St. Francis Xavier University
Department of Computer Science

CSCI 544: Computational Logic
Lecture 6: Predicate Logic I—Syntax and Semantics
Winter 2024

1 The Syntax of Predicate Logic

Predicate logic is the logical system that goes beyond individual atomic propositions and focuses on predicates, or functions over some domain that take variables and produce truth values. In this way, predicate logic should not be viewed as a replacement for propositional logic, but rather as an extension. Predicates allow us to capture properties of multiple elements at once, as well as to express relationships between elements.

Just as the proposition was the fundamental notion in propositional logic, so too is the predicate the fundamental notion of predicate logic.

Definition 1 (Predicate)**.** A predicate P takes some number of arguments $n \geq 1$ and produces a truth value. The number of arguments n is called the arity of P; alternatively, we say that P is an n-ary predicate.

We express a predicate using notation very similar to that used for writing a mathematical function: given an n-ary predicate P, we write $P(x_1, x_2, \ldots, x_n)$ to denote the predicate P taking as arguments some combination of *variables* and *constants* x_1 through x_n. (We occasionally refer to a constant using a different symbol, like a_i.) We refer to this singular predicate with variables/constants as an *atomic formula*.

Example 2. Let $L(x, y)$ denote the predicate "x is less than y", where x and y are integers. We can either take both x and y to be variables, or we can fix one of x and y to be a constant; for example, $L(x, 0)$ is an atomic formula checking whether some integer x is negative.

Following our observations, you might begin to see what sort of power predicates give to us. If P corresponds to some property of elements, then we can check at once whether any or each element x_1 through x_n satisfies that property. Depending on the property we want to check, we can determine whether *some* element satisfies that property, or whether *all* elements satisfy that property. To differentiate between the two, we use two symbols called *quantifiers*:

- The *existential quantifier* $\exists$ checks whether some element satisfies the predicate.
- The *universal quantifier* $\forall$ checks whether all elements satisfy the predicate.

These quantifiers are read as "there exists" and "for all", respectively.

We can interpret both of the logical quantifiers using logical connectives we learned during our previous study of propositional logic. If we're quantifying over elements from a set A, then

- writing $\forall x \in A\ P(x)$ is equivalent to writing $\forall x\ (x \in A \Rightarrow P(x))$, and
- writing $\exists x \in A\ P(x)$ is equivalent to writing $\exists x\ (x \in A \wedge P(x))$.

If we know the set over which we're quantifying (which will often be the case), then we can omit writing $\forall x \in A$ and simply write $\forall x$, and likewise for $\exists x$.

Example 3. Suppose a and b are students, s is a professor, and c is a course. Let $C(x, y)$ denote the predicate "x and y are classmates", $I(x, y, z)$ denote the predicate "x is y's instructor for course z", and $T(x, z)$ denote the predicate "professor x teaches course z". Then the expression

$$\forall a\ \forall b\ (\exists s\ \exists c\ T(s, c) \land I(s, a, c) \land I(s, b, c)) \Rightarrow C(a, b)$$

states that, for any pair of students a and b, if a and b are both registered in some course c taught by some professor s, then a and b are classmates.

Note that our previous example combines predicates, variables, and connectives from propositional logic to produce a complex expression encapsulating relationships between variables. This complex expression is, in fact, just a *formula* as we've seen before. As we did in our propositional logic system, we can create and write formulas in predicate logic using a set of recursive rules.

Definition 4 (Formula)**.** A statement in predicate logic is a formula if it can be constructed according to the following rules:

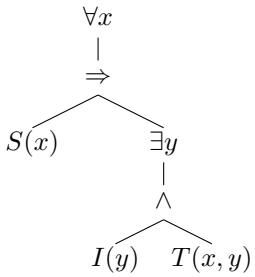
1. Every atomic formula is a formula;
2. If $P(\cdot)$ is a formula, then $\neg P(\cdot)$ is a formula;
3. If $P(\cdot)$ and $Q(\cdot)$ are formulas, then each of $P(\cdot) \land Q(\cdot)$, $P(\cdot) \lor Q(\cdot)$, $P(\cdot) \Rightarrow Q(\cdot)$, and $P(\cdot) \Leftrightarrow Q(\cdot)$ are formulas;
4. If $P(\cdot)$ is a formula, then $\exists x\ P(\cdot)$ and $\forall x\ P(\cdot)$ are formulas, where x is some variable; and
5. No other statement is a formula.

As you might expect, we can represent formulas either as a linear sequence of symbols or using a tree structure.

Example 5. Consider the following formula:

$$\forall x\ S(x) \Rightarrow (\exists y\ (I(y) \land T(x, y))).$$

This formula may be parsed, for example, to mean something like "if x is a student, then there is some instructor y where x is a student of y". This formula can be represented by the following tree:



Observe in this tree that the existential ($\exists$) and universal ($\forall$) quantifiers have a branching factor of 1; that is, a quantifier in a tree has exactly one child. In this way, quantifiers behave much the same as a unary connective like negation ($\neg$).

Since quantifiers have only a single child in the tree representation of a formula, it's reasonable to say that a quantifier applies to everything in the subtree beneath that quantifier. In other words, if we see a quantifier like $\forall x$ and, deeper in the subtree underneath that quantifier, we see a predicate involving x, we can conclude that the quantifier and the predicate are talking about the same x from the same domain of variables. Much the same situation arises in a programming environment, where we can define a variable in some context—such as within a function—and be sure that any reference to that variable in that context is to the same object in memory. Just as programming language theorists can speak of the *scope* of that variable, so too can we speak of the *scope* of a quantifier.

Definition 6 (Scope)**.** Given a variable x and a formula $P(x, \cdots)$ involving x, we say that either of the formulas $\exists x\ P(x, \cdots)$ or $\forall x\ P(x, \cdots)$ are quantified formulas, and x is a quantified variable in the scope P.

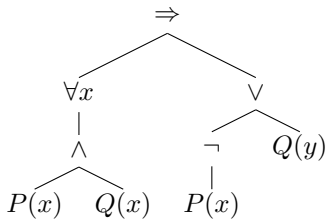
Focusing on the variables of a formula P, we say that a variable x in P is a *free variable* if and only if it does not appear in the scope of some quantified formula. Otherwise, we say that x is a *bound variable*.

For formulas, on the other hand, we say that a formula is *closed* if all of its variables x_1 through x_n are bound. If x_1 through x_n are free variables of a formula, then we can obtain the *closure* of the formula by taking either $\forall x_1 \ldots \forall x_n$ or $\exists x_1 \ldots \exists x_n$.

Example 7. Consider the formula

$$(\forall x\ (P(x) \wedge Q(x))) \Rightarrow (\neg P(x) \vee Q(y)).$$

(You may ascribe any meaning you like to this example formula; get creative!) The formula's corresponding tree representation is



In the left subtree, all occurrences of x beneath the quantifier $\forall x$ are bound variables. In the right subtree, the variable x appearing in $P(x)$ is not bound by the quantifier $\forall x$, so it is a free variable. Likewise, the variable y appearing in $Q(y)$ is a free variable. Thus, it is possible for a single variable (like x) to be both bound and free at different points within a single formula, but each individual occurrence of the variable is either bound or free; never both at once.

2 The Semantics of Predicate Logic

The notion of an *interpretation* differs slightly between our two systems of propositional and predicate logic. Recall that, in propositional logic, an interpretation was an assignment of truth values to propositions. If we were to extend this directly to predicate logic, then we would be assigning truth values to formulas. However, we require slightly more information: we must know also what is assigned to the variables and constants of each predicate in the formula. Thus, in our definition of an interpretation for predicate logic, we incorporate these aspects as well.

Definition 8 (Interpretation for a formula)**.** Let A be a formula where $\{P_1, \ldots, P_n\}$ is a set of predicates appearing in A and $\{a_1, \ldots, a_m\}$ is a set of constants appearing in A. An interpretation for A is a tuple

$$(D, \{R_1, \ldots, R_n\}, \{d_1, \ldots, d_m\}),$$

where D is a nonempty domain, R_i is an n-ary relation on D assigned to the n-ary predicate P_i, and d_j is an element of D assigned to the constant a_j.

We can define an interpretation for a set of formulas in a similar manner.

Example 9. Consider a formula $\forall x\ P(a, x)$. We can come up with any number of interpretations for this formula. For example:

- The interpretation $\mathscr{I}_1 = (\mathbb{N}, \{\geq\}, \{0\})$ ascribes to the formula the meaning "for every natural number x, $x \geq 0$";
- The interpretation $\mathscr{I}_2 = (\mathbb{N}, \{|\}, \{1\})$ ascribes to the formula the meaning "for every natural number x, $1 \mid x$";

- The interpretation $\mathscr{I}_3 = (\{\mathtt{0},\mathtt{1}\}^*, \{\text{sub}\}, \{\epsilon\})$ ascribes to the formula the meaning "for every word x over the alphabet $\{\mathtt{0},\mathtt{1}\}$, the empty word ϵ is a subword of x"; and
- The interpretation $\mathscr{I}_4 = (G, E, \{a\})$ ascribes to the formula the meaning "for every vertex x in a graph G, (a, x) is an edge in G".

Now, unlike with propositional logic, the truth value of a given formula may vary depending on the value taken by each variable in the formula. For example, if we have a formula $P(x, 10)$ that is taken to mean "x is greater than 10", this formula is not universally true or false; its truth value depends on the value of x. Thus, when we specify an interpretation for a formula, we must also specify an *assignment* of values to variables. Only then can we evaluate the truth value of a formula.

Definition 10 (Assignment to variables of a formula)**.** Let $\mathscr{I}_A$ be an interpretation for a formula A. An assignment $\sigma_{\mathscr{I}_A} : V \to D$ is a mapping of free variables $x \in V$ to elements $d \in D$ from the domain of $\mathscr{I}_A$.

If we fix the assignment of some element d_i to the free variable x_i in a formula A, then we write $\sigma_{\mathscr{I}_A}[x_i/d_i]$. As with interpretations, we can define an assignment to variables in a set of formulas in a similar manner.

Finally, we can define the notion of a truth value for a formula in predicate logic.

Definition 11 (Truth value of a formula)**.** Let $\mathscr{I}$ be an interpretation for a formula A (where A may be composed of smaller formulas A_1 and A_2), and let $\sigma_{\mathscr{I}_A}$ be an assignment. The truth value of A under $\mathscr{I}_A$ and $\sigma_{\mathscr{I}_A}$, denoted $v_{\sigma_{\mathscr{I}_A}}(A)$, is defined inductively according to the following rules:

- $v_{\sigma_{\mathscr{I}_A}}(A) = \text{T}$ if and only if:
  - $A = P_k(c_1, \ldots, c_n)$ is an atomic formula where each c_i is either a variable or a constant;
  - $(d_1, \ldots, d_n) \in R_k$, where R_k is the relation assigned to P_k by $\mathscr{I}_A$;
  - d_i is the element of D assigned to c_i, either by $\mathscr{I}_A$ if c_i is a constant or by $\sigma_{\mathscr{I}_A}$ if c_i is a variable;
- $v_{\sigma_{\mathscr{I}_A}}(\neg A) = \text{T}$ if and only if $v_{\sigma_{\mathscr{I}_A}}(A) = \text{F}$;
  $v_{\sigma_{\mathscr{I}_A}}(\neg A) = \text{F}$ if and only if $v_{\sigma_{\mathscr{I}_A}}(A) = \text{T}$;
- $v_{\sigma_{\mathscr{I}_A}}(A_1 \wedge A_2) = \text{T}$ if and only if $v_{\sigma_{\mathscr{I}_A}}(A_1) = \text{T}$ and $v_{\sigma_{\mathscr{I}_A}}(A_2) = \text{T}$;
  $v_{\sigma_{\mathscr{I}_A}}(A_1 \wedge A_2) = \text{F}$ otherwise;
  (and similarly for $\vee$, $\Rightarrow$, and $\Leftrightarrow$);
- $v_{\sigma_{\mathscr{I}_A}}(\forall x\ A) = \text{T}$ if and only if $v_{\sigma_{\mathscr{I}_A}[x/d]}(A) = \text{T}$ for all $d \in D$; and
- $v_{\sigma_{\mathscr{I}_A}}(\exists x\ A) = \text{T}$ if and only if $v_{\sigma_{\mathscr{I}_A}[x/d]}(A) = \text{T}$ for some $d \in D$.

2.1 Interlude: Interpretations and Assignments

Before we continue, we must make one small note about the kinds of formulas we will focus on in the following sections. Recall that a formula is closed if all of the variables in the formula are bound. In order for us to avoid having to deal with both interpretations and assignments at once, we will assume that all of the formulas we deal with in this and future sections are closed. In this way, we only need to concern ourselves with interpretations.

The property of the truth value of a closed formula being independent of assignments is summarized in the following proposition:

Proposition 12. *Let A be a closed formula, and let $\mathscr{I}_A$ be an interpretation for A. Then the truth value of A, $v_{\sigma_{\mathscr{I}_A}}(A)$, does not depend on the assignment $\sigma_{\mathscr{I}_A}$.*

Therefore, when we're dealing with the truth value of a closed formula A, we need only write $v_{\mathscr{I}}(A)$.

What if we're dealing with a non-closed formula A'? Using the notion of the closure of a formula, we can make observations about the truth value of A' via the following proposition.

Proposition 13. *Let $A' = P(x_1, \ldots, x_n)$ be a non-closed formula with free variables $x_1, \ldots, x_n$, and let $\mathscr{I}$ be an interpretation. Then*

- $v_{\sigma_{\mathscr{I}_{A'}}}(A') = T$ *for some assignment* $\sigma_{\mathscr{I}_{A'}}$ *if and only if* $v_{\mathscr{I}}(\exists x_1 \ldots \exists x_n\ A') = T$*; and*
- $v_{\sigma_{\mathscr{I}_{A'}}}(A') = T$ *for all assignments* $\sigma_{\mathscr{I}_{A'}}$ *if and only if* $v_{\mathscr{I}}(\forall x_1 \ldots \forall x_n\ A') = T$.

2.2 Logical Equivalence

In predicate logic, the notion of *logical equivalence* is just as it is in predicate logic, after applying the appropriate changes.

Definition 14 (Logical equivalence)**.** Let $U = \{A_1, A_2\}$ be a pair of closed formulas. If $v_{\mathscr{I}_U}(A_1) = v_{\mathscr{I}_U}(A_2)$ for all interpretations $\mathscr{I}_U$ of the set U, then we say that A_1 and A_2 are logically equivalent, and we denote this by $A_1 \equiv A_2$.

Of course, since predicate logic introduces the new existential and universal quantifiers as syntactic symbols, we can specify a number of logically equivalent formulations involving these quantifiers. Below, we summarize some of these identities (though beware: some of these formulations are not bidirectional/equivalent, but instead only unidirectional, and this is denoted by a standard implication symbol).

$$\exists x\ P(x) \equiv \neg\forall x\ \neg P(x)$$
$$\forall x\ P(x) \equiv \neg\exists x\ \neg P(x)$$

Table 1: Duality

Matching quantifiers

$$\exists x \exists y\ P(x,y) \equiv \exists y \exists x\ P(x,y)$$
$$\forall x \forall y\ P(x,y) \equiv \forall y \forall x\ P(x,y)$$

Mixed quantifiers

$$\exists x \forall y\ P(x,y) \Rightarrow \forall y \exists x\ P(x,y)$$

Table 2: Commutativity

Conjunction

$$\exists x\ (P(x) \wedge Q(x)) \Rightarrow (\exists x\ P(x) \wedge \exists x\ Q(x))$$
$$\forall x\ (P(x) \wedge Q(x)) \equiv (\forall x\ P(x) \wedge \forall x\ Q(x))$$

Disjunction

$$\exists x\ (P(x) \vee Q(x)) \equiv (\exists x\ P(x) \vee \exists x\ Q(x))$$
$$(\forall x\ P(x) \vee \forall x\ Q(x)) \Rightarrow \forall x\ (P(x) \vee Q(x))$$

Table 3: Distributivity over $\wedge$ and $\vee$

Implication

$$\exists x\ (P(x) \Rightarrow Q(x)) \equiv (\forall x\ P(x) \Rightarrow \exists x\ Q(x))$$
$$\exists x\ (P(x) \Rightarrow Q) \Rightarrow (\forall x\ P(x) \Rightarrow Q)$$
$$\exists x\ (P \Rightarrow Q(x)) \Rightarrow (P \Rightarrow \exists x\ Q(x))$$
$$\exists x\ P(x) \Rightarrow \exists x\ Q(x) \Rightarrow \forall x\ (P(x) \Rightarrow Q(x))$$
$$\forall x\ (P(x) \Rightarrow Q) \equiv (\exists x\ P(x) \Rightarrow Q)$$
$$\forall x\ (P \Rightarrow Q(x)) \equiv (P \Rightarrow \forall x\ Q(x))$$

Equivalence

$$\forall x\ (P(x) \Leftrightarrow Q(x)) \Rightarrow (\forall x\ P(x) \Leftrightarrow \forall x\ Q(x))$$
$$\exists x\ (P(x) \Leftrightarrow Q(x)) \Rightarrow (\exists x\ P(x) \Leftrightarrow \exists x\ Q(x))$$

Table 4: Distributivity over $\Rightarrow$ and $\Leftrightarrow$

2.3 Substitution

Recall that, when we introduced the notion of assignments, we used a notation $\sigma_{\mathscr{I}_A}[x_i/d_i]$ to denote the assignment of an element d_i to the variable x_i in a formula A. We can generalize this notion to apply not only to individual variables, but to all variables in an entire formula or subformula, and we achieve this generalization using *substitutions*.

Substitutions are perhaps most easily viewed through the lens of trees. If we wish to substitute a variable x located at some leaf of the tree corresponding to a formula, we will replace that leaf with a new subtree containing whatever we are substituting: another variable or some constant. For clarity, we will refer to variables and constants collectively as *terms*.

We can't necessarily substitute anything we want into a formula; for example, we can't substitute a variable with a predicate, since predicates can't take nested predicates as arguments. However, substitutions with

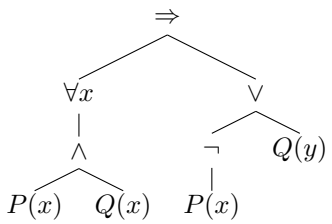
terms work perfectly fine. We can even take a function to be a term, like $f(x, y) = x + y$; unlike predicates, which return truth values, functions return values that we can then use as arguments in our original predicate.

Definition 15 (Substitution)**.** Given a variable x in a formula A and a term t, we take $A[x/t]$ to mean the formula obtained by replacing every free occurrence of x in A with t.

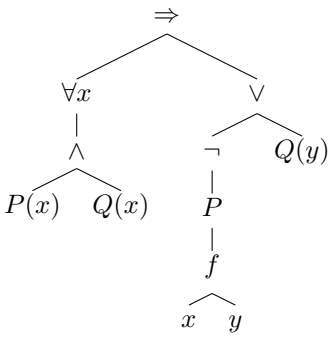
Observe that, when we write $A[x/t]$, we are not referring to a formula explicitly. We are instead referring to a formula that we obtain by applying the "operation" $[x/t]$ to the original formula A. Thus, the substitution itself is not a formula, but the result of the substitution is a formula.

Note also that, when we perform a substitution, we must take into account the scope of the variable we're substituting. We can only replace free occurrences of a variable in a substitution; from our tree perspective, we can only replace occurrences of a variable x if that occurrence doesn't appear in a subtree rooted at either $\exists x$ or $\forall x$.

Example 16. Recall the formula $A = (\forall x\ (P(x) \wedge Q(x))) \Rightarrow (\neg P(x) \vee Q(y))$ from an earlier example. As we saw, we could represent this formula as the following tree:



Suppose we perform the substitution $A[x/f(x, y)]$; that is, we replace all free occurrences of the variable x with the function $f(x, y)$. Doing so, we would obtain the following tree (while taking a few liberties to handle parentheses):



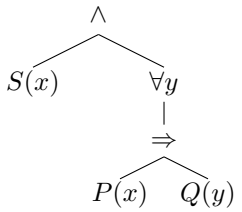
Observe that the occurrences of x in the left subtree were not modified, since both of those occurrences are bound by the quantifier $\forall x$. The only free occurrence of x appeared in the right subtree.

When we use substitution, we must be cautious not to fall into the trap of *variable capture*. In essence, variable capture occurs when we perform a substitution $A[x/t]$, where the term t contains another variable y, and some free occurrence of x in A appears in the scope of a quantifier involving y, like $\exists y$ or $\forall y$. What happens after the substitution is that the variable y occurring in the term t is now inadvertently captured by the quantifier, which ultimately changes the meaning of the formula.

To be sure that we don't fall victim to variable capture, we must make sure that the term t we're substituting is *free* for the variable x being substituted. Note that the definition of a term's "freeness" is distinct from the definition of a "free variable" we learned earlier, and the two uses are opposites, in a sense: a term *is* free for x if there *are not* any free occurrences of x within some scope of the term.

Definition 17 (Freeness)**.** Given a formula A, a variable x, and a term t, we say that t is free for x in A if no free occurrence of x in A occurs in the scope of either $\exists y$ or $\forall y$ for any variable y occurring in t.

Example 18. Consider the formula $B = S(x) \wedge (\forall y\ (P(x) \Rightarrow Q(y)))$. We can represent this formula as the following tree:



Suppose we take the term $t = f(y, y)$ and perform the substitution $B[x/f(y, y)]$. Both occurrences of x in B are free, and we could perform the substitution with no issues in the left subtree. However, the right subtree includes the quantifier $\forall y$, and substituting x with $f(y, y)$ in the right subtree introduces a new variable y that is captured by the quantifier.

2.4 Satisfiability and Validity

We conclude by defining the notions of *satisfiability* and *validity* in predicate logic. Fortunately, apart from our previous observations, satisfiability and validity are the same as in propositional logic.

Definition 19 (Satisfiability)**.** Let A be a formula. Then A is satisfiable if and only if $v_{\mathscr{I}}(A) = \mathrm{T}$ for some interpretation $\mathscr{I}$.

Naturally, if A is not satisfiable, then we say that it is *unsatisfiable*. In terms of notation, we occasionally denote the property of $v_{\mathscr{I}}(A) = \mathrm{T}$ by writing $\mathscr{I} \vDash A$, and in this case we say that A is *true* in $\mathscr{I}$ or, alternatively, $\mathscr{I}$ is a *model* for A.

Definition 20 (Validity)**.** Let A be a formula. Then A is valid if and only if $v_{\mathscr{I}}(A) = \mathrm{T}$ for all interpretations $\mathscr{I}$, and we denote the validity of A by $\vDash A$.

Likewise, if A is not valid, then we say that it is *falsifiable*.

Example 21. Let's consider a selection of different formulas and evaluate both their satisfiability and validity.

- Let $A_1 = \exists x\ \exists y\ (P(x) \wedge \neg P(y))$. This formula is satisfiable only over a domain D where $|D| \geq 2$. The formula is not valid since we could take an interpretation whose domain D is such that $|D| = 1$.
- Let $A_2 = \forall x\ \exists y\ P(x, y)$. This formula is satisfiable only in an interpretation that assigns a total function to the predicate P; for instance, $y = -2x$ for $x, y \in \mathbb{Z}$. The formula is not valid since we could take an interpretation whose domain makes the function non-total, like $\mathbb{N}$.
- Let $A_3 = \forall x\ \forall y\ (P(x, y) \Rightarrow P(y, x))$. This formula is satisfiable only in an interpretation that assigns a symmetric relation to the predicate P; for instance, equality. The formula is not valid since we could take an interpretation that assigns a non-symmetric relation to P, like $<$.
- Let $A_4 = \forall x\ (P(x) \wedge Q(x)) \Leftrightarrow (\forall x\ P(x) \wedge \forall x\ Q(x))$. This formula is valid, as we can prove that the universal quantifier distributes over conjunction, but not disjunction.
- Let $A_5 = \exists x\ (P(x) \vee Q(x)) \Leftrightarrow (\exists x\ P(x) \vee \exists x\ Q(x))$. This formula is valid, as we can prove that the existential quantifier distributes over disjunction, but not conjunction.