

IN3070/4070 – Logic – Autumn 2022

Lecture 8: First-order Resolution

Martin Giese

11th October 2021



DEPARTMENT OF
INFORMATICS



UNIVERSITY OF
OSLO

Today's Plan

- ▶ Reminder: Clausal Form Translations
- ▶ Reminder: Propositional Resolution
- ▶ Reminder: Unification
- ▶ First-Order Resolution
- ▶ Soundness and Completeness
- ▶ Compactness
- ▶ Summary

Outline

- ▶ **Reminder: Clausal Form Translations**
- ▶ Reminder: Propositional Resolution
- ▶ Reminder: Unification
- ▶ First-Order Resolution
- ▶ Soundness and Completeness
- ▶ Compactness
- ▶ Summary

Translation into Clausal Form – Example

Example: $\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y)$

Try to prove this formula based on refutation in **CNF**

Translation into Clausal Form – Example

Example: $\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y)$

Try to prove this formula based on refutation in **CNF**

► negate the formula: $\neg(\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y))$

Translation into Clausal Form – Example

Example: $\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y)$

Try to prove this formula based on refutation in **CNF**

- ▶ negate the formula: $\neg(\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y))$
- ▶ Eliminate implication $\rightarrow$: $\neg(\neg \forall x \exists y p(x, y) \vee \exists y \forall x p(x, y))$

Translation into Clausal Form – Example

Example: $\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y)$

Try to prove this formula based on refutation in **CNF**

- ▶ negate the formula: $\neg(\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y))$
- ▶ Eliminate implication $\rightarrow$: $\neg(\neg \forall x \exists y p(x, y) \vee \exists y \forall x p(x, y))$
- ▶ Push negation inwards: $\forall x \exists y p(x, y) \wedge \forall y \exists x \neg p(x, y)$

Translation into Clausal Form – Example

Example: $\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y)$

Try to prove this formula based on refutation in **CNF**

- ▶ negate the formula: $\neg(\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y))$
- ▶ Eliminate implication $\rightarrow$: $\neg(\neg \forall x \exists y p(x, y) \vee \exists y \forall x p(x, y))$
- ▶ Push negation inwards: $\forall x \exists y p(x, y) \wedge \forall y \exists x \neg p(x, y)$
- ▶ Rename bound variables: $\forall x \exists y p(x, y) \wedge \forall w \exists z \neg p(z, w)$

Translation into Clausal Form – Example

Example: $\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y)$

Try to prove this formula based on refutation in **CNF**

- ▶ negate the formula: $\neg(\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y))$
- ▶ Eliminate implication $\rightarrow$: $\neg(\neg \forall x \exists y p(x, y) \vee \exists y \forall x p(x, y))$
- ▶ Push negation inwards: $\forall x \exists y p(x, y) \wedge \forall y \exists x \neg p(x, y)$
- ▶ Rename bound variables: $\forall x \exists y p(x, y) \wedge \forall w \exists z \neg p(z, w)$
- ▶ Move quantifiers to prefix: $\forall x \exists y \forall w \exists z (p(x, y) \wedge \neg p(z, w))$

Translation into Clausal Form – Example

Example: $\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y)$

Try to prove this formula based on refutation in **CNF**

- ▶ negate the formula: $\neg(\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y))$
- ▶ Eliminate implication $\rightarrow$: $\neg(\neg \forall x \exists y p(x, y) \vee \exists y \forall x p(x, y))$
- ▶ Push negation inwards: $\forall x \exists y p(x, y) \wedge \forall y \exists x \neg p(x, y)$
- ▶ Rename bound variables: $\forall x \exists y p(x, y) \wedge \forall w \exists z \neg p(z, w)$
- ▶ Move quantifiers to prefix: $\forall x \exists y \forall w \exists z (p(x, y) \wedge \neg p(z, w))$
- ▶ Skolemize, i.e., replace $\exists$: $\forall x \forall w (p(x, f(x)) \wedge \neg p(g(x, w), w))$

Translation into Clausal Form – Example

Example: $\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y)$

Try to prove this formula based on refutation in **CNF**

- ▶ negate the formula: $\neg(\forall x \exists y p(x, y) \rightarrow \exists y \forall x p(x, y))$
- ▶ Eliminate implication $\rightarrow$: $\neg(\neg \forall x \exists y p(x, y) \vee \exists y \forall x p(x, y))$
- ▶ Push negation inwards: $\forall x \exists y p(x, y) \wedge \forall y \exists x \neg p(x, y)$
- ▶ Rename bound variables: $\forall x \exists y p(x, y) \wedge \forall w \exists z \neg p(z, w)$
- ▶ Move quantifiers to prefix: $\forall x \exists y \forall w \exists z (p(x, y) \wedge \neg p(z, w))$
- ▶ Skolemize, i.e., replace $\exists$: $\forall x \forall w (p(x, f(x)) \wedge \neg p(g(x, w), w))$
- ▶ Write in clausal form : $\{\{p(x, f(x))\}, \{\neg p(g(x, w), w)\}\}$

Outline

- ▶ Reminder: Clausal Form Translations
- ▶ **Reminder: Propositional Resolution**
- ▶ Reminder: Unification
- ▶ First-Order Resolution
- ▶ Soundness and Completeness
- ▶ Compactness
- ▶ Summary

The Resolution Rule

The resolution calculus is a **refutation procedure**.

- ▶ in order to determine whether a formula F (in clausal form) is valid, we check whether $\neg F$ is **unsatisfiable**

The Resolution Rule

The resolution calculus is a **refutation procedure**.

- ▶ in order to determine whether a formula F (in clausal form) is valid, we check whether $\neg F$ is **unsatisfiable**

Definition 2.1 (Complementary Literal).

The complementary literal $\bar{L}$ of a literal L is A if L is of the form $\neg A$, otherwise it is $\neg L$.

The Resolution Rule

The resolution calculus is a **refutation procedure**.

- ▶ in order to determine whether a formula F (in clausal form) is valid, we check whether $\neg F$ is **unsatisfiable**

Definition 2.1 (Complementary Literal).

The complementary literal $\bar{L}$ of a literal L is A if L is of the form $\neg A$, otherwise it is $\neg L$.

Definition 2.2 (Resolution Rule).

*Let C_1, C_2 be clauses with $L \in C_1$ and $\bar{L} \in C_2$. The **resolvent** C' of C_1 and C_2 is $(C_1 \setminus \{L\}) \cup (C_2 \setminus \{\bar{L}\})$. C_1 and C_2 are the **parents** of C' .*

The Resolution Rule

The resolution calculus is a **refutation procedure**.

- ▶ in order to determine whether a formula F (in clausal form) is valid, we check whether $\neg F$ is **unsatisfiable**

Definition 2.1 (Complementary Literal).

The complementary literal $\bar{L}$ of a literal L is A if L is of the form $\neg A$, otherwise it is $\neg L$.

Definition 2.2 (Resolution Rule).

*Let C_1, C_2 be clauses with $L \in C_1$ and $\bar{L} \in C_2$. The **resolvent** C' of C_1 and C_2 is $(C_1 \setminus \{L\}) \cup (C_2 \setminus \{\bar{L}\})$. C_1 and C_2 are the **parents** of C' .*

- ▶ the resolution rule maintains satisfiability: If $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then $\mathcal{I} \models C'$

The Resolution Rule

The resolution calculus is a **refutation procedure**.

- ▶ in order to determine whether a formula F (in clausal form) is valid, we check whether $\neg F$ is **unsatisfiable**

Definition 2.1 (Complementary Literal).

The complementary literal $\bar{L}$ of a literal L is A if L is of the form $\neg A$, otherwise it is $\neg L$.

Definition 2.2 (Resolution Rule).

*Let C_1, C_2 be clauses with $L \in C_1$ and $\bar{L} \in C_2$. The **resolvent** C' of C_1 and C_2 is $(C_1 \setminus \{L\}) \cup (C_2 \setminus \{\bar{L}\})$. C_1 and C_2 are the **parents** of C' .*

- ▶ the resolution rule maintains satisfiability: If $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then $\mathcal{I} \models C'$
- ▶ if a set of clauses S is satisfiable and $C_1, C_2 \in S$, then $S \cup \{C'\}$ is satisfiable.

The Resolution Rule – Example

Example: Let $C_1 = \{a, b, \neg c\}$ and $C_2 = \{b, c, \neg e\}$.

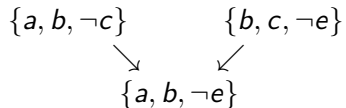
The Resolution Rule – Example

Example: Let $C_1 = \{a, b, \neg c\}$ and $C_2 = \{b, c, \neg e\}$.

$$\{a, b, \neg c\} \qquad \{b, c, \neg e\}$$

The Resolution Rule – Example

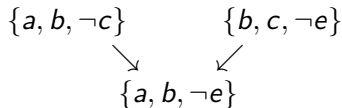
Example: Let $C_1 = \{a, b, \neg c\}$ and $C_2 = \{b, c, \neg e\}$.



The resolvent of C_1 and C_2 is $\{a, b, \neg e\}$.

The Resolution Rule – Example

Example: Let $C_1 = \{a, b, \neg c\}$ and $C_2 = \{b, c, \neg e\}$.



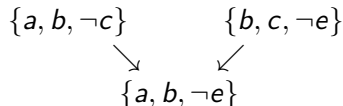
The resolvent of C_1 and C_2 is $\{a, b, \neg e\}$.

Observations:

- ▶ if $\{a, b, \neg c\}$ and $\{b, c, \neg e\} \equiv (a \vee b \vee \neg c) \wedge (b \vee c \vee \neg e)$ are **true** in $\mathcal{I}$, then $(a \vee b)$ is true (if c is true) or $(b \vee \neg e)$ is true (if c is false); hence $(a \vee b \vee \neg e)$ is **true**

The Resolution Rule – Example

Example: Let $C_1 = \{a, b, \neg c\}$ and $C_2 = \{b, c, \neg e\}$.



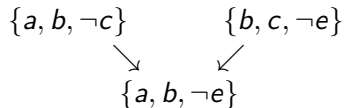
The resolvent of C_1 and C_2 is $\{a, b, \neg e\}$.

Observations:

- ▶ if $\{a, b, \neg c\}$ and $\{b, c, \neg e\} \equiv (a \vee b \vee \neg c) \wedge (b \vee c \vee \neg e)$ are **true** in $\mathcal{I}$, then $(a \vee b)$ is true (if c is true) or $(b \vee \neg e)$ is true (if c is false); hence $(a \vee b \vee \neg e)$ is **true**
- ▶ if resolvent is **unsatisfiable**, then conj. of parents is **unsatisfiable**
- ▶ the empty clause $\square$ is **unsatisfiable**

The Resolution Rule – Example

Example: Let $C_1 = \{a, b, \neg c\}$ and $C_2 = \{b, c, \neg e\}$.



The resolvent of C_1 and C_2 is $\{a, b, \neg e\}$.

Observations:

- ▶ if $\{a, b, \neg c\}$ and $\{b, c, \neg e\} \equiv (a \vee b \vee \neg c) \wedge (b \vee c \vee \neg e)$ are **true** in $\mathcal{I}$, then $(a \vee b)$ is true (if c is true) or $(b \vee \neg e)$ is true (if c is false); hence $(a \vee b \vee \neg e)$ is **true**
- ▶ if resolvent is **unsatisfiable**, then conj. of parents is **unsatisfiable**
- ▶ the empty clause $\square$ is **unsatisfiable**
- ▶ **goal:** derive empty clause $\square$

The Resolution Procedure

Definition 2.3 (Resolution Procedure).

Given a set of clauses S .

- 1. apply the resolution rule to a pair of clauses $\{C_1, C_2\} \subseteq S$ that has not been chosen before; let C' be the resolvent*

The Resolution Procedure

Definition 2.3 (Resolution Procedure).

Given a set of clauses S .

- 1. apply the resolution rule to a pair of clauses $\{C_1, C_2\} \subseteq S$ that has not been chosen before; let C' be the resolvent*
- 2. $S' := S \cup \{C'\}$, $S := S'$*

The Resolution Procedure

Definition 2.3 (Resolution Procedure).

Given a set of clauses S .

- 1. apply the resolution rule to a pair of clauses $\{C_1, C_2\} \subseteq S$ that has not been chosen before; let C' be the resolvent*
- 2. $S' := S \cup \{C'\}$, $S := S'$*
- 3. if $C' = \square$, then output “**unsatisfiable**”;*

The Resolution Procedure

Definition 2.3 (Resolution Procedure).

Given a set of clauses S .

1. apply the resolution rule to a pair of clauses $\{C_1, C_2\} \subseteq S$ that has not been chosen before; let C' be the resolvent
2. $S' := S \cup \{C'\}$, $S := S'$
3. if $C' = \square$, then output “*unsatisfiable*”;
4. if all possible resolvents have been considered, then output “*satisfiable*”; otherwise continue with 1.

The Formal Resolution Calculus

Definition 2.4 (Resolution Calculus).

*The **resolution calculus** has one axiom and one (inference) rule.*

The Formal Resolution Calculus

Definition 2.4 (Resolution Calculus).

The *resolution calculus* has one axiom and one (inference) rule.

$$\frac{}{C_1, \dots, \square, \dots C_n} \text{ axiom}$$

The Formal Resolution Calculus

Definition 2.4 (Resolution Calculus).

The *resolution calculus* has one axiom and one (inference) rule.

$$\frac{}{C_1, \dots, \square, \dots, C_n} \text{ axiom}$$

$$\frac{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n, C_i \cup C_j}{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n} \text{ resolution}$$

The Formal Resolution Calculus

Definition 2.4 (Resolution Calculus).

The *resolution calculus* has one axiom and one (inference) rule.

$$\frac{}{C_1, \dots, \square, \dots, C_n} \text{ axiom}$$

$$\frac{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n, C_i \cup C_j}{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n} \text{ resolution}$$

A *resolution proof* of a set of clauses S is a derivation of S in the resolution calculus.

The Formal Resolution Calculus

Definition 2.4 (Resolution Calculus).

The *resolution calculus* has one axiom and one (inference) rule.

$$\frac{}{C_1, \dots, \square, \dots, C_n} \text{ axiom}$$

$$\frac{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n, C_i \cup C_j}{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n} \text{ resolution}$$

A *resolution proof* of a set of clauses S is a derivation of S in the resolution calculus.

- ▶ in contrast to natural deduction or the sequent calculus, the resolution calculus has no rule with more than **one premise**

The Formal Resolution Calculus

Definition 2.4 (Resolution Calculus).

The *resolution calculus* has one axiom and one (inference) rule.

$$\frac{}{C_1, \dots, \square, \dots, C_n} \text{ axiom}$$

$$\frac{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n, C_i \cup C_j}{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n} \text{ resolution}$$

A *resolution proof* of a set of clauses S is a derivation of S in the resolution calculus.

- ▶ in contrast to natural deduction or the sequent calculus, the resolution calculus has no rule with more than **one premise**
- ▶ hence, a derivation in the resolution calculus has **only one branch**

The Formal Resolution Calculus

Definition 2.4 (Resolution Calculus).

The *resolution calculus* has one axiom and one (inference) rule.

$$\frac{}{C_1, \dots, \square, \dots, C_n} \text{ axiom}$$

$$\frac{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n, C_i \cup C_j}{C_1, \dots, C_i \cup \{L\}, \dots, C_j \cup \{\bar{L}\}, \dots, C_n} \text{ resolution}$$

A *resolution proof* of a set of clauses S is a derivation of S in the resolution calculus.

- ▶ in contrast to natural deduction or the sequent calculus, the resolution calculus has no rule with more than **one premise**
- ▶ hence, a derivation in the resolution calculus has **only one branch**
- ▶ **terminates**, if all clauses $C_i \cup \{L\}, C_j \cup \{\bar{L}\}$ have been considered

Outline

- ▶ Reminder: Clausal Form Translations
- ▶ Reminder: Propositional Resolution
- ▶ **Reminder: Unification**
- ▶ First-Order Resolution
- ▶ Soundness and Completeness
- ▶ Compactness
- ▶ Summary

Unification

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{\neg p(a, y)\} \}$$

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{\neg p(a, y)\} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{\neg p(a, y)\} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

- Should be OK to instantiate x with a and y with b

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{\neg p(a, y)\} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

- Should be OK to instantiate x with a and y with b
- Giving

$$\{ \{p(a, b)\}, \{\neg p(a, b)\} \}$$

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{\neg p(a, y)\} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

- Should be OK to instantiate x with a and y with b
- Giving

$$\{ \{p(a, b)\}, \{\neg p(a, b)\} \}$$

- Which can be resolved to $\square$

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{\neg p(a, y)\} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

- Should be OK to instantiate x with a and y with b
- Giving

$$\{ \{p(a, b)\}, \{\neg p(a, b)\} \}$$

- Which can be resolved to $\square$

Unification problem

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{\neg p(a, y)\} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

- Should be OK to instantiate x with a and y with b
- Giving

$$\{ \{p(a, b)\}, \{\neg p(a, b)\} \}$$

- Which can be resolved to $\square$

Unification problem

Let s and t be terms.

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{\neg p(a, y)\} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

- Should be OK to instantiate x with a and y with b
- Giving

$$\{ \{p(a, b)\}, \{\neg p(a, b)\} \}$$

- Which can be resolved to $\square$

Unification problem

Let s and t be terms. Find *all* substitutions that make s and t syntactically equal

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{\neg p(a, y)\} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

- Should be OK to instantiate x with a and y with b
- Giving

$$\{ \{p(a, b)\}, \{\neg p(a, b)\} \}$$

- Which can be resolved to $\square$

Unification problem

Let s and t be terms. Find *all* substitutions that make s and t syntactically equal, i.e. all σ with $\sigma(s) = \sigma(t)$.

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{ \neg p(a, y) \} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

- Should be OK to instantiate x with a and y with b
- Giving

$$\{ \{p(a, b)\}, \{ \neg p(a, b) \} \}$$

- Which can be resolved to $\square$

Unification problem

Let s and t be terms. Find *all* substitutions that make s and t syntactically equal, i.e. all σ with $\sigma(s) = \sigma(t)$.

- A substitution that makes s and t syntactically equal is called a **unifier** for s and t .

Unification

- Motivation: try refuting the following

$$\{ \{p(x, b)\}, \{ \neg p(a, y) \} \}$$

- Remember: these mean

$$\forall x p(x, b) \quad \text{and} \quad \forall y \neg p(a, y)$$

- Should be OK to instantiate x with a and y with b
- Giving

$$\{ \{p(a, b)\}, \{ \neg p(a, b) \} \}$$

- Which can be resolved to $\square$

Unification problem

Let s and t be terms. Find *all* substitutions that make s and t syntactically equal, i.e. all σ with $\sigma(s) = \sigma(t)$.

- A substitution that makes s and t syntactically equal is called a **unifier** for s and t .
- Two terms are **unifiable** if they have a unifier.

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms.

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms. A substitution σ is

- ▶ a *unifier* for s and t if $\sigma(s) = \sigma(t)$.

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms. A substitution σ is

- ▶ a *unifier* for s and t if $\sigma(s) = \sigma(t)$.
- ▶ a *most general unifier* (mgu) for s and t if

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms. A substitution σ is

- ▶ a *unifier* for s and t if $\sigma(s) = \sigma(t)$.
- ▶ a *most general unifier* (mgu) for s and t if
 - ▶ it is a unifier for s and t , and

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms. A substitution σ is

- ▶ a **unifier** for s and t if $\sigma(s) = \sigma(t)$.
- ▶ a **most general unifier** (mgu) for s and t if
 - ▶ it is a unifier for s and t , and
 - ▶ it is more general than any other unifiers for s and t .

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms. A substitution σ is

- ▶ a **unifier** for s and t if $\sigma(s) = \sigma(t)$.
- ▶ a **most general unifier** (mgu) for s and t if
 - ▶ it is a unifier for s and t , and
 - ▶ it is more general than any other unifiers for s and t .

We say that s and t are **unifiable** if they have a unifier.

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms. A substitution σ is

- ▶ a **unifier** for s and t if $\sigma(s) = \sigma(t)$.
- ▶ a **most general unifier** (mgu) for s and t if
 - ▶ it is a unifier for s and t , and
 - ▶ it is more general than any other unifiers for s and t .

We say that s and t are **unifiable** if they have a unifier.

Let $s = f(x)$ and $t = f(y)$.

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms. A substitution σ is

- ▶ a **unifier** for s and t if $\sigma(s) = \sigma(t)$.
- ▶ a **most general unifier** (mgu) for s and t if
 - ▶ it is a unifier for s and t , and
 - ▶ it is more general than any other unifiers for s and t .

We say that s and t are **unifiable** if they have a unifier.

Let $s = f(x)$ and $t = f(y)$.

- ▶ $\sigma_1 = \{x \backslash a, y \backslash a\}$ is a unifier for s and t

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms. A substitution σ is

- ▶ a **unifier** for s and t if $\sigma(s) = \sigma(t)$.
- ▶ a **most general unifier** (mgu) for s and t if
 - ▶ it is a unifier for s and t , and
 - ▶ it is more general than any other unifiers for s and t .

We say that s and t are **unifiable** if they have a unifier.

Let $s = f(x)$ and $t = f(y)$.

- ▶ $\sigma_1 = \{x \backslash a, y \backslash a\}$ is a unifier for s and t
- ▶ $\sigma_2 = \{x \backslash y\}$ and $\sigma_3 = \{y \backslash x\}$ are also unifiers for s and t

Most General Unifiers

Definition 3.1 (Unifier, Most General Unifier).

Let s and t be terms. A substitution σ is

- ▶ a **unifier** for s and t if $\sigma(s) = \sigma(t)$.
- ▶ a **most general unifier** (mgu) for s and t if
 - ▶ it is a unifier for s and t , and
 - ▶ it is more general than any other unifiers for s and t .

We say that s and t are **unifiable** if they have a unifier.

Let $s = f(x)$ and $t = f(y)$.

- ▶ $\sigma_1 = \{x \backslash a, y \backslash a\}$ is a unifier for s and t
- ▶ $\sigma_2 = \{x \backslash y\}$ and $\sigma_3 = \{y \backslash x\}$ are also unifiers for s and t
- ▶ σ_2 and σ_3 are the most general unifiers for s and t

Uniqueness “up to variable renaming”

Proposition 3.1.

If σ_1 and σ_2 are most general unifiers for two terms s and t , then there is a variable renaming η such that $\eta\sigma_1 = \sigma_2$.

Uniqueness “up to variable renaming”

Proposition 3.1.

If σ_1 and σ_2 are most general unifiers for two terms s and t , then there is a variable renaming η such that $\eta\sigma_1 = \sigma_2$.

- We leave out the proof.

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

while $(\sigma(t_1) \neq \sigma(t_2))$ **do**

end while

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

while $(\sigma(t_1) \neq \sigma(t_2))$ **do**

 choose a critical pair $\langle k_1, k_2 \rangle$ for $\sigma(t_1), \sigma(t_2);$

end while

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

while $(\sigma(t_1) \neq \sigma(t_2))$ **do**

 choose a critical pair $\langle k_1, k_2 \rangle$ for $\sigma(t_1), \sigma(t_2);$

if (neither k_1 nor k_2 are variables) **then**

end if

end while

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

while $(\sigma(t_1) \neq \sigma(t_2))$ **do**

 choose a critical pair $\langle k_1, k_2 \rangle$ for $\sigma(t_1), \sigma(t_2);$

if (neither k_1 nor k_2 are variables) **then**

 return *“not unifiable”*;

end if

end while

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

while $(\sigma(t_1) \neq \sigma(t_2))$ **do**

 choose a critical pair $\langle k_1, k_2 \rangle$ for $\sigma(t_1), \sigma(t_2)$;

if (neither k_1 nor k_2 are variables) **then**

 return *“not unifiable”*;

end if

$x :=$ the one of k_1, k_2 that is a variable (if both are, choose one)

$t :=$ the one of k_1, k_2 that is not x ;

end while

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

while $(\sigma(t_1) \neq \sigma(t_2))$ **do**

 choose a critical pair $\langle k_1, k_2 \rangle$ for $\sigma(t_1), \sigma(t_2)$;

if (neither k_1 nor k_2 are variables) **then**

 return “not unifiable”;

end if

$x :=$ the one of k_1, k_2 that is a variable (if both are, choose one)

$t :=$ the one of k_1, k_2 that is not x ;

if (x occurs in t) **then**

end if

end while

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

while $(\sigma(t_1) \neq \sigma(t_2))$ **do**

 choose a critical pair $\langle k_1, k_2 \rangle$ for $\sigma(t_1), \sigma(t_2)$;

if (neither k_1 nor k_2 are variables) **then**

 return “not unifiable”;

end if

$x :=$ the one of k_1, k_2 that is a variable (if both are, choose one)

$t :=$ the one of k_1, k_2 that is not x ;

if (x occurs in t) **then**

 return “not unifiable”;

end if

end while

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

while $(\sigma(t_1) \neq \sigma(t_2))$ **do**

 choose a critical pair $\langle k_1, k_2 \rangle$ for $\sigma(t_1), \sigma(t_2)$;

if (neither k_1 nor k_2 are variables) **then**

 return *“not unifiable”*;

end if

$x :=$ the one of k_1, k_2 that is a variable (if both are, choose one)

$t :=$ the one of k_1, k_2 that is not x ;

if (x occurs in t) **then**

 return *“not unifiable”*;

end if

$\sigma := \{x \backslash t\} \sigma;$

end while

Unification Algorithm

Algorithm: $\text{unify}(t_1, t_2)$

$\sigma := \epsilon;$

while $(\sigma(t_1) \neq \sigma(t_2))$ **do**

 choose a critical pair $\langle k_1, k_2 \rangle$ for $\sigma(t_1), \sigma(t_2)$;

if (neither k_1 nor k_2 are variables) **then**

 return “not unifiable”;

end if

$x :=$ the one of k_1, k_2 that is a variable (if both are, choose one)

$t :=$ the one of k_1, k_2 that is not x ;

if (x occurs in t) **then**

 return “not unifiable”;

end if

$\sigma := \{x \backslash t\} \sigma;$

end while

return σ ;

Properties of the Unification Algorithm

- ▶ If the terms t_1 and t_2 are unifiable, the algorithm returns a most general unifier for t_1 and t_2 .

Properties of the Unification Algorithm

- ▶ If the terms t_1 and t_2 are unifiable, the algorithm returns a most general unifier for t_1 and t_2 .
- ▶ If t_1 and t_2 are **not** unifiable, the algorithm returns *“not unifiable”*.

Outline

- ▶ Reminder: Clausal Form Translations
- ▶ Reminder: Propositional Resolution
- ▶ Reminder: Unification
- ▶ **First-Order Resolution**
- ▶ Soundness and Completeness
- ▶ Compactness
- ▶ Summary

The First-Order Resolution Calculus

The resolution rule is generalized by performing unification as part of the rule and an additional factoring rule is added.

The First-Order Resolution Calculus

The resolution rule is generalized by performing unification as part of the rule and an additional factoring rule is added.

Definition 4.1 (First-Order Resolution Calculus).

$$\frac{}{C_1, \dots, \{\}, \dots, C_n} \text{ axiom}$$

The First-Order Resolution Calculus

The resolution rule is generalized by performing unification as part of the rule and an additional factoring rule is added.

Definition 4.1 (First-Order Resolution Calculus).

$$\begin{array}{c}
 \overline{C_1, \dots, \{\}, \dots, C_n} \text{ axiom} \\
 \\
 \overline{C_1, \dots, C_i \cup \{L_1\}, \dots, C_j \cup \{L_2\}, \dots, C_n, \sigma(C_i \cup C_j)} \\
 \overline{C_1, \dots, C_i \cup \{L_1\}, \dots, C_j \cup \{L_2\}, \dots, C_n} \text{ resolution}
 \end{array}$$

with σ a m.g.u. of L_1 and $\overline{L_2}$.

The First-Order Resolution Calculus

The resolution rule is generalized by performing unification as part of the rule and an additional factoring rule is added.

Definition 4.1 (First-Order Resolution Calculus).

$$\frac{}{C_1, \dots, \{\}, \dots, C_n} \text{ axiom}$$

$$\frac{C_1, \dots, C_i \cup \{L_1\}, \dots, C_j \cup \{L_2\}, \dots, C_n, \sigma(C_i \cup C_j)}{C_1, \dots, C_i \cup \{L_1\}, \dots, C_j \cup \{L_2\}, \dots, C_n} \text{ resolution}$$

with σ a m.g.u. of L_1 and $\overline{L_2}$.

$$\frac{C_1, \dots, C_i \cup \{L_1, \dots, L_m\}, \dots, C_n, \sigma(C_i \cup \{L_1\})}{C_1, \dots, C_i \cup \{L_1, \dots, L_m\}, \dots, C_n} \text{ factoring}$$

with σ a m.g.u. of $L_1 \dots L_m$.

The First-Order Resolution Calculus

The resolution rule is generalized by performing unification as part of the rule and an additional factoring rule is added.

Definition 4.1 (First-Order Resolution Calculus).

$$\begin{array}{c}
 \overline{C_1, \dots, \{\}, \dots, C_n} \text{ axiom} \\
 \\
 \frac{C_1, \dots, C_i \cup \{L_1\}, \dots, C_j \cup \{L_2\}, \dots, C_n, \sigma(C_i \cup C_j)}{C_1, \dots, C_i \cup \{L_1\}, \dots, C_j \cup \{L_2\}, \dots, C_n} \text{ resolution} \\
 \text{with } \sigma \text{ a m.g.u. of } L_1 \text{ and } \overline{L_2}. \\
 \\
 \frac{C_1, \dots, C_i \cup \{L_1, \dots, L_m\}, \dots, C_n, \sigma(C_i \cup \{L_1\})}{C_1, \dots, C_i \cup \{L_1, \dots, L_m\}, \dots, C_n} \text{ factoring} \\
 \text{with } \sigma \text{ a m.g.u. of } L_1 \dots L_m.
 \end{array}$$

- a **resolution proof** for a set of clauses S is a derivation of S in the resolution calculus; the **substitution** σ is local for every rule application; variables in every clause C can be **renamed**

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$
8. $\neg q(a)$ — from 3 and 6 with $[x \backslash a]$

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$
8. $\neg q(a)$ — from 3 and 6 with $[x \setminus a]$
9. $\neg r'(a)$ — from 3 and 7 with $[x \setminus a]$

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$
8. $\neg q(a)$ — from 3 and 6 with $[x \setminus a]$
9. $\neg r'(a)$ — from 3 and 7 with $[x \setminus a]$
10. $q(a), r(a, f(a))$ — from 1 and 4 with $[x \setminus a]$

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$
8. $\neg q(a)$ — from 3 and 6 with $[x \setminus a]$
9. $\neg r'(a)$ — from 3 and 7 with $[x \setminus a]$
10. $q(a), r(a, f(a))$ — from 1 and 4 with $[x \setminus a]$
11. $q(a), r'(f(a))$ — from 2 and 4 with $[x \setminus a]$

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$
8. $\neg q(a)$ — from 3 and 6 with $[x \setminus a]$
9. $\neg r'(a)$ — from 3 and 7 with $[x \setminus a]$
10. $q(a), r(a, f(a))$ — from 1 and 4 with $[x \setminus a]$
11. $q(a), r'(f(a))$ — from 2 and 4 with $[x \setminus a]$
12. $r(a, f(a))$ — from 10 and 8 with $[]$ (empty substitution)

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$
8. $\neg q(a)$ — from 3 and 6 with $[x \setminus a]$
9. $\neg r'(a)$ — from 3 and 7 with $[x \setminus a]$
10. $q(a), r(a, f(a))$ — from 1 and 4 with $[x \setminus a]$
11. $q(a), r'(f(a))$ — from 2 and 4 with $[x \setminus a]$
12. $r(a, f(a))$ — from 10 and 8 with $[]$ (empty substitution)
13. $r'(f(a))$ — from 11 and 8 with $[]$

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$
8. $\neg q(a)$ — from 3 and 6 with $[x \setminus a]$
9. $\neg r'(a)$ — from 3 and 7 with $[x \setminus a]$
10. $q(a), r(a, f(a))$ — from 1 and 4 with $[x \setminus a]$
11. $q(a), r'(f(a))$ — from 2 and 4 with $[x \setminus a]$
12. $r(a, f(a))$ — from 10 and 8 with $[]$ (empty substitution)
13. $r'(f(a))$ — from 11 and 8 with $[]$
14. $p'(f(a))$ — from 12 and 5 with $[y \setminus f(a)]$

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$
8. $\neg q(a)$ — from 3 and 6 with $[x \setminus a]$
9. $\neg r'(a)$ — from 3 and 7 with $[x \setminus a]$
10. $q(a), r(a, f(a))$ — from 1 and 4 with $[x \setminus a]$
11. $q(a), r'(f(a))$ — from 2 and 4 with $[x \setminus a]$
12. $r(a, f(a))$ — from 10 and 8 with $[]$ (empty substitution)
13. $r'(f(a))$ — from 11 and 8 with $[]$
14. $p'(f(a))$ — from 12 and 5 with $[y \setminus f(a)]$
15. $\neg p'(f(a))$ — from 13 and 7 with $[x \setminus f(a)]$

First-Order Resolution Calculus – Example

1. $\neg p(x), q(x), r(x, f(x))$
2. $\neg p(x), q(x), r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y), p'(y)$
6. $\neg p'(x), \neg q(x)$
7. $\neg p'(x), \neg r'(x)$
8. $\neg q(a)$ — from 3 and 6 with $[x \setminus a]$
9. $\neg r'(a)$ — from 3 and 7 with $[x \setminus a]$
10. $q(a), r(a, f(a))$ — from 1 and 4 with $[x \setminus a]$
11. $q(a), r'(f(a))$ — from 2 and 4 with $[x \setminus a]$
12. $r(a, f(a))$ — from 10 and 8 with $[]$ (empty substitution)
13. $r'(f(a))$ — from 11 and 8 with $[]$
14. $p'(f(a))$ — from 12 and 5 with $[y \setminus f(a)]$
15. $\neg p'(f(a))$ — from 13 and 7 with $[x \setminus f(a)]$
16. $\square$ — from 14 and 15 with $[]$

The Necessity of Factoring

$$(1) : \quad p(x) \vee p(y)$$

$$(2) : \quad \neg p(u) \vee \neg p(v)$$

Using only resolution:

The Necessity of Factoring

$$(1) : \quad p(x) \vee p(y)$$

$$(2) : \quad \neg p(u) \vee \neg p(v)$$

Using only resolution:

$$(3) : \quad p(y') \vee \neg p(v') \quad \text{by resolving (1) and (2), with } x \setminus u$$

The Necessity of Factoring

$$(1): \quad p(x) \vee p(y)$$

$$(2): \quad \neg p(u) \vee \neg p(v)$$

Using only resolution:

$$(3): \quad p(y') \vee \neg p(v') \quad \text{by resolving (1) and (2), with } x \setminus u$$

$$(4): \quad \neg p(u') \vee \neg p(v'') \quad \text{by resolving (2) and (3), with } u \setminus y'$$

The Necessity of Factoring

$$(1): \quad p(x) \quad \vee \quad p(y)$$

$$(2): \quad \neg p(u) \quad \vee \quad \neg p(v)$$

Using only resolution:

$$(3): \quad p(y') \quad \vee \quad \neg p(v') \quad \text{by resolving (1) and (2), with } x \setminus u$$

$$(4): \quad \neg p(u') \quad \vee \quad \neg p(v'') \quad \text{by resolving (2) and (3), with } u \setminus y'$$

$$(5): \quad p(x') \quad \vee \quad p(y'') \quad \text{by resolving (1) and (3), with } y \setminus v'$$

The Necessity of Factoring

$$(1): \quad p(x) \vee p(y)$$

$$(2): \quad \neg p(u) \vee \neg p(v)$$

Using only resolution:

$$(3): \quad p(y') \vee \neg p(v') \quad \text{by resolving (1) and (2), with } x \setminus u$$

$$(4): \quad \neg p(u') \vee \neg p(v'') \quad \text{by resolving (2) and (3), with } u \setminus y'$$

$$(5): \quad p(x') \vee p(y'') \quad \text{by resolving (1) and (3), with } y \setminus v'$$

$$(6): \quad p(y''') \vee \neg p(v''') \quad \text{by resolving (3) and (3) ...}$$

The Necessity of Factoring

$$(1): \quad p(x) \vee p(y)$$

$$(2): \quad \neg p(u) \vee \neg p(v)$$

Using only resolution:

$$(3): \quad p(y') \vee \neg p(v') \quad \text{by resolving (1) and (2), with } x \setminus u$$

$$(4): \quad \neg p(u') \vee \neg p(v'') \quad \text{by resolving (2) and (3), with } u \setminus y'$$

$$(5): \quad p(x') \vee p(y'') \quad \text{by resolving (1) and (3), with } y \setminus v'$$

$$(6): \quad p(y''') \vee \neg p(v''') \quad \text{by resolving (3) and (3) ...}$$

Using factorisation:

$$(3): \quad p(x') \quad \text{by factoring (1) with } x \setminus y$$

The Necessity of Factoring

$$(1): \quad p(x) \vee p(y)$$

$$(2): \quad \neg p(u) \vee \neg p(v)$$

Using only resolution:

$$(3): \quad p(y') \vee \neg p(v') \quad \text{by resolving (1) and (2), with } x \setminus u$$

$$(4): \quad \neg p(u') \vee \neg p(v'') \quad \text{by resolving (2) and (3), with } u \setminus y'$$

$$(5): \quad p(x') \vee p(y'') \quad \text{by resolving (1) and (3), with } y \setminus v'$$

$$(6): \quad p(y''') \vee \neg p(v''') \quad \text{by resolving (3) and (3) ...}$$

Using factorisation:

$$(3): \quad p(x') \quad \text{by factoring (1) with } x \setminus y$$

$$(4): \quad \neg p(u') \quad \text{by factoring (2) with } u \setminus v$$

The Necessity of Factoring

$$(1): \quad p(x) \vee p(y)$$

$$(2): \quad \neg p(u) \vee \neg p(v)$$

Using only resolution:

$$(3): \quad p(y') \vee \neg p(v') \quad \text{by resolving (1) and (2), with } x \setminus u$$

$$(4): \quad \neg p(u') \vee \neg p(v'') \quad \text{by resolving (2) and (3), with } u \setminus y'$$

$$(5): \quad p(x') \vee p(y'') \quad \text{by resolving (1) and (3), with } y \setminus v'$$

$$(6): \quad p(y''') \vee \neg p(v''') \quad \text{by resolving (3) and (3) ...}$$

Using factorisation:

$$(3): \quad p(x') \quad \text{by factoring (1) with } x \setminus y$$

$$(4): \quad \neg p(u') \quad \text{by factoring (2) with } u \setminus v$$

$$(5): \quad \square \quad \text{by resolving (3) and (4) with } x' \setminus u'$$

Outline

- ▶ Reminder: Clausal Form Translations
- ▶ Reminder: Propositional Resolution
- ▶ Reminder: Unification
- ▶ First-Order Resolution
- ▶ **Soundness and Completeness**
- ▶ Compactness
- ▶ Summary

Soundness and Completeness

Theorem 5.1 (Soundness and Completeness of Resolution).

The resolution calculus is sound and complete, i.e.

- ▶ *if A is provable in the resolution calculus, then A is valid
(if $\vdash A$ then $\models A$)*
- ▶ *if A is valid, then A is provable in the resolution calculus
(if $\models A$ then $\vdash A$)*

Proof.

See Ben-Ari, section 10.5, [Robinson 1965].



Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

Soundness

Definition 5.1.

An interpretation $\mathcal{I}$ satisfies a clause C if for every variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Lemma 5.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in S$ to S is also satisfiable.

Proof.

Sketch: if $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$

Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Lemma 5.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in A$ to S is also satisfiable.

Proof.

Sketch: if $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then also $\mathcal{I} \models \sigma(C_1)$ and $\mathcal{I} \models \sigma(C_2)$ (where σ is the m.g.u.)

Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Lemma 5.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in A$ to S is also satisfiable.

Proof.

Sketch: if $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then also $\mathcal{I} \models \sigma(C_1)$ and $\mathcal{I} \models \sigma(C_2)$ (where σ is the m.g.u.) due to

Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Lemma 5.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in A$ to S is also satisfiable.

Proof.

Sketch: if $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then also $\mathcal{I} \models \sigma(C_1)$ and $\mathcal{I} \models \sigma(C_2)$ (where σ is the m.g.u.) due to the

Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Lemma 5.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in A$ to S is also satisfiable.

Proof.

Sketch: if $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then also $\mathcal{I} \models \sigma(C_1)$ and $\mathcal{I} \models \sigma(C_2)$ (where σ is the m.g.u.) due to the s

Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Lemma 5.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in A$ to S is also satisfiable.

Proof.

Sketch: if $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then also $\mathcal{I} \models \sigma(C_1)$ and $\mathcal{I} \models \sigma(C_2)$ (where σ is the m.g.u.) due to the sub

Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Lemma 5.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in A$ to S is also satisfiable.

Proof.

Sketch: if $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then also $\mathcal{I} \models \sigma(C_1)$ and $\mathcal{I} \models \sigma(C_2)$ (where σ is the m.g.u.) due to the substitution lemma.

Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Lemma 5.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in A$ to S is also satisfiable.

Proof.

Sketch: if $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then also $\mathcal{I} \models \sigma(C_1)$ and $\mathcal{I} \models \sigma(C_2)$ (where σ is the m.g.u.) due to the substitution lemma.

Then $\mathcal{I} \models \sigma((C_1 \setminus \{L_1\}) \cup (C_2 \setminus \{\overline{L_2}\}))$

Soundness

Definition 5.1.

*An interpretation $\mathcal{I}$ satisfies a clause C if for **every** variable assignment α , there is a $L \in C$ with $v_{\mathcal{I}}(\alpha, L) = T$.*

So $\mathcal{I} \models \{p(x), q(x)\}$ if either p or q holds for all domain elements.

Lemma 5.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in S$ to S is also satisfiable.

Proof.

Sketch: if $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then also $\mathcal{I} \models \sigma(C_1)$ and $\mathcal{I} \models \sigma(C_2)$ (where σ is the m.g.u.) due to the substitution lemma.

Then $\mathcal{I} \models \sigma((C_1 \setminus \{L_1\}) \cup (C_2 \setminus \{\overline{L_2}\}))$ like for propositional logic. □

Completeness

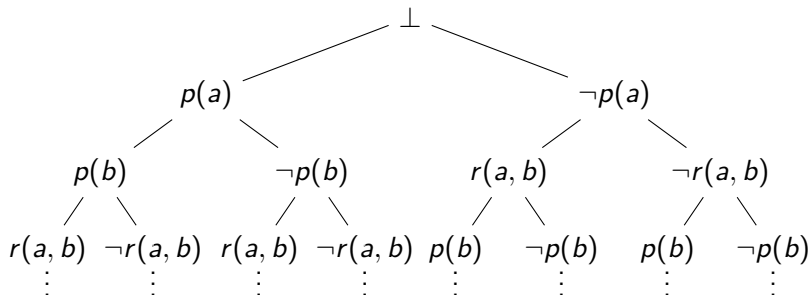
- ▶ Semantic Trees can be infinite

Completeness

- ▶ Semantic Trees can be infinite
- ▶ Define **complete** semantic trees for all closed literals

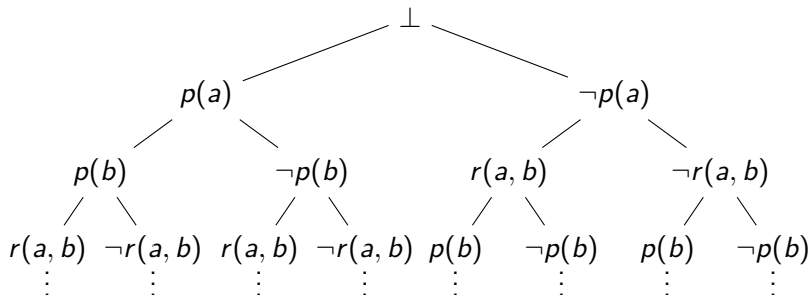
Completeness

- ▶ Semantic Trees can be infinite
- ▶ Define **complete** semantic trees for all closed literals



Completeness

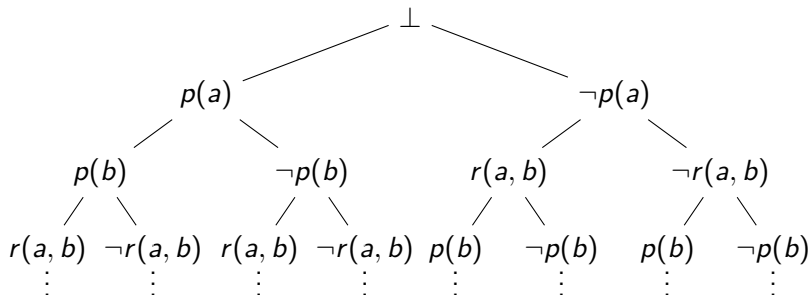
- ▶ Semantic Trees can be infinite
- ▶ Define **complete** semantic trees for all closed literals



- ▶ Same notions of failure nodes and closed semantic trees as before

Completeness

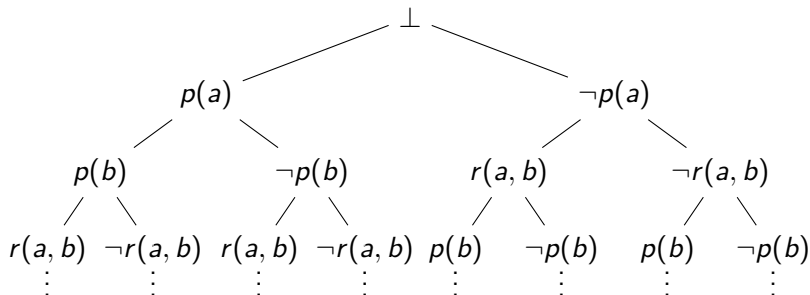
- ▶ Semantic Trees can be infinite
- ▶ Define **complete** semantic trees for all closed literals



- ▶ Same notions of failure nodes and closed semantic trees as before
- ▶ There are resolution steps from *closed instances* of clauses

Completeness

- ▶ Semantic Trees can be infinite
- ▶ Define **complete** semantic trees for all closed literals



- ▶ Same notions of failure nodes and closed semantic trees as before
- ▶ There are resolution steps from *closed instances* of clauses
- ▶ Lifting: There are corresponding steps using m.g.u.s

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint.

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$.

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11}$

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$.

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$. And $\bar{L} = \theta_2 L_{21} = \theta_2 L_{22} = \dots$ for $\{L_{21}, \dots\} \subseteq C_2$.

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$. And $\bar{L} = \theta_2 L_{21} = \theta_2 L_{22} = \dots$ for $\{L_{21}, \dots\} \subseteq C_2$. Let $\lambda_1 = \text{mgu}(L_{11}, L_{12}, \dots)$ and $\lambda_2 = \text{mgu}(L_{21}, L_{22}, \dots)$.

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$. And $\bar{L} = \theta_2 L_{21} = \theta_2 L_{22} = \dots$ for $\{L_{21}, \dots\} \subseteq C_2$. Let $\lambda_1 = \text{mgu}(L_{11}, L_{12}, \dots)$ and $\lambda_2 = \text{mgu}(L_{21}, L_{22}, \dots)$. λ_i is more general than θ_i . Let $\lambda = \lambda_1 \cup \lambda_2$.

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$. And $\bar{L} = \theta_2 L_{21} = \theta_2 L_{22} = \dots$ for $\{L_{21}, \dots\} \subseteq C_2$. Let $\lambda_1 = \text{mgu}(L_{11}, L_{12}, \dots)$ and $\lambda_2 = \text{mgu}(L_{21}, L_{22}, \dots)$. λ_i is more general than θ_i . Let $\lambda = \lambda_1 \cup \lambda_2$.

► If more than one L_{1j} , apply factoring to get $\lambda_1 C_1$

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$. And $\bar{L} = \theta_2 L_{21} = \theta_2 L_{22} = \dots$ for $\{L_{21}, \dots\} \subseteq C_2$. Let $\lambda_1 = \text{mgu}(L_{11}, L_{12}, \dots)$ and $\lambda_2 = \text{mgu}(L_{21}, L_{22}, \dots)$. λ_i is more general than θ_i . Let $\lambda = \lambda_1 \cup \lambda_2$.

► If more than one L_{1j} , apply factoring to get $\lambda_1 C_1 = \lambda C_1$.

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$. And $\bar{L} = \theta_2 L_{21} = \theta_2 L_{22} = \dots$ for $\{L_{21}, \dots\} \subseteq C_2$. Let $\lambda_1 = \text{mgu}(L_{11}, L_{12}, \dots)$ and $\lambda_2 = \text{mgu}(L_{21}, L_{22}, \dots)$. λ_i is more general than θ_i . Let $\lambda = \lambda_1 \cup \lambda_2$.

- ▶ If more than one L_{1j} , apply factoring to get $\lambda_1 C_1 = \lambda C_1$.
- ▶ If more than one L_{2j} , apply factoring to get $\lambda_2 C_2 = \lambda C_2$.

Now $\lambda L_{11} \in \lambda C_1$ and $\lambda \bar{L}_{21} \in \lambda C_2$ are unifiable

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$. And $\bar{L} = \theta_2 L_{21} = \theta_2 L_{22} = \dots$ for $\{L_{21}, \dots\} \subseteq C_2$. Let $\lambda_1 = \text{mgu}(L_{11}, L_{12}, \dots)$ and $\lambda_2 = \text{mgu}(L_{21}, L_{22}, \dots)$. λ_i is more general than θ_i . Let $\lambda = \lambda_1 \cup \lambda_2$.

- ▶ If more than one L_{1j} , apply factoring to get $\lambda_1 C_1 = \lambda C_1$.
- ▶ If more than one L_{2j} , apply factoring to get $\lambda_2 C_2 = \lambda C_2$.

Now $\lambda L_{11} \in \lambda C_1$ and $\lambda \bar{L}_{21} \in \lambda C_2$ are unifiable with mgu σ .

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$. And $\bar{L} = \theta_2 L_{21} = \theta_2 L_{22} = \dots$ for $\{L_{21}, \dots\} \subseteq C_2$. Let $\lambda_1 = \text{mgu}(L_{11}, L_{12}, \dots)$ and $\lambda_2 = \text{mgu}(L_{21}, L_{22}, \dots)$. λ_i is more general than θ_i . Let $\lambda = \lambda_1 \cup \lambda_2$.

- ▶ If more than one L_{1j} , apply factoring to get $\lambda_1 C_1 = \lambda C_1$.
- ▶ If more than one L_{2j} , apply factoring to get $\lambda_2 C_2 = \lambda C_2$.

Now $\lambda L_{11} \in \lambda C_1$ and $\lambda \bar{L}_{21} \in \lambda C_2$ are unifiable with mgu σ .

- ▶ There is a resolution step between λC_1 and λC_2 with mgu σ .

The Lifting Lemma

Lemma 5.2 (Lifting Lemma).

Let C'_1 and C'_2 be ground instances of C_1 , C_2 , respectively. Let C' be a ground resolvent of C'_1 and C'_2 . Then there is a resolution derivation of C of C_1 and C_2 such that C' is a ground instance of C .

Proof sketch. Make variables in C_1 and C_2 disjoint. Let $L \in C'_1 = \theta_1 C_1$, $\bar{L} \in C'_2 = \theta_2 C_2$. Then $L = \theta_1 L_{11} = \theta_1 L_{12} = \dots$ for $\{L_{11}, \dots\} \subseteq C_1$. And $\bar{L} = \theta_2 L_{21} = \theta_2 L_{22} = \dots$ for $\{L_{21}, \dots\} \subseteq C_2$. Let $\lambda_1 = \text{mgu}(L_{11}, L_{12}, \dots)$ and $\lambda_2 = \text{mgu}(L_{21}, L_{22}, \dots)$. λ_i is more general than θ_i . Let $\lambda = \lambda_1 \cup \lambda_2$.

► If more than one L_{1j} , apply factoring to get $\lambda_1 C_1 = \lambda C_1$.

► If more than one L_{2j} , apply factoring to get $\lambda_2 C_2 = \lambda C_2$.

Now $\lambda L_{11} \in \lambda C_1$ and $\lambda \bar{L}_{21} \in \lambda C_2$ are unifiable with mgu σ .

► There is a resolution step between λC_1 and λC_2 with mgu σ .

(Still need to show that C' is a ground instance of the resolvent.)

Outline

- ▶ Reminder: Clausal Form Translations
- ▶ Reminder: Propositional Resolution
- ▶ Reminder: Unification
- ▶ First-Order Resolution
- ▶ Soundness and Completeness
- ▶ **Compactness**
- ▶ Summary

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable there is a closed semantic tree

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable there is a closed semantic tree which enables a resolution step

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable there is a closed semantic tree which enables a resolution step that gives a smaller semantic tree.

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable there is a closed semantic tree which enables a resolution step that gives a smaller semantic tree.
- ▶ No need to use *all* of S

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable there is a closed semantic tree which enables a resolution step that gives a smaller semantic tree.
- ▶ No need to use *all* of S
- ▶ The closed tree is always finite (König's Lemma)

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable there is a closed semantic tree which enables a resolution step that gives a smaller semantic tree.
- ▶ No need to use *all* of S
- ▶ The closed tree is always finite (König's Lemma)
- ▶ To close the semantic tree we need only finitely many clauses $S' \subseteq S$.

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable there is a closed semantic tree which enables a resolution step that gives a smaller semantic tree.
- ▶ No need to use *all* of S
- ▶ The closed tree is always finite (König's Lemma)
- ▶ To close the semantic tree we need only finitely many clauses $S' \subseteq S$.
- ▶ Collect all clauses $S_0 \subseteq S$ that are used in a refutation

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable there is a closed semantic tree which enables a resolution step that gives a smaller semantic tree.
- ▶ No need to use *all* of S
- ▶ The closed tree is always finite (König's Lemma)
- ▶ To close the semantic tree we need only finitely many clauses $S' \subseteq S$.
- ▶ Collect all clauses $S_0 \subseteq S$ that are used in a refutation
- ▶ $S_0 \subseteq S$ is finite and unsatisfiable

Compactness

Observation

Nowhere in the definition of resolution do we need that S is finite.

- ▶ If S is unsatisfiable there is a closed semantic tree which enables a resolution step that gives a smaller semantic tree.
- ▶ No need to use *all* of S
- ▶ The closed tree is always finite (König's Lemma)
- ▶ To close the semantic tree we need only finitely many clauses $S' \subseteq S$.
- ▶ Collect all clauses $S_0 \subseteq S$ that are used in a refutation
- ▶ $S_0 \subseteq S$ is finite and unsatisfiable

Theorem 6.1 (Compactness).

Every unsatisfiable set of clauses S has a finite unsatisfiable subset

Compactness: Example

$$\exists x \neg p(x), p(a), p(fa), p(ffa), p(ffa), \dots$$

Compactness: Example

$$\exists x \neg p(x), p(a), p(fa), p(ffa), p(ffa), \dots$$

- ▶ Every finite subset is satisfiable.

Compactness: Example

$$\exists x \neg p(x), p(a), p(fa), p(ffa), p(fffa), \dots$$

- ▶ Every finite subset is satisfiable.
- ▶ E.g., let n be the largest n with $p(f^n a)$ in the set.

Compactness: Example

$$\exists x \neg p(x), p(a), p(fa), p(ffa), p(fffa), \dots$$

- ▶ Every finite subset is satisfiable.
- ▶ E.g., let n be the largest n with $p(f^n a)$ in the set.
- ▶ Interpret p such that $p'(f^{n+1}a) = F$, and therefore $\mathcal{I} \models \exists x \neg p(x)$.

Compactness: Example

$$\exists x \neg p(x), p(a), p(fa), p(ffa), p(fffa), \dots$$

- ▶ Every finite subset is satisfiable.
- ▶ E.g., let n be the largest n with $p(f^n a)$ in the set.
- ▶ Interpret p such that $p^{\iota}(f^{n+1}a) = F$, and therefore $\mathcal{I} \models \exists x \neg p(x)$.
- ▶ By compactness: The whole set is also satisfiable
- ▶ Note: the same construction of a satisfying model does not work!

Compactness: Counterexample

- Now we look at satisfiability 'over $\mathbb{N}$ '

Compactness: Counterexample

- ▶ Now we look at satisfiability 'over $\mathbb{N}$ '
- ▶ i.e. in interpretations with $D = \mathbb{N}$, $0^I = 0$, $1^I = 1, \dots$

Compactness: Counterexample

- ▶ Now we look at satisfiability 'over $\mathbb{N}$ '
- ▶ i.e. in interpretations with $D = \mathbb{N}$, $0^I = 0$, $1^I = 1, \dots$

$$\exists x \neg p(x), \quad p(0), \quad p(1), \quad p(2), \quad p(3), \quad \dots$$

Compactness: Counterexample

- ▶ Now we look at satisfiability 'over $\mathbb{N}$ '
- ▶ i.e. in interpretations with $D = \mathbb{N}$, $0^I = 0$, $1^I = 1, \dots$
 $\exists x \neg p(x), p(0), p(1), p(2), p(3), \dots$
- ▶ Every finite subset $S_0 \subseteq S$ is satisfiable over $\mathbb{N}$.

Compactness: Counterexample

- ▶ Now we look at satisfiability 'over $\mathbb{N}$ '
- ▶ i.e. in interpretations with $D = \mathbb{N}$, $0^I = 0$, $1^I = 1, \dots$

$$\exists x \neg p(x), p(0), p(1), p(2), p(3), \dots$$

- ▶ Every finite subset $S_0 \subseteq S$ is satisfiable over $\mathbb{N}$.
- ▶ E.g. let n be maximal with $p(n) \in S_0$

Compactness: Counterexample

- ▶ Now we look at satisfiability 'over $\mathbb{N}$ '
- ▶ i.e. in interpretations with $D = \mathbb{N}$, $0^I = 0$, $1^I = 1, \dots$

$$\exists x \neg p(x), p(0), p(1), p(2), p(3), \dots$$

- ▶ Every finite subset $S_0 \subseteq S$ is satisfiable over $\mathbb{N}$.
- ▶ E.g. let n be maximal with $p(n) \in S_0$
- ▶ Interpret $p(0) \dots p(n)$ as true, but $p(n+1)$ as false.

Compactness: Counterexample

- ▶ Now we look at satisfiability 'over $\mathbb{N}$ '
- ▶ i.e. in interpretations with $D = \mathbb{N}$, $0^{\iota} = 0$, $1^{\iota} = 1, \dots$
 $\exists x \neg p(x)$, $p(0)$, $p(1)$, $p(2)$, $p(3)$, $\dots$
- ▶ Every finite subset $S_0 \subseteq S$ is satisfiable over $\mathbb{N}$.
- ▶ E.g. let n be maximal with $p(n) \in S_0$
- ▶ Interpret $p(0) \dots p(n)$ as true, but $p(n+1)$ as false.
- ▶ Then all $p(\dots) \in S_0$ are satisfied and also $\exists x \neg p(x)$.

Compactness: Counterexample

- ▶ Now we look at satisfiability 'over $\mathbb{N}$ '
- ▶ i.e. in interpretations with $D = \mathbb{N}$, $0^I = 0$, $1^I = 1, \dots$
 $\exists x \neg p(x)$, $p(0)$, $p(1)$, $p(2)$, $p(3)$, $\dots$
- ▶ Every finite subset $S_0 \subseteq S$ is satisfiable over $\mathbb{N}$.
- ▶ E.g. let n be maximal with $p(n) \in S_0$
- ▶ Interpret $p(0) \dots p(n)$ as true, but $p(n+1)$ as false.
- ▶ Then all $p(\dots) \in S_0$ are satisfied and also $\exists x \neg p(x)$.
- ▶ But the whole set of formulas is **unsatisfiable over $\mathbb{N}$**

Compactness: Counterexample

- ▶ Now we look at satisfiability 'over $\mathbb{N}$ '
- ▶ i.e. in interpretations with $D = \mathbb{N}$, $0^\iota = 0$, $1^\iota = 1, \dots$

$$\exists x \neg p(x), p(0), p(1), p(2), p(3), \dots$$
- ▶ Every finite subset $S_0 \subseteq S$ is satisfiable over $\mathbb{N}$.
- ▶ E.g. let n be maximal with $p(n) \in S_0$
- ▶ Interpret $p(0) \dots p(n)$ as true, but $p(n+1)$ as false.
- ▶ Then all $p(\dots) \in S_0$ are satisfied and also $\exists x \neg p(x)$.
- ▶ But the whole set of formulas is **unsatisfiable over $\mathbb{N}$**

Theorem 6.2.

*Satisfiability over the natural numbers is **not** compact.*

Compactness: Counterexample

- ▶ Now we look at satisfiability 'over $\mathbb{N}$ '
- ▶ i.e. in interpretations with $D = \mathbb{N}$, $0^I = 0$, $1^I = 1, \dots$

$$\exists x \neg p(x), p(0), p(1), p(2), p(3), \dots$$
- ▶ Every finite subset $S_0 \subseteq S$ is satisfiable over $\mathbb{N}$.
- ▶ E.g. let n be maximal with $p(n) \in S_0$
- ▶ Interpret $p(0) \dots p(n)$ as true, but $p(n+1)$ as false.
- ▶ Then all $p(\dots) \in S_0$ are satisfied and also $\exists x \neg p(x)$.
- ▶ But the whole set of formulas is **unsatisfiable over $\mathbb{N}$**

Theorem 6.2.

*Satisfiability over the natural numbers is **not** compact.*

Reasoning about numbers involves more than just first-order logic.

Outline

- ▶ Reminder: Clausal Form Translations
- ▶ Reminder: Propositional Resolution
- ▶ Reminder: Unification
- ▶ First-Order Resolution
- ▶ Soundness and Completeness
- ▶ Compactness
- ▶ **Summary**

Summary

- ▶ **resolution calculus** is one of the most popular proof search calculi for (classical) first-order logic

Summary

- ▶ **resolution calculus** is one of the most popular proof search calculi for (classical) first-order logic
- ▶ consists of:
 - ▶ one axiom
 - ▶ resolution rule
 - ▶ factoring rule

Summary

- ▶ **resolution calculus** is one of the most popular proof search calculi for (classical) first-order logic
- ▶ consists of:
 - ▶ one axiom
 - ▶ resolution rule
 - ▶ factoring rule
- ▶ **unification** is used to unify terms of complementary literals

Summary

- ▶ **resolution calculus** is one of the most popular proof search calculi for (classical) first-order logic
- ▶ consists of:
 - ▶ one axiom
 - ▶ resolution rule
 - ▶ factoring rule
- ▶ **unification** is used to unify terms of complementary literals
- ▶ easy to implement, but for an **efficient proof search** the application of the resolution rule needs to be controlled
- ▶ implemented in popular **automated theorem provers**, e.g. Otter, Prover9, Vampire

Summary

- ▶ **resolution calculus** is one of the most popular proof search calculi for (classical) first-order logic
- ▶ consists of:
 - ▶ one axiom
 - ▶ resolution rule
 - ▶ factoring rule
- ▶ **unification** is used to unify terms of complementary literals
- ▶ easy to implement, but for an **efficient proof search** the application of the resolution rule needs to be controlled
- ▶ implemented in popular **automated theorem provers**, e.g. Otter, Prover9, Vampire
- ▶ **Compactness**: we can reason over (countably) infinite clause sets, but 1st-order reasoning is not strong enough for all of maths

Summary

- ▶ **resolution calculus** is one of the most popular proof search calculi for (classical) first-order logic
- ▶ consists of:
 - ▶ one axiom
 - ▶ resolution rule
 - ▶ factoring rule
- ▶ **unification** is used to unify terms of complementary literals
- ▶ easy to implement, but for an **efficient proof search** the application of the resolution rule needs to be controlled
- ▶ implemented in popular **automated theorem provers**, e.g. Otter, Prover9, Vampire
- ▶ **Compactness**: we can reason over (countably) infinite clause sets, but 1st-order reasoning is not strong enough for all of maths
- ▶ **Next Week**: logic programming and Prolog