Exploring Representation of Horn Clauses using GNNs

Chencheng Liang[1], Philipp Rümmer[1,2], Marc Brockschmidt[3]

[1]Uppsala University, Sweden
[2]University of Regensburg, Germany
[3]Microsoft Research, United Kingdom

August 11, PAAR 2022

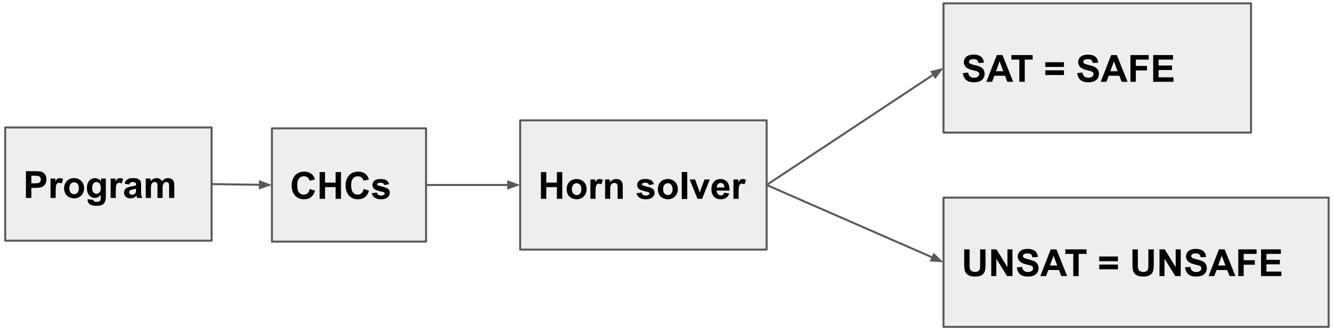
UPPSALA UNIVERSITET

Universität Regensburg

Microsoft

Solving program verification problems

- Program verification problems can be encoded to Constraint Horn clauses (CHCs) solving problems



Programs to CHCs

```
int x=-50;
int y;
 while (x<0){
   x=x+y;
   y=y+1;
 }
 assert(y>0);
```

(Si, et al 2020), Code2inv: A deep learning framework for program verification

Programs to CHCs

```
int x=-50;
int y;
while (x<0){
  x=x+y;
  y=y+1;
}
assert(y>0);
```

$$L(x, y) \text{ :- } true \land x = -50$$

Programs to CHCs

```
int x=-50;
int y;
 while (x<0){
   x=x+y;
   y=y+1;
 }
 assert(y>0);
```

$$L(x, y) \text{ :- } true \land x = -50$$

$$\boxed{L(x', y') \text{ :- } L(x, y) \land x < 0 \land y' = y + 1 \land x' = x + y}$$

Programs to CHCs

```
int x=-50;
int y;
while (x<0){
  x=x+y;
  y=y+1;
}
assert(y>0);
```

$$L(x, y) \text{ :- } true \land x = -50$$

$$L(x', y') \text{ :- } L(x, y) \land x < 0 \land y' = y + 1 \land x' = x + y$$

$$\boxed{false \text{ :- } L(x, y) \land x >= 0 \land y <= 0}$$

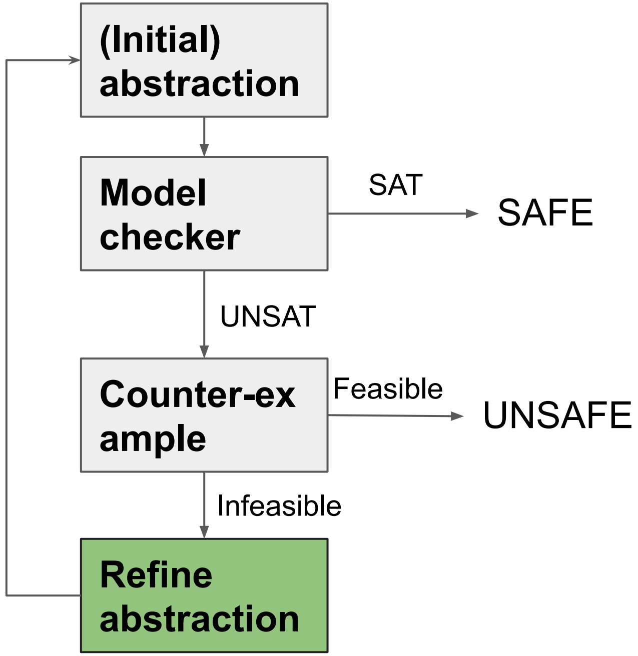
Programs to CHCs

```
int x=-50;
int y;
 while (x<0){
   x=x+y;
   y=y+1;
 }
 assert(y>0);
```

$$\forall x, y, x', y'$$
$$L(x, y) \text{ :- } \mathit{true} \wedge x = -50$$
$$L(x', y') \text{ :- } L(x, y) \wedge x < 0 \wedge y' = y + 1 \wedge x' = x + y$$
$$\mathit{false} \text{ :- } L(x, y) \wedge x >= 0 \wedge y <= 0$$

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

$$\forall x, y, x', y'$$
$$L(x, y) \text{ :- } true \land x = -50$$
$$L(x', y') \text{ :- } L(x, y) \land x < 0 \land y' = y + 1 \land x' = x + y$$
$$false \text{ :- } L(x, y) \land x >= 0 \land y <= 0$$

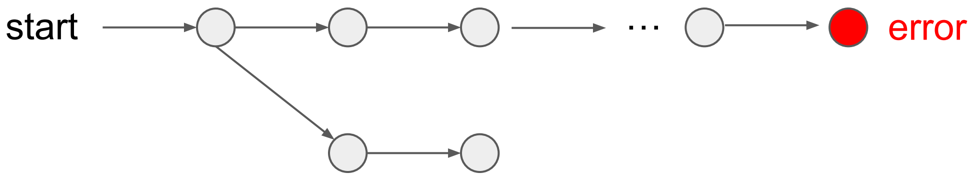


Framework overview

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

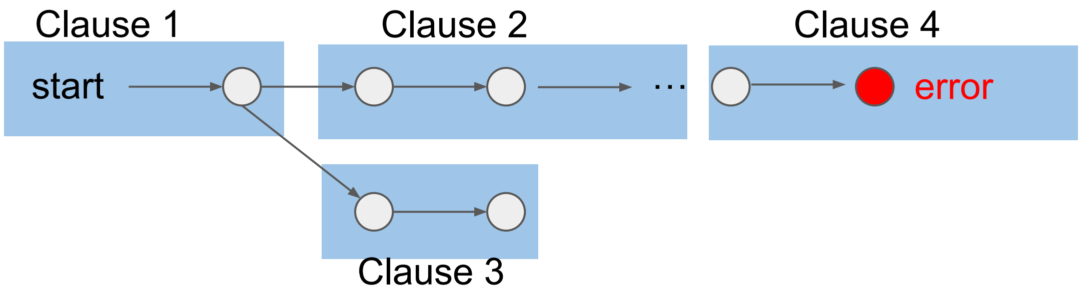
Five proxy tasks from simple to difficult

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



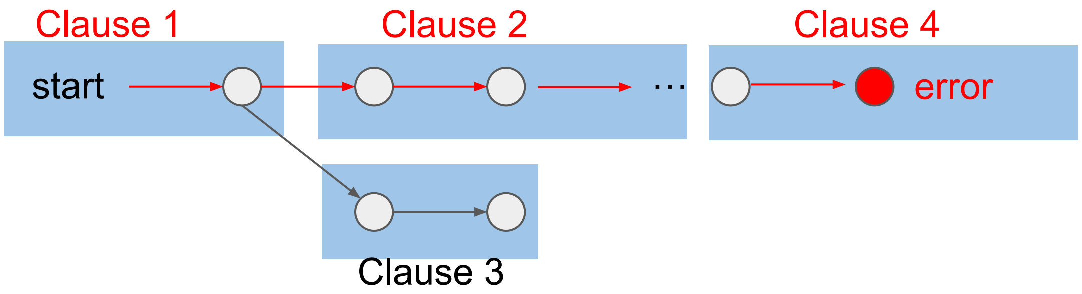
Five proxy tasks from simple to difficult

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

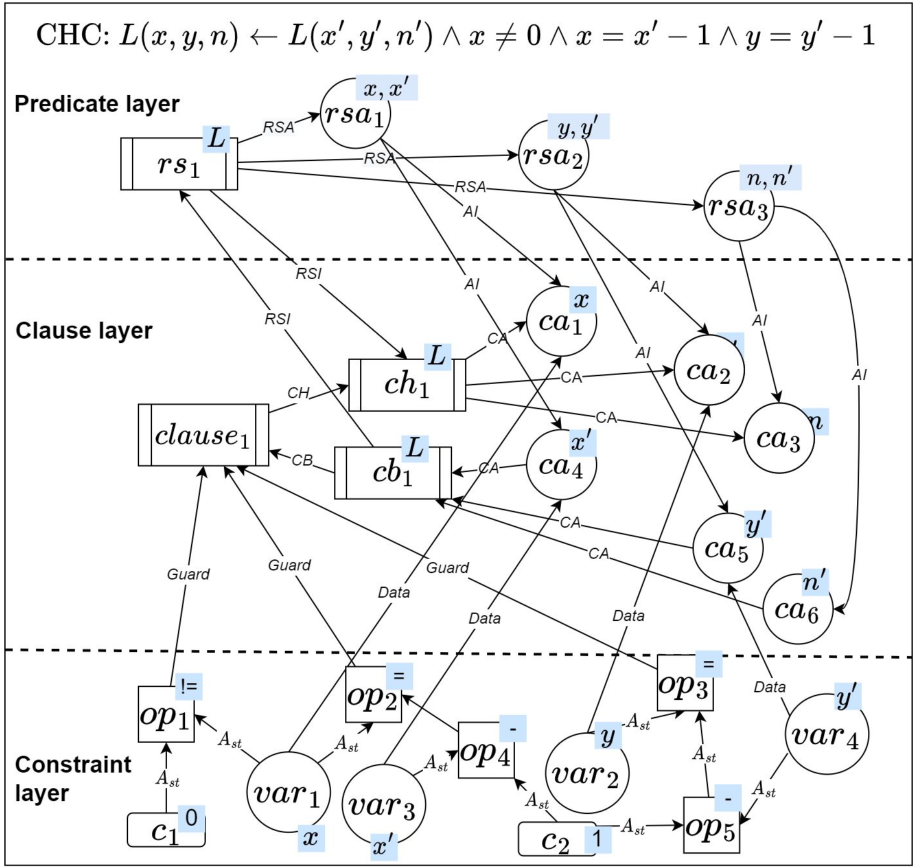
1. Predict if an graph node is an argument of relation symbol
2. Predict the number of occurrence of the relation symbols in all clauses
3. Predict if a relation symbol is in a cycle
4. Predict the existence of argument bound
5. Predict the clause membership in all/some minimal unsat cores



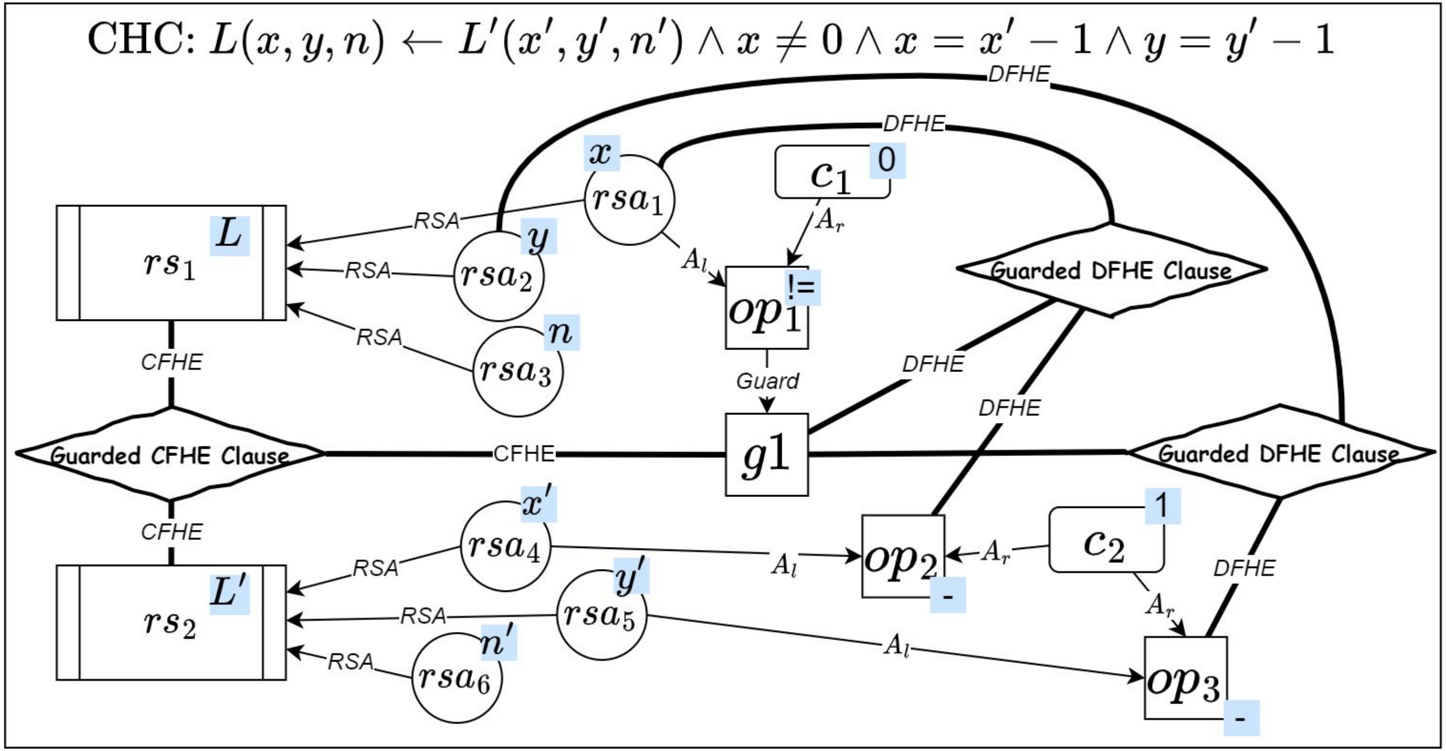
Framework overview

- Two graph representations for CHCs

Constraint graph (CG)

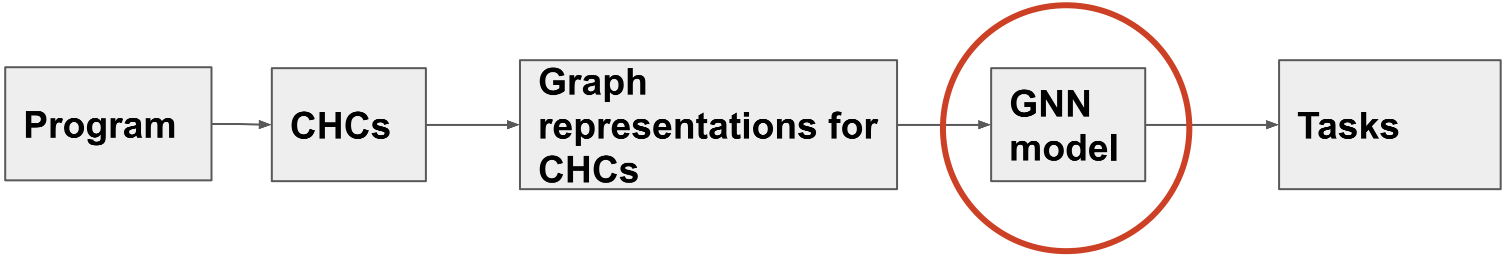


Control- and data-flow hypergraph (CDHG)

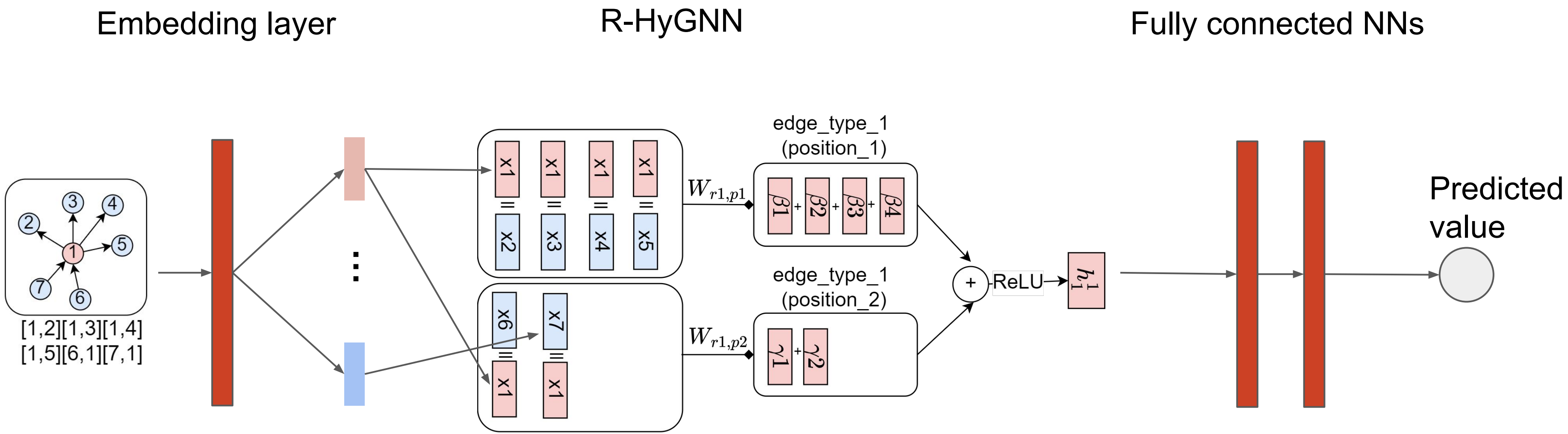


Framework overview

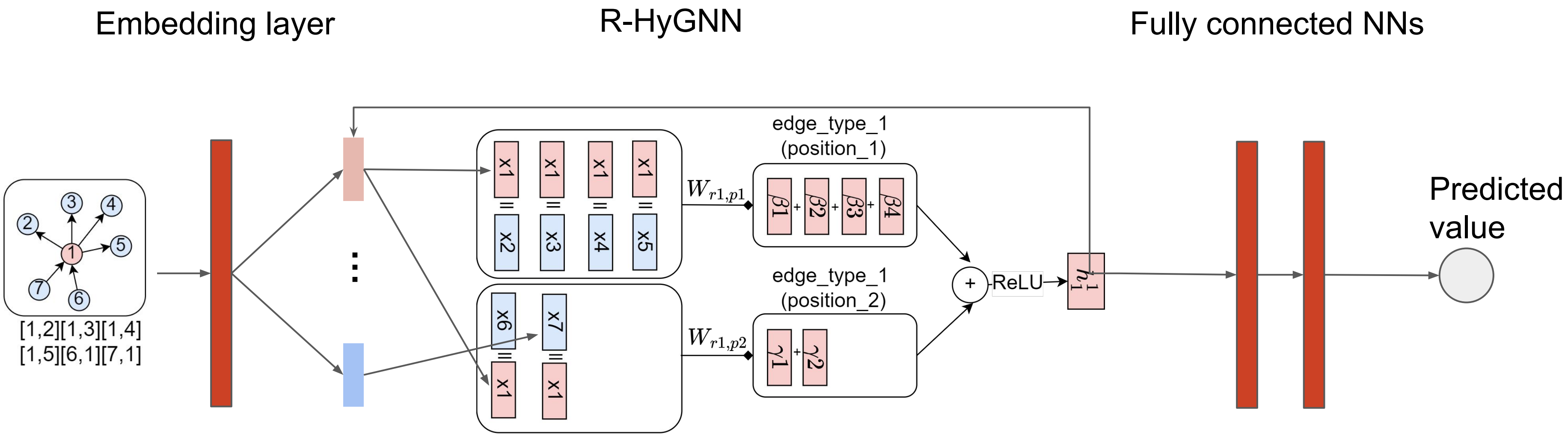
- Relational Hypergraph Neural Network (R-HyGNN)



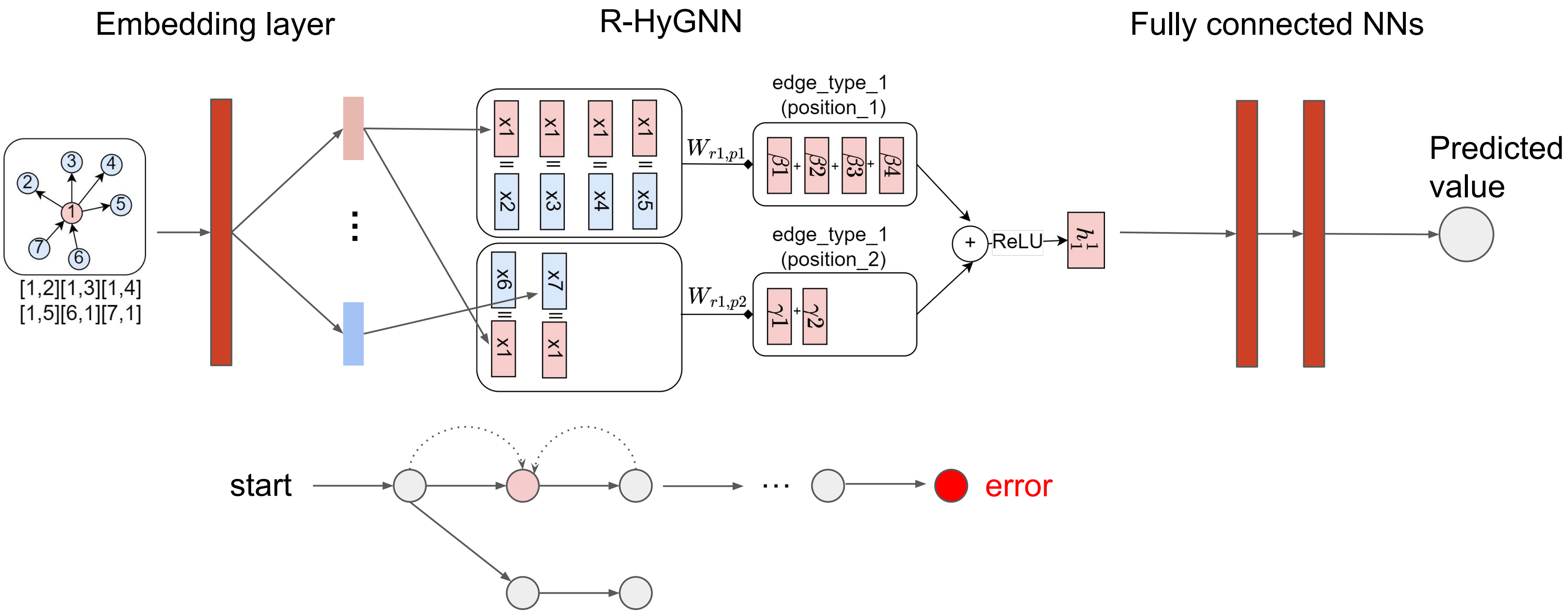
Training model



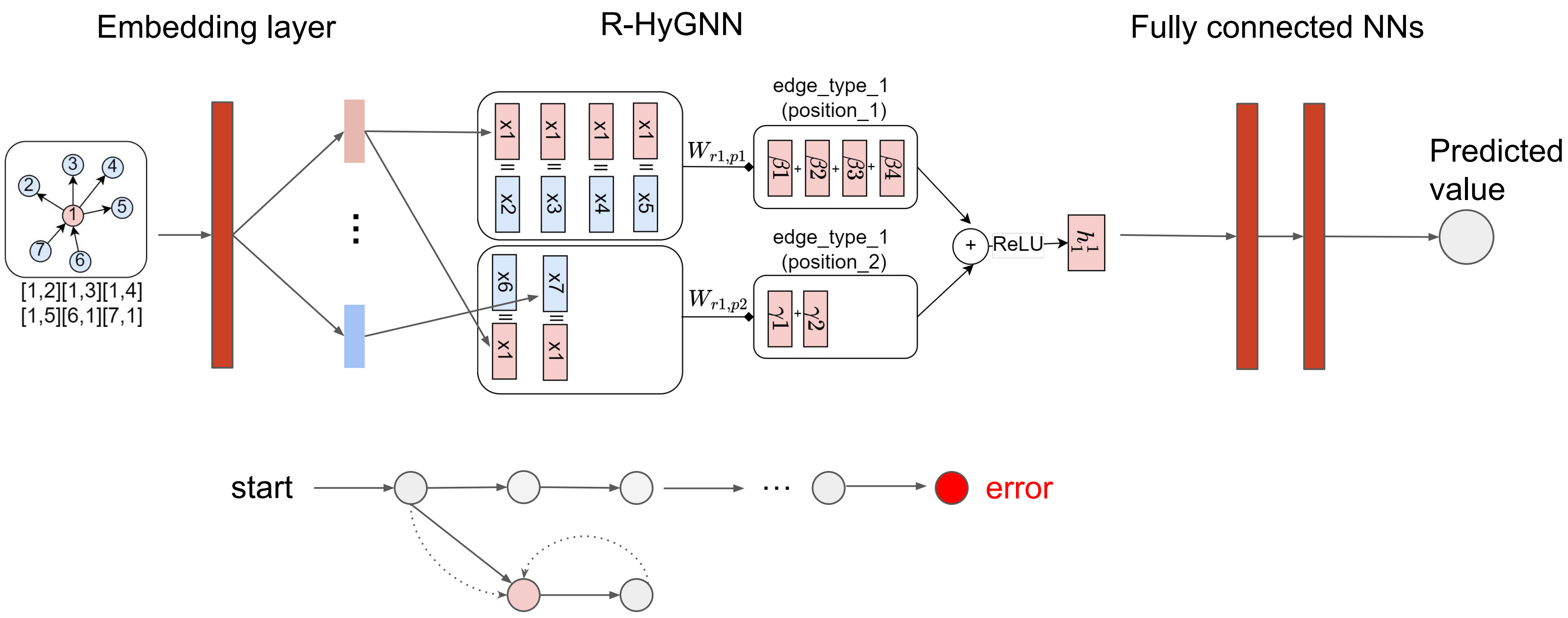
Training model



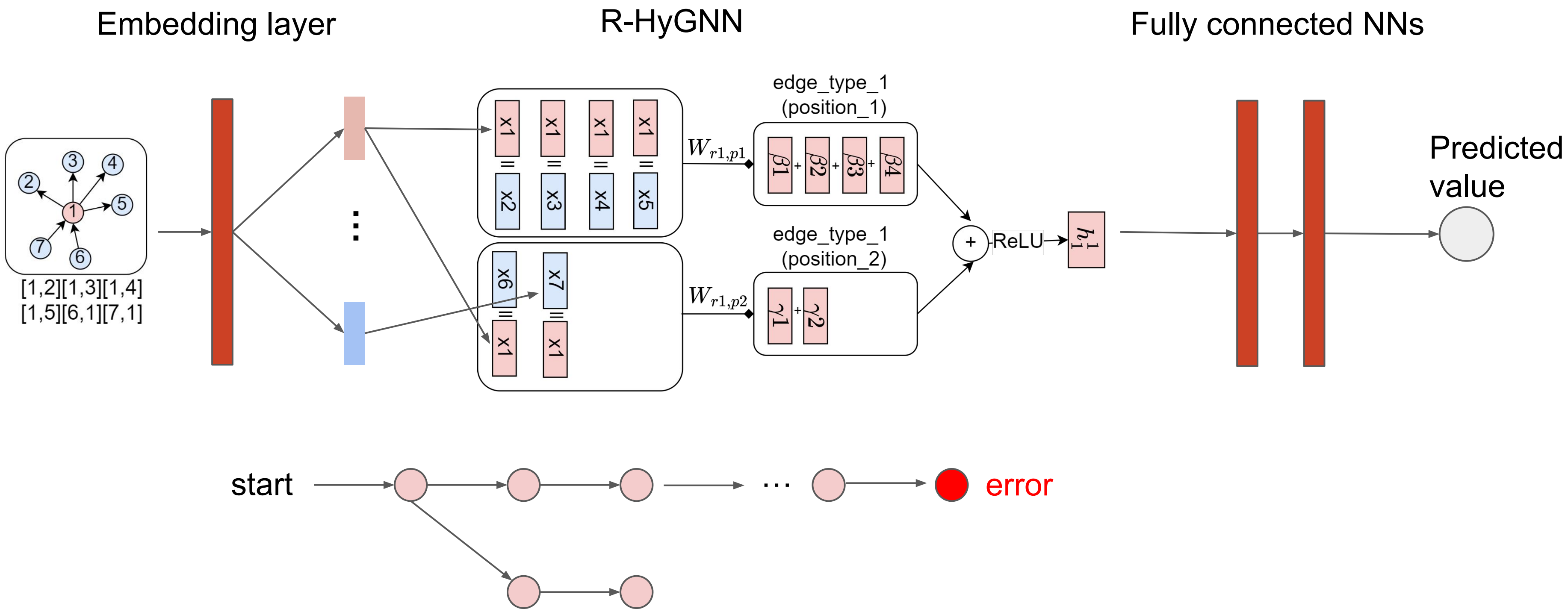
Training model



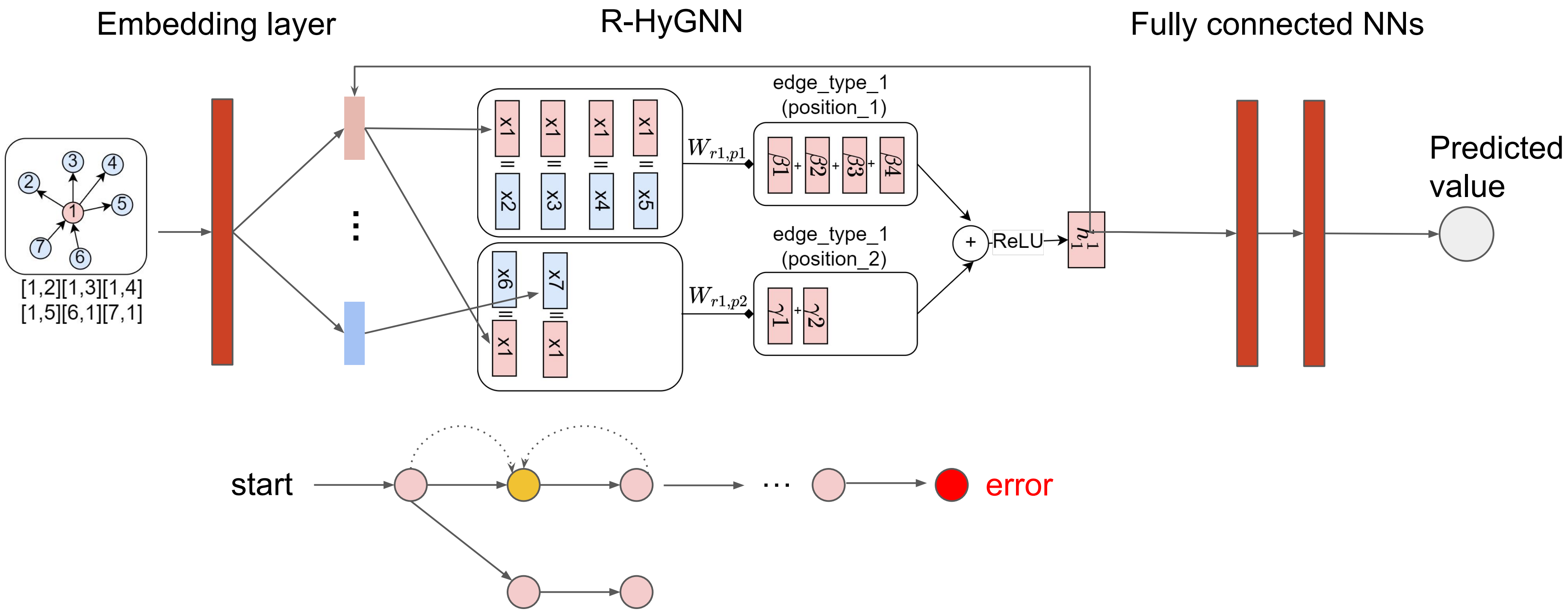
Training model



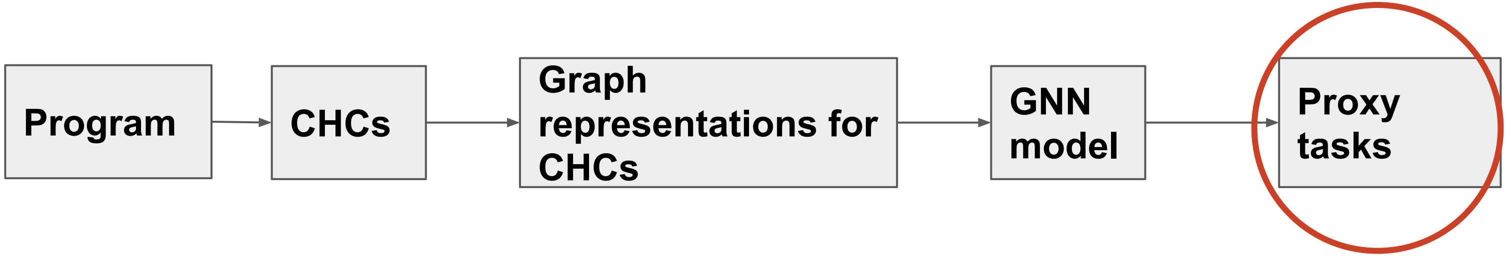
Training model



Training model



Framework overview



Evaluation on five proxy tasks

- Dominate distribution

			CG				CDHG			
Task	Files	T \ P	+	-	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	T \ P	+	-	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		

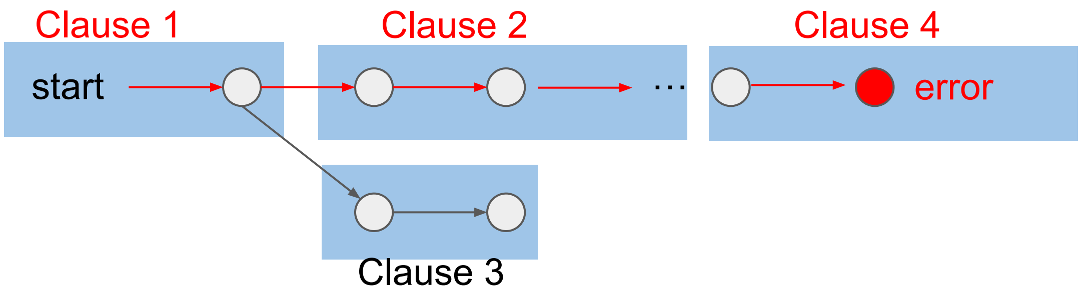
Evaluation on five proxy tasks (Task 5 results)

- 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	T \ P	+	-	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		

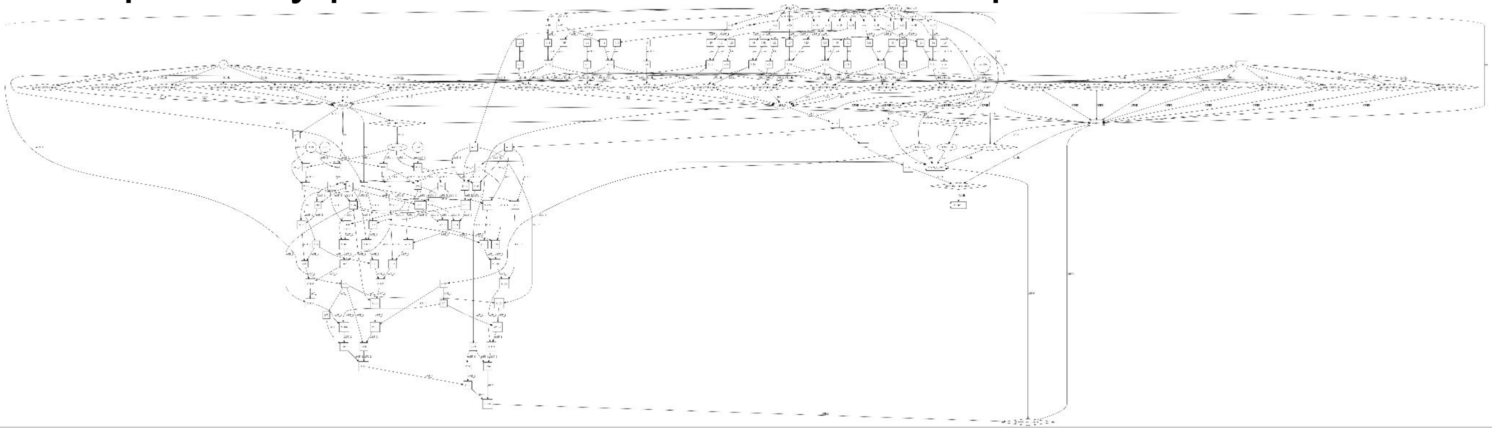
Predict the clause membership in minimal unsat cores

- 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.

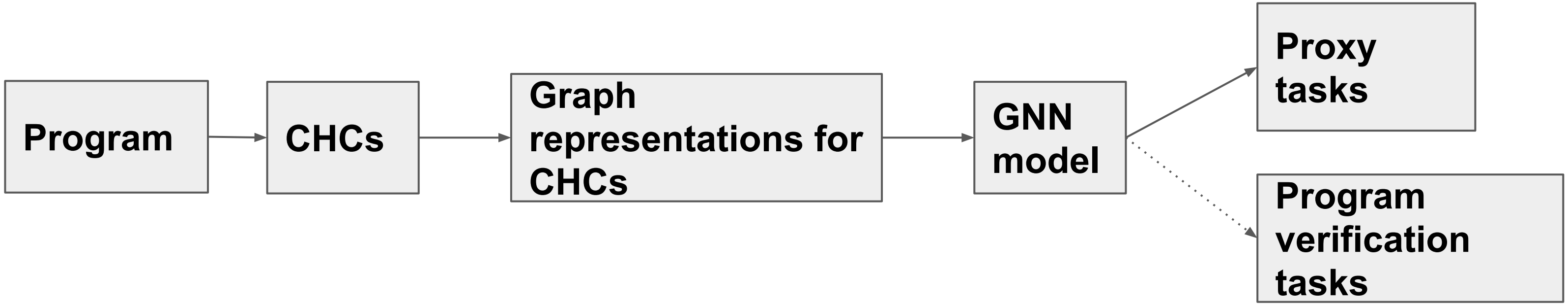


Predict the minimal unsatisfiable cores

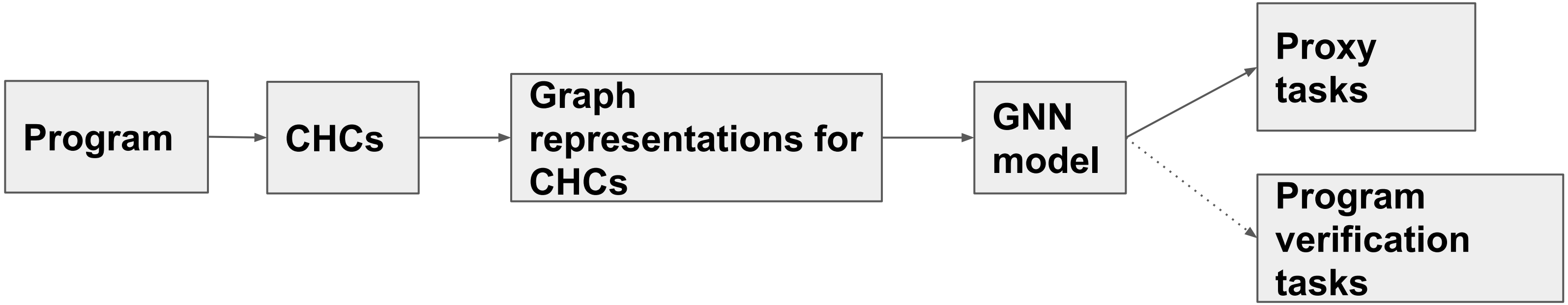
- Simple patterns
  - Clauses close to the assertions are likely in the minimal unsatisfiable cores
- Intricate patterns
  - In the verification problem that contain 290 clauses, the model can perfectly predict the clause membership in minimal unsat cores.



Summary & Future work



Summary & Future work



Thanks !

References

(J. Leroux, et al 2016) J. Leroux, P. Rümmer, P. Subotić, Guiding Craig interpolation with domain-specific abstractions, Acta Informatica 53 (2016) 387–424. URL: https://doi.org/10.1007/s00236-015-0236-z. doi:10.1007/s00236-015-0236-z.

(Y. Demyanova, et al 2017) Y. Demyanova, P. Rümmer, F. Zuleger, Systematic predicate abstraction using variable roles, in: C. Barrett, M. Davies, T. Kahsai (Eds.), NASA Formal Methods, Springer International Publishing, Cham, 2017, pp. 265–281.

(V. Tulsian, et al 2014) V. Tulsian, A. Kanade, R. Kumar, A. Lal, A. V. Nori, MUX: Algorithm selection for software model checkers, in: Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, Association for Computing Machinery, New York, NY, USA, 2014, p. 132–141. URL: https://doi.org/10.1145/2597073.2597080. doi:10.1145/2597073. 2597080.

(C. Richter, et al 2020) C. Richter, E. Hüllermeier, M.-C. Jakobs, H. Wehrheim, Algorithm selection for software validation based on graph kernels, Automated Software Engineering 27 (2020) 153–186.

(Si, et al 2020) X. Si, A. Naik, H. Dai, M. Naik, L. Song, Code2inv: A deep learning framework for program verification, in: Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part II, Springer-Verlag, Berlin, Heidelberg, 2020, p. 151–164. URL: https://doi.org/10.1007/978-3-030-53291-8_9. doi:10.1007/978-3-030-53291-8_9.

(A. Paliwal, et al 2019) A. Paliwal, S. M. Loos, M. N. Rabe, K. Bansal, C. Szegedy, Graph representations for higher-order logic and theorem proving, CoRR abs/1905.10006 (2019). URL: http://arxiv. org/abs/1905.10006. arXiv:1905.10006.

(D. Selsam, et al 2019) D. Selsam, N. Bjørner, Neurocore: Guiding high-performance SAT solvers with unsatcore predictions, CoRR abs/1903.04671 (2019). URL: http://arxiv.org/abs/1903.04671. arXiv:1903.04671.

(M. Schlichtkrull, et al 2017) M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, 2017. URL: https://arxiv.org/abs/1703.06103. doi:10.48550/ARXIV.1703.06103.

Related works

(T. Ben-Nun, et al 2018) T. Ben-Nun, A. S. Jakobovits, T. Hoefler, Neural code comprehension: A learnable representation of code semantics, CoRR abs/1806.07336 (2018). URL: http://arxiv.org/abs/1806.07336. arXiv:1806.07336.

Benchmark

Table 7
The number of labeled graph representations extracted from a collection of CHC-COMP benchmark [20]. For each SMT-LIB file, the graph representations for Task 1,2,3,4 are extracted together using the timeout of 3 hours, and for task 5 is extracted using 20 minutes timeout. Here, T. denotes Task.

	SMT-LIB files		CG			CDHGs		
	Total	Unsat	T. 1-4	T. 5 (a)	T. 5 (b)	T. 1-4	T. 5 (a)	T. 5 (b)
Linear LIA	8705	1659	2337	881	857	3029	880	883
Non-linear LIA	8425	3601	3376	1141	1138	4343	1497	1500
Aligned	17130	5260	5602	1927	1914	5602	1927	1914

CDHG node definition

Table 5

Node types for the CDHG. Note that Tables 2 and 5 use some same node types because they represent the same elements in the CHC. Some abstract nodes, such as *initial* and *guard*, do not have concrete symbol names since they do not directly associate with any element in the CHCs.

Node types X^{CDHG}	Explanation	Elements in CHCs	Shape
relation symbol (rs)	Relation symbols in head or body	L	
initial	Initial state	$\emptyset$	
false	*false* state	*false*	
relation symbol argument (rsa)	Arguments of the relation symbols	x, y	
variables (var)	Free variables	n	
operator (op)	Operators over a theory	=, +	
constant (c)	Constant over a theory	0, 1, $true$	
guard (g)	Guard for *CFHE* and *DFHE*	$\emptyset$	

CDHG edge definition

Table 6
Edge types for the CDHG. $rs, rsa, var, op, c, etc.$ are node types from Table 5.

Edge type R^{CDHG}	Edge arity	Definition	Explanation
Control Flow Hyperedge (*CFHE*)	Ternary	$(rs_1, rs_2 \mid false, g, CFHE)$	Connects the rs node in body and head, and abstract *guard* node
Data Flow Hyperedge (*DFHE*)	Ternary	$(a, op \mid c \mid var, g, DFHE)$	Connects the root node of right-hand and left-hand side of data flow sub-formulas, and a abstract *guard* node
Guard (*Guard*)	Binary	$(op \mid c, g, Guard)$	Connects all roots of ASTs of control flow sub-formulas and the $guard$ node
Relation Symbol Argument (*RSA*)	Binary	(a, rs, RSA)	Connects rsa nodes and their rs node
AST_left (A_l)	Binary	$(op, op \mid var \mid c, A_l)$	Connects left-hand side element of a binary operator or an element from a unary operator
AST_right (A_r)	Binary	$(op, op \mid var \mid c, A_r)$	It connects right-hand side element of a binary operator

CG node definition

Table 2
Node types for constraint graph. The shape corresponds to the shape of nodes in Figure 2 and is only used for visualizing the example.

Node types X^{CG}	Layer	Explanation	Elements in CHCs	Shape
relation symbol (rs)	Predicate layer	Relation symbols in head or body	L	▭
false	Predicate layer	*false* state	*false*	▭
relation symbol argument (rsa)	Predicate layer	Arguments of the relation symbols	x, y	○
clause (cla)	Clause layer	Represent clause as a abstract node	$\emptyset$	▭
clause head (ch)	Clause layer	Relation symbol in head	L	▭
clause body (cb)	Clause layer	Relation symbol in body	L	▭
clause argument (ca)	Clause layer	Arguments of relation symbol in head and body	x, y	○
variable (var)	Constraint layer	Free variables	n	○
operator (op)	Constraint layer	Operators over a theory	=, -	□
constant (c)	Constraint layer	Constants over a theory	0, 1, *true*	▭

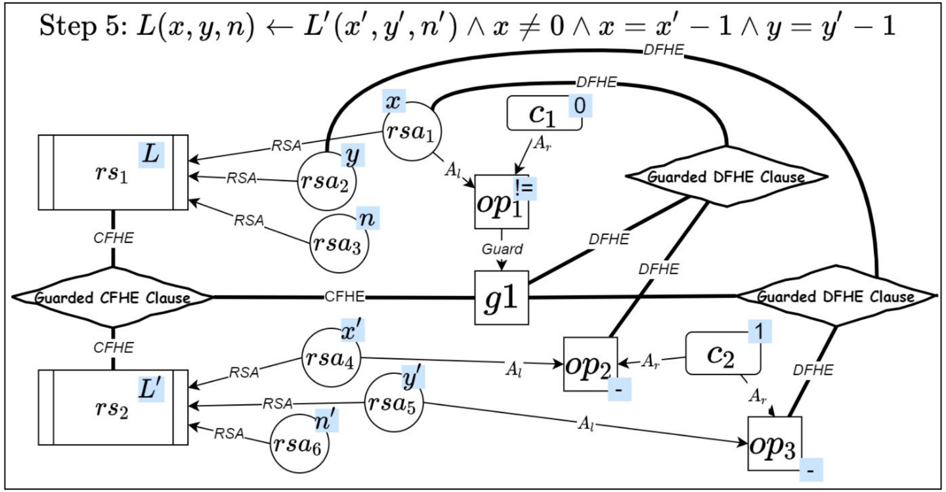
CG edge definition

Table 3
Edge types for constraint graph. Here, rs, rsa, cla, etc. are node types described in Table 2.Here, "|" means "or". For example, $c \mid op \mid var$ means this node could be node with type c, op, or var.

Edge type R^{CG}	Layer	Definition	Explanation
Relation Symbol Argument (*RSA*)	Predicate layer	(rs, rsa, RSA)	Connects relation symbols and their arguments
Relation Symbol Instance (*RSI*)	Between predicate and clause layer	$(rs, ch, \mathit{RSA}) \mid (cb, rs, \mathit{RSA})$	Connects relation symbols with their head and body
Argument Instance (*AI*)	Between predicate and clause layer	(pa, ca, AI)	Connects relation symbols and their arguments
Clause Head (*CH*)	Clause layer	(cla, ch, CH)	Connect *clause* node to its head
Clause Body (*CB*)	Clause layer	(cb, cla, CB)	Connect the *clause* node to its body
Clause Argument (*CA*)	Clause layer	$(ca, cb, \mathit{CA}) \mid (ch, ca, \mathit{CA})$	Connect ch or cb nodes with corresponding ca nodes
Guard (*GUARD*)	Between clause and constraint layer	$(c \mid op, cla, \mathit{GUARD})$	Connect the root node of the AST to corresponding *clause* node
Data (*DATA*)	Between clause and constraint layer	(var, ca, DATA)	Connect ca nodes to corresponding var nodes AST
AST sub-term (A_{st})	Constraint layer	$(c \mid var \mid op, op, A_{st})$	Connect nodes within AST

Evaluation on five proxy tasks (Task 1 example)

- Task 1: Predict if a node type is relation symbol argument.
- Binary classification task.



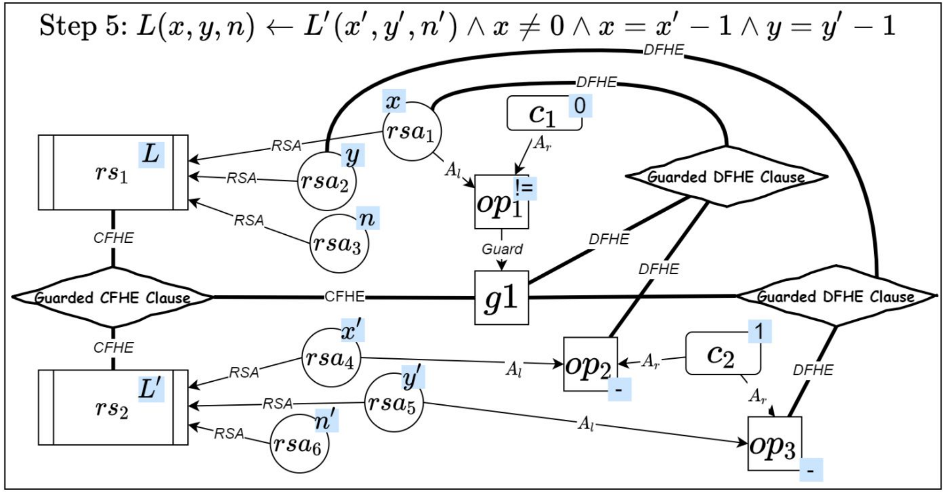
Evaluation on five proxy tasks (Task 1 results)

- Task 1: Predict if a node type is relation symbol argument.

			CG				CDHG			
Task	Files	T \ P	+	-	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		

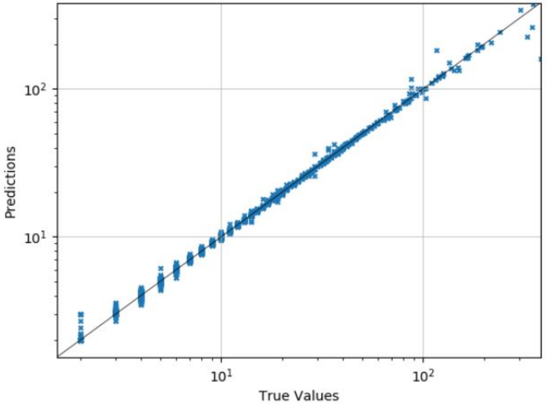
Evaluation on five proxy tasks (Task 2 example)

- Task 2: Predict how many times the relation symbols occur in all clauses.
- Regression task.

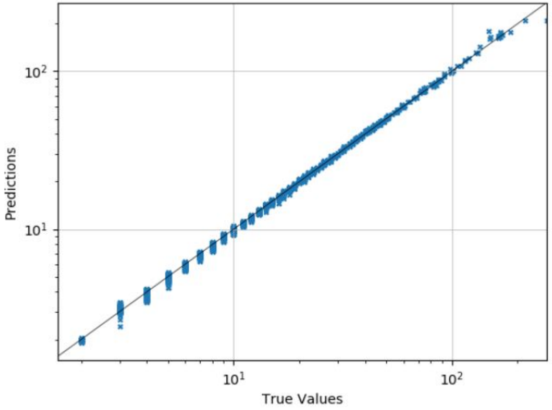


Evaluation on five proxy tasks (Task 2 results)

- Task 2: Predict how many times the relation symbols occur in all clauses.



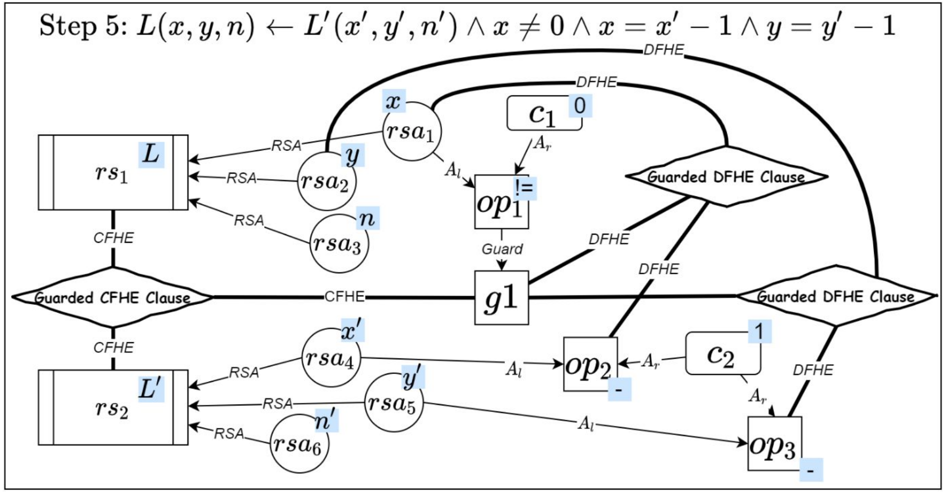
(a) Scatter plot for CDHG. The total rs node number is 16858. The mean square error is 4.22.



(b) Scatter plot for constraint graph. The total rs node number is 11131. The mean square error is 1.04.

Evaluation on five proxy tasks (Task 3 example)

- Task 3: Predict if relation symbol node is in a cycle.
- Binary Classification task.



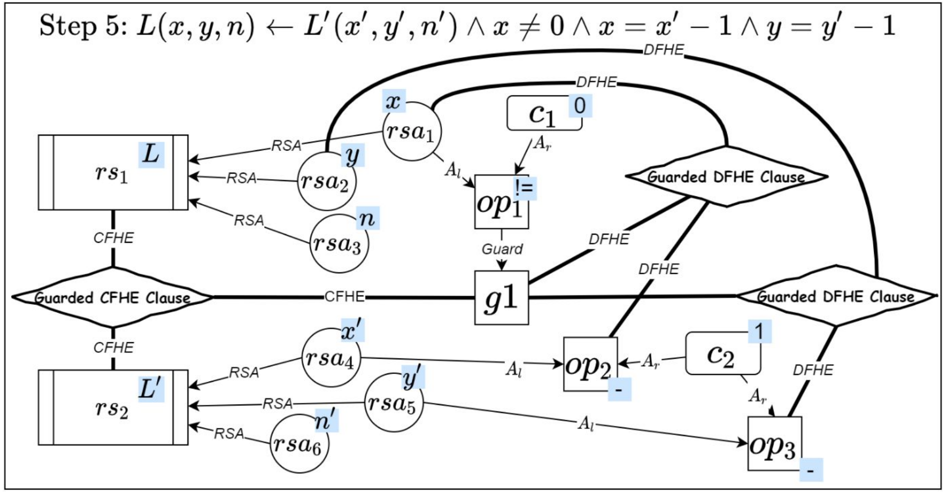
Evaluation on five proxy tasks (Task 3 results)

- Task 3: Predict if relation symbol node is in a cycle.

			CG				CDHG			
Task	Files	T \ P	+	-	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 4 example)

- Task 4: Predict lower and upper bound existence for relation symbol arguments.
- Binary classification task.



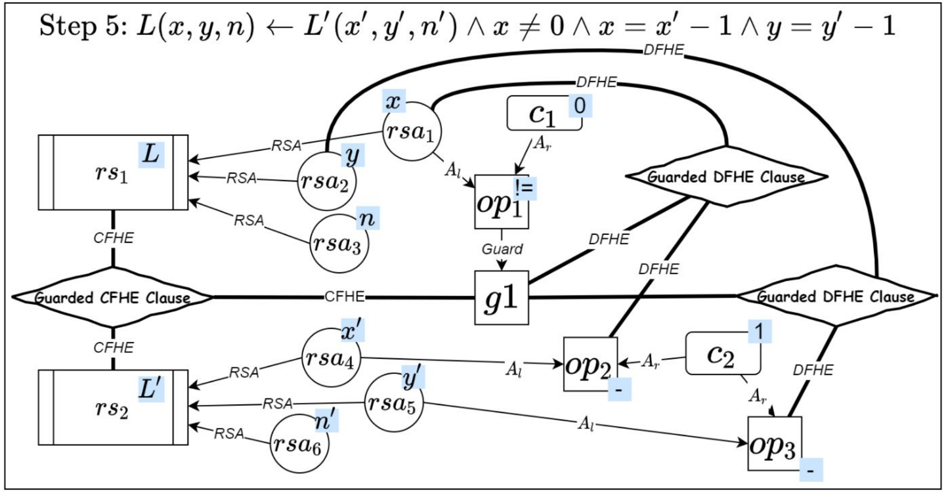
Evaluation on five proxy tasks (Task 4 results)

- Task 4: Predict lower and upper bound existence for relation symbol. arguments. (a) lower bound, (b) upper bound.

			CG				CDHG			
Task	Files	T \ P	+	-	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 example)

- Task 5: Predict if a clause occurs in the counter-examples.
- Binary classification.



Evaluation on five proxy tasks (Task 5 results)

- Task 5: Predict if a clause occurs in the counter-examples.
- (a) the intersection and (b) the union of the minimal unsatisfiable subsets of a clause set.

			CG				CDHG			
Task	Files	T \ P	+	-	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		

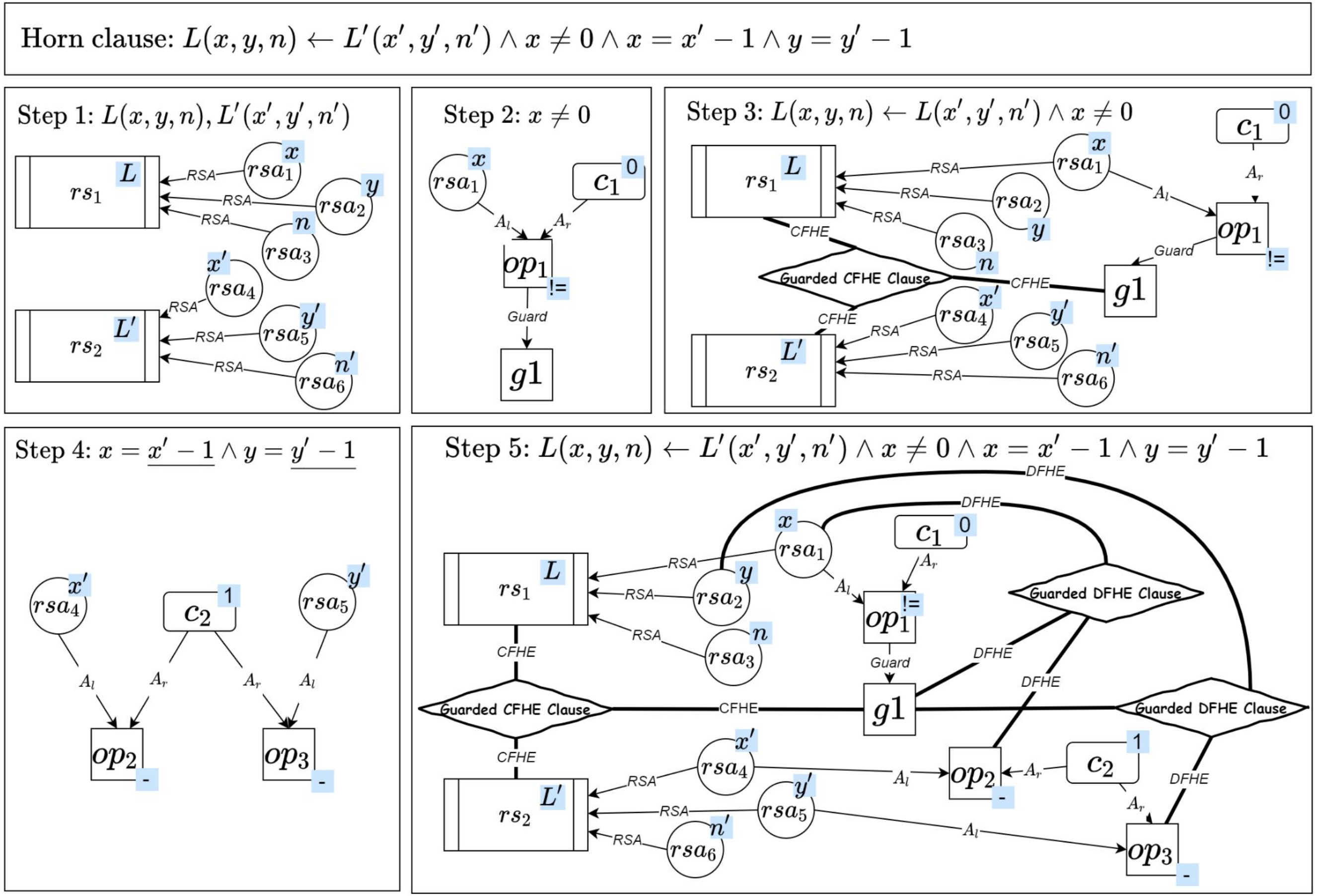
Motivation

1. Automatic program verification rely on domain-specific heuristics constructed from program features (J. Leroux, et al 2016, Y. Demyanova, et al 2017, V. Tulsian, et al 2014).
2. Deep learning, a popular automatic feature learning technique, has been used to extract program features in many applications (C. Richter, et al 2020, X. Si, et al 2020).
3. More and more studies represent code by graphs and extract program features by GNNs (A. Paliwal, et al 2019, D. Selsam, et al 2019).
4. Every study has unique graph representation and learning models.

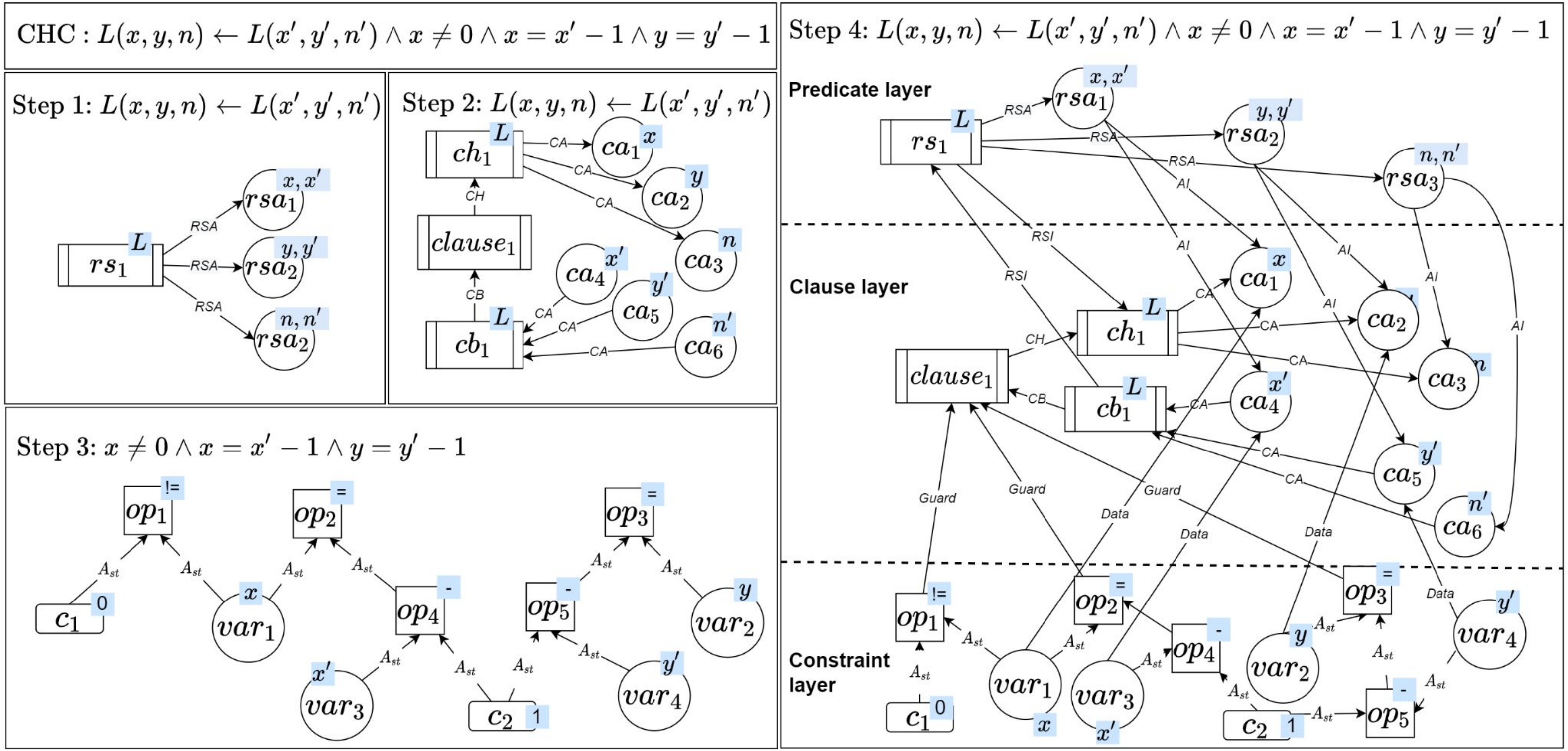
Challenges for learning program features to guide verification

1. The syntax of the source code varies depending on the programming languages, conventions, regulations, and even syntax sugar.
    - We use CHC as input.
2. Program feature extracting requires capturing intricate semantics from long-distance relational information based on re-occurring identifiers.
    - We introduce two graph encoding for CHCs.
3. Most of GNN models cannot deal with hypergraphs.
    - We introduce R-HyGNN, an extension of Relational Graph Convolutional Networks (M. Schlichtkrull, et al 2017).

Control- and data-flow hypergraph (CDHG)



Constraint graph (CG)

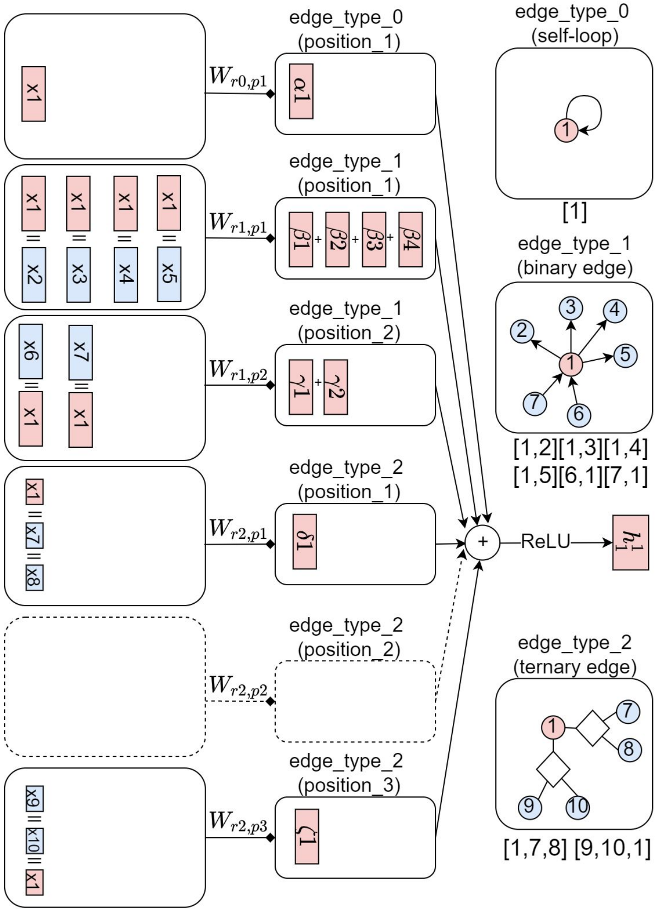


R-HyGNN

The updating rule for node representation in time step t :

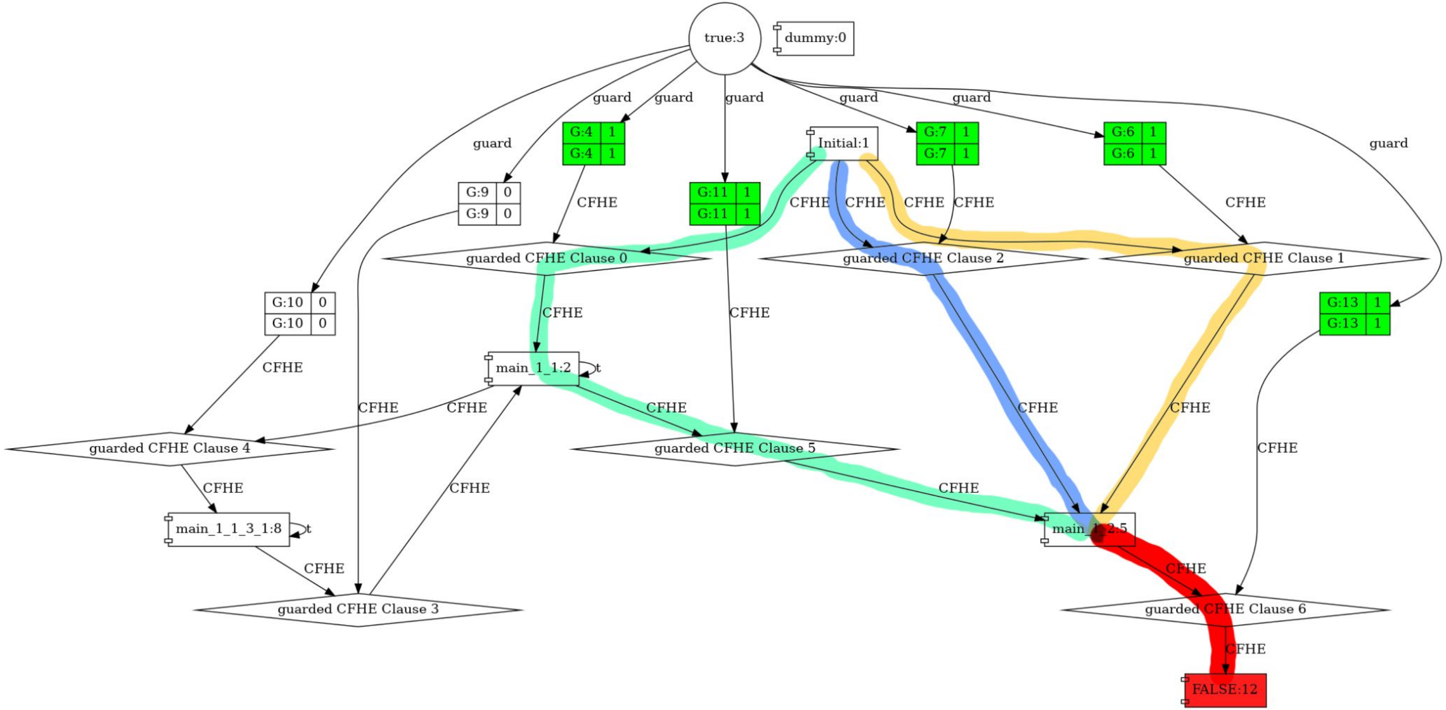
$$h_v^t = \mathrm{ReLU}\left(\sum_{r\in R}\sum_{p\in P_r}\sum_{e\in E_v^{r,p}} W_{r,p}^t \cdot \|[h_u^{t-1} \mid u\in e]\right),$$

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 pth 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.



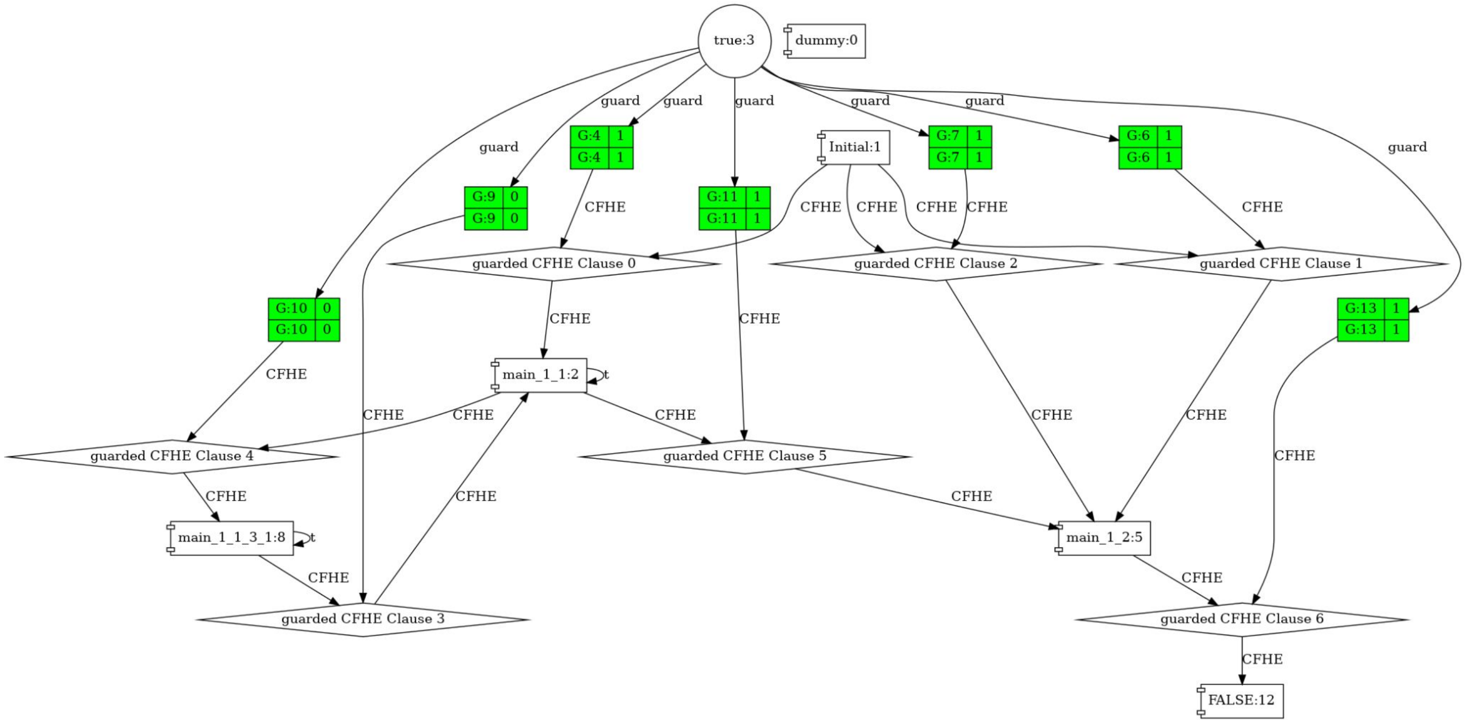
Predict the minimum unsatisfiable cores

- Three minimum unsatisfiable cores

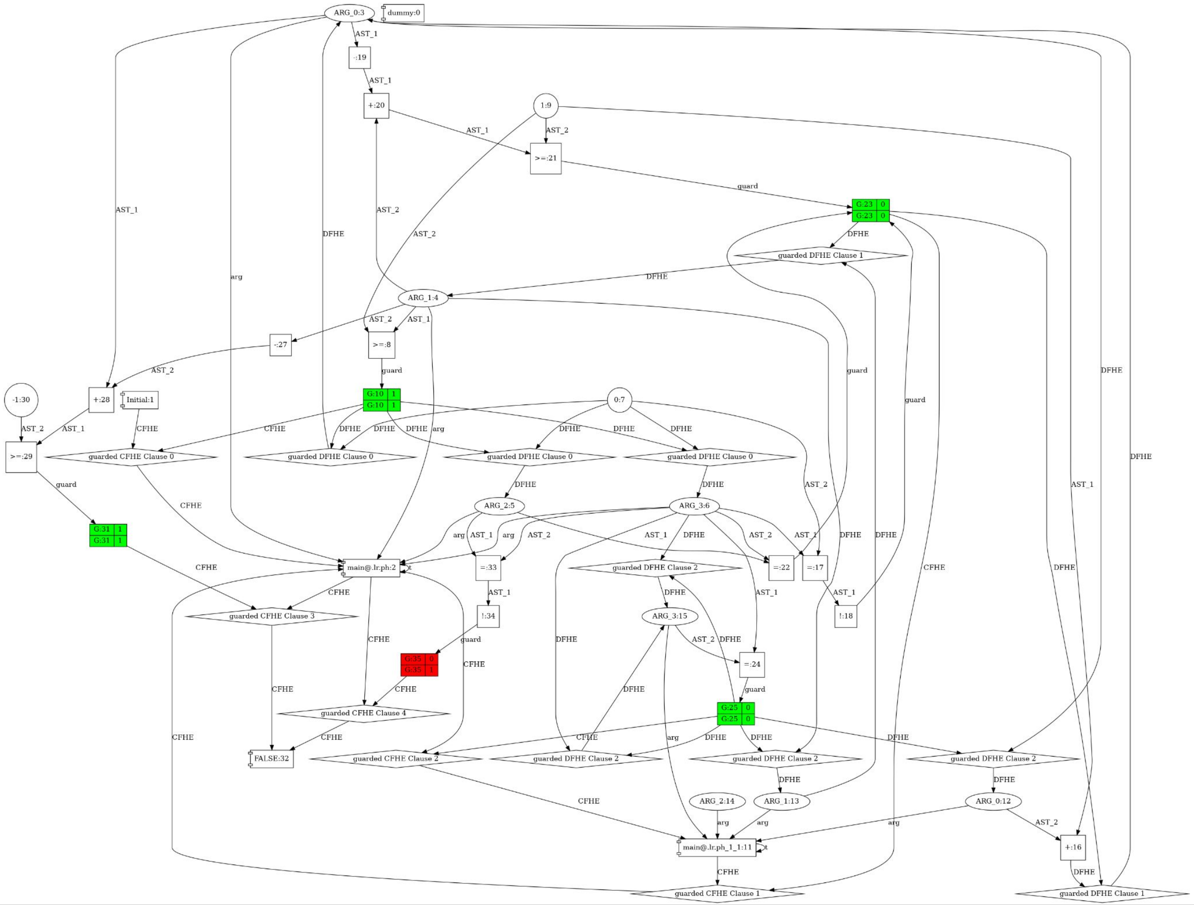


Predict the minimum unsatisfiable cores

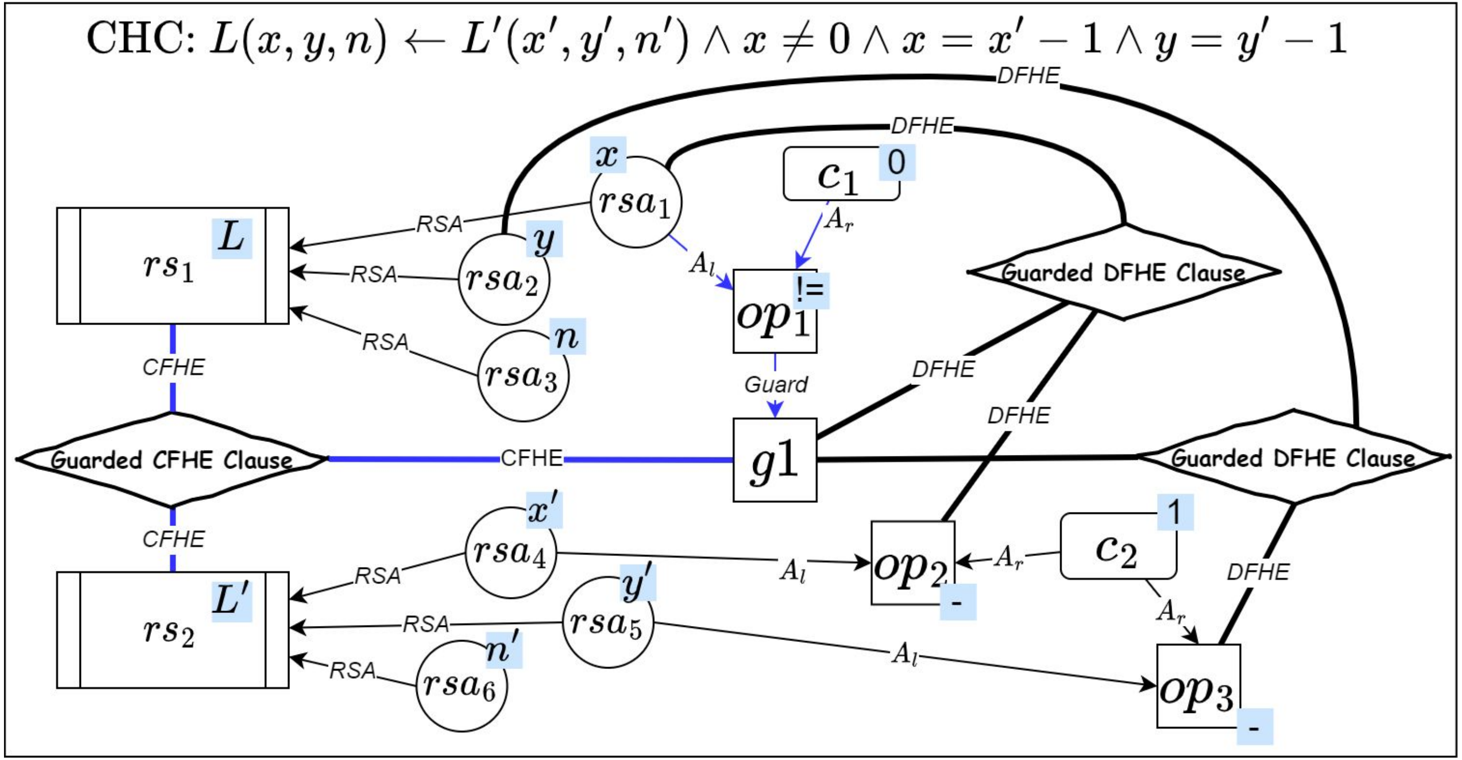
- Predict if a clause occurs in some minimum unsatisfiable cores



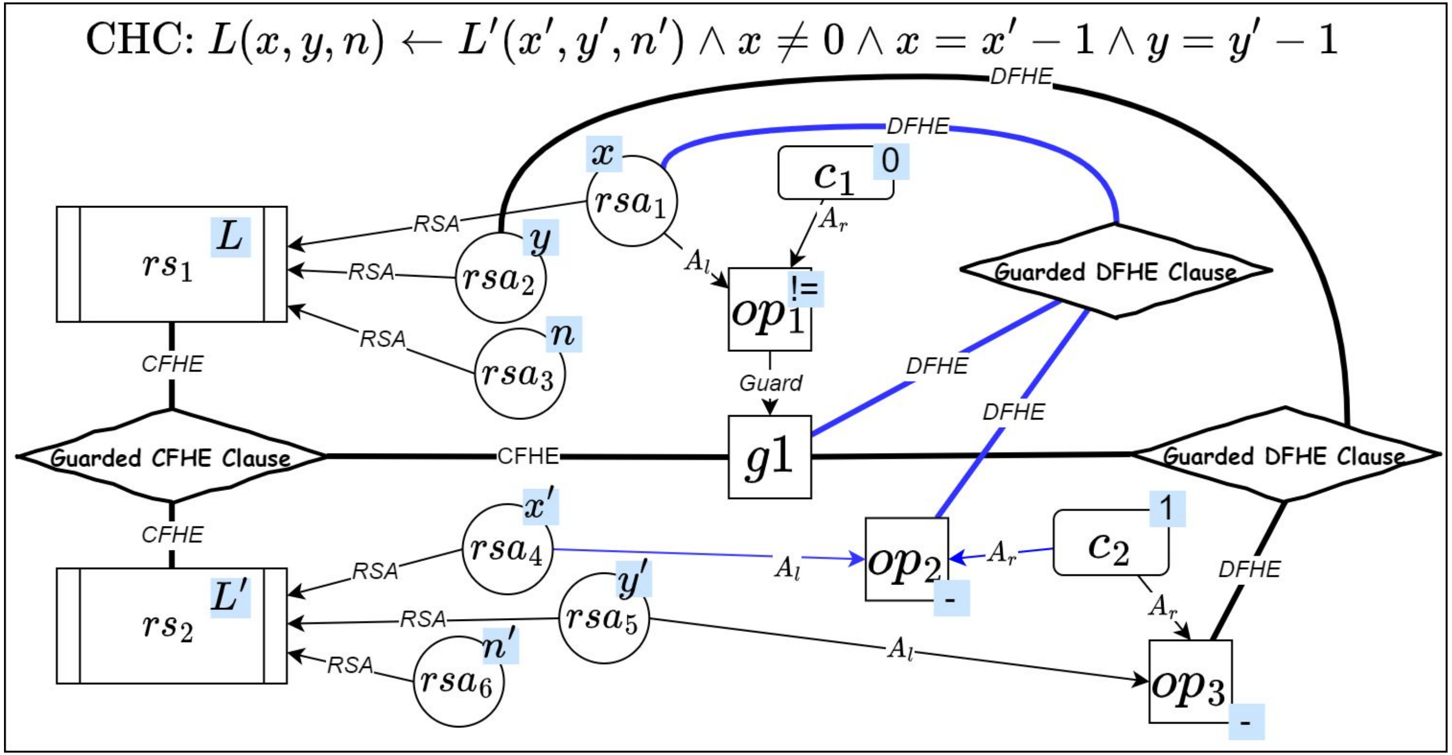
Technique details



Control- and data-flow hypergraph (CDHG)



Control- and data-flow hypergraph (CDHG)



Solving CHCs by model checking

$$
\begin{aligned}
&\forall x, y, x', y' \\
&L(x, y) \text{ :- } x = -50 \\
&L(x', y') \text{ :- } L(x, y) \wedge x < 0 \wedge y' = y + 1 \wedge x' = x + y \\
&false \text{ :- } x >= 0 \wedge y <= 0
\end{aligned}
$$

$$
\text{start} \xrightarrow[x = -50]{\boxed{y = 1,}} L(-50, 1) \xrightarrow{\substack{x < 0, \\ y' = y + 1, \\ x' = x + y}} L(-49, 2) \xrightarrow{\substack{x < 0, \\ y' = y + 1, \\ x' = x + y}} L(-47, 3) \cdots \xrightarrow{\substack{x < 0, \\ y' = y + 1, \\ x' = x + y}} L(6, 11) \xrightarrow{\substack{x >= 0, \\ y <= 0}} false
$$

Solving CHCs by model checking

$$\forall x, y, x', y'$$
$$L(x, y) \text{ :- } x = -50$$
$$L(x', y') \text{ :- } L(x, y) \wedge x < 0 \wedge y' = y + 1 \wedge x' = x + y$$
$$false \text{ :- } x >= 0 \wedge y <= 0$$

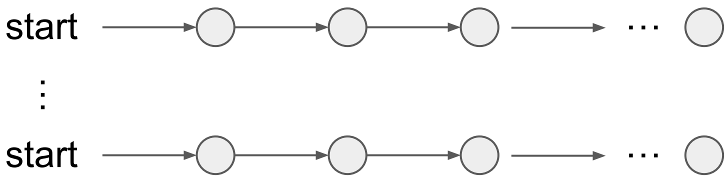
$$\text{start} \xrightarrow{\substack{y = 1, \\ x = -50}} L(-50, 1) \xrightarrow{\substack{x < 0, \\ y' = y + 1, \\ x' = x + y}} L(-49, 2) \xrightarrow{\substack{x < 0, \\ y' = y + 1, \\ x' = x + y}} L(-47, 3) \cdots \xrightarrow{\substack{x < 0, \\ y' = y + 1, \\ x' = x + y}} L(6, 11) \xrightarrow{\substack{x >= 0, \\ y <= 0}} false$$

$$\text{start} \xrightarrow{\substack{\boxed{y = 2,} \\ x = -50}} L(-50, 2) \xrightarrow{\substack{x < 0, \\ y' = y + 1, \\ x' = x + y}} L(-48, 3) \xrightarrow{\substack{x < 0, \\ y' = y + 1, \\ x' = x + y}} L(-45, 4) \cdots \xrightarrow{\substack{x < 0, \\ y' = y + 1, \\ x' = x + y}} L(4, 11) \xrightarrow{\substack{x >= 0, \\ y <= 0}} false$$

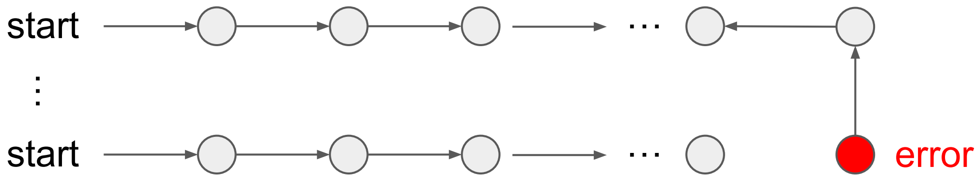
Solving CHCs by model checking

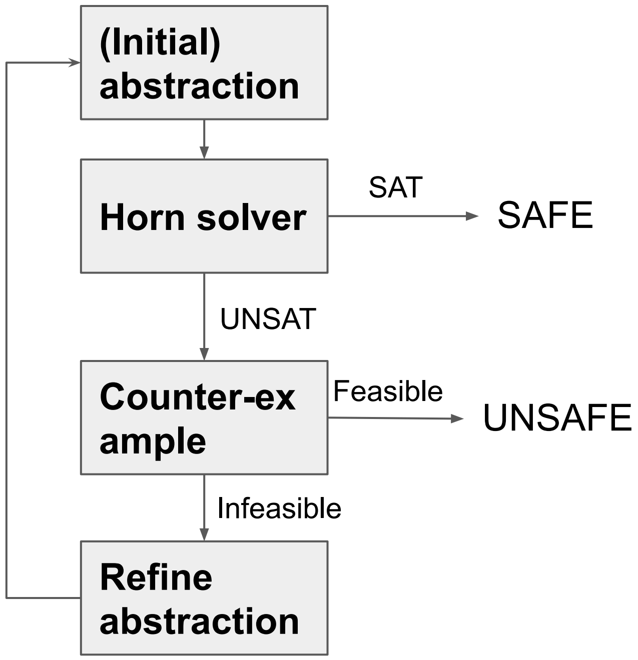
$$\text{start} \xrightarrow{\substack{y\,=\,1,\\ x\,=\,-50}} L(-50,1) \xrightarrow{\substack{x\,<\,0,\\ y'\,=\,y+1,\\ x'\,=\,x+y}} L(-49,2) \xrightarrow{\substack{x\,<\,0,\\ y'\,=\,y+1,\\ x'\,=\,x+y}} L(-47,3) \cdots \xrightarrow{\substack{x\,<\,0,\\ y'\,=\,y+1,\\ x'\,=\,x+y}} L(6,11) \xrightarrow{\substack{x\,>=\,0,\\ y\,<=\,0}} \mathit{false}$$

$$\text{start} \xrightarrow{\substack{\boxed{y\,=\,2,}\\ x\,=\,-50}} L(-50,2) \xrightarrow{\substack{x\,<\,0,\\ y'\,=\,y+1,\\ x'\,=\,x+y}} L(-48,3) \xrightarrow{\substack{x\,<\,0,\\ y'\,=\,y+1,\\ x'\,=\,x+y}} L(-45,4) \cdots \xrightarrow{\substack{x\,<\,0,\\ y'\,=\,y+1,\\ x'\,=\,x+y}} L(4,11) \xrightarrow{\substack{x\,>=\,0,\\ y\,<=\,0}} \mathit{false}$$

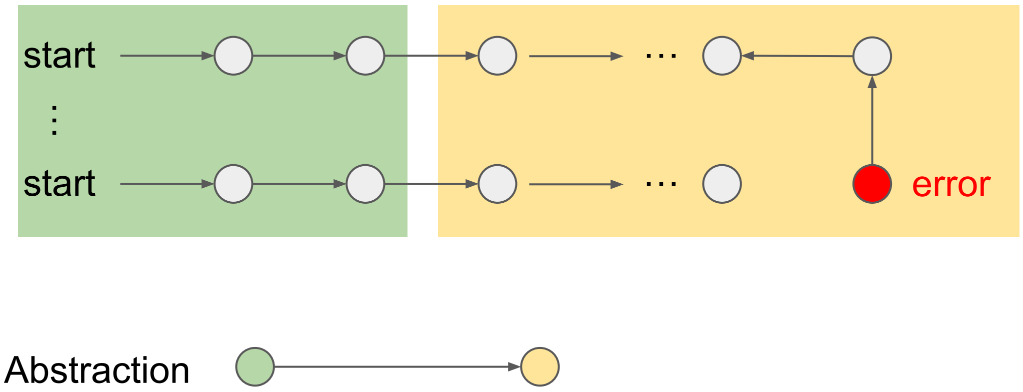


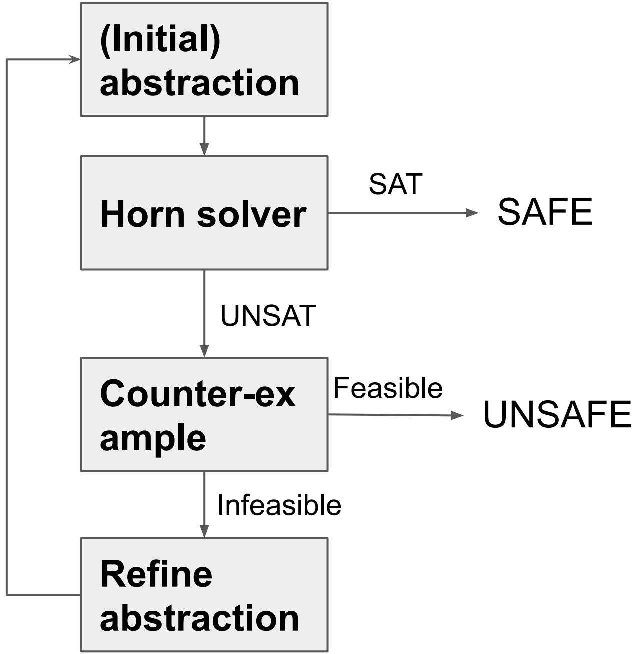
Solving CHCs by counter-example guided abstraction refinement (CEGAR) based model checking



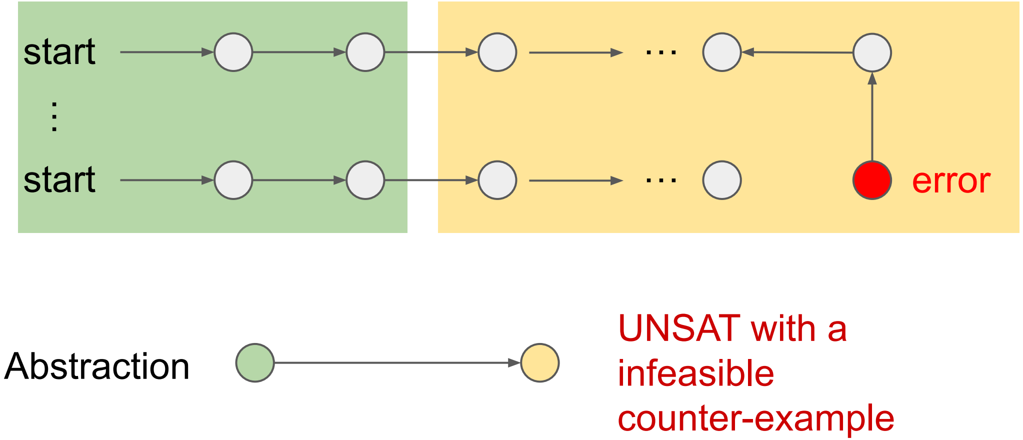


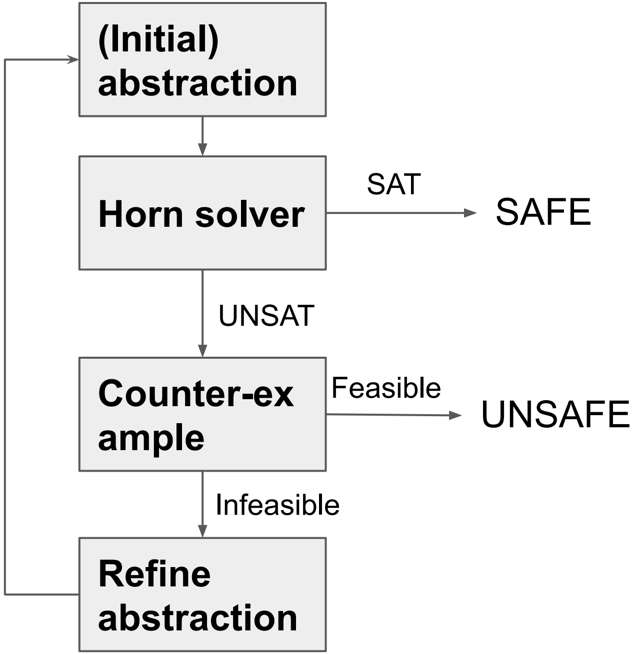
Solving CHCs by counter-example guided abstraction refinement (CEGAR) based model checking



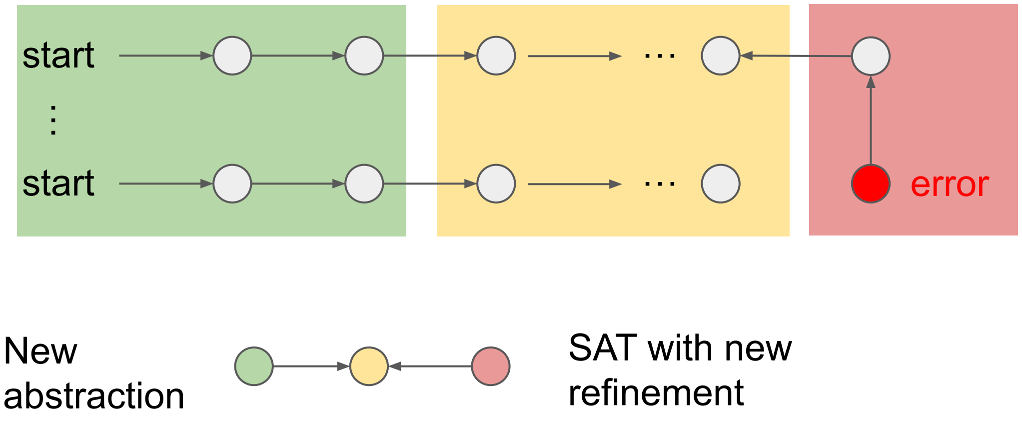


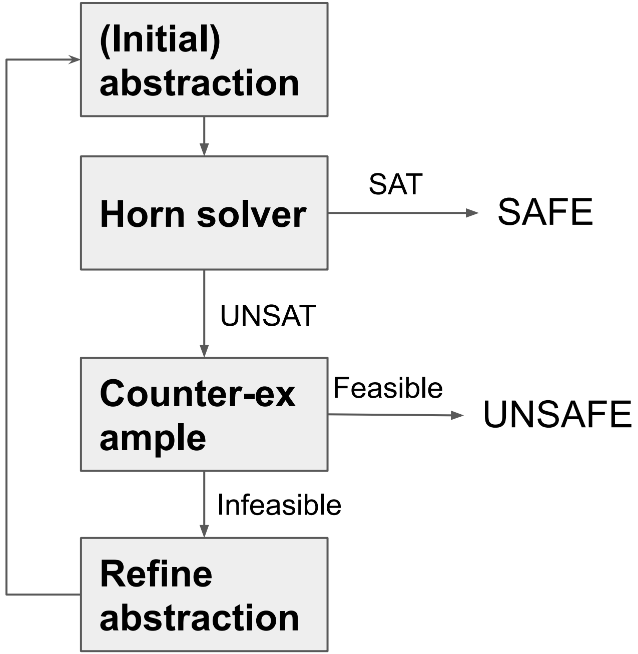
Solving CHCs by counter-example guided abstraction refinement (CEGAR) based model checking





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





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