PREMISE SELECTION FOR HIGHER-ORDER ATP PROBLEMS

Keneni W. Tesema, Jan Jakubuv, Martin Suda

Czech Technical University in Prague

Deeper4AI Project

Gothenburg CHAIR workshop, 28th May 2026

INTRO: PREMISE SELECTION

- Premise selection is the task of identifying, from a library of potentially thousands of axioms, a small subset necessary to prove a given conjecture.

- ATP without premise selection

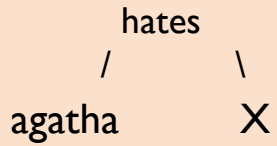
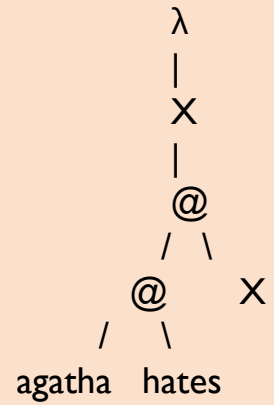
- ATP with premise selection

- Magic

INTRO: HIGER-ORDER LOGIC (HOL) PROBLEMS

- **First-order:** "Be confident."
- **Higher-order:** "Be confident in your ability to become more confident.“
- **First-order:** "Translate 'hello' to 'bonjour'."
- **Higher-order:** "Translate any instruction from English to French, including the word 'translate' itself."
- **First-order:** “Do what Simon says"
- **Higher-order:** “Do what Simon says when Simon says “Simon says….””

PREMISE SELECTION IN HOL

FOL	HOL
Variables range over **individuals** (people, numbers, etc.)	**Variables** can range over **FUNCTIONS(properties etc)**: ∀P. P(agatha) → P(butler)
Predicates apply to individuals: hates (agatha, X)	**Lambda abstraction**: λx. x + x (functions as data)
Functions return individuals: father_of (agatha)	**Partial application**: hates(agatha) returns a function λY. hates(agatha, Y)
Quantifiers only over individuals: ∀X. hates(agatha, X)	**Higher-order unification**: Matching P(agatha) against hates(agatha, butler) requires finding P = λX. hates(X, butler)
Simple Abstract Syntax Trees(AST) Eg: Ax: hates(agatha, X) 	Richer AST Eg: ^[X:$i]: (@(@(hates, agatha), X)) 

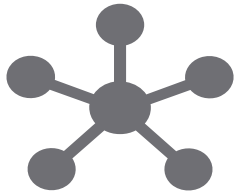
CHALLENGES IN PREMISE SELECTION IN HOL

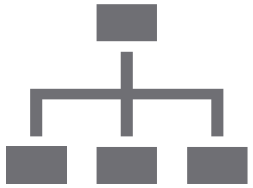
FOL	HOL
Challenge 1: Variable Arity and Partial Application	
hates(agatha, butler)	hates(agatha), (hates(agatha))(butler), @(@(hates, agatha), butler)
Challenge 2: **Lambda Abstractions (Anonymous functions)**	
None	λx. x + x
Challenge 3: **Rich Type system**	
Simple types (i, o)	Function types, Polymorphic types, Type variables and instantiation
Challenge 4: **Many Node Types**	
Symbol, variable, operator	Application (@), Lambda(^) , Type abstraction, Type application, implicit vs explicit quantifications

RELATED WORKS

Technique	Approach	FOL	**HOL**	Type	Key Limitation
MePo (Meng & Paulson, 2009)	Symbol overlap heuristics	✓	✗	Heuristic	Shallow, no structure
MaSh (Kühlwein et al., 2013)	Naive Bayes + kNN ensemble	✓	✗	Heuristic	Hand-crafted features only
SInE (Hoder & Voronkov, 2011)	Least common symbol trigger (used in Vampire/E-prover)	✓	✗	Heuristic	Approximate, no types
Wang et al. (2017)	Graph embeddings with shared symbol nodes	✓	✓	GNN	First graph-based for HOL
Olšák et al. (2019)	Graph embeddings + property-invariant hypergraph	✓	✗	GNN	Invariant to symbol renaming; focuses on unseen Skolem symbols
ARCG-NN (Liu et al., 2021)	Cross-graph attention (conjecture ↔ axioms)	✓	✓	GNN	Better relevance encoding
Bansal et al. (2019)	GNN + sequence models (HOL Light)	✓	✓	Hybrid	HOL4/HOL Light specific
Magnushammer (Mikuła et al., 2024)	Contrastive transformer + InfoNCE (SELECT+RERANK)	✓	✓	Sequence	No explicit type structure, (types as ato ms, not composed)
Paliwal et al. (2020)	Graph w/ types + binding	✓	✓	GNN	Explicit HOL structure
QUILL (Kogkalidis et al., 2024)	Structured attention + de Bruijn indices	✓	✓	GNN	Dependent types (Agda), not THF

METHODOLOGY





Graph-based networks

Sequence-based networks

Problems get parsed by vampire → **Build Graph** → Initialization → **Encoding** → Scoring → Loss

Problem:
conj,
ax 1 to ax 2000

Problem:
conj,
ax 1 :4
Ax 2: 7
Ax3: -2
Ax20 : 9

EXAMPLE: KNIGHTS AND KNAVES

A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet two inhabitants: Zoey and Mel. Zoey tells you that Mel is a knave. Mel says, `Neither Zoey nor I are knaves'. Who is a knight and who is a knave?

Conjecture:
$\exists Y\ Z.\ ((Y = knight) <\sim> (Y = knave)) \wedge ((Z = knight) <\sim> (Z = knave)) \wedge is\_a(mel, Y) \wedge is\_a(zoey, Z)$

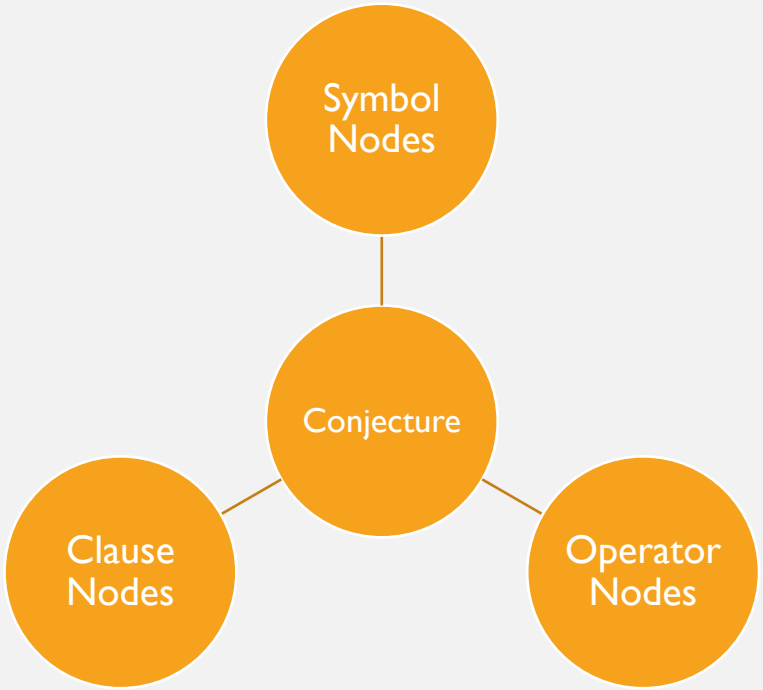
- **ax1:** $\forall X.\ is\_a(X, islander) \rightarrow (is\_a(X, knight) \vee is\_a(X, knave))$
  (Islanders are only knights or knaves)
- **ax2:** $\forall X.\ is\_a(X, knight) \rightarrow \forall A.\ (says(X, A) \rightarrow A)$
  (Knights tell the truth)
- **ax3:** $\forall X.\ is\_a(X, knave) \rightarrow \forall A.\ (says(X, A) \rightarrow \neg A)$
  (Knaves lie)
- **ax4:** $is\_a(zoey, islander) \wedge is\_a(mel, islander)$
  (Zoey and Mel are islanders)
- **ax5:** $says(zoey, is\_a(mel, knave))$
  (Zoey says Mel is a knave)
- **ax6:** $says(mel, \neg(is\_a(zoey, knave) \vee is\_a(mel, knave)))$
  (Mel says "Neither Zoey nor I are knaves")

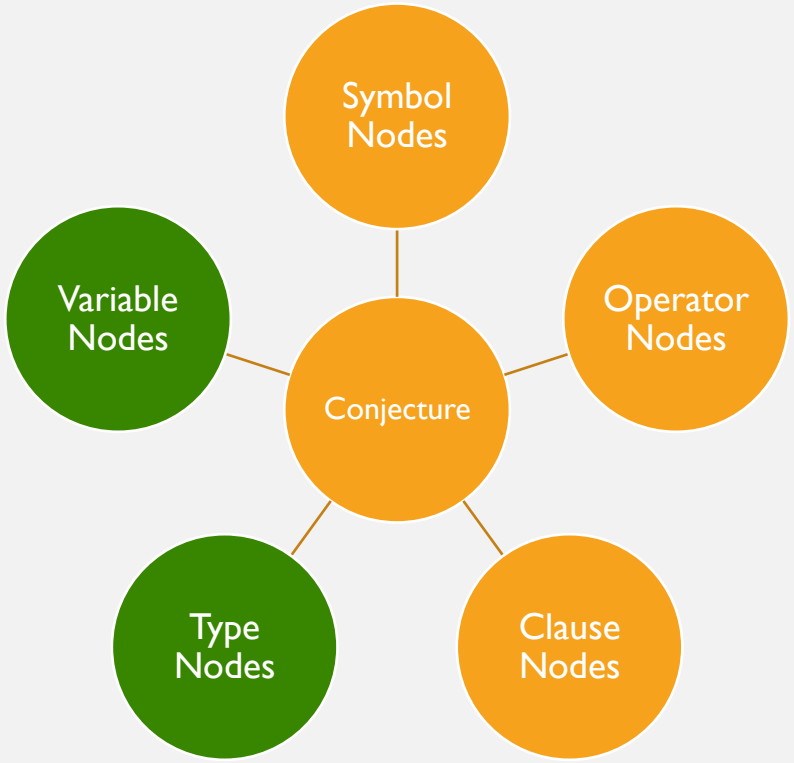
METHODOLOGY: GNN_V1

Problem (with too many axioms)

Graph based networks

Problem (with few relevant axioms)





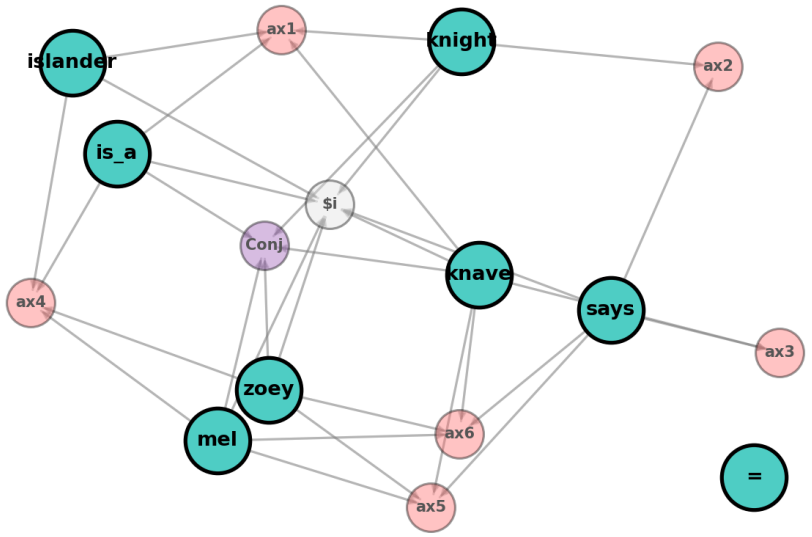
Olšák, Miroslav, Cezary Kaliszyk, and Josef Urban. "Property invariant embedding for automated reasoning." *arXiv preprint arXiv:1911.12073* (2019).

GNN_v1

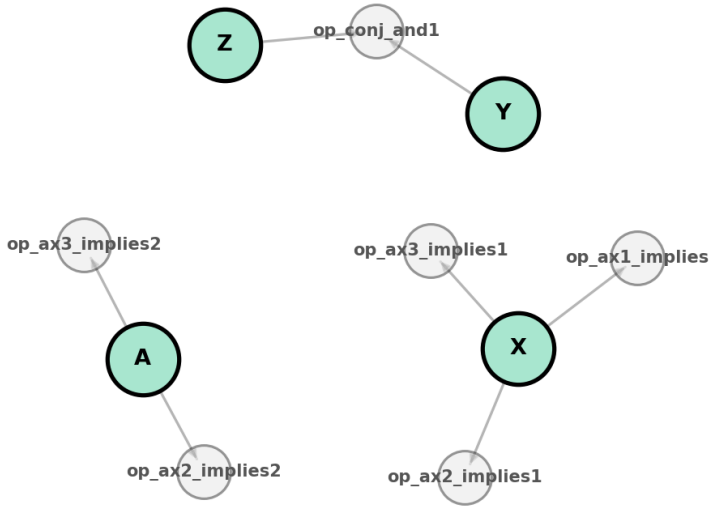
Load pt file from vampire → Build Graph → Initialization → Message Passing → Scoring → Loss

Symbol nodes → **Variable nodes** → **AST_OP nodes/clause** → **Type nodes** → **Clause nodes** → **Edges**

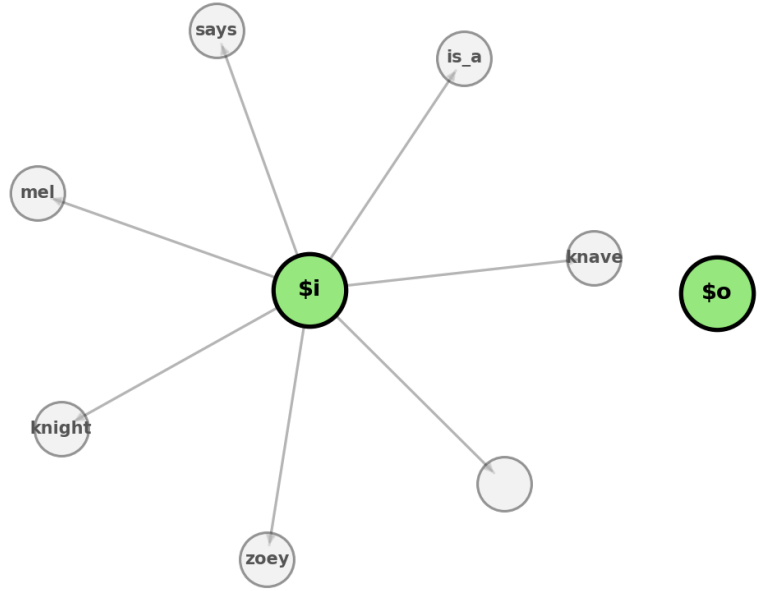
Symbol Nodes



Variable Nodes



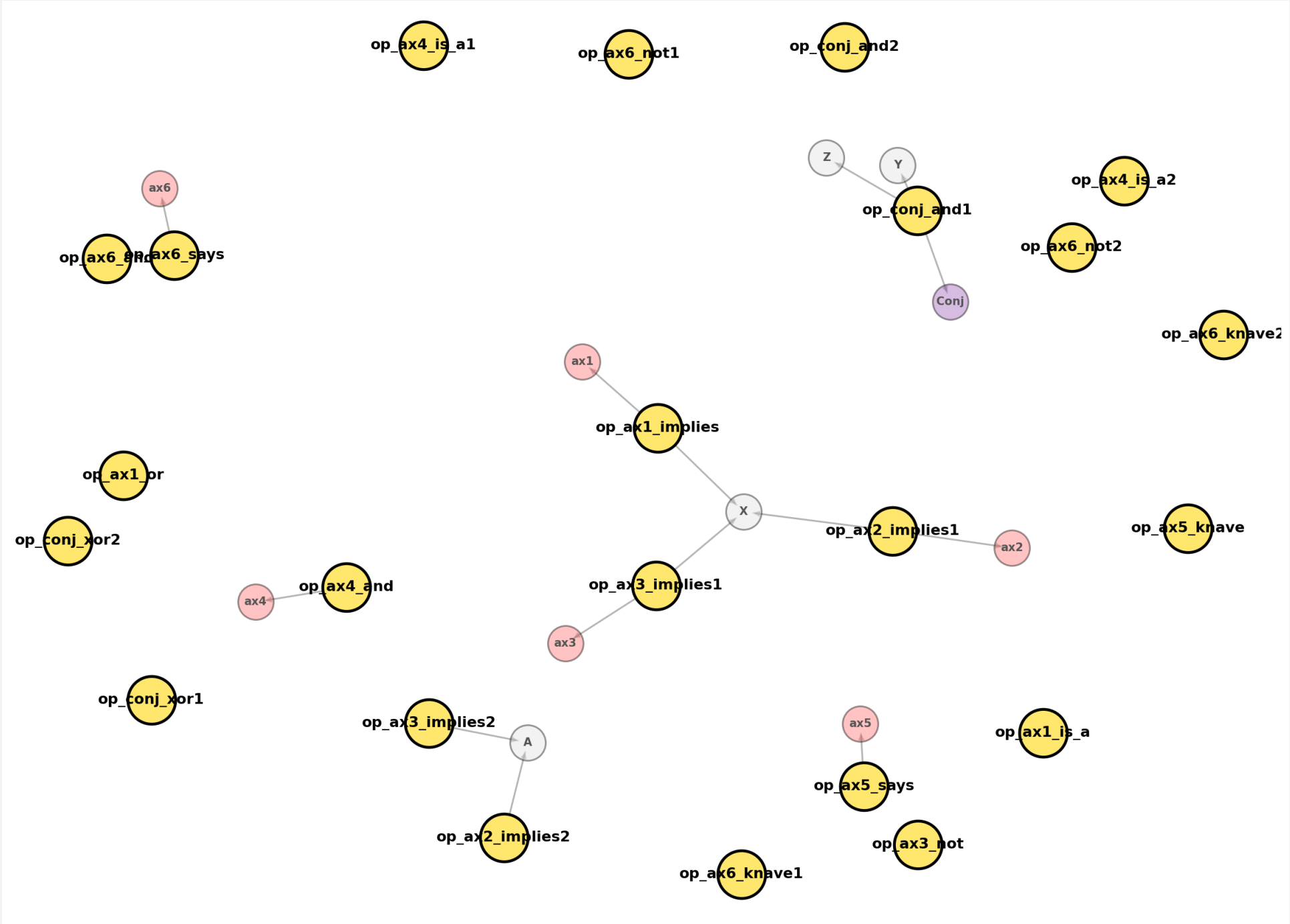
Type Nodes



types, sort_terms, symbols, formula_nodes, formula_roles

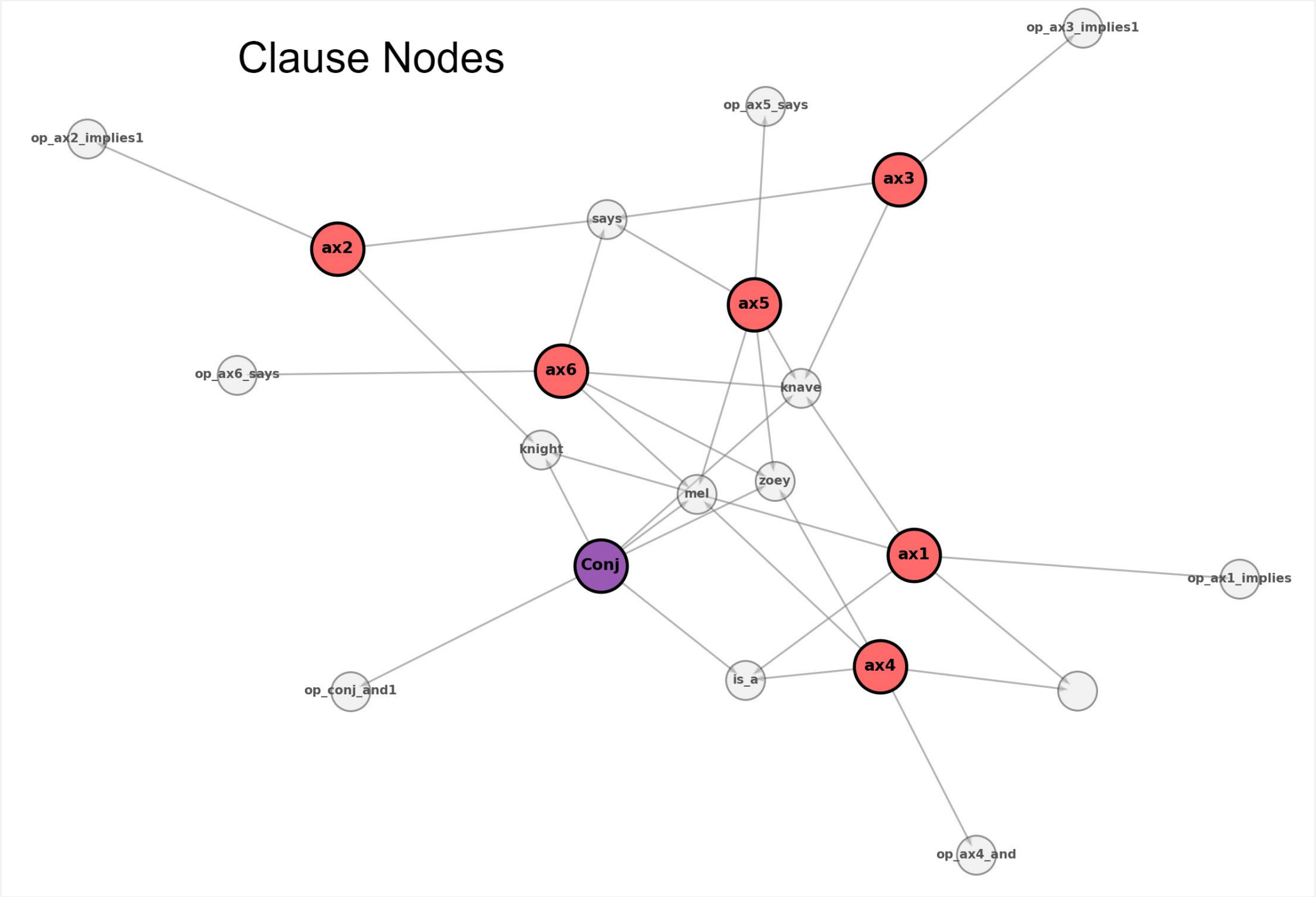
GNN_v1

- Symbol nodes
- Variable nodes
- **AST_OP nodes/clause**
- Type nodes
- Clause nodes
- Edges

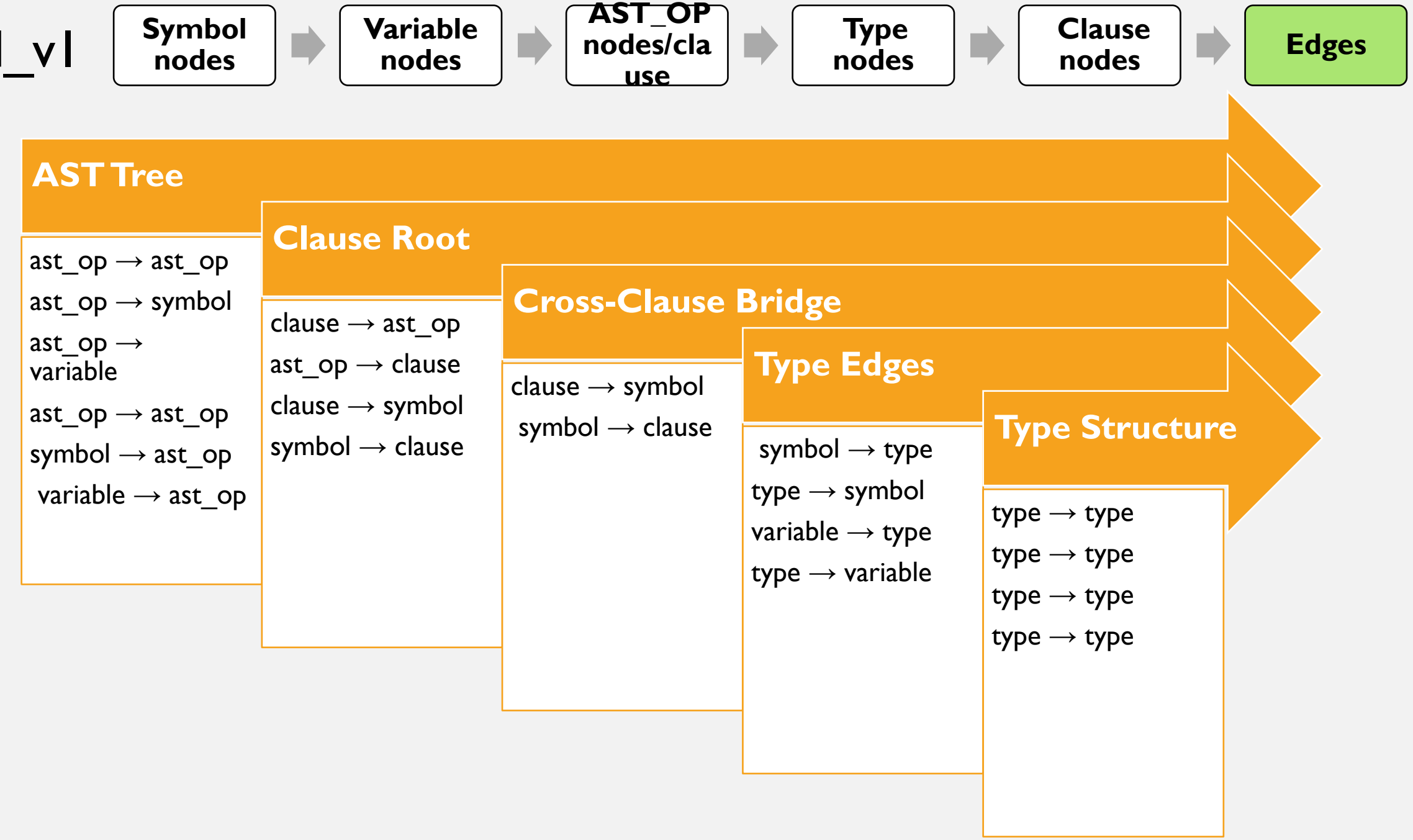


GNN_v1

- **Symbol nodes**
- **Variable nodes**
- **AST_OP nodes/clause**
- **Type nodes**
- **Clause nodes**
- **Edges**



GNN_v1


Symbol nodes
Variable nodes
AST_OP nodes/clause
Type nodes
Clause nodes
Edges
AST Tree
ast_op → ast_op
ast_op → symbol
ast_op → variable
ast_op → ast_op
symbol → ast_op
variable → ast_op
Clause Root
clause → ast_op
ast_op → clause
clause → symbol
symbol → clause
Cross-Clause Bridge
clause → symbol
symbol → clause
Type Edges
symbol → type
type → symbol
variable → type
type → variable
Type Structure
type → type
type → type
type → type
type → type

GNN_v1

Symbol nodes

↓

Variable nodes

↓

AST_OP nodes/clause

↓

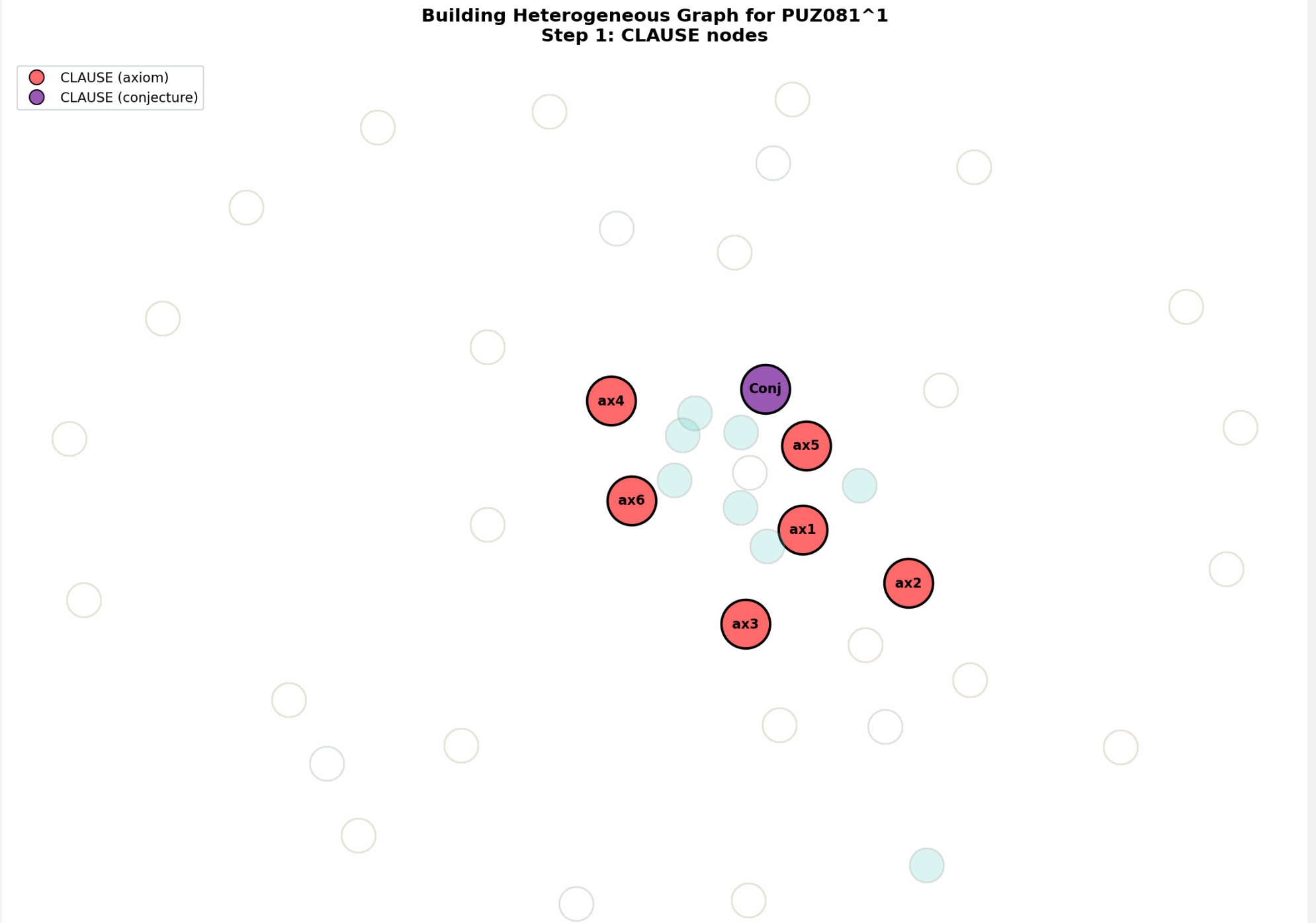
Type nodes

↓

Clause nodes

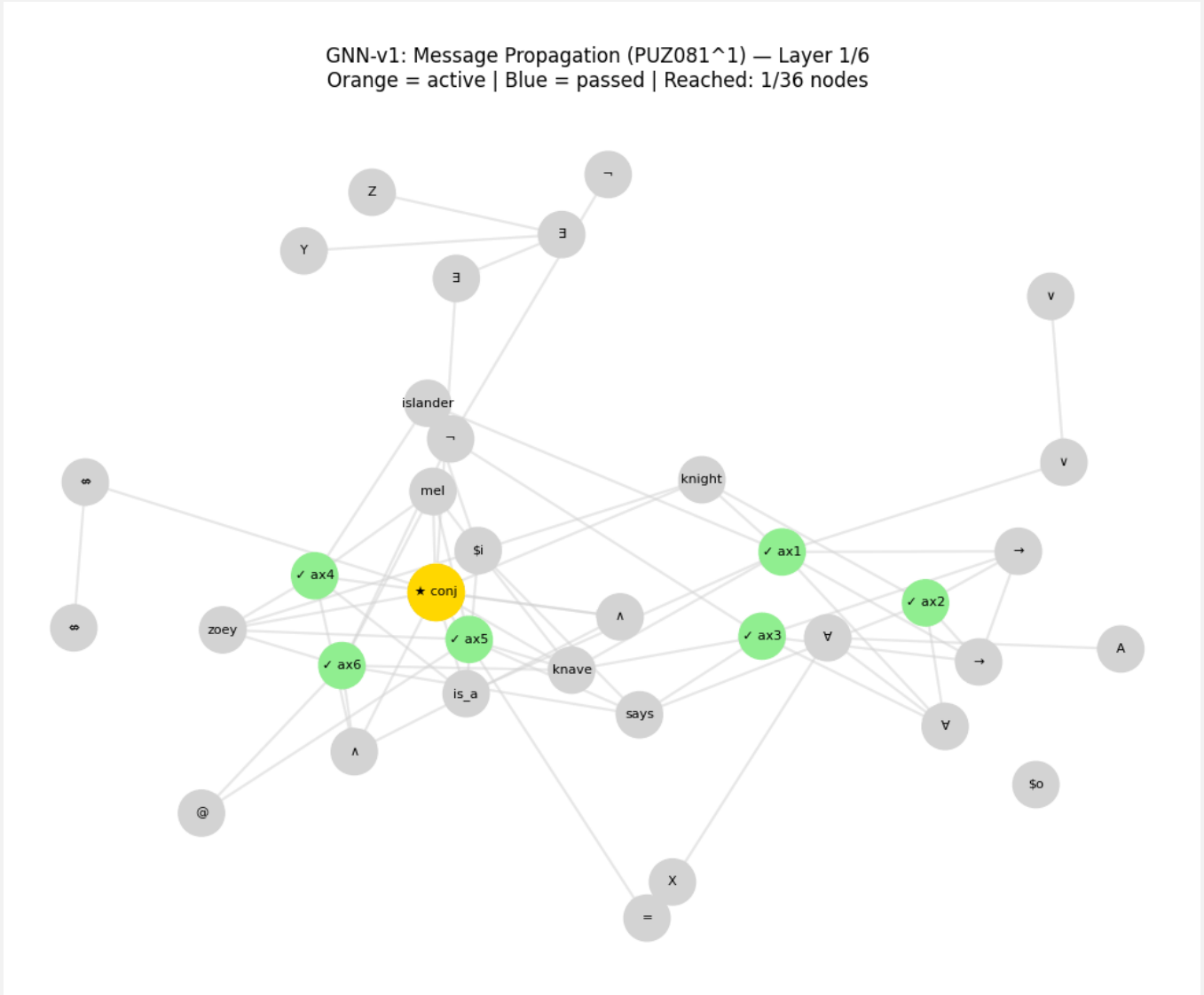
↓

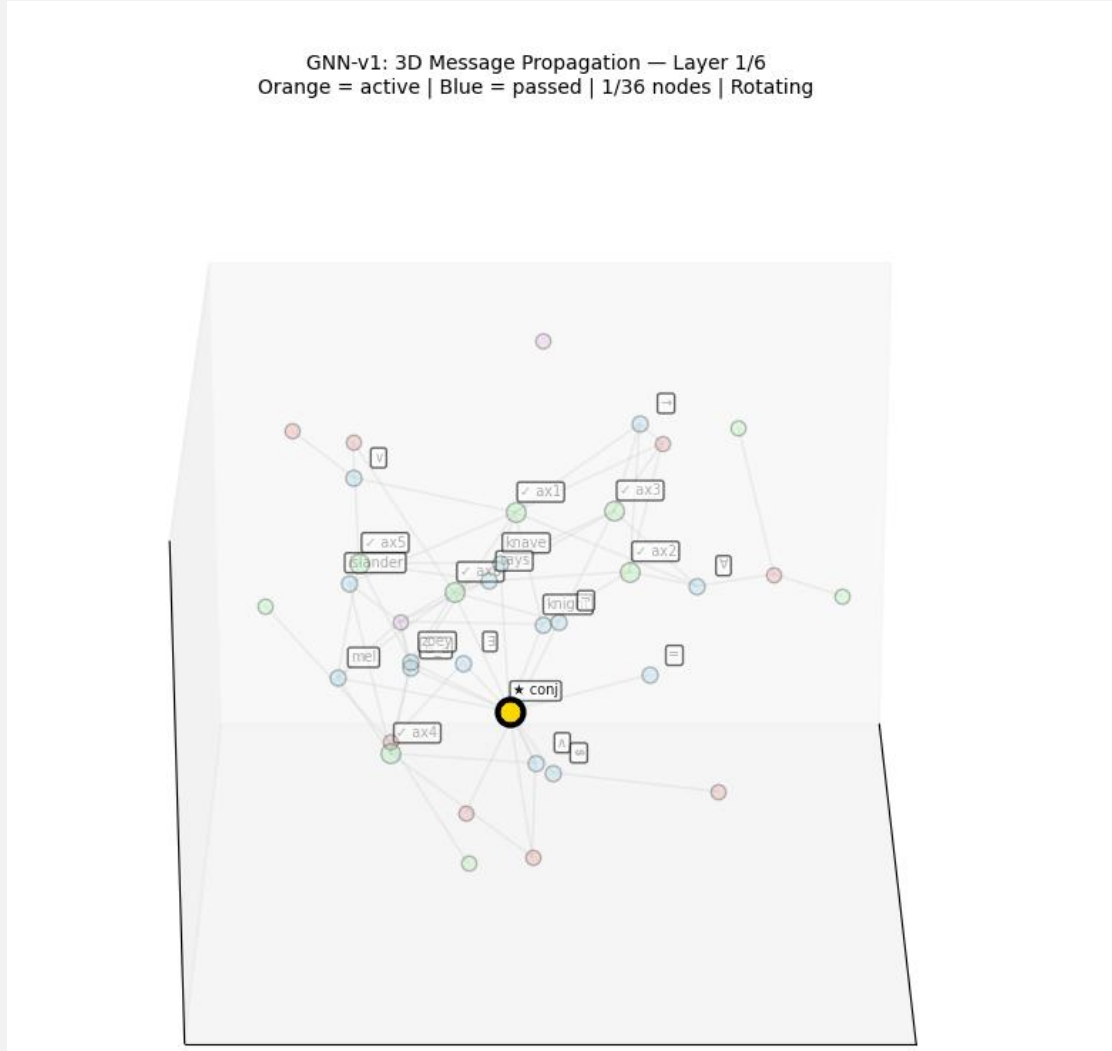
Edges



GNN_v1

Load pt file from vampire → Build Graph → Initialization → Message Passing → Scoring → Loss

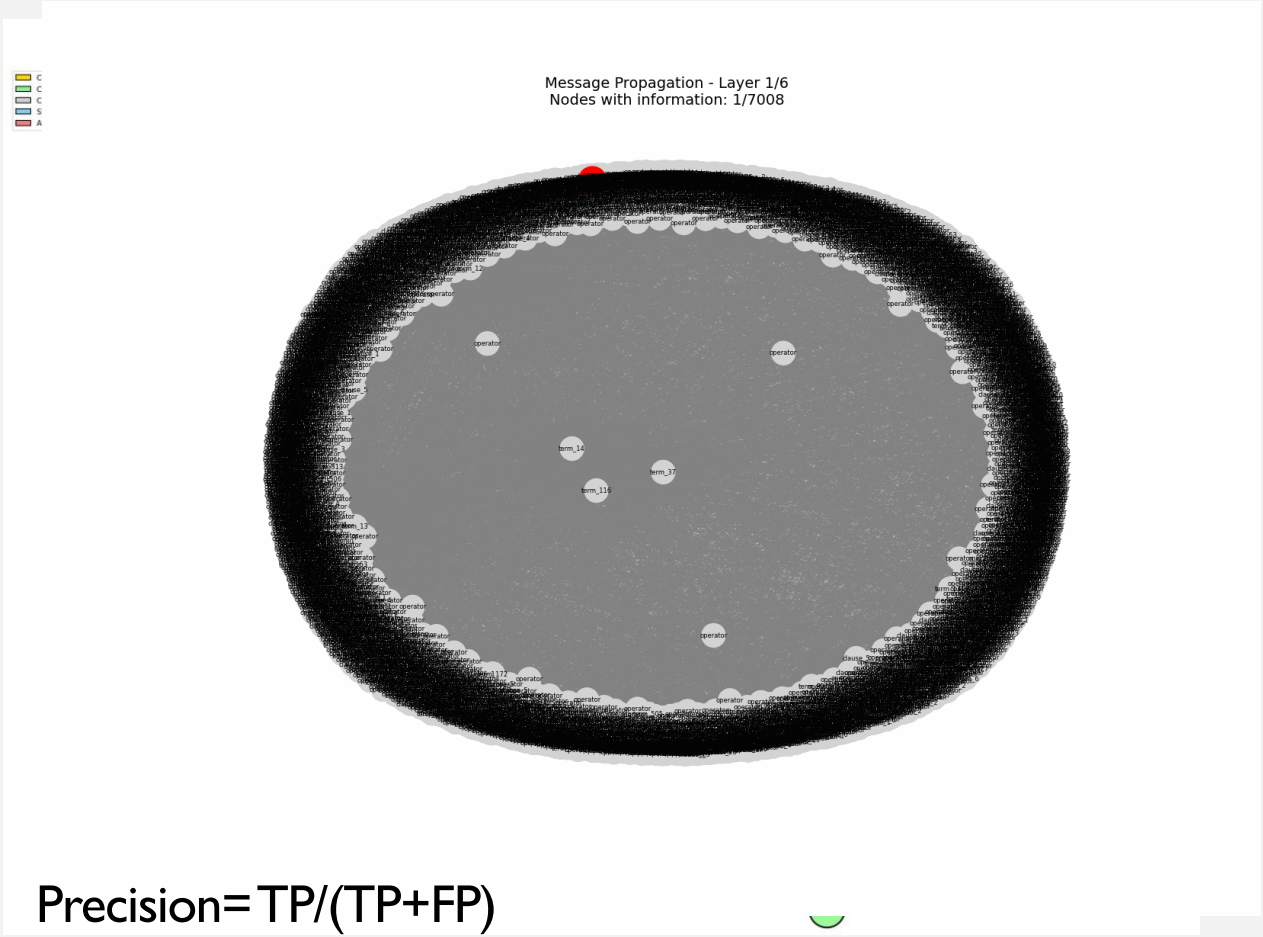




GNN_v1

Load pt file from vampire → Build Graph → Initialization → Message Passing → Scoring → Loss

Training	Testing
Graph of the training dataset	New Test Graph for the test set….
Fixed Trainig Graph	Fixed Graph Structure
GNN Weights (random)	Frozen weights
Training Embeddings	Inference Embeddings
Loss & Gradients	Predictions



Loss Function
1. Binary Cross Entropy Loss (BCE)
2. Focal Loss

Precision= TP/(TP+FP)
Recall = TP / (TP + FN)
Accuracy = Accuracy = (TP + TN) / (TP + TN + FP + FN)
F-score = 2 × (Precision × Recall) / (Precision + Recall)

Lin, Tsung-Yi, et al. "Focal loss for dense object detection." *Proceedings of the IEEE international conference on computer vision*. 2017.

IMPROVING GNN-V1

Design

- Too many edges and too many nodes

⇩

GNN_V2

Parameter Optimization

- Number of layers, Embedding size etc

⇩

Many variants

Architecture details

- Pre and postprocessing steps

⇩

GNN_V1-NoVocab

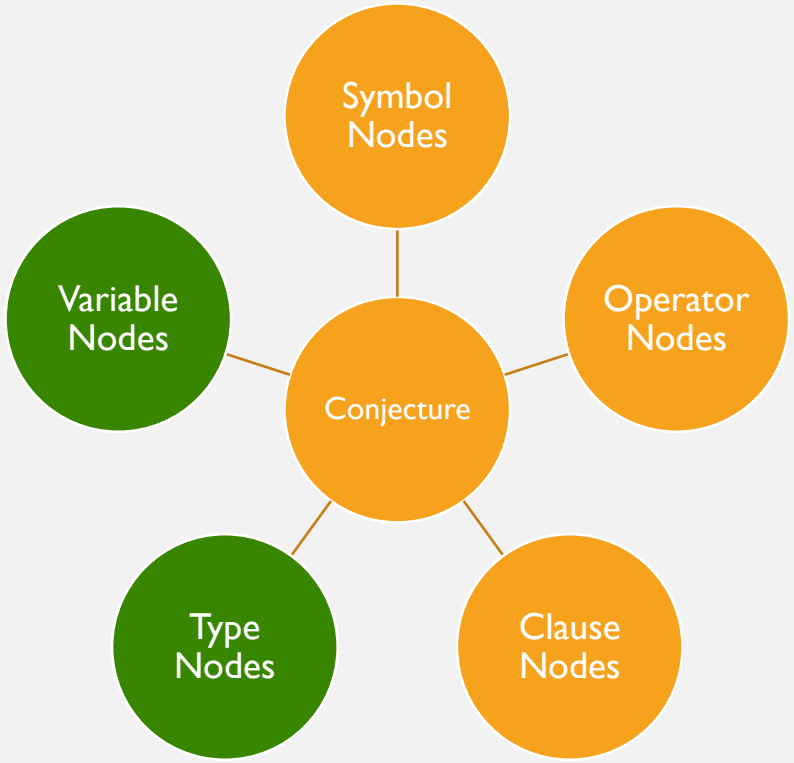
Model	Input (.pt)	Representation / Preprocessing	Core Layers	Special Components	Output	Loss
GNN-v1	Graph data	5-type graph → Init	6 × SAGEConv	—	Score	Focal Loss
GNN-v1	Graph data	5-type graph → Init	6 × SAGEConv	GNN_V1_NoVocab	Score	Focal Loss
GNN-v2	Graph data	3-type graph + 5D edge features → Init	6 × GATv2Conv	Attention mechanism	Score	Focal Loss

METHODOLOGY: GNN_V2

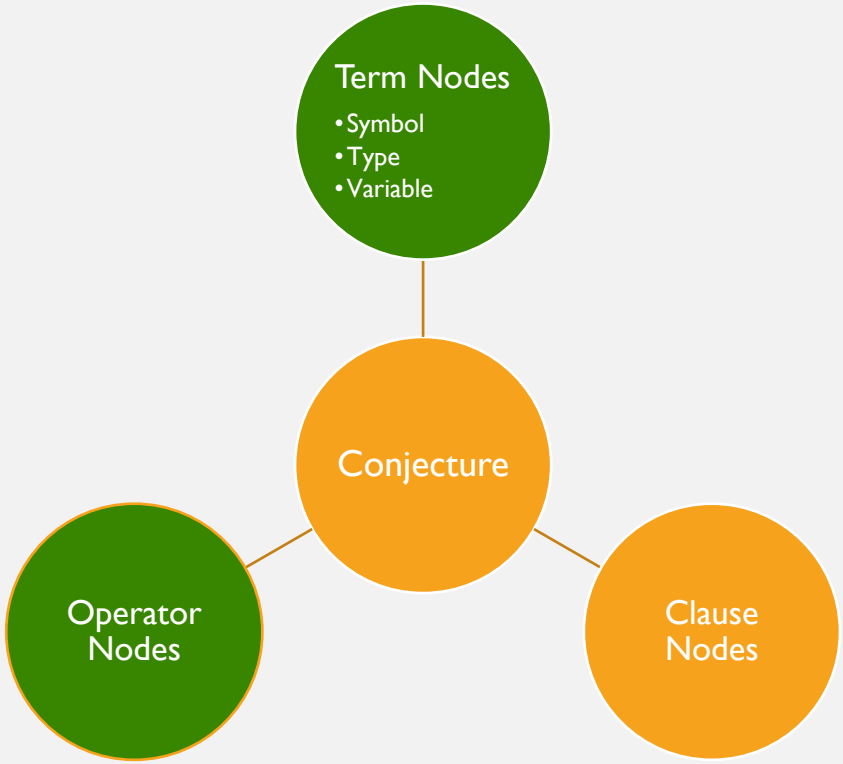
Problem (with too many axioms)

Graph based networks

Problem (with few relevant axioms)



GNN_V1



GNN_V2

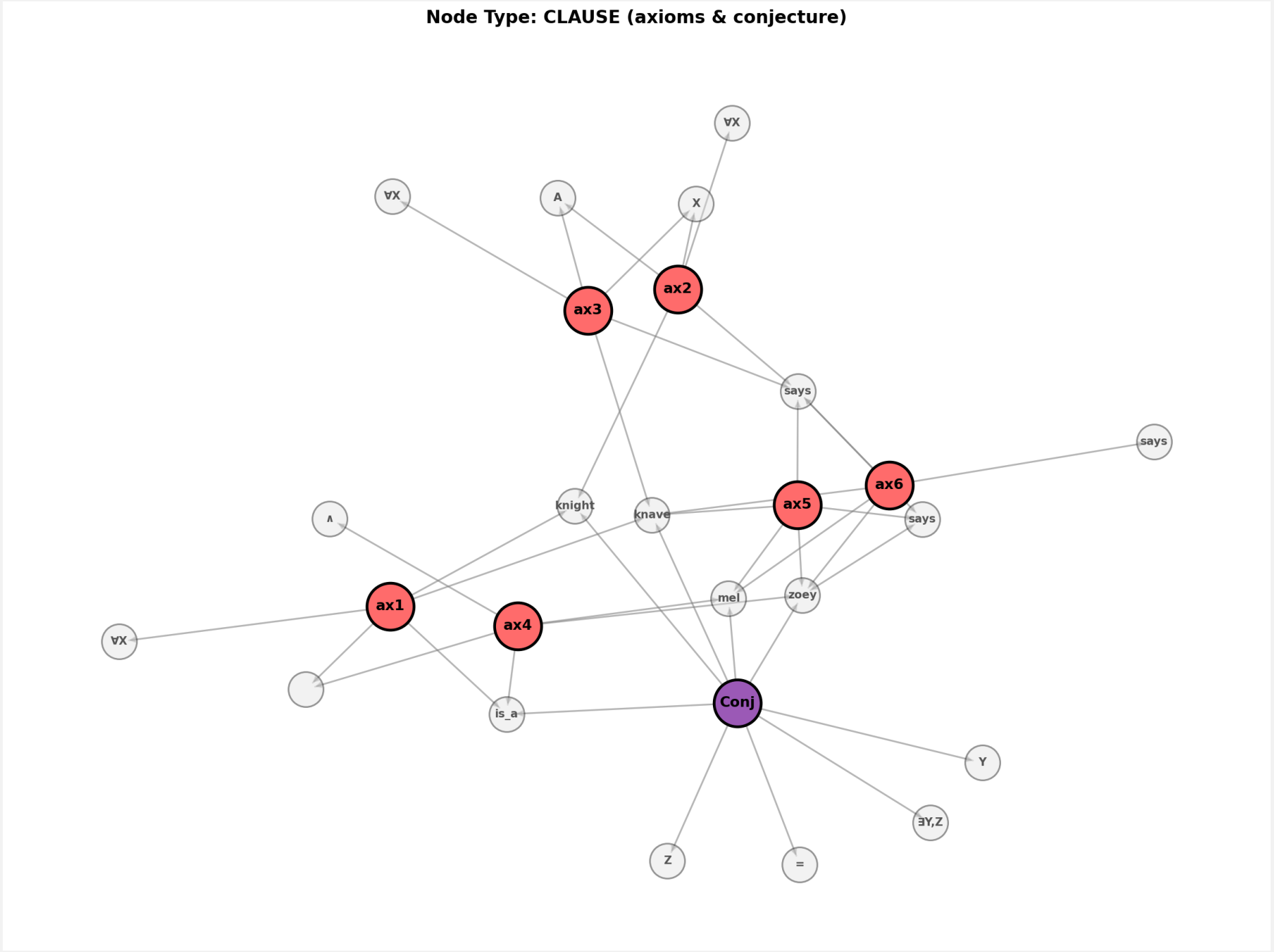
GNN_v2

Clause nodes

Term nodes (symbol, variable, type)

Operator nodes

Edges with 5D edge Features



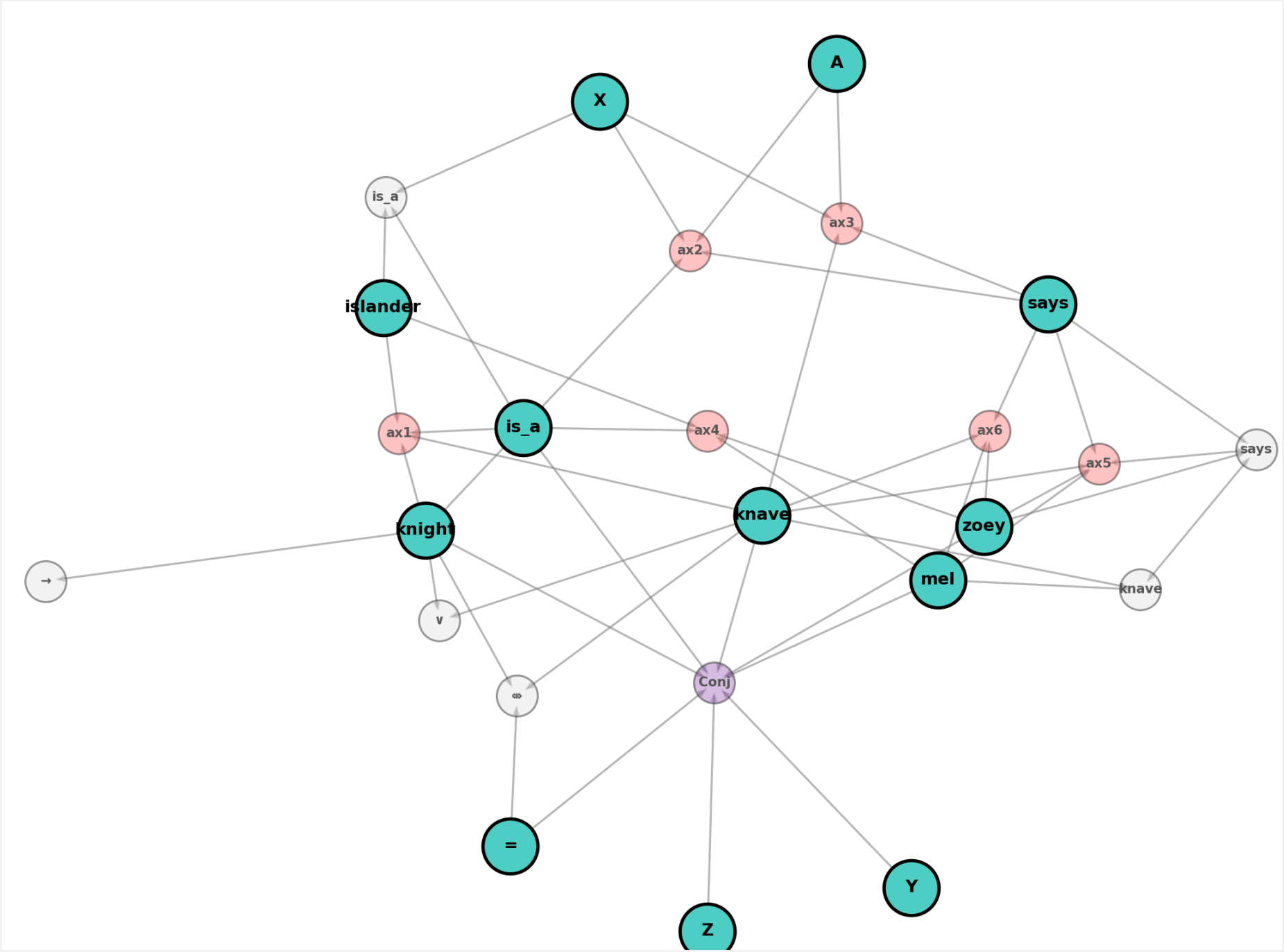
GNN_v2

Clause nodes

Term nodes (symbol, variable, type)

Operator nodes

Edges with 5D edge Features



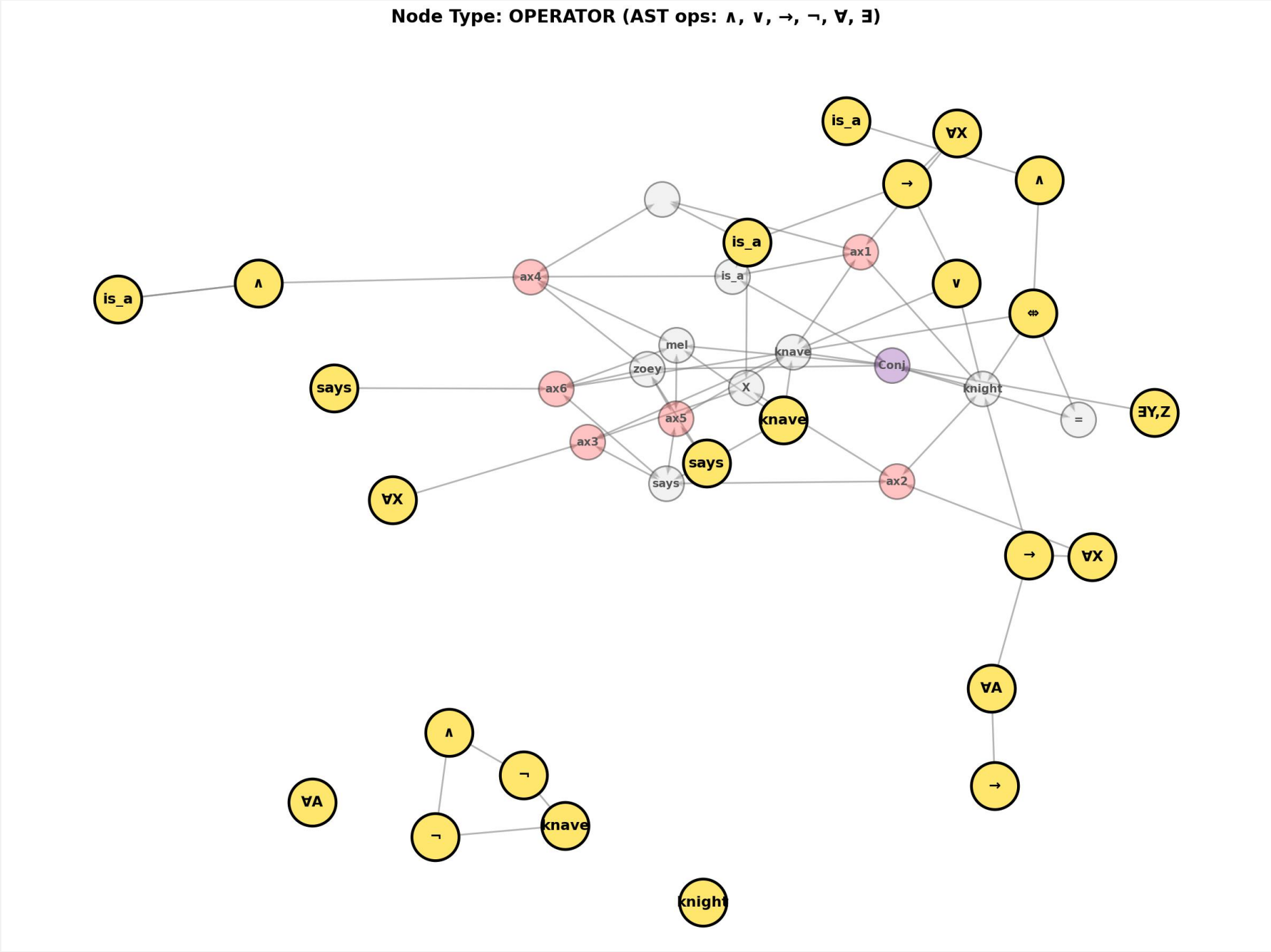
GNN_v2

Clause nodes

Term nodes (symbol, variable, type)

Operator nodes

Edges with 5D edge Features



GNN_v2

Clause nodes

Term nodes (symbol, variable, type)

Operator nodes

Edges with 5D edge Features

op_to_term — • operator → term

op_to_op — operator → operator

rev_op_to_term — term → operator

rev_op_to_op — operator → operator

clause_root — clause → operator

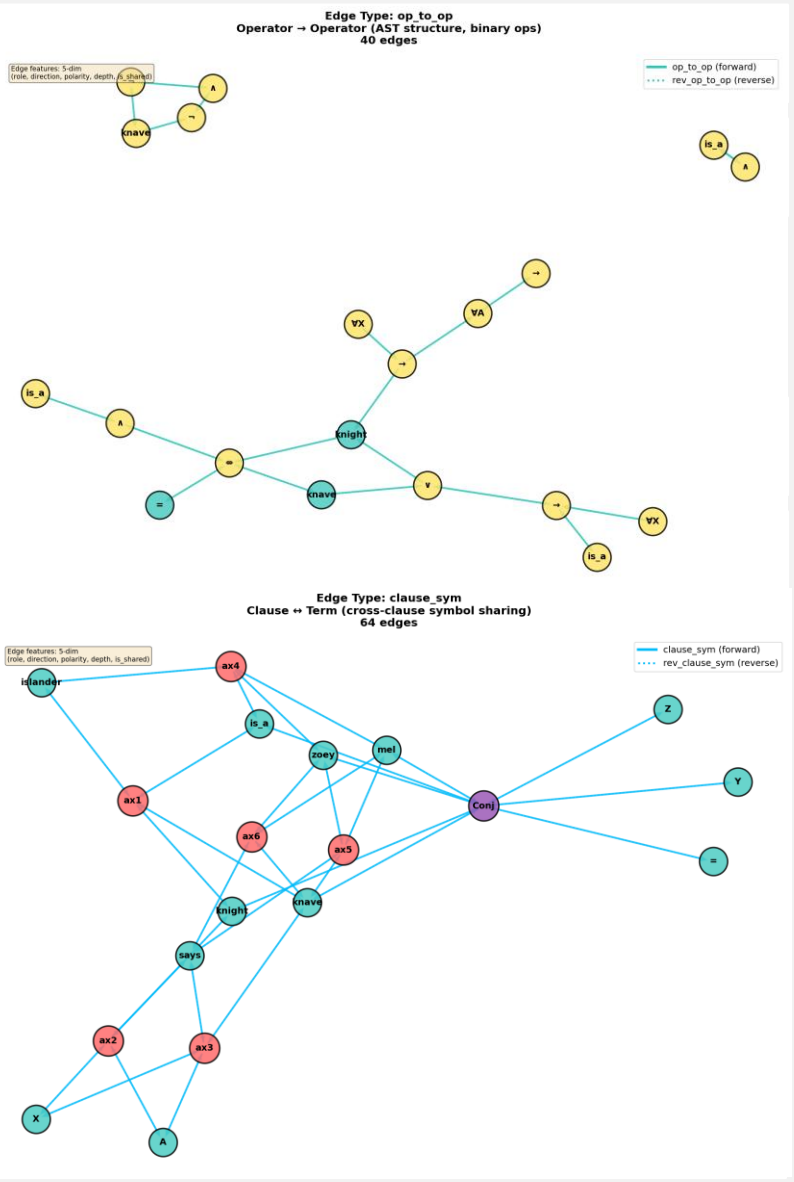
rev_clause_root — operator → clause

clause_root — clause → term

rev_clause_root — term → clause

clause_sym — clause → term

rev_clause_sym — term → clause



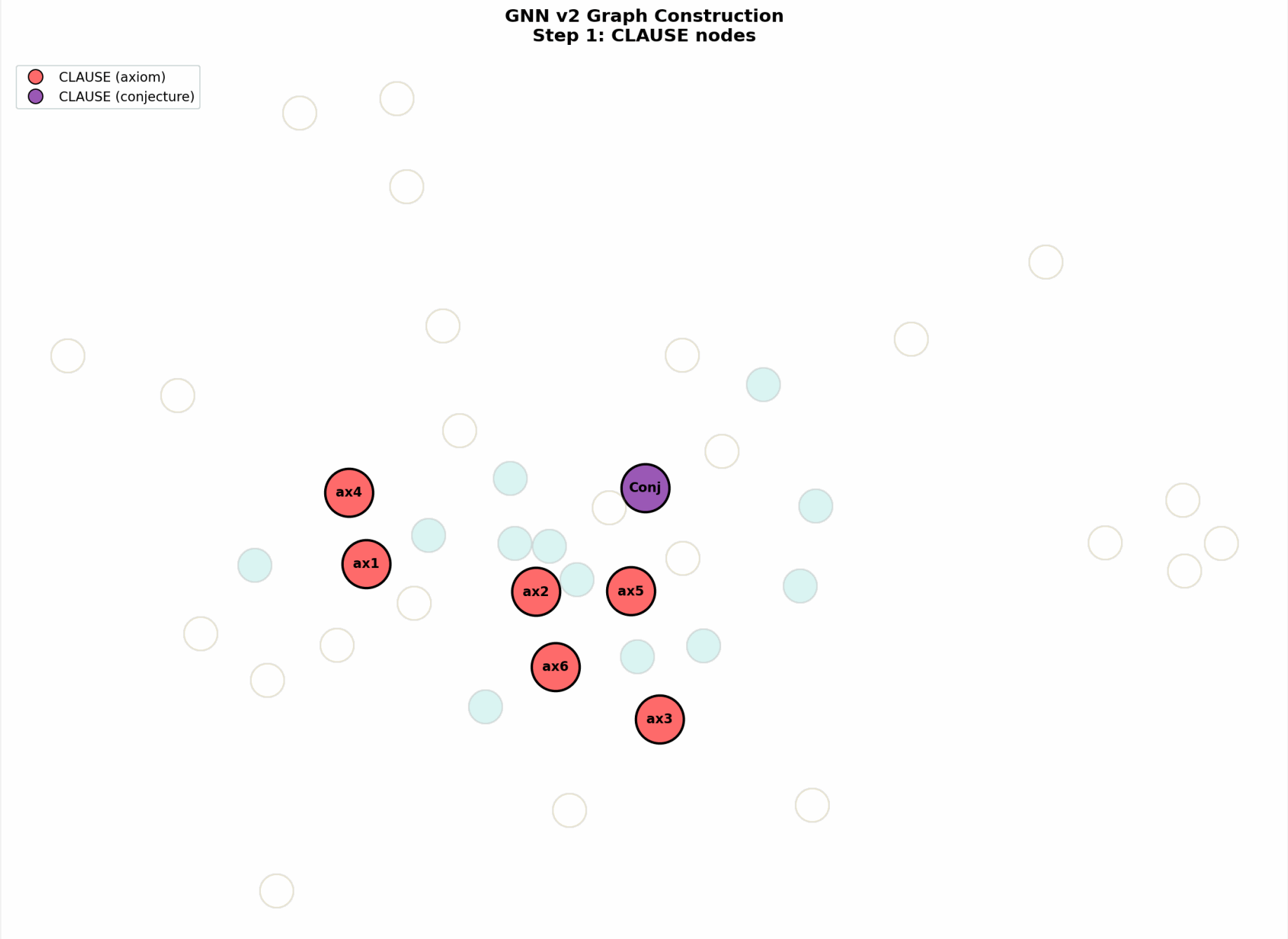
GNN_V2

Term nodes (symbol, variable, type)

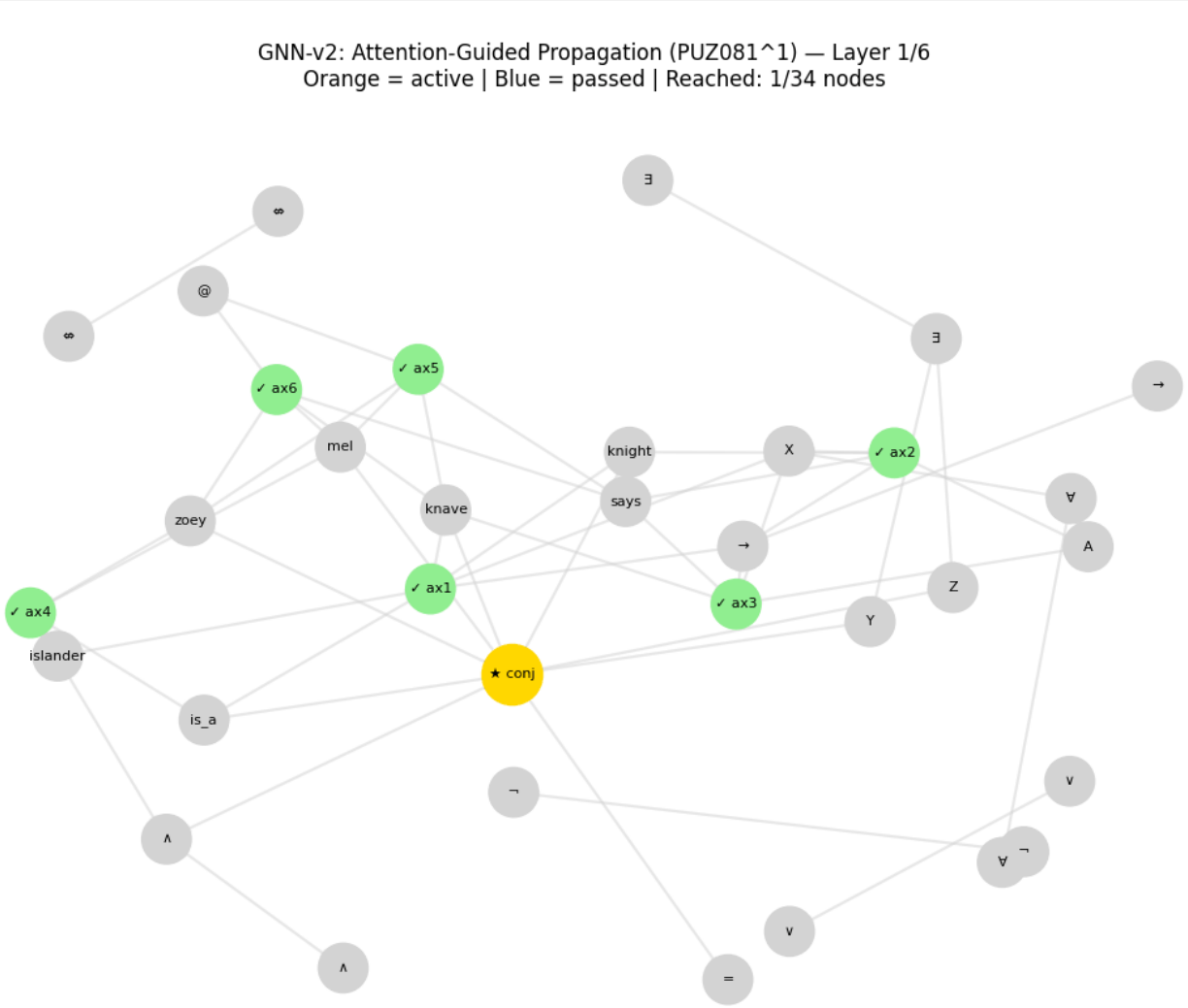
Operator nodes

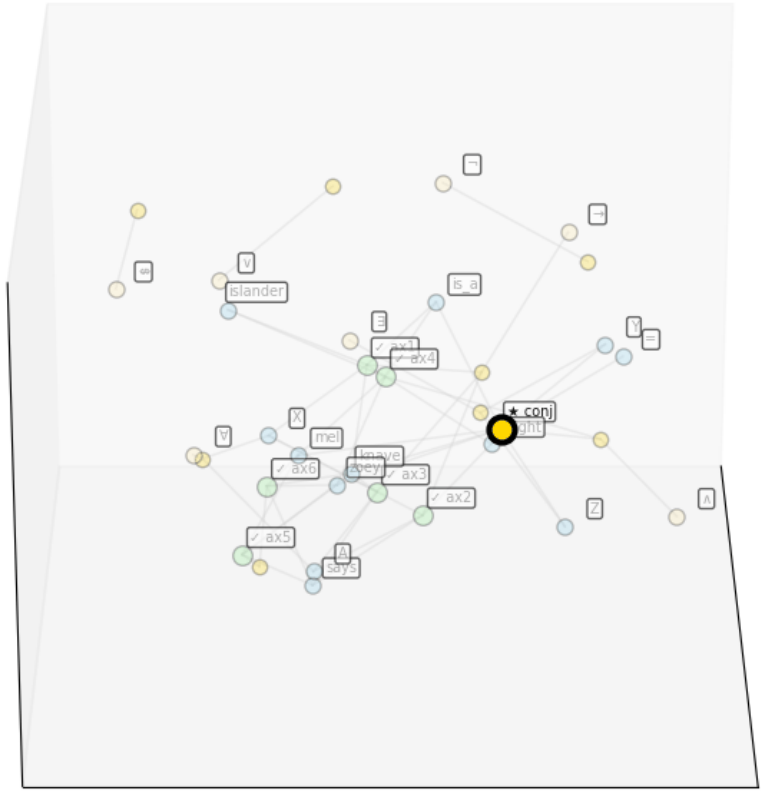
Clause nodes

Edges with 5D edge Features



GNN_V2


GNN-v2: Attention-Guided Propagation (PUZ081^1) — Layer 1/6
Orange = active | Blue = passed | Reached: 1/34 nodes
✓ ax6
✓ ax5
mel
knight
X
✓ ax2
zoey
knave
says
✓ ax1
✓ ax3
✓ ax4
islander
★ conj
is_a
Z
Y
A
∧
∨
=
¬


GNN-v2: 3D Attention-Guided Propagation — Layer 1/6
Orange = active | Blue = passed | 1/34 nodes | Rotating
islander
is_a
mel
knave
says
★ conj

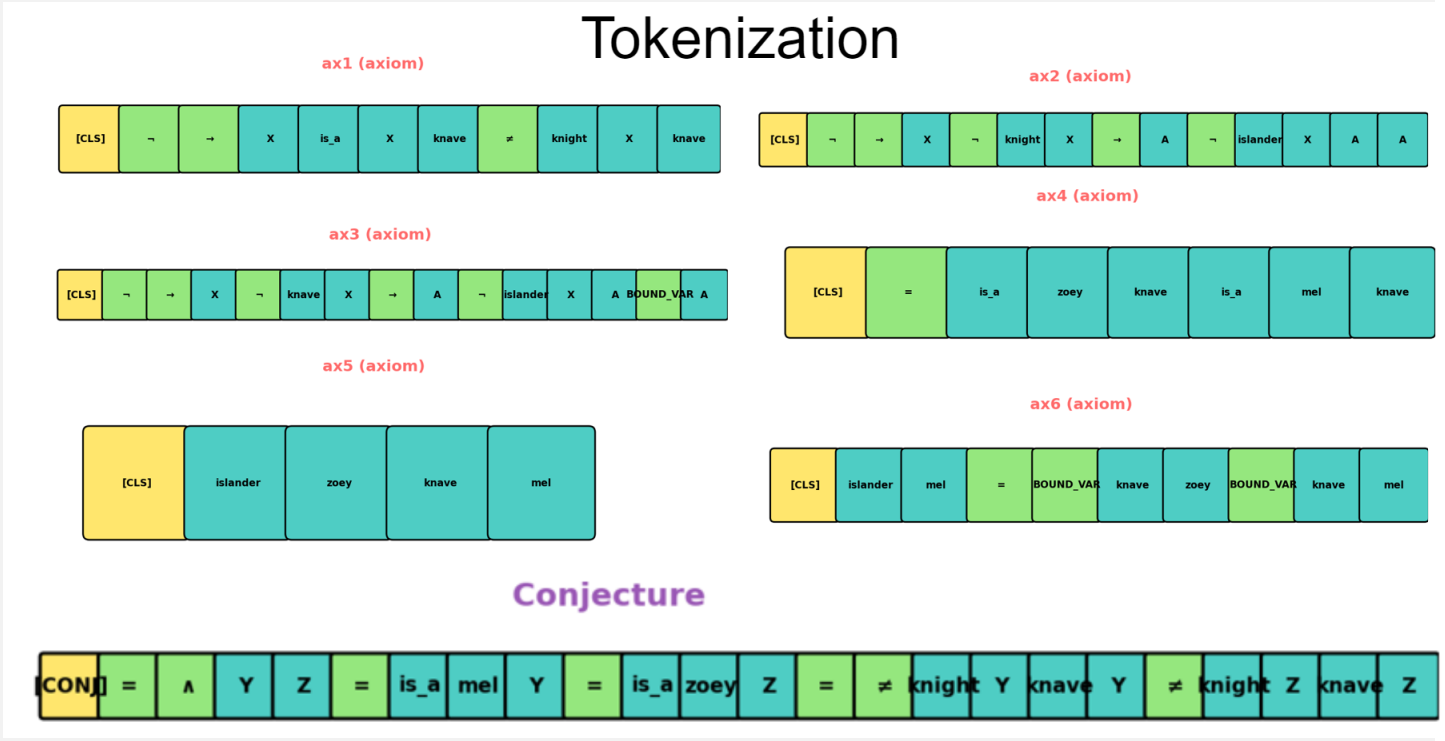
Improving GNN_v2

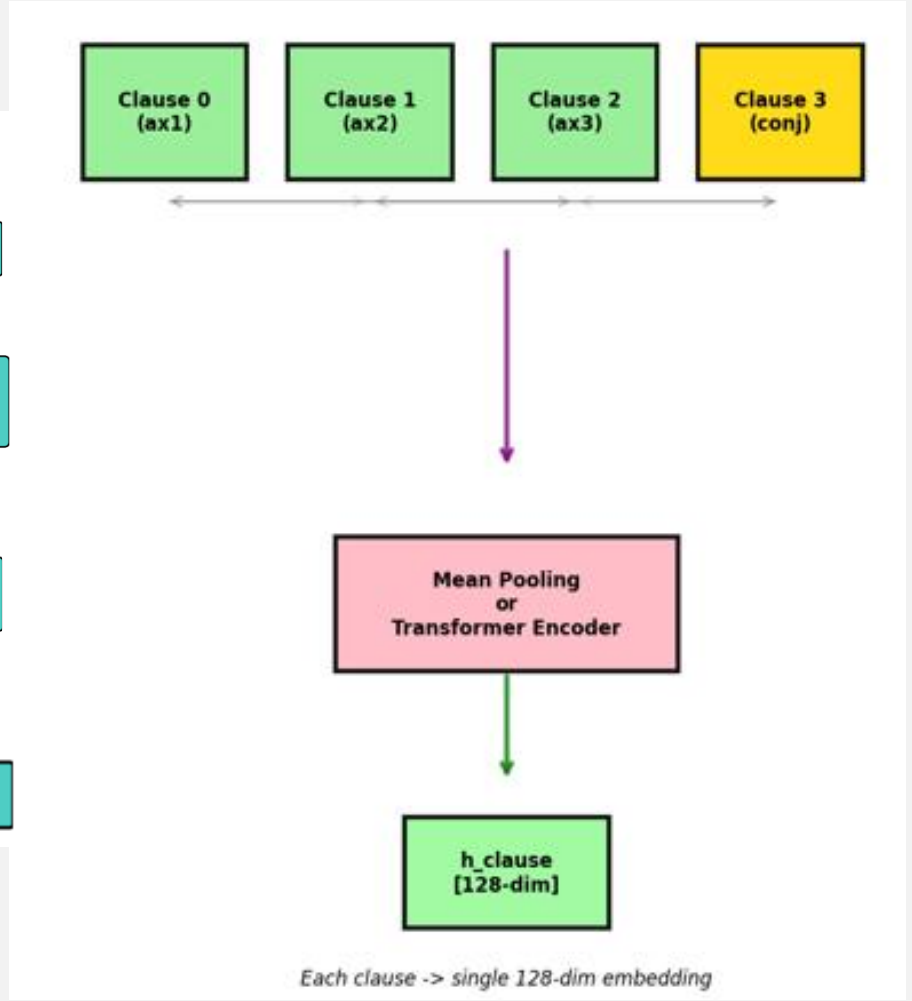
Design	Parameter Optimization	Architecture details
• try a totally new design	• Number of layers, Embedding size etc	• Pre and postprocessing steps
⇩	⇩	⇩
Transformer	Many variants	GNN_V2-NoVocab

Model	Input (.pt)	Representation / Preprocessing	Core Layers	Special Components	Output	Loss
GNN_V1	Graph data	5-type graph → Init	6 × SAGEConv	—	Score	Focal Loss
GNN_V1_NoVocab	Graph data	5-type graph → Init	6 × SAGEConv	GNN_V1_NoVocab	Score	Focal Loss
GNN_V2	Graph data	3-type graph + 5D edge features → Init	6 × GATv2Conv	Attention mechanism	Score	Focal Loss
GNN_V2_NoVocab	Graph data	3-type graph + 5D edge features → Init	6 × GATv2Conv	Attention mechanism	Score	Focal Loss

Transformer

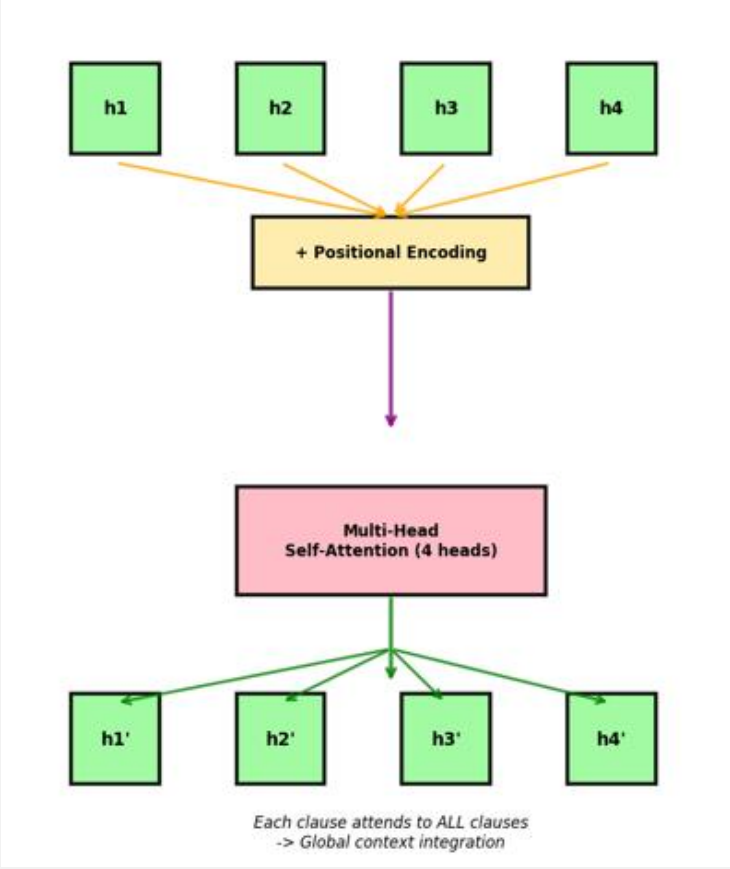
Tokenization → Per_ clause formula encoding → Cross-clause self-attention → Attention weight matrix → Conjecture-Conditioned Scoring

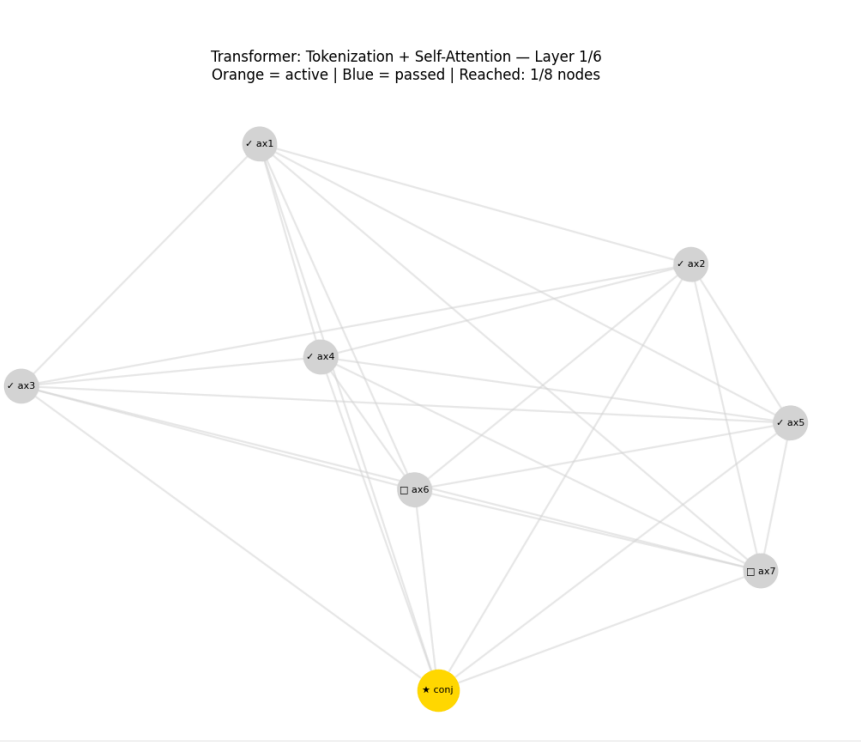


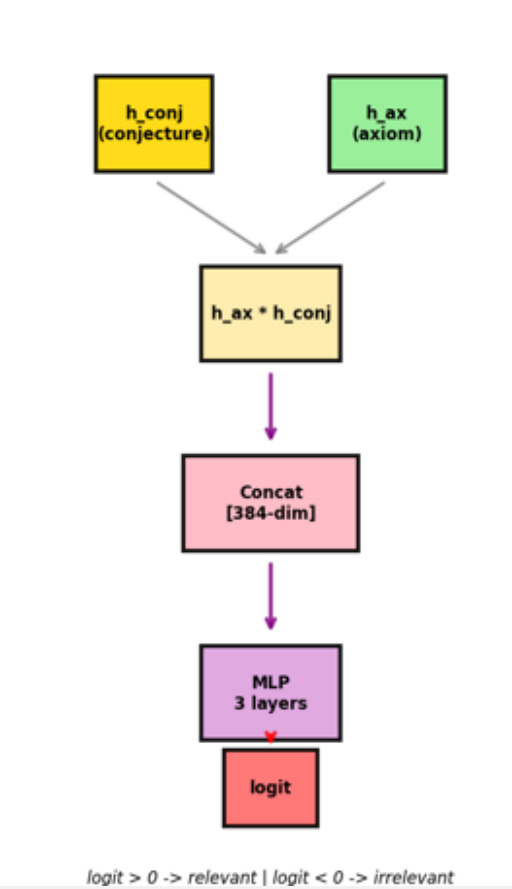


Transformer

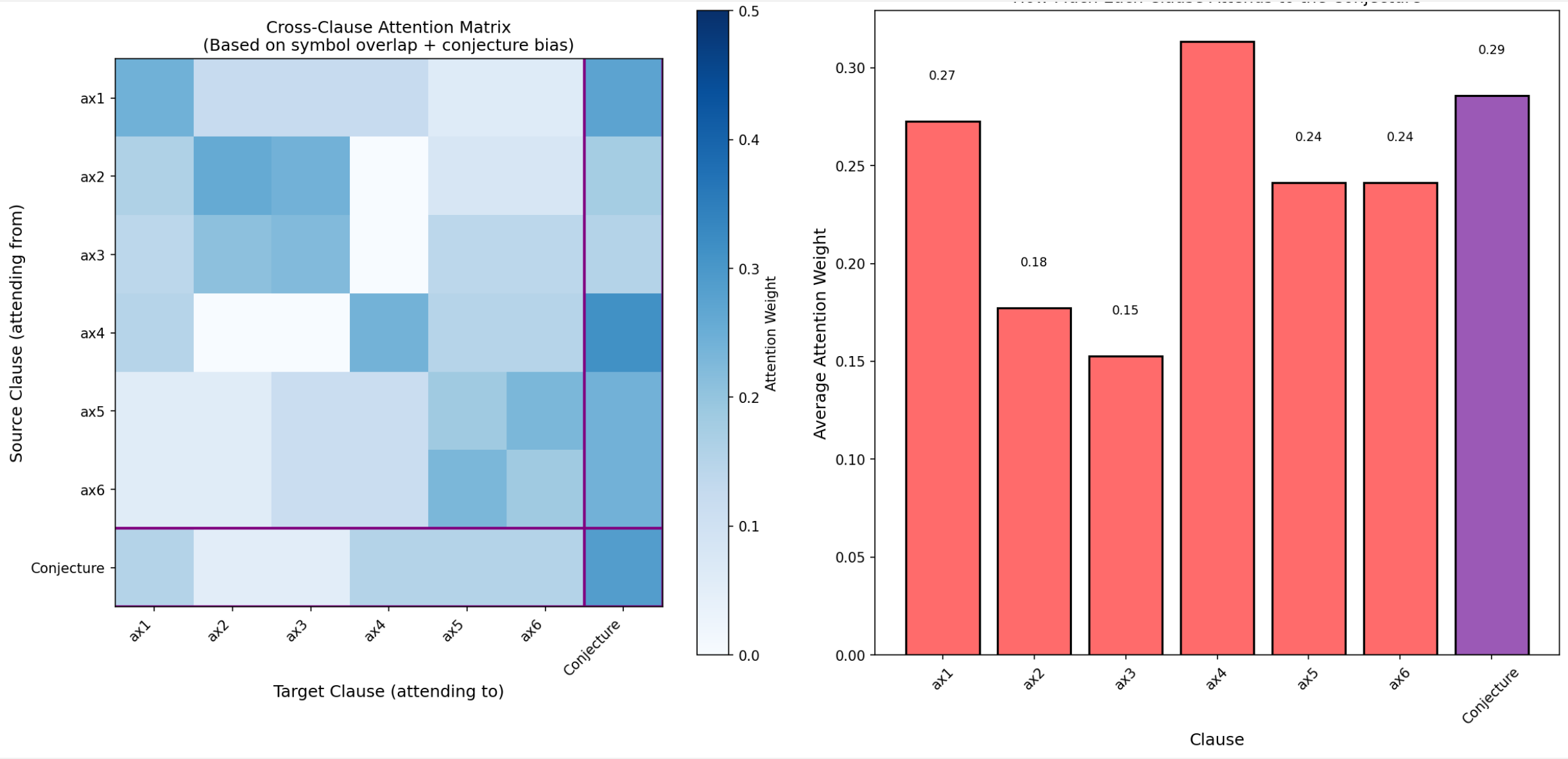
Tokenization → Per_ clause formula encoding → Cross-clause self-attention → Attention weight matrix → Conjecture-Conditioned Scoring







Transformer: Self-Attention Weight Matrix

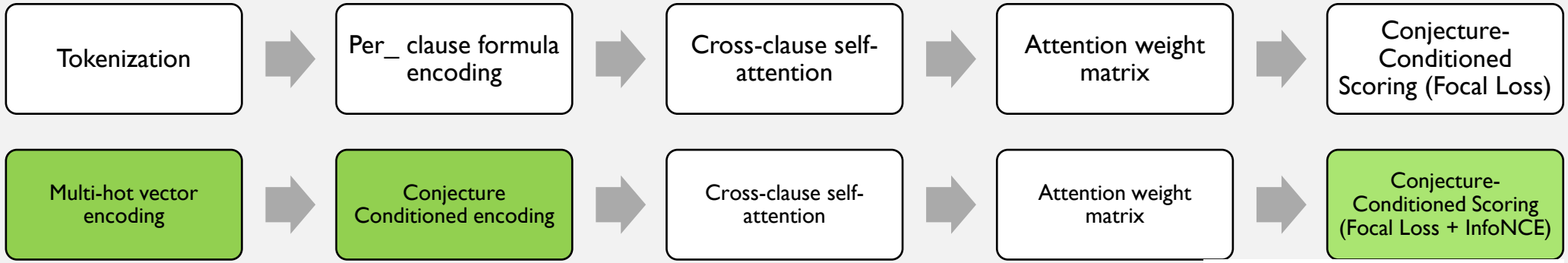


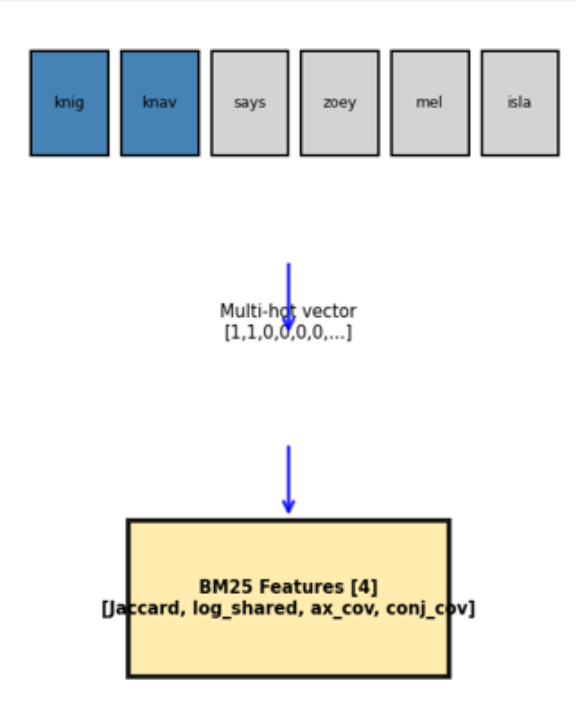
Improve it? IMPROVE IT?

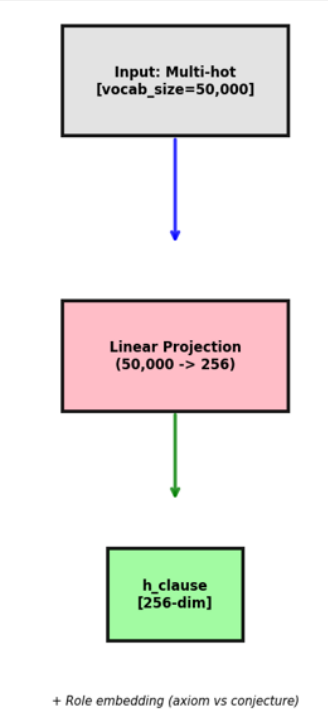
Design	Parameter Optimization	Architecture details
• Type information is lost	• Number of layers, Embedding size etc	• Pre and postprocessing steps
⇩	⇩	⇩
Contrastive Transformer	Many variants	w/o Positional Encoding, Typed Transformer

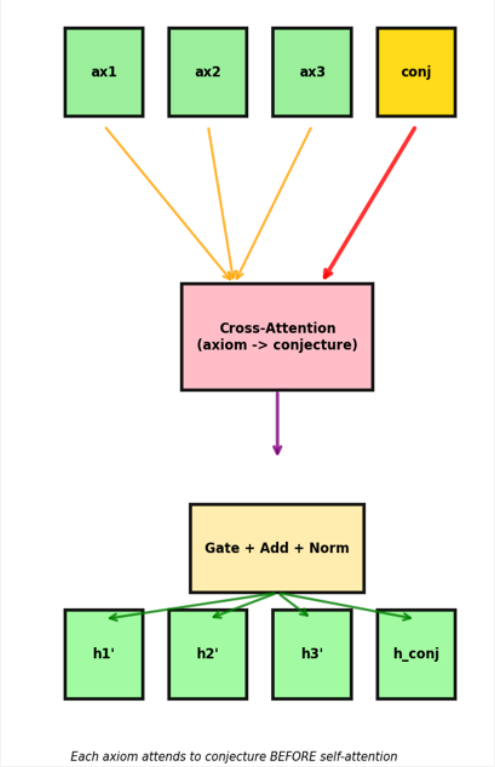
Model	Input (.pt)	Representation / Preprocessing	Core Layers	Special Components	Output	Loss
Transformer	Token data	Linearize → Token Embedding	Formula Encoder + 4 × Cross-Attention	Pooling / Transformer encoder	Score	—
Typed_ Transformer	Token data with type info	Linearize → Token Embedding	Formula Encoder + 4 × Cross-Attention	Pooling / Transformer encoder	Score	—
Transformer_w/o_ Positional Encoding	Token data	Linearize → Token Embedding	Formula Encoder w/o Positional encoding + 4 × Cross-Attention	Pooling / Transformer encoder	Score	—
Contrastive	Feature data	Multi-hot + BM25 (4) → Projection	2 × Cross-Attention + 4 × Self-Attention	Axiom → Conjecture alignment	Dual Score	Focal Loss + InfoNCE

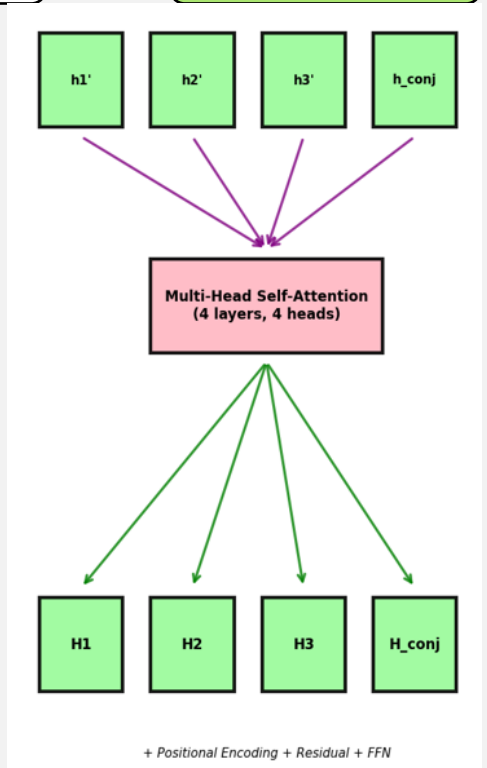
Contrastive Transformer







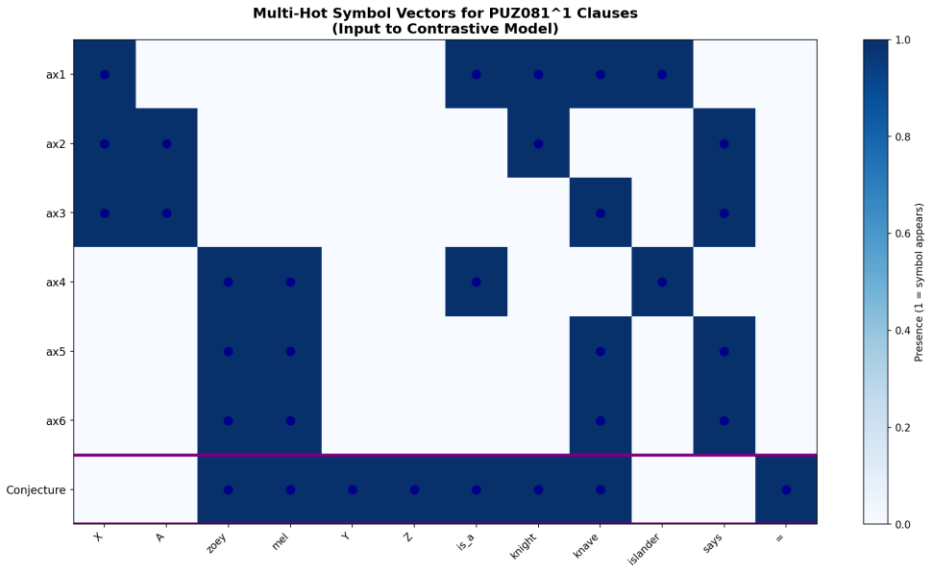


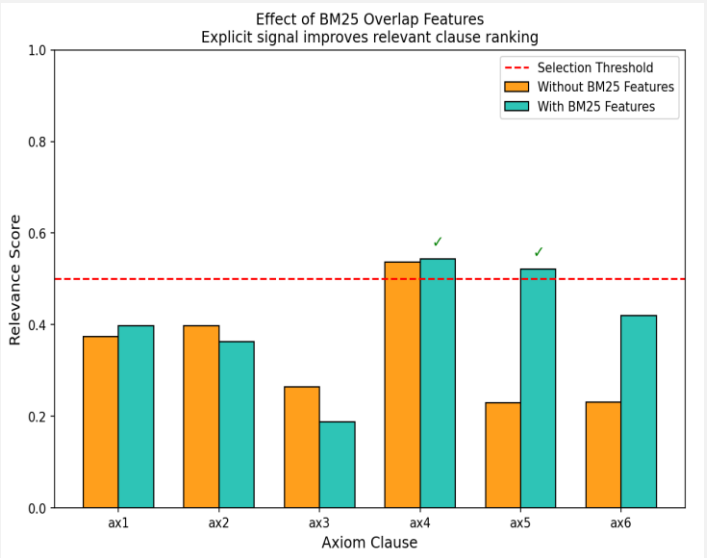


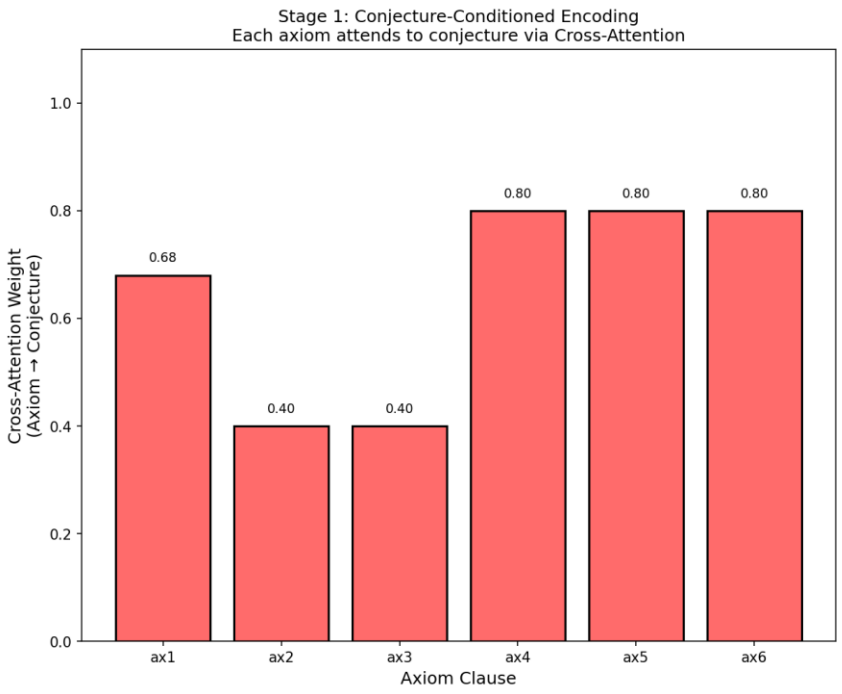
Mikuła, Maciej, et al. "Magnushammer: A transformer-based approach to premise selection." *International Conference on Learning Representations*. Vol. 2024. 2024.

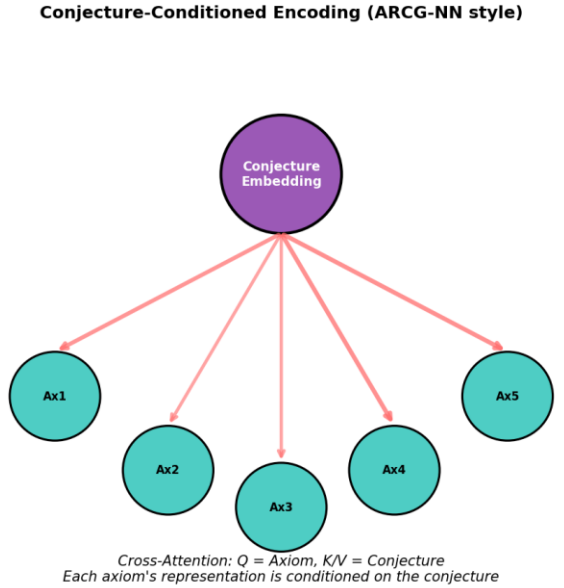
Contrastive Transformer

- Multi-hot vector encoding
- Conjecture Conditioned encoding
- Cross-clause self-attention
- Attention weight matrix
- Conjecture-Conditioned Scoring (Focal Loss + InfoNCE)

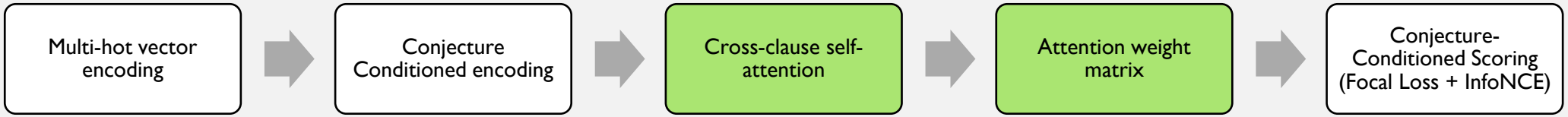


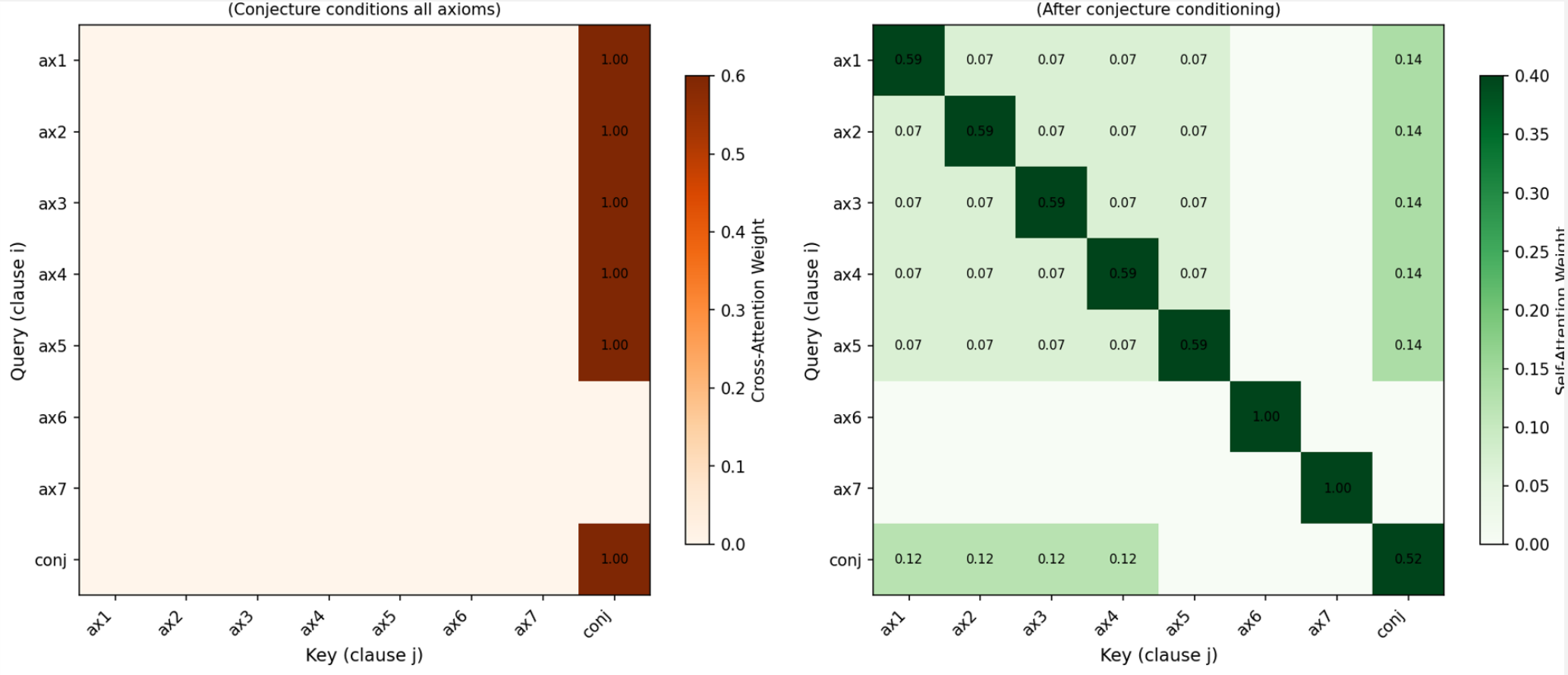




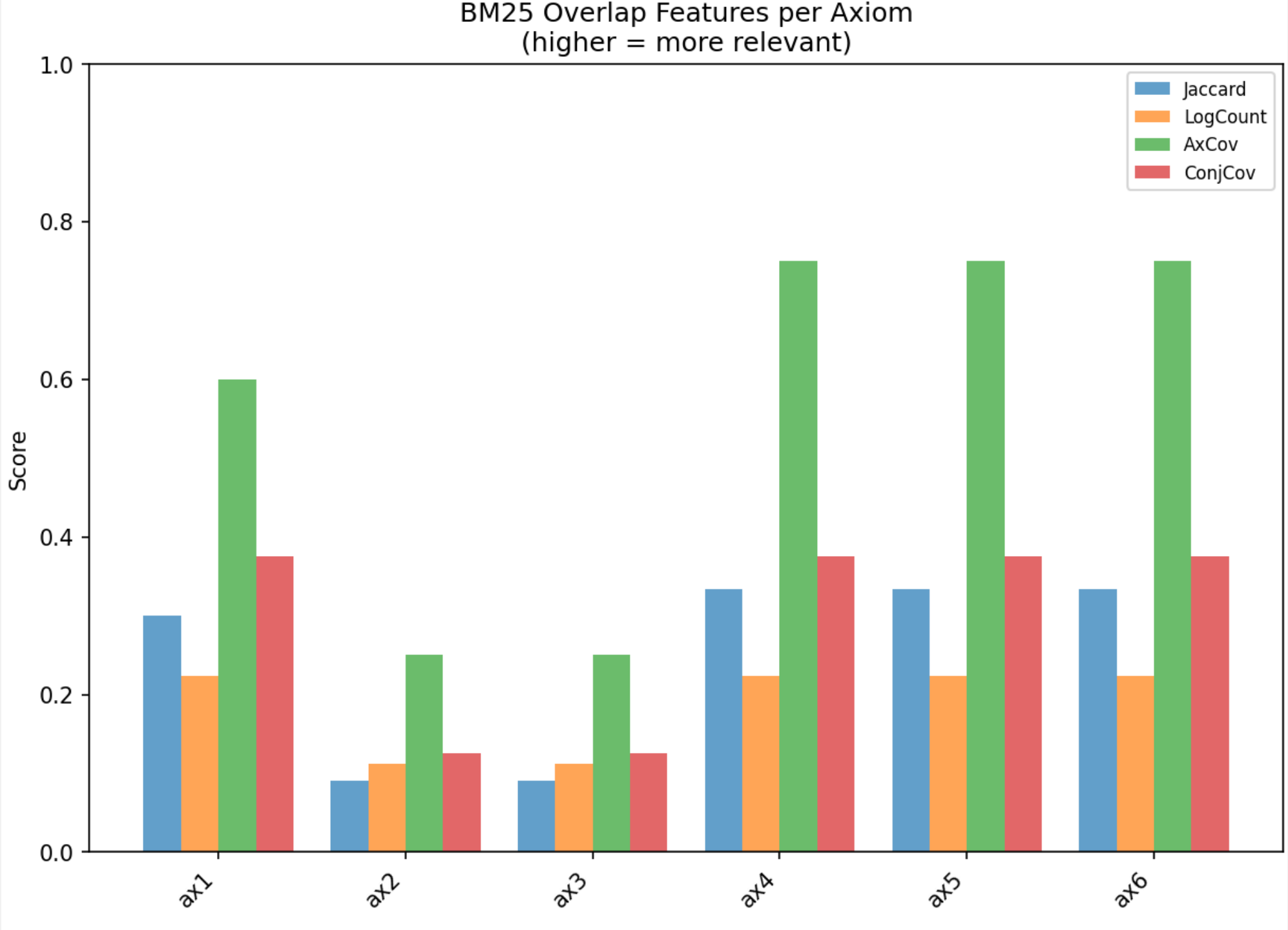


Contrastive Transformer





CROSS ATTENTION: AXIOM TO CONJECTURE
SELF ATTENTION ON ALL AXIOMS


BM25 Overlap Features per Axiom
(higher = more relevant)
Score
1.0
0.8
0.6
0.4
0.2
0.0
Jaccard
LogCount
AxCov
ConjCov
ax1
ax2
ax3
ax4
ax5
ax6

Improve it? IMPROVE IT?

Design	Parameter Optimization	Architecture details
• Future work	• Number of layers, Embedding size etc	• Pre and postprocessing steps
⇩	⇩	⇩
Future Work	Many variants	w/o Positional Encoding, Typed Transformer

Model	Input (.pt)	Representation / Preprocessing	Core Layers	Special Components	Output	Loss
Contrastive	Feature data	Multi-hot + BM25 (4) → Projection	2 × Cross-Attention + 4 × Self-Attention	Axiom → Conjecture alignment	Dual Score	Focal Loss + InfoNCE
Typed_Contrastive	Feature data including type	Multi-hot + BM25 (4) → Projection	2 × Cross-Attention + 4 × Self-Attention	Axiom → Conjecture alignment	Dual Score	Focal Loss + InfoNCE
Contrastive_No Positional Encoding	Feature data	Multi-hot + BM25 (4) → Projection	2 × Cross-Attention + 4 × Self-Attention, No Positional Encoding	Axiom → Conjecture alignment	Dual Score	Focal Loss + InfoNCE

EXPERIMENTS

- Two strategies:
  - Best X axioms (best 10,20, 50,100, 250, 512(baseline)
  - All axioms above Logit Score X (min 0.5, 0.0 , -0.5, -1.0 , -2.0, -3.0, -5.0
  - T- Time (1s, 10s)
- Dataset:
  - Training: 26,000 problems from Isabelle (80/20 train/val split)
  - Testing : 8900 holdout problems

Model

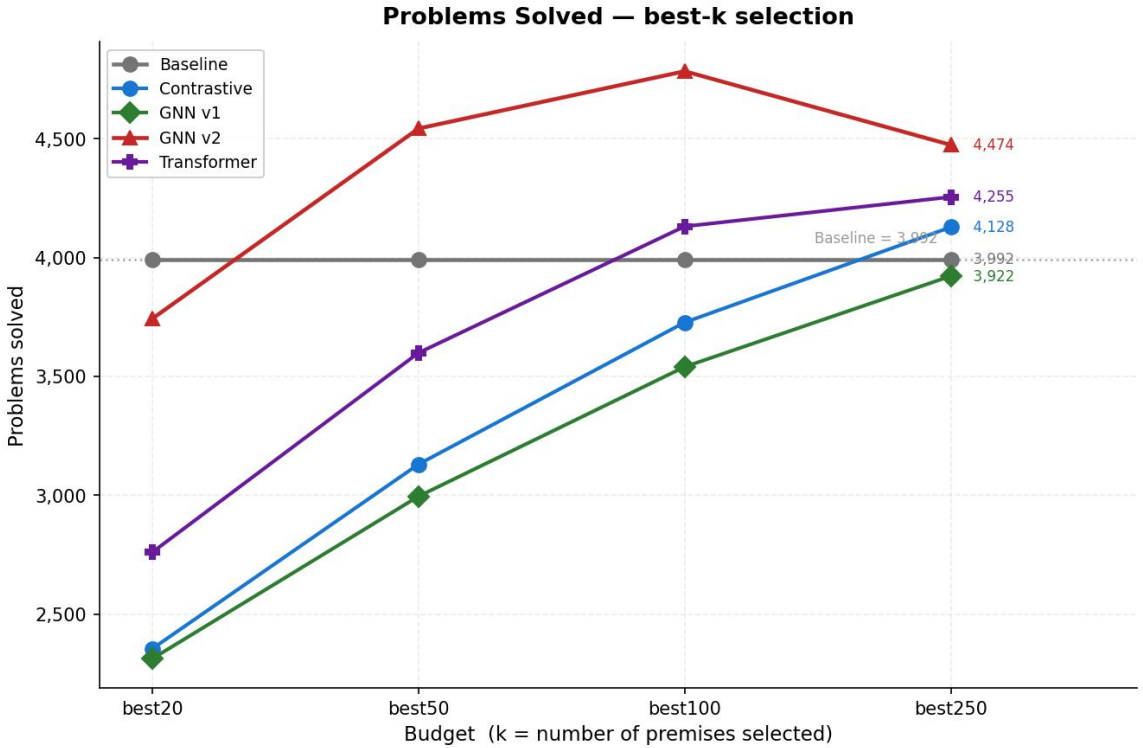
- Baseline (MePo)
- GNN_V1
- GNN_V1_NoVocab

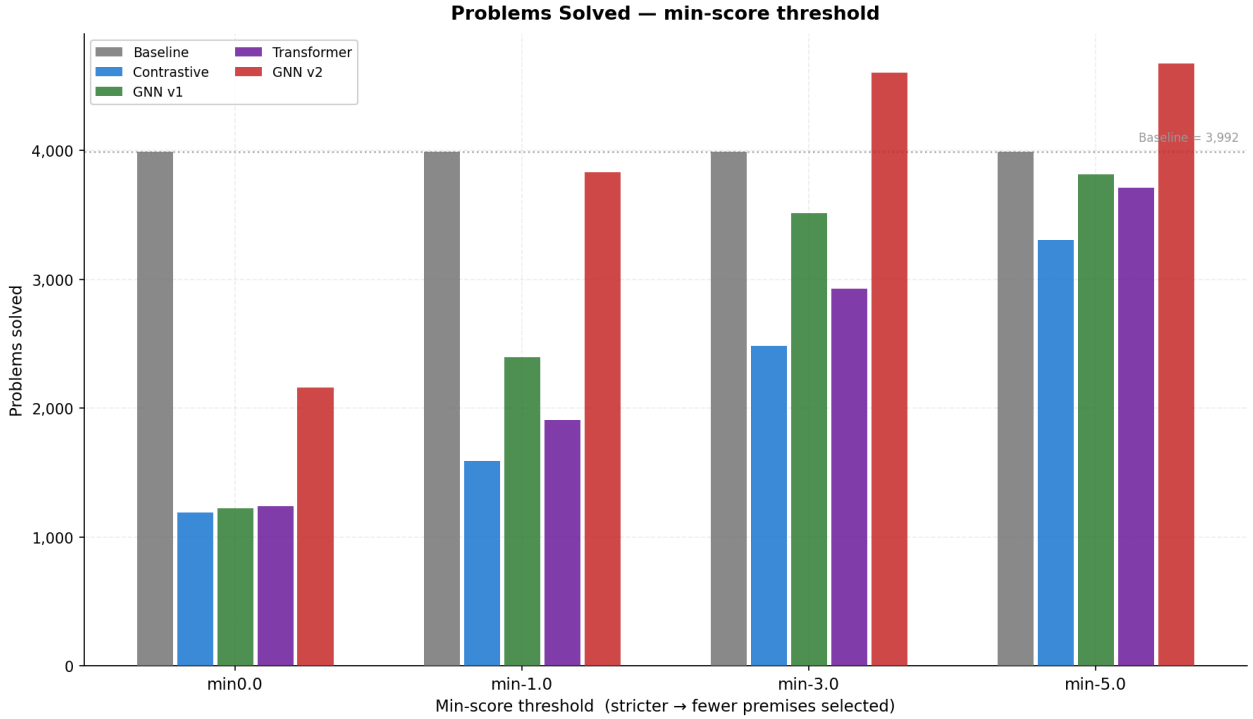
- GNN-V2
- GNN_V2_NoVocab

- Transformer
- TypedTransformer
- Transformer_NoPE

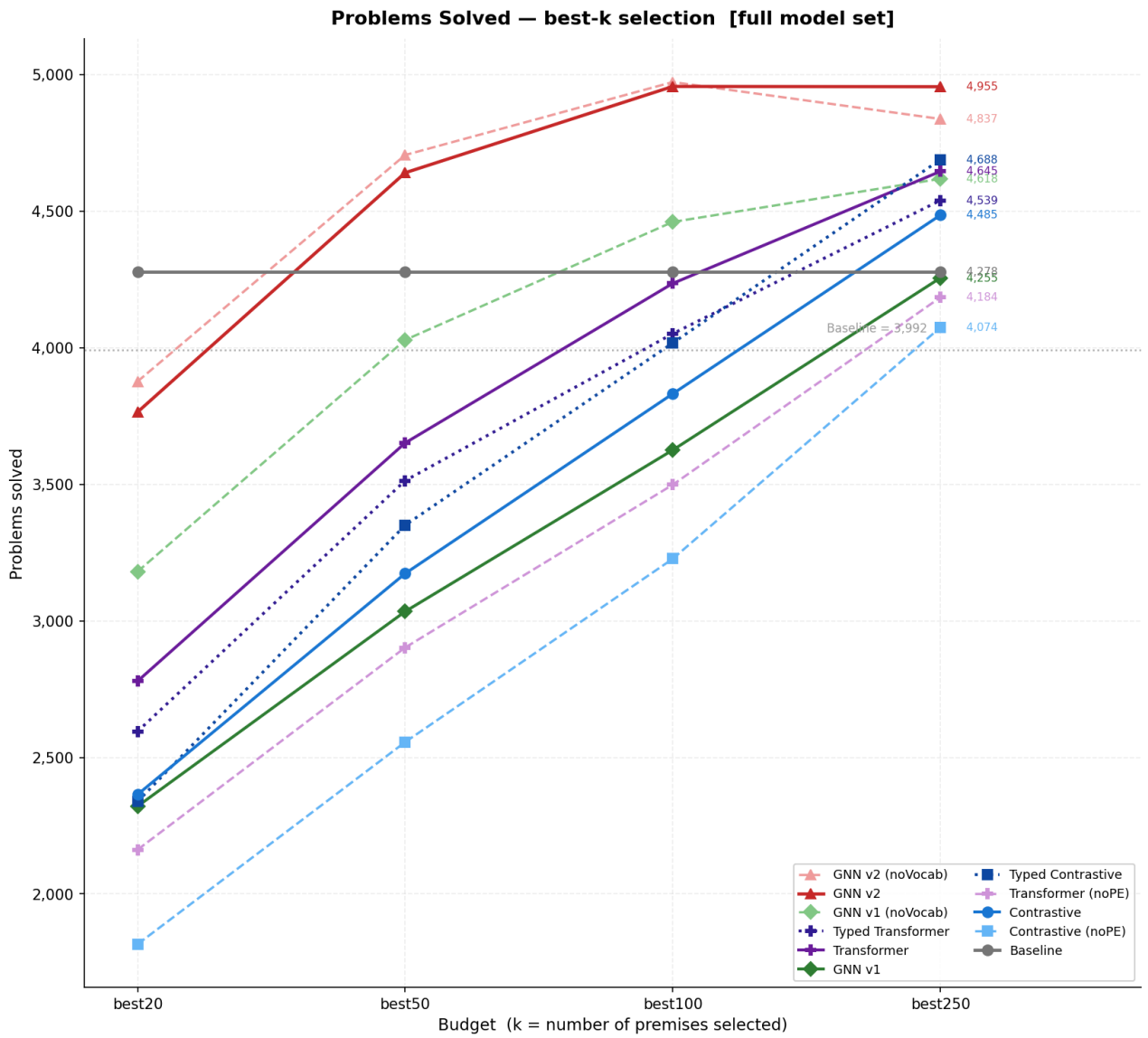
- Contrastive
- Typed Contrastive Transformer
- Contrastive Transformer NoPE

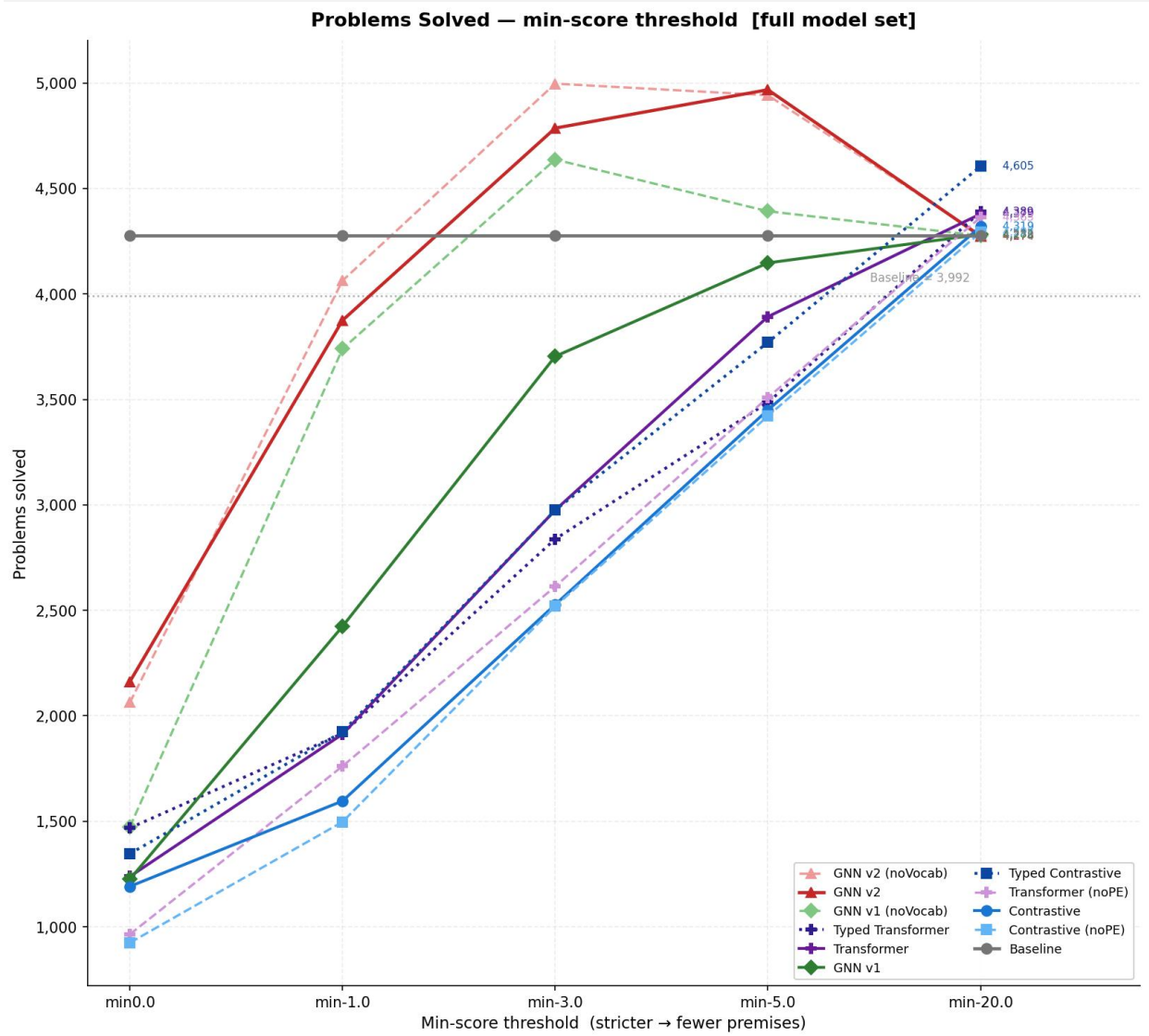
INITIAL RESULTS (T=1S)



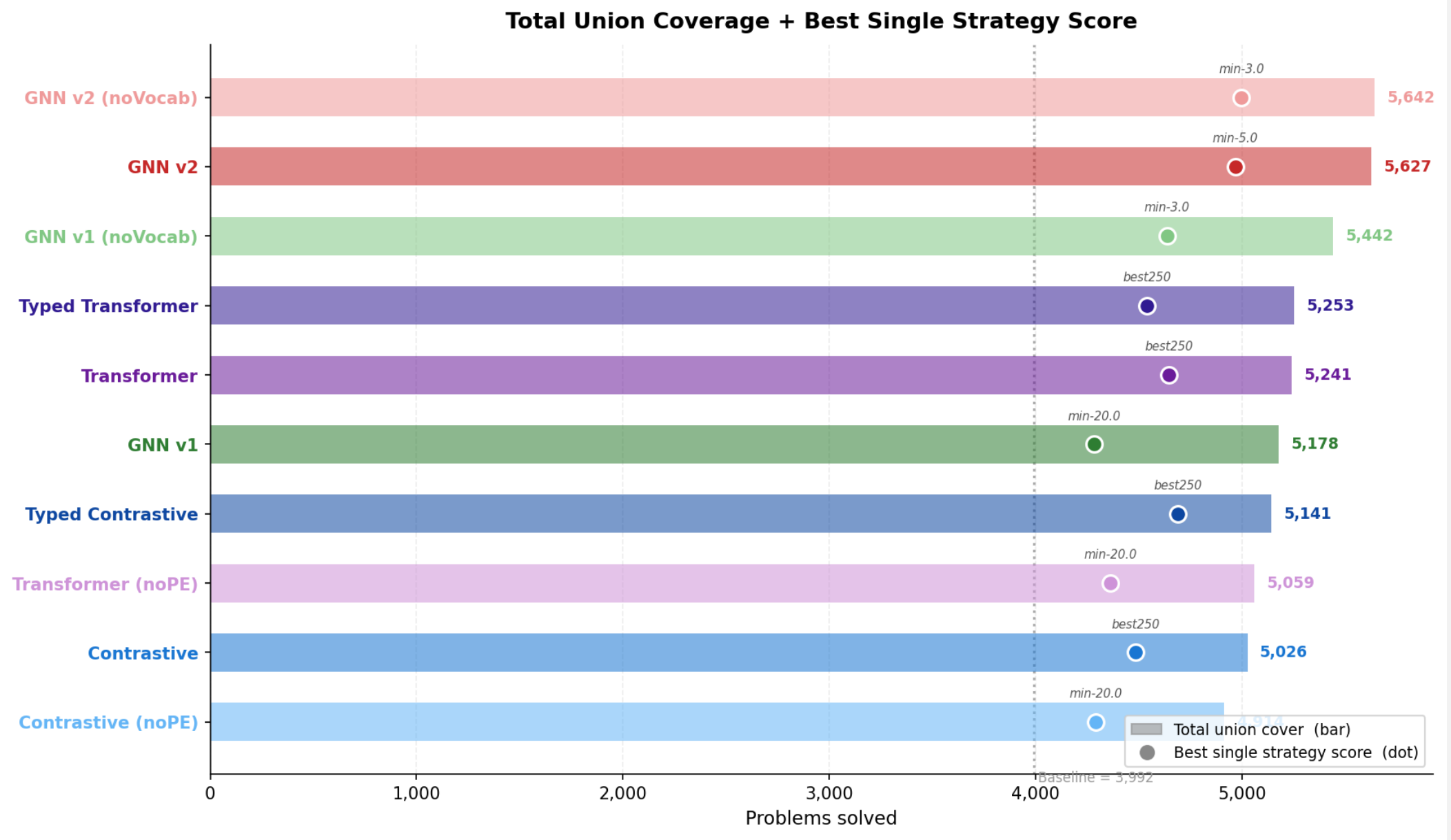


INITIAL RESULTS (T=10 S)

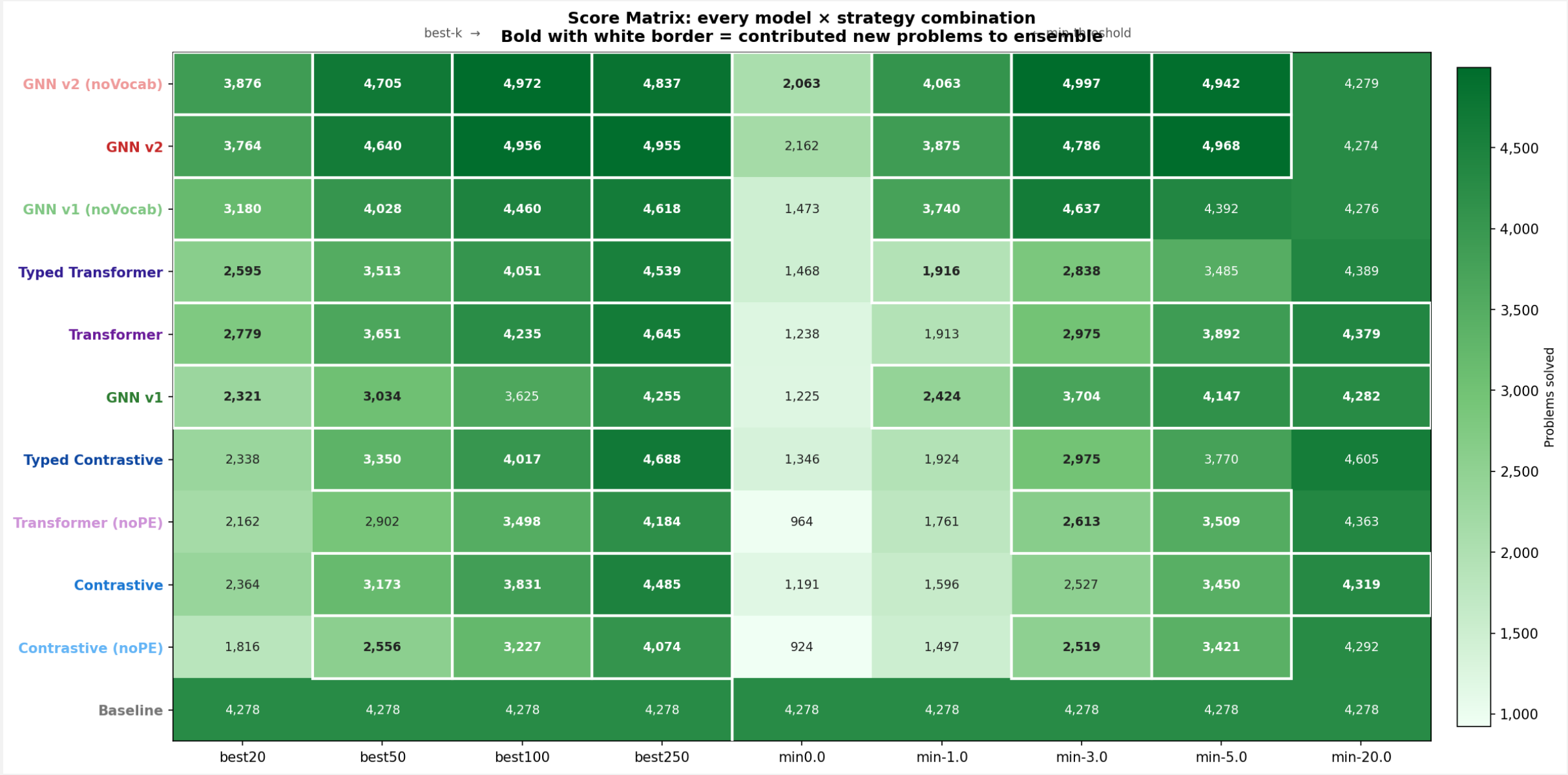




INITIAL RESULTS (T=10 S)



RESULTS (T=10 S, GREEDY COVER)



RESULTS (T=10 S, GREEDY COVER)

MODEL	VARIANT	SCORE	+NEW	+%	TOTAL	COVER%
gnn_v2_novocab	min-3.0	4997	4997	—	4997	55.65%
gnn_v2_novocab	best250	4837	333	6.66%	5330	59.35%
gnn_v2	best50	4640	158	2.96%	5488	61.11%
transformer	best250	4645	78	1.42%	5566	61.98%
gnn_v2	best100	4956	57	1.02%	5623	62.62%
gnn_v2_novocab	min-1.0	4063	45	0.80%	5668	63.12%
gnn_v1_novocab	best100	4460	32	0.56%	5700	63.47%
gnn_v1	min-20.0	4282	26	0.46%	5726	63.76%
transformer	best50	3651	21	0.37%	5747	64.00%
gnn_v2	min-5.0	4968	19	0.33%	5766	64.21%

CONCLUSION & FUTURE WORK

Premise Selection is a Promising direction towards solving more HOL problems faster.

Our models are also promising but they could be better.