Predicate Logic

Michael Franke

Formulas of predicate logic; predicate letters, variables & individual constants; domain of quantification; quantifier scope and binding; atomic sentences; predicate-logical meaning of natural language sentences; semantics of predicate logic; model, domain and interpretation function; assignment and valuation functions; validity; predicate logic with identity

1 *Motivation*

We can think of propositional logic as a system that formalizes the meaning of important functional terms, namely the sentential connectives (negation, conjunction, disjunction, implication). This carries a long way towards capturing what sound logical inference is. But it also fails to capture crucial patterns of inference. For example, if we know that

Alex is as tall as Bo

we also know that

Bo is as tall as Alex

in virtue of our knowledge of what the predicate "being as tall as" means. Propositional logic cannot capture this. It can model the first sentences as p (= "Alex is as tall as Bo") and the second as q (= "Bo is as tall as Alex"), but since we cannot look inside the structure of a simple proposition, there is no way in which we can say that p, based on its internal form and relatedness to the internal form of q, must necessarily entail q.[1]

[1] We can, of course, make the additional premise by writing down that $p \rightarrow q$, but that clearly does not explain the general relationship.

Predicate logic (PredLog) is an extension of PropLog which adds two things. Firstly, PredLog models the internal structure of atomic propositions in terms of *predicates* and *individual constants*. For example, we could have a (two-place) predicate T meaning ("being as tall as") and two symbols, so-called individual constants, a and b which represent Alex and Bo respectively. We can then translate the sentence "Alex is as tall as Bo" into the formula Tab, which is a minimal truth-evaluable unit, but does have internal structure. Similarly, the sentence "Bo is as tall as Alex" would translate into Tba.

Secondly, PredLog allows to capture *quantification*, which is extremely important in order to capture general rules, generalizations and key aspects of our semantic and world knowledge.[2] Imagine that you have a reasoning machine (computer, robot, friend ...) able to compute logical inferences in predicate logic. Even though we have not introduced any of the formal machinery (syntax, semantics, definition of validity, deduction system) necessary to make such formal reasoning precise, imagine you give this machine

[2] Semantic knowledge is what we know about the meaning of words and expressions. For example, semantic knowledge tells us that "being taller than" is a transitive relation, and that "being as tall as" is an equivalence relation. World knowledge is what we know about the world. For example, we know that Berlin is the capital of Germany.

the information Tab. You want it to represent “Alex is as tall as Bo,” but the machine only has the string of symbols to work with. Would that machine be able to conclude that Tba? No, it wouldn’t because it doesn’t know that you want the symbol T to mean “being as tall as” and not “being taller than” or anything else. But, using predicate logic, you *can* tell the system about the fundamental structural properties of the relation “being as tall as,” such as that it is reflexive. In other words, you can express that “being as tall as” is a symmetric predict directly in PREDLOG with the formula:

$$\forall x \forall y (Txy \to Tyx)$$

which can be read as “for all objects x and y, if x stands in relation T to y, then so does y to x.” This formulas uses the quantifier $\forall$ to express a generalization: something that holds of any pair of objects. Generalizations of this kind are essential for human reasoning and PREDLOG captures the most basic aspects of quantification in a system of logical reasoning. To be clear, the inference schema:

$$Tab, \forall x \forall y (Txy \to Tyx) / Tba$$

is logically valid in PREDLOG, but the schema:

$$Tab/Tba$$

is not.

2 The language of predicate logic

2.1 Basic ingredients of predicate-logical formulas

The formulas of PREDLOG consist:

- individual constants $a, b, c, \ldots, v$
- predicate letters $A, B, C, D \ldots$
- variables w, x, y, z
- parentheses ( )
- sentential connectives $\neg, \wedge, \vee, \to$ $, \leftrightarrow$
- quantifiers $\exists, \forall$

Individual constants are denoted by lower-case Roman letters $(a, b, c, \ldots, v)$ up to v.[3] Individual constants are like proper names: they refer to exactly one individual. For example, the individual constant a may be interpreted as referring to Alex and b as referring to Bo. Individuals in the sense of predicate logic need not be humans or animals. An individual is any kind of entity that can have properties or stand in some kind of relation to any other property. For example, constant m may denote a particular copy of *Moby Dick*.

[3] If need be, we can also use indices like a_1, a_2 etc. This also holds for variables and predicate letters.

Predicate letters are denoted with upper-case Roman letters $(A, B, C, D \ldots)$. Predicate letters will be used to denote properties or relations. Each predicate letter has a unique *arity*. The arity of a predicate letter is always an integer bigger than zero. It gives the number of elements that the predicate letter expects as an argument. A predicate letter with arity one is also called *unary* predicate letter and will be interpreted to refer to a property. For example, if B is a unary predicate letter denoting the property "x is a book", then the expression Bm can be interpreted as expressing that *Moby Dick* is a book. A predicate letter with arity bigger than one will be interpreted to denote a relation. For example, if the predicate letter L has arity two, it may stand for a two-place relations such as "x loves y". Consequently, we expect L to have two arguments, so that Lab, Lam or Lmb would be well-formed expressions (no matter whether true or meaningful), while $Labm$, $Laaaa$ or Lb would not be.

Variables are denoted by lower-case Roman letters (w, x, y, z), starting from w. Variables are only interpretable in the *scope* of a quantifier, an important technical concept we will introduce later. As a first intuitive guide, think of variables as similar to pronouns[4] which are used to refer to an unnamed individual introduced by a quantifying expression like in these examples:

[4] A proper logical treatment of pronouns in natural language requires more sophisticated logical systems, like *dynamic logics* or *discourse representation theory*.

For every boy it holds that *he* ... [*he* = some boy]
There is a boy for which it holds that *he* ... [*he* = some boy]

To build formulas, PREDLOG also uses *parentheses* and exactly the same *sentential connectives* as PROPLOG does.

Quantifiers are special functional elements of the language of PREDLOG. The quantifier $\exists$ is the *existential quantifier*. It is read as "there is" or "there exists." For example, the formula $\exists x(Bx \wedge Ix)$ would be read as "there is an x such that x has property B and I." It would mean that there is an individual which has the property denoted by B (e.g., it is a book) and the property denoted by I (e.g., it is interesting). In short, this formula would express that there is at least one interesting book. The quantifier $\forall$ is the *universal quantifier*. It is read as "for all," "all" or "every." For example, the formula $\forall x(Bx \rightarrow Ix)$ would be read as "for all x it holds that if x has property B, then it also has property I." This would express that all books are interesting.

2.2 *Formulas*

The language $\mathfrak{L}$ of PREDLOG is the set of all *formulas* which are recursively defined as follows:

(i) If A is an n-ary predicate letter and if $t_1, \ldots, t_n$ are individual constants or variables, then $At_1 \ldots t_n$ is a formula.

(ii) If φ is a formula, then so is $\neg\varphi$.

(iii) If φ and ψ are formulas, so are:[5]

[5] We allow ourselves to omit the outermost pair of parentheses as in PROPLOG.

a. $(\varphi \wedge \psi)$ b. $(\varphi \vee \psi)$ c. $(\varphi \to \psi)$ d. $(\varphi \leftrightarrow \psi)$

(iv) If φ is a formula and if x is a variable, then these are formulas:
a. $\forall x\, \varphi$ [*universal statement*] b. $\exists x\, \varphi$ [*existential statement*]

(v) Anything that cannot be constructed by (i)–(iv) is not a formula.

Here are examples of formulas of PREDLOG, together with intuitive paraphrases based on the interpretation that a is Alex, b is Bo, m is the book *Moby Dick*, Lxy means "x likes y," Bx means "x is a book" and Oxy means that "x owns y."

Lam	Alex likes Moby Dick.
$Lab \wedge Lba$	Alex likes Bo and Bo likes Alex.
$\neg Oba$	Bo does not own Alex.
$\exists x(Bx \wedge Oax)$	Alex owns a book.
$\forall x\,((Bx \wedge Obx) \to Lax)$	Alex likes every book Bo owns.
$\neg\exists x\,(Bx \wedge Oax)$	Alex does not own any books.
$\forall x\,(Bx \to \neg Oax)$	Alex does not own any books.

2.3 *Syntactic trees*

The recursive definition for formulas of PREDLOG gives an internal structure to each formula which we can represent using *syntactic trees*, just like for PROPLOG. For PREDLOG the syntactic structure of a formula is particularly important for the important concept of the *scope of a quantifier* and the crucial notion of *variable binding* (see below). The syntactic structure, and with it the scope of a quantifier, depends on proper use of parentheses.

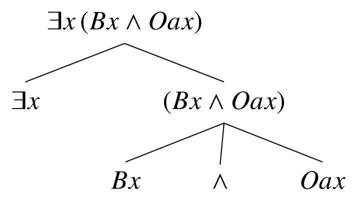
To illustrate, assume that we want to explicate the logical structure of the sentence:

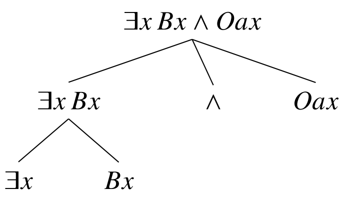
Alex owns a book.

Here are two candidate formulas, of which the left one is correct, the right one incorrect:

$$\exists x\,(Bx \wedge Oax) \qquad\qquad \exists x\, Bx \wedge Oax$$

A paraphrase of the second (incorrect) formula is, roughly: "There is a book and Alex owns them, him, her, it." The point is that in the second formula the x might but need not refer to the book, because it is not part of the formula "dominated by $\exists$," so to speak. This shows in the different syntactic trees.





2.4 *Terminology*

Just like in PropLog, formulas of PredLog can be named by their main operator. For example, the following formulas can be called *conjunctions*:[6]

$$Oam \land Lam \qquad \exists x\,(Bx \land Oax) \land \exists x\,(Bx \land Lax)$$

[6] The second example here expresses the idea that Alex owns a book, and that Alex likes a book, where these could be the same or different books.

A formula whose main operator is a universal quantifier can be called a *universal formula* or a *universal statement*; a formula whose main operator is an existential quantifier can be called an *existential formula* or an *existential statement*. For example, the formula $\exists x\,(Bx \land Oax)$ from above is an existential statement, but the formula $\exists x\,Bx \land Oax$ is a conjunction (as evidenced by the syntactic trees given above).

If A is an n-ary predicate letter and if $t_1, \ldots, t_n$ are individual constants, then $At_1 \ldots t_n$ is an ATOMIC SENTENCE. Atomic sentences are minimal truth-evaluable units of the language of PredLog, akin to the proposition letters of PropLog.

3 *Quantifier scope & binding*

Not every well-formed formula of PredLog is interpretable. Consider the formula Lax. We might paraphrase this as "Alex likes them, us, him, her or it." Without knowing what x refers to, this formula —though a formula of PredLog— is not interpretable. We therefore introduce terminology to speak about which occurrences of variables are interpretable, which are not, and how a variable that is interpretable is to be interpreted. The relevant technical terms are *scope*, as well as *bound* and *free* occurrences of a variable.

If $\forall x\,\psi$ is a subformula of φ, then ψ is the *scope* of this occurrence of the quantifier $\forall x$ in φ. The same holds for $\exists x$. An occurrence of a variable x in a formula φ (in a place where also an individual constant could appear, so not the x in "$\forall x$" and "$\exists x$"), is *free* in φ if x is not in the scope of a quantifier $\forall x$ or $\exists x$. If $\forall x\,\psi$ (or $\exists x\,\psi$) is a subformula of φ and if an occurrence of x is free in ψ, then this occurrence of x is *bound* by the quantifier $\forall x$ (or $\exists x$). A formula of predicate logic without any free occurrences of variables is called a *sentence*.

Here are examples:[7]

Px	x is free
$Px \land \forall x\,Qx$	the first occurrence of x is free, the second bound
$\exists x(\,Px \land Qx)$	both occurrences of x are existentially bound
$\exists x\,Px \land \forall x\,Qx$	first occur. existentially bound, second universally bound
$\exists x(\,Px \land \forall\, xQx)$	first occur. existentially bound, second universally bound

[7] The last formula is well-formed and interpretable, but not very cooperative for an interpreter. In practice, we would rather like to write $\exists x(\,Px \land \forall y\,Qy)$

4 *Domain of quantification*

Even if all variables in a formula are bound, in order to be able to interpret — even if only intuitively— what a formula of PredLog could mean, we need information about the *domain of quantification* D. Take the formula $\forall x\,(Lxa)$ with the interpretation of a and Lxy as before. We might take this to mean that everybody likes Alex, or that everything on earth (including the book *Moby Dick*) likes Alex. So, when we write down a formula with quantifiers in PredLog, it will only be interpretable if we specify which individuals the quantification should range over. We call this the *domain of quantification* D. Remember that you must always specify the domain of quantification D in translation exercises or other applications where your formulas are supposed to be meaningfully interpretable.

5 *Translations from natural langauge to* PredLog

Just like PropLog, PredLog is useful for uncovering the logical structure of sentences. Unlike PropLog, PredLog can lay bare the internal structure of atomic propositions and aspects of quantification.

Suppose we want to translate this sentence to predicate logic:

Alex likes Bo but if Bo likes Alex, Bo likes everybody.

A formula that captures the logical structure of this sentence is:

$$Lab \land (Lba \rightarrow \forall x\, Lbx)$$

Such a translation is only complete, strictly speaking, when we also explicitly state the *translation key*, which defines what each individual constant and predicate letter refers to, as well as the arity of each predicate letter. In the example at hand, the translation key would be:[8]

[8]Notice that the arity of the predicate L is fixed by the notation Lxy and that it is crucial for the translation key to specify exactly what a predicate like L means, i.e., is first argument the slot for the person doing or receiving the liking?

(i) a: Alex

(ii) b: Bo

(iii) Lxy: x likes y

Since the sentence to translate includes the word "everybody," the domain of quantification should be the set of all human beings for the above formula to be correct. If the domain of quantification also includes non-humans, we would have to adapt the formula like so:

$$Lab \land (Lba \rightarrow \forall x\,(Hx \rightarrow Lbx))$$

and also include the predicate letter H in the translation key like so:

(iv) Hx: x is a human being

Let us consider a few examples. The domain of quantification D is the set of all human beings.

a: Alex	Lxy: x likes y
b: Bo	Px: x is a pilot
Fx: x is friendly	Sxy: x and y are siblings

Here is a list with sentences and potential translations into PREDLOG.[9]

Alex is a pilot.	Pa
Bo is a friendly pilot.	$Fb \wedge Pb$
No pilot is friendly.	$\forall x\,(Px \to \neg Fx)$
	$\neg\exists x\,(Px \wedge Fx)$
Nobody likes pilots.	$\neg\exists x\,\exists y\,(Py \wedge Lxy)$
Bo has a friendly sibling.	$\exists x\,(Fx \wedge Sbx)$
Every pilot has a friendly sibling.	$\forall x\,(Px \to (\exists y\,(Fy \wedge Sxy)))$
	$\forall x\,\exists y\,(Px \to (Fy \wedge Sxy))$
	$\neg\exists x\,(Px \wedge \exists y\,(Fy \wedge Sxy))$
	$\neg\exists x\,\exists y\,(Px \wedge (Fy \wedge Sxy))$

[9] For some sentences, more than one translation is given. These alternatives are logically equivalent under the semantics of PREDLOG which we will introduce later.

Exercise 1. For each of the following strings, determine whether they are formulas of PredLog or not. Assume that P and Q are unary predicate letters, and that R is a binary predicate letter.

(i) $Px \to \exists\, x$

(ii) $\forall x(Px)$

(iii) $\forall xPx$

(iv) $(\forall xPx)$

(v) $Px \lor \exists\, xPx$

(vi) $\forall yPx \lor \exists\, xPx$

(vii) $\forall y(Rxy \lor \exists\, xPx)$

(viii) $\forall y(Rxy \lor \exists\, xPx)$

Exercise 2. For each of the following formulas of predicate logic, determine whether each occurrence of a variable is a free or bound occurrence. If it is a bound occurrence, determine which quantifier binds it.

(i) Px

(ii) $\exists x\, Lxj$

(iii) $\exists x\, Lxy$

(iv) $\exists x\, Px \land Lxj$

(v) $\exists x\,(Px \land Lxj)$

(vi) $\exists x\,(Px \land \forall x\, Lxj)$

Exercise 3. Translate the following sentences into the language of predicate logic. Preserve as much of the logical structure as possible and give the translation key and the domain of quantification (here: D : people).

(i) Everybody is friendly.

(ii) Everybody loves somebody.

(iii) Every pilot loves Bill.

(iv) If Mary is a pilot, someone loves her.

(v) Every pilot is unfriendly.

(vi) Some pilots are friendly.

(vii) No pilot is friendly.

(viii) Nobody loves anyone who is in love with a pilot.

6 *Semantics of predicate logic*

The semantics of propositional logic was formulated in terms of valuation functions, which mapped each formula to a truth value, true or false. We thought of these are possible worlds, possible states the world could be in. The semantics of PredLog is slightly more complicated because in order to say which formulas are true or false, we need to interpret not just atomic propositions, but their internal structure, so to speak; we need to interpret all terms (individual constants and variables) and the predicate letters. This is done in a so-called *model*, which can similarly be thought of as a possible world. Given a model, we can derive a valuation function. The model fixes the domain (which individuals are there) and the meaning of all predicates.

6.1 *Model, domain & interpretation function*

Let C be set of individual constants and $\mathcal{P}$ the set of predicate letters our language $\mathfrak{L}$ of PredLog. A *model* $M = \langle D, I \rangle$ for $\mathfrak{L}$ consists of:

a *domain* $D \neq \emptyset$, and   an *interpretation function* I.

The domain D is a set of entities.[10] The interpretation function I maps each element of C and $\mathcal{P}$ to a suitable relation on D:

[10] We will always assume that the domain has at least one element.

if $c \in C$, then $I(c) \in D$, and

if $P \in \mathcal{P}$ is an n-ary predicate letter, then $I(P) \subseteq D^n$.[11]

[11] D^n is the set of all n-tuples with elements from D.

The interpretation $I(c)$ of a constant c is the entity in the domain that c stands for (e.g., its name or identifier). The interpretation $I(P)$ of an n-ary predicate letter, is the set of all n-tuples which stand in the relation P to each other. For example, if R is the one-place predicate letter denoting the property "red," then $I(P)$ is the set of all red things (in the possible world we are modeling with the help of I). Of, if L is the two-place predicate letter denoting the relation "x likes y," then $I(L)$ is the set of all pairs of elements from the domain, e.g., $\langle d_1, d_2 \rangle$, such that the first likes the second (in the possible world we are modeling with the help of I).

6.2 *Assignment functions*

A model $M = \langle D, I \rangle$ interprets individual constants and predicate letters. But we also need a way to assign meaning to variables. This needs to take quantifier binding into account. For this purpose we can use assignment functions. Assignment functions and the interpretation function together allow us to interpret all terms (variables and individual constants).

An *assignment function* g for model $M = \langle D, I \rangle$ and language $\mathfrak{L}$ maps all variables in $\mathfrak{L}$ to elements in D. We write $g_{[x/d]}$ and read "g with x mapped to d", for the assignment function which is like g except that x is mapped to $d \in D$. We will use this latter construct, $g_{[x/d]}$ to define the semantics of existential and universal formulas (see below).

If t is a term of $\mathfrak{L}$, then $\llbracket t \rrbracket_{M,g}$ is the *term interpretation* relative to $M = \langle D, I \rangle$ and g:

$$\llbracket t \rrbracket_{M,g} = I(t) \text{ if } t \text{ is a constant, and} \qquad \llbracket t \rrbracket_{M,g} = g(t) \text{ otherwise.}$$

6.3 *Valuation functions*

We now can use a model $M = \langle D, I \rangle$ and an assignment function g to define a valuation function $V_{M,g}$, which uses M and g, to give a truth value for every formula of predicate logic. Here is the full definition, which we will go through systematically below.

$$\begin{array}{lll}
V_{M,g}(At_1 \dots t_n) = 1 & \text{iff} & \left\langle \llbracket t_1 \rrbracket_{M,g}, \dots, \llbracket t_1 \rrbracket_{M,g} \right\rangle \in I(A) \\
V_{M,g}(\neg\phi) = 1 & \text{iff} & V_{M,g}(\phi) = 0 \\
V_{M,g}(\phi \wedge \psi) = 1 & \text{iff} & V_{M,g}(\phi) = 1 \text{ and } V_{M,g}(\psi) = 1 \\
V_{M,g}(\phi \vee \psi) = 1 & \text{iff} & V_{M,g}(\phi) = 1 \text{ or } V_{M,g}(\psi) = 1 \\
V_{M,g}(\phi \rightarrow \psi) = 0 & \text{iff} & V_{M,g}(\phi) = 1 \text{ en } V_{M,g}(\psi) = 0 \\
V_{M,g}(\phi \leftrightarrow \psi) = 1 & \text{iff} & V_{M,g}(\phi) = V_{M,g}(\psi) \\
V_{M,g}(\forall x\, \phi) = 1 & \text{iff} & V_{M,g_{[x/d]}}(\phi) = 1 \text{ for all } d \in D \\
V_{M,g}(\exists x\, \phi) = 1 & \text{iff} & V_{M,g_{[x/d]}}(\phi) = 1 \text{ for at least one } d \in D
\end{array}$$

For the most basic kind of formula, built from an n-ary predicate letter A with n terms $t_1, \dots, t_n$ (where each t_i is either a variable or an individual constant), the formula $At_1 \dots t_n$ is true, given $M = \langle D, I \rangle$ and g, if the n-tuple of entities picked out by the interpretation of all terms, using I and g, is contained in the interpretation of A.

$$V_{M,g}(At_1 \dots t_n) = 1 \quad \text{iff} \quad \left\langle \llbracket t_1 \rrbracket_{M,g}, \dots, \llbracket t_n \rrbracket_{M,g} \right\rangle \in I(A)$$

For sentential connectives, valuation functions work exactly like in PROPLOG.

$$\begin{array}{lll}
V_{M,g}(\neg\phi) = 1 & \text{iff} & V_{M,g}(\phi) = 0 \\
V_{M,g}(\phi \wedge \psi) = 1 & \text{iff} & V_{M,g}(\phi) = 1 \text{ and } V_{M,g}(\psi) = 1 \\
V_{M,g}(\phi \vee \psi) = 1 & \text{iff} & V_{M,g}(\phi) = 1 \text{ or } V_{M,g}(\psi) = 1 \\
V_{M,g}(\phi \rightarrow \psi) = 0 & \text{iff} & V_{M,g}(\phi) = 1 \text{ en } V_{M,g}(\psi) = 0 \\
V_{M,g}(\phi \leftrightarrow \psi) = 1 & \text{iff} & V_{M,g}(\phi) = V_{M,g}(\psi)
\end{array}$$

Finally, to assign a truth-value to existential or universal formulas, we use the construction $g_{[x/d]}$, which represents an assignment function which is exactly like g but with a special, possibly different interpretation for the variable x.

$$\begin{array}{lll}
V_{M,g}(\forall x\, \phi) = 1 & \text{iff} & V_{M,g_{[x/d]}}(\phi) = 1 \text{ for all } d \in D \\
V_{M,g}(\exists x\, \phi) = 1 & \text{iff} & V_{M,g_{[x/d]}}(\phi) = 1 \text{ for at least one } d \in D
\end{array}$$

7 *Truth in a model*

Assignment functions are just auxiliary constructs. At the end of the day, we are interested in whether a formula is true in a possible world, i.e., a model $M = \langle D, I\rangle$, no matter what assignment function is used.

If $V_{M,g}(\varphi) = 1$ for all g, we write $V_M(\varphi) = 1$ and say "φ is true in M." If $V_{M,g}(\varphi) = o$ for all g, we write $V_M(\varphi) = 0$ and say "φ is false in M."

Notice that any *sentence* of PredLog (i.e., any formula without free variables) is either true or false in any given model M. In other words, formulas without free occurrences of variables are interpretable just with respect to a model, and these formulas are —usually— what we care about.

Let's consider a simple example of a model and the way it makes certain sentences true or false. The model $M = \langle D, I\rangle$ is for a language with just one individual constant a, two one-place predicate letters, P and Q, as well as one two-place predicate letter R. The following lines are a complete specification of a model for this language, fixing a domain and an interpretation of all relevant ingredients:

$$\begin{aligned} D &= \{1, 2, 3, 4, 5\} \\ I(a) &= 3 \\ I(P) &= \{1, 2, 5\} \\ I(Q) &= \{2, 5\} \\ I(R) &= \{\langle 1, 2\rangle, \langle 2, 2\rangle, \langle 1, 4\rangle, \langle 1, 5\rangle\} \end{aligned}$$

For better visual grasp, small models (with at most two-place predicate letters) can often be represented economically as a graph. Figure 1 is an example for the model at hand. The two-place relation is represented in terms of arrows, small letters ornamenting the elements of the domain show the interpretation of individual constants and one-place predicate letters.

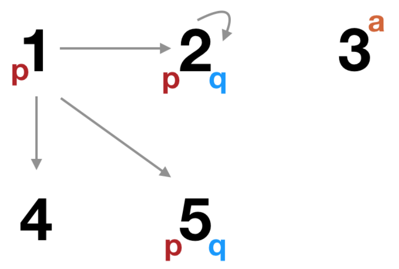


Figure 1: Example of a graphical representation of a simple model for predicate logic

A graphical representation like in Figure 1 greatly facilitates checking whether a model makes certain sentences true or false. For example, in order to decide whether a formula like $\exists x\,(Px \wedge Qx)$ is true or false in this model, we need to check whether there is an element that has both property P and property Q. Looking at the diagram in Figure 1, we see that this is true. Elements 2 and 5 both have both properties. To write this down in a formally correct way with reference to the model's mathematical structure and the definition of the semantics of PredLog, we would say that $V_M(\exists x\,(Px \wedge Qx)) = 1$ because, not matter which g we use, there exists an element in D, namely 2, such that $V_{M,g[x/d]}(Px \wedge Qx) = 1$. This is because $2 \in I(P)$ and $2 \in I(Q)$.

Take another example. What is the truth value that the model assigns to formula Pa? Well, that formula is false in the given model. The individual referred to by constant a is 3, but 3 does not have property P. In formal terms, we would then say that $V_M(Pa) = 0$, because, for any valuation function g, since $I(a) = 3$ and $3 \notin I(P)$, we have $V_{M,g}(P(a)) = 0$.

8 *Validity*

Validity is defined in the same way in PredLog as it was in PropLog.

If $\varphi_1, \ldots, \varphi_n$ and ψ are sentences of predicate logic, $\varphi_1, \ldots, \varphi_n/\psi$ is an *argument schema*. The argument schema $\varphi_1, \varphi_2, \ldots, \varphi_n/\psi$ is *valid* iff for all models M such that $V_M(\varphi_1) = V_M(\varphi_2) = \cdots = V_M(\varphi_n) = 1$ it also holds that $V_M(\psi) = 1$. If valid, we write $\varphi_1, \varphi_2, \ldots, \varphi_n \models \psi$.

In order to demonstrate that an argument schema is not valid, it suffices to give a single counterexample. In order to demonstrate that a given argument schema is valid, we need to give an informal proof, since we must reason over *all* models. The following gives examples for each.

8.1 *Demonstrating invalidity with a counterexample*

The argument schema $\exists x\,Px$ / $\forall x\,Px$ is not valid. Intuitively, just because there is a tall person, it does not mean that everybody is tall. A minimal counterexample for this case is a model M with domain $D = \{1, 2\}$ and interpretation function $I(P) = \{1\}$. It is clear that there is an element $d \in D$, namely 1, such that $V_{M,g[x/d]}(Px)$ is true, so that the premise $\exists x\,Px$ is true in M. But there is also an element $d \in D$, namely 2, such that $V_{M,g[x/d]}Px$ is false, so that the conclusion $\forall x\,Px$ is false in M.

8.2 *Demonstrating validity with an informal proof*

Claim 1. $\forall x\,Px \models \neg\exists x\,\neg Px$

Proof. We need to show that for all models M, if $V_M(\forall x\,Px) = 1$, then $V_M(\neg\exists x\,\neg Px) = 1$. So suppose that we have an arbitrary model $M = \langle D, I\rangle$, such that $V_M(\forall x\,Px) = 1$. This entails that for all $d \in D$, $d \in I(P)$. Suppose further, towards contradiction, using an indirect proof strategy, that for the same M we have $V_M(\neg\exists x\,\neg Px) = 0$. The latter means that $V_M(\exists x\,\neg Px) = 1$, which is the case when there is an element $d \in D$ such that $d \notin I(P)$. But that contradicts the inference we drew from our premise that for all $d \in D$, $d \in I(P)$. Consequently, there cannot be a model M which makes the premise true and the conclusion false. The argument schema must be valid. □

9 *Equivalences, tautologies and contradictions*

The notions of *logical equivalence*, *tautology* and *contradiction* for PREDLOG are all analogous to their counterparts in PROPLOG.

Two formulas φ and ψ of PREDLOG are logically equivalent iff for all models M we have $V_M(\varphi) = V_M(\psi)$. A formula φ of PREDLOG is a tautology iff for all models M we have $V_M(\varphi) = 1$. It is a contradiction iff for all models M we have $V_M(\varphi) = 0$.

In order to demonstrate that two formulas are logically equivalent, or that a formula is a tautology or contradiction, we can give informal proofs. For example, the previous section showed that $\forall x\, Px \models \neg\exists x\, \neg Px$. In fact, $\forall x\, Px$ and $\neg\exists x\, \neg Px$ are logically equivalent. Here is an informal proof for this:

Claim 2. $\forall x\, Px$ and $\neg\exists x\, \neg Px$ are logically equivalent.

Proof. Let M be a model in which $\neg\exists x\, \neg Px$ is true. This means that there is not a single element $d \in D$, such that $d \notin I(P)$. In other words, $I(P) = D$, i.e., all elements in the domain have property P. So M also makes $\forall x\, Px$ true. The reverse direction has already been shown when proving $\forall x\, Px \models \neg\exists x\, \neg Px$. □

Exercise 4. Use the model from the example in Figure 1 to check the truth value of the following formulas.

1. $\neg Qa$
2. $\forall x\,(Qx \to Px)$
3. $\neg\exists x\, Rxx$
4. $\exists x \exists y\,(Qx \wedge Py \wedge Rxy)$
5. $\forall x \exists y\, Rxy$
6. $\forall x \forall y\,(Rxy \to (Px \vee Qy))$

Exercise 5. Consider a model $M = \langle D, I\rangle$ for a language with two unary predicate letters P and Q and a two-place predicate letter R, given as follows.

$$\begin{aligned} D &= \{1,2,3,4,5\} \\ I(P) &= \{1,2,5\} \\ I(Q) &= \{2,5\} \\ I(R) &= \{\langle 1,2\rangle, \langle 2,2\rangle, \langle 1,4\rangle, \langle 1,5\rangle\} \end{aligned}$$

Evaluate for each of the following formulas whether they are true or false in the model. (Hint: It is highly recommended that you draw a graphical representation for yourself similar to Figure 1, even if that is not required to obtain points and/or a correct solution.)

1. $\exists x\,(Px \wedge Qx)$
2. $\forall x\,(Qx \to Px)$
3. $\neg\exists x\, Rxx$
4. $\exists x \exists y\,(Qx \wedge Py \wedge Rxy)$
5. $\forall x \exists y\, Rxy$
6. $\forall x \forall y\,(Rxy \to (Px \vee Qy))$

Exercise 6. Give informal proofs or counterexamples for these claims:

(i) $\forall x\,(Ax \vee Bx \vee Cx),\ \ \forall x\,(Ax \to Bx),\ \ \neg\exists Bx \models \exists x\, Cx$

(ii) $\exists x\, Ax,\ \exists x\, Bx \models \exists x\,(Ax \wedge Bx)$

(iii) $\neg\forall x\,(Ax \to Bx) \models \exists x\, Ax$

(iv) $\forall x \forall y \forall z\,((Rxy \wedge Ryz) \to Rxz),\ \ \exists x \exists y\,(x \neq y \wedge Rxy \wedge Ryx)\ \ /\ \ \exists x\, Rxx$

(v) $\forall x\,(Ax \vee Bx) \models \forall x\, Ax \vee \forall x\, Bx$

(vi) $\forall x\,(Ax \leftrightarrow Rxx),\ \exists x \forall y\,(Ryy \leftrightarrow x = y) \models \exists x\, Ax$

(vii) $\exists x \forall y\, Rxy,\ \forall x \forall y\,(Rxy \to Ryx) \models \forall x\, \exists y\, Rxy$

(viii) $\forall x\,(Ax \to Bx), \exists x\, Bx \models \neg\exists x\, Ax$

10 *Predicate logic with identity*

Suppose we want to translate the meaning of the following sentence into predicate logic:

Only Alex is a pilot.

We can write Pa (with the obvious translation key), expressing that Alex is a pilot. But we also want to express that nobody else is. We can express that nobody is a pilot: $\neq \forall x \neg Px$, but that's not actually true because Alex is. So writing $Pa \wedge \forall x \neg Px$ is a contradiction that does not express the meaning of "Only Alex is a pilot." Indeed, PredLog as we defined it so far is not expressive enough to capture the meaning of this sentence. We need to add something. We will add a special predicate: identity.

10.1 *Adding identity to* PredLog

We change clause (i) of the definition of formulas of predicate logic, given in Section 2.2, to also allow the use the special predicate two-place predicate symbol "=" for which we use infix notation.

(i) If A is an n-ary predicate letter and if $t_1, \dots, t_n$ are terms, then $At_1 \dots t_n$ is a formula. **If t_1 and t_2 are terms, then $t_1 = t_2$ and $t_1 \neq t_2$ are formulas.**

This is a special symbol also in the semantics, because it is interpreted in the same manner in every model. (Identity is the same concept in all possible worlds.) Concretely, we just add the following clause to the definition of the semantics or predicate logic:[12]

$$V_{M,g}(t_1 = t_1) = 1 \quad \text{iff} \quad [\![t_1]\!] = [\![t_2]\!]$$
$$V_{M,g}(t_1 \neq t_1) = 1 \quad \text{iff} \quad [\![t_1]\!] \neq [\![t_2]\!]$$

[12]Notice that there are three occurrences of the symbol "=" in the first line of this definition. The first occurrences refers to the two-place predicate letter of PredLog with identity. The latter two refer to our usual understanding of "=" as identity. Basically, this is a complicated way of saying that "=" is supposed to mean the same same thing in PredLog as what it means outside of it. Similarly, for the last line in the definition.

10.2 *Translations with* PredLog *with identity*

Using identity in the logical language, we can express the meaning of

Only Alex is a pilot.

in terms of the formula:

$$Pa \wedge \forall x (x \neq a \rightarrow \neg Px)$$

which can we read as: "Alex is a pilot, and everybody who is not Alex is not a pilot."[13]

Here are further examples for translations, making use of identity:[14]

(i) Alex is friendly but nobody else is. $\quad Fa \wedge \forall x (Fx \rightarrow x = a)$

(ii) At least two people are friendly. $\quad \exists x \exists y (Fx \wedge Fy \wedge x \neq y)$

[13]This is not the only paraphrase and formula we can give to this example. See the first example in the list that follows, for instance.

[14]Translation key as before.

(iii) Alex likes only pilots, except for Bo. $\forall x\,(Lax \rightarrow (Px \vee x = b))$

(iv) Exactly two people are pilots.
$$\exists x \exists y\,(x \neq y \wedge Px \wedge Py \wedge \forall z\,(Pz \rightarrow (z = x \vee z = y)))$$

(v) Bo likes at most two people.
$$\forall x \forall y\,(((Lbx \wedge Lby) \wedge x \neq y) \rightarrow \forall z\,(Lbz \rightarrow (z = x \vee z = y)))$$

Translations of quantified expressions like “at most two people,” as in the last example, can be difficult. Is this translation also adequate for the last example of “Bo likes at most two people?”

(v) Bo likes at most two people.
$$\forall x \forall y\,((Lbx \wedge Lby) \rightarrow \forall z\,(Lbz \rightarrow (z = x \vee z = y)))$$

No, because that one is false in a world in which Bo loves two people. Suppose Bo loves Alex and Charley, and that Alex, Bo and Charley are three different people. Let x and y refer to Alex. Then the antecedent $(Lbx \wedge Lby)$ becomes true. Let z refer to Charley. Bo loves Charley, but Charley is not identical to Alex (by assumption).

Exercise 7. Translate the following sentences into predicate logic with identity.

(i) Only Alex and Bo are friendly.

(ii) If Alex is friendly, nobody else is.

(iii) Alex is the only friendly pilot.

(iv) There is a pilot but it is not Alex.

(v) There is at most one pilot.

(vi) Alex likes nobody but Bo and Alex.