Chapter 6
Relational Logic

§6.1 Introduction

Propositional logic does a good job of allowing us to talk about relationships among propositions, and it gives us the machinery to derive logical conclusions based on these relationships. Suppose, for example, we believe that, if Jack knows Jill, then Jill knows Jack. Suppose we also believe that Jack knows Jill. From these two facts, we can conclude that Jill knows Jack using a simple application of Modus Ponens.

Unfortunately, when we want to say things more generally, we find that propositional logic is inadequate. Suppose, for example, that we wanted to say in general that, if one person knows a second person, then the second person knows the first. Here, propositional logic is inadequate; it gives us no way of encoding this more general belief.

Relational logic solves this problem by providing a finer grain of representation. In particular, it provides us with a way of talking about individual objects and their interrelationships. It allows us to assert the existence of objects with a given relationship and allows us to talk about all objects having a given relationship and to apply those facts to specific cases.

From a representational perspective, relational logic is more intuitive than propositional logic. Unfortunately, for beginners, the details are a little more complicated; and so we proceed more slowly than before. We start with a discussion of the notions of object and function and relation. Only then do we proceed as before, first looking at syntax and then at semantics.

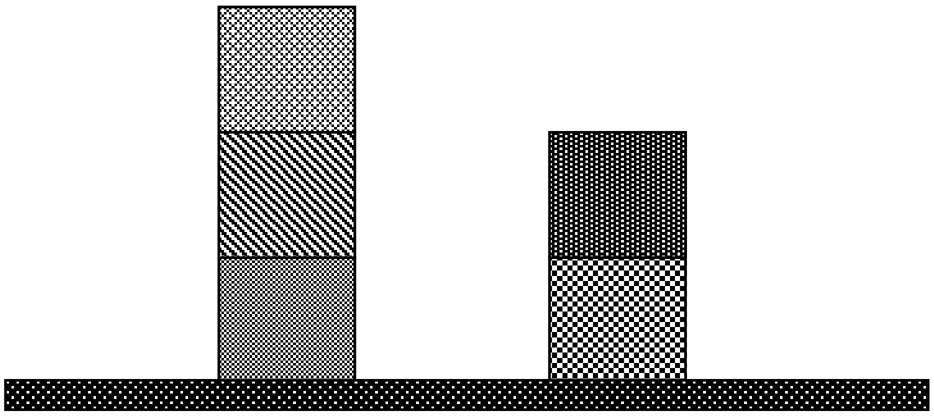
§6.2 Conceptualization

As we shall see, the semantics of relational logic is based on the notion of a *conceptualization* of the world. This includes the objects presumed or hypothesized to exist in the world and their interrelationships.

The notion of an *object* used here is quite broad. Objects can be concrete (e.g. this book, Confucius, the sun) or abstract (e.g. the number 2, the set of all integers, the concept of justice). Objects can be primitive or composite (e.g. a circuit that consists of many subcircuits). Objects can even be fictional (e.g. a unicorn, Sherlock Holmes, Miss Right). In short, an object can be anything about which we want to say something.

Not all knowledge-representation tasks require that we consider all the objects in the world; in some cases, only those objects in a particular set are relevant. For example, number theorists usually are concerned with the properties of numbers and usually are not concerned with physical objects such as resistors and transistors. Electrical engineers usually are concerned with resistors and transistors and usually are not concerned with buildings and bridges. The set of objects about which knowledge is being expressed is often called a *universe of discourse.*

As an example, consider the Blocks World scene in Figure n. Most people looking at this figure interpret it as a configuration of toy blocks. Some people conceptualize the table on which the blocks are resting as an object as well; but, for simplicity, we ignore it here. The universe of discourse corresponding to this conceptualization is the set consisting of the five blocks in the scene.



Although in this example there are finitely many elements in our universe of discourse, this need not always be the case. It is common in mathematics, for example, to consider the set of all integers or the set of all real numbers or the set of all n-tuples of real numbers, as universes with infinitely many elements.

An interrelationship among the objects in a universe of discourse is called a *relation*. Although we can invent many relations for a set of objects, in conceptualizing a portion of the world we usually emphasize some relations and ignore others. The set of relations emphasized in a conceptualization is called the *relational basis set*.

In a spatial conceptualization of the Blocks World, there are numerous meaningful relations. For example, it makes sense to think about the *on* relation that holds between two blocks if and only if one is immediately above the other. We might also think about the *above* relation that holds between two blocks if and only if one is anywhere above the other. The *stack* relation holds of three blocks that are stacked one on top of the other. The *clear* relation holds of a block if and only if there is no block on top of it. The *bottom* relation holds of a block if and only if that block is resting on the table.

One way of giving meaning to these intuitive notions in the context of a specific conceptualization of the world is to list the combinations of objects in the universe of discourse that satisfy each relation in the relational basis set.

For example, the *on* relation can be characterized as a set of all pairs of blocks that are *on*-related to each other. If we use the symbols a, b, c, d, and e to stand for the blocks in the scene shown above, then the table shown below encodes the *on* relation. There is a separate row in the table for each pair of blocks in which the first element is on the second.

a	b
b	c
d	e

The *above* relation can be characterized by a table similar to that for the *on* relation.

In this case, there is an additional row, since a block can be above another without being directly on top of it.

a	b
a	c
b	c
d	e

The *clear* relation is a property of a block in of itself, not with respect to other blocks. In this case, the table has just one column, and each row represents a block that is clear of other blocks.

a
d

The *bottom* relation, like the *clear* relation is a property of a block in of itself; and so its table also has just one column.

c
e

The *stack* relation is a relationship among three blocks; and so the table requires three columns, as shown below. In this case, there is just one stack in the scene, and so there is just one row in the table.

a	b	c

The generality of relations can be determined by comparing their elements. Thus, the *on* relation is less general than the *above* relation since, when viewed as a set of tuples, it is a subset of the *above* relation. Of course, some relations are empty (e.g. the *unsupported* relation), whereas others consist of all n-tuples over the universe of discourse (e.g. the *block* relation).

Logic allows us to capture the contents of such tables; and, more importantly, it allows us to assert relationships between these tables. For example, it allows us to say that, if one block is *on* a second block, then that block is *above* the second block. Also, the first block is not on the table. Also, the second block in not clear. Moreover, it allows us to say these things *general*, i.e. out of the context of any particular arrangement of blocks.

One thing that our tabular representation for relations makes clear is that, for a finite universe of discourse, there is an upper bound on the number of possible n-ary relations. In particular, for a universe of discourse of size b, there are b^n distinct n-tuples. Every n-ary relation is a subset of these b^n tuples. Therefore, an n-ary relation must be one of at most $2^{(b^n)}$ possible sets.

A *function* is a special kind of relation among the objects in a universe of discourse. For each combination of n objects in a universe of discourse (called the *arguments*, a function associates a single object (called the *value* of the function applied to the arguments). Note that there must be one and exactly one value for each combination of arguments. Functions that are not defined for all combinations of arguments are often called *partial functions*; and functions for which there can be more than value for a given combination of arguments are often called *multi-valued functions.* By and large, we will ignore partial and multi-valued functions in what follows.

Mathematically, a function can be viewed as a set of tuples of objects, one for each combination of possible arguments paired with the corresponding value. In other words, an n-ary function is a special kind of $n+1$-ary relation.

For example, consider the unary function shown on the left below. We can conceptualize this function as a binary relation shown on the right.

$$\begin{array}{ccc} a & \rightarrow & b \\ b & \rightarrow & c \\ c & \rightarrow & d \\ d & \rightarrow & e \\ e & \rightarrow & a \end{array}$$

a	b
b	c
c	d
d	e
e	a

In an analogous way, a binary function can be treated as a ternary relation; a ternary function can be treated as a 4-ary relation; and so forth.

Although the tabular representation for functions and relations is intuitively satisfying, it consumes a lot of space. Therefore, in what follows, we use a different way of encoding relational tables, viz. as sets of tuples, each tuple representing a single row of the corresponding table.

As an example, consider the *on* relation shown earlier. In what follows, we will treat the relation as the set of three 2-tuples shown below.

$$\{\langle a, b\rangle, \langle b, c\rangle, \langle d, e\rangle\}$$

Since functions are relations, we could use the same representation. However, to emphasize the special properties of functions, we use a slight variation, viz. the use of an arrow in each tuple to emphasize the fact that the last element is the value of the function applied to the element or elements before the arrow.

As an example, consider the *next* function shown above. In this case, we would write the function as shown below.

$$\{\langle a \rightarrow b\rangle, \langle b \rightarrow c\rangle, \langle c \rightarrow d\rangle, \langle d \rightarrow e\rangle, \langle e \rightarrow a\rangle\}$$

Finally, before we get on with the details of relational logic, it is worthwhile to make a few remarks about conceptualization. No matter how we choose to conceptualize the world, it is important to realize that there are other conceptualizations as well. Furthermore, there need not be any correspondence between the objects, functions, and relations in one conceptualization and the objects, functions, and relations in another.

In some cases, changing one's conceptualization of the world can make it impossible to express certain kinds of knowledge. A famous example of this is the controversy in the field of physics between the view of light as a wave phenomenon and the view of light in terms of particles. Each conceptualization allowed physicists to explain different aspects of the behavior of light, but neither alone sufficed. Not until the two views were merged in modern quantum physics were the discrepancies resolved.

In other cases, changing one's conceptualization can make it more difficult to express knowledge, without necessarily making it impossible. A good example of this, once again in the field of physics, is changing one's frame of reference. Given Aristotle's geocentric view of the universe, astronomers had great difficulty explaining the motions of the moon and other planets. The data were explained (with epicycles, etc.) in the Aristotelian conceptualization, although the explanation was extremely cumbersome. The switch to a heliocentric view quickly led to a more perspicuous theory.

This raises the question of what makes one conceptualization more appropriate than another for knowledge formalization. Currently, there is no comprehensive answer to this question. However, there are a few issues that are especially noteworthy.

One such issue is the *grain size* of the objects associated with a conceptualization. Choosing too small a grain can make knowledge formalization prohibitively tedious. Choosing too large a grain can make it impossible. As an example of the former problem, consider a conceptualization of the scene in Figure n in which the objects in the universe of discourse are the atoms composing the blocks in the picture. Each block is composed of enormously many atoms, so the universe of discourse is extremely large. Although it is in principle possible to describe the scene at this level of detail, it is senseless if we are interested in only the vertical relationship of the blocks made up of those atoms. Of course, for a chemist interested in the composition of blocks, the atomic view of the scene might be more appropriate. For this purpose, our conceptualization in terms of blocks has too large a grain.

Finally, there is the issue of *reification* of functions and relations as objects in the universe of discourse. The advantage of this is that it allows us to consider properties of properties. As an example, consider a Blocks World conceptualization in which there are five blocks, no functions, and three unary relations, each corresponding to a different color. This conceptualization allows us to consider the colors of blocks but not the properties of those colors.

We can remedy this deficiency by *reifying* various color relations as objects in their own right and by adding a partial function – such as *color* – to relate blocks to colors. Because the colors are objects in the universe of discourse, we can then add relations that characterize them, e.g. *nice*.

Note that, in this discussion, no attention has been paid to the question of whether the objects in one's conceptualization of the world really exist. We have adopted neither the standpoint of *realism*, which posits that the objects in one's conceptualization really exist, nor that of *nominalism*, which holds that one's concepts have no necessary external existence. Conceptualizations are our inventions, and their justification is based solely on their utility. This lack of commitment indicates the essential ontological promiscuity of logic: Any conceptualization of the world is accommodated, and we seek those that are useful for our purposes.

§6.3 Syntax

The vocabulary of relational logic is slightly different from that of propositional logic. There are no logical constants; and, in their place, we have two new classes of words – *variables* and *constants*.

By convention, variables begin with letters from the end of the alphabet, viz. u, v, w, x, y, z. In some cases, we use subscripts to differentiate variables, e.g. x_1, x_2, and so forth.

By convention, all constants begin with either upper case characters (other than u, v, w, x, y, z), mathematical characters (+, −, etc.), or digits. Again, we sometimes use subscripts to differentiate different constants, e.g. a_1, a_2, and so forth.

Constants are further subdivided into *object constants*, *function constants*, and *relation constants*. Each function constant or relation constant has an associated positive integer, called its *arity*, which determines how many *arguments* it accepts. Note that there is no inherent syntactic distinction between object constants, function constants, and relation constants. The type of each such word is determined by its usage or, in some cases, in an informal vocabulary specification.

As we shall see, function constants and relation constants are used in forming complex expressions by combining them with an appropriate number of "arguments". Accordingly, each function and relation constant has an associated *arity*, i.e. a number indicating the number of arguments to which that function or relation can be applied. A function or relation constant that can applied to a single argument is said to be it unary; one that applies to two arguments is said to be *binary*; more generally, a function or relation constant that applies to n arguments is said to be n-ary.

A *functional term* is an expression formed from an n-ary function constant and an appropriate number of terms enclosed in parentheses and separated by commas. For example, if f is a function constant with arity n and if t_1, ... t_n are terms, then $f(t_1, ..., t_n)$ is a functional term.

Functional terms can occur inside of other functional terms. For example, if a and b are object constants, f is a function constant with arity 2, and if g is a function constant with arity 1, then the expression $f(g(a), b)$ is a properly formed functional term, as is the expression $g(f(a, b))$.

We use the word *term* to refer to any variable, object constant, or functional term. As we shall see, terms refer to objects and, as such, are analogous to noun phrases of natural language.

There are three types of *sentences* in relational logic, viz. relational sentences (the analog of propositional constants in propositional logic), logical sentences (the analog of logical sentences in propositional logic), and quantified sentences (which have no analog in propositional logic).

A *relational sentence* is an expression formed from an n-ary relation constant and an appropriate number of terms. For example, if r is a relation constant with arity 3 and if a, b, and c are terms, then the expression shown below is a relational sentence.

$$r(a, b, c)$$

Remember that the arguments in a relational sentence can be terms of any sort, not just object constants. Variables and functional terms are equally acceptable.

Logical sentences are defined as in propositional logic. There are negations, conjunctions, disjunctions, implications, reductions, and equivalences. The syntax is exactly the same, except, of course, the elementary components are relational sentences rather than logical constants.

Quantified sentences are formed from a *quantifier*, a variable, and an embedded sentence. The embedded sentence is called the *scope* of the quantifier. There are two types of quantified sentences in relational logic, viz. universally quantified sentences and existentially quantified sentences.

A *universally quantified sentence* is used to assert that all objects have a certain property. For example, if p is a unary relation constant, then the following expression is a universally quantified sentence asserting that p holds of all objects in the universe of discourse.

$$\forall x.p(x)$$

An *existentially quantified sentence* is used to assert that some object has a certain property. For example, if p is a unary relation constant, then the following expression is an existentially quantified sentence asserting that there is some object in the universe of discourse for which p holds.

$$\exists x.p(x)$$

A sentence is *ground* if and only if it contains no variables. For example, the sentence $human(joe)$ is ground, whereas the sentence $\forall x.human(x)$ is not.

An occurrence of a variable is *free* if and only if it is not in the scope of a quantifier of that variable. Otherwise, it is *bound*. For example, y is free and x is bound in the following sentence.

$$\exists x.p(x, y))$$

A sentence is *open* if and only if it has free variables. Otherwise, it is *closed*. For example, the first sentence below is open and the second is closed.

$$p(y) \Rightarrow \exists x.q(x, y)$$

$$\forall y.(p(y) \Rightarrow \exists x.q(x, y))$$

As this example reveals, logical sentences and quantified sentences can be arbitrarily nested within other sentences.

§6.4 Semantics

An *interpretation* i is a mapping from the constants of the language into the elements of a conceptualization. We use the expression $\forall^i$ to refer to the universe of discourse corresponding to the conceptualization. Each object constant in our language is mapped into an element of the universe of discourse. Each function constant with arity n is mapped into an n-ary function on the universe of discourse. Each relation constant with arity n is mapped into an n-ary relation on the universe of discourse.

As an example, consider a relational language with object constants a and b, unary function constant f, and binary relation constant r.

As our universe of discourse, we take the set consisting of the three abstract objects shown below.

$$\circ \qquad \bullet \qquad \star$$

Admittedly, this is a very small universe of discourse. It is also somewhat abstract. This has the advantage that we can write out examples in their entirety. Of course, in practical situations, universes of discourse are likely to be more concrete and much larger.

The equations shown below define one interpretation of our language in terms of this universe of discourse. As we did with propositional logic, we are using informal, *metalevel* statements here. Remember that these are not themselves sentences in relational logic.

$$\begin{aligned}
&\forall^i = \{\circ, \bullet, \star\}\\
&a^i = \circ\\
&b^i = \bullet\\
&c^i = \bullet\\
&f^i = \{\langle \circ \rightarrow \bullet \rangle, \langle \bullet \rightarrow \bullet \rangle, \langle \star \rightarrow \circ \rangle\}\\
&r^i = \{\langle \circ\, , \circ \rangle, \langle \circ\, , \bullet \rangle, \langle \bullet\, , \bullet \rangle\}
\end{aligned}$$

Note that more than one object constant can refer to the same object. Note also that not every object in the universe of discourse need have a constant that refers to it.

A *variable assignment* for a universe of discourse $\forall^i$ is a mapping from the variables of the language into $\forall^i$.

$$\begin{aligned}
x^v &= \circ\\
y^v &= \circ\\
z^v &= \bullet
\end{aligned}$$

A *value assignment* based on interpretation i and variable assignment v for $\forall^i$ is a mapping from the terms of the language into $\forall^i$. The mapping must agree with i on constants, and it must agree with v on variables. The value of a functional term is the result of applying the function corresponding to the function constant to the objects designated by the terms.

$$\begin{aligned}
a^{iv} &= \circ\\
z^{iv} &= \bullet\\
f(a)^{iv} &= \bullet\\
f(z)^{iv} &= \circ
\end{aligned}$$

The *truth assignment* based on interpretation i and variable assignment v is a mapping from the sentences of the language to true or false defined as follows.

Given an interpretation i and a variable assignment v, a relational sentence is true if and only if the tuple of objects denoted by the arguments is contained in the relation denoted by the relation constant.

For example, given the interpretation and variable assignment presented above, we have the truth assignments shown below. The relational sentence $r(a, b)$ is true since $\langle\circ, \bullet\rangle$ is a row in r^i. The sentence $r(b, z)$ is false since $\langle\bullet, \circ\rangle$ is not in r^i.

$$r(a, b)^{iv} = true$$
$$r(b, z)^{iv} = false$$

The conditions for truth of logical sentences in relational logic are analogous to those for the truth of logical sentences in propositional logic. A negation $\neg\phi$ is true if and only if ϕ is false. A conjunction $\phi \wedge \psi$ is true if and only if both ϕ and ψ are true. A disjunction $\phi \wedge \psi$ is true if and only if ϕ is true or ψ is true (or both). An implication is true unless the antecedent is true and the consequent is false. A reduction is true unless the antecedent is true and the consequent is false. An equivalence is true if and only if the truth values of the two embedded sentences are the same.

Intuitively, a universally quantified sentence is satisfied if and only if the embedded sentence is satisfied for all values of the quantified variable. Intuitively, an existentially quantified sentence is satisfied if and only if the embedded sentence is satisfied for some values of the quantified variable.

Sadly, expressing this intuition rigorously is a little tricky. First, we need the notion of a "version" of a variable assignment. Then, we can give a rigorous definition to our intuition by talking about some variations or all variations of a given variable assignment.

A *version* $v : \nu/x$ of a variable assignment v is a variable assignment that assigns x to ν and agrees with v on all other variables.

Suppose, for example, we had the variable assignment v shown below.

$$x^v = \circ$$
$$y^v = \circ$$
$$z^v = \bullet$$

The variation $v : y/\star$ is shown below.

$$x^v = \circ$$
$$y^v = \star$$
$$z^v = \bullet$$

With this notion, we can formalize the semantics of quantified sentences as follows. An interpretation i and variable assignment v satisfy a universally quantified sentence $\forall\nu.\phi$ if and only if i and v satisy ϕ for all versions of v that can be defined over the same universe of discourse. An interpretation i and variable assignment v satisfy an existentially quantified sentence $\exists\nu.\phi$ if and only if i and v satisy ϕ for at least one version of v that can be defined over the same universe of discourse.

An interpretation i and variable assignment v are said to *satisfy* a sentence ϕ (written $\models_i \phi[v]$) if and only if the sentence is assigned the value *true*.

A sentence is *satisfiable* if and only if there is an interpretation and a variable assignment that satisfy it. A sentence is *valid* if and only if it is satisfied by *every* interpretation and variable assignment. A sentence is *unsatisfiable* if and only if it is satisfied by *no* interpretation and variable assignment.

An interpretation i is a *model* of a sentence ϕ (written $\models_i \phi$) if and only if the interpretation satisfies the sentence for *every* variable assignment (in other words, if $\models_i \phi[v]$ for all variable assignments). Note that, if an interpretation satisfies a sentence for one variable assignment, then it satisfies the sentence for every variable assignment (i.e. it is a model). Note also that, if an interpretation i satisfies an open sentence ϕ with free variable ν for every variable assignment, then it is a model of $\forall\nu.\phi$.

§6.5 Examples

6.5.1 Unary Relations in Natural Language

As an example of using relational logic, consider the task of formalizing the information in the following English sentences. We assume that the conceptualization underlying all the sentences is the same. The universe of discourse is the set of all plants. There is a unary relation for being a mushroom, another for being purple, and a third for being poisonous. We designate these relations with the unary relation symbols *mushroom*, *purple*, and *poisonous*. Each English sentence is followed by one or more translations into the language of relational logic. Where more than one translation is given, the alternatives are logically equivalent.

All purple mushrooms are poisonous.
$\forall x.(purple(x) \wedge mushroom(x) \Rightarrow poisonous(x))$
$\forall x.(purple(x) \Rightarrow (mushroom(x) \Rightarrow poisonous(x)))$
$\forall x.(mushroom(x) \Rightarrow (purple(x) \Rightarrow poisonous(x)))$

The use of the word *all* in this sentence is a clear indication that it is universally quantified. The equivalence of the three sentences should be obvious. The first states that, if an object is a mushroom and purple, then it is poisonous. The second states that, if an object is purple, then, if it is also a mushroom, it is poisonous. The third sentence states that, if an object is a mushroom, then, if it is also purple, it is poisonous. All three statements assert the poisonous quality of any purple mushroom.

A mushroom is poisonous only if it is purple.
$\forall x.mushroom(x) \wedge poisonous(x) \Rightarrow purple(x)$
$\forall x.mushroom(x) \Rightarrow (poisonous(x) \Rightarrow purple(x))$

Here we have the converse of the relationship in the preceding sentence. The argument for equivalence is the same as for the preceding sentence. (Caution: A conceptualization of the world in which this is a true sentence may be hazardous to your health!)

No purple mushroom is poisonous.
$\forall x.\neg(purple(x) \wedge mushroom(x) \wedge poisonous(x))$
$\neg(\exists x.(purple(x) \wedge mushroom(x) \wedge poisonous(x))$

The use of the word *no* here indicates that something is not true. The fact that, for all objects, something is not true (as suggested by the first rendition) is equivalent to the lack of existence of an object for which it is true (as suggested by the second).

There is exactly one mushroom.
$\exists x.mushroom(x) \wedge (\forall z.z \neq x \Rightarrow \neg mushroom(z))$

The easiest way to encode information about the number of objects having a property is to state the cardinality of the set of all objects having that property. Although the specified conceptualization includes neither this set nor the cardinality function, it is possible to express that there is only one mushroom using the equality relation. Note that the fact can be stated even though the identity of the individual mushroom is unknown.

6.5.2 Blocks World

As another example of expressing information in relational logic, once again consider the Blocks World scene in Figure n. For our vocabulary, we use the five object constants a, b, c, d, and e to refer to the five blocks; and we use the relation constants *on*, *clear*, *table*, and *above* to refer to their interrelationships.

The sentences that follow encode the essential information about this scene. Block a is on block b; block b is on block c; and block d is on block e. Block a is above b and c; b is above c; and d is above e. Finally, block a is clear, and so is block d. Block c is on the table, and so is block e.

$$\begin{array}{llll} on(a,b) & above(a,b) & clear(a) & table(c) \\ on(b,c) & above(b,c) & clear(d) & table(e) \\ on(d,e) & above(a,c) & & \\ & above(d,e) & & \end{array}$$

In addition to encoding simple sentences, we can encode more general facts. In the Blocks World, if one block is on another, then that block is above the other. Also, if one block is on another then the first is not on the table and the second is not clear. Furthermore, the *above* relation is transitive; if one block is above a second and the second is above a third, then the first also is above the third.

$$\forall x.\forall y.(on(x,y) \Rightarrow above(x,y))$$

$$\forall x.\forall y.(on(x,y) \Rightarrow \neg table(x) \wedge \neg clear(y))$$

$$\forall x.\forall y.\forall z.(above(x,y) \wedge above(y,z) \Rightarrow above(x,z))$$

One advantage to writing such general sentences is economy. If we record *on* information for every object and encode the relationship between the *on* relation and the *above* relation, there is no need to record any *above* information explicitly.

Another advantage is that these general sentences apply to Blocks World scenes other than the one pictured here. It is possible to create a Blocks World scene in which none of the specific sentences we have listed is true, but the general sentences are still correct.

Of course, there are other true sentences one can write about the Blocks World. Many of these sentences are redundant in that they are entailed by the preceding sentences. This notion of logical entailment is defined more precisely in the next chapter.

6.5.3 Lists

If $\tau_1, \ldots, \tau_n$ are legal terms in our language, then a *list* is a term of the following form, where n is any integer greater than or equal to zero.

$$[\tau_1, \ldots, \tau_n]$$

The primary use of lists is the representation of sequences of objects. For example, if we use numerals to designate numbers, we can use the following list to designate the sequence consisting of the first three integers in ascending order.

$$[1, 2, 3]$$

Because lists are themselves terms, we can nest lists within lists. For example, the following term is a list of all permutations of the first three positive integers.

$$[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]$$

To talk about lists of arbitrary length, we use the binary functional operator "`.`" in infix form. In particular, a term of the form $\tau_1 . \tau_2$ designates a sequence in which τ_1 is the first element and τ_2 is the rest of the list. With this operator, we can rewrite the list `[1,2,3]` as follows.

$$(1.(2.(3.[])))$$

The advantage of this representation is that it allows us to describe functions and relations on lists without regard to length.

As an example, consider the definition of the binary relation *member*, which holds of an object and a list if the object is a top-level member of the sequence denoted by the list. Using the "`.`" operator, we can describe the *member* relation as follows. Obviously, an object is a member of a sequence if it is the first element; however, it is also a member if it is member of the rest of the list.

$$\forall x.\forall l.member(x, x.l)$$
$$\forall x.\forall y.\forall l.(member(x, l) \Rightarrow member(x, y.l))$$

We also can define functions to manipulate lists in different ways. For example, the following axioms define a function called *append*. The value of *append* is a sequence consisting of the elements in the sequence supplied as its first argument followed by the elements in the sequence supplied as its second argument. Thus, *append*$([1, 2], [3, 4])$ denotes the same sequence as the list $[1, 2, 3, 4]$.

$$\forall m.append([], m) = m$$
$$\forall x.\forall l.\forall m.append(x.l, m) = (x.append(l, m))$$

Of course, we can also define relations that depend on the structure of the elements of a sequence. For example, the `Among` relation is true of an object and a sequence if the object is a member of the sequence, if it is a member of a sequence that is itself a member of the sequence, and so on.

$$\forall x.among(x, x)$$
$$\forall x.\forall y.\forall z.(among(x, y) \lor among(x, z)) \Rightarrow among(x, y.z))$$

Lists are an extremely versatile representational device, and the reader is encouraged to become as familiar as possible with the techniques of writing definitions for functions and relations on lists. As is true of many tasks, practice is the best approach to gaining skill.

Exercises

1. Syntax Test. Assume that *red*, *green*, *jim*, and *molly* are object constants. Assume that *color* is a unary function constant. And assume that *p* is a unary relation constant and *parent* is a binary relation constant. Under these assumptions, say whether each of the following expressions is a legal sentence in relational logic.

(a) $parent(red, green)$
(b) $\neg color(jim)$
(c) $color(blue, molly)$
(d) $color(molly, red \land green)$
(e) $parent(molly, molly)$
(f) $parent(molly)$
(g) $parent(molly, z)$
(h) $x(jim, molly) \Rightarrow x(molly, jim)$

2. *Translation.* Use the following vocabulary to express the information in the sentences below.

$male(x)$ means that the object denoted by x is male.
$female(x)$ means that x is female.
$vegetarian(x)$ means that x is a vegetarian.
$butcher(x)$ means that x is a butcher.

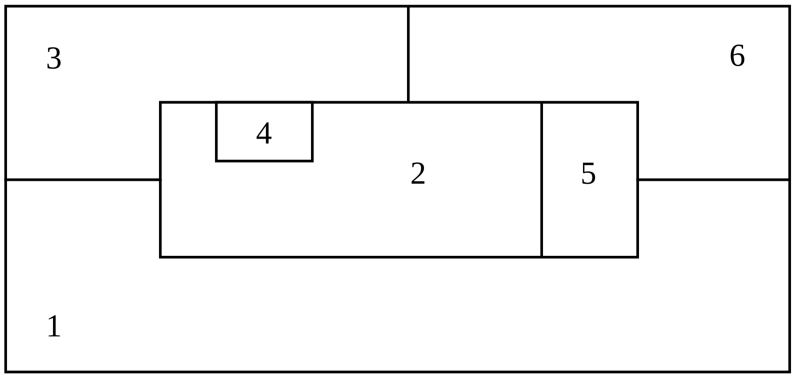
(a) No man is both a butcher and a vegetarian.
(b) All men except butchers like vegetarians.
(c) The only vegetarian butchers are women.
(d) No man likes a woman who is a vegetarian.

(e) No woman likes a man who does not like all vegetarians.

3. Lists. Write the axioms defining the function *reverse*, which takes a list as argument and returns as value the reverse of the list supplied as its argument.

4. *Resolution.* If a course is easy, some students are happy. If a course has a final, no students are happy. Use resolution to show that, if a course has a final, the course is not easy.

5. *Map coloring.* Consider the problem of coloring the following map, using only four colors, such that no two adjacent regions share the same color.



This problem can be set up as a relational logic problem. Write down the database and the query.