**The lecture 7**

**The first-order logic**

We begin this section by specifying more precisely the way in which the possible worlds of first-order logic reflect the ontological commitment to objects and relations. Then we introduce the various elements of the language, explaining their semantics as we go along.

**Models for first-order logic**

Each model links the vocabulary of the logical sentences to elements of the possible world, so that the truth of any sentence can be determined. Thus, models for propositional logic link proposition symbols to predefined truth values. Models for first-order logic are much more interesting. First, they have objects in them. The **domain** of a model is the set of objects or **domain elements** it contains. The domain is required to be *nonempty*—every possible world must contain at least one object. Mathematically speaking, it doesn’t matter *what* these objects are—all that matters is *how many* there are in each particular model—but for pedagogical purposes we’ll use a concrete example. The figure shows a model with five objects: Richard the Lionheart, King of England from 1189 to 1199; his younger brother, the evil King John, who ruled from 1199 to 1215; the left legs of Richard and John; and a crown. The objects in the model may be *related* in various ways. In the figure, Richard and John are brothers. Formally speaking, a relation is just the set of **tuples** of objects that are related. (A tuple is a collection of objects arranged in a fixed order and is written with angle brackets surrounding the objects.) Thus, the brotherhood relation in this model is the set {<Richard the Lionheart, King John>, <King John, Richard the Lionheart>}

(Here we have named the objects in English, but you may, if you wish, mentally substitute the pictures for the names.) The crown is on King John’s head, so the “on head” relation contains just one tuple, <the crown, King John>. The “brother” and “on head” relations are binary relations—that is, they relate pairs of objects. The model also contains unary relations, or properties: the “person” property is true of both Richard and John; the “king” property is true only of John (presumably because Richard is dead at this point); and the “crown” property is true only of the crown.

Certain kinds of relationships are best considered as functions, in that a given object must be related to exactly one object in this way. For example, each person has one left leg, so the model has a unary “left leg” function that includes the following mappings:

**<Richard the Lionheart> → Richard’s left leg**

**<King John> → John’s left leg.**

Strictly speaking, models in first-order logic require total functions, that is, there must be a

value for every input tuple. Thus, the crown must have a left leg and so must each of the left

legs. There is a technical solution to this awkward problem involving an additional “invisible” object that is the left leg of everything that has no left leg, including itself. Fortunately, as long as one makes no assertions about the left legs of things that have no left legs, these

technicalities are of no import.

So far, we have described the elements that populate models for first-order logic. The

other essential part of a model is the link between those elements and the vocabulary of the

logical sentences, which we explain next.



**Symbols and interpretations**

We turn now to the syntax of first-order logic. The impatient reader can obtain a complete

description from the formal grammar. The basic syntactic elements of first-order logic are the symbols that stand for objects, relations, and functions. The symbols, therefore, come in three kinds: **constant symbols**, which stand for objects; **predicate symbols**, which stand for relations; and **function symbols**, which stand for functions. We adopt the convention that these symbols will begin with uppercase letters. For example, we might use the constant symbols Richard and John; the predicate symbols Brother, OnHead, Person, King, and Crown; and the function symbol LeftLeg. As with proposition symbols, the choice of names is entirely up to the user. Each predicate and function symbol come with an **arity** that fixes the number of arguments.

As in propositional logic, every model must provide the information required to determine

if any given sentence is true or false. Thus, in addition to its objects, relations, and functions, each model includes an **interpretation** that specifies exactly which objects, relations and functions are referred to by the constant, predicate, and function symbols. One possible interpretation for our example—which a logician would call the **intended interpretation**—is as follows:

• Richard refers to Richard the Lionheart and John refers to the evil King John.

• Brother refers to the brotherhood relation, that is, the set of tuples. OnHead refers to the “on head” relation that holds between the crown and King John; Person, King, and Crown refer to the sets of objects that are persons, kings, and crowns.

• LeftLeg refers to the “left leg” function

There are many other possible interpretations, of course. For example, one interpretation maps Richard to the crown and John to King John’s left leg. There are five objects in the model, so there are 25 possible interpretations just for the constant symbols Richard and John. Notice that not all the objects need have a name—for example, the intended interpretation does not name the crown or the legs. It is also possible for an object to have several names; there is an interpretation under which both Richard and John refer to the crown. If you find this possibility confusing, remember that, in propositional logic, it is perfectly possible to have a model in which Cloudy and Sunny are both true; it is the job of the knowledge base to rule out models that are inconsistent with our knowledge.





In summary, a model in first-order logic consists of a set of objects and an interpretation that maps constant symbols to objects, predicate symbols to relations on those objects, and function symbols to functions on those objects. Just as with propositional logic, entailment, validity, and so on are defined in terms of *all possible models*. To get an idea of what the set of all possible models looks like, see Figure 8.4. It shows that models vary in how many objects they contain—from one up to infinity—and in the way the constant symbols map to objects. If there are two constant symbols and one object, then both symbols must refer to the same object; but this can still happen even with more objects. When there are more objects than constant symbols, some of the objects will have no names. Because the number of possible models is unbounded, checking entailment by the enumeration of all possible models is not feasible for first-order logic (unlike propositional logic). Even if the number of objects is restricted, the number of combinations can be very large.

**Terms**

A **term** is a logical expression that refers to an object. Constant symbols are therefore terms, but it is not always convenient to have a distinct symbol to name every object. For example,

in English we might use the expression “King John’s left leg” rather than giving a name to his leg. This is what function symbols are for: instead of using a constant symbol, we use LeftLeg(John). In the general case, a complex term is formed by a function symbol followed by a parenthesized list of terms as arguments to the function symbol. It is important to remember that a complex term is just a complicated kind of name. It is not a “subroutine call” that “returns a value.” There is no LeftLeg subroutine that takes a person as input and returns a leg. We can reason about left legs (e.g., stating the general rule that everyone has one and then deducing that John must have one) without ever providing a definition of LeftLeg. The formal semantics of terms is straightforward. Consider a term f(t1, . . . , tn). The function symbol f refers to some function in the model (call it F); the argument terms refer to objects in the domain (call them d1, . . . , dn); and the term as a whole refers to the object that is the value of the function F applied to d1, . . . , dn. For example, suppose the LeftLeg function symbol refers to the function shown in Equation and John refers to King John, then LeftLeg(John) refers to King John’s left leg. In this way, the interpretation fixes the referent of every term.

**Atomic sentences**

Now that we have both terms for referring to objects and predicate symbols for referring to

relations, we can put them together to make **atomic sentences** that state facts. An **atomic sentence** (or **atom** for short) is formed from a predicate symbol optionally followed by a

parenthesized list of terms, such as

Brother (Richard, John).

This states, under the intended interpretation given earlier, that Richard the Lionheart is the

brother of King John.6 Atomic sentences can have complex terms as arguments. Thus,

Married(Father (Richard),Mother (John))

states that Richard the Lionheart’s father is married to King John’s mother (again, under a

suitable interpretation).

*An atomic sentence is* ***true*** *in a given model if the relation referred to by the predicate symbol holds among the objects referred to by the arguments.*

**Complex sentences**

We can use **logical connectives** to construct more complex sentences, with the same syntax

and semantics as in propositional calculus. Here are four sentences that are true in the model

under our intended interpretation:

￢Brother (LeftLeg(Richard), John)

Brother (Richard, John) ∧ Brother (John,Richard)

King(Richard ) ∨ King(John)

￢King(Richard) ⇒ King(John).