

From Literals to Atomic Constraints

Generalising Conflict-Driven Clause Learning for Constraint Programming

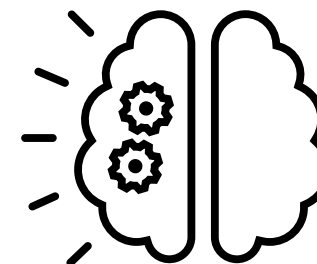
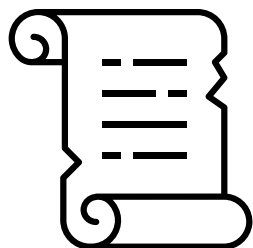
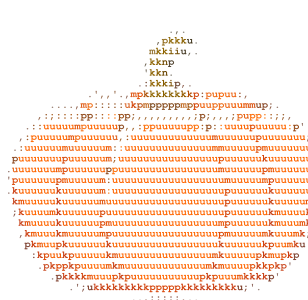
Imko Marijnissen, Maarten Flippo, and Emir Demirović

CP '26

CPMPy

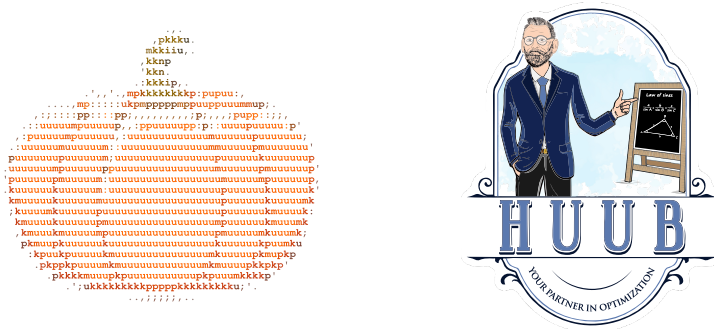
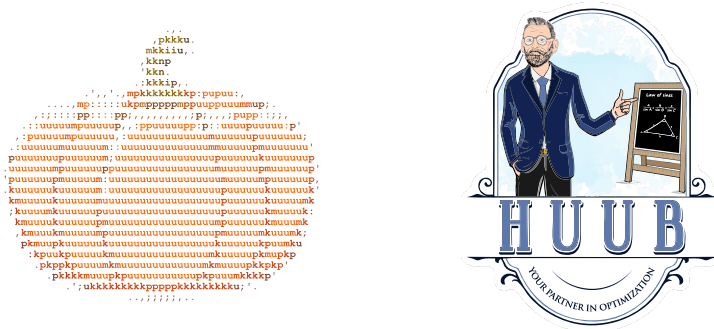


XCSP³



Model

Solve

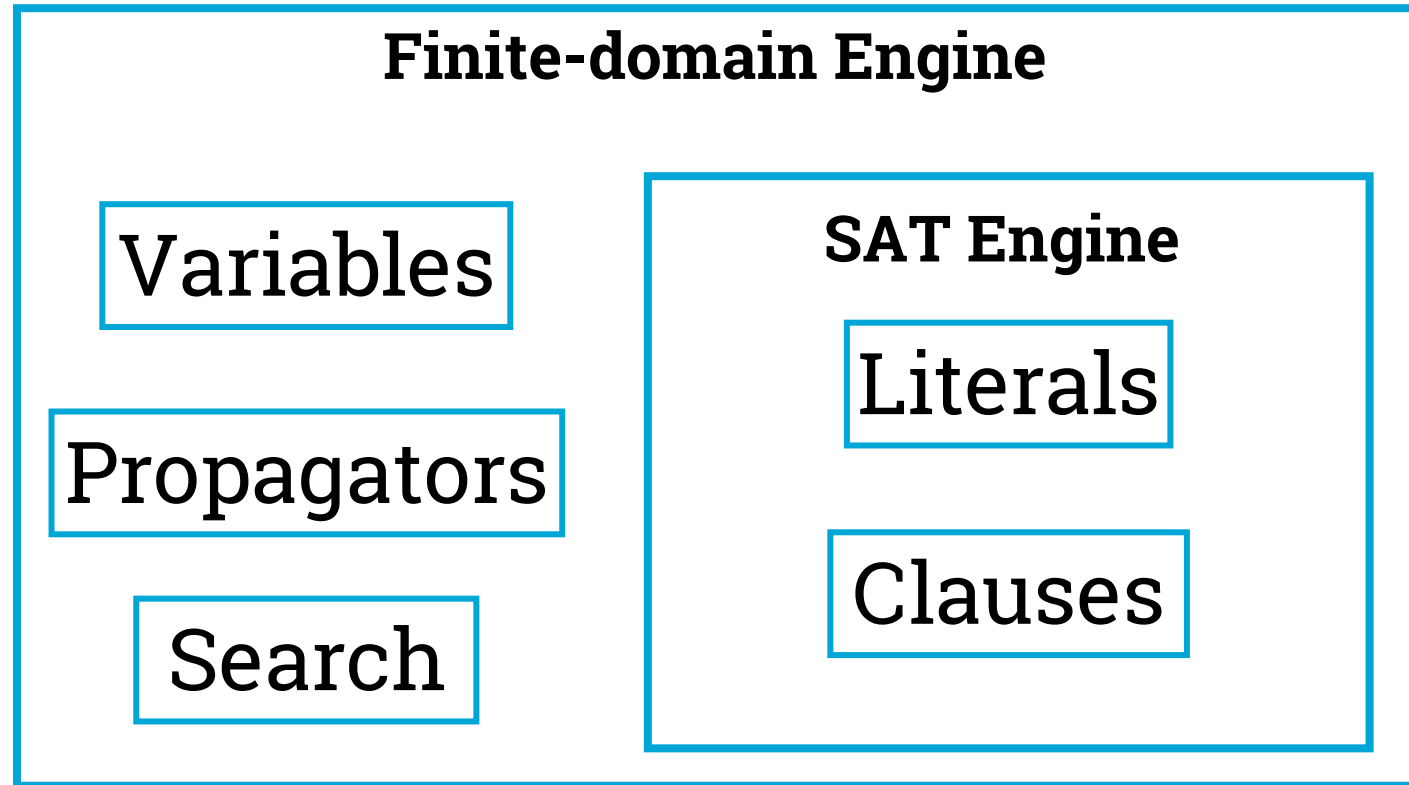


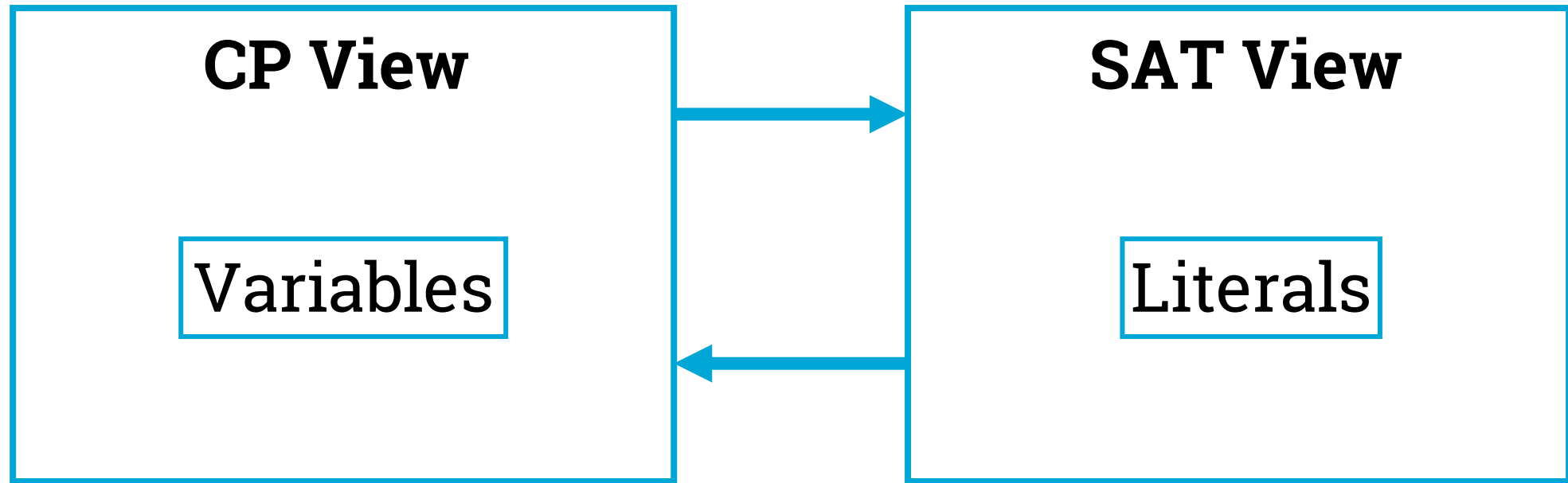
Finite-domain Engine

Variables

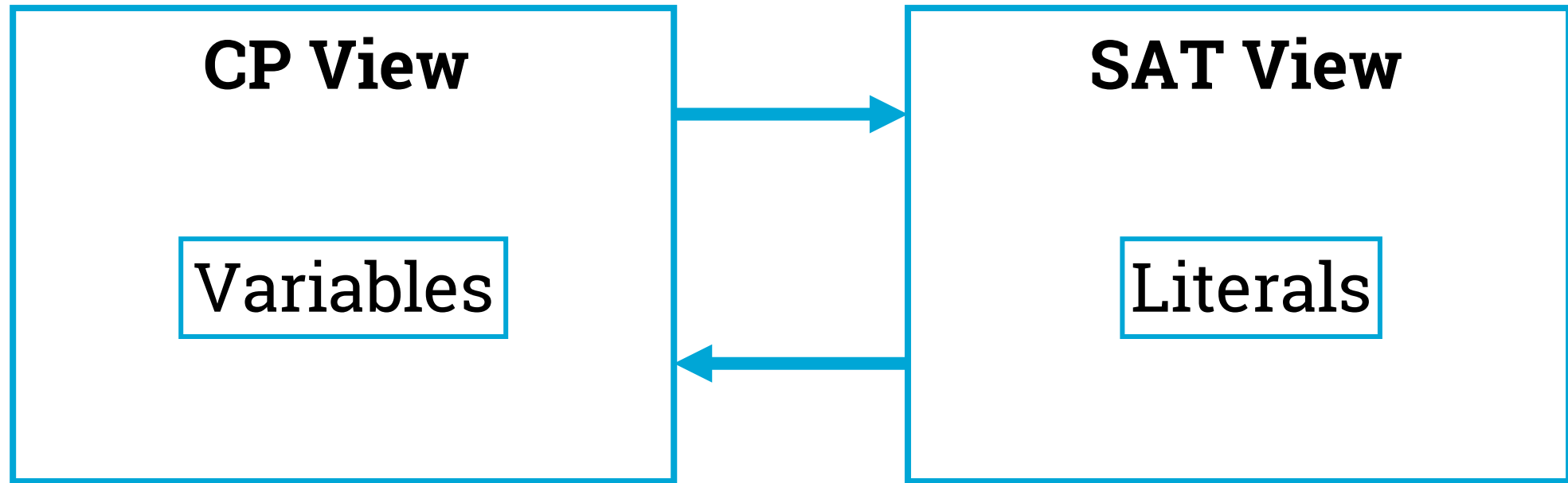
Propagators

Search





Transfer SAT
techniques
to CP!



How is the
SAT view
maintained?



Exploit
rich CP
information?



Literals

$$x \in \{1, 2, 3, 4\}$$

$$l_1 \leftrightarrow [x \geq v]$$

$$l_2 \leftrightarrow [x = v]$$

Literals

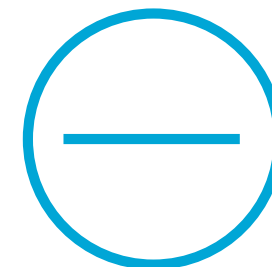
**Explain
Inferences**



$$y > x$$

$$[x \geq 0] \rightarrow [y \geq 1]$$

**Learn Nogoods
via Conflict
Analysis**



$$[x \geq 2] \wedge [y \neq 5] \rightarrow \perp$$

Literals

x

Literals

$x \in [0, 100000]$



Our Contributions

Analysis SAT
View in
Solvers



Remove SAT
View



Generalise
Nogood
Propagation/
Learning

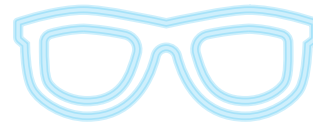


Our Contributions

Analysis SAT
View in
Solvers



Remove SAT
View



Generalise
Nogood
Propagation/
Learning



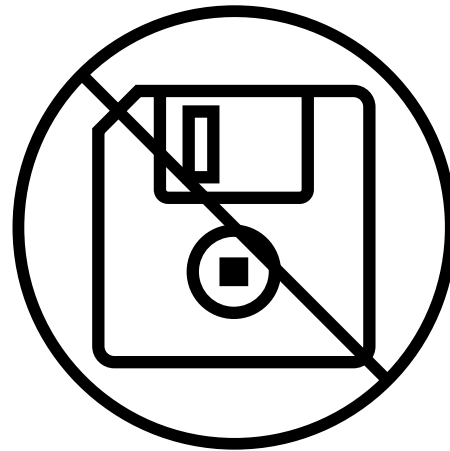
How do solvers maintain the SAT view?

Chuffed $\neq$ Huub $\neq$ OR-Tools $\neq$ Tempo $\neq$ Geas



Tradeoffs to adhere to
SAT view

Chuffed $\neq$ Huub $\neq$ OR-Tools $\neq$ Tempo $\neq$ Geas

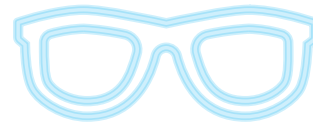


Our Contributions

Analysis SAT
View in
Solvers



Remove SAT
View



Generalise
Nogood
Propagation/
Learning



Our Contributions

Analysis SAT
View in
Solvers

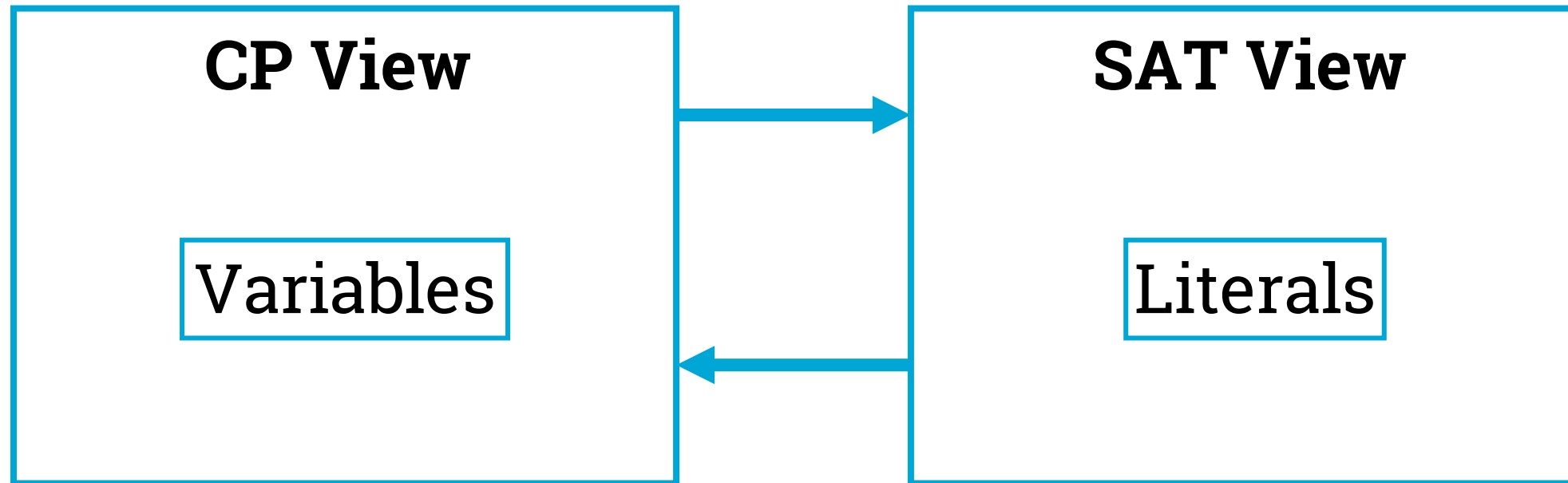


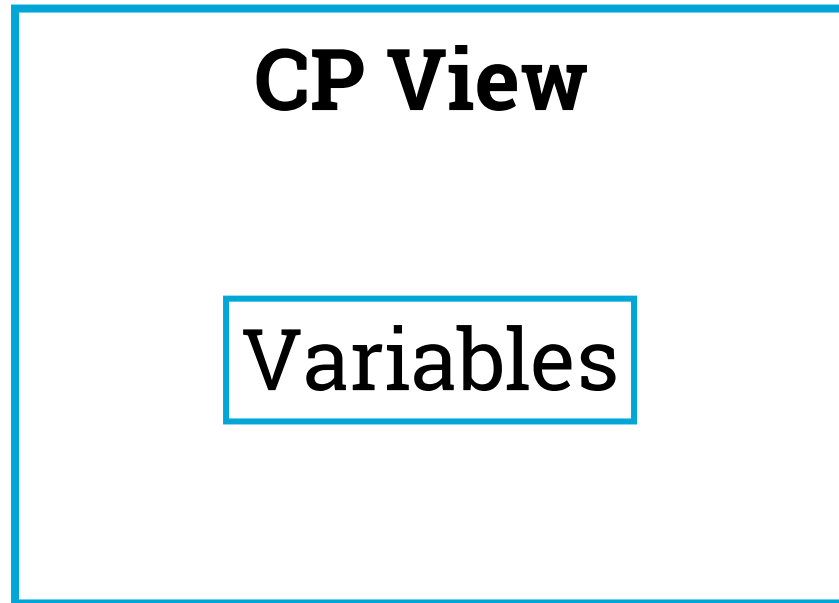
Remove SAT
View



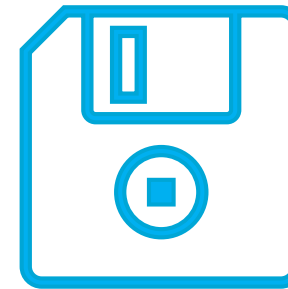
Generalise
Nogood
Propagation/
Learning

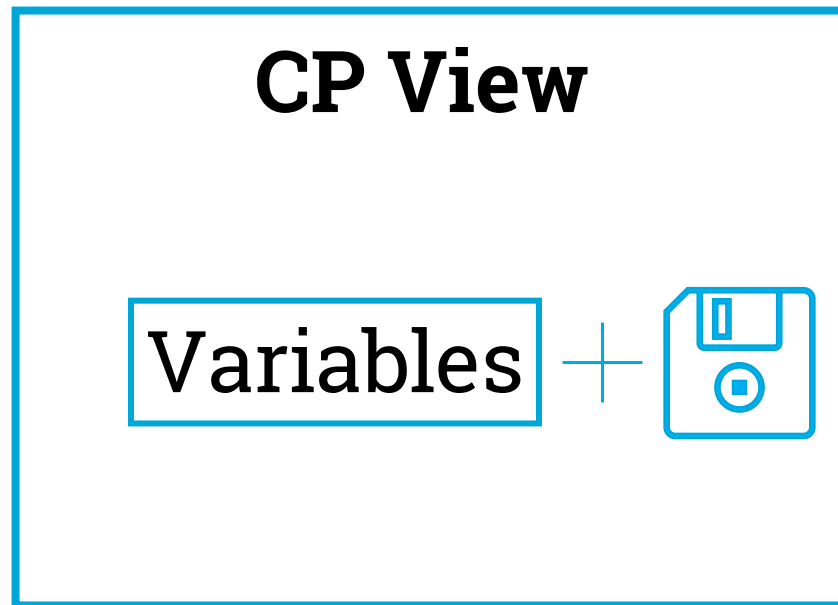




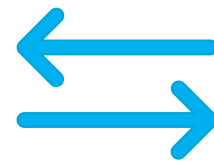


+





Overcome
Tradeoffs



Remove
Channelling

Our Contributions

Analysis SAT
View in
Solvers



Remove SAT
View



Generalise
Nogood
Propagation/
Learning



Our Contributions

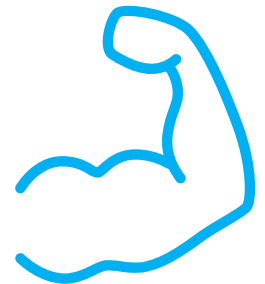
Analysis SAT
View in
Solvers

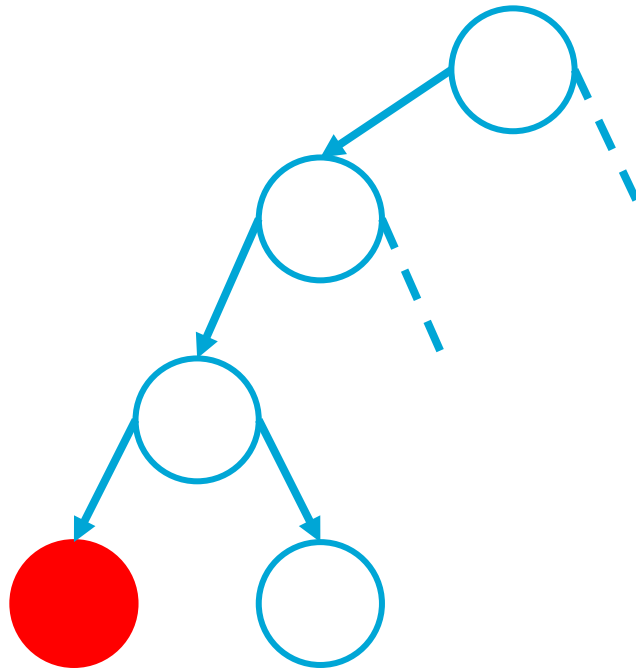


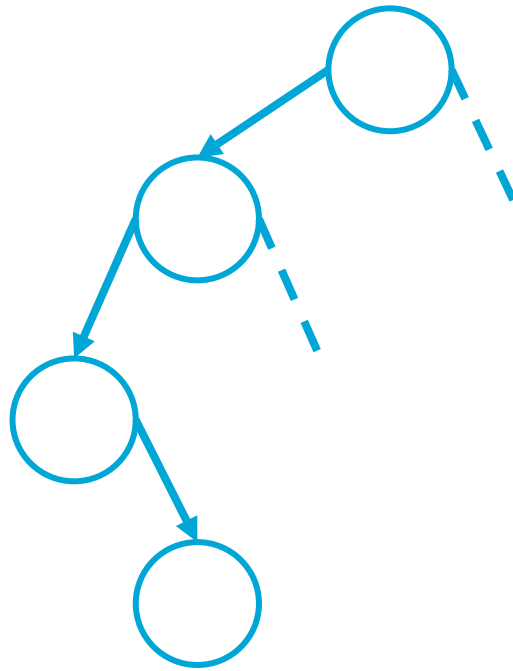
Remove SAT
View

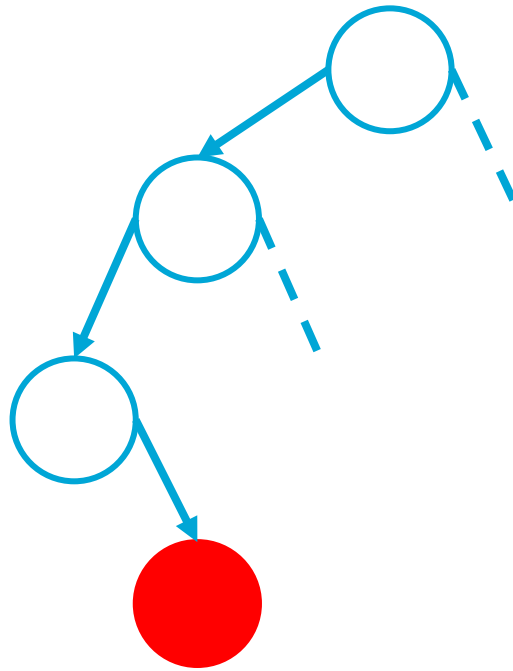


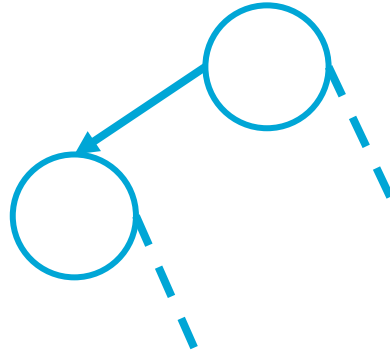
Generalise
Nogood
Propagation/
Learning

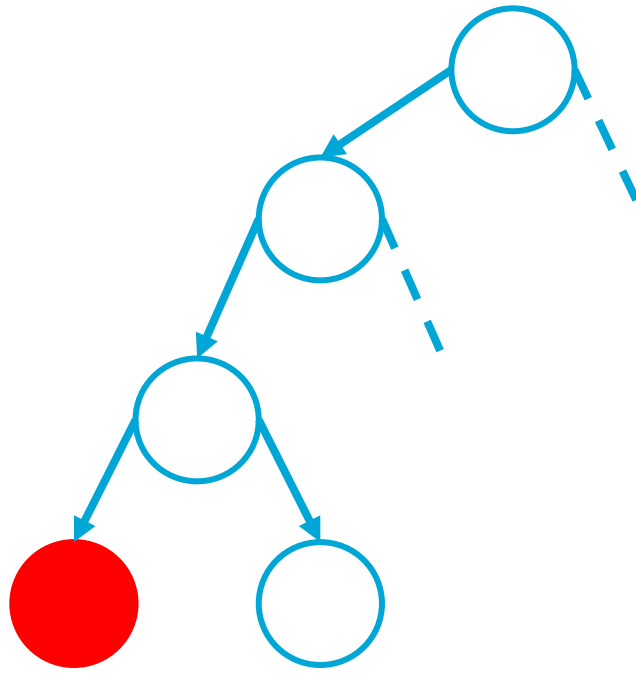


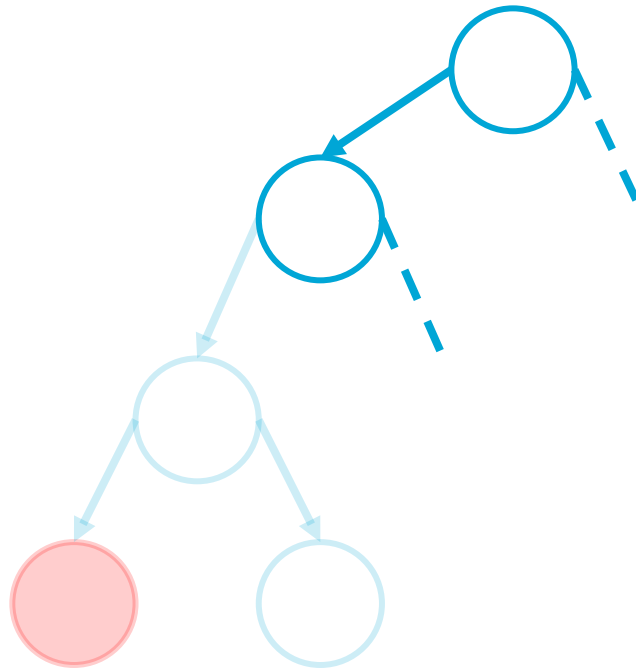












Learn **Nogoods**
using **Conflict**
Analysis!

SAT View

$$a \wedge b \wedge \neg c \rightarrow \perp$$

SAT View

$$a \wedge b \wedge \neg c \rightarrow \perp$$

SAT View

$$a \wedge b \wedge \neg c \rightarrow \perp$$

SAT View

$$a \wedge b \wedge \neg c \rightarrow \perp$$

Finally, we
can propagate!

SAT View

$$a \wedge b \wedge \neg c \rightarrow \perp$$

Unit
propagation

CP View

$$a \wedge b \wedge \neg c \rightarrow \perp$$

CP View

$$[x \geq 5] \wedge [x \leq 10] \wedge \neg[x = 7] \rightarrow \perp$$

Still use unit
propagation...

$$[x \geq 5] \wedge [x \leq 10] \wedge \neg[x = 7] \rightarrow \perp$$

Can we do
more?

$$[x \geq 5] \wedge [x \leq 10] \wedge \neg[x = 7] \rightarrow \perp$$

What if
[$x = 5$]?

$$[x \geq 5] \wedge [x \leq 10] \wedge \neg[x = 7] \rightarrow \perp$$

What if
[$x = 5$]?

$$[x \geq 5] \wedge [x \leq 10] \wedge \neg[x = 7] \rightarrow \perp$$

What if
[$x = 5$]?

$$[x \geq 5] \wedge [x \leq 10] \wedge \neg[x = 7] \rightarrow \perp$$

What if
[$x = 5$]?

$$[x \geq 5] \wedge [x \leq 10] \wedge \neg[x = 7] \rightarrow \perp$$

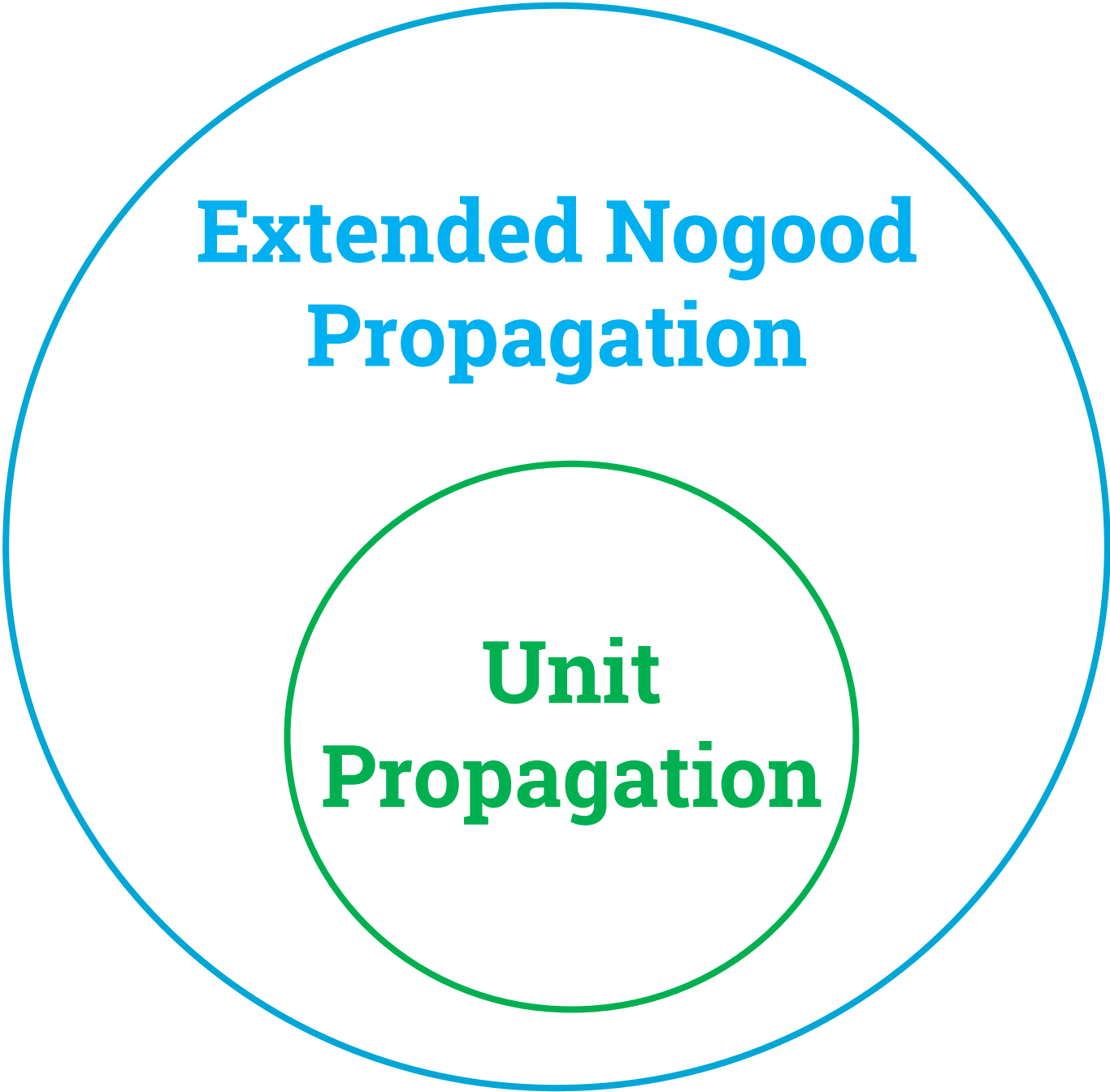
What if
[$x = 5$]?

$$[x \geq 5] \wedge [x \leq 10] \wedge \neg[x = 7] \rightarrow \perp$$

Propagate

$$[x \neq 5] \quad [x \neq 6] \quad [x \neq 8] \quad [x \neq 9] \quad [x \neq 10]$$

**Extended Nogood
Propagation**



The diagram consists of two concentric circles. The outer circle is blue and contains the text "Extended Nogood Propagation". The inner circle is green and contains the text "Unit Propagation". The green circle is centered within the blue circle, indicating that Unit Propagation is a subset of Extended Nogood Propagation.

**Extended Nogood
Propagation**

**Unit
Propagation**

What about Conflict Analysis?

SAT view

Resolve until
unit propagation



1UIP

CP view

Resolve until
extended nogood
propagation



CPIP

CP view

Lower backtrack
level (and LBD)



Fewer
explanations
necessary



Our Contributions

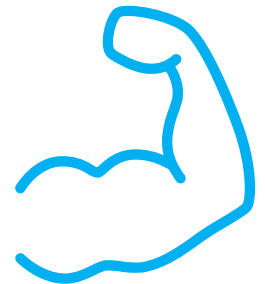
Analysis SAT
View in
Solvers



Remove SAT
View



Generalise
Nogood
Propagation/
Learning



Experiments

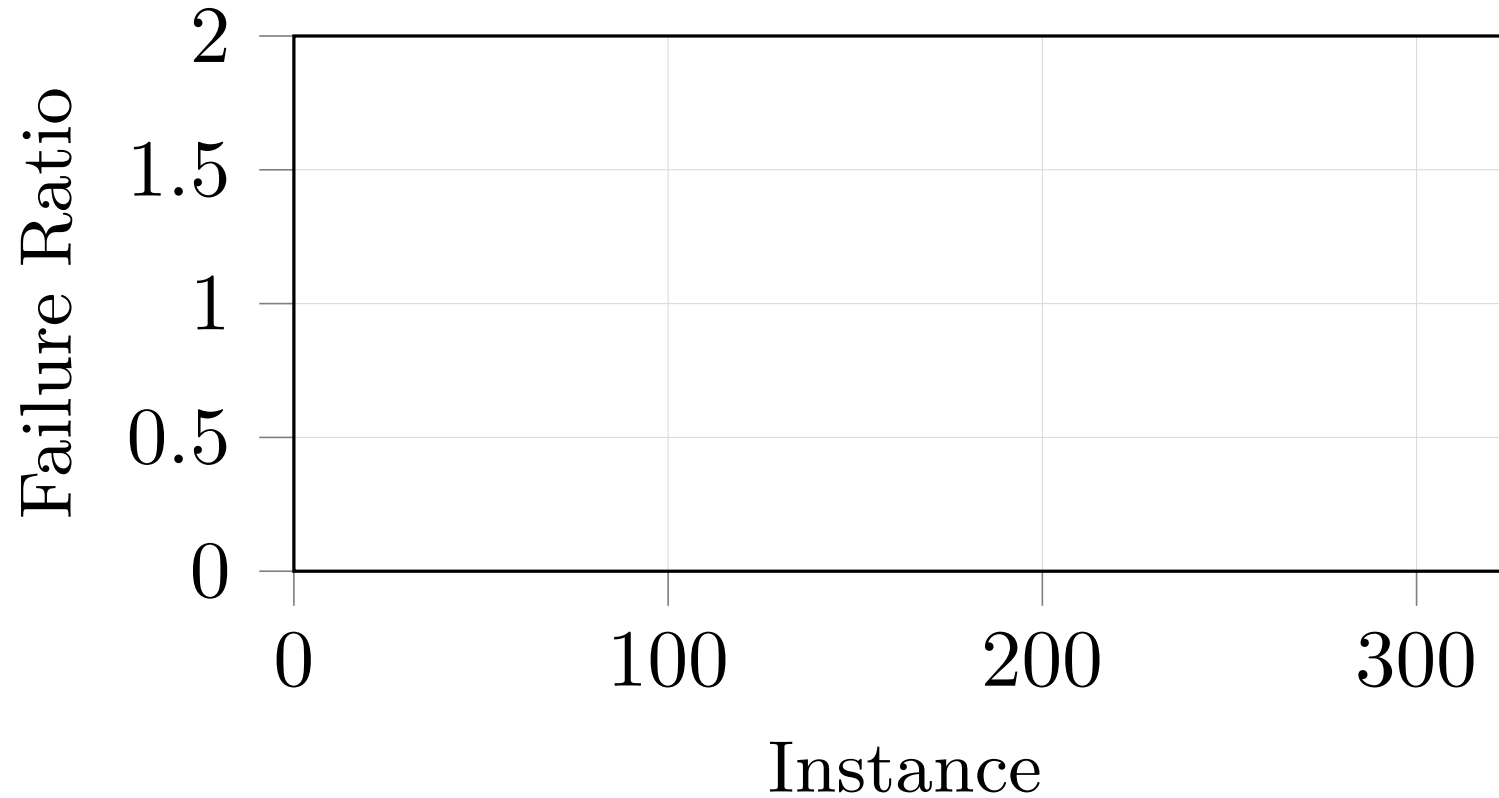
```

      .'.
      ,pkkku.
      mkkiiu, .
      ,kkn
      'kkn.
      .:kkkip, .
      .',',',mpkkkkkkkpp:pupuu:,
      .',',',mp:':':':ukpmpppppmppuoppuuummup; .
      .',',',pp:':':pp;:':':p;:':':pupp:':':
      .':uuuumpuuuup,,:ppuuuupp:p:uuuupuuuu:p'
      .:puuuuumpuuuuu, :uuuuuuuuuuuuuuuuuupuuuuuu;
      .:uuuuuuuuuuuuu: :uuuuuuuuuuuuuuuuuupmuuuuuu '
      puuuuuuupuuuuuu; uuuuuuuuuuuuuuupuuuuuukuuuuuup
      .uuuuuumpuuuuuuppuuuuuuuuuuuuuuuuuuupmuuuuuup '
      'puuuuupmuuuuuu: uuuuuuuuuuuuuuuuuuupmuuuuuup,
      .kuuuuuukuuuuuu: uuuuuuuuuuuuuuuuupuuuuuukuuuuuu '
      kmuuuuukuuuuuuuuuuuuuuuuuuuuuuupuuuuuukuuuuumk
      ;kuuuuukuuuuuupuuuuuuuuuuuuuuuupuuuuuukmuuuuk:
      kmuuuukuuuuuupmuuuuuuuuuuuuuuupuuuuuukmuuumk
      ,kmuuuukmuuuuupuuuuuuuuuuuuuuuupmuuuuukmuuumk;
      pkmuupkuuuuuukuuuuuuuuuuuuuuukuuuuuukpuumku
      :kpuukpuuuuukmuuuuuuuuuuuuuuukmuuuupkmpkp
      .pkppkpumuukmuuuuuuuuuuuuukmuuuupkpkp'
      .pkkkkmuuupkpumuuuuuuupkpumuukkkkp'
      .';ukkkkkkkkkppppppkkkkkkkkku; '.
      .'.

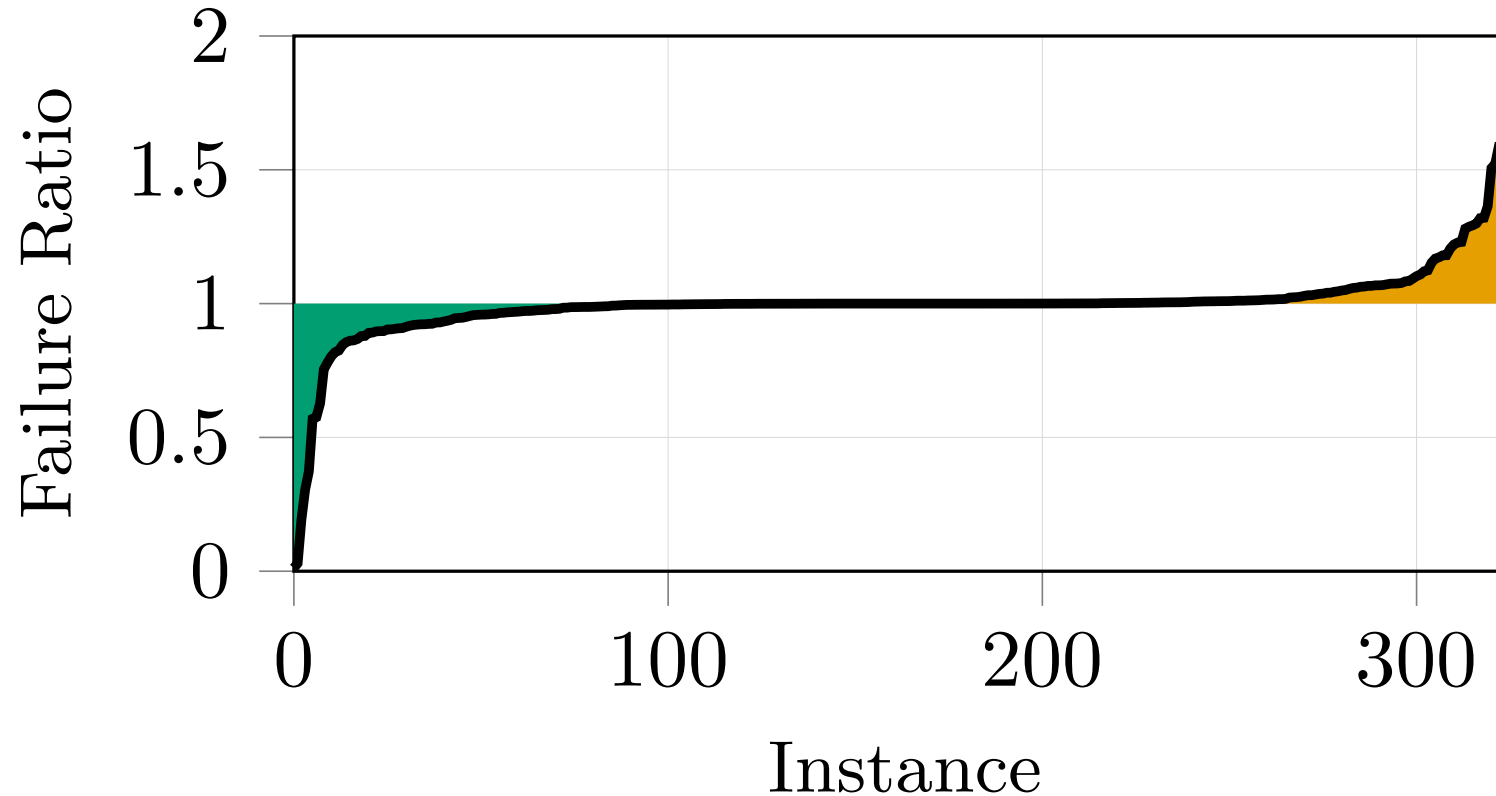
```



1UIP + Extended Propagation

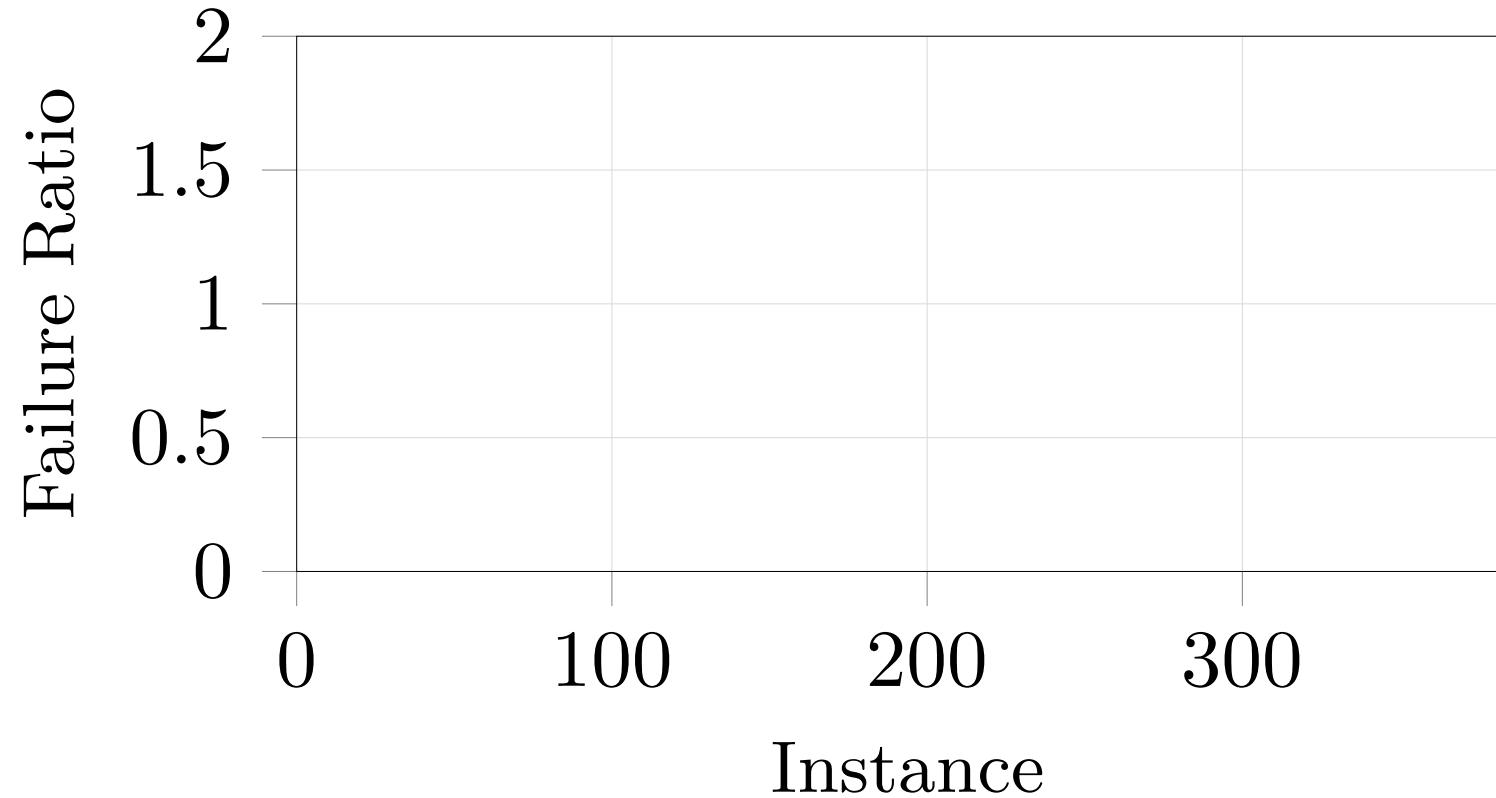


1UIP + Extended Propagation

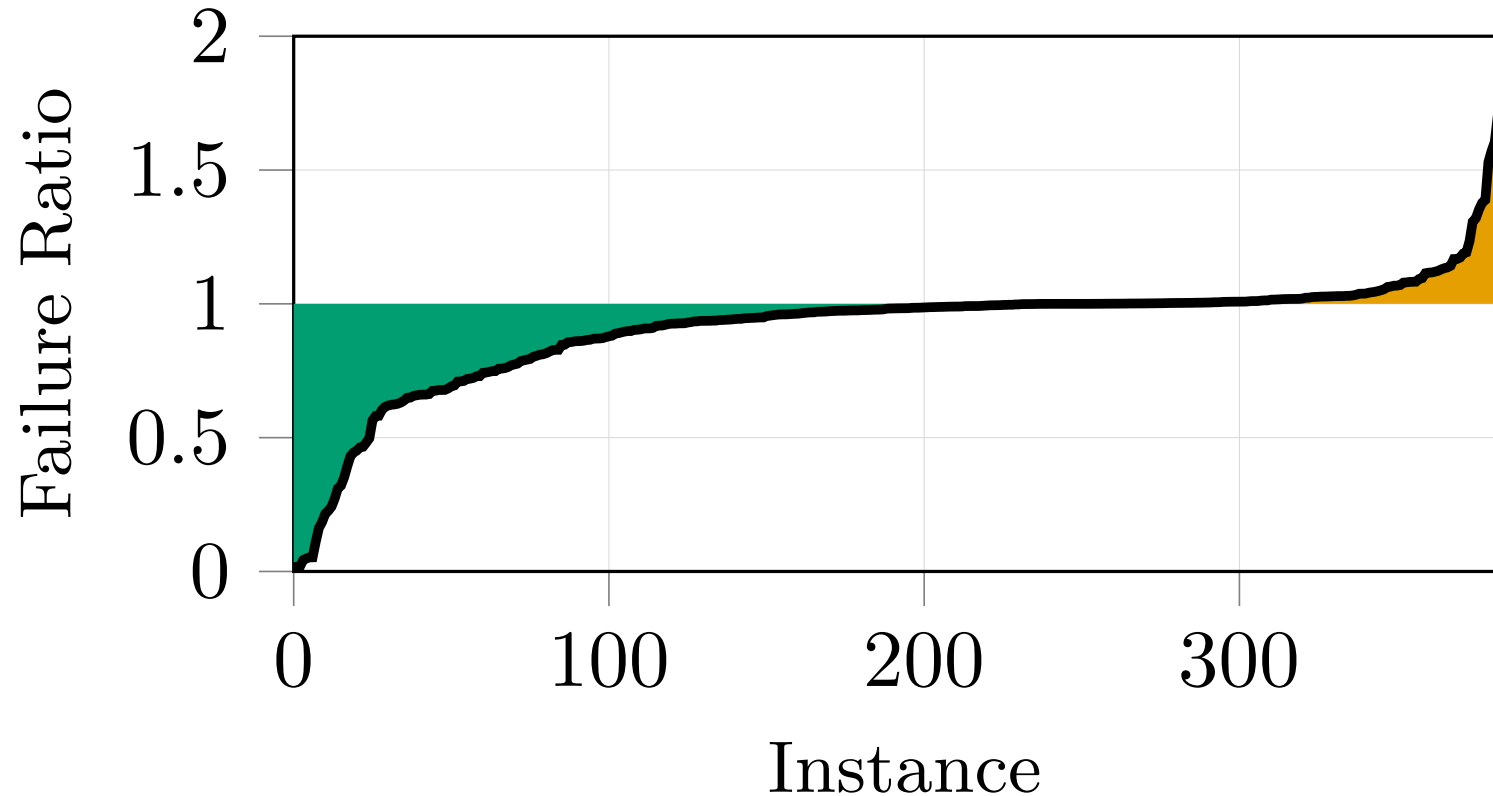


**Only propagation
does not work**

CPIP + Extended Propagation



CPIP + Extended Propagation

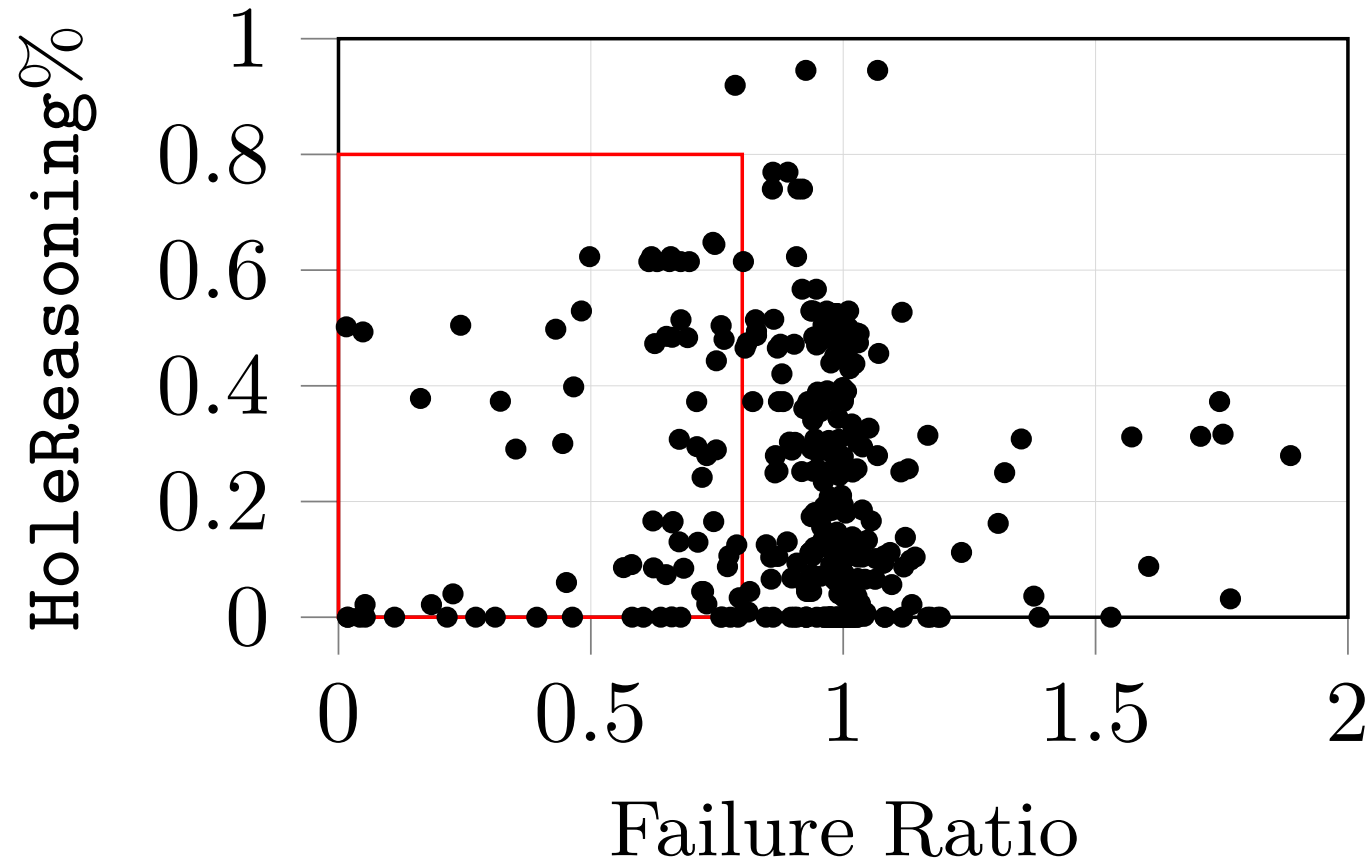


**Learning + Propagation
does work!**

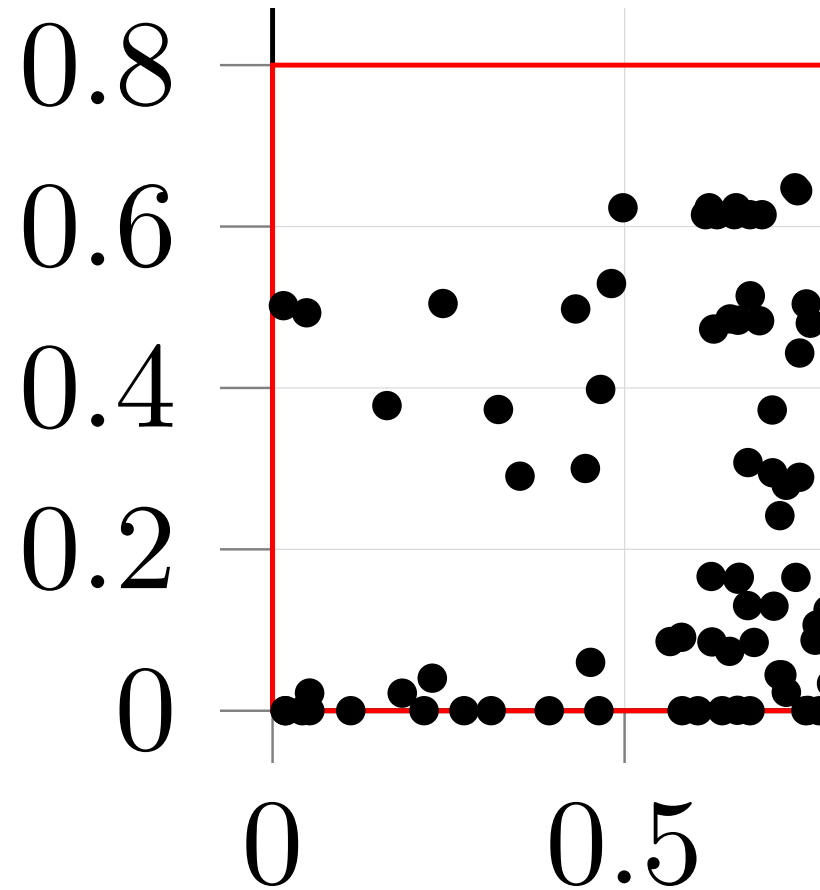
Learning + Propagation
does work!

But
why?

Learning + Propagation does work!



Learning + Propagation does work!



**Reasoning over holes
appears important!**

Conclusions

Analysis SAT
View in
Solvers



Generalise
CDCL for CP

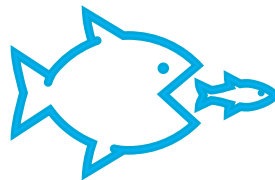


Reduces #failures
on many
instances



Future Work

Generalise Other
Techniques



Better Understanding
of Interaction



Generalising Conflict-Driven Clause Learning for Constraint Programming

Imko Marijnissen, Maarten Flippo, and Emir Demirović

Analysis SAT
View in
Solvers



Generalise
CDCL for CP



Reduces #failures
on many
instances



Paper

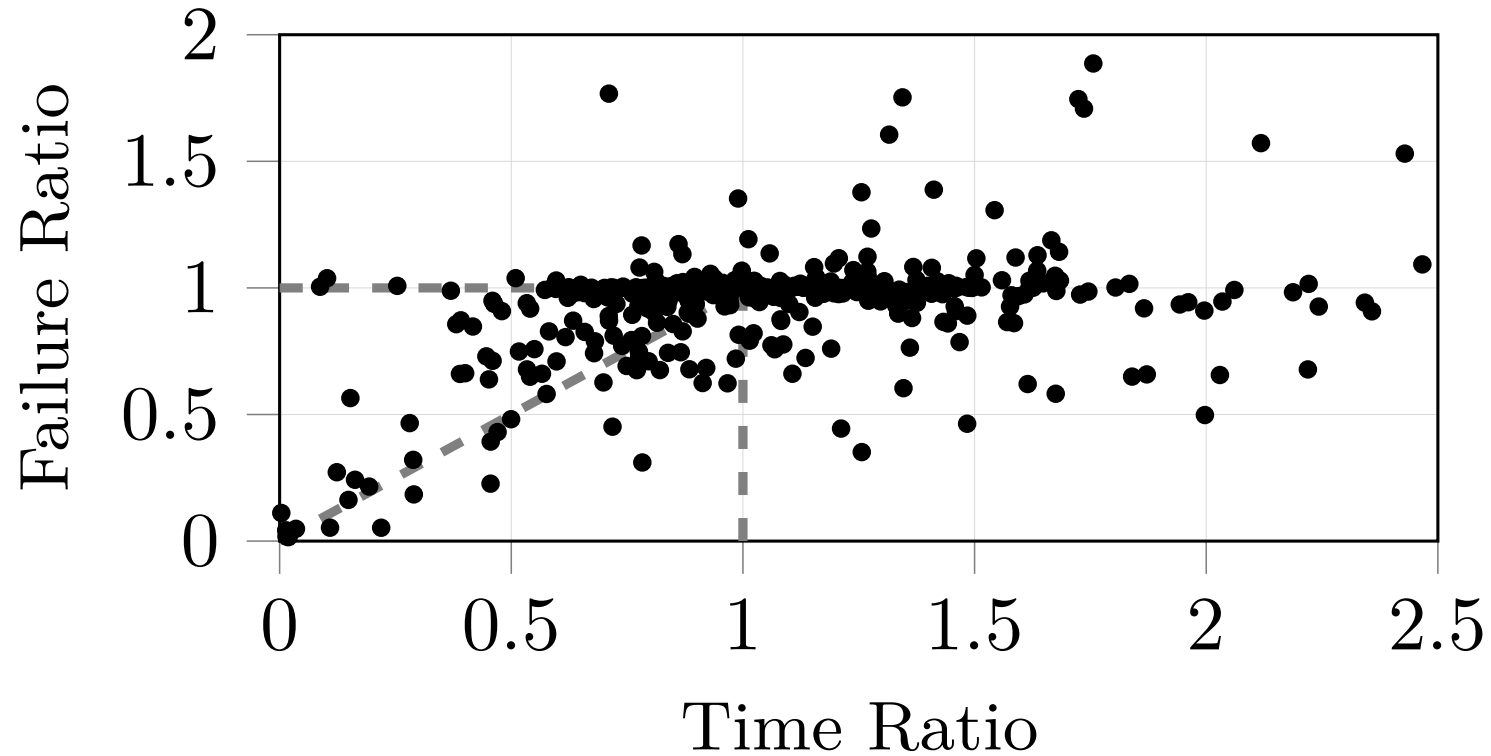


Pumpkin



Appendices

What about runtime?



**Many instances
on which runtime
also improves!**

Solver \ Feature	Explanation Lifting	Fully Lazy Explanations	No Linking Clauses	Allows Minimisation
Chuffed	✓	✗	✓	✓
Huub	$\sim^a$	$\sim^b$	✗	$\sim^a$
OR-Tools	$\sim^a$	$\sim^c$	✗	$\sim^a$
Tempo	✓	✓	✓	✓
Geas	✓	✗	✓	✗

- a) Limited Lifting/Minimisation if literals do not exist.
- b) By backtracking to this state and afterwards reapplying the trail
- c) By traversing the entire trail

■ **Table 1** An overview of features supported by state-of-the-art solvers. Each solver makes different trade-offs, apart from Tempo, which is restricted to bound atomic constraints.

■ Algorithm 2 SAT 1UIP

Input: C - Conflict Literals
 T - Trail (Assigned Literals)
 DL - Current decision level

Output: Learned Clause

```
1:  $i \leftarrow |T|$ 
2:  $LC \leftarrow C$ 
3: while  $\left| \begin{array}{l} l \in LC \wedge \\ \text{DecLvl}(l) = DL \end{array} \right| > 1$ 
   do
4:   if  $T[i] \notin LC$  then
5:     continue
6:    $l \leftarrow T[i]$ 
7:    $LC \leftarrow LC \odot \text{Reason}(l)$ 
8:    $i \leftarrow i - 1$ 
9: return  $LC$ 
```

■ Algorithm 3 CP 1UIP

Input: C - Conflict Atomic Constraints
 DL - Current decision level

Output: Learned Nogood

```
1:  $\text{CurrDL} \leftarrow \text{Heap} \left( \begin{array}{l} p \mid p \in C \wedge \\ \text{DecLvl}(p) = DL \end{array} \right)$ 
2:  $\text{PrevDL} \leftarrow \begin{array}{l} p \mid p \in C \wedge \\ 0 < \text{DecLvl}(p) < DL \end{array}$ 
3: while  $|\text{CurrDL}| > 1$  do
4:    $\text{next} \leftarrow \text{pop max element CurrDL}$ 
5:   for  $p \in \text{Reason}(\text{next})$  do
6:     if  $\text{DecLvl}(p) = DL$  then
7:        $\text{CurrDL} \leftarrow \text{CurrDL} \cup \{p\}$ 
8:     else
9:        $\text{PrevDL} \leftarrow \text{PrevDL} \cup \{p\}$ 
10: return  $\text{PrevDL} \cup \text{CurrDL}$ 
```

■ **Algorithm 1** Extended Nogood Propagation

Input: N - Semantically Minimised Nogood

x - Variable to propagate

Assumptions: No atomic constraints in N which are falsified or unassigned and not over x .

```
1: if UnitPropagate( $N$ ) then return
2:  $N' \leftarrow \{ \llbracket x \otimes v \rrbracket \mid \llbracket x \otimes v \rrbracket \in N : \llbracket x \otimes v \rrbracket \text{ is unassigned} \}$ 
3: if  $\exists v \mid \llbracket x \geq v \rrbracket \in N'$  then  $lb \leftarrow v$  else  $lb \leftarrow \min_{v \text{ s.t. } \llbracket x \neq v \rrbracket \in N'} v$ 
4: if  $\exists v \mid \llbracket x \leq v \rrbracket \in N'$  then  $ub \leftarrow v$  else  $ub \leftarrow \max_{v \text{ s.t. } \llbracket x \neq v \rrbracket \in N'} v$ 
5: if  $\nexists v \text{ s.t. } \llbracket x \leq v \rrbracket \in N' \wedge \exists v \text{ s.t. } \llbracket x \neq v \rrbracket \in N'$  then ▷ Upper-bound Propagation
6:   Post( $\llbracket x \leq ub \rrbracket$ )
7: if  $\nexists v \text{ s.t. } \llbracket x \geq v \rrbracket \in N' \wedge \exists v \text{ s.t. } \llbracket x \neq v \rrbracket \in N'$  then ▷ Lower-bound Propagation
8:   Post( $\llbracket x \geq lb \rrbracket$ )
9: for  $v \in \{p \mid p \in [lb, ub] \text{ s.t. } \llbracket x \neq p \rrbracket \notin N'\}$  do ▷ Disequality Propagation
10:   Post( $\llbracket x \neq v \rrbracket$ )
```
