Lecture Notes for

Logic and Computability

Course Number: IND04033UF

Contact

Bettina Könighofer
Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Austria
bettina.koenighofer@iaik.tugraz.at

6
Predicate Logic

The Limitations of Propositional Logic. Propositional logic has limitations when encoding declarative sentences. It is quite easy to encode logical sentence components like *not*, *and*, *or*, *if ... then*. However, we would also like to express sentence components like *there exists ...* and *for all ...* which is not possible in propositional logic. To overcome this limitation, we need a more powerful type of logic like *predicate logic*, also called *first-order logic*.

Example. When trying to model the following sentence in propositional logic:

"Every person who is 18 years or older is eligible to vote.",

we need to define p as $p :=$ "Every person who is 18 years or older is eligible to vote."

The statement above cannot be adequately expressed using only propositional logic, since propositional logic is not expressive enough to deal with quantified variables. Since the sentence does not refer to a specific person and the statement applies to all people who are 18 years or older, we are stuck. Therefore we need a richer logic.

6.1 Predicates and Quantifiers

Predicate logic is an extension of propositional logic. It adds the concept of *predicates* and *quantifiers* to better capture the meaning of statements that cannot be adequately expressed in propositional logic.

Predicates

Definition 6.1 (Predicate) A predicate $P : \mathcal{A}^n \to \mathbb{B}$ is a function that maps one or more variables from a domain $\mathcal{A}$ to a boolean value.

We denote predicates with *capital Roman letters* such as P, Q, and R.

Example. We want to translate "x is smaller than 5." into a formula φ in predicate logic. Since the truth value of this statement is simply dependent on the value of x, we define $\varphi \coloneqq S(x)$, where the predicate S returns true if $x < 5$ and false otherwise.

Example. When translating "If the sum of digits of n is divisible by 9 then n is divisible by 9" into a formula φ in predicate logic, we need to define two predicates:

- $P(n)$ is true if the sum of digits of n is divisible by 9, and
- $Q(n)$ is true when n is divisible by 9.

The sentence can then be modeled as $\varphi \coloneqq P(n) \to Q(n)$.

Quantifiers

Quantifiers are used to express the extent to which a predicate is true over a range of elements. Using quantifiers to create such propositions is called quantification. We distinguish between two types of quantification: the *universal quantification* and the *existential quantification.*

Universal Quantification

We can express statements that assert that *a property is true for all possible values of a variable in a particular domain* using universal quantification. The notation $\forall x P(x)$ denotes the universal quantification of $P(x)$. $\forall x P(x)$ is read as "for all x P(x)". *The formula $\forall x P(x)$ evaluates to true if $P(x)$ is true for all values of x in the specified domain.* It is very important to explicitly specify the domain when using universal quantification since the domain decides the possible values of x. Without the domain, the universal quantification has no meaning.

Example. We want to translate the statement: "For any natural number n, it holds that $n \cdot n$ is larger or equal to $n + n$" into a formula φ in predicate logic. We need to define the predicate $L(n) \coloneqq n \cdot n \geq n + n$, the domain $\mathcal{A} = \mathbb{N}$, and finally: $\varphi = \forall x L(x)$.

Existential Quantification

We can express statements that assert that *there is an element with a certain property* by existential quantification. The notation $\exists x P(x)$ denotes the existential quantification of $P(x)$, i.e., $\exists x P(x)$ is true if and only if $P(x)$ is true for at least one value of x in the domain. $\exists x P(x)$ is read as "There is at least one such x such that P(x)" and denotes the statement. *The formula $\exists x P(x)$ is true if there exists an element x in the domain such that P(x)."*

Example. We want to translate the sentence: "There exists a natural number n, such that n is larger than 0 and smaller than 1" into a formula φ in predicate

logic. We define two predicates:

- $G(n) \coloneqq n > 0$, and
- $L(n) \coloneqq n < 1$

in the domain $\mathcal{A} = \mathbb{N}$. We can then express the formula as $\varphi = \exists n(G(n) \wedge L(n))$.

Example. We want to translate the sentence: "For any natural number n, there exists a natural number m, such that $m > n$" into a formula φ in predicate logic. We define the predicate $L(n, m) \coloneqq m > n$ in the domain $\mathcal{A} = \mathbb{N}$. The statement then translates to $\varphi = \forall n \exists m L(n, m)$.

Example 1

Model the following sentence in predicate logic:

"Every person which is 18 years or older is eligible to vote."

Solution. Using the domain $x \in People$ and the predicates $P(x) \coloneqq$"x is 18 years", $R(x) \coloneqq$"x is older than 18 years" and $Q(x) \coloneqq$"x is eligible to vote" we get:

$$\forall x((P(x) \vee R(x)) \leftrightarrow Q(x)).$$

Example 2

Model the following sentence in predicate logic:

"Every lecturer is older than some student."

Solution. We define the following predicates for $x, y \in People$:

$$L(x) \coloneqq \text{“}x \text{ is a lecturer”}$$
$$S(x) \coloneqq \text{“}x \text{ is a student”}$$
$$O(x, y) \coloneqq \text{“}x \text{ is older than } y\text{”}$$

Using these predicates, we can model the sentence as follows:

$$\forall x(L(x) \rightarrow \exists y(S(y) \wedge O(x, y)))$$

Expressivity of Predicate Logic. We want to compare the expressivity of propositional logic to that of predicate logic via an example. **Example.** We want to model the following sentences using propositional logic and check whether the sequent can be proven:

"*Every child has a mother. Maria is a child. Therefore, Maria has a mother.*"

Solution. We define the following atomic propositions:

$$p := \text{“Every child has a mother.”}$$
$$q := \text{“Maria is a child.”}$$
$$r := \text{“Maria has a mother.”}$$

Using propositional logic results in the following sequent:

$$p, q \vdash r$$

The sequent can easily be disproven by the following counterexample: $\mathfrak{M} := \{p = \top, q = \top, r = \bot\}$.

Example. We formalize the same reasoning from above using predicate logic.

Solution. We define the following predicates:

$$C(x) := \text{“}x \text{ is a child”}$$
$$M(x) := \text{“}x \text{ has a mother”}$$

Using the domain $\mathcal{A} = People$, we can model the sequent as follows:

$$\forall x(C(x) \leftrightarrow M(x)), \quad C(maria) \vdash M(maria)$$

Using predicate logic, the meaning of the statements is preserved such that reasoning is possible. In the next chapter, we will extend the natural deduction calculus to predicate logic such that we are able to prove such sequents.

Example 3

Model the following sentence in predicate logic:

”Niki and Ben have the same maternal grandmother.“

Solution. We use the binary predicate $M(x, y) :=$“x *is the mother of* y.” with $\mathcal{A} = People$ and get the following formula:

$$\forall x \forall y \forall u \forall v(M(x, y) \land M(y, Niki) \land M(u, v) \land M(v, ben) \rightarrow x = u).$$

The formula states that, if y and v are Nikis and Bens mothers, respectively, and x and u are their mothers (i.e. Bens and Nikis maternal grandmothers, respectively), then x and u are the same person. Note, that we use the infix notation for the equality predicate instead of the prefix notation. Whenever it feels more natural you can use the infix notation. You can always use the equality predicate without defining it explicitly.

Functions

Definition 6.2 (Function) A *function* is an expression $f : \mathcal{A}^n \rightarrow \mathcal{A}$ that maps one or more variables in a specific domain $\mathcal{A}$ to a value in that domain. We denote

functions with lowercase capital Roman letters such as f, g, and h.

Example. We can express the statement from Example 3 as the formula φ using the function symbol $m(x)$ that returns the mother of x, i.e.

$$\begin{aligned} m : \text{People} &\to \text{People} \\ x &\mapsto \text{Mother of x.} \end{aligned}$$

The formula $\varphi = m(m(\text{niki})) = m(m(\text{ben}))$ models the statement.

6.2 Syntax of Predicate Logic

In this section, we define the syntactic rules that define well-formed formulas in predicate logic.

First, note that in predicate logic, we have two types of *sorts*. First, we have *terms* that talk about objects. Terms include individual objects (e..g, *ben*, *niki*), variables since they represent objects (e.g., x, y), and function symbols since they refer to objects (e.g., $m(x)$).

Second, we have *formulas* that have a truth value. For example, $P(x, f(y))$ is a formula and x, y, and $f(x)$ are terms.

$$\overbrace{\underbrace{P}_{\text{Predicate}}\left(\underbrace{x}_{\text{Term}}, \underbrace{f(y)}_{\text{Term}}\right)}^{\text{Formula}}$$

To define the syntax of predicate logic, we use the following notation:

- $\mathcal{V}$: Defines the set of variable symbols, e.g., x, y, z.
- $\mathcal{F}$: Defines the set of function symbols, e.g., f, g, h.
- $\mathcal{P}$: Defines the set of predicate symbols, e.g., P, Q, R.

Each predicate symbol and each function symbol comes with an arity, the number of arguments it expects. *Constants* are functions that dont take any arguments, i.e., nullary functions.

Terms

Definition 6.3 (Terms) We define *terms* as the following:

- Any variable is a term.
- If $c \in \mathcal{F}$ is a nullary function, then c is a term.
- If $t_1, t_2, ..., t_n$ are terms and $f \in \mathcal{F}$ has arity $n > 0$, then $f(t_1, t_2, ..., t_n)$ is a term.

Note that the first building blocks of terms are constants (nullary functions) and variables. More complex terms are built from function symbols using as many previously built terms as required by such function symbols.

Formulas

Definition 6.4 (Formula) We define *formulas* as the following:

- If $P \in \mathcal{P}$ is a predicate with arity $n > 0$ and t_1, t_2, ..., t_n are terms over $\mathcal{F}$, then $P(t_1, t_2, ..., t_n)$ is a formula.
- If φ is a formula, then $\neg\varphi$ is a formula.
- If φ and ψ are formulas, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, and $(\varphi \rightarrow \psi)$ are formulas.
- If φ is a formula and x is a variable, then $(\forall x\ \varphi)$ and $(\exists x\ \varphi)$ are formulas.

Note, that arguments, that are given to a predicate, are always terms.

Binding Priorities

We add to the binding priorities that we have defined for propositional logic the convention that $\forall x$ and $\exists x$ are binding as strong as $\neg$. Therefore, the precedence rules can be given by:

1. $\forall$, $\exists$, $\neg$ bind most tightly;
2. then $\wedge$;
3. then $\vee$;
4. then $\rightarrow$ which is right-associative.

Parse Tree

The parse tree is constructed in the same way as for formulas in propositional logic, but with two additional sorts of nodes:

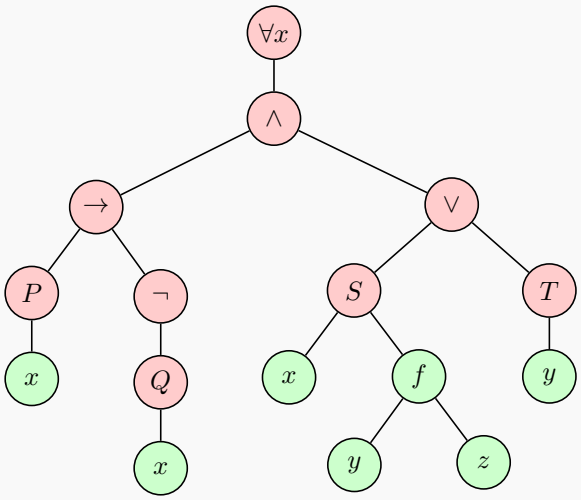
- Nodes for the quantifiers $\forall x$ and $\exists x$ have one subtree.
- Predicates of the form $P(t_1, t_2, \ldots, t_n)$ have the symbol P as a node with n subtrees, namely the parse trees of the terms $t_1, t_2, \ldots, t_n$.

Example 4

Draw the syntax tree to the following formula:

$$\forall x \Big(\big(P(x) \to \neg Q(x)\big) \wedge \big(S(x, f(y, z)) \vee T(y)\big) \Big).$$

Solution.



Red nodes are formulas, green nodes are terms.

6.3 Free and Bound Variables

With the introduction of the quantifiers $\forall$ and $\exists$, we also have to define the scope of quantifiers, i.e. whether a variable is quantified or whether it is free. We call quantified variables *bound*.

Definition 6.5 (Scope of a Quantifier.) For $\forall x\varphi$, the formula φ is the scope of $\forall x$. Likewise, for $\exists x\varphi$, the formula φ is the scope of $\exists x$.

In terms of parse trees, the scope of a quantifier is its subtree.

Definition 6.6 (Free and Bound Variables.) Let φ be a formula in predicate logic. An occurrence of x in φ is called *free*, if x is not in the scope of a $\forall x$ or $\exists x$ quantifier. Otherwise, that occurrence of x is called bound.

6.3.1 Substitution

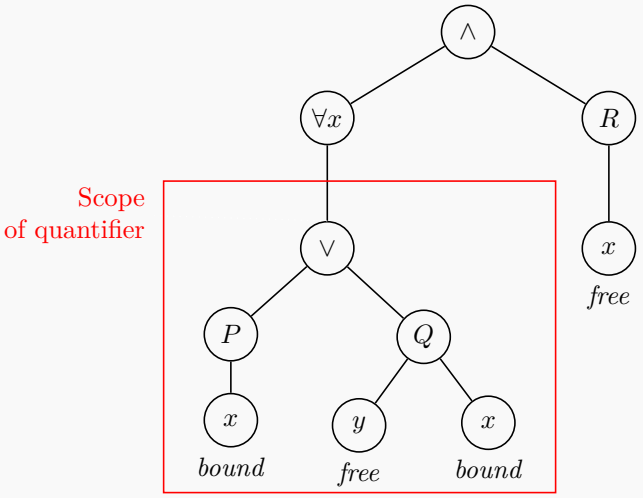
Variables are placeholders and can be *replaced* with more concrete information.

Example 5

Construct a parse tree for the following formula φ, determine the scope of its quantifiers and which occurrences of the variables are free and which are bound:

$$\varphi := \forall x\ (P(x) \vee Q(y,x)) \wedge R(x)$$

Solution.
We draw the syntax tree and annotate the free and bound variables.



The scope of $\forall x$ is $(P(x) \vee Q(y,x))$. Thus, the first two occurrences of x are bound to $\forall x$. Starting for the y leave node, the only quantifier we run into is $\forall x$ but that x has nothing to do with y. So y is free in this formula. The third occurrence of x is also free.

Definition 6.7 (Substitution) Given a variable x, a term t and a formula φ we define $\varphi[t/x]$ to be the formula obtained by replacing *each free occurrence* of variable x in φ with t.

$$\varphi[\underbrace{t}_{\text{Term}} / \underbrace{x}_{\text{Variable}}]$$

Therefore, if all occurrences of x are bound in φ, none of them gets substituted by t. Furthermore, it is *not allowed* to perform substitutions such that *a variable gets* captured *by a quantifier*: meaning that the variable was free before, and by carrying out the substitution it gets bound by a quantifier.

Example. We want to substitute x with $f(z)$, i.e. we want to compute $\varphi[f(z)/x]$

for the following formula:

$$\varphi \coloneqq \forall y\big(P(x) \land Q(y)\big) \lor \big(R(y) \land Q(x)\big).$$

All occurrences of x are free and can be replayed by $f(z)$:

$$\varphi\left[f(z)/x\right] = \forall y\ \big(P(f(z)) \land Q(y)\big) \lor \big(R(y) \land Q(f(z))\big)$$

Example. When computing $\psi[f(z)/x]$ for the following formula:

$$\psi = \forall x\big(P(x) \land Q(y)\big) \lor \big(R(y) \land Q(x)\big),$$

we can only substitute x in $Q(x)$, as the other occurrence x is bound to the $\forall x$ quantifier. The resulting formula is:

$$\psi\left[f(z)/x\right] = \forall x\ \big(P(x) \land Q(y)\big) \lor \big(R(y) \land Q(f(z))\big)$$

Example. When computing $\varphi[f(y)/x]$ for the following formula:

$$\varphi \coloneqq \forall y\big(P(x) \land Q(y) \lor \big(R(y) \land Q(x)\big)\big),$$

we are not allowed to replace x with $f(y)$, since x is free before the substitution and the term $f(y)$ contains a y which is in the scope of the $\forall y$ quantifier. Therefore, the variable would be captured which is not allowed. The resulting formula is:

$$\varphi\left[f(y)/x\right] \coloneqq \forall y\big(P(x) \land Q(y) \lor \big(R(y) \land Q(x)\big)\big)$$

6.4 Semantics of Predicate Logic

We will extend the notion of *models* that we have discussed for propositional logic to predicate logic. In propositional logic, a model defines an assignment of truth values to variables such that the formula evaluates to true or to false.

A model in predicate logic differs from a model in propositional logic in the treatment of predicates and functions.

6.4.1 Models

A model in predicate logic needs to define a concrete meaning to all predicate and function symbols involved. For example, the predicate P is defined in the model to be the relation *"greater than"* on the set of real numbers.

Definition 6.8 (Model) A model $\mathcal{M}$ consists of the following set of data:

- *A non-empty set $\mathcal{A}$, the universe/domain of concrete values*;
- *for each nullary function symbol $f \in \mathcal{F}$, a concrete element $f^{\mathcal{M}} \in \mathcal{A}$*;
- *for each nullary predicate symbol $P \in \mathbb{P}$, a truth value*;

- *for each function symbol* $f \in \mathcal{F}$ *with arity* $n > 0$*, a concrete function* $f^{\mathcal{M}} : \mathcal{A}^n \to \mathcal{A}$;
- *for each predicate symbol* $P \in \mathbb{P}$ *with arity* $n > 0$ *: subset* $P^{\mathcal{M}} \subseteq \mathcal{A}^n$.
- *for any free variable* `var`: a lookup-table $l : \texttt{var} \to \mathcal{A}$.

To denote a *concrete* instance of a function f or a predicate P in a model $\mathcal{M}$, we use the notation $f^{\mathcal{M}}$ and $P^{\mathcal{M}}$. We often define $P^{\mathcal{M}}$ as tuples which make P true and use function tables to define $f^{\mathcal{M}}$.

Example. A model $\mathcal{M}$ for the following formula:

$$\varphi := \forall x \exists y P(x, y),$$

needs to consist of a domain $\mathcal{A}$ and a concrete predicate instance $P^{\mathcal{M}}$. One possible model for φ would be:

- $\mathcal{A} = \{a, b\}$
- $P^{\mathcal{M}} = \{(a, b), (b, a)\}$ (meaning $P(a, b) = true$, $P(b, a) = true$, for all other cases, P evaluates to $false$)

6.4.2 Evaluating a Formula under a Model

Given a model $\mathcal{M}$, we define the satisfaction relation $\mathcal{M} \vDash \varphi$ for formulas in predicate logic by structural induction:

- P : If φ is of the form $P(t_1, t_2, \ldots, t_n)$, then we interpret the terms $t_1, t_2, \ldots, t_n$ in our set $\mathcal{A}$ by replacing all variables with their values according to the look-up table l and interpret any function symbols $f \in \mathcal{F}$ by $f^{\mathcal{M}}$. In this way we compute concrete values $a_1, a_2, ..., a_n$ of $\mathcal{A}$ for each of these terms. Now $\mathcal{M} \vDash P(t_1, t_2, \ldots, t_n)$ holds if and only if $(a_1, a_2, ..., a_n)$ is in the set $P^{\mathcal{M}}$.
- $\forall x$: The relation $\mathcal{M} \vDash \forall x \psi$ holds if and only if $\mathcal{M} \vDash \psi[x \leftarrow a]$ holds *for all* $a \in \mathcal{A}$.
- $\exists x$: Dually, $\mathcal{M} \vDash \exists x \psi$ holds if and only if $\mathcal{M} \vDash \psi[x \leftarrow a]$ holds *for some* $a \in \mathcal{A}$.

Example. We want to evaluate $\varphi := \forall x \exists y P(x, y)$ under the model $\mathcal{M} : \ \mathcal{A} = \{a, b\}, \ P^{\mathcal{M}} = \{(a, b), (b, a)\}$. We need to show that for any possible value for x, there exists a value for y, such that P evaluates to true. Since $P(a, b)$ and $P(b, a)$ are both true, this is the case and it holds that $\mathcal{M} \vDash \varphi$.

Example. We want to evaluate $\psi = \exists x \forall y \ P(x, y)$ under the model $\mathcal{M} : \ \mathcal{A} = \{a, b\}, \ P^{\mathcal{M}} = \{(a, b), (b, a)\}$. We need to show that there is a value for x, such that for all possible values for y, P evaluates to true. We perform the following substitutions:

- x and y substituted with a: $P(a, a) = \bot$. Therefore, we try the next substitution for x.
- x with b and y with a: $P(b, a) = \top$
- x with b and y with b: $P(b, b) = \bot$

Therefore, there is no such x for which P evaluates to true under all possible values for y. Therefore, $M \not\models \psi$.

Example. Given the formula $\psi = \exists x \forall y\ P(x, y)$ and a model $\mathcal{M}:\ A = \mathbb{N},\ P^{\mathcal{M}} = x \leq y \mid P^M = \{(1,1), (1,2) \dots (2,2), \dots\}$. We want to check whether $\mathcal{M} \vDash \psi$? Let us substitute x with 1. We need to show that P evaluates to true for all values of y.

- y substituted with 1: $P(1,1) = \top$
- y substituted with 2: $P(1,2) = \top$
- $\dots$
- y substituted with n: $P(1,n) = \top$ for any $n > 1$

Therefore, we can conclude that $\mathcal{M} \vDash \psi$.

List of Definitions