

# First Order Logic

Mădălina Eraşcu Tudor Jebelean

# Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Formula Classification

Substitution

# Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Formula Classification

Substitution

# Syntax

The language of FOL consists in **terms** and **formulae**.

**Terms** are defined recursively as follows:

1. A constant is a term.
2. A variable is a term.
3. If  $f$  is an  $n$ -place function symbol, and  $t_1, \dots, t_n$  are terms then  $f[t_1, \dots, t_n]$  is a term.
4. All terms are generated by applying the above rules.

If  $P$  is an  $n$ -place predicate symbol and  $t_1, \dots, t_n$  are terms then  $P[t_1, \dots, t_n]$  is an atom.

An **atom** is  $\mathbb{T}$ ,  $\mathbb{F}$ , or an  $n$ -ary predicate applied to  $n$  terms.

A **literal** is an atom or its negation.

# Syntax

The language of FOL consists in **terms** and **formulae**.

**Terms** are defined recursively as follows:

1. A constant is a term.
2. A variable is a term.
3. If  $f$  is an  $n$ -place function symbol, and  $t_1, \dots, t_n$  are terms then  $f[t_1, \dots, t_n]$  is a term.
4. All terms are generated by applying the above rules.

If  $P$  is an  $n$ -place predicate symbol and  $t_1, \dots, t_n$  are terms then  $P[t_1, \dots, t_n]$  is an atom.

An **atom** is  $\mathbb{T}$ ,  $\mathbb{F}$ , or an  $n$ -ary predicate applied to  $n$  terms.

A **literal** is an atom or its negation.

# Syntax

The language of FOL consists in **terms** and **formulae**.

**Terms** are defined recursively as follows:

1. A constant is a term.
2. A variable is a term.
3. If  $f$  is an  $n$ -place function symbol, and  $t_1, \dots, t_n$  are terms then  $f[t_1, \dots, t_n]$  is a term.
4. All terms are generated by applying the above rules.

If  $P$  is an  $n$ -place predicate symbol and  $t_1, \dots, t_n$  are terms then  $P[t_1, \dots, t_n]$  is an atom.

An **atom** is  $\mathbb{T}$ ,  $\mathbb{F}$ , or an  $n$ -ary predicate applied to  $n$  terms.

A **literal** is an atom or its negation.

# Syntax

The language of FOL consists in **terms** and **formulae**.

**Terms** are defined recursively as follows:

1. A constant is a term.
2. A variable is a term.
3. If  $f$  is an  $n$ -place function symbol, and  $t_1, \dots, t_n$  are terms then  $f[t_1, \dots, t_n]$  is a term.
4. All terms are generated by applying the above rules.

If  $P$  is an  $n$ -place predicate symbol and  $t_1, \dots, t_n$  are terms then  $P[t_1, \dots, t_n]$  is an atom.

An **atom** is  $\mathbb{T}$ ,  $\mathbb{F}$ , or an  $n$ -ary predicate applied to  $n$  terms.

A **literal** is an atom or its negation.

# Syntax

The language of FOL consists in **terms** and **formulae**.

**Terms** are defined recursively as follows:

1. A constant is a term.
2. A variable is a term.
3. If  $f$  is an  $n$ -place function symbol, and  $t_1, \dots, t_n$  are terms then  $f[t_1, \dots, t_n]$  is a term.
4. All terms are generated by applying the above rules.

If  $P$  is an  $n$ -place predicate symbol and  $t_1, \dots, t_n$  are terms then  $P[t_1, \dots, t_n]$  is an atom.

An **atom** is  $\mathbb{T}$ ,  $\mathbb{F}$ , or an  $n$ -ary predicate applied to  $n$  terms.

A **literal** is an atom or its negation.



# Syntax

The language of FOL consists in **terms** and **formulae**.

**Terms** are defined recursively as follows:

1. A constant is a term.
2. A variable is a term.
3. If  $f$  is an  $n$ -place function symbol, and  $t_1, \dots, t_n$  are terms then  $f[t_1, \dots, t_n]$  is a term.
4. All terms are generated by applying the above rules.

If  $P$  is an  $n$ -place predicate symbol and  $t_1, \dots, t_n$  are terms then  $P[t_1, \dots, t_n]$  is an atom.

An **atom** is  $\mathbb{T}$ ,  $\mathbb{F}$ , or an  $n$ -ary predicate applied to  $n$  terms.

A **literal** is an atom or its negation.

# Syntax

The language of FOL consists in **terms** and **formulae**.

**Terms** are defined recursively as follows:

1. A constant is a term.
2. A variable is a term.
3. If  $f$  is an  $n$ -place function symbol, and  $t_1, \dots, t_n$  are terms then  $f[t_1, \dots, t_n]$  is a term.
4. All terms are generated by applying the above rules.

If  $P$  is an  $n$ -place predicate symbol and  $t_1, \dots, t_n$  are terms then  $P[t_1, \dots, t_n]$  is an atom.

An **atom** is  $\mathbb{T}$ ,  $\mathbb{F}$ , or an  $n$ -ary predicate applied to  $n$  terms.

A **literal** is an atom or its negation.

# Syntax

The language of FOL consists in **terms** and **formulae**.

**Terms** are defined recursively as follows:

1. A constant is a term.
2. A variable is a term.
3. If  $f$  is an  $n$ -place function symbol, and  $t_1, \dots, t_n$  are terms then  $f[t_1, \dots, t_n]$  is a term.
4. All terms are generated by applying the above rules.

If  $P$  is an  $n$ -place predicate symbol and  $t_1, \dots, t_n$  are terms then  $P[t_1, \dots, t_n]$  is an atom.

An **atom** is  $\mathbb{T}$ ,  $\mathbb{F}$ , or an  $n$ -ary predicate applied to  $n$  terms.

A **literal** is an atom or its negation.

# Syntax

The language of FOL consists in **terms** and **formulae**.

**Terms** are defined recursively as follows:

1. A constant is a term.
2. A variable is a term.
3. If  $f$  is an  $n$ -place function symbol, and  $t_1, \dots, t_n$  are terms then  $f[t_1, \dots, t_n]$  is a term.
4. All terms are generated by applying the above rules.

If  $P$  is an  $n$ -place predicate symbol and  $t_1, \dots, t_n$  are terms then  $P[t_1, \dots, t_n]$  is an atom.

An **atom** is  $\mathbb{T}$ ,  $\mathbb{F}$ , or an  $n$ -ary predicate applied to  $n$  terms.

A **literal** is an atom or its negation.

# Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$
2.  $\neg (\exists_x E[0, f[x]])$
3.  $\forall_x \exists_y (E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]))$

## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$
2.  $\neg (\exists_x E[0, f[x]])$
3.  $\forall_x \exists_y (E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]))$

## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$
2.  $\neg (\exists_x E[0, f[x]])$
3.  $\forall_x \exists_y (E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]))$

## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$
2.  $\neg (\exists_x E[0, f[x]])$
3.  $\forall_x \exists_y (E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]))$



## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$

2.  $\neg (\exists_x E[0, f[x]])$

3.  $\forall_x \exists_y (E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]))$

## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the below

1.  $\forall_x x + 1 \geq x$

2.  $\neg (\exists_x E[0, f[x]])$

3.  $\forall_x \exists_y (E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]))$

## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$

2.  $\neg (\exists_x E[0, f[x]])$

3.  $\forall_x \exists_y (E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]))$

## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$
2.  $\neg \left( \exists_x E[0, f[x]] \right)$
3.  $\forall_x \exists_y \left( E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]) \right)$

## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$
2.  $\neg \left( \exists_x E[0, f[x]] \right)$
3.  $\forall_x \exists_y \left( E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]) \right)$

## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$
2.  $\neg \left( \exists_x E[0, f[x]] \right)$
3.  $\forall_x \exists_y \left( E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]) \right)$

## Syntax (cont'd)

**Formulae** are defined as follows:

1. An atom is a formula.
2. If  $F$  and  $G$  are formulae then  $\neg F$ ,  $F \vee G$ ,  $F \wedge G$ ,  $F \implies G$ , and  $F \iff G$  are formulae.
3. If  $F$  is a formula and  $x$  is a variable, then  $\forall_x F$  and  $\exists_x F$  are formulae.
4. Formulae are generated only by a finite number of applications of the above rules.

A variable  $x$  is **bound** in the formula  $F$  if there is an occurrence of  $x$  in the scope of a binding quantifier  $\forall_x$  or  $\exists_x$ .

A variable  $x$  is **free** in the formula  $F$  if there is an occurrence of  $x$  that is not bound by any quantifier.

**Examples:** Identify constants, variables (free, bound), quantifiers, functions, predicates, atoms, terms, formulae from the bellow

1.  $\forall_x x + 1 \geq x$
2.  $\neg \left( \exists_x E[0, f[x]] \right)$
3.  $\forall_x \exists_y \left( E[y, f[x]] \wedge \forall_z (E[z, f[x]] \implies E[y, z]) \right)$

# Outline

Syntax

**Semantics**

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Formula Classification

Substitution



# Semantics

An **interpretation**  $I$  of a formula  $F$  in FOL consists of a nonempty domain  $D$  and an assignment of values to each constant, function, symbol and predicate symbol occurring in  $F$  as follows:

- ▶ to each constant we assign an element in  $D$
- ▶ to each function symbol we assign a mapping from  $D^n$  to  $D$
- ▶ to each predicate symbol we assign a mapping from  $D^n$  to  $\{\mathbf{T}, \mathbf{F}\}$ .

Then the semantics of the formula  $F$  is a function  $f : \mathcal{I} \rightarrow \{\mathbf{T}, \mathbf{F}\}$ , where  $I \in \mathcal{I}$  and  $\mathcal{I}$  is the set of all interpretations of the formula  $F$ .

# Semantics

An **interpretation**  $I$  of a formula  $F$  in FOL consists of a nonempty domain  $D$  and an assignment of values to each constant, function, symbol and predicate symbol occurring in  $F$  as follows:

- ▶ to each constant we assign an element in  $D$
- ▶ to each function symbol we assign a mapping from  $D^n$  to  $D$
- ▶ to each predicate symbol we assign a mapping from  $D^n$  to  $\{\mathbf{T}, \mathbf{F}\}$ .

Then the semantics of the formula  $F$  is a function  $f : \mathcal{I} \rightarrow \{\mathbf{T}, \mathbf{F}\}$ , where  $I \in \mathcal{I}$  and  $\mathcal{I}$  is the set of all interpretations of the formula  $F$ .

# Semantics

An **interpretation**  $I$  of a formula  $F$  in FOL consists of a nonempty domain  $D$  and an assignment of values to each constant, function, symbol and predicate symbol occurring in  $F$  as follows:

- ▶ to each constant we assign an element in  $D$
- ▶ to each function symbol we assign a mapping from  $D^n$  to  $D$
- ▶ to each predicate symbol we assign a mapping from  $D^n$  to  $\{\mathbf{T}, \mathbf{F}\}$ .

Then the semantics of the formula  $F$  is a function  $f : \mathcal{I} \rightarrow \{\mathbf{T}, \mathbf{F}\}$ , where  $I \in \mathcal{I}$  and  $\mathcal{I}$  is the set of all interpretations of the formula  $F$ .

# Semantics

An **interpretation**  $I$  of a formula  $F$  in FOL consists of a nonempty domain  $D$  and an assignment of values to each constant, function, symbol and predicate symbol occurring in  $F$  as follows:

- ▶ to each constant we assign an element in  $D$
- ▶ to each function symbol we assign a mapping from  $D^n$  to  $D$
- ▶ to each predicate symbol we assign a mapping from  $D^n$  to  $\{\mathbb{T}, \mathbb{F}\}$ .

Then the semantics of the formula  $F$  is a function  $f : \mathcal{I} \rightarrow \{\mathbb{T}, \mathbb{F}\}$ , where  $I \in \mathcal{I}$  and  $\mathcal{I}$  is the set of all interpretations of the formula  $F$ .

# Semantics

An **interpretation**  $I$  of a formula  $F$  in FOL consists of a nonempty domain  $D$  and an assignment of values to each constant, function, symbol and predicate symbol occurring in  $F$  as follows:

- ▶ to each constant we assign an element in  $D$
- ▶ to each function symbol we assign a mapping from  $D^n$  to  $D$
- ▶ to each predicate symbol we assign a mapping from  $D^n$  to  $\{\mathbb{T}, \mathbb{F}\}$ .

Then the semantics of the formula  $F$  is a function  $f : \mathcal{I} \rightarrow \{\mathbb{T}, \mathbb{F}\}$ , where  $I \in \mathcal{I}$  and  $\mathcal{I}$  is the set of all interpretations of the formula  $F$ .

## Semantics (cont'd)

Example: Find the truth value of the formulae:

$$\blacktriangleright F_1 : \iff \forall_x \forall_y x \leq y, \text{ where } I : \begin{cases} D = \{0, 1\} \\ \leq_I \rightarrow \leq_{\mathbb{Z}} \end{cases}$$

$$\blacktriangleright F_2 : \iff \forall_x \exists_y x + y > c, \text{ where } I : \begin{cases} D = \{0, 1\} \\ c_I = 0 \\ +_I \rightarrow +_{\mathbb{Z}} \\ >_I \rightarrow >_{\mathbb{Z}} \end{cases}$$

$$\blacktriangleright F_3 : \iff \forall_x (P[x] \implies Q[f[x], a]), \text{ where}$$

$$I : \begin{cases} D = \{1, 2\} \\ a_I = 1 \\ f_I : D \rightarrow D \\ P_I : D \rightarrow \{\mathbb{T}, \mathbb{F}\} \\ Q_I : D^2 \rightarrow \{\mathbb{T}, \mathbb{F}\} \end{cases} \begin{cases} f_I[1] = 1 \\ f_I[2] = 1 \\ P_I[1] = \mathbb{T} \\ P_I[2] = \mathbb{F} \\ Q_I[1, 1] = \mathbb{T} & Q_I[1, 2] = \mathbb{F} \\ Q_I[2, 1] = \mathbb{F} & Q_I[2, 2] = \mathbb{T} \end{cases}$$

# Outline

Syntax

Semantics

**(Un)Satisfiability & (In)Validity**

Equivalences of Formulas

Normal Forms

Formula Classification

Substitution

# (Un)Satisfiability & (In)Validity

A formula  $F$  is **satisfiable (consistent)** iff there exists an interpretation  $I$  such that  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **unsatisfiable (inconsistent)** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $F$  is **valid** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **invalid** iff there exists an interpretation  $I$ , such that  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $G$  is a **logical consequence** of formulae  $F_1, F_2, \dots, F_n$  iff for every interpretation  $I$ , if  $F_1 \wedge F_2 \wedge \dots \wedge F_n$  is true in  $I$ ,  $G$  is also true in  $I$ .

Note that validity and satisfiability applies to closed formulae.

Examples: Prove that

▶  $\forall_x P[x] \wedge \exists_y \neg P[y]$  is inconsistent.



# (Un)Satisfiability & (In)Validity

A formula  $F$  is **satisfiable (consistent)** iff there exists an interpretation  $I$  such that  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **unsatisfiable (inconsistent)** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $F$  is **valid** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **invalid** iff there exists an interpretation  $I$ , such that  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $G$  is a **logical consequence** of formulae  $F_1, F_2, \dots, F_n$  iff for every interpretation  $I$ , if  $F_1 \wedge F_2 \wedge \dots \wedge F_n$  is true in  $I$ ,  $G$  is also true in  $I$ .

Note that validity and satisfiability applies to closed formulae.

Examples: Prove that

▶  $\forall_x P[x] \wedge \exists_y \neg P[y]$  is inconsistent.

# (Un)Satisfiability & (In)Validity

A formula  $F$  is **satisfiable (consistent)** iff there exists an interpretation  $I$  such that  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **unsatisfiable (inconsistent)** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $F$  is **valid** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **invalid** iff there exists an interpretation  $I$ , such that  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $G$  is a **logical consequence** of formulae  $F_1, F_2, \dots, F_n$  iff for every interpretation  $I$ , if  $F_1 \wedge F_2 \wedge \dots \wedge F_n$  is true in  $I$ ,  $G$  is also true in  $I$ .

Note that validity and satisfiability applies to closed formulae.

Examples: Prove that

▶  $\forall_x P[x] \wedge \exists_y \neg P[y]$  is inconsistent.

# (Un)Satisfiability & (In)Validity

A formula  $F$  is **satisfiable (consistent)** iff there exists an interpretation  $I$  such that  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **unsatisfiable (inconsistent)** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $F$  is **valid** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **invalid** iff there exists an interpretation  $I$ , such that  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $G$  is a **logical consequence** of formulae  $F_1, F_2, \dots, F_n$  iff for every interpretation  $I$ , if  $F_1 \wedge F_2 \wedge \dots \wedge F_n$  is true in  $I$ ,  $G$  is also true in  $I$ .

Note that validity and satisfiability applies to closed formulae.

Examples: Prove that

▶  $\forall_x P[x] \wedge \exists_y \neg P[y]$  is inconsistent.

# (Un)Satisfiability & (In)Validity

A formula  $F$  is **satisfiable (consistent)** iff there exists an interpretation  $I$  such that  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **unsatisfiable (inconsistent)** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $F$  is **valid** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **invalid** iff there exists an interpretation  $I$ , such that  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $G$  is a **logical consequence** of formulae  $F_1, F_2, \dots, F_n$  iff for every interpretation  $I$ , if  $F_1 \wedge F_2 \wedge \dots \wedge F_n$  is true in  $I$ ,  $G$  is also true in  $I$ .

Note that validity and satisfiability applies to closed formulae.

Examples: Prove that

- ▶  $\forall_x P[x] \wedge \exists_y \neg P[y]$  is inconsistent.

# (Un)Satisfiability & (In)Validity

A formula  $F$  is **satisfiable (consistent)** iff there exists an interpretation  $I$  such that  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **unsatisfiable (inconsistent)** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $F$  is **valid** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **invalid** iff there exists an interpretation  $I$ , such that  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $G$  is a **logical consequence** of formulae  $F_1, F_2, \dots, F_n$  iff for every interpretation  $I$ , if  $F_1 \wedge F_2 \wedge \dots \wedge F_n$  is true in  $I$ ,  $G$  is also true in  $I$ .

**Note that validity and satisfiability applies to closed formulae.**

Examples: Prove that

- ▶  $\forall_x P[x] \wedge \exists_y \neg P[y]$  is inconsistent.

# (Un)Satisfiability & (In)Validity

A formula  $F$  is **satisfiable (consistent)** iff there exists an interpretation  $I$  such that  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **unsatisfiable (inconsistent)** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $F$  is **valid** iff for all interpretations  $I$ ,  $F$  is evaluated to  $\mathbb{T}$  in  $I$ .

A formula  $F$  is **invalid** iff there exists an interpretation  $I$ , such that  $F$  is evaluated to  $\mathbb{F}$  in  $I$ .

A formula  $G$  is a **logical consequence** of formulae  $F_1, F_2, \dots, F_n$  iff for every interpretation  $I$ , if  $F_1 \wedge F_2 \wedge \dots \wedge F_n$  is true in  $I$ ,  $G$  is also true in  $I$ .

**Note that validity and satisfiability applies to closed formulae.**

**Examples:** Prove that

- ▶  $\forall_x P[x] \wedge \exists_y \neg P[y]$  is inconsistent.

# Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

**Equivalences of Formulas**

Normal Forms

Formula Classification

Substitution

# Equivalences of Formulae

Two formulae  $F$  and  $G$  are **equivalent** iff the truth values of  $F$  and  $G$  are the same under any interpretation.

$$F \iff G \equiv (F \Rightarrow G) \wedge (G \Rightarrow F)$$

$$F \Rightarrow G \equiv \neg F \vee G$$

$$F \vee G \equiv G \vee F$$

$$(F \vee G) \vee H \equiv F \vee (G \vee H)$$

$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$$

$$F \vee \mathbb{T} \equiv \mathbb{T}$$

$$F \vee \mathbb{F} \equiv F$$

$$F \vee \neg F \equiv \mathbb{T}$$

$$\neg(\neg F) \equiv F$$

$$\neg(F \vee G) \equiv \neg F \wedge \neg G$$

$$(Qx)F[x] \vee G \equiv (Qx)(F[x] \vee G)$$

$$\neg \forall_x F[x] \equiv \exists_x \neg F[x]$$

$$\forall_x F[x] \vee \forall_x G[x] \not\equiv \forall_x (F[x] \vee G[x])$$

$$\exists_x F[x] \vee \exists_x G[x] \equiv \exists_x (F[x] \vee G[x])$$

$$F \wedge G \equiv G \wedge F$$

$$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$$

$$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$

$$F \wedge \mathbb{T} \equiv F$$

$$F \wedge \mathbb{F} \equiv \mathbb{F}$$

$$F \wedge \neg F \equiv \mathbb{F}$$

$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$

$$(Qx)F[x] \wedge G \equiv (Qx)(F[x] \wedge G)$$

$$\neg(\exists_x)F[x] \equiv \forall_x \neg F[x]$$

$$\forall_x F[x] \wedge \forall_x G[x] \equiv \forall_x (F[x] \wedge G[x])$$

$$\exists_x F[x] \wedge \exists_x G[x] \not\equiv \exists_x (F[x] \wedge G[x])$$

Which implications do not hold in the  $\not\equiv$  above?





# Equivalences of Formulae

$$F \iff G \equiv (F \Rightarrow G) \wedge (G \Rightarrow F)$$

$$F \Rightarrow G \equiv \neg F \vee G$$

$$F \vee G \equiv G \vee F$$

$$(F \vee G) \vee H \equiv F \vee (G \vee H)$$

$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$$

$$F \vee \mathbb{T} \equiv \mathbb{T}$$

$$F \vee \mathbb{F} \equiv F$$

$$F \vee \neg F \equiv \mathbb{T}$$

$$\neg(\neg F) \equiv F$$

$$\neg(F \vee G) \equiv \neg F \wedge \neg G$$

$$(Qx)F[x] \vee G \equiv (Qx)(F[x] \vee G)$$

$$\neg \forall_x F[x] \equiv \exists_x \neg F[x]$$

$$\forall_x F[x] \vee \forall_x G[x] \not\equiv \forall_x (F[x] \vee G[x])$$

$$\exists_x F[x] \vee \exists_x G[x] \equiv \exists_x (F[x] \vee G[x])$$

$$F \wedge G \equiv G \wedge F$$

$$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$$

$$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$

$$F \wedge \mathbb{T} \equiv F$$

$$F \wedge \mathbb{F} \equiv \mathbb{F}$$

$$F \wedge \neg F \equiv \mathbb{F}$$

$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$

$$(Qx)F[x] \wedge G \equiv (Qx)(F[x] \wedge G)$$

$$\neg(\exists_x)F[x] \equiv \forall_x \neg F[x]$$

$$\forall_x F[x] \wedge \forall_x G[x] \equiv \forall_x (F[x] \wedge G[x])$$

$$\exists_x F[x] \wedge \exists_x G[x] \not\equiv \exists_x (F[x] \wedge G[x])$$

Which implications do not hold in the  $\not\equiv$  above?



# Equivalences of Formulae

$$F \iff G \equiv (F \Rightarrow G) \wedge (G \Rightarrow F)$$

$$F \Rightarrow G \equiv \neg F \vee G$$

$$F \vee G \equiv G \vee F$$

$$(F \vee G) \vee H \equiv F \vee (G \vee H)$$

$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$$

$$F \vee \mathbb{T} \equiv \mathbb{T}$$

$$F \vee \mathbb{F} \equiv F$$

$$F \vee \neg F \equiv \mathbb{T}$$

$$\neg(\neg F) \equiv F$$

$$\neg(F \vee G) \equiv \neg F \wedge \neg G$$

$$(Qx)F[x] \vee G \equiv (Qx)(F[x] \vee G)$$

$$\neg \forall_x F[x] \equiv \exists_x \neg F[x]$$

$$\forall_x F[x] \vee \forall_x G[x] \not\equiv \forall_x (F[x] \vee G[x])$$

$$\exists_x F[x] \vee \exists_x G[x] \equiv \exists_x (F[x] \vee G[x])$$

$$F \wedge G \equiv G \wedge F$$

$$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$$

$$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$

$$F \wedge \mathbb{T} \equiv F$$

$$F \wedge \mathbb{F} \equiv \mathbb{F}$$

$$F \wedge \neg F \equiv \mathbb{F}$$

$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$

$$(Qx)F[x] \wedge G \equiv (Qx)(F[x] \wedge G)$$

$$\neg(\exists_x)F[x] \equiv \forall_x \neg F[x]$$

$$\forall_x F[x] \wedge \forall_x G[x] \equiv \forall_x (F[x] \wedge G[x])$$

$$\exists_x F[x] \wedge \exists_x G[x] \not\equiv \exists_x (F[x] \wedge G[x])$$

Which implications do not hold in the  $\not\equiv$  above?



# Equivalences of Formulae

$$F \iff G \equiv (F \Rightarrow G) \wedge (G \Rightarrow F)$$

$$F \Rightarrow G \equiv \neg F \vee G$$

$$F \vee G \equiv G \vee F$$

$$(F \vee G) \vee H \equiv F \vee (G \vee H)$$

$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$$

$$F \vee \mathbb{T} \equiv \mathbb{T}$$

$$F \vee \mathbb{F} \equiv F$$

$$F \vee \neg F \equiv \mathbb{T}$$

$$\neg(\neg F) \equiv F$$

$$\neg(F \vee G) \equiv \neg F \wedge \neg G$$

$$(Qx)F[x] \vee G \equiv (Qx)(F[x] \vee G)$$

$$\neg \forall_x F[x] \equiv \exists_x \neg F[x]$$

$$\forall_x F[x] \vee \forall_x G[x] \not\equiv \forall_x (F[x] \vee G[x])$$

$$\exists_x F[x] \vee \exists_x G[x] \equiv \exists_x (F[x] \vee G[x])$$

$$F \wedge G \equiv G \wedge F$$

$$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$$

$$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$

$$F \wedge \mathbb{T} \equiv F$$

$$F \wedge \mathbb{F} \equiv \mathbb{F}$$

$$F \wedge \neg F \equiv \mathbb{F}$$

$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$

$$(Qx)F[x] \wedge G \equiv (Qx)(F[x] \wedge G)$$

$$\neg(\exists_x)F[x] \equiv \forall_x \neg F[x]$$

$$\forall_x F[x] \wedge \forall_x G[x] \equiv \forall_x (F[x] \wedge G[x])$$

$$\exists_x F[x] \wedge \exists_x G[x] \not\equiv \exists_x (F[x] \wedge G[x])$$

Which implications do not hold in the  $\not\equiv$  above?

$\implies$

$\not\Leftarrow$

$\nRightarrow$

$\Leftarrow$

## Equivalences of Formulae (cont'd)

Note that

$$\begin{aligned}\forall_x F[x] \vee \forall_x G[x] &\equiv \forall_x F[x] \vee \forall_y G[y] \equiv \forall_{x,y} F[x] \vee G[y] \\ \exists_x F[x] \wedge \exists_x G[x] &\equiv \exists_x F[x] \wedge \exists_y G[y] \equiv \exists_{x,y} F[x] \wedge G[y]\end{aligned}$$

# Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

**Normal Forms**

Formula Classification

Substitution

# Normal Forms

Normal forms:

1. CNF
2. DNF
3. negation normal form (NNF)
4. prenex normal form (PNF)
5. Skolem standard form

**Negation normal form (NNF)** requires that  $\neg$ ,  $\wedge$ , and  $\vee$  to be the only logical connectives and that negations appear only in literals.

A formula  $F$  in FOL is said to be in **prenex normal form (PNF)** iff the formula is in the form  $(Q_1x_1)\dots(Q_nx_n) M$ , where  $Q_i \in \{\forall, \exists\}$  and  $M$  is quantifier-free.

A FOL formula is in **Skolem standard form** if it is of the form  $\forall_{x_1, \dots, x_n} M$ , where  $M$  is a quantifier-free formula in CNF.

# Normal Forms

Normal forms:

1. CNF
2. DNF
3. negation normal form (NNF)
4. prenex normal form (PNF)
5. Skolem standard form

**Negation normal form (NNF)** requires that  $\neg$ ,  $\wedge$ , and  $\vee$  to be the only logical connectives and that negations appear only in literals.

A formula  $F$  in FOL is said to be in **prenex normal form (PNF)** iff the formula is in the form  $(Q_1x_1)\dots(Q_nx_n) M$ , where  $Q_i \in \{\forall, \exists\}$  and  $M$  is quantifier-free.

A FOL formula is in **Skolem standard form** if it is of the form  $\forall_{x_1, \dots, x_n} M$ , where  $M$  is a quantifier-free formula in CNF.

# Normal Forms

Normal forms:

1. CNF
2. DNF
3. negation normal form (NNF)
4. prenex normal form (PNF)
5. Skolem standard form

**Negation normal form (NNF)** requires that  $\neg$ ,  $\wedge$ , and  $\vee$  to be the only logical connectives and that negations appear only in literals.

A formula  $F$  in FOL is said to be in **prenex normal form (PNF)** iff the formula is in the form  $(Q_1x_1)\dots(Q_nx_n) M$ , where  $Q_i \in \{\forall, \exists\}$  and  $M$  is quantifier-free.

A FOL formula is in **Skolem standard form** if it is of the form  $\forall_{x_1, \dots, x_n} M$ , where  $M$  is a quantifier-free formula in CNF.



# Normal Forms

Normal forms:

1. CNF
2. DNF
3. negation normal form (NNF)
4. prenex normal form (PNF)
5. Skolem standard form

**Negation normal form (NNF)** requires that  $\neg$ ,  $\wedge$ , and  $\vee$  to be the only logical connectives and that negations appear only in literals.

A formula  $F$  in FOL is said to be in **prenex normal form (PNF)** iff the formula is in the form  $(Q_1x_1)\dots(Q_nx_n) M$ , where  $Q_i \in \{\forall, \exists\}$  and  $M$  is quantifier-free.

A FOL formula is in **Skolem standard form** if it is of the form  $\forall_{x_1, \dots, x_n} M$ , where  $M$  is a quantifier-free formula in CNF.

# Normal Forms

Normal forms:

1. CNF
2. DNF
3. negation normal form (NNF)
4. prenex normal form (PNF)
5. Skolem standard form

**Negation normal form (NNF)** requires that  $\neg$ ,  $\wedge$ , and  $\vee$  to be the only logical connectives and that negations appear only in literals.

A formula  $F$  in FOL is said to be in **prenex normal form (PNF)** iff the formula is in the form  $(Q_1x_1)\dots(Q_nx_n) M$ , where  $Q_i \in \{\forall, \exists\}$  and  $M$  is quantifier-free.

A FOL formula is in **Skolem standard form** if it is of the form  $\forall_{x_1, \dots, x_n} M$ , where  $M$  is a quantifier-free formula in CNF.

# Normal Forms

Normal forms:

1. CNF
2. DNF
3. negation normal form (NNF)
4. prenex normal form (PNF)
5. Skolem standard form

**Negation normal form (NNF)** requires that  $\neg$ ,  $\wedge$ , and  $\vee$  to be the only logical connectives and that negations appear only in literals.

A formula  $F$  in FOL is said to be in **prenex normal form (PNF)** iff the formula is in the form  $(Q_1x_1)\dots(Q_nx_n) M$ , where  $Q_i \in \{\forall, \exists\}$  and  $M$  is quantifier-free.

A FOL formula is in **Skolem standard form** if it is of the form  $\forall_{x_1, \dots, x_n} M$ , where  $M$  is a quantifier-free formula in CNF.

# Normal Forms

Normal forms:

1. CNF
2. DNF
3. negation normal form (NNF)
4. prenex normal form (PNF)
5. Skolem standard form

**Negation normal form (NNF)** requires that  $\neg$ ,  $\wedge$ , and  $\vee$  to be the only logical connectives and that negations appear only in literals.

A formula  $F$  in FOL is said to be in **prenex normal form (PNF)** iff the formula is in the form  $(Q_1x_1)\dots(Q_nx_n) M$ , where  $Q_i \in \{\forall, \exists\}$  and  $M$  is quantifier-free.

A FOL formula is in **Skolem standard form** if it is of the form  $\forall_{x_1, \dots, x_n} M$ , where  $M$  is a quantifier-free formula in CNF.

# Normal Forms

Normal forms:

1. CNF
2. DNF
3. negation normal form (NNF)
4. prenex normal form (PNF)
5. Skolem standard form

**Negation normal form (NNF)** requires that  $\neg$ ,  $\wedge$ , and  $\vee$  to be the only logical connectives and that negations appear only in literals.

A formula  $F$  in FOL is said to be in **prenex normal form (PNF)** iff the formula is in the form  $(Q_1x_1)\dots(Q_nx_n) M$ , where  $Q_i \in \{\forall, \exists\}$  and  $M$  is quantifier-free.

A FOL formula is in **Skolem standard form** if it is of the form  $\forall_{x_1, \dots, x_n} M$ , where  $M$  is a quantifier-free formula in CNF.

# Normal Forms

Normal forms:

1. CNF
2. DNF
3. negation normal form (NNF)
4. prenex normal form (PNF)
5. Skolem standard form

**Negation normal form (NNF)** requires that  $\neg$ ,  $\wedge$ , and  $\vee$  to be the only logical connectives and that negations appear only in literals.

A formula  $F$  in FOL is said to be in **prenex normal form (PNF)** iff the formula is in the form  $(Q_1x_1)\dots(Q_nx_n) M$ , where  $Q_i \in \{\forall, \exists\}$  and  $M$  is quantifier-free.

A FOL formula is in **Skolem standard form** if it is of the form  $\forall_{x_1, \dots, x_n} M$ , where  $M$  is a quantifier-free formula in CNF.

# Normal Forms (cont'd)

## Examples:

1. Prove the following by bringing the formulae into conjunctive normal form

$$\left(\forall_x P[x]\right) \Rightarrow Q \equiv \exists_x (P[x] \Rightarrow Q).$$

2. Bring the following formulae into Skolem standard form



$$\forall_{x,y,z} ((\neg P[x,y] \wedge Q[x,z]) \vee R[x,y,z])$$



$$\forall_{x,y} \left( \exists_z (P[x,z] \wedge P[y,z]) \right) \Rightarrow \exists_u Q[x,y,u]$$

# Normal Forms (cont'd)

## Examples:

1. Prove the following by bringing the formulae into conjunctive normal form

$$\left(\forall_x P[x]\right) \Rightarrow Q \equiv \exists_x (P[x] \Rightarrow Q).$$

2. Bring the following formulae into Skolem standard form

$$\forall_{x,y,z} ((\neg P[x,y] \wedge Q[x,z]) \vee R[x,y,z])$$

$$\forall_{x,y} \left( \exists_z P[x,z] \wedge P[y,z] \right) \Rightarrow \exists_u Q[x,y,u]$$



# Normal Forms (cont'd)

## Examples:

1. Prove the following by bringing the formulae into conjunctive normal form

$$\left(\forall_x P[x]\right) \Rightarrow Q \equiv \exists_x (P[x] \Rightarrow Q).$$

2. Bring the following formulae into Skolem standard form

▶

$$\forall_{x,y,z} ((\neg P[x,y] \wedge Q[x,z]) \vee R[x,y,z])$$

▶

$$\forall_{x,y} \left( \exists_z (P[x,z] \wedge P[y,z]) \right) \Rightarrow \exists_u Q[x,y,u]$$

# Normal Forms (cont'd)

## Examples:

1. Prove the following by bringing the formulae into conjunctive normal form

$$\left(\forall_x P[x]\right) \Rightarrow Q \equiv \exists_x (P[x] \Rightarrow Q).$$

2. Bring the following formulae into Skolem standard form



$$\forall_{x,y,z} ((\neg P[x,y] \wedge Q[x,z]) \vee R[x,y,z])$$



$$\forall_{x,y} \left( \exists_z (P[x,z] \wedge P[y,z]) \right) \Rightarrow \exists_u Q[x,y,u]$$

# Normal Forms (cont'd)

## Examples:

1. Prove the following by bringing the formulae into conjunctive normal form

$$\left(\forall_x P[x]\right) \Rightarrow Q \equiv \exists_x (P[x] \Rightarrow Q).$$

2. Bring the following formulae into Skolem standard form



$$\forall_x \exists_{y,z} ((\neg P[x,y] \wedge Q[x,z]) \vee R[x,y,z])$$



$$\forall_{x,y} \left( \exists_z P[x,z] \wedge P[y,z] \right) \Rightarrow \exists_u Q[x,y,u]$$

# Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

**Formula Classification**

Substitution

## Formula Clausification

A **clause** is a disjunction of literals.

Examples:  $\neg P[x] \vee Q[y, f[x]]$ ,  $P[x]$

A **set of clauses**  $S$  is regarded as a conjunction of all clauses in  $S$ , where every variable in  $S$  is considered governed by a universal quantifier.

Example: Let

$$\forall_x \exists_{y,z} ((\neg P[x, y] \wedge Q[x, z]) \vee R[x, y, z])$$

The standard form of the formula above, that is

$$\forall_x ((\neg P[x, f[x]] \vee R[x, f[x], g[x]]) \wedge (Q(x, g[x]) \vee R[x, f[x], g[x]]))$$

can be represented by the following set of clauses

$$\{\neg P[x, f[x]] \vee R[x, f[x], g[x]], Q(x, g[x]) \vee R[x, f[x], g[x]]\}$$

**Note that**, if  $S$  is a set of clauses that represents a standard form of a formula  $F$ , then  $F$  is inconsistent iff  $S$  is inconsistent.

## Formula Clausification

A **clause** is a disjunction of literals.

**Examples:**  $\neg P[x] \vee Q[y, f[x]], P[x]$

A **set of clauses**  $S$  is regarded as a conjunction of all clauses in  $S$ , where every variable in  $S$  is considered governed by a universal quantifier.

**Example:** Let

$$\forall_x \exists_{y,z} ((\neg P[x, y] \wedge Q[x, z]) \vee R[x, y, z])$$

The standard form of the formula above, that is

$$\forall_x ((\neg P[x, f[x]] \vee R[x, f[x], g[x]]) \wedge (Q(x, g[x]) \vee R[x, f[x], g[x]]))$$

can be represented by the following set of clauses

$$\{\neg P[x, f[x]] \vee R[x, f[x], g[x]], Q(x, g[x]) \vee R[x, f[x], g[x]]\}$$

**Note that,** if  $S$  is a set of clauses that represents a standard form of a formula  $F$ , then  $F$  is inconsistent iff  $S$  is inconsistent.

## Formula Clausification

A **clause** is a disjunction of literals.

**Examples:**  $\neg P[x] \vee Q[y, f[x]]$ ,  $P[x]$

A **set of clauses**  $S$  is regarded as a conjunction of all clauses in  $S$ , where every variable in  $S$  is considered governed by a universal quantifier.

**Example:** Let

$$\forall_x \exists_{y,z} ((\neg P[x, y] \wedge Q[x, z]) \vee R[x, y, z])$$

The standard form of the formula above, that is

$$\forall_x ((\neg P[x, f[x]] \vee R[x, f[x], g[x]]) \wedge (Q(x, g[x]) \vee R[x, f[x], g[x]]))$$

can be represented by the following set of clauses

$$\{\neg P[x, f[x]] \vee R[x, f[x], g[x]], Q(x, g[x]) \vee R[x, f[x], g[x]]\}$$

**Note that,** if  $S$  is a set of clauses that represents a standard form of a formula  $F$ , then  $F$  is inconsistent iff  $S$  is inconsistent.

## Formula Clausification

A **clause** is a disjunction of literals.

**Examples:**  $\neg P[x] \vee Q[y, f[x]]$ ,  $P[x]$

A **set of clauses**  $S$  is regarded as a conjunction of all clauses in  $S$ , where every variable in  $S$  is considered governed by a universal quantifier.

**Example:** Let

$$\forall_x \exists_{y,z} ((\neg P[x, y] \wedge Q[x, z]) \vee R[x, y, z])$$

The standard form of the formula above, that is

$$\forall_x ((\neg P[x, f[x]] \vee R[x, f[x], g[x]]) \wedge (Q(x, g[x]) \vee R[x, f[x], g[x]]))$$

can be represented by the following set of clauses

$$\{\neg P[x, f[x]] \vee R[x, f[x], g[x]], Q(x, g[x]) \vee R[x, f[x], g[x]]\}$$

**Note that,** if  $S$  is a set of clauses that represents a standard form of a formula  $F$ , then  $F$  is inconsistent iff  $S$  is inconsistent.



## Formula Clausification

A **clause** is a disjunction of literals.

**Examples:**  $\neg P[x] \vee Q[y, f[x]], P[x]$

A **set of clauses**  $S$  is regarded as a conjunction of all clauses in  $S$ , where every variable in  $S$  is considered governed by a universal quantifier.

**Example:** Let

$$\forall_x \exists_{y,z} ((\neg P[x, y] \wedge Q[x, z]) \vee R[x, y, z])$$

The standard form of the formula above, that is

$$\forall_x ((\neg P[x, f[x]] \vee R[x, f[x], g[x]]) \wedge (Q(x, g[x]) \vee R[x, f[x], g[x]]))$$

can be represented by the following set of clauses

$$\{\neg P[x, f[x]] \vee R[x, f[x], g[x]], Q(x, g[x]) \vee R[x, f[x], g[x]]\}$$

**Note that,** if  $S$  is a set of clauses that represents a standard form of a formula  $F$ , then  $F$  is inconsistent iff  $S$  is inconsistent.

## Formulae Classification (cont'd)

Example :

Transform the formulae  $F_1, F_2, F_3, F_4$ , and  $\neg G$  into a set of clauses, where

$$F_1 : \forall_{x,y} \exists_z P[x, y, z]$$

$$F_2 : \forall_{x,y,z,u,v,w} ((P[x, y, u] \wedge P[y, z, v] \wedge P[u, z, w]) \Rightarrow P[x, v, w]) \\ \wedge \forall_{x,y,z,u,v,w} (P[x, y, u] \wedge (P[y, z, v] \wedge P[x, v, w]) \Rightarrow P[u, z, w])$$

$$F_3 : \forall_x P[x, e, x] \wedge \forall_x P[e, x, x]$$

$$F_4 : \forall_x P[x, i[x], e] \wedge \forall_x P[i[x], x, e]$$

$$G : \left( \forall_x P[x, x, e] \right) \Rightarrow \forall_{u,v,w} (P[u, v, w] \Rightarrow P[v, u, w])$$

# Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Formula Classification

**Substitution**

# Substitution

Example: Let

$$\begin{aligned}C_1 &: P[x] \vee Q[x] \\C_2 &: \neg P[f[x]] \vee R[x]\end{aligned}$$

# Substitution

Example: Let

$$\begin{aligned} C_1 &: P[x] \vee Q[x] \\ C_2 &: \neg P[f[x]] \vee R[x] \end{aligned}$$

# Substitution

Example: Let

$$\begin{aligned}C_1 &: P[x] \vee Q[x] \\C_2 &: \neg P[f[x]] \vee R[x]\end{aligned}$$

Let  $x \rightarrow f[a]$  in  $C_1$ ,  $x \rightarrow a$  in  $C_2$ .

# Substitution

Example: Let

$$C_1 : P[x] \vee Q[x]$$

$$C_2 : \neg P[f[x]] \vee R[x]$$

Let  $x \rightarrow f[a]$  in  $C_1$ ,  $x \rightarrow a$  in  $C_2$ .

We have

$$C'_1 : P[f[a]] \vee Q[f[a]]$$

$$C'_2 : \neg P[f[a]] \vee R[a]$$

# Substitution

Example: Let

$$C_1 : P[x] \vee Q[x]$$

$$C_2 : \neg P[f[x]] \vee R[x]$$

Let  $x \rightarrow f[a]$  in  $C_1$ ,  $x \rightarrow a$  in  $C_2$ .

We have

$$C'_1 : P[f[a]] \vee Q[f[a]]$$

$$C'_2 : \neg P[f[a]] \vee R[a]$$

$C'_1$  and  $C'_2$  are **ground** instances.



# Substitution

Example: Let

$$\begin{aligned}C_1 &: P[x] \vee Q[x] \\C_2 &: \neg P[f[x]] \vee R[x]\end{aligned}$$

Let  $x \rightarrow f[a]$  in  $C_1$ ,  $x \rightarrow a$  in  $C_2$ .

We have

$$\begin{aligned}C'_1 &: P[f[a]] \vee Q[f[a]] \\C'_2 &: \neg P[f[a]] \vee R[a]\end{aligned}$$

$C'_1$  and  $C'_2$  are **ground** instances.

A **resolvent** of  $C'_1$  and  $C'_2$  is

$$C'_3 : Q[f[a]] \vee R[a]$$

# Substitution

Example: Let

$$C_1 : P[x] \vee Q[x]$$

$$C_2 : \neg P[f[x]] \vee R[x]$$

Let  $x \rightarrow f[x]$  in  $C_1$ . We have

$$C_1^* : P[f[x]] \vee Q[f[x]]$$

$C_1^*$  is an instance of  $C_1$ .

A resolvent of

$$C_2 : \neg P[f[x]] \vee R[x]$$

$$C_1^* : P[f[x]] \vee Q[f[x]]$$

is

$$C_3 : Q[f[x]] \vee R[x]$$

$C_3'$  is an instance of  $C_3$ .  $C_3$  is the most general clause.

# Substitution

Example: Let

$$\begin{aligned}C_1 &: P[x] \vee Q[x] \\C_2 &: \neg P[f[x]] \vee R[x]\end{aligned}$$

Let  $x \rightarrow f[x]$  in  $C_1$ . We have

$$C_1^* : P[f[x]] \vee Q[f[x]]$$

$C_1^*$  is an **instance** of  $C_1$ .

A resolvent of

$$\begin{aligned}C_2 &: \neg P[f[x]] \vee R[x] \\C_1^* &: P[f[x]] \vee Q[f[x]]\end{aligned}$$

is

$$C_3 : Q[f[x]] \vee R[x]$$

$C_3'$  is an instance of  $C_3$ .  $C_3$  is the **most general clause**.

# Substitution

Example: Let

$$C_1 : P[x] \vee Q[x]$$

$$C_2 : \neg P[f[x]] \vee R[x]$$

Let  $x \rightarrow f[x]$  in  $C_1$ . We have

$$C_1^* : P[f[x]] \vee Q[f[x]]$$

$C_1^*$  is an **instance** of  $C_1$ .

A resolvent of

$$C_2 : \neg P[f[x]] \vee R[x]$$

$$C_1^* : P[f[x]] \vee Q[f[x]]$$

is

$$C_3 : Q[f[x]] \vee R[x]$$

$C_3$  is an instance of  $C_3$ .  $C_3$  is the **most general clause**.

# Substitution

Example: Let

$$C_1 : P[x] \vee Q[x]$$

$$C_2 : \neg P[f[x]] \vee R[x]$$

Let  $x \rightarrow f[x]$  in  $C_1$ . We have

$$C_1^* : P[f[x]] \vee Q[f[x]]$$

$C_1^*$  is an **instance** of  $C_1$ .

A resolvent of

$$C_2 : \neg P[f[x]] \vee R[x]$$

$$C_1^* : P[f[x]] \vee Q[f[x]]$$

is

$$C_3 : Q[f[x]] \vee R[x]$$

$C_3$  is an instance of  $C_3$ .  $C_3$  is the **most general clause**.

# Substitution

Example: Let

$$C_1 : P[x] \vee Q[x]$$

$$C_2 : \neg P[f[x]] \vee R[x]$$

Let  $x \rightarrow f[x]$  in  $C_1$ . We have

$$C_1^* : P[f[x]] \vee Q[f[x]]$$

$C_1^*$  is an **instance** of  $C_1$ .

A resolvent of

$$C_2 : \neg P[f[x]] \vee R[x]$$

$$C_1^* : P[f[x]] \vee Q[f[x]]$$

is

$$C_3 : Q[f[x]] \vee R[x]$$

$C_3'$  is an instance of  $C_3$ .  $C_3$  is the **most general clause**.

## Substitution (cont'd)

A **substitution**  $\sigma$  is a finite set of the form  $\{v_1 \rightarrow t_1, \dots, v_n \rightarrow t_n\}$  where every  $t_i$  is a term different from  $v_i$  and no two elements in the set have the same variable  $v_i$ .

Let  $\sigma$  be defined as above and  $E$  be an expression. Then  $E\sigma$  is an expression obtained from  $E$  by replacing simultaneously each occurrence of  $v_i$  in  $E$  by the term  $t_i$ .

Example: Let  $\sigma = \{x \rightarrow z, z \rightarrow h[a, y]\}$  and  $E = f[z, a, g[x], y]$ . Then  $E\sigma = f[h[a, y], a, g[z], y]$ .

## Substitution (cont'd)

A **substitution**  $\sigma$  is a finite set of the form  $\{v_1 \rightarrow t_1, \dots, v_n \rightarrow t_n\}$  where every  $t_i$  is a term different from  $v_i$  and no two elements in the set have the same variable  $v_i$ .

Let  $\sigma$  be defined as above and  $E$  be an expression. Then  $E\sigma$  is an expression obtained from  $E$  by replacing simultaneously each occurrence of  $v_i$  in  $E$  by the term  $t_i$ .

Example: Let  $\sigma = \{x \rightarrow z, z \rightarrow h[a, y]\}$  and  $E = f[z, a, g[x], y]$ . Then  $E\sigma = f[h[a, y], a, g[z], y]$ .



## Substitution (cont'd)

A **substitution**  $\sigma$  is a finite set of the form  $\{v_1 \rightarrow t_1, \dots, v_n \rightarrow t_n\}$  where every  $t_i$  is a term different from  $v_i$  and no two elements in the set have the same variable  $v_i$ .

Let  $\sigma$  be defined as above and  $E$  be an expression. Then  $E\sigma$  is an expression obtained from  $E$  by replacing simultaneously each occurrence of  $v_i$  in  $E$  by the term  $t_i$ .

**Example:** Let  $\sigma = \{x \rightarrow z, z \rightarrow h[a, y]\}$  and  $E = f[z, a, g[x], y]$ . Then  $E\sigma = f[h[a, y], a, g[z], y]$ .

## Substitution (cont'd)

Let

$$\theta = \{x_1 \rightarrow t_1, \dots, x_n \rightarrow t_n\}$$

$$\lambda = \{y_1 \rightarrow u_1, \dots, y_n \rightarrow u_n\}$$

Then the **composition** of  $\theta$  and  $\lambda$  ( $\theta \circ \lambda$ ) is obtained from the set

$$\{x_1 \rightarrow t_1\lambda, \dots, x_n \rightarrow t_n\lambda, y_1 \rightarrow u_1, \dots, y_n \rightarrow u_n\}$$

by deleting any element  $x_j \rightarrow t_j\lambda$  for which  $x_j = t_j\lambda$  and any element  $y_i \rightarrow u_i$  such that  $y_i$  is among  $\{x_1, \dots, x_n\}$ .

## Substitution (cont'd)

Example 1:

$$\theta = \{x \rightarrow f[y], y \rightarrow z\}$$

$$\lambda = \{x \rightarrow a, y \rightarrow b, z \rightarrow y\}$$

Then

$$\begin{aligned}\theta \circ \lambda &= \{x \rightarrow f[b], y \rightarrow y, x \rightarrow a, y \rightarrow b, z \rightarrow y\} \\ &= \{x \rightarrow f[b], z \rightarrow y\}\end{aligned}$$

Example 2:

$$\theta_1 = \{x \rightarrow a, y \rightarrow f[z], z \rightarrow y\}$$

$$\theta_2 = \{x \rightarrow b, y \rightarrow z, z \rightarrow g[x]\}$$

Then

$$\begin{aligned}\theta_1 \circ \theta_2 &= \{x \rightarrow a, y \rightarrow f[g[x]], z \rightarrow z, x \rightarrow b, y \rightarrow z, z \rightarrow g[x]\} \\ &= \{x \rightarrow a, y \rightarrow f[g[x]]\}\end{aligned}$$

## Substitution (cont'd)

Example 1:

$$\begin{aligned}\theta &= \{x \rightarrow f[y], y \rightarrow z\} \\ \lambda &= \{x \rightarrow a, y \rightarrow b, z \rightarrow y\}\end{aligned}$$

Then

$$\begin{aligned}\theta \circ \lambda &= \{x \rightarrow f[b], y \rightarrow y, x \rightarrow a, y \rightarrow b, z \rightarrow y\} \\ &= \{x \rightarrow f[b], z \rightarrow y\}\end{aligned}$$

Example 2:

$$\begin{aligned}\theta_1 &= \{x \rightarrow a, y \rightarrow f[z], z \rightarrow y\} \\ \theta_2 &= \{x \rightarrow b, y \rightarrow z, z \rightarrow g[x]\}\end{aligned}$$

Then

$$\begin{aligned}\theta_1 \circ \theta_2 &= \{x \rightarrow a, y \rightarrow f[g[x]], z \rightarrow z, x \rightarrow b, y \rightarrow z, z \rightarrow g[x]\} \\ &= \{x \rightarrow a, y \rightarrow f[g[x]]\}\end{aligned}$$

## Substitution (cont'd)

Example 1:

$$\theta = \{x \rightarrow f[y], y \rightarrow z\}$$

$$\lambda = \{x \rightarrow a, y \rightarrow b, z \rightarrow y\}$$

Then

$$\begin{aligned}\theta \circ \lambda &= \{x \rightarrow f[b], y \rightarrow y, x \rightarrow a, y \rightarrow b, z \rightarrow y\} \\ &= \{x \rightarrow f[b], z \rightarrow y\}\end{aligned}$$

Example 2:

$$\theta_1 = \{x \rightarrow a, y \rightarrow f[z], z \rightarrow y\}$$

$$\theta_2 = \{x \rightarrow b, y \rightarrow z, z \rightarrow g[x]\}$$

Then

$$\begin{aligned}\theta_1 \circ \theta_2 &= \{x \rightarrow a, y \rightarrow f[g[x]], z \rightarrow z, x \rightarrow b, y \rightarrow z, z \rightarrow g[x]\} \\ &= \{x \rightarrow a, y \rightarrow f[g[x]]\}\end{aligned}$$

## Substitution (cont'd)

Example 1:

$$\theta = \{x \rightarrow f[y], y \rightarrow z\}$$

$$\lambda = \{x \rightarrow a, y \rightarrow b, z \rightarrow y\}$$

Then

$$\begin{aligned}\theta \circ \lambda &= \{x \rightarrow f[b], y \rightarrow y, x \rightarrow a, y \rightarrow b, z \rightarrow y\} \\ &= \{x \rightarrow f[b], z \rightarrow y\}\end{aligned}$$

Example 2:

$$\theta_1 = \{x \rightarrow a, y \rightarrow f[z], z \rightarrow y\}$$

$$\theta_2 = \{x \rightarrow b, y \rightarrow z, z \rightarrow g[x]\}$$

Then

$$\begin{aligned}\theta_1 \circ \theta_2 &= \{x \rightarrow a, y \rightarrow f[g[x]], z \rightarrow z, x \rightarrow b, y \rightarrow z, z \rightarrow g[x]\} \\ &= \{x \rightarrow a, y \rightarrow f[g[x]]\}\end{aligned}$$