Existential Instantiation

Similarly, the rule of **Existential Instantiation** replaces an existentially quantified variable with a single *new constant symbol*. The formal statement is as follows: for any sentence $\alpha$, variable $v$, and constant symbol $k$ that does not appear elsewhere in the knowledge base,

$$\frac{\exists v \ \alpha}{\text{SUBST}(\{v/k\},\alpha)}.$$

For example, from the sentence

$$\exists x \ \ Crown(x) \wedge OnHead(x, John)$$

we can infer the sentence

$$Crown(C_1) \wedge OnHead(C_1, John)$$

as long as $C_1$ does not appear elsewhere in the knowledge base. Basically, the existential sentence says there is some object satisfying a condition, and applying the existential instantiation rule just gives a name to that object. Of course, that name must not already belong to another object. Mathematics provides a nice example: suppose we discover that there is a number that is a little bigger than 2.71828 and that satisfies the equation $d(x^y)/dy = x^y$ for $x$. We can give this number the name $e$, but it would be a mistake to give it the name of an existing object, such as $\pi$. In logic, the new name is called a **Skolem constant**.

Skolem constant

Whereas Universal Instantiation can be applied many times to the same axiom to produce many different consequences, Existential Instantiation need only be applied once, and then the existentially quantified sentence can be discarded. For example, we no longer need $\exists x \ Kill(x, Victim)$ once we have added the sentence $Kill(Murderer, Victim)$.

### 9.1.1 Reduction to propositional inference

We now show how to convert any first-order knowledge base into a propositional knowledge base. The first idea is that, just as an existentially quantified sentence can be replaced by one instantiation, a universally quantified sentence can be replaced by the set of *all possible* instantiations. For example, suppose our knowledge base contains just the sentences

$$\begin{array}{l} \forall x \ \ King(x) \wedge Greedy(x) \Rightarrow Evil(x) \\ King(John) \\ Greedy(John) \\ Brother(Richard, John). \end{array} \tag{9.1}$$

and that the only objects are *John* and *Richard*. We apply UI to the first sentence using all possible substitutions, $\{x/John\}$ and $\{x/Richard\}$. We obtain

$$\begin{array}{l} King(John) \wedge Greedy(John) \Rightarrow Evil(John) \\ King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard). \end{array}$$

Next replace ground atomic sentences, such as $King(John)$, with proposition symbols, such as *JohnIsKing*. Finally, apply any of the complete propositional algorithms in Chapter 7 to obtain conclusions such as *JohnIsEvil*, which is equivalent to $Evil(John)$.

Propositionalization

This technique of **propositionalization** can be made completely general, as we show in Section 9.5. However, there is a problem: when the knowledge base includes a function symbol, the set of possible ground-term substitutions is infinite! For example, if the knowledge base mentions the *Father* function, then infinitely many nested terms such as $Father(Father(Father(John)))$ can be constructed.