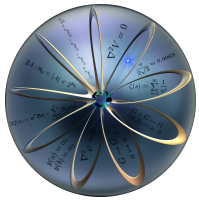

planetmath.org

Math for the people, by the people.

generalized sequential machine

Canonical name	GeneralizedSequentialMachine
Date of creation	2013-03-22 18:57:22
Last modified on	2013-03-22 18:57:22
Owner	CWoo (3771)
Last modified by	CWoo (3771)
Numerical id	15
Author	CWoo (3771)
Entry type	Definition
Classification	msc 03D05
Classification	msc 68Q45
Synonym	gsm
Synonym	deterministic generalized sequential machine
Related topic	StateOutputMachine
Defines	gsm mapping
Defines	inverse gsm mapping
Defines	deterministic gsm

Definition

Generalized sequential machines are generalizations of finite state sequential machines. In a finite state machine $M = (S, \Sigma, \Delta, \delta, \lambda)$, the next-state ($\delta$) and output ($\lambda$) functions work independently of one another in the sense that no next states affect the outputs, and vice versa: given an input symbol a and current state s, if p, q are next states, and c, d are output symbols, then all four output configurations are possible: $(p, c), (p, d), (q, c), (q, d)$. In addition, for a given input symbol a, the corresponding outputs are individual symbols in the output alphabet, rather than words: $\lambda(s, a)$ is a subset of Δ. When these restrictions are removed, we have a generalized sequential machine.

Formally, a *generalized sequential machine* is a quadruple $M = (S, \Sigma, \Delta, \tau)$, such that

1. S is an alphabet whose elements are called *states*,
2. Σ is an alphabet whose elements are called *input symbols*,
3. Δ is an alphabet whose elements are called *output symbols*, and
4. τ is a function from $S \times \Sigma$ to $P(S \times \Delta^*)$, such that $\tau(s, a)$ is finite for all $(s, a) \in S \times \Sigma$. τ is called the *transition function*.

A generalized sequential machine is also called a *gsm* for short. Note that a gsm M becomes a finite state machine if τ can be written as (δ, λ), and λ is a function from $S \times \Sigma$ to $P(\Delta)$. A gsm is said to be *deterministic* if each $\tau(s, a)$ is a singleton. So a Mealy machine is just a deterministic gsm such that $\tau(s, a)$ consists of an output symbol.

Like a finite state machine, the function τ can be extended so its first component applies to sets of states, rather than individual states:

$$\tau(T, a) = \bigcup \{\tau(s, a) \mid s \in T\}.$$

As usual, $\tau(\varnothing, a) = \varnothing$.

Next, we can extend τ so M can process input words rather than input symbols. The key idea, like in the case of a fsm, is that the input word is processed one symbol at a time from left to right. Each time an input symbol is processed or consumed, a set of output configurations are produced. The states in these output configurations are used to process the next input

symbols, and the outputs are appended to the right of the outputs that have already been generated. More precisely,

$$\tau(T, ua) := \{(s, vw) \mid (s, w) \in \tau(t, a) \text{ for some } t \in S \text{ such that } (t, v) \in \tau(T, u)\}.$$

When the input word is the empty word λ, we require its output to be the empty word also, without changing the state, hence $\tau(s, \lambda) := \{(s, \lambda)\}$.

Here's an example. Let $M = (\{s, t\}, \{a, b\}, \{c, d\}, \tau)$ be a gsm whose transition function τ is given by the following table

(x, y)	$\tau(x, y)$
(s, a)	$\{(t, c)\}$
(s, b)	$\{(s, cd), (s, d^2)\}$
(t, a)	$\{(t, dc)\}$
(t, b)	$\{(s, d), (s, c^2)\}$

To fine $\tau(s, ab)$, first find $\tau(s, a)$, which consists of a single output configuration (t, c) according to the table above. So c will serve as the prefix of the output(s). The next state is now t, to be applied to the second symbol b in ab. By the table above, $\tau(t, b)$ has two output configurations: (s, d) and (s, c^2). Therefore, the final output configurations are (s, cd) and (s, c^3), when c is appended to the left of d and c^2. We may also apply the formula above:

$$\begin{aligned} \tau(s, ab) &= \{(p, vw) \mid (p, w) \in \tau(q, b) \text{ where } (q, v) \in \tau(s, a)\} \\ &= \{(p, cw) \mid (p, w) \in \tau(t, b)\} \\ &= \{(s, cd), (s, c^3)\}. \end{aligned}$$

Language Translation

A gsm can be used to translate languages. In other words, an input language L over Σ is fed into a gsm M so that an output language $M(L)$ is produced, or "translated". This can be done by fixing a start state s_0 and a set F of final states, much like an automaton. First, consider an input word u, then $\tau(s_0, u)$ is the resulting set of output configurations. Define the set of outputs translated from u by M as:

$$\mathrm{GSM}_M(u) := \{v \mid (t, v) \in \tau(s_0, u) \text{ for some } t \in F\}.$$

More generally, let L be an input language (a set of inputs), define the output language translated from L by M as:

$$\mathrm{GSM}_M(L) := \{v \mid v \in \mathrm{GSM}(u) \text{ for some } u \in L\}.$$

It is clear that GSM_M is a mapping from $P(\Sigma^*)$ to $P(\Delta^*)$, and it is in this regard that we see M as a language translator. Of course, different start states and different sets of final states produce different translations, which is the reason why a gsm is usually regarded as a 6-tuple $(S, \Sigma, \Delta, \tau, s_0, F)$ rather than a quadruple.

Given a set K of output words, one can ask the inverse question: what are all the input words whose output configurations contain an output word in K? In other words, we are forming the set:

$$\mathrm{GSM}_M^{-1}(K) := \{u \mid \mathrm{GSM}_M(u) \cap K \neq \varnothing\}.$$

Note, however, that if $L = \mathrm{GSM}_M^{-1}(K)$, then L in general does not get translated to K: the function GSM_M^{-1} is not a functional inverse of GSM_M:

$$\mathrm{GSM}_M \circ \mathrm{GSM}_M^{-1}(K) \neq K \qquad \text{and} \qquad \mathrm{GSM}_M^{-1} \circ \mathrm{GSM}_M(L) \neq L$$

in general.

GSM as an Acceptor

A gsm may be turned into a language acceptor in the following way: let $M = (S, \Sigma, \Delta, \tau, s_0, F)$ be a gsm with starting state s_0 and F the set of final states. Then the language

$$L(M) := \{u \in \Sigma^* \mid \mathrm{GSM}(u) \neq \varnothing\}$$

is called the language *accepted by M*. Now, given M, we may construct a gsm $M' = (S, \Sigma, \varnothing, \tau', s_0, F)$, such that $\tau'(s, a) := \{(t, \epsilon) \mid (t, v) \in \tau(s, a) \text{ for some } v \in \Delta^*\}$, where ϵ denotes the empty word. It is easy to see that $L(M) = L(M')$. Given any $(s, a) \in S \times \Sigma$, define $s \to at$ iff $(t, \epsilon) \in \tau'(s, a)$. From this, it is readily seen that the formal grammar $G = (\Sigma \cup S, F, P, s_0)$, where the productions in P are of the form $s \to at$ defined earlier, generates $L(M')$. As a result, every language accepted by a gsm is regular. It is not hard to see that the converse is also true: every regular language is accepted by some gsm.

GSM Mappings

The gsm mapping of a gsm M is the function $\text{GSM}_M : P(\Sigma^*) \to P(\Delta^*)$ defined earlier. A GSM mapping is a function GSM_M for some gsm M. An inverse GSM mapping is defined similarly. A GSM mapping is said to be λ-free if $(t, \epsilon) \notin \tau(s, a)$ for any $(s, a) \in S \times \Sigma$.

Examples

- Any language homomorphism is a gsm mapping. Given a homomorphism $h : \Sigma \to \Delta^*$, define $M = (S, \Sigma, \Delta, \tau, s_0, F)$ such that $S = F = \{s_0\}$ and $\tau(s_0, a) = \{(s_0, h(a))\}$. Then M is a gsm with $\text{GSM}_M = h$. Similarly, any inverse homomorphism is an inverse gsm mapping.

- The mapping $L \mapsto L \cap R$, where R is a regular language, is a gsm mapping. To see this, let G be a grammar generating R consisting of productions of the form $A \to aB$ or the form $A \to \lambda$. Define $M = (S, \Sigma, \Sigma, \tau, s_0, F)$ as follows: S consists of the non-terminals of G as well as a new symbol s, s_0 is the starting symbol for G, and $F = \{s\}$. Next, set $\tau(A, a) = (B, a)$ iff $A \to aB$, and $\tau(A, a) = (s, a)$ iff $A \to \lambda$. Then, for any word u, it is easy to see that $\text{GSM}_M(u) = \{u\}$ iff $u \in R$. As a result, $\text{GSM}_M(L) = L \cap R$.

A family $\mathscr{F}$ of languages is said to be closed under GSM mappings if for any $L \in \mathscr{F}$, $\text{GSM}_M(L) \in \mathscr{F}$ for any gsm M. Closure under λ-free GSM mappings and inverse GSM mappings are similarly defined. It can be shown that each of the four families in the Chomsky hierarchy is closed under inverse GSM mappings. While the families of regular, context-free, and T0 languages are closed under GSM mappings, the family of context-sensitive languages is only closed under λ-free GSM mappings.

References

[1] A. Salomaa, *Formal Languages*, Academic Press, New York (1973).

[2] J.E. Hopcroft, J.D. Ullman, *Formal Languages and Their Relation to Automata*, Addison-Wesley, (1969).