Automata: a subset of your wishes is their command

David Wilding, April 2012

http://dpw.me/mathematics/

An automaton is a device for computing a property of a string of symbols. They are typically used to compute membership of formal languages, but they can also count features of strings and decode strings.

Contents

1 Alphabets and words

An *alphabet* A is a non-empty finite set whose elements we call *symbols* or *letters*. The set $\{\mathtt{0}, \mathtt{1}\}$ of binary digits and the set $\{\mathtt{0}, \mathtt{1}, \ldots, \mathtt{9}\}$ of decimal digits should be familiar alphabets. Note that the symbols in an alphabet are simply distinct abstract elements with no intrinsic meaning or value; the alphabet $\{\mathtt{9}, \mathtt{8}\}$ could be used to represent binary numbers instead of $\{\mathtt{0}, \mathtt{1}\}$.

We can *concatenate* (stick together) symbols from an alphabet A to form *words* (the "strings of symbols" referred to above) of arbitrary finite length. For example

$$\varepsilon, \quad \mathtt{0}, \quad \mathtt{101} \quad \text{and} \quad \mathtt{01}^{724} \tag{1}$$

are all words over the alphabet $\{\mathtt{0}, \mathtt{1}\}$, where ε denotes the *empty* word of length zero and $\mathtt{1}^{724}$ means $\mathtt{1}$ written 724 times. Formally, words over an alphabet A are elements of the free monoid

$$A^* = A^0 \cup A^1 \cup A^2 \cup \cdots \tag{2}$$

on A. The product of two elements of A^* is simply their concatenation, so the identity element for this operation is the empty word.

If we are being very picky there is a distinction between symbols (elements of A) and words of length one (elements of A^1), but it is customary to ignore this fact and regard A and A^1 as one and the same.

2 Decoding numbers

Ternary (base 3) numbers are conventionally written using the alphabet $\{\mathtt{0}, \mathtt{1}, \mathtt{2}\}$. The word $\mathtt{201}$ then represents the number 19—but how exactly? Let us write $f(w)$ for the ternary value of a word $w \in \{\mathtt{0}, \mathtt{1}, \mathtt{2}\}^*$. Then we can work out $f(\mathtt{201})$ recursively:

$$f(\mathtt{201}) = 1 + f(\mathtt{20}) \cdot 3 \tag{3}$$
$$f(\mathtt{20}) = 0 + f(\mathtt{2}) \cdot 3 \tag{4}$$
$$f(\mathtt{2}) = 2 + f(\varepsilon) \cdot 3 \tag{5}$$

with $f(\varepsilon) = 0$. It may not be immediately obvious, but we can rewrite this recursion in terms of matrices:

$$\begin{bmatrix} 1 & f(\mathtt{201}) \end{bmatrix} = \begin{bmatrix} 1 & f(\mathtt{20}) \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 0 & 3 \end{bmatrix} \tag{6}$$
$$\begin{bmatrix} 1 & f(\mathtt{20}) \end{bmatrix} = \begin{bmatrix} 1 & f(\mathtt{2}) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \tag{7}$$
$$\begin{bmatrix} 1 & f(\mathtt{2}) \end{bmatrix} = \begin{bmatrix} 1 & f(\varepsilon) \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \tag{8}$$

with $\begin{bmatrix} 1 & f(\varepsilon) \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$. This form is nicer because it allows us to do away with the recursion and write

$$f(\mathtt{201}) = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}}_{\mathtt{2}\text{ matrix}} \cdot \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}}_{\mathtt{0}\text{ matrix}} \cdot \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 3 \end{bmatrix}}_{\mathtt{1}\text{ matrix}} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{9}$$

That is, we can compute $f(\texttt{201})$ by taking the product of a row vector, one matrix for each symbol and a column vector.

There is no reason to restrict ourselves to base 3 numbers of course; we can compute the value of a number in any base by forming an appropriate matrix product. For example, the decimal value of $\texttt{55} \in \{\texttt{0}, \texttt{1}, \ldots, \texttt{9}\}^*$ is given by

$$\begin{bmatrix}1 & 0\end{bmatrix} \cdot \underbrace{\begin{bmatrix}1 & 5\\ 0 & 10\end{bmatrix}}_{\texttt{5 matrix}} \cdot \underbrace{\begin{bmatrix}1 & 5\\ 0 & 10\end{bmatrix}}_{\texttt{5 matrix}} \cdot \begin{bmatrix}0\\ 1\end{bmatrix}. \tag{10}$$

3 The definition of an automaton

A *semiring* is a set S with binary operations $+$ and $\cdot$ such that $(S, +, 0)$ is a commutative monoid, $(S, \cdot, 1)$ is a monoid, $\cdot$ distributes over $+$ from both sides and $0 \cdot s = s \cdot 0$ for all $s \in S$. For instance, the set $\{0, 1, 2, \ldots\}$ of natural numbers is a semiring with the usual operations of addition and multiplication.

In order to state the definition of an automaton we must fix an alphabet A and a semiring S. Then an (A, S)-*automaton* (or *weighted automaton*, or just *automaton*) comprises the following.

- A positive integer n called the number of *states*.
- A row vector $U \in S^{1\times n}$ called the *entry* vector.
- A column vector $U \in S^{n\times 1}$ called the *exit* vector.
- A function $E\colon A \to S^{n\times n}$ that associates each symbol with a *transition* matrix.

This function E extends to a monoid morphism $E\colon A^* \to S^{n\times n}$ in a unique way: we set

$$E(a_1 \cdots a_m) = E(a_1) \cdots E(a_m) \tag{11}$$

for all $a_1, \ldots, a_m \in A$ and we take $E(\varepsilon)$ to be the $n \times n$ identity matrix over S. The automaton is then said to *recognise* the function $A^* \to S$ given by

$$w \mapsto U \cdot E(w) \cdot X \tag{12}$$

for all $w \in A^*$. Any function $A^* \to S$ that is recognised by some (A, S)-automaton is called, unsurprisingly, *recognisable*.

The function that sends a word $w \in \{0,1,2\}^*$ to its ternary value is recognisable because there is a $\big(\{0,1,2\},\mathbf{N}\big)$-automaton that recognises it. This automaton has $n = 2$ states, entry and exit vectors

$$U = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{13}$$

respectively and transition matrices

$$E(0) = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}, \quad E(1) = \begin{bmatrix} 1 & 1 \\ 0 & 3 \end{bmatrix} \quad \text{and} \quad E(2) = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}. \tag{14}$$

We illustrated why this automaton recognises the ternary valuation function in the previous section.

4 Automata as graphs

Given an (A,S)-automaton with n states we can construct a weighted directed graph with n vertices labelled $1,\ldots,n$. Each edge (i,j) of this graph, for $1 \le i,j \le n$, is weighted by the function that sends $a \in A$ to $E(a)_{i,j} \in S$. Each vertex i has associated values $U_i \in S$ and $X_i \in S$ from the entry and exit vectors of the automaton. We indicate these values (when drawing the graph) by arrows entering and exiting each vertex.

The graph of the 2-state automaton described in the previous section is shown in Figure 1. For convenience the edge $(1,2)$ is not shown because it is weighted by the function $a \mapsto 0$ for all $a \in \{0,1,2\}$. Similarly the exit value for vertex 1 and the entry value for vertex 2 are not shown because they are both zero.

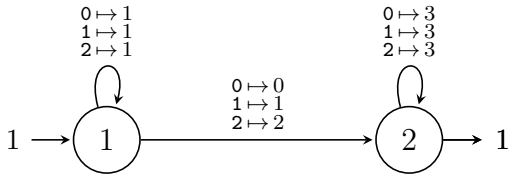

Figure 1: An automaton that computes the ternary value of a word.

We can use the graph of an automaton to evaluate the function $f\colon A^* \to S$ it recognises at a word $w \in A^*$. This computation requires a "memory location" that can store a value from S.

First pick any vertex i and write its entry value to the memory. Then pick another

(possibly the same) vertex j and multiply the memory by the value of the edge (i, j) function applied to the first symbol of w. Then pick yet another vertex k and multiply the memory by the value of the edge (j, k) function applied to the second symbol of w. Continue this process until there are no more symbols in w, then multiply the memory by the exit value of the final vertex.

To compute $f(w)$ we take not the final memory value, but the sum of the final memory value over all possible sequences of vertices we could have picked when applying the above procedure.

5 The computing power of automata

The set of functions $A^* \to S$ (for an alphabet A and a semiring S) can be given the structure of a "monoid algebra over a semiring" (compare with the definition of a group algebra over a ring), and it is for this reason that we will denote it SA^*. The left and right actions of $s \in S$ on $f \in SA^*$ are defined by

$$(sf)(w) = s \cdot f(w) \quad \text{and} \quad (fs)(w) = f(w) \cdot s \tag{15}$$

respectively for all $w \in A^*$, and the sum of $f, g \in SA^*$ is defined by

$$(f + g)(w) = f(w) + g(w) \tag{16}$$

for all $w \in A^*$.

The product of $f, g \in SA^*$ is defined by

$$(f \cdot g)(w) = \sum_{w=xy} f(x) \cdot g(y) \tag{17}$$

for all $w \in A^*$, where "$w = xy$" is shorthand for "$(x, y) \in A^* \times A^*$ with $w = xy$". Each word $v \in A^*$ has an associated *characteristic* function $\hat{v} \in SA^*$ defined by

$$\hat{v}(w) = \begin{cases} 1 & \text{if } w = v, \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

for all $w \in A^*$, and it is clear that $\hat{\varepsilon}$ is the identity element for the product operation.

Thanks to the structure of A^* the algebra SA^* admits a unary operation, called *star*,

which is defined for $f \in SA^*$ by

$$f^*(\varepsilon) = 1 \tag{19}$$

and

$$f^*(wa) = \sum_{w=xy} f^*(x) \cdot f(ya) \tag{20}$$

for all $w \in A^*$ and all $a \in A$. It is no coincidence that the recursive part of this definition looks like a product; if f is *proper*, that is, if $f(\varepsilon) = 0$, then it can be shown that $f^* = f^0 + (f^* \cdot f)$. Intuitively, if f is proper then $f^* = f^0 + f^1 + f^2 + \cdots$.

The Kleene-Schützenberger theorem (1956–1961) tells us precisely which functions in SA^* are recognised by some (A, S)-automaton:

(i) $\hat{a}$ for all $a \in A$;

(ii) sf and fs for all $s \in S$ and all recognisable $f \in SA^*$;

(iii) $f + g$ for all recognisable $f, g \in SA^*$;

(iv) $f \cdot g$ for all recognisable $f, g \in SA^*$;

(v) f^* for all recognisable proper $f \in SA^*$.

In particular, the theorem gives us a suite of operations with which we can express every recognisable function. For example, the ternary valuation function $\{\mathtt{0}, \mathtt{1}, \mathtt{2}\} \to \mathbf{N}$ can be written as

$$\left(\hat{\mathtt{0}} + \hat{\mathtt{1}} + \hat{\mathtt{2}}\right)^* \cdot \left(0\hat{\mathtt{0}} + 1\hat{\mathtt{1}} + 2\hat{\mathtt{2}}\right) \cdot \left(3\hat{\mathtt{0}} + 3\hat{\mathtt{1}} + 3\hat{\mathtt{2}}\right)^*. \tag{21}$$

The three terms in this product correspond to, respectively, the left loop, the middle horizontal arrow and the right loop in Figure 1.

If S is the Boolean semiring $\mathbf{B} = \{\bot, \top\}$ (with addition $\vee$, multiplication $\wedge$, zero $\bot$ and one $\top$) then the elements of SA^* "are" subsets of A^*. That is, the elements of $\mathbf{B}A^*$ are formal languages. The languages that are recognised by some $(\mathbf{B}, A)$-automaton are called *regular* and the Kleene-Schützenberger theorem tells us exactly which languages are regular:

(i) $\{a\}$ for all $a \in A$;

(ii) $\emptyset$;

(iii) $B \cup C$ for all regular $B, C \subseteq A^*$;

(iv) $BC = \{vw : v \in B \text{ and } w \in C\}$ for all regular $B, C \subseteq A^*$;

(v) B^* for all regular $B \subseteq A^* \setminus \{\varepsilon\}$.

The classic example of a language that is not regular is $\{\mathtt{0}^m\mathtt{1}^m : m \in \mathbf{N}\} \subseteq \{\mathtt{0}, \mathtt{1}\}^*$.

References

[1] J. Berstel and C. Reutenauer. *Noncommutative Rational Series with Applications.* Cambridge University Press, Cambridge, 2011.

[2] J. H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, London, 1971.

[3] S. Eilenberg. *Automata, Languages, and Machines, volume A.* Academic Press, New York, 1974.