The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

Solution Manual
Chapter 3, Problem 3.20

1. This language is regular. In processing the symbols, the FA need only store in its finite state whether or not the next symbol encountered should generate or propagate a carry. Since we read the input in the "wrong" direction (from the most significant bit to the least significant), what really happens is that we detect the use of a carry before we get to the point where it is generated—in effect, our FA will check that every carry used is also generated. The FA has just two states; the start state is accepting, the other rejecting. We stay in the start state when the next vector read involves no carry generation, propagation, or use, for triples $\begin{pmatrix}0\\0\\0\end{pmatrix}$, $\begin{pmatrix}0\\1\\1\end{pmatrix}$, and $\begin{pmatrix}1\\0\\1\end{pmatrix}$; we move to the other state when a carry is used and not propagated, for triple $\begin{pmatrix}0\\0\\1\end{pmatrix}$. Once in the other state, we stay there when a carry is propagated, for triples $\begin{pmatrix}0\\1\\0\end{pmatrix}$, $\begin{pmatrix}1\\0\\0\end{pmatrix}$, and $\begin{pmatrix}1\\1\\1\end{pmatrix}$, but return to the start state when a carry is generated, for triple $\begin{pmatrix}1\\1\\0\end{pmatrix}$.
2. This language is not regular. Basically, this is due to the fact that the number of bits needed to express the product of x and y can grows as fast as (although no faster than) the sum of the number of bits needed to express each of x and y. We can set up a product where xy takes $|x| + |y|$ bits and where we only pump in the area where x and y have leading 0s, thereby increasing the number of bits used in xy without increasing the number of bits used in x or y—or, viewed differently, thereby altering the product without altering the terms.

Let n be the constant of the pumping lemma. We choose the following string:

$$z = \begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}0\\0\\0\end{pmatrix}^{n}\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}0\\0\\0\end{pmatrix}^{n+1}$$

which corresponds to the product $2^{n+1} \cdot 2^{n+1} = 2^{2n+2}$ and is sufficiently long. We use the extended pumping lemma and write $z = xuvwy$, where $x = \begin{pmatrix}0\\0\\1\end{pmatrix}$, $y = \begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}0\\0\\0\end{pmatrix}^{n+1}$, and $uvw = \begin{pmatrix}0\\0\\0\end{pmatrix}^{n}$, so that v is a string of one or more triples of 0s. Pumping up or down once does not affect the two product terms, but doubles or halves the result, which is then incorrect for the multiplicands given, so that the new string is not in the language.