

Application of DGSM to DFA State Minimization

Teodor Rus

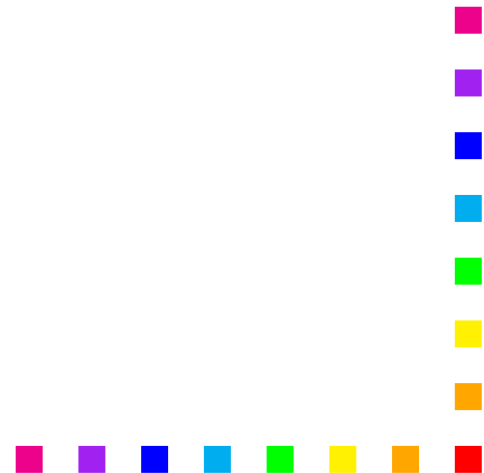
`rus@cs.uiowa.edu`

The University of Iowa, Department of Computer Science



Observations

1. DFA simulators are easy to implement;
2. NFA-s are easy to design but their simulation requires parallelism in their manipulation;
3. Mapping NFA to DFA can be carried out automatically. But the resulting DFA is exponential in the number of NFA states.



Processing technology

1. Specify languages using NFA-s;
2. Map the NFA-s into DFA-s and optimize the resulting DFA.

Note: DFA-optimization requires us to develop algorithms that map a DFA into an "equivalent" DFA with a minimum number of states.



Facts

1. Sipser does not approach the problem of DFA minimization;
2. Hopcroft, Motwany, and Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley 2001, dedicates Section 4.4 (page 154) to this problem. But do not offer an easy understandable and implementable algorithm to solve it;
3. Fleck approaches this problem using DGSM-s and develop a manageable algorithm that map a DFA into an equivalent DFA with a minimum number of states.



Definitions

1. Let $M = (Q_m, \Sigma, \Delta, \delta_m, \rho_m, q_m^0)$ and $N = (Q_n, \Sigma, \Delta, \delta_n, \rho_n, q_n^0)$ be two DGSMs with $\Sigma_M = \Sigma_N = \Sigma$ and $\Delta_m = \Delta_n = \Delta$. For each integer $k > 0$, state $q \in Q_m$ is k -equivalent to state $p \in Q_n$ if $\rho_m(q, w) = \rho_n(p, w)$ for all $w \in \Sigma^*$ with $|w| \leq k$ and write it $q \stackrel{k}{\equiv} p$.
2. States q and p are equivalent, written $q \equiv p$, if $q \stackrel{k}{\equiv} p$ for all $k \geq 0$;
3. A DGSM is called distinguished provided it has no two distinct states that are equivalent.



Fact

k -equivalences are equivalent relations, as one can check directly:

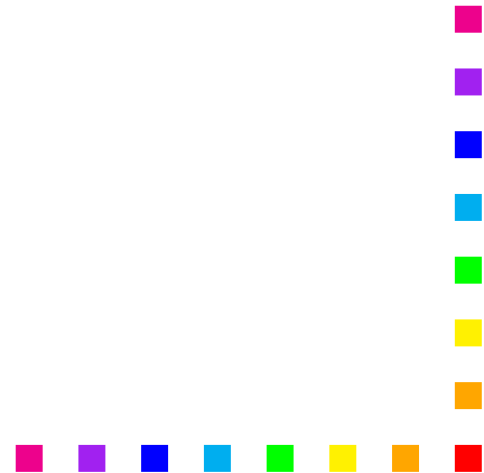
1. Reflexive: $q \stackrel{k}{\equiv} q$ obviously;
2. Symmetric: If $q \stackrel{k}{\equiv} p$ then $p \stackrel{k}{\equiv} q$ because $\rho(q, w) = \rho(p, w)$ implies $\rho(p, w) = \rho(q, w)$;
3. Transitive: result from transitivity of set equality.



Notation

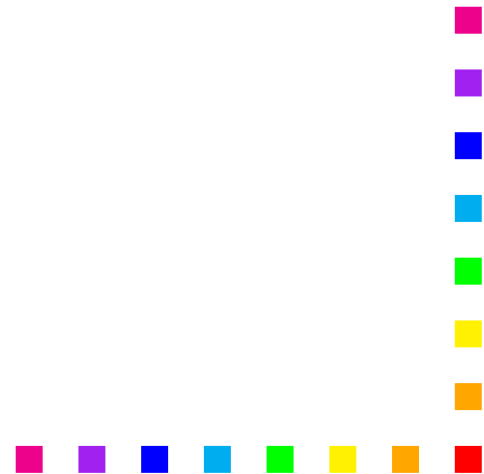
Π and Π_k are classes of equivalence defined by:

1. $[q] \in \Pi$ is defined by $[q] = \{p \mid q \equiv p\}$
2. $[q]_k \in \Pi_k$ is defined by $[q]_k = \{p \mid q \stackrel{k}{\equiv} p\}$



DGSM equivalence

Two DGSM M , N , with input alphabets $\Sigma_m = \Sigma_n$ and output alphabets $\Delta_m = \Delta_n$ are equivalent provided that for each $q \in Q_m$ there is $p \in Q_n$ such that $q \equiv p$, and conversely.



Questions

- Given that Σ^* is infinite is there a finite process to determine state equivalence?
- Can k -equivalence be used to develop a series of increasingly better approximations that converge to the equivalence we seek?



Example equivalences

Consider DGSM M in Figure 1

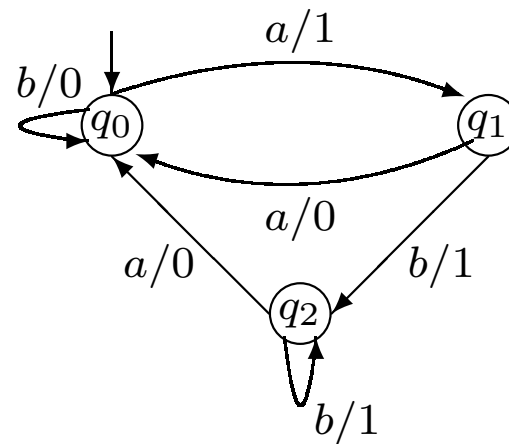
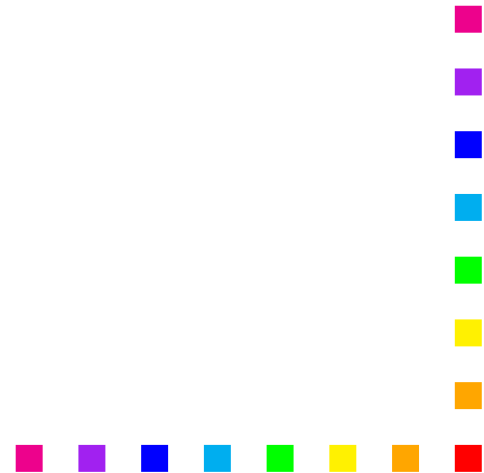


Figure 1: DGSM M



Facts

1. q_0 and q_1 are not 1-equivalent because $\rho(q_0, a) = 1$ $\rho(q_1, a) = 0$;
2. q_1 and q_2 are 1-equivalent because $\rho(q_1, a) = 0$, $\rho(q_2, a) = 0$ and $\rho(q_1, b) = 1$, $\rho(q_2, b) = 1$;
3. $\Pi_1 = \{[q_0], [q_1, q_2]\}$
4. One can verify by observing that:
 - (a) for any input beginning with "a" both states in $[q_1, q_2]$ initially yield "0" and then continue from $[q_0]$
 - (b) for any input beginning with "b" both states in $[q_1, q_2]$ yield "1" and then continue with $[q_1, q_2]$

Hence they are equivalent. i.e., in this case $\Pi = \Pi_1$



A more challenging example

Consider DGSM M in Figure 2

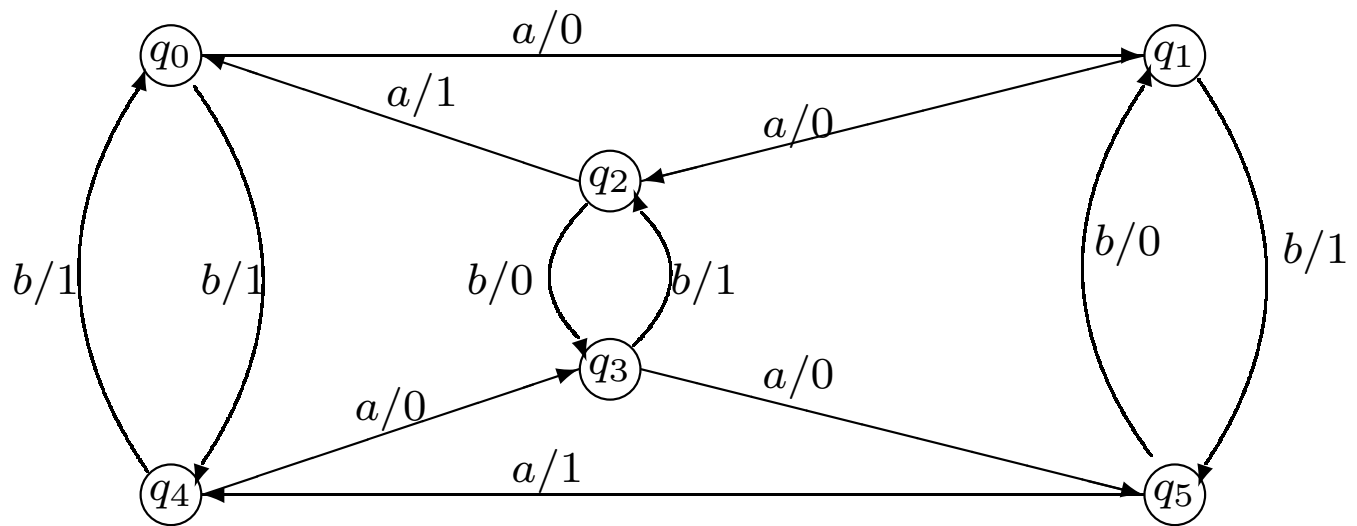


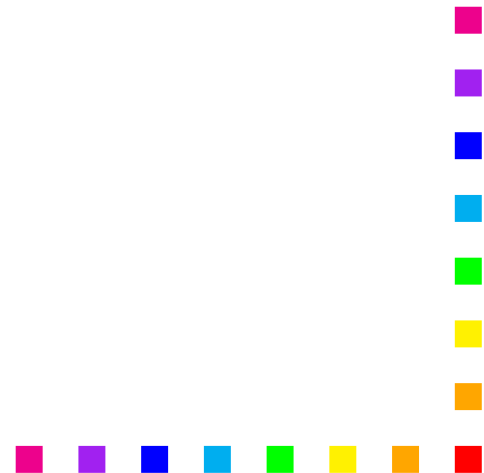
Figure 2: DGSM M



By direct inspection

Classes of 1-equivalence:

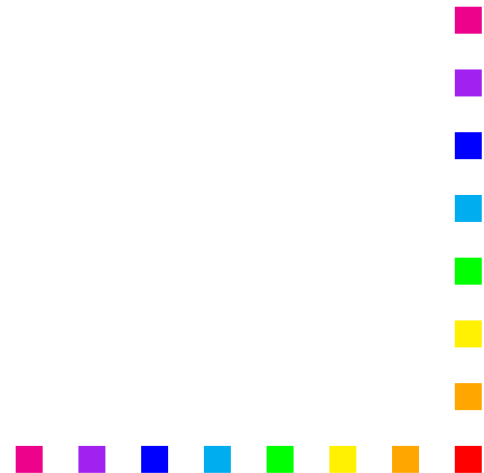
- $q_2 \stackrel{1}{\equiv} q_5$: $\rho(q_2, a) = \rho(q_5, a) = 1$, $\rho(q_2, b) = \rho(q_5, b) = 0$;
Note: q_0 is not equivalent to q_2 or q_5 since $\rho(q_0, b) = 1$.
- q_0, q_1, q_3, q_4 are all 1-equivalent as they yield output 0 for input "a" and 1 for input "b";
That is, $\Pi_1 = [q_0, q_1, q_3, q_4], [q_2, q_5]$.



Observation

Classes of equivalence are more difficult to show.

1. q_0 and q_3 are not 2-equivalent because on "ab", q_0 yield "01" and q_3 yields "00";
2. however, q_2 and q_5 are 2-equivalent, as can be easily verified.



Major result

Let $M = (Q, \Sigma, \Delta, \delta, \rho, q_0)$ be a DGSM and $q, p \in Q$. Then for every $k > 0$:

1. $q \stackrel{k+1}{\equiv} p$ iff $q \stackrel{1}{\equiv} p$ and $\delta(q, x) \stackrel{k}{\equiv} \delta(p, x)$ for all $x \in \Sigma$;
2. $\Pi_{k+1} = \Pi_k$ implies $\Pi_n = \Pi_k$ for all $n > k$;
3. $\Pi_{k+1} = \Pi_k$ implies $q \equiv p$ iff $q \stackrel{k}{\equiv} p$.

Proof: see Fleck 151-152.



Fact

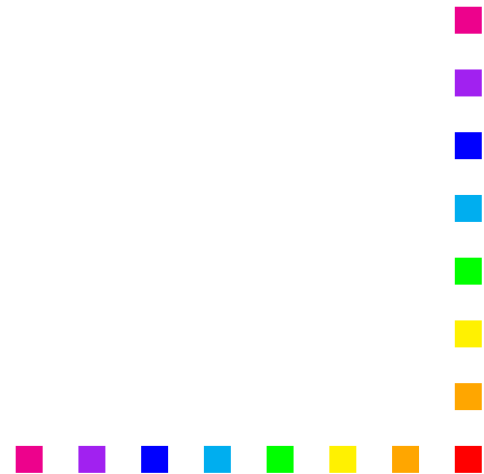
- The key insight revealed by this result is that we can tell whether or not two k -equivalent states are also $k + 1$ -equivalent by examining k -equivalence of the next states under all letters of Σ .
- **Consequence:** from the analysis of k -equivalence and single transitions we can deduce $k + 1$ equivalence.



Theorem 3.2.2 (Fleck)

Let DGSM $M = (Q, \Sigma, \Delta, \delta, \rho, q_0)$ where $|Q| > n - 2$ states. $\forall q, p \in Q [q \equiv p]$ iff $q \stackrel{n-1}{\equiv} p$

Proof: See Fleck 152,152



Application

The application of theorem 3.2.2 is the following algorithm for determining the equivalence of states in a DGSM:

1. Compute Π_1 by inspection of ρ for single letter input;
2. Repeat for $k = 1, 2, \dots$ until no class split or $k = n - 1$:

Given Π_k compute Π_{k+1} by:

for each pair $q \stackrel{k}{\equiv} p$:

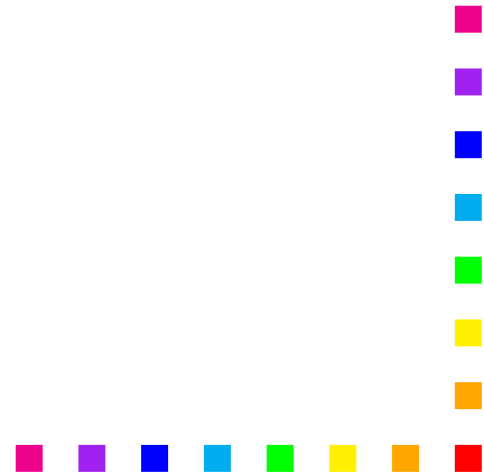
- if $\delta(q, x) \stackrel{k}{\equiv} \delta(p, x)$ for all $x \in \Sigma$ then $q \stackrel{k+1}{\equiv} p$;
- otherwise split q and p , that is, set q and p in different classes in Π_{k+1} .

3. At termination set $\Pi = \Pi_{k+1}$.



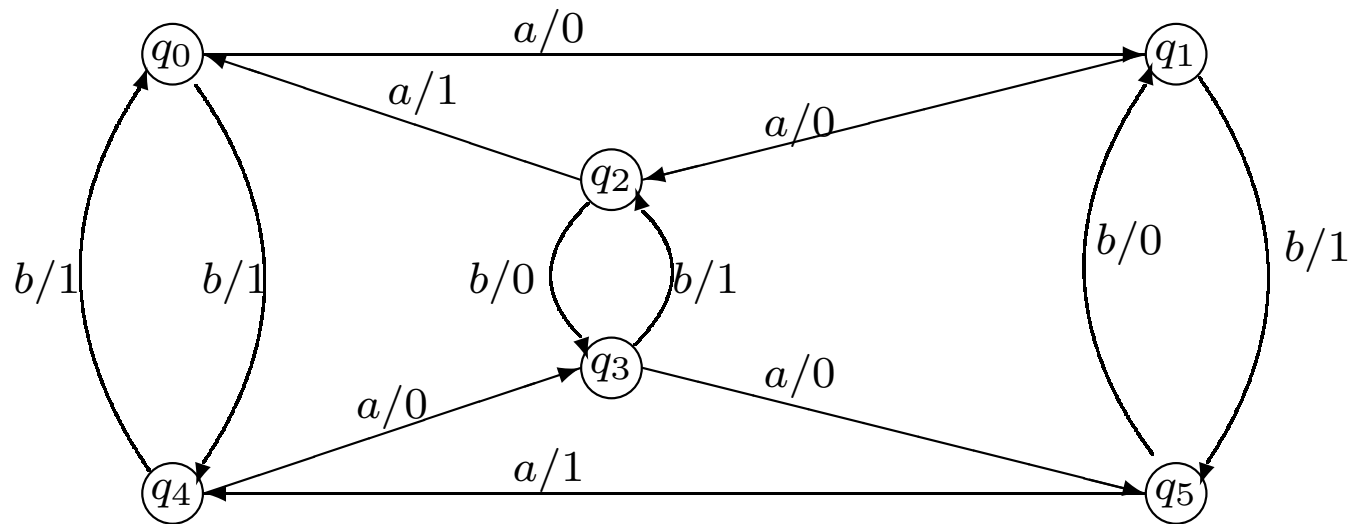
Observations

1. $\Pi_k = \{C_1^k, C_2^k, \dots, C_m^k\}$;
2. if $p \stackrel{k}{\equiv} q$ and $\delta(q, x) \stackrel{k}{\equiv} \delta(p, x)$ it means that $\exists C_i^k, C_j^k \in \Pi_k$, such that $p, q \in C_i^k$ and $\delta(p, x), \delta(q, x) \in C_j^k$
3. Condition $\delta(q, x)$ not k -equivalent to $\delta(p, x)$ means that $\delta(p, x) \in C_j^k$ and $\delta(q, x) \in C_r^k$ for $j \neq r$.



Applying the algorithm

Consider again the DGSM M in the figure:



Equivalence classes

1. Step 1: as we already know, $\Pi_1 = \{[q_0, q_1, q_3, q_4], [q_2, q_5]\}$

2. Step 2: $k = 1$:

$\delta(q_0, a) = q_1, \delta(q_1, a) = q_2$; since q_1 and q_2 are not in the same class q_0, q_1 are not 2-equivalent; hence q_0 and q_1 are placed in different classes in Π_2

$\delta(q_0, a) = q_1, \delta(q_4, a) = q_3, q_1, q_3$ are 1-equivalent; $\delta(q_0, b) = q_4, \delta(q_4, b) = q_0, q_0, q_4$ are 1-equivalent. Hence, q_0, q_4 are 2-equivalent. Similarly, $[q_2, q_5]$ does not split. Consequently, $\Pi_2 = \{[q_0, q_4], [q_1, q_3], [q_2, q_5]\}$

Step 2; $k = 2$: Similar analysis shows $\Pi_3 = \Pi_2$

3. Step 3: termination occurs with

$$\Pi_2 = \Pi_3 = \Pi = \{[q_0, q_4], [q_1, q_3], [q_2, q_5]\}$$



Theorem 3.2.3 (Fleck)

In a distinguished DGSM M with n states, for each k , $1 \leq k < n$, a k -equivalence has at least $k + 1$ equivalence classes, and no k -equivalence class has more than $n - k$ states

Proof: by induction on k .

Note: For $n = 1$ this is vacuously true because there is no k with $1 \leq k < 1$.



Induction basis: $k = 1$

- When $n > 1$, if 1-equivalence were to have a single equivalence class then for any two states, p and q , and any letter $w \in \Sigma$,
 $\delta(p, w) = \delta(q, w)$
- Then by Lemma 3.2.1, $q \stackrel{2}{\equiv} p$; i.e., $\Pi_2 = \Pi_1$ and again, by the same lemma $\Pi_2 = \Pi_1 = \Pi$
- This implies that all states are equivalent, which contradicts the assumption that M is distinguished and $n > 1$. Hence, 1-equivalence has at least 2 classes



Induction step: $k := k + 1$

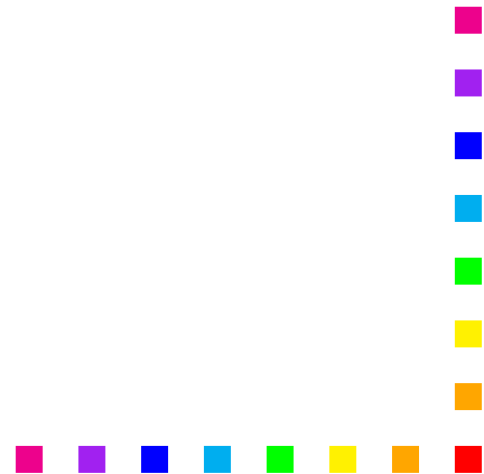
Assume that for $1 \leq k \leq n - 2$, k -equivalence has at least $k + 1$ classes and consider $(k + 1)$ -equivalence.

- Since Π_{k+1} is a refinement of Π_k , either (a) Π_{k+1} has more classes than Π_k or (b) $\Pi_{k+1} = \Pi_k$;
- In case (a) Π_{k+1} has at least $k + 2$ classes (because Π_k has $k + 1$ classes by induction step);
- In case (b), by Lemma 3.2.1, $\Pi_{k+1} = \Pi_n$ and therefore has $n > k + 1$ classes.



Conclusion

Since each equivalence class contains at least one state, if there are $k + 1$ classes, k of them contain at least k elements, leaving at most $n - k$ for the other class.



Lemma 3.2.4

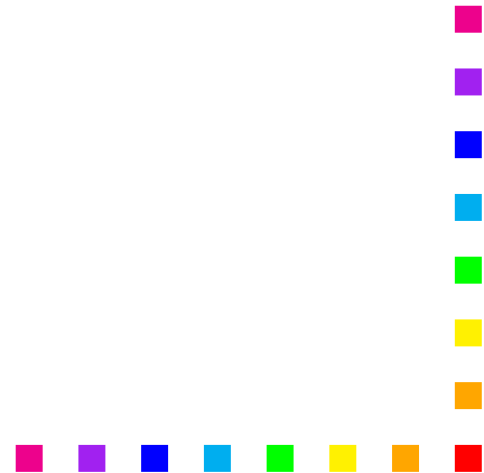
Let

$$M = (Q_1, \Sigma, \Delta, \delta_1, \rho_1, q_0)$$

$$N = (Q_2, \Sigma, \Delta, \delta_2, \rho_2, p_0)$$

be DGSMs and $q \in Q_1, p \in Q_2$.

If $q \equiv p$ then $\delta_1^*(q, x) \equiv \delta_2^*(p, x)$ for all $x \in \Sigma^*$



Proof

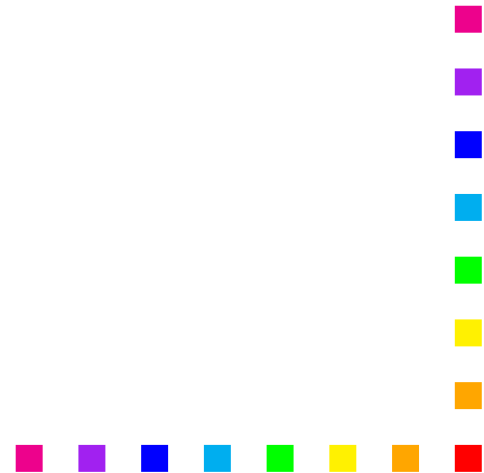
By contradiction: assume that $q \equiv p$ but $\delta_1^*(q, x) \neq \delta_2^*(p, x)$ for some $x \in \Sigma^*$. Then there is $y \in \Sigma^*$ so that $\rho_1(\delta_1(q, x), y) \neq \rho_2(\delta_2(p, x), y)$.

Then: $\rho_1(q, xy) = \rho_1(q, x)\rho_1(\delta_1(q, x), y) \neq$

$\rho_1(q, x)\rho_2(\delta_2(p, x), y) = \rho_2(p, x)\rho_2(\delta_2(p, x), y) = \rho_2(p, xy)$

which contradicts the equivalence of q and p and hence,

$\delta_1(q, x) = \delta_2(p, x)$ for all $x \in \Sigma^*$



Isomorphic DGSMs

Let $M = (Q_1, \Sigma, \Delta, \delta_1, \rho_1, q_0)$ and $N = (Q_2, \Sigma, \Delta, \delta_2, \rho_2, p_0)$ be DGSMs.

M is state isomorphic to N if there is a function $\alpha : Q_1 \rightarrow Q_2$ which is one-to-one and onto (i.e., a bijection) so that for all $q \in Q_1$ and $w \in \Sigma$:

- (i) $\rho_1(q, w) = \rho_2(\alpha(q), w)$, and
- (ii) $\alpha(\delta_1(q, w)) = \delta_2(\alpha(q), w)$

See Figure 3 for the graphic illustration of this definition.



Isomorphism, illustration

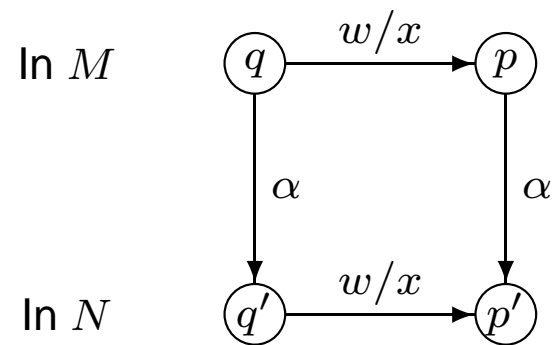


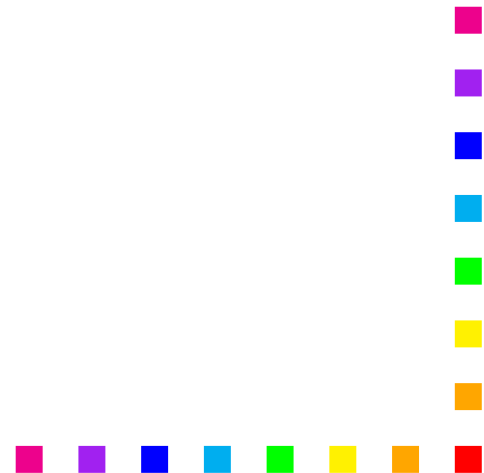
Figure 3: DGSM Isomorphism

Note: the critical structural relationship is: for each transition in M , $\delta_1(q, w) = p$, if $\alpha(q) = q'$ and $\alpha(p) = p'$ then $\delta_2(q', w) = p'$ in N .



Facts

- If two DGSMs do not have the same number of states, they cannot be isomorphic;
- For two DGSMs with n states, there are $n!$ different 1-1 and onto mappings to consider as potential isomorphisms.



Fundamental theorem

Among all DGSMs equivalent to a given DGSM M , there is a unique one (up to isomorphism) which has the fewest states and is distinguished. This is called the reduced machine of M .

Proof: by construction. We will show that if $M = (Q, \Sigma, \Delta, \delta, \rho, q_0)$ then its reduced machine is $N = (\Pi, \Sigma, \Delta, \delta', \rho', [q_0])$ where for all $[q] \in \Pi$ and $w \in \Sigma$:

$$\delta'([q], w) = [\delta(q, w)] \text{ and } \rho'([q], w) = \rho(q, w)$$



Observations

- Both functions δ' and ρ' are well defined;
- It is immaterial which of the states are chosen from the class;
- Thus, for $q, p \in [q]$, $q \equiv p$ and so $\rho(q, w) = \rho(p, w)$;
- By Lemma 3.2.1, $\delta(q, w) \equiv \delta(p, w)$, i.e., $[\delta(q, w)] = [\delta(p, w)]$.



Claim 1

For each $q \in Q$, $q \equiv [q]$ and hence $M \equiv N$.

Proof: by induction on the length of the input:

- **Induction basis:** $|w| = 0$: $\rho'([q], \epsilon) = \rho(q, \epsilon) = \epsilon$, for all $q \in Q$.
- **Induction step:** assume claim 1 for all $w \in \Sigma^*$, $|w| = k$, and consider the input wx , $x \in \Sigma$. Then we have:

$$\begin{aligned}\rho'([q], wx) &= \rho'([q], w)\rho'(\delta'([q], w), x) \text{ (by DGSM property)} \\ &= \rho(q, w)\rho'(\delta'([q], w), x) \text{ (by induction hypothesis)} \\ &= \rho(q, w)\rho'(\delta(q, w), x) \text{ (by } \delta' \text{ definition)} \\ &= \rho(q, w)\rho(\delta(q, w), x) \text{ (by } \rho' \text{ definition)} \\ &= \text{rho}(q, wx) \text{ (by DGSM property)}\end{aligned}$$

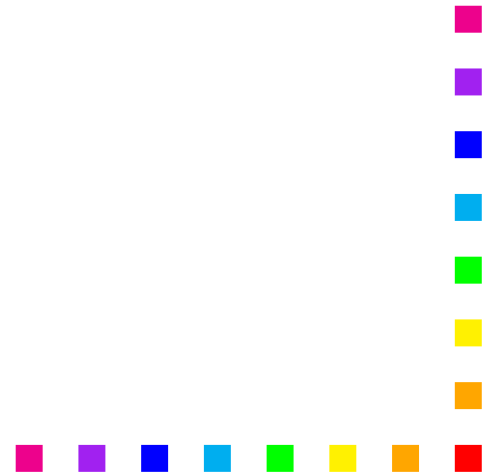
Since q and $[q]$ yield the same output for all inputs, this proves Claim 1.

Claim 2

N is distinguished.

Proof: let $[q]$ and $[p]$ be two distinct equivalence classes. Then $q \not\equiv p$ and $\exists x \in \Sigma^* [\rho'([q], x) = \rho(q, x) \neq \rho(p, x) = \rho'([p], x)]$.

Hence, $[q] \neq [p]$, thus proving Claim 2.



Claim 3

Any distinguished DGSM M_1 equivalent to M is isomorphic to N .

Proof: by construction. Let $M_1 = (Q_1, \Sigma, \Delta, \delta_1, \rho_1, q_1^0)$.

- Define $\alpha : Q_1 \rightarrow \Pi$ as follows: since $M_1 \equiv M$, for each $q_1 \in Q_1$ there is $q \in Q$ so that $q_1 \equiv q$. Let $\alpha(q_1) = [q]$. Since $M_1 \equiv M \equiv N$, α is total and onto. Since M_1 and N are both distinguished, α is one-to-one.
- One can check directly that α satisfies the properties of an isomorphism.



Example

Consider again DGSM M in Figure 4

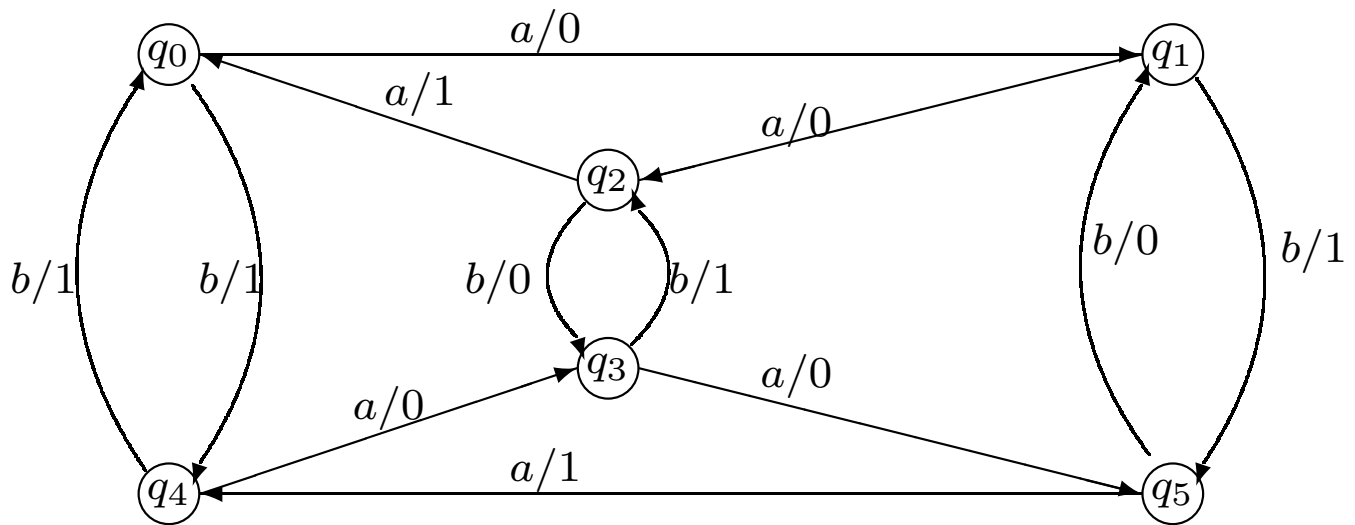
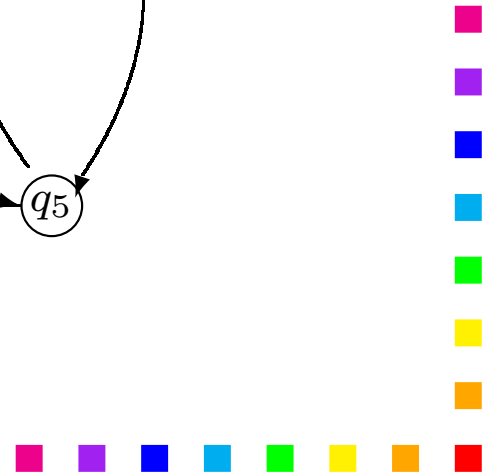


Figure 4: DGSM M



Reduced DGSM of M

The reduced machine is obtained by taking the classes $[q_0, q_4], [q_1, q_3], [q_2, q_5]$ as states and by associating the input/output behavior of the states in a class to the entire class, Figure 5.

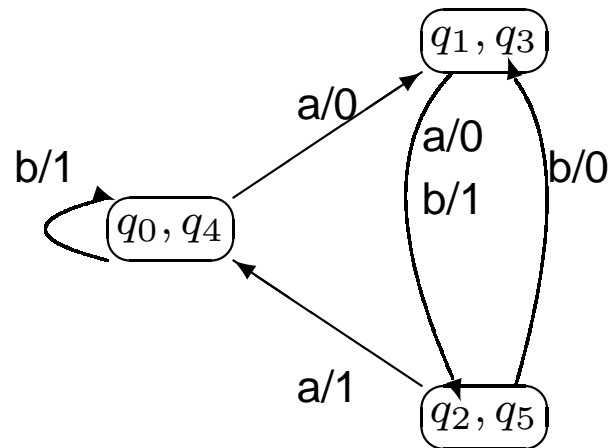


Figure 5: Reduced DGSM of M in Figure 4



Application to DFA

- We can view a DFA as a DGSM that has a binary output alphabet:
 1. entering a final state corresponds to "1" output and
 2. entering a non-final state corresponds to "0" output.
- Then the DGSM reduction process can be applied to DFA by the following procedure:



DFA reduction algorithm

1. Discard all states that are unreachable from the start state of the DFA. They affect the transducer's capability but not the recognizer.
2. Add outputs from $\{0, 1\}$ to the DFA defining ρ by:

$$\rho(q, w) = \begin{cases} 0, & \text{if } \delta(q, w) \text{ is non-final;} \\ 1, & \text{if } \delta(q, w) \text{ is final.} \end{cases}$$

3. Perform the reduction process on DGSM produced by step (2);
Do not place final and non-final states in the same Π_1 class because ϵ does not distinguish states in a DGSM but it does distinguish final/non-final states in a DFA.
4. Ignore the output associated with the reduced DGSM obtained in step (3); this is the minimal DFA.



Final observations

- The unique reduced machine property is valid only with DFA. For NFA this property is lost because ϵ input cannot distinguish states;
- There is little to differentiate the recognizer and transducer orientations, except that recognizers effectively employ a binary output alphabet;
- The recognizer view ignores the intermediate steps in a computation and focus on a single final response;
- The disregarded details of the intermediate responses can still be extracted, if one so wishes.

