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;

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]

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 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

[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.

[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.

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:

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

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:

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

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 of a number of building blocks.

- individual constants $a, b, c, \ldots, v$
- predicate letters $A, B, C, D \ldots$
- variables w, x, y, z
- brackets ( )
- sentential connectives (like PropLog) $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- 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.[4]

Predicate letters are denoted with upper-case Roman letters ($A, B, C, D \ldots$). Predicate letters will be used to denote relations. Each predicate letter has a unique *arity*, i.e., the number of elements that the relevant relation requires. For example, the predicate letter L may stand for a two-place relations such

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

[4] 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 Michael's copy of *Moby Dick*.

as "x loves y". The arity of L would therefore be 2. We expect L to have two arguments, so that Lab, Lax or Lxy would be well-formed expressions (see below), while $Labc$ or La 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, a technical concept we will introduce later. As a first intuitive guide, think of variables as similar to pronouns which are used to refer to an unnamed individual introduced by a quantifying expression like in these examples:

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 *brackets* 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). 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." 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]
a. $(\varphi \wedge \psi)$ b. $(\varphi \vee \psi)$ c. $(\varphi \rightarrow \psi)$ 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.

[5] We allow ourselves to omit the outermost pair of brackets as in PropLog.

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 "x owns y."

Lam	Alex likes Moby Dick.
$Lab \wedge Lba$	Alex likes Bo and Bo likes Alex.
$\exists x(Bx \wedge Oax)$	Alex owns a book.
$\forall x\,((Bx \wedge Obx) \to Lax)$	Alex likes every book Bo owns.

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 *Domain of quantification*

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.

4 *Quantifier scope & binding*

Even with an explicit domain of quantification, not every 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* occurrence 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 (outside of a quantifier $\forall x$ or $\exists x$), 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 $\exists x$).

Here are examples:[6]

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

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

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 sentences to predicate logic:

Alex likes Bo but if Bo likes Alex, Bo likes everyone.

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:[7]

(i) a: Alex

(ii) b: Bo

(iii) Lxy: x likes y

[7] 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?

The domain of quantification should be the set of all human beings. If the domain of quantification should also include non-humans, we would have to adapt the formula:

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

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

(iv) Hx: x is a human being

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) $Px \rightarrow \exists x$

(ii) $\forall x(Px)$

(iii) $\forall xPx$

(iv) $(\forall xPx)$

(v) $Px \vee \exists xPx$

(vi) $\forall yPx \vee \exists xPx$

(vii) $\forall y(Rxy \vee \exists xPx)$

(viii) $\forall y(Rxy \vee \exists xPx)$

Exercise 2. 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.

Exercise 3. 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) $\exists x\, Px \wedge Lxj$

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

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