Exercise 2.20: Design a 1-bit synchronous adder `1BitAdder` by composing instances of And, Or, Not, and Xor gates. The component `1BitAdder` has three input variables *x*, *y*, and *carry-in* and two output variables *z* and *carry-out*. In each round, the value encoded by the two output bits *z* and *carry-out*, where *z* is the least significant bit, should equal the sum of the values of three input variables. Then, design a 3-bit synchronous adder `3BitAdder` by composing three instances of the component `1BitAdder`. The component `3BitAdder` has input variables x_0, x_1, x_2, y_0, y_1, y_2, and *carry-in* and has output variables z_0, z_1, z_2, and *carry-out*. In each round, the 4-bit number encoded by the output variables z_0, z_1, z_2, and *carry-out* should equal the sum of the 3-bit number encoded by the input variables x_0, x_1, and x_2, the 3-bit number encoded by the input variables y_0, y_1, and y_2, and the input value of *carry-in*. ■

Exercise 2.22: Consider the leader election algorithm in synchronous networks (figure 2.35). Argue that if the value of *id* does not change in a given round, then there is no need to send it in the following round (that is, the output *out* can be absent in the next round). This can reduce the number of messages sent. Modify the description of the component `SyncLENode` to implement this change. ■

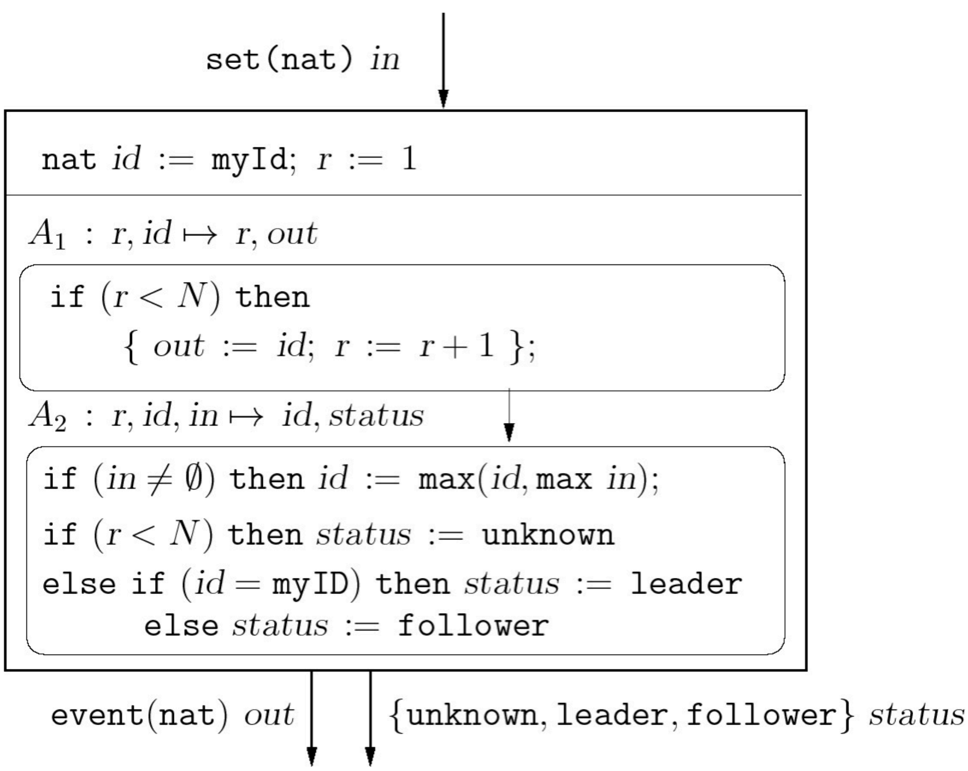

Figure 2.35: Component `SyncLENode` for Synchronous Leader Election

Exercise 3.6: Consider a transition system T with two integer variables x and y and a Boolean variable z. All the variables are initially 0. The transitions of the system correspond to executing the conditional statement

$$\texttt{if } (z = 0) \texttt{ then } \{x := x + 1;\ z := 1\} \texttt{ else } \{y := y + 1;\ z := 0\}.$$

Consider the property ϕ given by $(x = y) \vee (x = y + 1)$. Is ϕ an invariant of the transition system T? Is ϕ an inductive invariant of the transition system T? Find a formula ψ such that ψ is stronger than ϕ and is an inductive invariant of the transition system T. Justify your answers. ■

Exercise 3.7: Recall the transition system $\texttt{Mult}(m, n)$ from exercise 3.1. First, show that the invariant property $(mode = \texttt{stop}) \rightarrow (y = m \cdot n)$ is not an inductive invariant. Then find a stronger property that is an inductive invariant. Justify your answers. ■

Exercise 3.1: Given two natural numbers *m* and *n*, consider the program Mult that multiplies the input numbers using two variables *x* and *y*, of type nat, as shown in figure 3.2. Describe the transition system `Mult`(*m, n*) that captures the behavior of this program on input numbers *m* and *n*, that is, describe the states, initial states, and transitions. Argue that when the value of the variable *x* is 0, the value of the variable *y* must equal the product of the input numbers *m* and *n*, that is, the following property is an invariant of this transition system:

$$(mode = \texttt{stop}) \;\rightarrow\; (y = m \cdot n)$$

■

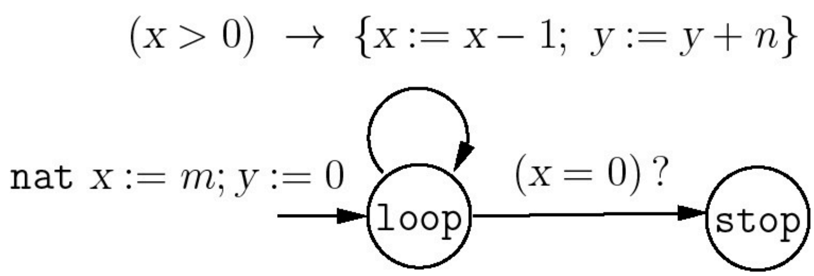

Figure 3.2: Program for Multiplication

Exercise 3.3: The reaction description for the controller `Controller2` consists of three tasks as shown in figure 3.8. `Split` the task A_3 into four tasks, each of which writes exactly one of the state variables *east*, *west*, $near_W$, and $near_E$. Each task should be described by its read-set, write-set, and update code, along with the necessary precedence constraints. The revised description should have the same set of reactions as the original description. Does this splitting impact output/ input await dependencies? If not, what would be the potential benefits and/ or drawbacks of the revised description compared to the original description? ■

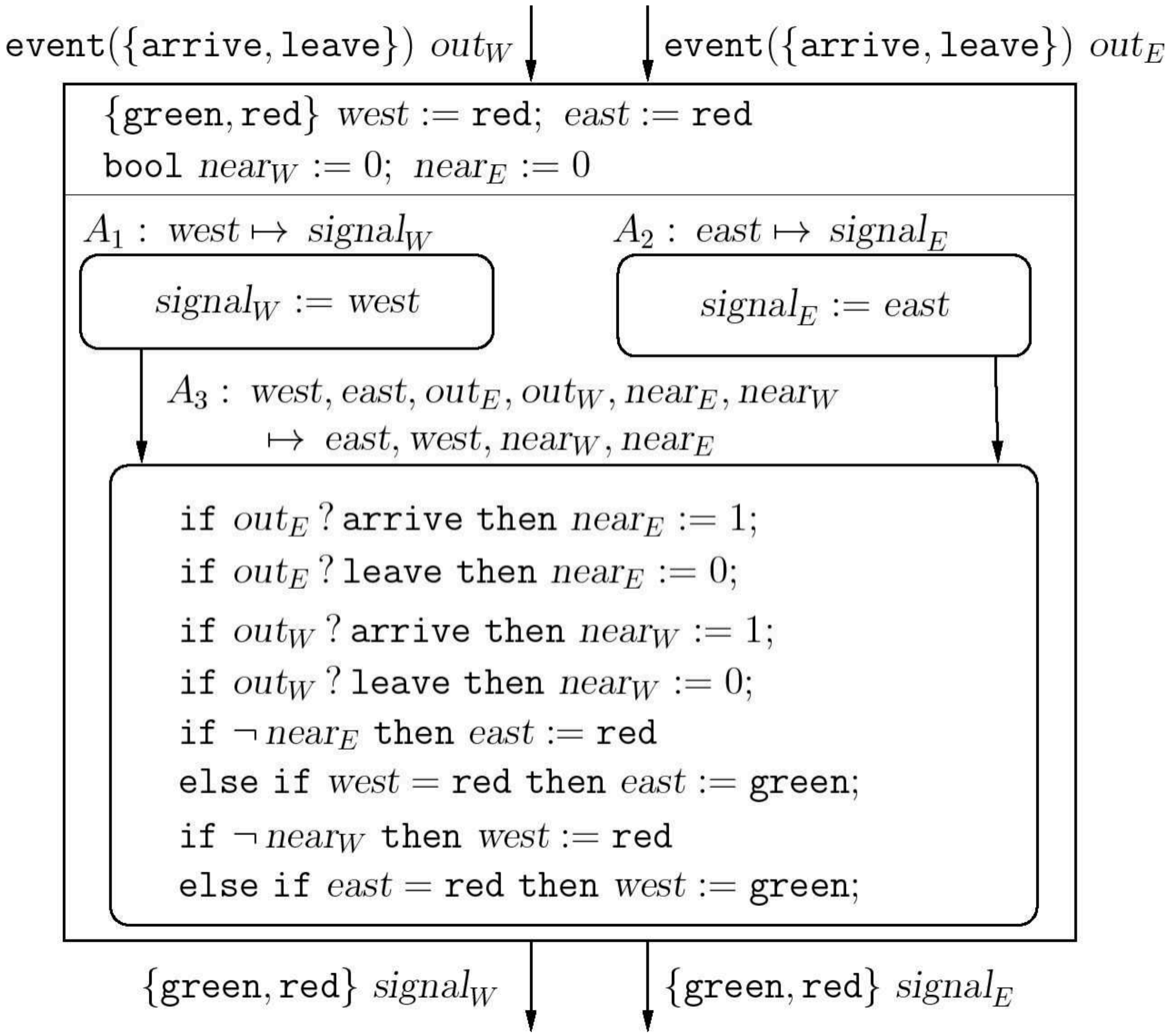

Figure 3.8: A Safe Controller for the Railroad Problem

Exercise 3.4: Consider a component *C* with an output variable *x* of type `int`. Design a safety monitor to capture the requirement that the sequence of values output by the component *C* is strictly increasing (that is, the output in each round should be strictly greater than the output in the preceding round). ■