# Program Reasoning
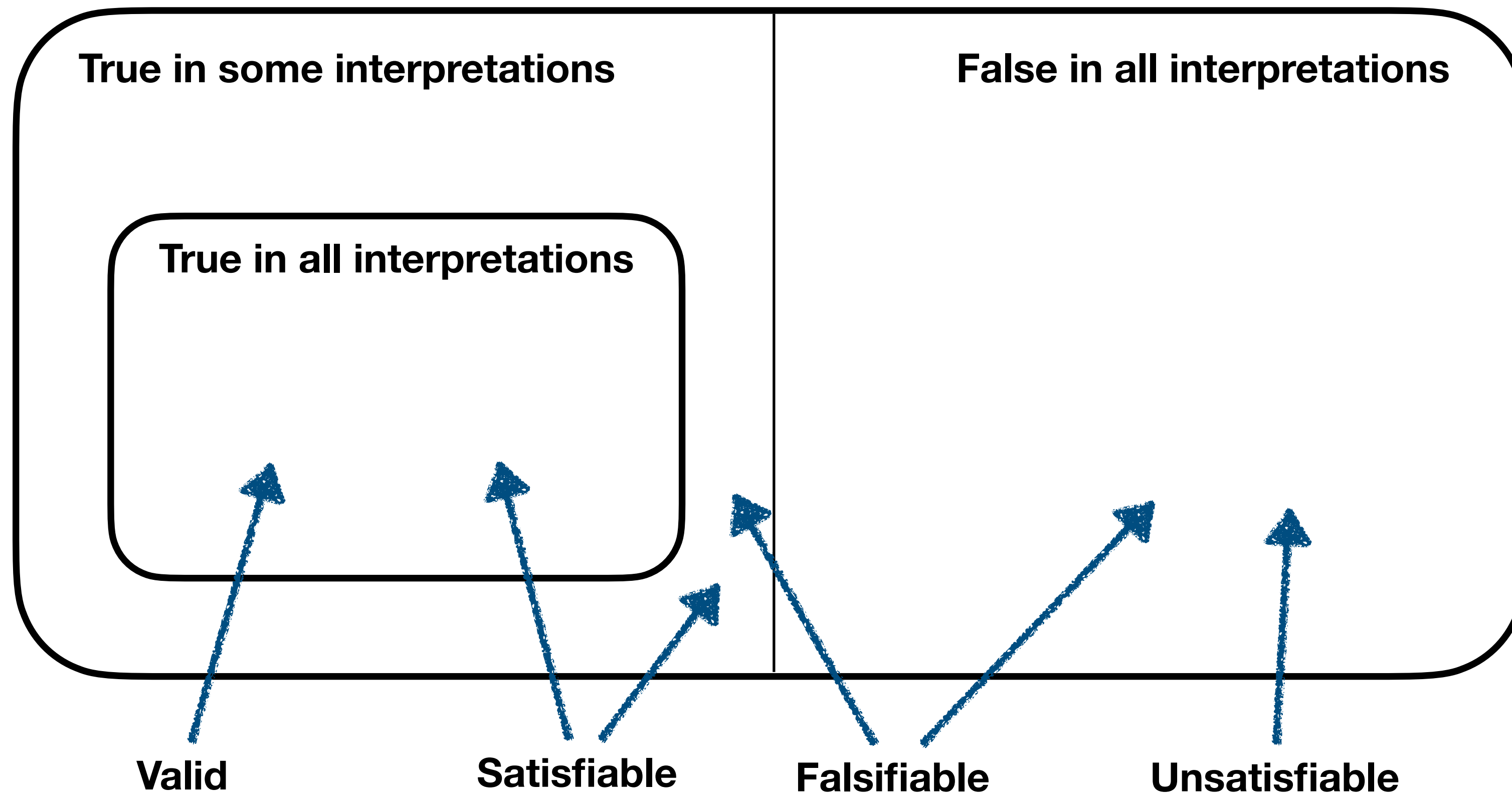
## 5. First-order Logic

Kihong Heo

**KAIST**

# First-order Logic

- An extension of propositional logic with predicates, functions and quantifiers

- FOL is more expressive than propositional logic

  - Expressive enough to reason about programs

- Not admit completely automated reasoning (i.e., undecidable)

  - "Yes, $F$ is valid" (so, $\neg F$ is unsatisfiable)

  - "Yes, $\neg F$ is valid" (so, $F$ is unsatisfiable)

  - "…" (may not terminate if $F$ is invalid)

  - Note: "$F$ is invalid" $\neq$ "$\neg F$ is valid"

# Valid, Satisfiable, Falsifiable and Unsatisfiable

True in some interpretations

False in all interpretations

True in all interpretations

Valid

Satisfiable

Falsifiable

Unsatisfiable

# Syntax (1): Terms

- Objects that we are reasoning about

- Terms evaluate to values in an underlying domain (e.g., integers, strings, lists, etc)

  - C.f., All formulae in PL evaluate to true or false

- Basic terms: variables ($x$, $y$, $z$, …) and constants ($a$, $b$, $c$, …)

- Composite terms: $n$-ary functions applied to $n$ terms

  - A constant can be viewed as a 0-ary function

- Example:

  - $a$, $x$, $f(a)$, $g(x, b)$, $f(g(x, f(b))$

# Syntax (2): Predicates

- Generalization of propositional variables in PL ($p,\ q,\ r,\ \dots$)

- An n-ary predicate takes n terms as arguments

  - A FOL propositional variable is a 0-ary predicate ($P,\ Q,\ R,\ \dots$)

- Example:

  - $P,\ p\,(f(x),\ g\,(x,\ f(x)))$

  - *isHappy*($x$), *love*($x,\ y$), *betterThan*($x,\ y$)

# Predicates and Functions

- They look similar but different

- Function terms can be nested within each other and inside relation constants

  - E.g., $f(f(x))$, $p(f(x))$

- Predicates cannot be nested within function terms or other predicates

  - E.g., $f(p(x))$, $p(p(x))$

# Syntax (3): Formula

- Atom: basic elements

  - truth symbols ($\bot$ and $\top$), $n$-ary predicates applied to $n$ terms

- Literal: an atom $\alpha$ or its negation $\neg\alpha$

- Formula: literal, the app. of a logical conn. to formulae, or the app. of a quantifier to a formula

$$
\begin{aligned}
F \quad \rightarrow \quad & \bot \mid \top \mid p(t_1, \ldots, t_n) \\
\mid \quad & \neg F \\
\mid \quad & F_1 \wedge F_2 \\
\mid \quad & F_1 \vee F_2 \\
\mid \quad & F_1 \rightarrow F_2 \\
\mid \quad & F_1 \leftrightarrow F_2 \\
\mid \quad & \exists x.F[x] \\
\mid \quad & \forall x.F[x]
\end{aligned}
$$

# Quantification

$$\exists x.F[x]$$
$$\forall x.F[x]$$

"$x$ is bound in $F[x]$"

scope of
quantifier

scope of y

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

scope of x

- A variable is free in $F[x]$ if it is not bound

- free$(F)$ and bound$(F)$ denote the free and bound variables of $F$

- A formula $F$ is closed if $F$ has no free variables

- If free$(F) = \{x_1, \ldots, x_n\}$, the universal closure is $\forall x_1, \ldots, x_n . F$ (usually $\forall * . F$)
  and its existential closure is $\exists x_1, \ldots, x_n . F$ (usually $\exists * . F$)

# Example

- Every dog has its day   $\forall x.dog(x) \to \exists y.day(y) \land itsDay(x, y)$

- Some dogs have more days than others   $\exists x, y.dog(x) \land dog(y) \land \#days(x) > \#days(y)$

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

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

- Fermat's Last Theorem

$$\forall n.integer(n) \land n > 2$$
$$\to \forall x, y, z.$$
$$integer(x) \land integer(y) \land integer(z) \land x > 0 \land y > 0 \land z > 0$$
$$\to x^n + y^n \neq z^n$$

# Interpretation (1)

- A FOL interpretation $I : (D_I, \alpha_I)$ is a pair of a domain and an assignment

  - $D_I$ : a nonempty set of values such as integers, real numbers, etc

  - $\alpha_I$ : a mapping from variables, constants, functions, and predicate symbols to elements, functions, and predicates over $D_I$

    - Each variable $x$ is assigned to a value from $D_I$

    - Each $n$-ary function symbol $f$ is assigned an $n$-ary function $f_I : D_I^n \to D_I$

    - Each $n$-ary predicate symbol $p$ is assigned an $n$-ary predicate $p_I : D_I^n \to \{\text{true}, \text{false}\}$

# Interpretation (2)

- Interpretation of complicated atoms: recursively defined

- Evaluate arbitrary terms recursively:

  - $\alpha_I[f(t_1, \ldots, t_n)] = \alpha_I[f](\alpha_I[t_1], \ldots, \alpha_I[t_n])$

- Evaluate arbitrary terms recursively:

  - $\alpha_I[p(t_1, \ldots, t_n)] = \alpha_I[p](\alpha_I[t_1], \ldots, \alpha_I[t_n])$

# Example

$$F : x + y > z \rightarrow y > z - x$$

- Note: +, -, > are just symbols and no meaning is given without an interpretation

  - Alternative form: $p(f(x,y), z) \rightarrow p(y, g(z, x))$

- The standard interpretation

  - Domain $D_I = \mathbb{Z}$

  - Assignment $\alpha_I = \{+ \mapsto +_{\mathbb{Z}}, - \mapsto -_{\mathbb{Z}}, > \mapsto >_{\mathbb{Z}}, x \mapsto 13, y \mapsto 42, z \mapsto 1, \ldots\}$

# Semantics

- Given an interpretation $I : (D_I, \alpha_I)$, $I \vDash F$ or $I \nvDash F$

$$
\begin{aligned}
&I \models \top \\
&I \not\models \bot \\
&I \models p(t_1, \ldots, t_n) \quad && \text{iff } \alpha_I[p(t_1, \ldots, t_n)] = \text{true} \\
&I \models \neg F && \text{iff } I \not\models F \\
&I \models F_1 \wedge F_2 && \text{iff } I \models F_1 \text{ and } I \models F_2 \\
&I \models F_1 \vee F_2 && \text{iff } I \models F_1 \text{ or } I \models F_2 \\
&I \models F_1 \rightarrow F_2 && \text{iff, if } I \models F_1 \text{ then } I \models F_2 \\
&I \models F_1 \leftrightarrow F_2 && \text{iff, if } I \models F_1 \text{ and } I \models F_2, \text{ or if } I \not\models F_1 \text{ and } I \not\models F_2 \\
&I \models \forall x.F && \text{iff for all } v \in D_I, I \triangleleft \{x \mapsto v\} \models F \\
&I \models \exists x.F && \text{iff there exists } v \in D_I, I \triangleleft \{x \mapsto v\} \models F
\end{aligned}
$$

where $J : I \triangleleft \{x \mapsto v\}$ denotes an $\boldsymbol{x}$-variant of $\boldsymbol{I}$

- $D_J = D_I$

- $\alpha_J[y] = \alpha_I[y]$ for all constant, free variable, function, and predicate symbols $y$ except that $\alpha_J(x) = v$

# Example

$$F : \exists x. f(x) = g(x)$$

- Consider the interpretation $I : (D_I, \alpha_I)$

  - $D_I = \{0, 1\}$

  - $\alpha_I = \{f(0) \mapsto 0, \, f(1) \mapsto 1, \, g(0) \mapsto 1, \, g(1) \mapsto 0\}$

- Compute the truth value of $F$ under $I$

  - $I \lhd \{x \mapsto v\} \nvDash f(x) = g(x)$ for $v \in D_I$

  - $I \nvDash \exists x . f(x) = g(x)$ since $v \in D_I$ is arbitrary

# "First"-order?

- First-order: quantified variables over domain elements

- What is second-order logic?

  - Quantified variables over functions or predicates

  - $\exists f \, \forall x \, . \, f(x) = f(-x)$ satisfiable?

# Satisfiability and Validity

- A formula $F$ is <span style="color:red">satisfiable</span> iff there exists an interpretation $I$ such that $I \vDash F$

- A formula $F$ is <span style="color:red">valid</span> iff for all interpretations $I$, $I \vDash F$

- Satisfiability and validity are dual: $F$ is valid iff $\neg F$ is unsatisfiable

- Satisfiability and validity are defined for closed FOL, but conventionally

  - A formula with free variables is valid : $\forall * . F$ is valid

  - A formula with free variables is satisfiable : $\exists * . F$ is satisfiable

# Proof Rules (1)

- According to the semantics of negation,

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

- According to the semantics of conjunction,

$$\frac{I \models F \wedge G}{I \models F, I \models G} \qquad\qquad \frac{I \not\models F \wedge G}{I \not\models F \mid I \not\models G}$$

# Proof Rules (2)

- According to the semantics of disjunction,

$$\frac{I \models F \vee G}{I \models F \quad | \quad I \models G}$$

$$\frac{I \not\models F \vee G}{I \not\models F, \ I \not\models G}$$

- According to the semantics of implication,

$$\frac{I \models F \rightarrow G}{I \not\models F \quad | \quad I \models G}$$

$$\frac{I \not\models F \rightarrow G}{I \models F, \ I \not\models G}$$

# Proof Rules (3)

- According to the semantics of iff,

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \mid I \models \neg F \wedge \neg G}$$

$$\frac{I \not\models F \leftrightarrow G}{I \models F \wedge \neg G \mid I \models \neg F \wedge G}$$

# Proof Rules (4)

- A contradiction exists if two variants of the original interpretation $I$ disagree

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

- Example

$$\frac{I \triangleleft \{x \mapsto a\} \models p(x), \ I \triangleleft \{y \mapsto a\} \not\models p(y)}{I \models \bot}$$

# Proof Rules (5)

- According to the semantics of universal quantification

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

(Usually applied using a domain element $v$ that was introduced earlier in the proof)

- According to the semantics of existential quantification

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

(Usually applied using a domain element $v$ that was introduced earlier in the proof)

# Example

- Prove $F : (\forall x \, . \, p(x)) \rightarrow (\exists y \, . \, p(y))$ is valid

# Proof Rules (6)

- According to the semantics of universal quantification

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

($v$ must not have been previously used in the proof)

- Example: prove $F : p(a) \rightarrow \forall x \, . \, p(x)$ is valid

(what happens if you ignore the warning?)

# Proof Rules (7)

- According to the semantics of existential quantification

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

($v$ must not have been previously used in the proof)

- Example: prove $F : (\exists x . p(x)) \rightarrow p(a)$ is valid

  (what happens if you ignore the warning?)

# Example (1)

- Prove $F : (\forall x \,.\, p(x)) \to (\forall y \,.\, p(y))$ is valid

# Example (2)

- Prove $F : (\forall x \,.\, p(x)) \rightarrow (\neg \exists y \,.\, \neg p(y))$ is valid

# Example (3)

- Prove $F : p(a) \rightarrow ( \exists x \, . \, p(x))$ is valid

# Example (4)

- Show $F : (\forall x . p(x, x)) \rightarrow (\exists x . \forall y . p(x, y))$ is invalid

# Soundness and Completeness

- Soundness: if every branch of a semantic argument proof reach $I \vDash \bot$ then $F$ is valid

- Completeness: each valid formula $F$ has a semantic argument proof in which every branch reaches $I \vDash \bot$

  - Gödel's completeness theorem: "Anything universally true is provable"

- Note: DO NOT get confused with Gödel's **incompleteness** theorem

  - First-order logic: complete (completeness theorem)

  - First-order logic in (Peano) arithmetic: incomplete (incompleteness theorem)

# Decidability

- Does there exist an algorithm to solve a problem?

  - Solve: eventually halt and return a correct answer

  - E.g., Halting problem

- Our problem: satisfiability (or dually, validity) of FOL

- Satisfiability of PL: decidable

- Satisfiability of FOL: semi-undecidable (by Church and Turing)

  - If $F$ is valid, the algorithm says "Yes"

  - If $F$ is invalid, the algorithm may not terminate

# Summary

- FOL: an extension of PL with predicates, functions and quantifiers

  - Powerful enough to reason about properties of software

- Proof system (semantic argument method) for validity

  - Sound and complete

  - Undecidable

- How to use FOL for program reasoning, mathematical reasoning, etc?

  - Next topic: First-order theories