signal on the output terminal that flows along another wire. To determine what these signals will be, we need to know how the gates transform their input signals. There are four types of gates: AND, OR, and XOR gates have two input terminals, and NOT gates have one. All gates have one output terminal. Circuits, like gates, have input and output terminals.

To reason about functionality and connectivity, we do not need to talk about the wires themselves, the paths they take, or the junctions where they come together. All that matters is the connections between terminals—we can say that one output terminal is connected to another input terminal without having to say what actually connects them. Other factors such as the size, shape, color, or cost of the various components are irrelevant to our analysis.

If our purpose were something other than verifying designs at the gate level, the ontology would be different. For example, if we were interested in debugging faulty circuits, then it would probably be a good idea to include the wires in the ontology, because a faulty wire can corrupt the signal flowing along it. For resolving timing faults, we would need to include gate delays. If we were interested in designing a product that would be profitable, then the cost of the circuit and its speed relative to other products on the market would be important.

Decide on a vocabulary

We now know that we want to talk about circuits, terminals, signals, and gates. The next step is to choose functions, predicates, and constants to represent them. First, we need to be able to distinguish gates from each other and from other objects. Each gate is represented as an object named by a constant, about which we assert that it is a gate with, say, $Gate(X_1)$. The behavior of each gate is determined by its type: one of the constants AND, OR, XOR, or NOT. Because a gate has exactly one type, a function is appropriate: $Type(X_1) = XOR$. Circuits, like gates, are identified by a predicate: $Circuit(C_1)$.

Next we consider terminals, which are identified by the predicate $Terminal(x)$. A gate or circuit can have one or more input terminals and one or more output terminals. We use the function $In(1, X_1)$ to denote the first input terminal for gate X_1. A similar function Out is used for output terminals. The function $Arity(c, i, j)$ says that circuit c has i input and j output terminals. The connectivity between gates can be represented by a predicate, $Connected$, which takes two terminals as arguments, as in $Connected(Out(1, X_1), In(1, X_2))$.

Finally, we need to know whether a signal is on or off. One possibility is to use a unary predicate, $On(t)$, which is true when the signal at a terminal is on. This makes it a little difficult, however, to pose questions such as "What are all the possible values of the signals at the output terminals of circuit C_1 ?" We therefore introduce as objects two signal values, 1 and 0, and a function $Signal(t)$ that denotes the signal value for the terminal t.

Encode general knowledge of the domain

One sign that we have a good ontology is that we require only a few general rules, which can be stated clearly and concisely. These are all the axioms we will need:

1. If two terminals are connected, then they have the same signal:
   $\forall t_1, t_2 \;\; Terminal(t_1) \wedge Terminal(t_2) \wedge Connected(t_1, t_2) \Rightarrow$
   $Signal(t_1) = Signal(t_2)$.

2. The signal at every terminal is either 1 or 0:
   $\forall t \;\; Terminal(t) \Rightarrow Signal(t) = 1 \vee Signal(t) = 0$.
3. Connected is commutative:
   $\forall t_1, t_2 \;\; Connected(t_1, t_2) \Leftrightarrow Connected(t_2, t_1)$.
4. There are four types of gates:
   $\forall g \;\; Gate(g) \wedge k = Type(g) \Rightarrow k = AND \vee k = OR \vee k = XOR \vee k = NOT$.
5. An AND gate's output is 0 if and only if any of its inputs is 0:
   $\forall g \;\; Gate(g) \wedge Type(g) = AND \Rightarrow$
   $Signal(Out(1, g)) = 0 \Leftrightarrow \exists n \;\; Signal(In(n, g)) = 0$.
6. An OR gate's output is 1 if and only if any of its inputs is 1:
   $\forall g \;\; Gate(g) \wedge Type(g) = OR \Rightarrow$
   $Signal(Out(1, g)) = 1 \Leftrightarrow \exists n \;\; Signal(In(n, g)) = 1$.
7. An XOR gate's output is 1 if and only if its inputs are different:
   $\forall g \;\; Gate(g) \wedge Type(g) = XOR \Rightarrow$
   $Signal(Out(1, g)) = 1 \Leftrightarrow Signal(In(1, g)) \neq Signal(In(2, g))$.
8. A NOT gate's output is different from its input:
   $\forall g \;\; Gate(g) \wedge (Type(g) = NOT) \Rightarrow$
   $Signal(Out(1, g)) \neq Signal(In(1, g))$.
9. The gates (except for NOT) have two inputs and one output.
   $\forall g \;\; Gate(g) \wedge Type(g) = NOT \Rightarrow Arity(g, 1, 1)$.
   $\forall g \;\; Gate(g) \wedge k = Type(g) \wedge (k = AND \vee k = OR \vee k = XOR) \Rightarrow$
   $Arity(g, 2, 1)$
10. A circuit has terminals, up to its input and output arity, and nothing beyond its arity:
    $\forall c, i, j \;\; Circuit(c) \wedge Arity(c, i, j) \Rightarrow$
    $\forall n \;\; (n \leq i \Rightarrow Terminal(In(c, n))) \wedge (n > i \Rightarrow In(c, n) = Nothing) \wedge$
    $\forall n \;\; (n \leq j \Rightarrow Terminal(Out(c, n))) \wedge (n > j \Rightarrow Out(c, n) = Nothing)$
11. Gates, terminals, signals, gate types, and $Nothing$ are all distinct.
    $\forall g, t \;\; Gate(g) \wedge Terminal(t) \Rightarrow$
    $g \neq t \neq 1 \neq 0 \neq OR \neq AND \neq XOR \neq NOT \neq Nothing$.
12. Gates are circuits.
    $\forall g \;\; Gate(g) \Rightarrow Circuit(g)$

Encode the specific problem instance

The circuit shown in Figure 8.6 is encoded as circuit C_1 with the following description. First, we categorize the circuit and its component gates:

$Circuit(C_1) \wedge Arity(C_1, 3, 2)$
$Gate(X_1) \wedge Type(X_1) = XOR$
$Gate(X_2) \wedge Type(X_2) = XOR$
$Gate(A_1) \wedge Type(A_1) = AND$
$Gate(A_2) \wedge Type(A_2) = AND$
$Gate(O_1) \wedge Type(O_1) = OR$.