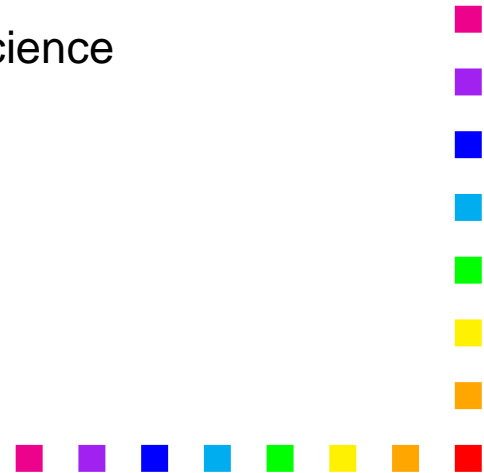


Transducers

Teodor Rus

`rus@cs.uiowa.edu`

The University of Iowa, Department of Computer Science



Transducers

- Transducers are DFA that produce output;
- Transducers have interesting applications in natural language processing;
- Textbook introduces transducers shallowly;
- We follow Fleck's book for a more detailed presentation of transducers.



Transducer by Sipser

- Problem 1.24 in Sipser's book introduces transducers as follows:

A finite state transducer (FST) is a type of finite automaton whose output is a string and not just *accept* or *reject*.

- Sipser illustrates this definition with the two transducers in Figures 1,2, whose output are then examined when these machines receive some input.



Example 1

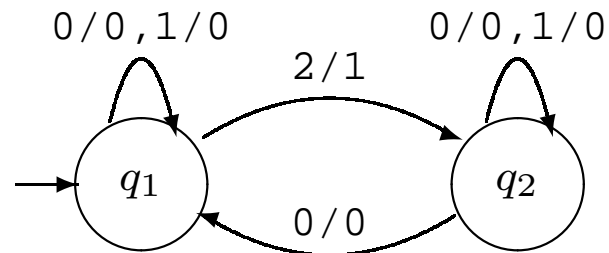


Figure 1: Transducer T_1

Computations:

On input $w = 011$ T_1 performs: $q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1$ and outputs 000;

On input $w = 211$ T_1 performs: $q_1 \rightarrow q_2 \rightarrow q_2 \rightarrow q_2$ and outputs 100.



Example 2

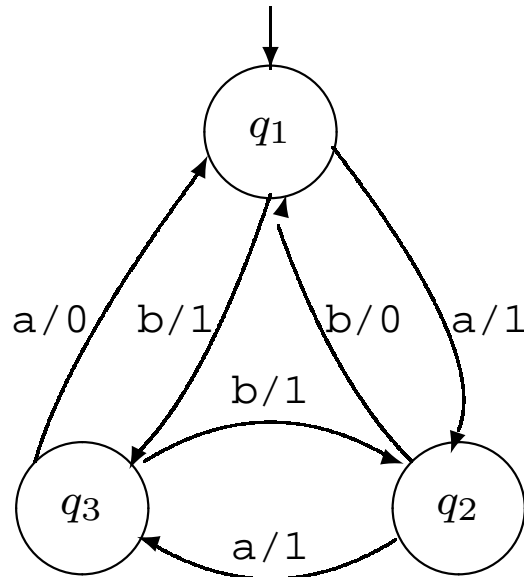


Figure 2: Transducer T_2

Computations: On input b T_2 performs $q_1 \rightarrow q_3$ and outputs 1

On input ϵ T_2 performs q_1 and outputs ϵ

On input $bbab$ T_2 performs $q_1 \rightarrow q_3 \rightarrow q_2 \rightarrow q_3 \rightarrow q_2$ and outputs 1111.



A formal definition

A finite state transducer is a 5-tuple $T = (Q, \Sigma, \Gamma, \delta, q_0)$ where

1. Q is a finite set called the set of states of T ;
2. Σ is a finite set of symbols called the input alphabet of T ;
3. Γ is a finite set of symbols called the output alphabet of T ;
4. $\delta : Q \times \Sigma \rightarrow Q \times \Gamma$ is the transition function of T ;
5. $q_0 \in Q$ is the start state of T .

Note: unlike a DFA a transducer has no final states!



Computation

Let $T = (Q, \Sigma, \Gamma, \delta, q_0)$ be a transducer. For an input $w = w_1w_2 \dots w_n \in \Sigma^*$, T outputs a string $v = v_1v_2 \dots v_n \in \Gamma^*$ if a sequence of states $r_0, r_1, \dots, r_n \in Q$ exists such that the following two conditions hold:

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = (r_{i+1}, v_{i+1})$, for $i = 0, 1, \dots, n - 1$.

Note: Obviously T compute a function $f_T : \Sigma^* \rightarrow \Gamma^*$.

How can we describe this function ?



Computation model

- Transducers regard the input to a DFA as being transformed by the machine into the output.
- This may be considered as a more realistic model of computation than simple acceptance or rejection.
- Fleck observed that adding output to the DFA amounts to extending the model of computation.



Transition systems

Definition: A deterministic transition system, DTS, is a triple $T = (Q, \Sigma, \delta)$, where Q is a finite set of states, Σ is an alphabet, and $\delta : Q \times \Sigma \rightarrow Q$ is the transition function.

Note: the set of states is regarded here as the memory of computation:

- A DTS captures the way memory configurations of a computation device change when a single input symbol is presented to it;
- The memory (states) permit the model to retain information about its past input; this may affect the reaction to later input symbols.



Observations

- In a real computer the states are mappings from variables to values while transitions are operators (instructions) mapping states to states;
- Past behavior of a computation can be seen as the **computation trace** consisting of the string of transitions performed so far.



Extending δ

If $T = (Q, \Sigma, \delta)$ is a DTS then $\delta^* : Q \times \Sigma^* \rightarrow Q$ is defined inductively for all $q \in Q$ by:

- $\delta^*(q, \epsilon) = q$;
- For all $w \in \Sigma^*$ and $x \in \Sigma$, $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$

Example:

$$\delta^*(q, x_1x_2) = \delta(\delta^*(q, x_1), x_2) = \delta(\delta(\delta^*(q, \epsilon), x_1), x_2) = \delta(\delta(q, x_1), x_2)$$

Note: δ^* consumes the input from left to the right, one symbol at the time.



Example 3

Take $Q = \{q_0, q_1, q_2\}$ and $\Sigma = \{0, 1\}$. Define δ by the Table:

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_2

That is: $\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$, $\delta(q_1, 0) = q_2$, $\delta(q_1, 1) = q_1$,
 $\delta(q_2, 0) = q_2$, $\delta(q_2, 1) = q_2$.

The transition diagram of this system is in Figure 3



Transition diagram

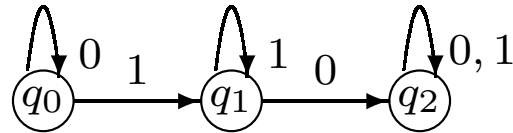


Figure 3: Example transition system

Essential: on a DTS transition diagram we have:

- Transition function show how states change;
- δ is described by edges and records the next state;
- δ^* is described by *paths*, i.e., sequence of edges.



Deterministic finite acceptor

- The concept of a deterministic finite acceptor is the same as the concept of deterministic finite automaton, DFA.
- Using the notion of transition system, a DFA A is defined by the 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$ where:
 1. (Q, Σ, δ) is a DTS;
 2. $q_0 \in Q$ is the start state;
 3. $F \subseteq Q$ is the set of accepting states.



Contrasting diagrams

The transition diagram of a DFA differ from the transition diagram of a DTS, see Figure 4, by:

1. q_0 is marked by an arrow from nowhere;
2. states in F are double-circled.

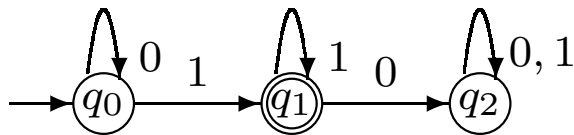


Figure 4: Example transition system



Transducers

Definition: A transducer M is a 6-tuple $M = (Q, \Sigma, \Delta, \delta, \rho, q_0)$ where:

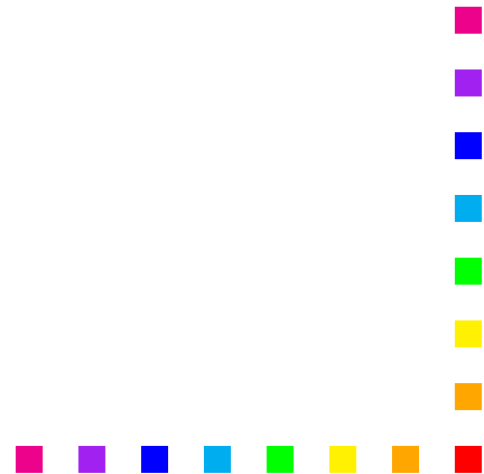
1. (Q, Σ, δ) is a deterministic transition system;
2. Δ is a finite non-empty set called the output alphabet;
3. $q_0 \in Q$ is the start state;
4. $\rho : Q \times \Sigma \rightarrow \Delta^*$ is the output function.

Note: termination occurs when all input has been consumed.



Terminology

Fleck calls transducers Deterministic Generalized Sequential Machines (DGSM).



Extending ρ

Here we extend $\rho : Q \times \Sigma \rightarrow \Delta^*$ to $\rho^* : Q \times \Sigma^* \rightarrow \Delta^*$ by:

- $\rho^*(q, \epsilon) = \epsilon$, for each $q \in Q$
- $\rho^*(q, wx) = \rho^*(q, w)\rho(\delta^*(q, w), x)$
for all $w \in \Sigma^*$ and $x \in \Sigma$

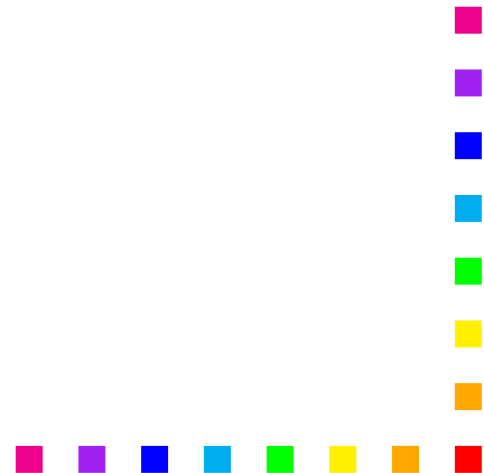
Note: $\rho^*(q, wx)$ is obtained by concatenating $\rho^*(q, w)$ and $\rho(\delta^*(q, w), x)$.



Consequence

The extension of ρ to ρ^* allows us to regard M as defining a function $M : \Sigma^* \rightarrow \Delta^*$.

Namely $M(w) = \rho^*(q_0, w)$ and refer to it as the DGSM function.



DGSM Computation

- Computation performed by a transducer parallels that of a DFA except that there is a sequence of outputs as well as a sequence of states;
- Since ρ^* captures exactly this information we do not formalize further this notion.

Note: a transducer terminates when all symbols of the input are consumed.



Example transducer

Consider the transducer $M = (Q, \Sigma, \Delta, \delta, \rho, q_0)$ where $\Sigma = \Delta = \{a, b\}$ which copies its input to its output while deleting the second occurrence (if any) of the symbol a .

Note:

- The state diagram of this machine is in Figure 5.
- The arrows are labeled by y/x where $y \in \Sigma$ is the input and $x \in \Delta^*$ is the output.



Transition diagram of M

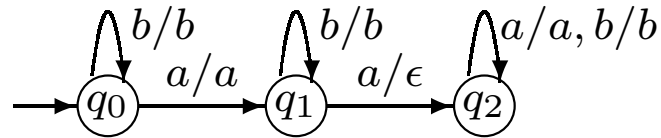
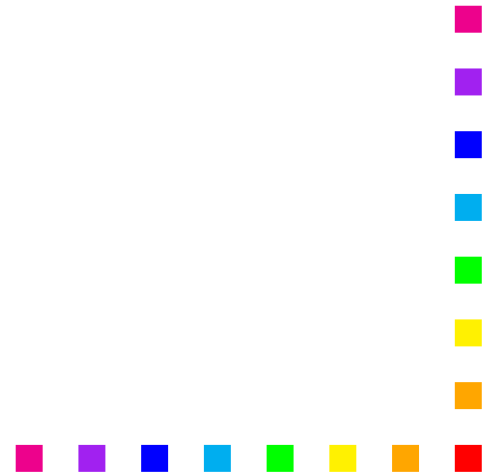


Figure 5: Example transition system



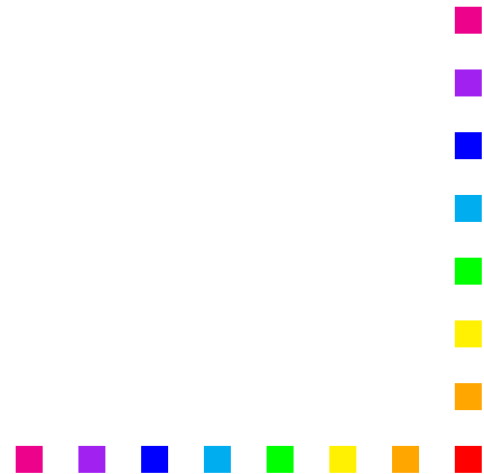
Machine M at work

- This machine copies “b”s in state q_0 until first “a” occurs;
- When first “a” occurs, M copies it to the output and then goes to state q_1 . In this state it copies “b”s to output until first “a” occurs;
- When first “a” occurs in state q_1 the machine removes it and goes to state q_2 where it copies all input to output.



Facts

- The length of any symbol in the alphabet is 1;
- DGSM may output strings of length 0; this justifies the usage of ρ^* .



Example 4

In this example $\Sigma = \Delta = \{0, 1, \#\}$ and the DGSM M_2 computes the odd parity, i.e., it transforms $w\#$ where $w \in \{0, 1\}^*$ into $w\pi\#$, $\pi \in \{0, 1\}$ such that the sequence $w\pi$ contains an odd number of 1s, Figure 6.

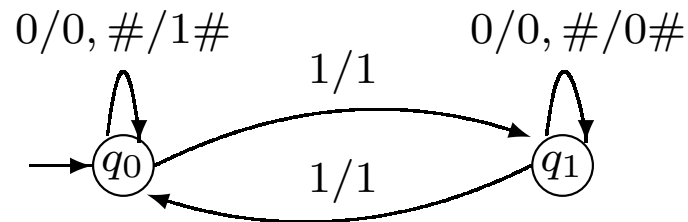


Figure 6: DGSM M_2 computing odd parity

Interpretation

- M_2 is in state q_0 whenever input has even parity and it is in state q_1 whenever input has an odd parity;
- When the input $\#$ (i.e., end of string) is encountered in state q_0 the machine adds 1 to the output thus making it of the odd parity;
- When the input $\#$ is encountered in state q_1 the machine adds a 0 to the output thus maintaining the odd parity.



Observations

1. With DFA our interest is in the language (i.e., a set of strings) a DFA D recognizes;
2. With transducers our interest is in the function $f : \Sigma^* \rightarrow \Delta^*$ a transducer M computes (i.e., the transformation the input undergoes).



Extendible functions

Definition: A function $f : \Sigma^* \rightarrow \Delta^*$ is called extendible if $f(\epsilon) = \epsilon$ and for each $x, y \in \Sigma^*$, there is $z \in \Delta^*$ so that $f(xy) = f(x)z$

Note:

- For an extendible function, results of the function acting on prefixes of an argument occur as prefixes of the result of a longer argument;
- This captures the DGSM characteristic of output accumulating as input is processed from left to right.



Observations

- Functions realized by DGSMs are always extendible, i.e., $\forall x, y \in \Sigma^*$:

$$M(xy) = \rho^*(q_0, xy) = \rho^*(q_0, x)\rho(\delta^*(q_0, x), y) = M(x)z$$

- Consequently, if a string-to-string function is not extendible, it is not computable by a DGSM;
- But not all extendible functions are computable by DGSM.



Example 5

Consider the function $f : \{a, b\}^* \rightarrow \{0\}^*$ defined by: $f(x) = 0^k$ where x contains k "a"s

Note: if $f(xy) = 0^m$ then xy contains m "a"s and so $f(x) = 0^k, k \leq m$.

Hence, $f(xy) = 0^k 0^{m-k}$, i.e., $f(xy) = f(x)0^{m-k}$ and thus f is extendible.



Example 6

Consider the function $f : \{a, b\}^* \rightarrow \{0, 1\}^*$ defined by:

$$f(x) = \begin{cases} 0^k, & \text{if } x \text{ contains } k \text{ more "a"s than "b"s} \\ 1^k, & \text{if } x \text{ contains } k \text{ more "b"s than "a"s} \\ \epsilon, & \text{if } x \text{ contains the same number of "a"s and "b"s.} \end{cases}$$

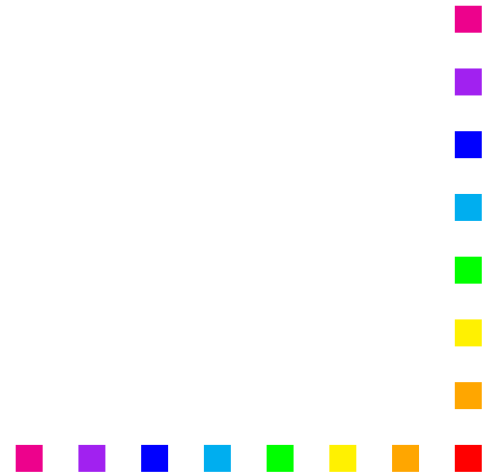
Since $f(a) = 0$ while $f(ab) = \epsilon$ and $\epsilon \neq f(a)z$ for some z , it results that f is not extendible.



Fact

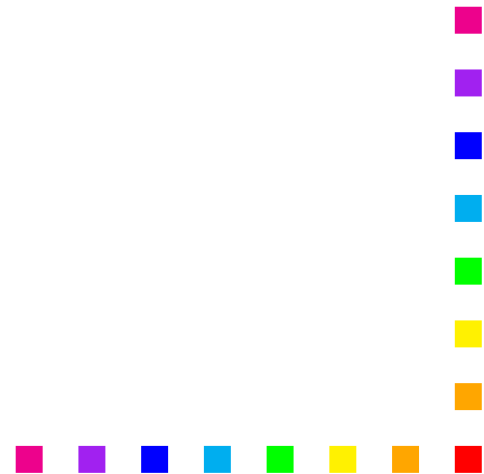
For a given extendible function f and string x , for each string y the string z (whose existence is guaranteed by the definition, $f(xy) = f(x)z$) is uniquely determined by y .

Hence: $f(x)$ and $f(xy)$ are uniquely determined.



Conclusion

- There can be only one z that can satisfy the extensibility conditions.
- That is, with f and x fixed, z is a function of y .



Derived function

Given an extendible function $f : \Sigma^* \rightarrow \Delta^*$ and $x \in \Sigma^*$, the function $f_x : \Sigma^* \rightarrow \Delta^*$ defined by: $\forall y \in \Sigma^* f_x(y) = z$ if $f(xy) = f(x)z$ is called a **derived function associated to f** .

Note: if f is extendible then:

- $\forall x, y \in \Sigma^* [f(xy) = f(x)f_x(y)]$;
- Since Σ^* is infinite there is a potential for infinite many derived functions associated to f ;
- However, it may also be that $f_{x_1} = f_{x_2}$ even though $x_1 \neq x_2$.



Definition

An extendible function $f : \Sigma^* \rightarrow \Delta^*$ is said to be of finite index if its set of distinct derived function $\{f_x | x \in \Sigma^*\}$ is finite.

Cardinality of $\{f_x | x \in \Sigma^*\}$ is called the index of f ; f is of infinite index if $|\{f_x | x \in \Sigma^*\}| = \infty$.

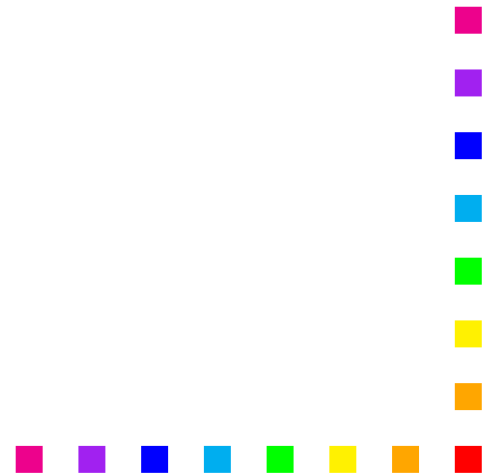


Example 7

Construct the derived function of the identity $id : \Sigma^* \rightarrow \Sigma^*$ defined by $\forall x \in \Sigma^* [id(x) = x]$.

- Since $id(xy) = id(x)y$, id is extendible.
- Since $id_x = id$ for all $x \in \Sigma^*$, i.e., $id_x(y) = y = id(y)$, the index of id is 1

That is, id is of finite index.



Example 8

Construct the derived function of the *unit delay* function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ defined by:

$f(\epsilon) = \epsilon$, $f(w_1w_2 \dots w_k) = w_1w_2 \dots w_{k-1}$, for $k \geq 1$ and $w_i \in \Sigma$, $1 \leq i \leq k$.

- $f(w_1) = \epsilon$, $f(w_1w_2) = f(w_1)w_1$, $f(w_1w_2w_3) = f(w_1w_2)w_2$, \dots
and f is clearly extendible.



Derived functions for f

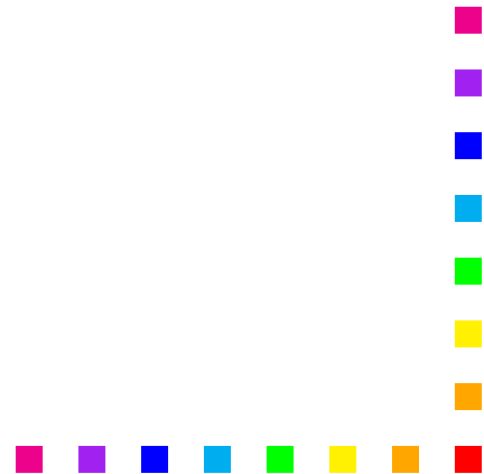
There are three distinct derived functions for f :

1. $f(0) = f(1) = \epsilon$ and $\forall x, y \in \Sigma^*$, and $y \neq \epsilon$, $f(0y) = 0f(y)$ from the definition of f
2. $f(0y) = f(0)f_0(y)$ by extensibility condition; but $f(0) = \epsilon$, so $f_0(y) = f(0y) = 0f(y)$; similarly $f_1(y) = 1f(y)$.
3. $f(x0y) = x0f(y)$ (by definition) and $f(x0y) = f(x0)f_{x0}(y)$ (by extensibility); but $f(x0) = x$ (def) and thus $f(x0y) = xf_{x0}(y)$. I.e., $x0f(y) = xf_{x0}(y)$, hence, $\forall x \in \Sigma^* [f_{x0} = f_0]$;
4. Similarly, $\forall x \in \Sigma^* [f_{x1} = f_1]$ and $f_\epsilon = f$.



Conclusion

$$\{f_x \mid x \in \Sigma^*\} = \{f_\epsilon, f_0, f_1\}.$$



Fundamental results

- If $f : \Sigma^* \rightarrow \Delta^*$ is extendible then $f_\epsilon = f$ and $\forall x \in \Sigma^* [f_x(\epsilon) = \epsilon]$;
- A function $f : \Sigma^* \rightarrow \Delta^*$ is a DGSM function iff it is extendible and of finite index;
- If M is a DGSM and $L \subseteq \Sigma^*$ is regular then $M(L) = \{w \in \Delta^* \mid \exists x \in L \wedge M(x) = w\}$ is regular.

See proof in Fleck, pages 143–147



Problem 3.10

One of the functions $f, g : \{0, 1\}^* \rightarrow \{0, 1\}^*$ defined below is extendible and of finite index (i.e., can be realized by a DGSM) and one is not. Which is which and why?

f : For each $x \in \{0, 1\}^*$, $f(x) = w$ where w is obtained from x by deleting all instances of 1

g : $g(\epsilon) = \epsilon$ and for each $w \in \{0, 1\}^*$ and $x \in \{0, 1\}$, $g(wx) = g(w)wx$



Solution

Function f is extendible and has index 1.
This is because:

$\forall x, y \in \Sigma$, $f(xy)$ is xy with all '1'-s removed. This is the same as $f(x)f(y)$. That is, $f(xy) = f(x)f_x(y) = f(x)f(y)$, and thus $f_x(y) = f(y)$ for every x . Therefore one-state DGSM realizes f .



More on solution

Function g is extendible but not of finite index.

g is extendible because for $y = x_1x_2 \dots x_k$ we have:

$$g(wy) = g(wx_1x_2 \dots x_k) = g(wx_1x_2 \dots x_{k-1})wx_1x_2 \dots x_k \text{ (by definition)}$$

$$g(wy) = g(wx_1x_2 \dots x_{k-2})wx_1x_2 \dots x_{k-1}wx_1x_2 \dots x_k \text{ (second applic.)}$$

$$g(wy) = g(w)wx_1wx_1x_2w \dots wx_1x_2 \dots x_{k-1}wx_1x_2 \dots x_k.$$

However g is not of finite index because:

$$g(01) = g(0)01 = g(0)g_0(1), \text{ so } g_0(1) = 01$$

$$g(001) = g(00)001 = g(00)g_{00}(1), \text{ so } g_{00}(1) = 001$$

$$g(0001) = g(000)0001 = g(000)g_{000}(1), \text{ so } g_{000}(1) = 0001, \dots \text{ That is}$$

functions g_x for $x = 0, 00, \dots, 0^k, \dots$ are all different. Therefore g cannot

be realized by any DGSM.



Assignment (100 points)

Solve the following problems taken from Fleck's book, page 181. You receive 25 bonus-points for each problem correctly solved.

Problem 3.7 Show that the function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ defined by:

$$f(x) = \begin{cases} 0^n 1^n, & \text{if } x = 0^{2n} \\ x, & \text{otherwise} \end{cases}$$

cannot be realized by a DGSM.



Problem 3.8

Given $L \subseteq \Sigma^*$, define the function $f : \Sigma^* \rightarrow \{0, 1\}^*$ inductively as follows:

1. $f(\epsilon) = \epsilon$
2. For each $w \in \Sigma^*$ and $x \in \Sigma$

$$f(wx) = \begin{cases} f(w)0, & \text{if } wx \notin L \\ f(w)1, & \text{if } wx \in L \end{cases}$$

Show that f is a DGSM function iff L is regular.



Problem 3.12

Consider the function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ defined by:

- $f(\epsilon) = \epsilon$, $f(0) = 0$, $f(1) = 1$
- For $w_i \in \{0, 1\}$ and $k \geq 2$,
$$f(w_1w_2 \dots w_k) = w_1w_2w_1w_3 \dots w_1w_kw_1$$

Determine whether or not f is a DGSM function and prove it.

