Guiding Interpolation for Model Checking by Deep Learning Techniques

PhD student: Chencheng Liang

Supervisor: Philipp Rümmer, Marc Brockschmidt

Uppsala University

Sweden

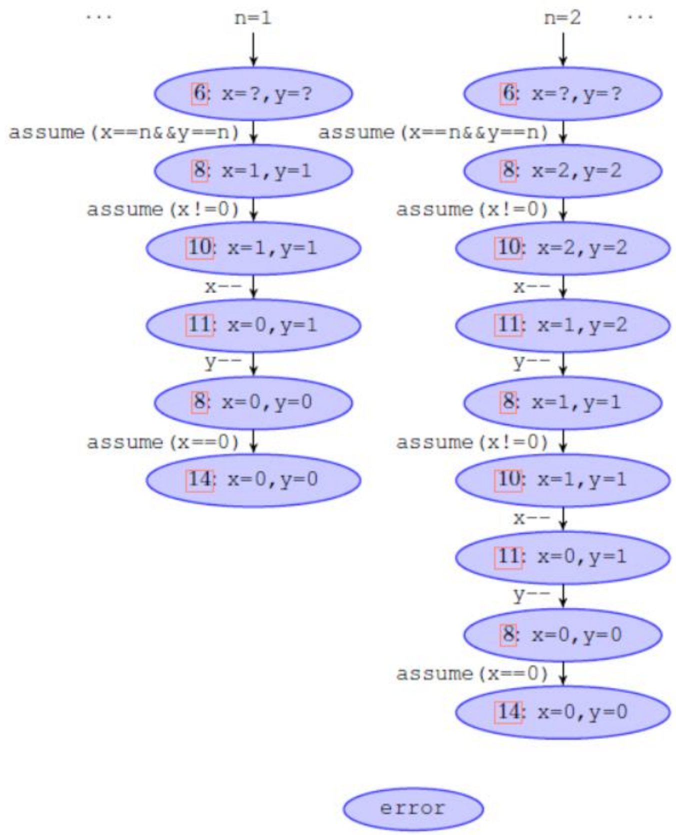
Outline

- Background
  - Model checking
  - CEGAR
  - Craig Interpolation
- Guiding Interpolation for Model Checking
- Summary
- Future works

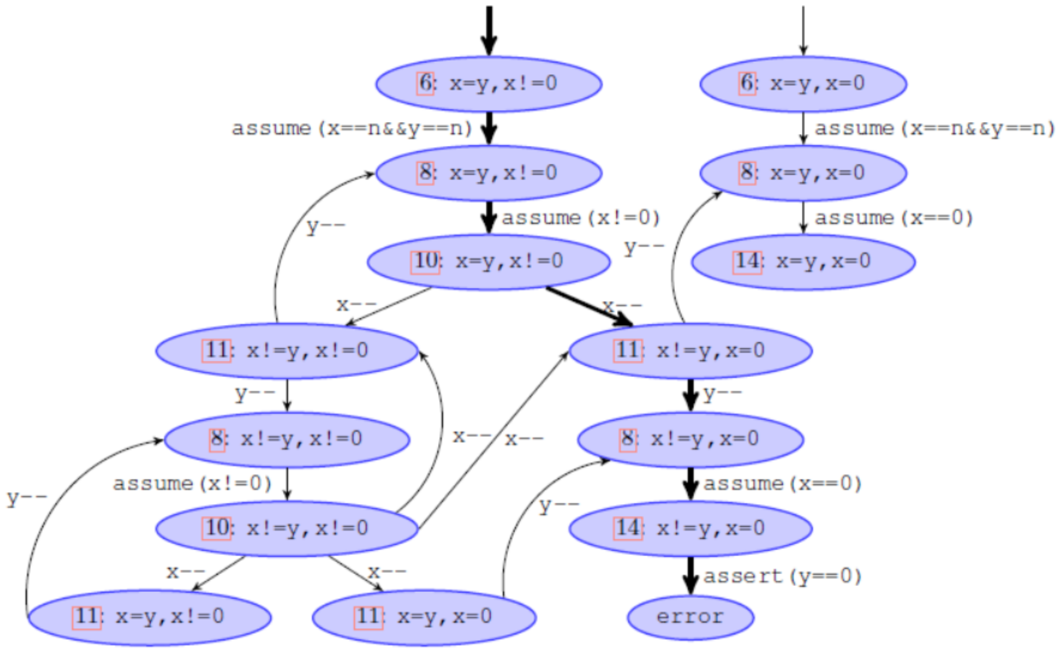
Abstraction-based model checking

```
extern int n;

void main()
{
  int x, y;
  assume(x==n && y==n);

  while (x!=0)
  {
    x--;
    y--;
  }

  assert(y==0);
}
```

Source code



Labelled concrete transition system (infinite states)

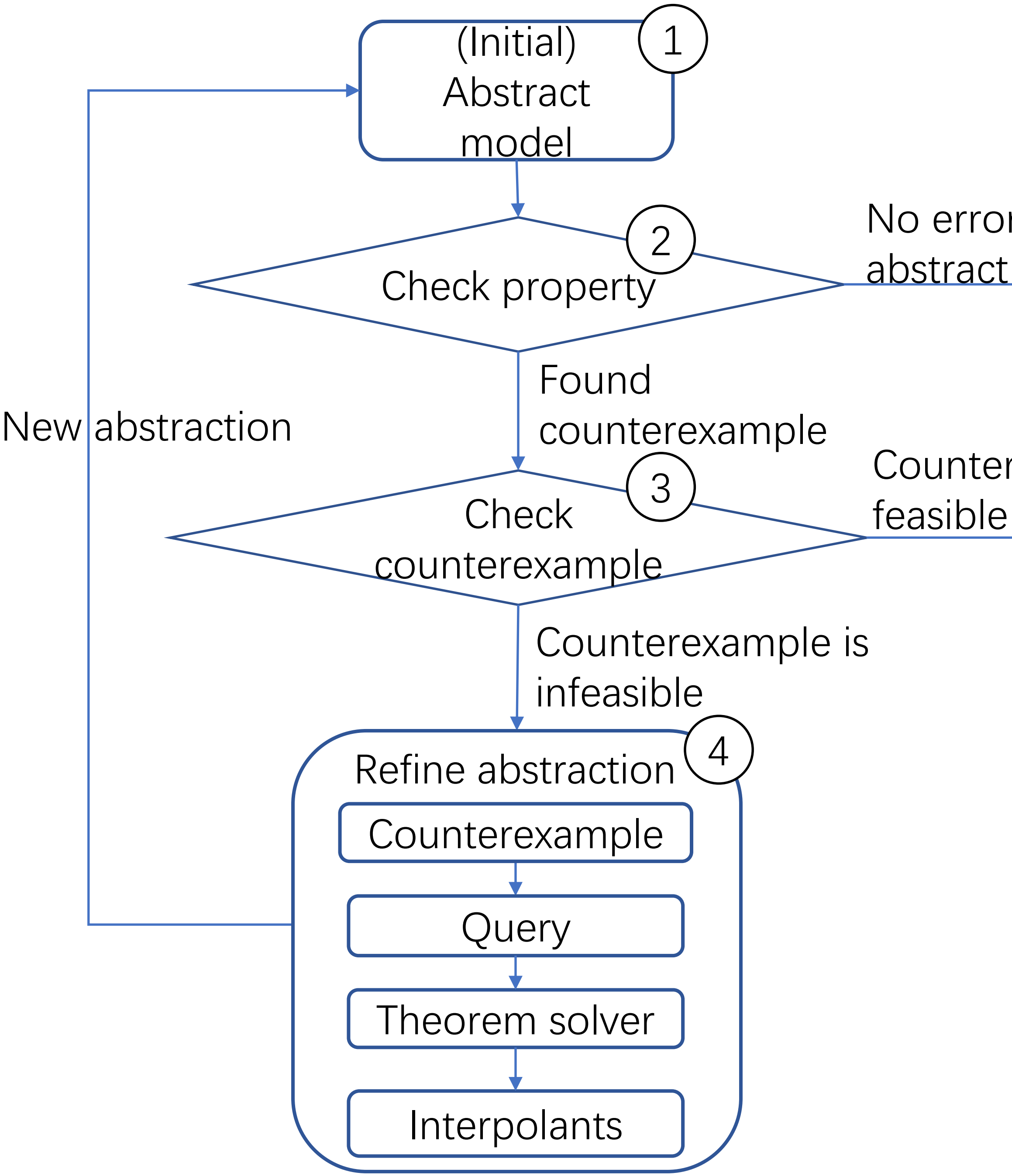


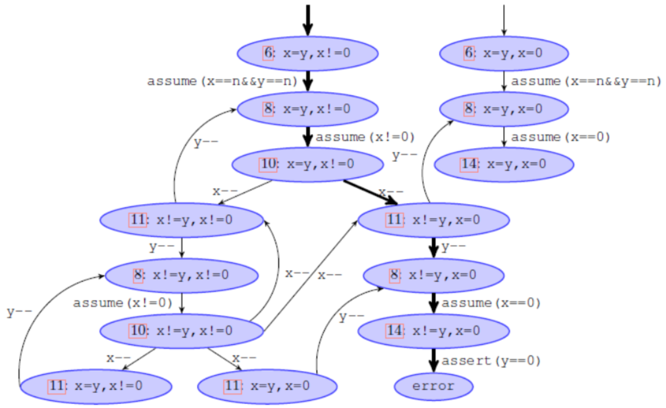
Abstract labelled transition system

$P_1 = \{x = y, x = 0\}$

CEGAR framework

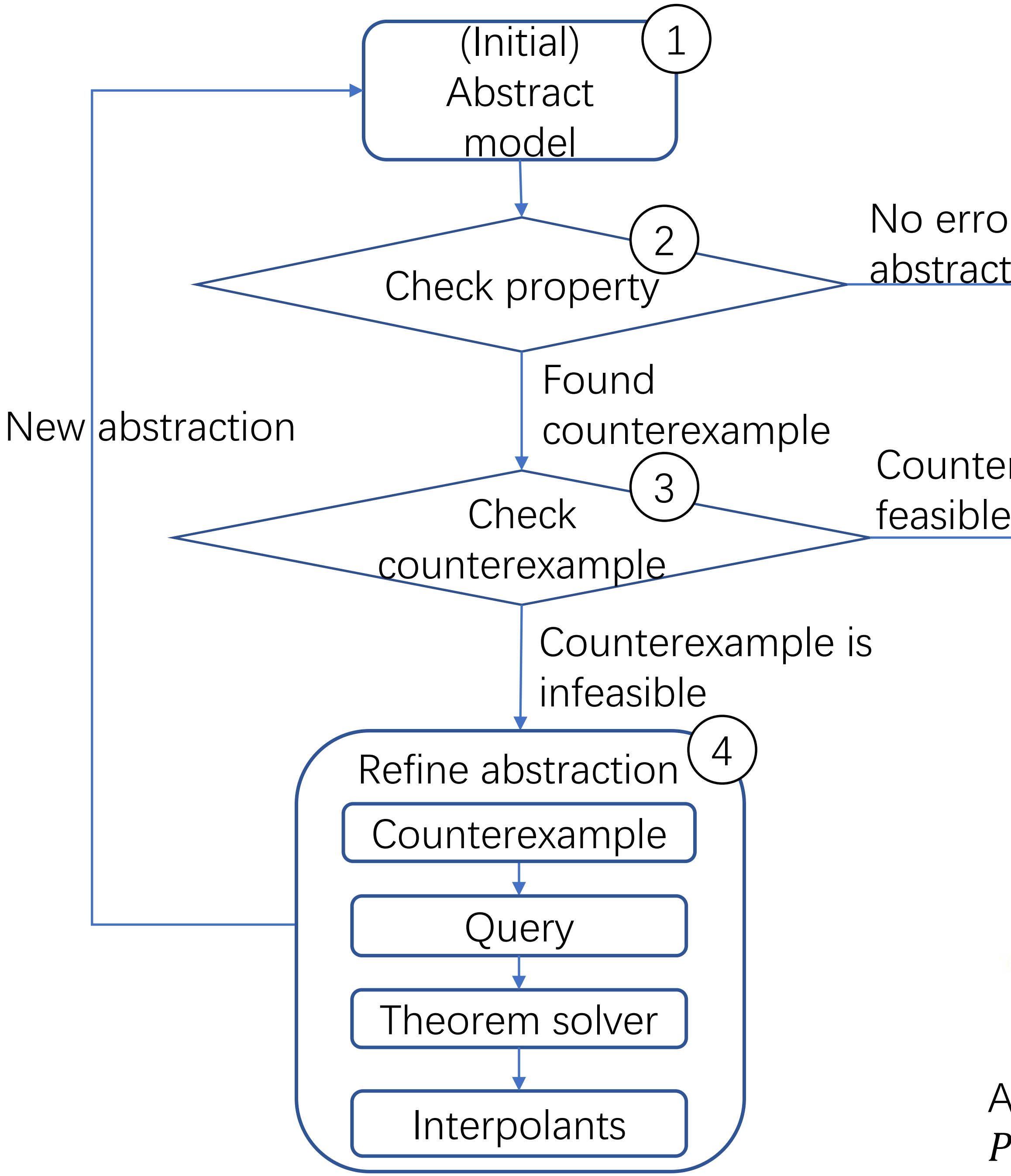
[Ceg00] Edmund Clarke et al. Counterexample-guided abstraction refinement. Computer Aided Verification





Abstract labelled transition system

$P_1 = \{x = y, x = 0\}$

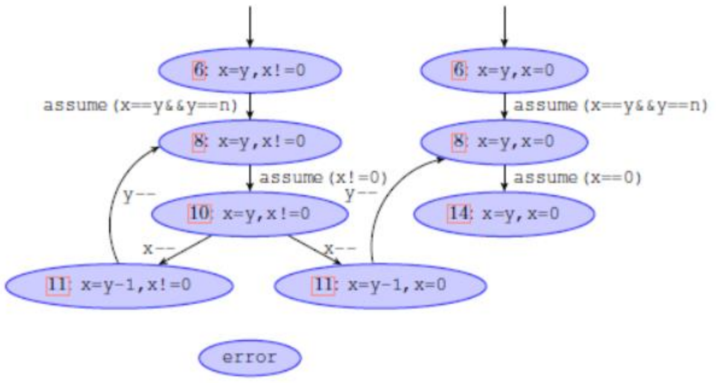


CEGAR framework

[Ceg00] Edmund Clarke et al. Counterexample-guided abstraction refinement. Computer Aided Verification

Craig Interpolation

[Cra57] William Craig. Linear reasoning. a new form of the herbrand-gentzen theorem



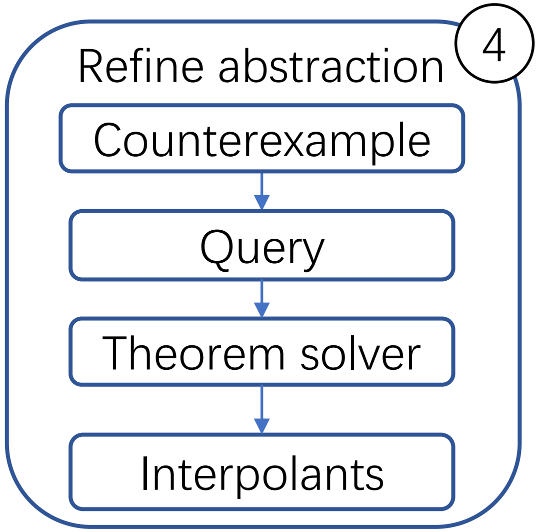
Abstract labelled transition system

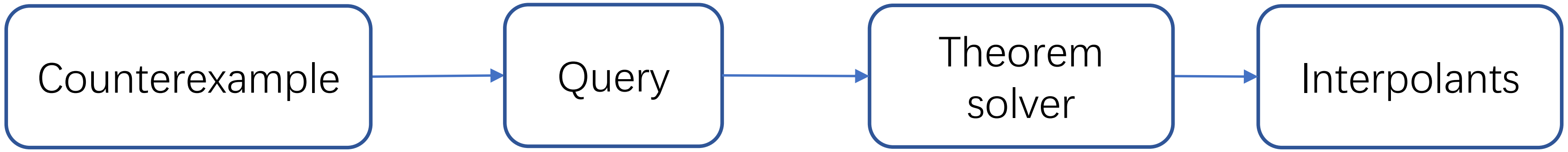
$P_2 = \{x = y, x = 0, \boldsymbol{x = y - 1}\}$

Refine abstraction

Craig Interpolation

[Cra57] William Craig. Linear reasoning. a new form of the herbrand-gentzen theorem



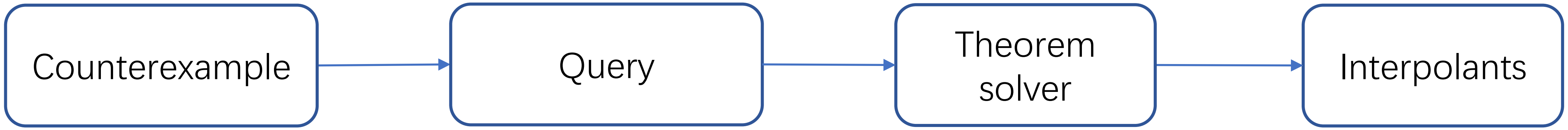


$$A = (x = 1 \wedge y = 2)$$
$$B = (x > y)$$

$$I_1 = (x = 1 \wedge y = 2)$$
$$I_2 = (y = x + 1)$$

Eldarica (abstract interpolation)

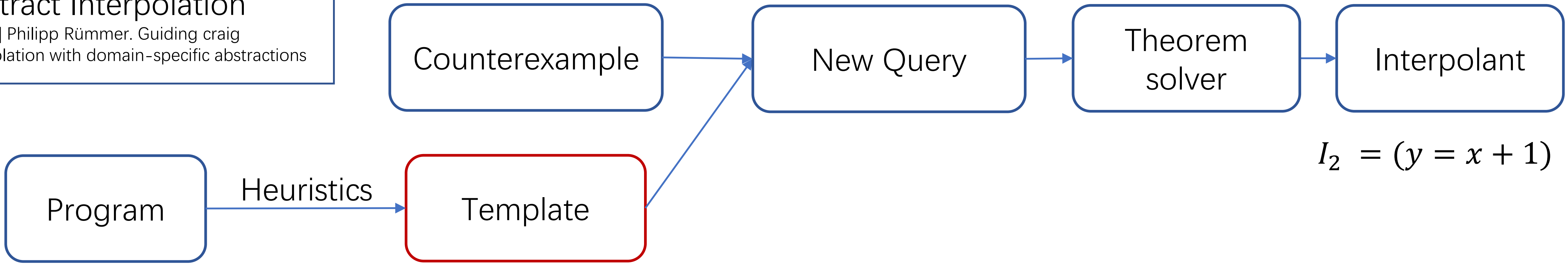
Craig Interpolation
[Cra57] William Craig. Linear reasoning. a new form of the herbrand-gentzen theorem



$$A = (x = 1 \land y = 2)$$
$$B = (x > y)$$

$$I_1 = (x = 1 \land y = 2)$$
$$I_2 = (y = x + 1)$$

Abstract Interpolation
[Gui16] Philipp Rümmer. Guiding craig interpolation with domain-specific abstractions



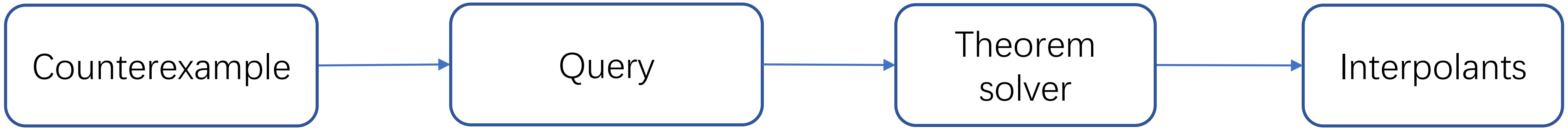
$$I_2 = (y = x + 1)$$

$$\{x - y\}$$

Eldarica (abstract interpolation)

Craig Interpolation

[Cra57] William Craig. Linear reasoning. a new form of the herbrand-gentzen theorem

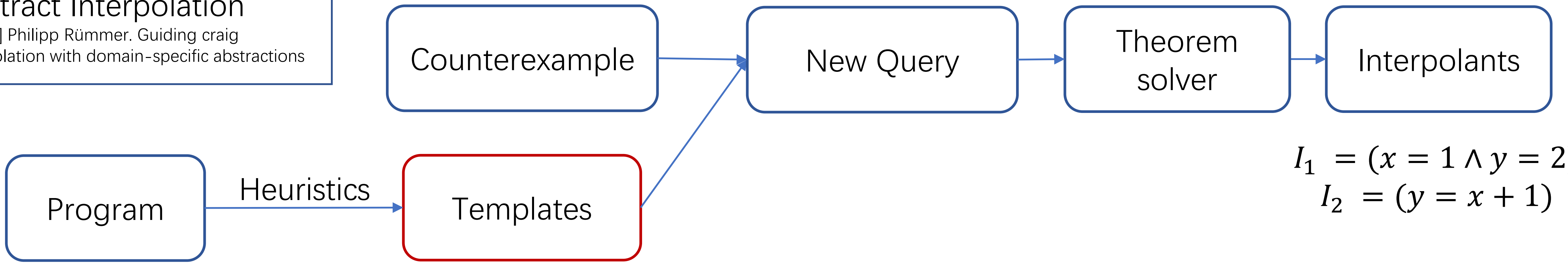


$$A = (x = 1 \land y = 2)$$
$$B = (x > y)$$

$$I_1 = (x = 1 \land y = 2)$$
$$I_2 = (y = x + 1)$$

Abstract Interpolation

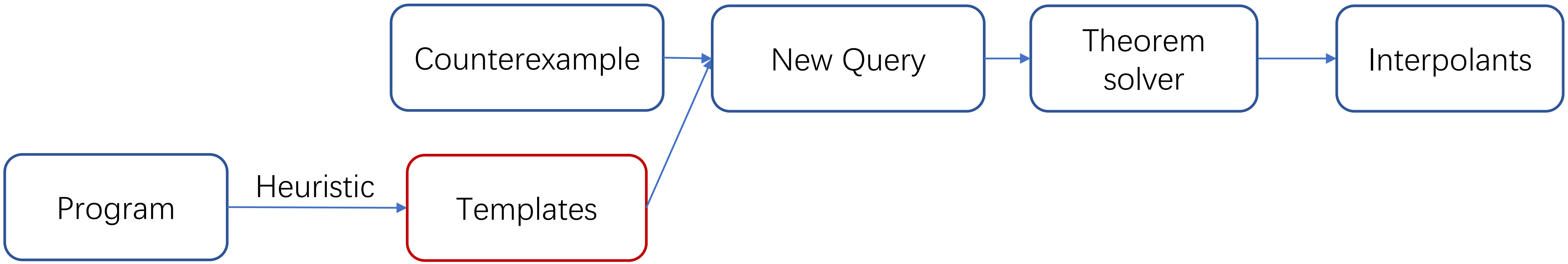
[Gui16] Philipp Rümmer. Guiding craig interpolation with domain-specific abstractions

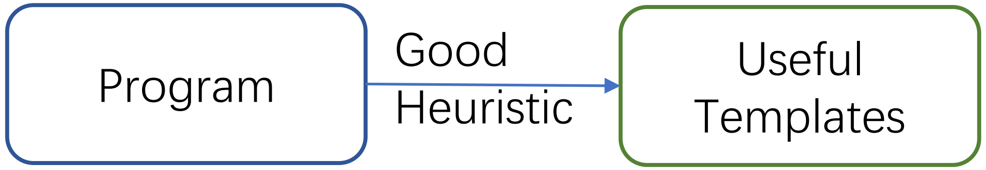


$$I_1 = (x = 1 \land y = 2)$$
$$I_2 = (y = x + 1)$$

$$\{x, y\}$$
$$\{x - y\}$$

Guiding interpolation





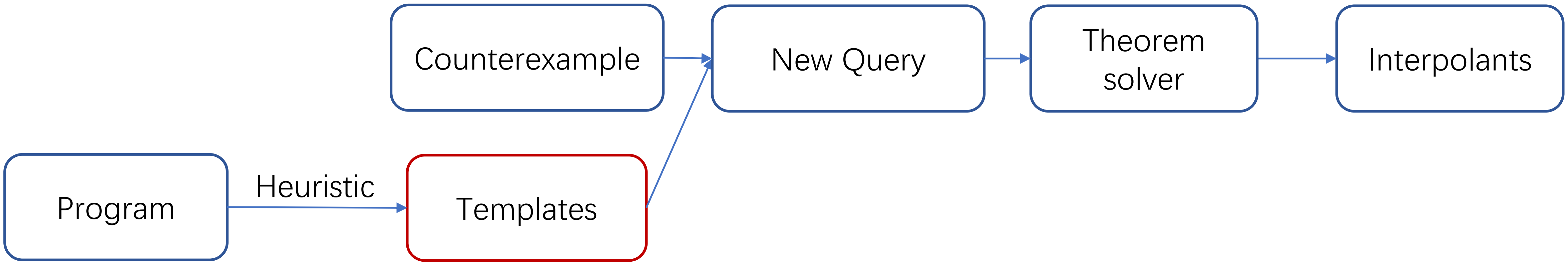
Abstract Interpolation

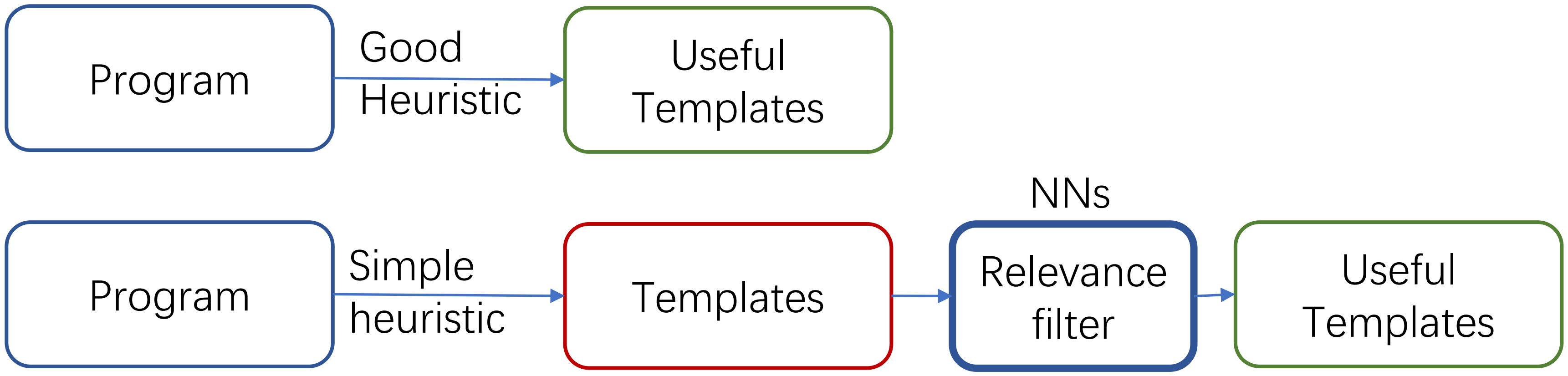
[Gui16] Philipp Rümmer. Guiding craig interpolation with domain-specific abstractions

Variable role

[Var17] Systematic predicate abstraction using variable roles

Relevance filter for templates

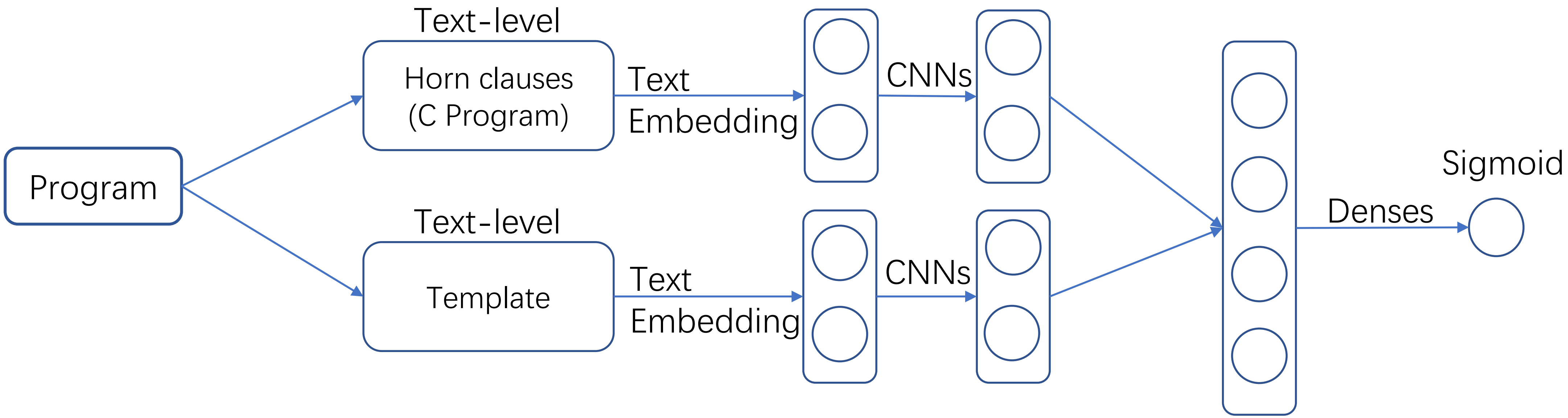




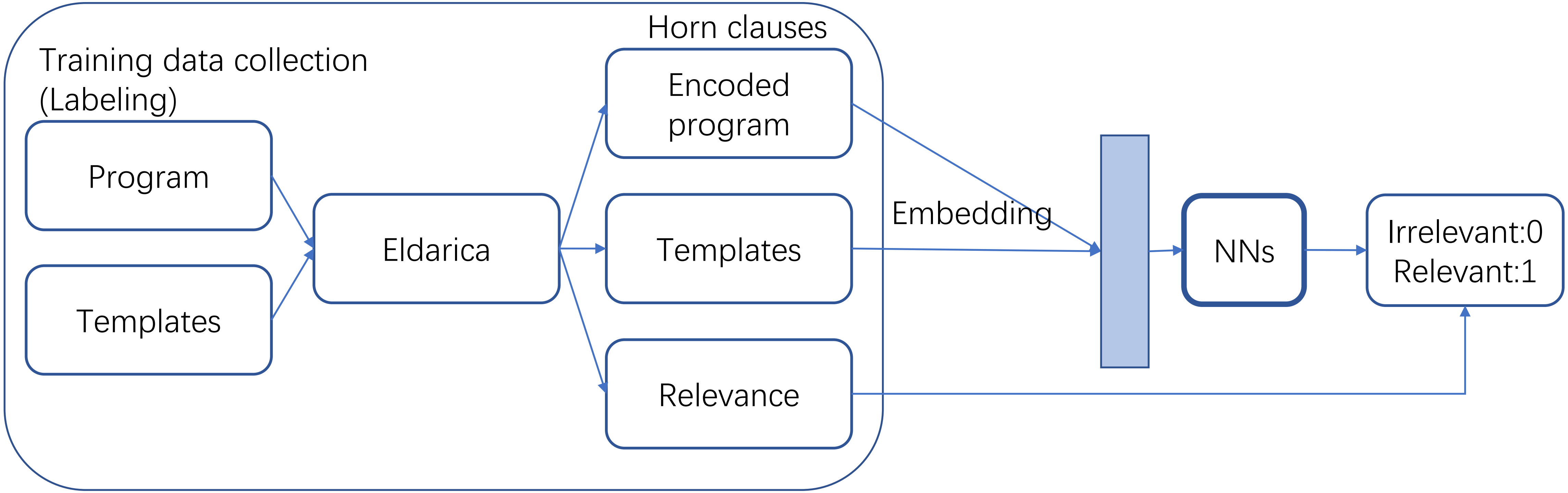
Abstract Interpolation
[Gui16] Guiding craig interpolation with domain-specific abstractions

Variable role
[Var17] Yulia Demyanova, et al. Systematic predicate abstraction using variable roles

Network structure

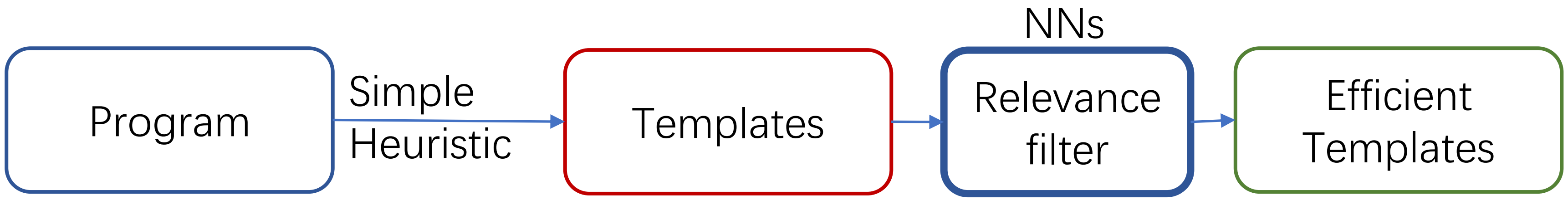


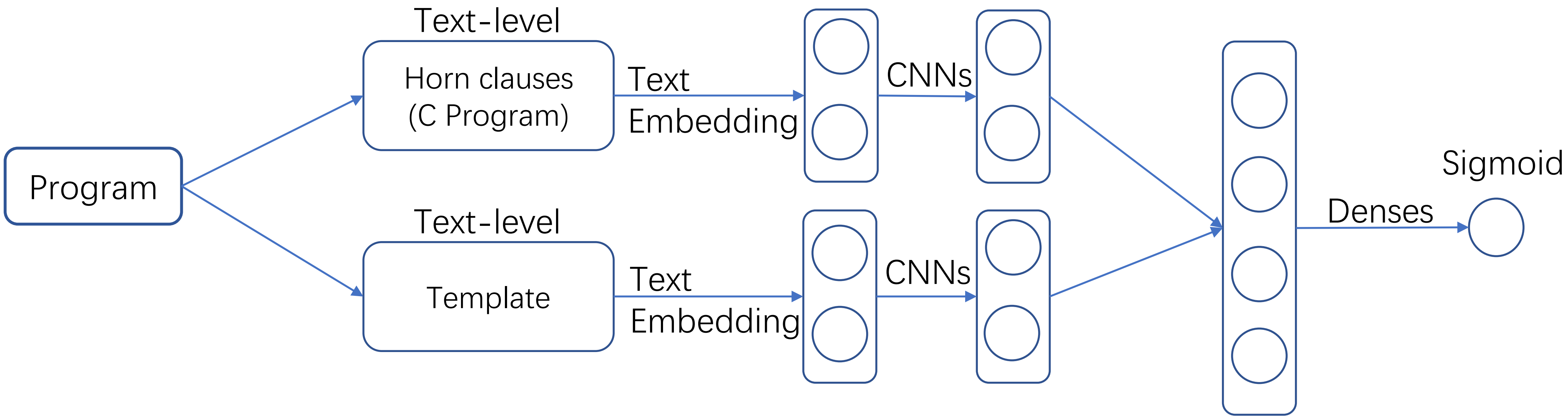
Labelling and training process



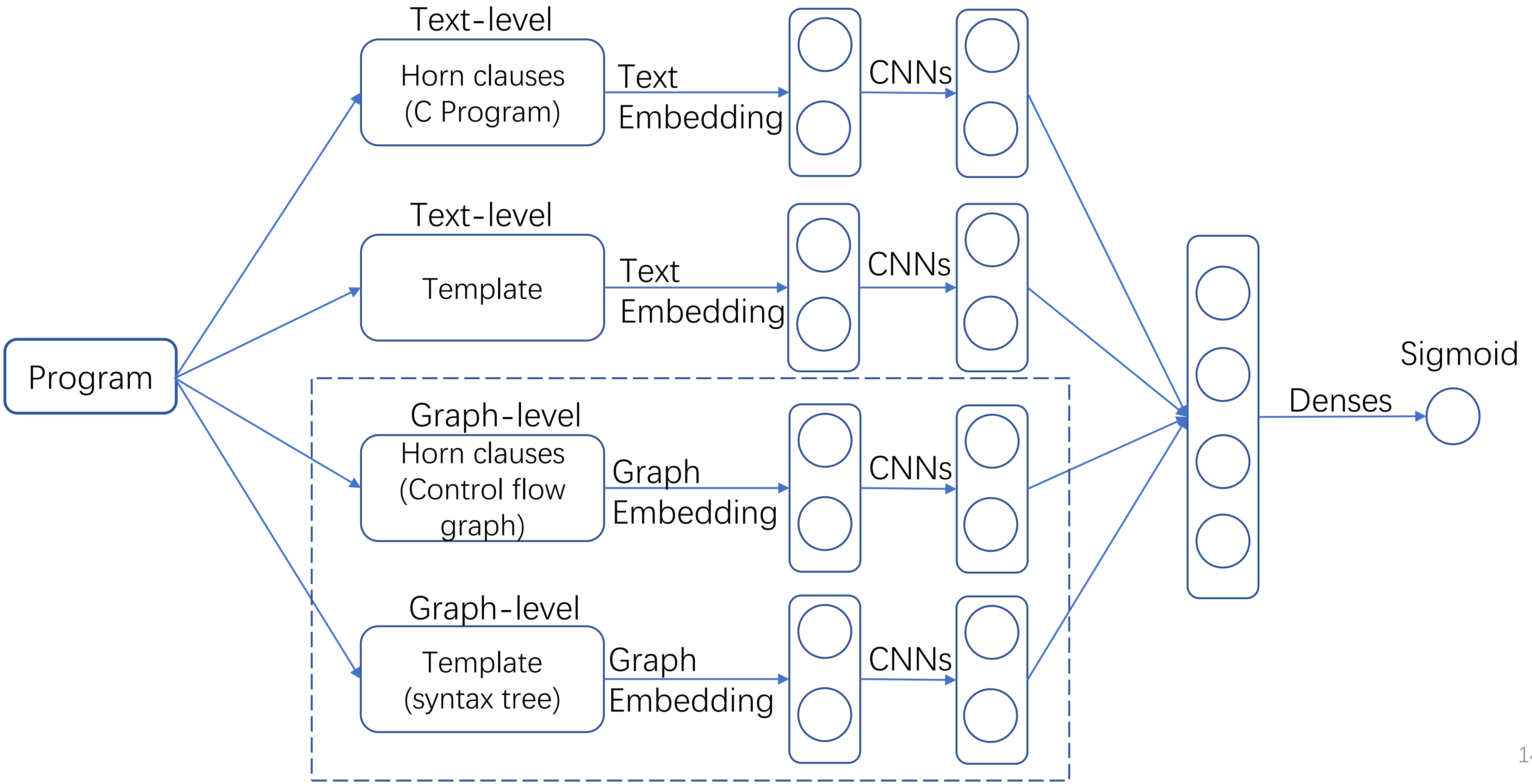
Network structure

DeepMath
[Dee16] Geoffrey Irving. Deepmath-deep sequence models for premise selection

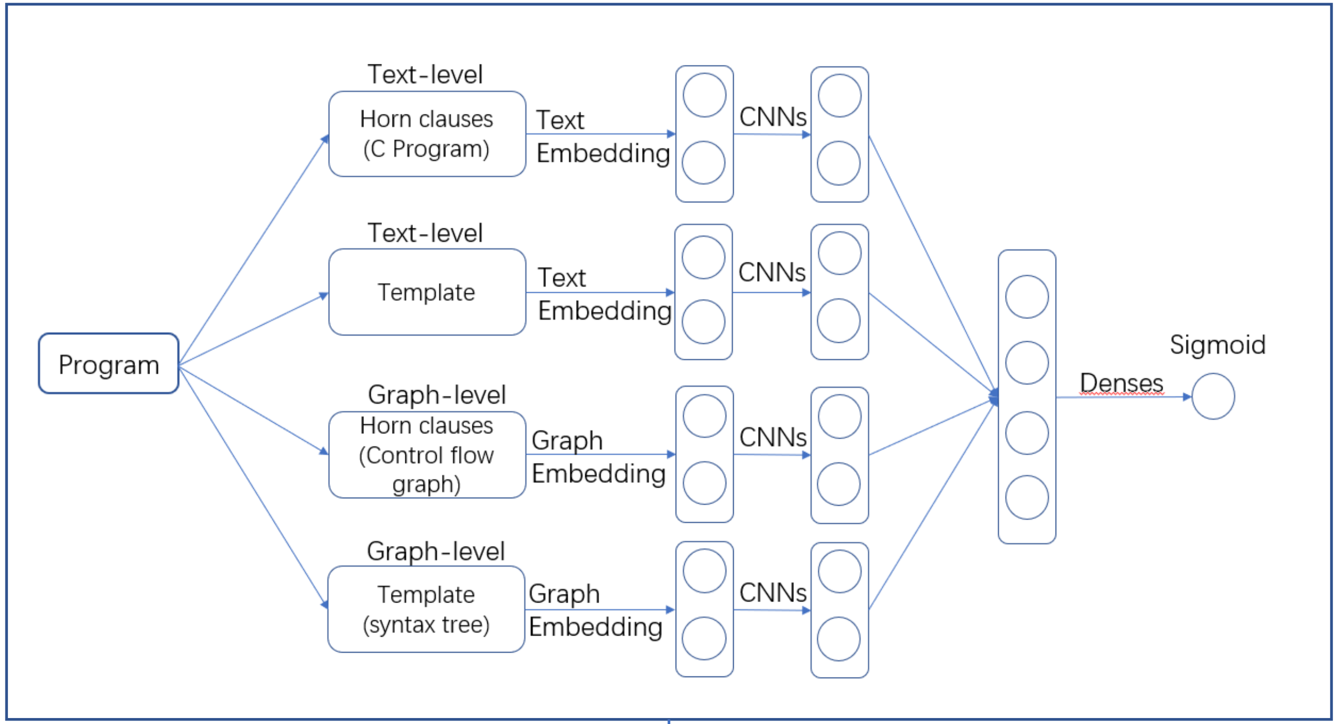


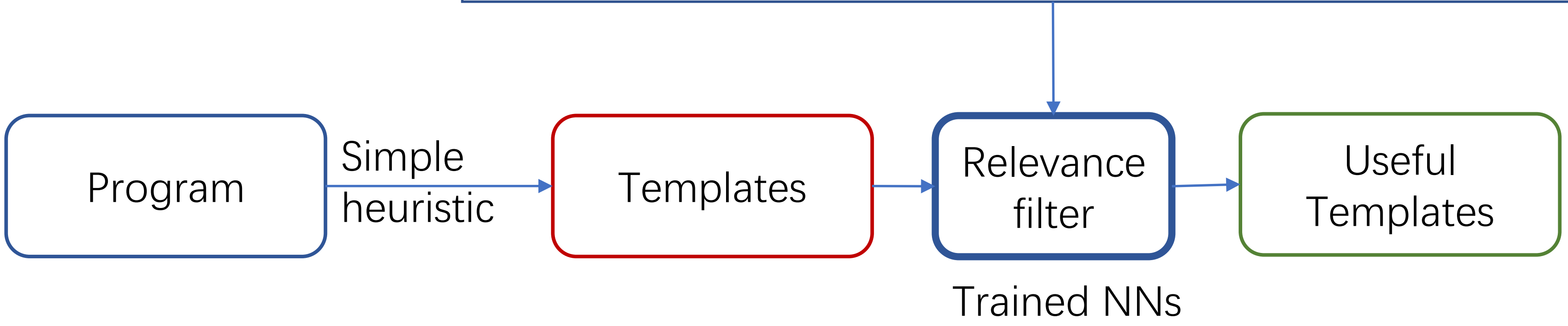


Training structure in graph-level

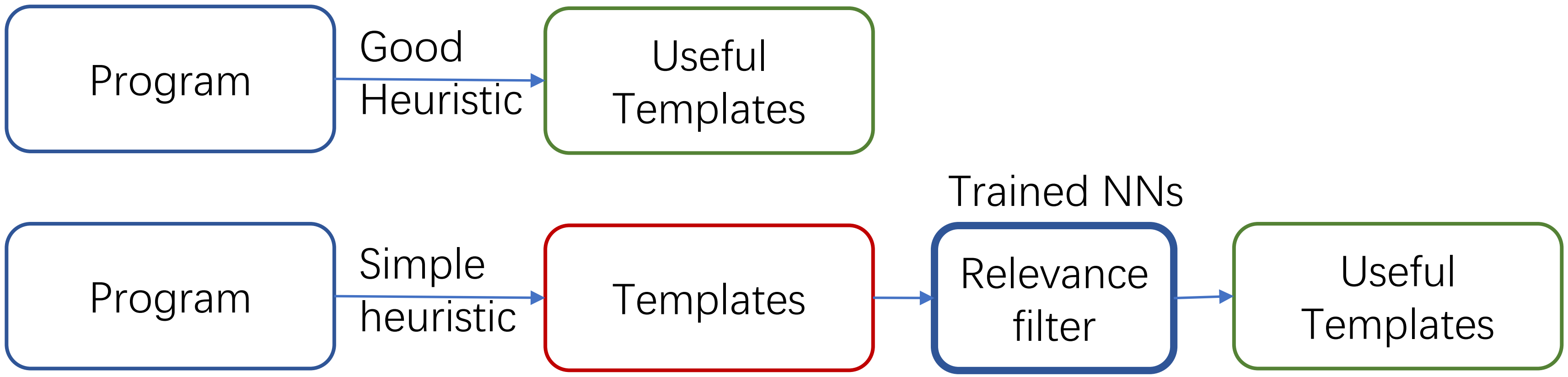


Overview

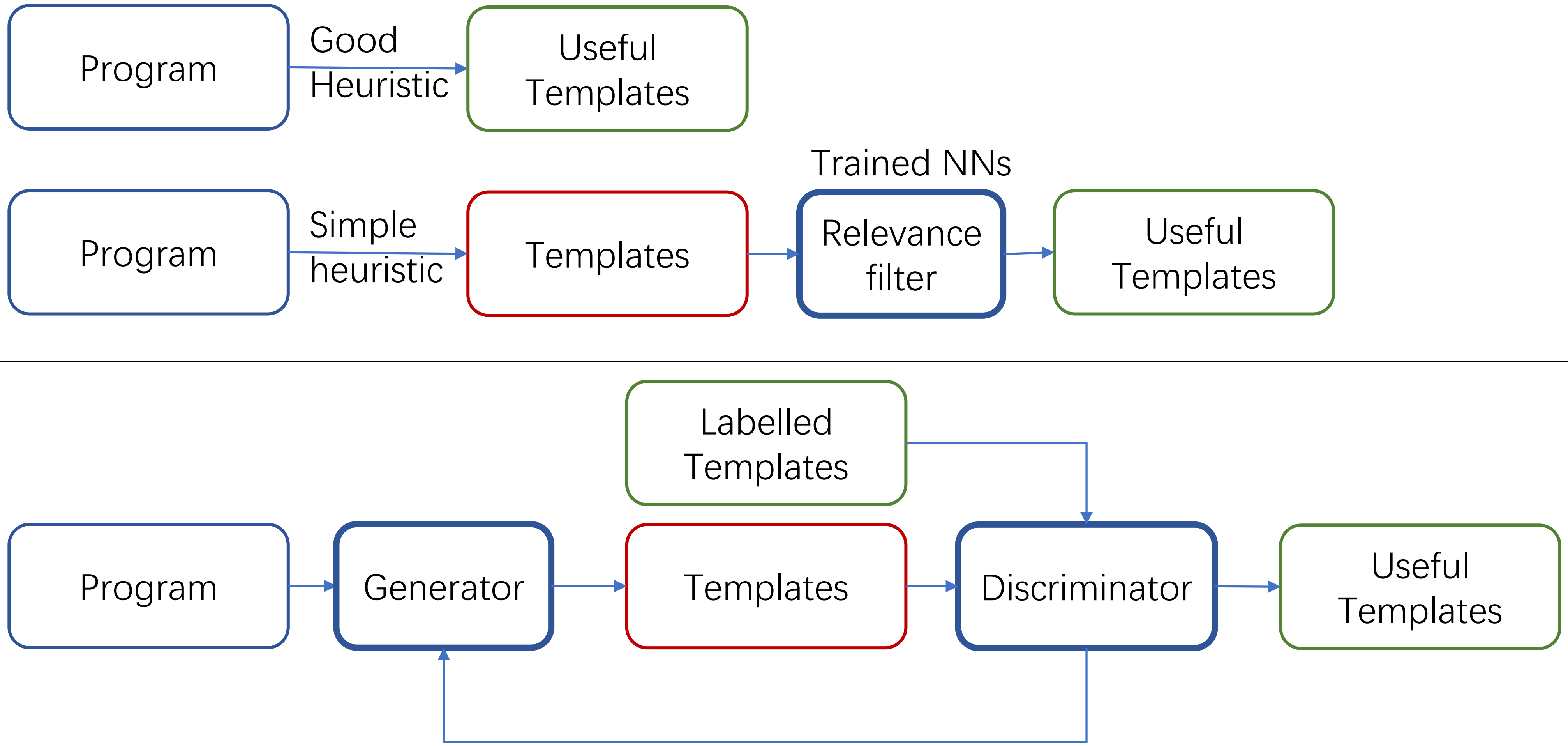

Text-level
Horn clauses
(C Program)
Text
Embedding
CNNs
Text-level
Template
Text
Embedding
CNNs
Program
Graph-level
Horn clauses
(Control flow
graph)
Graph
Embedding
CNNs
Graph-level
Template
(syntax tree)
Graph
Embedding
CNNs
Sigmoid
Denses


Program
Simple
heuristic
Templates
Relevance
filter
Useful
Templates
Trained NNs

Relevance filter for templates



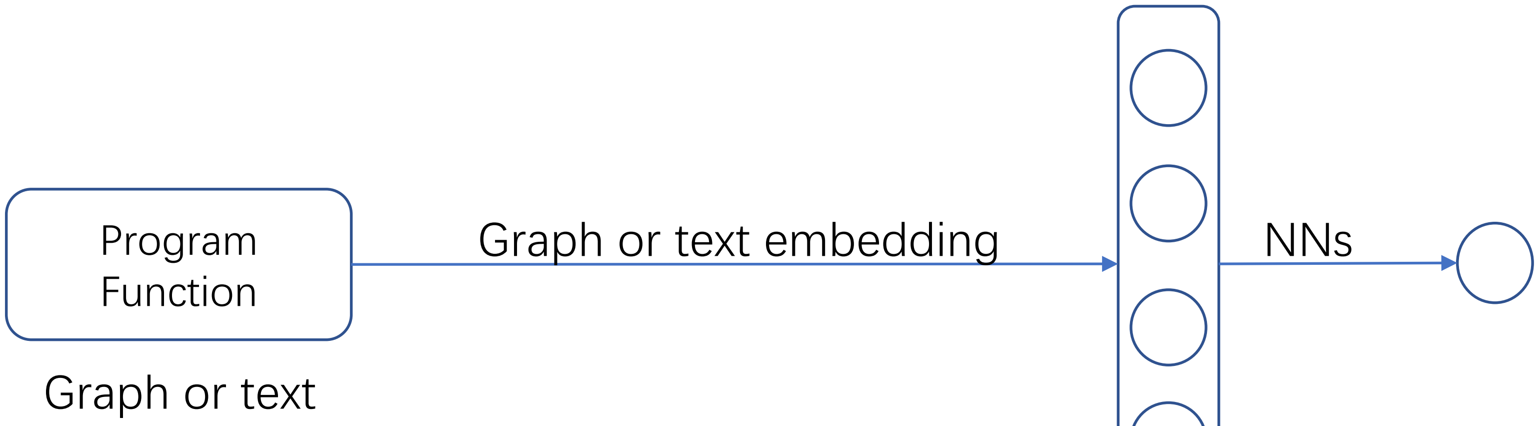
Train a generator to generate templates



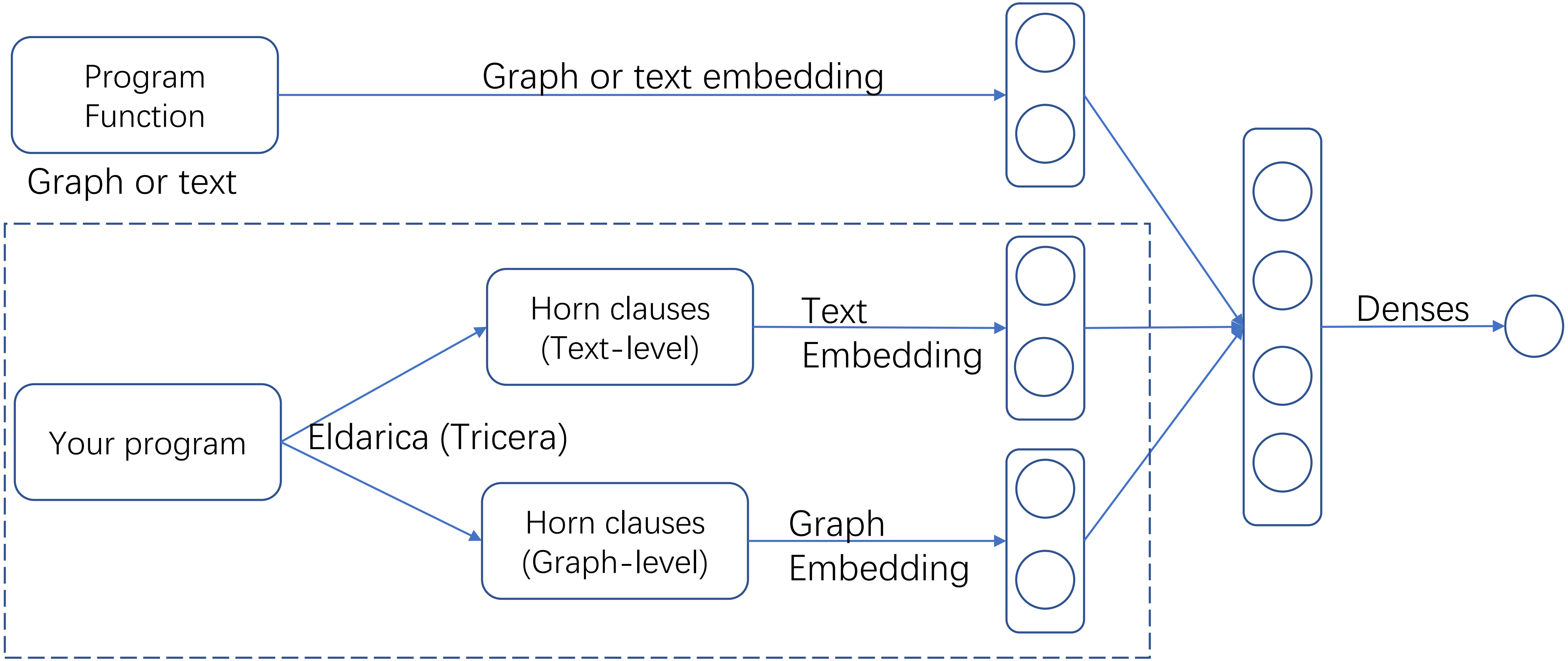
Summary

- Goal: filter templates to guiding interpolation for model checking
- Preliminary results:
  - Built a relevance filter
  - General tool: provide horn clauses in text-level and graph-level embedding.

Example: function name prediction



Additional training vector from horn clauses

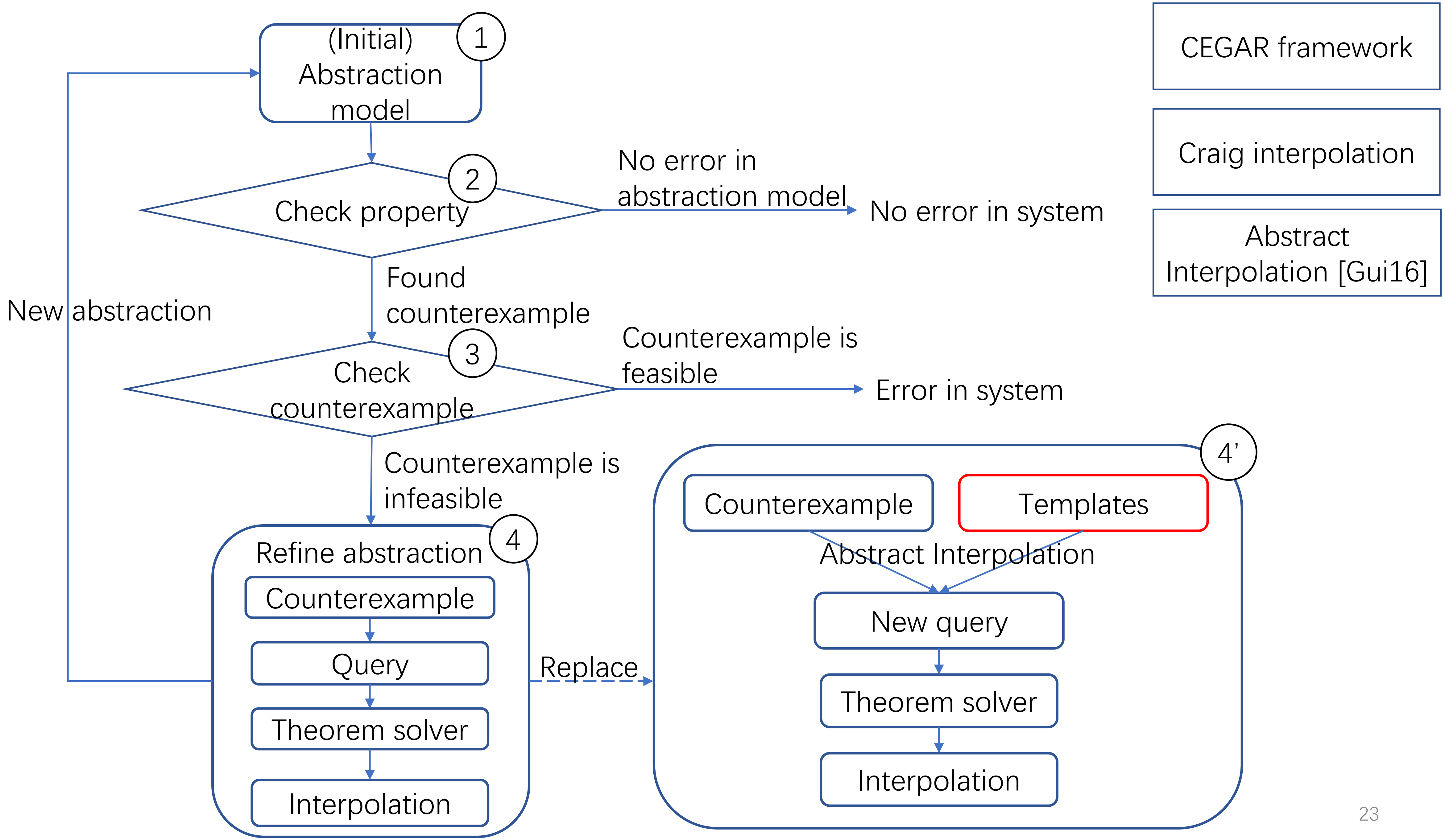


Next step

- Feed the filtered templates back to model checker.
- Transform horn clauses to graph representation.
- Bigger and more diverse datasets.

Thank you!

Chencheng Liang
Uppsala University
Sweden



(Initial) Abstraction model
1
Check property
2
No error in abstraction model
No error in system
Found counterexample
New abstraction
Check counterexample
3
Counterexample is feasible
Error in system
Counterexample is infeasible
Refine abstraction
4
Counterexample
Query
Theorem solver
Interpolation
Replace
4'
Counterexample
Templates
Abstract Interpolation
New query
Theorem solver
Interpolation
CEGAR framework
Craig interpolation
Abstract Interpolation [Gui16]

Benchmarks and tools

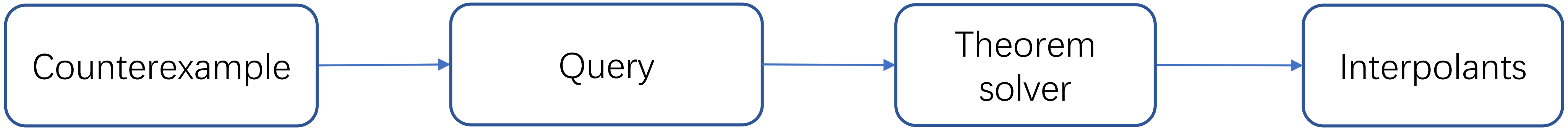
- Benchmarks (C programs)
  - SV-COMP’16 categories "Integers and Control Flow" and "Loops“, and loop invariant generation [Svc16]

- Tools
  - Model checker: Eldarica [Eld18]
  - Theorem Solver: Princess[Pri08]
  - Text embedding: Doc2vec [Dis14]
  - Graph embedding: Graph2vec [nod16]
  - Graph processing: Graphviz [Gra12] and NetworkX [Net19]
  - Neural network structure: CNNs [Ima12] and Denses [Dee16]

References

- [Dee16] Geoffrey Irving, Christian Szegedy, Alexander A Alemi, Niklas Een, Francois Chollet, and Josef Urban. Deepmath - deep sequence models for premise selection. Advances in Neural Information Processing Systems 29, pages 2235–2243. Curran Associates, Inc., 2016.
- [Ceg00] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith Counterexample-guided abstraction refinement. Computer Aided Verification, pages 154–169, Berlin, Heidelberg, 2000.Springer Berlin Heidelberg
- [Var17] Yulia Demyanova, Philipp Rümmer, and Florian Zuleger. Systematic predicate abstraction using variable roles. NASA Formal Methods, pages 265–281, Cham, 2017. Springer International Publishing.
- [Eld18] Hossein Hojjat and Philipp Rümmer. The eldarica horn solver. In 2018 Formal Methods in Computer Aided Design (FMCAD), pages 1–7, Oct 2018
- [Gui16] Jérome Leroux, Philipp Rümmer, and Pavle Suboti´c. Guiding craig interpolation with domain-specific abstractions. Acta Informatica, 53(4):387–424, Jun 2016.
- [Dis14] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. CoRR, abs/1405.4053, 2014
- [Cra57] William Craig. Linear reasoning. a new form of the herbrand-gentzen theorem. Journal of Symbolic Logic, 22(3):250–268, 1957.
- [Gra12] Narayanan, Annamalai and Chandramohan, Mahinthan and Venkatesan, Rajasekar and Chen, Lihui and Liu, Yang. graph2vec: Learning distributed representations of graphs. MLG 2017, 13th International Workshop on Mining and Learning with Graphs (MLGWorkshop 2017).
- [Ima12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.
- [Dee16 p.164-167] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. https://www.deeplearningbook.org
- [Pri08] Philipp Rümmer. A Constraint Sequent Calculus for First-Order Logic with Linear Integer Arithmetic. 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR), Doha, Qatar, 2008.Springer-Verlag, LNCS 5330, pages 274-289
- [Svc16] https://sv-comp.sosy-lab.org/2016/benchmarks.php
- [Gra19] https://www.graphviz.org
- [Net19] https://networkx.github.io

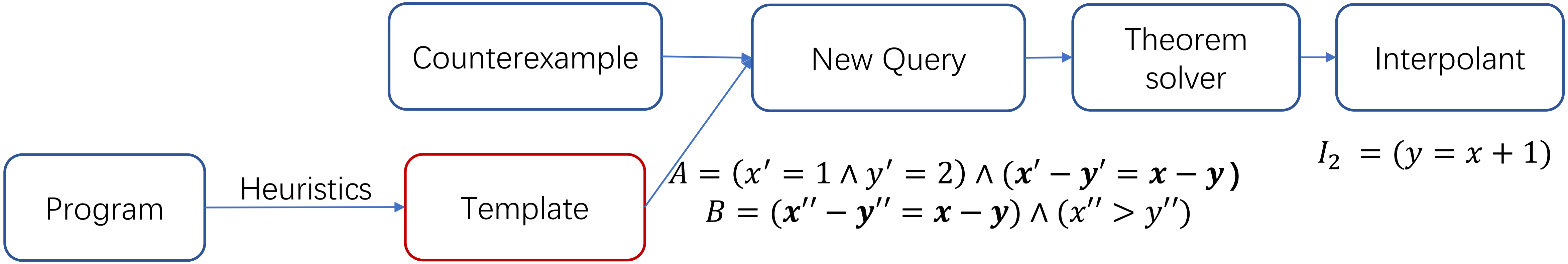
Eldarica (abstract interpolation)



$$A = (x = 1 \land y = 2)$$
$$B = (x > y)$$

$$I_1 = (x = 1 \land y = 2)$$
$$I_2 = (y = x + 1)$$

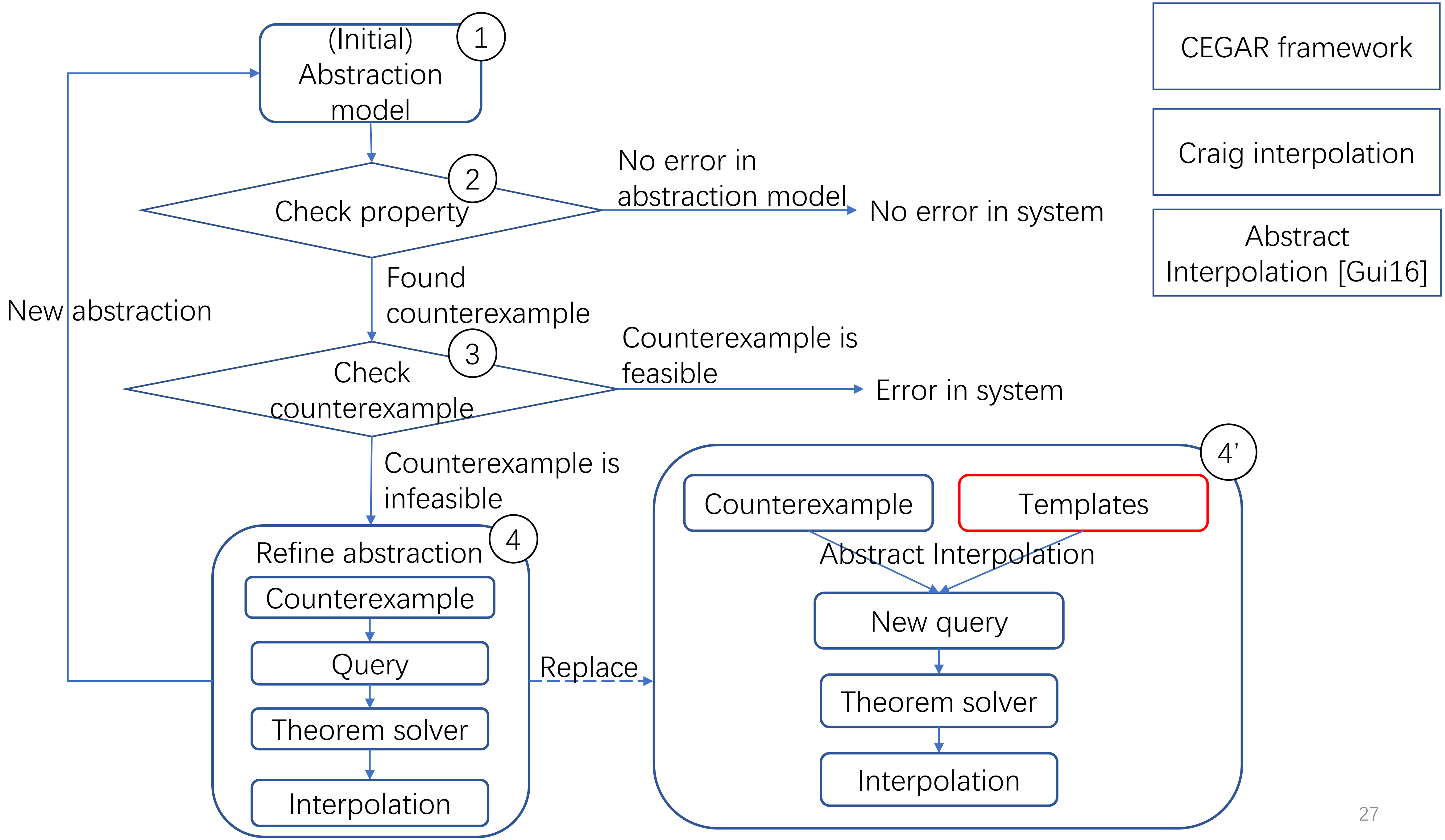
Abstract Interpolation [Gui16]



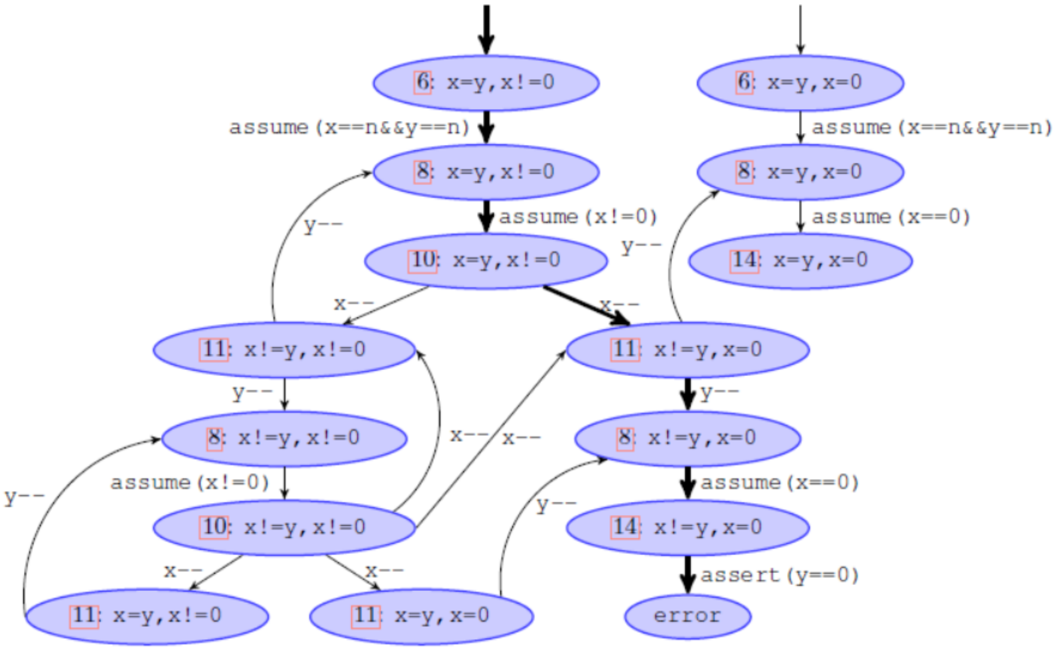
$$I_2 = (y = x + 1)$$

$$A = (x' = 1 \land y' = 2) \land (\boldsymbol{x' - y' = x - y})$$
$$B = (\boldsymbol{x'' - y'' = x - y}) \land (x'' > y'')$$

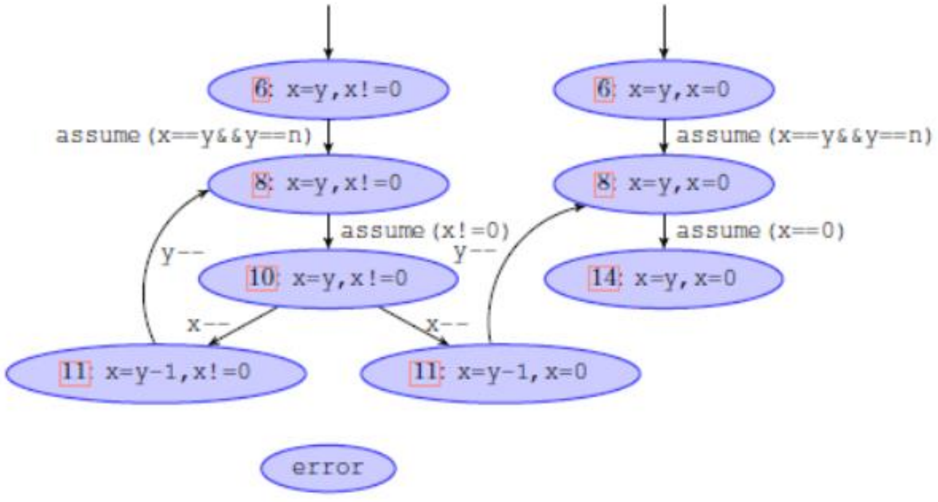
$$\{x - y\}$$



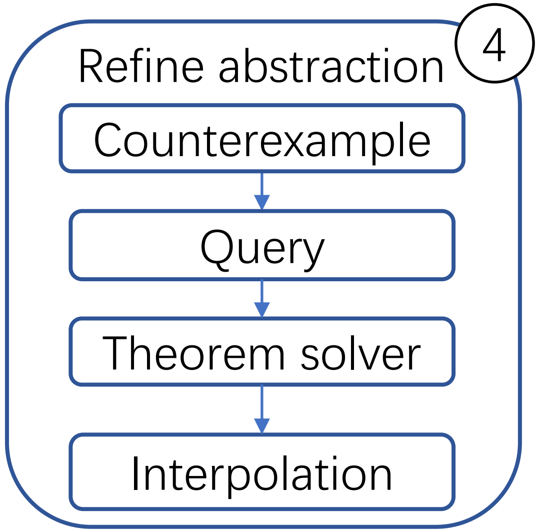
(Initial) Abstraction model
1
Check property
2
No error in abstraction model
No error in system
Found counterexample
New abstraction
Check counterexample
3
Counterexample is feasible
Error in system
Counterexample is infeasible
Refine abstraction
4
Counterexample
Query
Theorem solver
Interpolation
Replace
4'
Counterexample
Templates
Abstract Interpolation
New query
Theorem solver
Interpolation
CEGAR framework
Craig interpolation
Abstract Interpolation [Gui16]



Abstract labelled transition system
$P_1 = \{x = y, x = 0\}$



Abstract labelled transition system
$P_2 = \{x = y, x = 0, \boldsymbol{x = y - 1}\}$

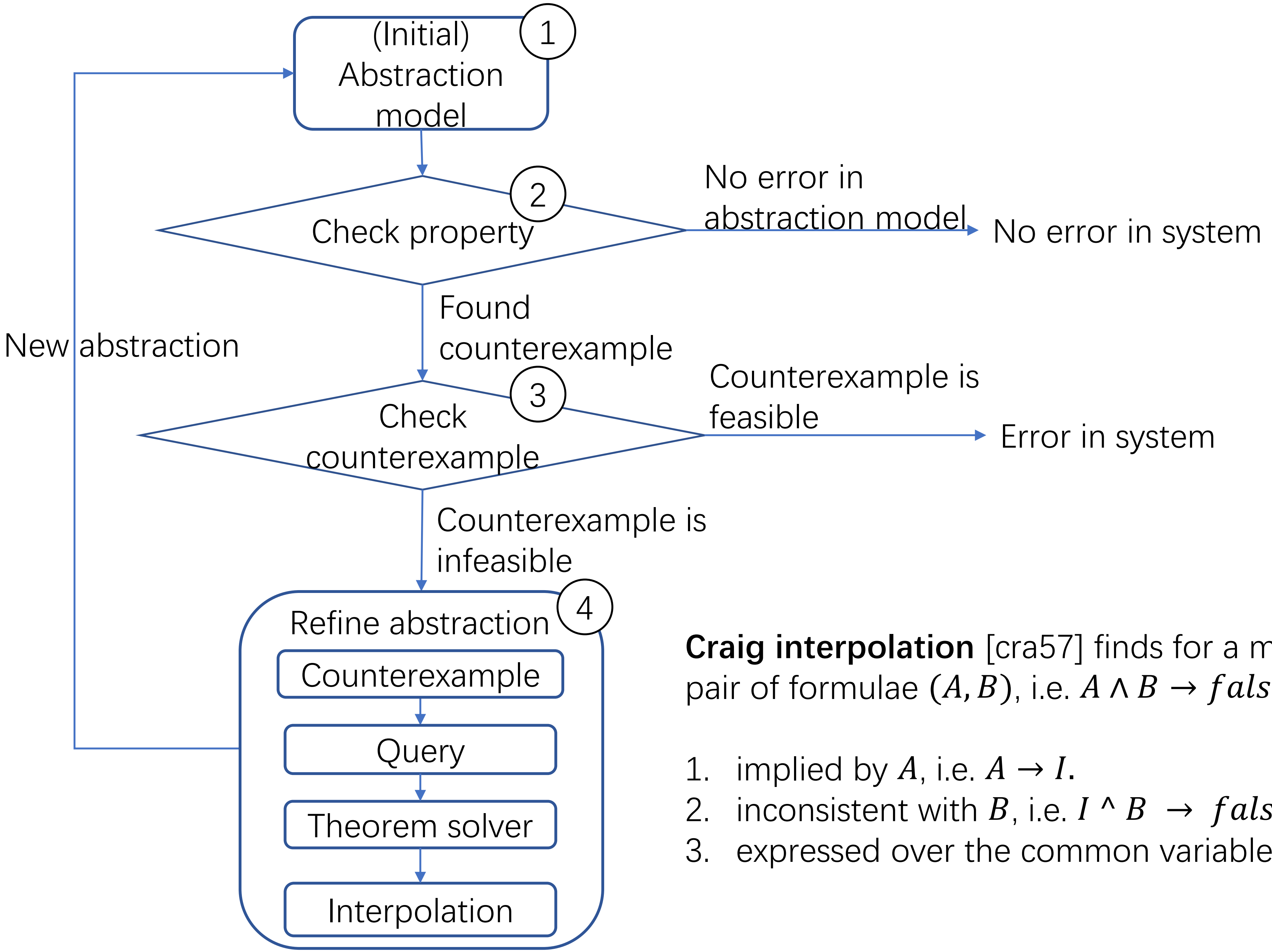


Counterexample (path): $x \overset{6}{=} n \wedge y \overset{6}{=} n \wedge x \overset{8}{\neq} 0 \wedge x' \overset{10}{=} x - 1 \wedge y' \overset{11}{=} y - 1 \wedge x' \overset{8}{=} 0 \wedge y' \overset{14}{!=} 0.$

Separated path (query): $A = (x = n \wedge y = n \wedge x \mathrel{!=} 0 \wedge x' = x - 1)$ and $B = (y' = y - 1 \wedge x' = 0 \wedge y' \mathrel{!=} 0).$

1. implied by A, i.e. $A \rightarrow I$.
2. inconsistent with B, i.e. $I \wedge B \rightarrow false$ and
3. expressed over the common variables of A and B.

Interpolation (new abstraction): $I = (x' = y - 1)$

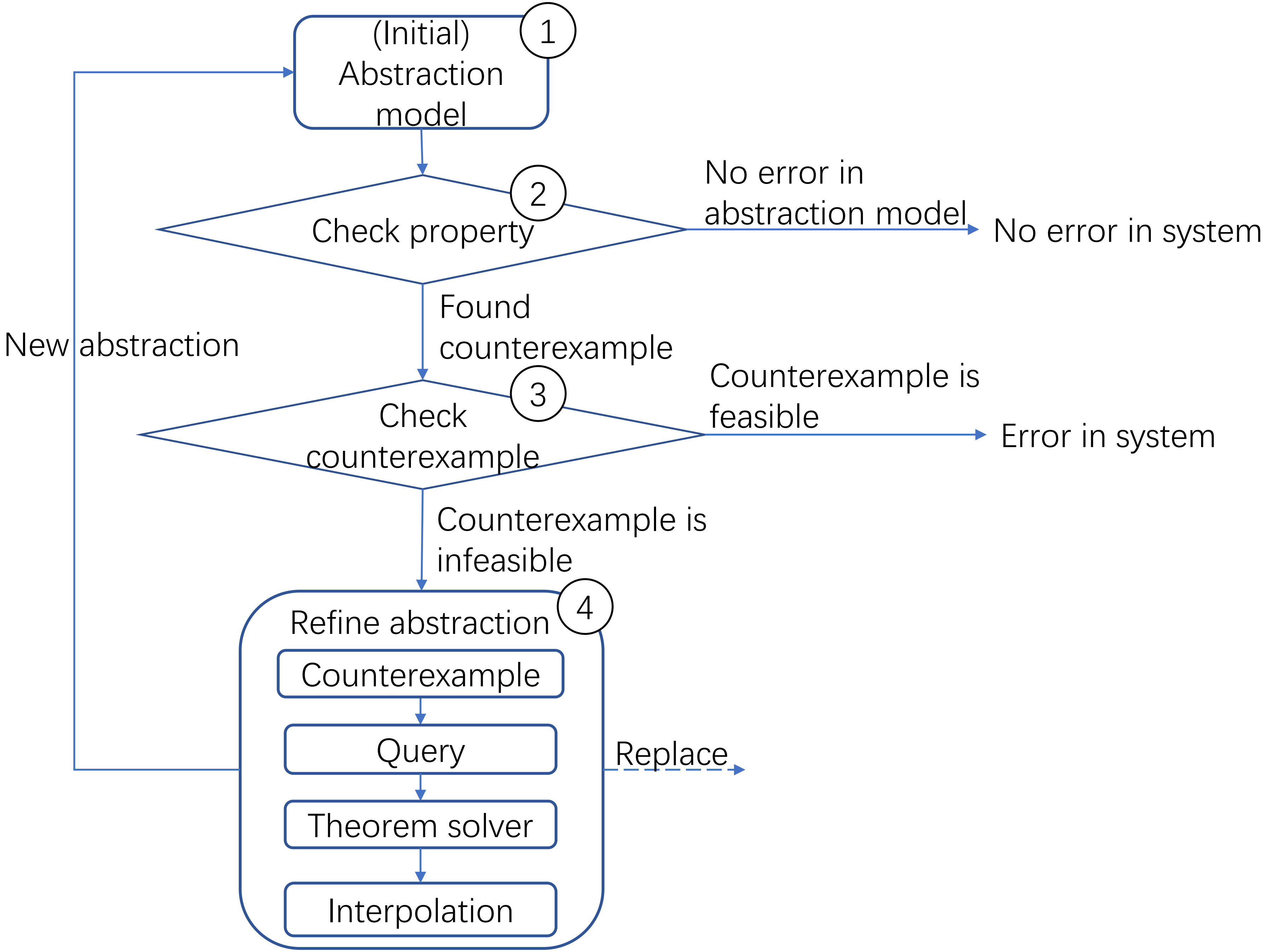


CEGAR framework

Craig interpolation

Craig interpolation [cra57] finds for a mutually inconsistent pair of formulae (A, B), i.e. $A \wedge B \rightarrow false$, a formula I which is

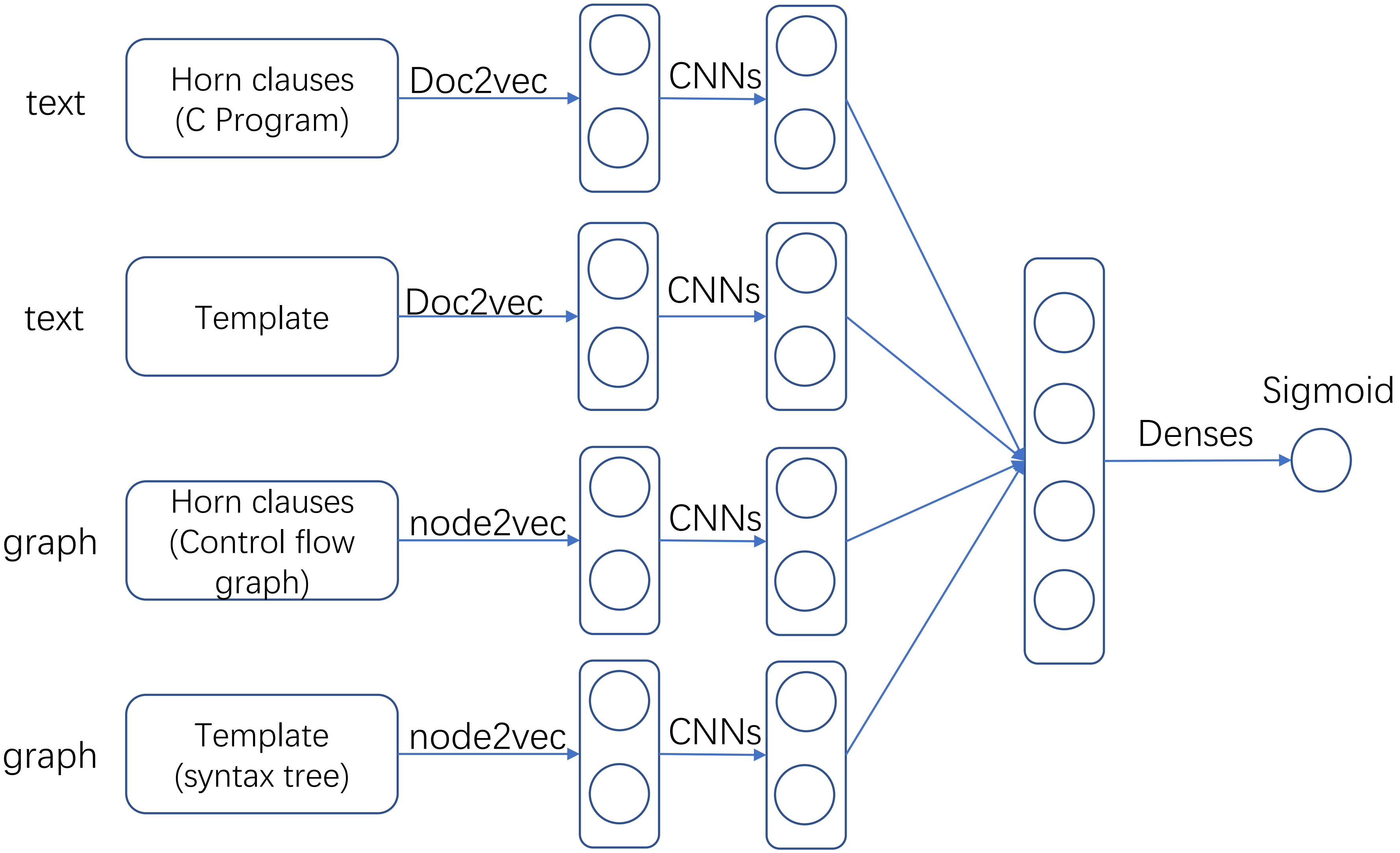
1. implied by A, i.e. $A \rightarrow I$.
2. inconsistent with B, i.e. $I \wedge B \rightarrow false$ and
3. expressed over the common variables of A and B.



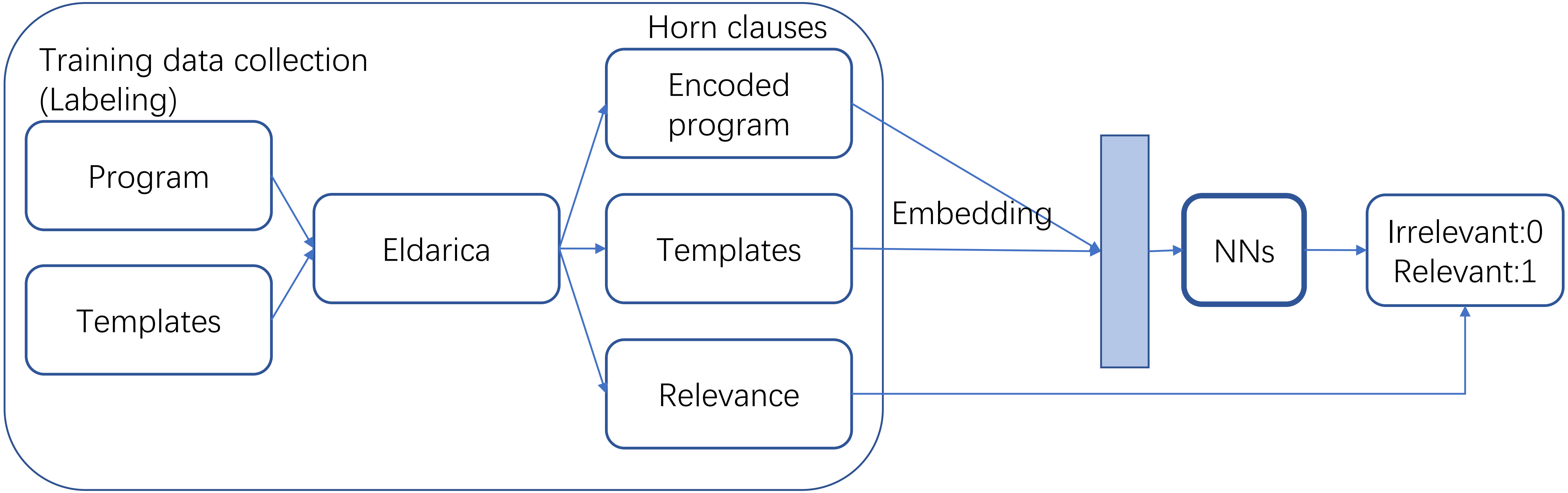
CEGAR framework

Craig interpolation

Abstract Interpolation [Gui16]


text
Horn clauses
(C Program)
Doc2vec
CNNs
text
Template
Doc2vec
CNNs
graph
Horn clauses
(Control flow
graph)
node2vec
CNNs
graph
Template
(syntax tree)
node2vec
CNNs
Denses
Sigmoid

Template selection (training process)



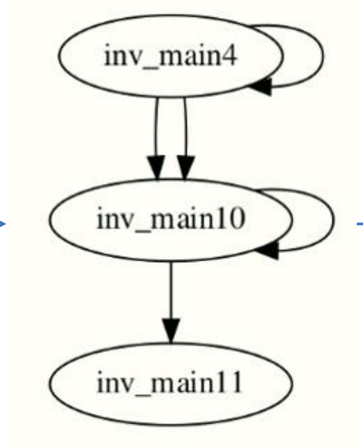
```c
void errorFn() {assert(0);}

int unknown1();
int unknown2();
int unknown3();
int unknown4();

int main()
{
	int i = 1;
	int j = 0;
	int z = i-j;
	int x = 0;
	int y = 0;
	int w = 0;

	while(unknown2())
	{
		z+=x+y+w;
		y++;
		if(z%2==1)
		  x++;
		w+=2;
	}

	if(!(x==y))
		errorFn();
}
```

C program

Eldarica

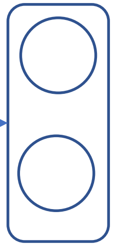
```
break_single_merged_safe.c.annot.c.horn
    main4(x, 0).
    main6(_0, _1) :- main4(_0, _1), 10 >= _1.
    main5(_0, _1) :- main4(_0, _1), _1 >= 11.
    main8(_0, _1) :- main6(_0, _1), _1 = _0.
    main9(_0, _1) :- main6(_0, _1), _1 != _0.
    main5(_0, _1) :- main8(_0, _1).
    main7(_0, _1) :- main9(_0, _1).
    main4(_0, _1 + 1) :- main7(_0, _1).
    main10(_0, _1, 0) :- main5(_0, _1).
    main12(_0, _1, _2) :- main10(_0, _1, _2), 10 >= _2 & _2 != _0.
    main11(_0, _1, _2) :- main10(_0, _1, _2), _2 >= 11 | _2 = _0.
    main10(_0, _1, _2 + 1) :- main12(_0, _1, _2).
    main1 :- main11(_0, _1, _2).
    main0(__result) :- main1.

Assertions:
  false :- main11(_0, _1, _2), _1 != _2.
```

Horn clauses



Graph representation

Node2vec

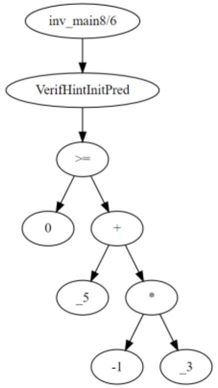


Fixed-length vector

---

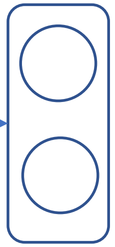
inv_main8
VerifHintInitPred(((_5 + -1 * _3) >= 0))

Template



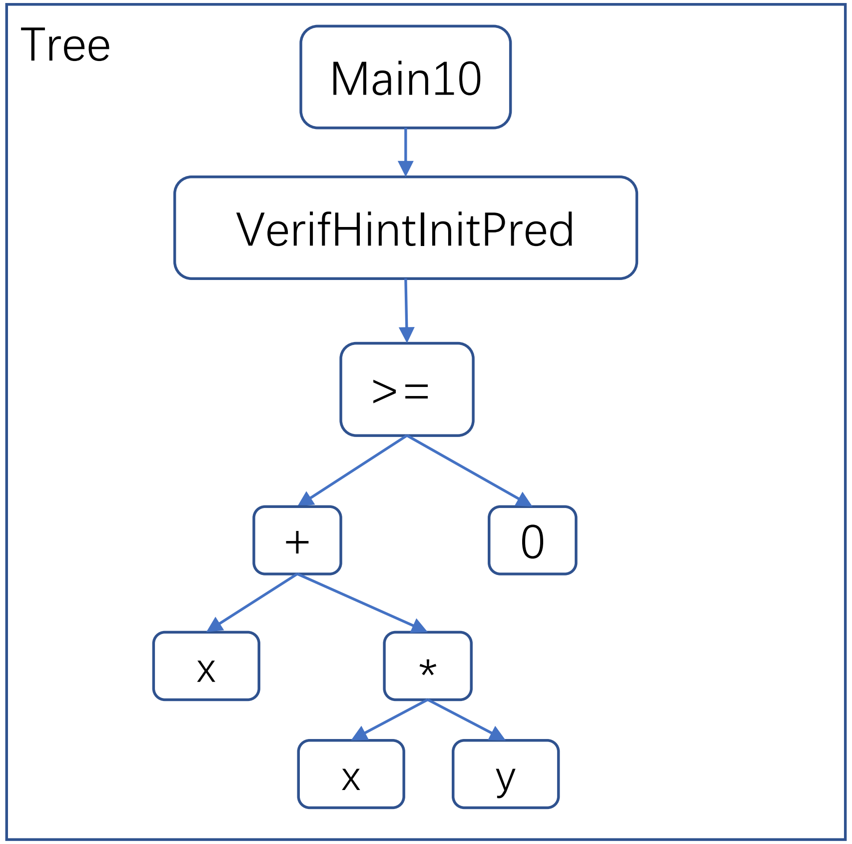
Node2vec

Fixed-length vector



Template

Main10
VerifHintInitPred ((x + -1 * y) >= 0)



Program + Hints

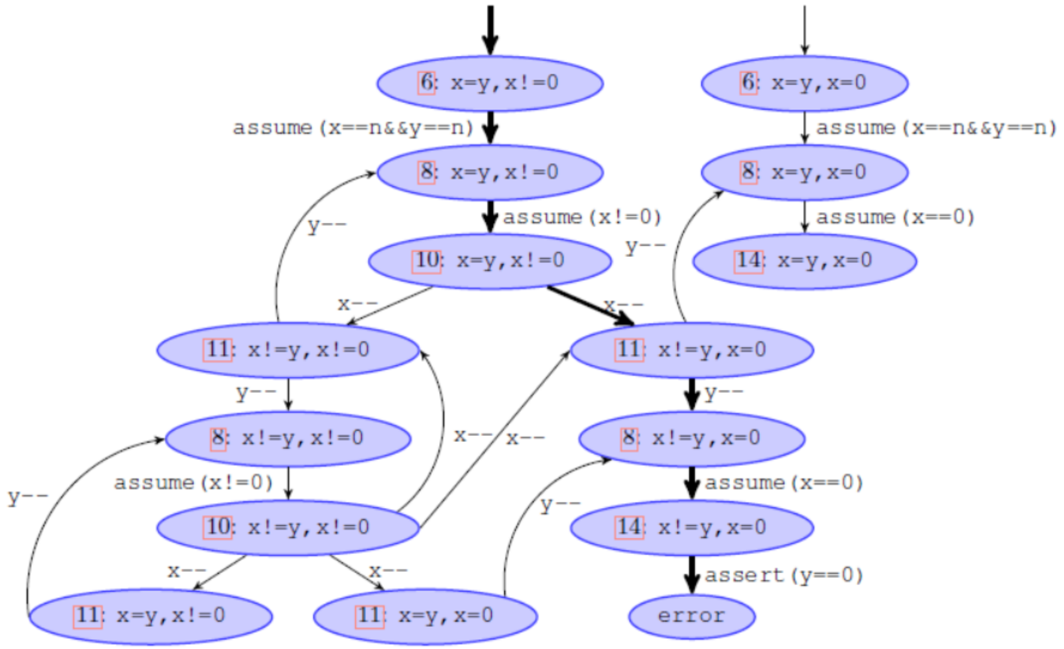
```
# 1 "/tmp/tmp.iQTEZnXQj2.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 1 "<command-line>" 2
# 1 "/tmp/tmp.iQTEZnXQj2.c"
void errorFn(){assert(0);}

int unknown1();
int unknown2();
int unknown3();
int unknown4();

int main()
{
 int /*@  predicates{i==1} predicates_tpl{0==0} @*/ i = 1;
 int /*@  predicates{j==0} @*/ j = 0;
 int z = i-j;
 int /*@  predicates{x<=z,x>=z} terms_tpl{x-z} @*/ x = 0;
 int /*@  predicates{y<=x,y<=z,y>=x,y>=z} terms_tpl{y-x,y-z} @*/ y = 0;
 int /*@  predicates{w<=x,w<=y,w<=z,w>=x,w>=y,w>=z} terms_tpl{w-2*x,w-2*y,w-z} @*/ w = 0;

 while(unknown2())
 {
  z+=x+y+w;
  y++;
  if(z%2==1)
    x++;
  w+=2;
 }

 if(!(x==y))
  errorFn();
}
```

Eldarica

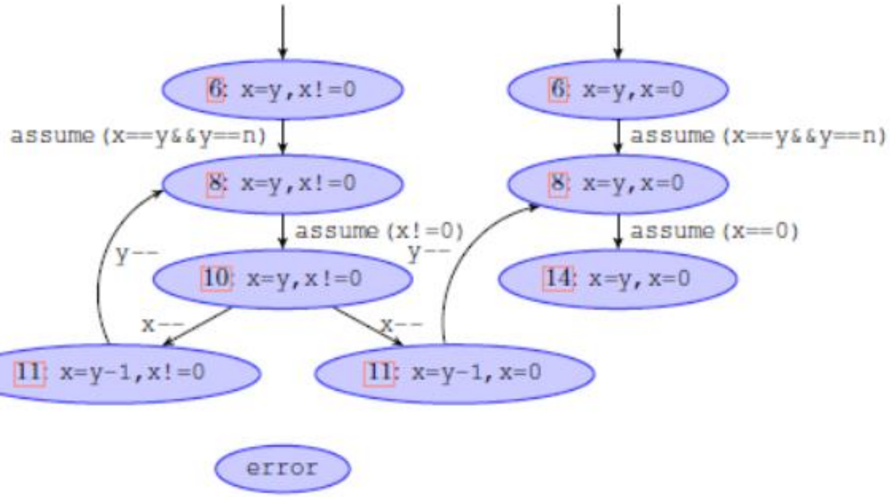
```
simpHints Hints:
inv_main9/4
VerifHintTplPred((0 = 0),1)
VerifHintTplEqTerm((_1 + -1 * _0),1)
VerifHintTplEqTerm((_2 + -1 * _0),1)
VerifHintTplEqTerm((_2 + -1 * _1),1)
VerifHintTplEqTerm((_3 + -1 * _0),1)
VerifHintTplEqTerm((_3 + -1 * 2 * _2),1)
VerifHintTplEqTerm((_3 + -1 * 2 * _1),1)
VerifHintTplEqTerm(_0,10000)
VerifHintTplEqTerm(_1,10000)
VerifHintTplEqTerm(_2,10000)
VerifHintTplEqTerm(_3,10000)
inv_main10/5
VerifHintTplPred((0 = 0),1)
VerifHintTplEqTerm((_1 + -1 * _0),1)
VerifHintTplEqTerm((_2 + -1 * _0),1)
VerifHintTplEqTerm((_2 + -1 * _1),1)
VerifHintTplEqTerm((_3 + -1 * _0),1)
VerifHintTplEqTerm((_3 + -1 * 2 * _2),1)
VerifHintTplEqTerm((_3 + -1 * 2 * _1),1)
VerifHintTplEqTerm(_0,10000)
VerifHintTplEqTerm(_1,10000)
VerifHintTplEqTerm(_2,10000)
VerifHintTplEqTerm(_3,10000)
VerifHintTplEqTerm(_4,10000)
inv_main19/4
VerifHintTplPred((0 = 0),1)
VerifHintTplEqTerm((_1 + -1 * _0),1)
VerifHintTplEqTerm((_2 + -1 * _0),1)
VerifHintTplEqTerm((_2 + -1 * _1),1)
VerifHintTplEqTerm((_3 + -1 * _0),1)
VerifHintTplEqTerm((_3 + -1 * 2 * _2),1)
VerifHintTplEqTerm((_3 + -1 * 2 * _1),1)
VerifHintTplEqTerm(_0,10000)
VerifHintTplEqTerm(_1,10000)
VerifHintTplEqTerm(_2,10000)
VerifHintTplEqTerm(_3,10000)

Optimized Hints:
!@@@@
inv_main9/4
VerifHintTplEqTerm(_0,10000)
VerifHintTplEqTerm(_1,10000)
VerifHintTplEqTerm(_2,10000)
VerifHintTplEqTerm(_3,10000)
@@@@!
```

```
extern int n;

void main()
{
  int x, y;
  assume(x==n && y==n);

  while (x!=0)
  {
    x--;
    y--;
  }

  assert(y==0);
}
```

Source code



Abstract labelled transition system
$P_1 = \{x = y, x = 0\}$



Abstract labelled transition system
$P_2 = \{x = y, x = 0, \boldsymbol{x = y - 1}\}$

Counterexample (path): $x \overset{6}{=} n \wedge y \overset{6}{=} n \wedge x \overset{8}{\neq} 0 \wedge x' \overset{10}{=} x - 1 \wedge y' \overset{11}{=} y - 1 \wedge x' \overset{8}{=} 0 \wedge y' \overset{14}{!=} 0.$

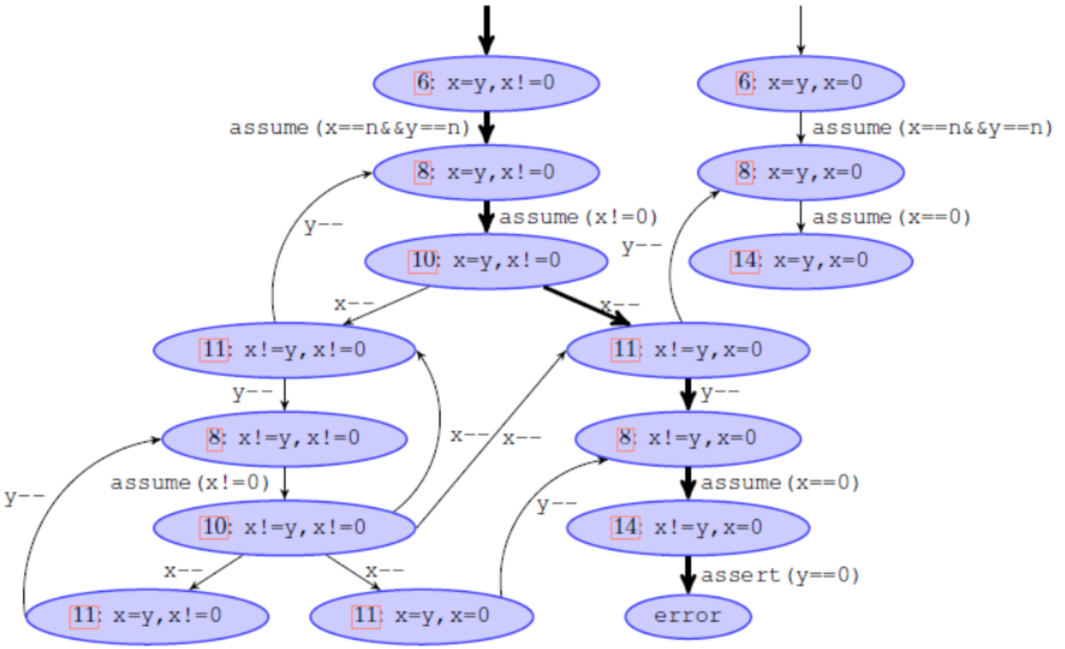
Separated path (query): $A = (x = n \wedge y = n \wedge x \mathrel{!=} 0 \wedge x' = x - 1)$ and
$B = (y' = y - 1 \wedge x' = 0 \wedge y' \mathrel{!=} 0).$

Separated path (query): $I = (x' = y - 1)$

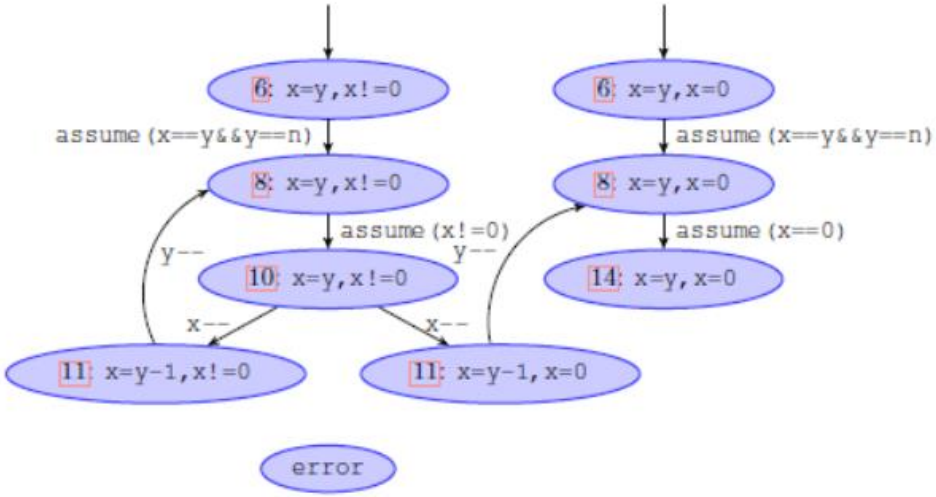
1. $x = n \wedge y = n \wedge x' = x - 1 \rightarrow \boldsymbol{x' = y - 1}$ and
2. $\boldsymbol{x' = y - 1} \wedge y' = y - 1 \wedge x' = 0 \wedge y' \mathrel{!=} 0 \rightarrow \mathit{false}$ and
3. I uses the common variables of A and B.

```
1 extern int n;
2
3 void main()
4 {
5   int x, y;
6   assume(x==n && y==n);
7
8   while (x!=0)
9   {
10    x--;
11    y--;
12  }
13
14  assert(y==0);
15 }
```

Source code



Abstract labelled transition system $P_1 = \{x = y, x = 0\}$



Abstract labelled transition system $P_2 = \{x = y, x = 0, \boldsymbol{x = y - 1}\}$

Counterexample (path): $x \overset{6}{=} n \wedge y \overset{6}{=} n \wedge x \overset{8}{\neq} 0 \wedge x' \overset{10}{=} x - 1 \wedge y' \overset{11}{=} y - 1 \wedge x' \overset{8}{=} 0 \wedge y' \overset{14}{!=} 0.$

Separated path (query): $A = (x = n \wedge y = n \wedge x \mathrel{!=} 0 \wedge x' = x - 1)$ and $B = (y' = y - 1 \wedge x' = 0 \wedge y' \mathrel{!=} 0)$.

Separated path (query): $I = (x' = y - 1)$

1. $x = n \wedge y = n \wedge x' = x - 1 \rightarrow \boldsymbol{x' = y - 1}$ and
2. $\boldsymbol{x' = y - 1} \wedge y' = y - 1 \wedge x' = 0 \wedge y' \mathrel{!=} 0 \rightarrow false$ and
3. I uses the common variables of A and B.

Abstract Interpolation

```
i = 0; x = j;
while (i<50) {i++; x++;}
if (j == 0) assert (x >= 50);
```

Source code

$i_0 \doteq 0 \wedge x_0 \doteq j \wedge i_0 < 50 \wedge i_1 \doteq i_0+1 \wedge x_1 \doteq x_0+1$
$\wedge\, i_1 \geq 50 \wedge j \doteq 0 \wedge x_1 < 50$

Original query

Interpolation: $I_1 = (i_1 \leq 1)$

Template1: $x_1 - i_1$
Template2: j

$(i_0 \doteq 0 \wedge x_0 \doteq j' \wedge i_0 < 50 \wedge i_1' \doteq i_0+1 \wedge x_1' \doteq x_0+1 \wedge x_1' - i_1' \doteq x_1 - i_1 \wedge j' \doteq j\,)$
$\wedge\, (\,x_1 - i_1 \doteq x_1'' - i_1'' \wedge j \doteq j'' \wedge i_1'' \geq 50 \wedge j'' \doteq 0 \wedge x_1'' < 50)$

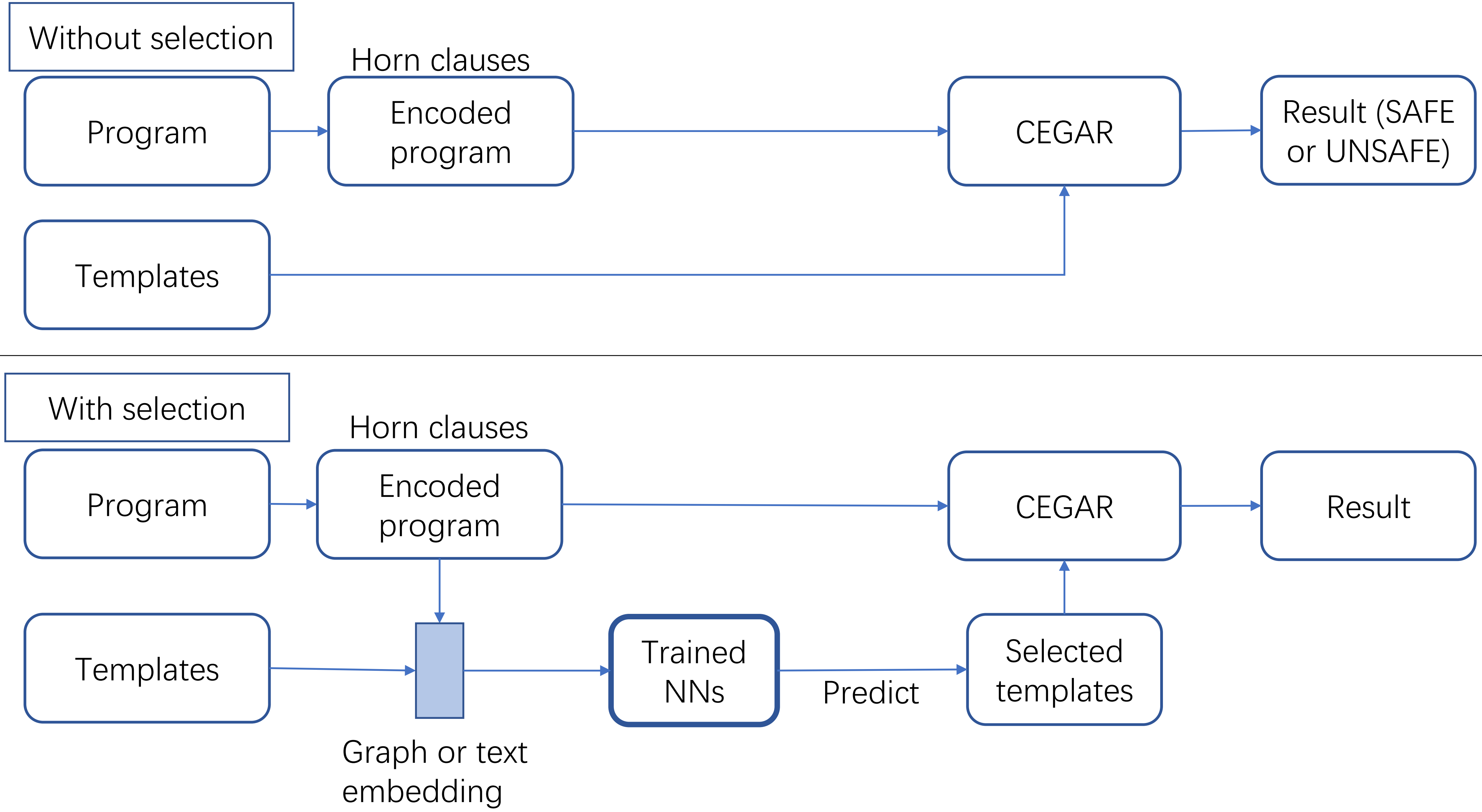
Modified query

Interpolation: $I_2 = (x_1 \geq i + j)$

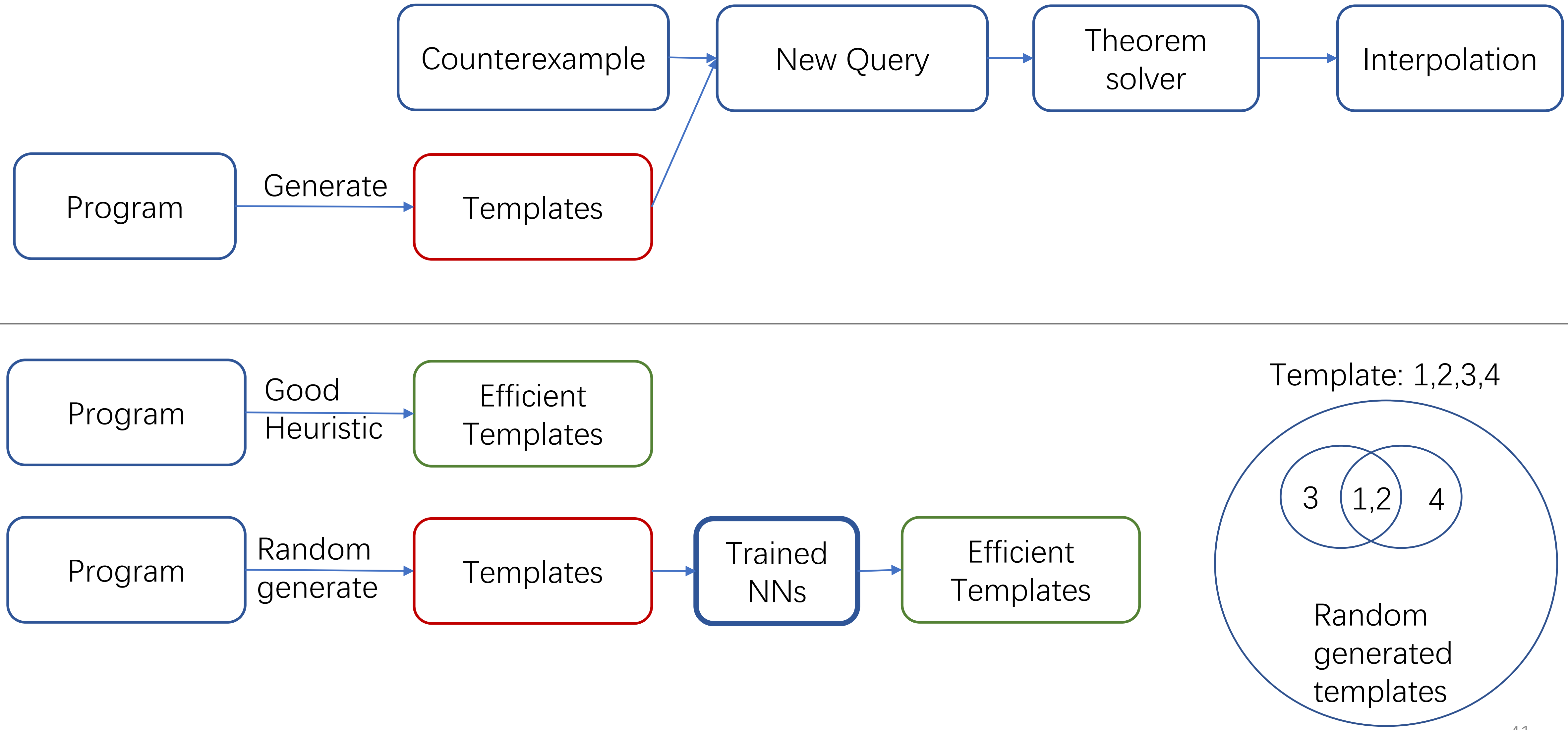
Preliminary results

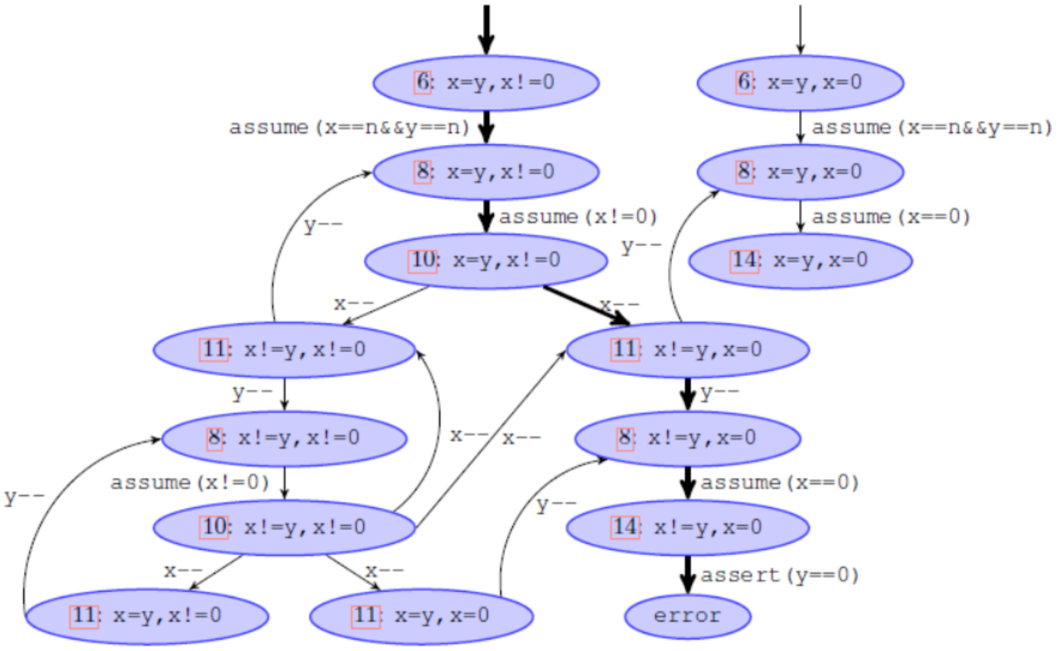
- 80% accuracy in binary classification task (46 programs and 11900 lines of training data).
  - Eliminate large part of redundant templates
  - Give rankings to templates

Template selection (predicting process)

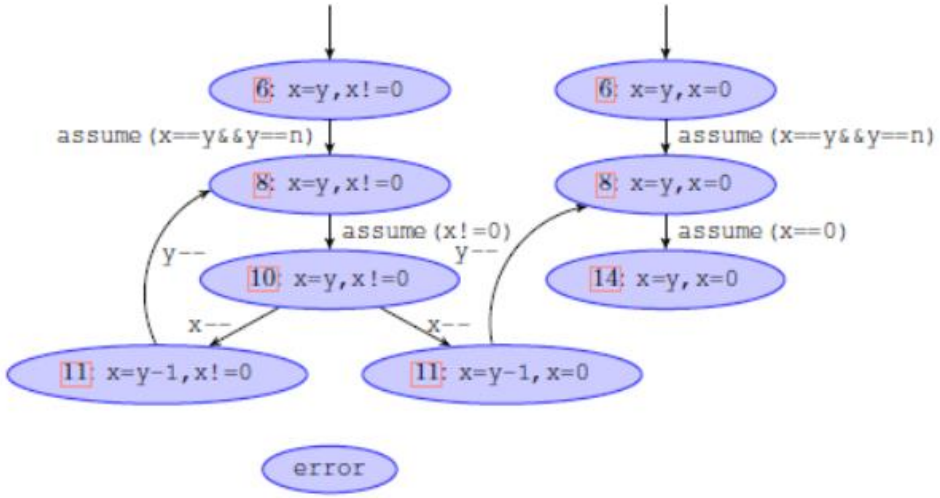


Select random generated templates

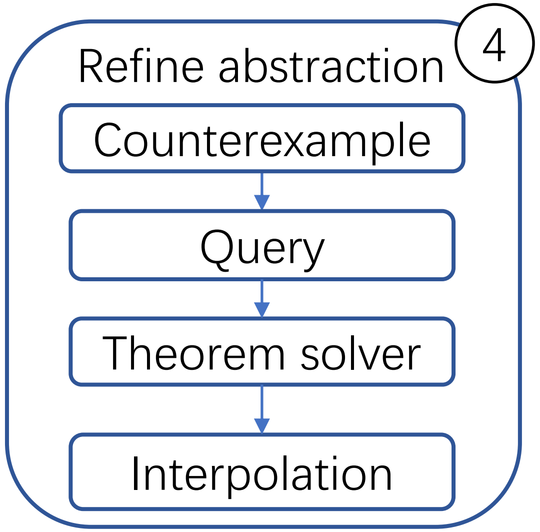




Abstract labelled transition system
$P_1 = \{x = y, x = 0\}$



Abstract labelled transition system
$P_2 = \{x = y, x = 0, \boldsymbol{x = y - 1}\}$



Counterexample (path): $x \overset{6}{=} n \wedge y \overset{6}{=} n \wedge x \overset{8}{\neq} 0 \wedge x' \overset{10}{=} x - 1 \wedge y' \overset{11}{=} y - 1 \wedge x' \overset{8}{=} 0 \wedge y' \overset{14}{!=} 0.$

Separated path (query): $A = (x = n \wedge y = n \wedge x \mathrel{!=} 0 \wedge x' = x - 1)$ and
$B = (y' = y - 1 \wedge x' = 0 \wedge y' \mathrel{!=} 0)$.

Interpolation (new abstraction): $I = (x' = y - 1)$

1. $x = n \wedge y = n \wedge x' = x - 1 \rightarrow \boldsymbol{x' = y - 1}$ and
2. $\boldsymbol{x' = y - 1} \wedge y' = y - 1 \wedge x' = 0 \wedge y' \mathrel{!=} 0 \rightarrow false$ and
3. I uses the common variables of A and B.