Logic 1
First-Order Logic

Mădălina Erașcu    Tudor Jebelean

Research Institute for Symbolic Computation,
Johannes Kepler University, Linz, Austria

{merascu,tjebelea}@risc.jku.at

November 21, 2013
Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Formula Clausification

Substitution
Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Formula Clausification

Substitution
The language of FOL consists in terms and formulas. 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 $t_1, ..., t_n$ are terms then $f[t_1, ..., t_n]$ is a term.
4. All terms are generated by applying the above rules.

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

An atom is $T$, $F$, 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 formulas. 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 $t_1, ..., t_n$ are terms then $f[t_1, ..., t_n]$ is a term.
4. All terms are generated by applying the above rules.

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

An atom is $\top$, $\bot$, 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 formulas. 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 $t_1, ..., t_n$ are terms then $f[t_1, ..., t_n]$ is a term.

4. All terms are generated by applying the above rules.

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

An atom is $T$, $F$, 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 formulas. **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 $t_1, ..., t_n$ are terms then $f[t_1, ..., t_n]$ is a term.
4. All terms are generated by applying the above rules.

If $P$ is an $n$-place predicate symbol and $t_1, ..., t_n$ are terms then $P[t_1, ..., t_n]$ 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.
The language of FOL consists in terms and formulas. 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 $t_1, \ldots, t_n$ are terms then $f[t_1, \ldots, t_n]$ is a term.
4. All terms are generated by applying the above rules.

If $P$ is an $n$-place predicate symbol and $t_1, \ldots, t_n$ are terms then $P[t_1, \ldots, t_n]$ 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.
The language of FOL consists in terms and formulas. 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 \( t_1, \ldots, t_n \) are terms then \( f[t_1, \ldots, t_n] \) is a term.
4. All terms are generated by applying the above rules.

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

An atom is \( \top, \bot \), 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 formulas.

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 $t_1, \ldots, t_n$ are terms then $f[t_1, \ldots, t_n]$ is a term.
4. All terms are generated by applying the above rules.

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

An atom is $T$, $F$, or an $n$-ary predicate applied to $n$ terms.

A literal is an atom or its negation.
The language of FOL consists in terms and formulas. 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 $t_1, ..., t_n$ are terms then $f[t_1, ..., t_n]$ is a term.
4. All terms are generated by applying the above rules.

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

An atom is $\top, \bot$, 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 formulas. 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 $t_1, \ldots, t_n$ are terms then $f[t_1, \ldots, t_n]$ is a term.
4. All terms are generated by applying the above rules.

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

An atom is $\top$, $\bot$, or an $n$-ary predicate applied to $n$ terms.

A literal is an atom or its negation.
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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, formulas from the bellow

1. $\forall x \, x + 1 \geq x$
2. $\neg \left( \exists x \, E[0, f[x]] \right)$
3. $\forall \exists x \exists y \left( E[y, f[x]] \land \forall z \left( E[z, f[x]] \implies E[y, z] \right) \right)$
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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, formulas from the bellow

1. $\forall x \; x + 1 \geq x$
2. $\neg \left( \exists x \; E[0, f[x]] \right)$
3. $\forall x \exists y \left( E[y, f[x]] \land \forall z \left( E[z, f[x]] \implies E[y, z] \right) \right)$
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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, formulas from the bellow

1. $\forall x (x + 1 \geq x)$

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

3. $\forall x \exists y \left( E[y, f[x]] \land \forall z (E[z, f[x]] \implies E[y, z]) \right)$
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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 $\forall$ or $\exists$.

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, formulas from the bellow

1. $\forall x \ x + 1 \geq x$
2. $\neg (\exists x \ E[0, f[x]])$
3. $\forall \exists (\forall y (E[y, f[x]] \land \forall z (E[z, f[x]] \implies E[y, z])))$
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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, formulas from the below

1. $\forall x x + 1 \geq x$
2. $\neg \left( \exists x E[0, f(x)] \right)$
3. $\forall x \exists y \left( E[y, f(x)] \land \forall z \left( E[z, f(x)] \implies E[y, z] \right) \right)$
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If \( F \) and \( G \) are formulas then \( \neg F, F \lor G, F \land G, F \implies G \), and \( F \iff G \) are formulas.
3. If \( F \) is a formula and \( x \) is a variable, then \( \forall x F \) and \( \exists x F \) are formulas.
4. Formulas 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, formulas from the below

1. \( \forall x x + 1 \geq x \)
2. \( \neg \left( \exists x E[0, f[x]] \right) \)
3. \( \forall x \exists y \left( E[y, f[x]] \land \forall z (E[z, f[x]] \implies E[y, z]) \right) \)
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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 $\forall$ or $\exists$.

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, formulas from the bellow

1. $\forall x (x + 1 \geq x)$
2. $\neg (\exists x E[0, f[x]])$
3. $\forall x \exists y (E[y, f[x]] \land \forall z (E[z, f[x]] \implies E[y, z]))$
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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, formulas from the bellow

1. $\forall x x + 1 \geq x$
2. $\neg (\exists x E[0, f[x]])$
3. $\forall x \exists y (E[y, f[x]] \land \forall z (E[z, f[x]] \implies E[y, z]))$
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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, formulas from the bellow

1. $\forall x \quad x + 1 \geq x$

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

3. $\forall x \exists y (E[y, f[x]] \land \forall z (E[z, f[x]] \implies E[y, z]))$
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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, formulas from the bellow

1. $\forall x x + 1 \geq x$
2. $\neg \left( \exists x E[0, f[x]] \right)$
3. $\forall x \exists y \left( E[y, f[x]] \land \forall z (E[z, f[x]] \implies E[y, z]) \right)$
Syntax (cont’d)

Formulas are defined as follows:

1. An atom is a formula.
2. If $F$ and $G$ are formulas then $\neg F$, $F \lor G$, $F \land G$, $F \implies G$, and $F \iff G$ are formulas.
3. If $F$ is a formula and $x$ is a variable, then $\forall x F$ and $\exists x F$ are formulas.
4. Formulas 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, formulas from the bellow

1. $\forall x \ x + 1 \geq x$
2. $\neg \left( \exists x \ E[x, f[x]] \right)$
3. $\forall x \exists y \left( E[y, f[x]] \land \forall z \left( E[z, f[x]] \implies E[y, z] \right) \right)$
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 $\{T, F\}$.

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

Then the semantics of the formula $F$ is a function $f : \mathcal{I} \rightarrow \{T, F\}$, 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 \{T, F\}.

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

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 $\{T, F\}$.

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

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

Semantics
Example: Find the truth value of the formulas:

- \( F_1 \leftarrow \iff \forall \forall x \leq y, \) where \( I : \left\{ \begin{array}{l} D = \{0, 1\} \\
\leq_I \rightarrow \leq_{\mathbb{Z}} \end{array} \right. \)

- \( F_2 \leftarrow \iff \forall \exists x + y > c, \) where \( I : \left\{ \begin{array}{l} D = \{0, 1\} \\
c_I = 0 \\
+I \rightarrow +\mathbb{Z} \\
>_{I} \rightarrow >_{\mathbb{Z}} \end{array} \right. \)

- \( F_3 \leftarrow \iff \forall (P[x] \implies Q[f[x], a]), \) where

\[
\begin{align*}
I : & \left\{ \begin{array}{l} D = \{1, 2\} \\
a_I = 1 \\
f_I : D \rightarrow D \\
P_I : D \rightarrow \{\mathbb{T}, \mathbb{F}\} \\
Q_I : D^2 \rightarrow \{\mathbb{T}, \mathbb{F}\} \\
\end{array} \right. \\
& \left\{ \begin{array}{l} f_I[1] = 1 \\
f_I[2] = 1 \\
P_I[1] = \mathbb{T} \\
P_I[2] = \mathbb{F} \\
Q_I[1, 1] = \mathbb{T} \\
Q_I[1, 2] = \mathbb{F} \\
Q_I[2, 1] = \mathbb{F} \\
Q_I[2, 2] = \mathbb{T} \end{array} \right. \end{align*}
\]
Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Formula Clausification

Substitution
(Un)Satisfiability & (In)Validity

A formula $F$ is satisfiable (consistent) iff there exists an interpretation $I$ such that $F$ is evaluated to $\top$ in $I$.

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

A formula $F$ is valid iff for all interpretations $I$, $F$ is evaluated to $\top$ in $I$.

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

A formula $G$ is a logical consequence of formulas $F_1$, $F_2$, ..., $F_n$ iff for every interpretation $I$, if $F_1 \land F_2 \land \ldots \land F_n$ is true in $I$, $G$ is also true in $I$.

Note that validity and satisfiability applies to closed formulas.

Examples: Prove that

$\forall x P[x] \land \exists y \neg P[y]$ is inconsistent.
A formula $F$ is **satisfiable (consistent)** iff there exists an interpretation $I$ such that $F$ is evaluated to $\top$ in $I$.

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

A formula $F$ is **valid** iff for all interpretations $I$, $F$ is evaluated to $\top$ in $I$.

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

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

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

**Examples:** Prove that

$\forall x P[x] \land \exists y \neg P[y]$ is inconsistent.
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 formulas $F_1$, $F_2$, ..., $F_n$ iff for every interpretation $I$, if $F_1 \land F_2 \land ... \land F_n$ is true in $I$, $G$ is also true in $I$.

Note that validity and satisfiability applies to closed formulas.

**Examples:** Prove that

$\forall x P[x] \land \exists y \neg P[y]$ is inconsistent.
A formula $F$ is **satisfiable (consistent)** iff there exists an interpretation $I$ such that $F$ is evaluated to $\top$ in $I$.

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

A formula $F$ is **valid** iff for all interpretations $I$, $F$ is evaluated to $\top$ in $I$.

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

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

Note that validity and satisfiability applies to closed formulas.

**Examples:** Prove that

$\forall x P[x] \land \exists y \neg P[y]$ is inconsistent.
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 formulas $F_1, F_2, ..., F_n$ iff for every interpretation $I$, if $F_1 \land F_2 \land ... \land F_n$ is true in $I$, $G$ is also true in $I$.

Note that validity and satisfiability applies to closed formulas.

Examples: Prove that

$\forall x P[x] \land \exists y \neg P[y]$ is inconsistent.
(Un)Satisfiability & (In)Validity

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

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

A formula $F$ is **valid** iff for all interpretations $I$, $F$ is evaluated to $\top$ in $I$.

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

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

Note that validity and satisfiability applies to closed formulas.

**Examples:** Prove that

\[
\forall x P[x] \land \exists y \neg P[y]
\]

is inconsistent.
A formula $F$ is **satisfiable (consistent)** iff there exists an interpretation $I$ such that $F$ is evaluated to $\top$ in $I$.

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

A formula $F$ is **valid** iff for all interpretations $I$, $F$ is evaluated to $\top$ in $I$.

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

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

Note that validity and satisfiability applies to closed formulas.

**Examples:** Prove that

$\forall x P[x] \land \exists y \neg P[y]$ is inconsistent.
Outline

Syntax

Semantics

(Un)satisfiability & (In)validity

Equivalences of Formulas

Normal Forms

Formula Clausification

Substitution
Equivalences of Formulas

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

\[
\begin{align*}
F \iff G & \equiv (F \Rightarrow G) \land (G \Rightarrow F) \\
F \Rightarrow G & \equiv \neg F \lor G \\
F \lor G & \equiv G \lor F \\
(F \lor G) \lor H & \equiv F \lor (G \lor H) \\
F \lor (G \land H) & \equiv (F \lor G) \land (F \lor H) \\
F \lor T & \equiv T \\
F \lor F & \equiv F \\
F \lor \neg F & \equiv T \\
\neg (\neg F) & \equiv F \\
\neg (F \lor G) & \equiv \neg F \land \neg G \\
(Qx)F[x] \lor G & \equiv (Qx)(F[x] \lor G) \\
\neg \forall F[x] & \equiv \exists \neg F[x] \\
\forall F[x] \lor \forall G[x] & \not\equiv \forall (F[x] \lor G[x]) \\
\exists F[x] \lor \exists G[x] & \equiv \exists (F[x] \lor G[x]) \\
\forall F[x] \land \forall G[x] & \equiv \forall (F[x] \land G[x]) \\
\exists F[x] \land \exists G[x] & \not\equiv \exists (F[x] \land G[x])
\end{align*}
\]
### Equivalences of Formulas

| $F$ $\iff$ $G$ $\equiv$ $(F \Rightarrow G) \land (G \Rightarrow F)$ | $F \land G$ $\equiv$ $G \land F$ |
| $F \Rightarrow G$ $\equiv$ $\neg F \lor G$ | $(F \land G) \land H$ $\equiv$ $F \land (G \land H)$ |
| $F \lor G$ $\equiv$ $G \lor F$ | $F \land (G \lor H)$ $\equiv$ $(F \land G) \lor (F \land H)$ |
| $(F \lor G) \lor H$ $\equiv$ $F \lor (G \lor H)$ | $F \lor F$ $\equiv$ $F$ |
| $F \lor (G \land H)$ $\equiv$ $(F \lor G) \land (F \lor H)$ | $F \lor T$ $\equiv$ $T$ |
| $F \lor \top$ $\equiv$ $\top$ | $F \land T$ $\equiv$ $F$ |
| $F \lor \bot$ $\equiv$ $\bot$ | $F \land \bot$ $\equiv$ $F$ |
| $\neg (\neg F)$ $\equiv$ $F$ | $\neg (F \land G)$ $\equiv$ $\neg F \land \neg G$ |
| $\neg (F \lor G)$ $\equiv$ $\neg F \land \neg G$ | $(Qx) F[x] \land G$ $\equiv$ $(Qx) (F[x] \land G)$ |
| $(Qx) F[x] \lor G$ $\equiv$ $(Qx) (F[x] \lor G)$ | $\neg (\exists x) F[x]$ $\equiv$ $\forall \neg F[x]$ |
| $\forall F[x] \equiv \exists \neg F[x]$ | $\forall F[x] \land \forall G[x]$ $\equiv$ $\forall (F[x] \land G[x])$ |
| $\forall F[x] \lor \forall G[x]$ $\not\equiv$ $\forall (F[x] \lor G[x])$ | $\exists F[x] \land \exists G[x]$ $\equiv$ $\exists (F[x] \land G[x])$ |
| $\exists F[x] \lor \exists G[x]$ $\equiv$ $\exists (F[x] \lor G[x])$ | $\exists F[x] \land \exists G[x]$ $\not\equiv$ $\exists (F[x] \land G[x])$ |

Which implications do not hold in the $\not\equiv$ above?
### Equivalences of Formulas

<table>
<thead>
<tr>
<th>Equivalence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F \iff G \equiv (F \Rightarrow G) \land (G \Rightarrow F) )</td>
<td>( F \Rightarrow G \equiv \neg F \lor G )</td>
</tr>
<tr>
<td>( F \lor G \equiv G \lor F )</td>
<td>( F \lor (G \land H) \equiv (F \lor G) \land (F \lor H) )</td>
</tr>
<tr>
<td>( (F \lor G) \lor H \equiv F \lor (G \lor H) )</td>
<td>( F \lor T \equiv T )</td>
</tr>
<tr>
<td>( F \lor \neg F \equiv T )</td>
<td>( F \lor F \equiv F )</td>
</tr>
<tr>
<td>( \neg (\neg F) \equiv F )</td>
<td>( \neg (F \lor G) \equiv \neg F \land \neg G )</td>
</tr>
<tr>
<td>( (Qx)F[x] \lor G \equiv (Qx)(F[x] \lor G) )</td>
<td>( (Qx)F[x] \lor G \equiv (Qx)(F[x] \land G) )</td>
</tr>
<tr>
<td>( \neg \forall F[x] \equiv \exists \neg F[x] )</td>
<td>( \neg (\exists x)F[x] \equiv \forall \neg F[x] )</td>
</tr>
<tr>
<td>( \forall F[x] \lor \forall G[x] \neq \forall (F[x] \lor G[x]) )</td>
<td>( \forall F[x] \land \forall G[x] \equiv \forall (F[x] \land G[x]) )</td>
</tr>
<tr>
<td>( \exists F[x] \lor \exists G[x] \equiv \exists (F[x] \lor G[x]) )</td>
<td>( \exists F[x] \land \exists G[x] \neq \exists (F[x] \land G[x]) )</td>
</tr>
</tbody>
</table>

Which implications do not hold in the \( \neq \) above?
### Equivalences of Formulas

<table>
<thead>
<tr>
<th>$F \iff G$</th>
<th>$\equiv (F \implies G) \land (G \implies F)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F \implies G$</td>
<td>$\equiv \neg F \lor G$</td>
</tr>
<tr>
<td>$F \lor G$</td>
<td>$\equiv G \lor F$</td>
</tr>
<tr>
<td>$(F \lor G) \lor H$</td>
<td>$\equiv F \lor (G \lor H)$</td>
</tr>
<tr>
<td>$F \lor (G \land H)$</td>
<td>$\equiv (F \lor G) \land (F \lor H)$</td>
</tr>
<tr>
<td>$F \lor \top$</td>
<td>$\equiv \top$</td>
</tr>
<tr>
<td>$F \lor \bot$</td>
<td>$\equiv \bot$</td>
</tr>
<tr>
<td>$\neg (\neg F)$</td>
<td>$\equiv F$</td>
</tr>
<tr>
<td>$\neg (F \lor G)$</td>
<td>$\equiv \neg F \land \neg G$</td>
</tr>
<tr>
<td>$(Qx)F[x] \lor G$</td>
<td>$\equiv (Qx)(F[x] \lor G)$</td>
</tr>
<tr>
<td>$\neg \forall F[x]$</td>
<td>$\equiv \exists \neg F[x]$</td>
</tr>
<tr>
<td>$\forall F[x] \lor \forall G[x]$</td>
<td>$\not\equiv \forall (F[x] \lor G[x])$</td>
</tr>
<tr>
<td>$\exists F[x] \lor \exists G[x]$</td>
<td>$\equiv \exists (F[x] \lor G[x])$</td>
</tr>
</tbody>
</table>

| $F \land G$ | $\equiv G \land F$ |
| $(F \land G) \land H$ | $\equiv F \land (G \land H)$ |
| $F \land (G \lor H)$ | $\equiv (F \land G) \lor (F \land H)$ |
| $F \land \top$ | $\equiv F$ |
| $F \land \bot$ | $\equiv \bot$ |
| $\neg (F \land G)$ | $\equiv \neg F \lor \neg G$ |
| $(Qx)F[x] \land G$ | $\equiv (Qx)(F[x] \land G)$ |
| $\neg (\exists x)F[x]$ | $\equiv \forall \neg F[x]$ |
| $\forall F[x] \land \forall G[x]$ | $\equiv \forall (F[x] \land G[x])$ |
| $\exists F[x] \land \exists G[x]$ | $\equiv \exists (F[x] \land G[x])$ |

Which implications do not hold in the $\not\equiv$ above?
Equivalences of Formulas (cont’d)

Note that

\[ \forall x F[x] \lor \forall x G[x] \equiv \forall x F[x] \lor \forall y G[y] \equiv \forall x F[x] \lor G[y] \]

\[ \exists x F[x] \land \exists x G[x] \equiv \exists x F[x] \land \exists y G[y] \equiv \exists x F[x] \land G[y] \]
Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Formula Clausification

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 $\neg$, $\land$, and $\lor$ 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 $(Q_1x_1)\ldots(Q_nx_n)M$, 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 $\forall_{x_1,\ldots,x_n}M$, 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 $\neg$, $\land$, and $\lor$ 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 $(Q_1 x_1) \ldots (Q_n x_n) M$, 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 $\forall_{x_1, \ldots, x_n} M$, 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 $\neg$, $\land$, and $\lor$ 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 $(Q_1x_1)...(Q_nx_n) M$, 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 $\forall_{x_1,...,x_n} M$, 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 $\neg$, $\land$, and $\lor$ 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 $\forall x_1, \ldots, x_n M$, 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 $\forall x_1, \ldots, x_n M$, 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 ¬, ∧, and ∨ 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 \((Q_1x_1)...(Q_nx_n) M\), 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 \( \forall_{x_1,...,x_n} M \), 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 $\neg$, $\land$, and $\lor$ 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 $(Q_1x_1)...(Q_nx_n)M$, 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 $\forall_{x_1,\ldots,x_n}M$, 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 $\neg$, $\land$, and $\lor$ 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 $(Q_1 x_1) \ldots (Q_n x_n) M$, 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 $\forall x_1, \ldots, x_n M$, 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 ¬, ∧, and ∨ 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 $\left( Q_1 x_1 \right) \ldots \left( Q_n x_n \right) M$, 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 $\forall_{x_1, \ldots, x_n} M$, 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 $\neg$, $\land$, and $\lor$ 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 $(Q_1x_1)...(Q_nx_n) M$, 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 $\forall_{x_1,...,x_n} M$, where $M$ is a quantifier-free formula in CNF.
Normal Forms (cont’d)

Examples:

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

\[
\left( \forall x P[x] \right) \Rightarrow Q \equiv \exists x (P[x] \Rightarrow Q).
\]

2. Bring the following formulas into Skolem standard form

\[
\forall x \exists y, z \left( \neg P[x, y] \land Q[x, z] \right) \lor R[x, y, z]
\]

\[
\forall x, y \left( \exists z P[x, z] \land P[y, z] \right) \Rightarrow \exists u Q[x, y, u]
\]
Normal Forms (cont’d)

Examples:

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

\[ \left( \forall x P[x] \right) \Rightarrow Q \equiv \exists x (P[x] \Rightarrow Q). \]

2. Bring the following formulas into Skolem standard form

\[ \forall x \exists y, z (\neg P[x, y] \land Q[x, z]) \lor R[x, y, z] \]

\[ \forall x, y \left( \exists z P[x, z] \land P[y, z] \right) \Rightarrow \exists u Q[x, y, u] \]
Normal Forms (cont’d)

Examples:

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

\[
\left( \forall x \, P[x] \right) \Rightarrow Q \equiv \exists x \, (P[x] \Rightarrow Q).
\]

2. Bring the following formulas into Skolem standard form

\[
\forall x \, \exists y, z \, ((\neg P[x, y] \land Q[x, z]) \lor R[x, y, z])
\]

\[
\forall x, y \left( \exists z \, P[x, z] \land P[y, z] \right) \Rightarrow \exists u \, Q[x, y, u]
\]
Normal Forms (cont’d)

Examples:

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

\[
\left( \forall \ P[x] \right) \Rightarrow Q \equiv \exists \ (P[x] \Rightarrow Q).
\]

2. Bring the following formulas into Skolem standard form

\[
\forall \ (\exists \ z \ (\neg P[x, y] \land Q[x, z]) \lor R[x, y, z])
\]

\[
\forall \ (\exists \ z \ (\exists P[x, z] \land P[y, z]) \Rightarrow \exists \ Q[x, y, u])
\]
Normal Forms (cont’d)

Examples:

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

\[
(\forall P[x]) \Rightarrow Q \equiv \exists_x (P[x] \Rightarrow Q).
\]

2. Bring the following formulas into Skolem standard form

\[
\forall x, \exists y, z ((\neg P[x, y] \land Q[x, z]) \lor R[x, y, z])
\]

\[
\forall x, y \left( \exists z (P[x, z] \land P[y, z]) \Rightarrow \exists u Q[x, y, u] \right)
\]
Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

**Formula Clausification**

Substitution
**Formula Clausification**

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

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

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

$$\forall \exists (\neg P[x, y] \land Q[x, z]) \lor R[x, y, z])$$

The standard form of the formula above, that is

$$\forall \neg P[x, f[x]] \lor R[x, f[x], g[x]]) \land (Q(x, g[x]) \lor R[x, f[x], g[x]]))$$

can be represented by the following set of clauses

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

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: \( \neg P[x] \lor Q[y, f[x]], P[x] \)

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

\[
\forall x \exists y, z ((\neg P[x, y] \land Q[x, z]) \lor R[x, y, z])
\]

The standard form of the formula above, that is

\[
\forall x ((\neg P[x, f[x]] \lor R[x, f[x], g[x]]) \land (Q(x, g[x]) \lor R[x, f[x], g[x]]))
\]

can be represented by the following set of clauses

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

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:** \( \neg P[x] \lor Q[y, f[x]], P[x] \)

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

\[
\forall \exists \exists_{x, y, z} ((\neg P[x, y] \land Q[x, z]) \lor R[x, y, z])
\]

The standard form of the formula above, that is

\[
\forall_{x} ((\neg P[x, f[x]] \lor R[x, f[x], g[x]]) \land (Q(x, g[x]) \lor R[x, f[x], g[x]]))
\]

can be represented by the following set of clauses

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

**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: $\neg P[x] \lor Q[y, f[x]], P[x]$.

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

$$
\forall x \exists y, z ((\neg P[x, y] \land Q[x, z]) \lor R[x, y, z])
$$

The standard form of the formula above, that is

$$
\forall x ((\neg P[x, f[x]] \lor R[x, f[x], g[x]]) \land (Q(x, g[x]) \lor R[x, f[x], g[x]]))
$$

can be represented by the following set of clauses

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

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: \( \neg P[x] \lor Q[y, f[x]], P[x] \)

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

\[
\forall x \exists y, z ( (\neg P[x, y] \land Q[x, z]) \lor R[x, y, z] )
\]

The standard form of the formula above, that is

\[
\forall x ( (\neg P[x, f[x]] \lor R[x, f[x], g[x]]) \land (Q(x, g[x]) \lor R[x, f[x], g[x]]) )
\]

can be represented by the following set of clauses

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

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.
Formulas Clausification (cont’d)

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

$F_1$:
$$\forall_{x,y} \exists_z P[x, y, z]$$
$$\forall_{x,y,z,u,v,w} \left( (P[x, y, u] \land P[y, z, v] \land P[u, z, w]) \Rightarrow P[x, v, w] \right)$$

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

$F_3$:
$$\forall_{x} P[x, e, x] \land \forall_{x} P[e, x, x]$$

$F_4$:
$$\forall_{x} P[x, i[x], e] \land \forall_{x} P[i[x], x, e]$$

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

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Formula Clausification

Substitution
Substitution

Example: Let

\[ C_1 : \quad P[x] \lor Q[x] \]
\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]
**Substitution**

Example: Let

\[ C_1 : \quad P[x] \lor Q[x] \]

\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]
Example: Let

\[ C_1 : \quad P[x] \lor Q[x] \]
\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]

Let \( x \rightarrow f[a] \) in \( C_1 \), \( x \rightarrow a \) in \( C_2 \).
Substitution

Example: Let

\[ C_1 : \quad P[x] \lor Q[x] \]
\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]

Let \( x \to f[a] \) in \( C_1 \), \( x \to a \) in \( C_2 \).

We have

\[ C'_1 : \quad P[f[a]] \lor Q[f[a]] \]
\[ C'_2 : \quad \neg P[f[a]] \lor R[a] \]
Example: Let

\[ C_1 : \quad P[x] \lor Q[x] \]
\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]

Let \( x \to f[a] \) in \( C_1 \), \( x \to a \) in \( C_2 \).

We have

\[ C_1' : \quad P[f[a]] \lor Q[f[a]] \]
\[ C_2' : \quad \neg P[f[a]] \lor R[a] \]

\( C_1' \) and \( C_2' \) are ground instances.
Substitution

Example: Let

\[ C_1 : \quad P[x] \lor Q[x] \]
\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]

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

We have

\[ C'_1 : \quad P[f[a]] \lor Q[f[a]] \]
\[ C'_2 : \quad \neg P[f[a]] \lor R[a] \]

\( C'_1 \) and \( C'_2 \) are ground instances.

A resolvent of \( C'_1 \) and \( C'_2 \) is

\[ C'_3 : \quad Q[f[a]] \lor R[a] \]
Example: Let

\[ C_1 : \quad P[x] \lor Q[x] \]
\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]

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

\[ C_1^* : \quad P[f[x]] \lor Q[f[x]] \]

\( C_1^* \) is an instance of \( C_1 \).

A resolvent of

\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]
\[ C_1^* : \quad P[f[x]] \lor Q[f[x]] \]

is

\[ C_3 : \quad Q[f[x]] \lor R[x] \]

\( C_3' \) is an instance of \( C_3 \). \( C_3 \) is the most general clause.
Example: Let

\[ C_1 : \quad P[x] \lor Q[x] \]
\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]

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

\[ C^*_1 : \quad P[f[x]] \lor Q[f[x]] \]

\( C^*_1 \) is an instance of \( C_1 \).

A resolvent of

\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]
\[ C^*_1 : \quad P[f[x]] \lor Q[f[x]] \]

is

\[ C_3 : \quad Q[f[x]] \lor R[x] \]

\( C'_3 \) is an instance of \( C_3 \). \( C_3 \) is the most general clause.
**Substitution**

**Example:** Let

\[ C_1 : \quad P[x] \lor Q[x] \]
\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]

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

\[ C_1^* : \quad P[f[x]] \lor Q[f[x]] \]

\( C_1^* \) is an *instance* of \( C_1 \).

A resolvent of

\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]
\[ C_1^* : \quad P[f[x]] \lor Q[f[x]] \]

is

\[ C_3 : \quad Q[f[x]] \lor R[x] \]

\( C_3' \) is an instance of \( C_3 \). \( C_3 \) is the *most general clause*. 
Example: Let

\[ C_1 : P[x] \lor Q[x] \]
\[ C_2 : \neg P[f[x]] \lor R[x] \]

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

\[ C_1^* : P[f[x]] \lor Q[f[x]] \]

\( C_1^* \) is an instance of \( C_1 \).

A resolvent of

\[ C_2 : \neg P[f[x]] \lor R[x] \]
\[ C_1^* : P[f[x]] \lor Q[f[x]] \]

is

\[ C_3 : Q[f[x]] \lor R[x] \]

\( C_3^* \) is an instance of \( C_3 \). \( C_3 \) is the most general clause.
Substitution

Example: Let

\[ C_1 : \quad P[x] \lor Q[x] \]
\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]

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

\[ C_1^* : \quad P[f[x]] \lor Q[f[x]] \]

\( C_1^* \) is an instance of \( C_1 \).

A resolvent of

\[ C_2 : \quad \neg P[f[x]] \lor R[x] \]
\[ C_1^* : \quad P[f[x]] \lor Q[f[x]] \]

is

\[ C_3 : \quad Q[f[x]] \lor R[x] \]

\( C_3' \) 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 $\{v_1 \rightarrow t_1, \ldots, v_n \rightarrow t_n\}$ 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 $E\sigma = f[h[a, y], a, g[z], y]$. 
A substitution \( \sigma \) is a finite set of the form \( \{ v_1 \rightarrow t_1, \ldots, v_n \rightarrow t_n \} \) 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 \( E\sigma = f[h[a, y], a, g[z], y] \).
A substitution $\sigma$ is a finite set of the form $\{v_1 \rightarrow t_1, ..., v_n \rightarrow t_n\}$ 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 $E\sigma = f[h[a, y], a, g[z], y]$. 
Let

\[ \theta = \{ x_1 \rightarrow t_1, \ldots, x_n \rightarrow t_n \} \]
\[ \lambda = \{ y_1 \rightarrow u_1, \ldots, y_n \rightarrow u_n \} \]

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

\[ \{ x_1 \rightarrow t_1 \lambda, \ldots, x_n \rightarrow t_n \lambda, y_1 \rightarrow u_1, \ldots, y_n \rightarrow u_n \} \]

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 \( \{ x_1, \ldots, x_n \} \).
**Substitution (cont’d)**

Example 1:

\[ \theta = \{ x \rightarrow f[y], y \rightarrow z \} \]
\[ \lambda = \{ x \rightarrow a, y \rightarrow b, z \rightarrow y \} \]

Then

\[ \theta \circ \lambda = \{ x \rightarrow f[b], y \rightarrow y, x \rightarrow a, y \rightarrow b, z \rightarrow y \} \]
\[ = \{ x \rightarrow f[b], z \rightarrow y \} \]

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

\[ \theta_1 \circ \theta_2 = \{ x \rightarrow a, y \rightarrow f[g[x]], z \rightarrow z, x \rightarrow b, y \rightarrow z, z \rightarrow g[x] \} \]
\[ = \{ x \rightarrow a, y \rightarrow f[g[x]] \} \]
Substitution (cont’d)

Example 1:

\[ \theta = \{x \rightarrow f[y], y \rightarrow z\} \]
\[ \lambda = \{x \rightarrow a, y \rightarrow b, z \rightarrow y\} \]

Then

\[ \theta \circ \lambda = \{x \rightarrow f[b], y \rightarrow y, x \rightarrow a, y \rightarrow b, z \rightarrow y\} \]
\[ = \{x \rightarrow f[b], z \rightarrow y\} \]

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

\[ \theta_1 \circ \theta_2 = \{x \rightarrow a, y \rightarrow f[g[x]], z \rightarrow z, x \rightarrow b, y \rightarrow z, z \rightarrow g[x]\} \]
\[ = \{x \rightarrow a, y \rightarrow f[g[x]]\} \]
Substitution (cont’d)

Example 1:

\[ \theta = \{x \to f[y], y \to z\} \]
\[ \lambda = \{x \to a, y \to b, z \to y\} \]

Then

\[ \theta \circ \lambda = \{x \to f[b], y \to y, x \to a, y \to b, z \to y\} \]
\[ = \{x \to f[b], z \to y\} \]

Example 2:

\[ \theta_1 = \{x \to a, y \to f[z], z \to y\} \]
\[ \theta_2 = \{x \to b, y \to z, z \to g[x]\} \]

Then

\[ \theta_1 \circ \theta_2 = \{x \to a, y \to f[g[x]], z \to z, x \to b, y \to z, z \to g[x]\} \]
\[ = \{x \to a, y \to f[g[x]]\} \]
Substitution (cont’d)

Example 1:

\[ \theta = \{ x \rightarrow f[y], y \rightarrow z \} \]
\[ \lambda = \{ x \rightarrow a, y \rightarrow b, z \rightarrow y \} \]

Then

\[ \theta \circ \lambda = \{ x \rightarrow f[b], y \rightarrow y, x \rightarrow a, y \rightarrow b, z \rightarrow y \} \]
\[ = \{ x \rightarrow f[b], z \rightarrow y \} \]

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

\[ \theta_1 \circ \theta_2 = \{ x \rightarrow a, y \rightarrow f[g[x]], z \rightarrow z, x \rightarrow b, y \rightarrow z, z \rightarrow g[x] \} \]
\[ = \{ x \rightarrow a, y \rightarrow f[g[x]] \} \]