# Inprocessing Rules

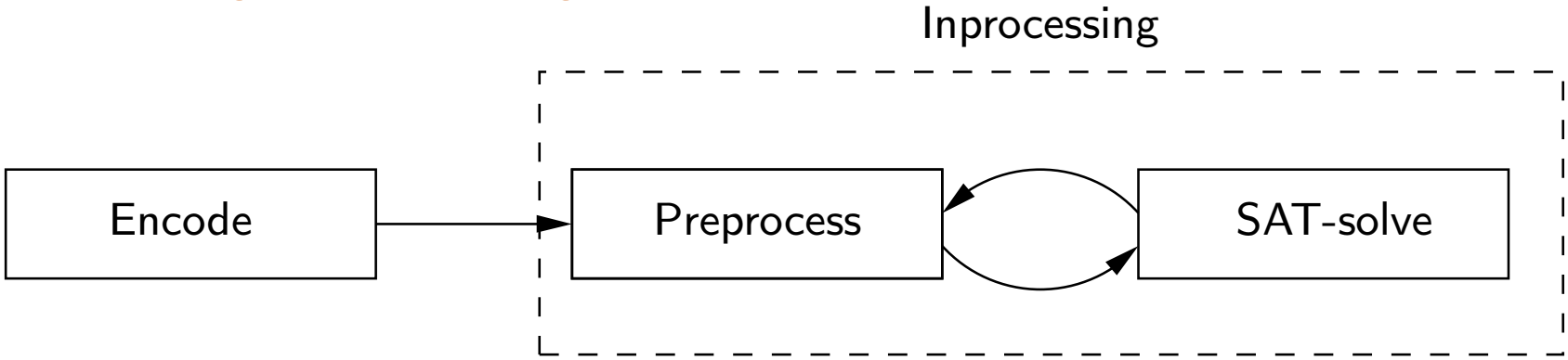
Marijn J.H. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

*Joint work with*
Matti Järvisalo (University of Helsinki)
Armin Biere (Johannes Kepler University, Linz)

January 22, 2014

# Inprocessing: Advantages




- *Exploits already learned clauses dynamically*
- Pushes CDCL solvers further
  - Most state-of-the-art solvers are *inprocessing* solvers: Precosat, Lingeling, Cryptominisat, ...

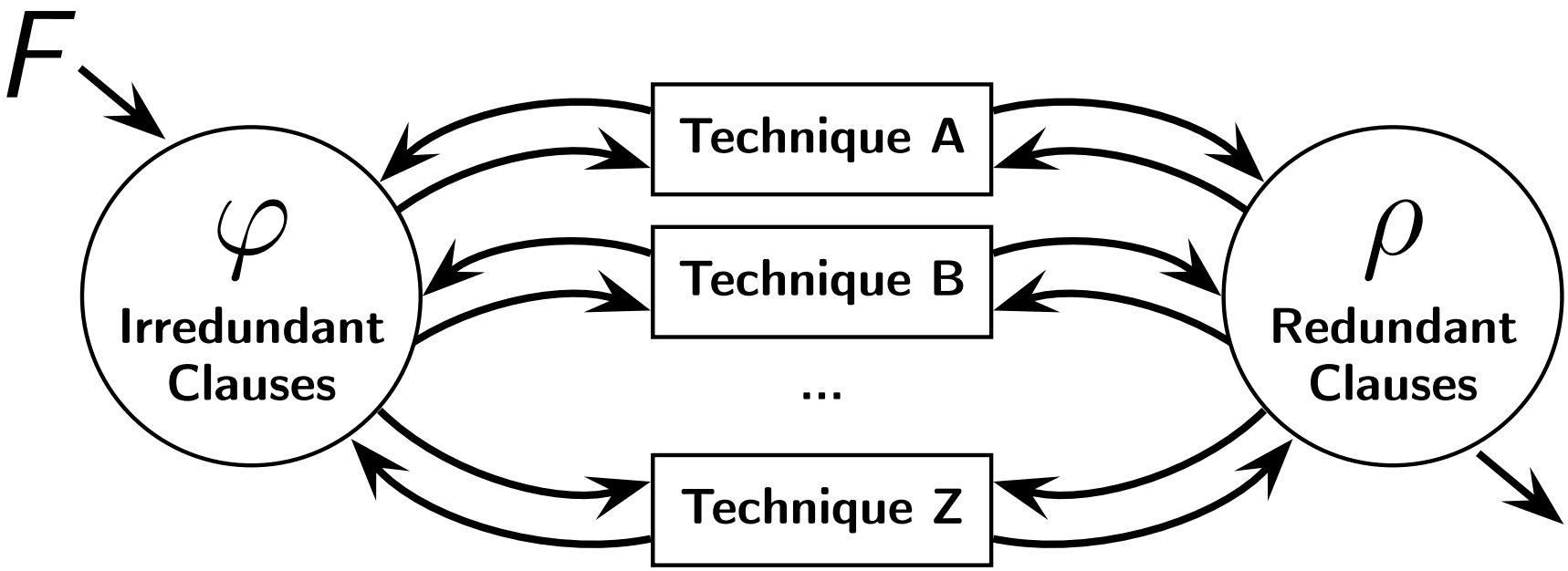
Lingeling ats [Biere, 2013]

SAT Competition 2013 Applications SAT+UNSAT instances

2xx randomly selected instances, 1-h timeout per instance

| **Configuration** | **#solved** | **SAT** | **UNSAT** | flags |
|---|---|---|---|---|
| default | 182 | 90 | 92 | |
| no inprocessing | 158 | 89 | 69 | –inprocessing=0 |
| no pre/inprocessing | 144 | 80 | 64 | –plain=1 |

# Inprocessing: Irredundant versus Redundant




- Main invariant: $\varphi$ is satisfiability-equivalent to $\varphi \wedge \rho$
- Understanding what is correct is nontrivial
- Set of models is not preserved – may grow or shrink
- Model reconstruction is required for many applications

# Inprocessing: Towards a general framework

- **Abstract inprocessing framework**
  - Four elegant deduction rules
  - Characterize inprocessing as a transition system
- **Benefits**:
  - **Correctness of existing and new techniques** can be checked against the generic preconditions of the rules
  - **Model reconstruction captured** by a *simple linear-time algorithm*
- **Covers generally used inprocessing techniques** *even when restricting the rule preconditions on a single polynomial-time computable redundancy property*
- Unifying view to *inprocessing* SAT solving

# Abstract Inprocessing

# Abstract Inprocessing

- Characterize inprocessing solving as a transition system
- State $\varphi \,[\, \rho \,]\, \sigma$
  - $\varphi$: current "irredundant" clauses
  - $\rho$: current "redundant" clauses
  - $\varphi$ and $\varphi \wedge \rho$ are satisfiability-equivalent, $\varphi \models \rho$ is not required

# Abstract Inprocessing

- Characterize inprocessing solving as a transition system
- State $\varphi\,[\,\rho\,]\,\sigma$
  - $\varphi$: current "irredundant" clauses
  - $\rho$: current "redundant" clauses
  - $\varphi$ and $\varphi \wedge \rho$ are satisfiability-equivalent, $\varphi \models \rho$ is not required
  - $\sigma$: sequence of literal-clause pairs $\langle l{:}C \rangle$ for model reconstruction

# Abstract Inprocessing

- Characterize inprocessing solving as a transition system
- State $\varphi\,[\,\rho\,]\,\sigma$
  - $\varphi$: current "irredundant" clauses
  - $\rho$: current "redundant" clauses
  - $\varphi$ and $\varphi \wedge \rho$ are satisfiability-equivalent, $\varphi \models \rho$ is not required
  - $\sigma$: sequence of literal-clause pairs $\langle l{:}C \rangle$ for model reconstruction
- Legal next states $\varphi'\,[\,\rho'\,]\,\sigma'$ of $\varphi\,[\,\rho\,]\,\sigma$ expressed by rules: $\dfrac{\varphi\,[\,\rho\,]\,\sigma}{\varphi'\,[\,\rho'\,]\,\sigma'}$

# The Rules

**Learn** $\dfrac{\varphi\,[\,\rho\,]\,\sigma}{\varphi\,[\,\rho \wedge C\,]\,\sigma}\ \sharp$

**Forget** $\dfrac{\varphi\,[\,\rho \wedge C\,]\,\sigma}{\varphi\,[\,\rho\,]\,\sigma}$

**Strengthen** $\dfrac{\varphi\,[\,\rho \wedge C\,]\,\sigma}{\varphi \wedge C\,[\,\rho\,]\,\sigma}$

**Weaken** $\dfrac{\varphi \wedge C\,[\,\rho\,]\,\sigma}{\varphi\,[\,\rho \wedge C\,]\,\sigma \cup \langle l{:}C \rangle}\ \flat$

**Learn** new redundant clause $C$ to $\rho$.

- Generic precondition $\sharp$:
  $\varphi \wedge \rho$ and $\varphi \wedge \rho \wedge C$ are satisfiability-equivalent.

**Forget** redundant clause $C$ from $\rho$.

**Strengthen** $\varphi$ by making redundant $C$ irredundant

**Weaken** $\varphi$ by making irredundant $C$ redundant

- Generic precondition $\flat$:
  $\varphi$ and $\varphi \wedge C$ are satisfiability-equivalent.

- A sound and complete proof system

# Intuition why Learn has to take redundancy into account

$$\text{Learn} \qquad \frac{\varphi\,[\,\rho\,]\,\sigma}{\varphi\,[\,\rho \wedge C\,]\,\sigma} \;\sharp$$

- Q: Could the precondition $\sharp$ of Learn

  "$\varphi \wedge \rho$ *and* $\varphi \wedge \rho \wedge C$ *are satisfiability-equivalent*"

  be weakened to

  "$\varphi$ *and* $\varphi \wedge C$ *are satisfiability-equivalent*"

  i.e., must the redundant clauses be taken into account for Learn?

# Intuition why Learn has to take redundancy into account

$$\text{Learn} \qquad \frac{\varphi\,[\,\rho\,]\,\sigma}{\varphi\,[\,\rho \wedge C\,]\,\sigma} \;\sharp$$

- Q: Could the precondition $\sharp$ of Learn

  *"$\varphi \wedge \rho$ and $\varphi \wedge \rho \wedge C$ are satisfiability-equivalent"*

  be weakened to

  *"$\varphi$ and $\varphi \wedge C$ are satisfiability-equivalent"*

  i.e., must the redundant clauses be taken into account for Learn?

- A: $\rho$ is essential: ignoring $\rho$ breaks main invariant $\varphi$ sat-eq $\varphi \wedge \rho$
  - Consider $F = (a)$.

  1. Initial state $(a)\,[\emptyset]\,\langle\rangle$
  2. Obtain $\emptyset\,[(a)]\,\langle a{:}(a)\rangle$ through Weaken.
  3. In case $\rho$ were ignored in $\sharp$:
     apply Learn and derive $\emptyset\,[(a) \wedge (\neg a)]\,\langle a{:}(a)\rangle$.

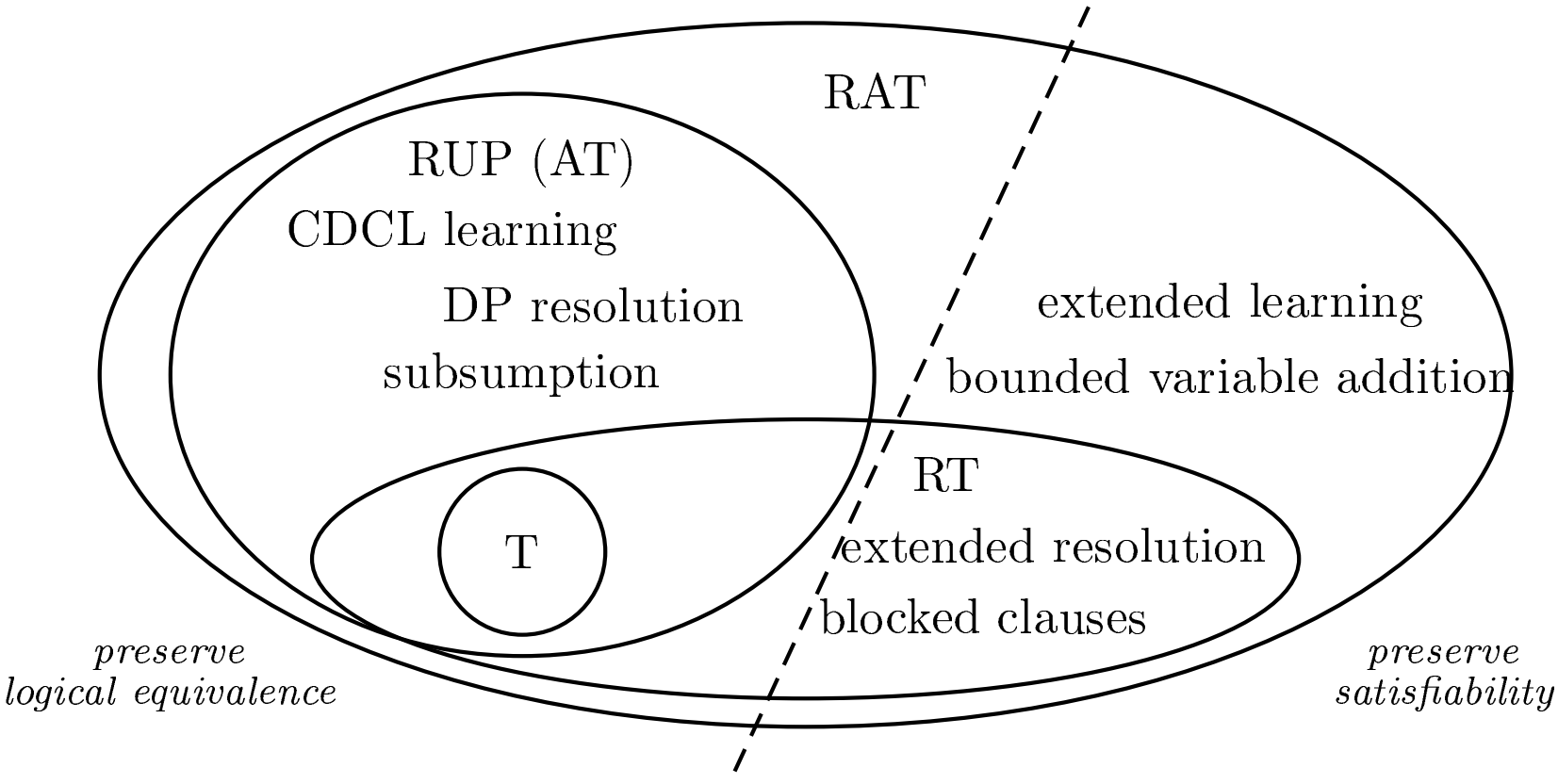
  - Does not preserve satisfiability: $(a) \wedge (\neg a)$ is unsatisfiable.

# Polynomial-Time Computable Redundancy

# Towards Practice: Instantiating the Rules

- The generic preconditions $\sharp$ and $\flat$ for Learn and Weaken are impractical: checking satisfiablity-equivalence is NP-complete
- In practice: procedures are based on polynomial-time computable redundancy properties
- Moreover: a single polynomial-time computable clause redundancy property is enough for a generic system!
  - RAT: *resolution asymmetric tautologies*

# Relationship between Redundancy Properties




All known techniques can be expressed using RAT [IJCAR'12]

# RAT: Resolution Asymmetric Tautologies

- Clause $C$ has AT (Asymmetric Tautology) w.r.t. $F \setminus C$ iff unit propagation derives a conflict in $(F \setminus C) \wedge \neg C$.
  - E.g. $(a \vee b)$ has AT w.r.t. $(a \vee c) \wedge (\neg c \vee \neg d) \wedge (b \vee d)$
  - Tautologies have AT

# RAT: Resolution Asymmetric Tautologies

- Clause $C$ has AT (Asymmetric Tautology) w.r.t. $F \setminus C$ iff unit propagation derives a conflict in $(F \setminus C) \wedge \neg C$.
  - E.g. $(a \vee b)$ has AT w.r.t. $(a \vee c) \wedge (\neg c \vee \neg d) \wedge (b \vee d)$
  - Tautologies have AT

- Clause $C$ *has* RAT (Resolution Asymmetric Tautology) w.r.t. $F \setminus C$ iff
  - there exists a literal $l \in C$ such that
    for each clause $C' \in F$ with $\neg l \in C'$
    clause $(C' \setminus \neg l) \cup C$ has AT w.r.t. $F \setminus C$.
  - E.g. $(a)$ has RAT w.r.t. $(a \vee b) \wedge (\neg a \vee c) \wedge (b \vee c)$
  - Clauses with AT w.r.t. $F$ have RAT w.r.t. $F$

# Capturing Inprocessing Solvers using RAT

Learn $\dfrac{\varphi\,[\,\rho\,]\,\sigma}{\varphi\,[\,\rho \wedge C\,]\,\sigma}$ ♯

Forget $\dfrac{\varphi\,[\,\rho \wedge C\,]\,\sigma}{\varphi\,[\,\rho\,]\,\sigma}$

Strengthen $\dfrac{\varphi\,[\,\rho \wedge C\,]\,\sigma}{\varphi \wedge C\,[\,\rho\,]\,\sigma}$

Weaken $\dfrac{\varphi \wedge C\,[\,\rho\,]\,\sigma}{\varphi\,[\,\rho \wedge C\,]\,\sigma \cup \langle l{:}C \rangle}$ ♭

Polynomial-time computable preconditions:

- ♯: $C$ has RAT w.r.t. $\varphi \wedge \rho$.
- ♭: $C$ has RAT (on $l$) w.r.t. $\varphi$.

# Capturing Inprocessing Solvers using RAT

Learn $\dfrac{\varphi\,[\,\rho\,]\,\sigma}{\varphi\,[\,\rho \wedge C\,]\,\sigma}$ ♯

Forget $\dfrac{\varphi\,[\,\rho \wedge C\,]\,\sigma}{\varphi\,[\,\rho\,]\,\sigma}$

Strengthen $\dfrac{\varphi\,[\,\rho \wedge C\,]\,\sigma}{\varphi \wedge C\,[\,\rho\,]\,\sigma}$

Weaken $\dfrac{\varphi \wedge C\,[\,\rho\,]\,\sigma}{\varphi\,[\,\rho \wedge C\,]\,\sigma \cup \langle l{:}C \rangle}$ ♭

Polynomial-time computable preconditions:

♯: $C$ has RAT w.r.t. $\varphi \wedge \rho$.

♭: $C$ has RAT (on $l$) w.r.t. $\varphi$.

- Simulates generally used inprocessing techniques
  - Pure literal elimination, clause elimination (including subsumption, blocked clause elimination, . . . ), clause addition, variable elimination, hyper-binary resolution, self-subsuming resolution, equivalent literal reasoning, hidden literal elimination, clause learning, extended resolution, . . .
- Has a unifying linear-time model reconstruction algorithm *covering all these techniques*

# Examples

# Example of incorrect clause elimination

Idea: eliminate $C$ if it is redundant w.r.t. $\varphi \wedge \rho$.

- This would allow using redundant learned clauses in $\rho$, which can later be forgotten, for weakening $\varphi$.

# Example of incorrect clause elimination

Idea: eliminate $C$ if it is redundant w.r.t. $\varphi \wedge \rho$.

- This would allow using redundant learned clauses in $\rho$, which can later be forgotten, for weakening $\varphi$.

Bad Idea:

- Consider $\rho_0 = \emptyset$ and the minimally unsatisfiable formula $\varphi_0 = (a \vee \neg b) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee b \vee c) \wedge (a \vee b \vee \neg c)$

# Example of incorrect clause elimination

Idea: eliminate $C$ if it is redundant w.r.t. $\varphi \wedge \rho$.

- This would allow using redundant learned clauses in $\rho$, which can later be forgotten, for weakening $\varphi$.

Bad Idea:

- Consider $\rho_0 = \emptyset$ and the minimally unsatisfiable formula $\varphi_0 = (a \vee \neg b) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee b \vee c) \wedge (a \vee b \vee \neg c)$
- The clause $(a \vee b)$ has $\mathrm{AT}$ w.r.t. $\varphi_0$

# Example of incorrect clause elimination

Idea: eliminate $C$ if it is redundant w.r.t. $\varphi \wedge \rho$.

- This would allow using redundant learned clauses in $\rho$, which can later be forgotten, for weakening $\varphi$.

Bad Idea:

- Consider $\rho_0 = \emptyset$ and the minimally unsatisfiable formula $\varphi_0 = (a \vee \neg b) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee b \vee c) \wedge (a \vee b \vee \neg c)$
- The clause $(a \vee b)$ has $\mathrm{AT}$ w.r.t. $\varphi_0$
- Applying Learn gives $\varphi_1 = \varphi_0$ and $\rho_1 = (a \vee b)$.
- $(a \vee b) \in \rho_1$ subsumes $(a \vee b \vee c) \in \varphi_1$

# Example of incorrect clause elimination

Idea: eliminate $C$ if it is redundant w.r.t. $\varphi \wedge \rho$.

- This would allow using redundant learned clauses in $\rho$, which can later be forgotten, for weakening $\varphi$.

Bad Idea:

- Consider $\rho_0 = \emptyset$ and the minimally unsatisfiable formula $\varphi_0 = (a \vee \neg b) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee b \vee c) \wedge (a \vee b \vee \neg c)$
- The clause $(a \vee b)$ has $\mathrm{AT}$ w.r.t. $\varphi_0$
- Applying Learn gives $\varphi_1 = \varphi_0$ and $\rho_1 = (a \vee b)$.
- $(a \vee b) \in \rho_1$ subsumes $(a \vee b \vee c) \in \varphi_1$
- Weaken would give $\varphi_2 = \varphi_1 \setminus (a \vee b \vee c)$
- However, $\varphi_2$ is satisfiable

# Example of incorrect clause elimination

Idea: eliminate $C$ if it is redundant w.r.t. $\varphi \wedge \rho$.

- This would allow using redundant learned clauses in $\rho$, which can later be forgotten, for weakening $\varphi$.

Bad Idea:

- Consider $\rho_0 = \emptyset$ and the minimally unsatisfiable formula $\varphi_0 = (a \vee \neg b) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee b \vee c) \wedge (a \vee b \vee \neg c)$
- The clause $(a \vee b)$ has $\mathrm{AT}$ w.r.t. $\varphi_0$
- Applying Learn gives $\varphi_1 = \varphi_0$ and $\rho_1 = (a \vee b)$.
- $(a \vee b) \in \rho_1$ subsumes $(a \vee b \vee c) \in \varphi_1$
- Weaken would give $\varphi_2 = \varphi_1 \setminus (a \vee b \vee c)$
- However, $\varphi_2$ is satisfiable

Fixed Idea:

*The clauses in $\rho$ cannot be used to eliminate clauses in $\varphi$*

- First move the desired clauses from $\rho$ to $\varphi$ (Strengthen)

# Examples: Simulating Resolution and More

## Resolution and Clause Learning

- For any $\varphi$, $(C \vee D)$ is an $\mathrm{AT}$ w.r.t. $\varphi \wedge (C \vee x) \wedge (D \vee \neg x)$
- Thus $(C \vee D)$ can be learned by applying Learn.

$\Rightarrow$ Covers resolution-based techniques such as hyper-binary resolution

# Examples: Simulating Resolution and More

## Resolution and Clause Learning

- For any $\varphi$, $(C \vee D)$ is an $\mathrm{AT}$ w.r.t. $\varphi \wedge (C \vee x) \wedge (D \vee \neg x)$
- Thus $(C \vee D)$ can be learned by applying Learn.

$\Rightarrow$ Covers resolution-based techniques such as hyper-binary resolution

## Extended resolution

- Extension rule: Introduce fresh definitions of the form $x \equiv a \wedge b$ i.e. the CNF formula $(x \vee \neg a \vee \neg b) \wedge (\neg x \vee a) \wedge (\neg x \vee b)$
- Simulation:
  1. $(x \vee \neg a \vee \neg b)$ has $\mathrm{RAT}$ on $x$ w.r.t. $\varphi \wedge \rho$ (Learn);
  2. $(\neg x \vee a)$ and $(\neg x \vee b)$ have $\mathrm{RAT}$ on $\neg x$ w.r.t. $\varphi \wedge (x \vee \neg a \vee \neg b) \wedge \rho$ (Learn)

# Examples: Simulating Resolution and More

## Resolution and Clause Learning

- For any $\varphi$, $(C \vee D)$ is an $\mathrm{AT}$ w.r.t. $\varphi \wedge (C \vee x) \wedge (D \vee \neg x)$
- Thus $(C \vee D)$ can be learned by applying Learn.

$\Rightarrow$ Covers resolution-based techniques such as hyper-binary resolution

## Extended resolution

- Extension rule: Introduce fresh definitions of the form $x \equiv a \wedge b$
  i.e. the CNF formula $(x \vee \neg a \vee \neg b) \wedge (\neg x \vee a) \wedge (\neg x \vee b)$
- Simulation:
  1. $(x \vee \neg a \vee \neg b)$ has $\mathrm{RAT}$ on $x$ w.r.t. $\varphi \wedge \rho$ (Learn);
  2. $(\neg x \vee a)$ and $(\neg x \vee b)$ have $\mathrm{RAT}$ on $\neg x$ w.r.t. $\varphi \wedge (x \vee \neg a \vee \neg b) \wedge \rho$ (Learn)

## Bounded Variable Elimination

- Perhaps the most important SAT preprocessing technique
- Generate all resolvents w.r.t. variable $x$, then forget all antecedents
- Simulation:
  1. Learn and Strengthen resolvents; 2. Weaken and Forget antecedents

# Model Reconstruction

# Model Reconstruction

- Weaken may introduce new models

$$\text{Weaken} \quad \frac{\varphi \wedge C\,[\,\rho\,]\,\sigma}{\varphi\,[\,\rho \wedge C\,]\,\sigma \cup \langle l{:}C \rangle} \; \flat$$

Given a model $\tau$ for the current $\varphi$:

```
1   while σ is not empty do
2     remove the last literal-clause pair ⟨l:C⟩ from σ
3     if C is not satisfied by τ then τ := (τ \ {l = 0}) ∪ {l = 1}
4   return τ
```

# Conclusions

# Conclusion

## Inprocessing SAT solving

- Interleaving preprocessing and CDCL search
- Improves efficiency of CDCL solvers
- Correctness issues are nontrivial:
  Understanding how the individual preprocessing and search techniques work together

## Abstract Inprocessing Framework

- Captures generally used inprocessing and CDCL techniques
- Check individual techniques for correctness against the inprocessing rules
- Yields a generic and simple model reconstruction algorithm
- A basis for developing novel inprocessing techniques