Predicate Logic: Part 2

CS 4710 Course Notes

February 8, 2019

Interpretation Semantics

Now we define meaning for PL statements in a way that assures every statement can be assigned unambiguous meaning. To do this we introduce the concept of an **interpretation**, which is a generalization of the valuations we saw in PC.

Definition. An **interpretation** M consists of the following:
(1) A **universe** U, which is any non-empty set.
(2) An assignment to each name of $\mathcal{L}$ a definite element in U.
(3) An assignment to each n-ary function symbol to a corresponding n-ary function in U.
(4) An assignment to each n-ary predicate symbol P of a definite n-place relation among elements of U. If $n = 1$, we call this a **property**, otherwise we call it a **relation**.

Now we can determine a definite truth value for each statement by giving $=, \forall, \exists$ their natural meanings as follows:

Truth-conditions of statements:
(1) $\exists x \varphi x$ is true iff there exists a $u \in U$ such that u has the property referred to by φ.
(2) $\forall x \varphi x$ is true iff for all $u \in U$ the property referred to by φ holds for u.
(3) $x = y$ is true iff the referent of x and the referent of y in U are the same element of U. (For example, "the morning star" and "the evening star" are names that both refer to the same object, Venus.)

Example. Suppose we only have one predicate symbol P and name a. Let interpretation M be such that U is the set of non-zero natural numbers. That is, let $U = \{1, 2, \ldots\}$. Let name a refer to 1, and P refer to the arithmetic relation $<$ ("less-than").

Consider the following statements and ask yourself if they're true or false given the truth-conditions of statements:
(1) $\exists x Lax$
(2) $\forall x Lax$
(3) $\forall x \exists y Lxy$

Tree Rules for PL

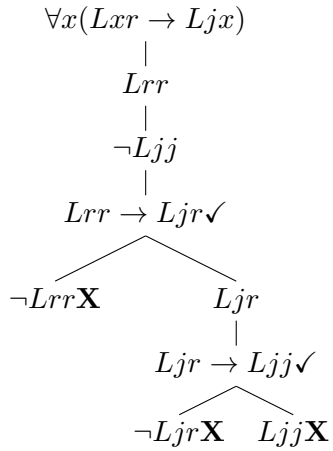
In part 1 we showed how to extend PC to a PL language $\mathcal{L}$. In a similar fashion, we can extend PC tree rules (PCT) to PL tree rules (PLT). We'll have to come up with a set of rules for $\forall, \exists$, and $=$.

Universal Instantiation (UI). When statement φ with form $\forall x\varphi(x)$ occupies an open path ψ of a tree:

(1) If some name n is in ψ, write $\varphi(n)$ at the end unless $\varphi(n)$ is already in ψ.
(2) If no name appears in ψ, choose some name n and write $\varphi(n)$ at the end.
(3) Do **not** tick $\forall x\varphi(x)$.

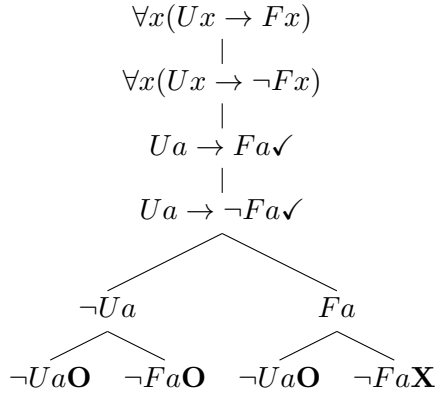
Example. Suppose we have individual names r and j, two-place predicate L, and the following argument: $\forall x(Lxr \to Ljx), Lrr \vdash Ljj$. We'll use PLT to check for the validity of this argument. **Note:** all of the test functions in tree procedures (tests for validity, tautology, etc) are the same as in PCT. Ask yourself why.

Two applications of UI to $\forall x(Lxr \to Ljx)$ and two applications of rule I produce the following PLT:



Example. Suppose we have predicates P and Q, and set of statements:
$S = \{\forall x(Ux \to Fx), \forall x(Ux \to \neg Fx)\}$

Use PLT to test the satisfiability of S. We apply UI once to each statement. Notice nothing in UI stops us from using the same name a, and as a result we produce the following tree:



Each finished open path on a finished PLT can be considered a possible universe in which all of the statements on the path are true. Thus if there's an open path in a finished tree, the set of premises considered are satisfiable.

Note: we really only need one of our quantifiers, as they are inter-definable.

$\neg\exists x\varphi(x) \equiv \forall x\neg\varphi(x)$ and $\neg\forall x\varphi(x) \equiv \exists x\neg\varphi(x)$

This is easy to see why in finite cases. Consider names a, b, c and predicate P.

$\forall xPx \Rightarrow Pa \wedge Pb \wedge Pc$

$\exists xPx \Rightarrow Pa \vee Pb \vee Pc$

Now consider $\neg\exists xPx \equiv \neg(Pa \vee Pb \vee Pc)$ to see the connection between the inter-definability of $\forall, \exists$ and the expressive completeness of $\{\neg, \wedge\}$, and $\{\neg, \vee\}$.

Back to some tree rules.

Negated Quantification (NQ). If $\neg\forall x\varphi(x)$ or $\neg\exists x\varphi(x)$ appears on an open path, tick it and write "$\exists x\neg$" in the place of "$\neg\forall x$" (or "$\forall x\neg$" in the place of "$\neg\exists x$").

$$\begin{array}{c}\neg\forall x\varphi(x)\checkmark\\ \exists x\neg\varphi(x)\end{array}$$

$$\begin{array}{c}\neg\exists x\varphi(x)\checkmark\\ \forall x\neg\varphi(x)\end{array}$$

Existential Instantiation (EI). Given unticked statement of the form $\exists x\varphi(x)$ on open path ψ, check to see if ψ contains $\varphi(n)$ for some name n. If not, pick a name that doesn't appear in ψ and write $\varphi(n)$ at the end. When this is done for each open path $\exists x\varphi(x)$ is on, tick it.

$$\begin{array}{c}\exists x\varphi(x)\checkmark\\ \psi\\ \varphi(n)\end{array}$$

where $n \notin \psi$. The idea in the name constraint is that we wish to produce a statement we have assumed nothing about, save that it is a witness to the existential statement (all we know from the statement alone is that *some* name fulfills it).

Suppose for example we were to misapply the rule when checking the satisfiability of following satisfiable set: $S = \{\exists xPx, \neg Pn\}$. If we allowed choice of n in using EI, we'd get the following tree:

$$\begin{array}{c}\exists xPx\checkmark\\ \neg Pn\\ Pn\mathbf{X}\end{array}$$

which would mean S is not satisfiable! But we know it is, and so we can see why this application of EI is prone to error. Let's see what happens when we do it right and choose a different name than n:

$$\exists x Px\checkmark$$
$$\neg Pn$$
$$Pa\mathbf{O}$$

which accords with the satisfiability of S.

Identity (ID). If an open path has a statement of the form $x = y$, and also a statement P in which at least one of x or y has at least one occurrence, then write Q where Q is generated by replacing any number of occurrences of x with y in P, and vice versa (y for x). Do not tick $x = y$.

$$x = y$$
$$P$$
$$Q$$

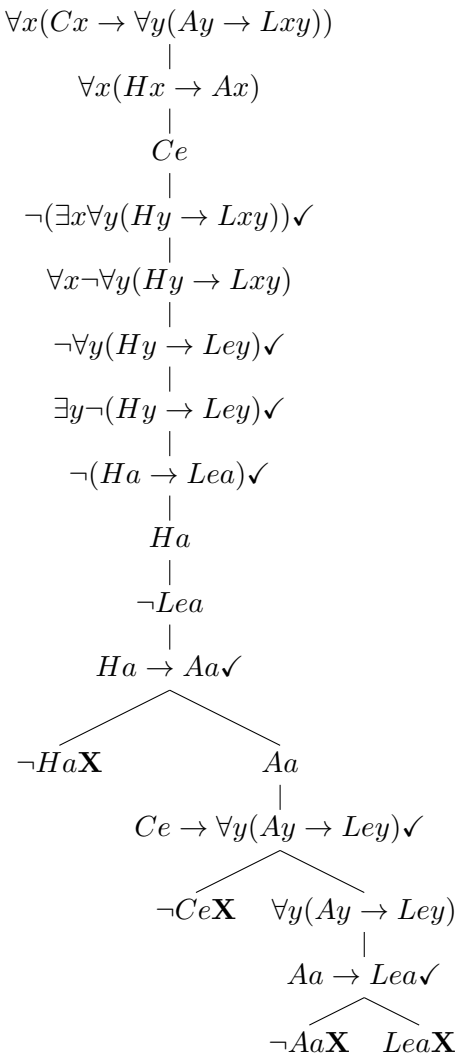
Non-identity (NID). If a statement of the form $a \neq a$ appears in a path, immediately close it.

$$a \neq a\mathbf{X}$$

(Note these two PLT rules ID and NID get us the four laws of identity: substitutivity, reflexivity, symmetry, and transitivity. Consider how to demonstrate each using PLT.)

To round out our discussion of PLT, let's see a bigger example.

Let our argument be $\forall x(Cx \to \forall y(Ay \to Lxy)), \forall x(Hx \to Ax), Ce \vdash \exists x \forall y(Hy \to Lxy)$.


$\forall x(Cx \to \forall y(Ay \to Lxy))$
$\forall x(Hx \to Ax)$
Ce
$\neg(\exists x \forall y(Hy \to Lxy))\checkmark$
$\forall x \neg \forall y(Hy \to Lxy)$
$\neg \forall y(Hy \to Ley)\checkmark$
$\exists y \neg(Hy \to Ley)\checkmark$
$\neg(Ha \to Lea)\checkmark$
Ha
$\neg Lea$
$Ha \to Aa\checkmark$
$\neg Ha\mathbf{X}$
Aa
$Ce \to \forall y(Ay \to Ley)\checkmark$
$\neg Ce\mathbf{X}$
$\forall y(Ay \to Ley)$
$Aa \to Lea\checkmark$
$\neg Aa\mathbf{X}$
$Lea\mathbf{X}$