Math 4680, Topics in Logic and Computation, Winter 2012

Lecture Notes 3: The Language of First-Order Logic

Peter Selinger

1 The language of first-order logic

1.1 Signatures

Definition. A ***first-order signature*** consists of two disjoint sets $\mathcal{F}$ and $\mathcal{P}$ of ***function symbols***, respectively, ***predicate symbols***, together with an ***arity function*** $\mathrm{ar} : \mathcal{F} \cup \mathcal{P} \to \mathbb{N}$. If $f \in \mathcal{F}$ and $\mathrm{ar}(f) = n$, then we call f an ***n-ary function symbol***. If $P \in \mathcal{P}$ and $\mathrm{ar}(P) = n$, then we call P an ***n-ary predicate symbol***. We also use the words "nullary", "unary", "binary", etc., for 0-ary, 1-ary, 2-ary and so on. A nullary function symbol is also called a ***constant symbol***. A nullary predicate symbol is sometimes called a ***sentence symbol***.

Example 1. The ***signature of elementary arithmetic*** has a constant symbol "0", a unary function symbol "S", binary function symbols "$+$", "$\cdot$", and "E", and a binary predicate symbol "$<$". The intended interpretations of these symbols are respectively zero, the successor function, addition, multiplication, exponentiation, and the less than relation.

Example 2. The ***signature of set theory*** has at least a binary predicate symbol "$\in$". Sometimes one also adds a constant symbol "$\emptyset$", binary function symbols "$\cup$" and "$\cap$", and so forth.

1.2 Alphabet

First-order terms and formulas will be defined relative to a given signature. Given a signature, the ***alphabet*** of first-order logic consists of the elements of $\mathcal{F}$ and $\mathcal{P}$, plus the following symbols (which we assume are not in $\mathcal{F}$ and $\mathcal{P}$):

$\neg \quad \wedge \quad \vee \quad \to \quad \top \quad \bot \quad \forall \quad \exists$	(connectives)
$\approx$	(equality)
$(\quad) \quad ,$	(auxiliary symbols)
$x_1 \quad x_2 \quad x_3 \quad \ldots$	(variables)

Notice that there is a countable supply of variables. We denote the set of variables by $\mathcal{V}$, and we often use lower-case roman letters such as x, y, z to denote variables.

Notice that when we discussed sentential logic, we included the connective "$\leftrightarrow$" as a primitive connective of our language. We have dropped this connective now, since it constituted an unnecessary luxury, and it did not mesh well with natural deduction. However, we do not go as far as Enderton, who drops all connectives except for "$\to$", "$\neg$", and "$\forall$".

Let $\mathcal{A}^*$ denote the set of finite strings over the alphabet $\mathcal{A}$. If α and β are strings, then we denote their concatenation by $\alpha\beta$.

1.3 Well-formed terms

Definition 3. The set $\mathcal{T} \subseteq \mathcal{A}^*$ of ***well-formed terms*** (or simply ***terms***) over a given first-order signature is the smallest subset of $\mathcal{A}^*$ such that

1. If x is a variable, then $x \in \mathcal{T}$.
2. If c is a constant symbol, then $c \in \mathcal{T}$.
3. If f is an n-ary function symbol, where $n \geqslant 1$, and if $t_1, \ldots, t_n \in \mathcal{T}$, then $f(t_1, \ldots, t_n) \in \mathcal{T}$.

Example 4. Relative to the signature of elementary arithmetic, the following are well-formed terms:

$$+(x,y) \qquad S(S(S(0))) \qquad \cdot(S(S(0)),+(E(x,0),S(0)))$$

We will soon start using a more readable, informal syntax for terms, writing, for instance $x+y$, 3, and $2\cdot(x^0+1)$ for the above three terms. However, the formal syntax, while cumbersome, has technical advantages because it guarantees us a unique readability result similar to that proved for well-formed formulas of sentential logic. When we use the informal syntax, it will always be understood that we actually mean the corresponding well-formed terms of the formal language.

1.4 Well-formed formulas

Definition 5. The set $\mathcal{W} \subseteq \mathcal{A}^*$ of ***well-formed formulas (wff's)*** over a given first-order signature is the smallest subset of $\mathcal{A}^*$ such that

1. If Q is a nullary predicate symbol, then $Q \in \mathcal{W}$.

2. If P is an n-ary predicate symbol, where $n \geqslant 1$, and if $t_1, \ldots, t_n$ are well-formed terms, then $P(t_1, \ldots, t_n) \in \mathcal{W}$.

3. If t_1 and t_2 are well-formed terms, then $\approx(t_1, t_2) \in \mathcal{W}$.

4. $\top, \bot \in \mathcal{W}$.

5. If $\alpha \in \mathcal{W}$ then $(\neg \alpha) \in \mathcal{W}$.

6. If $\alpha, \beta \in \mathcal{W}$ then $(\alpha \wedge \beta) \in \mathcal{W}$, $(\alpha \vee \beta) \in \mathcal{W}$, and $(\alpha \to \beta) \in \mathcal{W}$,

7. If $\alpha \in \mathcal{W}$ and x is a variable, then $(\forall x\, \alpha) \in \mathcal{W}$ and $(\exists x\, \alpha) \in \mathcal{W}$.

The formulas build by rules 1–4 are called ***atomic formulas***. All other well-formed formulas are called ***composite formulas***.

The sets of terms and formulas that arise from a given signature are also called the ***language*** of that signature. For instance, we speak of the language of elementary arithmetic, or the language of set theory, etc.

Remark. In Definitions 3 and 5, we speak of the "smallest" set of strings satisfying certain conditions. In order for this to be well-defined, we must prove in each case that such a smallest set actually exists. The proof is very similar to the case of sentential logic, and we omit it here.

Notice that equality symbol $\approx$ is automatically included in every first-order language. We regard $\approx$ as a logical symbol with a fixed interpretation, namely equality. In this, it differs from the other predicate symbols, which are non-logical symbols whose meaning depends on the context. The symbol $\approx$ is not part of the signature of a language, since the signature only fixes the non-logical symbols. However, we do occasionally refer to $\approx$ as a binary predicate symbol.

1.5 Examples

Relative to the signature of elementary arithmetic, the following are well-formed formulas:

1. $\approx(x, S(0))$.

2. $(\forall x\, (\exists y\, (\neg \approx(x, y))))$.

3. $(\forall x\, (\forall y\, (\forall z\, ((<(x, y) \wedge <(y, z)) \to <(x, z)))))$.

Relative to the signature of set theory, the following are well-formed formulas:

4. $\in(x, y)$.

5. $(\exists x\, (\forall y\, (\neg \in(y, x))))$.

6. $(\forall x\, (\forall y$
$((\forall z\, ((\in(z, x) \to \in(z, y)) \wedge (\in(z, y) \to \in(z, x)))) \to \approx(x, y))))$.

1.6 Backus-Naur Form

In computer science, where one needs to define languages all the time, a useful notation has emerged for defining sets of strings. The notation is called BNF (for Backus-Naur Form). The BNF notation is nothing but a succinct way of stating the content of Definitions 3 and 5, and other such similar definitions. The following is the BNF for well-formed terms and formulas of first-order logic:

$$\begin{array}{ll}
\text{Terms:} & t \ ::= x \ \big|\ c \ \big|\ f(t_1, \ldots, t_n) \\
\text{Formulas:} & \alpha ::= Q \ \big|\ P(t_1, \ldots, t_n) \ \big|\ \approx(t_1, t_2) \ \big|\ \top \ \big|\ \bot \\
 & \qquad \big|\ (\neg \alpha) \ \big|\ (\alpha_1 \wedge \alpha_2) \ \big|\ (\alpha_1 \vee \alpha_2) \ \big|\ (\alpha_1 \to \alpha_2) \\
 & \qquad \big|\ (\forall x\, \alpha) \ \big|\ (\exists x\, \alpha)
\end{array}$$

Here, the symbols t, and α are meta-variables that range over terms and formulas, respectively. Moreover, x ranges over variables, c ranges over constant symbols, f ranges over n-ary function symbols with $n \geqslant 1$, Q ranges over nullary predicate symbols, and P ranges over n-ary predicate symbols with $n \geqslant 1$.

Each clause of the BNF defines a syntactic class, such as terms, formulas, etc. The vertical lines in each clause separate the different alternative forms that a string in that syntactic class can have. The meaning of a BNF is that it defines the smallest set(s) of strings closed under certain operations. Thus, it is really just a short-hand notation for the inductive clauses spelled out in Definitions 3 and 5.

1.7 Induction, recursion, and unique readability

As in the case of sentential logic, there are induction and recursion principles for terms and formulas of first-order logic. Also, a unique readability result holds.

Since the proofs are very similar to those of sentential logic, we omit them here, and we will henceforth use induction and recursion without further ado.

1.8 Informal syntax

We will also adopt a more liberal syntax for writing terms and formulas informally. For the sentential connectives, we adopt the same precedence rules as before. We also write $\forall x\,\alpha$ or $\forall x.\alpha$ instead of $\mathtt{(}\forall x\,\alpha\mathtt{)}$, and similar for $\exists$. The convention is that when we use a dot, as in $\forall x.\alpha$, then the scope of the quantifier extends as far as possible, so that $\forall x.\alpha\wedge\beta$ means $\forall x\mathtt{(}\alpha\wedge\beta\mathtt{)}$ and not $\mathtt{(}\forall x\,\alpha\mathtt{)}\wedge\beta$.

For terms, we use the usual infix notation where appropriate. For instance, we write $x+y\cdot z$ instead of $+\mathtt{(}x,\cdot\mathtt{(}y,z\mathtt{))}$. We do the same for certain binary predicates, for instance we write $x<y$, $x\in y$, and $x\approx y$ instead of $<\mathtt{(}x,y\mathtt{)}$, $\in\mathtt{(}x,y\mathtt{)}$, and $\approx\mathtt{(}x,y\mathtt{)}$, respectively.

If P is a predicate symbol of arity $n=0$, then we will sometimes also write $P(t_1,\ldots,t_n)$ to denote P itself. This convention allows us to handle the cases $n=0$ and $n\geqslant 1$ uniformly in case distinctions. With this convention, the atomic formulas are precisely the formulas of the form $P(t_1,\ldots,t_n)$, $t_1\approx t_2$, $\top$, or $\bot$.

We also adopt certain other conventions, such as writing $x\not< y$ instead of $\neg\, x<y$, and $\alpha\leftrightarrow\beta$ instead of $(\alpha\to\beta)\wedge(\beta\to\alpha)$. The following are the well-formed formulas from Section 1.5 written in the informal syntax.

1. $x\approx S(0)$.
2. $\forall x\exists y\, x\notin y$.
3. $\forall x\forall y\forall z.\, x<y\wedge y<z\to x<z$.
4. $x\in y$.
5. $\exists x\forall y\, y\notin x$.
6. $\forall x\forall y\,((\forall z.\, z\in x\leftrightarrow z\in y)\to x\approx y)$.

1.9 Free variables

Definition. The set $\mathrm{FV}(t)$ of free variables of a term t, and the set $\mathrm{FV}(\alpha)$ of free variables of a formula α, are recursively defined as follows:

$$\begin{array}{llcll}
\text{Terms:} & \mathrm{FV}(x) & = & \{x\} & \\
& \mathrm{FV}(f(t_1,\ldots,t_n)) & = & \mathrm{FV}(t_1)\cup\ldots\cup\mathrm{FV}(t_n) & (n\geqslant 0)\\[1ex]
\text{Formulas:} & \mathrm{FV}(P(t_1,\ldots,t_n)) & = & \mathrm{FV}(t_1)\cup\ldots\cup\mathrm{FV}(t_n) & (n\geqslant 0)\\
& \mathrm{FV}(t_1\approx t_2) & = & \mathrm{FV}(t_1)\cup\mathrm{FV}(t_2) & \\
& \mathrm{FV}(\top) & = & \emptyset & \\
& \mathrm{FV}(\bot) & = & \emptyset & \\
& \mathrm{FV}(\mathtt{(}\neg\alpha\mathtt{)}) & = & \mathrm{FV}(\alpha) & \\
& \mathrm{FV}(\mathtt{(}\alpha_1\,\square\,\alpha_2\mathtt{)}) & = & \mathrm{FV}(\alpha)\cup\mathrm{FV}(\beta) & \\
& \mathrm{FV}(\mathtt{(}\forall x\,\alpha\mathtt{)}) & = & \mathrm{FV}(\alpha)-\{x\} & \\
& \mathrm{FV}(\mathtt{(}\exists x\,\alpha\mathtt{)}) & = & \mathrm{FV}(\alpha)-\{x\} &
\end{array}$$

We say a variable x is ***free*** in α if $x\in\mathrm{FV}(\alpha)$. This is the case if x occurs somewhere in α, but not in the scope of a quantifier. A formula α is called a ***sentence*** if $\mathrm{FV}(\alpha)=\emptyset$. We sometimes denote sentences by the letters σ and τ.

2 Truth and Models

2.1 Structures

To interpret the formulas of first-order logic, we need to know three things: first, we need to know in which set to interpret the variables. Second, we need to know how to interpret the basic constant and function symbols. Third, we need to know how to interpret the basic predicate symbols.

Definition. Fix a first-order signature $\langle\mathcal{F},\mathcal{P},\mathrm{ar}\rangle$. A ***structure*** $\mathfrak{A}$ for the signature consists of the following data:

1. A non-empty set $|\mathfrak{A}|$, called the ***carrier*** of $\mathfrak{A}$. The elements of $|\mathfrak{A}|$ are also called ***objects*** or ***individuals***.
2. For each n-ary function symbol f in $\mathcal{F}$, a function

   $$f^{\mathfrak{A}}:|\mathfrak{A}|^n\to|\mathfrak{A}|,$$

   called the ***interpretation*** of f.

3. For each n-ary predicate symbol P in $\mathcal{P}$, a relation
$$P^{\mathfrak{A}} \subseteq |\mathfrak{A}|^n,$$
called the ***interpretation*** of P.

Note that we require the carrier to be non-empty. Also note that the interpretation of each function symbol is a total function, i.e., it is everywhere defined. We do not allow function symbols to denote partial functions. So for instance, if we were to interpret the language of fields, we would be forced to interpret the inversion operation $(-)^{-1}$ by a total function. We handle this situation by interpreting 0^{-1} by some dummy value, for instance, we could let $0^{-1} = 0$.

2.2 Examples

The ***standard model*** for the language of elementary arithmetic is the following structure:

$$\begin{array}{lcll}
|\mathfrak{A}| & = & \mathbb{N} = \{0,1,2,\ldots\} & \\
<^{\mathfrak{A}} & = & \{\langle x,y\rangle \in \mathbb{N} \mid x < y\} & \\
0^{\mathfrak{A}} & = & 0 & \\
S^{\mathfrak{A}}(x) & = & x+1, & \text{for all } x \in \mathbb{N} \\
+^{\mathfrak{A}}(x,y) & = & x+y, & \text{for all } x,y \in \mathbb{N} \\
\cdot^{\mathfrak{A}}(x,y) & = & x\cdot y, & \text{for all } x,y \in \mathbb{N} \\
E^{\mathfrak{A}}(x,y) & = & x^y, & \text{for all } x,y \in \mathbb{N}
\end{array}$$

The following defines another structure for the language of elementary arithmetic:

$$\begin{array}{lcll}
|\mathfrak{B}| & = & \{0,1,2,3,4\} & \\
<^{\mathfrak{B}} & = & \{\langle x,y\rangle \in \mathbb{N} \mid y = x+1 \pmod 5\} & \\
0^{\mathfrak{B}} & = & 0 & \\
S^{\mathfrak{B}}(x) & = & x+1 \pmod 5, & \text{for all } x \in \mathbb{N} \\
+^{\mathfrak{B}}(x,y) & = & x+y \pmod 5, & \text{for all } x,y \in \mathbb{N} \\
\cdot^{\mathfrak{B}}(x,y) & = & x\cdot y \pmod 5, & \text{for all } x,y \in \mathbb{N} \\
E^{\mathfrak{B}}(x,y) & = & x^y \pmod 5, & \text{for all } x,y \in \mathbb{N}
\end{array}$$

The structure $\mathfrak{A}$ satisfies different sentences than the structure $\mathfrak{B}$. For example, the following sentences hold in $\mathfrak{B}$, but not in $\mathfrak{A}$:
$$\forall x\exists y(x \approx S(y)).$$

$$S(S(S(S(S(0))))) \approx 0.$$

The following sentences hold in $\mathfrak{A}$, but not in $\mathfrak{B}$:
$$\forall x\forall y\forall z(x < y \wedge y < z \rightarrow x < z).$$
$$\forall x\forall y(x + y \approx 0 \rightarrow x \approx 0 \wedge y \approx 0).$$

In particular, not every structure of elementary arithmetic makes all the sentences true that would be true of the natural numbers. If a structure satisfies, say, the axioms of number theory, then we call it a ***model*** of number theory. Notice the difference in the usage of the words "structure" and "model": we speak of a *structure for a language*, but of a *model of the axioms*. But we are getting ahead of ourselves: axioms and models are discussed in Section 2.5 below.

2.3 The interpretation of terms and formulas

In the previous section, we have given some examples of structures, and we have informally spoken of a structure "satisfying" certain sentences but not others. We now need to define formally what we mean by a structure satisfying a sentence. To this end, we need to be able to interpret terms and formulas in a given structure. We first need to define the concept of a valuation of variables. Throughout the rest of this section, fix a language. Recall that we wrote $\mathcal{V}$ for the set of variables and $\mathcal{T}$ for the set of terms of a language.

Definition. A ***valuation*** in a structure $\mathfrak{A}$ is a map $s : \mathcal{V} \to |\mathfrak{A}|$ which assigns an element of the carrier to each variable of the language. If s is a valuation, $x \in V$ a variable, and $a \in |\mathfrak{A}|$ is an individual, then we denote by $s(x|a)$ the valuation s' that behaves like s, except that it assigns a to x:
$$s'(z) = \begin{cases} a & \text{if } z = x, \\ s(z) & \text{if } z \neq x. \end{cases}$$

Any valuation $s : \mathcal{V} \to |\mathfrak{A}|$ can be recursively extended to an interpretation $\bar{s} : \mathcal{T} \to |\mathfrak{A}|$ of arbitrary terms as follows:
$$\begin{array}{lcll}
\bar{s}(x) & = & s(x) & \\
\bar{s}(f(t_1,\ldots,t_n)) & = & f^{\mathfrak{A}}(\bar{s}(t_1),\ldots,\bar{s}(t_n)) & (n \geqslant 0)
\end{array}$$

If $\mathfrak{A}$ is a structure, $s : V \to |\mathfrak{A}|$ a valuation, and φ a formula, then we define whether $|\mathfrak{A}|$ ***satisfies φ under the valuation s***, in symbols $\models_{\mathfrak{A}} \varphi\,[s]$, by the following recursive clauses:

$\models_{\mathfrak{A}} P(t_1,\ldots,t_n)\,[s]$	iff	$\langle \bar{s}(t_1),\ldots,\bar{s}(t_n)\rangle \in P^{\mathfrak{A}}$
$\models_{\mathfrak{A}} t_1 \approx t_2\,[s]$	iff	$\bar{s}(t_1) = \bar{s}(t_2)$
$\models_{\mathfrak{A}} \top\,[s]$	always	
$\models_{\mathfrak{A}} \bot\,[s]$	never	
$\models_{\mathfrak{A}} (\neg\varphi)\,[s]$	iff	not $\models_{\mathfrak{A}} \varphi\,[s]$
$\models_{\mathfrak{A}} (\varphi_1 \wedge \varphi_2)\,[s]$	iff	$\models_{\mathfrak{A}} \varphi\,[s]$ and $\models_{\mathfrak{A}} \psi\,[s]$
$\models_{\mathfrak{A}} (\varphi_1 \vee \varphi_2)\,[s]$	iff	$\models_{\mathfrak{A}} \varphi\,[s]$ or $\models_{\mathfrak{A}} \psi\,[s]$
$\models_{\mathfrak{A}} (\varphi_1 \to \varphi_2)\,[s]$	iff	$\models_{\mathfrak{A}} \varphi\,[s]$ implies $\models_{\mathfrak{A}} \psi\,[s]$
$\models_{\mathfrak{A}} (\forall x\,\varphi)\,[s]$	iff	for all $a \in |\mathfrak{A}|$, $\models_{\mathfrak{A}} \varphi\,[s(x|a)]$
$\models_{\mathfrak{A}} (\exists x\,\varphi)\,[s]$	iff	there exists $a \in |\mathfrak{A}|$ such that $\models_{\mathfrak{A}} \varphi\,[s(x|a)]$

Remark. Strictly speaking, the above definition does not fit into the schema of our recusion principle, because we are definition a predicate by recursion, and not a function. More formally, one would first define, by recursion, a function $\text{models}_{\mathfrak{A}}$ which assigns a truth value $\text{models}_{\mathfrak{A}}(\varphi, s) \in \{T, F\}$ to each pair of a well-formed formula φ and a valuation s. One would then define $\models_{\mathfrak{A}} \varphi\,[s]$ if and only if $\text{models}_{\mathfrak{A}}(\varphi, s) = T$.

2.4 Logical entailment

Definition. Let Γ be a set of well-formed formulas, and φ a well-formed formula. We say that Γ ***logically implies*** (or ***entails***) φ, in symbols $\Gamma \models \varphi$, if for every structure $\mathfrak{A}$ and every valuation s in $\mathfrak{A}$: if $\models_{\mathfrak{A}} \psi\,[s]$ holds for every $\psi \in \Gamma$, then $\models_{\mathfrak{A}} \varphi\,[s]$.

As before in the case of sentential logic, we use certain shorthand notations. Thus we write $\psi_1,\ldots,\psi_n \models \varphi$ instead of $\{\psi_1,\ldots,\psi_n\} \models \varphi$, and $\models \varphi$ instead of $\emptyset \models \varphi$. Thus, $\models \varphi$ means that every structure and every valuation satisfies φ. In this case, we say that the formula φ is ***valid***. We say that two formulas φ and ψ ***logically equivalent*** if $\varphi \models \psi$ and $\psi \models \varphi$. In this case, we also write $\varphi \models\!\!\!\dashv \psi$.

Notice that, as compared to our terminology in sentential logic, we have dropped the "tauto" from "tautological entailment" and "tautological equivalence". One usually speaks of tautologies only in the case of sentential logic.

Example. Prove that $\forall x.\, Qx \models Qy$. Take an arbitrary structure $\mathfrak{A}$ and an arbitrary valuation s such that $\models_{\mathfrak{A}} \forall x.\, Qx\,[s]$. Then, by the interpretation of the universal quantifier, for all $a \in \mathfrak{A}$, we have $\models_{\mathfrak{A}} Qx\,[s(x|a)]$, and thus, for all $a \in \mathfrak{A}$, $a \in Q^{\mathfrak{A}}$. In particular, $s(y) \in Q^{\mathfrak{A}}$, and thus $\models_{\mathfrak{A}} Qy\,[s]$. Since $\mathfrak{A}$ and s were arbitrary, this shows $\forall x.\, Qx \models Qy$.

Example. Prove that $Qy \not\models \forall x.\, Qx$. It suffices to give a particular structure and a valuation such that the left-hand side is satisfied and the right-hand side is not. So let $|\mathfrak{A}| = \{0, 1\}$, and let $Q^{\mathfrak{A}} = \{0\}$. Let s be a valuation such that $s(y) = 0$. Then $\models_{\mathfrak{A}} Qy\,[s]$, but not $\models_{\mathfrak{A}} \forall x.\, Qx\,[s]$.

Notice that the situation is similar to doing proofs with truth tables: To prove that a logical entailment holds, we need to consider ***all*** structures and valuations, whereas to prove that a logical entailment doesn't hold, it suffices to give a single counterexample. The difference is that structures, unlike truth tables, need not be finite, and thus there is no obvious mechanical method for deciding whether a given logical entailment holds or not. (In fact, one can prove that there isn't a non-obvious method either).

2.5 Theories and models

We first observe that the interpretation of a sentence does not depend on the valuation s.

Lemma 6. *1. If s, s' are valuations which agree on the free variables of a term t, then $\bar{s}(t) = \bar{s}'(t)$.*

2. If s, s' are valuations which agree on the free variables of a formula φ, then $\models_{\mathfrak{A}} \varphi\,[s]$ if and only if $\models_{\mathfrak{A}} \varphi\,[s']$.

Proof. Exercise. □

Corollary 7. *If σ is a sentence and $\mathfrak{A}$ is a structure, then $\models_{\mathfrak{A}} \sigma\,[s]$ is independent of the valuation s (i.e., either it holds for all valuations, or for no valuation).* □

We say that $\mathfrak{A}$ is a ***model*** of σ if $\models_{\mathfrak{A}} \sigma\,[s]$ for some s. Since this notion does not depend on s (by the previous corollary), we simply write $\models_{\mathfrak{A}} \sigma$ in this case. Similarly, if Σ is a set of sentences, then we say that $\mathfrak{A}$ is a ***model*** of Σ, in symbols $\models_{\mathfrak{A}} \Sigma$, if $\models_{\mathfrak{A}} \sigma$ for all $\sigma \in \Sigma$.

Let $\mathfrak{A}$ be a structure. The set of all sentences that are satisfied in $\mathfrak{A}$ is called the ***theory*** of $\mathfrak{A}$. It is also denoted $\text{Th}(\mathfrak{A})$. More generally, if $\mathscr{C}$ is a class of structures, the set of those sentences that are satisfied in *every* structure $\mathfrak{A} \in \mathscr{C}$ is called the ***theory*** of $\mathscr{C}$, and written $\mathrm{Th}(\mathscr{C})$.

We can also take the opposite point of view: starting from a set Σ of sentences, we can ask for the class of structures which satisfy Σ. This class is denoted by $\mathrm{Mod}(\Sigma)$. In this context, the sentences in Σ are often called ***axioms***, and we say that the class $\mathrm{Mod}(\Sigma)$ is ***axiomatized*** by Σ.

Definition. We say that $\mathscr{C}$ is ***axiomatizable*** if $\mathscr{C} = \mathrm{Mod}(\Sigma)$ for some set Σ of sentences. We say that a class $\mathscr{C}$ of structures is ***finitely axiomatizable*** if $\mathscr{C} = \mathrm{Mod}(\sigma)$ for some sentence σ.

A finitely axiomatizable class is also called an ***elementary class***. A general axiomatizable class is also called an ***elementary class in the wider sense***.

Remark. Since $\mathrm{Mod}(\sigma_1, \ldots, \sigma_n) = \mathrm{Mod}(\sigma_1 \wedge \ldots \wedge \sigma_n)$, any class that is axiomatizable by finitely many sentences is already axiomatizable by a single sentence. Thus, no generality is lost in the previous definition by defining an elementary class to be a class axiomatized by a single sentence.

Notice that for any structure $\mathfrak{A}$ and any sentence σ, the following holds:

$$\sigma \in \mathrm{Th}(\mathfrak{A}) \qquad \Longleftrightarrow \qquad \models_{\mathfrak{A}} \sigma \qquad \Longleftrightarrow \qquad \mathfrak{A} \in \mathrm{Mod}(\sigma).$$

Proposition 8. *The following hold, for any class $\mathscr{C}$ of structures and any set Σ of sentences:*

1. $\Sigma \subseteq \mathrm{Th}(\mathscr{C})$ *if and only if* $\mathscr{C} \subseteq \mathrm{Mod}(\Sigma)$.
2. $\mathscr{C} \subseteq \mathrm{Mod}(\mathrm{Th}(\mathscr{C}))$.
3. $\Sigma \subseteq \mathrm{Th}(\mathrm{Mod}(\Sigma))$.
4. $\mathrm{Th}(\mathscr{C}) = \mathrm{Th}(\mathrm{Mod}(\mathrm{Th}(\mathscr{C})))$.
5. $\mathrm{Mod}(\Sigma) = \mathrm{Mod}(\mathrm{Th}(\mathrm{Mod}(\Sigma)))$.
6. $\mathrm{Mod}(\Sigma) = \bigcap_{\sigma \in \Sigma} \mathrm{Mod}(\sigma)$.
7. $\mathrm{Th}(\mathscr{C}) = \bigcap_{\mathfrak{A} \in \mathscr{C}} \mathrm{Th}(\mathfrak{A})$.

Proof. Exercise. □