person, and all kings are persons:

$$\text{TELL}(KB, King(John)).$$
$$\text{TELL}(KB, Person(Richard)).$$
$$\text{TELL}(KB, \forall x \; King(x) \Rightarrow Person(x)).$$

We can ask questions of the knowledge base using ASK. For example,

$$\text{ASK}(KB, King(John))$$

returns *true*. Questions asked with ASK are called **queries** or **goals**. Generally speaking, any query that is logically entailed by the knowledge base should be answered affirmatively. For example, given the three assertions above, the query

$$\text{ASK}(KB, Person(John))$$

should also return *true*. We can ask quantified queries, such as

$$\text{ASK}(KB, \exists x \; Person(x)).$$

The answer is *true*, but this is perhaps not as helpful as we would like. It is rather like answering "Can you tell me the time?" with "Yes." If we want to know what value of x makes the sentence true, we will need a different function, which we call ASKVARS,

$$\text{ASKVARS}(KB, Person(x))$$

and which yields a stream of answers. In this case there will be two answers: $\{x/John\}$ and $\{x/Richard\}$. Such an answer is called a **substitution** or **binding list**. ASKVARS is usually reserved for knowledge bases consisting solely of Horn clauses, because in such knowledge bases every way of making the query true will bind the variables to specific values. That is not the case with first-order logic; in a *KB* that has been told only that $King(John) \vee King(Richard)$ there is no single binding to x that makes the query $\exists x \; King(x)$ true, even though the query is in fact true.

Substitution
Binding list

8.3.2 The kinship domain

The first example we consider is the domain of family relationships, or kinship. This domain includes facts such as "Elizabeth is the mother of Charles" and "Charles is the father of William" and rules such as "One's grandmother is the mother of one's parent."

Clearly, the objects in our domain are people. Unary predicates include *Male* and *Female*, among others. Kinship relations—parenthood, brotherhood, marriage, and so on—are represented by binary predicates: *Parent*, *Sibling*, *Brother*, *Sister*, *Child*, *Daughter*, *Son*, *Spouse*, *Wife*, *Husband*, *Grandparent*, *Grandchild*, *Cousin*, *Aunt*, and *Uncle*. We use functions for *Mother* and *Father*, because every person has exactly one of each of these, biologically (although we could introduce additional functions for adoptive mothers, surrogate mothers, etc.).

We can go through each function and predicate, writing down what we know in terms of the other symbols. For example, one's mother is one's parent who is female:

$$\forall m, c \; Mother(c) = m \Leftrightarrow Female(m) \wedge Parent(m, c).$$

One's husband is one's male spouse:

$$\forall w, h \; Husband(h, w) \Leftrightarrow Male(h) \wedge Spouse(h, w).$$

Parent and child are inverse relations:

$$\forall p, c \; Parent(p, c) \Leftrightarrow Child(c, p).$$