Automated Theorem Proving (Demonstrarea Automata a Teoremelor)

First-Order Logic

Mădălina Erașcu[1] Tudor Jebelean[2]

[1]West University of Timișoara and Institute e-Austria Timișoara
bvd. V. Parvan 4, Timișsoara, Romania

[2]Research Institute for Symbolic Computation,
Johannes Kepler University, Linz, Austria

merascu@info.uvt.ro tjebelea@risc.jku.at

May 6 and 8, 2014

Outline

Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

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.

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.

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.

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.

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.

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.

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.

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.

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.

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \implies G$, and $F \iff G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\ x + 1 \geq x$
2. $\neg \left(\underset{x}{\exists}\ E[0, f[x]] \right)$
3. $\underset{x}{\forall} \underset{y}{\exists} \Big(E[y, f[x]] \ \wedge\ \underset{z}{\forall} \big(E[z, f[x]] \Rightarrow E[y, z] \big) \Big)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \implies G$, and $F \iff G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\, x + 1 \geq x$
2. $\neg \left(\underset{x}{\exists}\, E[0, f[x]] \right)$
3. $\underset{x}{\forall} \underset{y}{\exists} \Big(E[y, f[x]] \wedge \underset{z}{\forall} \big(E[z, f[x]] \Rightarrow E[y, z] \big) \Big)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \implies G$, and $F \iff G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\; x + 1 \geq x$
2. $\neg\left(\underset{x}{\exists}\; E[0, f[x]]\right)$
3. $\underset{x}{\forall}\underset{y}{\exists}\Big(E[y, f[x]] \;\wedge\; \underset{z}{\forall}\big(E[z, f[x]] \Rightarrow E[y, z]\big)\Big)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \implies G$, and $F \iff G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\ x + 1 \geq x$
2. $\neg \left(\underset{x}{\exists}\ E[0, f[x]] \right)$
3. $\underset{x}{\forall} \underset{y}{\exists} \left(E[y, f[x]] \ \wedge \ \underset{z}{\forall} (E[z, f[x]] \Rightarrow E[y, z]) \right)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \implies G$, and $F \Longleftrightarrow G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\, x + 1 \geq x$
2. $\neg\left(\underset{x}{\exists}\, E[0, f[x]]\right)$
3. $\underset{x}{\forall}\underset{y}{\exists}\left(E[y, f[x]] \ \wedge\ \underset{z}{\forall}\left(E[z, f[x]] \Rightarrow E[y, z]\right)\right)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \Longrightarrow G$, and $F \Longleftrightarrow G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\ x + 1 \geq x$
2. $\neg \left(\underset{x}{\exists}\ E[0, f[x]] \right)$
3. $\underset{x}{\forall} \underset{y}{\exists} \Big(E[y, f[x]] \ \wedge\ \underset{z}{\forall} \big(E[z, f[x]] \Rightarrow E[y, z] \big) \Big)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \Longrightarrow G$, and $F \Longleftrightarrow G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\ x + 1 \geq x$
2. $\neg\left(\underset{x}{\exists}\ E[0, f[x]]\right)$
3. $\underset{x}{\forall}\underset{y}{\exists}\left(E[y, f[x]] \ \wedge\ \underset{z}{\forall}\left(E[z, f[x]] \Rightarrow E[y, z]\right)\right)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \Longrightarrow G$, and $F \Longleftrightarrow G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\, x + 1 \geq x$
2. $\neg \left(\underset{x}{\exists}\, E[0, f[x]] \right)$
3. $\underset{x}{\forall} \underset{y}{\exists} \Big(E[y, f[x]] \;\wedge\; \underset{z}{\forall} (E[z, f[x]] \Rightarrow E[y, z]) \Big)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \implies G$, and $F \iff G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\, x + 1 \geq x$
2. $\neg \left(\underset{x}{\exists}\, E[0, f[x]] \right)$
3. $\underset{x}{\forall}\, \underset{y}{\exists} \left(E[y, f[x]] \;\wedge\; \underset{z}{\forall} (E[z, f[x]] \Rightarrow E[y, z]) \right)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \Longrightarrow G$, and $F \Longleftrightarrow G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\, x + 1 \geq x$
2. $\neg \left(\underset{x}{\exists}\, E[0, f[x]] \right)$
3. $\underset{x}{\forall}\, \underset{y}{\exists} \left(E[y, f[x]] \;\wedge\; \underset{z}{\forall} (E[z, f[x]] \Rightarrow E[y, z]) \right)$

Syntax (cont'd)

Formulas are defined as follows:

1. An atom is a formula.
2. If F and G are formulas then $\neg F$, $F \vee G$, $F \wedge G$, $F \implies G$, and $F \Longleftrightarrow G$ are formulas.
3. If F is a formula and x is a variable, then $\underset{x}{\forall} F$ and $\underset{x}{\exists} F$ 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 $\underset{x}{\forall}$ or $\underset{x}{\exists}$.

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. $\underset{x}{\forall}\ x + 1 \geq x$
2. $\neg \left(\underset{x}{\exists}\ E[0, f[x]] \right)$
3. $\underset{x}{\forall}\ \underset{y}{\exists} \Big(E[y, f[x]] \ \wedge\ \underset{z}{\forall} (E[z, f[x]] \Rightarrow E[y, z]) \Big)$

Outline

- Syntax
- **Semantics**
- (Un)Satisfiability & (In)Validity
- Equivalences of Formulas
- Normal Forms

Semantics

Excursion: Semantics of propositional logic.

The semantics of a formula F is a function f_F, $f_F : \mathcal{I} \rightarrow \{\mathbb{T}, \mathbb{F}\}$, where $\mathcal{I}$ is the set of all interpretations I defined as bellow.

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 $\{\mathbb{T}, \mathbb{F}\}$.

Semantics

Excursion: Semantics of propositional logic.

The semantics of a formula F is a function f_F, $f_F : \mathcal{I} \rightarrow \{\mathbb{T}, \mathbb{F}\}$, where $\mathcal{I}$ is the set of all interpretations I defined as bellow.

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 $\{\mathbb{T}, \mathbb{F}\}$.

Semantics

Excursion: Semantics of propositional logic.

The semantics of a formula F is a function f_F, $f_F : \mathcal{I} \rightarrow \{\mathbb{T}, \mathbb{F}\}$, where $\mathcal{I}$ is the set of all interpretations I defined as bellow.

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 $\{\mathbb{T}, \mathbb{F}\}$.

Semantics

Excursion: Semantics of propositional logic.

The semantics of a formula F is a function f_F, $f_F : \mathcal{I} \to \{\mathbb{T}, \mathbb{F}\}$, where $\mathcal{I}$ is the set of all interpretations I defined as bellow.

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 $\{\mathbb{T}, \mathbb{F}\}$.

Semantics

Excursion: Semantics of propositional logic.

The semantics of a formula F is a function f_F, $f_F : \mathcal{I} \rightarrow \{\mathbb{T}, \mathbb{F}\}$, where $\mathcal{I}$ is the set of all interpretations I defined as bellow.

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 $\{\mathbb{T}, \mathbb{F}\}$.

Semantics

Excursion: Semantics of propositional logic.

The semantics of a formula F is a function f_F, $f_F : \mathcal{I} \rightarrow \{\mathbb{T}, \mathbb{F}\}$, where $\mathcal{I}$ is the set of all interpretations I defined as bellow.

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 $\{\mathbb{T}, \mathbb{F}\}$.

Semantics (cont'd)

Truth Evaluation of $\langle\varphi\rangle_I$

Formula

$\mathbb{T}$, $\mathbb{F}$	
$P(t_1, \ldots, t_n)$	$\langle P(t_1, \ldots, t_n)\rangle^I_\sigma = P_I(\langle t_1\rangle^I_\sigma, \ldots, \langle t_n\rangle^I_\sigma)$
$\neg\varphi$	$\langle\neg\varphi\rangle^I_\sigma = \mathcal{B}_\neg(\langle\varphi\rangle^I_\sigma)$
$\varphi \underbrace{\bowtie}_{\in\{\wedge,\vee,\Rightarrow,\Longleftrightarrow\}} \psi$	$\langle\varphi \bowtie \psi\rangle^I_\sigma = \mathcal{B}_\bowtie(\langle\varphi\rangle^I_\sigma, \langle\psi\rangle^I_\sigma)$
$\underset{\upsilon}{\forall}\ \varphi$ (υ – variable symbol)	$\left\langle \underset{\upsilon}{\forall}\ \varphi \right\rangle^I_\sigma = \mathbb{T}$ iff (for each $d \in D$, $\langle\varphi\rangle^I_{\sigma\uplus\{\upsilon\leftarrow d\}} = \mathbb{T}$)
$\underset{\upsilon}{\exists}\ \varphi$ (υ – variable symbol)	$\left\langle \underset{\upsilon}{\exists}\ \varphi \right\rangle^I_\sigma = \mathbb{T}$ iff (for some $d \in D$, $\langle\varphi\rangle^I_{\sigma\uplus\{\upsilon\leftarrow d\}} = \mathbb{T}$)

(the domain D of the interpretation I)

Semantics (cont'd)

Truth Evaluation of $\langle\varphi\rangle_I$

Term

υ ($\in$ variable symbol set)	$\langle\upsilon\rangle^I_\sigma = \langle\upsilon\rangle^I_{\{\ldots\upsilon\leftarrow d\ldots\}} = d$ (assuming there is such an assignment $\upsilon \leftarrow d$)
c ($\in$ constant symbol set)	$\langle c\rangle^I_\sigma = c_I$
f ($\in$ function symbol set)	$\langle f(t_1, \ldots, t_n)\rangle^I_\sigma = f_I(\langle t_1\rangle^I_\sigma, \ldots, \langle t_n\rangle^I_\sigma)$

Semantics (cont'd)

Example: Find the truth value of the formulas:

- $F_1 \; :\Longleftrightarrow \underset{x}{\forall} \underset{y}{\forall} x \leq y$, where $I : \begin{cases} D = \{0,1\} \\ \leq_I \; \rightarrow \; \leq_{\mathbb{Z}} \end{cases}$
- $F_2 \; :\Longleftrightarrow \underset{x}{\forall} \underset{y}{\exists} x + y > c$, where $I : \begin{cases} D = \{0,1\} \\ c_I = 0 \\ +_I \; \rightarrow \; +_{\mathbb{Z}} \\ >_I \; \rightarrow \; >_{\mathbb{Z}} \end{cases}$
- $F_3 \; :\Longleftrightarrow \underset{x}{\forall} \left(P[x] \Longrightarrow Q[f[x], a]\right)$, where

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

Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

(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 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.

(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 formulas F_1, F_2, ..., F_n iff for every interpretation I, if $F_1 \wedge F_2 \wedge \ldots \wedge F_n$ is true in I, G is also true in I.

Note that validity and satisfiability applies to closed formulas.

(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 formulas F_1, F_2, ..., F_n iff for every interpretation I, if $F_1 \wedge F_2 \wedge \ldots \wedge F_n$ is true in I, G is also true in I.

Note that validity and satisfiability applies to closed formulas.

(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 formulas F_1, F_2, ..., F_n iff for every interpretation I, if $F_1 \wedge F_2 \wedge \ldots \wedge F_n$ is true in I, G is also true in I.

Note that validity and satisfiability applies to closed formulas.

(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 formulas F_1, F_2, ..., F_n iff for every interpretation I, if $F_1 \wedge F_2 \wedge \ldots \wedge F_n$ is true in I, G is also true in I.

Note that validity and satisfiability applies to closed formulas.

(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 formulas F_1, F_2, ..., F_n iff for every interpretation I, if $F_1 \wedge F_2 \wedge \ldots \wedge F_n$ is true in I, G is also true in I.

Note that validity and satisfiability applies to closed formulas.

Outline

Syntax

Semantics

(Un)Satisfiability & (In)Validity

Equivalences of Formulas

Normal Forms

Equivalences of Formulas

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

$$F \Longleftrightarrow G \equiv (F \Rightarrow G) \wedge (G \Rightarrow F)$$
$$F \Rightarrow G \equiv \neg F \vee G$$
$$F \vee G \equiv G \vee F$$
$$(F \vee G) \vee H \equiv F \vee (G \vee H)$$
$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$$
$$F \vee \mathbb{T} \equiv \mathbb{T}$$
$$F \vee \mathbb{F} \equiv F$$
$$F \vee \neg F \equiv \mathbb{T}$$
$$\neg(\neg F) \equiv F$$
$$\neg(F \vee G) \equiv \neg F \wedge \neg G$$
$$(Qx) F[x] \vee G \equiv (Qx)(F[x] \vee G)$$
$$\neg \underset{x}{\forall} F[x] \equiv \underset{x}{\exists} \neg F[x]$$
$$\underset{x}{\forall} F[x] \vee \underset{x}{\forall} G[x] \not\equiv \underset{x}{\forall}(F[x] \vee G[x])$$
$$\underset{x}{\exists} F[x] \vee \underset{x}{\exists} G[x] \equiv \underset{x}{\exists}(F[x] \vee G[x])$$

$$F \wedge G \equiv G \wedge F$$
$$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$$
$$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$
$$F \wedge \mathbb{T} \equiv F$$
$$F \wedge \mathbb{F} \equiv \mathbb{F}$$
$$F \wedge \neg F \equiv \mathbb{F}$$
$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$
$$(Qx) F[x] \wedge G \equiv (Qx)(F[x] \wedge G)$$
$$\neg(\underset{x}{\exists} x) F[x] \equiv \underset{x}{\forall} \neg F[x]$$
$$\underset{x}{\forall} F[x] \wedge \underset{x}{\forall} G[x] \equiv \underset{x}{\forall}(F[x] \wedge G[x])$$
$$\underset{x}{\exists} F[x] \wedge \underset{x}{\exists} G[x] \not\equiv \underset{x}{\exists}(F[x] \wedge G[x])$$

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

$\Longrightarrow$ ✓

$\not\Longrightarrow$

Equivalences of Formulas

$F \Longleftrightarrow G \quad \equiv \quad (F \Rightarrow G) \wedge (G \Rightarrow F)$	
$F \Rightarrow G \quad \equiv \quad \neg F \vee G$	
$F \vee G \quad \equiv \quad G \vee F$	$F \wedge G \quad \equiv \quad G \wedge F$
$(F \vee G) \vee H \quad \equiv \quad F \vee (G \vee H)$	$(F \wedge G) \wedge H \quad \equiv \quad F \wedge (G \wedge H)$
$F \vee (G \wedge H) \quad \equiv \quad (F \vee G) \wedge (F \vee H)$	$F \wedge (G \vee H) \quad \equiv \quad (F \wedge G) \vee (F \wedge H)$
$F \vee \mathbb{T} \quad \equiv \quad \mathbb{T}$	$F \wedge \mathbb{T} \quad \equiv \quad F$
$F \vee \mathbb{F} \quad \equiv \quad F$	$F \wedge \mathbb{F} \quad \equiv \quad \mathbb{F}$
$F \vee \neg F \quad \equiv \quad \mathbb{T}$	$F \wedge \neg F \quad \equiv \quad \mathbb{F}$
$\neg(\neg F) \quad \equiv \quad F$	
$\neg(F \vee G) \quad \equiv \quad \neg F \wedge \neg G$	$\neg(F \wedge G) \quad \equiv \quad \neg F \vee \neg G$
$(Qx)F[x] \vee G \quad \equiv \quad (Qx)(F[x] \vee G)$	$(Qx)F[x] \wedge G \quad \equiv \quad (Qx)(F[x] \wedge G)$
$\neg \underset{x}{\forall} F[x] \quad \equiv \quad \underset{x}{\exists} \neg F[x]$	$\neg(\underset{x}{\exists} x)F[x] \quad \equiv \quad \underset{x}{\forall} \neg F[x]$
$\underset{x}{\forall} F[x] \;\vee\; \underset{x}{\forall} G[x] \quad \not\equiv \quad \underset{x}{\forall}(F[x] \;\vee\; G[x])$	$\underset{x}{\forall} F[x] \wedge \underset{x}{\forall} G[x] \quad \equiv \quad \underset{x}{\forall}(F[x] \wedge G[x])$
$\underset{x}{\exists} F[x] \vee \underset{x}{\exists} G[x] \quad \equiv \quad \underset{x}{\exists}(F[x] \vee G[x])$	$\underset{x}{\exists} F[x] \;\wedge\; \underset{x}{\exists} G[x] \quad \not\equiv \quad \underset{x}{\exists}(F[x] \;\wedge\; G[x])$

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

$\Longrightarrow$	$\not\Longrightarrow$
$\not\Longleftarrow$	$\Longleftarrow$

Equivalences of Formulas

$$
\begin{aligned}
F \Longleftrightarrow G &\equiv (F \Rightarrow G) \wedge (G \Rightarrow F) \\
F \Rightarrow G &\equiv \neg F \vee G \\
F \vee G &\equiv G \vee F \\
(F \vee G) \vee H &\equiv F \vee (G \vee H) \\
F \vee (G \wedge H) &\equiv (F \vee G) \wedge (F \vee H) \\
F \vee \mathbb{T} &\equiv \mathbb{T} \\
F \vee \mathbb{F} &\equiv F \\
F \vee \neg F &\equiv \mathbb{T} \\
\neg(\neg F) &\equiv F \\
\neg(F \vee G) &\equiv \neg F \wedge \neg G \\
(Qx)F[x] \vee G &\equiv (Qx)(F[x] \vee G) \\
\neg \forall_x F[x] &\equiv \exists_x \neg F[x] \\
\forall_x F[x] \vee \forall_x G[x] &\not\equiv \forall_x (F[x] \vee G[x]) \\
\exists_x F[x] \vee \exists_x G[x] &\equiv \exists_x (F[x] \vee G[x])
\end{aligned}
$$

$$
\begin{aligned}
F \wedge G &\equiv G \wedge F \\
(F \wedge G) \wedge H &\equiv F \wedge (G \wedge H) \\
F \wedge (G \vee H) &\equiv (F \wedge G) \vee (F \wedge H) \\
F \wedge \mathbb{T} &\equiv F \\
F \wedge \mathbb{F} &\equiv \mathbb{F} \\
F \wedge \neg F &\equiv \mathbb{F} \\
\neg(F \wedge G) &\equiv \neg F \vee \neg G \\
(Qx)F[x] \wedge G &\equiv (Qx)(F[x] \wedge G) \\
\neg(\exists x)F[x] &\equiv \forall_x \neg F[x] \\
\forall_x F[x] \wedge \forall_x G[x] &\equiv \forall_x (F[x] \wedge G[x]) \\
\exists_x F[x] \wedge \exists_x G[x] &\not\equiv \exists_x (F[x] \wedge G[x])
\end{aligned}
$$

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

$\Longrightarrow$ $\quad\quad$ $\not\Longrightarrow$

$\not\Longleftarrow$ $\quad\quad$ $\Longleftarrow$

Equivalences of Formulas

$$
\begin{array}{rcl||rcl}
F \Longleftrightarrow G & \equiv & (F \Rightarrow G) \wedge (G \Rightarrow F) & & & \\
F \Rightarrow G & \equiv & \neg F \vee G & & & \\
F \vee G & \equiv & G \vee F & F \wedge G & \equiv & G \wedge F \\
(F \vee G) \vee H & \equiv & F \vee (G \vee H) & (F \wedge G) \wedge H & \equiv & F \wedge (G \wedge H) \\
F \vee (G \wedge H) & \equiv & (F \vee G) \wedge (F \vee H) & F \wedge (G \vee H) & \equiv & (F \wedge G) \vee (F \wedge H) \\
F \vee \mathbb{T} & \equiv & \mathbb{T} & F \wedge \mathbb{T} & \equiv & F \\
F \vee \mathbb{F} & \equiv & F & F \wedge \mathbb{F} & \equiv & \mathbb{F} \\
F \vee \neg F & \equiv & \mathbb{T} & F \wedge \neg F & \equiv & \mathbb{F} \\
\neg(\neg F) & \equiv & F & & & \\
\neg(F \vee G) & \equiv & \neg F \wedge \neg G & \neg(F \wedge G) & \equiv & \neg F \vee \neg G \\
(Qx)F[x] \vee G & \equiv & (Qx)(F[x] \vee G) & (Qx)F[x] \wedge G & \equiv & (Qx)(F[x] \wedge G) \\
\neg \underset{x}{\forall} F[x] & \equiv & \underset{x}{\exists} \neg F[x] & \neg(\underset{x}{\exists} x)F[x] & \equiv & \underset{x}{\forall} \neg F[x] \\
\underset{x}{\forall} F[x] \vee \underset{x}{\forall} G[x] & \not\equiv & \underset{x}{\forall} (F[x] \vee G[x]) & \underset{x}{\forall} F[x] \wedge \underset{x}{\forall} G[x] & \equiv & \underset{x}{\forall} (F[x] \wedge G[x]) \\
\underset{x}{\exists} F[x] \vee \underset{x}{\exists} G[x] & \equiv & \underset{x}{\exists} (F[x] \vee G[x]) & \underset{x}{\exists} F[x] \wedge \underset{x}{\exists} G[x] & \not\equiv & \underset{x}{\exists} (F[x] \wedge G[x])
\end{array}
$$

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

$$
\begin{array}{c||c}
\Longrightarrow & \not\Longrightarrow \\
\not\Longleftarrow & \Longleftarrow
\end{array}
$$

Equivalences of Formulas (cont'd)

Note that

$$\underset{x}{\forall} F[x] \lor \underset{x}{\forall} G[x] \quad \equiv \quad \underset{x}{\forall} F[x] \lor \underset{y}{\forall} G[y] \quad \equiv \quad \underset{x,y}{\forall} F[x] \lor G[y]$$

$$\underset{x}{\exists} F[x] \;\land\; \underset{x}{\exists} G[x] \quad \equiv \quad \underset{x}{\exists} F[x] \land \underset{y}{\exists} G[y] \quad \equiv \quad \underset{x,y}{\exists} F[x] \land G[y]$$

Outline

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 $(Q_1 x_1) \ldots (Q_n x_n)\ M$, 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 $\underset{x_1, \ldots, x_n}{\forall} M$, 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 $(Q_1 x_1)...(Q_n x_n)\ M$, 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 $\underset{x_1,\ldots,x_n}{\forall} M$, 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 $(Q_1 x_1)...(Q_n x_n)\ M$, 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 $\underset{x_1,\ldots,x_n}{\forall} M$, 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 $(Q_1 x_1) \ldots (Q_n x_n)\ M$, 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 $\underset{x_1,\ldots,x_n}{\forall} M$, 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 $(Q_1 x_1)\ldots(Q_n x_n)\ M$, 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 $\underset{x_1,\ldots,x_n}{\forall} M$, 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 $(Q_1 x_1)...(Q_n x_n)\ M$, 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 $\underset{x_1,\ldots,x_n}{\forall} M$, 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 $(Q_1 x_1) \ldots (Q_n x_n)\ M$, 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 $\underset{x_1,\ldots,x_n}{\forall} M$, 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 $(Q_1 x_1) \ldots (Q_n x_n)\ M$, 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 $\underset{x_1,\ldots,x_n}{\forall} M$, 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 $(Q_1 x_1) \ldots (Q_n x_n)\ M$, 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 $\underset{x_1, \ldots, x_n}{\forall} M$, 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
$$\Big(\underset{x}{\forall} P[x]\Big) \Rightarrow Q \quad \equiv \quad \underset{x}{\exists}\,(P[x] \Rightarrow Q)\,.$$

2. Bring the following formulas into Skolem standard form

   - $$\underset{x}{\forall}\ \underset{y,z}{\exists}\,((\neg P[x,y] \ \wedge \ Q[x,z]) \ \vee \ R[x,y,z])$$

   - $$\underset{x,y}{\forall}\Big(\underset{z}{\exists}(P[x,z] \wedge P[y,z]) \ \Rightarrow \ \underset{u}{\exists} Q[x,y,u]\Big)$$

Normal Forms (cont'd)

Examples:

1. Prove the following by bringing the formulas into conjunctive normal form
$$\left(\underset{x}{\forall} P[x]\right) \Rightarrow Q \quad \equiv \quad \underset{x}{\exists}\,(P[x] \Rightarrow Q)\,.$$

2. Bring the following formulas into Skolem standard form
   - $$\underset{x}{\forall}\ \underset{y,z}{\exists}\,((\neg P[x,y] \ \wedge \ Q[x,z]) \ \vee \ R[x,y,z])$$
   - $$\underset{x,y}{\forall}\left(\underset{z}{\exists}\,(P[x,z] \wedge P[y,z]) \ \Rightarrow \ \underset{u}{\exists} Q[x,y,u]\right)$$

Normal Forms (cont'd)

Examples:

1. Prove the following by bringing the formulas into conjunctive normal form
$$\left(\underset{x}{\forall} P[x]\right) \Rightarrow Q \;\equiv\; \underset{x}{\exists}\, (P[x] \Rightarrow Q)\,.$$
2. Bring the following formulas into Skolem standard form
   - $$\underset{x}{\forall}\ \underset{y,z}{\exists} ((\neg P[x,y] \;\wedge\; Q[x,z]) \;\vee\; R[x,y,z])$$
   - $$\underset{x,y}{\forall} \left(\underset{z}{\exists} (P[x,z] \wedge P[y,z]) \;\Rightarrow\; \underset{u}{\exists} Q[x,y,u]\right)$$

Normal Forms (cont'd)

Examples:

1. Prove the following by bringing the formulas into conjunctive normal form
$$\left(\underset{x}{\forall} P[x]\right) \Rightarrow Q \quad \equiv \quad \underset{x}{\exists} (P[x] \Rightarrow Q)\,.$$
2. Bring the following formulas into Skolem standard form
   - $$\underset{x}{\forall}\ \underset{y,z}{\exists} ((\neg P[x,y] \ \wedge \ Q[x,z]) \ \vee \ R[x,y,z])$$
   - $$\underset{x,y}{\forall} \left(\underset{z}{\exists} (P[x,z] \wedge P[y,z]) \ \Rightarrow \ \underset{u}{\exists} Q[x,y,u]\right)$$

Normal Forms (cont'd)

Examples:

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

$$\left(\underset{x}{\forall} P[x]\right) \Rightarrow Q \quad \equiv \quad \underset{x}{\exists}\left(P[x] \Rightarrow Q\right).$$

2. Bring the following formulas into Skolem standard form

   - $$\underset{x}{\forall}\ \underset{y,z}{\exists}\left(\left(\neg P[x,y] \ \wedge \ Q[x,z]\right) \ \vee \ R[x,y,z]\right)$$

   - $$\underset{x,y}{\forall}\left(\underset{z}{\exists}\left(P[x,z] \wedge P[y,z]\right) \ \Rightarrow \ \underset{u}{\exists} Q[x,y,u]\right)$$