Abstract Conflict Driven Clause Learning

Leopold Haller

Joint work with
Vijay D'Silva, Alberto Griggio, Michael Tautschnig, Daniel Kroening

Anglo-EU Translation Guide

What the British say	What the British mean	What others understand
That's not bad.	That's good	Could be better.
Oh, by the way ...	The primary purpose of our discussion is ...	It's not very important, but ...

“Everything is Abstract Interpretation ...”

Abstract Interpretation-Everyone Else Translation Guide		
What an Abs. Int. person says	**What they might mean**	**What others understand**
Isn’t this an instance of abstract interpretation?	I think there is a simple top-down characterisation of this in the language of algebra, fixed points and abstraction.	This is a trivial consequence of abstract interpretation.
Technique X computes an abstract fixed point.	There is a view of X that allows for the application of a rich body of results.	Details are unimportant.

“Everything is Abstract Interpretation ...”

Abstract Interpretation-Everyone Else Translation Guide

What an Abs. Int. person says	What they might mean	What others understand
Isn’t this an instance of abstract interpretation?	I think there is a simple top-down characterisation of this in the language of algebra, fixed points and abstraction.	This is a trivial consequence of abstract interpretation.
Technique X computes an abstract fixed point.	There is a view of X that allows for the application of a rich body of results.	Details are unimportant.

... including SAT solvers

(Satisfiability Solvers are Static Analysers. D’Silva, Haller, Kroening, SAS 2012)

Propositional Satisfiability (SAT)

literal

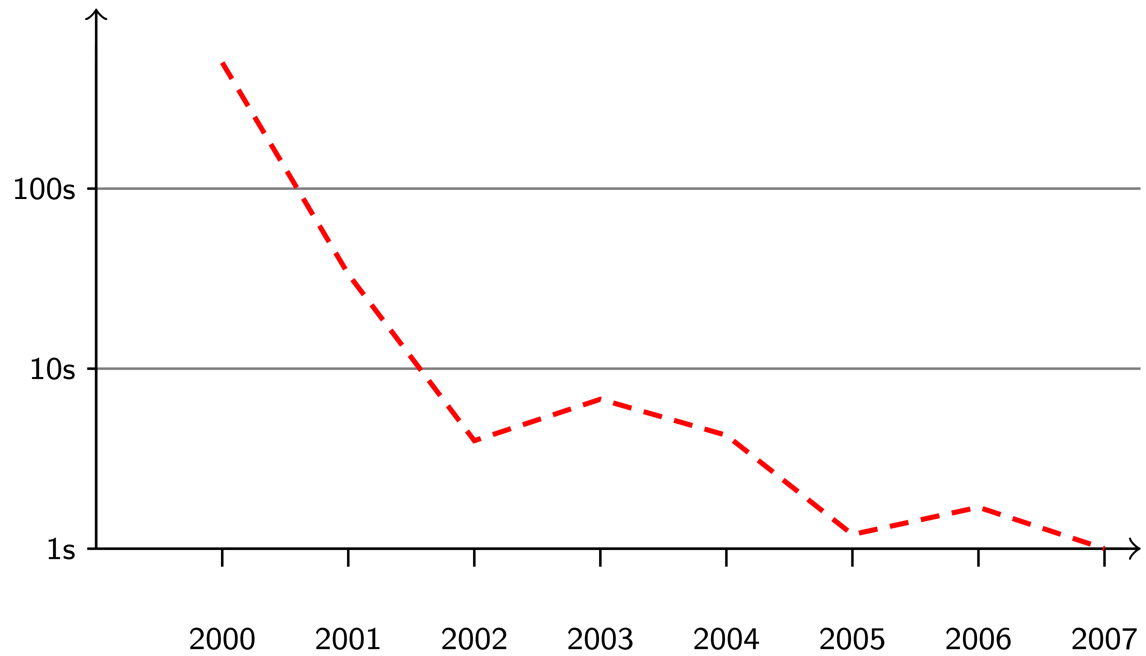
$$\varphi = (p \vee \neg q) \wedge \ldots \wedge (\neg r \vee w \vee q)$$

clause

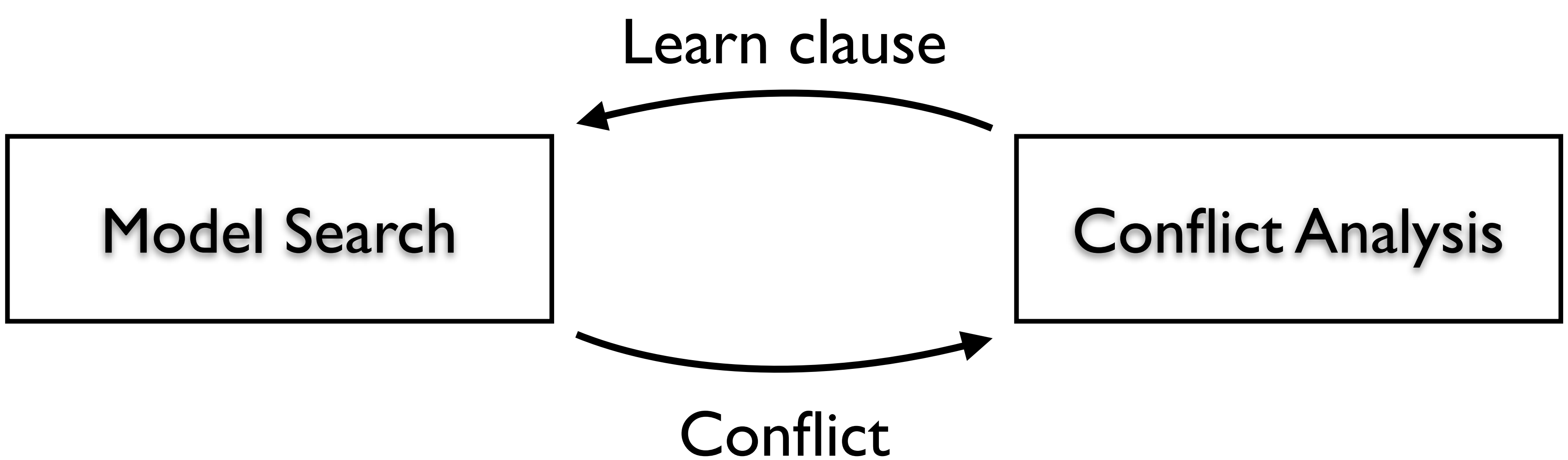
Given a propositional formula φ, is there a propositional truth assignment σ such that $\sigma \models \varphi$.

- Solvers are based on Conflict Driven Clause Learning (CDCL)
- Basis of modern Satisfiability Module Theory (SMT) solvers
- Critical components of program verification techniques

CDCL



(Malik and Zhang, 2009)



Work on CDCL has resulted in an exponential decrease in runtimes.

Can we lift this success to other domains?

SMT via DPLL(T)

Solve satisfiability for (QF) first order formula with background theory

$$(\underbrace{x + y \leq 3}_{p} \vee \underbrace{2x - y \geq 1}_{q}) \wedge (\underbrace{x = 5}_{r} \vee \underbrace{y = x}_{s})$$

$$(p \vee q) \wedge (r \vee s)$$

$$\boxed{\text{Propositional CDCL}} \longleftrightarrow \boxed{\text{Theory Solver}}$$

CDCL enumerates candidate propositional truth assignments,
theory solver checks consistency.

DPLL(T) is a <u>mathematical recipe</u> and <u>implementation framework</u> for building SMT decision procedures!

SMT via DPLL(T)

DPLL(T) can be viewed to partition the space of potential models using the structure of the formula.
Measures have to be taken to avoid enumeration behaviour.

$$(x = 0 \vee x = 2 \vee x = 4 \vee \ldots \vee x = 2k) \wedge$$
$$(y = 0 \vee y = 2 \vee y = 4 \vee \ldots \vee y = 2k) \wedge$$
$$(x + y = 2c + 1)$$

x\y	0	2	4	...
0				...
2				...
4				...
...	...	...	...	...

DPLL(T) explores truth assignments to predicates

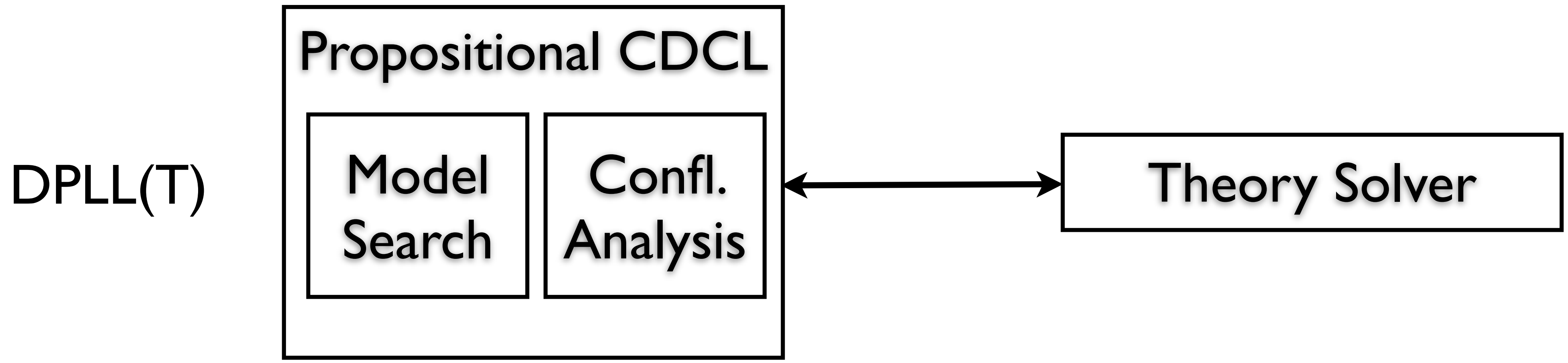
x\y	even	odd
even		
odd		

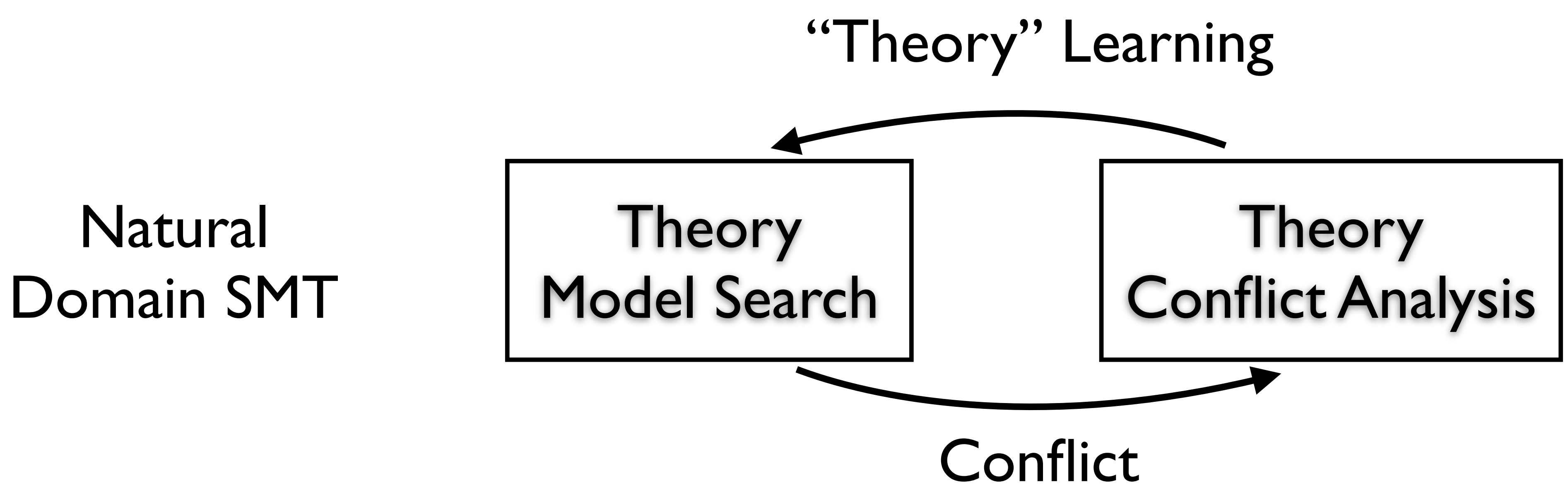
Full even / odd partitioning

CDCL SMT

Natural Domain SMT

Natural Domain SMT





Abstract Interpretation

Abstract Interpretation by Example: Intervals

Track possible range for variable

```
int a = 5;
int b;
```

```
if(*)
  b = 3;
else
  b = -3;
```

```
a += b;
```

```
assert(a == 0);
```

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with
strongest postcondition

```
int a = 5;
int b;
```

```
if(*)
  b = 3;
else
  b = -3;
```

```
a += b;
```

```
assert(a == 0);
```

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

$\top$

```
int a = 5;
int b;
```

```
if(*)
  b = 3;
else
  b = -3;
```

```
a += b;
```

```
assert(a == 0);
```

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

$\top$

$a \mapsto [5,5]$

```
int a = 5;
int b;
```

```
if(*)
  b = 3;
else
  b = -3;
```

```
a += b;
```

```
assert(a == 0);
```

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

$\top$

$a \mapsto [5, 5]$	`int a = 5;` `int b;`
Imprecise OA: $a \mapsto [5, 5], b \mapsto [-3, 3]$	`if(*)` `  b = 3;` `else` `  b = -3;`
	`a += b;`
	`assert(a == 0);`

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

$\top$

$a \mapsto [5, 5]$	`int a = 5;` `int b;`
Imprecise OA: $a \mapsto [5, 5], b \mapsto [-3, 3]$	`if(*)` `  b = 3;` `else` `  b = -3;`
$a \mapsto [2, 8], b \mapsto [-3, 3]$	`a += b;`
	`assert(a == 0);`

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

Underapproximate Analysis with weakest precondition

$\top$

$a \mapsto [5, 5]$

```
int a = 5;
int b;
```

Imprecise OA:

$a \mapsto [5, 5], b \mapsto [-3, 3]$

```
if(*)
  b = 3;
else
  b = -3;
```

$a \mapsto [2, 8], b \mapsto [-3, 3]$

```
a += b;
```

```
assert(a == 0);
```

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

Underapproximate Analysis with weakest precondition

$\top$

```
int a = 5;
int b;
```

$a \mapsto [5,5]$

```
if(*)
  b = 3;
else
  b = -3;
```

Imprecise OA:

$a \mapsto [5,5], b \mapsto [-3,3]$

```
a += b;
```

$a \mapsto [2,8], b \mapsto [-3,3]$

$\{a \mapsto [-\infty,-1], a \mapsto [1,\infty]\}$

```
assert(a == 0);
```

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition		Underapproximate Analysis with weakest precondition
$\top$		
$a \mapsto [5,5]$	`int a = 5;` `int b;`	
Imprecise OA: $a \mapsto [5,5], b \mapsto [-3,3]$	`if(*)` `b = 3;` `else` `b = -3;`	UA "guess": $\{(a \mapsto [4,\infty], b \mapsto [-3,3])\}$
$a \mapsto [2,8], b \mapsto [-3,3]$	`a += b;`	$\{a \mapsto [-\infty,-1], a \mapsto [1,\infty]\}$
	`assert(a == 0);`	

```
int a = 5;
int b;
if(*)
  b = 3;
else
  b = -3;
a += b;
assert(a == 0);
```

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition		Underapproximate Analysis with weakest precondition
$\top$		
$a \mapsto [5, 5]$	`int a = 5;` `int b;`	$\{a \mapsto [4, \infty]\}$
Imprecise OA: $a \mapsto [5, 5], b \mapsto [-3, 3]$	`if(*)` `b = 3;` `else` `b = -3;`	UA “guess”: $\{(a \mapsto [4, \infty], b \mapsto [-3, 3])\}$
$a \mapsto [2, 8], b \mapsto [-3, 3]$	`a += b;`	$\{a \mapsto [-\infty, -1], a \mapsto [1, \infty]\}$
	`assert(a == 0);`	

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition		Underapproximate Analysis with weakest precondition
$\top$		$\top$
$a \mapsto [5,5]$	`int a = 5;` `int b;`	$\{a \mapsto [4,\infty]\}$
Imprecise OA: $a \mapsto [5,5], b \mapsto [-3,3]$	`if(*)` `b = 3;` `else` `b = -3;`	UA "guess": $\{(a \mapsto [4,\infty], b \mapsto [-3,3])\}$
$a \mapsto [2,8], b \mapsto [-3,3]$	`a += b;`	$\{a \mapsto [-\infty,-1], a \mapsto [1,\infty]\}$
	`assert(a == 0);`	

Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition		Underapproximate Analysis with weakest precondition
$\top$		$\top$
$a \mapsto [5,5]$	`int a = 5;` `int b;`	$\{a \mapsto [4,\infty]\}$
Imprecise OA: $a \mapsto [5,5], b \mapsto [-3,3]$	`if(*)` `b = 3;` `else` `b = -3;`	UA "guess": $\{(a \mapsto [4,\infty], b \mapsto [-3,3])\}$
$a \mapsto [2,8], b \mapsto [-3,3]$	`a += b;`	$\{a \mapsto [-\infty,-1], a \mapsto [1,\infty]\}$
	`assert(a == 0);`	

Sound, but incomplete

Abstract Interpretation

Concrete Lattice

$(\wp(States), \subseteq, \cap, \cup)$

Galois connection:

$$\underset{\alpha}{\overset{\gamma}{\leftrightarrows}}$$

Abstract Lattice

$(Intervals, \sqsubseteq, \sqcap, \sqcup)$

Abstract Interpretation

Concrete Lattice

$$(\wp(States), \subseteq, \cap, \cup)$$

Galois connection:

$$\xrightleftharpoons[\alpha]{\gamma}$$

Abstract Lattice

$$(Intervals, \sqsubseteq, \sqcap, \sqcup)$$

Abstraction and concretisation function

$$\alpha(\{x \mapsto 3, x \mapsto 1, x \mapsto 9\}) = x \mapsto [1, 9]$$

$$\gamma(x \mapsto [4, 6]) = \{x \mapsto 4, x \mapsto 5, x \mapsto 6\}$$

Abstract Interpretation

Concrete Lattice

$$(\wp(States), \subseteq, \cap, \cup)$$

Galois connection:

$$\underset{\alpha}{\overset{\gamma}{\leftrightarrows}}$$

Abstract Lattice

$$(Intervals, \sqsubseteq, \sqcap, \sqcup)$$

Abstraction and concretisation function

$$\alpha(\{x \mapsto 3, x \mapsto 1, x \mapsto 9\}) = x \mapsto [1, 9]$$

$$\gamma(x \mapsto [4, 6]) = \{x \mapsto 4, x \mapsto 5, x \mapsto 6\}$$

Concrete transformer $\quad post : \wp(States) \to \wp(States)$

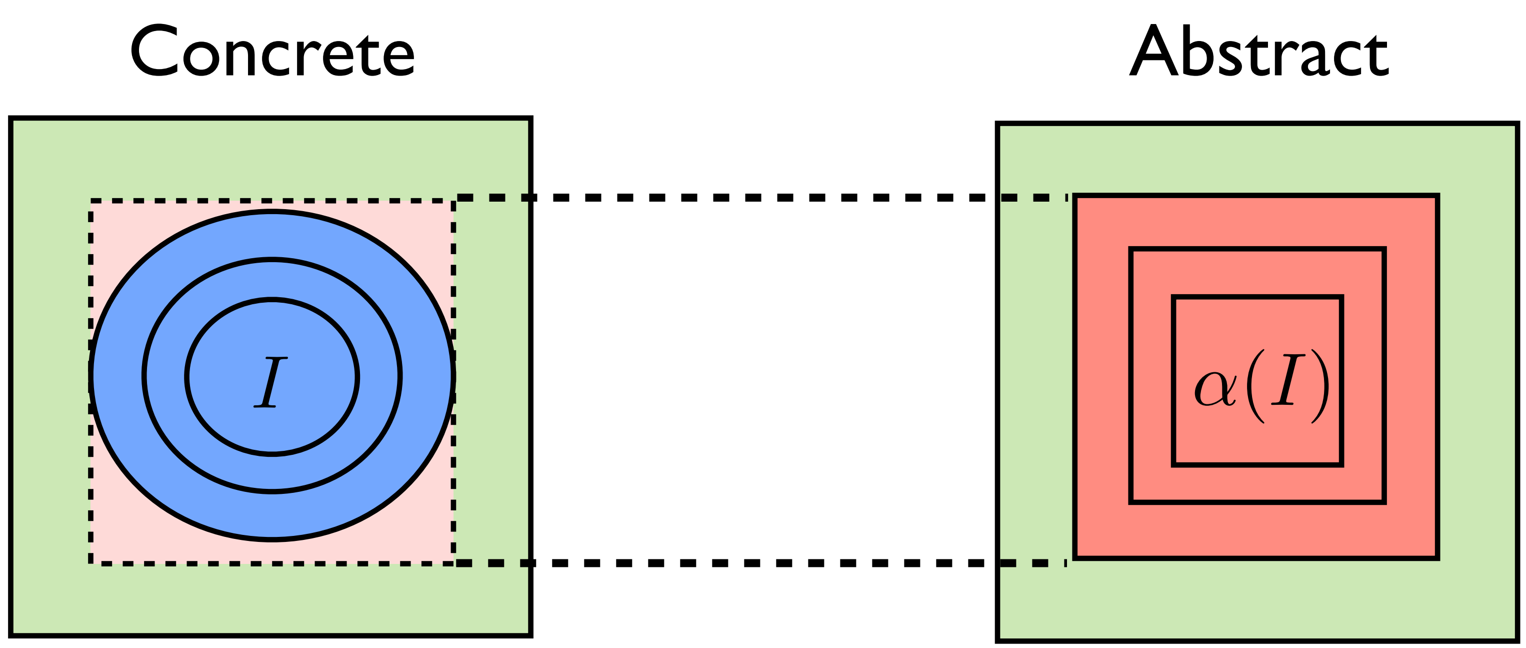
Sound abstr. transformer $\quad \hat{post} : Intervals \to Intervals$

$$post \circ \gamma \subseteq \gamma \circ \hat{post}$$

Approximating Fixed Points

Fixed points can be computed in the abstract

$$\mathsf{lfp}\ X.\ I \cup post(X) \subseteq \gamma(\mathsf{lfp}\ X.\alpha(I) \sqcup \hat{post}(X))$$



Accelerating Fixed Point Computations

```
x = 0;
while(x < 1000)
  x++;
```

Fixed point computations might take a long time (or fail to terminate):

$F_0 : x \mapsto [0, 0] \qquad F_1 : x \mapsto [0, 1] \qquad F_2 : x \mapsto [0, 2] \qquad F_3 : x \mapsto [0, 3] \qquad \ldots$

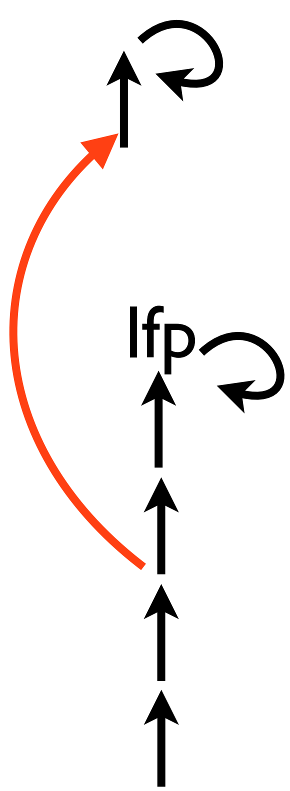
Accelerating Fixed Point Computations

```
x = 0;
while(x < 1000)
  x++;
```

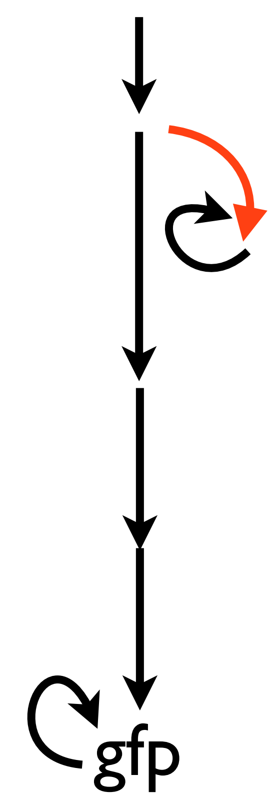
Fixed point computations might take a long time (or fail to terminate):

$$F_0 : x \mapsto [0,0] \qquad F_1 : x \mapsto [0,1] \qquad F_2 : x \mapsto [0,2] \qquad F_3 : x \mapsto [0,3] \qquad \ldots$$

Widening
(jumps above least fixed point)



Narrowing
(stay above greatest fixed point)



Abstract Interpretation

↓

Interpreting Logic

Abstractly Interpreting Logic

Check satisfiability of the following formula

$$\varphi = p \wedge (\neg p \vee q) \wedge (\neg p \vee \neg q)$$

Prove the following program safe

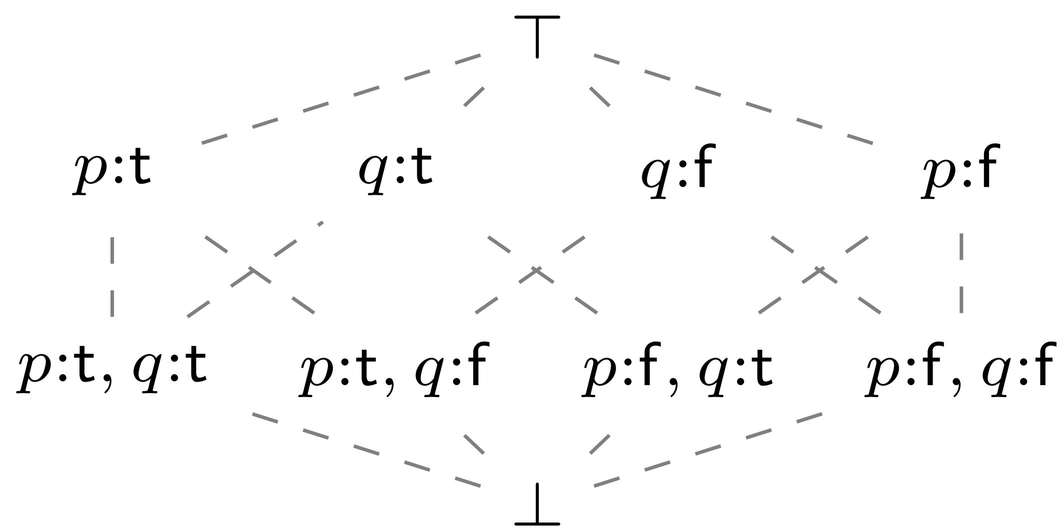
```
int main()
{

  if(p)
    if(!p || q)
      if(!p || !q)
        assert(false);

}
```

Abstractly Interpreting Logic

Constants analysis



`int main()` `{`	$\top$
`  if(p)`	p:t
`    if(!p || q)`	p:t, q:t
`      if(!p || !q)`	$\perp$
`        assert(false);` `}`	

Abstractly Interpreting Logic

Set of formulae

$$Form$$

Set of structures

$$Struct$$

Semantic entailment relation

$$\models \quad \in \wp(Struct \times Form)$$

Concrete Domain

$$(\wp(Struct), \subseteq, \cap, \cup)$$

Abstractly Interpreting Logic

Set of formulae

$Form$

Set of structures

$Struct$

Semantic entailment relation

$\models \quad \in \wp(Struct \times Form)$

Concrete Domain

$(\wp(Struct), \subseteq, \cap, \cup)$

E.g., propositional logic:

$Lit = \{p, \neg p \mid p \in Props\}$

$Clauses = \wp(Lit)$

$Form = \wp(Clauses)$

$Struct = Props \to \{\mathsf{t}, \mathsf{f}\}$

$\sigma \models \varphi$ iff

$\forall C \in \varphi.\ \exists l \in C.\ \ (l = p \wedge \sigma(p) = \mathsf{t}) \vee (l = \neg p \wedge \sigma(p) = \mathsf{f})$

Abstract Satisfaction

Structure transformers;

$$mods_{\varphi}(S) = \{\sigma \mid \sigma \in S \wedge \sigma \models \varphi\} \qquad confs_{\varphi}(S) = \{\sigma \mid \sigma \in S \vee \sigma \not\models \varphi\}$$

$$mods_{\varphi} = post_{assume(\varphi)} \qquad confs_{\varphi} = \tilde{pre}_{assume(\varphi)}$$

Abstract Satisfaction

Structure transformers;

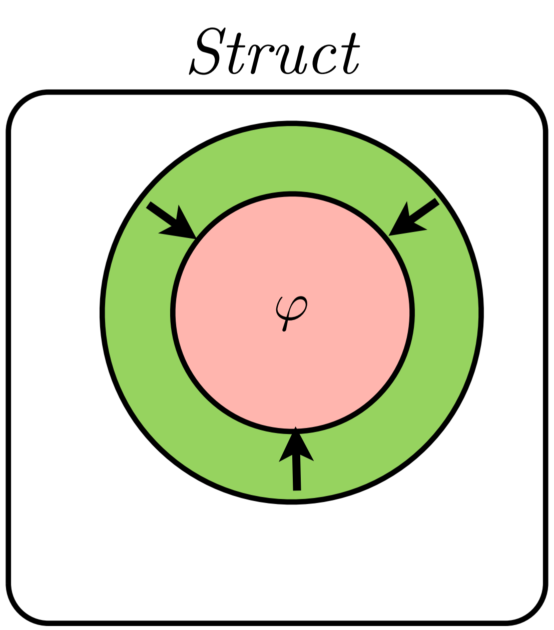
$$mods_\varphi(S) = \{\sigma \mid \sigma \in S \wedge \sigma \models \varphi\} \qquad confs_\varphi(S) = \{\sigma \mid \sigma \in S \vee \sigma \not\models \varphi\}$$

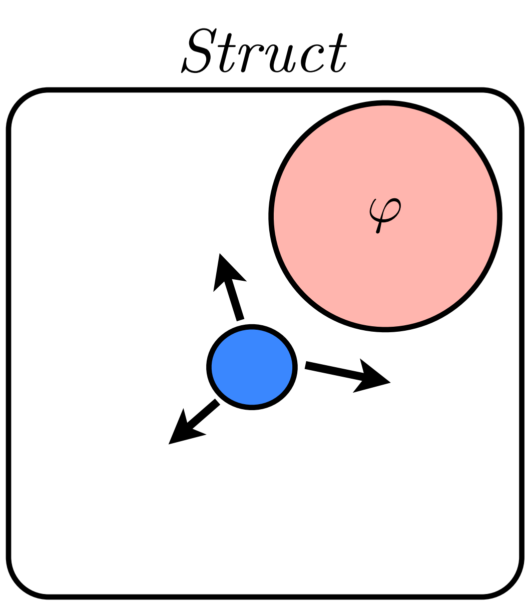
$$mods_\varphi = post_{assume(\varphi)} \qquad confs_\varphi = \tilde{pre}_{assume(\varphi)}$$

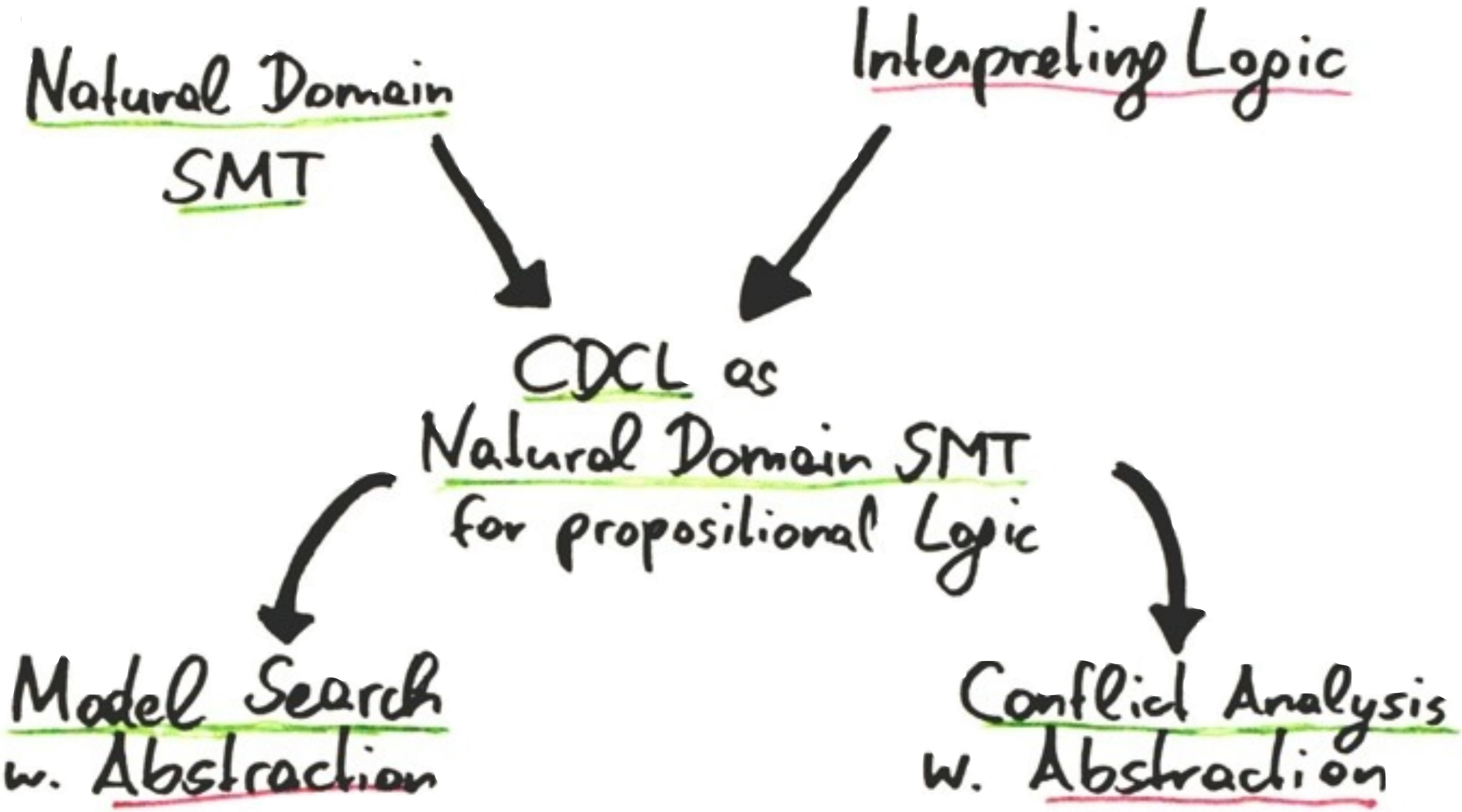
Overapproximation $amods_\varphi$ of $mods_\varphi$

Underapproximation $aconfs_\varphi$ of $confs_\varphi$

$$\boxed{\begin{array}{l} \mathsf{gfp}\ amods_\varphi = \bot \text{ or} \\ \mathsf{lfp}\ aconfs_\varphi = \top \end{array} \implies \varphi \text{ is unsatisfiable}}$$

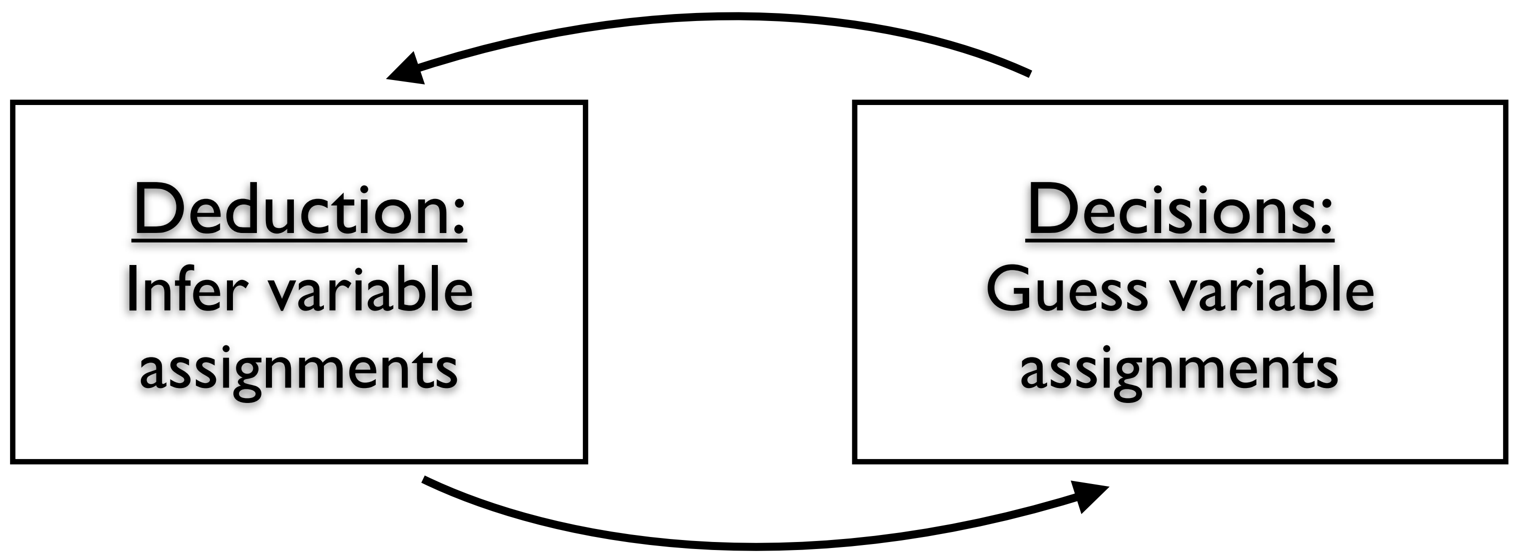





Natural Domain SMT
Interpreting Logic
CDCL as Natural Domain SMT for propositional Logic
Model Search w. Abstraction
Conflict Analysis w. Abstraction

Model Search

Find either a satisfying assignment or a conflicting partial assignment



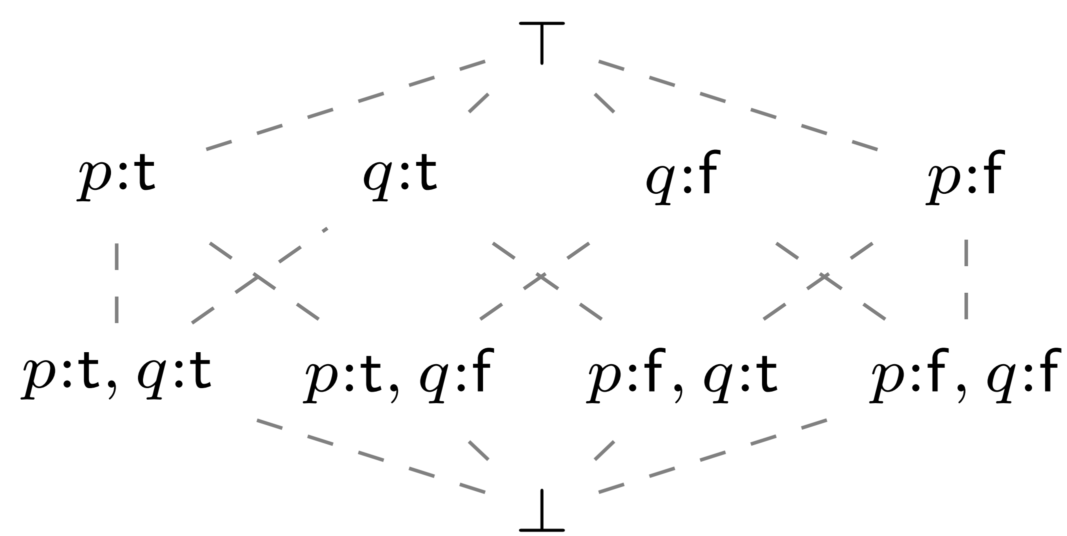
Partial Assignments are an Abstract Domain

```
#define  l_True   (lbool (( uint8_t )0))
#define  l_False  (lbool (( uint8_t )1))
#define  l_Undef  (lbool (( uint8_t )2))

class lbool {  [...]  };

class Solver {
   [...]
   // FALSE means solver is in a conflicting state
   bool           okay () const;
   vec<lbool>     assigns;  // The current assignments.
   // Enqueue a literal. Assumes value of literal is undefined.
```

$$\wp(\mathit{Props} \to \{\mathsf{t}, \mathsf{f}\}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}}$$



Deduction Computes a Greatest Fixed Point

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

$$\text{gfp } mods_{\varphi}$$

Deduction Computes a Greatest Fixed Point

$$\neg p \wedge (p \vee q) \wedge (\neg q \vee r)$$

$$\top$$

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

$$\text{gfp } mods_\varphi$$

Deduction Computes a Greatest Fixed Point

$$\neg p \wedge (p \vee q) \wedge (\neg q \vee r)$$

$$\begin{array}{c} \top \\ \big\downarrow \\ (p{:}\mathsf{f}) \end{array}$$

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

$$\text{gfp } \mathit{mods}_\varphi$$

Deduction Computes a Greatest Fixed Point

$$\neg p \wedge (p \vee q) \wedge (\neg q \vee r)$$

$$\begin{array}{c} \top \\ \downarrow \\ (p{:}\mathsf{f}) \\ \downarrow \\ (p{:}\mathsf{f}, q{:}\mathsf{t}) \end{array}$$

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

$$\mathsf{gfp}\ \ mods_{\varphi}$$

Deduction Computes a Greatest Fixed Point

$$\neg p \wedge (p \vee q) \wedge (\neg q \vee r)$$

$$\begin{tikzcd}
\top \arrow[d] \\
(p{:}\mathsf{f}) \arrow[d] \\
(p{:}\mathsf{f}, q{:}\mathsf{t}) \arrow[d] \\
(p{:}\mathsf{f}, q{:}\mathsf{t}, r{:}\mathsf{t}) \arrow[loop right]
\end{tikzcd}$$

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

$$\mathrm{gfp}\ \mathit{mods}_{\varphi}$$

Deduction Computes a Greatest Fixed Point

$$\neg p \wedge (p \vee q) \wedge (\neg q \vee r)$$

$$\begin{tikzcd}
\top \arrow[d] \\
(p{:}\mathsf{f}) \arrow[d] \\
(p{:}\mathsf{f}, q{:}\mathsf{t}) \arrow[d] \\
(p{:}\mathsf{f}, q{:}\mathsf{t}, r{:}\mathsf{t}) \arrow[loop right]
\end{tikzcd}$$

$$bcp(\pi) = \mathsf{gfp}\ X.\ unit(\pi \sqcap X)$$

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

$$\mathsf{gfp}\ mods_\varphi$$

Decision Making is Dual Widening

Once no more new facts can be deduced,
a solver heuristically picks a truth value for an unassigned variable

$$p \wedge (q \vee r) \wedge (q \vee \neg r)$$

$$\top$$

Deduction $\downarrow$

p:t

Decision Making is Dual Widening

Once no more new facts can be deduced,
a solver heuristically picks a truth value for an unassigned variable

$$p \wedge (q \vee r) \wedge (q \vee \neg r)$$

$\top$

Deduction

p:t

Decision q:f

p:t, q:f

Decision Making is Dual Widening

Once no more new facts can be deduced,
a solver heuristically picks a truth value for an unassigned variable

$$p \wedge (q \vee r) \wedge (q \vee \neg r)$$

$\top$

Deduction

p:t

Decision q:f

p:t, q:f

Deduction

p:t, q:f, r:t

$\perp$

Decision Making is Dual Widening

Once no more new facts can be deduced,
a solver heuristically picks a truth value for an unassigned variable

$$p \wedge (q \vee r) \wedge (q \vee \neg r)$$

$\top$

Deduction

p:t

Decision q:f

p:t, q:f

Deduction

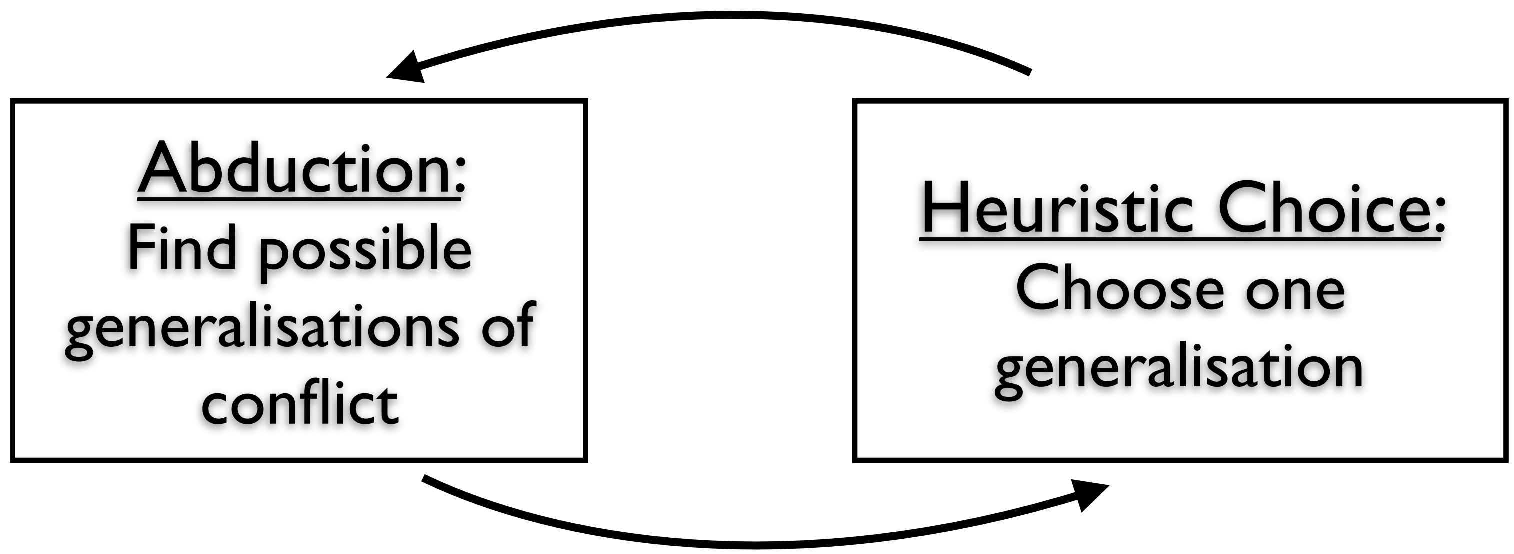
p:t, q:f, r:t

$\bot$

Recall: Widenings jump <u>over a least fixed point</u>

Decisions jump <u>under a greatest fixed point</u> (unusual: unsound!)

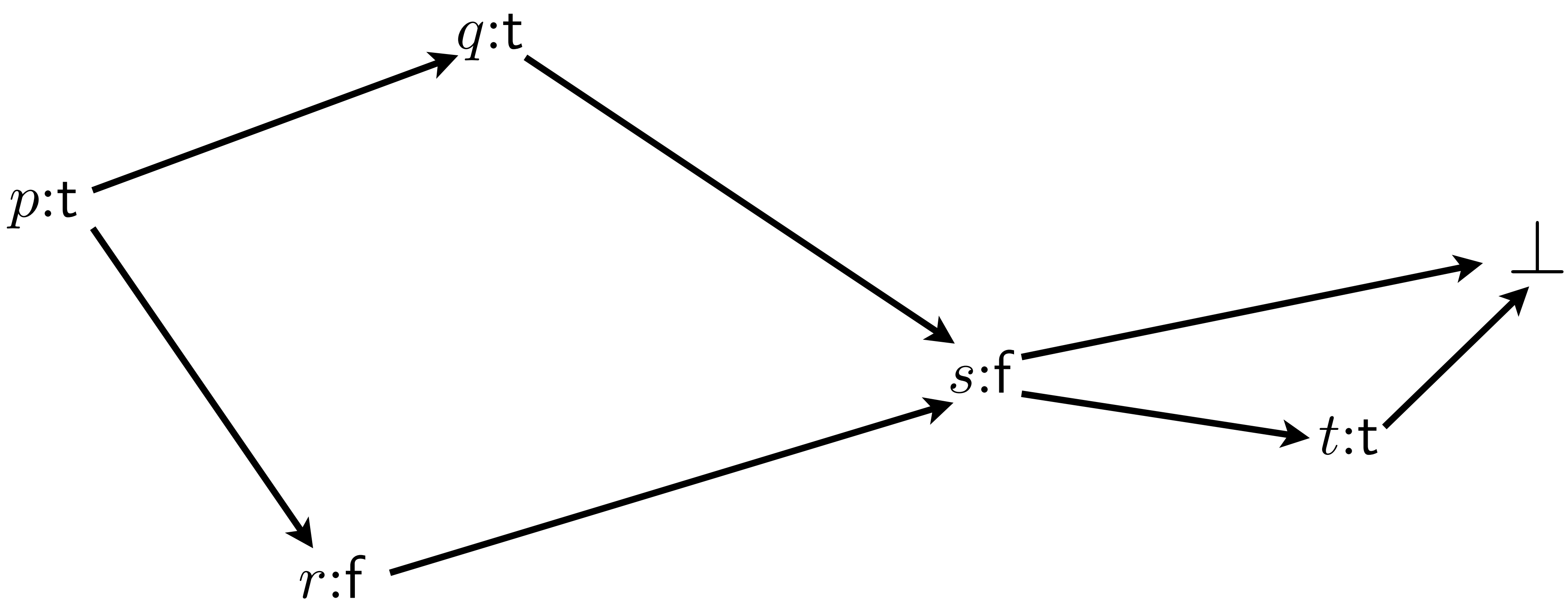
Conflict Analysis



Implication Graph Cutting

CDCL solvers record deductions in data structure called implication graph

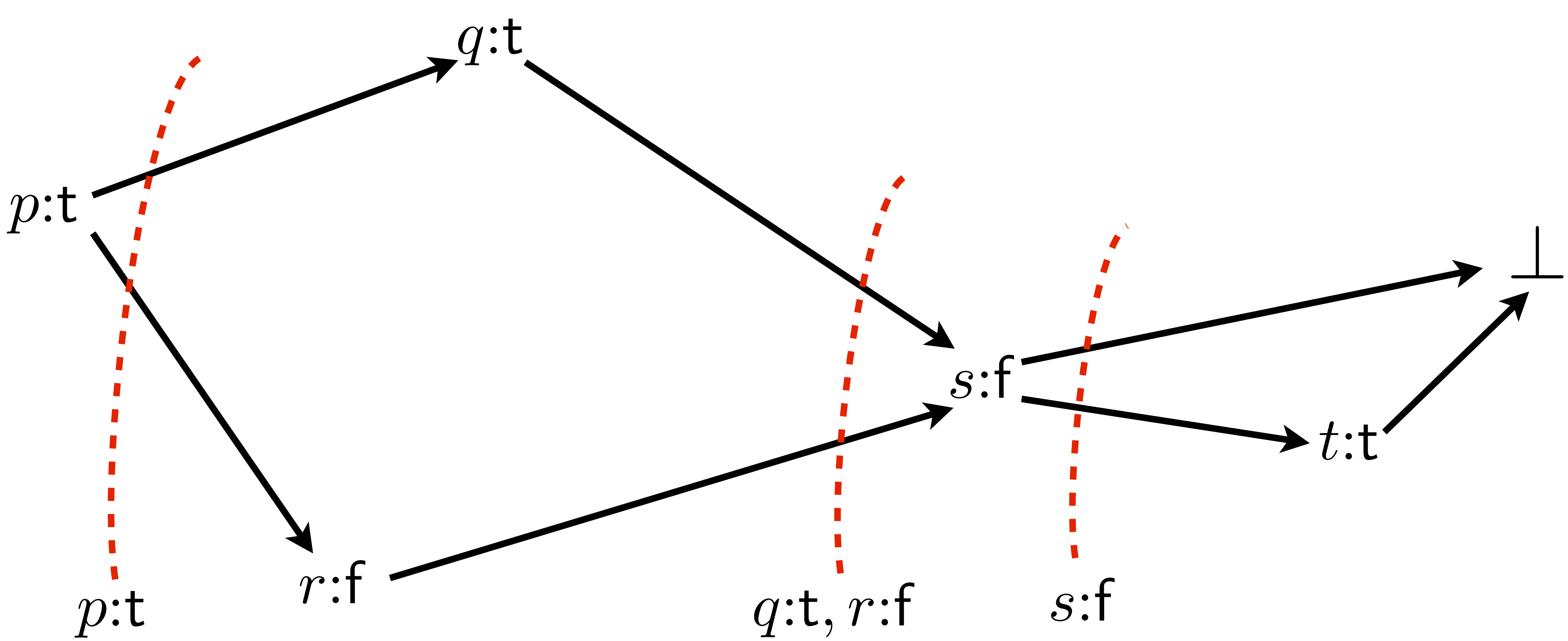
$$(\neg p \vee q) \wedge (\neg p \vee \neg r) \wedge (\neg q \vee r \vee \neg s) \wedge (s \vee t) \wedge (s \vee \neg t)$$



Implication Graph Cutting

CDCL solvers record deductions in data structure called implication graph

$$(\neg p \vee q) \wedge (\neg p \vee \neg r) \wedge (\neg q \vee r \vee \neg s) \wedge (s \vee t) \wedge (s \vee \neg t)$$



Conflict abduction is performed by obtaining cuts through the graph

Original conflict $\pi = (p\text{:t}, q\text{:t}, r\text{:f}, s\text{:f}, t\text{:t})$

Possible generalisations from cuts $cut(\{\pi\}) = \{(p\text{:t}), (q\text{:t}, r\text{:f}), (s\text{:f})\}$

Abduction computes a least fixed point

$(r{:}\mathrm{f})$ $(s{:}\mathrm{t})$ generalisation from clause minimisation

$(p{:}\mathrm{t}, q{:}\mathrm{t})$ $(r{:}\mathrm{f}, s{:}\mathrm{t})$ generalisation from graph cuts

$(p{:}\mathrm{t}, q{:}\mathrm{t}, r{:}\mathrm{f}, s{:}\mathrm{t})$

Original conflict

Abduction computes a least fixed point

$$\begin{tikzcd}[column sep=small]
& & (r{:}\mathrm{f}) & & (s{:}\mathrm{t}) \\
(p{:}\mathrm{t}, q{:}\mathrm{t}) & & & (r{:}\mathrm{f}, s{:}\mathrm{t}) \arrow[ul] \arrow[ur] & \\
& (p{:}\mathrm{t}, q{:}\mathrm{t}, r{:}\mathrm{f}, s{:}\mathrm{t}) \arrow[ul] \arrow[urr] & & &
\end{tikzcd}$$

generalisation from clause minimisation

generalisation from graph cuts

Original conflict

$$\begin{tikzcd}
\{(p{:}\mathrm{t}, q{:}\mathrm{t}), (r{:}\mathrm{f}), (s{:}\mathrm{t})\} \arrow[loop right] \\
\{(p{:}\mathrm{t}, q{:}\mathrm{t}), (r{:}\mathrm{f}, s{:}\mathrm{t})\} \arrow[u] \\
\{(p{:}\mathrm{t}, q{:}\mathrm{t}, r{:}\mathrm{f}, s{:}\mathrm{t})\} \arrow[u]
\end{tikzcd}$$

Collecting all conflicts

Abduction computes a least fixed point

$$
\begin{tikzcd}[column sep=small]
 & & (r{:}\mathsf{f}) & & (s{:}\mathsf{t}) \\
(p{:}\mathsf{t}, q{:}\mathsf{t}) & & & (r{:}\mathsf{f}, s{:}\mathsf{t}) \arrow[ul] \arrow[ur] & \\
 & (p{:}\mathsf{t}, q{:}\mathsf{t}, r{:}\mathsf{f}, s{:}\mathsf{t}) \arrow[ul] \arrow[urr] & & &
\end{tikzcd}
$$

generalisation from clause minimisation

generalisation from graph cuts

Original conflict

$$
\begin{tikzcd}
\{(p{:}\mathsf{t}, q{:}\mathsf{t}), (r{:}\mathsf{f}), (s{:}\mathsf{t})\} \arrow[loop right] \\
\{(p{:}\mathsf{t}, q{:}\mathsf{t}), (r{:}\mathsf{f}, s{:}\mathsf{t})\} \arrow[u] \\
\{(p{:}\mathsf{t}, q{:}\mathsf{t}, r{:}\mathsf{f}, s{:}\mathsf{t})\} \arrow[u]
\end{tikzcd}
$$

Collecting all conflicts

Abduction underapproximately computes the fixed point lfp confs_φ

Heuristic Choice is Dual Narrowing

$$\{(p{:}\mathrm{t}, q{:}\mathrm{t}), (r{:}\mathrm{f}), (s{:}\mathrm{t})\}$$

$$\uparrow$$

$$\{(p{:}\mathrm{t}, q{:}\mathrm{t}), (r{:}\mathrm{f}, s{:}\mathrm{t})\}$$

$$\uparrow$$

$$\{(p{:}\mathrm{t}, q{:}\mathrm{t}, r{:}\mathrm{f}, s{:}\mathrm{t})\}$$

Collecting all conflicts

Heuristic Choice is Dual Narrowing

$$\{(p{:}\mathrm{t}, q{:}\mathrm{t}), (r{:}\mathrm{f}), (s{:}\mathrm{t})\}$$
$$\uparrow$$
$$\{(p{:}\mathrm{t}, q{:}\mathrm{t}), (r{:}\mathrm{f}, s{:}\mathrm{t})\}$$
$$\uparrow$$
$$\{(p{:}\mathrm{t}, q{:}\mathrm{t}, r{:}\mathrm{f}, s{:}\mathrm{t})\}$$

Collecting all conflicts

$$\{(r{:}\mathrm{f})\}$$
$$\uparrow$$
$$\{(r{:}\mathrm{f}, s{:}\mathrm{t})\}$$
$$\uparrow$$
$$\{(p{:}\mathrm{t}, q{:}\mathrm{t}, r{:}\mathrm{f}, s{:}\mathrm{t})\}$$

SAT Solvers choose *one* reason

Heuristic Choice is Dual Narrowing

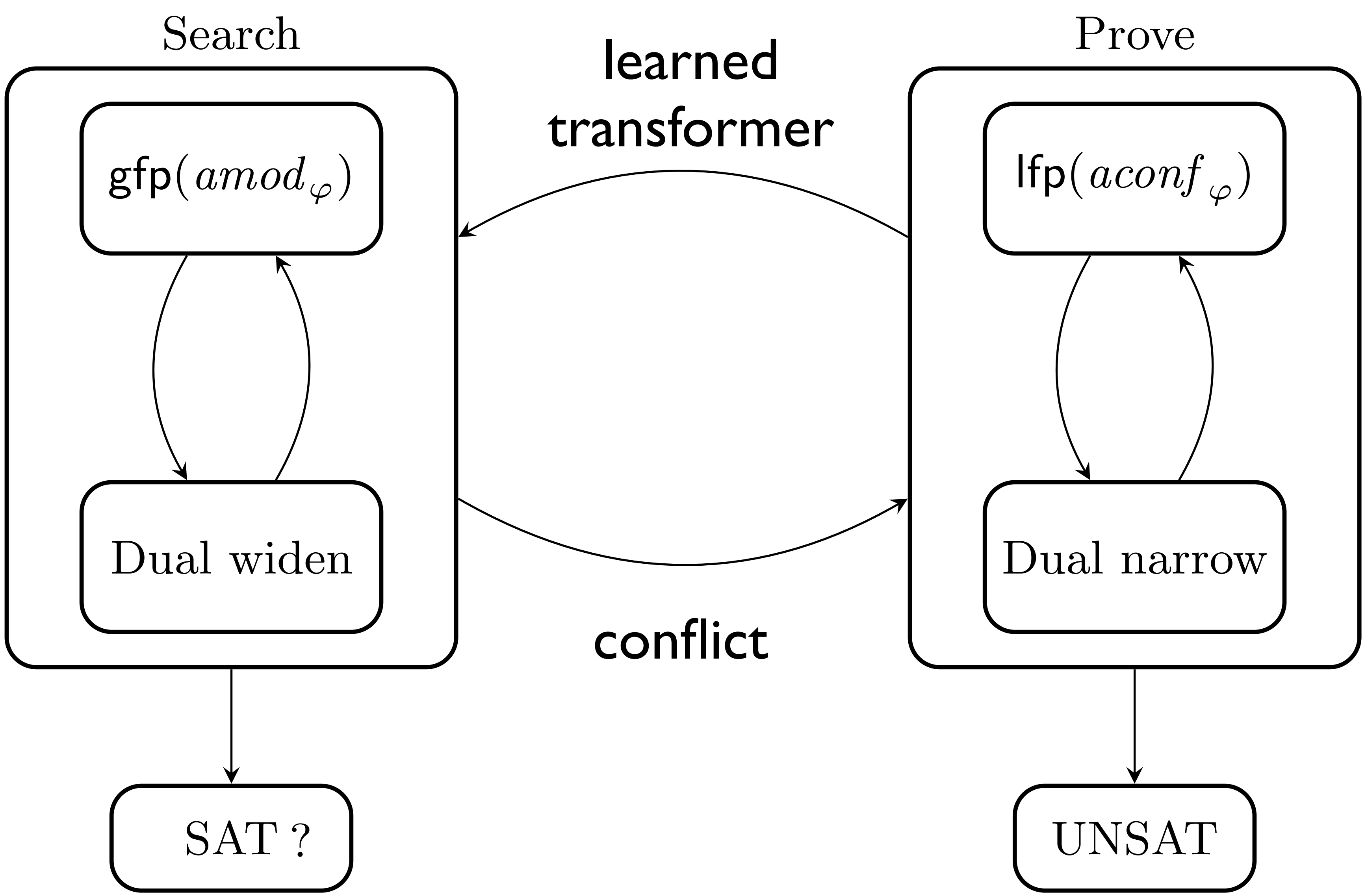
$$\begin{array}{ccc}
\{(p{:}\mathrm{t}, q{:}\mathrm{t}), (r{:}\mathrm{f}), (s{:}\mathrm{t})\} & \sqsupseteq & \{(r{:}\mathrm{f})\} \\
\uparrow & & \uparrow \\
\{(p{:}\mathrm{t}, q{:}\mathrm{t}), (r{:}\mathrm{f}, s{:}\mathrm{t})\} & \sqsupseteq & \{(r{:}\mathrm{f}, s{:}\mathrm{t})\} \\
\uparrow & & \uparrow \\
\{(p{:}\mathrm{t}, q{:}\mathrm{t}, r{:}\mathrm{f}, s{:}\mathrm{t})\} & & \{(p{:}\mathrm{t}, q{:}\mathrm{t}, r{:}\mathrm{f}, s{:}\mathrm{t})\}
\end{array}$$

Collecting all conflicts

SAT Solvers choose *one* reason

Recall that narrowing is used to converge above a greatest fixed point.
Heuristic choice of conflict reasons leads to convergence below a least fixed point!

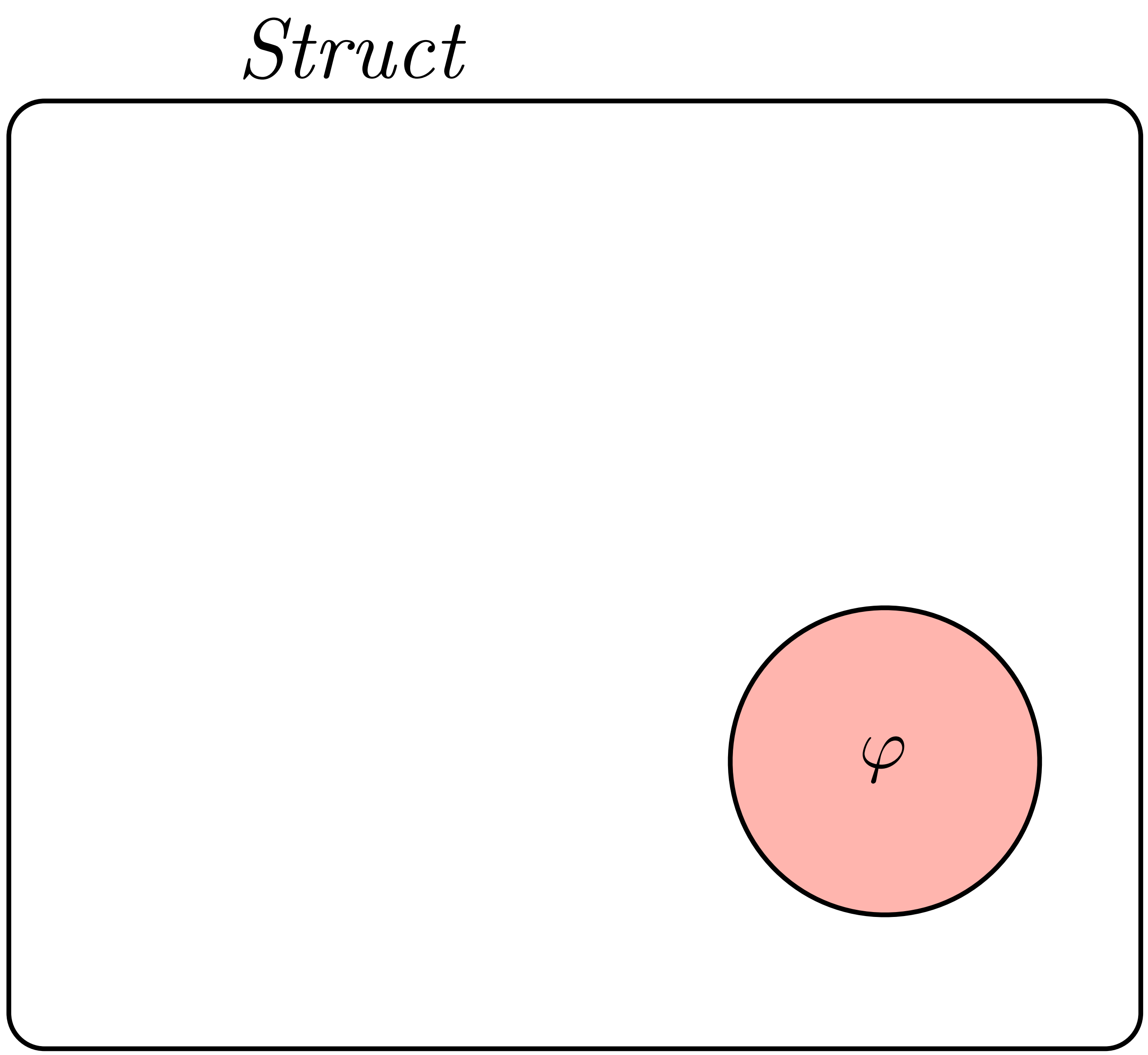
ACDCL: A recipe for deriving natural domain SMT solvers from abstract domains



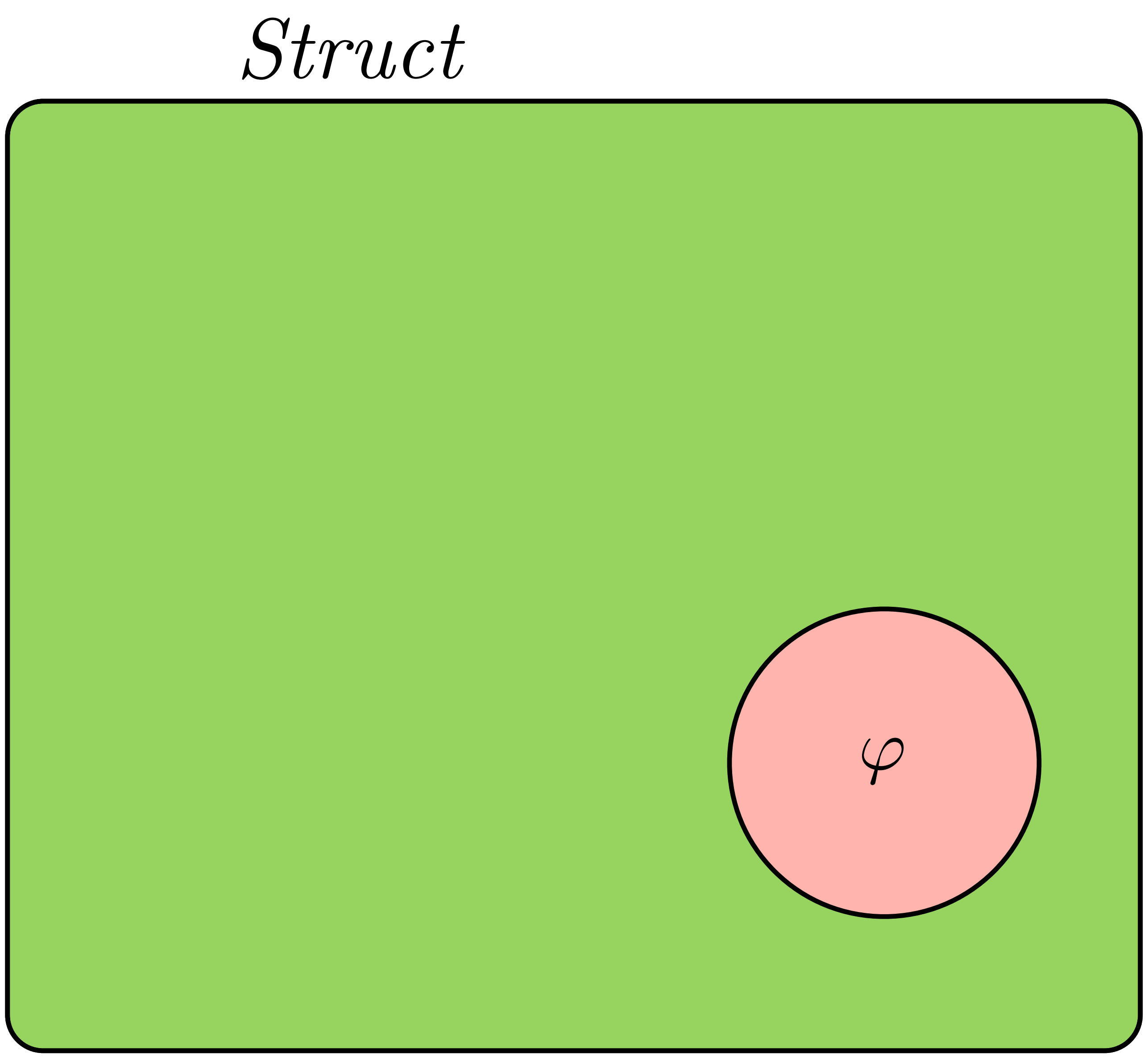
Overapproximating domain O

Underapproximating domain U

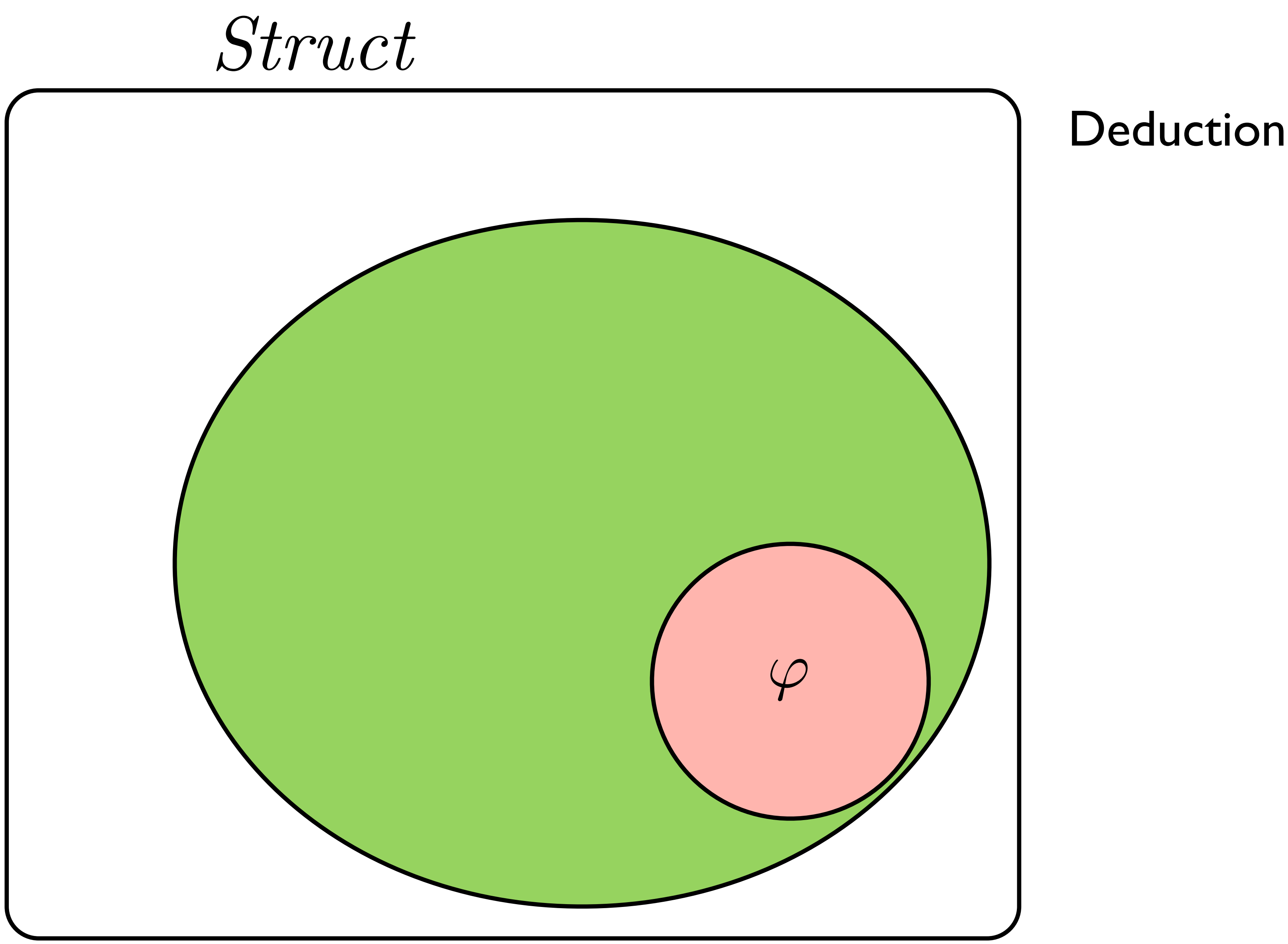
Model Search and Conflict Analysis with Abstract Domains



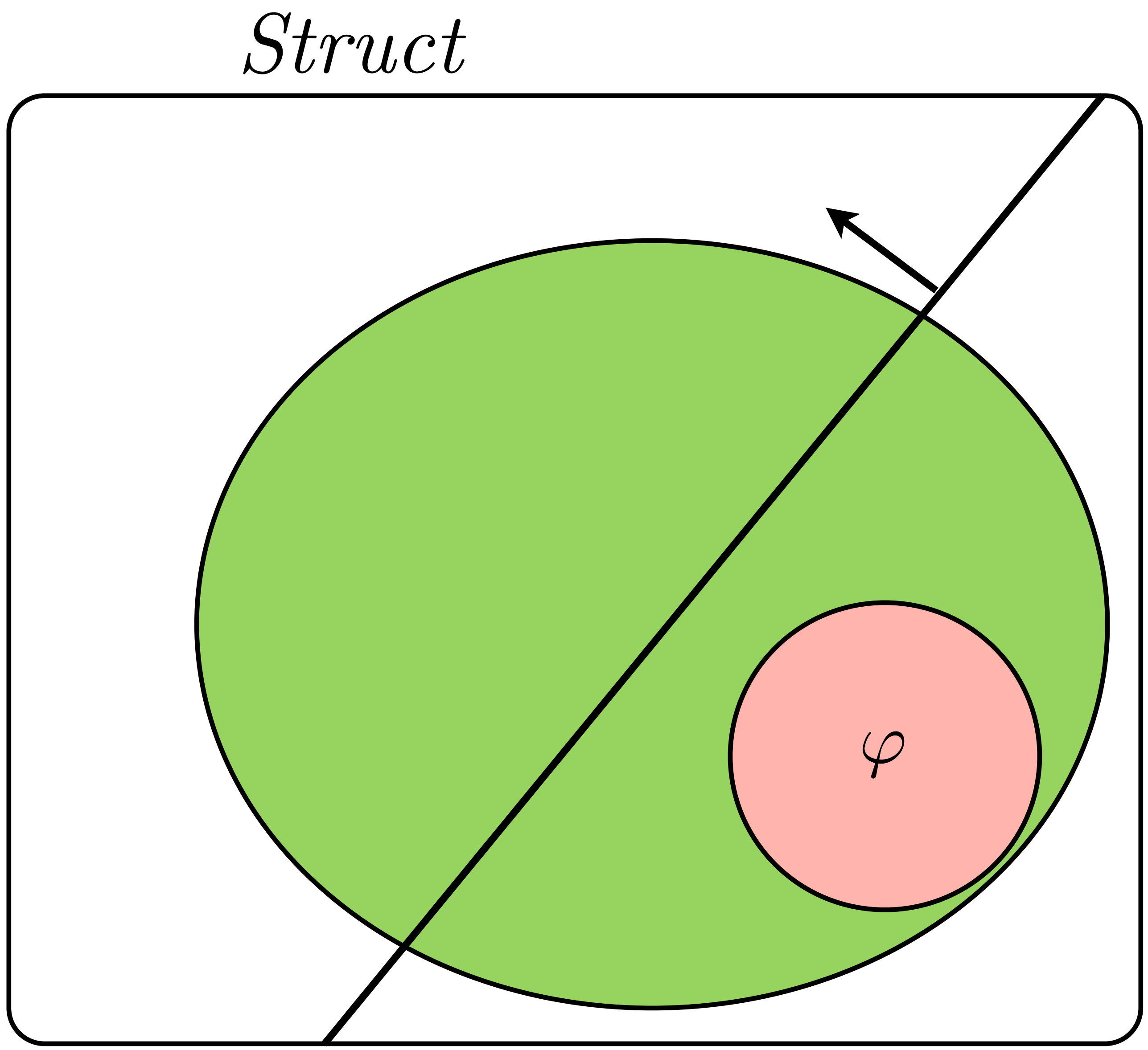
Model Search and Conflict Analysis with Abstract Domains



Model Search and Conflict Analysis with Abstract Domains

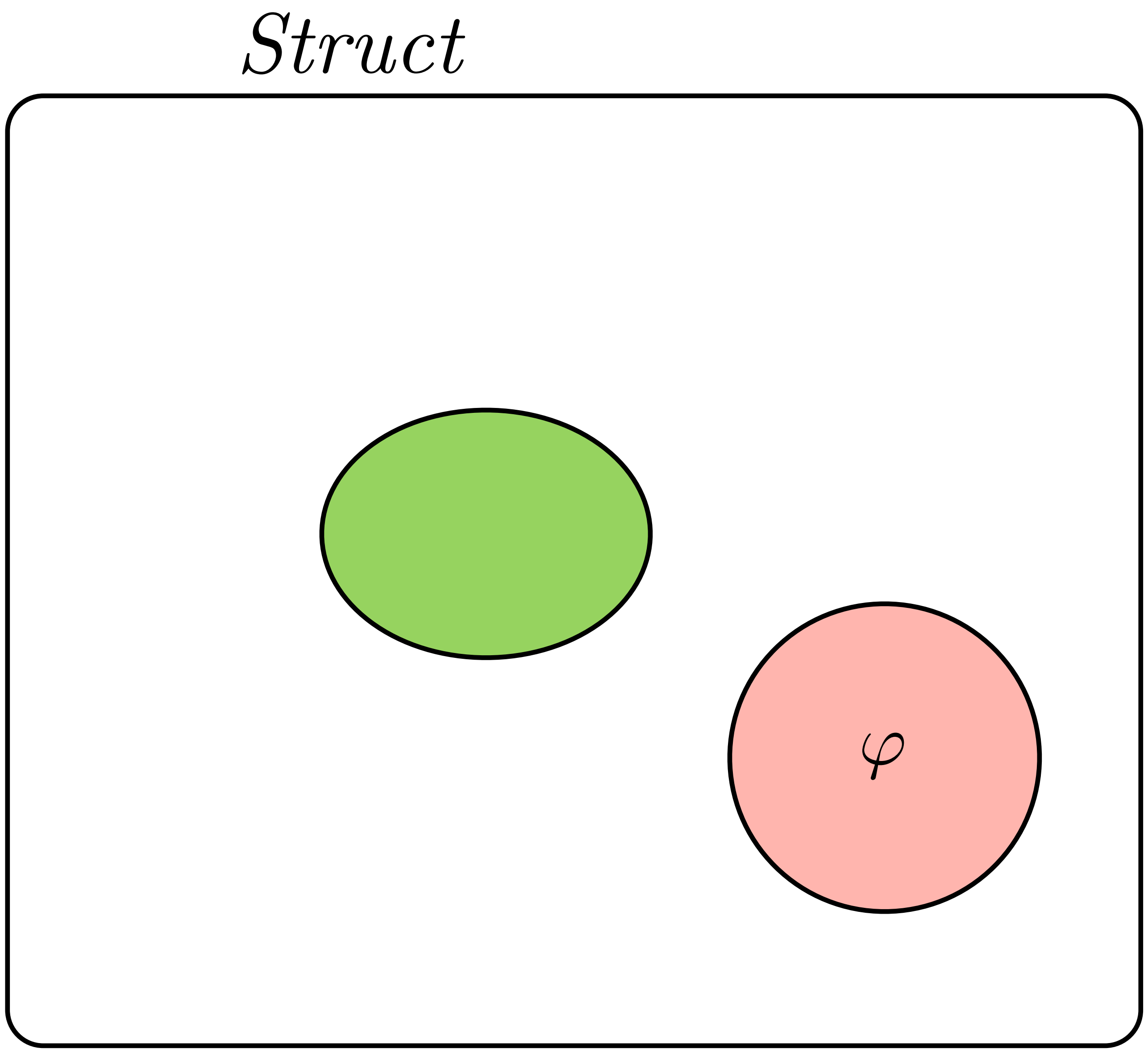


Model Search and Conflict Analysis with Abstract Domains



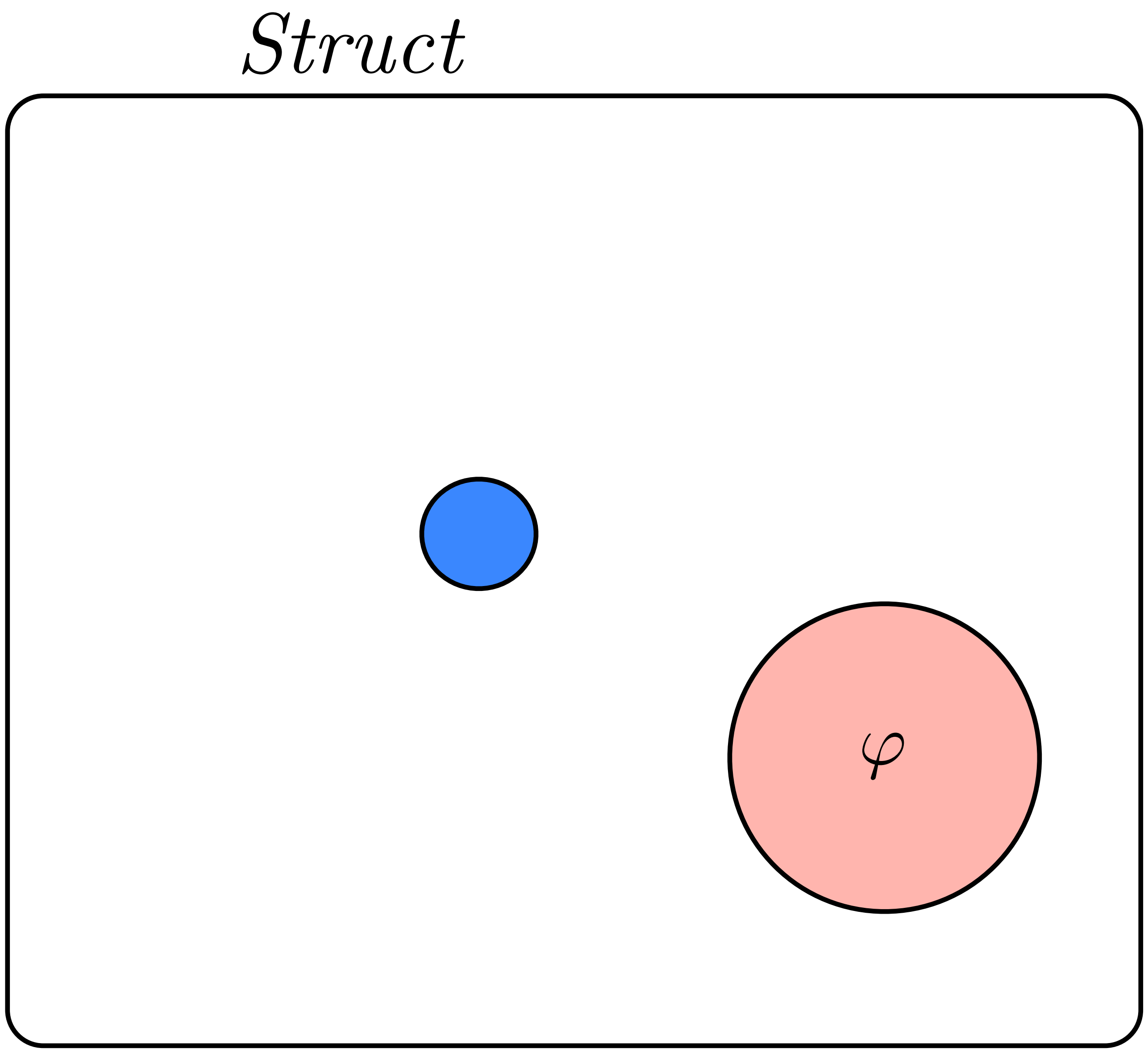
Deduction
Decision

Model Search and Conflict Analysis with Abstract Domains



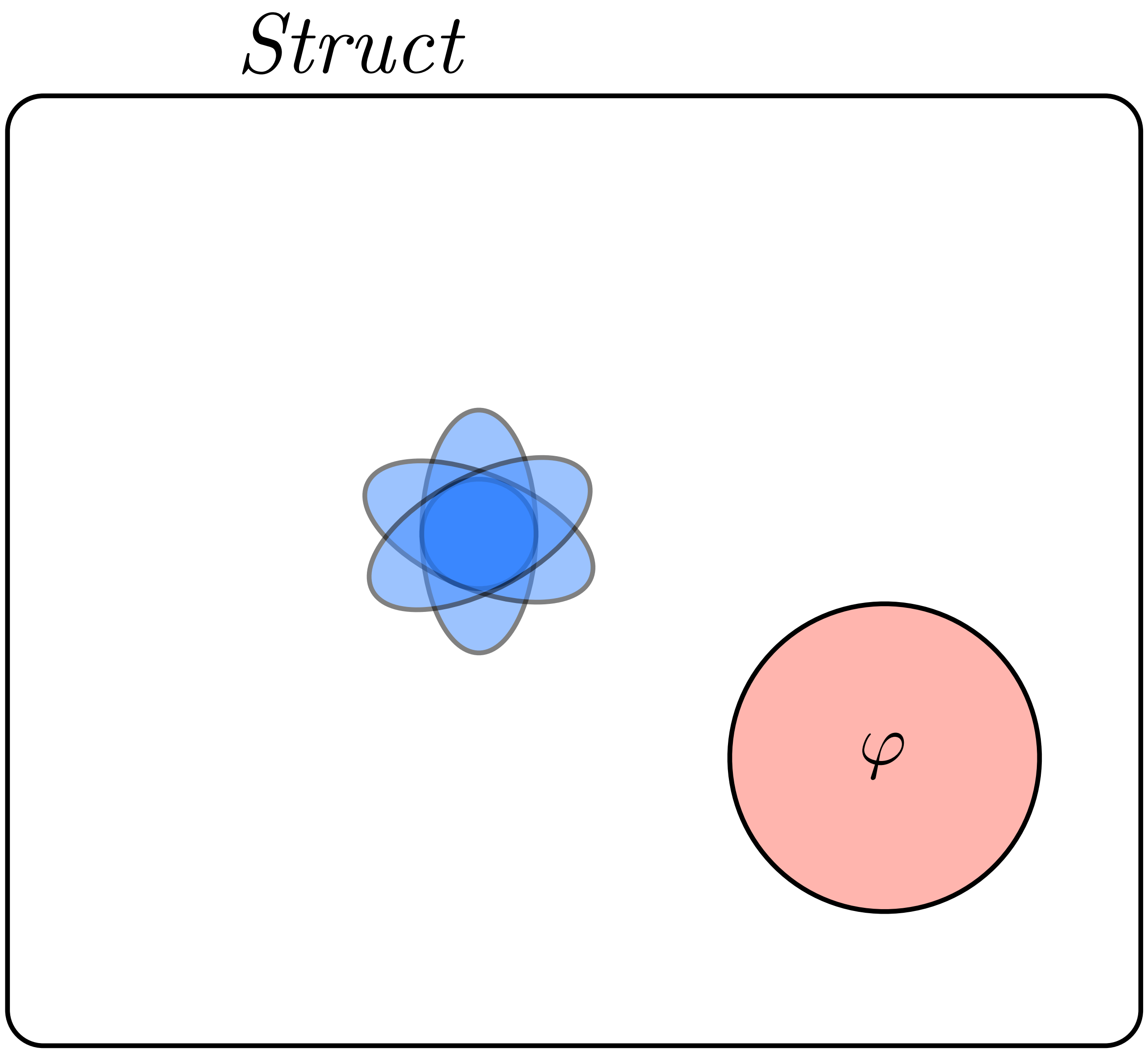
Deduction
Decision
Deduction

Model Search and Conflict Analysis with Abstract Domains



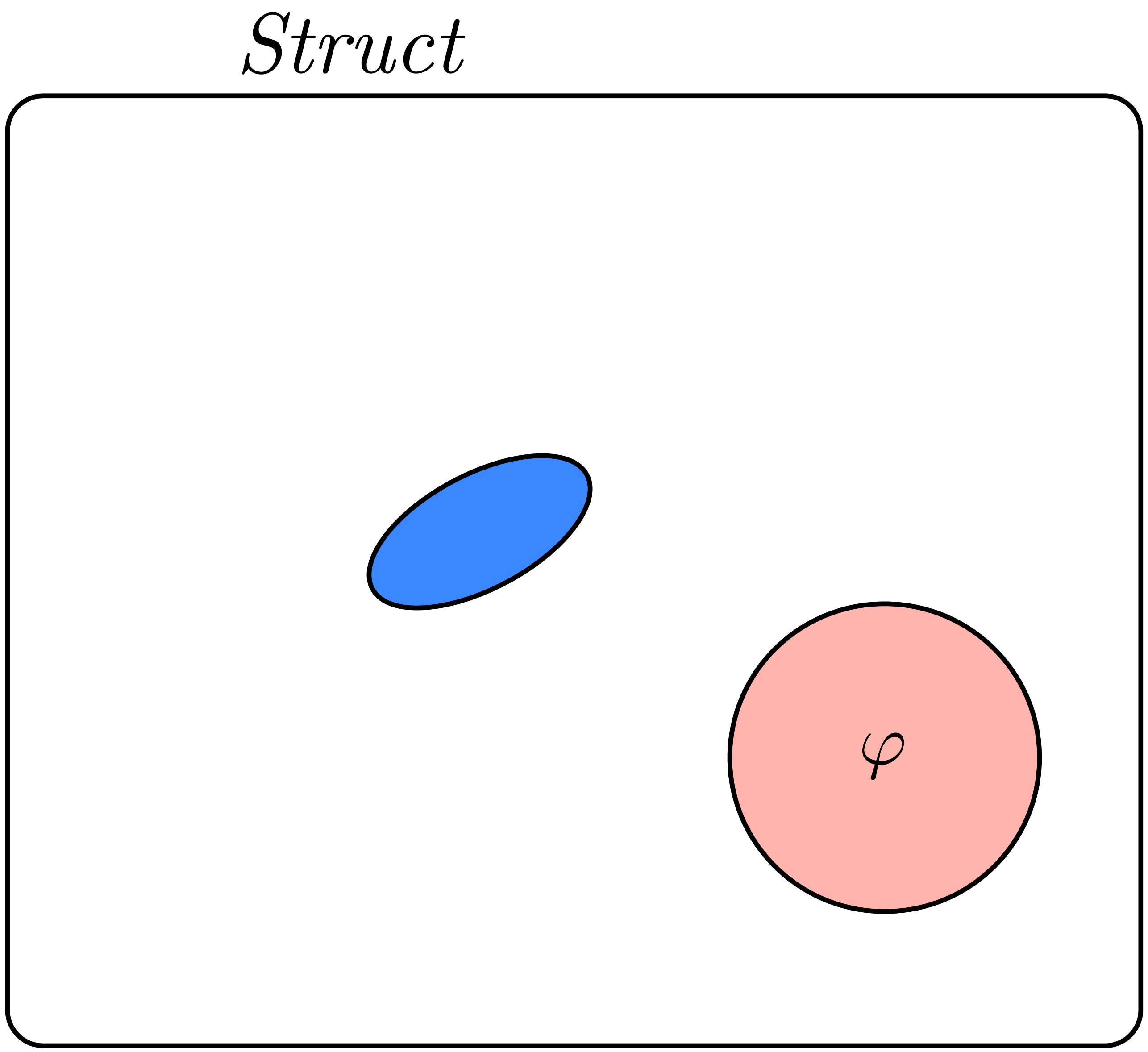
Deduction
Decision
Deduction
Conflict

Model Search and Conflict Analysis with Abstract Domains



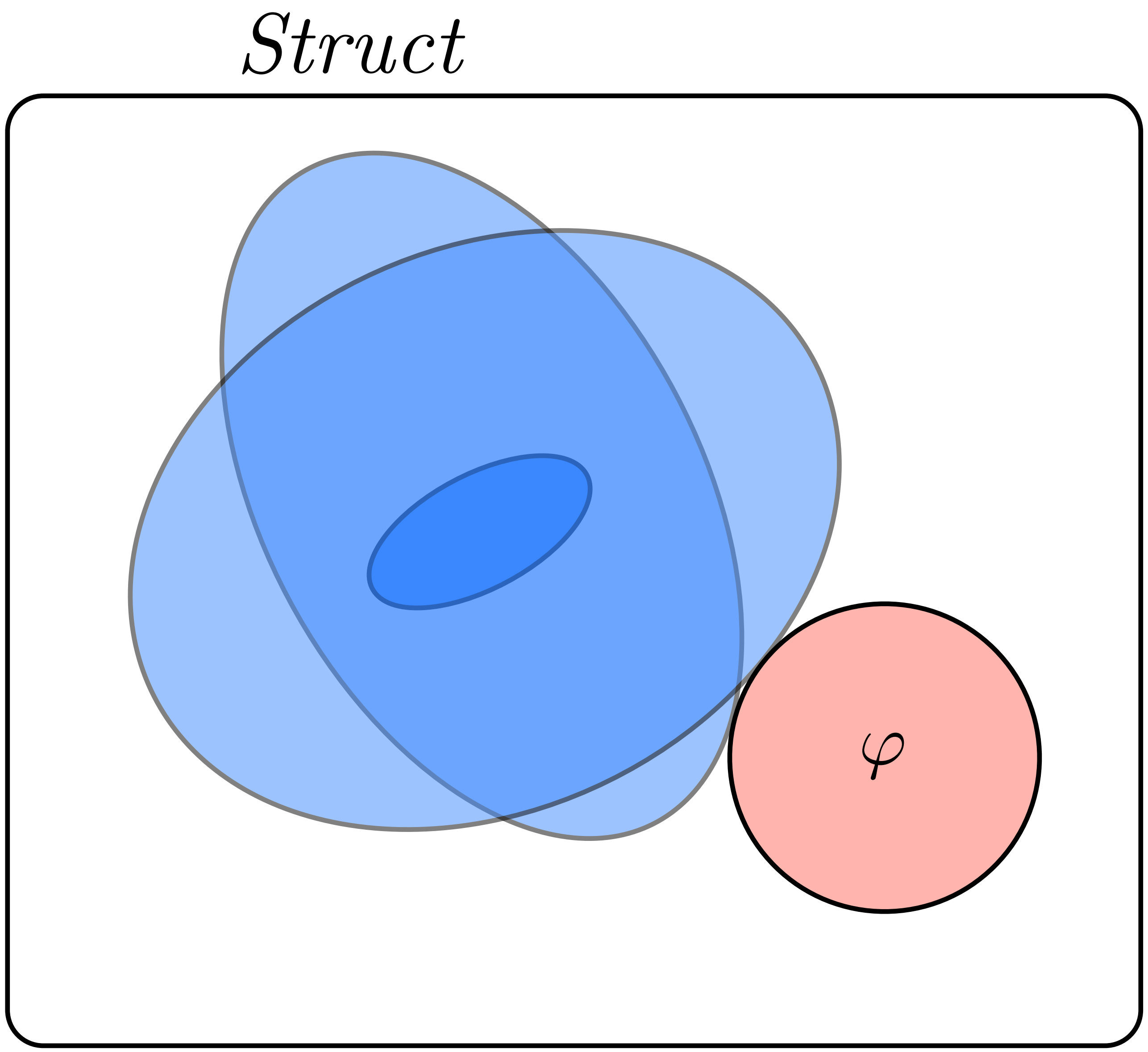
Deduction
Decision
Deduction
Conflict
Abduction

Model Search and Conflict Analysis with Abstract Domains



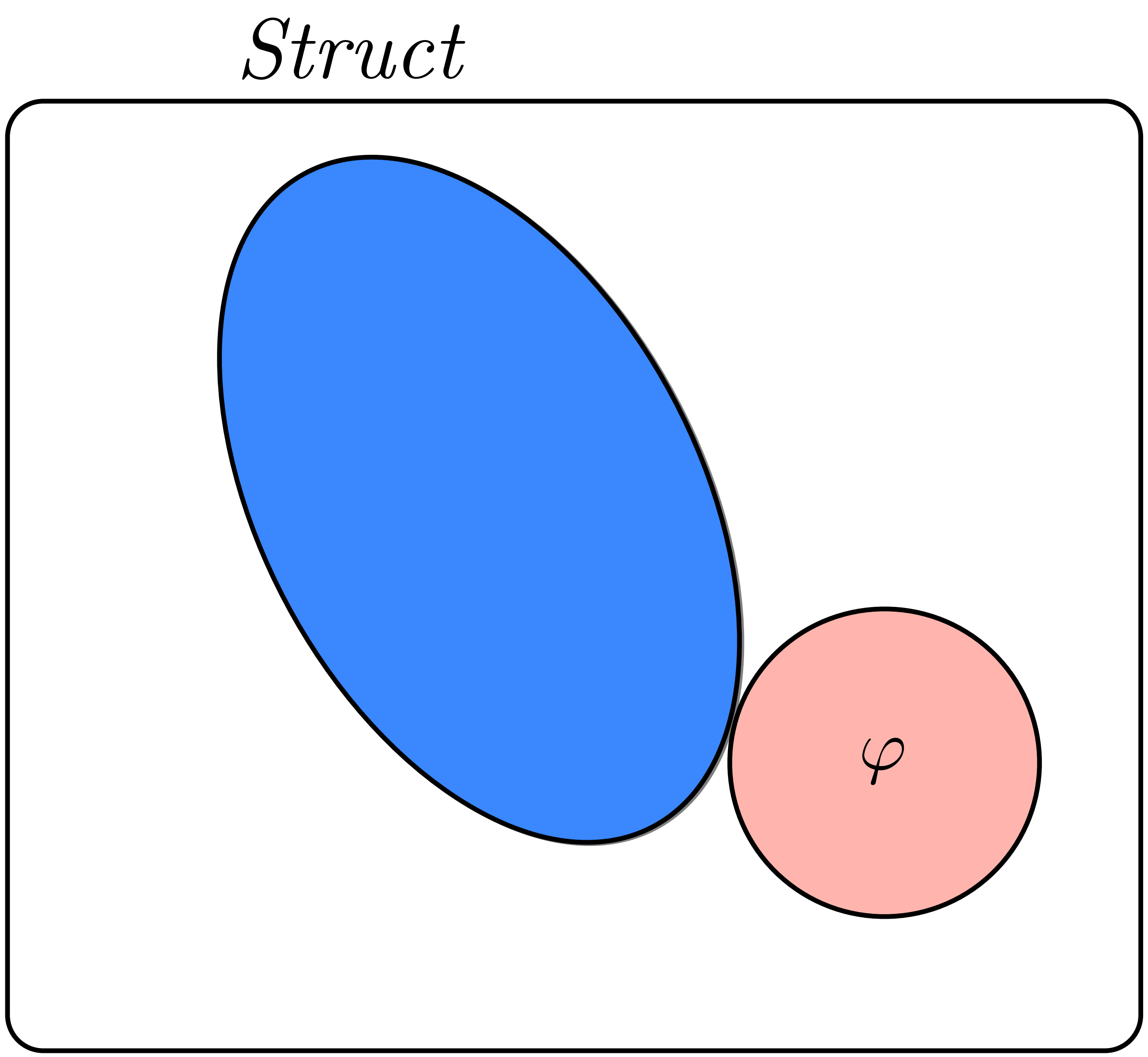
Deduction
Decision
Deduction
Conflict
Abduction
Choice

Model Search and Conflict Analysis with Abstract Domains

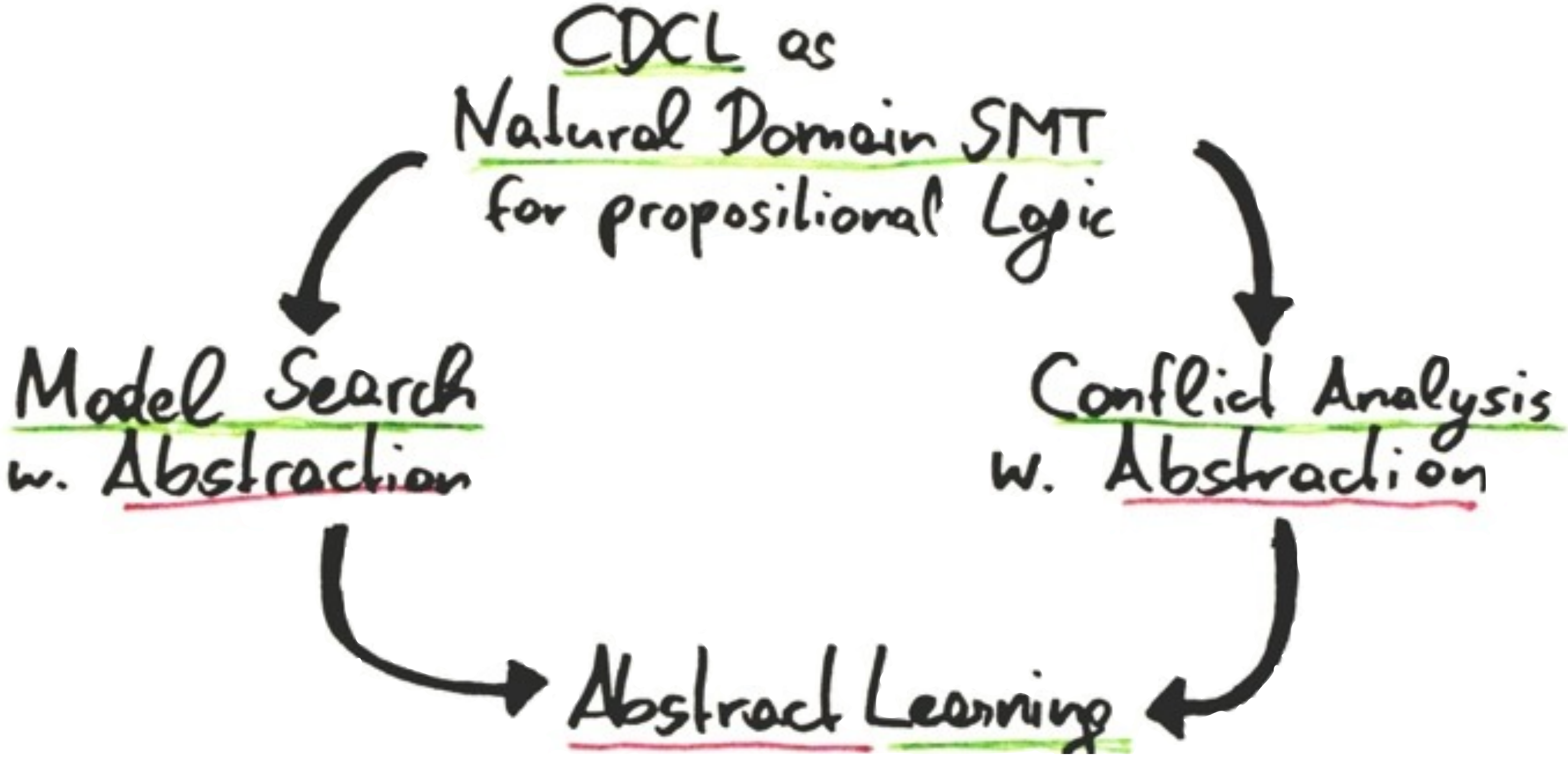


Deduction
Decision
Deduction
Conflict
Abduction
Choice
Abduction

Model Search and Conflict Analysis with Abstract Domains

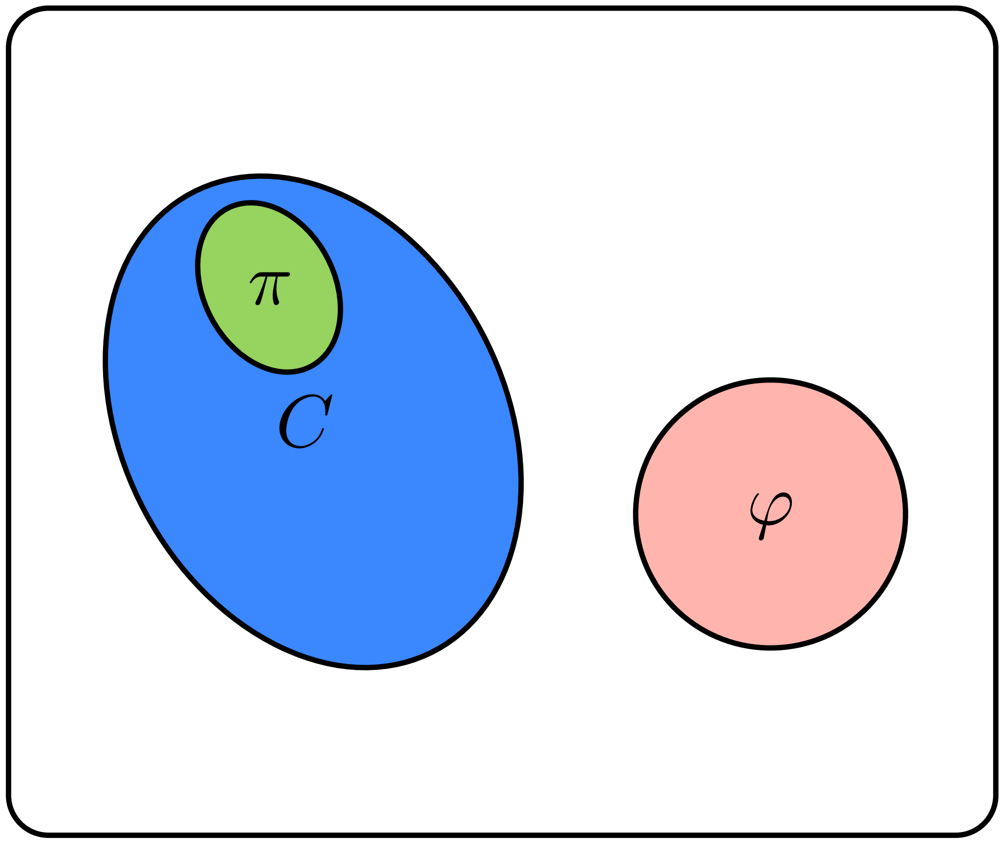


Deduction
Decision
Deduction
Conflict
Abduction
Choice
Abduction
Choice


CDCL as
Natural Domain SMT
for propositional Logic
Model Search
w. Abstraction
Conflict Analysis
w. Abstraction
Abstract Learning

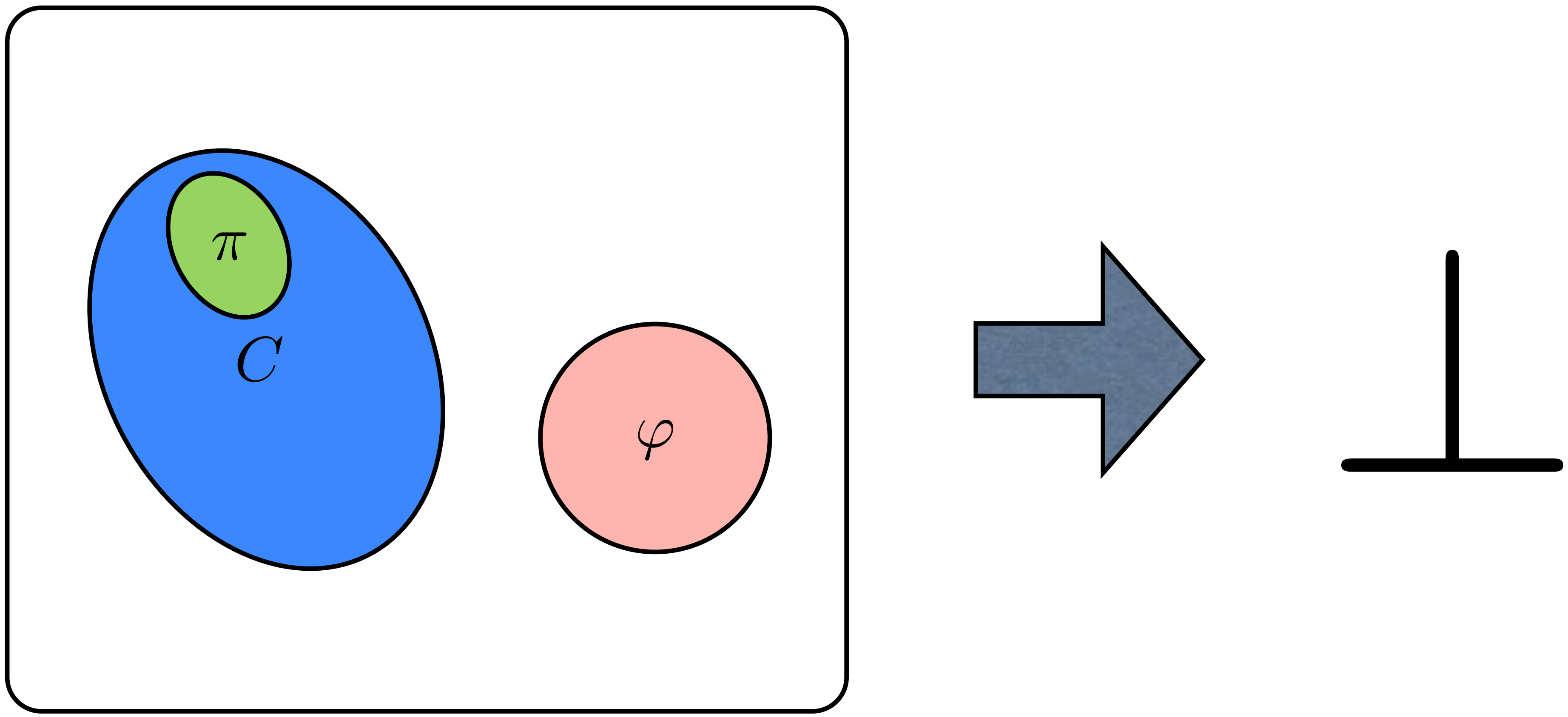
Tabu Learning

Simple but weak form of learning:
When the conflict region is reentered immediately deduce conflict



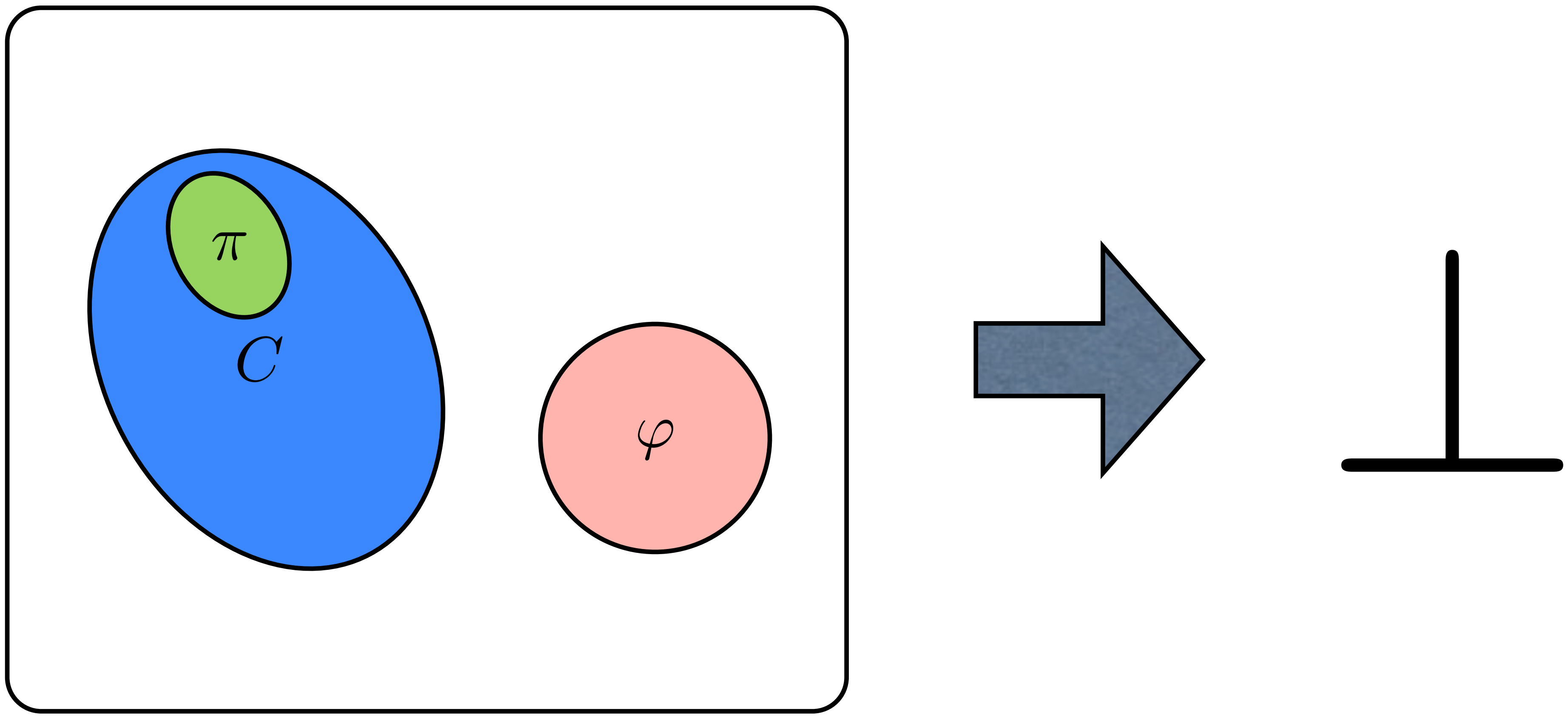
Tabu Learning

Simple but weak form of learning:
When the conflict region is reentered immediately deduce conflict



Tabu Learning

Simple but weak form of learning:
When the conflict region is reentered immediately deduce conflict

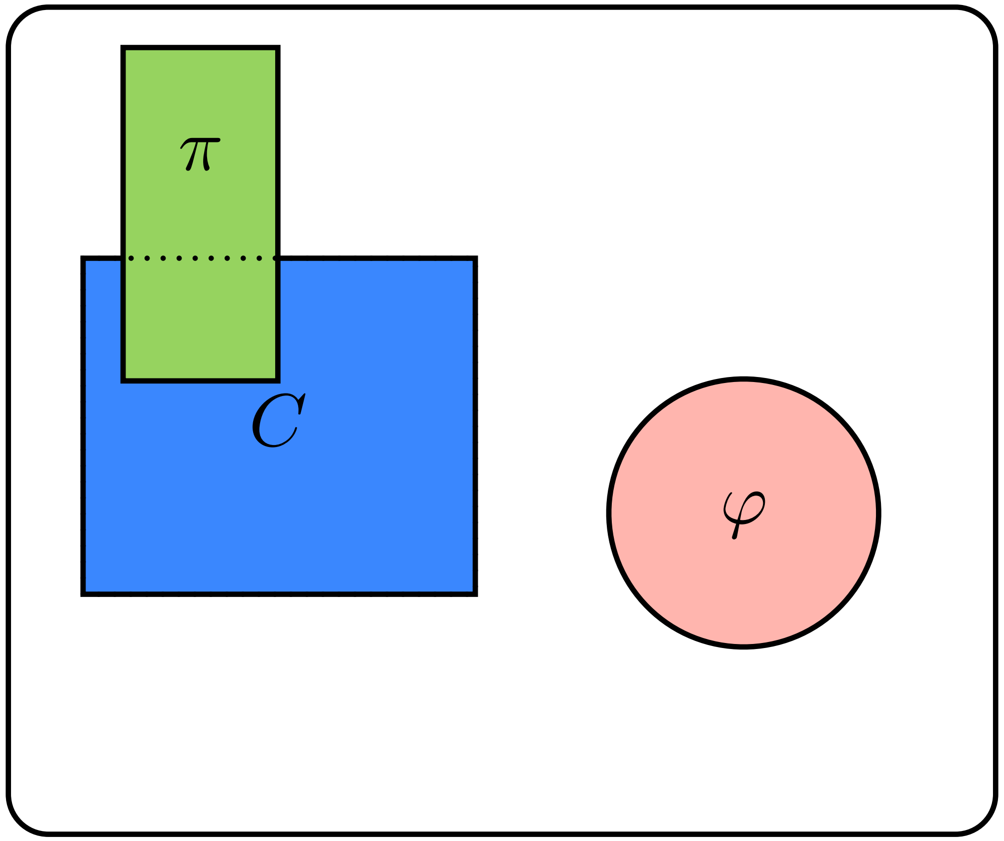


No lattice theoretic prerequisites, possible over any domain

$$tabu_C(\pi) = \begin{cases} \perp & \text{if } \pi \sqsubseteq C \\ \pi & \text{otherwise} \end{cases}$$

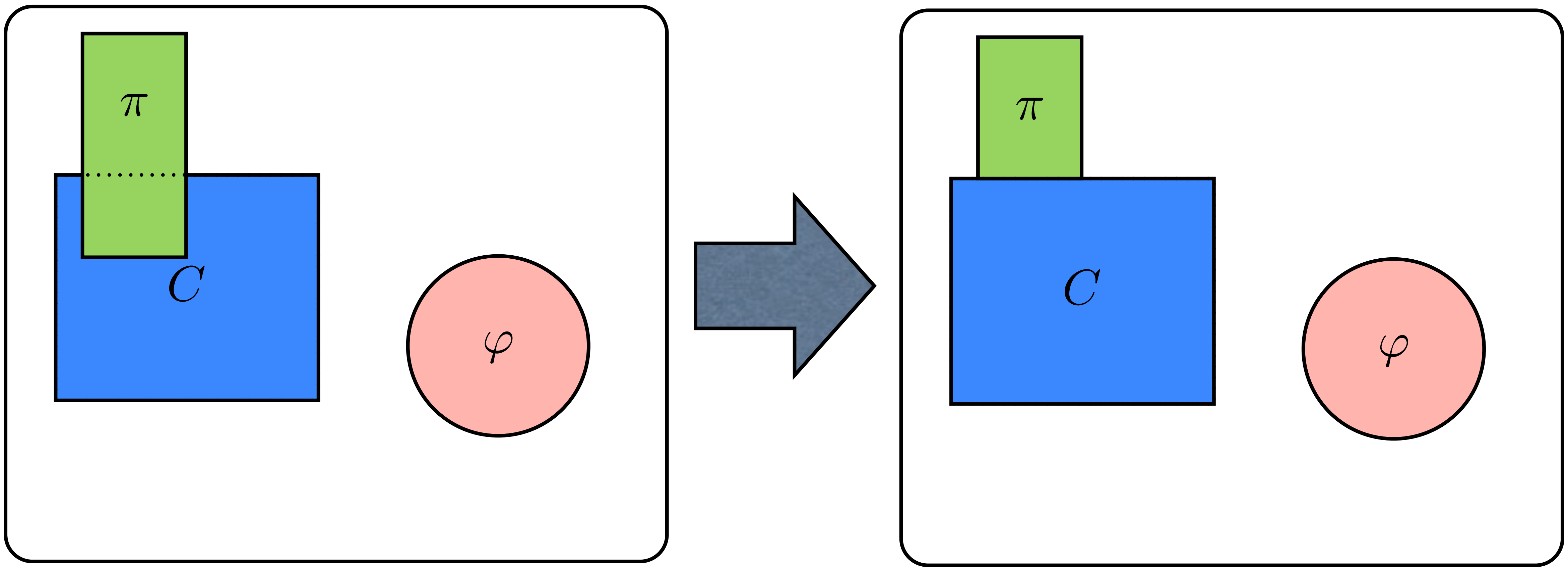
Propositional Clause Learning

When assignment is "nearly conflicting", drive the search away from the conflict



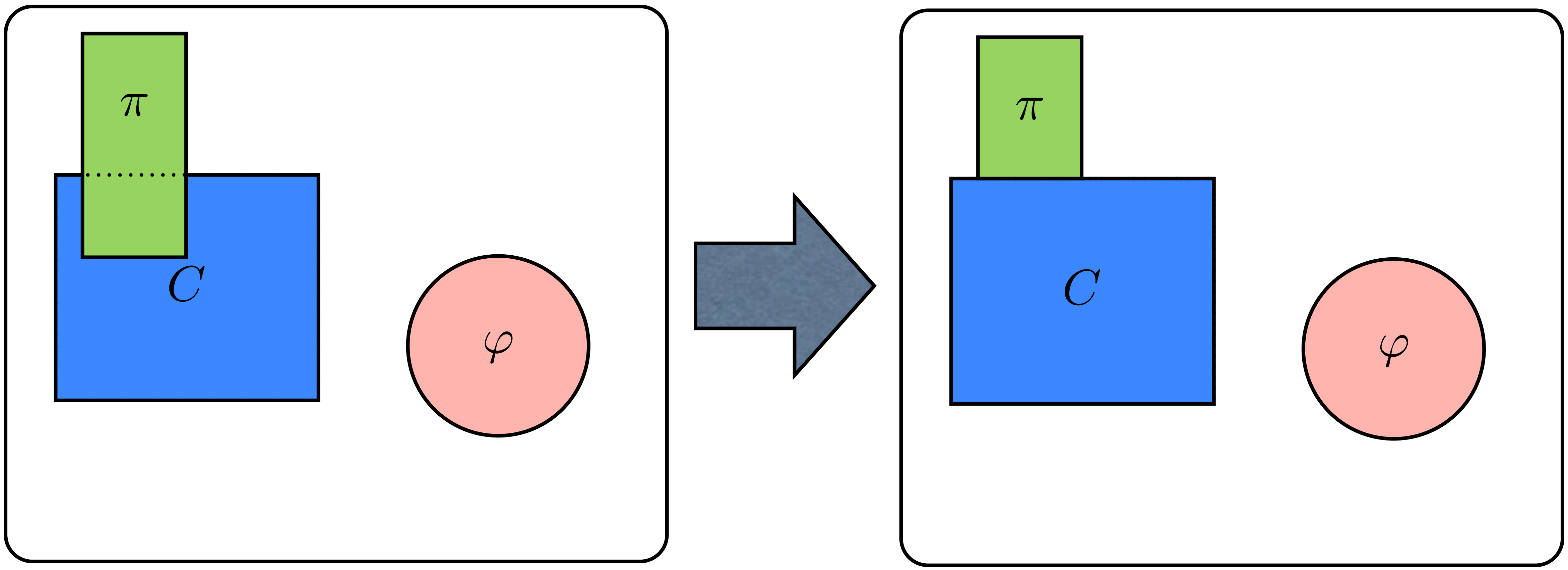
Propositional Clause Learning

When assignment is "nearly conflicting", drive the search away from the conflict



Propositional Clause Learning

When assignment is "nearly conflicting", drive the search away from the conflict



$$C = (p{:}\mathrm{t}, q{:}\mathrm{t}, r{:}\mathrm{f}) = (p{:}\mathrm{t}) \sqcap (q{:}\mathrm{t}) \sqcap (r{:}\mathrm{f})$$

complements drive the search away from conflict

decomposition allows us to express "nearly conflicting"

$$unit_{(p:\mathrm{t},q:\mathrm{t},r:\mathrm{f})}(\pi) = \begin{cases} \pi \sqcap \neg(p{:}\mathrm{t}) & \pi \sqsubseteq (q{:}\mathrm{t}) \wedge \pi \sqsubseteq (r{:}\mathrm{f}) \\ \pi \sqcap \neg(q{:}\mathrm{t}) & \pi \sqsubseteq (p{:}\mathrm{t}) \wedge \pi \sqsubseteq (r{:}\mathrm{f}) \\ \pi \sqcap \neg(r{:}\mathrm{f}) & \pi \sqsubseteq (p{:}\mathrm{t}) \wedge \pi \sqsubseteq (q{:}\mathrm{t}) \end{cases}$$

Complementable Meet Irreducibles

Clause learning requires a weak complementation property of the abstraction

$$\underbrace{(p{:}\mathrm{t}, q{:}\mathrm{f})}_{\text{No precise complement}} = \underbrace{(p{:}\mathrm{t})}_{\text{Precise complement}} \sqcap \underbrace{(q{:}\mathrm{f})}_{\text{Precise complement}}$$

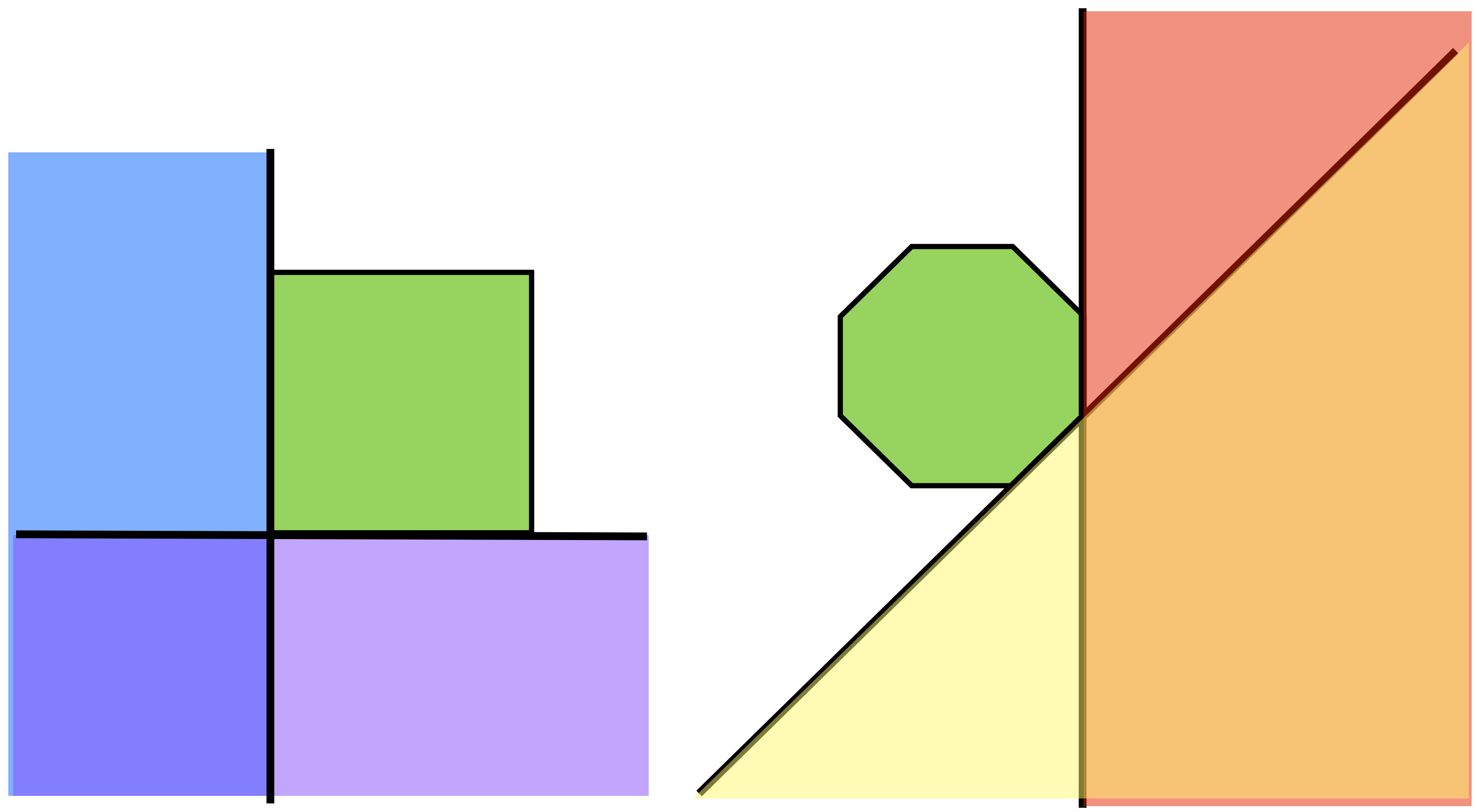
Every element needs to have a decomposition into precisely complementable elements.

Complementable Meet Irreducibles

Examples of lattices with complementable meet irreducibles

Complementable Meet Irreducibles

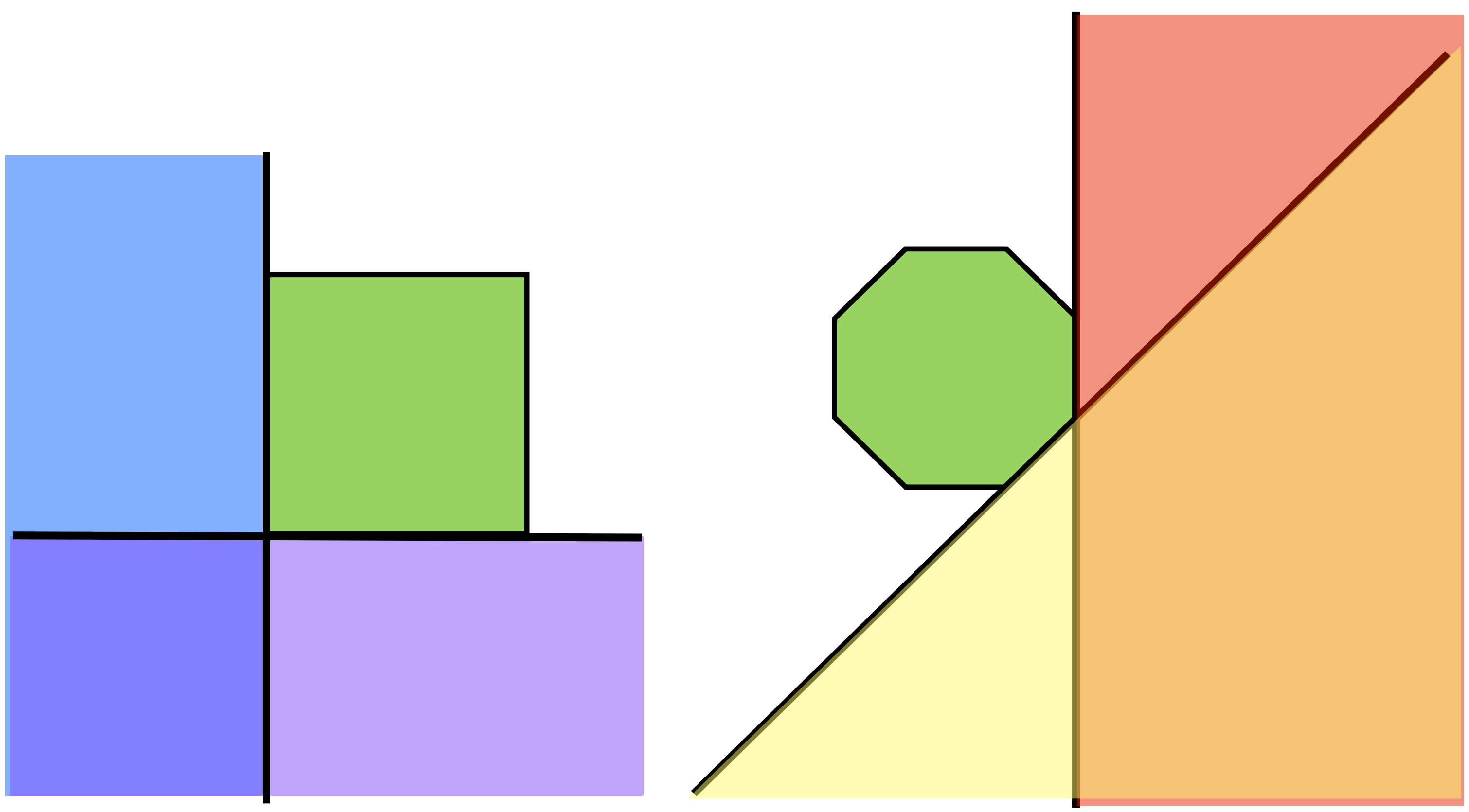
Examples of lattices with complementable meet irreducibles



Intervals and Octagons are intersections of complementable half-spaces

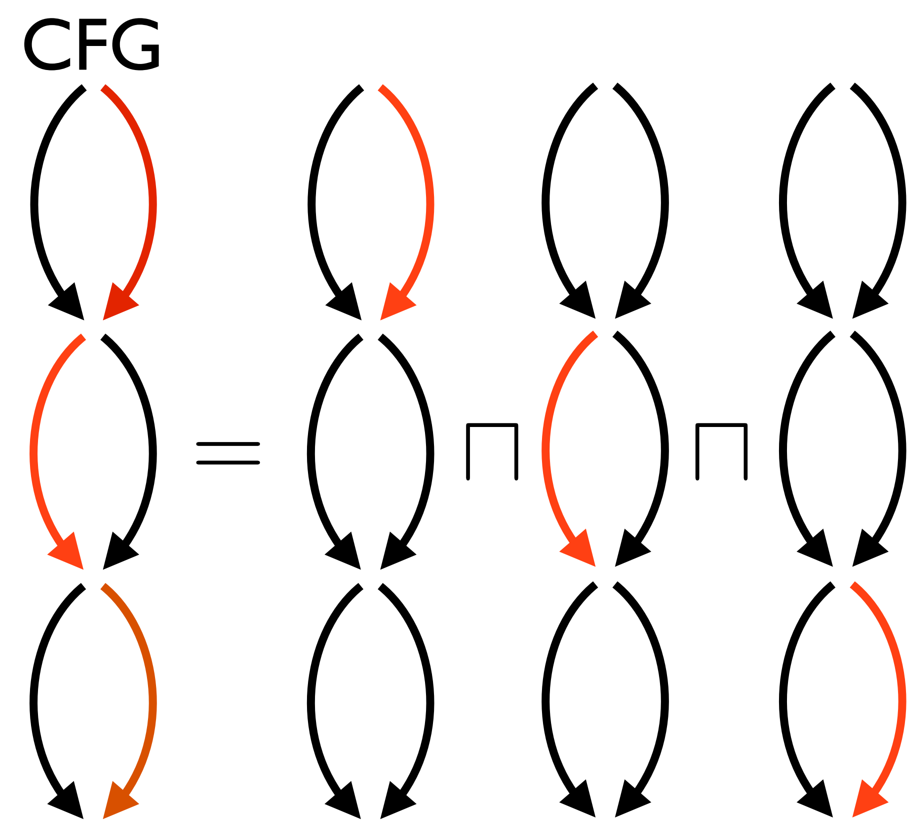
Complementable Meet Irreducibles

Examples of lattices with complementable meet irreducibles



Intervals and Octagons are intersections of complementable half-spaces

$$Branches \to \{\text{left}, \text{right}, \top\}$$

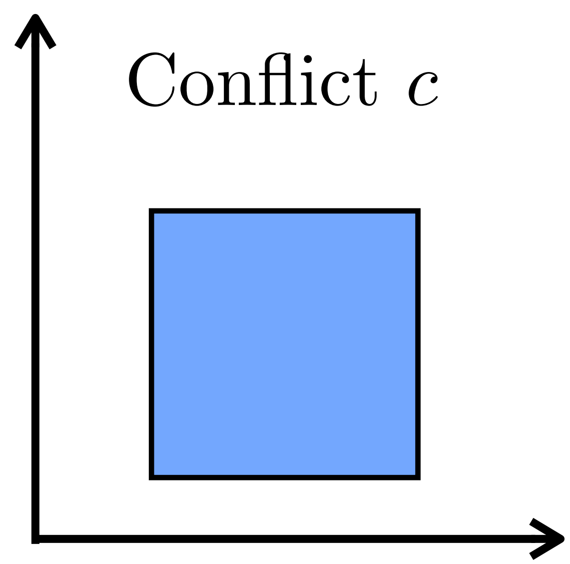


Trace abstraction based on control history

Generalised Unit Rule

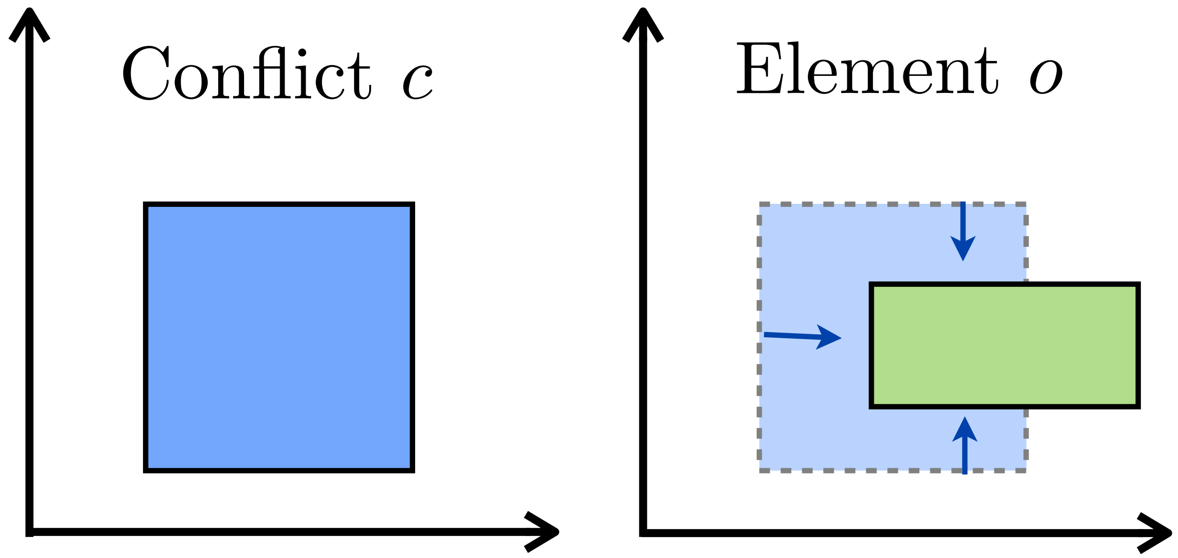
Generalised Unit Rule

Intervals



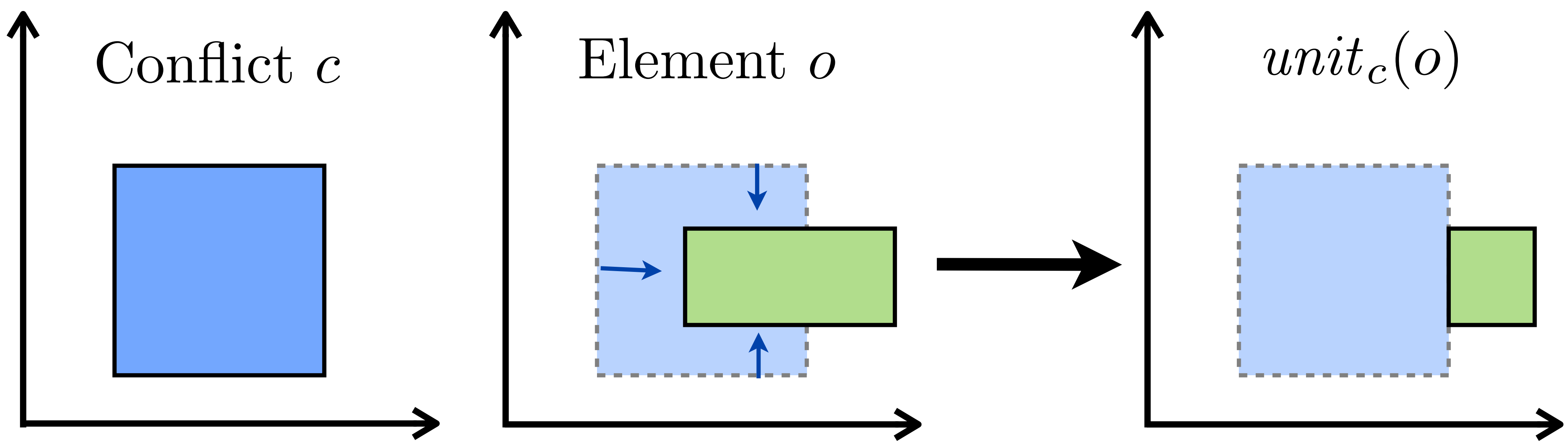
Generalised Unit Rule

Intervals



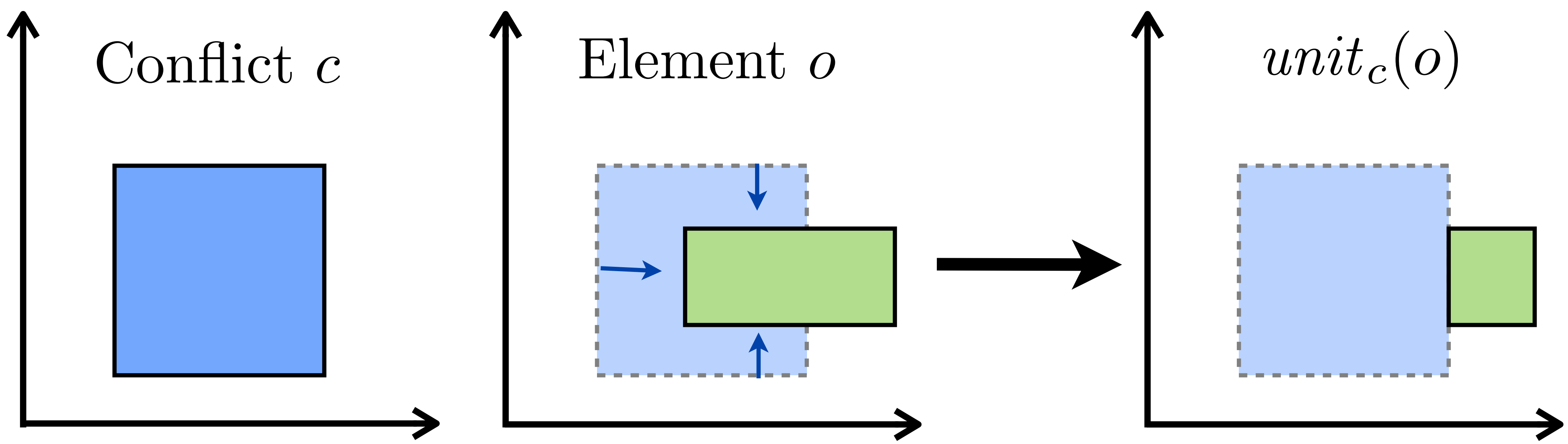
Generalised Unit Rule

Intervals

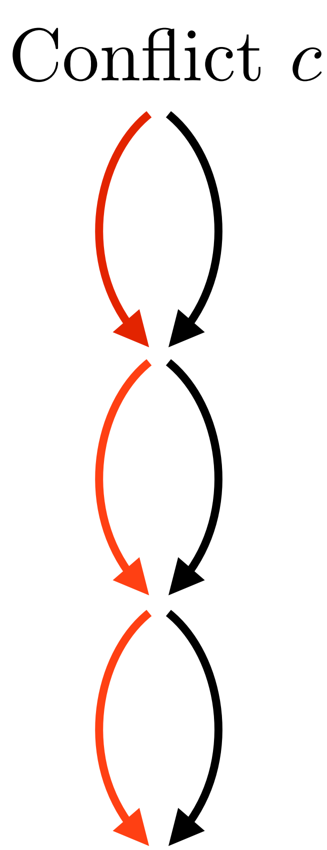


Generalised Unit Rule

Intervals

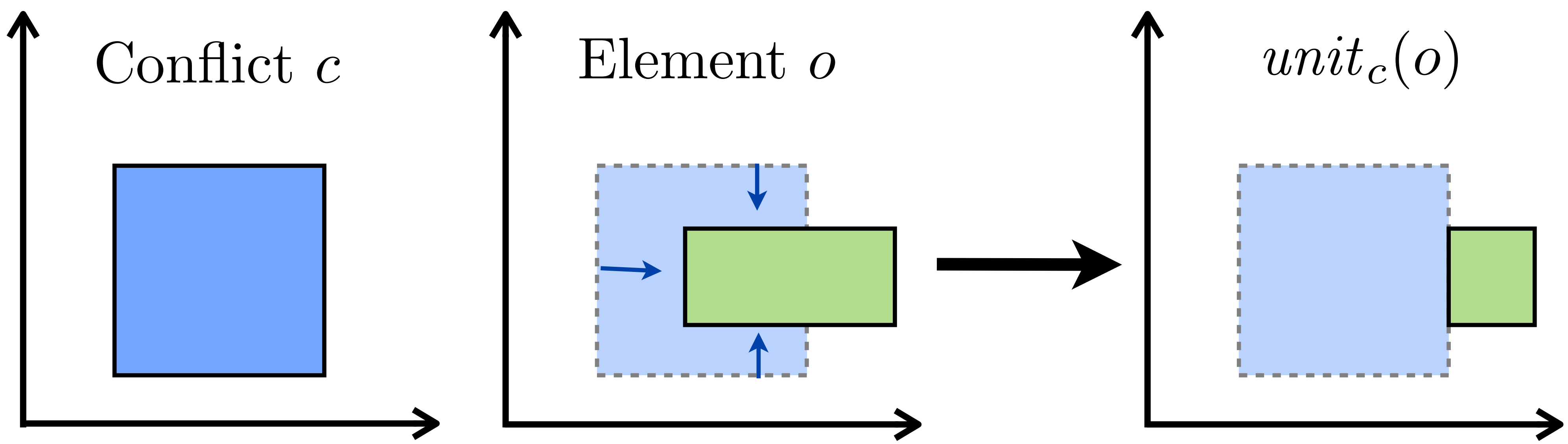


Trace abstractions:

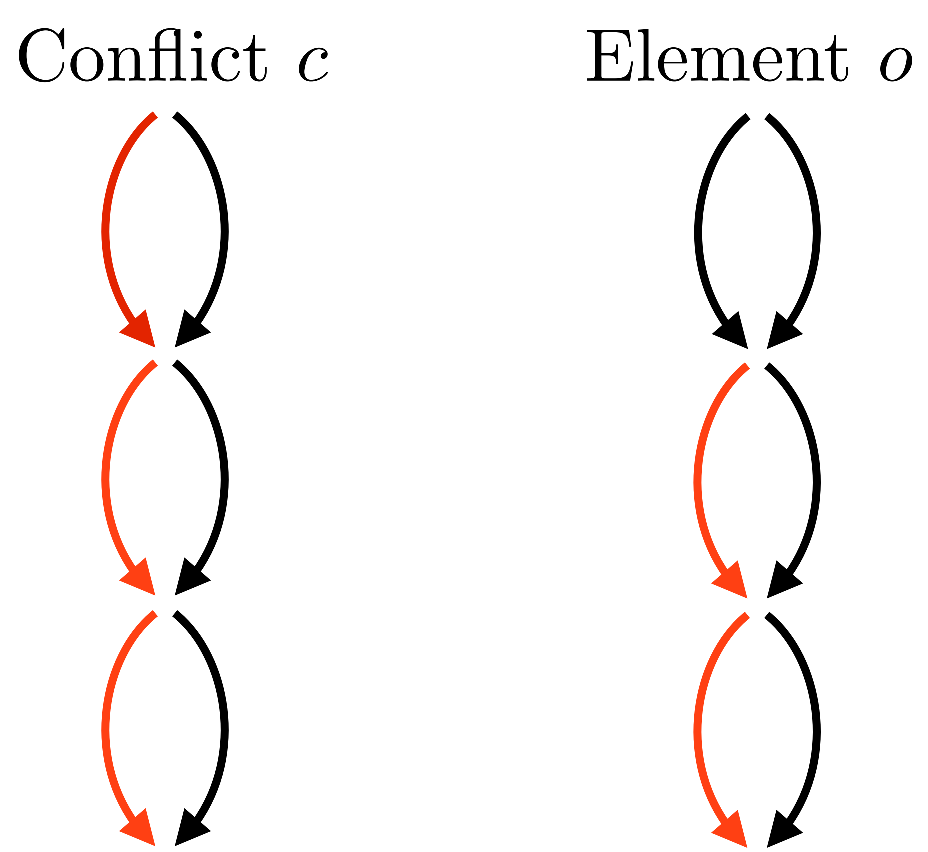


Generalised Unit Rule

Intervals

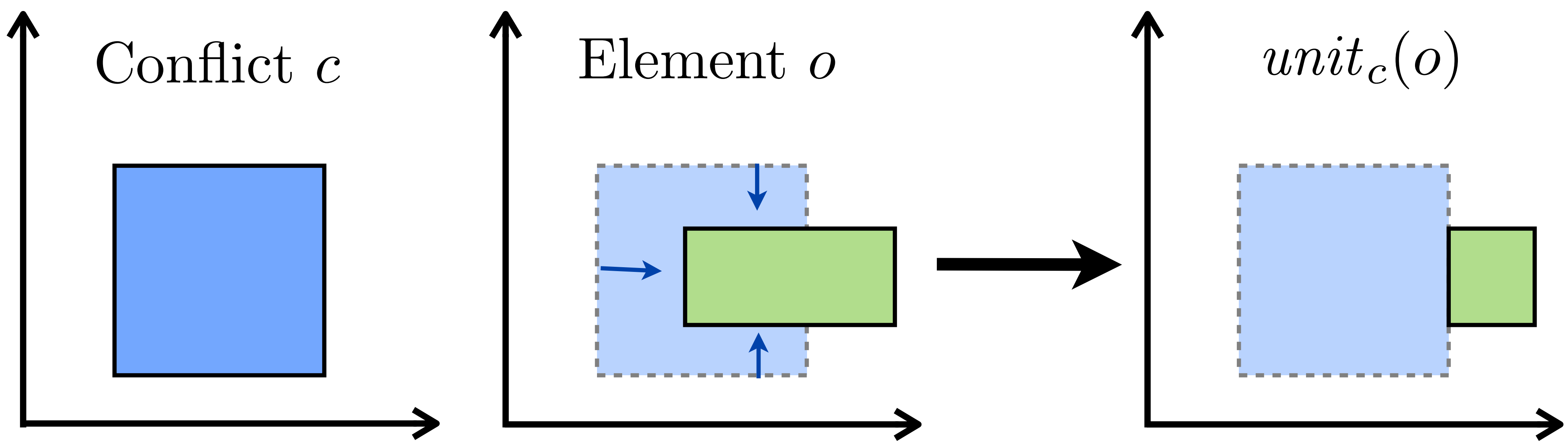


Trace abstractions:

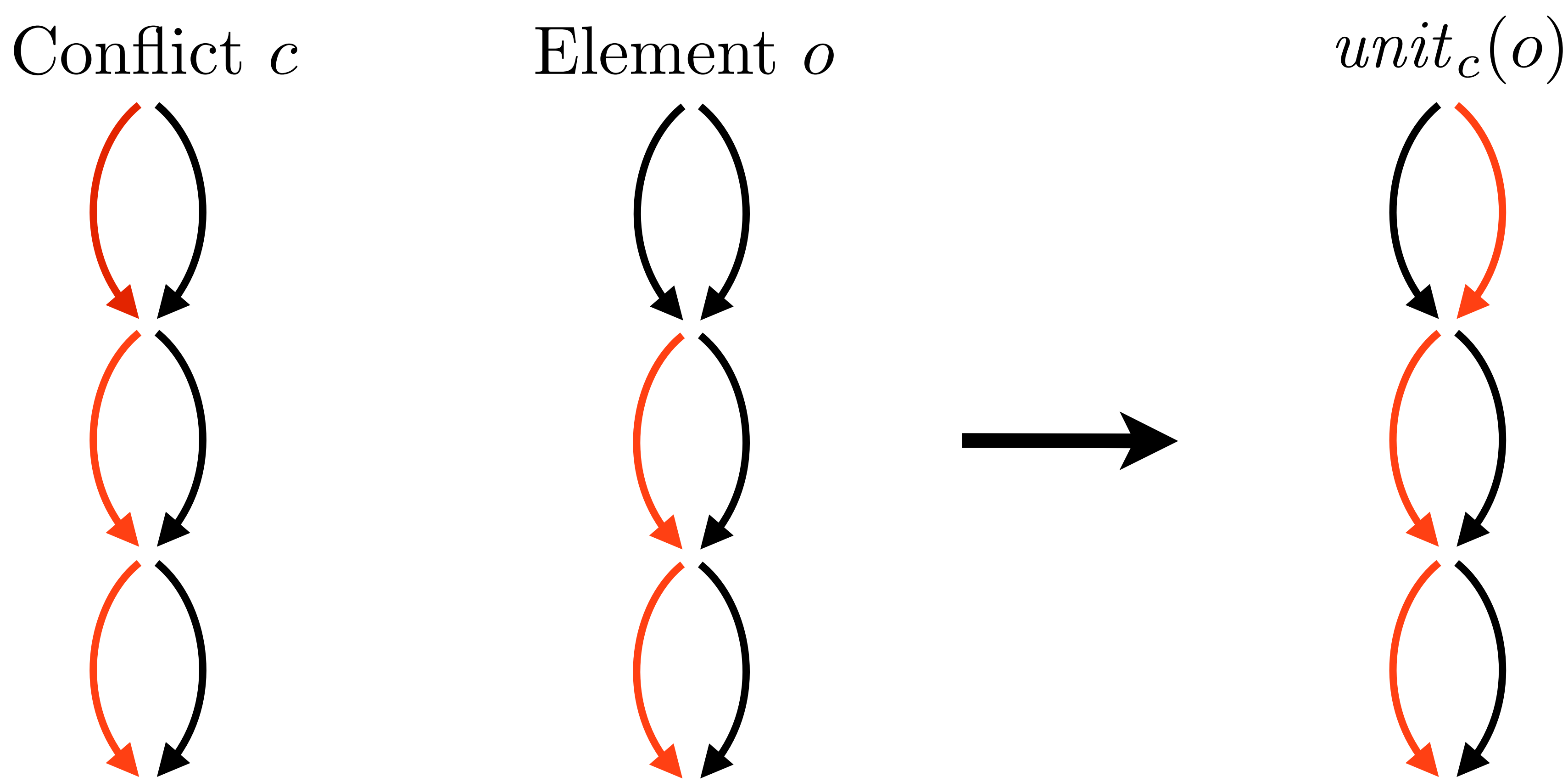


Generalised Unit Rule

Intervals



Trace abstractions:

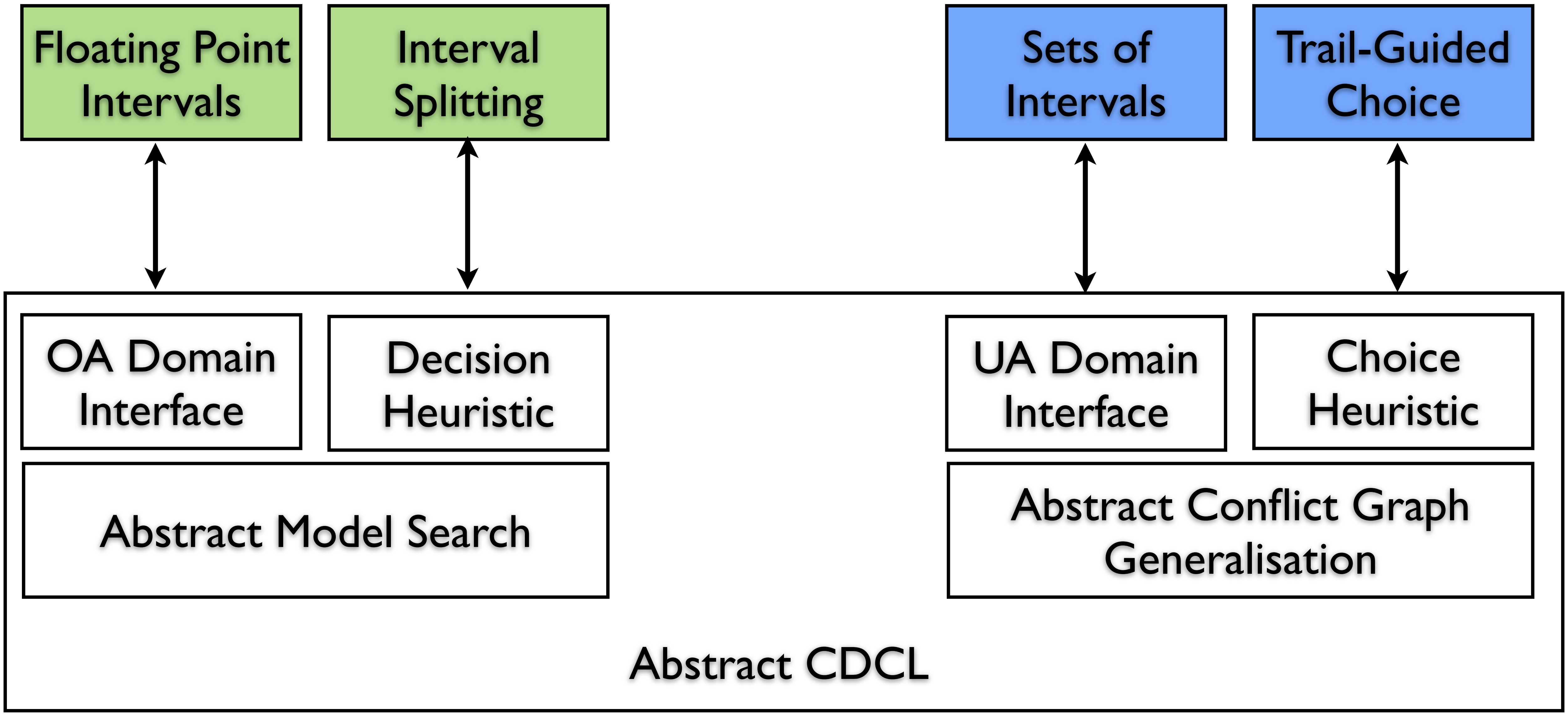


Instances/Applications

Abs. Interpretation based SMT Solver

CDCL-Style Static Analysis

An SMT Solver based on ACDCL



(Joint work with Alberto Griggio, implemented using MathSAT infrastructure)

Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

$$x = y \ \wedge \ x + y \geq 10$$

Graph nodes are meet irreducibles (e.g., half spaces)

Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

$$x = y \ \wedge \ x + y \geq 10$$

Graph nodes are meet irreducibles (e.g., half spaces)

$x \in [-\infty, 0]$

Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

$$x = y \ \wedge \ x + y \geq 10$$

Graph nodes are meet irreducibles (e.g., half spaces)

$$x \in [-\infty, 0] \longrightarrow y \in [-\infty, 0]$$

Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

$$x = y \ \wedge \ x + y \geq 10$$

Graph nodes are meet irreducibles (e.g., half spaces)

$$\begin{tikzcd}
x \in [-\infty, 0] \arrow[dr] \arrow[drr] & & \\
& y \in [-\infty, 0] \arrow[r] & \lightning
\end{tikzcd}$$

Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

$$x = y \ \wedge \ x + y \geq 10$$

Graph nodes are meet irreducibles (e.g., half spaces)

$$\begin{tikzcd}
x \in [-\infty, 0] \arrow[dr] \arrow[drr] & & \\
& y \in [-\infty, 0] \arrow[r] & \lightning \\
& \color{red}{y \in [-\infty, 7]} &
\end{tikzcd}$$

Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

$$x = y \ \wedge \ x + y \geq 10$$

Graph nodes are meet irreducibles (e.g., half spaces)

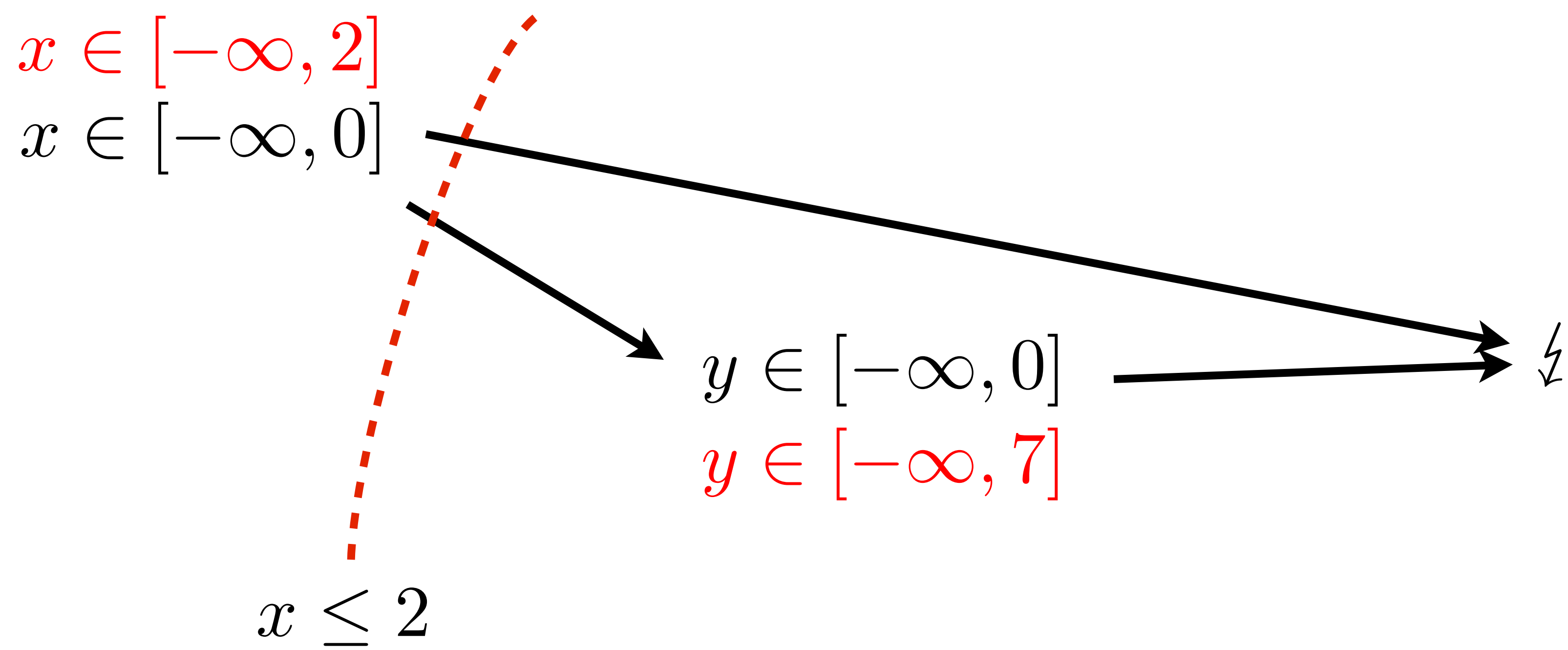
$$\begin{tikzcd}
\begin{array}{c} x \in [-\infty, 2] \\ x \in [-\infty, 0] \end{array} \arrow[dr] \arrow[drr] & & \\
& \begin{array}{c} y \in [-\infty, 0] \\ y \in [-\infty, 7] \end{array} \arrow[r] & \lightning
\end{tikzcd}$$

Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

$$x = y \ \wedge \ x + y \geq 10$$

Graph nodes are meet irreducibles (e.g., half spaces)

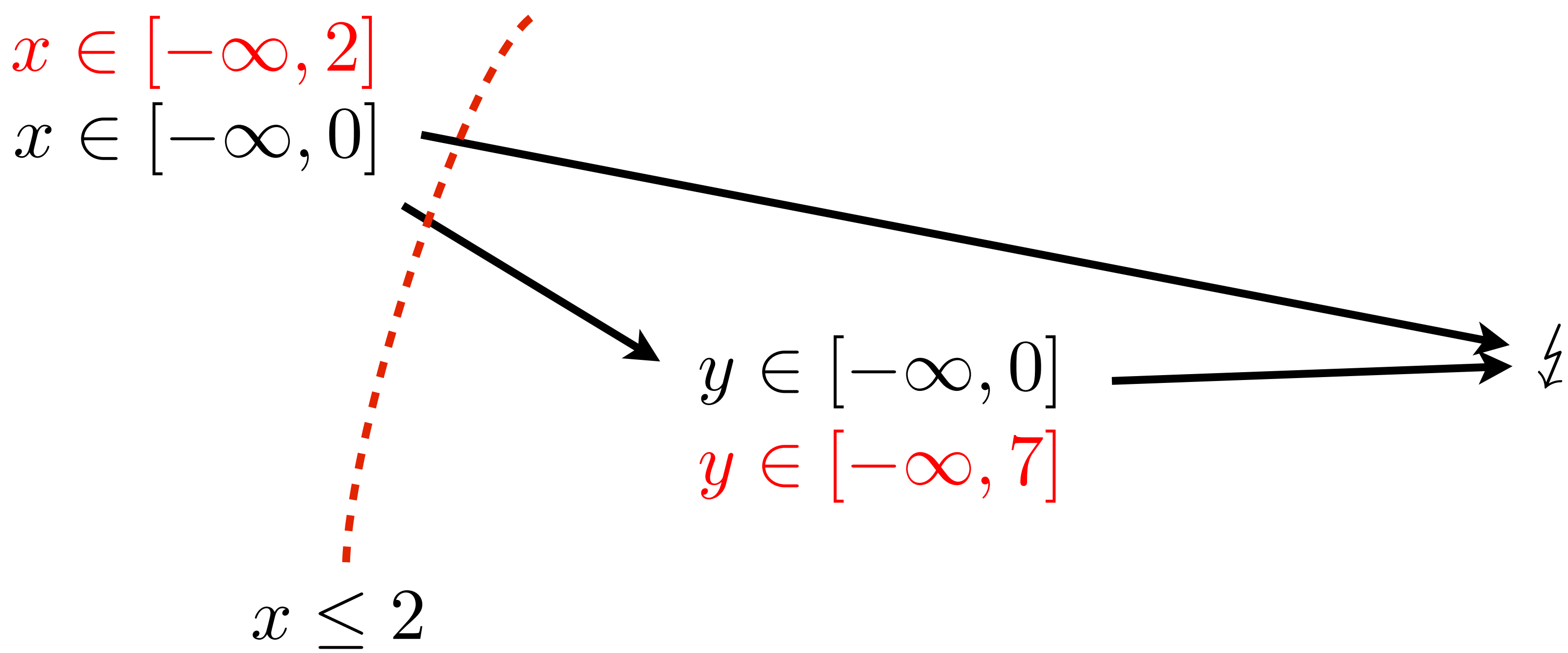


Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

$$x = y \;\wedge\; x + y \geq 10$$

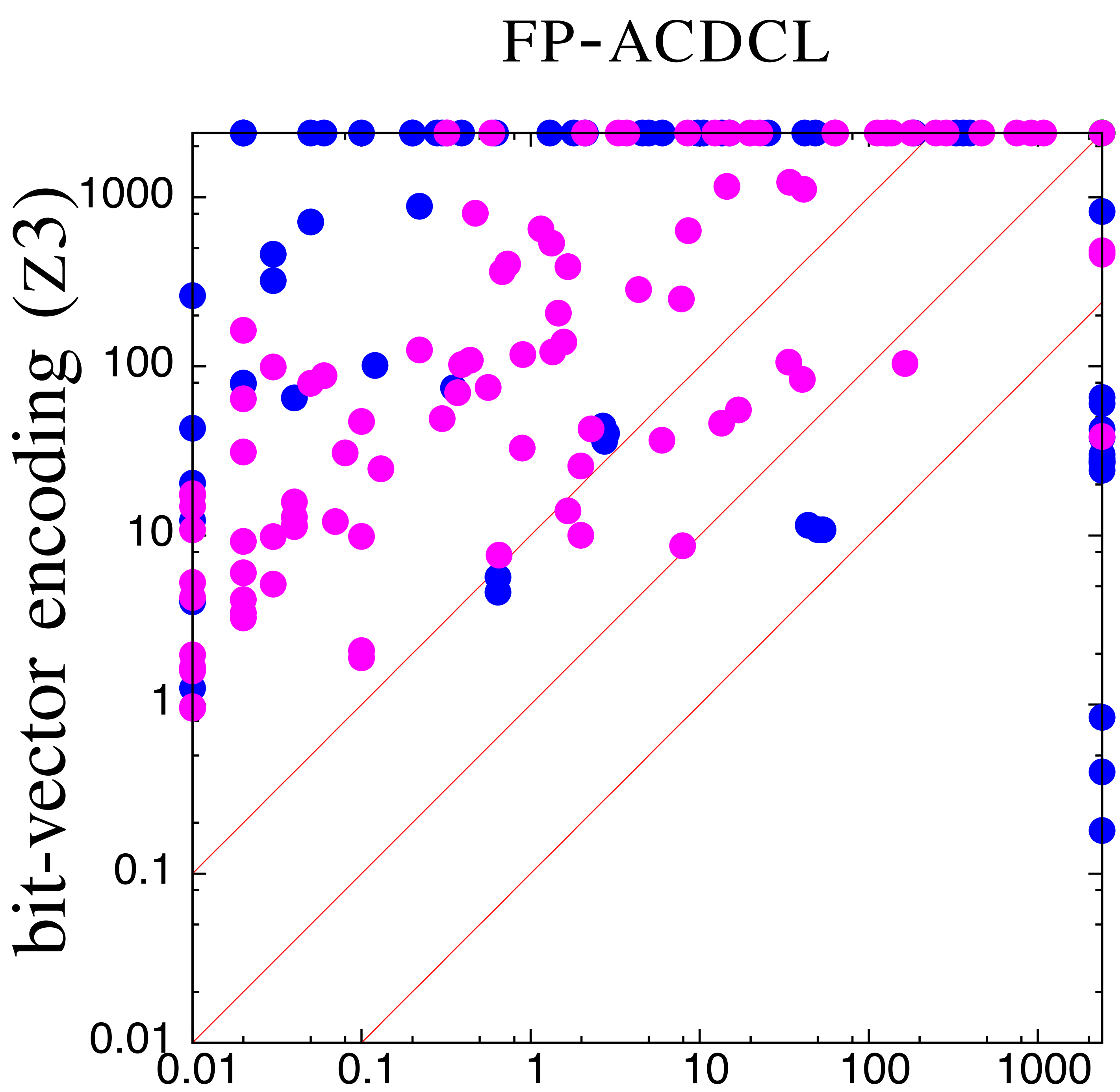
Graph nodes are meet irreducibles (e.g., half spaces)



1. Generalise each node of the conflict graph using heuristic choice

2. Cut the graph

Experiments



(Bit-vector encoding generated by MathSAT, solved by Z3)

ACDCL for Programs

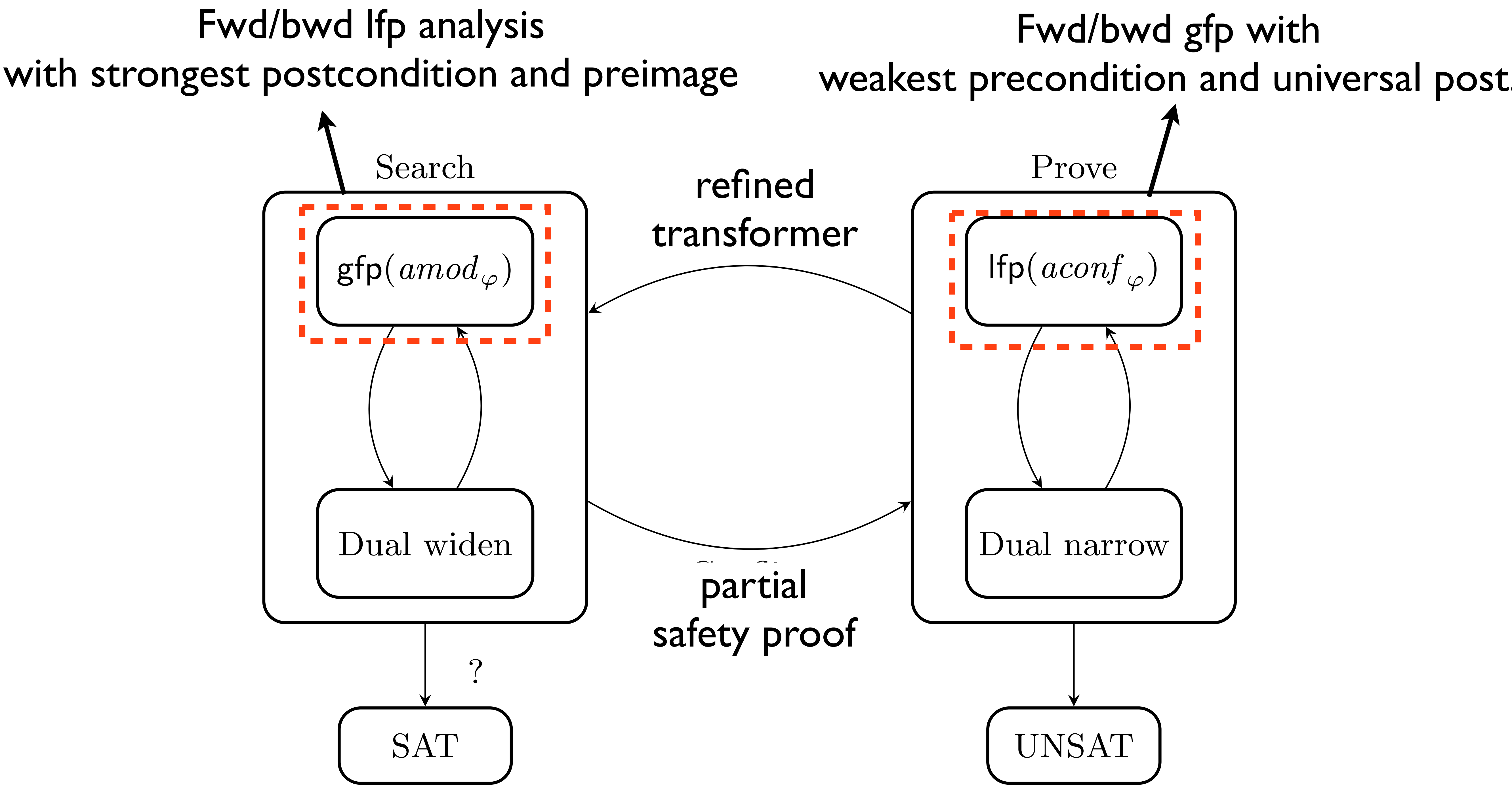
Treat program analysis as a logical problem:

$\pi \models P$ *iff* trace π is an erroneous trace generated by program P

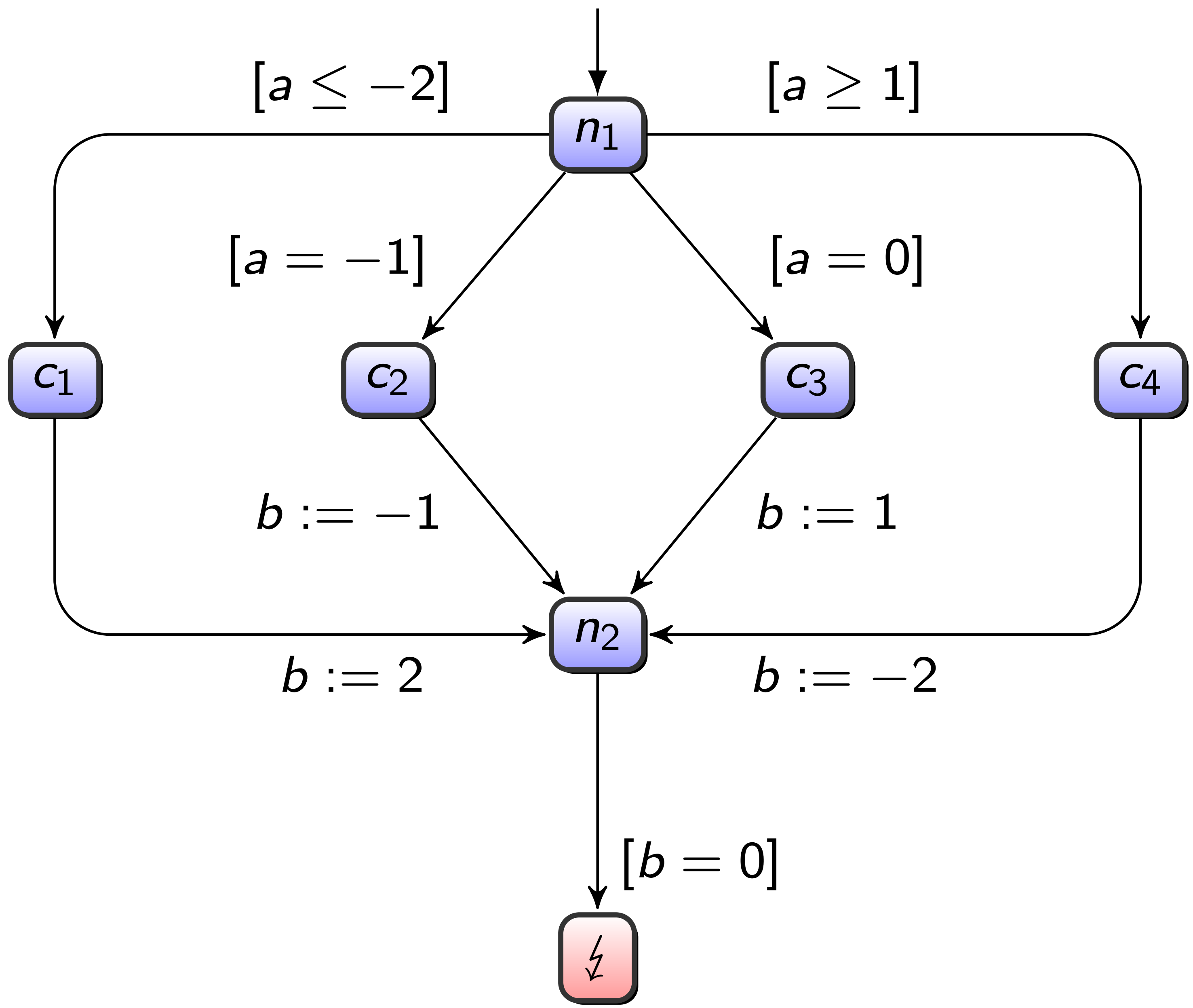
ACDCL for Programs

Treat program analysis as a logical problem:

$$\pi \models P \ \textit{iff} \ \text{trace } \pi \text{ is an erroneous trace generated by program } P$$

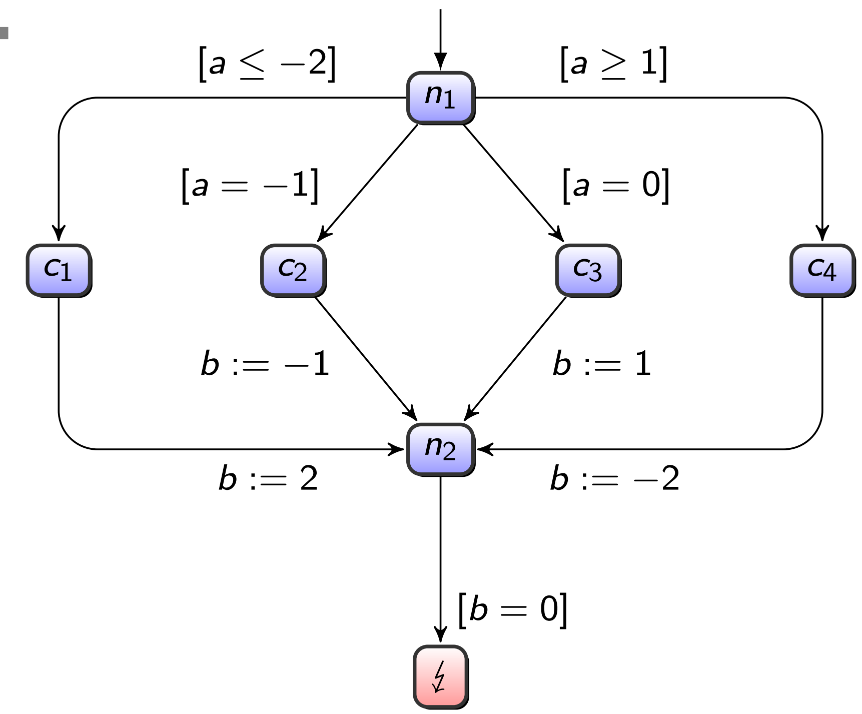


Example 1: Interval Conflict Graphs

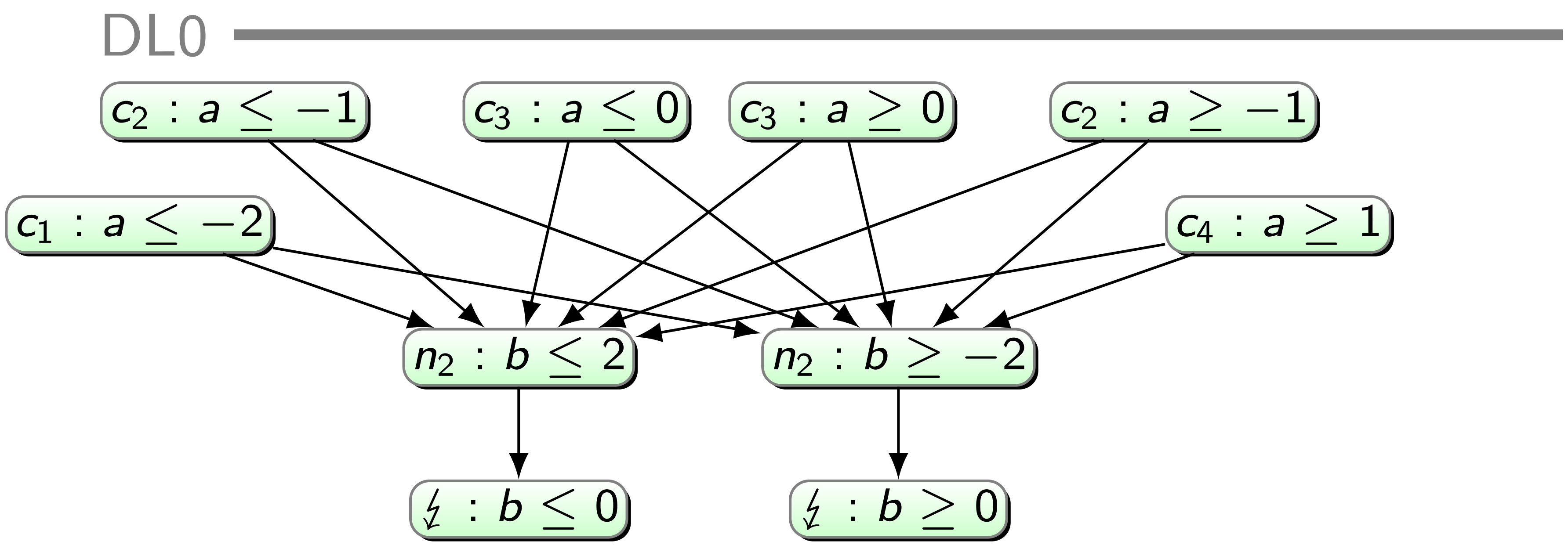


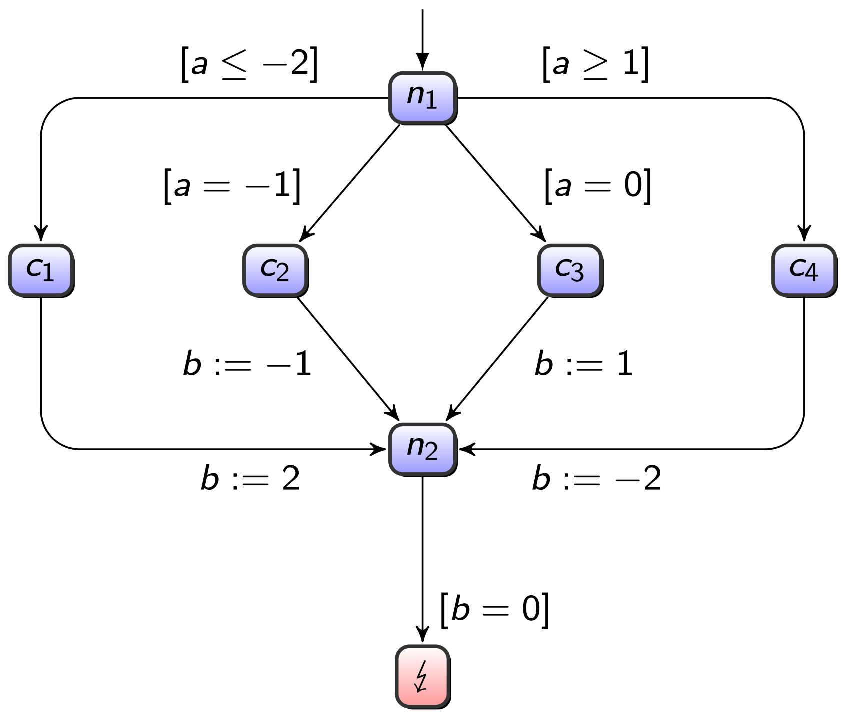
Example 1: Interval Conflict Graphs

DL0

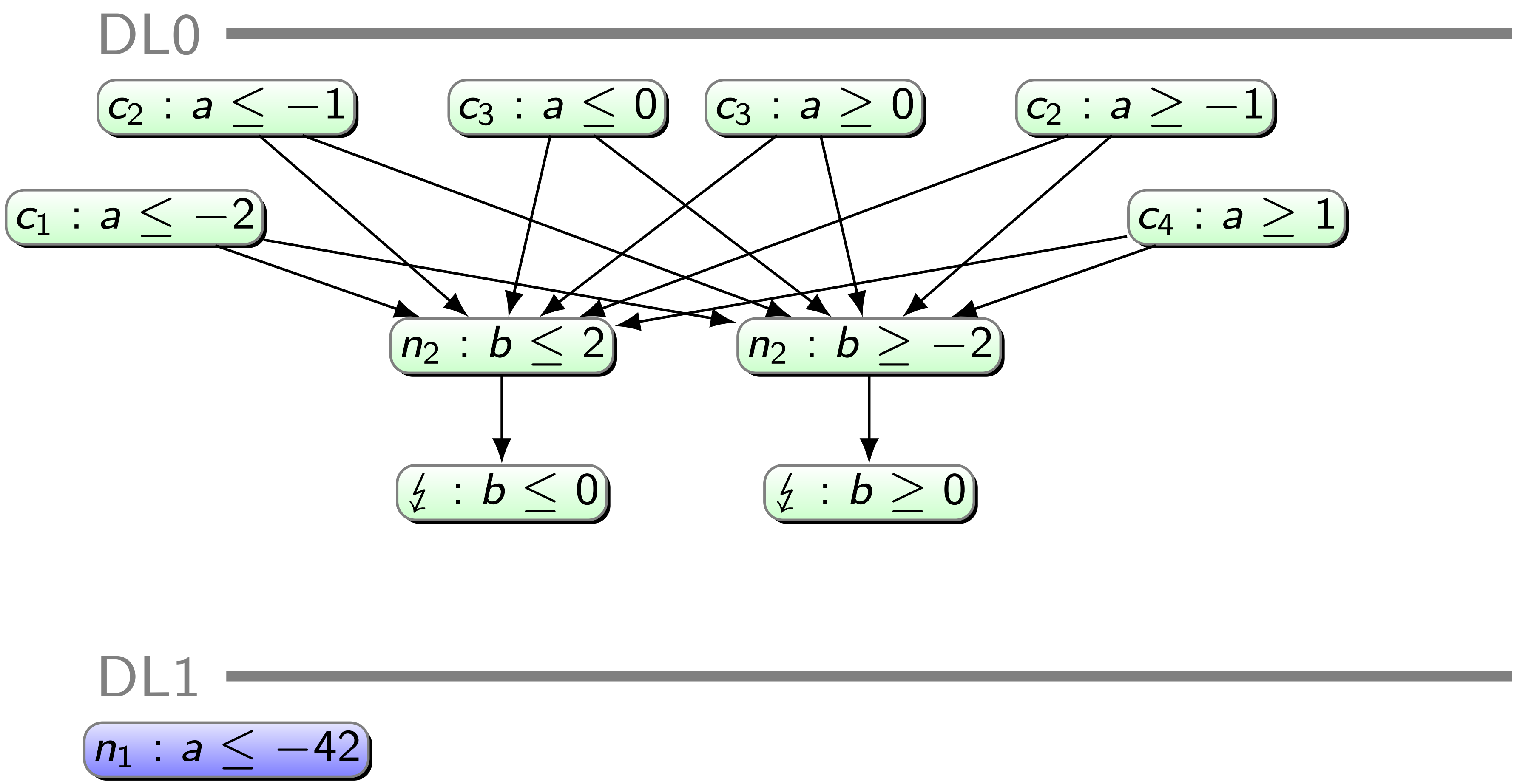


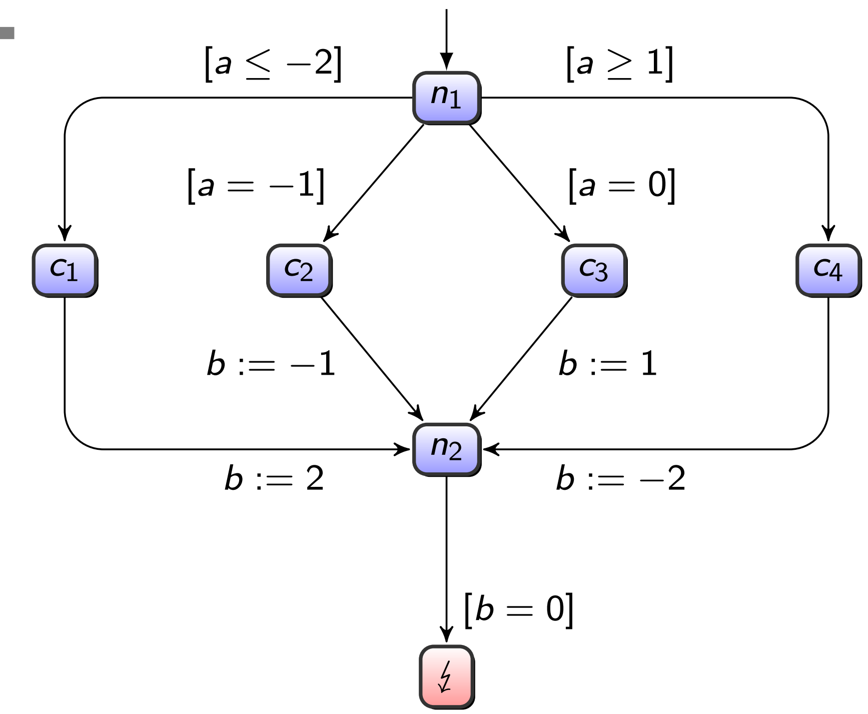
Example 1: Interval Conflict Graphs



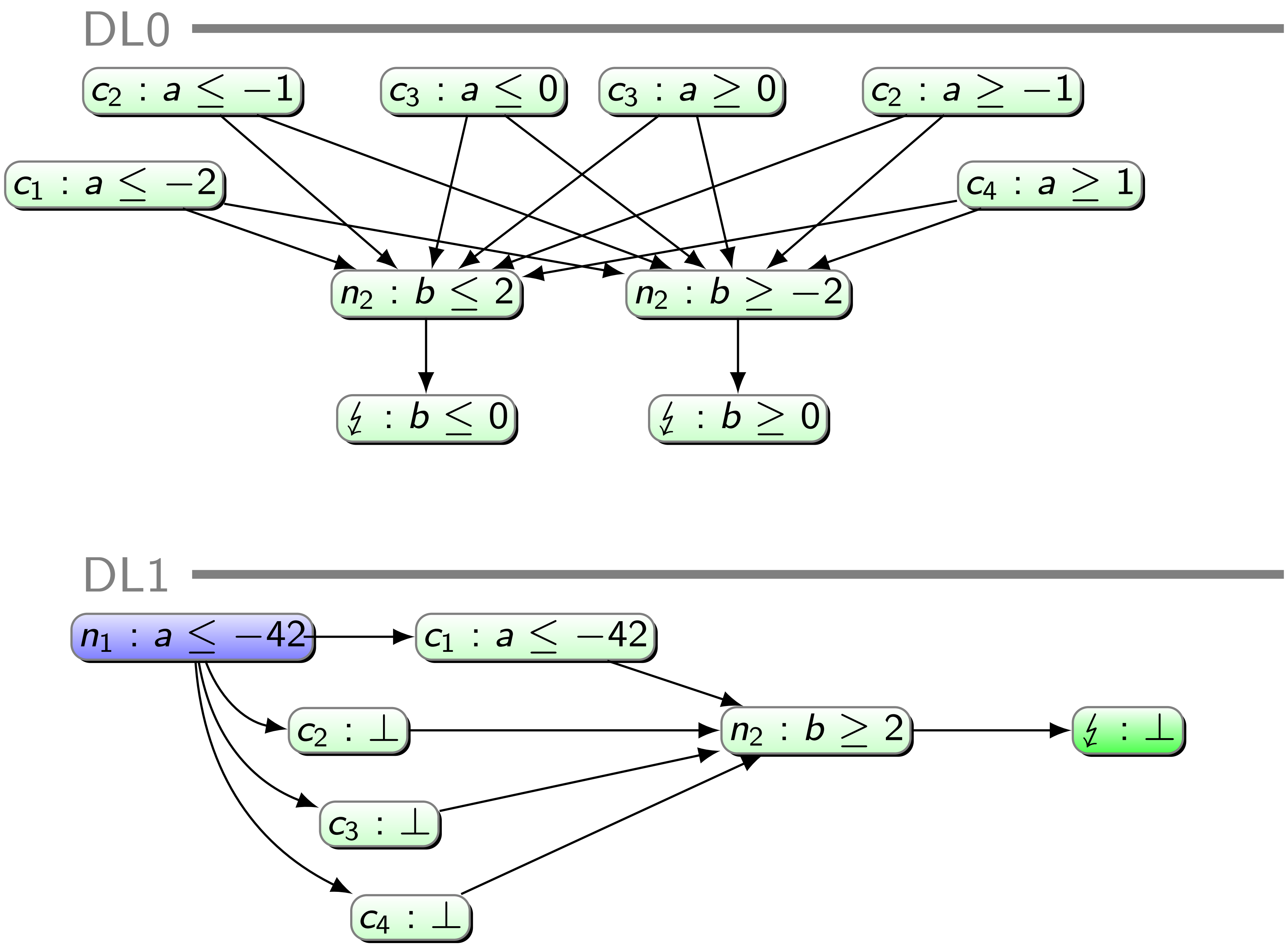


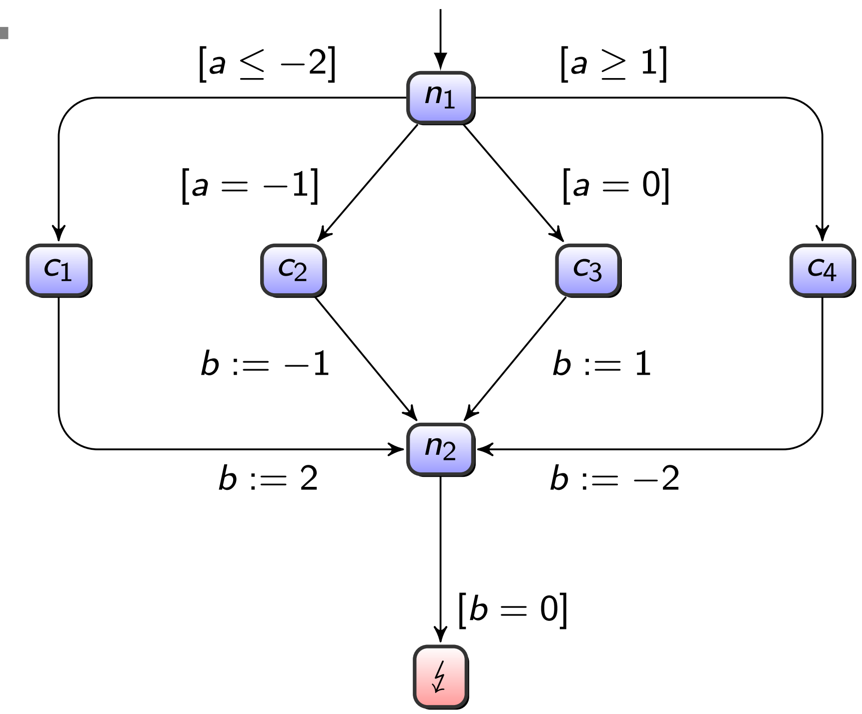
Example 1: Interval Conflict Graphs





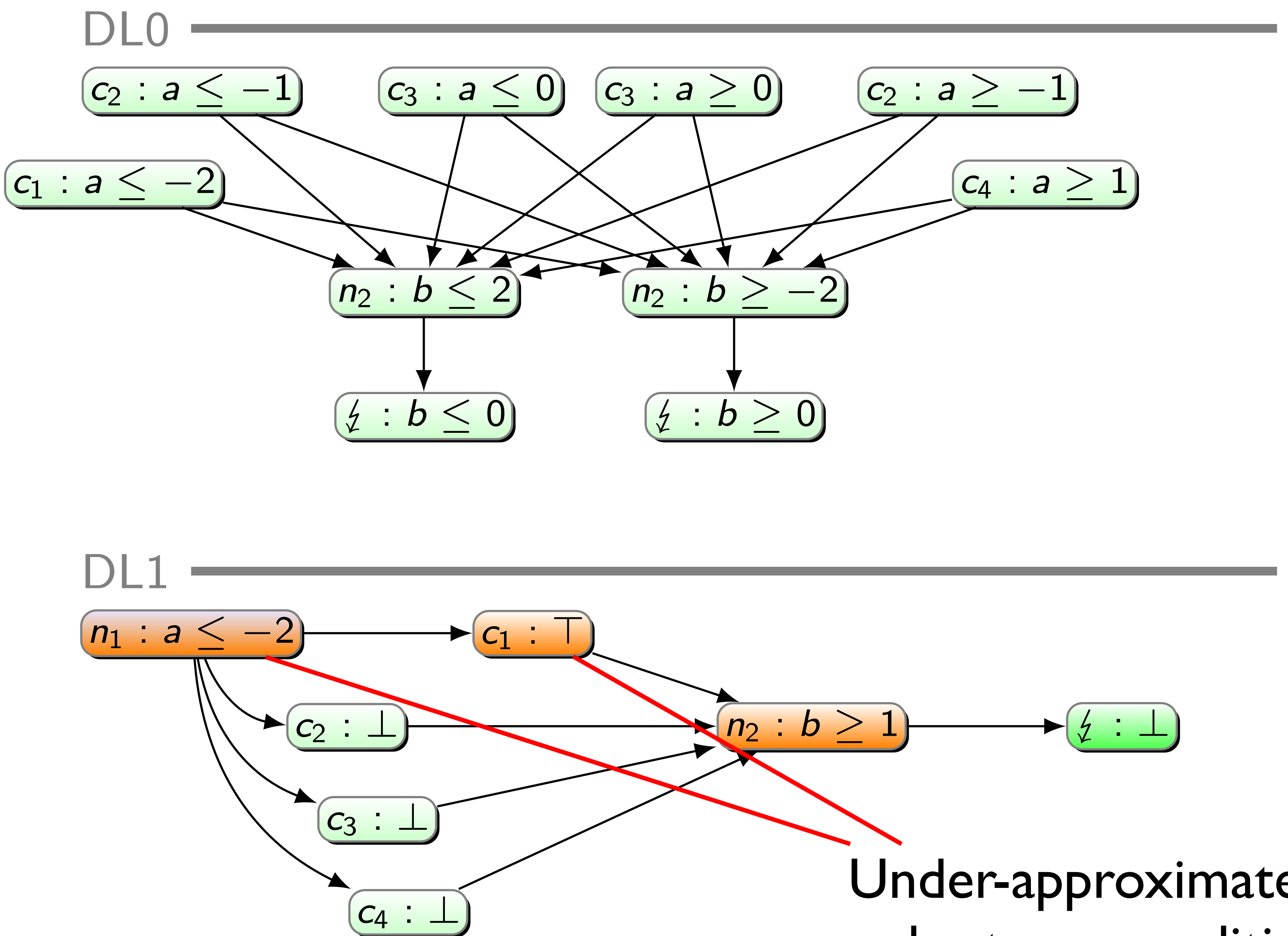
Example 1: Interval Conflict Graphs

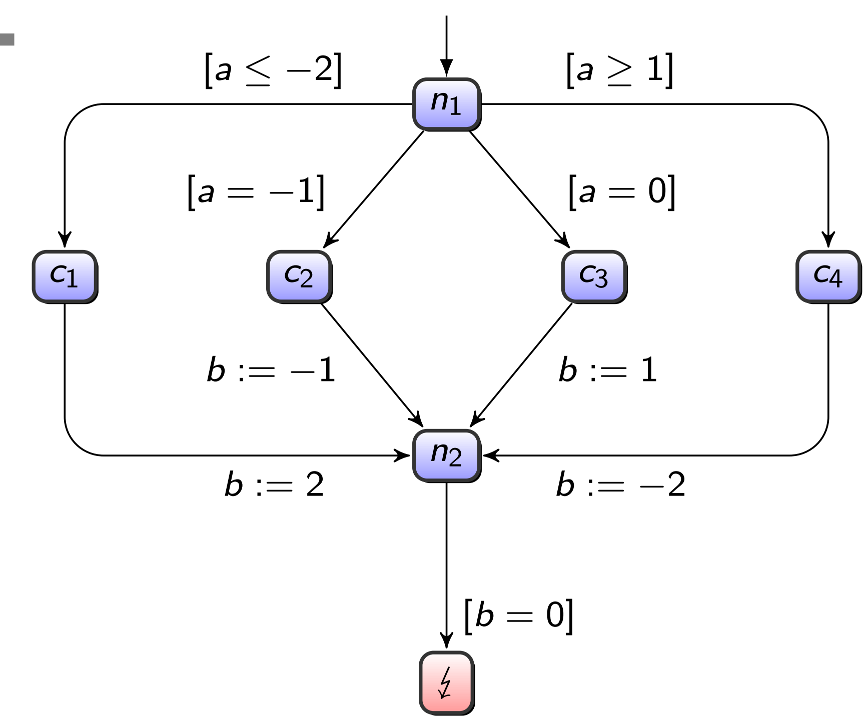




SAFE

Example 1: Interval Conflict Graphs

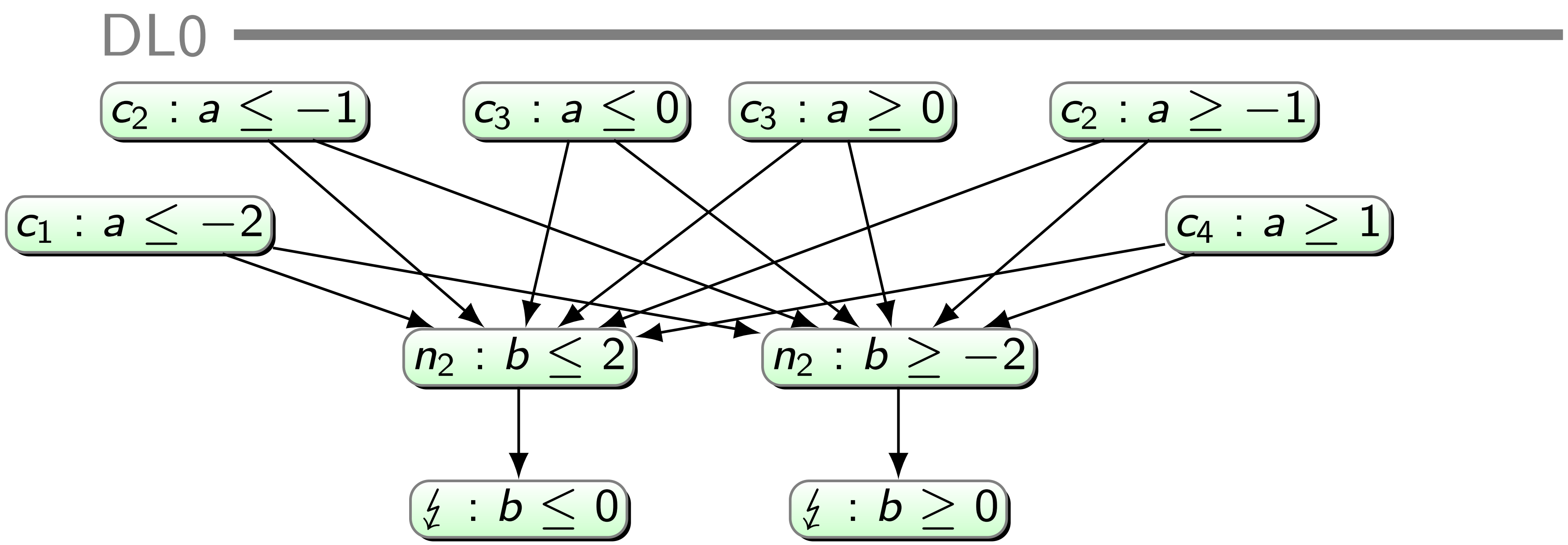


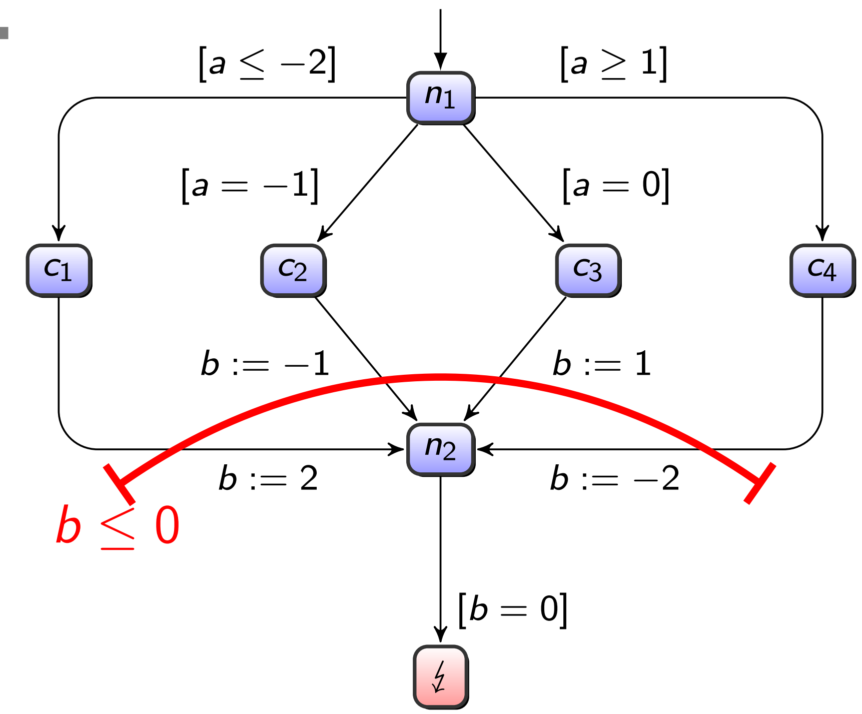


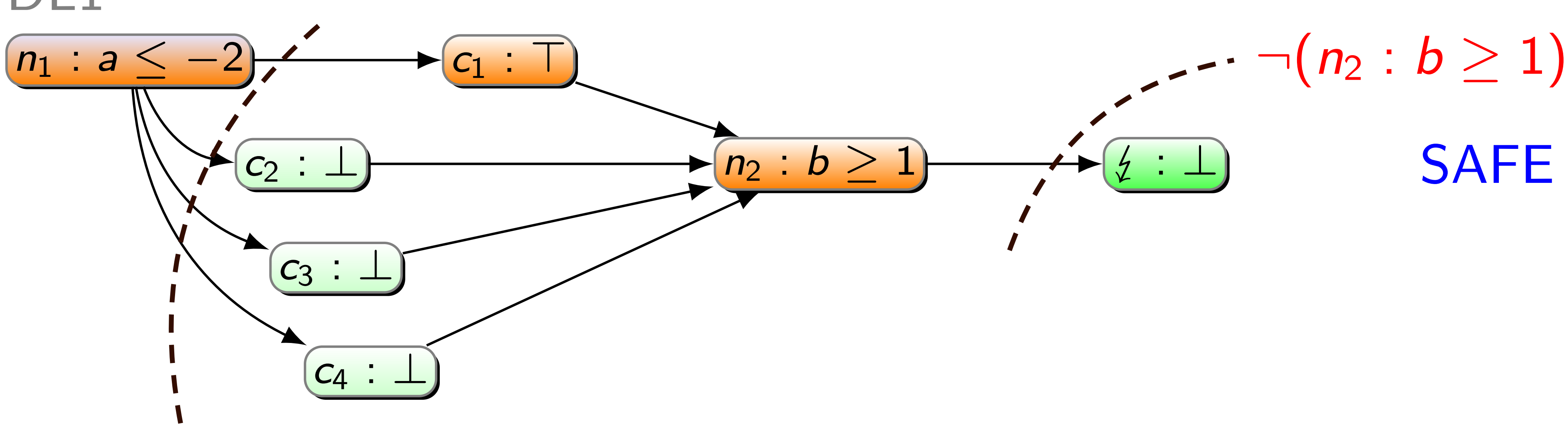
SAFE $\rightarrow$ Generalise!

Under-approximate
weakest pre-condition

Example 1: Interval Conflict Graphs



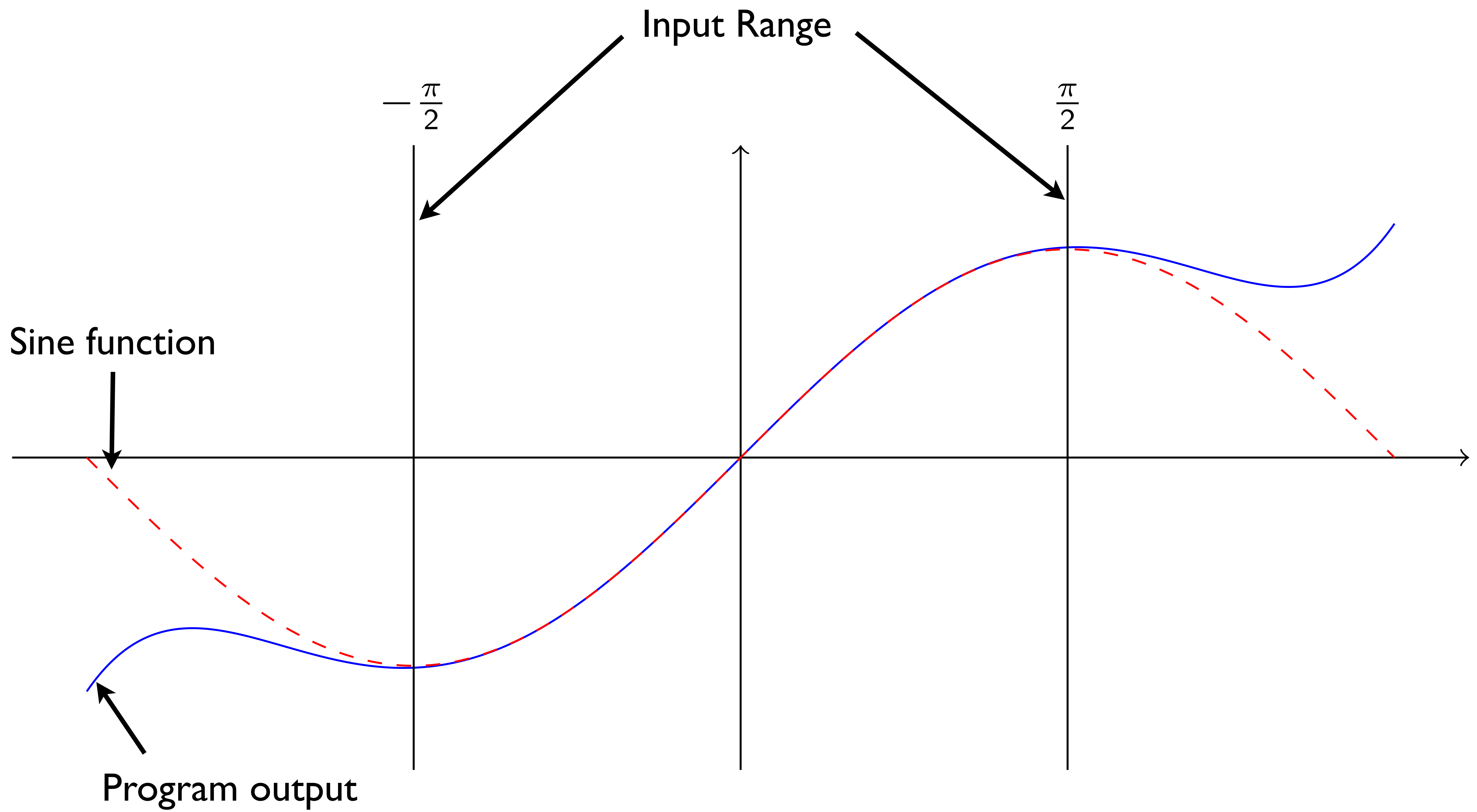




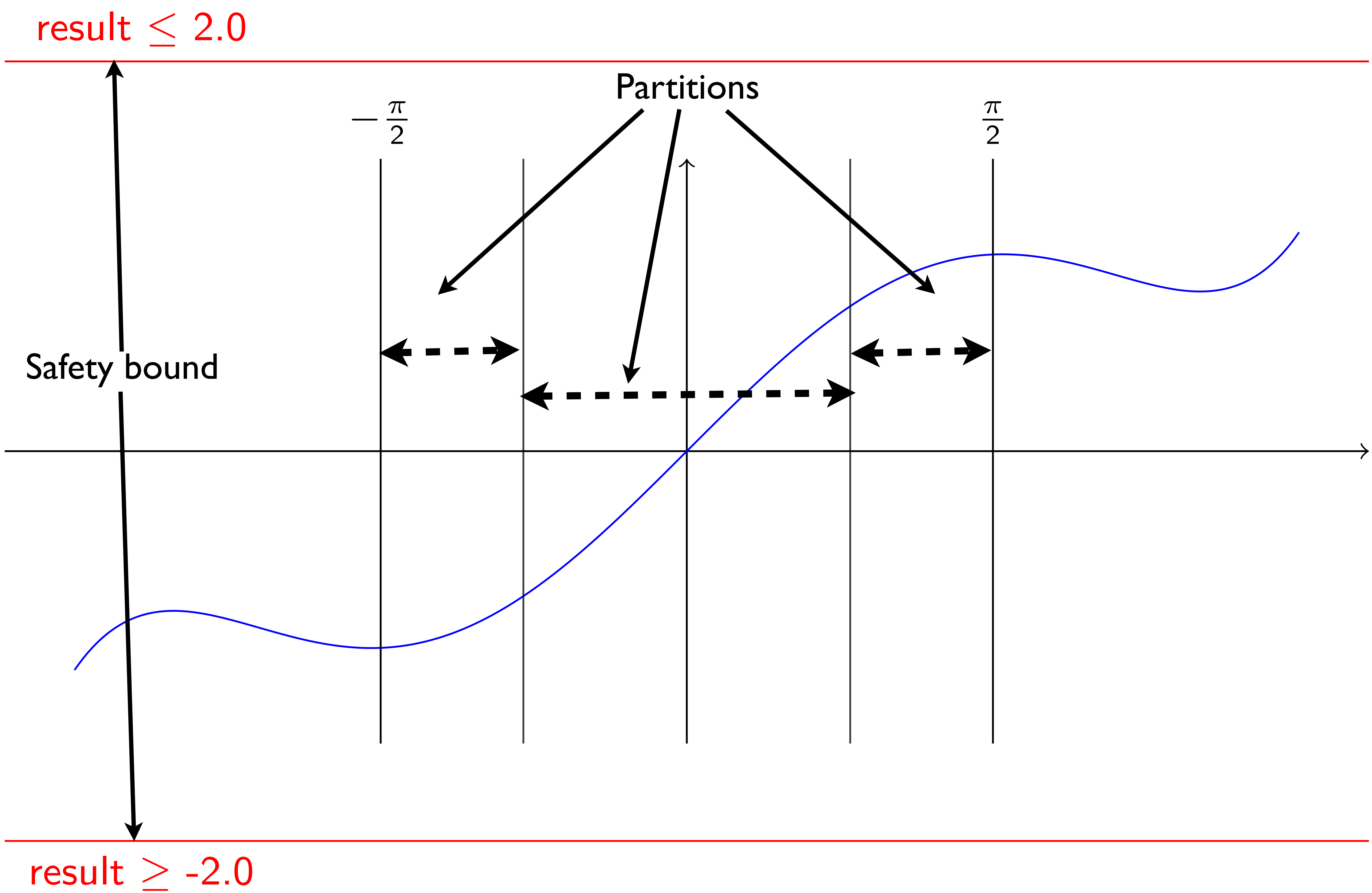
SAFE $\rightarrow$ find cut

ACDCL "intelligently" decomposes the problem

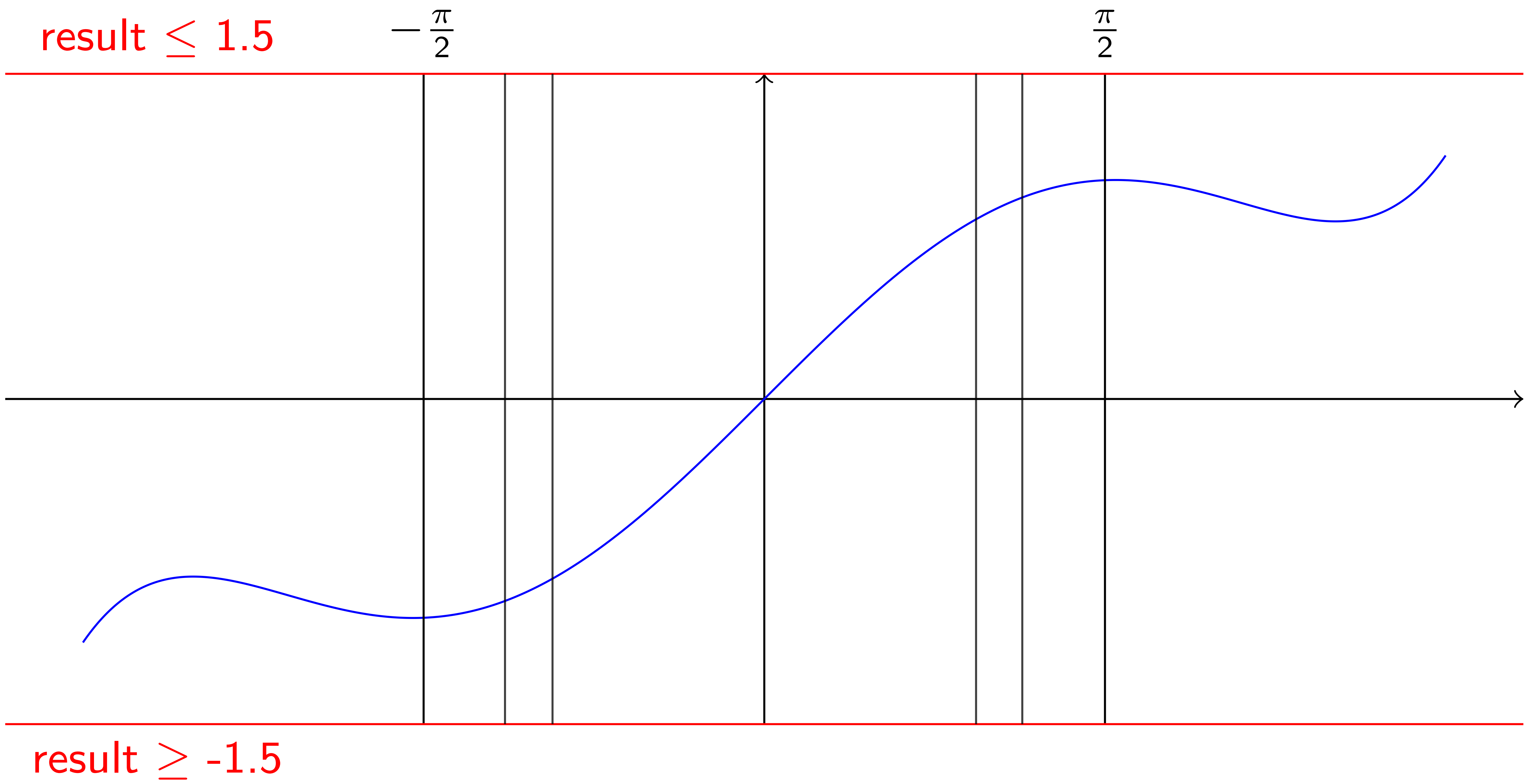
Example 2: Problem Dependent Decomposition



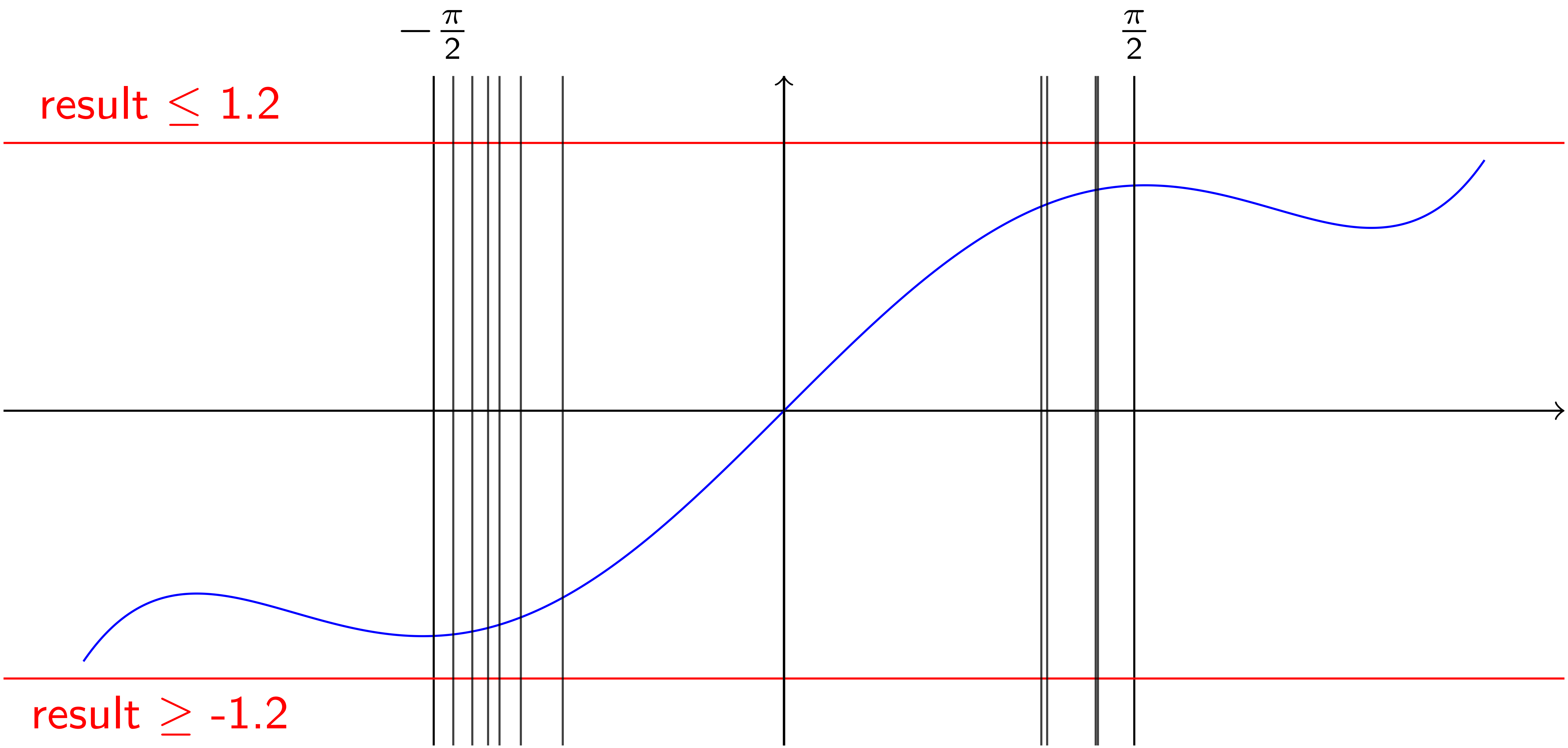
Example 2: Problem Dependent Decomposition



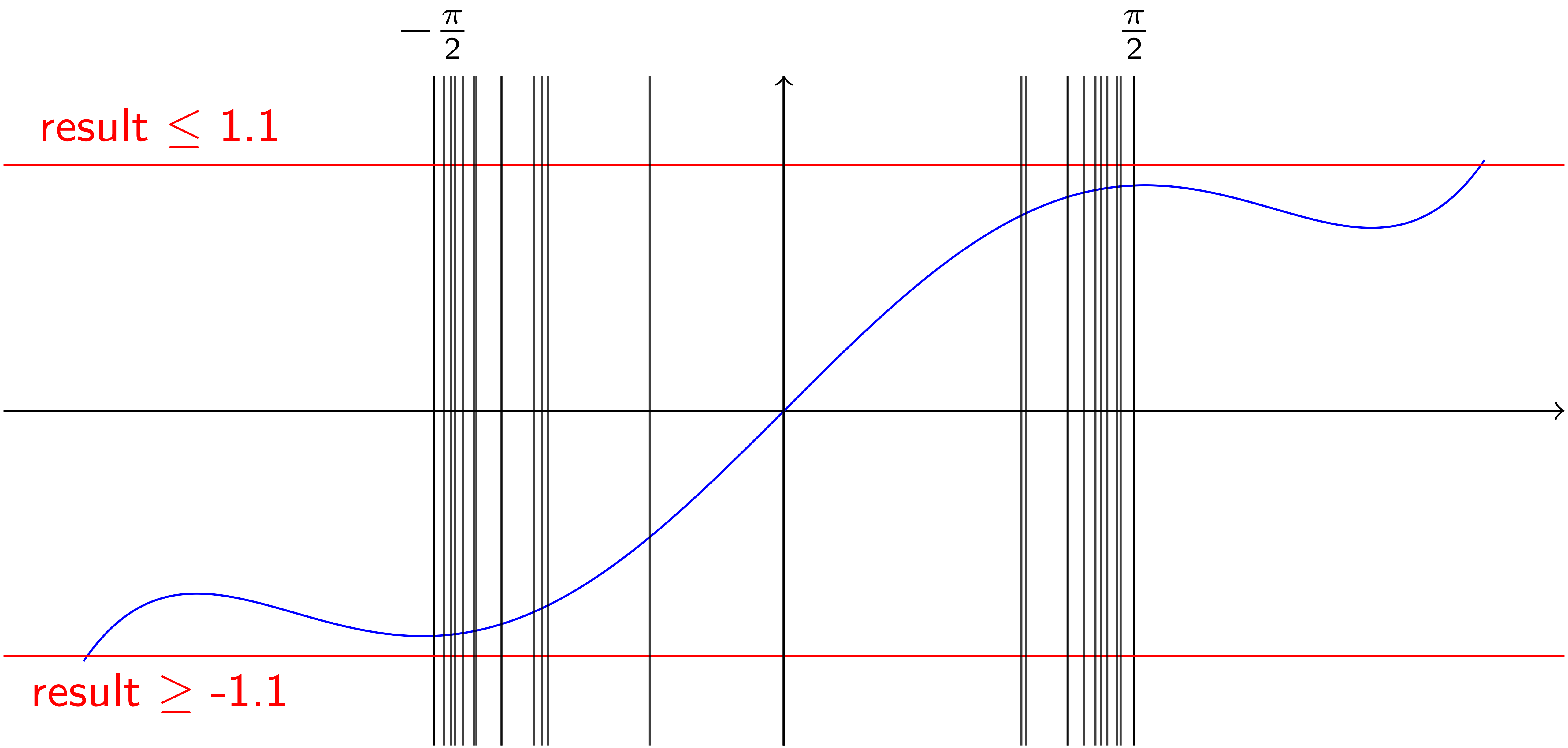
Example 2: Problem Dependent Decomposition



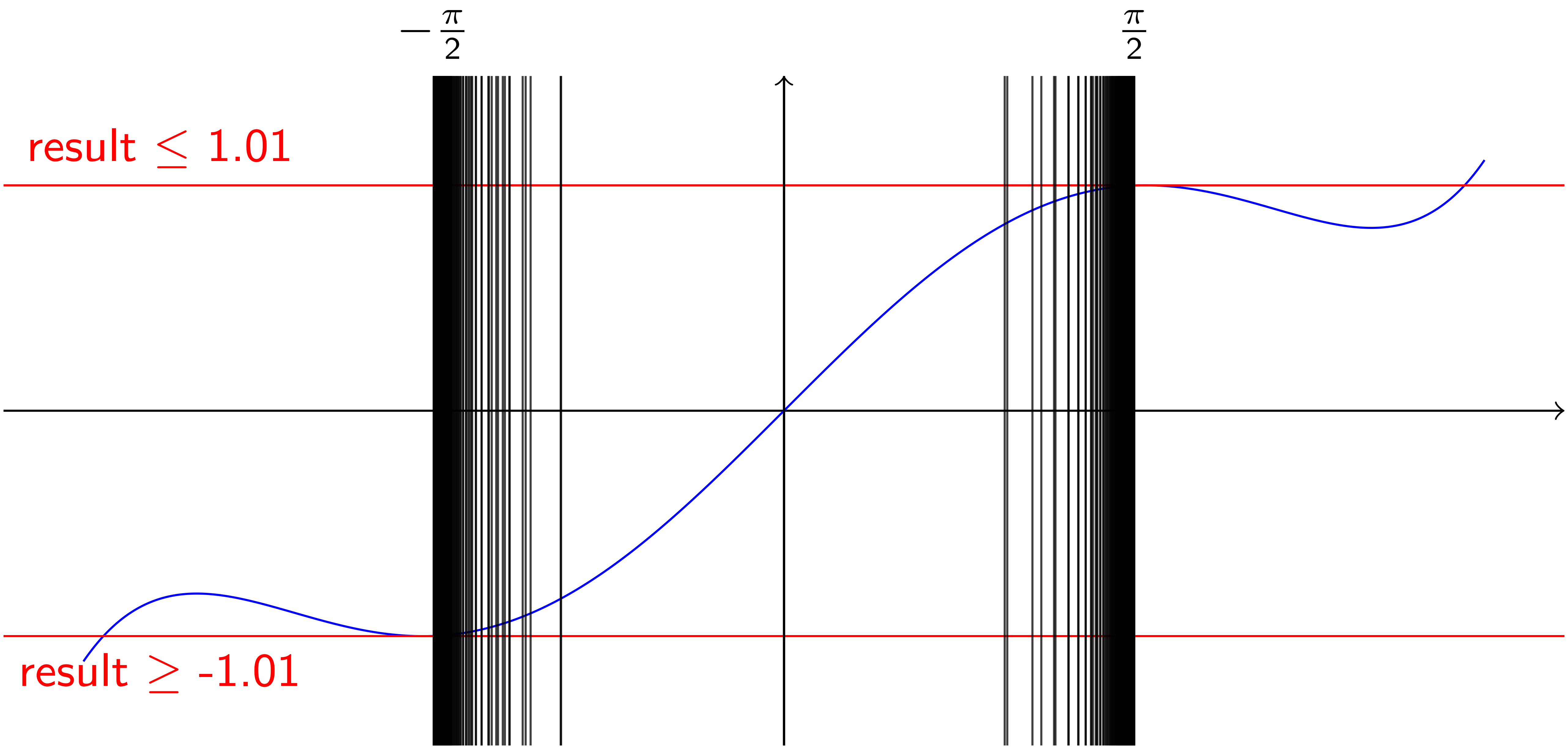
Example 2: Problem Dependent Decomposition



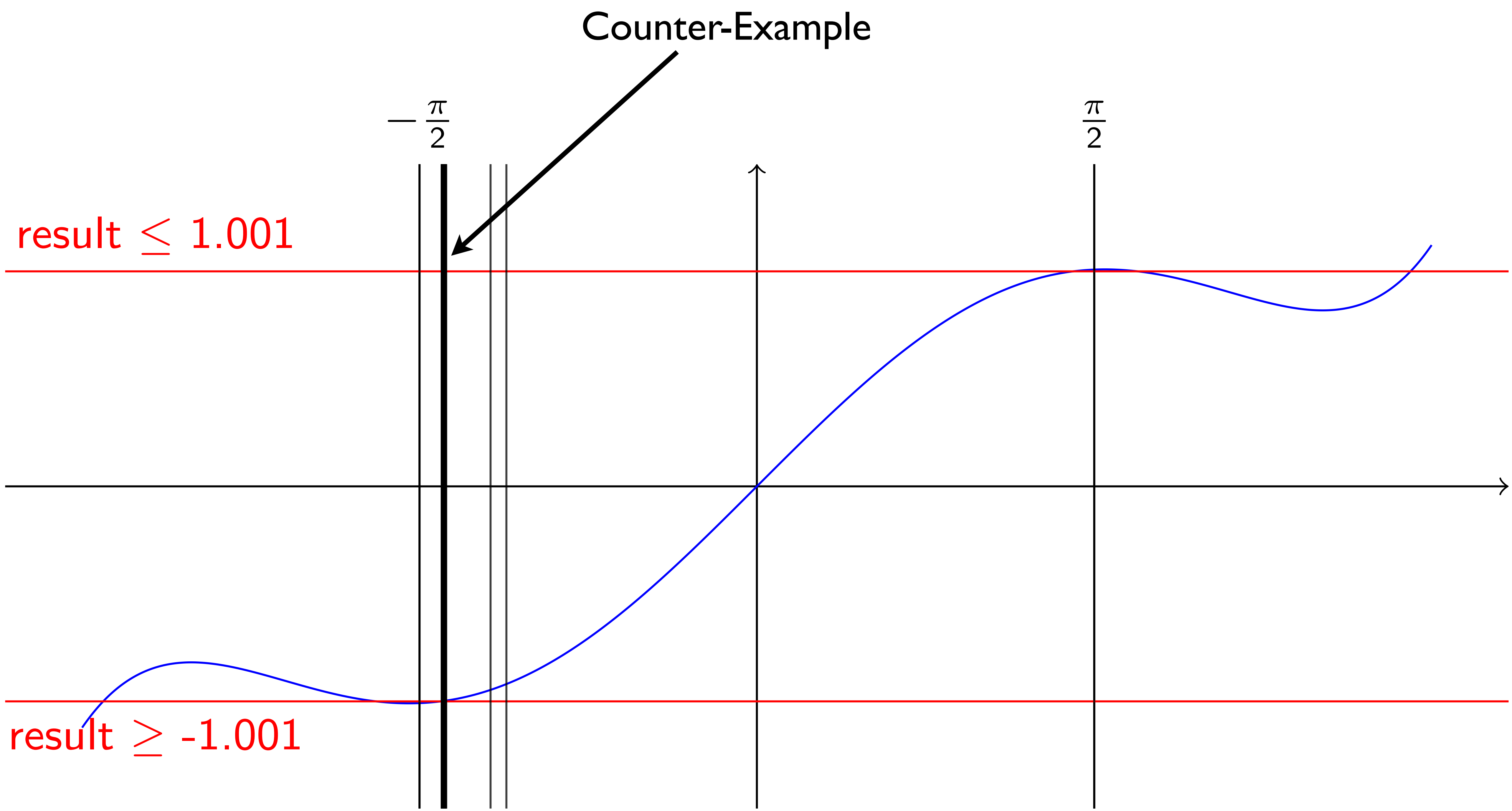
Example 2: Problem Dependent Decomposition



Example 2: Problem Dependent Decomposition



Example 2: Problem Dependent Decomposition



Intelligent decomposition of the analysis

And never the twain shall meet?

Oh, East is East, and West is West, and never the twain shall meet,
Till Earth and Sky stand presently at God's great Judgment Seat;
But there is neither East nor West, Border, nor Breed, nor Birth,
When two strong men stand face to face, tho' they come from the ends of the earth!

Thanks for your attention!