Models

The formula $(p \vee \neg q) \rightarrow (q \rightarrow p)$ is evaluated by computing a truth value (T or F) for it, based on a given valuation (assumed truth values for p and q). This activity is essentially the construction of one line in the truth table of $(p \vee \neg q) \rightarrow (q \rightarrow p)$. How can we evaluate formulas in predicate logic, e.g. $\forall x \exists y ((P(x) \vee \neg Q(y)) \rightarrow (Q(x) \rightarrow P(y)))$ which 'enriches' the formula of propositional logic above? Could we simply assume truth values for P(x), Q(y), Q(x) and P(y) and compute a truth value as before? Not quite, since we have to reflect the meaning of the quantifiers $\forall x$ and $\exists y$, their dependences and the actual parameters of P and Q – a formula $\forall x \exists y R(x, y)$ generally means something else other than $\exists y \forall x R(x, y)$; why? The problem is that variables are place holders for any, or some, unspecified concrete values. Such values can be of almost any kind: students, birds, numbers, data structures, programs and so on.

Thus, if we encounter a formula $\exists y \psi$, we try to find some instance of y (some concrete value) such that ψ holds for that particular instance of y. If this succeeds (i.e. there is such a value of y for which ψ holds), then $\exists y \psi$ evaluates to T; otherwise (i.e. there is no concrete value of y which realises ψ) it returns F. Dually, evaluating $\forall x \psi$ amounts to showing that ψ evaluates to T for all possible values of x; if this is successful, we know that $\forall x \psi$ evaluates to T; otherwise (i.e. there is some value of x such that ψ computes F) it returns F. Of course, such evaluations of formulas require a fixed universe of concrete values, the things we are, so to speak, talking about. Thus, the truth value of a formula in predicate logic depends on, and varies with, the actual choice of values and the meaning of the predicate and function symbols involved.

If variables can take on only finitely many values, we can write a program that evaluates formulas in a compositional way. If the root node of φ is $\wedge$, $\vee$, $\rightarrow$ or $\neg$, we can compute the truth value of φ by using the truth table of the respective logical connective and by computing the truth values of the subtree(s) of that root. If the root is a quantifier, we have sketched above how to proceed. This leaves us with the case of the root node being a predicate symbol P (in propositional logic this was an atom and we were done already). Such a predicate requires n arguments which have to be terms t1, t2,...,tn. Therefore, we need to be able to assign truth values to formulas of the form P(t1, t2,...,tn).

For formulas P(t1, t2,...,tn), there is more going on than in the case of propositional logic. For n = 2, the predicate P could stand for something like 'the number computed by t1 is less than, or equal to, the number computed by t2.' Therefore, we cannot just assign truth values to P directly without knowing the meaning of terms. We require a model of all function and predicate symbols involved. For example, terms could denote real numbers and P could denote the relation 'less than or equal to' on the set of real numbers.

Semantic entailment

In propositional logic, the semantic entailment $\varphi 1, \varphi 2,...,\varphi n \models \psi$ holds iff: whenever all $\varphi 1, \varphi 2,...,\varphi n$ evaluate to T, the formula ψ evaluates to T as well. How can we define such a notion for formulas in predicate logic.

Let Γ be a (possibly infinite) set of formulas in predicate logic and ψ a formula of predicate logic.

1. Semantic entailment $\Gamma \models \psi$ holds iff for all models M and look-up tables l, whenever $M\ l \varphi$ holds for all $\varphi \in \Gamma$, then $M\ l \psi$ holds as well.

2. Formula ψ is satisfiable iff there is some model M and some environment l such that $M\ l \psi$ holds.

3. Formula ψ is valid iff $M\ l \psi$ holds for all models M and environments l in which we can check ψ.

4. The set Γ is consistent or satisfiable iff there is a model M and a look-up table l such that $M\ l \varphi$ holds for all $\varphi \in \Gamma$.

In predicate logic, the symbol $\models$ is overloaded: it denotes model checks '$M \models \varphi$' and semantic entailment '$\varphi 1, \varphi 2,...,\varphi n \models \psi$.' Computationally, each of these notions means trouble. First, establishing $M \models \varphi$ will cause problems, if done on a machine, as soon as the universe of values A of M is infinite. In that case, checking the sentence $\forall x\ \psi$, where x is free in ψ, amounts to verifying $M\ [x \rightarrow a]\ \psi$ for infinitely many elements a. Second, and much more seriously, in trying to verify that $\varphi 1, \varphi 2,...,\varphi n \models \psi$ holds, we have to check things out for all possible models, all models which are equipped with the right structure (i.e. they have functions and predicates with the matching number of arguments). This task is impossible to perform mechanically. This should be contrasted to the situation in propositional logic, where the computation of the truth tables for the propositions involved was the basis for computing this relationship successfully.

However, we can sometimes reason that certain semantic entailments are valid. We do this by providing an argument that does not depend on the actual model at hand. Of course, this works only for a very limited number of cases. The most prominent ones are the quantifier equivalences which we already encountered in the section on natural deduction. Let us look at a couple of examples of semantic entailment.