Logic for computer science- Week 11
Normal forms for first order logic

In this lecture we define what equivalent formulae are in the case of first order logic and we introduce the prenex normal form for this class of formulae.

1 Equivalent formulae

In some contexts, several formulae have the same meaning. For instance, the formulae $\forall x.P(x,x)$ and $\forall y.P(y,y)$ have the same meaning in any context. Another example of formulae with the same meaning in $\neg\forall x.Q(x)$ and $\exists x.\neg Q(x)$. We call this formulae *equivalent*.

Some formulae have the same meaning only for some interpretation of predicate and functional symbols. For instance, if we work in a structure in which the predicate symbol P is interpreted as a symetric predicate, the formulae $P(x,y)$ and $P(y,x)$ have the same meaning. These formulae are called *equivalent in the structure S* in which we work.

These aspects are contained in the following definitions.

Definition 1.1. *Two formulae $\varphi_1 \in$ `LP1` and $\varphi_2 \in$ `LP1` are equivalent in a structure S if, for any S-assignment α,*

$$S, \alpha \models \varphi_1 \textit{ ddacă } S, \alpha \models \varphi_2.$$

The fact that φ_1 and φ_2 are equivalent in the structure S is denoted $\varphi_1 \overset{S}{\equiv} \varphi_2$.

In other words, two formulae are equivalent in a given structure S if, after evaluating the truth values of the formulae in the structure S, we obtain the same result for both formulae (both are true or both are false), for any assignment α with which we work.

Example 1.1. *We continue the examples from previous lectures. Let consider the signature $\Sigma = (\{P\}, \{f, i, e\})$ and Σ-structure $S_1 = (\mathbb{Z}, \{=\}, \{+, -, 0\})$.*

1. *We have that $P(x,y) \overset{S_1}{\equiv} P(y,x)$. Why?*

   Let α any S_1-assignment.

Avem $S_1, \alpha \models P(x, y)$ *iff (by definition of the relation* $\models$*)* $P^{S_1}(\bar{\alpha}(x), \bar{\alpha}(y)) = 1$
iff $\bar{\alpha}(x) = \bar{\alpha}(y)$
iff $\alpha(x) = \alpha(y)$
iff (by the symmetry of equality) $\alpha(y) = \alpha(y)$
iff (by definition of the relation $\models$*)* $S_1, \alpha \models P(y, x)$.

So, for any S_1*-assignment* α*, we have:* $S_1, \alpha \models P(x, y)$ *iff* $S_1, \alpha \models P(y, x)$*, which is exactly the definition for* $P(x, y) \overset{S_1}{\equiv} P(y, x)$*.*

2. *We have that* $P(x_1, x_3) \overset{S_1}{\not\equiv} P(x_2, x_3)$*. Why?*

   Because there is an S_1*-assignment* $\alpha : \mathcal{X} \to \mathbb{Z}$*, defined by* $\alpha(x_1) = 42, \alpha(x_2) = 7, \alpha(x_3) = 42$ *and* $\alpha(x) = 0$*,* $\forall x \in \mathcal{X} \setminus \{x_1, x_2, x_3\}$ *with the property that*

   $S_1, \alpha \models P(x_1, x_3)$ *(because* $42 = 42$*), but*
   $S_1, \alpha \not\models P(x_2, x_3)$ *(because* $42 \neq 7$*).*

In the case that the structure is not given, we have the following definition:

Definition 1.2. *Two formulae* $\varphi_1 \in$ `LP1` and $\varphi_2 \in$ `LP1` are equivalent *if, for any structure* S *and for any* S*-assignment* α*,*

$$S, \alpha \models \varphi_1 \text{ iff } S, \alpha \models \varphi_2.$$

The fact that φ_1 *and* φ_2 *are equivalent is denoted by* $\varphi_1 \equiv \varphi_2$*.*

Example 1.2. *We continue the previous example.*

1. *We have that* $P(x, y) \not\equiv P(y, x)$*. Why?*

   Because there is an Σ*-structure and an assignment in that structure such that the two formulae have idfferent truth values.*

   Let take the structure $S_5 = (\mathbb{Z}, \{<\}, \{+, -, 0\})$ *defined in previous lectures and* S_5*-assignment* $\alpha_6 : \mathcal{X} \to \mathbb{Z}$*, defined by* $\alpha(x) = 2, \alpha(y) = 3$ *and* $\alpha(z) = 1$ *for any* $z \in \mathcal{X} \setminus \{x, y\}$*.*

   Note that the only difference between S_1 *and* S_5 *is the fact that the predicate symbol* P *is interpreted by the predicate* $=$ *in* S_1*, while in* S_5 *is interpreted by* $<$ *(the relation less then is interpreted over integers).*

   We have $S_5, \alpha_6 \models P(x, y)$*, because* $2 < 3$*, but* $S_5, \alpha_6 \not\models P(y, x)$*, because* $3 \not< 2$*. So the formulae* $P(x, y)$ *and* $P(y, x)$ *are not equivalent (even if they are equivalent in the structure* S_1*).*

2. *We have that* $\forall x.P(x,z) \equiv \forall y.P(y,z)$. *Why?*

   Let S *be any* Σ*-structure with the domain* D *and* $\alpha : \mathcal{X} \to D$ *any* S*-assignment.*

   We have that

$$\begin{array}{ll}
S,\alpha \models \forall x.P(x,z) & \textit{iff} \\
\textit{for any } u \in D, \textit{ we have } S,\alpha[x \mapsto u] \models P(x,z) & \textit{iff} \\
\textit{for any } u \in D, \textit{ we have } P^S\Big(\overline{\alpha[x \mapsto u]}(x), \overline{\alpha[x \mapsto u]}(z)\Big) = 1 & \textit{iff} \\
\textit{for any } u \in D, \textit{ we have } P^S\Big(u, \alpha(z)\Big) = 1 & \textit{iff} \\
\textit{for any } u \in D, \textit{ we have } P^S\Big(\overline{\alpha[y \mapsto u]}(y), \overline{\alpha[y \mapsto u]}(z)\Big) = 1 & \textit{iff} \\
\textit{for any } u \in D, \textit{ we have } S,\alpha[y \mapsto u] \models P(y,z) & \textit{iff} \\
S,\alpha \models \forall y.P(y,z). &
\end{array}$$

So, for any Σ*-structure* S*, for any* S*-assignment* α*, we have that*

$$S,\alpha \models \forall x.P(x,z) \textit{ iff } S,\alpha \models \forall y.P(y,z),$$

which is exactly the definition of $\forall x.P(x,z) \equiv \forall y.P(y,z)$.

2 Substitutions

As in the case of propositional logic, for first order logic, there are some normal forms corresponding to formulae.

In order to define the normal forms, we recall the notion of *substitution*:

- A *substitution* is a function $\sigma : \mathcal{X} \to \mathcal{T}$, with the property that $\sigma(x) \neq x$ for a finite number of variables $x \in \mathcal{X}$;
- *The domain of a substitution* σ is the set $dom(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$;
- *The extension* of the substitution σ to the set of terms is the function $\sigma^\sharp : \mathcal{T} \to \mathcal{T}$, defined as:

  1. $\sigma^\sharp(x) = \sigma(x)$, for any $x \in \mathcal{X}$;
  2. $\sigma^\sharp(c) = c$, for any constant symbol $c \in \mathcal{F}_0$;
  3. $\sigma^\sharp(f(t_1,\ldots,t_n)) = f(\sigma^\sharp(t_1),\ldots,\sigma^\sharp(t_n))$, for any functional symbol $f \in \mathcal{F}_n$ of arity $n \in \mathbb{N}$ and any terms $t_1,\ldots,t_n \in \mathcal{T}$.

  If $t \in \mathcal{T}$ is a term, then $\sigma^\sharp(t) \in \mathcal{T}$ is the term obtained from t by *applying* the substitution σ (each occurrence of a variable x in t is replaced by the term $\sigma(x)$).

- If $dom(\sigma) = \{x_1,\ldots,x_n\}$, then the substitution σ may be also written as follows:

$$\sigma = \{x_1 \mapsto \sigma(x_1),\ldots,x_n \mapsto \sigma(x_n)\}.$$

- *The restriction of the substitutioni $\sigma$to the set V is a new substitution denoted by $\sigma|_V : \mathcal{X} \to \mathcal{T}$*, and defined as:

  1. $\sigma|_V(x) = \sigma(x)$ for any $x \in V$;
  2. $\sigma|_V(x) = x$ for any $x \in \mathcal{X} \setminus V$.

  In other words, by restricting a substitution to a set of variables, we remove the other variables from the domain of the substitution.

- *The extension* of σ to the set of formulae is the function $\sigma^\flat : \texttt{LP1} \to \texttt{LP1}$, defined as:

  1. $\sigma^\flat(P(t_1, \ldots, t_n)) = P(\sigma^\sharp(t_1), \ldots, \sigma^\sharp(t_n))$;
  2. $\sigma^\flat(\neg\varphi) = \neg\sigma^\flat(\varphi)$;
  3. $\sigma^\flat(\varphi_1 \land \varphi_2) = \sigma^\flat(\varphi_1) \land \sigma^\flat(\varphi_2)$;
  4. $\sigma^\flat(\varphi_1 \lor \varphi_2) = \sigma^\flat(\varphi_1) \lor \sigma^\flat(\varphi_2)$;
  5. $\sigma^\flat(\varphi_1 \to \varphi_2) = \sigma^\flat(\varphi_1) \to \sigma^\flat(\varphi_2)$;
  6. $\sigma^\flat(\varphi_1 \leftrightarrow \varphi_2) = \sigma^\flat(\varphi_1) \leftrightarrow \sigma^\flat(\varphi_2)$;
  7. $\sigma^\flat(\forall x.\varphi) = \forall x.(\rho^\flat(\varphi))$, where $\rho = \sigma|_{dom(\sigma)\setminus\{x\}}$;
  8. $\sigma^\flat(\exists x.\varphi) = \exists x.(\rho^\flat(\varphi))$, where $\rho = \sigma|_{dom(\sigma)\setminus\{x\}}$.

  In other words, in order to obtain the formula $\sigma^\flat(\varphi)$ from φ, each **free** occurence of a variable x from the formula φ is replaced by the term $\sigma(x)$.

3 Prenex Normal Form

Definition 3.1. *A formula φ* is in prenex normal form *if*

$$\varphi = Q_1 x_1.Q_2 x_2.\ldots.Q_n x_n.\varphi',$$

where:

1. *$Q_i \in \{\forall, \exists\}$ (for any $1 \le i \le n$);*
2. *φ' does not contain quantifiers.*

In other words, a formula is in prenex normal form (PNF) if all quantifiers are "in front of the formula".

Example 3.1.
1. *the formula $\forall x.\exists y.(P(x,y) \land \neg P(z,y))$ is in prenex normal form;*
2. *the formula $\forall x.((\exists y.P(x,y)) \land \neg P(z,y))$ is not in prenex normal form.*

Any formula can be written in prenex normal form, fact that is captured by the following theorem:

Theorem 3.1. *For any formula $\varphi \in$ `LP1`, there is $\varphi' \in$ `LP1` such that:*

1. *φ' is in prenex normal form;*
2. *$\varphi \equiv \varphi'$.*

The formula φ' is a prenex normal form for the formula *φ.*

In the following er prove the above theorem by presenting an algorithm that computes φ' starting from φ. But first we need the following theorem:

Lemma 3.1 (Renaming lemma)**.** *Let $\varphi \in$ `LP1` be a formula and $x, y \in \mathcal{X}$ two variables with the property that $y \notin vars(\varphi)$.*

Then the following equivalence holds:

$$\forall x.\varphi \equiv \forall y.(\sigma^\flat(\varphi)) \textit{ și } \exists x.\varphi \equiv \exists y.(\sigma^\flat(\varphi)),$$

where $\sigma = \{x \mapsto y\}$.

In other words, the above lemma says that in the formula $\forall x.\varphi$ can replace the quantifier $\forall x$ ($\exists x$ respectively) with a quantifier $\forall y$($\exists y$ respectively) with the condition that y is not a variable of φ. Also, the free occurrences of x in φ have to be replaced by y by applying the substitution $\sigma = \{x \mapsto y\}$ over the formula φ.

Example 3.2. *Let consider the formula $\forall x.P(x, y)$. Because $z \notin free(P(x, y))$, we have by the renaming lemma that $\forall x.P(x, y) \equiv \forall z.P(z, y)$.*

Note that $\forall x.P(x, y) \not\equiv \forall y.P(y, y)$ (the equivalence cannot be explained by the renaming lemma because $y \in free(P(x, y))$ and anyway it does not hold).

Now we are ready to present the sketch of the proof for Theorem 3.1.

Proof. We apply the following equivalences, from left to right:

1. $(\forall x.\varphi_1) \land \varphi_2 \equiv \forall x.(\varphi_1 \land \varphi_2)$, if $x \notin free(\varphi_2)$;
2. $(\forall x.\varphi_1) \lor \varphi_2 \equiv \forall x.(\varphi_1 \lor \varphi_2)$, if $x \notin free(\varphi_2)$;
3. $(\exists x.\varphi_1) \land \varphi_2 \equiv \exists x.(\varphi_1 \land \varphi_2)$, if $x \notin free(\varphi_2)$;
4. $(\exists x.\varphi_1) \lor \varphi_2 \equiv \exists x.(\varphi_1 \lor \varphi_2)$, if $x \notin free(\varphi_2)$;
5. $\neg\forall x.\varphi \equiv \exists x.\neg\varphi$;
6. $\neg\exists x.\varphi \equiv \forall x.\neg\varphi$.

In the case one of first four equivalences cannot be applied because of the restriction $x \notin free(\varphi_2)$, we first have to apply the renaming lemma in order to rename the bound variable x.

Also, we may use the commutativity of connectors $\land$ and $\lor$:

1. $\varphi_1 \land \varphi_2 \equiv \varphi_2 \land \varphi_1$;

2. $\varphi_1 \vee \varphi_2 \equiv \varphi_2 \vee \varphi_1$.

The effect of forst 6 equivalences from above is to move the quantifiers, in the abstract tree of the formula, above the quantifiers $\wedge, \vee, \neg$, ensuring in this way that the algorithm terminates and that all quantifiers will be as close to the root of the tree as possible.

In order to treat the operators $\rightarrow, \leftrightarrow$, we can use their "translation" using the connectors $\wedge, \vee, \neg$:

1. $\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$;

2. $\varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$.

□

Further, we give an example to compute the prenex normal form for a formula using the above algorithm.

Example 3.3. *Let consider the formula* $\varphi = \Big(\forall x.\neg\big(P(x,x) \wedge \neg\exists y.P(x,y)\big)\Big) \wedge P(x,x)$.

We cannot apply the equivalence te bring the quantifier $\forall x$ *in front of the formula because* $x \in \mathit{free}(P(x,x))$. *Therefore, we first have to apply the renaming lemma (R.L.):*

$$\begin{array}{lcl}
\varphi & = & \Big(\forall x.\neg\big(P(x,x) \wedge \neg\exists y.P(x,y)\big)\Big) \wedge P(x,x) \\
 & \overset{R.L.}{\equiv} & \Big(\forall z.\neg\big(P(z,z) \wedge \neg\exists y.P(z,y)\big)\Big) \wedge P(x,x) \\
 & \overset{1}{\equiv} & \forall z.\Big(\neg\big(P(z,z) \wedge \neg\exists y.P(z,y)\big) \wedge P(x,x)\Big) \\
 & \overset{6}{\equiv} & \forall z.\Big(\neg\big(P(z,z) \wedge \forall y.\neg P(z,y)\big) \wedge P(x,x)\Big) \\
 & \overset{commutativity\ \wedge}{\equiv} & \forall z.\Big(\neg\big((\forall y.\neg P(z,y)) \wedge P(z,z)\big) \wedge P(x,x)\Big) \\
 & \overset{1}{\equiv} & \forall z.\Big(\neg\big(\forall y.(\neg P(z,y) \wedge P(z,z))\big) \wedge P(x,x)\Big) \\
 & \overset{commutativity\ \wedge}{\equiv} & \forall z.\Big(\neg\big(\forall y.(P(z,z) \wedge \neg P(z,y))\big) \wedge P(x,x)\Big) \\
 & \overset{5}{\equiv} & \forall z.\Big(\big(\exists y.\neg(P(z,z) \wedge \neg P(z,y))\big) \wedge P(x,x)\Big) \\
 & \overset{3}{\equiv} & \forall z.\exists y.(\neg(P(z,z) \wedge \neg P(z,y)) \wedge P(x,x)).
\end{array}$$

Therefore, we found that the formula $\forall z.\exists y.(\neg(P(z,z) \wedge \neg P(z,y)) \wedge P(x,x))$ *is a prenex normal form for the formula* $\Big(\forall x.\neg\big(P(x,x) \wedge \neg\exists y.P(x,y)\big)\Big) \wedge P(x,x)$.

When we do the calculus, we usually do not mark explicitly the step where we apply the commutativity and we write shorter as follows:

$$
\begin{aligned}
\varphi &= \Big(\forall x.\neg\big(P(x,x) \land \neg\exists y.P(x,y)\big)\Big) \land P(x,x) \\
&\overset{R.L.}{\equiv} \Big(\forall z.\neg\big(P(z,z) \land \neg\exists y.P(z,y)\big)\Big) \land P(x,x) \\
&\overset{1}{\equiv} \forall z.\Big(\neg\big(P(z,z) \land \neg\exists y.P(z,y)\big) \land P(x,x)\Big) \\
&\overset{6}{\equiv} \forall z.\Big(\neg\big(P(z,z) \land \forall y.\neg P(z,y)\big) \land P(x,x)\Big) \\
&\overset{1}{\equiv} \forall z.\Big(\neg\big(\forall y.(P(z,z) \land \neg P(z,y))\big) \land P(x,x)\Big) \\
&\overset{5}{\equiv} \forall z.\Big(\big(\exists y.\neg(P(z,z) \land \neg P(z,y))\big) \land P(x,x)\Big) \\
&\overset{3}{\equiv} \forall z.\exists y.(\neg(P(z,z) \land \neg P(z,y)) \land P(x,x)).
\end{aligned}
$$

4 Closed formulae

Definition 4.1. *A formula $\varphi \in$ LP1 is* closed *if free$(\varphi) = \emptyset$.*

In other words, if a formula does not have free variables, it is called closed. Closed formulae are also called *sentences*.

Definition 4.2. *A formula that is not closed is called* opened.

Example 4.1. *The formula $\forall x.P(x,x) \land \exists y.P(y,x)$ is a closed formula because free$(\forall x.P(x,x) \land \exists y.P(y,x)) = \emptyset$.*

The formula $\forall z.\exists y.(\neg(P(z,z) \land \neg P(z,y)) \land P(x,x))$ is not closed (is opened) because free$(\forall z.\exists y.(\neg(P(z,z) \land \neg P(z,y)) \land P(x,x))) = \{x\}$.

Definition 4.3. *Let $\varphi \in$ LP1 be a formula and free$(\varphi) = \{x_1, \ldots, x_n\}$ the set of its free variables.*

The formula

$$\exists x_1.\exists x_2.\ldots.\exists x_n.\varphi$$

is is called the existential closure of the formula φ.

Remark 4.1. *The existential closure of a formula is a closed formula.*

Example 4.2. *The existential closure of the formula $\forall z.\exists y.(\neg(P(z,z) \land \neg P(z,y)) \land P(x,x))$ is $\exists x.\forall z.\exists y.(\neg(P(z,z) \land \neg P(z,y)) \land P(x,x))$.*

Definition 4.4. *Two formulae $\varphi_1 \in$ LP1 and $\varphi_2 \in$ LP1 are* equisatisfiable, *if:*

1. *either both φ_1 and φ_2 are satisfiable;*
2. *or neither φ_1 or φ_2 are satisfiable.*

In other qirds, the only cases in which the two formulae are not equisatisfiable are when one of the formula is satisfiable and the other not.

Theorem 4.1. *Any formula is equisatisfiable with its existential closure.*

Definition 4.5. *Let $\varphi \in$ `LP1` be a formula and free$(\varphi) = \{x_1, \ldots, x_n\}$ the set of its free variables.*

The formula

$$\forall x_1.\forall x_2.\ldots.\forall x_n.\varphi$$

is called universal closure *of the formula φ.*

Remark 4.2. *The universal closure of a formula is a closed formula.*

Example 4.3. *The universal closure of the formula $\forall z.\exists y.(\neg(P(z,z) \wedge \neg P(z,y)) \wedge P(x,x))$ is $\forall x.\forall z.\exists y.(\neg(P(z,z) \wedge \neg P(z,y)) \wedge P(x,x))$.*

Theorem 4.2. *A formula is valid if and only if its universal closure is valid.*