Substitution

Variables are place holders so we must have some means of replacing them with more concrete information. On the syntactic side, we often need to replace a leaf node x by the parse tree of an entire term t. In substituting t for x we have to leave untouched the bound leaves x since they are in the scope of some ∃x or ∀x, i.e. they stand for some unspecified or all values respectively.

Given a variable x,aterm t and a formula φ we define φ[t/x] to be the formula obtained by replacing each free occurrence of variable x in φ with t.

Substitutions are easily understood by looking at some examples. Let f be a function symbol with two arguments and φ the formula with the parse tree. This is true because all occurrences of x are bound in φ, so none of them gets substituted.

Note that the bound x leaves are unaffected by this operation. You can see that the process of substitution is straightforward, but requires that it be applied only to the free occurrences of the variable to be substituted. A word on notation: in writing φ[t/x], we really mean this to be the formula obtained by performing the operation [t/x] on φ. Strictly speaking, the chain of symbols φ[t/x] is not a logical formula, but its result will be a formula, provided that φ was one in the first place.

Unfortunately, substitutions can give rise to undesired side effects. In performing a substitution φ[t/x], the term t may contain a variable y, where free occurrences of x in φ are under the scope of ∃y or ∀y in φ. By carrying out this substitution φ[t/x], the value y, which might have been fixed by a concrete context, gets caught in the scope of ∃y or ∀y. This binding capture overrides the context specification of the concrete value of y, for it will now stand for ‘some unspecified’ or ‘all ,’ respectively. Such undesired variable captures are to be avoided at all costs.