

Predicate Logic – first order logic

Syntax — Well formed formulae (wff):

- **term**: constant symbols, variables, and n-place function symbols followed by n *terms* enclosed in parenthesis
- **atomic formula**: n-place predicate symbols followed by n *terms* enclosed in parenthesis
- **wff**: *atomic formulae* and expressions of the form $(\alpha \rightarrow \beta)$, $(\alpha \vee \beta)$, $(\alpha \wedge \beta)$, $(\neg\alpha)$, $\forall x\alpha$, or $\exists x\alpha$, where α and β are *wffs* and x is a variable

Examples:

$$\forall x(p(x) \rightarrow q(x))$$

$$\forall x\exists y(p(x) \vee q(f(y)))$$

A *wff* is **closed** if it does not contain any free variables.

To simplify the discussion, we will assume that all *wffs* are closed.

Predicate Logic – first order logic

Semantics — Interpretations:

An **interpretation I** consists of

- a non-empty set D , called the *universe*
- a mapping that assigns to each constant c a fixed element $c^I \in D$
- a mapping that assigns to each n-place function symbol f an n-ary function $f^I: D^n \rightarrow D$
- a mapping that assigns to each n-place predicate symbol p an n-ary predicate $p^I \subseteq D^n \rightarrow \{ \mathbf{true}, \mathbf{false} \}$

Note:

logic connectives $\rightarrow, \vee, \wedge$, and \neg have their usual propositional meaning

$\forall x$ and $\exists x$ mean “for all x in D ” and “there exists an x in D ”, respectively

Predicate Logic – first order logic

An interpretation I **satisfies** wff α ($\models_I \alpha$) iff α evaluates to true under the interpretation I . Such an I is called a **model** of α .

A wff α is **valid** ($\models \alpha$) iff it is satisfied for all of its interpretations.

A wff α is **unsatisfiable** iff it is not satisfiable in any of its interpretations.

A set Γ of wffs **logically implies** a wff α ($\Gamma \models \alpha$) iff for every interpretation that satisfies all members in Γ , α is also satisfied.

Question: Is there a mechanical way to derive all wffs that are logically implied by a set of wffs Γ ?

Predicate Logic – first order logic

A **deductive calculus** consists of

- A set Δ of *wffs*, called **logical axioms**
- A set of inference rules

A *wff* α is a **theorem** of a set Γ of *wffs* ($\Gamma \vdash \alpha$) iff α belongs to the set of formulae that can be generated from $\Gamma \cup \Delta$ by a finite sequence of inference rule applications. Such a sequence is called a **deduction**.

Without going into details about its infinite set Δ , there is a deductive calculus that uses only a single inference rule, called *modus ponens*:

$$\frac{\alpha, \alpha \rightarrow \beta}{\beta}$$

Example:

Assume that Δ contains $\forall x p(x) \rightarrow p(a)$. Then we can proof

$$\{\forall x p(x), (p(a) \rightarrow q(a))\} \vdash q(a)$$

with two applications of the inference rule.

Predicate Logic – first order logic

Two major properties of a deductive calculus:

- A deductive calculus is **sound** iff
 $\Gamma \vdash \alpha$ implies $\Gamma \models \alpha$
- A deductive calculus is **complete** iff
 $\Gamma \models \alpha$ implies $\Gamma \vdash \alpha$

In other words, *soundness* means “whatever can be proven is valid”, and *completeness* means “whatever is valid can be proven”. A calculus that is not sound is pretty useless. A calculus that is sound and complete is the best we can do.

Gödel showed in 1930 that first order logic is complete, i.e., there is a sound deductive calculus for first order logic that is complete.

Questions:

- Is propositional logic decidable?
- Is $\Gamma \vdash \alpha$ decidable?
- Is second order logic (allows quantification over predicates) complete?

Logic Programming and Prolog

Logic programming languages are not procedural or functional.

- Specify *relations* between objects
 - `larger(3,2)`
 - `father(tom,jane)`
- Separate logic from control:
 - Programmer declares **what** facts and relations are true
 - System determines **how** to use facts to solve problems
- Based on Predicate Logic (first order logic)
- Computation engine: theorem-proving and recursion (Unification, Resolution, Backward Chaining, Backtracking)

Free and bound variables

A variable x can occur in a *wff*

- within a term (**use**), or
- next to a quantifier \forall or \exists (**definition**).

Any use of a variable is **bound** to the “closest” surrounding definition of the variable, if such a definition exists.

For the *wffs* $\forall x\alpha$ or $\exists x\alpha$, α is called the **scope** of definition x .

Example:

$$(\forall x(p(x) \rightarrow \exists y\exists x(q(x) \vee q(y)))) \vee x)$$

Question: What are the **bindings** between the uses and definitions of variables? Note: the same variable can be defined or used several times in a *wff*.

The use of a variable is **free** in a *wff* if there is no matching definition. Otherwise it is **bound**.