SYNTACTIC THEORY
STRINGS

We will concentrate on what is called a **syntactic theory** of mental objects.[11] In this approach, we represent mental objects with **strings** written in a representation language. (We will use first-order logic itself as the representation language, but we are not required to.) A string is just a list of symbols, so the event $Flies(Clark)$ can be represented by the string of characters $[F, l, i, e, s, (, C, l, a, r, k,)]$, which we will abbreviate as "$Flies(Clark)$". In this formulation, "$Clark$" $\neq$ "$Superman$" because they are two different strings consisting of different symbols. The idea is that a knowledge-based agent has a knowledge base consisting of strings that were added either via TELL or through inference. The syntactic theory models the knowledge base and the strings that are in it.

Now all we have to do is provide a syntax, semantics, and proof theory for the string representation language, just as we did in Chapter 6. The difference is that we have to define them all in first-order logic. We start by defining Den as the function that maps a string to the object that it denotes, and $Name$ as a function that maps an object to a string that is the name of a constant that denotes the object. For example, the denotation of both "$Clark$" and "$Superman$" is the object referred to by the constant symbol $ManOfSteel$, and the name of that object could be either "$Superman$", "$Clark$", or some other constant, such as "K_{11}".

$$Den(\text{“}Clark\text{”}) = ManOfSteel \land Den(\text{“}Superman\text{”}) = ManOfSteel$$
$$Name(ManOfSteel) = \text{“}K_{11}\text{”}$$

The next step is to define inference rules for logical agents. For example, we might want to say that a logical agent can do Modus Ponens: if it believes p and believes $p \Rightarrow q$ then it will also believe q. The first attempt at writing this axiom is

$$\forall\, a, p, q \;\; LogicalAgent(a) \land Believes(a, p) \land Believes(a, \text{“}p \Rightarrow q\text{”}) \;\Rightarrow\; Believes(a, q)$$

But this is not right because the string "$p \Rightarrow q$" contains the letters 'p' and 'q' but has nothing to do with the strings that are the values of the variables p and q. In fact, "$p \Rightarrow q$" is not even a syntactically correct sentence, because only variables can be lower-case letters. The correct formulation is:

$$\forall\, a, p, q \;\; LogicalAgent(a) \land Believes(a, p) \land Believes(a, Concat(p, \text{“}{\Rightarrow}\text{”}, q)$$
$$\Rightarrow\; Believes(a, q)$$

where $Concat$ is a function on strings that concatenates their elements together. We will abbreviate $Concat(p, \text{“}{\Rightarrow}\text{”}, q)$ as "$\underline{p} \Rightarrow \underline{q}$". That is, an occurrence of $\underline{x}$ within a string means to substitute in the value of the variable x. Lisp programmers will recognize this as the backquote operator.

Once we add in the other inference rules besides Modus Ponens, we will be able to answer questions of the form "given that a logical agent knows these premises, can it draw that conclusion?" Besides the normal inference rules, we need some rules that are specific to belief. For example, the following rule says that if a logical agent believes something, then it believes that it believes it.

$$\forall\, a, p \;\; LogicalAgent(a) \land Believes(a, p) \;\Rightarrow\; Believes(a, \text{“}Believes(\underline{Name(a)}, \underline{p})\text{”})$$

Note that it would not do to have just $\underline{a}$ as part of the string, because a is an agent, not a description of an agent. We use $\underline{Name(a)}$ to get a string that names the agent.

[11] An alternative based on **modal logic** is covered in the historical notes section.