CSE4051: Program Verification

First-Order Logic

2025 Fall

Woosuk Lee

Review: Calculus of Computation

- Calculus: a set of symbols + rules for manipulating the symbols
  - e.g., Differential calculus: rules for manipulating integral symbols over a polynomial
- We may ask questions about computations
  - Does this program terminate?
  - Does this program output a sorted array for a given array?
  - Does this program access unallocated memory?
- We need a calculus to reason about computation to answer these questions.

Review: Propositional Logic and First-Order Logic

- Also known as propositional calculus and predicate calculus
- calculi for reasoning about propositions and predicates
- Propositions: statements that can be true or false
  - e.g., "It is raining", "2 + 2 = 4"
- **Predicates**: statements that can be true or false depending on the values given to them
  - e.g., "x is greater than 2", "y is a prime number"

First-Order Logic

- First-order logic (FOL) (also called predicate logic) extends propositional logic (PL) with, most importantly, **predicates.**
- A predicate takes arguments and returns a truth value.
  - *n*-ary predicate takes *n* arguments
- A propositional variable can be regarded as a 0-ary predicate.
- Examples of predicates (x, y are propositional variables, p, q are predicates)
  - $p(x, y), q(y)$
  - $\neg p(x, y) \wedge q(y)$
  - $Love(Alice, Bob)$:"Alice loves Bob"

First-Order Logic

- First-order logic (FOL) extends propositional logic (PL) also with **functions** and **quantifiers**.
- In PL, all parts of a formula evaluate to true or false.

$$(p \land q) \lor \neg r$$

- In FOL, thanks to **functions**, parts of a formula may evaluate to values other than truth values such as integers, strings, etc.

$$x + y > 5 \land f(z) = \text{"hi"}$$

Function (+) · Predicate (>) · Function (f) · Constant ("hi")

Terms

- Representations of objects that we are reasoning about
- Constants and variables are terms
- Functions taking terms as arguments are also terms.
- Examples:
  - $f(1)$: a unary function f applied to a constant 1
  - $g(x,2)$: a binary function g applied to a variable x and a constant 2
  - $f(g(x, f(1)))$

First-Order Logic

- **Quantifiers**: symbols telling you how many things a statement is talking about
  - Universal quantifier(∀ — "for all"): a statement is true for every object

    $$\forall x\,.\,Human(x) \implies Mortal(x)$$

    "For every x, if x is a human, then x is mortal."
  - Existential quantifier(∃ — "there exists"): a statement is true for at least one object

    $$\exists x\,.\,Student(x) \wedge StudiesCSE4051(x)$$

    "There exists x such that x is a student and x studies the CSE4051 course."

Syntax

- Atom: Basic elements
  - Truth symbols $\bot$, $\top$, n-ary predicates applied to n terms
- Literal: an atom or its negation
- Formula: a literal or application of a logical connective to formulas, or the application of a quantifier to a formula

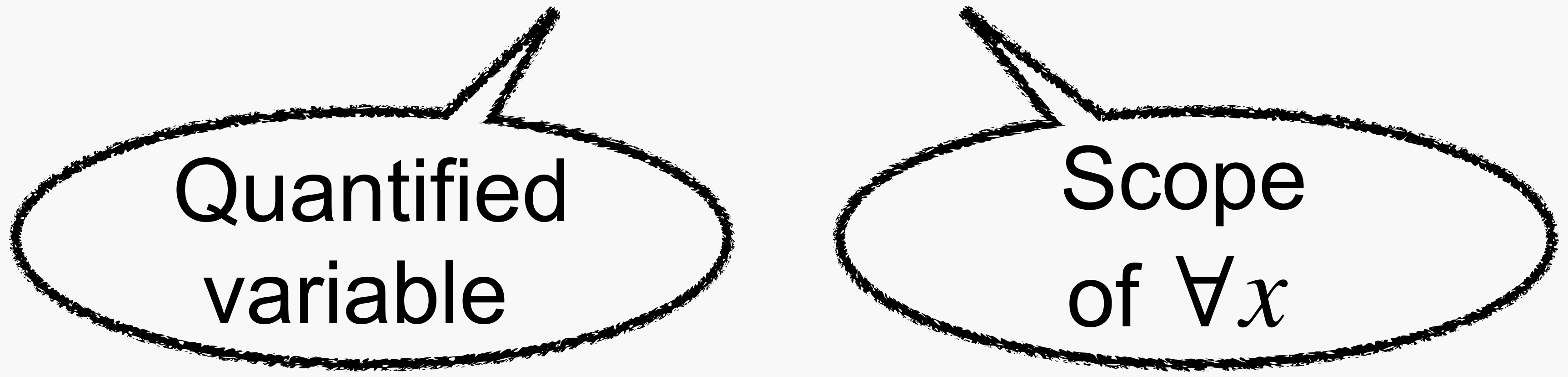
$F \rightarrow \bot \mid \top \mid p(t_1, \ldots, t_n)$ (Atom)

$\mid \neg F \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid F_1 \Rightarrow F_2$ (Logical connectives)

$\mid \forall x \,.\, F[x] \mid \exists x \,.\, F[x]$ (Quantification)

More about Quantifiers

$$\forall x\,.\;F[x]$$



- x in $F[x]$ is bound (by the quantifier)
- The scope of the quantified variable extends as far as possible
  - Which are the scopes of $\forall x$ and $\exists y$?

$$\forall x\,.\,p(f(x), x) \Rightarrow (\,\exists y\,.\,p(f(g(x, y)), g(x, y))) \land q(x, f(x))$$

- A variable occurrence is free in $F[x]$ if it is not bound.
- Which variable occurrences are free / bound?

$$\forall x\,.\,p(f(x), y) \Rightarrow \forall y\,.\,p(f(x), y)$$

Examples of FOL Formulas

- Every dog has its day.

$$\forall x.\ dog(x) \ \rightarrow\ \exists y.\ day(y) \ \wedge\ itsDay(x, y)$$

- Some dogs have more days than others.

$$\exists x, y.\ dog(x) \ \wedge\ dog(y) \ \wedge\ \#days(x) > \#days(y)$$

- All cats have more days than dogs.

$$\forall x, y.\ dog(x) \ \wedge\ cat(y) \ \rightarrow\ \#days(y) > \#days(x)$$

Examples of FOL Formulas

- Fido is a dog. Furrball is a cat. Fido has fewer days than does Furrball.

$$dog(Fido) \ \wedge \ cat(Furrball) \ \wedge \ \#days(Fido) < \#days(Furrball)$$

- The length of one side of a triangle is less than the sum of the lengths of the other two sides.

$$\forall x, y, z.\ triangle(x, y, z) \ \rightarrow \ length(x) < length(y) + length(z)$$

- Fermat's Last Theorem.

$$\begin{aligned} &\forall n.\ integer(n) \ \wedge \ n > 2 \\ &\quad \rightarrow \ \forall x, y, z. \\ &\qquad integer(x) \ \wedge \ integer(y) \ \wedge \ integer(z) \ \wedge \ x > 0 \ \wedge \ y > 0 \ \wedge \ z > 0 \\ &\qquad\quad \rightarrow \ x^n + y^n \neq z^n \end{aligned}$$

Semantics (Meaning) of FOL

- Formulae of FOL evaluate to the truth values true and false.
- However, terms of FOL formulae evaluate to values from a specified **domain**.
- A FOL interpretation $I = (D_I, \alpha_I)$
  - The **domain** D_I : a nonempty (possibly infinite) set of values (e.g., integers, people, …)
  - The **assignment** α_I maps
    - a variable symbol x to a value in D_I
    - an n-ary function symbol f to a function $f_I : D_I^n \to D_I$
    - an n-ary predicate symbol p to an n-ary predicate $p_I : D_I^n \to \{\text{true}, \text{false}\}$
    - Each constant (0-ary function) and propositional variable (0-ary predicate) are assigned a value in D_I and a truth value, respectively.

Example

- Given a formula $F : x + y > z \Rightarrow y > z - x$
- Note that $+, -, >$ are just symbols: we could have written

$$p(f(x, y), z) \Rightarrow p(y, g(z, x))$$

- An interpretation $I = (D_I, \alpha_I)$ where
  - $D_I = \mathbb{Z}$ (set of integers)
  - $\alpha_I = \{ + \mapsto +_{\mathbb{Z}}, - \mapsto -_{\mathbb{Z}}, > \mapsto >_{\mathbb{Z}}, x \mapsto 13, y \mapsto 42, z \mapsto 1\}$

Semantics of FOL

- Given a FOL formula F and interpretation $I = (D_I, \alpha_I)$, $I \models F$ or $I \not\models F$.
- The meaning of truth symbols:
  - $I \models \top$, $I \not\models \bot$
- For more complicated atoms, α_I gives meaning $\alpha_I(x)$, $\alpha_I(c)$, and $\alpha_I(f)$ to variables x, constants c, and functions f. Evaluate arbitrary terms recursively:

  $$\alpha_I(f(t_1, \ldots, t_n)) = \alpha_I(f)(\alpha_I(t_1), \ldots, \alpha_I(t_n))$$

  for function symbol f and terms $t_1, \ldots, t_n$. Similarly, for predicate symbol p

  $$\alpha_I(p(t_1, \ldots, t_n)) = \alpha_I(p)(\alpha_I(t_1), \ldots, \alpha_I(t_n))$$

  Then, $I \models p(t_1, \ldots, t_n)$ iff $\alpha_I[p(t_1, \ldots, t_n)] =$ true

Semantics of FOL

- Connectives:

$$\begin{array}{lcll} I & \models & \neg F & \text{iff } I \not\models F \\ I & \models & F_1 \wedge F_2 & \text{iff } I \models F_1 \text{ and } I \models F_2 \\ I & \models & F_1 \vee F_2 & \text{iff } I \models F_1 \text{ or } I \models F_2 \\ I & \models & F_1 \rightarrow F_2 & \text{iff, if } I \models F_1 \text{ then } I \models F_2 \end{array}$$

- Quantifiers:

$$\begin{array}{lcll} I & \models & \forall x.\ F & \text{iff for all } \mathsf{v} \in D_I,\ I \triangleleft \{x \mapsto \mathsf{v}\} \models F \\ I & \models & \exists x.\ F & \text{iff there exists } \mathsf{v} \in D_I \text{ such that } I \triangleleft \{x \mapsto \mathsf{v}\} \models F \end{array}$$

I
with an updated assignment
where x maps to v

Example

- Given a formula $F : x + y > z \Rightarrow y > z - x$
- The previous interpretation $I = (D_I, \alpha_I)$
  - $D_I = \mathbb{Z}$ (set of integers)
  - $\alpha_I = \{ + \mapsto +_{\mathbb{Z}}, - \mapsto -_{\mathbb{Z}}, > \mapsto >_{\mathbb{Z}}, x \mapsto 13, y \mapsto 42, z \mapsto 1\}$
- is satisfying because

1. $I \models x + y > z$ since $\alpha_I[x + y > z] = 13 + 42 >_{\mathbb{Z}} 1$
2. $I \models y > z - x$ since $\alpha_I[y > z - x] = 42 >_{\mathbb{Z}} 1 -_{\mathbb{Z}} 13$
3. $I \models F$ by 1, 2, and the semantics of $\Rightarrow$

Satisfiability and Validity

- A formula F is satisfiable iff there exists an interpretation I such that $I \models F$.
- A formula F is valid iff for all interpretations $I, I \models F$.
- Technically, satisfiability and validity only apply to *closed* FOL formulae, which do not have free variables.
- However, there's a convention:
  - If we say a formula having free variables is valid, we treat free variables as universally quantified variables (e.g.,"$\forall x \, . \, x > y$ is valid" means "$\forall x, y \, . \, x > y$ is valid") (Similar for $\exists$)
- Duality holds: $\forall * . F$ is valid $\iff$ $\exists * . \neg F$ is unsatisfiable

Review: Semantic Argument Method

- Assume a formula is invalid, and check if it leads to a contradiction by applying *proof rules.*
- A proof rule has one or more premises (assumed facts) and deductions (deduced facts)

$$\frac{\text{Assumed fact1 , …, Assumed fact n}}{\text{Deduced fact1, … , Deduced fact n}}$$

Review: Semantic Argument Method for PL

$$\frac{I \models \neg F}{I \not\models F} \qquad \frac{I \not\models \neg F}{I \models F}$$

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \qquad \frac{I \not\models F \wedge G}{I \not\models F \mid I \not\models G} \qquad \frac{I \models F \vee G}{I \models F \mid I \models G} \qquad \frac{I \not\models F \vee G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

OR

$$\frac{I \models F \rightarrow G}{I \not\models F \mid I \models G} \qquad \frac{I \not\models F \rightarrow G}{\begin{array}{l} I \models F \\ I \not\models G \end{array}} \qquad \frac{\begin{array}{l} I \models F \\ I \not\models F \end{array}}{I \models \bot}$$

Contradiction!

Semantic Argument Method for FOL

- The rules for PL +
- Universal elimination I

$$\frac{I \models \forall x.\ F}{I \triangleleft \{x \mapsto \mathsf{v}\} \models F} \quad \text{for any } \mathsf{v} \in D_I$$

- Existential elimination I

$$\frac{I \not\models \exists x.\ F}{I \triangleleft \{x \mapsto \mathsf{v}\} \not\models F} \quad \text{for any } \mathsf{v} \in D_I$$

There rules are usually applied using a domain element v that was introduced earlier in the proof.

Semantic Argument Method for FOL

- Universal elimination 2

$$\frac{I \models \exists x.\ F}{I \triangleleft \{x \mapsto \mathsf{v}\} \models F} \quad \text{for a } \textit{fresh}\ \mathsf{v} \in D_I$$

Not used before

- Existential elimination 2

$$\frac{I \not\models \forall x.\ F}{I \triangleleft \{x \mapsto \mathsf{v}\} \not\models F} \quad \text{for a } \textit{fresh}\ \mathsf{v} \in D_I$$

When applying these rules, v must not have been previously used in the proof.

Semantic Argument Method for FOL

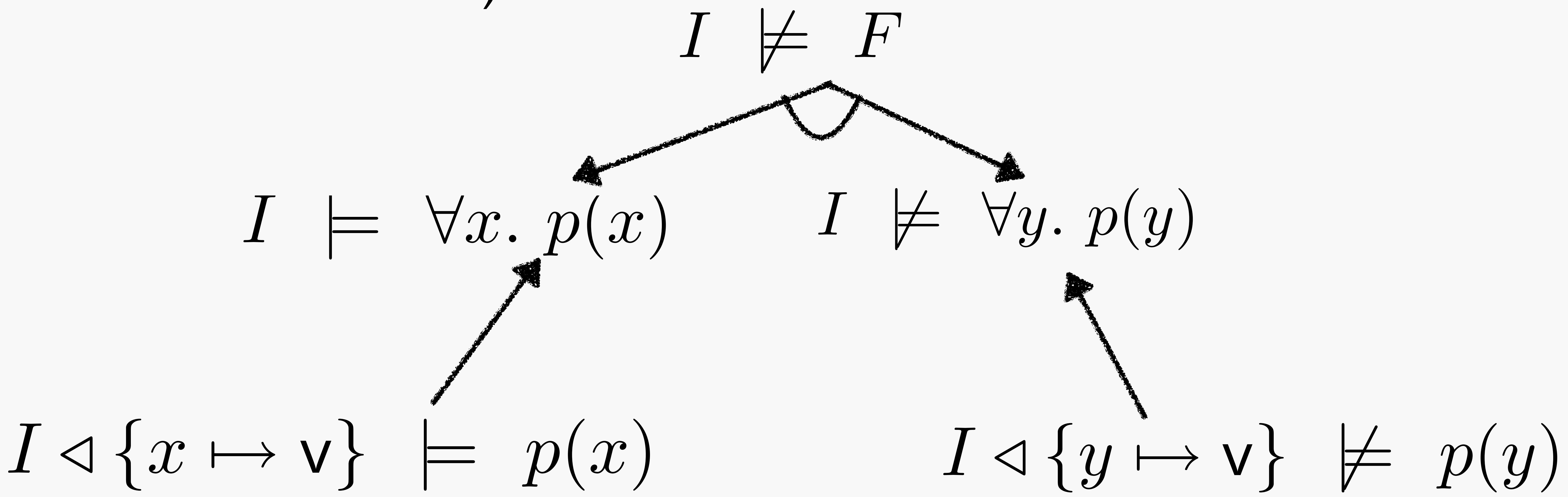
- Contradiction

$$\frac{\begin{array}{l} J : I \lhd \cdots \models p(s_1, \ldots, s_n) \\ K : I \lhd \cdots \not\models p(t_1, \ldots, t_n) \end{array}}{I \models \bot} \quad \text{for } i \in \{1, \ldots, n\},\ \alpha_J[s_i] = \alpha_K[t_i]$$

The inputs to p are semantically equivalent under J and K but the outcome of p is different, which is a contradiction.

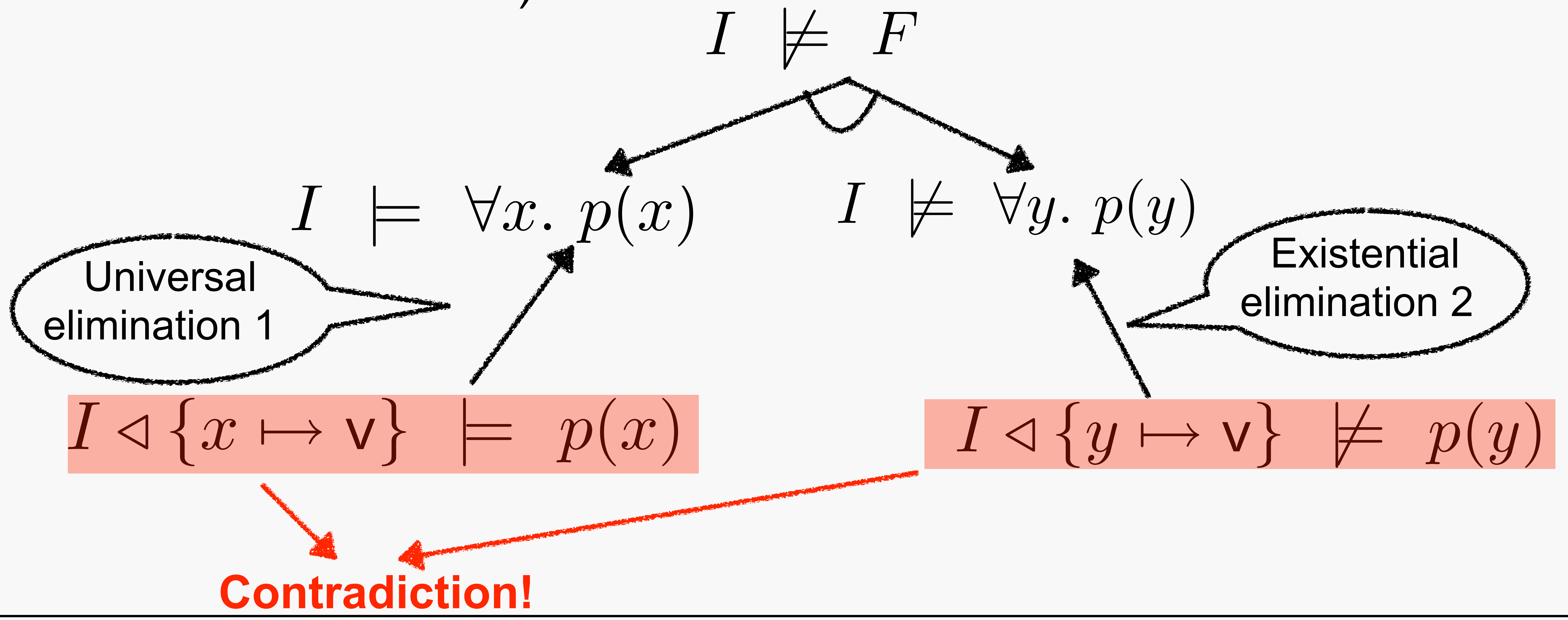
Example

- Let's prove $F: \; (\forall x.\; p(x)) \;\rightarrow\; (\forall y.\; p(y))$ is valid
- Assume it is invalid and derives a contradiction (then, the assumption is wrong, which means F is valid).



Example

- Let's prove $F : (\forall x.\ p(x)) \rightarrow (\forall y.\ p(y))$ is valid
- Assume it is invalid and derives a contradiction (then, the assumption is wrong, which means F is valid).



Example

- Let's prove $F : \; p(a) \; \rightarrow \; \exists x.\; p(x)$ is valid
- Assume it is invalid and derives a contradiction.

$$I \not\models F$$

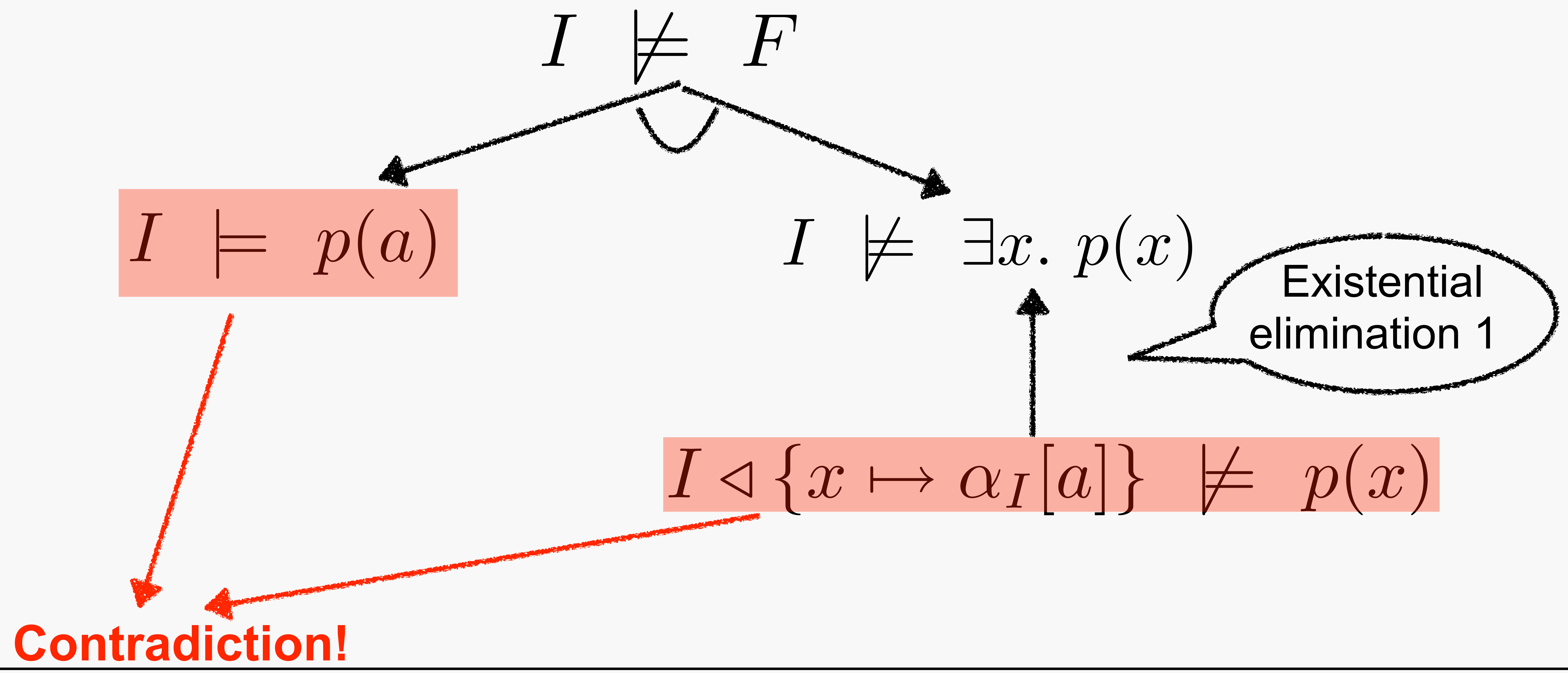
$$I \models p(a) \qquad I \not\models \exists x.\; p(x)$$

Existential elimination 1

$$I \triangleleft \{x \mapsto \alpha_I[a]\} \not\models p(x)$$

Example

- Let's prove $F: \; p(a) \rightarrow \exists x.\, p(x)$ is valid
- Assume it is invalid and derives a contradiction.



Example

- Let's prove $F: \ (\forall x.\ p(x,x)) \ \rightarrow \ (\exists x.\ \forall y.\ p(x,y))$ is invalid
- To show that it is invalid, we find a *counterexample* I such that

$$I \ \models \ \neg((\forall x.\ p(x,x)) \ \rightarrow \ (\exists x.\ \forall y.\ p(x,y)))$$

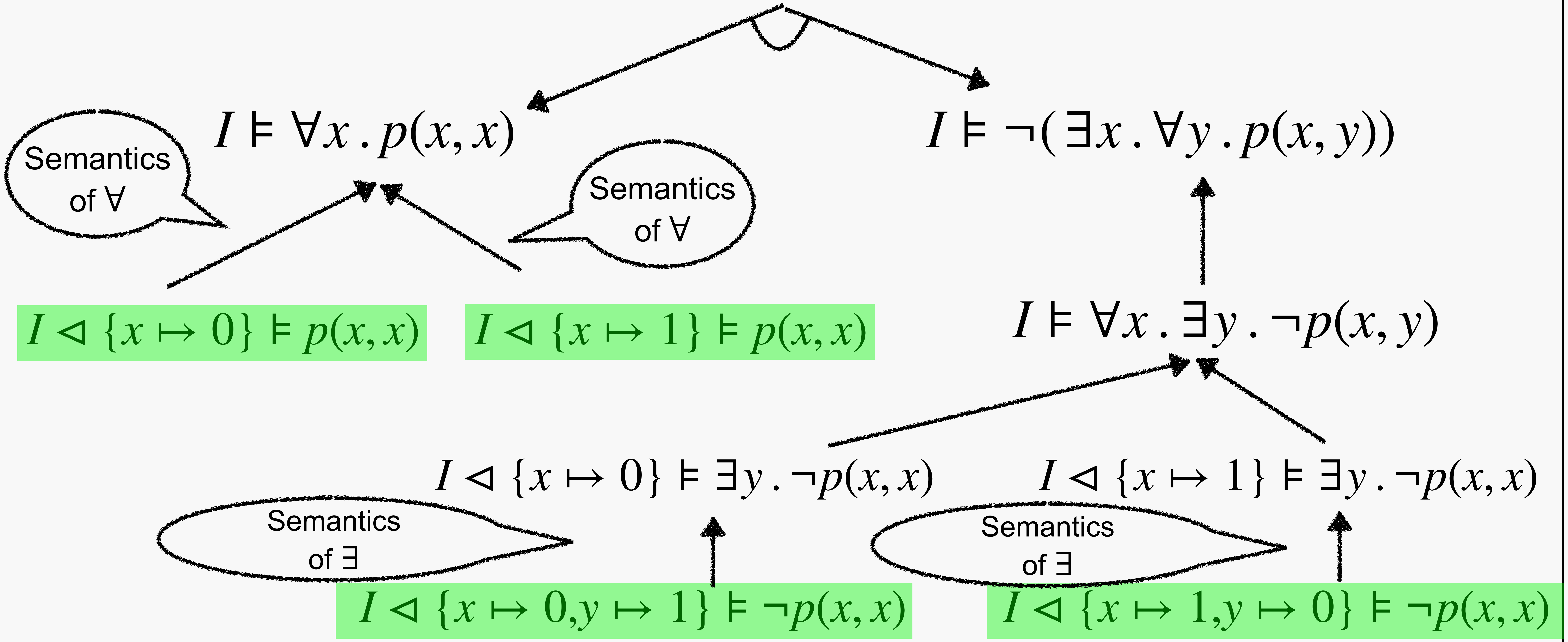
equivalently,

$$I \ \models \ (\forall x.\ p(x,x)) \ \wedge \ \neg(\exists x.\ \forall y.\ p(x,y))$$

- Choose $D_I = \{0,1\}$ and $p_I = \{(0,0) \mapsto$ true, $(1,1) \mapsto$ true$\}$

Example

$$I \models (\forall x.\ p(x,x)) \land \neg(\exists x.\ \forall y.\ p(x,y))$$



Why is FOL Called "First-Order"?

- In logic, the order refers to what kind of variables quantifiers can bind.
- In first-order logic, we can write $\forall x \, . \, P(x)$ ("for every individual x, $P(x)$ holds")
- But we cannot write $\forall P \, . \, \exists x \, . \, P(x)$ ("for every predicate P, there exists …)
- The above formula is allowed in *second-order logic*.
- *First-order entities*: objects in a domain, *second-order entities*: predicates and functions on the first-order entities, …
- FOL restricts quantification to **first-order entities** and **not** over predicates or functions.

Why is FOL Called "First-Order"?

Logic Type	Can Quantify Over	Example
Propositional	Nothing	N/A
First-Order (FOL)	Objects	$\forall x \,.\, P(x)$
Second-Order (SOL)	Predicates/functions over objects	$\forall P \,.\, \exists x \,.\, P(x)$
Higher-Order	Predicates of predicates, etc	$\forall Q \,.\, Q(P)$

Summary

- Syntax and semantics of first-order logic
- Terms, functions, predicates
- Quantifiers