A Unifying Principle for Clause Elimination in First-Order Logic

Benjamin Kiesl Martin Suda

Institute for Logic and Computation, TU Wien

Topic of the Talk

- Preprocessing techniques for first-order theorem provers.
  - Improve the efficiency of provers by simplifying the input.

Topic of the Talk

- Preprocessing techniques for first-order theorem provers.
  - Improve the efficiency of provers by simplifying the input.
- In particular, clause-elimination techniques:
  - Remove redundant clauses from a formula in CNF.

Topic of the Talk

- Preprocessing techniques for first-order theorem provers.
  - Improve the efficiency of provers by simplifying the input.
- In particular, clause-elimination techniques:
  - Remove redundant clauses from a formula in CNF.
- Many clause-elimination techniques are used in SAT solving but not in first-order logic yet.

Topic of the Talk

- Preprocessing techniques for first-order theorem provers.
  - Improve the efficiency of provers by simplifying the input.
- In particular, clause-elimination techniques:
  - Remove redundant clauses from a formula in CNF.
- Many clause-elimination techniques are used in SAT solving but not in first-order logic yet.
- We lifted SAT techniques to first-order logic without equality.

Topic of the Talk

- Preprocessing techniques for first-order theorem provers.
  - Improve the efficiency of provers by simplifying the input.
- In particular, clause-elimination techniques:
  - Remove redundant clauses from a formula in CNF.
- Many clause-elimination techniques are used in SAT solving but not in first-order logic yet.
- We lifted SAT techniques to first-order logic without equality.
  - We proved correctness in a uniform way by introducing the principle of implication modulo resolution.

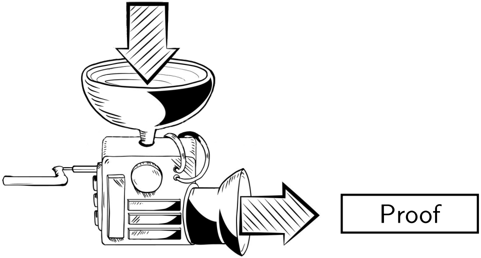
Outline

- First-order theorem proving and preprocessing in a nutshell.
- Details on one successful approach for preprocessing:
  - Clause-elimination techniques.
- Overview of techniques we lifted.
- The unifying principle of implication modulo resolution.
- Confluence results.
- Future work.

First-Order Theorem Proving

- Input: Formula in first-order logic.
- Output: Proof

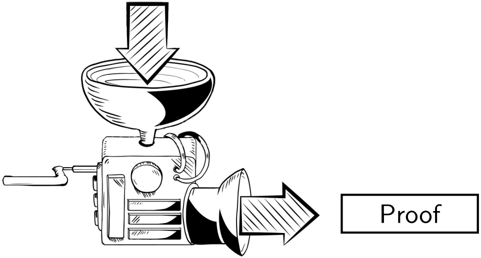
$$Q(a, b) \land ((\forall x \forall y P(x, y) \leftrightarrow P(y, x)) \rightarrow (\neg P(a, b) \lor P(b, a)))$$



First-Order Theorem Proving

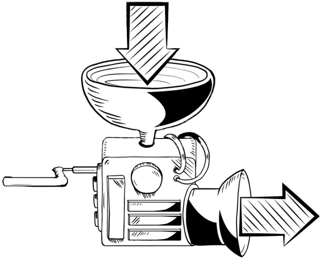
- Input: Formula in first-order logic.
- Output: Proof
- Applications: Mathematics, verification of software and hardware, reasoning over knowledge bases, etc.

$$Q(a,b) \wedge ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \vee P(b,a)))$$



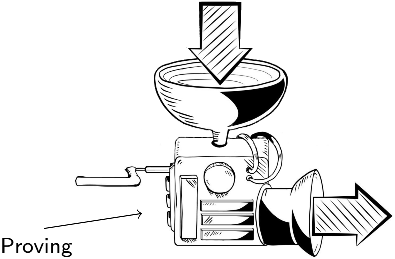
Automatic First-Order Theorem Proving

$$Q(a, b) \wedge ((\forall x \forall y P(x, y) \leftrightarrow P(y, x)) \rightarrow (\neg P(a, b) \vee P(b, a)))$$



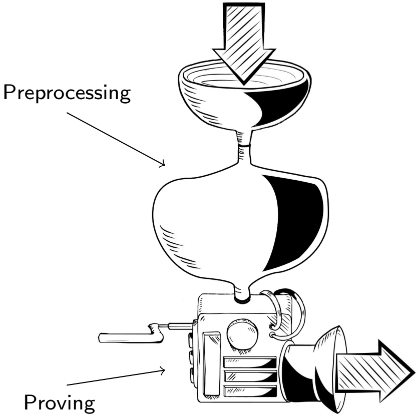
Automatic First-Order Theorem Proving

$$Q(a, b) \wedge ((\forall x \forall y P(x, y) \leftrightarrow P(y, x)) \rightarrow (\neg P(a, b) \vee P(b, a)))$$



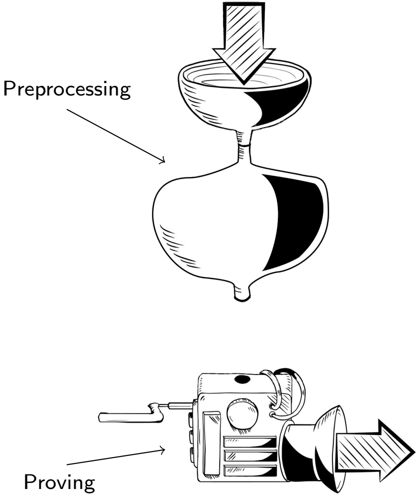
Automatic First-Order Theorem Proving

$$Q(a,b) \wedge ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \vee P(b,a)))$$



Automatic First-Order Theorem Proving

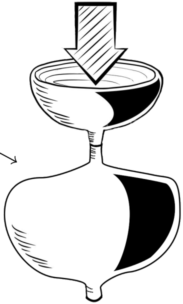
$$Q(a, b) \wedge ((\forall x \forall y P(x, y) \leftrightarrow P(y, x)) \rightarrow (\neg P(a, b) \vee P(b, a)))$$



Automatic First-Order Theorem Proving

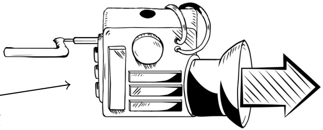
$$Q(a,b) \wedge ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \vee P(b,a)))$$

Preprocessing



Simplified Formula
in CNF (negated)

$$(\neg P(x,y) \vee P(y,x)) \wedge P(a,b) \wedge \neg P(b,a)$$



Proving

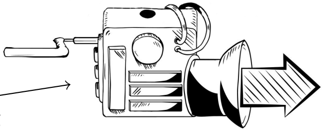
Automatic First-Order Theorem Proving

$$Q(a,b) \wedge ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \vee P(b,a)))$$

Preprocessing

Simplified Formula
in CNF (negated)

$$(\neg P(x,y) \vee P(y,x)) \wedge P(a,b) \wedge \neg P(b,a)$$



Proving

Refutation

Resolution Refutations (Propositional Logic)

- Resolution Rule: Derive $C \vee D$ from $C \vee L$ and $\neg L \vee D$:

$$\frac{C \vee L \qquad \neg L \vee D}{C \vee D}$$

Resolution Refutations (Propositional Logic)

- Resolution Rule: Derive $C \vee D$ from $C \vee L$ and $\neg L \vee D$:

$$\frac{C \vee L \qquad \neg L \vee D}{C \vee D}$$

➥ $C \vee D$ is a resolvent of $C \vee L$ upon L.

Resolution Refutations (Propositional Logic)

- Resolution Rule: Derive $C \vee D$ from $C \vee L$ and $\neg L \vee D$:

$$\frac{C \vee L \qquad \neg L \vee D}{C \vee D}$$

➥ $C \vee D$ is a resolvent of $C \vee L$ upon L.

- Every unsatisfiable formula can be refuted by resolution.

Resolution Refutations (Propositional Logic)

- Resolution Rule: Derive $C \vee D$ from $C \vee L$ and $\neg L \vee D$:

$$\frac{C \vee L \qquad \neg L \vee D}{C \vee D}$$

➥ $C \vee D$ is a resolvent of $C \vee L$ upon L.

- Every unsatisfiable formula can be refuted by resolution.
- Example: $F = (\neg P \vee Q) \wedge (P) \wedge (\neg Q)$

$$\frac{\dfrac{\neg P \vee Q \qquad P}{Q} \qquad \neg Q}{\perp}$$

Resolution Refutations (First-Order Logic)

- Resolution Rule: Derive $(C \vee D)\sigma$ from $C \vee L(t_1, \ldots, t_n)$ and $\neg L(s_1, \ldots, s_n) \vee D$ if σ unifies $L(t_1, \ldots, t_n)$ and $L(s_1, \ldots, s_n)$:

Resolution Refutations (First-Order Logic)

- Resolution Rule: Derive $(C \vee D)\sigma$ from $C \vee L(t_1, \ldots, t_n)$ and $\neg L(s_1, \ldots, s_n) \vee D$ if σ unifies $L(t_1, \ldots, t_n)$ and $L(s_1, \ldots, s_n)$:
- Intuitively, a mapping σ unifies literals if it makes them equal:
  - $P(x, y)$ and $P(a, b)$ are unifiable $\rightarrow \sigma(x) = a$ and $\sigma(y) = b$.

Resolution Refutations (First-Order Logic)

- Resolution Rule: Derive $(C \vee D)\sigma$ from $C \vee L(t_1, \ldots, t_n)$ and $\neg L(s_1, \ldots, s_n) \vee D$ if σ unifies $L(t_1, \ldots, t_n)$ and $L(s_1, \ldots, s_n)$:
- Intuitively, a mapping σ unifies literals if it makes them equal:
  - $P(x, y)$ and $P(a, b)$ are unifiable $\rightarrow$ $\sigma(x) = a$ and $\sigma(y) = b$.
  - $P(b, a)$ and $P(b, a)$ are unifiable $\rightarrow$ no mapping necessary.

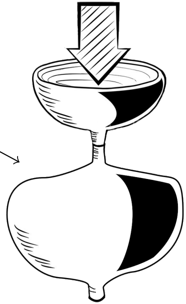
Resolution Refutations (First-Order Logic)

- Resolution Rule: Derive $(C \vee D)\sigma$ from $C \vee L(t_1, \ldots, t_n)$ and $\neg L(s_1, \ldots, s_n) \vee D$ if σ unifies $L(t_1, \ldots, t_n)$ and $L(s_1, \ldots, s_n)$:
- Intuitively, a mapping σ unifies literals if it makes them equal:
  - $P(x, y)$ and $P(a, b)$ are unifiable $\rightarrow$ $\sigma(x) = a$ and $\sigma(y) = b$.
  - $P(b, a)$ and $P(b, a)$ are unifiable $\rightarrow$ no mapping necessary.
- Example Refutation:
  $F = (\neg P(x, y) \vee P(y, x)) \wedge P(a, b) \wedge \neg P(b, a)$

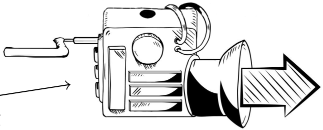
$$\dfrac{\dfrac{\neg P(x,y) \vee P(y,x) \qquad P(a,b)}{P(b,a)} \qquad \neg P(b,a)}{\bot}$$

Automatic First-Order Theorem Proving

$$Q(a,b) \land ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \to (\neg P(a,b) \lor P(b,a)))$$



$$(\neg P(x,y) \lor P(y,x)) \land P(a,b) \land \neg P(b,a)$$

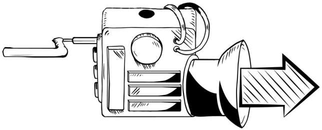


Automatic First-Order Theorem Proving

$$Q(a, b) \land ((\forall x \forall y P(x, y) \leftrightarrow P(y, x)) \rightarrow (\neg P(a, b) \lor P(b, a)))$$

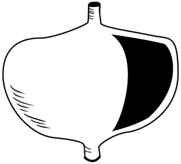
What's going on here?

$$(\neg P(x, y) \lor P(y, x)) \land P(a, b) \land \neg P(b, a)$$



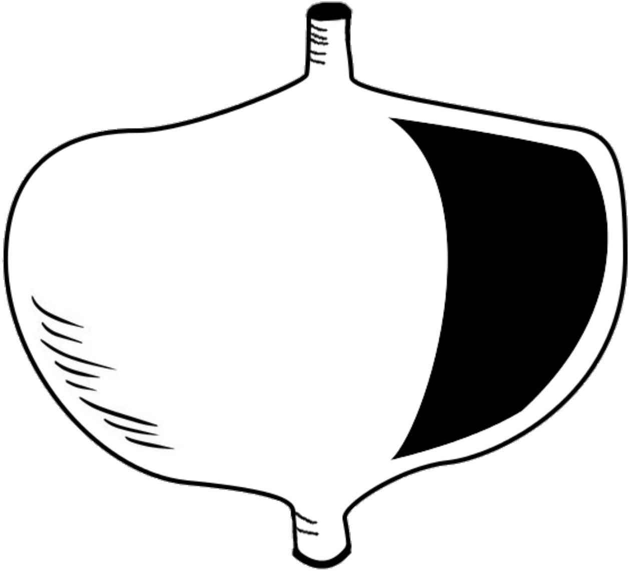
Resolution Refutation

Automatic First-Order Theorem Proving

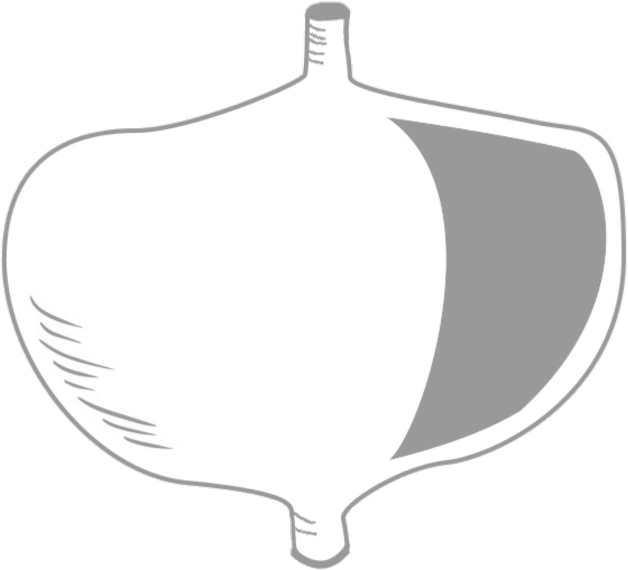


What's going on here?

Preprocessing Pipeline

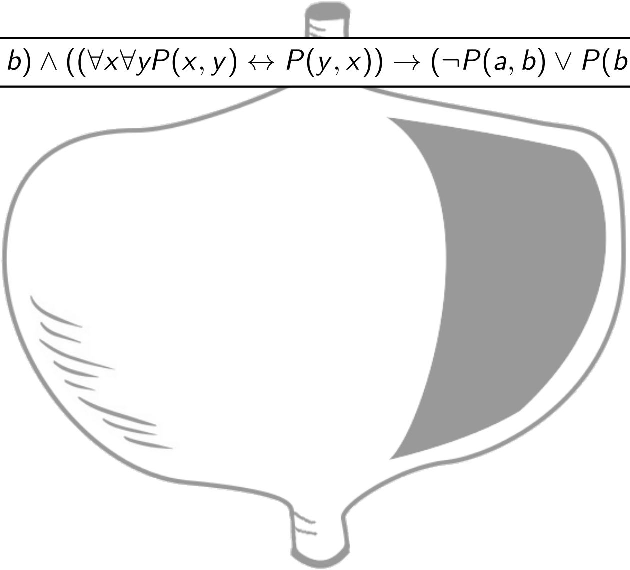


Preprocessing Pipeline



Preprocessing Pipeline

$$\boxed{Q(a,b) \wedge ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \vee P(b,a)))}$$

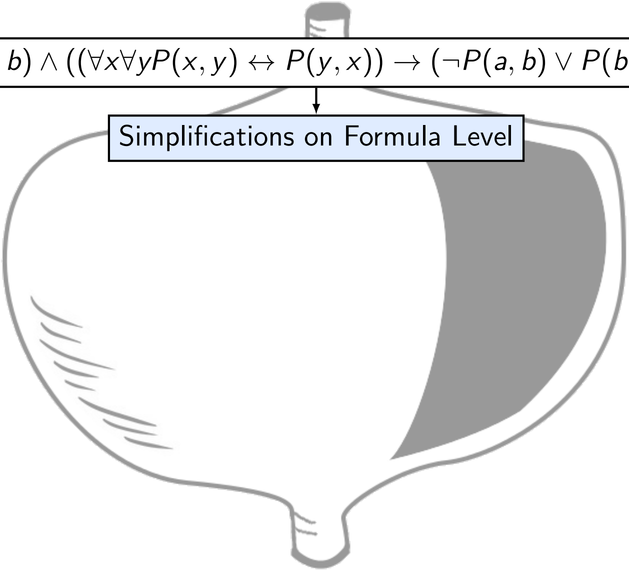


Preprocessing Pipeline

$$Q(a,b) \land ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \lor P(b,a)))$$

$\downarrow$

Simplifications on Formula Level



Preprocessing Pipeline

$$Q(a,b) \land ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \lor P(b,a)))$$

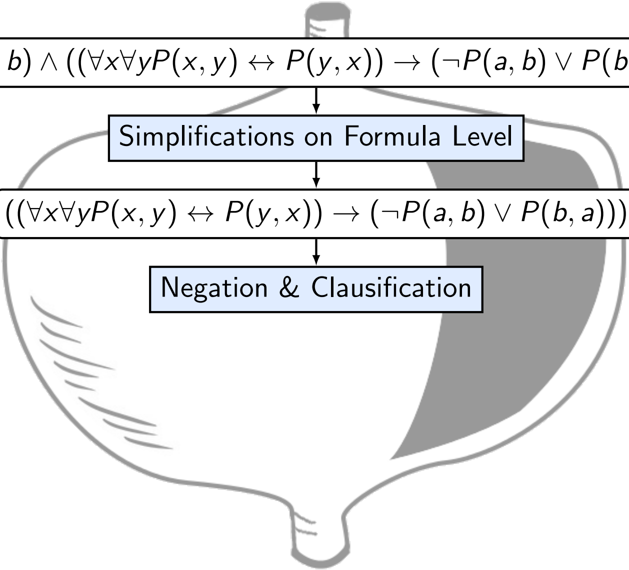
$$\downarrow$$

Simplifications on Formula Level

$$\downarrow$$

$$((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \lor P(b,a)))$$

Preprocessing Pipeline



$Q(a,b) \land ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \lor P(b,a)))$

↓

Simplifications on Formula Level

↓

$((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \lor P(b,a)))$

↓

Negation & Clausification

Preprocessing Pipeline

$$Q(a,b) \wedge ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \vee P(b,a)))$$

↓

Simplifications on Formula Level

↓

$$((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \vee P(b,a)))$$

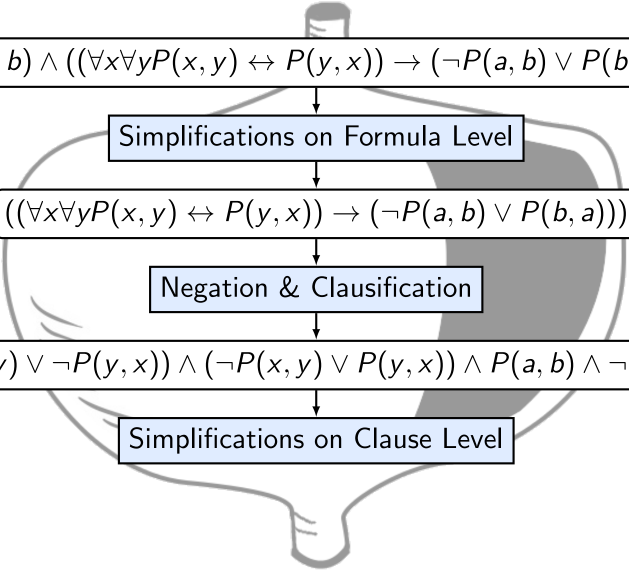
↓

Negation & Clausification

↓

$$(P(x,y) \vee \neg P(y,x)) \wedge (\neg P(x,y) \vee P(y,x)) \wedge P(a,b) \wedge \neg P(b,a)$$

Preprocessing Pipeline



$$Q(a,b) \land ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \lor P(b,a)))$$

$\downarrow$

Simplifications on Formula Level

$\downarrow$

$$((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \lor P(b,a)))$$

$\downarrow$

Negation & Clausification

$\downarrow$

$$(P(x,y) \lor \neg P(y,x)) \land (\neg P(x,y) \lor P(y,x)) \land P(a,b) \land \neg P(b,a)$$

$\downarrow$

Simplifications on Clause Level

Preprocessing Pipeline

$$Q(a,b) \wedge ((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \vee P(b,a)))$$

↓

Simplifications on Formula Level

↓

$$((\forall x \forall y P(x,y) \leftrightarrow P(y,x)) \rightarrow (\neg P(a,b) \vee P(b,a)))$$

↓

Negation & Clausification

↓

$$(P(x,y) \vee \neg P(y,x)) \wedge (\neg P(x,y) \vee P(y,x)) \wedge P(a,b) \wedge \neg P(b,a)$$

↓

Simplifications on Clause Level

↓

$$(\neg P(x,y) \vee P(y,x)) \wedge P(a,b) \wedge \neg P(b,a)$$

Preprocessing Pipeline

- Topic of this talk: Simplifications on the clause level.

$$\boxed{(P(x,y) \vee \neg P(y,x)) \wedge (\neg P(x,y) \vee P(y,x)) \wedge P(a,b) \wedge \neg P(b,a)}$$

$$\downarrow$$

$$\boxed{\text{Simplifications on Clause Level}}$$

$$\downarrow$$

$$\boxed{(\neg P(x,y) \vee P(y,x)) \wedge P(a,b) \wedge \neg P(b,a)}$$

Clause-Elimination Techniques in Theory

- Clause-elimination techniques remove redundant clauses.

Clause-Elimination Techniques in Theory

- Clause-elimination techniques remove redundant clauses.
- A clause is redundant if its removal preserves unsatisfiability.

Clause-Elimination Techniques in Theory

- Clause-elimination techniques remove redundant clauses.
- A clause is redundant if its removal preserves unsatisfiability.
  - ➥ If we can refute the formula before removing the clause, we can still refute it afterwards.

Clause-Elimination Techniques in Theory

- Clause-elimination techniques remove redundant clauses.
- A clause is redundant if its removal preserves unsatisfiability.
  - ➥ If we can refute the formula before removing the clause, we can still refute it afterwards.

Definition

A clause C is redundant with respect to a formula F if F and $F \setminus \{C\}$ are equisatisfiable.

Clause-Elimination Techniques in Theory

- Clause-elimination techniques remove redundant clauses.
- A clause is redundant if its removal preserves unsatisfiability.
  - ➥ If we can refute the formula before removing the clause, we can still refute it afterwards.

Definition

A clause C is redundant with respect to a formula F if F and $F \setminus \{C\}$ are equisatisfiable.

- Remark: Redundant clauses need not be implied!

Clause-Elimination Techniques in Practice

- Problem: Checking if a clause is redundant is undecidable.

Clause-Elimination Techniques in Practice

- Problem: Checking if a clause is redundant is undecidable.

➥ Define efficiently decidable criteria that ensure redundancy.

Clause-Elimination Techniques in Practice

- Problem: Checking if a clause is redundant is undecidable.

➥ Define efficiently decidable criteria that ensure redundancy.

- Examples:

Clause-Elimination Techniques in Practice

- Problem: Checking if a clause is redundant is undecidable.

➥ Define efficiently decidable criteria that ensure redundancy.

- Examples: A clause C is redundant if ...

Clause-Elimination Techniques in Practice

- Problem: Checking if a clause is redundant is undecidable.

➥ Define efficiently decidable criteria that ensure redundancy.

- Examples: A clause C is redundant if ...
  - ... it contains two complementary literals L and $\neg L$. (Tautology)

Clause-Elimination Techniques in Practice

- Problem: Checking if a clause is redundant is undecidable.

➥ Define efficiently decidable criteria that ensure redundancy.

- Examples: A clause C is redundant if ...
  - ... it contains two complementary literals L and $\neg L$. (Tautology)
  - ... all resolvents upon one of its literals are tautologies. (Blocked clause)

Clause-Elimination Techniques in Practice

- Problem: Checking if a clause is redundant is undecidable.

➥ Define efficiently decidable criteria that ensure redundancy.

- Examples: A clause C is redundant if ...
  - ...it contains two complementary literals L and $\neg L$. (Tautology)
  - ...all resolvents upon one of its literals are tautologies. (Blocked clause)
  - ...there exist another clause D and a substitution λ such that $D\lambda \subseteq C$. (Subsumed clause)

Clause-Elimination Techniques in Practice

- Problem: Checking if a clause is redundant is undecidable.

➥ Define efficiently decidable criteria that ensure redundancy.

- Examples: A clause C is redundant if ...
  - ... it contains two complementary literals L and $\neg L$. (Tautology)
  - ... all resolvents upon one of its literals are tautologies. (Blocked clause)
  - ... there exist another clause D and a substitution λ such that $D\lambda \subseteq C$. (Subsumed clause)
  - ...

Clause-Elimination Techniques: Success Stories

- Clause-elimination is successfully used in SAT and QSAT solving:
  - *Effective Preprocessing in SAT Through Variable and Clause Elimination* (Eén and Biere, SAT, 2005)
  - *Clause Elimination for SAT and QSAT* (Heule et al., JAIR, 2010)
  - *Covered Clause Elimination* (Heule et al., LPAR, 2010)
  - *Blocked Clause Elimination* (Järvisalo et al., TACAS, 2010)
  - *Enhancing Search-Based QBF solving by Dynamic Blocked Clause Elimination* (Lonsing et al., LPAR, 2015)
  - ...

Clause-Elimination Techniques: Success Stories

- Clause-elimination is successfully used in SAT and QSAT solving:
  - *Effective Preprocessing in SAT Through Variable and Clause Elimination* (Eén and Biere, SAT, 2005)
  - *Clause Elimination for SAT and QSAT* (Heule et al., JAIR, 2010)
  - *Covered Clause Elimination* (Heule et al., LPAR, 2010)
  - *Blocked Clause Elimination* (Järvisalo et al., TACAS, 2010)
  - *Enhancing Search-Based QBF solving by Dynamic Blocked Clause Elimination* (Lonsing et al., LPAR, 2015)
  - ...
- Blocked-clause elimination can speed up first-order provers:
  - *Blocked Clauses in First-Order Logic* (Kiesl, Suda, Seidl, Tompits, and Biere, LPAR, 2017)

(Some) Types of Redundant Clauses in SAT Solving

Asymmetric Tautologies

Covered Clauses

Resolution Asymmetric Tautologies

Resolution Subsumed Clauses

Blocked Clauses

Tautologies

Asymmetric Blocked Clauses

Subsumed Clauses

Asymmetric Covered Clauses

(Some) Types of Redundant Clauses in SAT Solving

Asymmetric Tautologies

Covered Clauses

Resolution Asymmetric Tautologies

Resolution Subsumed Clauses

Asymmetric Blocked Clauses

Asymmetric Covered Clauses

- Not available in first-order logic before!

(Some) Types of Redundant Clauses in SAT Solving

Asymmetric Tautologies

Covered Clauses

Resolution Asymmetric Tautologies

Resolution Subsumed Clauses

Asymmetric Blocked Clauses

Asymmetric Covered Clauses

- Not available in first-order logic before!
  - ➥ We lifted them.

Example: Blocked Clauses in Propositional Logic

- A clause C is blocked in a formula F if all resolvents upon one of its literals are tautologies.

$$\neg S \vee P \vee Q$$

$$P \vee Q \vee R \qquad \neg R \vee \neg Q$$

$$\neg R \vee \neg P$$

$$\neg T \vee S \vee Q$$

Example: Blocked Clauses in Propositional Logic

- A clause C is blocked in a formula F if all resolvents upon one of its literals are tautologies.

$\neg S \vee P \vee Q$

$\neg R \vee \neg Q$

$P \vee Q \vee R$

$\neg R \vee \neg P$

$\neg T \vee S \vee Q$

$P \vee Q \vee \neg Q$

Example: Blocked Clauses in Propositional Logic

- A clause C is blocked in a formula F if all resolvents upon one of its literals are tautologies.

$\neg S \vee P \vee Q$

$\neg R \vee \neg Q$

$P \vee Q \vee R$

$\neg R \vee \neg P$

$\neg T \vee S \vee Q$

$P \vee Q \vee \neg Q$ $\qquad$ $P \vee Q \vee \neg P$

Example: Blocked Clauses in Propositional Logic

- A clause C is blocked in a formula F if all resolvents upon one of its literals are tautologies.

$$P \lor Q \lor R$$

$$\neg S \lor P \lor Q$$

$$\neg R \lor \neg Q$$

$$\neg R \lor \neg P$$

$$\neg T \lor S \lor Q$$

$$P \lor Q \lor \neg Q \qquad P \lor Q \lor \neg P$$

➥ $P \lor Q \lor R$ is a blocked clause.

Blocked Clauses in First-Order Logic

- Blocked clauses for first-order logic can be defined in a similar way as in propositional logic.

Blocked Clauses in First-Order Logic

- Blocked clauses for first-order logic can be defined in a similar way as in propositional logic.
- Proving redundancy of blocked clauses in propositional logic is (relatively) simple.

Blocked Clauses in First-Order Logic

- Blocked clauses for first-order logic can be defined in a similar way as in propositional logic.
- Proving redundancy of blocked clauses in propositional logic is (relatively) simple.
- Proving redundancy of blocked clauses in first-order logic requires heavy machinery.
  - Herbrand's theorem,
  - factorization,
  - non-trivial properties of (most general) unification, etc.
- Required: A general theorem that helps us prove redundancy of several types of clauses in a unified way.

The Principle of Implication Modulo Resolution

- To prove correctness of the new techniques, we introduced the principle of implication modulo resolution.
  - A first-order variant of quantified implied outer resolvents (Heule, Seidl, and Biere, JAR, 2017).

The Principle of Implication Modulo Resolution

- To prove correctness of the new techniques, we introduced the principle of implication modulo resolution.
  - A first-order variant of quantified implied outer resolvents (Heule, Seidl, and Biere, JAR, 2017).

Definition

A clause C is implied modulo resolution by a formula F if all resolvents of C upon one of its literals are implied by $F \setminus \{C\}$.

The Principle of Implication Modulo Resolution

- To prove correctness of the new techniques, we introduced the principle of implication modulo resolution.
  - A first-order variant of quantified implied outer resolvents (Heule, Seidl, and Biere, JAR, 2017).

Definition

A clause C is implied modulo resolution by a formula F if all resolvents of C upon one of its literals are implied by $F \setminus \{C\}$.

➥ $F \setminus \{C\}$ might not imply C, but it implies all conclusions derived from C via resolution upon one of its literals.

The Principle of Implication Modulo Resolution

- To prove correctness of the new techniques, we introduced the principle of implication modulo resolution.
  - A first-order variant of quantified implied outer resolvents (Heule, Seidl, and Biere, JAR, 2017).

Definition

A clause C is implied modulo resolution by a formula F if all resolvents of C upon one of its literals are implied by $F \setminus \{C\}$.

Theorem (Main Result)

If a formula F implies a clause C modulo resolution, then C is redundant with respect to F.

Implication Modulo Resolution: Examples

Definition

A clause C is implied modulo resolution by a formula F if all resolvents of C upon one of its literals are implied by $F \setminus \{C\}$.

- Blocked clauses are implied modulo resolution:
  - Every resolvent is a tautology $\Rightarrow$ every resolvent is implied.

Implication Modulo Resolution: Examples

Definition

A clause C is implied modulo resolution by a formula F if all resolvents of C upon one of its literals are implied by $F \setminus \{C\}$.

- Blocked clauses are implied modulo resolution:
  - Every resolvent is a tautology $\Rightarrow$ every resolvent is implied.
- Clauses with pure literals:
  - Pure literals are literals whose predicate symbol occurs in only one polarity in F.

Implication Modulo Resolution: Examples

Definition

A clause C is implied modulo resolution by a formula F if all resolvents of C upon one of its literals are implied by $F \setminus \{C\}$.

- Blocked clauses are implied modulo resolution:
  - Every resolvent is a tautology $\Rightarrow$ every resolvent is implied.
- Clauses with pure literals:
  - Pure literals are literals whose predicate symbol occurs in only one polarity in F.
  - There are no resolvents upon a pure literal $\Rightarrow$ every resolvent is implied.

Implication Modulo Resolution: Examples

Definition

A clause C is implied modulo resolution by a formula F if all resolvents of C upon one of its literals are implied by $F \setminus \{C\}$.

- Blocked clauses are implied modulo resolution:
  - Every resolvent is a tautology $\Rightarrow$ every resolvent is implied.
- Clauses with pure literals:
  - Pure literals are literals whose predicate symbol occurs in only one polarity in F.
  - There are no resolvents upon a pure literal $\Rightarrow$ every resolvent is implied.
- Resolution asymmetric tautologies (RATs), resolution-subsumed clauses, etc.

Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.

Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):

Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):

Confluent Clause-Elimination Techniques

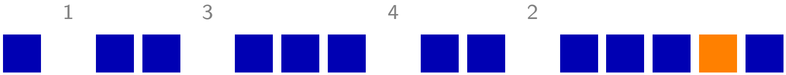
- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):

Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):

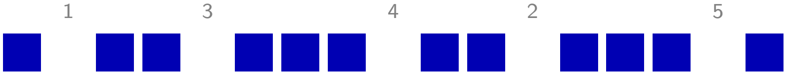
Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):



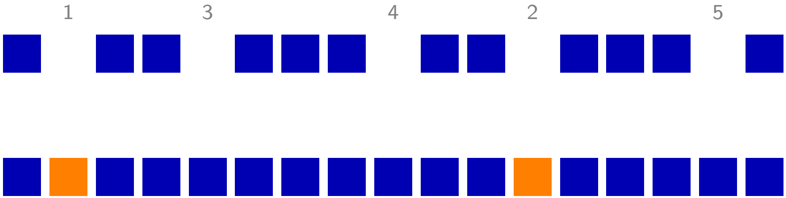
Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):



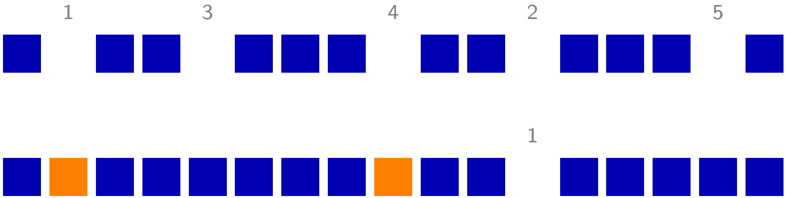
Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):



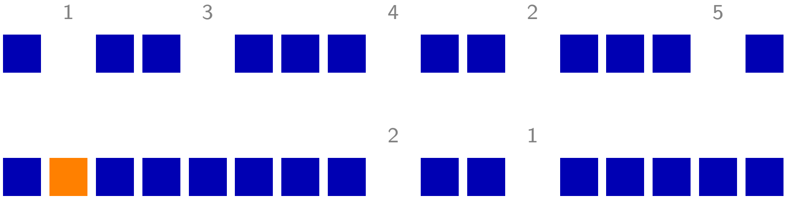
Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):



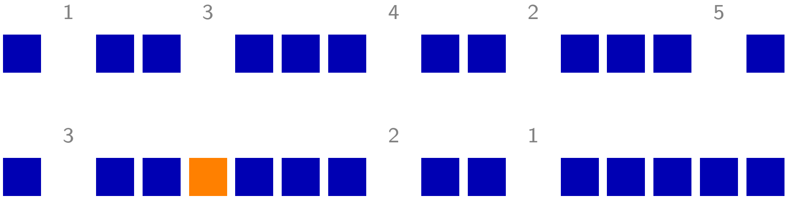
Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):



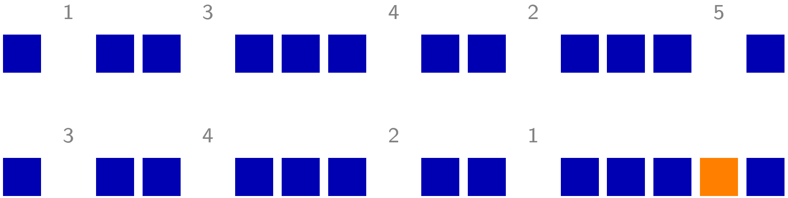
Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):



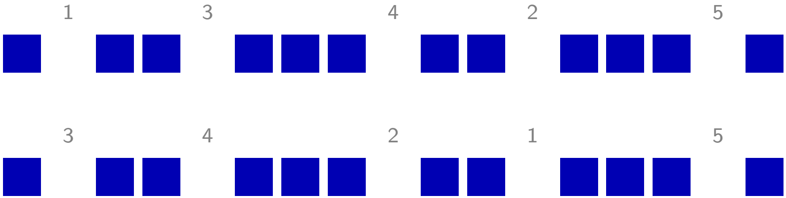
Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):



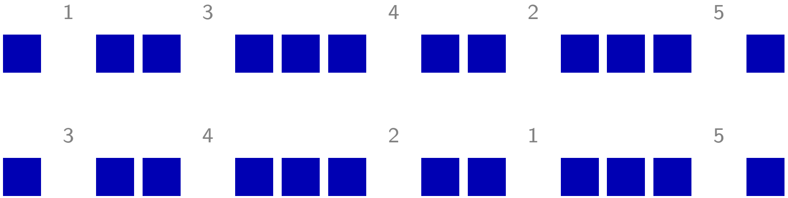
Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):



Confluent Clause-Elimination Techniques

- Confluence: Eliminating clauses in a different order yields the same result.
- Example (boxes are clauses, orange clauses are redundant according to some redundancy notion):



➥ We don't need to bother about the elimination order.

Confluence Results

Technique	Confluent
Blocked-Clause Elimination	✓
Covered-Clause Elimination	✓
Asymmetric-Tautology Elimination	✗
Resolution-Asymmetric-Tautology Elimination	✗
Resolution-Subsumed-Clause Elimination	✗

Confluence Results

Technique	**Confluent**
Blocked-Clause Elimination	✓
Covered-Clause Elimination	✓
Asymmetric-Tautology Elimination	✗
Resolution-Asymmetric-Tautology Elimination	✗
Resolution-Subsumed-Clause Elimination	✗
Covered-Literal Addition	✓
Asymmetric-Literal Addition	✓

Future Work

- Implication modulo resolution for first-order logic with equality.
  - ➥ Lift all preprocessing techniques to first-order logic with equality.
- Implement and evaluate a preprocessor with our techniques.
  - Blocked-clause elimination is already implemented.
  - Preprocessor is based on Vampire.

Summary

- Lifted clause-elimination techniques from SAT to first-order logic.
- Correctness proofs via principle of implication modulo resolution.
- Confluence analysis.
- Not in this talk but in the paper:
  - Short correctness proof for predicate elimination (Khasidashvili and Korovin, SAT, 2016) via implication modulo resolution.