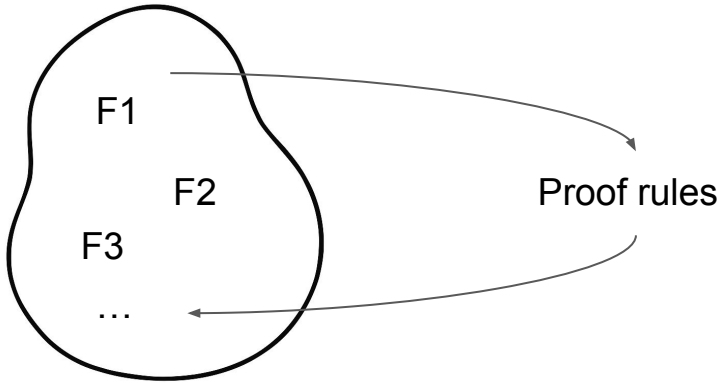
$$\text{Prove} : \forall x.\, P(x) \to Q(x) \land \forall x.\, Q(x) \to R(x) \land P(a) \implies R(a)$$

F1: $\forall x.\, P(x) \to Q(x)$

F2: $\forall x.\, Q(x) \to R(x)$

F3: $P(a)$

Goal: $R(a)$



Motivating Examples (Theorem Proving)

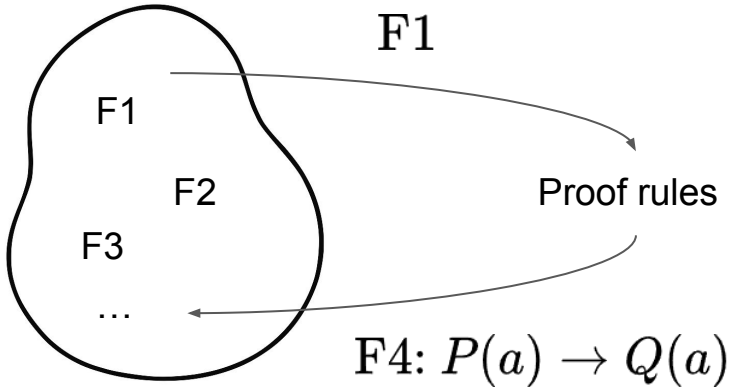
$$\text{Prove} : \forall x.\, P(x) \to Q(x) \land \forall x.\, Q(x) \to R(x) \land P(a) \implies R(a)$$

F1: $\forall x.\, P(x) \to Q(x)$

F2: $\forall x.\, Q(x) \to R(x)$

F3: $P(a)$

Goal: $R(a)$



Motivating Examples (Theorem Proving)

$$\text{Prove} : \forall x.\, P(x) \to Q(x) \land \forall x.\, Q(x) \to R(x) \land P(a) \implies R(a)$$

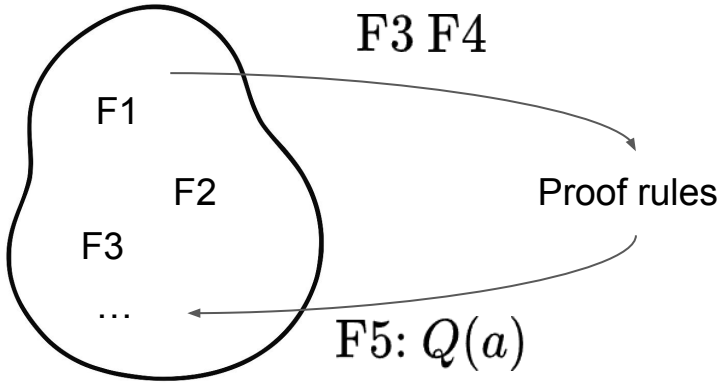
F1: $\forall x.\, P(x) \to Q(x)$

F2: $\forall x.\, Q(x) \to R(x)$

F3: $P(a)$

F4: $P(a) \to Q(a)$

Goal: $R(a)$



Guide CHC Solver (Example)

- Rank clauses before solving

$$
\begin{array}{lll}
C1: & L_1(x) & \leftarrow true \\
C2: & L_2(x) & \leftarrow L_1(x) \wedge x > 0 \\
C3: & L_1(x') & \leftarrow L_2(x) \wedge x' = x - 1 \\
C4: & L_3(x) & \leftarrow L_1(x) \wedge x \leq 0 \\
C5: & false & \leftarrow L_3(x) \wedge x = 0
\end{array}
$$

[6] H. Hojjat and P. Ruemmer, “The ELDARICA Horn solver,” 2018.

Guide CHC Solver (Example)

- Rank clauses before solving

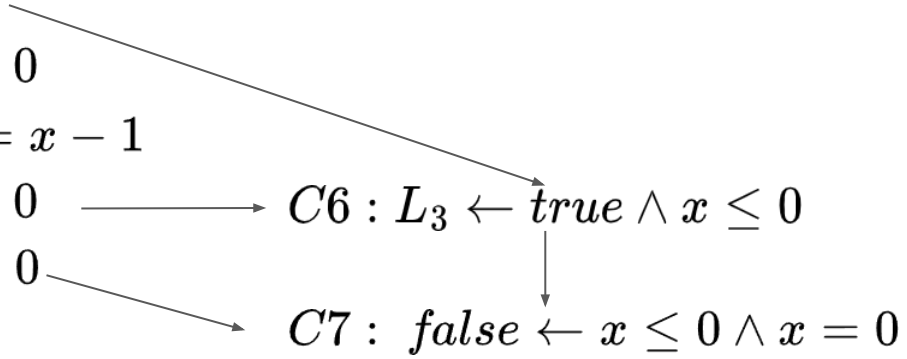
$$
\begin{array}{lll}
C1: & L_1(x) & \leftarrow true \\
C2: & L_2(x) & \leftarrow L_1(x) \wedge x > 0 \\
C3: & L_1(x') & \leftarrow L_2(x) \wedge x' = x - 1 \\
C4: & L_3(x) & \leftarrow L_1(x) \wedge x \leq 0 \\
C5: & false & \leftarrow L_3(x) \wedge x = 0
\end{array}
\qquad \longrightarrow \qquad C6 : L_3 \leftarrow true \wedge x \leq 0
$$

[6] H. Hojjat and P. Ruemmer, “The ELDARICA Horn solver,” 2018.

Guide CHC Solver (Example)

- Rank clauses before solving

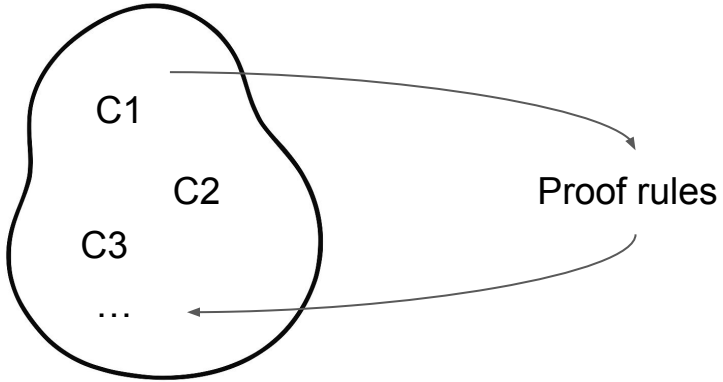
$$
\begin{aligned}
C1:\ & L_1(x) && \leftarrow true \\
C2:\ & L_2(x) && \leftarrow L_1(x) \wedge x > 0 \\
C3:\ & L_1(x') && \leftarrow L_2(x) \wedge x' = x - 1 \\
C4:\ & L_3(x) && \leftarrow L_1(x) \wedge x \leq 0 \\
C5:\ & false && \leftarrow L_3(x) \wedge x = 0
\end{aligned}
$$



[6] H. Hojjat and P. Ruemmer, “The ELDARICA Horn solver,” 2018.

Guide CHC Solver (Example)

- Rank clauses before solving



$$
\begin{aligned}
C1:\ & L_1(x) && \leftarrow true \\
C2:\ & L_2(x) && \leftarrow L_1(x) \wedge x > 0 \\
C3:\ & L_1(x') && \leftarrow L_2(x) \wedge x' = x - 1 \\
C4:\ & L_3(x) && \leftarrow L_1(x) \wedge x \leq 0 \\
C5:\ & false && \leftarrow L_3(x) \wedge x = 0
\end{aligned}
$$

$$C6: L_3 \leftarrow true \wedge x \leq 0$$

$$C7:\ false \leftarrow x \leq 0 \wedge x = 0$$

[6] H. Hojjat and P. Ruemmer, “The ELDARICA Horn solver,” 2018.