Overview

- Last lecture: Started talking about formal semantics for FOL
Overview

▶ Last lecture: Started talking about formal semantics for FOL

▶ Agenda for today:
Overview

▶ Last lecture: Started talking about formal semantics for FOL

▶ Agenda for today:

▶ Finish semantics of FOL
Overview

- **Last lecture:** Started talking about formal semantics for FOL

- **Agenda for today:**
  - Finish semantics of FOL
  - Semantics argument method for proving FOL validity
Overview

- **Last lecture:** Started talking about formal semantics for FOL

- **Agenda for today:**
  - Finish semantics of FOL
  - Semantics argument method for proving FOL validity
  - Important properties of FOL
We evaluate formulas $F$ under structure $S = \langle U, I \rangle$ and variable assignment $\sigma$. If $F$ evaluates to true under $U, I, \sigma$, we write $U, I, \sigma \models F$. If $F$ evaluates to false under $U, I, \sigma$, we write $U, I, \sigma \not\models F$. Semantics of $\models$ is defined inductively. Already defined semantics of terms, predicates, and logical connectives.
We evaluate formulas $F$ under structure $S = \langle U, I \rangle$ and variable assignment $\sigma$.

If $F$ evaluates to true under $U, I, \sigma$, we write $U, I, \sigma \models F$.
Review

- We evaluate formulas $F$ under structure $S = \langle U, I \rangle$ and variable assignment $\sigma$.

- If $F$ evaluates to true under $U, I, \sigma$, we write $U, I, \sigma \models F$.

- If $F$ evaluates to false under $U, I, \sigma$, we write $U, I, \sigma \not\models F$. 
We evaluate formulas $F$ under structure $S = \langle U, I \rangle$ and variable assignment $\sigma$.

- If $F$ evaluates to true under $U, I, \sigma$, we write $U, I, \sigma \models F$.
- If $F$ evaluates to false under $U, I, \sigma$, we write $U, I, \sigma \not\models F$.
- Semantics of $\models$ defined inductively.
We evaluate formulas $F$ under structure $S = \langle U, I \rangle$ and variable assignment $\sigma$.

- If $F$ evaluates to true under $U, I, \sigma$, we write $U, I, \sigma \models F$.

- If $F$ evaluates to false under $U, I, \sigma$, we write $U, I, \sigma \not\models F$.

- Semantics of $\models$ defined inductively.

- Already defined semantics of terms, predicates, and logical connectives.
Example

Consider universe \( \{\ast, \bullet\} \), variable assignment \( \sigma : \{x \mapsto \ast\} \), and interpretation \( I \):

\[
I(a) = \ast \quad I(b) = \bullet \\
I(f) = \{\ast \mapsto \bullet, \bullet \mapsto \ast\} \\
I(p) = \{\langle \bullet, \ast \rangle, \langle \bullet, \bullet \rangle\}
\]
Example

Consider universe \{\star, \bullet\}, variable assignment \(\sigma : \{x \mapsto \star\}\), and interpretation \(I\):

\[
\begin{align*}
I(a) &= \star \\
I(b) &= \bullet \\
I(f) &= \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) &= \{\langle \bullet, \star \rangle, \langle \bullet, \bullet \rangle\}
\end{align*}
\]

Under \(U\), \(I\) and \(\sigma\), what do these formulas evaluate to?

\[
p(f(b), f(x)) \rightarrow p(f(x), f(b)) =
\]
Example

Consider universe \(\{\ast, \bullet\}\), variable assignment \(\sigma : \{x \mapsto \ast\}\), and interpretation \(I\):  

\[
I(a) = \ast \quad I(b) = \bullet \\
I(f) = \{\ast \mapsto \bullet, \bullet \mapsto \ast\} \\
I(p) = \{\langle \bullet, \ast \rangle, \langle \bullet, \bullet \rangle\}
\]

Under \(U, I\) and \(\sigma\), what do these formulas evaluate to?

\[
p(f(b), f(x)) \rightarrow p(f(x), f(b)) = true
\]
Consider universe \( \{\ast, \bullet\} \), variable assignment \( \sigma : \{x \mapsto \ast\} \), and interpretation \( I \):

\[
I(a) = \ast \\
I(b) = \bullet \\
I(f) = \{\ast \mapsto \bullet, \bullet \mapsto \ast\} \\
I(p) = \{\langle \bullet, \ast \rangle, \langle \bullet, \bullet \rangle\}
\]

Under \( U \), \( I \) and \( \sigma \), what do these formulas evaluate to?

\[
p(f(b), f(x)) \rightarrow p(f(x), f(b)) = \text{true} \\
p(f(x), f(b)) \rightarrow p(f(b), f(x)) = \text{false}
\]
Example

Consider universe \{\star, \bullet\}, variable assignment \(\sigma : \{x \mapsto \star\}\), and interpretation \(I\):

\[
\begin{align*}
I(a) &= \star & I(b) &= \bullet \\
I(f) &= \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) &= \{\langle \bullet, \star \rangle, \langle \bullet, \bullet \rangle\}
\end{align*}
\]

Under \(U, I\) and \(\sigma\), what do these formulas evaluate to?

\[
\begin{align*}
p(f(b), f(x)) \rightarrow p(f(x), f(b)) &= \text{true} \\
p(f(x), f(b)) \rightarrow p(f(b), f(x)) &= \text{false}
\end{align*}
\]
Variant of Variable Assignment

- We still need to evaluate formulas containing quantifiers!
We still need to evaluate formulas containing quantifiers!

But to do that, we first define an $x$-variant of a variable assignment.

An $x$-variant of assignment $\sigma$, written $\sigma \left[x \mapsto c\right]$, is the assignment that agrees with $\sigma$ for assignments to all variables except $x$ and assigns $x$ to $c$. 

Example: If $\sigma : \{x \mapsto 1, y \mapsto 2\}$, what is $\sigma \left[x \mapsto 3\right]$?
Variant of Variable Assignment

- We still need to evaluate formulas containing quantifiers!

- But to do that, we first define an $x$-variant of a variable assignment.

- An $x$-variant of assignment $\sigma$, written $\sigma[x \mapsto c]$, is the assignment that agrees with $\sigma$ for assignments to all variables except $x$ and assigns $x$ to $c$. 
Variant of Variable Assignment

- We still need to evaluate formulas containing quantifiers!

- But to do that, we first define an $x$-variant of a variable assignment.

- An $x$-variant of assignment $\sigma$, written $\sigma[x \mapsto c]$, is the assignment that agrees with $\sigma$ for assignments to all variables except $x$ and assigns $x$ to $c$.

- Example: If $\sigma : \{x \mapsto 1, y \mapsto 2\}$, what is $\sigma[x \mapsto 3]$?
Variant of Variable Assignment

- We still need to evaluate formulas containing quantifiers!
- But to do that, we first define an \( x \)-variant of a variable assignment.
- An \( x \)-variant of assignment \( \sigma \), written \( \sigma[x \mapsto c] \), is the assignment that agrees with \( \sigma \) for assignments to all variables except \( x \) and assigns \( x \) to \( c \).
- Example: If \( \sigma : \{ x \mapsto 1, y \mapsto 2 \} \), what is \( \sigma[x \mapsto 3] \)? \( \sigma : \{ x \mapsto 3, y \mapsto 2 \} \)
Evaluation of Formulas II

- We can now give semantics to quantifiers:
Evaluation of Formulas II

- We can now give semantics to quantifiers:

  - Universal quantifier:

    $U, I, \sigma \models \forall x. F$ iff for all $o \in U$, $U, I, \sigma[x \mapsto o] \models F$
Evaluation of Formulas II

- We can now give semantics to quantifiers:

- Universal quantifier:
  \[ U, I, \sigma \models \forall x. F \iff \text{for all } o \in U, U, I, \sigma[x \mapsto o] \models F \]

- Existential quantifier:
  \[ U, I, \sigma \models \exists x. F \iff \text{there exists } o \in U \text{ s.t. } U, I, \sigma[x \mapsto o] \models F \]
Consider universe \{\ast, \bullet\}, variable assignment \(\sigma : \{x \mapsto \ast\}\), and interpretation \(I:\)

\[
I(a) = \ast \quad I(b) = \bullet \\
I(f) = \{\ast \mapsto \bullet, \bullet \mapsto \ast\} \\
I(p) = \{\langle \bullet, \ast \rangle, \langle \bullet, \bullet \rangle\}
\]
Example III: Evaluation of Formulas

Consider universe \{\star, \bullet\}, variable assignment \(\sigma : \{x \mapsto \star\}\), and interpretation \(I:\)

\[
\begin{align*}
I(a) &= \star & I(b) &= \bullet \\
I(f) &= \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) &= \{\langle \bullet, \star \rangle, \langle \bullet, \bullet \rangle\}
\end{align*}
\]

Under \(U, I\) and \(\sigma\), what do these formulas evaluate to?

\[
\forall x. p(x, a)
\]

=
Example III: Evaluation of Formulas

- Consider universe \{\star, \bullet\}, variable assignment \(\sigma : \{x \mapsto \star\}\), and interpretation \(I:\)

\[
\begin{align*}
I(a) &= \star & I(b) &= \bullet \\
I(f) &= \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) &= \{\langle \bullet, \star \rangle, \langle \bullet, \bullet \rangle\}
\end{align*}
\]

- Under \(U, I\) and \(\sigma\), what do these formulas evaluate to?

\[
\forall x. p(x, a) = false
\]
Example III: Evaluation of Formulas

- Consider universe \{\star, \bullet\}, variable assignment \(\sigma: \{x \mapsto \star\}\), and interpretation \(I:\)

\[
I(a) = \star \quad I(b) = \bullet \\
I(f) = \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) = \{\langle\bullet, \star\rangle, \langle\bullet, \bullet\rangle\}
\]

- Under \(U, I\) and \(\sigma\), what do these formulas evaluate to?

\[
\forall x.p(x, a) = false \\
\forall x.p(b, x) =
\]
Example III: Evaluation of Formulas

- Consider universe \( \{\star, \bullet\} \), variable assignment \( \sigma : \{x \mapsto \star\} \), and interpretation \( I \):

\[
I(a) = \star \quad I(b) = \bullet \\
I(f) = \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) = \{\langle \bullet, \star\rangle, \langle \bullet, \bullet\rangle\}
\]

- Under \( U, I \) and \( \sigma \), what do these formulas evaluate to?

\[
\forall x.p(x, a) = \text{false} \\
\forall x.p(b, x) = \text{true}
\]
Example III: Evaluation of Formulas

- Consider universe \( \{\star, \bullet\} \), variable assignment \( \sigma : \{x \mapsto \star\} \), and interpretation \( I \):

\[
\begin{align*}
I(a) &= \star & I(b) &= \bullet \\
I(f) &= \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) &= \{\langle \bullet, \star\rangle, \langle \bullet, \bullet\rangle\}
\end{align*}
\]

- Under \( U, I \) and \( \sigma \), what do these formulas evaluate to?

\[
\begin{align*}
\forall x. p(x, a) &= \text{false} \\
\forall x. p(b, x) &= \text{true} \\
\exists x. p(a, x) &= \text{false}
\end{align*}
\]
Example III: Evaluation of Formulas

- Consider universe \{\star, \bullet\}, variable assignment \(\sigma : \{x \mapsto \star\}\), and interpretation \(I:\)

\[
I(a) = \star \quad I(b) = \bullet \\
I(f) = \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) = \{\langle \bullet, \star \rangle, \langle \bullet, \bullet \rangle\}
\]

- Under \(U, I\) and \(\sigma\), what do these formulas evaluate to?

\[
\forall x.p(x, a) = \text{false} \\
\forall x.p(b, x) = \text{true} \\
\exists x.p(a, x) = \text{false}
\]
Example III: Evaluation of Formulas

Consider universe \{\star, \bullet\}, variable assignment \(\sigma : \{x \mapsto \star\}\), and interpretation \(I:\)

\[
I(a) = \star \quad I(b) = \bullet \\
I(f) = \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) = \{\langle \bullet, \star \rangle, \langle \bullet, \bullet \rangle\}
\]

Under \(U, I\) and \(\sigma\), what do these formulas evaluate to?

\[
\forall x. p(x, a) = false \\
\forall x. p(b, x) = true \\
\exists x. p(a, x) = false \\
\forall x. (p(a, x) \rightarrow p(b, x)) =
\]
Example III: Evaluation of Formulas

Consider universe \{⋆, •\}, variable assignment \(σ : \{x \mapsto ⋆\}\), and interpretation \(I:\)

\[
I(a) = ⋆ \quad I(b) = • \\
I(f) = \{⋆ \mapsto •, • \mapsto ⋆\} \\
I(p) = \{⟨•, ⋆⟩, ⟨•, •⟩\}
\]

Under \(U, I\) and \(σ\), what do these formulas evaluate to?

\[
\forall x. p(x, a) = \text{false} \\
\forall x. p(b, x) = \text{true} \\
\exists x. p(a, x) = \text{false} \\
\forall x. (p(a, x) \rightarrow p(b, x)) = \text{true}
\]
Example III: Evaluation of Formulas

Consider universe \{⋆, ⋅\}, variable assignment \(\sigma: \{x \mapsto ⋆\}\), and interpretation \(I\):

\[
I(a) = ⋆ \quad I(b) = ⋅ \\
I(f) = \{⋆ \mapsto ⋅, ⋅ \mapsto ⋆\} \\
I(p) = \{⟨⋅, ⋆⟩, ⟨⋅, ⋅⟩\}
\]

Under \(U, I\) and \(σ\), what do these formulas evaluate to?

\[
\forall x. p(x, a) = \text{false} \\
\forall x. p(b, x) = \text{true} \\
\exists x. p(a, x) = \text{false} \\
\forall x. (p(a, x) \rightarrow p(b, x)) = \text{true} \\
\exists x. (p(f(x), f(x)) \rightarrow p(x, x)) = \text{true}
\]
Example III: Evaluation of Formulas

- Consider universe \{\star, \bullet\}, variable assignment \(\sigma : \{x \mapsto \star\}\), and interpretation \(I:\)

\[
I(a) = \star \quad I(b) = \bullet \\
I(f) = \{\star \mapsto \bullet, \bullet \mapsto \star\} \\
I(p) = \{\langle \bullet, \star \rangle, \langle \bullet, \bullet \rangle\}
\]

- Under \(U, I\) and \(\sigma\), what do these formulas evaluate to?

\[
\forall x. p(x, a) = \text{false} \\
\forall x. p(b, x) = \text{true} \\
\exists x. p(a, x) = \text{false} \\
\forall x. (p(a, x) \rightarrow p(b, x)) = \text{true} \\
\exists x. (p(f(x), f(x)) \rightarrow p(x, x)) = \text{true}
\]
A first-order formula $F$ is **satisfiable** iff there exists a structure $S$ and variable assignment $\sigma$ such that

$$S, \sigma \models F$$
A first-order formula $F$ is **satisfiable** iff there exists a structure $S$ and variable assignment $\sigma$ such that

$$S, \sigma \models F$$

Otherwise, $F$ is **unsatisfiable**.

A structure $S$ is a model of $F$, written $S \models F$, if for all variable assignments $\sigma$,

$$S, \sigma \models F$$
Satisfiability and Validity of First-Order Formulas

- A first-order formula $F$ is **satisfiable** iff there exists a structure $S$ and variable assignment $\sigma$ such that
  
  $$S, \sigma \models F$$

- Otherwise, $F$ is **unsatisfiable**.

- A structure $S$ is a **model** of $F$, written $S \models F$, if for all variable assignments $\sigma$, $S, \sigma \models F$. 
A first-order formula $F$ is satisfiable iff there exists a structure $S$ and variable assignment $\sigma$ such that

$$S, \sigma \models F$$

Otherwise, $F$ is unsatisfiable.

A structure $S$ is a model of $F$, written $S \models F$, if for all variable assignments $\sigma$, $S, \sigma \models F$.

A first-order formula $F$ is valid, written $\models F$ if for all structures $S$, $S, \sigma \models F$. 
Satisfiability and Validity Examples

▶ Is the formula $\forall x. \exists y. p(x, y)$ satisfiable?

▶ Is this formula valid? no
▶ Falsifying interpretation: $U = \{\ast\}$, $I(p) = \emptyset$

▶ Is the formula $\forall x. (p(x, x) \rightarrow \exists y. p(x, y))$ valid? yes
▶ Intuition: Consider any object $o$. If $p(o, o)$ is false, then implication satisfied. If $p(o, o)$ is true, there there exists a $y$ (namely $o$) s.t $p(x, y)$ is also true.
Satisfiability and Validity Examples

- Is the formula $\forall x. \exists y. p(x, y)$ satisfiable? yes
Satisfiability and Validity Examples

- Is the formula $\forall x. \exists y. p(x, y)$ satisfiable? yes

- Satisfying interpretation:
Satisfiability and Validity Examples

- Is the formula $\forall x. \exists y. p(x, y)$ satisfiable?  yes

- Satisfying interpretation: $U = \{\star\}$, $I(p) = \{\langle \star, \star \rangle\}$
Satisfiability and Validity Examples

- Is the formula $\forall x. \exists y. p(x, y)$ satisfiable? yes

  - Satisfying interpretation: $U = \{\star\}$, $I(p) = \{\langle \star, \star \rangle\}$

- Is this formula valid?
  
  Intuition: Consider any object $o$. If $p(o, o)$ is false, then implication satisfied. If $p(o, o)$ is true, there there exists a $y$ (namely $o$) s.t $p(x, y)$ is also true.
Satisfiability and Validity Examples

- Is the formula $\forall x. \exists y. p(x, y)$ satisfiable? yes

- Satisfying interpretation: $U = \{\star\}$, $I(p) = \{\langle\star, \star\rangle\}$

- Is this formula valid? no
Satisfiability and Validity Examples

- Is the formula $\forall x. \exists y. p(x, y)$ satisfiable? yes

- Satisfying interpretation: $U = \{\star\}$, $I(p) = \{\langle\star, \star\rangle\}$

- Is this formula valid? no

- Falsifying interpretation:
Satisfiability and Validity Examples

- Is the formula $\forall x. \exists y. p(x, y)$ satisfiable? yes

- Satisfying interpretation: $U = \{ \star \}$, $I(p) = \{ (\star, \star) \}$

- Is this formula valid? no

- Falsifying interpretation: $U = \{ \star \}$, $I(p) = \{ \}$
Satisfiability and Validity Examples

▶ Is the formula \( \forall x. \exists y. p(x, y) \) satisfiable? yes

▶ Satisfying interpretation: \( U = \{\star\}, I(p) = \{\langle \star, \star \rangle\} \)

▶ Is this formula valid? no

▶ Falsifying interpretation: \( U = \{\star\}, I(p) = \{\} \)

▶ Is the formula \( \forall x. (p(x, x) \rightarrow \exists y. p(x, y)) \) valid?
Satisfiability and Validity Examples

- Is the formula $\forall x. \exists y. p(x, y)$ satisfiable? yes

- Satisfying interpretation: $U = \{\star\}$, $I(p) = \{\langle \star, \star \rangle\}$

- Is this formula valid? no

- Falsifying interpretation: $U = \{\star\}$, $I(p) = \{\}$

- Is the formula $\forall x. (p(x, x) \rightarrow \exists y. p(x, y))$ valid? yes
Satisfiability and Validity Examples

- Is the formula $\forall x. \exists y. p(x, y)$ satisfiable? yes
  - Satisfying interpretation: $U = \{\star\}$, $I(p) = \{\langle\star, \star\rangle\}$
  - Is this formula valid? no
  - Falsifying interpretation: $U = \{\star\}$, $I(p) = \emptyset$
  - Is the formula $\forall x. (p(x, x) \rightarrow \exists y. p(x, y))$ valid? yes
  - Intuition: Consider any object $o$. If $p(o, o)$ is false, then implication satisfied. If $p(o, o)$ is true, there there exists a $y$ (namely $o$) s.t $p(x, y)$ is also true.
More Satisfiability and Validity Examples

▶ Is the formula $(\exists x. p(x)) \rightarrow p(x)$ contingent, unsat, or valid?
More Satisfiability and Validity Examples

- Is the formula \((\exists x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? contingent
More Satisfiability and Validity Examples

▶ Is the formula $(\exists x. p(x)) \rightarrow p(x)$ contingent, unsat, or valid? contingent

▶ Satisfying $U, I, \sigma$: 
More Satisfiability and Validity Examples

- Is the formula $(\exists x.p(x)) \rightarrow p(x)$ contingent, unsat, or valid? contingent

- Satisfying $U, I, \sigma$: $U = \{\star, \circ\}$, $I(p) = \{\}$, $\sigma(x) = \circ$
More Satisfiability and Validity Examples

- Is the formula \((\exists x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? contingent

- Satisfying \(U, I, \sigma\): \(U = \{\star, \circ\}, I(p) = \{\}, \sigma(x) = \circ\)

- Falsifying interpretation:
More Satisfiability and Validity Examples

▶ Is the formula \((\exists x.p(x)) \rightarrow p(x)\) contingent, unsat, or valid? contingent

▶ Satisfying \(U, I, \sigma:\ U = \{\star, \circ\}, I(p) = \{\}, \sigma(x) = \circ\)

▶ Falsifying interpretation: \(U = \{\star, \circ\}, I(p) = \{\langle \star \rangle\}, \sigma(x) = \circ\)
More Satisfiability and Validity Examples

- Is the formula \((\exists x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? contingent

- Satisfying \(U, I, \sigma:\) \(U = \{\star, \circ\}, I(p) = \{\}, \sigma(x) = \circ\)

- Falsifying interpretation: \(U = \{\star, \circ\}, I(p) = \{\langle \star \rangle\}, \sigma(x) = \circ\)

- Is the formula \((\forall x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid?
More Satisfiability and Validity Examples

- Is the formula \((\exists x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? **contingent**

- Satisfying \(U, I, \sigma\): \(U = \{\star, \circ\}, I(p) = \{\}, \sigma(x) = \circ\)

- Falsifying interpretation: \(U = \{\star, \circ\}, I(p) = \{\langle\star\rangle\}, \sigma(x) = \circ\)

- Is the formula \((\forall x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? **valid**
More Satisfiability and Validity Examples

- Is the formula \((\exists x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? **contingent**

- Satisfying \(U, I, \sigma\): \(U = \{\star, \circ\}, I(p) = \{\}, \sigma(x) = \circ\)

- Falsifying interpretation: \(U = \{\star, \circ\}, I(p) = \{\langle \star \rangle\}, \sigma(x) = \circ\)

- Is the formula \((\forall x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? **valid**

- What about \((\forall x. (p(x) \rightarrow q(x))) \rightarrow (\exists x. (p(x) \land q(x)))\)?

More Satisfiability and Validity Examples

- Is the formula \((\exists x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? contingent

- Satisfying \(U, I, \sigma\): \(U = \{\star, \circ\}, I(p) = \{\}, \sigma(x) = \circ\)

- Falsifying interpretation: \(U = \{\star, \circ\}, I(p) = \{\langle \star \rangle\}, \sigma(x) = \circ\)

- Is the formula \((\forall x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? valid

- What about \((\forall x. (p(x) \rightarrow q(x))) \rightarrow (\exists x. (p(x) \land q(x)))\)? contingent
More Satisfiability and Validity Examples

- Is the formula \((\exists x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? **contingent**

- Satisfying \(U, I, \sigma\): \(U = \{\star, \circ\}, I(p) = \{\}, \sigma(x) = \circ\)

- Falsifying interpretation: \(U = \{\star, \circ\}, I(p) = \{\langle \star \rangle\}, \sigma(x) = \circ\)

- Is the formula \((\forall x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? **valid**

- What about \((\forall x. (p(x) \rightarrow q(x))) \rightarrow (\exists x. (p(x) \land q(x)))\)? **contingent**

- Satisfying interpretation:
More Satisfiability and Validity Examples

▶ Is the formula \((\exists x.p(x)) \rightarrow p(x)\) contingent, unsat, or valid? contingent

▶ Satisfying \(U, I, \sigma\): \(U = \{\star, \circ\}, I(p) = \{\}, \sigma(x) = \circ\)

▶ Falsifying interpretation: \(U = \{\star, \circ\}, I(p) = \{\langle \star \rangle\}, \sigma(x) = \circ\)

▶ Is the formula \((\forall x.p(x)) \rightarrow p(x)\) contingent, unsat, or valid? valid

▶ What about \((\forall x.(p(x) \rightarrow q(x))) \rightarrow (\exists x.(p(x) \land q(x)))\)? contingent

▶ Satisfying interpretation: \(U = \{\star\}, I(p) = \{\langle \star \rangle\}, I(q) = \{\langle \star \rangle\}\)
More Satisfiability and Validity Examples

- Is the formula \((\exists x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? **contingent**

- Satisfying \(U, I, \sigma: U = \{\star, \circ\}, I(p) = \{\}, \sigma(x) = \circ\)**

- Falsifying interpretation: \(U = \{\star, \circ\}, I(p) = \{\langle \star \rangle\}, \sigma(x) = \circ\)**

- Is the formula \((\forall x. p(x)) \rightarrow p(x)\) contingent, unsat, or valid? **valid**

- What about \((\forall x. (p(x) \rightarrow q(x))) \rightarrow (\exists x. (p(x) \land q(x)))\)? **contingent**

- Satisfying interpretation: \(U = \{\star\}, I(p) = \{\langle \star \rangle\}, I(q) = \{\langle \star \rangle\}\)**

- Falsifying interpretation:
More Satisfiability and Validity Examples

- Is the formula $(\exists x. p(x)) \rightarrow p(x)$ contingent, unsat, or valid? **contingent**

- Satisfying $U, I, \sigma$: $U = \{\star, \circ\}, I(p) = \emptyset, \sigma(x) = \circ$

- Falsifying interpretation: $U = \{\star, \circ\}, I(p) = \{\langle\star\rangle\}, \sigma(x) = \circ$

- Is the formula $(\forall x. p(x)) \rightarrow p(x)$ contingent, unsat, or valid? **valid**

- What about $(\forall x. (p(x) \rightarrow q(x))) \rightarrow (\exists x. (p(x) \land q(x)))$? **contingent**

- Satisfying interpretation: $U = \{\star\}, I(p) = \{\langle\star\rangle\}, I(q) = \{\langle\star\rangle\}$

- Falsifying interpretation: $U = \{\star\}, I(p) = \emptyset, I(q) = \{\langle\star\rangle\}$
True/False Exercises

Consider a formula $F$ such that $S, \sigma \models F$. Is $S$ a model $F$?
True/False Exercises

Consider a formula $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$? not necessarily
True/False Exercises

- Consider a formula $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$? **not necessarily**

- Consider a sentence $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$?
True/False Exercises

- Consider a formula $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$?  
  not necessarily

- Consider a sentence $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$?  
  yes
True/False Exercises

- Consider a formula $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$? not necessarily

- Consider a sentence $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$? yes

- Consider a ground formula $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$?
True/False Exercises

- Consider a formula $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$? not necessarily

- Consider a sentence $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$? yes

- Consider a ground formula $F$ such that $S, \sigma \models F$. Is $S$ a model of $F$? yes
Motivation for semantic argument method

- So far, we defined what it means for a first-order formula to be satisfiable and valid.
Motivation for semantic argument method

- So far, we defined what it means for a first-order formula to be satisfiable and valid.

- However, we haven’t talked about how to prove that a formula in FOL is valid.
Motivation for semantic argument method

- So far, we defined what it means for a first-order formula to be satisfiable and valid.

- However, we haven’t talked about how to prove that a formula in FOL is valid.

- Will use semantic argument method to prove validity of first-order formulas
Motivation for semantic argument method

- So far, we defined what it means for a first-order formula to be satisfiable and valid.

- However, we haven’t talked about how to prove that a formula in FOL is valid.

- Will use semantic argument method to prove validity of first-order formulas

- Extension of same technique from propositional logic
Duality of Satisfiability and Validity

- **Recall:** In propositional logic, satisfiability and validity are dual concepts:

  $F$ is valid iff $\neg F$ is unsatisfiable

- This duality also holds in first-order logic.

  Thus, if we have a technique for deciding validity in FOL, this immediately yields a way to decide satisfiability.

- Hence, we'll only focus on proving validity in this lecture.
Duality of Satisfiability and Validity

- **Recall:** In propositional logic, satisfiability and validity are dual concepts:
  \[ F \text{ is valid iff } \neg F \text{ is unsatisfiable} \]

- This duality also holds in first-order logic.
Duality of Satisfiability and Validity

- **Recall:** In propositional logic, satisfiability and validity are dual concepts:
  \[ F \text{ is valid iff } \neg F \text{ is unsatisfiable} \]

- This duality also holds in first-order logic.

- Thus, if we have a technique for deciding validity in FOL, this immediately yields a way to decide satisfiability.
Recall: In propositional logic, satisfiability and validity are dual concepts:

\[ F \text{ is valid iff } \neg F \text{ is unsatisfiable} \]

This duality also holds in first-order logic.

Thus, if we have a technique for deciding validity in FOL, this immediately yields a way to decide satisfiability.

Hence, we’ll only focus on proving validity in this lecture.
Semantic Argument Method to Prove Validity

- We will use the semantic argument technique from earlier to prove validity of first-order formulas.

Recall: Semantic argument method is a proof by contradiction. Basic idea: Assume that $F$ is not valid, i.e., there exists some $S, σ$ such that $S, σ ⊭ F$. Then, apply proof rules. If can derive contradiction on every branch of proof, $F$ is valid.
Semantic Argument Method to Prove Validity

- We will use the semantic argument technique from earlier to prove validity of first-order formulas.

- This technique is not particularly amenable to automation, but is useful for paper-and-pencil proofs of validity.
Semantic Argument Method to Prove Validity

- We will use the semantic argument technique from earlier to prove validity of first-order formulas.

- This technique is not particularly amenable to automation, but is useful for paper-and-pencil proofs of validity.

- **Recall:** Semantic argument method is a proof by contradiction.
We will use the semantic argument technique from earlier to prove validity of first-order formulas.

This technique is not particularly amenable to automation, but is useful for paper-and-pencil proofs of validity.

Recall: Semantic argument method is a proof by contradiction.

Basic idea: Assume that $F$ is not valid, i.e., there exists some $S, \sigma$ such that $S, \sigma \not\models F$. 
Semantic Argument Method to Prove Validity

- We will use the semantic argument technique from earlier to prove validity of first-order formulas.

- This technique is not particularly amenable to automation, but is useful for paper-and-pencil proofs of validity.

- **Recall**: Semantic argument method is a proof by contradiction.

- **Basic idea**: Assume that $F$ is not valid, i.e., there exists some $S, \sigma$ such that $S, \sigma \not\models F$

- Then, apply proof rules.
Semantic Argument Method to Prove Validity

- We will use the semantic argument technique from earlier to prove validity of first-order formulas.

- This technique is not particularly amenable to automation, but is useful for paper-and-pencil proofs of validity.

- **Recall:** Semantic argument method is a proof by contradiction.

- **Basic idea:** Assume that $F$ is not valid, i.e., there exists some $S, \sigma$ such that $S, \sigma \not\models F$

- Then, apply proof rules.

- If can derive contradiction on every branch of proof, $F$ is valid.
Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.
Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.

- As before, proof rules come in pairs, for each connective, we have one case for $\models$, one case for $\not\models$.
Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.

- As before, proof rules come in pairs, for each connective, we have one case for $\models$, one case for $\not\models$

- Negation elimination:

\[
\frac{S, \sigma \models \neg F}{S, \sigma \not\models F}
\]
Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.

- As before, proof rules come in pairs, for each connective, we have one case for $\models$, one case for $\not\models$

- Negation elimination:

$$S, \sigma \models \neg F \quad \frac{S, \sigma \not\models F}{S, \sigma \not\models \neg F}$$
Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.

- As before, proof rules come in pairs, for each connective, we have one case for $\models$, one case for $\not\models$

- Negation elimination:

$$
\frac{S, \sigma \models \neg F}{S, \sigma \not\models F} \quad \frac{S, \sigma \not\models \neg F}{S, \sigma \models F}
$$
Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.

- As before, proof rules come in pairs, for each connective, we have one case for $\models$, one case for $\not\models$

- Negation elimination:

\[
\begin{align*}
S, \sigma & \models \neg F \\
S, \sigma & \not\models F \\
S, \sigma & \not\models \neg F
\end{align*}
\]

- And elimination rule:

\[
S, \sigma \models F \land G
\]
Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.

- As before, proof rules come in pairs, for each connective, we have one case for $\models$, one case for $\not\models$

- Negation elimination:
  
  $\vdash S, \sigma \models \neg F$
  
  $\vdash S, \sigma \models F$

- And elimination rule:

  $\vdash S, \sigma \models F \land G$
  
  $\vdash S, \sigma \models F$
  
  $\vdash S, \sigma \models G$
Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.

- As before, proof rules come in pairs, for each connective, we have one case for $\models$, one case for $\not\models$

- Negation elimination:

  \[
  S, \sigma \models \neg F \quad S, \sigma \not\models \neg F
  \]
  
  \[
  S, \sigma \not\models F \quad S, \sigma \models F
  \]

- And elimination rule:

  \[
  S, \sigma \models F \land G \quad S, \sigma \not\models F \land G
  \]
  
  \[
  S, \sigma \models F \\
  S, \sigma \models G
  \]
Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.

- As before, proof rules come in pairs, for each connective, we have one case for $\models$, one case for $\not\models$

- Negation elimination:

$$
\begin{align*}
S, \sigma & \models \neg F \\
S, \sigma & \not\models F
\end{align*}$$

- And elimination rule:

$$
\begin{align*}
S, \sigma & \models F \land G \\
S, \sigma & \models F \\
S, \sigma & \models G
\end{align*}$$

$$
\begin{align*}
S, \sigma & \not\models F \land G \\
S, \sigma & \not\models F \\
S, \sigma & \not\models G
\end{align*}$$
Proof Rules II (Review)

- Or elimination:

\[
S, \sigma \models F \lor G
\]

- Implication elimination:

\[
S, \sigma \models F \rightarrow G
\]

- If and only if elimination:

\[
S, \sigma \models F \leftrightarrow G
\]
Proof Rules II (Review)

▶ Or elimination:

\[
\begin{align*}
S, \sigma &\models F \lor G \\
S, \sigma &\models F \\
S, \sigma &\models G
\end{align*}
\]
Proof Rules II (Review)

- Or elimination:

\[
\begin{align*}
S, \sigma & \models F \lor G \\
S, \sigma & \models F \\
S, \sigma & \models G \\
S, \sigma & \not\models F \lor G
\end{align*}
\]
Proof Rules II (Review)

- Or elimination:

\[
\begin{align*}
S, \sigma & \models F \lor G \\
S, \sigma & \not\models F \quad \text{or} \\
S, \sigma & \not\models G \\
\end{align*}
\]
Proof Rules II (Review)

- Or elimination:

\[
\begin{align*}
S, \sigma & \models F \lor G \\
S, \sigma & \models F \\
S, \sigma & \models G
\end{align*}
\]

- Implication elimination:

\[
S, \sigma \models F \rightarrow G
\]
Proof Rules II (Review)

- Or elimination:

\[
\begin{align*}
S, \sigma &\models F \lor G \\
S, \sigma &\not\models F \\
S, \sigma &\not\models G
\end{align*}
\]

- Implication elimination:

\[
\begin{align*}
S, \sigma &\models F \to G \\
S, \sigma &\not\models F \\
S, \sigma &\models G
\end{align*}
\]
Proof Rules II (Review)

- Or elimination:

\[
S, \sigma \models F \lor G \\
\quad \quad \quad S, \sigma \not\models F \\
\quad \quad \quad S, \sigma \models G
\]

- Implication elimination:

\[
S, \sigma \models F \rightarrow G \\
\quad \quad \quad S, \sigma \not\models F \\
\quad \quad \quad S, \sigma \models G
\]
Proof Rules II (Review)

- Or elimination:
  \[
  \frac{S, \sigma |= F \lor G}{S, \sigma |= F} \quad \frac{S, \sigma |= F \lor G}{S, \sigma |= G}
  \]

- Implication elimination:
  \[
  \frac{S, \sigma |= F \rightarrow G}{S, \sigma \not|= F} \quad \frac{S, \sigma |= F \rightarrow G}{S, \sigma |= G}
  \]

  \[
  \frac{S, \sigma \not|= F \lor G}{S, \sigma \not|= F} \quad \frac{S, \sigma \not|= F \lor G}{S, \sigma \not|= G}
  \]
Proof Rules II (Review)

- Or elimination:

\[
\frac{S, \sigma \models F \lor G}{S, \sigma \models F, S, \sigma \models G}
\]

- Implication elimination:

\[
\frac{S, \sigma \models F \rightarrow G}{S, \sigma \not\models F, S, \sigma \models G}
\]

- If and only if elimination:

\[
S, \sigma \models F \iff G
\]
Proof Rules II (Review)

- **Or elimination:**

\[
\begin{aligned}
S, \sigma &\models F \lor G \\
\hline
S, \sigma &\models F \\
S, \sigma &\models G
\end{aligned}
\]

\[
\begin{aligned}
S, \sigma &\not\models F \lor G \\
\hline
S, \sigma &\not\models F \\
S, \sigma &\not\models G
\end{aligned}
\]

- **Implication elimination:**

\[
\begin{aligned}
S, \sigma &\models F \rightarrow G \\
\hline
S, \sigma &\not\models F \\
S, \sigma &\models G
\end{aligned}
\]

\[
\begin{aligned}
S, \sigma &\not\models F \rightarrow G \\
\hline
S, \sigma &\models F \\
S, \sigma &\not\models G
\end{aligned}
\]

- **If and only if elimination:**

\[
\begin{aligned}
S, \sigma &\models F \leftrightarrow G \\
\hline
S, \sigma &\models F \land G \\
S, \sigma &\models \neg F \land \neg G
\end{aligned}
\]
Proof Rules II (Review)

- **Or elimination:**

\[
\begin{align*}
S, \sigma \models F \lor G & \quad \frac{S, \sigma \models F}{S, \sigma \models G} \quad S, \sigma \not\models F \lor G \quad \frac{S, \sigma \not\models F}{S, \sigma \not\models G}
\end{align*}
\]

- **Implication elimination:**

\[
\begin{align*}
S, \sigma \models F \implies G & \quad \frac{S, \sigma \not\models F}{S, \sigma \models G} \quad S, \sigma \not\models F \implies G \quad \frac{S, \sigma \models F}{S, \sigma \not\models G}
\end{align*}
\]

- **If and only if elimination:**

\[
\begin{align*}
S, \sigma \models F \iff G & \quad \frac{S, \sigma \models F \land G}{S, \sigma \models \neg F \land \neg G} \quad S, \sigma \not\models F \iff G \quad \frac{S, \sigma \not\models F}{S, \sigma \not\models G}
\end{align*}
\]
Proof Rules II (Review)

- **Or elimination:**

\[
\begin{align*}
S, \sigma & \models F \lor G \\
S, \sigma & \nmod F \quad \text{or} \quad S, \sigma & \nmod G
\end{align*}
\]

- **Implication elimination:**

\[
\begin{align*}
S, \sigma & \models F \rightarrow G \\
S, \sigma & \nmod F \quad \text{or} \quad S, \sigma & \models G
\end{align*}
\]

- **If and only if elimination:**

\[
\begin{align*}
S, \sigma & \models F \leftrightarrow G \\
S, \sigma & \models F \land G \quad \text{or} \quad S, \sigma & \models \neg F \land \neg G
\end{align*}
\]

\[
\begin{align*}
S, \sigma & \nmod F \leftrightarrow G \\
S, \sigma & \models F \land \neg G \quad \text{or} \quad S, \sigma & \models \neg F \land G
\end{align*}
\]
Proof Rules III (New)

- We need new rules to eliminate universal and existential quantifiers.
Proof Rules III (New)

- We need new rules to eliminate universal and existential quantifiers.

- Universal elimination I:

\[
U, I, \sigma \models \forall x. F
\]
Proof Rules III (New)

- We need new rules to eliminate universal and existential quantifiers.

- Universal elimination I:

\[
\frac{U, I, \sigma \models \forall x.F}{U, I, \sigma[x \mapsto o] \models F} \quad \text{(for any } o \in U) \]

Example: Suppose \( U, I, \sigma \models \forall x.\text{hates}(\text{jack}, x) \)

Using the above proof rule, we can conclude:

\( U, I, \sigma[x \mapsto \text{I}(\text{jack})] \models \text{hates}(\text{jack}, x) \)
We need new rules to eliminate universal and existential quantifiers.

Universal elimination I:

\[
U, I, \sigma \models \forall x. F \quad \text{(for any } o \in U)
\]

\[
\frac{U, I, \sigma[x \mapsto o] \models F}{U, I, \sigma \models F}
\]

Example: Suppose \( U, I, \sigma \models \forall x. hates(jack, x) \)
Proof Rules III (New)

- We need new rules to eliminate universal and existential quantifiers.

- **Universal elimination I:**

  \[
  U, I, \sigma \models \forall x.F \quad \text{(for any } o \in U) \\
  U, I, \sigma[x \mapsto o] \models F
  \]

- **Example:** Suppose \( U, I, \sigma \models \forall x.\text{hates}(\text{jack}, x) \)

- Using the above proof rule, we can conclude:

  \[
  U, I, \sigma[x \mapsto I(\text{jack})] \models \text{hates}(\text{jack}, x)
  \]
Universal Elimination Rule II

Universal elimination II:

\[ U, I, \sigma \not\models \forall x. F \]

By a fresh object constant, we mean an object that has not been previously used in the proof.

Why do we have this restriction?

If \( U, I, \sigma \) do not entail \( \forall x. F \), we know there is some object for which \( F \) does not hold, but we don't know which one.

If we have used an object \( o \) before in the proof, we might know something else about \( o \).
Universal Elimination Rule II

- Universal elimination II:

\[
\frac{U, I, \sigma \not\models \forall x. F}{U, I, \sigma[x \mapsto o] \not\models F} \quad \text{(for a fresh } o \in U)\]

- By a fresh object constant, we mean an object that has not been previously used in the proof.

- Why do we have this restriction?

- If \( U, I, \sigma \) do not entail \( \forall x. F \), we know there is some object for which \( F \) does not hold, but we don't know which one.

- If we have used an object \( o \) before in the proof, we might know something else about \( o \).
Universal Elimination Rule II

- Universal elimination II:

\[
\frac{U, I, \sigma \nvdash \forall x.F}{U, I, \sigma[x \mapsto o] \nvdash F} \quad \text{(for a fresh } o \in U)\]

- By a fresh object constant, we mean an object that has not been previously used in the proof
Universal Elimination Rule II

- Universal elimination II:

\[
U, I, \sigma \not\models \forall x. F \quad \text{(for a fresh } o \in U) \\
\]

\[
U, I, \sigma[x \mapsto o] \not\models F
\]

- By a fresh object constant, we mean an object that has not been previously used in the proof.

- Why do we have this restriction?
Universal Elimination Rule II

- **Universal elimination II:**

  \[ U, I, \sigma \not\models \forall x. F \quad \text{(for a fresh } o \in U) \]

  \[ \frac{}{U, I, \sigma[x \mapsto o] \not\models F} \]

- By a fresh object constant, we mean an object that has not been previously used in the proof

- Why do we have this restriction?

- If \( U, I, \sigma \) do not entail \( \forall x. F \), we know there is some object for which \( F \) does not hold, but we don’t know which one
Universal Elimination Rule II

- **Universal elimination II:**

  \[
  \frac{\neg U, I, \sigma \models \forall x. F}{U, I, \sigma[x \mapsto o] \models \neg F}
  \]  
  (for a fresh \(o \in U\))

- By a fresh object constant, we mean an object that has not been previously used in the proof

- Why do we have this restriction?

- If \(U, I, \sigma\) do not entail \(\forall x. F\), we know there is some object for which \(F\) does not hold, but we don’t know which one

- If we have used an object \(o\) before in the proof, we might know something else about \(o\)
Existential Elimination Rule 1

- **Existential elimination I:**
  
  \[ U, I, \sigma \models \exists x. F \]

- Again, fresh means an object that has not been used before.
- If \( U, I, \sigma \) entail \( \exists x. F \), we know there is some object for which \( F \) holds, but we don't know which object.
- If we introduce an object \( o \) we have previously used, we might know something else about \( o \).
Existential Elimination Rule 1

- Existential elimination I:

\[
U, I, \sigma \models \exists x. F \quad \text{(for a fresh } o \in U) \\
\frac{}{U, I, \sigma[x \mapsto o] \models F}
\]
Existential Elimination Rule 1

- **Existential elimination I:**
  \[
  \frac{U, I, \sigma \models \exists x. F}{U, I, \sigma[x \mapsto o] \models F} \quad \text{(for a fresh } o \in U) \]

- Again, **fresh** means an object that has not been used before
Existential Elimination Rule 1

- **Existential elimination I:**
  \[ U, I, \sigma \models \exists x. F \quad (\text{for a fresh } o \in U) \]
  \[ U, I, \sigma[x \leftrightarrow o] \models F \]

- Again, fresh means an object that has not been used before

- If \( U, I, \sigma \) entail \( \exists x. F \), we know there is some object for which \( F \) holds, but we don’t know which object
Existential Elimination Rule 1

- Existential elimination I:
  \[ U, I, \sigma \models \exists x. F \quad \text{(for a fresh } o \in U) \]
  \[ U, I, \sigma [x \mapsto o] \models F \]

- Again, fresh means an object that has not been used before

- If \( U, I, \sigma \) entail \( \exists x. F \), we know there is some object for which \( F \) holds, but we don’t know which object

- If we introduce an object \( o \) we have previously used, we might know something else about \( o \)
Existential Elimination Rule II

- Existential elimination II:

\[ U, I, \sigma \not\models \exists x. F \]
Existential Elimination Rule II

- Existential elimination II:

\[
U, I, \sigma \not\models \exists x. F \quad \text{(for any } o \in U) \\
\overline{U, I, \sigma[x \mapsto o] \not\models F}
\]
Existential Elimination Rule II

- Existential elimination II:

\[
\frac{U, I, \sigma \not\models \exists x . F}{U, I, \sigma[x \mapsto o] \not\models F} \quad \text{(for any } o \in U)\]

- Why can we instantiate \( x \) with any object?

Because if \( U, I, \sigma \) do not entail \( \exists x . F \), this means there does not exist any object for which \( F \) holds. Thus, no matter what object \( x \) maps to, it still won't entail \( F \). Therefore, it's ok to instantiate \( x \) with any object, regardless of whether it has been used before.
Existential Elimination Rule II

▶ Existential elimination II:

\[
\frac{U, I, \sigma \models \exists x. F \quad (\text{for any } o \in U)}{U, I, \sigma[x \mapsto o] \not\models F}
\]

▶ Why can we instantiate \( x \) with any object?

▶ Because if \( U, I, \sigma \) do not entail \( \exists x. F \), this means there does not exist any object for which \( F \) holds
Existential Elimination Rule II

Existential elimination II:

\[
U, I, \sigma \not\models \exists x. F \quad \text{(for any } o \in U) \\
\Rightarrow U, I, \sigma[x \mapsto o] \not\models F
\]

Why can we instantiate \( x \) with any object?

Because if \( U, I, \sigma \) do not entail \( \exists x. F \), this means there does not exist any object for which \( F \) holds

Thus, no matter what object \( x \) maps to, it still won’t entail \( F \)
Existential Elimination Rule II

- Existential elimination II:
  \[ \frac{U, I, \sigma \not\models \exists x. F}{U, I, \sigma[x \mapsto o] \not\models F} \text{ (for any } o \in U) \]

- Why can we instantiate \( x \) with any object?

- Because if \( U, I, \sigma \) do not entail \( \exists x. F \), this means there does not exist any object for which \( F \) holds

- Thus, no matter what object \( x \) maps to, it still won’t entail \( F \)

- Therefore, ok to instantiate \( x \) with any object, regardless of whether it has been used before
Proof Rules V (New)

- Finally, we need a rule for deriving for contradictions.
Proof Rules V (New)

- Finally, we need a rule for deriving for contradictions

- Contradiction rule:

\[
\begin{align*}
U, I, \sigma & \models p(s_1, \ldots, s_n) \\
U, I, \sigma & \not\models p(t_1, \ldots, t_n)
\end{align*}
\]

\[\text{(I, } \sigma)\text{(s}_i\text{) = (I, } \sigma\text{(t}_i\text{) for all } i \in [1, n]}\]
Finally, we need a rule for deriving for contradictions

Contradiction rule:

\[
\begin{align*}
U, I, \sigma & \models p(s_1, \ldots, s_n) \\
U, I, \sigma & \not\models p(t_1, \ldots, t_n) \\
(I, \sigma)(s_i) &= (I, \sigma)(t_i) \text{ for all } i \in [1, n]
\end{align*}
\]
Finally, we need a rule for deriving for contradictions

Contradiction rule:

\[\begin{align*}
U, I, \sigma & \models p(s_1, \ldots, s_n) \\
U, I, \sigma & \not\models p(t_1, \ldots, t_n) \\
(I, \sigma)(s_i) & = (I, \sigma)(t_i) \text{ for all } i \in [1, n] \\
\hline
U, I, \sigma & \models \bot
\end{align*}\]
Finally, we need a rule for deriving for contradictions

Contradiction rule:

\[ U, I, \sigma \models p(s_1, \ldots, s_n) \]
\[ U, I, \sigma \nvdash p(t_1, \ldots, t_n) \]
\[ (I, \sigma)(s_i) = (I, \sigma)(t_i) \text{ for all } i \in [1, n] \]
\[ U, I, \sigma \models \bot \]

Example: Suppose we have \( S, \{ x \mapsto a \} \models p(x) \) and \( S, \{ y \mapsto a \} \nvdash p(y) \)
Proof Rules V (New)

- Finally, we need a rule for deriving for contradictions

- **Contradiction rule:**

\[
\begin{align*}
U, I, \sigma & \models p(s_1, \ldots, s_n) \\
U, I, \sigma & \not\models p(t_1, \ldots, t_n) \\
(I, \sigma)(s_i) & = (I, \sigma)(t_i) \text{ for all } i \in [1, n] \\
U, I, \sigma & \models \bot
\end{align*}
\]

- **Example:** Suppose we have \( S, \{x \mapsto a\} \models p(x) \) and \( S, \{y \mapsto a\} \not\models p(y) \)

- The proof rule for contradiction allows us to derive \( \bot \)
Example 1: Proving Validity

- Prove the validity of formula:

\[ F : (\forall x. p(x)) \rightarrow (\forall y. p(y)) \]
Example 1: Proving Validity

- Prove the validity of formula:

\[ F : (\forall x. p(x)) \rightarrow (\forall y. p(y)) \]

- We start by assuming it is not valid, i.e., there exists some \( S, \sigma \) such that \( S, \sigma \not\models F \).

1. \( S, \sigma \not\models (\forall x. p(x)) \rightarrow (\forall y. p(y)) \) assumption
Example 1: Proving Validity

- Prove the validity of formula:

\[ F : (\forall x.p(x)) \rightarrow (\forall y.p(y)) \]

- We start by assuming it is not valid, i.e., there exists some \( S, \sigma \) such that \( S, \sigma \not\models F \).

1. \( S, \sigma \not\models (\forall x.p(x)) \rightarrow (\forall y.p(y)) \) \hspace{1cm} assumption
2. \( S, \sigma \models \forall x.p(x) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models \forall y.p(y) \) \hspace{1cm} 1 and \( \rightarrow \)
Example 1: Proving Validity

Prove the validity of formula:

\[ F : (\forall x. p(x)) \rightarrow (\forall y. p(y)) \]

We start by assuming it is not valid, i.e., there exists some \( S, \sigma \) such that \( S, \sigma \not\models F \).

1. \( S, \sigma \not\models (\forall x. p(x)) \rightarrow (\forall y. p(y)) \) \hspace{1cm} assumption
2. \( S, \sigma \models \forall x. p(x) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models \forall y. p(y) \) \hspace{1cm} 1 and \( \rightarrow \)
4. \( S, \sigma[y \mapsto o] \not\models p(y) \) \hspace{1cm} 3 and \( \not\models \forall x \)
Example 1: Proving Validity

Prove the validity of formula:

\[ F : (\forall x.p(x)) \rightarrow (\forall y.p(y)) \]

We start by assuming it is not valid, i.e., there exists some \( S, \sigma \) such that \( S, \sigma \not\models F \).

1. \( S, \sigma \not\models (\forall x.p(x)) \rightarrow (\forall y.p(y)) \) assumption
2. \( S, \sigma \models \forall x.p(x) \) 1 and \( \rightarrow \)
3. \( S, \sigma \not\models \forall y.p(y) \) 1 and \( \rightarrow \)
4. \( S, \sigma[y \mapsto o] \not\models p(y) \) 3 and \( \not\models \forall x \)
5. \( S, \sigma[x \mapsto o] \models p(x) \) 2 and \( \models \forall x \)
Example 1: Proving Validity

» Prove the validity of formula:

\[ F : (\forall x. p(x)) \rightarrow (\forall y. p(y)) \]

» We start by assuming it is not valid, i.e., there exists some \( S, \sigma \) such that \( S, \sigma \not\models F \).

1. \( S, \sigma \not\models (\forall x. p(x)) \rightarrow (\forall y. p(y)) \) [assumption]
2. \( S, \sigma \models \forall x. p(x) \) [1 and \( \rightarrow \)]
3. \( S, \sigma \not\models \forall y. p(y) \) [1 and \( \rightarrow \)]
4. \( S, \sigma[y \mapsto o] \not\models p(y) \) [3 and \( \not\models \forall x \)]
5. \( S, \sigma[x \mapsto o] \models p(x) \) [2 and \( \models \forall x \)]
6. \( S, \sigma \models \bot \) [4,5]
Example 2

▶ Is this formula valid?

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x.p(x) \lor \forall x.q(x)) \]
Example 2

- Is this formula valid? Yes!

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]
Example 2

▶ Is this formula valid? Yes!

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

▶ Informal argument: Suppose \( \forall x. (p(x) \lor q(x)) \) holds

▶ Thus, antecedent implies \( \exists x. p(x) \lor \forall x. q(x) \)
Example 2

- Is this formula valid? Yes!

\[
F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x))
\]

- Informal argument: Suppose \( \forall x. (p(x) \lor q(x)) \) holds

- This means either \( q(x) \) for all objects (i.e., \( \forall x. q(x) \))
Example 2

▶ Is this formula valid? Yes!

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

▶ Informal argument: Suppose \( \forall x.(p(x) \lor q(x)) \) holds

▶ This means either \( q(x) \) for all objects (i.e., \( \forall x. q(x) \))

▶ Or if \( q(x) \) does not hold for some object \( o \), then \( p(x) \) must hold for that object \( o \) (i.e, \( \exists x. p(x) \))
Example 2

- Is this formula valid? Yes!

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

- Informal argument: Suppose \( \forall x. (p(x) \lor q(x)) \) holds

- This means either \( q(x) \) for all objects (i.e., \( \forall x. q(x) \))

- Or if \( q(x) \) does not hold for some object \( o \), then \( p(x) \) must hold for that object \( o \) (i.e, \( \exists x. p(x) \))

- Thus, antecedent implies \( \exists p(x) \lor \forall x. q(x) \)
Example 2, cont

Let’s now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]
Example 2, cont

- Let’s now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

- Let’s assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches
Example 2, cont

- Let’s now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

- Let’s assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

1. \( S, \sigma \not\models F \) assumption
Example 2, cont

Let’s now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

Let’s assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

1. \( S, \sigma \not\models F \) \hspace{1cm} assumption
2. \( S, \sigma \models \forall x. (p(x) \lor q(x)) \) \hspace{1cm} 1 and \( \to \)
3. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} 1 and \( \to \)
Example 2, cont

Let's now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

Let's assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

1. \( S, \sigma \not\models F \) \hspace{1cm} assumption
2. \( S, \sigma \models \forall x. (p(x) \lor q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} 1 and \( \rightarrow \)
4. \( S, \sigma \not\models \exists x. p(x) \) \hspace{1cm} 3 and \( \lor \)
5. \( S, \sigma \not\models \forall x. q(x) \) \hspace{1cm} 3 and \( \lor \)
Example 2, cont

Let’s now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

Let’s assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

1. \( S, \sigma \not\models F \) \hspace{1cm} \text{assumption}
2. \( S, \sigma \models \forall x. (p(x) \lor q(x)) \) \hspace{1cm} \text{1 and } \rightarrow
3. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} \text{1 and } \rightarrow
4. \( S, \sigma \not\models \exists x. p(x) \) \hspace{1cm} \text{3 and } \lor
5. \( S, \sigma \not\models \forall x. q(x) \) \hspace{1cm} \text{3 and } \lor
6. \( S, \sigma[x \mapsto o] \not\models q(x) \) \hspace{1cm} \text{5 and } \not\models \forall x, \text{ fresh } o
Example 2, cont

- Let's now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

- Let's assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

1. \( S, \sigma \not\models F \) \hspace{1cm} \text{assumption}
2. \( S, \sigma \models \forall x. (p(x) \lor q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} 1 and \( \rightarrow \)
4. \( S, \sigma \not\models \exists x. p(x) \) \hspace{1cm} 3 and \( \lor \)
5. \( S, \sigma \not\models \forall x. q(x) \) \hspace{1cm} 3 and \( \lor \)
6. \( S, \sigma[x \mapsto o] \not\models q(x) \) \hspace{1cm} 5 and \( \not\models \forall x \), fresh \( o \)
7. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{1cm} 4 and \( \not\models \exists x \), any \( o \)
Let's now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

Let's assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

1. \( S, \sigma \not\models F \) \hspace{1cm} \text{assumption}
2. \( S, \sigma \models \forall x. (p(x) \lor q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} 1 and \( \rightarrow \)
4. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} 1 and \( \rightarrow \)
5. \( S, \sigma \not\models \forall x. q(x) \) \hspace{1cm} 1 and \( \rightarrow \)
6. \( S, \sigma[x \mapsto o] \not\models q(x) \) \hspace{1cm} 5 and \( \not\models \forall x \), fresh \( o \)
7. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{1cm} 4 and \( \not\models \exists x \), any \( o \)
8. \( S, \sigma[x \mapsto o] \models p(x) \lor q(x) \) \hspace{1cm} 2 and \( \models \forall x \), any \( o \)
Example 2, cont

- Let’s now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

- Let’s assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

1. \( S, \sigma \not\models F \) \hspace{1cm} assumption
2. \( S, \sigma \models \forall x. (p(x) \lor q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} 1 and \( \rightarrow \)
4. \( S, \sigma \not\models \exists x. p(x) \) \hspace{1cm} 3 and \( \lor \)
5. \( S, \sigma \not\models \forall x. q(x) \) \hspace{1cm} 3 and \( \lor \)
6. \( S, \sigma[x \mapsto o] \not\models q(x) \) \hspace{1cm} 5 and \( \not\models \forall x \), fresh \( o \)
7. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{1cm} 4 and \( \not\models \exists x \), any \( o \)
8. \( S, \sigma[x \mapsto o] \models p(x) \lor q(x) \) \hspace{1cm} 2 and \( \models \forall x \), any \( o \)
9a. \( S, \sigma[x \mapsto o] \models p(x) \) \hspace{1cm} 8 and \( \lor \)
9b. \( S, \sigma[x \mapsto o] \models q(x) \) \hspace{1cm} 8 and \( \lor \)
Let’s now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \to (\exists x. p(x) \lor \forall x. q(x)) \]

Let’s assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

1. \( S, \sigma \not\models F \) \hspace{1cm} assumption
2. \( S, \sigma \models \forall x. (p(x) \lor q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} 1 and \( \rightarrow \)
4. \( S, \sigma \not\models \exists x. p(x) \) \hspace{1cm} 3 and \( \lor \)
5. \( S, \sigma \not\models \forall x. q(x) \) \hspace{1cm} 3 and \( \lor \)
6. \( S, \sigma[x \mapsto o] \not\models q(x) \) \hspace{1cm} 5 and \( \not\models \forall x, \) fresh \( o \)
7. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{1cm} 4 and \( \not\models \exists x, \) any \( o \)
8. \( S, \sigma[x \mapsto o] \models p(x) \lor q(x) \) \hspace{1cm} 2 and \( \not\models \forall x, \) any \( o \)
9a. \( S, \sigma[x \mapsto o] \models p(x) \) \hspace{1cm} 8 and \( \lor \)
9b. \( S, \sigma[x \mapsto o] \models q(x) \) \hspace{1cm} 8 and \( \lor \)
10a. \( S, \sigma \models \bot \) \hspace{1cm} 7, 9a
Example 2, cont

- Let’s now prove validity using semantic argument method

\[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

- Let’s assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

1. \( S, \sigma \not\models F \) \hspace{1cm} assumption
2. \( S, \sigma \models \forall x. (p(x) \lor q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} 1 and \( \rightarrow \)
4. \( S, \sigma \not\models \exists x. p(x) \lor \forall x. q(x) \) \hspace{1cm} 3 and \( \lor \)
5. \( S, \sigma \not\models \forall x. q(x) \) \hspace{1cm} 3 and \( \lor \)
6. \( S, \sigma[x \mapsto o] \not\models q(x) \) \hspace{1cm} 5 and \( \not\models \forall x \), fresh \( o \)
7. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{1cm} 4 and \( \not\models \exists x \), any \( o \)
8. \( S, \sigma[x \mapsto o] \models p(x) \lor q(x) \) \hspace{1cm} 2 and \( \models \forall x \), any \( o \)
9a. \( S, \sigma[x \mapsto o] \models p(x) \) \hspace{1cm} 8 and \( \lor \)
9b. \( S, \sigma[x \mapsto o] \models q(x) \) \hspace{1cm} 8 and \( \lor \)
10a. \( S, \sigma \models \bot \) \hspace{1cm} 7, 9a
10b. \( S, \sigma \models \bot \) \hspace{1cm} 6, 9b
Example 3

- Is this formula valid?

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]
Example 3

- Is this formula valid? No!

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]
Example 3

- Is this formula valid? No!

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

- Intuitively, antecedent says \( p(o, o) \) holds for every object \( o \)
Example 3

- Is this formula valid? No!

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

- Intuitively, antecedent says \( p(o, o) \) holds for every object \( o \)

- Consequent says there exists some object, say \( o_1 \), for which \( p(o_1, \_ \) holds
Example 3

- Is this formula valid? **No!**

  \[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

- Intuitively, antecedent says \( p(o, o) \) holds for every object \( o \)

- Consequent says there exists some object, say \( o_1 \), for which \( p(o_1, \_ ) \) holds

- Clearly, these mean very different things
Example 3, cont

Now, how do we formally prove this formula is not valid?

\[ F : (\forall x.p(x, x)) \rightarrow (\exists x.\forall y.p(x, y)) \]
Example 3, cont

- Now, how do we formally prove this formula is not valid?

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

- We have to come up with \(U, I, \sigma\) such that \(U, I, \sigma \not\models F\)
Example 3, cont

- Now, how do we formally prove this formula is not valid?

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

- We have to come up with \( U, I, \sigma \) such that \( U, I, \sigma \not\models F \)

- In this case, \( \sigma \) not necessary since no free variables
Example 3, cont

- Now, how do we formally prove this formula is not valid?

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

- We have to come up with \( U, I, \sigma \) such that \( U, I, \sigma \not\models F \)

- In this case, \( \sigma \) not necessary since no free variables

- Choose \( U = \{\star, \bullet\} \), and \( I(p) = \{\langle \star, \star \rangle, \langle \bullet, \bullet \rangle\} \)
Now, how do we formally prove this formula is not valid?

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

We have to come up with \( U, I, \sigma \) such that \( U, I, \sigma \nvDash F \)

In this case, \( \sigma \) not necessary since no free variables

Choose \( U = \{\star, \bullet\} \), and \( I(p) = \{\langle \star, \star \rangle, \langle \bullet, \bullet \rangle\} \)

Clearly, under \( I \), \( \forall x. p(x, x) \) evaluates to true.
Example 3, cont

▶ Now, how do we formally prove this formula is not valid?

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

▶ We have to come up with \( U, I, \sigma \) such that \( U, I, \sigma \not\models F \)

▶ In this case, \( \sigma \) not necessary since no free variables

▶ Choose \( U = \{ \star, \bullet \} \), and \( I(p) = \{ (\star, \star), (\bullet, \bullet) \} \)

▶ Clearly, under \( I \), \( \forall x. p(x, x) \) evaluates to true.

▶ Furthermore, under \( I \), \( (\exists x. \forall y. p(x, y)) \) evaluates to false.
Example 3, cont

- Now, how do we formally prove this formula is not valid?

\[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

- We have to come up with \( U, I, \sigma \) such that \( U, I, \sigma \not\models F \)

- In this case, \( \sigma \) not necessary since no free variables

- Choose \( U = \{\star, \bullet\} \), and \( I(p) = \{\langle \star, \star \rangle, \langle \bullet, \bullet \rangle\} \)

- Clearly, under \( I \), \( \forall x. p(x, x) \) evaluates to true.

- Furthermore, under \( I \), \( \exists x. \forall y. p(x, y) \) evaluates to false.

- Thus, \( I \) is a falsifying interpretation of \( F \).
Example 4

- Is the following formula valid?

\[ (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]
Example 4

Is the following formula valid? **Yes**

\[(\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x))\]
Example 4

- Is the following formula valid? Yes

\[(\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x))\]

- Suppose \((\forall x. p(x) \land q(x))\) holds, we know \(p(x)\) and \(q(x)\) hold for every object \(o\)
Example 4

- Is the following formula valid? Yes

\[(\forall x.(p(x) \land q(x))) \rightarrow (\forall x.p(x)) \land (\forall x.q(x))\]

- Suppose \((\forall x.p(x) \land q(x))\) holds, we know \(p(x)\) and \(q(x)\) hold for every object \(o\)

- Thus, \(p(x)\) must hold for every object (i.e., \(\forall x.p(x)\)) and \(q(x)\) must hold for every object (i.e., \(\forall x.q(x)\))
Example 4

▶ Is the following formula valid? Yes

\[(\forall x.(p(x) \land q(x))) \rightarrow (\forall x.p(x)) \land (\forall x.q(x))\]

▶ Suppose \((\forall x.p(x) \land q(x))\) holds, we know \(p(x)\) and \(q(x)\) hold for every object \(o\)

▶ Thus, \(p(x)\) must hold for every object (i.e., \(\forall x.p(x)\)) and \(q(x)\) must hold for every object (i.e., \(\forall x.q(x)\))

▶ Thus, we also have \(\forall x.p(x) \land \forall x.q(x)\)
Example 4, cont

- Let’s prove validity using semantic argument method:
  \[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]
Example 4, cont

Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)
Example 4, cont

- Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

- Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) assumption
Example 4, cont

Let’s prove validity using semantic argument method:

\[ F : (\forall x.(p(x) \land q(x))) \rightarrow (\forall x.p(x)) \land (\forall x.q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) \hspace{1cm} assumption
2. \( S, \sigma \models \forall x.(p(x) \land q(x)) \hspace{1cm} 1 \text{ and } \rightarrow \)
3. \( S, \sigma \not\models (\forall x.p(x)) \land (\forall x.q(x)) \hspace{1cm} 1 \text{ and } \rightarrow \)
Example 4, cont

- Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

- Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) assumption
2. \( S, \sigma \models \forall x. (p(x) \land q(x)) \) 1 and \( \rightarrow \)
3. \( S, \sigma \not\models (\forall x. p(x)) \land (\forall x. q(x)) \) 1 and \( \rightarrow \)
4a. \( S, \sigma \not\models (\forall x. p(x)) \) 3 and \( \land \)
Example 4, cont

Let’s prove validity using semantic argument method:

\[ F : (\forall x.(p(x) \land q(x))) \rightarrow (\forall x.p(x)) \land (\forall x.q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) \hspace{1cm} \text{assumption}
2. \( S, \sigma \models \forall x.(p(x) \land q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models (\forall x.p(x)) \land (\forall x.q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
4a. \( S, \sigma \not\models (\forall x.p(x)) \) \hspace{1cm} 3 and \( \land \)
4b. \( S, \sigma \not\models (\forall x.q(x)) \) \hspace{1cm} 3 and \( \land \)
Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) assumption
2. \( S, \sigma \models \forall x. (p(x) \land q(x)) \) 1 and \( \rightarrow \)
3. \( S, \sigma \not\models (\forall x. p(x)) \land (\forall x. q(x)) \) 1 and \( \rightarrow \)
4a. \( S, \sigma \not\models (\forall x. p(x)) \) 3 and \( \land \)
4b. \( S, \sigma \not\models (\forall x. q(x)) \) 3 and \( \land \)
5a. \( S, \sigma[x \mapsto o] \not\models p(x) \) 4a and \( \forall \)
Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) \hspace{1cm} \text{assumption}
2. \( S, \sigma \models (\forall x. (p(x) \land q(x))) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models (\forall x. p(x)) \land (\forall x. q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
4a. \( S, \sigma \not\models (\forall x. p(x)) \) \hspace{1cm} 3 and \( \land \)
4b. \( S, \sigma \not\models (\forall x. q(x)) \) \hspace{1cm} 3 and \( \land \)
5a. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{1cm} 4a and \( \forall \)
6a. \( S, \sigma[x \mapsto o] \models p(x) \land q(x) \) \hspace{1cm} 2 and \( \forall \)
Example 4, cont

Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \nvdash F \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>( S, \sigma \nvdash F )</td>
</tr>
<tr>
<td>2.</td>
<td>( S, \sigma \models \forall x. (p(x) \land q(x)) )</td>
</tr>
<tr>
<td>3.</td>
<td>( S, \sigma \nvdash (\forall x. p(x)) \land (\forall x. q(x)) )</td>
</tr>
<tr>
<td>4a.</td>
<td>( S, \sigma \nvdash (\forall x. p(x)) )</td>
</tr>
<tr>
<td>4b.</td>
<td>( S, \sigma \nvdash (\forall x. q(x)) )</td>
</tr>
<tr>
<td>5a.</td>
<td>( S, \sigma[x \mapsto o] \nvdash p(x) )</td>
</tr>
<tr>
<td>6a.</td>
<td>( S, \sigma[x \mapsto o] \models p(x) \land q(x) )</td>
</tr>
<tr>
<td>7a.</td>
<td>( S, \sigma[x \mapsto o] \models p(x) )</td>
</tr>
</tbody>
</table>

assumption

1 and \( \rightarrow \)

1 and \( \rightarrow \)

3 and \( \land \)

3 and \( \land \)

4a and \( \forall \)

2 and \( \forall \)

6a and \( \land \)
Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) \hspace{1cm} assumption
2. \( S, \sigma \models \forall x. (p(x) \land q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models (\forall x. p(x)) \land (\forall x. q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
4a. \( S, \sigma \not\models (\forall x. p(x)) \) \hspace{1cm} 3 and \( \land \)
4b. \( S, \sigma \not\models (\forall x. q(x)) \) \hspace{1cm} 3 and \( \land \)
5a. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{1cm} 4a and \( \forall \)
6a. \( S, \sigma[x \mapsto o] \models p(x) \land q(x) \) \hspace{1cm} 2 and \( \forall \)
7a. \( S, \sigma[x \mapsto o] \models p(x) \) \hspace{1cm} 6a and \( \land \)
8a. \( S, \sigma[x \mapsto o] \models \bot \) \hspace{1cm} 5a, 7a
Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \nvDash F \)

1. \( S, \sigma \nvDash F \)  
   \[ \text{assumption} \]
2. \( S, \sigma \models \forall x. (p(x) \land q(x)) \)  
   \[ 1 \text{ and } \rightarrow \]
3. \( S, \sigma \nvDash (\forall x. p(x)) \land (\forall x. q(x)) \)  
   \[ 1 \text{ and } \rightarrow \]
4a. \( S, \sigma \nvDash (\forall x. p(x)) \)  
   \[ 3 \text{ and } \land \]
4b. \( S, \sigma \nvDash (\forall x. q(x)) \)  
   \[ 3 \text{ and } \land \]
5a. \( S, \sigma[x \mapsto o] \nvDash p(x) \)  
   \[ 4a \text{ and } \forall \]
6a. \( S, \sigma[x \mapsto o] \models p(x) \land q(x) \)  
   \[ 2 \text{ and } \forall \]
7a. \( S, \sigma[x \mapsto o] \models p(x) \)  
   \[ 6a \text{ and } \land \]
8a. \( S, \sigma[x \mapsto o] \models \bot \)  
   \[ 5a, 7a \]
5b. \( S, \sigma[x \mapsto o'] \nvDash q(x) \)  
   \[ 4b \text{ and } \forall \]
Example 4, cont

Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) \hspace{2cm} \text{assumption}
2. \( S, \sigma \models \forall x. (p(x) \land q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models (\forall x. p(x)) \land (\forall x. q(x)) \) \hspace{1cm} 1 and \( \rightarrow \)
4a. \( S, \sigma \not\models (\forall x. p(x)) \) \hspace{1cm} 3 and \( \land \)
4b. \( S, \sigma \not\models (\forall x. q(x)) \) \hspace{1cm} 3 and \( \land \)
5a. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{1cm} 4a and \( \forall \)
6a. \( S, \sigma[x \mapsto o] \models p(x) \land q(x) \) \hspace{1cm} 2 and \( \forall \)
7a. \( S, \sigma[x \mapsto o] \models p(x) \) \hspace{1cm} 6a and \( \land \)
8a. \( S, \sigma[x \mapsto o] \models \bot \) \hspace{1cm} 5a, 7a
5b. \( S, \sigma[x \mapsto o'] \not\models q(x) \) \hspace{1cm} 4b and \( \forall \)
6b. \( S, \sigma[x \mapsto o'] \models p(x) \land q(x) \) \hspace{1cm} 2 and \( \forall \)
Let’s prove validity using semantic argument method:

\[ F : (\forall x. (p(x) \land q(x))) \rightarrow (\forall x. p(x)) \land (\forall x. q(x)) \]

Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) \hspace{2cm} \text{assumption}
2. \( S, \sigma \models \forall x. (p(x) \land q(x)) \) \hspace{2cm} 1 and \( \rightarrow \)
3. \( S, \sigma \not\models (\forall x. p(x)) \land (\forall x. q(x)) \) \hspace{2cm} 1 and \( \rightarrow \)
4a. \( S, \sigma \not\models (\forall x. p(x)) \) \hspace{2cm} 3 and \( \land \)
4b. \( S, \sigma \not\models (\forall x. q(x)) \) \hspace{2cm} 3 and \( \land \)
5a. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{2cm} 4a and \( \forall \)
6a. \( S, \sigma[x \mapsto o] \models p(x) \land q(x) \) \hspace{2cm} 2 and \( \forall \)
7a. \( S, \sigma[x \mapsto o] \models p(x) \) \hspace{2cm} 6a and \( \land \)
8a. \( S, \sigma[x \mapsto o] \models \bot \) \hspace{2cm} 5a, 7a
5b. \( S, \sigma[x \mapsto o'] \not\models q(x) \) \hspace{2cm} 4b and \( \forall \)
6b. \( S, \sigma[x \mapsto o'] \models p(x) \land q(x) \) \hspace{2cm} 2 and \( \forall \)
7b. \( S, \sigma[x \mapsto o'] \models q(x) \) \hspace{2cm} 6b and \( \land \)
Example 4, cont

Let’s prove validity using semantic argument method:

\[
F : (\forall x.(p(x) \land q(x))) \rightarrow (\forall x.p(x)) \land (\forall x.q(x))
\]

Assume there is a \( S, \sigma \) such that \( S, \sigma \not\models F \)

1. \( S, \sigma \not\models F \) \hspace{2cm} \text{assumption}
2. \( S, \sigma \models (\forall x.(p(x) \land q(x))) \) \hspace{2cm} 1 \text{ and } \rightarrow
3. \( S, \sigma \not\models (\forall x.p(x)) \land (\forall x.q(x)) \) \hspace{2cm} 1 \text{ and } \rightarrow
4a. \( S, \sigma \not\models (\forall x.p(x)) \) \hspace{2cm} 3 \text{ and } \land
4b. \( S, \sigma \not\models (\forall x.q(x)) \) \hspace{2cm} 3 \text{ and } \land
5a. \( S, \sigma[x \mapsto o] \not\models p(x) \) \hspace{2cm} 4a \text{ and } \forall
6a. \( S, \sigma[x \mapsto o] \models p(x) \land q(x) \) \hspace{2cm} 2 \text{ and } \forall
7a. \( S, \sigma[x \mapsto o] \models p(x) \) \hspace{2cm} 6a \text{ and } \land
8a. \( S, \sigma[x \mapsto o] \models \bot \) \hspace{2cm} 5a, 7a
5b. \( S, \sigma[x \mapsto o'] \not\models q(x) \) \hspace{2cm} 4b \text{ and } \forall
6b. \( S, \sigma[x \mapsto o'] \models p(x) \land q(x) \) \hspace{2cm} 2 \text{ and } \forall
7b. \( S, \sigma[x \mapsto o'] \models q(x) \) \hspace{2cm} 6b \text{ and } \land
8b. \( S, \sigma[x \mapsto o'] \models \bot \) \hspace{2cm} 5b, 7b
Soundness and Completeness of Proof Rules

- The proof rules we used are sound and complete.
Soundness and Completeness of Proof Rules

- The proof rules we used are sound and complete.

- **Soundness:** If every branch of semantic argument proof derives a contradiction, then $F$ is indeed valid.
The proof rules we used are sound and complete.

**Soundness:** If every branch of semantic argument proof derives a contradiction, then $F$ is indeed valid.

**Translation:** The proof system does not reach wrong conclusions.

Completeness in this context also called refutational completeness.
Soundness and Completeness of Proof Rules

- The proof rules we used are sound and complete.

- **Soundness**: If every branch of semantic argument proof derives a contradiction, then $F$ is indeed valid.

- **Translation**: The proof system does not reach wrong conclusions

- **Completeness**: If formula $F$ is valid, then there exists a finite-length proof in which every branch derives $\bot$
Soundness and Completeness of Proof Rules

- The proof rules we used are sound and complete.

- **Soundness:** If every branch of semantic argument proof derives a contradiction, then $F$ is indeed valid.

- **Translation:** The proof system does not reach wrong conclusions

- **Completeness:** If formula $F$ is valid, then there exists a finite-length proof in which every branch derives ⊥

- **Translation:** There are no valid first-order formulas which we cannot prove to be valid using our proof rules.
Soundness and Completeness of Proof Rules

- The proof rules we used are sound and complete.

- **Soundness:** If every branch of semantic argument proof derives a contradiction, then $F$ is indeed valid.

- **Translation:** The proof system does not reach wrong conclusions

- **Completeness:** If formula $F$ is valid, then there exists a finite-length proof in which every branch derives $\bot$

- **Translation:** There are no valid first-order formulas which we cannot prove to be valid using our proof rules.

- Completeness in this context also called *refutational completeness*
Important Properties of First Order Logic

- **Really important result:** It is undecidable whether a first-order formula is valid. (Church and Turing)
Important Properties of First Order Logic

- **Really important result**: It is undecidable whether a first-order formula is valid. (Church and Turing)

- **Review**: A problem is decidable iff there exists a procedure $P$ such that, for any input:
  1. If the answer is positive, $P$ halts and says "yes".
  2. If the answer is negative, $P$ halts and says "no".

- But, what about the completeness result? Doesn't this contradict undecidability?

- No, because completeness says we will find proof of validity if it exists, but if the formula is invalid, we might search forever.
Important Properties of First Order Logic

- **Really important result:** It is undecidable whether a first-order formula is valid. (Church and Turing)

- **Review:** A problem is decidable iff there exists a procedure $P$ such that, for any input:
  
  1. $P$ halts and says “yes” if the answer is positive
Important Properties of First Order Logic

- **Really important result:** It is undecidable whether a first-order formula is valid. (Church and Turing)

- **Review:** A problem is decidable iff there exists a procedure $P$ such that, for any input:

  1. $P$ halts and says “yes” if the answer is positive

  2. Halts and says “no” if the answer is negative

But, what about the completeness result? Doesn’t this contradict undecidability?

No, because completeness says we will find proof of validity if it exists, but if formula is invalid, we might search forever.
Important Properties of First Order Logic

- **Really important result:** It is undecidable whether a first-order formula is valid. (Church and Turing)

- **Review:** A problem is decidable iff there exists a procedure $P$ such that, for any input:
  1. $P$ halts and says “yes” if the answer is positive
  2. halts and says “no” if the answer is negative

- But, what about the completeness result? Doesn’t this contradict undecidability?
Important Properties of First Order Logic

- **Really important result:** It is undecidable whether a first-order formula is valid. (Church and Turing)

- **Review:** A problem is decidable iff there exists a procedure $P$ such that, for any input:
  1. $P$ halts and says “yes” if the answer is positive
  2. halts and says “no” if the answer is negative

- But, what about the completeness result? Doesn’t this contradict undecidability?

- No, because completeness says we will find proof of validity if it exists, but if formula is invalid, we might search forever.
Semidecidability of First-Order Logic

- First-order logic is semidecidable
Semidecidability of First-Order Logic

- First-order logic is **semidecidable**
- A decision problem is semidecidable iff there exists a procedure $P$ such that, for any input:
Semidecidability of First-Order Logic

- First-order logic is semidecidable

- A decision problem is semidecidable iff there exists a procedure $P$ such that, for any input:

  1. $P$ halts and says “yes” if the answer is positive
Semidecidability of First-Order Logic

- First-order logic is **semidecidable**

- A decision problem is semidecidable iff there exists a procedure $P$ such that, for any input:
  
  1. $P$ halts and says “yes” if the answer is positive
  
  2. $P$ may not terminate if the answer is negative
Semidecidability of First-Order Logic

- First-order logic is **semidecidable**

- A decision problem is semidecidable iff there exists a procedure $P$ such that, for any input:

  1. $P$ halts and says “yes” if the answer is positive

  2. $P$ may not terminate if the answer is negative

- Thus, there exists an algorithm that always terminates and says if any arbitrary FOL formula is valid
Semidecidability of First-Order Logic

- First-order logic is semidecidable

- A decision problem is semidecidable iff there exists a procedure $P$ such that, for any input:
  1. $P$ halts and says “yes” if the answer is positive
  2. $P$ may not terminate if the answer is negative

- Thus, there exists an algorithm that always terminates and says if any arbitrary FOL formula is valid

- But no algorithm is guaranteed to terminate if the FOL formula is not valid
Decidable Fragments of First-Order Logic

- Although full-first order logic is not decidable, there are fragments of FOL that are decidable.
Decidable Fragments of First-Order Logic

- Although full-first order logic is not decidable, there are fragments of FOL that are decidable.

- A fragment of FOL is a syntactically restricted subset of full FOL: e.g., no functions, or only universal quantifiers, etc.
Decidable Fragments of First-Order Logic

- Although full-first order logic is not decidable, there are fragments of FOL that are decidable.

- A fragment of FOL is a syntactically restricted subset of full FOL: e.g., no functions, or only universal quantifiers, etc.

- Some decidable fragments:
  - Quantifier-free first order logic
  - Monadic first-order logic
  - Bernays-Schönfinkel class
The quantifier-free fragment of FOL is the syntactically restricted subset of FOL where formulas do not contain universal or existential quantifiers.
Quantifier-Free Fragment of FOL

- The quantifier-free fragment of FOL is the syntactically restricted subset of FOL where formulas do not contain universal or existential quantifiers.

- Determining validity and satisfiability in quantifier-free FOL is decidable (NP-complete).
The quantifier-free fragment of FOL is the syntactically restricted subset of FOL where formulas do not contain universal or existential quantifiers.

Determining validity and satisfiability in quantifier-free FOL is decidable (NP-complete).

This fragment can be reduced to a theory we will explore later, theory of equality with uninterpreted functions.
Monadic First-Order Logic

- Pure monadic FOL: all predicates are monadic (i.e., arity 1) and no function constants.
Monadic First-Order Logic

- **Pure monadic FOL**: all predicates are monadic (i.e., arity 1) and no function constants.

- **Impure monadic FOL**: both monadic predicates and monadic function constants allowed
Monadic First-Order Logic

- **Pure monadic FOL**: all predicates are *monadic* (i.e., arity 1) and no function constants.

- **Impure monadic FOL**: both monadic predicates and monadic function constants allowed

- **Result**: Monadic first-order logic is decidable (both versions)

However, if we add even a single binary predicate, the logic becomes undecidable.
Monadic First-Order Logic

- **Pure monadic FOL**: all predicates are monadic (i.e., arity 1) and no function constants.

- **Impure monadic FOL**: both monadic predicates and monadic function constants allowed

- **Result**: Monadic first-order logic is decidable (both versions)

- However, if we add even a single binary predicate, the logic becomes undecidable.
The Bernays-Schönfinkel class is a fragment of FOL where:

1. there are no function constants,
2. only formulas of the form:
   \[ \exists x_1, \ldots, \exists x_n, \forall y_1, \ldots, \forall y_m. F(x_1, \ldots, x_n, y_1, \ldots, y_m) \]

Result: The Bernays-Schönfinkel fragment of FOL is decidable.

However, it has additional restriction that all clauses are Horn clauses (i.e., at most one positive literal in each clause).
The Bernays-Schönfinkel class is a fragment of FOL where:

1. there are no function constants,
The Bernays-Schönfinkel class is a fragment of FOL where:

1. there are no function constants,

2. only formulas of the form:

\[ \exists x_1, \ldots, \exists x_n. \forall y_1, \ldots, \forall y_m. F(x_1, \ldots, x_n, y_1, \ldots y_m) \]
The Bernays-Schönfinkel class is a fragment of FOL where:

1. there are no function constants,

2. only formulas of the form:

   $$\exists x_1, \ldots, \exists x_n \forall y_1, \ldots, \forall y_m. F(x_1, \ldots, x_n, y_1, \ldots, y_m)$$

**Result:** The Bernays-Schönfinkel fragment of FOL is decidable
Bernays-Schönfinkel Class

► The Bernays-Schönfinkel class is a fragment of FOL where:

   1. there are no function constants,

   2. only formulas of the form:

\[ \exists x_1, \ldots, \exists x_n. \forall y_1, \ldots, \forall y_m. F(x_1, \ldots, x_n, y_1, \ldots, y_m) \]

► Result: The Bernays-Schönfinkel fragment of FOL is decidable

► Database query language Datalog is based on Bernays-Schönfinkel class of FOL
Bernays-Schönfinkel Class

- The Bernays-Schönfinkel class is a fragment of FOL where:
  1. there are no function constants,
  2. only formulas of the form:

\[
\exists x_1, \ldots, \exists x_n. \forall y_1, \ldots, \forall y_m. F(x_1, \ldots, x_n, y_1, \ldots y_m)
\]

- Result: The Bernays-Schönfinkel fragment of FOL is decidable

- Database query language Datalog is based on Bernays-Schönfinkel class of FOL

- However, it has additional restriction that all clauses are Horn clauses (i.e., at most one positive literal in each clause)
Datalog

- Datalog is a programming language that allows adding/querying facts in a deductive databases
Datalog

- Datalog is a programming language that allows adding/querying facts in a deductive databases

- An example Datalog program:

  ```prolog
  parent(bill, mary). % Bill is Mary’s parent
  parent(mary, john). % Mary is John’s parent

  ancestor(X,Y) :- parent(X,Y).
  ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z).

  ?-ancestor(X, john).
  ```
Datalog

- Datalog is a programming language that allows adding/querying facts in a deductive databases

- An example Datalog program:

  ```
  parent(bill, mary).  % Bill is Mary’s parent
  parent(mary, john).  % Mary is John’s parent

  ancestor(X,Y) :- parent(X,Y).
  ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z).

  %-ancestor(X, john).
  ```

- Last statement is a query: Is there anyone in the database who is John’s ancestor (and if so, who?)
parent(bill, mary). % Bill is Mary’s parent
parent(mary, john). % Mary is John’s parent

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z).

?-ancestor(X, john).

- This program is just syntactic sugar for FOL:
Datalog, cont.

\begin{verbatim}
parent(bill, mary).  % Bill is Mary’s parent
parent(mary, john). % Mary is John’s parent

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z).

?-ancestor(X, john).
\end{verbatim}

- This program is just syntactic sugar for FOL:

\[
\text{parent}(\text{bill}, \text{mary}) \land \text{parent}(\text{mary}, \text{john}) \land
\]

\[
\]
Datalog, cont.

parent(bill, mary). % Bill is Mary’s parent
parent(mary, john). % Mary is John’s parent

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z).

?-ancestor(X, john).

▶ This program is just syntactic sugar for FOL:

\[
\text{parent}(\text{bill}, \text{mary}) \land \text{parent}(\text{mary}, \text{john}) \land
(\forall x, y. \text{parent}(x, y) \rightarrow \text{ancestor}(x, y)) \land
\]

Datalog, cont.

parent(bill, mary). % Bill is Mary’s parent
parent(mary, john). % Mary is John’s parent

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z).

?-ancestor(X, john).

▶ This program is just syntactic sugar for FOL:

\[
\text{parent}(\text{bill}, \text{mary}) \land \text{parent}(\text{mary}, \text{john}) \land \\
(\forall x,y. \text{parent}(x,y) \rightarrow \text{ancestor}(x,y)) \land \\
(\forall x,y,z. \text{parent}(x,y) \land \text{parent}(y,z) \rightarrow \text{ancestor}(x,z)) \land
\]

Thus, if this formula is satisfiable, there is someone in our database who is John’s ancestor.
Datalog, cont.

parent(bill, mary). % Bill is Mary’s parent
parent(mary, john). % Mary is John’s parent

ancestor(X, Y) :- parent(X, Y).
ancestor(X, Z) :- parent(X, Y), ancestor(Y, Z).

?- ancestor(X, john).

▶ This program is just syntactic sugar for FOL:

\[
\begin{align*}
&\text{parent}(\text{bill}, \text{mary}) \land \text{parent}(\text{mary}, \text{john}) \land \\
&(\forall x, y. \text{parent}(x, y) \rightarrow \text{ancestor}(x, y)) \land \\
&(\forall x, y, z. \text{parent}(x, y) \land \text{parent}(y, z) \rightarrow \text{ancestor}(x, z)) \land \\
&(\exists x. \text{ancestor}(x, \text{john}))
\end{align*}
\]
Datalog, cont.

```
parent(bill, mary). % Bill is Mary’s parent
parent(mary, john). % Mary is John’s parent

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z).

?-ancestor(X, john).
```

▶ This program is just syntactic sugar for FOL:

```
parent(bill, mary) ∧ parent(mary, john) ∧
(∀x, y. parent(x, y) → ancestor(x, y)) ∧
(∀x, y, z. parent(x, y) ∧ parent(y, z) → ancestor(x, z)) ∧
(∃x. ancestor(x, john))
```

▶ Thus, if this formula is satisfiable, there is someone in our database who is John’s ancestor
A Datalog interpreter is nothing more than a solver for Bernays-Schönfinkel fragment of FOL
Datalog and Logic Programming Languages

▶ A Datalog interpreter is nothing more than a solver for Bernays-Schönfinkel fragment of FOL

▶ Since this fragment is decidable, Datalog programs always terminate
Datalog and Logic Programming Languages

- A Datalog interpreter is nothing more than a solver for Bernays-Schönfinkel fragment of FOL
- Since this fragment is decidable, Datalog programs always terminate
- In general, interpreters for all logic programming languages decide satisfiability in FOL or a fragment
A Datalog interpreter is nothing more than a solver for Bernays-Schönfinkel fragment of FOL

Since this fragment is decidable, Datalog programs always terminate

In general, interpreters for all logic programming languages decide satisfiability in FOL or a fragment

A popular logic programming language is Prolog
Datalog and Logic Programming Languages

- A Datalog interpreter is nothing more than a solver for Bernays-Schönfinkel fragment of FOL
- Since this fragment is decidable, Datalog programs always terminate
- In general, interpreters for all logic programming languages decide satisfiability in FOL or a fragment
- A popular logic programming language is Prolog
- Unlike Datalog, it is based on full FOL, so Prolog programs may not terminate
Another important property of FOL is compactness.
Another important property of FOL is compactness.

A logic is called compact if an infinite set of sentences $\Gamma$ is satisfiable iff every finite subset of $\Gamma$ is satisfiable.
Compactness of First-Order Logic

- Another important property of FOL is compactness.

- A logic is called **compact** if an infinite set of sentences $\Gamma$ is satisfiable iff every finite subset of $\Gamma$ is satisfiable.

- Theorem (due to Gödel): First-order logic is compact.
Another important property of FOL is compactness.

A logic is called compact if an infinite set of sentences $\Gamma$ is satisfiable iff every finite subset of $\Gamma$ is satisfiable.

Theorem (due to Gödel): First-order logic is compact.

Proof of compactness of FOL follows from the completeness of proof rules.
Proof of Compactness

- **Recall:** Completeness means that if a formula is unsatisfiable, then there exists a finite-length proof of unsatisfiability.
Proof of Compactness

- **Recall**: Completeness means that if a formula is unsatisfiable, then there exists a *finite-length* proof of unsatisfiability.

- Suppose FOL was not compact, i.e., there is an infinite set of sentences $\Gamma$ that are unsat, but every finite subset $\Sigma$ is sat.
Proof of Compactness

- **Recall:** Completeness means that if a formula is unsatisfiable, then there exists a finite-length proof of unsatisfiability.

- Suppose FOL was not compact, i.e., there is an infinite set of sentences $\Gamma$ that are unsat, but every finite subset $\Sigma$ is sat.

- By completeness of proof rules, if $\Gamma$ is unsat, there exists a finite-length proof of unsatisfiability.
Proof of Compactness

- **Recall**: Completeness means that if a formula is unsatisfiable, then there exists a *finite-length* proof of unsatisfiability.

- Suppose FOL was not compact, i.e., there is an infinite set of sentences $\Gamma$ that are unsat, but every finite subset $\Sigma$ is sat.

- By completeness of proof rules, if $\Gamma$ is unsat, there exists a finite-length proof of unsatisfiability.

- But this means the proof must use a finite subset of sentences $\Sigma$ of $\Gamma$, otherwise proof could not be finite.
Proof of Compactness

- **Recall:** Completeness means that if a formula is unsatisfiable, then there exists a finite-length proof of unsatisfiability.

- Suppose FOL was not compact, i.e., there is an infinite set of sentences $\Gamma$ that are unsat, but every finite subset $\Sigma$ is sat.

- By completeness of proof rules, if $\Gamma$ is unsat, there exists a finite-length proof of unsatisfiability.

- But this means the proof must use a finite subset of sentences $\Sigma$ of $\Gamma$, otherwise proof could not be finite.

- But this implies there is also a proof of unsatisfiability of $\Sigma$. 

Proof of Compactness

- **Recall**: Completeness means that if a formula is unsatisfiable, then there exists a finite-length proof of unsatisfiability.

- Suppose FOL was not compact, i.e., there is an infinite set of sentences $\Gamma$ that are unsat, but every finite subset $\Sigma$ is sat.

- By completeness of proof rules, if $\Gamma$ is unsat, there exists a finite-length proof of unsatisfiability.

- But this means the proof must use a finite subset of sentences $\Sigma$ of $\Gamma$, otherwise proof could not be finite.

- But this implies there is also a proof of unsatisfiability of $\Sigma$.

- Thus, by soundness of proof rules, $\Sigma$ must be unsat. ∎
Proving of compactness might look like a useless property, but it has very interesting consequences!
Consequences of Compactness

- Proof of compactness might look like a useless property, but it has very interesting consequences!

- Compactness can be used to show that a variety of interesting properties are not expressible in first-order logic.
Consequences of Compactness

- Proof of compactness might look like a useless property, but it has very interesting consequences!

- Compactness can be used to show that a variety of interesting properties are not expressible in first-order logic.

- For instance, we can use compactness theorem to show that transitive closure is not expressible in first order logic.
Transitive Closure

Given a directed graph $G = (V, E)$, the transitive closure of $G$ is defined as the graph $G^* = (V, E^*)$ where:

$$E^* = \{(n, n') \mid \text{if there is a path from vertex } n \text{ to } n'\}$$
Transitive Closure

Given a directed graph $G = (V, E)$, the transitive closure of $G$ is defined as the graph $G^* = (V, E^*)$ where:

$$E^* = \{(n, n') \mid \text{if there is a path from vertex } n \text{ to } n'\}$$

Observe: A binary predicate $p(t, t')$ be viewed as a graph containing an edge from node $t$ to $t'$
Given a directed graph $G = (V, E)$, the **transitive closure** of $G$ is defined as the graph $G^* = (V, E^*)$ where:

$$E^* = \{(n, n') \mid \text{if there is a path from vertex } n \text{ to } n'\}$$

**Observe:** A binary predicate $p(t, t')$ be viewed as a graph containing an edge from node $t$ to $t'$

Thus, the concept of transitive closure applies to binary predicates as well.
Given a directed graph $G = (V, E)$, the transitive closure of $G$ is defined as the graph $G^* = (V, E^*)$ where:

$$E^* = \{(n, n') \mid \text{if there is a path from vertex } n \text{ to } n'\}$$

Observe: A binary predicate $p(t, t')$ be viewed as a graph containing an edge from node $t$ to $t'$

Thus, the concept of transitive closure applies to binary predicates as well

A binary predicate $T$ is the transitive closure of predicate $p$ if $⟨t_0, t_n⟩ \in T$ iff there exists some sequence $t_0, t_1 \ldots, t_n$ such that $⟨t_i, t_{i+1}⟩ \in p$
“Expressing” Transitive Closure in FOL

- At first glance, it looks like transitive closure $T$ of binary relation $p$ is expressible in FOL:

\[ \forall x, \forall z. (T(x, z) \iff (p(x, z) \lor \exists y. p(x, y) \land T(y, z))) \]
At first glance, it looks like transitive closure $T$ of binary relation $p$ is expressible in FOL:

$$\forall x, \forall z. (T(x, z) \leftrightarrow (p(x, z) \lor \exists y. p(x, y) \land T(y, z)))$$

But this formula does not describe transitive closure at all!
“Expressing” Transitive Closure in FOL

- At first glance, it looks like transitive closure $T$ of binary relation $p$ is expressible in FOL:

$$\forall x, \forall z. (T(x, z) \leftrightarrow (p(x, z) \lor \exists y. p(x, y) \land T(y, z)))$$

- But this formula does not describe transitive closure at all!

- To see why, consider $U = \mathbb{N}$, $p$ is equality predicate, and $T$ is relation that is true for any number $x, y$. 
“Expressing” Transitive Closure in FOL

- At first glance, it looks like transitive closure $T$ of binary relation $p$ is expressible in FOL:

$$\forall x, \forall z. (T(x, z) \iff (p(x, z) \lor \exists y. p(x, y) \land T(y, z)))$$

- But this formula does not describe transitive closure at all!

- To see why, consider $U = \mathbb{N}$, $p$ is equality predicate, and $T$ is relation that is true for any number $x, y$.

- Clearly, this $T$ is not the transitive closure of equality, but this structure is actually a model of the formula.
“Expressing” Transitive Closure in FOL

- At first glance, it looks like transitive closure \( T \) of binary relation \( p \) is expressible in FOL:

\[
\forall x, \forall z. (T(x, z) \leftrightarrow (p(x, z) \lor \exists y. p(x, y) \land T(y, z)))
\]

- But this formula does not describe transitive closure at all!

- To see why, consider \( U = \mathbb{N} \), \( p \) is equality predicate, and \( T \) is relation that is true for any number \( x, y \).

- Clearly, this \( T \) is not the transitive closure of equality, but this structure is actually a model of the formula.

- Thus, the formula above is not a definition of transitive closure at all!
In fact, no matter how hard we try to correct this definition, we cannot express transitive closure in FOL.
In fact, no matter how hard we try to correct this definition, we cannot express transitive closure in FOL.

Will use compactness theorem to show that transitive closure is not expressible in FOL.
In fact, no matter how hard we try to correct this definition, we cannot express transitive closure in FOL.

Will use compactness theorem to show that transitive closure is not expressible in FOL.

Compactness: An infinite set of sentences $\Gamma$ is satisfiable iff every finite subset of $\Gamma$ is satisfiable.
In fact, no matter how hard we try to correct this definition, we cannot express transitive closure in FOL.

Will use compactness theorem to show that transitive closure is not expressible in FOL.

**Compactness**: An infinite set of sentences $\Gamma$ is satisfiable iff every finite subset of $\Gamma$ is satisfiable.

For contradiction, suppose transitive closure is expressible in first order logic.
In fact, no matter how hard we try to correct this definition, we cannot express transitive closure in FOL.

Will use compactness theorem to show that transitive closure is not expressible in FOL.

Compactness: An infinite set of sentences $\Gamma$ is satisfiable iff every finite subset of $\Gamma$ is satisfiable.

For contradiction, suppose transitive closure is expressible in first order logic.

Let $\Gamma$ be a (possibly infinite) set of sentences expressing that $T$ is the transitive closure of $p$. 
Proof 1

- $\Psi^n(a, b)$ encode the proposition: there is no path of length $n$ from $a$ to $b$. 
Proof 1

- $\Psi^n(a, b)$ encode the proposition: there is no path of length $n$ from $a$ to $b$.

- In particular, $\Psi^1 = \neg p(a, b)$
Proof 1

- $\Psi^n(a, b)$ encode the proposition: there is no path of length $n$ from $a$ to $b$.

- In particular, $\Psi^1 = \neg p(a, b)$

- Similarly,

$$\Psi^n = \neg \exists x_1, \ldots, x_{n-1}. (p(a, x_1) \land p(x_1, x_2) \land \ldots \land p(x_{n-1}, b))$$
Recall: \( \Gamma \) is a set of propositions encoding \( T \) is transitive closure of \( p \).
Proof II

- **Recall:** $\Gamma$ is a set of propositions encoding $T$ is transitive closure of $p$.

- Now, construct $\Gamma'$ as follows:

$$\Gamma' = \Gamma \cup \{ T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots \}$$
Proof II

- **Recall:** $\Gamma$ is a set of propositions encoding $T$ is transitive closure of $p$.

- **Now, construct $\Gamma'$ as follows:**

  $$\Gamma' = \Gamma \cup \{ T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots, \}$$

- **Observe:** $\Gamma'$ is unsatisfiable because:

  1. Since $\Gamma$ encodes that $T$ is transitive closure of $p$, $T(a, b)$ says there is some path from $a$ to $b$.
  2. The infinite set of propositions $\Psi^1, \Psi^2, \ldots$ say that there is no path of any length from $a$ to $b$. 
Proof II

- **Recall:** $\Gamma$ is a set of propositions encoding $T$ is transitive closure of $p$.

- **Now, construct $\Gamma'$ as follows:**

  $$\Gamma' = \Gamma \cup \{ T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots \}$$

- **Observe:** $\Gamma'$ is unsatisfiable because:

  1. Since $\Gamma$ encodes that $T$ is transitive closure of $p$, $T(a, b)$ says there is some path from $a$ to $b$. 
Proof II

- **Recall:** \( \Gamma \) is a set of propositions encoding \( T \) is transitive closure of \( p \).

- **Now,** construct \( \Gamma' \) as follows:

\[
\Gamma' = \Gamma \cup \{ T(a, b), \Psi_1, \Psi_2, \Psi_3, \ldots \}
\]

- **Observe:** \( \Gamma' \) is unsatisfiable because:

1. Since \( \Gamma \) encodes that \( T \) is transitive closure of \( p \), \( T(a, b) \) says there is some path from \( a \) to \( b \)

2. The infinite set of propositions \( \Psi_1, \Psi_2, \ldots \) say that there is no path of any length from \( a \) to \( b \)
Proof III

- Now, consider any finite subset of $\Gamma'$:

$$
\Gamma' = \Gamma \cup \{T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots, \}
$$
Proof III

- Now, consider any finite subset of $\Gamma'$:

$$\Gamma' = \Gamma \cup \{T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots\}$$

- Clearly, any finite subset does not contain $\Psi_i$ for some $i$. 

Thus, transitive closure cannot be expressed in FOL!
Proof III

- Now, consider any finite subset of $\Gamma'$:
  \[
  \Gamma' = \Gamma \cup \{ T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots, \}
  \]

- Clearly, any finite subset does not contain $\Psi_i$ for some $i$.

- **Observe:** This finite subset is satisfied by a model where there is a path of length $i$ from $a$ to $b$.
Proof III

- Now, consider any finite subset of $\Gamma'$:

$$\Gamma' = \Gamma \cup \{ T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots, \}$$

- Clearly, any finite subset does not contain $\Psi_i$ for some $i$.

- Observe: This finite subset is satisfied by a model where there is a path of length $i$ from $a$ to $b$.

- Thus, every finite subset of $\Gamma'$ is satisfiable.
Proof III

- Now, consider any finite subset of $\Gamma'$:
  \[
  \Gamma' = \Gamma \cup \{T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots, \}
  \]

- Clearly, any finite subset does not contain $\Psi_i$ for some $i$.

- **Observe**: This finite subset is satisfied by a model where there is a path of length $i$ from $a$ to $b$

- Thus, every finite subset of $\Gamma'$ is satisfiable.

- By the compactness theorem, this would imply $\Gamma'$ is also satisfiable
Proof III

- Now, consider any finite subset of $\Gamma'$:

\[ \Gamma' = \Gamma \cup \{ T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots, \} \]

- Clearly, any finite subset does not contain $\Psi_i$ for some $i$.

- **Observe:** This finite subset is satisfied by a model where there is a path of length $i$ from $a$ to $b$.

- Thus, every finite subset of $\Gamma'$ is satisfiable.

- By the compactness theorem, this would imply $\Gamma'$ is also satisfiable.

- But we just showed that $\Gamma'$ is unsatisfiable!
Now, consider any finite subset of $\Gamma'$:

$$\Gamma' = \Gamma \cup \{T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots, \}$$

Clearly, any finite subset does not contain $\Psi_i$ for some $i$.

Observe: This finite subset is satisfied by a model where there is a path of length $i$ from $a$ to $b$.

Thus, every finite subset of $\Gamma'$ is satisfiable.

By the compactness theorem, this would imply $\Gamma'$ is also satisfiable.

But we just showed that $\Gamma'$ is unsatisfiable!

Thus, transitive closure cannot be expressed in FOL!
Summary

- Semantic argument method for proving validity in FOL
Summary

- Semantic argument method for proving validity in FOL
- Soundness and completeness of semantic argument method
Summary

- Semantic argument method for proving validity in FOL
- Soundness and completeness of semantic argument method
- Important properties of FOL: undecidability, semidecidability, compactness

Next lecture: Basics of automated first-order theorem provers (much less theoretical)
Summary

- Semantic argument method for proving validity in FOL
- Soundness and completeness of semantic argument method
- Important properties of FOL: undecidability, semidecidability, compactness
- Compactness: useful to show what is not expressible in FOL
Summary

- Semantic argument method for proving validity in FOL
- Soundness and completeness of semantic argument method
- Important properties of FOL: undecidability, semidecidability, compactness
- Compactness: useful to show what is not expressible in FOL

Next lecture: Basics of automated first-order theorem provers (much less theoretical)