1 Structural Induction and Recursion for First-Order Formulae

Since first-order formulae are defined inductively, we can prove general facts about first-order formulae by Structural Induction and define operations on them by Structural Recursion. The modifications of these principles from propositional logic are straightforward.

Theorem 21.1.1 (Principle of Structural Induction) Suppose that X is a set of first-order formulae which satisfies the following clauses:

Basis step. All atomic formulae are in X.

Inductive step.
(i) If α is in X, then so are $(\neg\,\alpha)$, $(\forall x)\alpha$ and $(\exists x)\alpha$.
(ii) If α and β are in X, then so is $(\alpha \diamond \beta)$ where $\diamond \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Then X is the set of all formulae.

Remark 21.1.2 In propositional logic, we took propositions to be formation trees, from which it followed that every proposition could be uniquely decomposed. We will take first-order formulae to be formation trees, as defined in 20.1.10, so unique decomposability follows. This is required to justify the Principle of Structural Recursion. We will consider a slightly more general notion of a formation tree in the next section as well as a slight broadening of the Principle of Structural Recursion.

Theorem 21.1.3 (Principle of Structural Recursion) There is one and only one function F defined on the set of first-order formulae such that:

Basis clause. The value of F is specified explicitly on atomic formulae.

Recursive clause.
(i) The value of F on $(\neg\,\alpha)$ is specified in terms of the value of F on α.
(ii) The value of of F on $(\alpha \diamond \beta)$ is specified in terms of the value of F on α and β for each $\diamond \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$. (iii) The value of F on $(\forall x)\alpha$ and $(\exists x)\alpha$ is specified in terms of the value of F on α. This value may also depend on the variable x.

Example 21.1.4 (Depth of a Formula) The *depth* of a first-order formula is the longest path from root to leaf (where an atomic formula resides). Compare the definition here to the corresponding notion of depth of a proposition from Example 2.2.3.

$$\begin{aligned}
depth(A) &= 0 \qquad \text{for all atomic formulae } A \\
depth(\neg\,\phi) &= 1 + depth(\phi) \\
depth(\phi \diamond \psi) &= 1 + depth(\phi) + depth(\psi) \qquad \diamond \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \\
depth((\forall x)\phi) &= depth((\exists x)\phi) = 1 + depth(\phi).
\end{aligned}$$

2 Free and Bound Variables

Remark 21.2.1 Before defining the important notions of *free* and *bound* variable we look at some examples. In arithmetic we have equations such as

$$y = 3 \cdot x + 2.$$

The equation, as it stands, is neither true nor false. It attains a truth value only upon substitution for the variables y and x. For example, if we take $x = 2$ and $y = 8$, we have the true equation $8 = 3 \cdot 2 + 2$; if we take $x = 3$ and $y = 10$, we have the false equation $10 = 3 \cdot 3 + 2$. The important point is that the truth or

falsity of this equation depends on a choice of values for x and for y.

Now consider the formula

$$(\exists y) y = 3 \cdot x + 2.$$

The truth or falsity of this formula depends on the choice for x only, no such choice is relevant. If we set $x = 3$, then the formula becomes

$$(\exists y) y = 3 \cdot 3 + 2,$$

which is true, requiring no choice for y, even though y occurs in the formula. This reflects the fact that x occurs *free* in the formula but y does not, it occurs *bound* in the formula (by the existential quantifier $(\exists y)$.

Finally, the formula

$$(\forall x)(\exists y) y = 3 \cdot x + 2$$

is true, with no need for choosing values for x and y. All variables occurring in this formula are bound. We call such a formula a *sentence* to distinguish it from a formula, which may have free occurrences of variables.

Before defining the distinction between free and bound occurrences of a variable in a formula we formulate a notion of substitution of a term for a variable in a formula. (Compare this to the case with propositions **??**.)

Definition 21.2.2 (Substitution for a variable) Let t be any term, c a constant, and x a variable, then

$$t_c^x = \begin{cases} c & \text{if } t = x, \\ t & \text{otherwise.} \end{cases}$$

For every formulae ϕ, variable x and constant c, we define the substitution for x by c into a formula by recursion as follows: where ϕ and ψ are any formulae

$$\begin{aligned}
\left[Rt_1 \dots t_n\right]_c^x &= R([t_1]_c^x, \dots [t_n]_c^x) \qquad \text{for each } n\text{-ary } R \in \mathbf{PS} \\
\left[\bot\right]_c^x = \bot \qquad & \left[\top\right]_c^x = \top \\
\left[\neg\, \phi\right]_c^x &= \neg \left[\phi\right]_c^x \\
\left[\neg\, \phi \diamond \psi\right]_c^x &= \phi_c^x \diamond \psi_c^x \qquad \diamond \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \\
\left[(\forall y)\phi\right]_c^x &= \begin{cases} (\forall y)\phi & \text{if } x = y \\ \forall y)\left[\phi\right]_c^x & \text{otherwise.} \end{cases} \\
\left[(\exists y)\phi\right]_c^x &= \begin{cases} (\exists y)\phi & \text{if } x = y \\ \exists y)\left[\phi\right]_c^x & \text{otherwise.} \end{cases}
\end{aligned}$$

For example, if c is a constant

$$\left[(\forall x)P(x,y) \rightarrow (\exists y)P(x,y)\right]_c^x = (\forall x)P(x,y) \rightarrow (\exists y)P(c,y)$$

Definition 21.2.3 (sentence, free variable, bound variable) A variable x which occurs in a formula ϕ, has a *free occurrence* in ϕ if $\phi_c^x \neq \phi$. Otherwise, all occurrences of x in ϕ are *bound*. A *sentence* is formula ϕ such that every variable x and constant c, $\phi_c^x = \phi$. That is, a sentence is a formula with no free variables.

It is possible for a variable to have both free and bound occurrences in the same formula. In the formula $(\forall x)P(x,y) \rightarrow (\exists y)P(x,y)$ from the previous example, x occurs bound in the antecedent of the conditional and free in the consequent of the conditional.

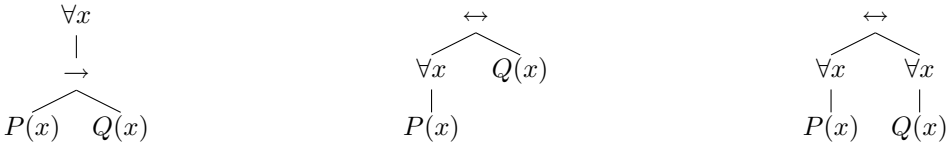
Here is another way to think about what a sentence. An particular occurrence of a variable x occurs *bound* in x, if it occurs in the formation tree in an atomic formula at a leaf below a node labeled by a

quantifier of the form $(\forall x)$ or $(\exists x)$. If a particular occurrence of a variable is not bound, then it is a *free occurrence* of the variable x. If a particular occurrence of x occurs bound in ϕ, then substitution ϕ_c^x has no effect on x. (Do you see why this is so based on the recursive definition of substitution?) A particular occurrence of a variable x occurs *free* in ϕ if it is not bound. So an occurrence of a variable occurs free if there is no node above it with a quantifier of the form $(\forall x)$ or $(\exists x)$, which means that if we make the substitution ϕ_c^x we have replaced that occurrence of x with c, so $\phi \neq \phi_c^x$. A sentence ϕ no free variables, that is, every occurrence of a variable x on a leaf occurs beneath a node with $(\forall x)$.

Example 21.2.4 Consider the following three formulae (the superscripts distinguishing different occurrences of the variable x):

$$\begin{array}{ll} (1) & (\forall x)((P(x^1) \to Q(x^2)) \\ (2) & ((\forall x)(P(x^1) \to Q(x^2))) \\ (3) & ((\forall x)(P(x^1)) \to ((\forall x)Q(x^2))) \end{array}$$

In (1) both occurrences are bound by the same instance of $(\forall x)$, that is, they both lie beneath the quantifier in the formation tree (see below). In (2) x^1 is bound, but x^2 is not. In (3) x^1 and x^2 are bound, but by different occurrences of $(\forall x)$. The formation tree makes these differences graphic:



The next example will have significance in the next section. But it is an application of Proof by Structural Induction.

Example 21.2.5 Let ϕ be any formula, x a variable and c a constant. Then

$$depth(\phi) = depth(\phi_c^x).$$

The proof is by Structural Induction. Let X be the set of first-order formulae ϕ for which $depth(\phi) = depth(\phi_c^x)$. For any n-place predicate symbol P and terms $t_1, \ldots, t_n$,

$$\big[Rt_1 \ldots t_n\big]_c^x = R([t_1]_c^x, \ldots [t_n]_c^x)$$

and since each formula is atomic, they have the same depth. Since substitution has no effect on $\bot, \top$, it follows that all substitutions into atomic formulae have depth zero, and so are in X.

Suppose $depth(\phi) = depth(\phi_c^x)$. Then

$$depth\big(\big[\neg\,\phi\big]_c^x\big) = depth\big(\neg\,\phi_c^x\big) = 1 + depth(\phi) = depth(\neg\,\phi),$$

so $\neg\,\phi \in X$. If $depth(\psi) = depth(\psi_c^x)$, then for any $\diamond \in \{\wedge, \vee, \to, \leftrightarrow\}$.

$$depth\big(\big[\phi \diamond \psi\big]_c^x\big) = depth\big(\phi_c^x \diamond \psi_c^x\big) = 1 + depth\big(\phi_c^x\big) + depth\big(\psi_c^x\big) = 1 + depth\big(\phi\big) + depth\big(\psi\big) = depth(\phi \diamond \psi)$$

So, $\phi \diamond \psi \in X$.

Finally consider $(\forall y)\phi$ and $(\exists y)\phi$. If $x = y$, then $\big[(\forall x)\phi\big]_c^x = (\forall x)\phi$ and similarly for $(\exists x)\phi$. So, $(\forall x)\phi, (\exists x)\phi \in X$. If $x \neq y$, then

$$depth\big(\big[(\forall y)\phi\big]_c^x\big) = depth\big((\forall y)\phi_c^x\big) = 1 + depth(\phi) = depth((\forall y)\phi),$$

and similarly for $(\exists y)\phi$. So, $(\forall y)\phi, (\exists y)\phi \in X$.

So, X is the set of all first-order formulae, that is $depth(\phi) = depth(\phi_c^x)$.

3 Expansions of a Language and General Formation Trees

The Principle of Structural recursion is not quite sufficient for justifying the semantic notion of truth we will give in the next lecture. For this we will need to consider formation trees of a different sort.

Definition 21.3.1 (Expansion of a language by parameters) Let $\mathcal{L}(\mathbf{CS}, \mathbf{PS})$ be a first-order language and A a nonempty set of constant symbols, distinct from those in $\mathbf{CS}$. We will call the new symbols in A *parameters* to distinguish those in $\mathbf{CS}$. Then the *expansion* of $\mathcal{L}$ by A, denoted by $\mathcal{L}^A$ is the new language $\mathcal{L}(\mathbf{CS} \cup A, \mathbf{PS})$.

Definition 21.3.2 (Immediate Subformula) Let $\mathcal{L}^A$ be an expansion of a language by parameters from A. The notion of an *immediate subformulae* is given by the following recursive definition.

1. An atomic formula has no immediate subformulae.
2. The immediate subformula of $\neg\,\phi$ is ϕ. The immediate subformulae of $\phi \diamond \psi$ are ϕ and ψ, where $\diamond \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
3. For any parameter $a \in A$, variable x and formula ϕ. ϕ^x_a is an immediate subformula of both $(\forall x)\phi$ and $(\exists x)\psi$

Note that if the set of parameters A is infinite, then the formulae $(\forall x)\phi$ and $(\exists x)\phi$ will have infinitely many subformulae.

We now define a formation tree in the *extended sense*.

Definition 21.3.3 (Formation Tree) By a *formation tree* (in the extended sense) for a language $\mathcal{L}$ extended by parameters A is meant a tree in which each leaf is labeled with an atomic formula, and such that all other nodes are determined by one of the following conditions

1. The node is labeled with $\neg$ and has one successor node which is the root of the formation tree for ϕ.
2. The node is labeled with a binary propositional connective $\diamond$ and has two successor nodes, the left node is the root of the formation tree for ϕ and the right node is the root of a formation tree for ψ.
3. The node is labeled by $\forall x$ or $\exists x$ and it has successors which are the roots of formation trees for $\phi^x_{a_1}, \phi^x_{a_2}, \phi^x_{a_3}, \ldots$, where $A = \{a_1, a_2, a_3, \ldots\}$. (The order of the successor nodes is not important, but we will assume it corresponds to some fixed ordering of the parameters of A.)

If the set of parameters A is infinite, the formation trees will not necessary be finitely generated (since quantifiers will correspond to infinite branching.) However, it is apparent that all paths through a formation tree are finite. In fact the depth of a formation tree (the length of the longest path) for ϕ in this extended sense is the same as the depth of the formation tree for ϕ given by Definition 20.1.10, by the argument given in Example 21.2.5.

We can reformulate the Principles of Structural Induction and Structural Recursion for formulae using the extended sense of formation trees.

Theorem 21.3.4 (Principle of Structural Induction–extended sense) Suppose that X is a set of first-order formulae which satisfies the following clauses:

Basis step. All atomic formulae are in X.

Inductive step.
(i) If α is in X, then so is $(\neg\,\alpha)$
(ii) If α and β are in X, then so is $(\alpha \diamond \beta)$ where $\diamond \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$,
(iii) If α^c_x is in X for every constant symbol c, then $(\forall x)\alpha$ is in X and $(\exists x)\alpha$ is in X.

Then X is the set of all formulae.

The main use for Structural Recursion in the extended sense is with the semantic definition of truth.

Theorem 21.3.5 (Principle of Structural Recursion–extended sense) There is one and only one function F defined on the set of first-order formulae such that:

BASIS CLAUSE. The value of F is specified explicitly on atomic formulae.

RECURSIVE CLAUSE.
(i) The value of F on $(\neg\,\alpha)$ is specified in terms of the value of F on α.
(ii) The value of of F on $(\alpha \diamond \beta)$ is specified in terms of the value of F on α and β for each $\diamond \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$. (iii) The value of F on $(\forall x)\alpha$ and $(\exists x)\alpha$ is specified in terms of the value of F on α_c^x for every constant c.

Remark 21.3.6 Here is the significance of this formulation of Structural Recursion. In the previous formula 21.1.3, the recursion went from $(\forall x)\alpha$ to α, but α may not be a sentence, even if $(\forall x)\alpha$ is. The whole point of distinguishing formulae with free variables and those without (sentences), is that the latter formulae will be given a definite truth value, while the former formulae have free variables, which do not refer to individuals in the domain, so the formulae are neither true nor false. In the formulation of Structural Recursion 21.3.5, the recursion goes from a universal sentence $(\forall x)\alpha$ to the sentences of the form α_c^x. So, we can define the truth of $(\forall x)\alpha$ based on the truth of the sentences α_c^x.