6 Machines

Exercise 6.1. Babbage's design for the Analytical Engine called for a store holding 1000 variables, each of which is a 50-digit (decimal) number. How many bits could the store of Babbage's Analytical Engine hold?

Solution.

Exercise 6.2. Define a Scheme procedure, *logical-or*, that takes two inputs and outputs the logical or of those inputs.

Solution.

Exercise 6.3. What is the meaning of composing *not* with itself? For example, (*not* (*not A*)).

Solution.

Exercise 6.4. Define the *xor* function using only *nand* functions.

Solution.

Exercise 6.5. [⋆] Our definition of (*not A*) as (*nand A A*) assumes there is a way to produce two copies of a given input. Design a component for our wine machine that can do this. It should take one input, and produce two outputs, both with the same value as the input. (Hint: when the input is *true*, we need to produce two full bottles as outputs, so there must be a source similarly to the *not* component.)

Solution.

Exercise 6.6. [⋆] The digital abstraction works fine as long as actual values stay close to the value they represent. But, if we continue to compute with the outputs of functions, the actual values will get increasingly fuzzy. For example, if the inputs to the *and3* function in Figure 6.3 are initially all $\frac{3}{4}$ full bottles (which should be interpreted as *true*), the basin for the first *and* function will fill to $1\frac{1}{2}$, so only $\frac{1}{2}$ bottle will be output from the first *and*. When combined with the third input, the second basin will contain $1\frac{1}{4}$ bottles, so only $\frac{1}{4}$ will spill into the output bottle. Thus, the output will represent *false*, even though all three inputs represent *true*. The solution to this problem is to use an *amplifier* to restore values to their full representations. Design a wine machine amplifier that takes one input and produces a strong representation of that input as its output. If that input represents *true* (any value that is half full or more), the amplifier should output *true*, but with a strong, full bottle representation. If that input represents *false* (any value

that is less than half full), the amplifier should output a strong *false* value (completely empty).

Solution.

Exercise 6.7. Adding logically.

a. What is the logical formula for r_3?

Solution.

b. Without simplification, how many functions will be composed to compute the addition result bit r_4?

Solution.

c. [⋆] Is it possible to compute r_4 with fewer logical functions?

Solution.

Exercise 6.8. Show how to compute the result bits for binary multiplication of two 2-bit inputs using only logical functions.

Solution.

Exercise 6.9. [⋆] Show how to compute the result bits for binary multiplication of two inputs of any length using only logical functions.

Solution.

Exercise 6.10. Follow the rules to simulate the checking parentheses Turing Machine on each input (assume the beginning and end of the input are marked with a `#`):

a. `)`

b. `()`

c. *empty input*

d. `(()(()))()`

e. `(()))(()`

Exercise 6.11. [⋆] Design a Turing Machine for adding two arbitrary-length binary numbers. The input is of the form $a_{n-1} \ldots a_1 a_0$ + $b_{m-1} \ldots b_1 b_0$ (with `#` markers at both ends) where each a_k and b_k is either `0` or `1`. The output tape should contain bits that represent the sum of the two inputs.

Solution.