Weight-Aware Core Extraction in SAT-Based MaxSAT Solving

Jeremias Berg Matti Järvisalo

HIIT, Dept. Computer Science,
University of Helsinki, Finland

CP 2017
Melbourne
September 1, 2017

Motivation

- Constraint optimization, a generic, effective approach to several NP-hard problems in AI and industrial domains.
  - planning, debugging, and diagnosis, machine learning, systems biology etc.
- Maximum Satisfiability, a competitive paradigm alternative
- *Core-Guided MaxSAT*, most promising for large-scale industrial problems.
  - Extensive use of SAT-solvers.
  - Iteratively extract and rule out unsatisfiable sub-formulas
  - Increase complexity of working formula on each iteration

Argelich et al.

Narodytska and Bacchus [2014]

Bjørner and Narodytska [2015]

Bonet et al. [2007]

Ansótegui et al. [2013]

Zhang and Bacchus [2012]

Argelich et al. [2010]

Chen et al. [2010]

Zhu et al. [2011]

Guerra and Lynce [2012]

Bunte et al. [2014]

Marques-Silva et al. [2015]

Ansótegui et al. [2015]

Berg et al. [2014]

Motivation

- Constraint optimization, a generic, effective approach to several NP-hard problems in AI and industrial domains.
  - planning, debugging, and diagnosis, machine learning, systems biology etc.
- Maximum Satisfiability, a competitive paradigm alternative
- *Core-Guided MaxSAT*, most promising for large-scale industrial problems.
  - Extensive use of SAT-solvers.
  - Iteratively extract and rule out unsatisfiable sub-formulas
  - Increase complexity of working formula on each iteration

Argelich et al.

Narodytska and Bacchus [2014]

Bjørner and Narodytska [2015]

Bonet et al. [2007]

Ansótegui et al. [2013]

Zhang and Bacchus [2012]

Argelich et al. [2010]

Chen et al. [2010]

Zhu et al. [2011]

Guerra and Lynce [2012]

Bunte et al. [2014]

Marques-Silva et al. [2015]

Ansótegui et al. [2015]

Berg et al. [2014]

Our Contributions

- Weight Aware Core Extraction (WCE)
  - Delay ruling out found cores, keep working formula simpler
  - Generic refinement to core-extraction and weight handling used in many solvers
  - Applicable to many modern MaxSAT algorithms
- In this paper, WCE for the PMRES algorithm Narodytska and Bacchus [2014].

Our Contributions

- Weight Aware Core Extraction (WCE)
  - Delay ruling out found cores, keep working formula simpler
  - Generic refinement to core-extraction and weight handling used in many solvers
  - Applicable to many modern MaxSAT algorithms
- In this paper, WCE for the PMRES algorithm Narodytska and Bacchus [2014].

Our Contributions

- Weight Aware Core Extraction (WCE)
  - Delay ruling out found cores, keep working formula simpler
  - Generic refinement to core-extraction and weight handling used in many solvers
  - Applicable to many modern MaxSAT algorithms
- In this paper, WCE for the PMRES algorithm Narodytska and Bacchus [2014].

Outline

- Satisfiability (SAT) and Maximum Satisfiability (MaxSAT)
- Core Guided MaxSAT solving.
- Weight Aware Core Extraction (WCE)
- Implementing WCE
- Experimental Evaluation (with PMRES)
- Summary

Satisfiability

- Satisfiability (SAT): Given a set F of Boolean Constraints, decide if its satisfiable

$$F = \{(x \vee z), (x \vee y), (\neg x), (\neg z), (\neg y)\}$$

Weighted Partial Maximum Satisfiability

- (Weighted Partial) Maximum Satisfiability (MaxSAT):
  - Two sets of constraints: F_h, (hard clauses) and F_s (soft clauses, weighted)
  - Find assignment τ satisfying F_h and a maximum sum of weights of clauses in F_s.

$$F_h = \{(x \vee z), (x \vee y)\}$$

$$F_s = \{((\neg x), 3), ((\neg z), 1), ((\neg y), 1)\}$$

COST: 2

Weighted Partial Maximum Satisfiability

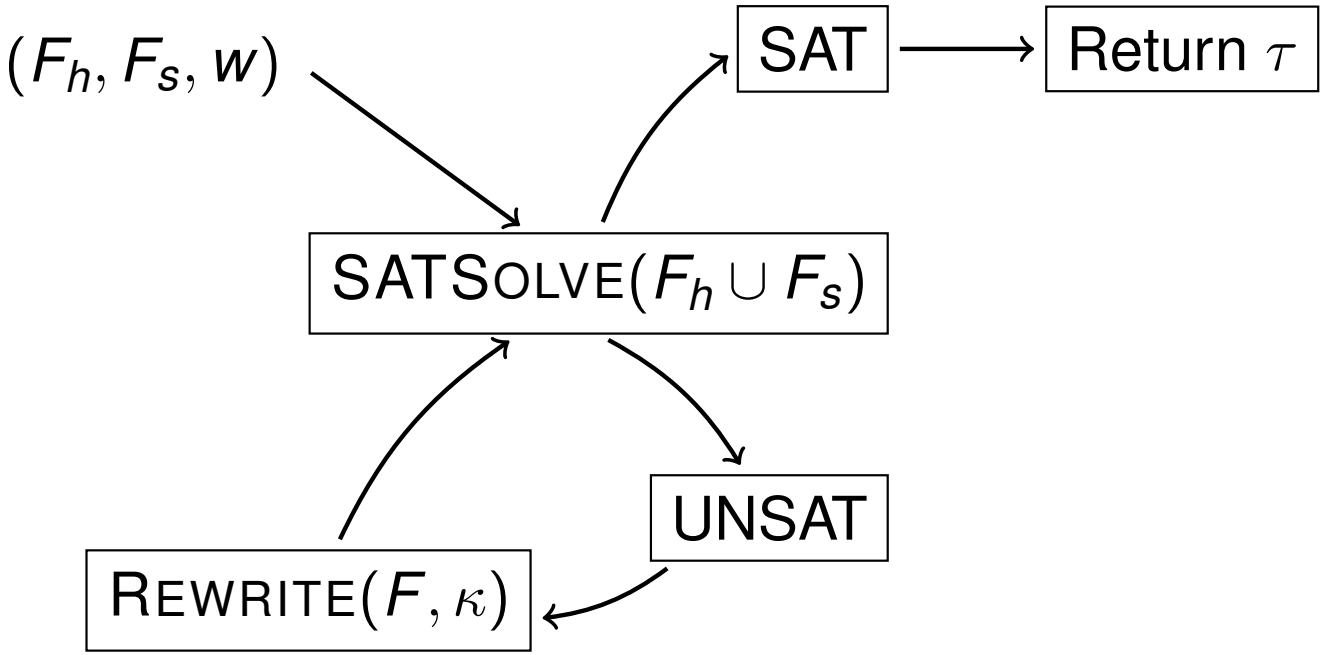
- A unsatisfiable core κ: a subset $\kappa \subseteq F_s$ s.t. $\kappa \wedge F_h$ is unsatisfiable.

$$F_h = \{(\mathbf{x} \vee \mathbf{z}), (x \vee y)$$

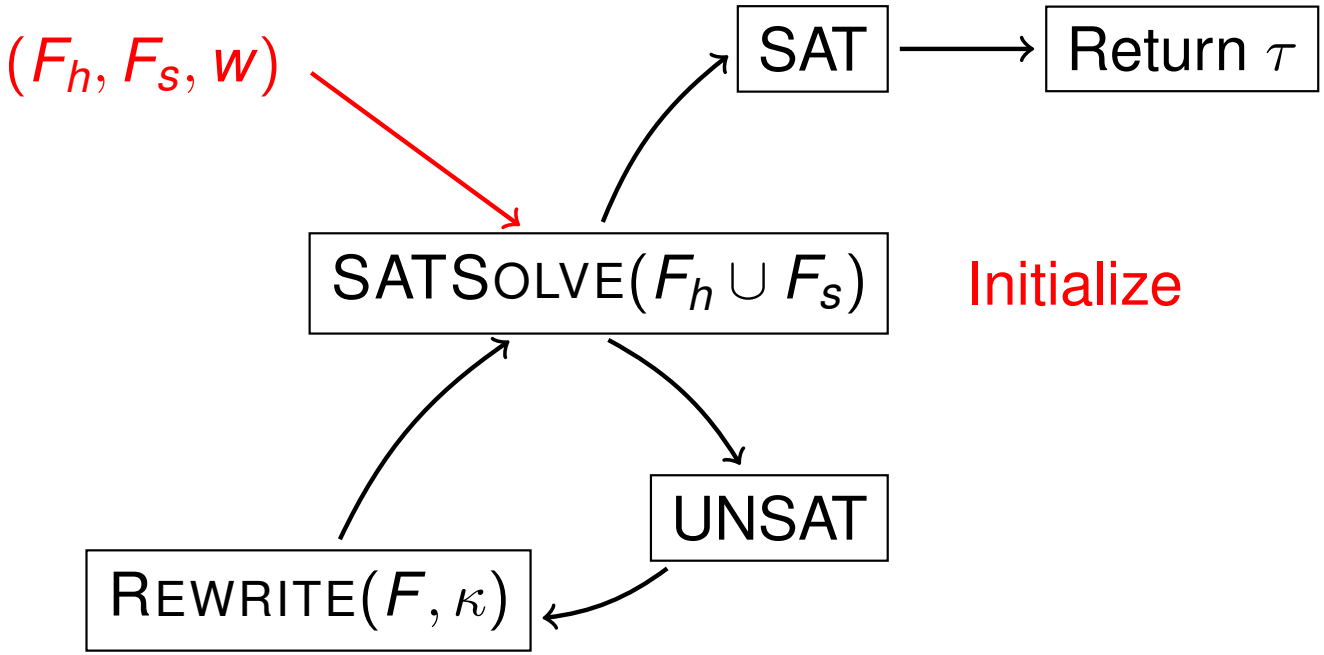
$$F_s = \{(\neg\mathbf{x}), (\neg\mathbf{z}), (\neg y)\}$$

Core Guided MaxSAT solving

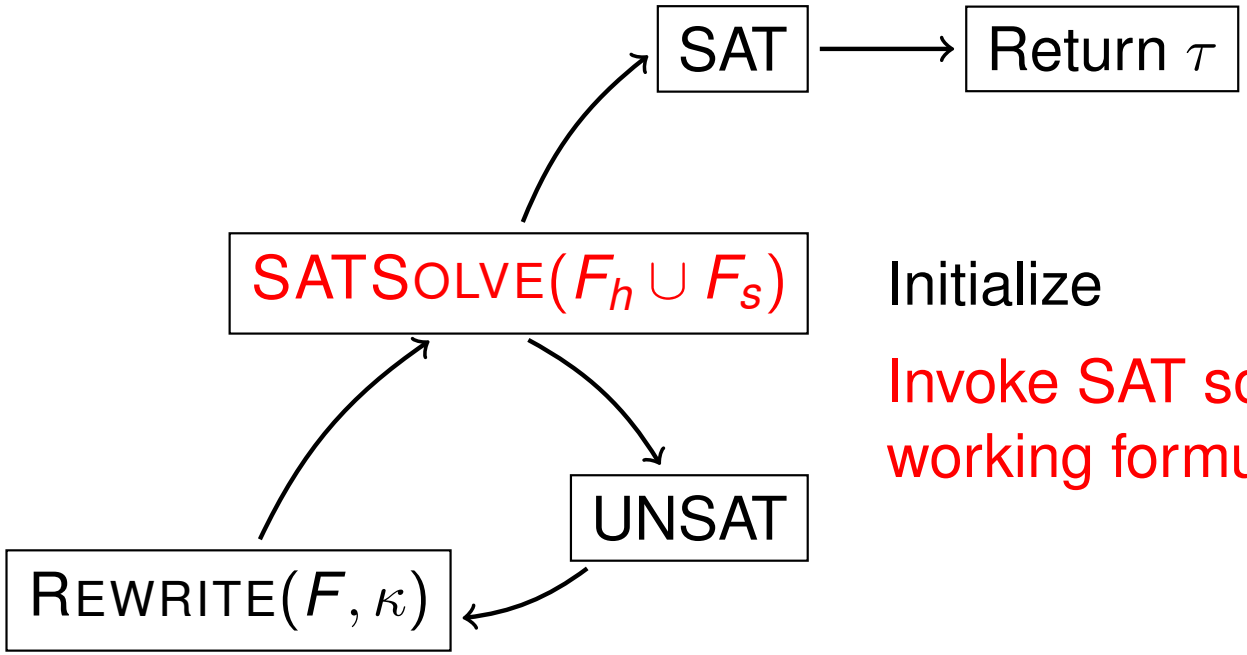
Core Guided MaxSAT solving



Core Guided MaxSAT solving



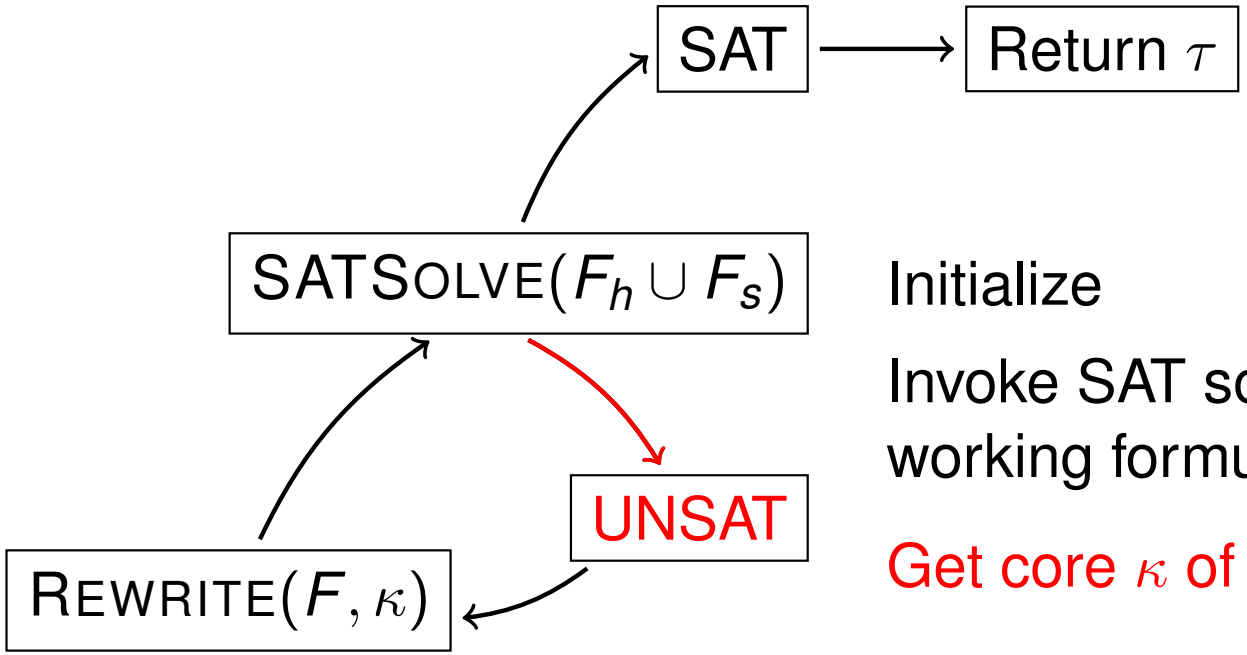
Core Guided MaxSAT solving



Initialize

Invoke SAT solver on working formula

Core Guided MaxSAT solving

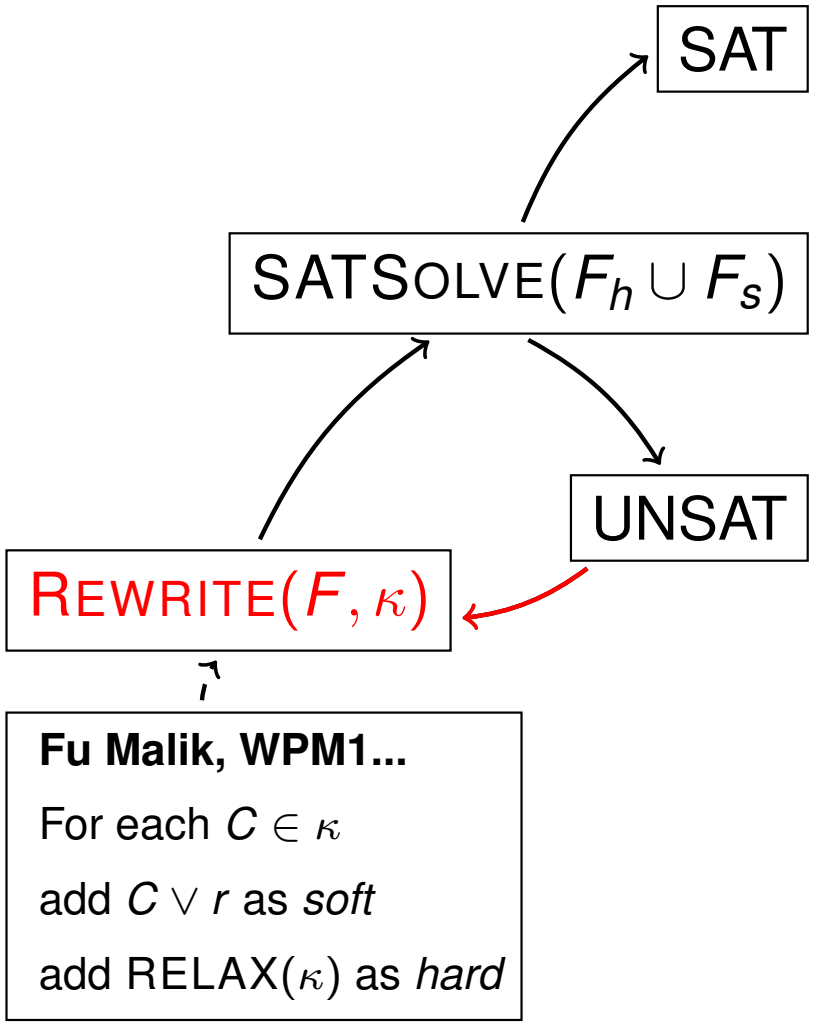


Initialize

Invoke SAT solver on working formula

Get core κ of F

Core Guided MaxSAT solving Ansótegui et al. [2013]



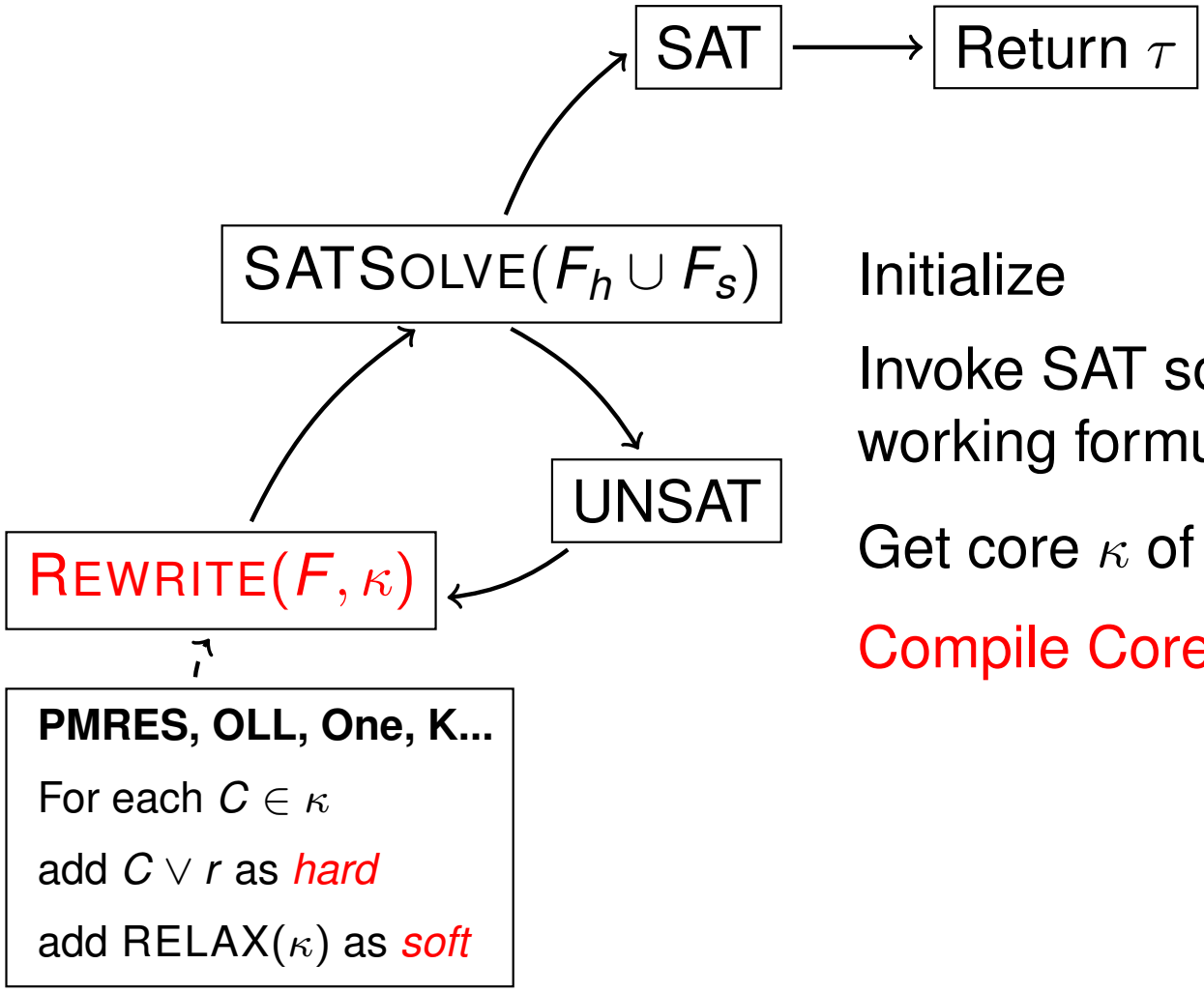
Initialize

Invoke SAT solver on working formula

Get core κ of F

Compile Core into the formula

Core Guided MaxSAT solving Narodytska and Bacchus [2014]



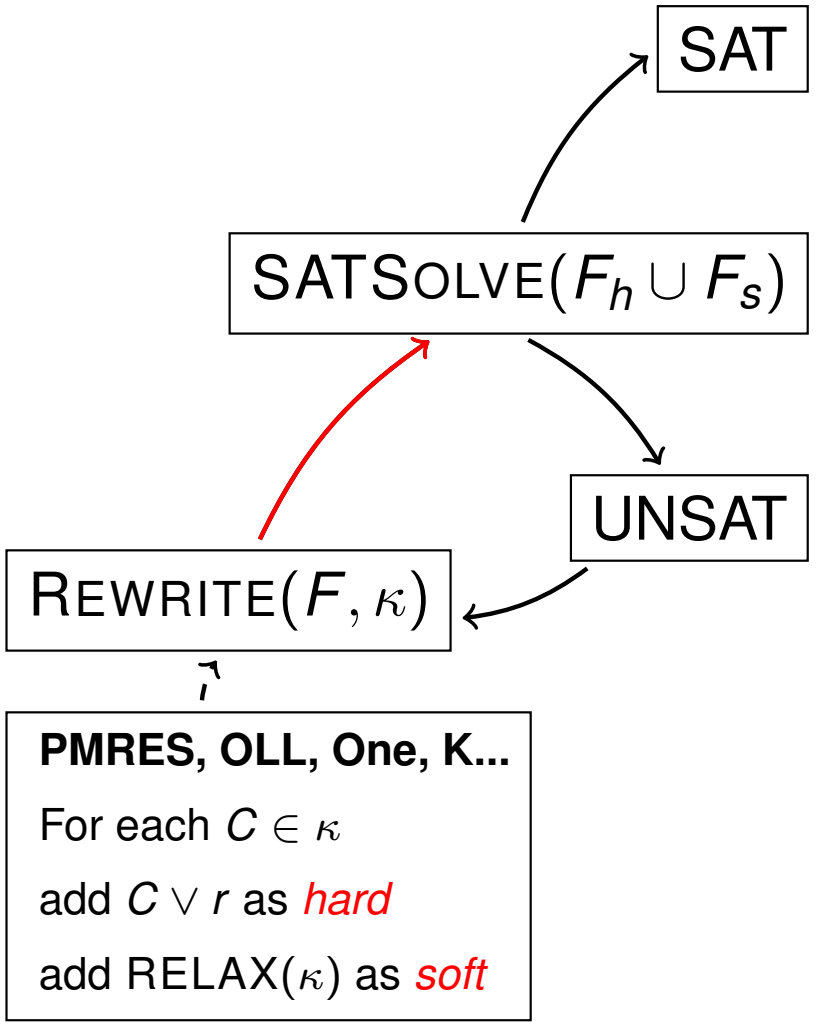
Initialize

Invoke SAT solver on working formula

Get core κ of F

Compile Core into the formula

Core Guided MaxSAT solving



Initialize

Invoke SAT solver on working formula

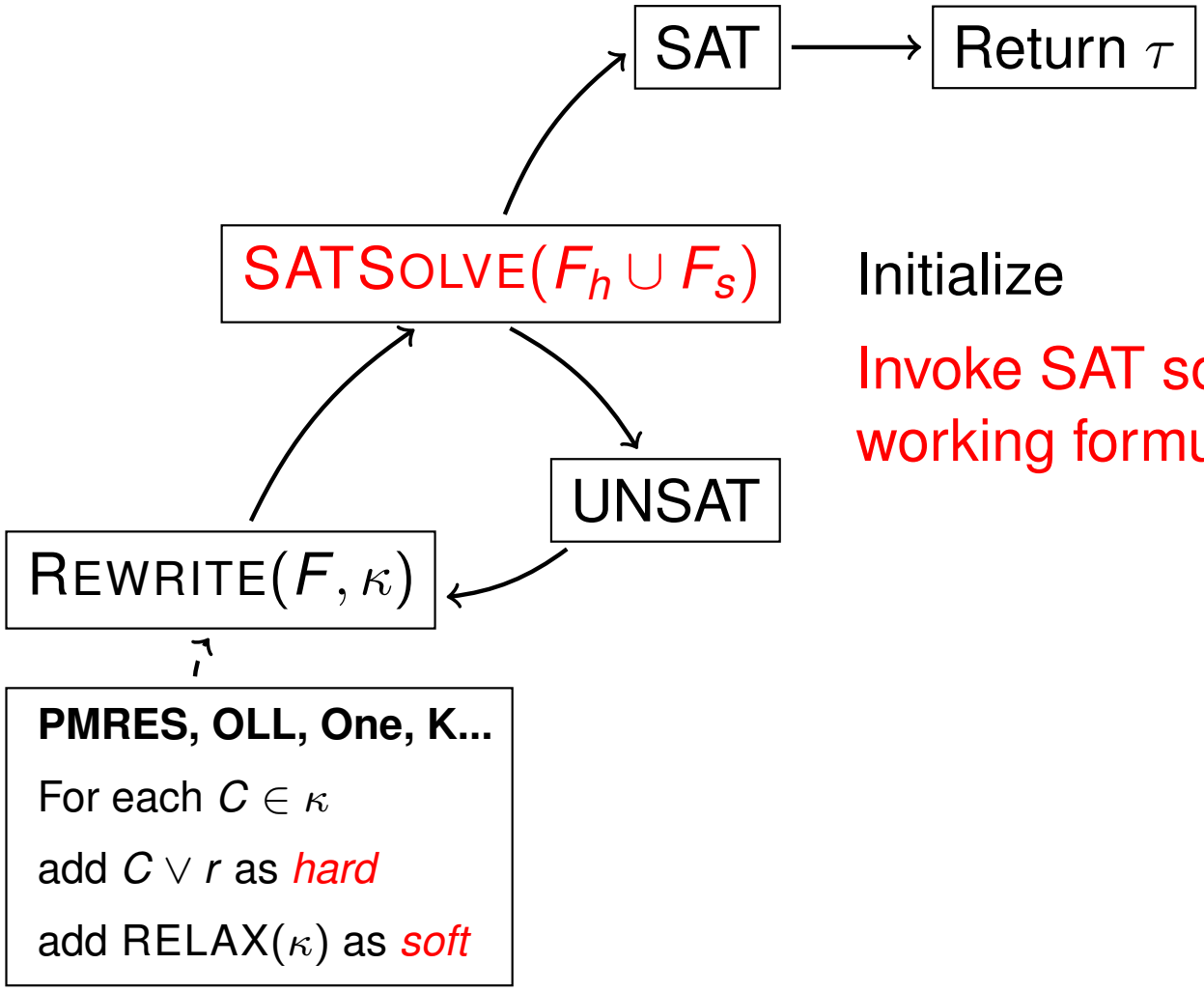
Get core κ of F

Compile Core into the formula

Reiterate

NOTE: Complexity increases

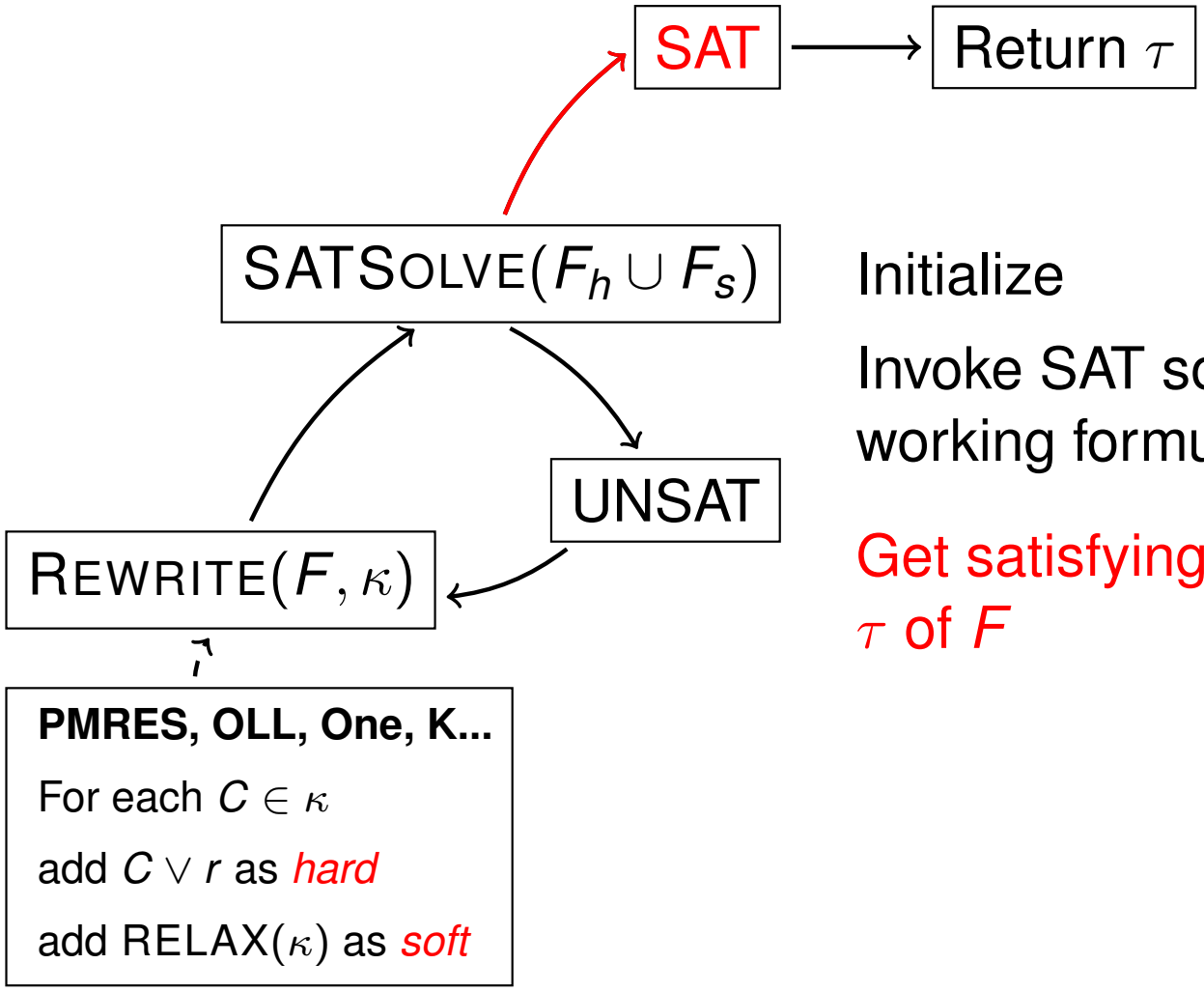
Core Guided MaxSAT solving



Initialize

Invoke SAT solver on working formula

Core Guided MaxSAT solving

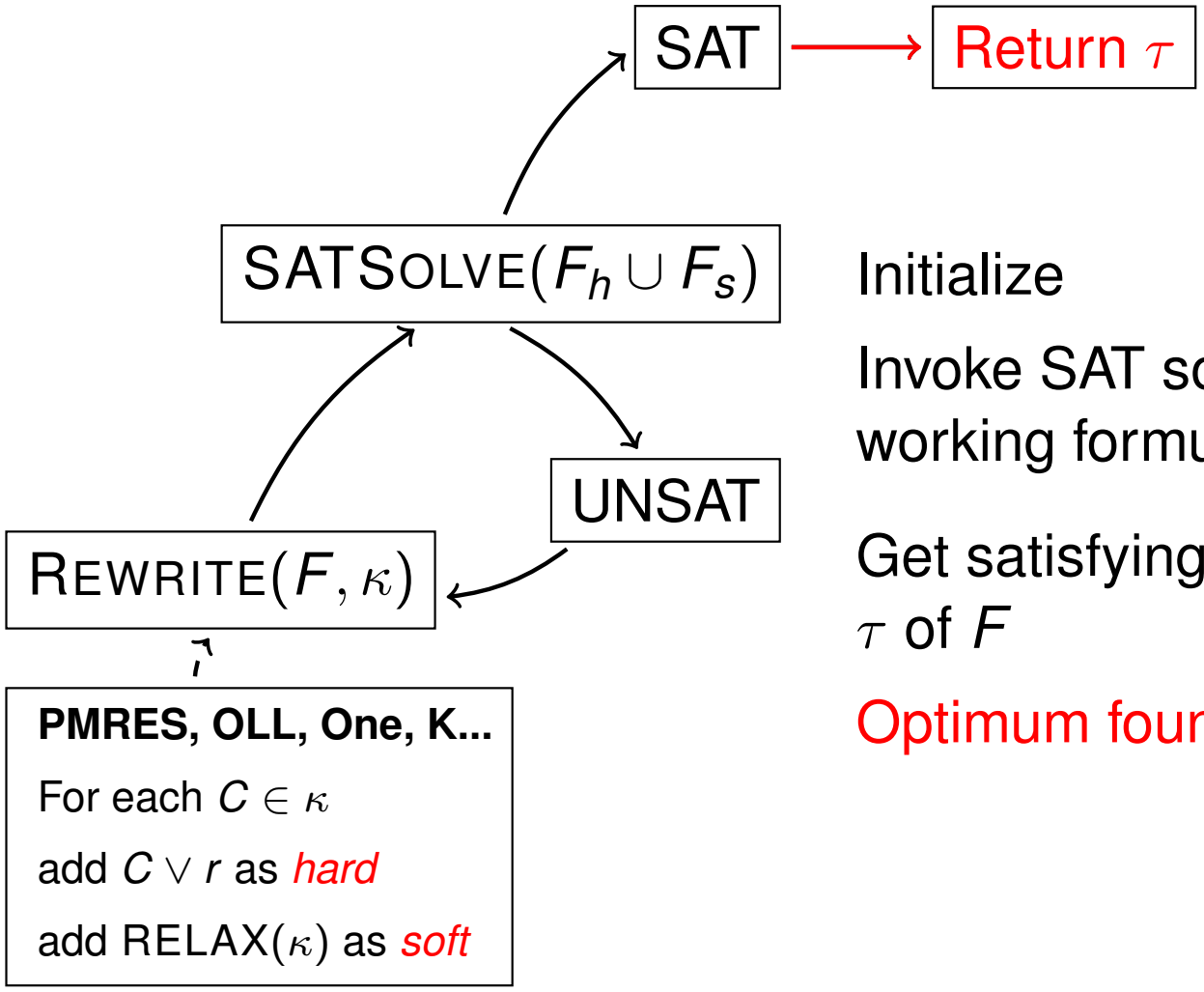


Initialize

Invoke SAT solver on working formula

Get satisfying assignment τ of F

Core Guided MaxSAT solving



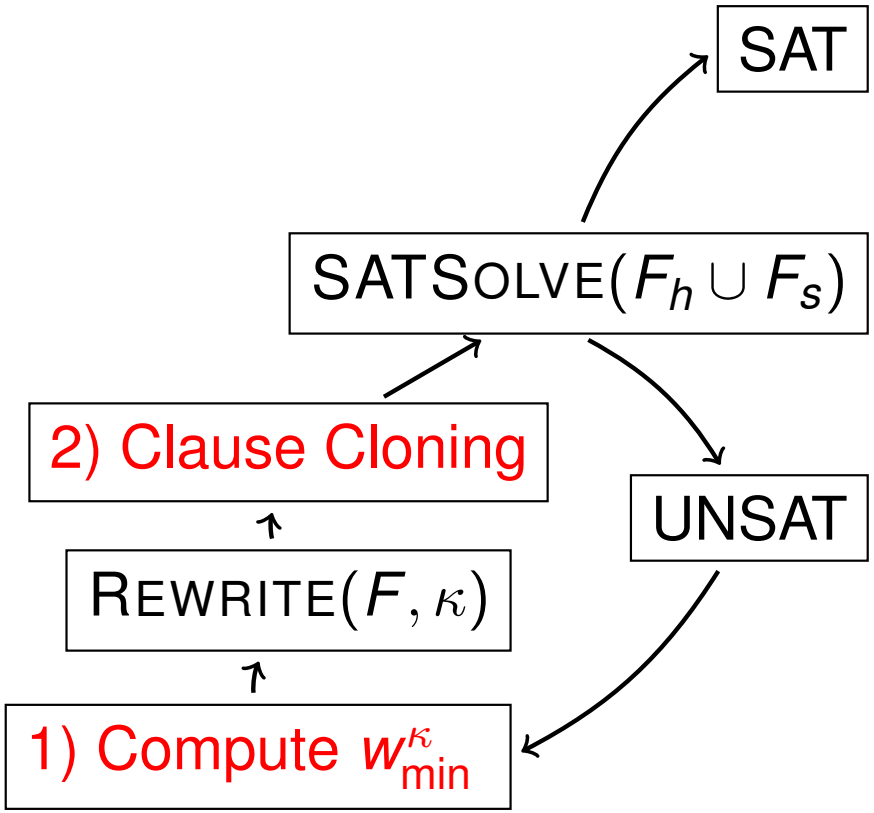
Initialize

Invoke SAT solver on working formula

Get satisfying assignment τ of F

Optimum found

Core Guided **weighted** MaxSAT solving



Clause Cloning:

$w^{\kappa}_{\min} = \min\{w(C) \mid C \in \kappa\}$ (1)

Weight of new soft clauses $w^{\kappa}_{\min}$

For each $C \in \kappa$ s.t $w(C) > w^{\kappa}_{\min}$

Add clone of C with weight $w(C) - w^{\kappa}_{\min}$ (2)

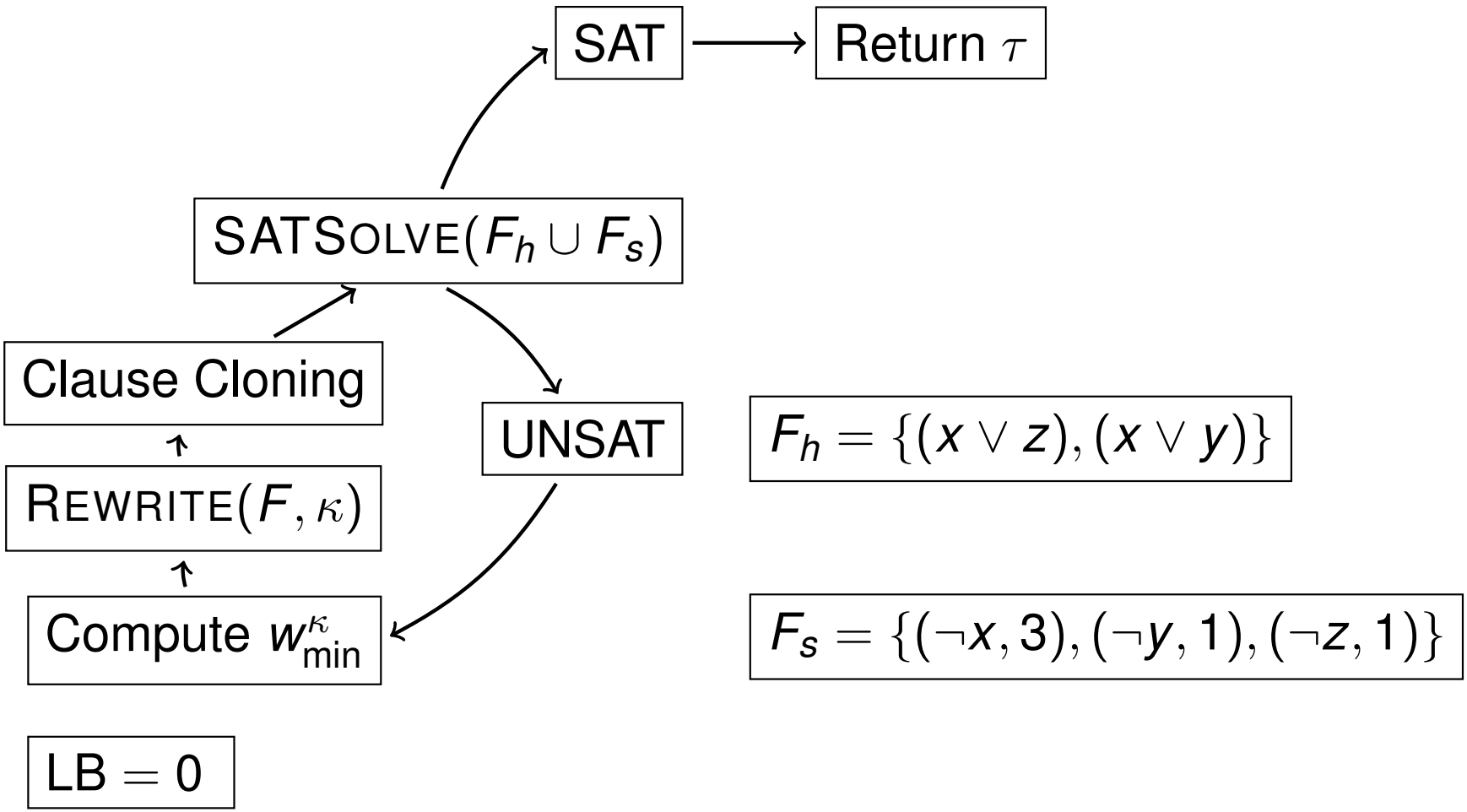
NOTE: Sum of $w^{\kappa}_{\min}$

is a Lower Bound on solution cost

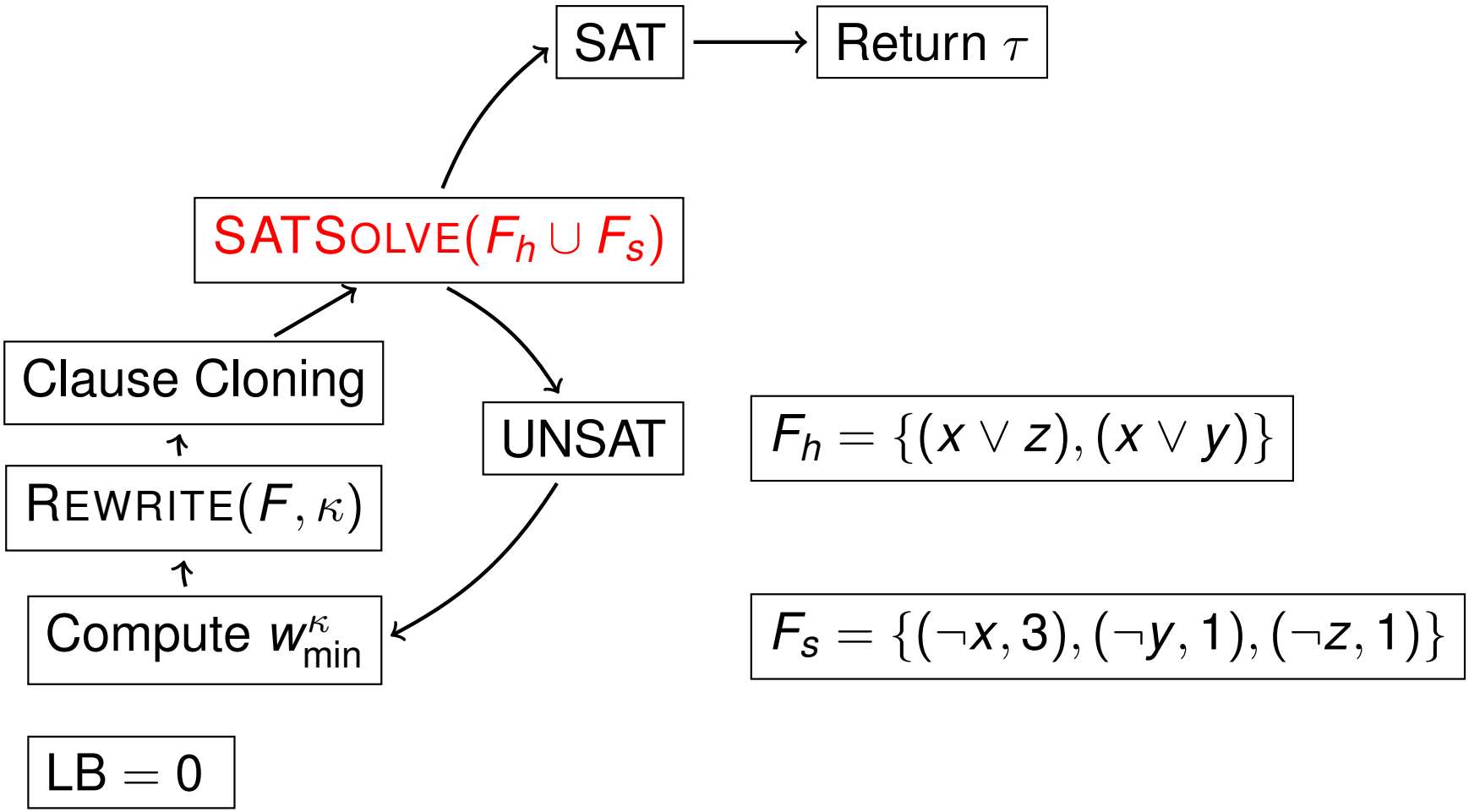
Example

$$F_h = \{(x \vee z), (x \vee y)\}$$
$$F_s = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$$

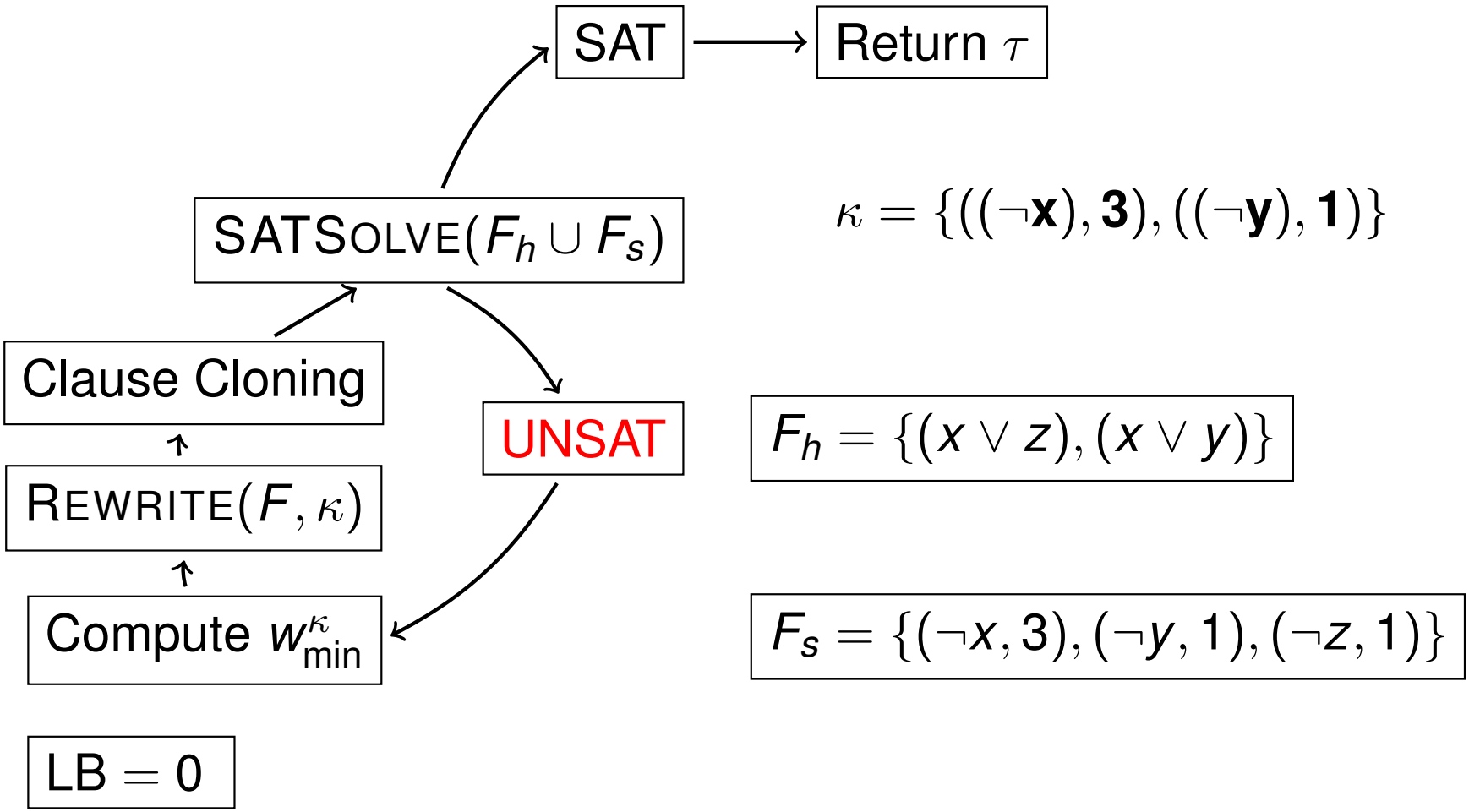
Example


SAT
Return τ
SATSOLVE($F_h \cup F_s$)
Clause Cloning
UNSAT
REWRITE(F, κ)
Compute w^{κ}_{min}
LB $= 0$
$F_h = \{(x \vee z), (x \vee y)\}$
$F_s = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$

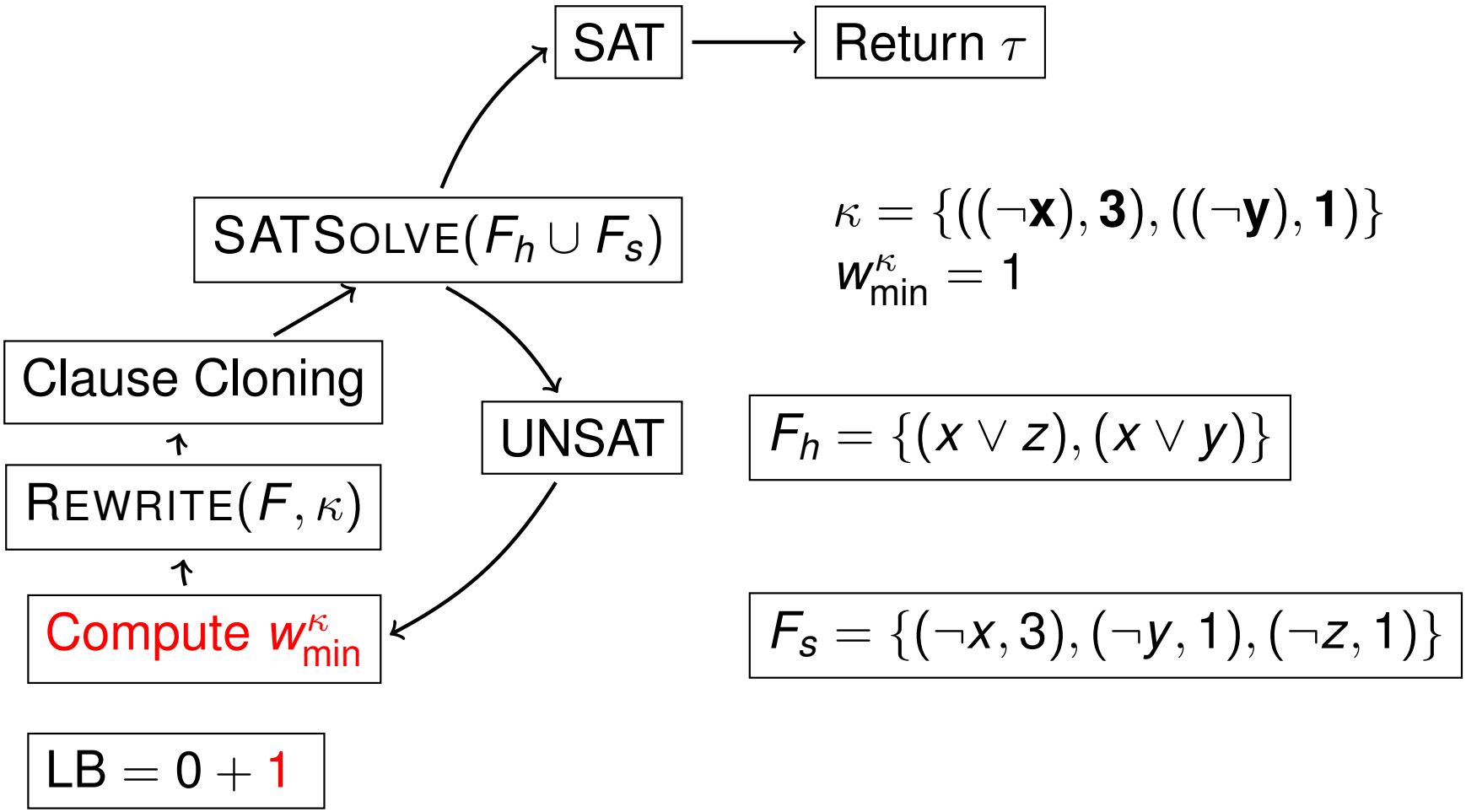
Example


SAT
Return τ
SATSOLVE($F_h \cup F_s$)
Clause Cloning
UNSAT
REWRITE(F, κ)
Compute w^{κ}_{min}
LB $= 0$
$F_h = \{(x \vee z), (x \vee y)\}$
$F_s = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$

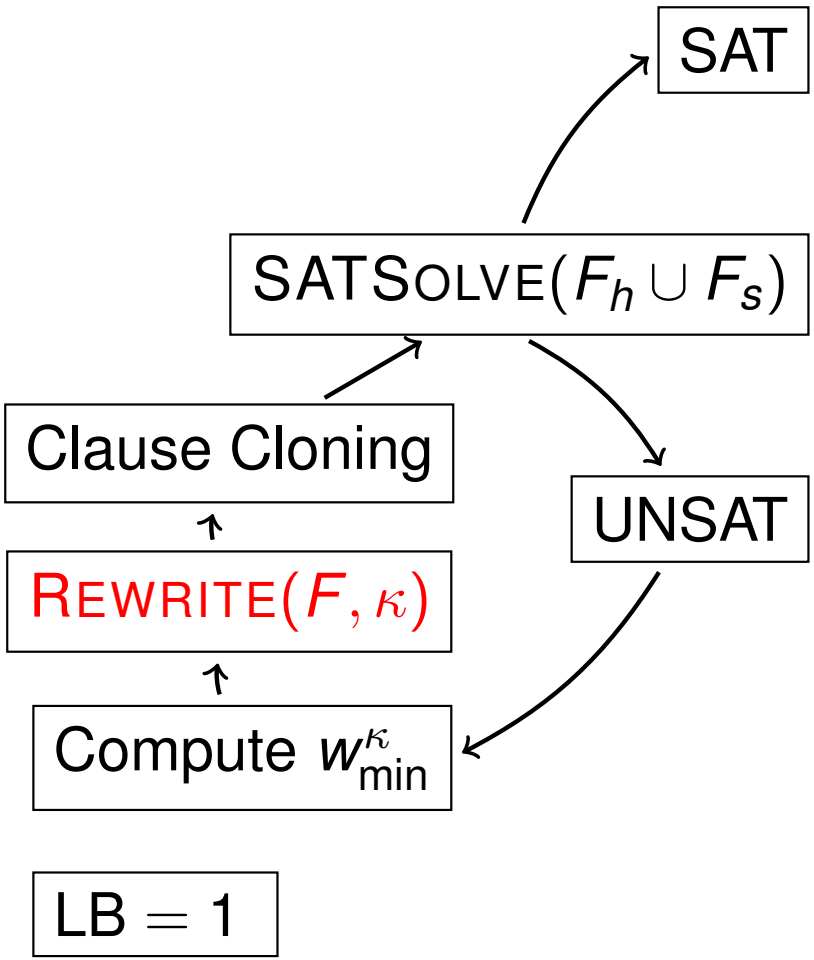
Example


SAT
Return τ
SATSOLVE($F_h \cup F_s$)
Clause Cloning
UNSAT
REWRITE(F, κ)
Compute w^{κ}_{min}
LB $= 0$
$\kappa = \{((\neg \mathbf{x}), \mathbf{3}), ((\neg \mathbf{y}), \mathbf{1})\}$
$F_h = \{(x \vee z), (x \vee y)\}$
$F_s = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$

Example


SAT
Return τ
SATSOLVE($F_h \cup F_s$)
Clause Cloning
UNSAT
REWRITE(F, κ)
Compute $w^{\kappa}_{\min}$
LB $= 0 + 1$
$\kappa = \{((\neg \mathbf{x}), \mathbf{3}), ((\neg \mathbf{y}), \mathbf{1})\}$
$w^{\kappa}_{\min} = 1$
$F_h = \{(x \vee z), (x \vee y)\}$
$F_s = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$

Example

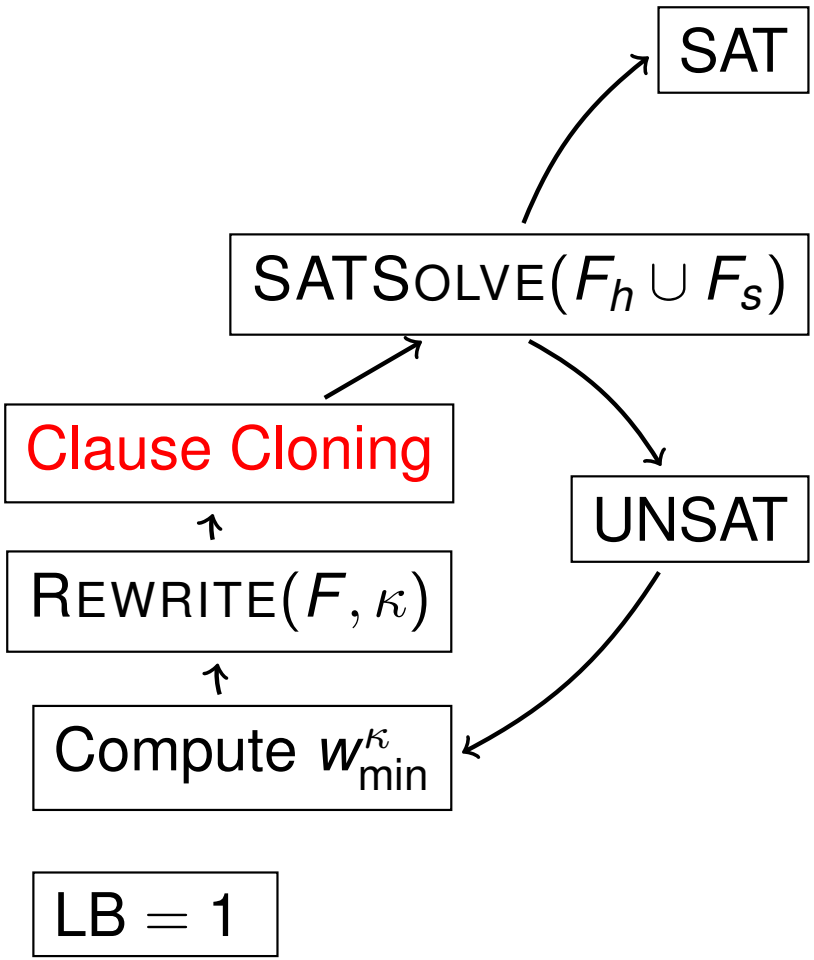


$\kappa = \{((\neg \mathbf{x}), \mathbf{3}), ((\neg \mathbf{y}), \mathbf{1})\}$
$w^{\kappa}_{\min} = 1$

$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2)\}$

$F_s = \{\cancel{(\neg x, 3)}, \cancel{(\neg y, 1)}, (\neg z, 1), (\text{CNF}(r_1 + r_2 = 1), 1)\}$

Example

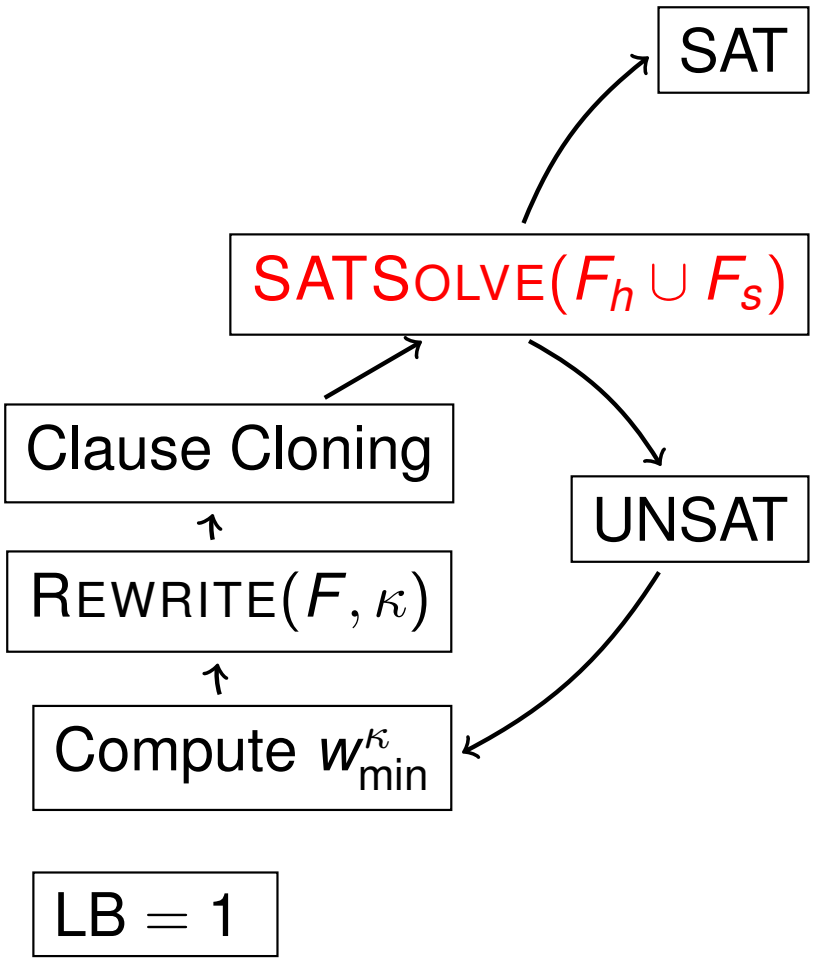


$\kappa = \{((\neg \mathbf{x}), \mathbf{3}), ((\neg \mathbf{y}), \mathbf{1})\}$
$w^{\kappa}_{\min} = 1$

$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2)\}$

$F_s = \{(\neg z, 1), (\neg x, 2), (\text{CNF}(r_1 + r_2 = 1), 1)\}$

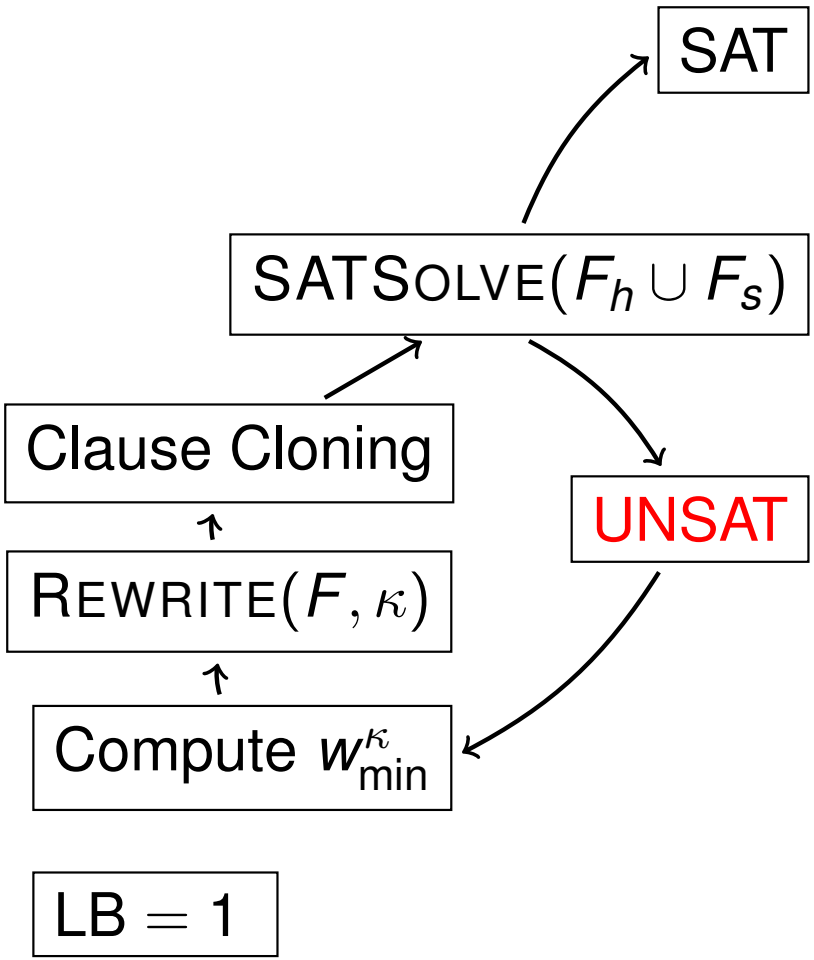
Example



$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2)\}$

$F_s = \{(\neg z, 1), (\neg x, 2), (\text{CNF}(r_1 + r_2 = 1), 1)\}$

Example

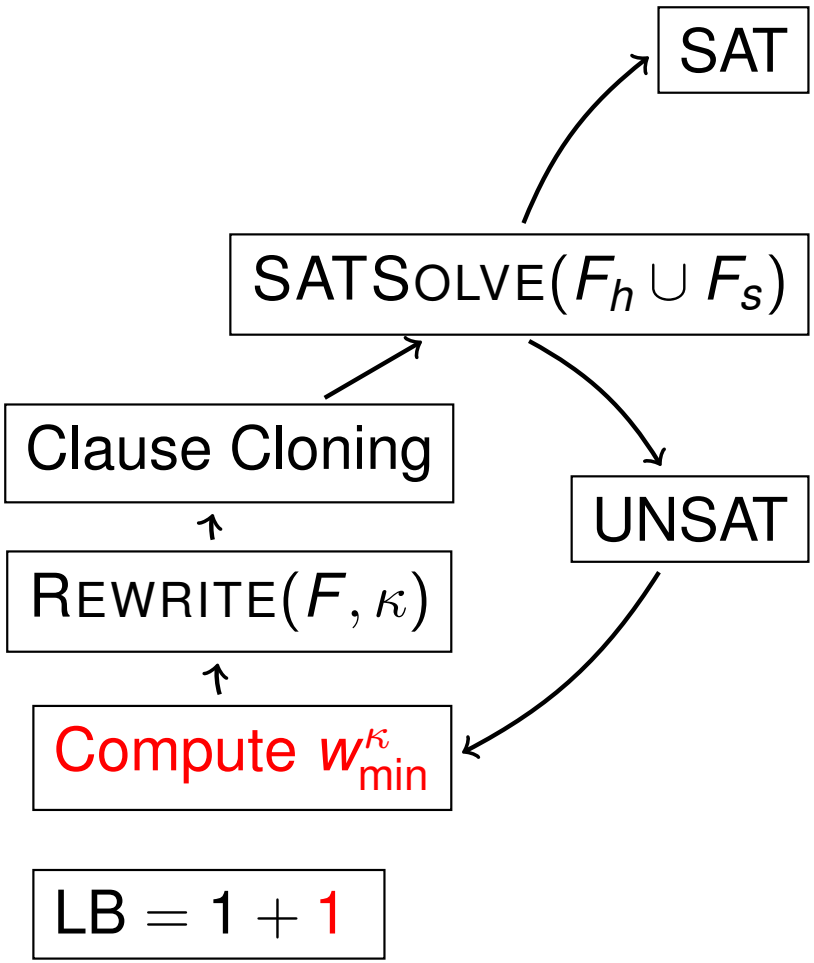


$\kappa = \{((\neg \mathbf{x}), \mathbf{2}), ((\neg \mathbf{z}), \mathbf{1})\}$

$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2)\}$

$F_s = \{(\neg z, 1), (\neg x, 2), (\text{CNF}(r_1 + r_2 = 1), 1)\}$

Example

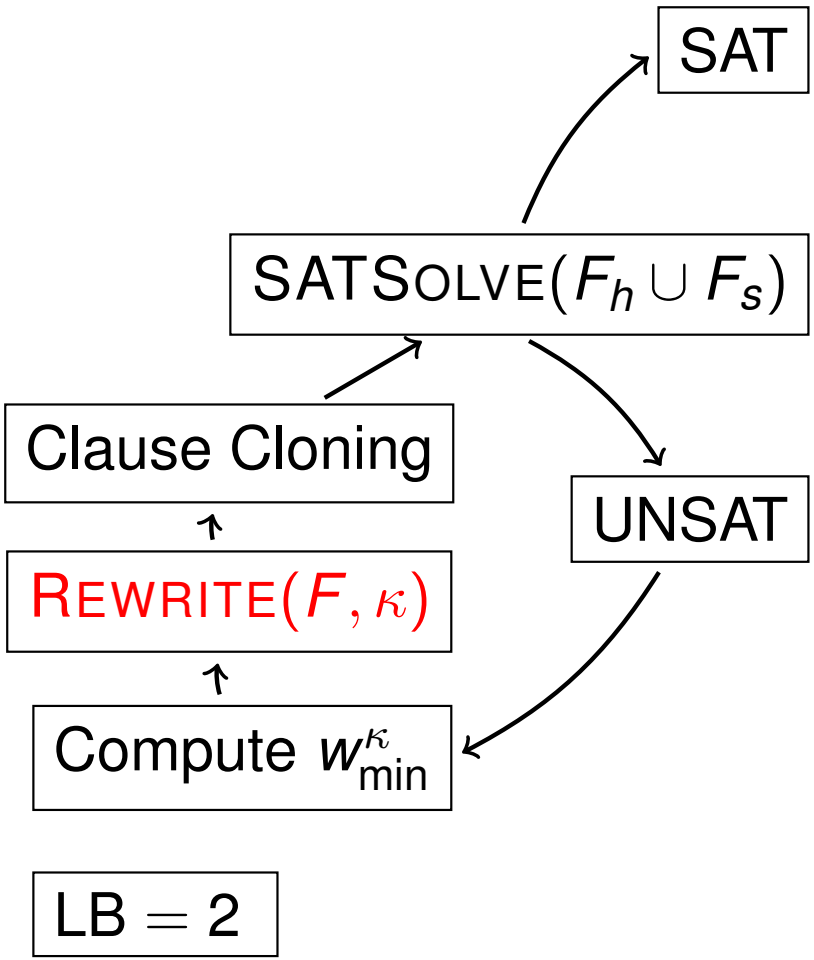


$\kappa = \{((\neg \mathbf{x}), \mathbf{2}), ((\neg \mathbf{z}), \mathbf{1})\}$
$w^{\kappa}_{\min} = 1$

$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2)\}$

$F_s = \{(\neg z, 1), (\neg x, 2), (\text{CNF}(r_1 + r_2 = 1), 1)\}$

Example

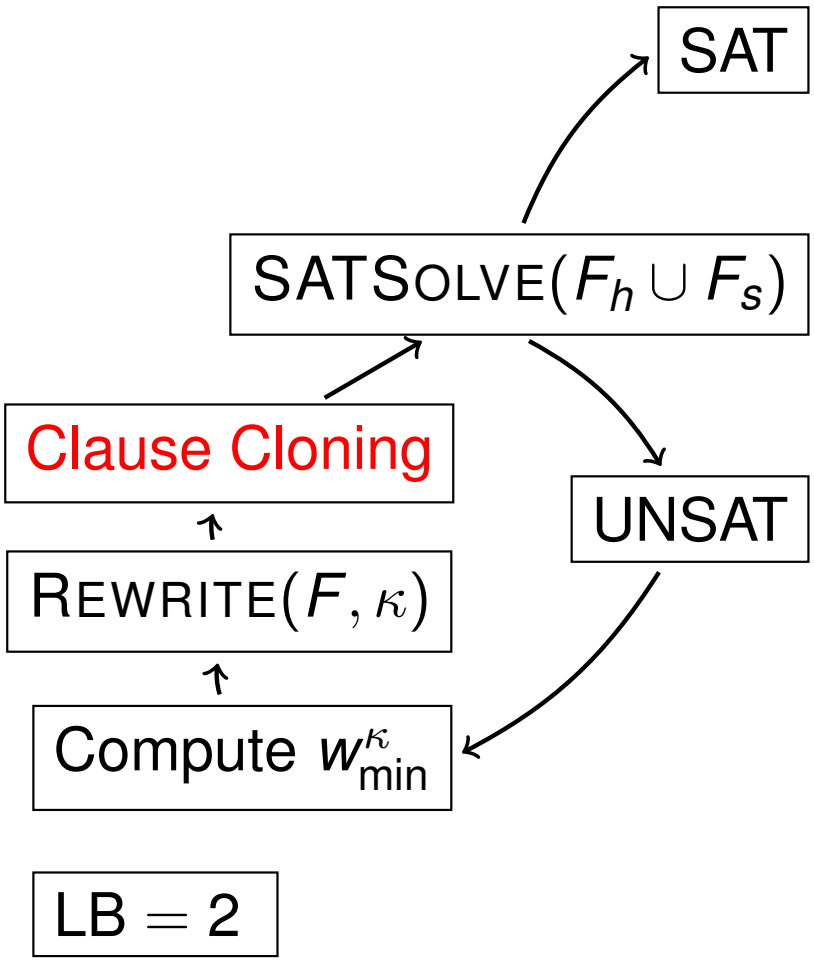


$\kappa = \{((\neg \mathbf{x}), \mathbf{2}), ((\neg \mathbf{z}), \mathbf{1})\}$
$w^{\kappa}_{\min} = 1$

$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1),$
$(\neg y \vee r_2), (\neg x \vee r_3), (\neg z \vee r_4)\}$

$F_s = \{\cancel{(\neg z, 1)}, \cancel{(\neg x, 2)},$
$(\text{CNF}(r_1 + r_2 = 1), 1),$
$(\text{CNF}(r_3 + r_4 = 1), 1)\}$

Example

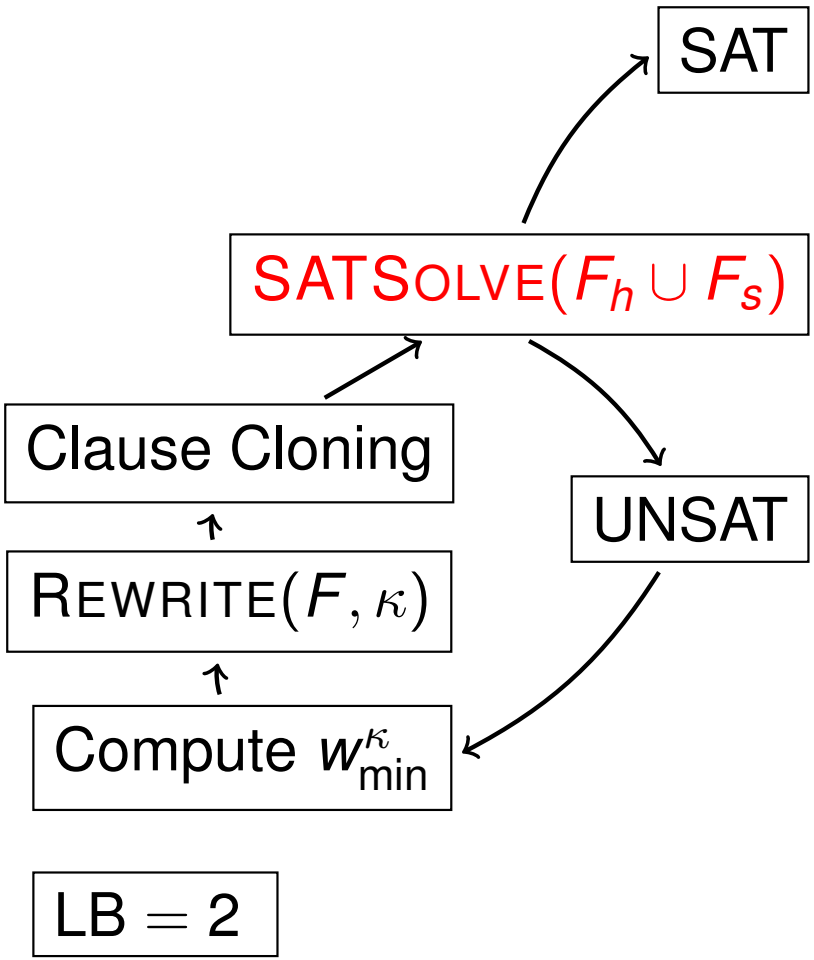


$\kappa = \{((\neg \mathbf{x}), \mathbf{2}), ((\neg \mathbf{z}), \mathbf{1})\}$
$w^{\kappa}_{\min} = 1$

$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2), (\neg x \vee r_3), (\neg z \vee r_4)\}$

$F_s = \{(\neg x, 1), (\text{CNF}(r_1 + r_2 = 1), 1), (\text{CNF}(r_3 + r_4 = 1), 1)\}$

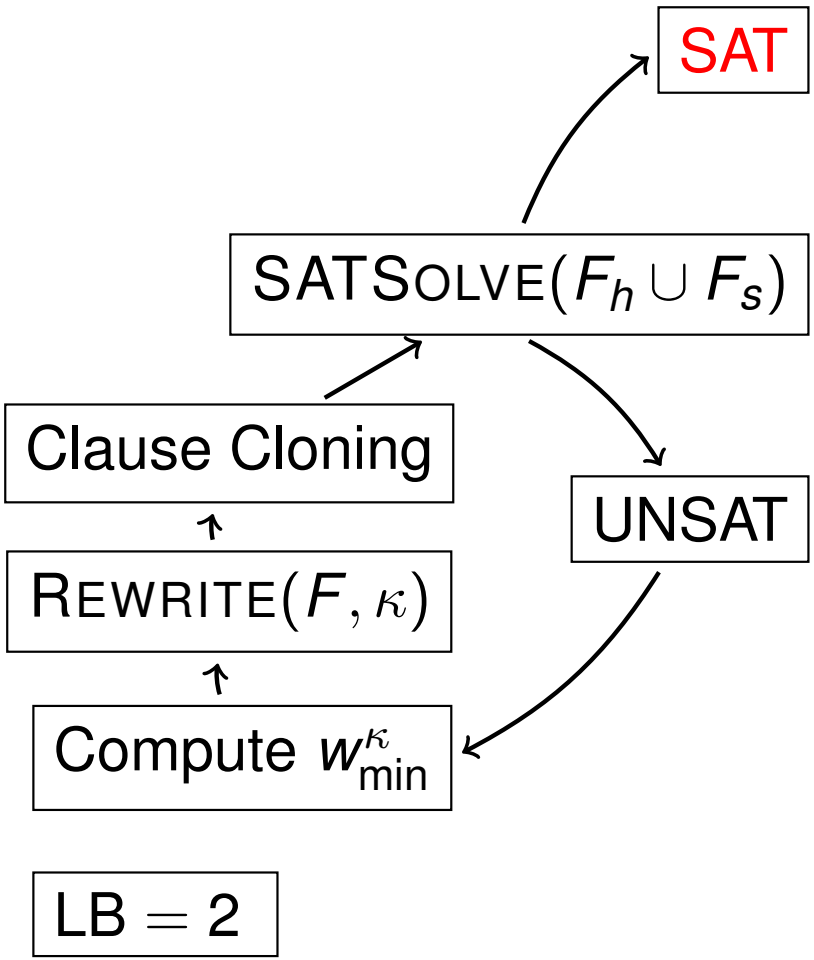
Example



$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2), (\neg x \vee r_3), (\neg z \vee r_4)\}$

$F_s = \{(\neg x, 1), (\text{CNF}(r_1 + r_2 = 1), 1), (\text{CNF}(r_3 + r_4 = 1), 1)\}$

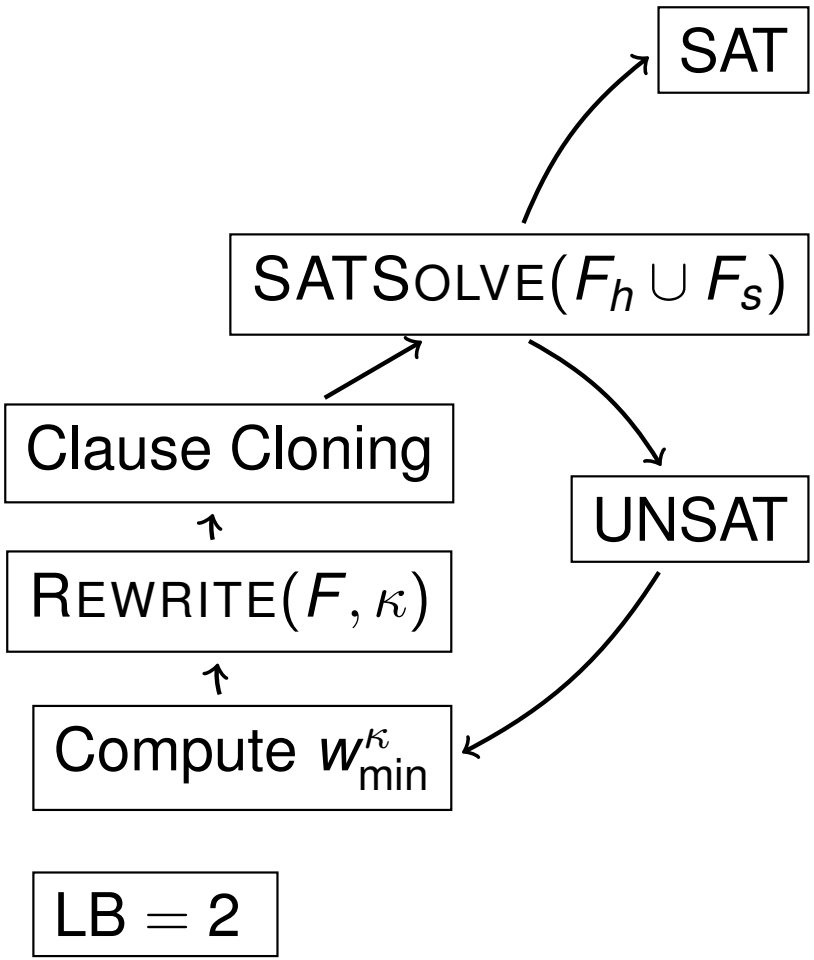
Example



$$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2), (\neg x \vee r_3), (\neg z \vee r_4)\}$$

$$F_s = \{(\neg x, 1), (\text{CNF}(r_1 + r_2 = 1), 1), (\text{CNF}(r_3 + r_4 = 1), 1)\}$$

Example



$$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2), (\neg x \vee r_3), (\neg z \vee r_4)\}$$

$$F_s = \{(\neg x, 1), (\text{CNF}(r_1 + r_2 = 1), 1), (\text{CNF}(r_3 + r_4 = 1), 1)\}$$

Observations

- Due to clause weights, both cores in previous example need to be extracted.
- Both can be extracted without rewriting.
- Adding relaxation variables to clause clones introduces implicit equivalences
- Relaxation variables can be reused.

Observations

- Due to clause weights, both cores in previous example need to be extracted.
- Both can be extracted without rewriting.
- Adding relaxation variables to clause clones introduces implicit equivalences
- Relaxation variables can be reused.

Previous Example, κ valid without rewriting

$\kappa = \{((\neg x), 2), ((\neg z), 1)\}$

$F_h = \{(x \vee z), (x \vee y), \cancel{(\neg x \vee r_1)}, \cancel{(\neg y \vee r_2)}\}$

$F_s = \{((\neg z), 1), ((\neg x), 2), \cancel{(\mathrm{CNF}(r_1 + r_2 = 1), 1)}\}$

Observations

- Due to clause weights, both cores in previous example need to be extracted.
- Both can be extracted without rewriting.
- Adding relaxation variables to clause clones introduces implicit equivalences
- Relaxation variables can be reused.

Equivalence between relaxation variables

$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2), (\neg x \vee r_3), (\neg z \vee r_4)\}$
$F_s = \{(\neg x, 1), (\mathrm{CNF}(r_1 + r_2 = 1), 1), (\mathrm{CNF}(r_3 + r_4 = 1), 1)\}$
$r_1 \leftrightarrow r_3$ in any optimal solution.

Observations

- Due to clause weights, both cores in previous example need to be extracted.
- Both can be extracted without rewriting.
- Adding relaxation variables to clause clones introduces implicit equivalences
- Relaxation variables can be reused.

Equivalence between relaxation variables

$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2), (\neg x \vee r_1), (\neg z \vee r_4)\}$
$F_s = \{(\neg x, 1), (\text{CNF}(r_1 + r_2 = 1), 1), (\text{CNF}(r_1 + r_4 = 1), 1)\}$
$r_1 \leftrightarrow r_3$ in any optimal solution.

Observations

- Due to clause weights, both cores in previous example need to be extracted.
- Both can be extracted without rewriting.
- Adding relaxation variables to clause clones introduces implicit equivalences
- Relaxation variables can be reused.

Equivalence between relaxation variables

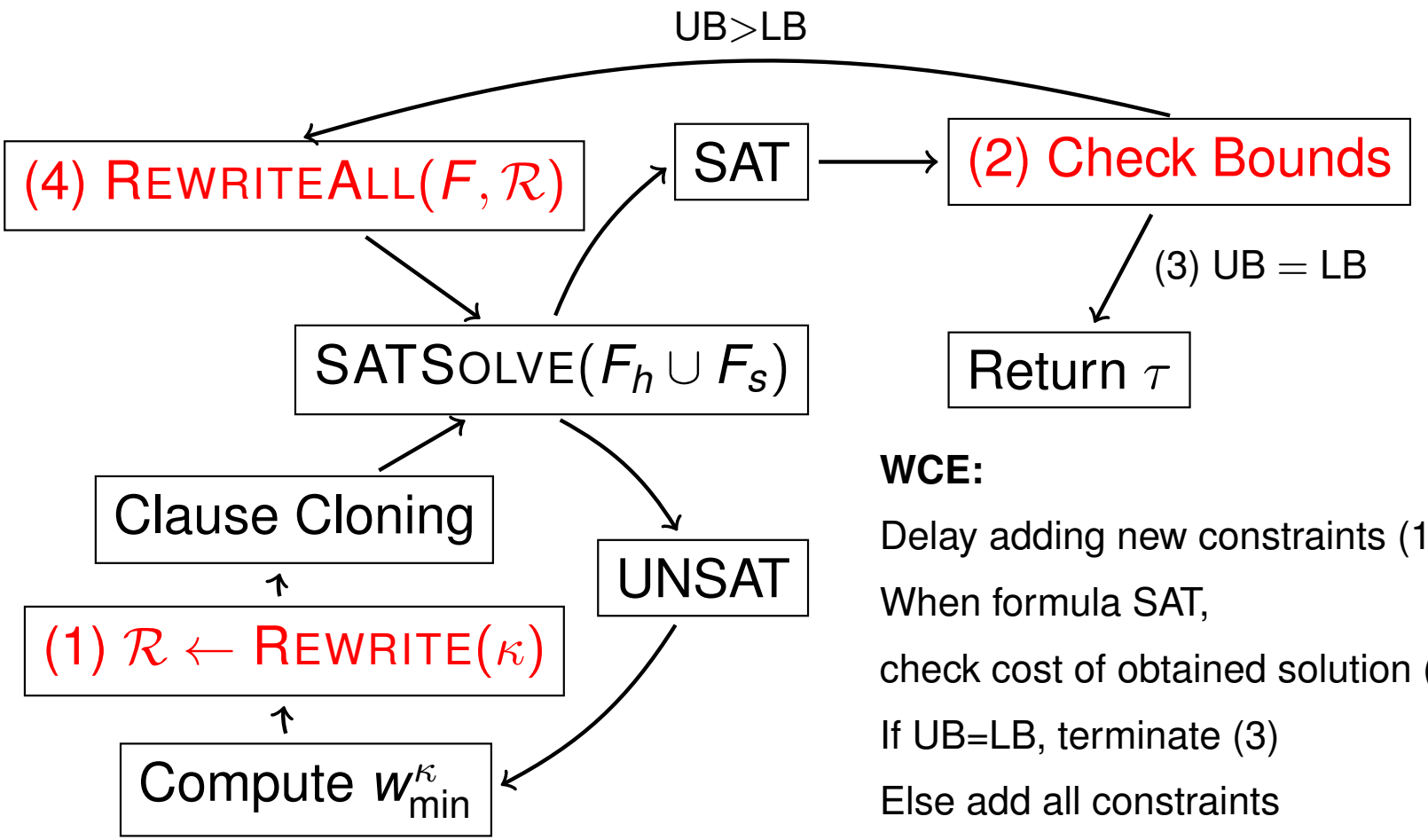
$F_h = \{(x \vee z), (x \vee y), (\neg x \vee r_1), (\neg y \vee r_2), (\neg z \vee r_4)\}$
$F_s = \{(\neg x, 1), (\text{CNF}(r_1 + r_2 = 1), 1), (\text{CNF}(r_1 + r_4 = 1), 1)\}$
$r_1 \leftrightarrow r_3$ in any optimal solution.

Weight Aware Core Extraction

Weight Aware Core Extraction (WCE)

- Delay relaxation steps, extract more valid cores.
- Easier on the SAT solver.
- Generic technique.
- Resembles earlier work on computing lower bounds Heras et al. [2012]

Core Guided weighted MaxSAT with WCE



WCE:

Delay adding new constraints (1)

When formula SAT,

check cost of obtained solution (UB) (2)

If UB=LB, terminate (3)

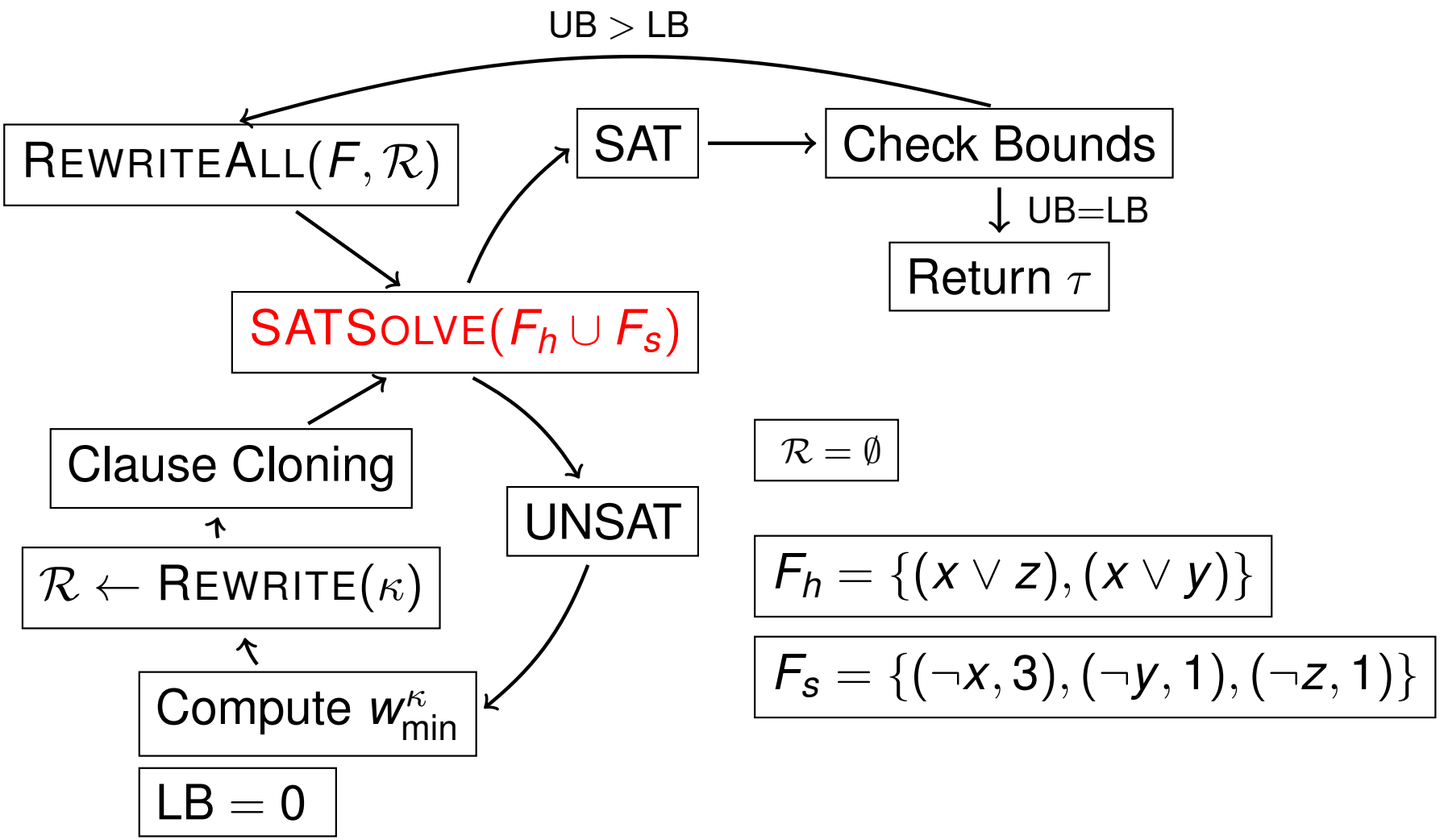
Else add all constraints

to formula (4)

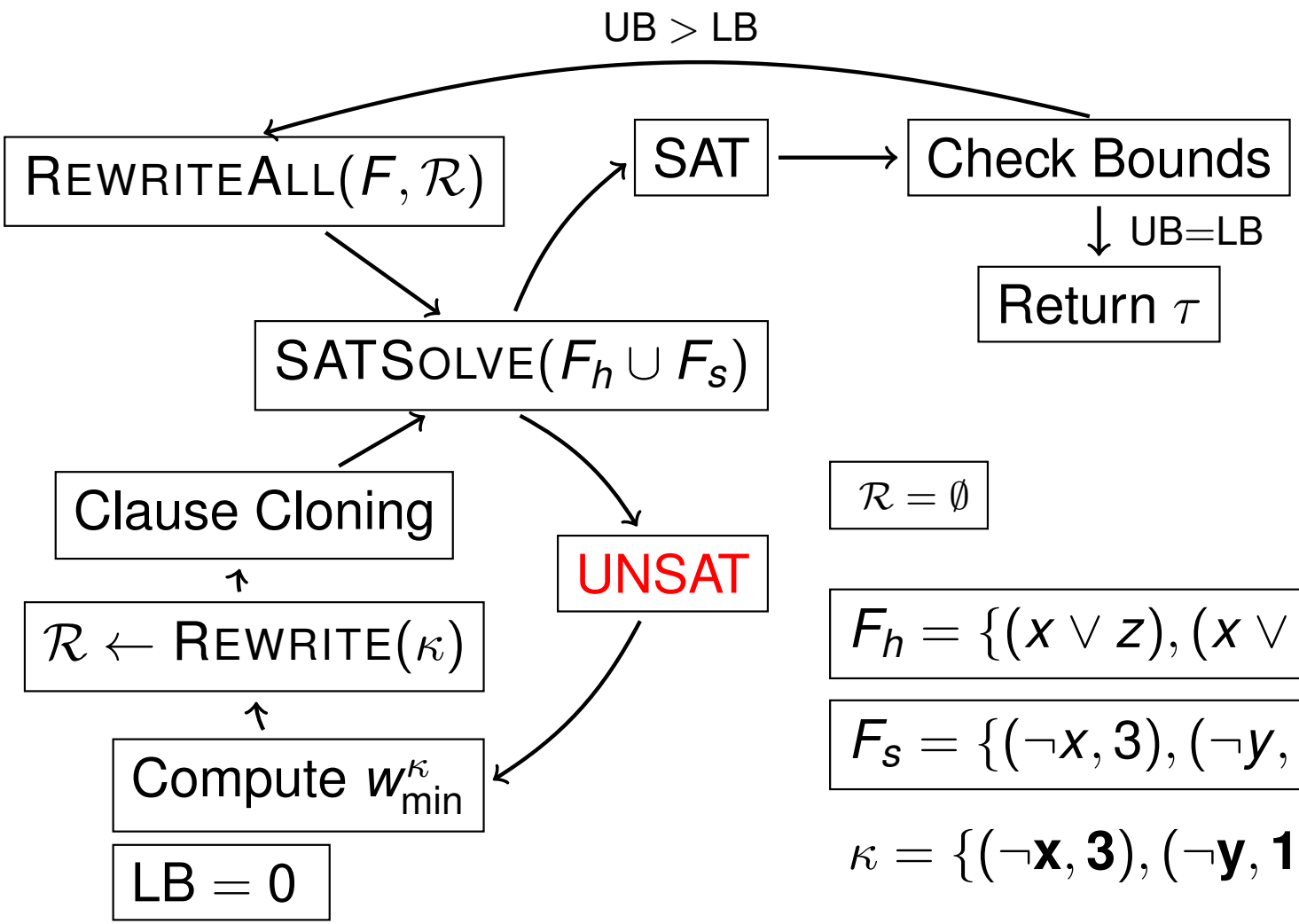
Example with WCE

$$F_h = \{(x \vee z), (x \vee y)\}$$
$$F_s = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$$

Example with WCE



Example with WCE



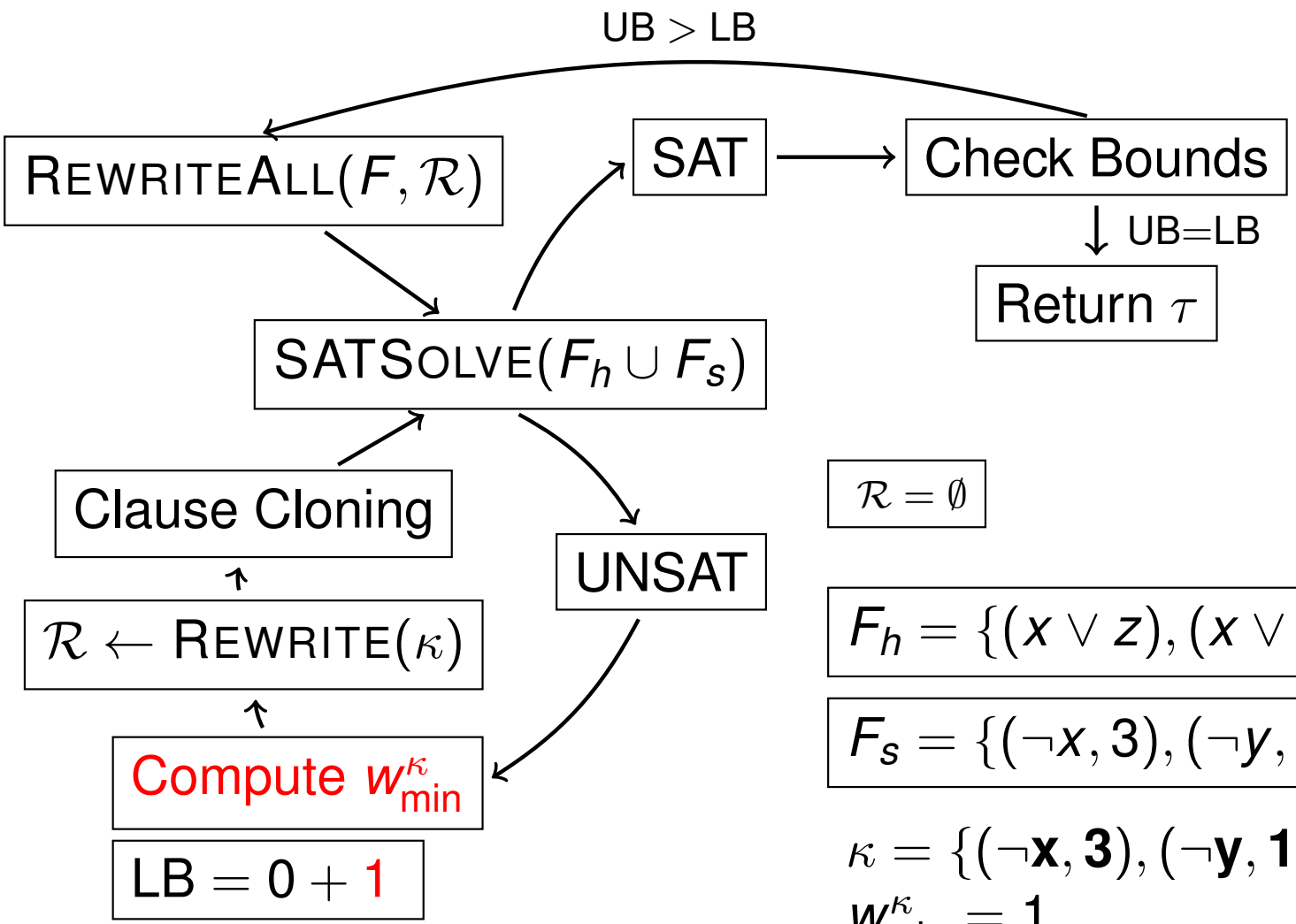
$\mathcal{R} = \emptyset$

$F_h = \{(x \vee z), (x \vee y)\}$

$F_s = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$

$\kappa = \{(\neg \mathbf{x}, \mathbf{3}), (\neg \mathbf{y}, \mathbf{1})\}$

Example with WCE



$\mathcal{R} = \emptyset$

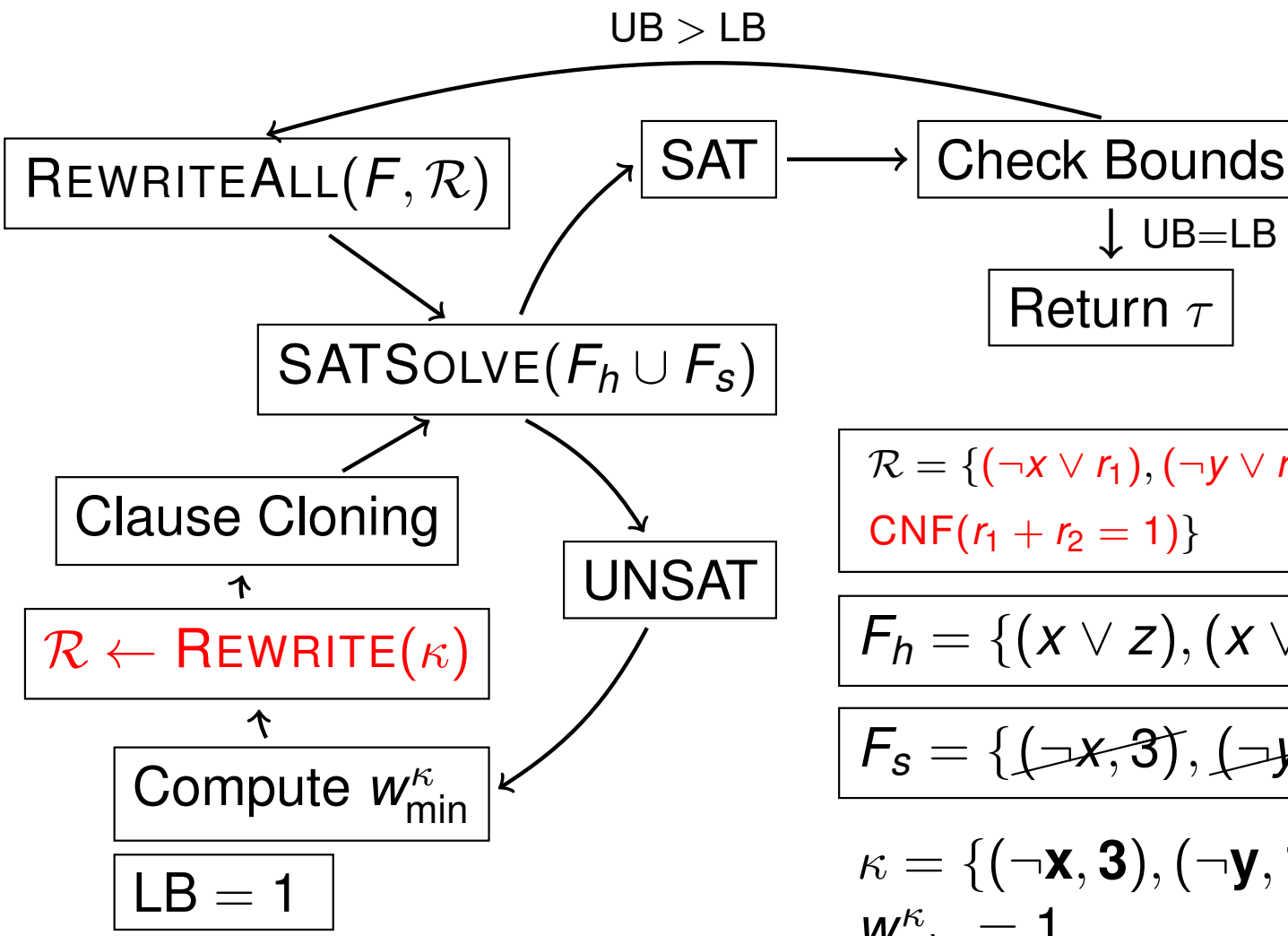
$F_h = \{(x \vee z), (x \vee y)\}$

$F_s = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$

$\kappa = \{(\neg \mathbf{x}, \mathbf{3}), (\neg \mathbf{y}, \mathbf{1})\}$

$w^{\kappa}_{\text{min}} = 1$

Example with WCE



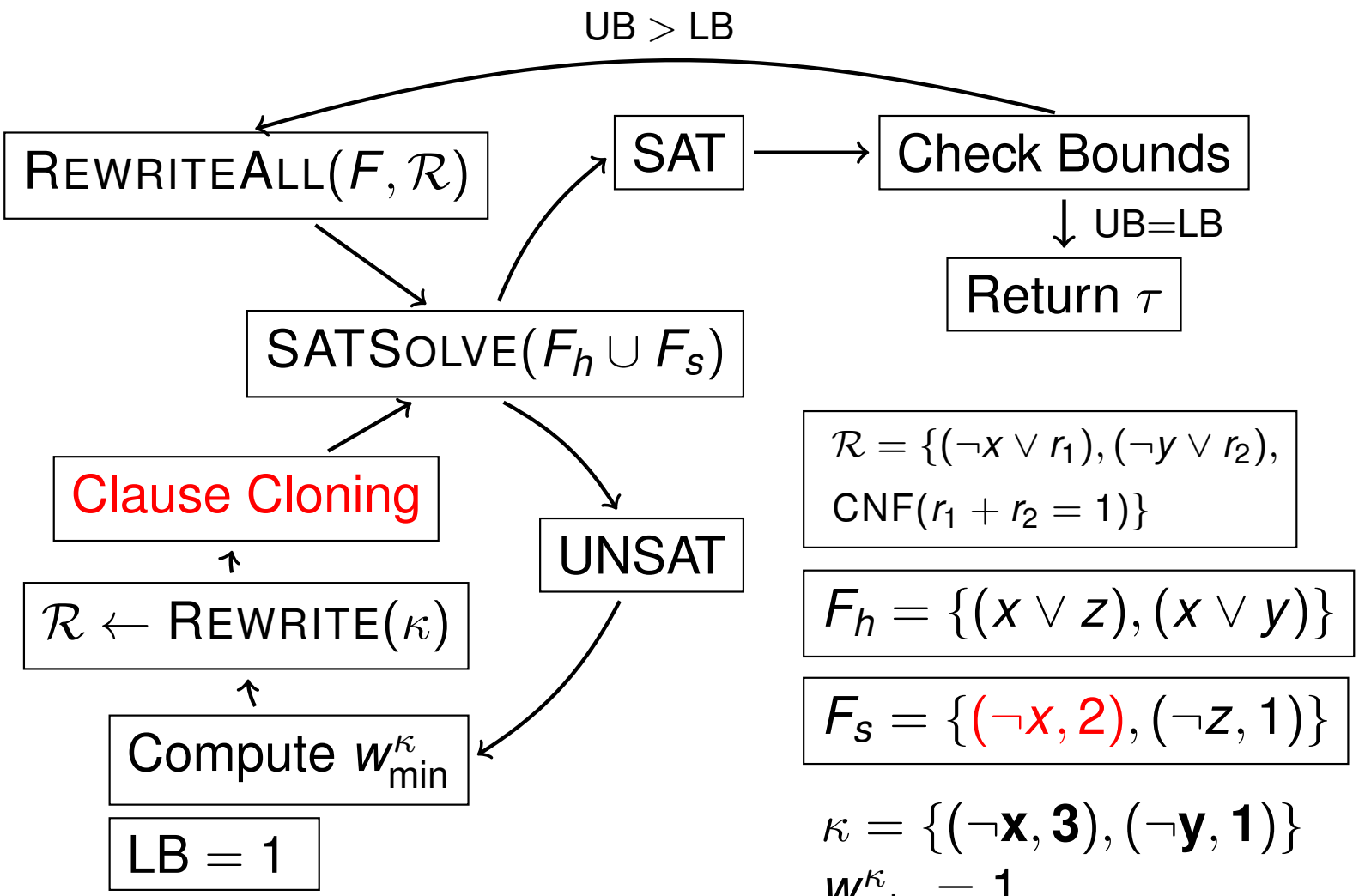
$\mathcal{R} = \{(\neg x \vee r_1), (\neg y \vee r_2),$
$\mathrm{CNF}(r_1 + r_2 = 1)\}$

$F_h = \{(x \vee z), (x \vee y)\}$

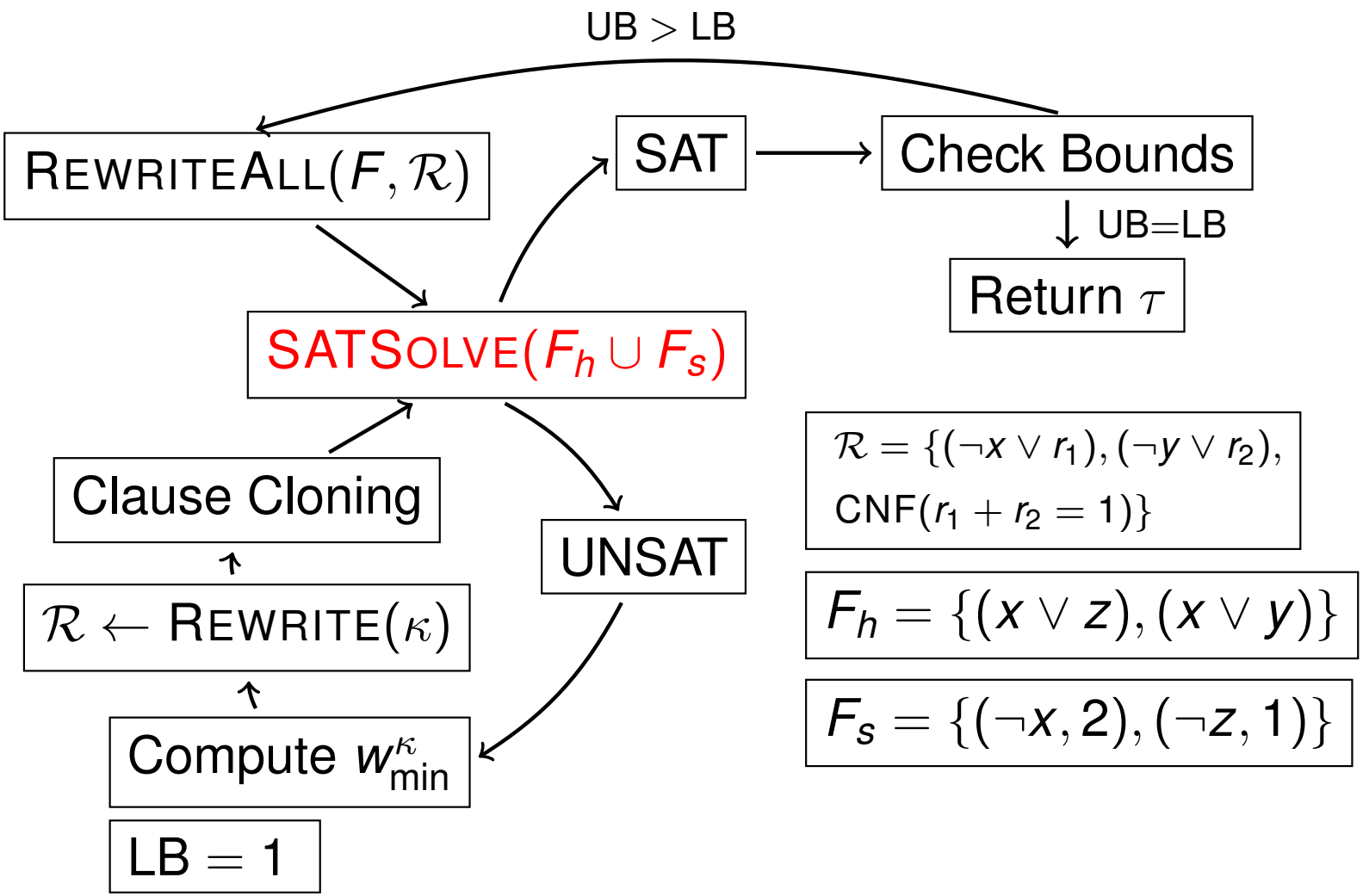
$F_s = \{\cancel{(\neg x, 3)}, \cancel{(\neg y, 1)}, (\neg z, 1)\}$

$\kappa = \{(\neg \mathbf{x}, \mathbf{3}), (\neg \mathbf{y}, \mathbf{1})\}$
$w^{\kappa}_{\min} = 1$

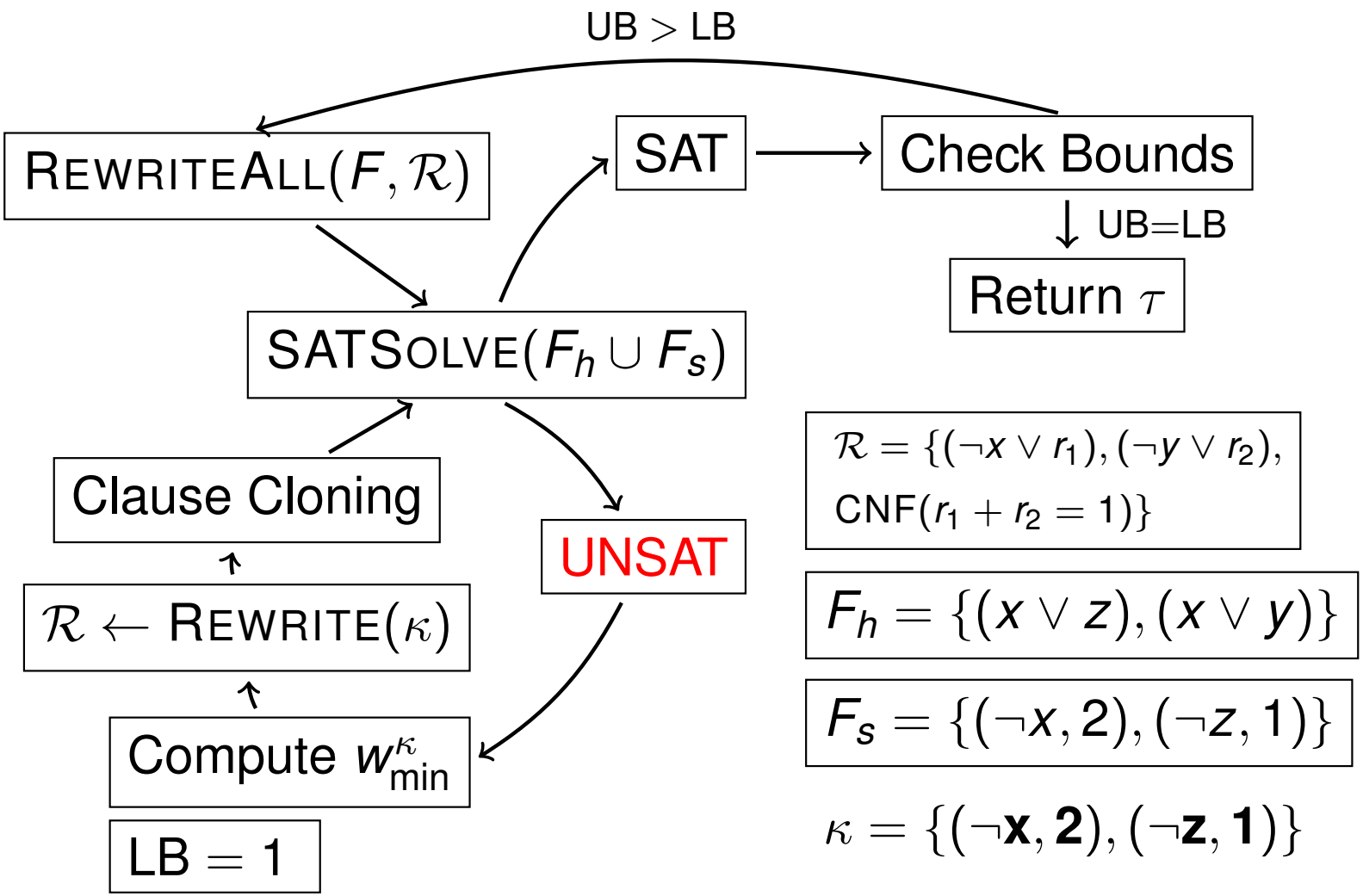
Example with WCE



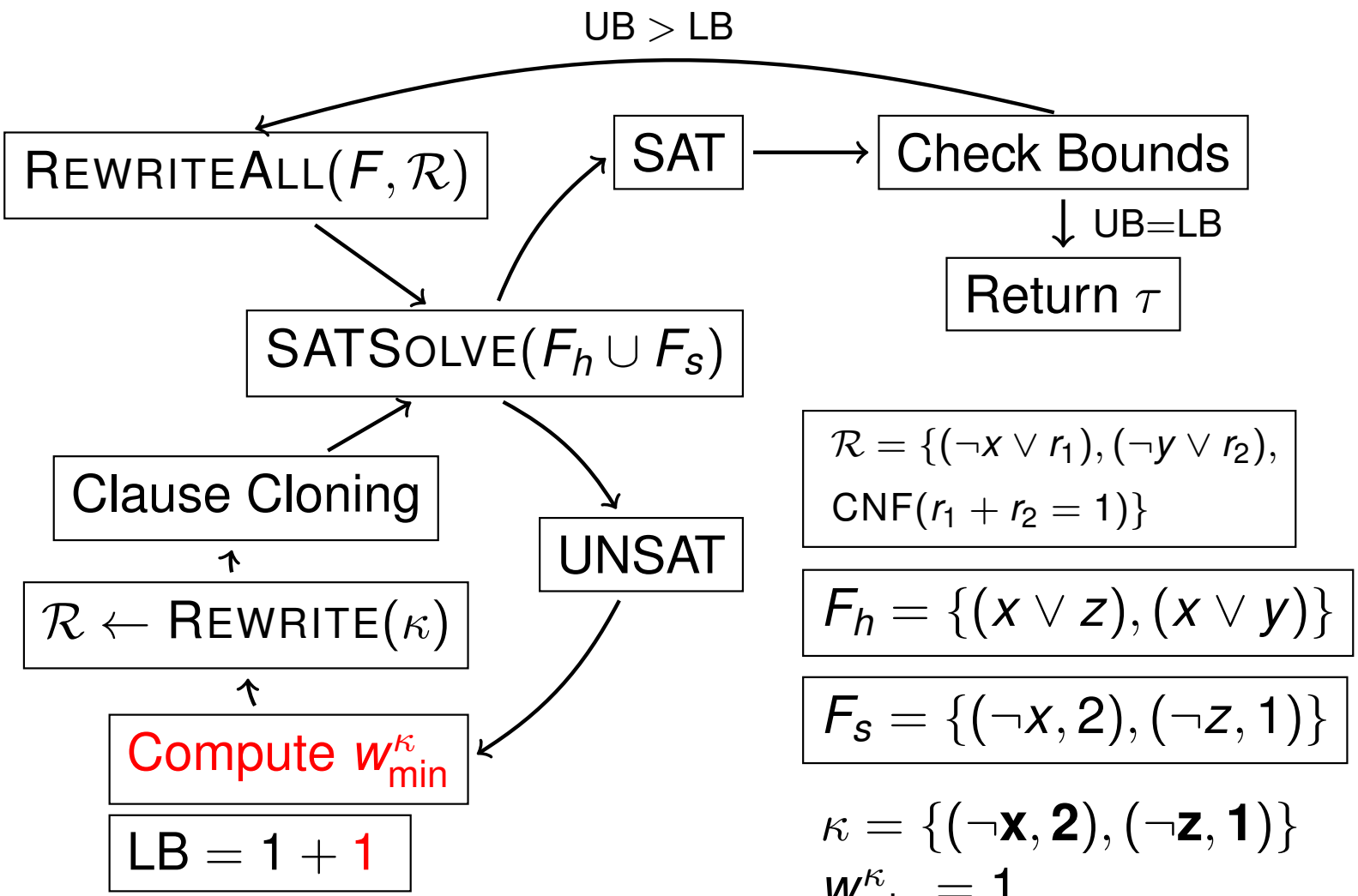
Example with WCE



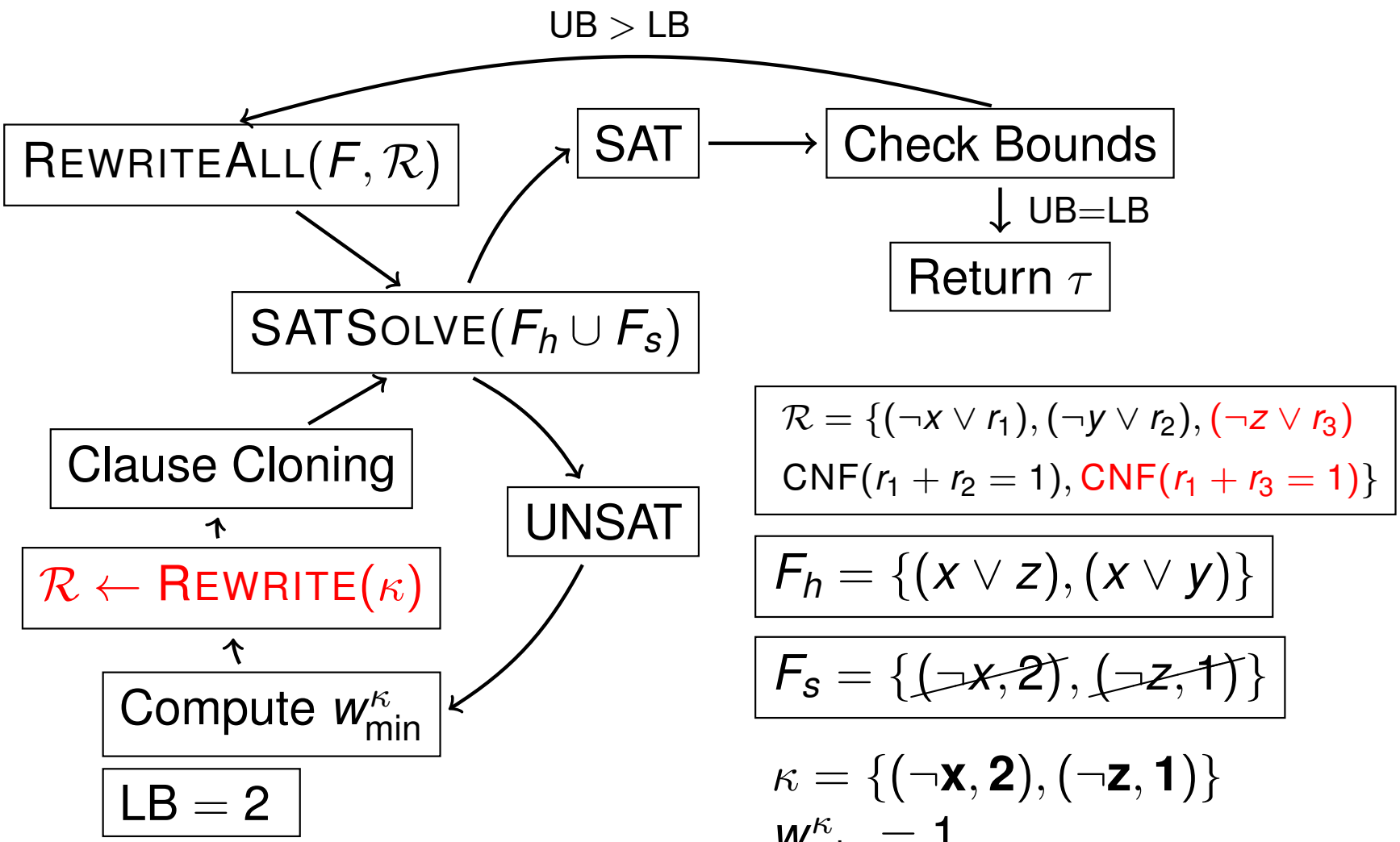
Example with WCE



Example with WCE



Example with WCE



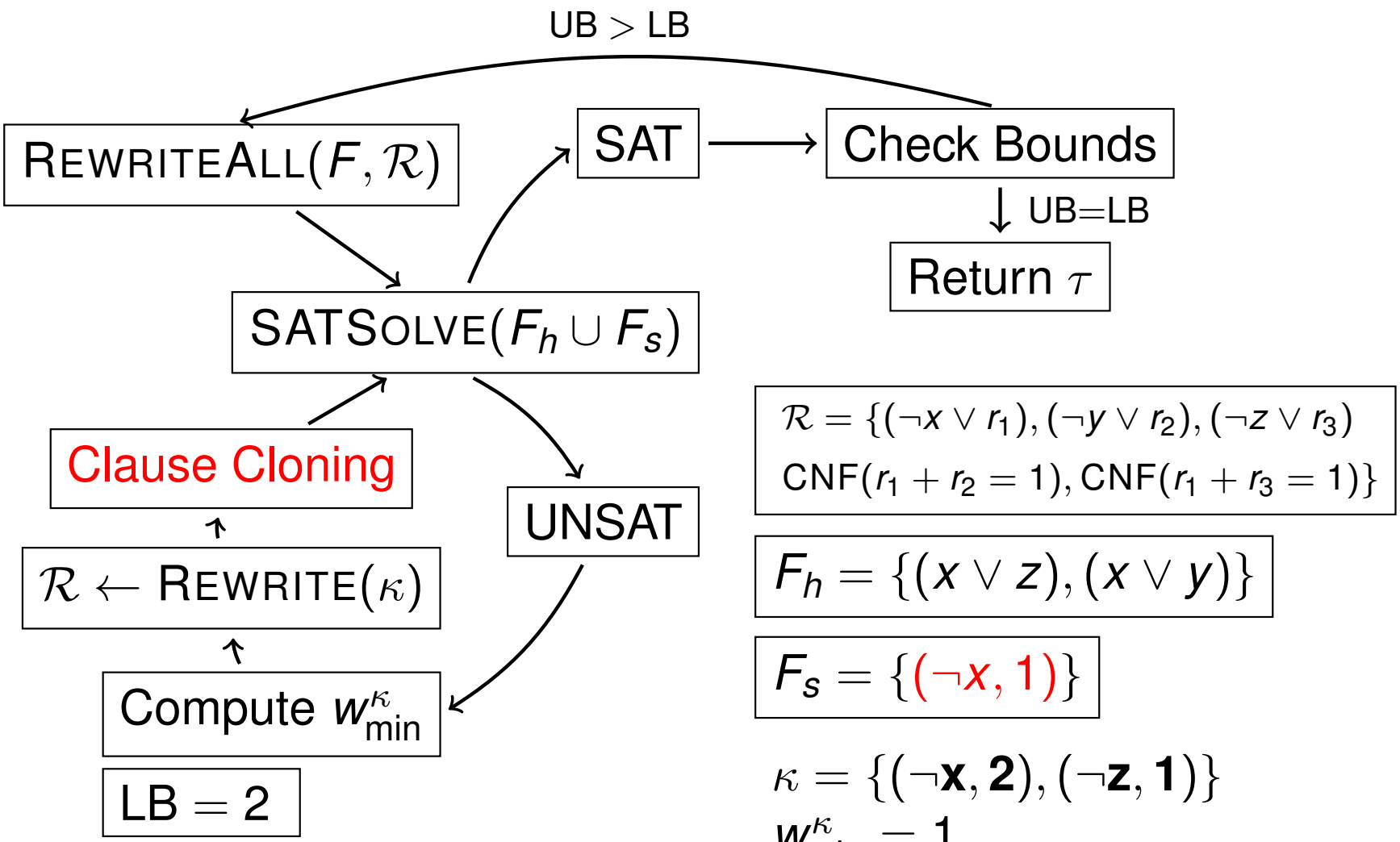
$\mathcal{R} = \{(\neg x \vee r_1), (\neg y \vee r_2), (\neg z \vee r_3)$
$\text{CNF}(r_1 + r_2 = 1), \text{CNF}(r_1 + r_3 = 1)\}$

$F_h = \{(x \vee z), (x \vee y)\}$

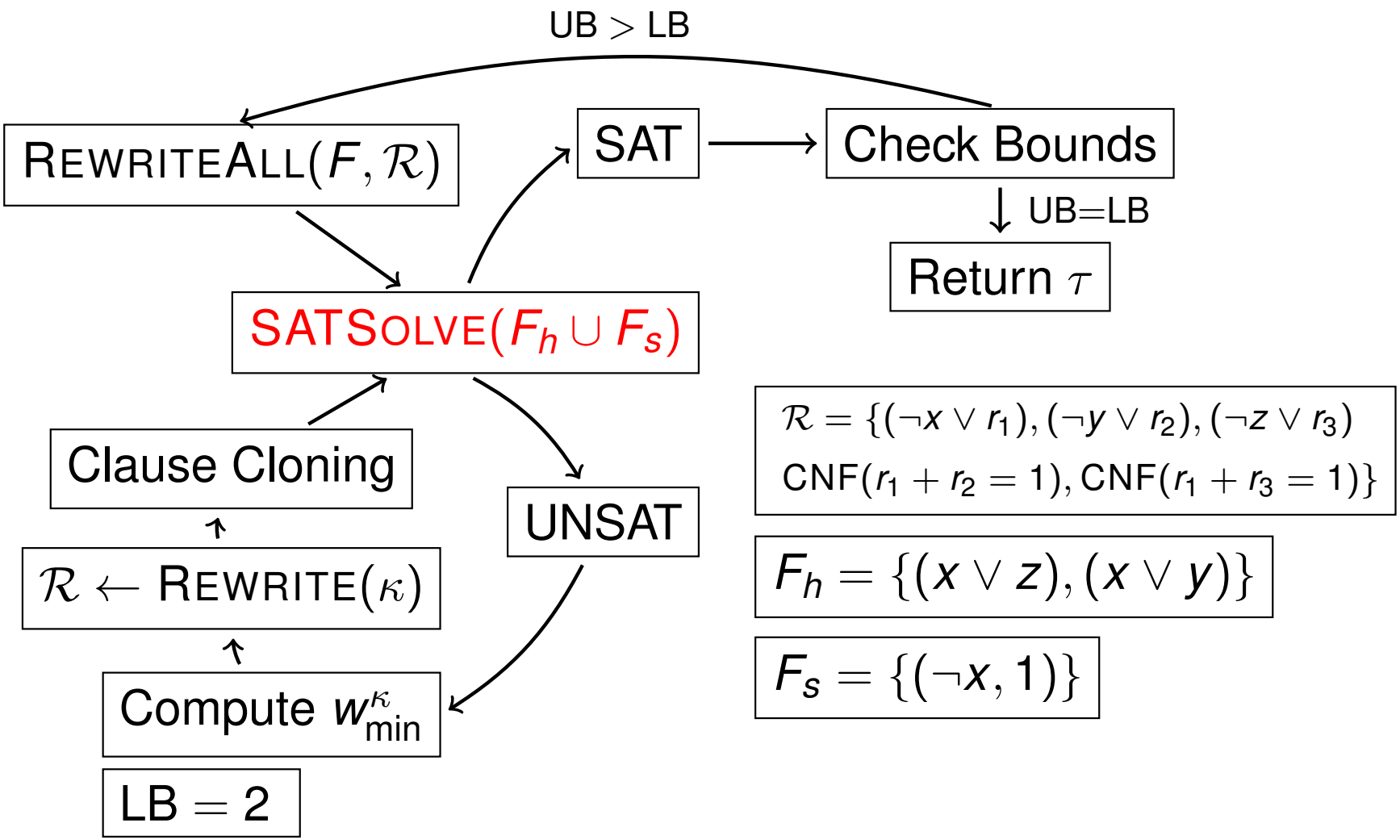
$F_s = \{\cancel{(\neg x, 2)}, \cancel{(\neg z, 1)}\}$

$\kappa = \{(\neg \mathbf{x}, \mathbf{2}), (\neg \mathbf{z}, \mathbf{1})\}$
$w^{\kappa}_{\text{min}} = 1$

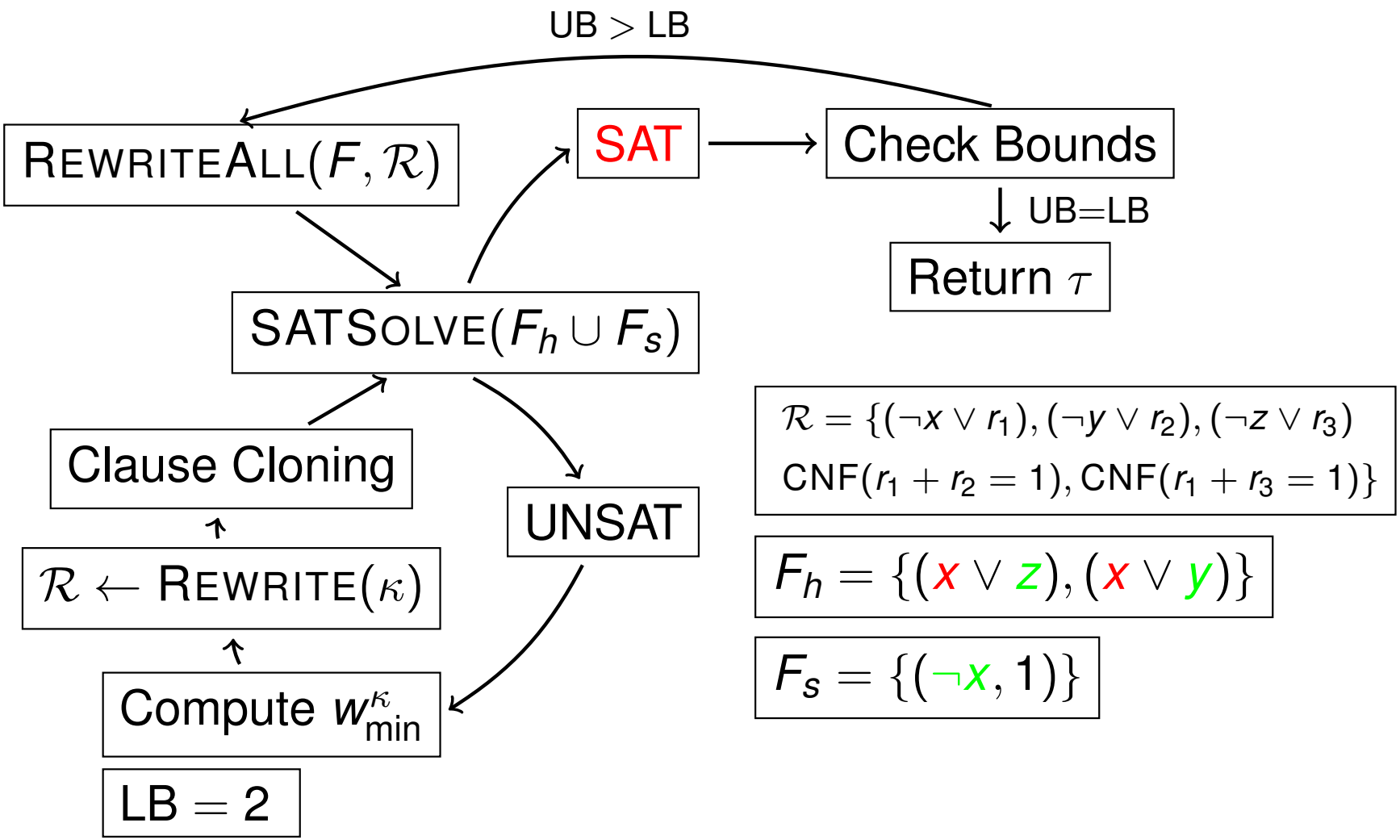
Example with WCE



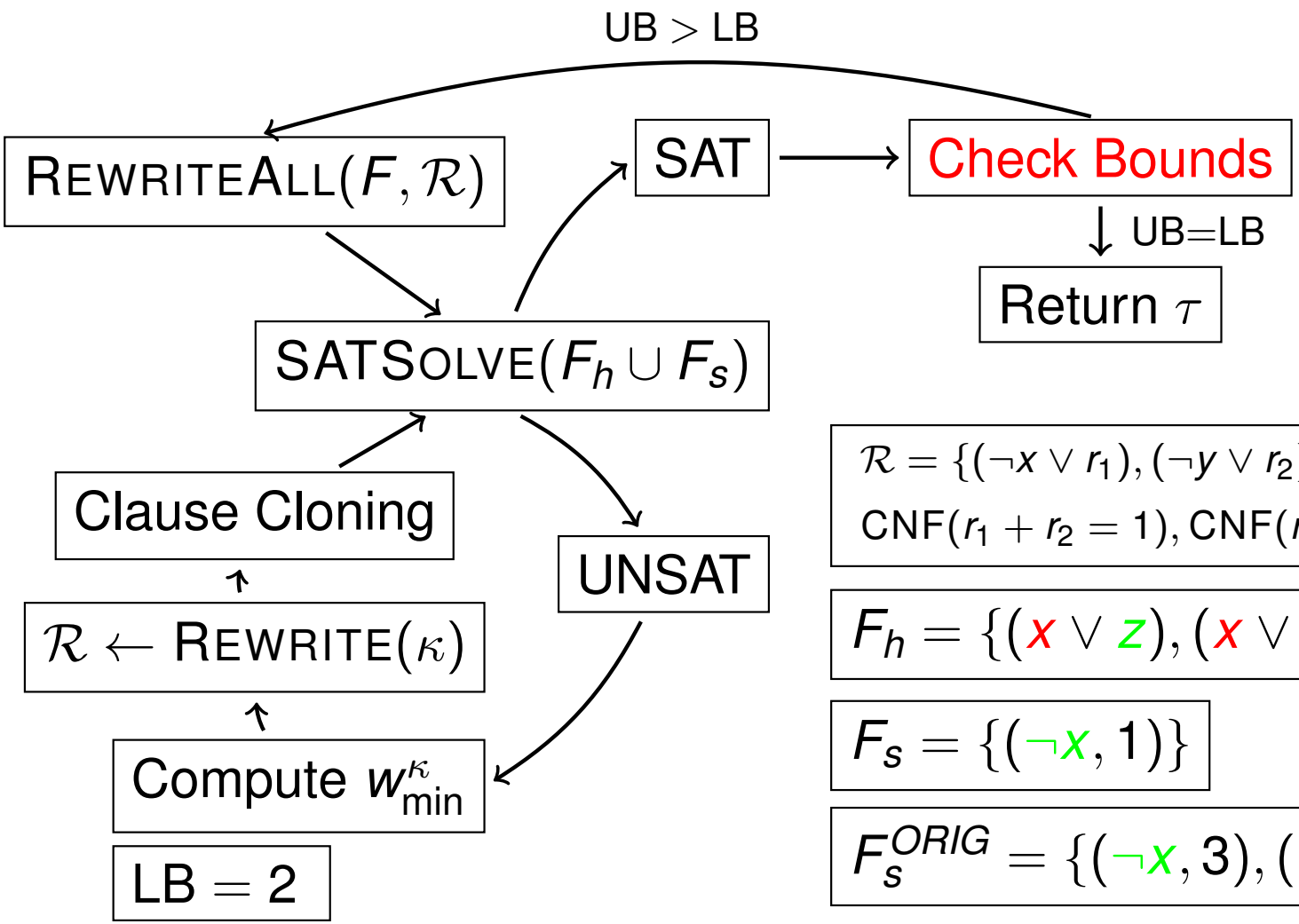
Example with WCE



Example with WCE



Example with WCE



$\mathcal{R} = \{(\neg x \vee r_1), (\neg y \vee r_2), (\neg z \vee r_3)$
$\text{CNF}(r_1 + r_2 = 1), \text{CNF}(r_1 + r_3 = 1)\}$

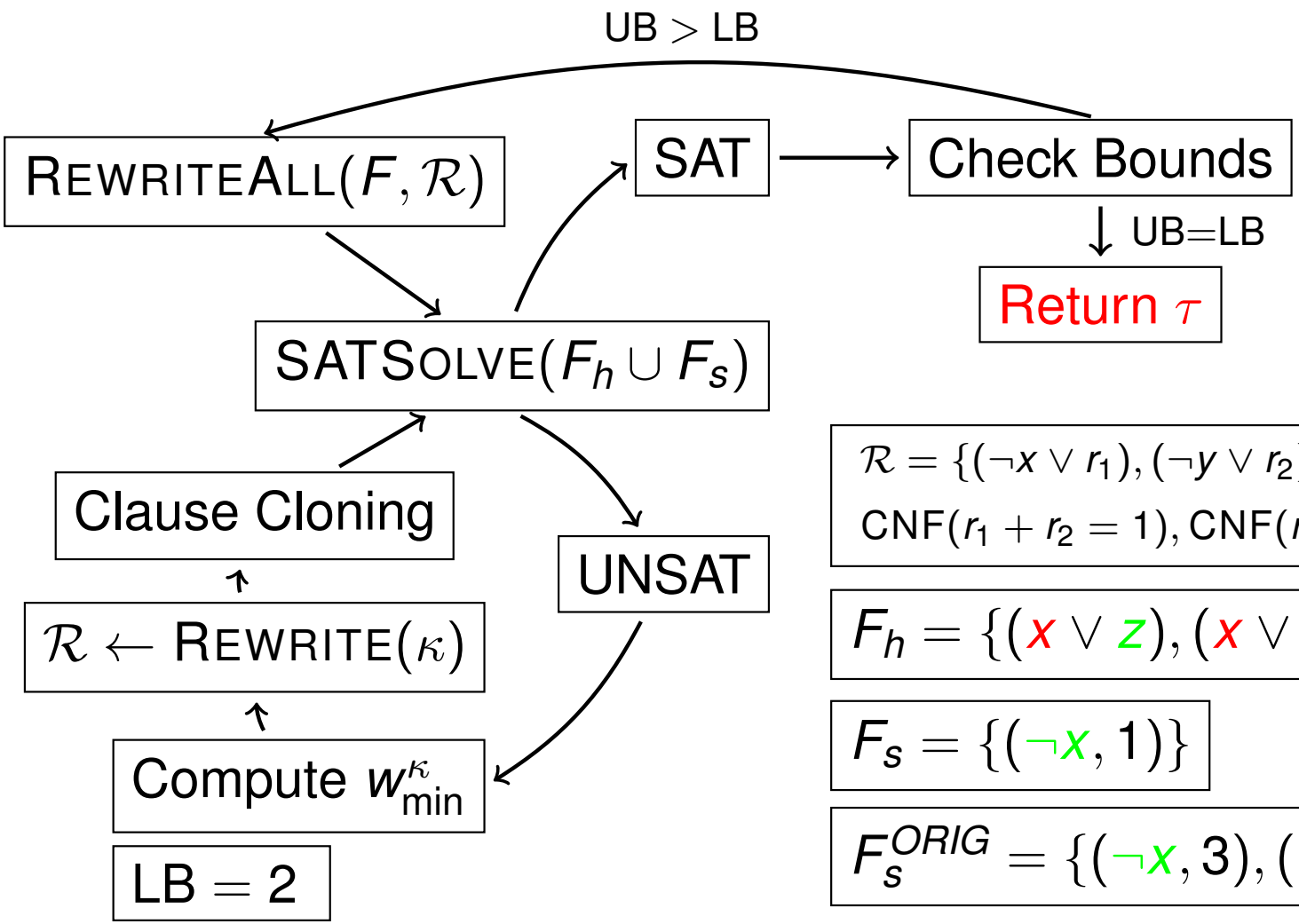
$F_h = \{(x \vee z), (x \vee y)\}$

$F_s = \{(\neg x, 1)\}$

$F_s^{ORIG} = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$

UB=2

Example with WCE



$\mathcal{R} = \{(\neg x \vee r_1), (\neg y \vee r_2), (\neg z \vee r_3)$
$\text{CNF}(r_1 + r_2 = 1), \text{CNF}(r_1 + r_3 = 1)\}$

$F_h = \{(x \vee z), (x \vee y)\}$

$F_s = \{(\neg x, 1)\}$

$F_s^{ORIG} = \{(\neg x, 3), (\neg y, 1), (\neg z, 1)\}$

Implementing WCE

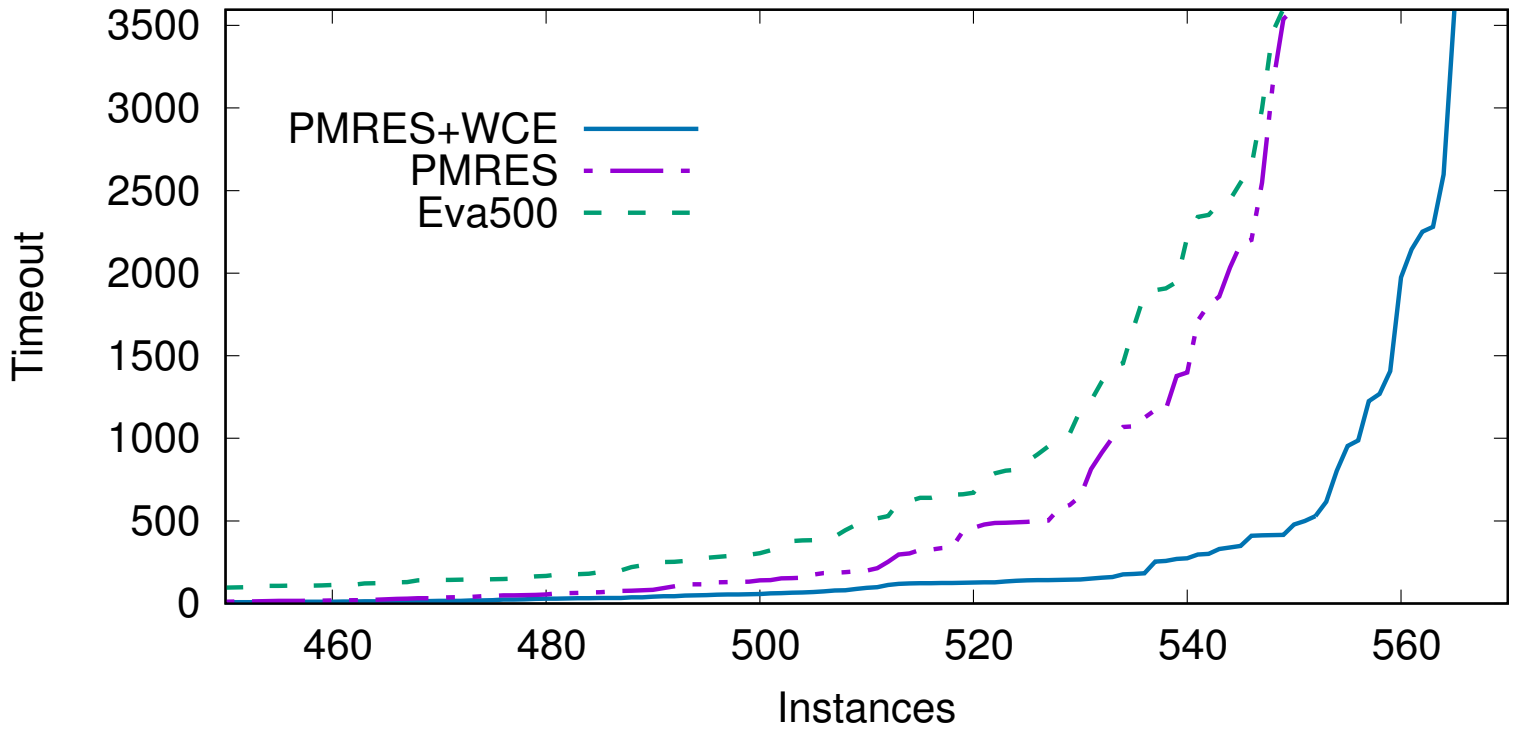
- Simple observation: a hardened clause $C \vee r$ does not appear in future cores.
- Allows implementing clause cloning through assumption variables (and no actual cloning)
- Makes WCE simple to implement.
- PMRES, Primal-Dual, OLL, K, etc. Morgado et al. [2014]; Narodytska and Bacchus [2014]; Bjørner and Narodytska [2015]; Alviano et al. [2015].

Experimental Evaluation

Set up of Experiments

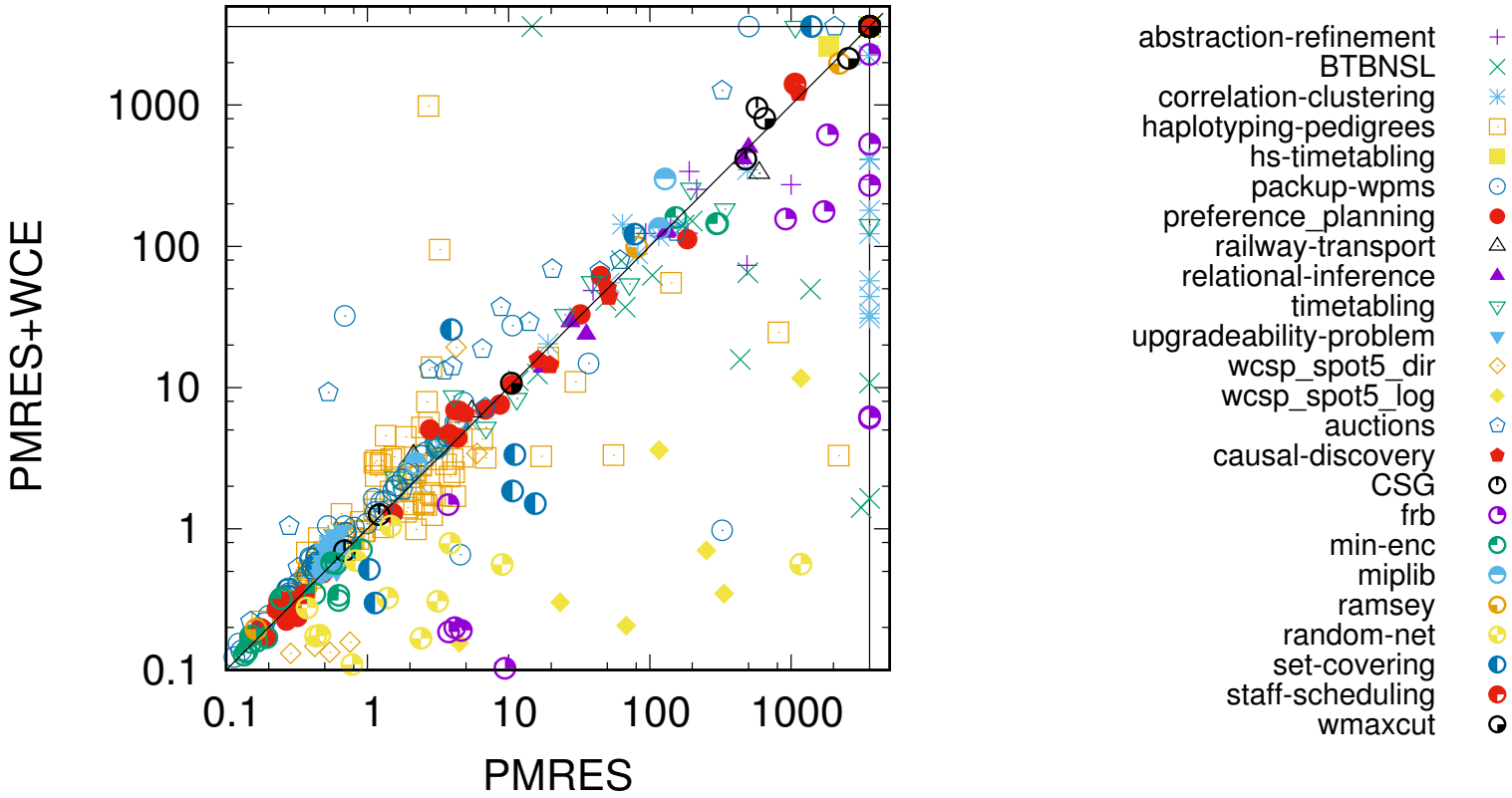
- Reimplemented the PMRES algorithm on top of Open-WBO
  Narodytska and Bacchus [2014]; Martins et al. [2014b]
- Compare to the original implementation, Eva500
- Benchmarks: MaxSAT Evaluation 2016 WPMS industrial (630) and crafted (321) instances.

Experimental Evaluation



Number of instances solved (x-axis) by the individual solvers under different per-instance time limits (y-axis)

Experimental Evaluation



Per-instance running time comparison of PMRES (x-axis) and PMRES+WCE (y-axis)

Comparison to OLL Morgado et al. [2014]

Industrial Domain (#instances)	MSCG15b	PMRES+WCE
abstraction refinement (11)	10 (17096)	10 (**2147**)
BTBNSL (60)	13 (885)	**19** (**679**)
correlation clustering (129)	**33** (19780)	19 (4499)
haplotyping pedigrees (100)	100 (**1343**)	100 (1374)
hs-timetabling (14)	1 (**167**)	1 (2596)
packup-wpms (99)	**99** (410)	94 (191)
preference planning (29)	29 (2021)	29 (**264**)
railway transport (11)	3 (**283**)	3 (340)
relational inference (9)	4 (3167)	**8** (**1360**)
timetabling (26)	12 (**764**)	12 (878)
upgradeability (100)	100 (118)	100 (**59**)
wcsp_spot5_dir (21)	**17** (2776)	14 (24)
wcsp_spot5_log (21)	14 (19)	14 (**17**)

Comparison to OLL Morgado et al. [2014]

Industrial Domain (#instances)	MSCG15b	PMRES+WCE
abstraction refinement (11)	10 (17096)	10 (**2147**)
BTBNSL (60)	13 (885)	**19** (**679**)
correlation clustering (129)	**33** (19780)	19 (4499)
haplotyping pedigrees (100)	100 (**1343**)	100 (1374)
hs-timetabling (14)	1 (**167**)	1 (2596)
packup-wpms (99)	**99** (410)	94 (191)
preference planning (29)	29 (2021)	29 (**264**)
railway transport (11)	3 (**283**)	3 (340)
relational inference (9)	4 (3167)	**8** (**1360**)
timetabling (26)	12 (**764**)	12 (878)
upgradeability (100)	100 (118)	100 (**59**)
wcsp_spot5_dir (21)	**17** (2776)	14 (24)
wcsp_spot5_log (21)	14 (19)	14 (**17**)

Comparison to OLL Morgado et al. [2014]

Industrial Domain (#instances)	MSCG15b	PMRES+WCE
abstraction refinement (11)	10 (17096)	10 (**2147**)
BTBNSL (60)	13 (885)	**19** (**679**)
correlation clustering (129)	**33** (19780)	19 (4499)
haplotyping pedigrees (100)	100 (**1343**)	100 (1374)
hs-timetabling (14)	1 (**167**)	1 (2596)
packup-wpms (99)	**99** (410)	94 (191)
preference planning (29)	29 (2021)	29 (**264**)
railway transport (11)	3 (**283**)	3 (340)
relational inference (9)	4 (3167)	**8** (**1360**)
timetabling (26)	12 (**764**)	12 (878)
upgradeability (100)	100 (118)	100 (**59**)
wcsp_spot5_dir (21)	**17** (2776)	14 (24)
wcsp_spot5_log (21)	14 (19)	14 (**17**)

Summary

- Core Guided MaxSAT shown to be effective on industrial MaxSAT instances.
- Most Core Guided MaxSAT algorithms deal with weights through clause cloning
- We propose WCE, a refinement to clause cloning.
- Prove correctness and discus applicability of WCE on modern MaxSAT algorithms
- Experimentally show improvement on PMRES

Bibliography I

M. Alviano, C. Dodaro, and F. Ricca. A MaxSAT algorithm using cardinality constraints of bounded size. In *Proc. IJCAI*, pages 2677–2683. AAAI Press, 2015.

C. Ansótegui, M.L. Bonet, and J. Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196:77–105, 2013.

Carlos Ansótegui, Frédéric Didier, and Joel Gabàs. Exploiting the structure of unsatisfiable cores in MaxSAT. In *Proc. IJCAI*, pages 283–289. AAAI Press, 2015.

J. Argelich, D. Le Berre, I. Lynce, J.P. Marques-Silva, and P. Rapicault. Solving Linux upgradeability problems using Boolean optimization. In *Proc. LoCoCo*, volume 29 of *Electronic Proceedings in Theoretical Computer Science*, pages 11–22, 2010.

Josep Argelich, Chu Min Li, Felip Manyà, and Jordi Planes. MaxSAT Evaluations. `http://maxsat.ia.udl.cat/`.

J. Berg, M. Järvisalo, and B. Malone. Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In *Proc. AISTATS*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 86–95. JMLR.org, 2014.

J. Berg, P. Saikko, and M. Järvisalo. Improving the effectiveness of SAT-based preprocessing for MaxSAT. In *Proc. IJCAI*, pages 239–245. AAAI Press, 2015.

N. Bjørner and N. Narodytska. Maximum satisfiability using cores and correction sets. In *Proc. IJCAI*, pages 246–252. AAAI Press, 2015.

Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8-9):606–618, 2007.

K. Bunte, M. Järvisalo, J. Berg, P. Myllymäki, J. Peltonen, and S. Kaski. Optimal neighborhood preserving visualization by maximum satisfiability. In *Proc. AAAI*, pages 1694–1700. AAAI Press, 2014.

Y. Chen, S. Safarpour, J. Marques-Silva, and A.G. Veneris. Automated design debugging with maximum satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(11):1804–1817, 2010.

J. Davies and F. Bacchus. Exploiting the power of MIP solvers in MaxSAT. In *Proc. SAT*, volume 7962 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2013.

Z. Fu and S. Malik. On solving the partial MaxSAT problem. In *Proc. SAT*, volume 4121 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2006.

Bibliography II

J. Guerra and I. Lynce. Reasoning over biological networks using maximum satisfiability. In *Proc. CP*, volume 7514 of *Lecture Notes in Computer Science*, pages 941–956. Springer, 2012.

Federico Heras, Antonio Morgado, and Joao Marques-Silva. Lower bounds and upper bounds for MaxSAT. In *Proc. LION*, volume 7219 of *Lecture Notes in Computer Science*, pages 402–407. Springer, 2012.

Chu Min Li, Felip Manyà, and Jordi Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *Proc. AAAI*, pages 86–91. AAAI Press, 2006.

Chu Min Li, Felip Manyà, Nouredine Ould Mohamedou, and Jordi Planes. Resolution-based lower bounds in MaxSAT. *Constraints*, 15(4):456–484, 2010.

Han Lin and Kaile Su. Exploiting inference rules to compute lower bounds for MAX-SAT solving. In *Proc. IJCAI*, pages 2334–2339, 2007.

J. Marques-Silva, M. Janota, A. Ignatiev, and A. Morgado. Efficient model based diagnosis with maximum satisfiability. In *Proc. IJCAI*, pages 1966–1972. AAAI Press, 2015.

R. Martins, S. Joshi, V.M. Manquinho, and I. Lynce. Incremental cardinality constraints for MaxSAT. In *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, pages 531–548. Springer, 2014a.

R. Martins, V.M. Manquinho, and I. Lynce. Open-WBO: A modular MaxSAT solver. In *Proc. SAT*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2014b.

A. Morgado, C. Dodaro, and J. Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2014.

N. Narodytska and F. Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proc. AAAI*, pages 2717–2723. AAAI Press, 2014.

P. Saikko, J. Berg, and M. Järvisalo. LMHS: A SAT-IP hybrid MaxSAT solver. In *Proc. SAT*, volume 9710 of *Lecture Notes in Computer Science*, pages 539–546. Springer, 2016.

Lei Zhang and Fahiem Bacchus. MAXSAT heuristics for cost optimal planning. In *Proc. AAAI*. AAAI Press, 2012.

C.S. Zhu, G. Weissenbacher, and S. Malik. Post-silicon fault localisation using maximum satisfiability and backbones. In *Proc. FMCAD*, pages 63–66. FMCAD Inc., 2011.

Extra Slides