Then, we show the connections between them:

\[
\begin{align*}
\text{Connected}(\text{Out}(1, M(1, X2))) & \quad \text{Connected}(\text{In}(1, C1), \text{In}(1, X1)) \\
\text{Connected}(\text{Out}(1, X1), \text{In}(2, A2)) & \quad \text{Connected}(\text{In}(1, C1), \text{In}(1, A1)) \\
\text{Connected}(\text{Out}(1, A2), \text{In}(1, O1)) & \quad \text{Connected}(\text{In}(2, C1), \text{In}(2, X1)) \\
\text{Connected}(\text{Out}(1, A1), \text{In}(2, O1)) & \quad \text{Connected}(\text{In}(2, C1), \text{In}(2, A1)) \\
\text{Connected}(\text{Out}(1, X2), \text{Out}(*, C1)) & \quad \text{Connected}(\text{In}(3, C1), \text{In}(2, X2)) \\
\text{Connected}(\text{Out}(1, \text{Out}(2, C1))) & \quad \text{Connected}(\text{In}(3, C1), \text{In}(1, A2)).
\end{align*}
\]

Pose queries to the inference procedure:

What combinations of inputs would cause the first output of C1 (the sum bit) to be 0 and the second output of C1 (the carry bit) to be 1?

\[
\exists i_1, i_2, i_3 \quad \text{Signal}(\text{In}(1, CO)) = 1 \land \text{Signal}(\text{In}(2, CO)) = 1 \land \text{Signal}(\text{Out}(1, C1)) = i_1 \land \text{Signal}(\text{Out}(2, C1)) = i_2.
\]

The answers are substitutions for the variables \(i_1, i_2, i_3\), and such that the resulting sentence is entailed by the knowledge base. `ASKVARS` will give us three such substitutions:

\[
\{i_1/0, i_2/1, i_3/1\}, \{i_1/0, i_2/0, i_3/1\}, \{i_1/1, i_2/1, i_3/1\}.
\]

What are the possible sets of values of all the terminals for the adder circuit?

\[
\exists i_1, i_2, i_3, 0, 1, 2 \quad \text{Signal}(\text{In}(1, CO)) = i_1 \land \text{Signal}(\text{In}(2, CO)) = i_2 \land \text{Signal}(\text{Out}(1, C1)) = i_1 \land \text{Signal}(\text{Out}(2, C1)) = i_2.
\]

This final query will return a complete input-output table for the device, which can be used to check that it does in fact add its inputs correctly. This is a simple example of circuit verification. We can also use the definition of the circuit to build larger digital systems, for which the same kind of verification procedure can be carried out. (See Exercise 8.26.) Many domains are amenable to the same kind of structured knowledge-base development, in which more complex concepts are defined on top of simpler concepts.

Debug the knowledge base

We can perturb the knowledge base in various ways to see what kinds of erroneous behaviors emerge. For example, suppose we fail to read Section 8.2.8 and hence forget to assert that 1 \# 0. Suddenly, the system will be unable to prove any outputs for the circuit, except for the input cases 000 and 110. We can pinpoint the problem by asking for the outputs of each gate. For example, we can ask

\[
\exists i_1, i_2, i_3 \quad \text{Signal}(\text{In}(1, CO)) = 2r \land \text{Signal}(\text{In}(2, C1)) = i_2 \land \text{Signal}(\text{Out}(1, X1)),
\]

which reveals that no outputs are known at X1 for the input cases 10 and 01. Then, we look at the axiom for XOR gates, as applied to X1:

\[
\text{Signal}(\text{Out}(1, X1)) = 1 \iff \text{Signal}(\text{In}(1, X1)) \neq \text{Signal}(\text{In}(2, X1)).
\]

If the inputs are known to be, say, 1 and 0, then this reduces to

\[
\text{Signal}(\text{Out}(1, X1)) = 1 \iff 1 \neq 0.
\]

Now the problem is apparent: the system is unable to infer that \(\text{Signal}(\text{Out}(1, X1)) = 1\), so we need to tell it that 1 \# 0.
8.5 SUMMARY

This chapter has introduced **first-order logic**, a representation language that is far more powerful than propositional logic. The important points are as follows:

- Knowledge representation languages should be declarative, compositional, expressive, context independent, and unambiguous.
- Logics differ in their **ontological commitments** and **epistemological commitments**. While propositional logic commits only to the existence of facts, first-order logic commits to the existence of objects and relations and thereby gains expressive power.
- The syntax of first-order logic builds on that of propositional logic. It adds terms to represent objects, and has universal and existential quantifiers to construct assertions about all or some of the possible values of the quantified variables.
- A possible world, or model, for first-order logic includes a set of objects and an **interpretation** that maps constant symbols to objects, predicate symbols to relations among objects, and function symbols to functions on objects.
- An atomic sentence is true just when the relation named by the predicate holds between the objects named by the terms. **Extended interpretations**, which map quantifier variables to objects in the model, define the truth of quantified sentences.
- Developing a knowledge base in first-order logic requires a careful process of analyzing the domain, choosing a vocabulary, and encoding the axioms required to support the desired inferences.

BIBLIOGRAPHICAL AND HISTORICAL NOTES

Although Aristotle’s logic deals with generalizations over objects, it fell far short of the expressive power of first-order logic. A major barrier to its further development was its concentration on one-place predicates to the exclusion of many-place relational predicates. The first systematic treatment of relations was given by Augustus De Morgan (1864), who cited the following example to show the sorts of inferences that Aristotle’s logic could not handle: “All horses are animals; therefore, the head of a horse is the head of an animal.” This inference is inaccessible to Aristotle because any valid rule that can support this inference must first analyze the sentence using the two-place predicate “x is the head of”. The logic of relations was studied in depth by Charles Sanders Peirce (1870, 2004).

True first-order logic dates from the introduction of quantifiers in Gottlob Frege’s (1879) **Begriffschrift** (“Concept Writing” or “Conceptual Notation”). Peirce (1883) also developed first-order logic independently of Frege, although slightly later, Frege’s ability to nest quantifiers was a big step forward, but he used an awkward notation. The present notation for first-order logic is due substantially to Giuseppe Peano (1889), but the semantics is virtually identical to Frege’s. Oddly enough, Peano’s axioms were due in large measure to Grossmann (1861) and Dedekind (1888).