Model Theory: Logical Languages

Logical languages have a **syntax** and a **semantics**. The syntax tells us which sequences of symbols are well-formed and can be interpreted. The semantics tells us how to interpret them. The interpretation of a logical language is usually one of the Boolean values `True` or `False`. However, there are also *weighted* logical languages in which case the interpretation of a logical language can be just about anything: whole numbers, real numbers, probabilities, negative log likelihoods, strings, languages, basically elements of any **semiring**. For now, we will interpret sentences of our logical languages as `True` or `False` for simplicity.

We will look at three logical languages, in the following order.

1. propositional logic
2. monadic-second order (MSO) logic, and
3. first-order (FO) logic.

For now, the relational models we studied before will become relevant for FO and MSO logic. For each of these logical languages, the syntax is defined *recursively* and the semantics is determined *compositionally* on the basis of the syntactic structure.

Propositional Logic Propositional logic is also sometimes called *Boolean logic.* The syntax of this logical language is defined as follows. Assume a countable set of atomic propositions $\mathbb{P} = \{p_1, p_2, \ldots\}$ We often use letters such as p, q, and r to indicate atomic propositions from this set.

Syntax Each $p \in P$ is a sentence of propositional logic. If α, β are sentences of propositional logic, then so are $\neg\alpha$ and $(\alpha \wedge \beta)$. The set of sentences of propositional logic are denoted $\text{PROP}(\mathbb{P})$.

Semantics A **valuation** is a total function from $v : \mathbb{P} \to \{\texttt{True}, \texttt{False}\}$. For each sentence of propositional logic φ, the interpretation of φ according to v, denoted $[\![\varphi]\!](v)$, is determined according to one of the following three cases.

1. There exists $p \in \mathbb{P}$ such that $\varphi = p$. Then $[\![\varphi]\!](v) = [\![p]\!](v) = v(p)$.
2. There exists $\alpha \in \text{PROP}(\mathbb{P})$ such that $\varphi = \neg\alpha$. Then $[\![\varphi]\!](v) = [\![\neg\alpha]\!](v) = \neg[\![\varphi]\!](v)$.
3. There exists $\alpha, \beta \in \text{PROP}(\mathbb{P})$ such that $\varphi = \alpha \wedge \beta$. Then $[\![\varphi]\!](v) = [\![\alpha \wedge \beta]\!](v) = \Big([\![\alpha]\!](v) \wedge [\![\beta]\!](v)\Big)$.

Exercise 1. Which of the following are sentences of propositional logic with $P = \{p, q, r, p_1, p_2, \ldots\}$?

1. $\neg(p \wedge q)$
2. $((p \wedge q) \wedge r)$
3. $(\neg p \wedge \neg q)$
4. $(\neg p)$
5. $p \wedge q \wedge r$
6. $p \vee q$

Exercise 2. Assume the valuation function is as follows: $v[p \mapsto \texttt{True}, q \mapsto \texttt{False}, r \mapsto \texttt{True}]$. (This means $v(p) = \texttt{True}$, $v(q) = \texttt{False}$, and $v(r) = \texttt{True}$.) For each φ in #1-3 in Exercise 1, what is $[\![\varphi]\!](v)$?

This is a 'bare bones' approach to propositional logic. Mathematically, it is valuable because it limits the number of cases in the proofs to one that are absolutely essential. Of course we can define propositional logic to include other logical connectives such as $\vee$ (disjunction), $\rightarrow$ (implication), and $\leftrightarrow$ (biconditional). However, they can all be derived from negation ($\neg$) conjunction ($\wedge$).

Exercise 3. Which logical connective is expressed by the sentence in Exercise 1 #3?

In practice, we often violate the simple but strict syntax defined above and write sentences such as those in #4-6 in Exercise 1.

MSO and FO Logic Both MSO and FO logic use *quantification* and *variables*. MSO makes us of two kinds of variables: variables that range over individual elements of the domain and variables that range over sets of individual elements of the domain. The former are denoted with lowercase letters such as x, y, z and the latter with uppercase letters X, Y, Z. While MSO uses both kinds of variables, FO logic only uses the former. That is the difference between the two logics. Formulas of FO logic are literally those formulas of MSO logic *without* quantificaton over sets of individual elements of the domain (so without variables like X, Y, Z).

MSO and FO logical languages depend on a model signature. Once we have a model signature (which recall gives us representations of objects belonging to some class), we can employ the recipes for syntax and semantics below to define logical languages for these model signatures. In what follows, I provide a recipe for MSO and FO logical languages for purely relational models (so signatures without functions).

Syntax of MSO Logic First, we establish symbols denoting variables. Consider two countable sets, $\{x_0, x_1, \ldots\}$ and $X \in \{X_0, X_1, \ldots\}$, which will ultimately be interpreted as variables over individual elements in a domain and sets of individuals in the domain, respectively.

Next consider an arbitrary relational model signature. So the signature looks something like this: $\mathbb{M} = \langle \mathcal{D}; R_1, R_2, \ldots R_m \rangle$. For all variables $x, y \in \{x_0, x_1, \ldots\}$, $X \in \{X_0, X_1, \ldots\}$, **formulas** of MSO logic over the relational model signature $\mathbb{M}$ include the following base cases.

• $x = y$	(equality)
• $x \in X$	(membership)
• $R(x_1, \ldots x_n)$ for each n-ary relation $R \in \mathbb{M}$	(atomic relational formulas)

Next we have the inductive cases. If α, β are formulas of MSO logic, then so are

• $\neg\,\alpha$	(negation)
• $(\alpha \wedge \beta)$	(conjunction)
• $\forall x\,[\alpha]$	(universal quantification for individuals)
• $\forall X\,[\alpha]$	(universal quantification for sets of individuals)

Nothing else is a formula of MSO logic. We denote this logical language MSO($\mathbb{M}$).

Convenient notation As before, all the other logical connectives ($\vee, \rightarrow, \leftrightarrow$) can be derived from negation ($\neg$) and conjunction ($\wedge$). Inequality ($\neq$) can be derived from negation and equality. Finally, existential quantification is derived from universal quantification: $\exists x[\varphi] \overset{\text{def}}{=} \neg\forall x[\neg\varphi]$. Existential quantification over sets of individuals is done similarly.

Formulas of FO Logic The logical language FO($\mathbb{M}$) is defined as all formulas of MSO($\mathbb{M}$) which do not contain quantification over sets of indviduals.

Sentences of MSO and FO logic The *free* variables of a formula φ are those variables in φ that are not quantified. Variables that are not free are called *bound*. A formula is a *sentence* if none of its variables are *free*. Only sentences can be interpreted.

Exercise 4. Consider the formulas below from FO($\mathbb{M}^{\triangleleft}$). Which of them are sentences? For the formulas that are not sentences, which variables are free and which are bound?

1. $\neg\exists y\ (x \triangleleft y)$
2. $\neg\exists y\ (y \triangleleft x)$
3. $\neg\exists x, y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \triangleleft y)$
4. $\forall x \exists y\ (\mathtt{n}(x) \rightarrow (\mathtt{t}(y) \wedge x \triangleleft y))$
5. $\mathtt{n}(x) \vee \mathtt{m}(x) \vee \mathtt{ŋ}(x)$
6. $\exists y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \triangleleft y)$
7. $\exists x, y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \triangleleft y)$
8. $\forall x, y\ (\ \neg(\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \triangleleft y))$
9. $\neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z)$
10. $\exists x, y, z\ (\neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z))$
11. $x \triangleleft y \wedge y \triangleleft z$

Semantics of MSO logic How do we evaluate sentences belonging to MSO($\mathbb{M}$) and FO($\mathbb{M}$) logic? In order to examine expressions of equality, such as $(x = y)$ and $(x \in X)$, the variables need to be assigned values. Also in order to evaluate an expression like $R(x_1, \ldots x_n)$, there needs to be a model M whose relational structure accords to the signature $\mathbb{M}$.

Therefore, we interpret a sentence φ with respect to a model M and a **variable assignment function** v which assigns values to the variables in φ. We will treat the interpretation of φ, denoted $[\![\varphi]\!]$, as a function, which takes two arguments M and v. In other words, given M and v, we are interested in evaluating $[\![\varphi]\!](M, v)$.

The function v is a partial function because not all variables may have an assignment. If we want to assign variable x to some element e in the domain $\mathcal{D}$ of M (written $[x \mapsto e]$), we write $v \leftarrow [x \mapsto e]$.

Example 1. Suppose currently v maps x_1 to e_1 and x_2 to e_2 and no other variable has an assignment. We indicate this by writing $v[x_1 \mapsto e_1, x_2 \mapsto e_2]$. Now suppose we want to add an assignment, say variable x_3 to element e_3. We can write this as follows.

$$v[x_1 \mapsto e_1, x_2 \mapsto e_2] \leftarrow [x_3 \mapsto e_3] = v[x_1 \mapsto e_1, x_2 \mapsto e_2, x_3 \mapsto e_3]$$

We can also use the same notation to *reassign* a variable to another domain element. However here we assume without loss of generality that every variable introduced by quantification is distinct from every other one, which avoids this issue. Finally, let v_0 denote empty function v (so no variables have an assignment). When variables assignments are clear from context, we just write v instead of $v[x_1 \mapsto e_1, \ldots x_n \mapsto e_n]$.

The base cases are as follows. For all models M, variables $x, y \in \{x_0, x_1, \ldots\}$, $X \in \{X_0, X_1, \ldots\}$, and n-ary relations R in the model signature $\mathbb{M}$, and sentences $\varphi \in$ MSO($\mathbb{M}$), we have:

- $[\![x = y]\!](M, v[x \mapsto e_1, y \mapsto e_2]) \quad = \texttt{True} \quad \leftrightarrow \quad e_1 = e_2$
- $[\![x \in X]\!](M, v[x \mapsto e, X \mapsto S]) \quad = \texttt{True} \quad \leftrightarrow \quad e \in S$
- $\left[\!\!\left[R(x_1, \dots x_n)\right]\!\!\right]\Big(M, v[x_1 \mapsto e_1, \dots x_n \mapsto e_n]\Big) \quad = \texttt{True} \quad \leftrightarrow \quad (e_1, \dots e_n) \in R$

The inductive cases are as follow.

- $[\![\neg\varphi]\!](M, v) \quad \leftrightarrow \quad \neg[\![\varphi]\!](M, v)$
- $\left[\!\!\left[(\varphi \wedge \psi)\right]\!\!\right](M, v) \quad \leftrightarrow \quad [\![\varphi]\!](M, v) \wedge [\![\psi]\!](M, v)$
- $\left[\!\!\left[\forall x[\varphi]\right]\!\!\right](M, v) \quad \leftrightarrow \quad \bigwedge_{e \in \mathcal{D}} \Big([\![\varphi]\!](M, v \leftarrow [x \mapsto e])\Big)$
- $\left[\!\!\left[\forall X[\varphi]\right]\!\!\right](M, v) \quad \leftrightarrow \quad \bigwedge_{S \subseteq \mathcal{D}} \Big([\![\varphi]\!](M, v \leftarrow [X \mapsto S])\Big)$

If $[\![\varphi]\!](M, v_0) = \texttt{True}$, we say M **satisfies**, or **models** $\varphi \in$, which we write $M \models \varphi$. Otherwise we write $M \not\models \varphi$.

Exercise 5. Assume the model signature $\mathbb{M}^{\triangleleft}$ and consider the logical language FO($\mathbb{M}^{\triangleleft}$). Evaluate the sentences in #3,4 in Exercise 4 with respect to the model of the word *sent*.

References

[1] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2nd edition, 2001.

[2] Shawn Hedman. *A First Course in Logic*. Oxford University Press, 2004.

[3] H. Jerome Keisler and Joel Robbin. *Mathematical Logic and Computability*. McGraw-Hill, 1996.