Logic for Computer Science LECTURE DATE : 5/2/2024

Lecture 7: Proof of Compactness Theorem (Continued) - Models

Lecture: Sujata Ghosh *Scribe: Kuntal Das, N. Vamsi Krishna*

1 Recapitulation

We are attempting to prove the compactness theorem (to be precise, the backwards implication); ie, we assume Γ, an infinite set of formulas to be finitely satisfiable and we aim to prove that Γ is satisfiable.

The steps taken so far are -

- **STEP 1** :
  We propose the following claims :
  - **Claim** 1: Every finitely satisfiable set of formulas can be extended to a finitely satisfiable and complete set of formulas.
  - **Claim** 2 : Every finitely satisfiable and complete set of formulas is a model set.
  - **Claim** 3 : Every model set is satisfiable
- **STEP 2**:
  Assuming the claims to be true, the sketch of our proof will go along the lines as following:
  $$\Gamma \xrightarrow{\text{extend}} \Delta \dashrightarrow \text{Model Set.}$$
  where Γ is a finitely satisfiable set of formulas and Δ is a finitely satisfiable and complete set of formulas.
- **STEP 3**: Since $\Gamma \subseteq \Delta$ and Δ has a model, then Γ also has a model, hence is satisfiable.
- **STEP 4**: We proceed to prove claim 1 and claim 2.
- **STEP 5**: We start proving claim 3 via method of induction.
  We take a model set Δ. To show that Δ is satisfiable, ie, to find a model $\mathcal{M}_\Delta$ such that $\mathcal{M}_\Delta \models \Delta$, we need to get $\mathcal{M}_\Delta = (\mathcal{D}_\Delta, \mathcal{I}_\Delta, \mathcal{G}_\Delta)$ such that $\mathcal{M}_\Delta \models \phi \iff \phi \subset \Delta$.
  Assume we somehow get hold of $\mathcal{M}_\Delta \models \phi$.
  To show $\mathcal{M}_\Delta \models \phi \iff \phi \subset \Delta$, we use induction on the size of ϕ.
  - **Base Case:** In base case, the formula ϕ can be of the following form (basically. the atomic formula case).
    i $\phi : t_1 = t_2$
    ii $\mathrm{p}_i^n t_1 t_2 .... t_n$

- **Induction Hypothesis:** Suppose the result holds for all formulas of size $\leq m$.
- **Induction Step:** Consider ϕ such that size of ϕ is $m+1$.
  Then ϕ is one of the following forms -
  * (a) $\phi : \neg\psi$
  * (b) $\phi : \psi \vee \chi$ or $\psi \wedge \chi$ or $\psi \rightarrow \chi$ or $\psi \longleftrightarrow \chi$
  * (c) $\phi : \forall$ x ψ or $\exists$ x ψ

- **STEP 6**: We assumed the base cases to be true and then proved (a) and (b) of the inductive step.
- **STEP 7**: Now we need to prove the base cases and (c) of the inductive step.

2 Model

Let us now move on to the base cases. We first try to define $\mathcal{M}_\Delta$.

Define $\mathcal{D}_\Delta := \{t : t \text{ is a term}\}$

Define $\mathcal{I}_\Delta$ as follows -

1. $\mathcal{I}_\Delta(c) = c$ for all constants c.
2. $\mathcal{I}_\Delta(\mathrm{f}_i^n) : \mathcal{D}_\Delta^n \longrightarrow \mathcal{D}_\Delta$ given as follows - $\mathcal{I}_\Delta(\mathrm{f}_i^n)(t_1, t_2, ..., t_n) = \mathrm{f}_n^i t_1 t_2 .....t_n$
3. $\mathcal{I}_\Delta(\mathrm{p}_i^n) \subseteq \mathcal{D}_\Delta^n$ given as follows - $(t_1, t_2, ..., t_n) \in \mathcal{I}_\Delta(\mathrm{p}_i^n)$ iff $\mathrm{p}_i^n t_1 t_2 .....t_n \in \Delta$

Define $\mathcal{G}_\Delta$: $\mathcal{G}_\Delta(x) = x$ for all variables x.

Thus, $\mathcal{M}_\Delta = (\mathcal{D}_\Delta, \mathcal{I}_\Delta, \mathcal{G}_\Delta)$

Now, remember(by a claim proved in Lecture 2) that this $\mathcal{G}_\Delta$ can be uniquely extended to a map $\mathcal{G}'_\Delta : \mathcal{T} \rightarrow \mathcal{D}_\Delta$

Exercise 1 *Show that $\mathcal{G}'_\Delta(t) = t$ for all terms t.*

Solution
- $\mathcal{G}'_\Delta(x_i) = x_i$ *(since $\mathcal{G}'_\Delta$ is extension of $\mathcal{G}_\Delta$, hence both should agree on variables).*
- $\mathcal{G}'_\Delta(c) = \mathcal{I}_\Delta(c) = c$ *for all constants c.*
- $\mathcal{G}'_\Delta(f_i^n t_1 t_2 .....t_n) = \mathcal{I}_\Delta(f_i^n)(\mathcal{G}'_\Delta(t_1), \mathcal{G}'_\Delta(t_2), ..., \mathcal{G}'_\Delta(t_n))$
  Then, inductively, $\mathcal{G}'_\Delta(|text f_i^n t_1, t_2, ..., t_n) = f_i^n t_1, t_2, ..., t_n$

(if t_i is a variable or constant, we know the value $\mathcal{G}'_\Delta(t_i) = t_i$; if not, then it is again of the form $f_m^j t_{i_1}, t_{i_2}, ..., t_{i_j}$; and continue recursively)
Thus,$\mathcal{G}'_\Delta(t) = t$ for all terms $t \in \mathcal{T}$.

Remark 1 *Since the extension $\mathcal{G}'_\Delta$ is unique, we do not distinguish between $\mathcal{G}_\Delta$ and $\mathcal{G}'_\Delta$, rather we mention $\mathcal{G}'_\Delta$ too as $\mathcal{G}_\Delta$.*

2.1 Base Cases

Recall that we had 2 situations in the base case -

i $\phi : t_1 \equiv t_2$

ii $\phi : \mathrm{f}_i^n t_1 t_2 .....t_n$

Let us prove (ii) first-
Proof.
Proof of base case (ii):
$\mathcal{M}_\Delta(t) \models \phi$
iff $\mathcal{M}_\Delta(t) \models \mathrm{p}_i^n t_1 t_2 .....t_n$
iff $(\mathcal{G}_\Delta(t_1), \mathcal{G}_\Delta(t_2), ..., \mathcal{G}_\Delta(t_n)) \in \mathcal{I}_\Delta(\mathrm{p}_i^n)$
iff $(t_1, t-2, ..., t_n) \in \mathcal{I}_\Delta(\mathrm{p}_i^n)$
iff $\mathrm{p}_i^n t_1 t_2 .....t_n \in \Delta$
iff $\phi \in \Delta$

This completes the proof for this case.

Now, for the proof of (i)-
Proof of base case (i):
We want to prove the following - $\mathcal{M}_\Delta \models t_1 \equiv t_2$ iff $t_1 \equiv t_2 \in \Delta$
While trying to prove this, we see that we face an issue, which is explained below -
$\mathcal{M}_\Delta \models t_1 \equiv t_2$
iff $\mathcal{G}_\Delta(t_1) \equiv \mathcal{G}_\Delta(t_2)$
iff $t_1 = t_2$ ——-(*)
iff $t_1 \equiv t_2$ □

The proof, on the surface, seems to be clean: except for the fact that we need to be very cautious about the (*) marked line.
$t_1 = t_2$ should be valid in true essence, not just by expression. An example would help understand this in a much easier way.

Example 1 *In the language of arithmetic,* $3 = 3$*, but we want the expressions* $2 + 1$ *or* $1 + 2$ *also t bear the same interpretation, and hence, denote the same term. We want both of these expressions to have the same meaning.*

To account for this problem, we go back to the definition of our domain and re-define it in suitable way.

2.2 Model : Re-defined

We had the definition of the domain as -
$\mathcal{D}_\Delta = \{t : t \text{ is a term}\}$
Define a relation $\sim$ on $\mathcal{T}$ as follows: $t_1 \sim t_2$ iff $t_1 \equiv t_2$

Claim $\sim$*, as defined above, is an equivalence relation.*

Proof.

1. $\sim$ **is reflexive**:
   We have to show that, for any term t; $t \equiv t \in \Delta$
   Suppose it is not true. Then there is a term t such that $t \equiv t \in \Delta \notin \Delta$. Note that $t \equiv t$ is a formula. Since Δ is complete, and $t \equiv t \notin \Delta$, then, $\neg(t \equiv t) \in \Delta$.
   So, $\{\neg(t \equiv t)\}$ is satisfiable - which is a contradiction.
   Thus, $t \equiv t \in \Delta$ for all terms $t \in \Delta$. Hence, $\sim$ is reflexive.

2. $\sim$ **is symmetric**:
   Suppose $t_1 \equiv t_2 \in \Delta$ for some $t_1, t_2 \in \mathcal{T}$; ie, $t_1 \sim t_2$.
   We have to show that $t_1 \sim t_2$.
   Suppose it is not true. Then $t_1 \equiv t_2 \notin \Delta$. Δ being complete, it must be the case that $\neg(t_1 \equiv t_2) \in \Delta$. Then, $\{t_1 \equiv t_2, \neg(t_1 \equiv t_2\}$ is satisfiable - which is a contradiction.
   Hence, $\sim$ is symmetric.

3. $\sim$**is transitive**:
   Suppose $t_1 \sim t_2$ and $t_2 \sim t_3$ for some $t_1, t_2, t_3 \in \mathcal{T}$; ie, $t_1 \equiv t_2 \in \Delta$ and $t_2 \equiv t_3 \in \Delta$.
   We have to show that $t_1 \equiv t_3 \in \Delta$, ir, $t_1 \equiv t_3 \in \Delta$.
   Suppose it is not true. Then $\neg(t_1 \equiv t_3) \in \Delta$. Therefore, $\{(t_1 \equiv t_2), (t_2 \equiv t_3), \neg(t_1 \equiv t_3) \subseteq \Delta\}$ is satisfiable - which is a contradiction.
   Then, $t_1 \equiv t_3 \in \Delta$, ie, $t_1 \sim t_3$.
   Hence, $\sim$ is transitive.

This completes the proof that $\sim$ is an equivalence relation. □

This equivalence relation gives rise to a collection of equivalence classes over $\mathcal{D}_\Delta = \mathcal{T}$. Define $\mathcal{D}'_\Delta = \{[t] : t \text{ is a term}\}$, where $[t]$ is the equivalence class of t under the equivalence relation $\sim$, as defined above.

Logic for Computer Science LECTURE DATE : 7/2/2024

Lecture 8: Proof of Compactness Theorem(Continued)- Substitutions

Lecture: Sujata Ghosh *Scribe: Kuntal Das, N. Vamsi Krishna*

1 Model : Re-defined (Continued)

We defined $\mathcal{D}'_\Delta = \{[t] : t \text{ is a term}\}$.
Now, we wish to define $\mathcal{I}'_\Delta$ and $\mathcal{G}'_\Delta$.
Define

- $\mathcal{I}'_\Delta(c) := [c]$
- $\mathcal{I}'_\Delta(\mathrm{f}_i^n) : (\mathcal{D}'_\Delta)^n \to \mathcal{D}'_\Delta$ defined as -
  $\mathcal{I}'_\Delta(\mathrm{f}_i^n)([t_1], [t_2], ..., [t_n]) = [\mathrm{f}_n^i t_1 t_2 .....t_n]$
- $\mathcal{I}'_\Delta(\mathrm{p}_i^n) \subseteq (\mathcal{D}'_\Delta)^n$ defined as - $([t_1], [t_2], ..., [t_n]) \in \mathcal{I}'_\Delta(\mathrm{p}_i^n)$ iff $\mathrm{p}_i^n t_1 t_2 .....t_n \in \Delta$

Next, we need to ensure well defined-ness of the above definitions.

1. **$\mathcal{I}'_\Delta(c) := c$ is well defined**:
   Let c_1, c_2 be two constants such that $c_1 \sim c_2$. Then $[c_1] = [c_2]$
   Hence, $\mathcal{I}'_\Delta(c_1) = [c_1] = [c_2] = \mathcal{I}'_\Delta(c_2)$
   Hence $\mathcal{I}'_\Delta(c)$ is well defined.

2. **$\mathcal{I}'_\Delta(\mathrm{f}_i^n)$ is well defined**:
   Let $t_1, t_2, ..., t_n$ and $t'_1, t'_2, ..., t'_n$ be terms such that
   $t_1 \equiv t'_1, t_2 \equiv t'_2, ..., t_n \equiv t'_n$
   ie, $t_1 \equiv t'_1, t_2 \equiv t'_2, ..., t_n \equiv t'_n \in \Delta$
   We want to show that $f_i^n t_1 t_2 .....t_n \equiv f_i^n t'_1 t'_2 .....t'_n \in \Delta$
   We use the following lemma-

   Lemma 1 *Let ϕ be a formula and Δ be a set of formulas, which is finitely satisfiable and complete. Then $\models \phi$ implies $\phi \in \Delta$*

   Proof. Suppose $\models \phi$ but $\phi \notin \Delta$
   Then $\neg\phi \in \Delta$. Thus, $\{\neg\phi\}$ is satisfiable - a contradiction.
   Hence, $\phi \in \Delta$ □

   Claim $\models ((t_1 \equiv t'_1) \wedge (t_2 \equiv t'_2) \wedge ..... \wedge (t_n \equiv t'_n)) \to (f_i^n t_1 t_2 .....t_n \equiv f_i^n t'_1 t'_2 .....t'_n)$

   Proof. We want to show that $f_i^n t_1 t_2 .....t_n \equiv f_i^n t'_1 t'_2 .....t'_n$
   ie, we want to show that $\mathcal{I}'_\Delta(f'_n)(t_1, t_2, ..., t_n) = \mathcal{I}_\Delta(f_i^n)(t'_1, t'_2, ..., t'_n)$
   ie, want to show $f_i^n(t_1, t_2, ..., t_n) \equiv f_i^n(t'_1, t'_2, ..., t'_n)$
   But the above is anyway true, since
   $[t_1] = [t'_1] \wedge [t_2] = [t'_2] \wedge ..... \wedge [t_n] = [t'_n]$

implies $\mathcal{I}'_\Delta(f_i^n)([t_1],[t_2],.....,[t_n]) = \mathcal{I}'_\Delta(f_i^n)([t'_1],[t'_2],.....,[t'_n])$
implies $[f_i^n(t_1,t_2,.....,t_n] = [f_i^n(t'_1,t'_2,.....,t'_n)]$
implies $f_i^n(t_1,t_2,.....,t_n) \equiv f_i^n(t'_1,t'_2,.....,t'_n)$
Thus, the validity is proved. □

Now, as $t_1 \equiv t'_1, t_2 \equiv t'_2, ..., t_n = t'_n\Delta$, we have $f_i^n t_1 t_2 .....t_n \equiv f_i^n t'_1 t'_2 .....t'_n \in \Delta$
This completes the proof of well defined-ness of $\mathcal{I}'_\Delta(f_i^n)$

3. **$\mathcal{I}'_\Delta(p_i^n)$ is well defined**.
Let $t_1, t_2, ..., t_n$ and $t'_1, t'_2, ..., t'_n$ be terms such that $t_1 \sim t'_1, t_2 \sim t'_2, ..., t_n \sim t'_n$. We want to show that if $(t_1, t_2, ..., t_n) \in \mathcal{I}'_\Delta(p_i^n)$ then $(t'_1, t'_2, ..., t'_n) \in \mathcal{I}'_\Delta(p_i^n)$ too and vice versa.
But this is obvious from the definition of $\mathcal{I}'_\Delta(p_i^n)$, since $[t_1] = [t'_1], [t_2] = [t'_2], ..., [t_n] = [t'_n]$.

Now, let us define $\mathcal{G}'_\Delta$ as follows: $\mathcal{G}'_\Delta(x) = [x]$, for all variables x
So, we have our model: $\mathcal{M}'_\Delta = (\mathcal{D}'_\Delta, \mathcal{I}'_\Delta, \mathcal{G}'_\Delta)$
As earlier, we can show that $\mathcal{G}'_\Delta(t) = [t]$ for all terms $t \in \mathcal{T}$(The proof is an exact imitation of solution of the exercise in the scribe of lecture 7).

Let us go back to the result we have to prove:

Claim *For all formulas $\phi, \mathcal{M}'_\Delta \models \phi$ iff $\phi \in \Delta$.*

Proof. We again use induction on the size of the formula.

- **Base case:**

1. ϕ is of the form $t_1 \equiv t_2$:

   $\mathcal{M}'_\Delta \models \phi$ iff $\mathcal{M}'_\Delta \models t_1 \equiv t_2$ iff $\mathcal{G}'_\Delta(t_1) = \mathcal{G}'_\Delta(t_2)$ iff $[t_1] = [t_2]$
   iff $t_1 \sim t_2$ iff $t_1 \equiv t_2 \in \Delta$ iff $\phi \in \Delta$
2. ϕ is of the form $p_i^n t_1 t_2 .....t_n$:

   $\mathcal{M}'_\Delta \models \phi$ iff $\mathcal{M}'_\Delta \models p_i^n t_1 t_2 .....t_n$ iff $(\mathcal{G}'_\Delta(t_1), \mathcal{G}'_\Delta(t_2), ..., \mathcal{G}'_\Delta(t_n)) \in \mathcal{I}_\Delta(p_i^n)$
   iff $([t_1],[t_2],.....,[t_n]) \in \mathcal{I}'_\Delta(p_i^n)$ iff $p_i^n t_1 t_2 .....t_n \in \Delta$ iff $\phi \in \Delta$

- So we are done with the base cases. Now, by our previous argument, we are also done for the formulas which are of the form $\neg\psi, \psi \vee \chi, \psi \wedge \chi, \psi \to \chi, \psi \longleftrightarrow \chi$.

Thus, if we consider only the atomic formulas (base cases) and their Boolean combinations (mentioned in the previous line), then we have the **Compactness Theorem** for the logical language consisting of the formulas ϕ, ψ of the following type:
$\phi, \psi : t_1 \equiv t_2$ or $p_i^n t_1 t_2 .....t_n$ or $\neg\phi$ or $\phi \vee \psi$ or $\phi \wedge \psi$ or $\phi \to \psi$ or $\phi \longleftrightarrow \psi$

We call this language the zeroth order language

□

Note that the proof is incomplete, since we are yet to prove the claim for formulas with quantifiers. We will now see that some new concept is needed for the quantifier case.

1.1 The quantifier case

Let us now consider the quantified formulas $\forall x\psi$ and $\exists x\psi$. We want to show that -
$\mathcal{M}'_\Delta \models \forall x\psi$ iff $\forall x\psi \in \Delta$ and $\mathcal{M}'_\Delta \models \exists x\psi$ iff $\exists x\psi \in \Delta$

Now, $\mathcal{M}'_\Delta \models \forall x\psi$ iff for all $d \in \mathcal{D}'_\Delta, \mathcal{M}'_{\Delta_{[x\to d}} \models \psi$ ——-(*)
(Remember that here, $\mathcal{D}'_\Delta = \{[t] : t \text{ is a term}\}$

To make sense of the notations in (*) marked statement, we have to talk about terms replacing variables in formulas. We introduce the concept of substitutions.

2 Substitutions

In this section, we will discuss an important operation of first-order logic: the substitution of a term for a variable. This operation will replace any occurrence of a variable in a formula by the given term. For instance, we will be able to substitute the term $f(m, x)$ for the variable y in the formula $P(y) \wedge Q(r, y)$ to obtain $P(f(m, x)) \wedge Q(r, f(m, x))$.
However, the substitution operation is surprisingly complex, as we need to deal with variables that are bound by quantifiers. The aim of this section is to give a rigorous presentation of variables and binding that allows us to safely carry out substitutions.

2.1 The Difficulty of Names and Variables

Let us first discuss two questions that arise in FOL:

1. Are the formulas $\forall x.P(x)$ and $\forall y.P(y)$ expressing the same?
2. What is the scope of variables?

- **Formula Equality and Renaming:**
  The first question can be answered by reading the formula without explicitly naming variables: **"P holds for all objects".**
  Clearly, this sentence corresponds to both formulas $\forall x.P(x)$ and $\forall y.P(y)$, and we should consider both formulas to be the same, already syntactically!
  This gives us a first rule that we will have to adhere to for FOL:
  Two formulas are considered to be the same, if we can bijectively rename the variables of one formula to obtain the other.
  For instance, consider the formulas $\phi = \forall x.\forall y.Q(x, y)$ and $\psi = \forall w.\forall z.Q(w, z)$. Then, a bijective renaming would be to rename x to w and y to z, which allows us to transform ϕ into ψ. However, renaming x to r and y to r is not bijective. Thus, we do not consider the formula $\forall r.\forall r.Q(r, r)$ to be same as ϕ. Note, that in the latter formula, the r refers to the inner-most quantifier and the outer quantifier has no effect!

(a) The plain tree (b) Tree with indication of the variable references

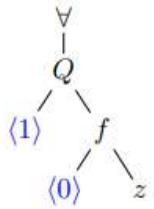
Figure 1: de Bruijn-Tree Representations of $\forall x.\, \forall y.\, Q(x, f(y, z))$

- **Scoping:**
  The second question concerns the scope of variables. This requires us to determine which objects a variable refers to. For instance, the variable x in the subformula $P(x) of \forall x.P(x)$ refers to what the quantifier ranges over. We say that x is in the scope of $\forall x$ in this formula. However, the variable y in $\forall x.Q(x, y)$ is in the scope of no quantifier and is thus a global reference. If we were now naively substituting x for y in this formula, then we would obtain $\forall x.Q(x, x)$. Here, the scope of the second argument of Q, and with it the meaning of the formula, has suddenly changed. This leads to a second rule:
  Substituting a term in a formula should not change the scoping of variables.

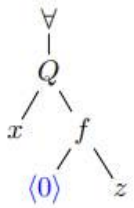
2.2 De Bruijn Trees

There are several ways to deal with variables, binding and substitution. An intuitively understandable way is to represent terms and formulas as **de Bruijn** trees. The idea is that bound variables are represented by a number that points to the quantifier that binds this variables. All free variables keep their name. Figure 1 shows the (de Bruijn-) tree representation of the formula $\forall x.\forall y.Q(x, f(y, z))$, where the left figure shows the actual tree and the right figure indicates to which quantifier the numerical name refers. We see that the bound variables are numbered starting from the inner-most quantifier. Also note that the variable z is free and thus keeps its name in the tree representation.
Since the nesting depth of quantifiers is important, we also note that tree representations are *not closed under subtrees!*. For instance, the tree in fig. 2a is not a valid tree representation because (1) is a dangling reference. Instead, if we want to remove the outer quantifier, we need to pick a name that we replace (1) with, say x, and then obtain the tree representation in fig. 2b.
In what follows, we will not work explicitly with tree representations, but will use another approach. However, the tree representation shows how wecan solve the initial problems:

1. The formulas $\forall x.P(x)$ and $\forall y.P(y)$ have the same tree representation, see fig. 3.
2. Substitutions can be carried out without changing the binding of variables, see fig. 4.



(a) Invalid de Bruijn-Tree



(b) Correct tree representing $\forall y.\, Q(x, f(y, z))$

Figure 2: Attempts of finding trees that represent the subformula $\forall y.\, Q(x, f(y, z))$ of $\forall x.\, \forall y.\, Q(x, f(y, z))$

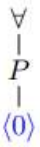
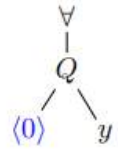
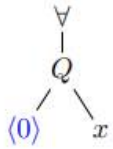


Figure 3: De Bruijn-Tree representing both formulas $\forall x.\, P(x)$ and $\forall y.\, P(y)$



(a) Representation before substitution



(b) Representation after substitution

Figure 4: Substitution of the free variable x for the free variable y by using the tree representation of $\forall x.\, Q(x, y)$

Note that the bound variable x is represented by (0) and thus there is no danger of substituting the unbound variable x for the unbound variable y. Hence, the problem of accidentally binding a variable does not exist for the tree representation.

2.3 Axiomatising Terms and Substitutions

The tree representations of formulas that we saw in section 1 solves our problems but it is difficult to work with. In fact, these trees are good way of implementing FOL on a computer, but not for humans to work with. In this section, we try to introduce a bunch of axioms that formulas and substitution have to fulfil. These axioms are fulfilled if we were to represent terms by de Bruijn trees, but for now, we will leave unspecified how formulas are implemented.

Definition 1: A *substitution* is a map $\sigma : \text{Var} \to \text{Term}$. Given $t \in \text{Term}$ and $x \in \text{Var}$, we

write $\sigma[x := t]$ for the updated substitution defined by

$$\sigma[x := t](y) = \begin{cases} t, & \text{if } x = y \\ \sigma(y), & \text{otherwise} \end{cases} \tag{1}$$

Similarly, we write $[x := t]$ for the substitution given by

$$[x := t](y) = \begin{cases} t, & \text{if } x = y \\ y, & \text{otherwise} \end{cases} \tag{2}$$

Given a term t, we write $t[\sigma]$ for the application of the substitution σ to t, defined by iteration on terms as follows.
$x[\sigma] = \sigma(x)$
$c[\sigma] = c$
$f(t_1, t_2, ..., t_n)[\sigma] = f(t_1[\sigma], ..., t_n[\sigma])$
The notation $\sigma[x := t]$ to update a substitution σ should be read like an assignment in an imperative programming language: the term that σ assigned to x will be overwritten by t. Similarly, the notation $[x := t]$ starts with a storage in which all variables y have the default value y, except for x which gets t assigned as initial value. It should also be noted that there is a substitution $\eta : \text{Var} \to \text{Term}$; that assigns to each variable y the term y, that is, we have $\eta(y) = y$. The notation $[x := t]$ is then a shorthand for $\eta[x := t]$.
The following example illustrates these notations.

Example 1 *Let $g(x, y)$ be a term with two free variables x and y, and one binary function symbol g. Moreover, let $\sigma = [x := y][y := x]$.*

1. *The substitution σ exchanges the two variables: $g(x, y)[\sigma] = g(x[\sigma], y[\sigma]) = g(y, x)$*
2. *Let now $\tau = \sigma[y := f(x)]$. In τ, the assignment of x to y is overwritten with the assignment of $f(x)$ to y. We thus have the following $g(x, y)[\tau] = g(x[\tau], y[\tau]) = g(y, f(x))$*
3. *It is sometimes convenient to exploit the notation and leave out square brackets. For example, instead of $g(x, y)[[y := f(x)]]$ we could write $g(x, y)[y := f(x)]$ and obtain $g(x, y)[y := f(x)] = g(x, f(x))$*

Next, we need to be able to apply substitutions to formulas. To circumvent the difficulties described in section 1, we assume that there is a set of terms on which we can carry out substitutions. This set of terms can in principle be defined by appealing to the tree representation. However, this is tedious and instead we just assume that formulas and the application of substitutions fulfil certain axioms.

Assumption *Given a formula ϕ, a variable x and a substitution σ, we say that x is fresh for σ in ϕ, if we have for all $y \in fv(\phi) - \{x\}$ that $x \notin fv(\sigma(y))$. We assume that there is an operation $\phi[\sigma]$ that applies a substitution σ to an FOL formula ϕ. Further, we assume for all $\diamond \in \forall, \exists$ and $\triangle \in \{\wedge, \vee, \to\}$ that the equality on formulas fulfils the following six axioms.*
$\phi \triangle \psi = \phi' \triangle \psi'$ *iff* $\phi = \phi'$ *and* $\psi = \psi'$ *(EC)*

$\diamond x.\phi = \diamond y.\psi$ *iff* $y \notin fv(\phi)$ *and* $\psi = \phi[x := y]$ *(EQ)*
$\tau[\sigma] = \tau$, *where* τ *is an atomic formula* *(SB)*
$P(t_1, ..., t_n)[\sigma] = P(t_1[\sigma], ..., t_n[\sigma])$ *(SP)*
$(\phi \triangle \psi)[\sigma] = \phi[\sigma] \triangle \psi[\sigma]$ *(SC)*
$(\diamond x.\phi)[\sigma] = \diamond x.\phi[\sigma[x := x]]$ *if* x *is fresh for* σ *in* ϕ *(SQ)*

Let us explain the intuition behind these axioms. First of all, there are two groups of axioms: those that define the equality on formulas (starting with E) and those that define the action of substitution on formulas (starting with S). The axiom (EQ) allows us to bijectively rename bound variables without illegally binding other variables. For instance, we have
$\forall x.Q(x, y) = \forall z.Q(z, y)$, because $z \notin fv(Q(x, y))$.
However, we have
$\forall x.Q(x, y) \notin \forall y.Q(y, y)$ because $y \in fv(Q(x, y))$.
The axiom (EC) allows us to carry out this renaming also in complex formulas that involve other connectives than quantifiers. Implicitly, we also use equality on terms and atoms, in the sense that
$P(t_1, ..., t_n) = P'(t'_1, ..., t'_n)$ iff $P = P'$ and $t_k = t'_k$ for all $1 \leq k \leq n$
and that equality is an equivalence relation (reflexive, symmetric and transitive).
The second group of axioms describes how substitution can be computed iteratively. Axioms (SB), (SP) and (SC) are doing what we would expect: no action on the atom τ, reduce subsitution on predicates to substitution in terms, and distribute substitution over propositional connectives. Complications arise only in the axiom (SQ), which has to make sure that the use of a bound variable is not changed and that variables are not accidentally bound. On the one hand, that the use of a bound variable is not changed is achieved by updating the substitution σ to $\sigma[x := x]$. For instance, if $\sigma(x) = g(y)$, then naively carrying out this substitution on $\forall x.P(x)$ would lead to $\forall x.P(g(y))$, which is certainly not what we want! Instead, we have by (SQ) and (SP)
$(\forall x.P(x))[\sigma] = \forall x.P(x)[\sigma[x := x]] = \forall x.P(x)$.
Accidental binding is preventing, on the other hand, by the precondition that x must be fresh for σ in ϕ. This condition ensures that none of the terms we want to substitute for the free variables in ϕ contains the variable x, which would become then bound by the quantifier. These rules and their interaction are best illustrated through some examples.

Example 2 *In the following, we use the substitution* σ *given by* $\sigma = [y := x][z := m]$.

1. *Let* $\phi = \forall z.Q(z, y)$. *First, we note that* $fv(Q(z, y)) \setminus z = y$ *and* $fv(\sigma(y)) = x$. *Thus,* $z \notin fv(\sigma(y))$ *and* z *is fresh. This gives us*
   $\phi[\sigma] = \forall z.Q(z, y)[\sigma[z := z]]$ *(SQ)*
   $= \forall z.Q(z[\forall[z := z]], y[\sigma[z := z]])$ *(SP)*
   $= \forall z.Q(z, x)$
   Note that $\sigma[z := z] = [y := x]$ *and the substitution of* m *for* z *in* σ *was "forgotten" when we applied the substitution under the quantifier. This is intuitively expected, as the bound variable* z *in* ϕ *is a local reference, while the* z *in* σ *refers to a global variable* z *that has the same name but is distinct from the local variable.*

2. *Let* $\psi = \forall x.Q(x, y)$. *First, we note that* $fv(Q(x, y)) \setminus \{x\} = \{y\}$ *and* $fv(\sigma(y)) = \{x\}$. *Thus,* $x \in fv(\sigma(y))$ *and* x *is not fresh. However, we have* $z \notin fv(Q(x, y))$ *and*

$Q(x,y)[x := z] = Q(z,y)$. *This allows us to rename the bound variable* x *in* ψ *to* z *and then carry out the substitution as above:*
$\psi[\sigma] = (\forall z.Q(z,y))[\sigma]$ *(EQ)*
$= \forall z.Q(z,x)$ *by the previous part of the example.*
Note that we cannot safely rename z *back to* x*, as we would otherwise illegally bind* x*.*

From now on, we will not make explicit use of the axioms provided in assumption (Then what will we do? This is explained in the next subsection - substitutability). Instead, we will rename bound variables whenever necessary before carrying out substitutions. For instance, we would just write $(\forall x.Q(x,y))[y := x] = \forall z.Q(z,x)$ without explicitly referring to the axioms. However, we know that in cause of doubt, we can always go back to the axioms and formally carry out the renaming and substitution.

Let's summarize the above discussion into important points which will be the take-away from this section.

Substitutions:

- *in terms:* Let t and t' be 2 terms and x be a variable. We write $t[t'/x]$ for $[t']$ replacing x in the term t.
  - $x[t'] = t'$
  - $y[t'] = y, y \neq x$
  - $c[t'/x] = c$
  - $f_i^n t_1 t_1 ..... t_n[t'/n] = f_i^n t_1[t'/x] t_2[t'/x] ..... t_n[t'_n/x]$
- *in formulas:* Let ϕ be a formula, t be a term and x be a variable. Then, we write $\phi[t/x]$ for x replaced by t in ϕ.
  - $(t_1 \equiv t_2)[t/x] = (t_1[t/x] \equiv t_2[t/x])$
  - $p_i^n t_1 t_2 .....{}_n[t/x] = p_i^n t_1[t/x] t_2[t/x] ..... t_n[t/x]$
  - $\neg\phi[t/x] = \neg(\phi[t/x])$
  - $(\psi \vee \chi)[t/x] = \psi[t/x] \vee \chi[t/x]$
  - $(\psi \wedge \chi)[t/x] = \psi[t/x] \wedge \chi[t/x]$
  - $(\psi \rightarrow \chi)[t/x] = \psi[t/x] \rightarrow \chi[t/x]$
  - $(\psi \longleftrightarrow \chi)[t/x] = \psi[t/x] \longleftrightarrow \chi[t/x]$
  - $\begin{cases} \forall x\psi[t/x] = \forall x\psi \\ \forall y\psi[t/x] = \forall y(\psi[t/x]), \quad y \neq x \end{cases}$
  - $\begin{cases} \exists x\psi[t/x] = \exists x\psi \\ \exists y\psi[t/x] = \exists y(\psi[t/x]), \quad y \neq x \end{cases}$

And the basic examples we should keep in mind, are here -

Examples:

- $(x \equiv y)[y/x] = y \equiv y$
- $(x \equiv y)[x/y] = x \equiv x$
- $\forall x(x \equiv y)[y/x] = (x \equiv y)$
- $\forall x(x \equiv y)[x/y] = (x \equiv x)$

Now, let's talk about *satisfiability of such substituted formulas*, an explicit formulation of our axiomatic assumptions as to which term is substitutable for which term in various cases (since, working with axioms is not a practical thing).
Let $\mathcal{M}$ be a model and consider a substituted formula $\phi[t/x]$. What would be a natural way to think about $\mathcal{M} \models \phi[t/x]$, where $\mathcal{M} = (\mathcal{D}, \mathcal{I}, \mathcal{G})$?
A natural guess would be : $\mathcal{M} \models \phi[t/x]$ iff $\mathcal{M}_{[x \to \mathcal{G}(t)]} \models \phi$
Question: Would this always hold?
Answer: *NO!!!*
Let's see this with an example -
$\phi : \exists y(\neg(y = x))$ —satisfiable
$\phi[y/x] : \exists y(\neg(y = y))$ —unsatisfiable
This brings up the question of substitutability, ie, which term is substitutable by which term in various scenario.

2.4 Substitutability

We want to have the term t said to be substitutable for a variable x in a formula ϕ. In this case, the definition of substitutability depends on ϕ.
A term t is said to be substitutable for a variable x in a formula ϕ if -

- **Case 1** *ϕ is $\neg\psi$:* t is always substitutable for x in ψ
- **Case 2** *ϕ is of the form $\neg\psi$:* t is substitutable for x in ψ
- **Case 3** *ϕ is of the form $\psi \vee \chi$:* t is substitutable for x in ψ and t is substitutable for x in χ
- **Case 4** *ϕ is of the form $\psi \leftarrow \chi$:* t is substitutable for x in ψ and t is substitutable for x in χ
- **Case 5** *ϕ is of the form $\psi \longleftrightarrow \chi$:* t is substitutable for x in ψ and t is substitutable for x in χ
- **Case 6** *ϕ is of the form $\forall y\psi$ or $\exists y\psi$:*

    1. x does not occur as a free variable in ψ
       OR
    2. y should not appear in the term t and t is substitutable for x in ψ

We end this lecture with one proposition, which is very basic and intuitive and captures the spirit of substitution.

Proposition 2 *Let ϕ be a formula, x be a variable, t be a term and $\mathcal{M}$ be a model. Then, $\mathcal{M} \models \phi[t/x]$ iff $\mathcal{M}_{[x \to \mathcal{G}(t)]} \models \phi$, provided t is substitutable for x in ϕ.*

We continue our discussion on substitutability in the next lecture.