E-8

8.2, 8.3
8.5, 8.6

* key points from lecture
  - discrete-event systems
    - to model control computations
  - model and analyze
    - deterministic / nondeterministic automata
    - deadlock, livelock, blocking
    - safety

* extra notes
  - definition $A = (Q, E, \delta, q_0, Q_m)$
    - E^* denotes all finite strings of elements of E, together with ε
      $E = \{a, d\}$, $E^* = \{\varepsilon, a, d, aa, ad, dd, aaa \ldots\}$
    - $q = \delta(q_0, s)$, $s \in E^*$. state reached after executing the string s
    - language L, a set of finite strings. $L \subseteq E^*$
      - concatenation of L_1, L_2
    - languge generated by A
      $$L(A) = \{s \in E^* : \delta(q_0, s) \text{ is defined}\}$$
    - languge marked by A
      $$L_m(A) = \{s \in E^* : \delta(q_0, s) \in Q_m\}$$
  - equivalent Automata
    A_1 equivalent to A_2 if $L(A_1) = L(A_2)$
    $L_m(A_1) = L_m(A_2)$

- deadlock:

  unmarked states can be reached and, no further event can be executed

- livelock

  a subset of unmarked states can be reached, but no transition is going out from the subset.

- prefix-closure of $\mathcal{L}m(A)$

$$\overline{\mathcal{L}m(A)} = \{s \in E^* : \exists s' \in E^*, ss' \in \mathcal{L}m(A)\}$$

$$\mathcal{L}m(A) \subseteq \overline{\mathcal{L}m(A)} \subseteq \mathcal{L}(A)$$

- blocking

  automaton A is blocking if $\overline{\mathcal{L}m(A)}$ is a strict subset of $\mathcal{L}(A)$.

  nonblocking if $\overline{\mathcal{L}m(A)} = \mathcal{L}(A)$

- nondeterministic

$$\delta : Q \times E \cup \{\varepsilon\} \rightarrow 2^Q$$

  q_0 is a set

- automaton with smallest number of states that mark a languge

  minimal realization

- equivalent states

  $\mathcal{L}m(A(x)) = \mathcal{L}m(A(y))$, $A(x)$ the same automaton as A with initial state x

- safety

  make sure reachable states aviod bad states

Ex. 8.2

extra notes

(a) model the vending machine using a discrete-event system

$$A = (Q, E, \delta, q_0, Q_m)$$ P6 of 28

analyze:

Soda: \$0.45

accepts: \$0.10 (dime) \$0.25 (quarter)

Soda dispensed only if \$0.45 is inserted

otherwise: no soda, no return change

States: $Q = \{0, 10, 25, 35, 45, 20, 30, 40, \square\}$ overflow

event: $E = \{D, Q\}$

initial state: $q_0 = 0$, marked state / final state $q_m = 45$

transition map: $\delta(0, D) = 10$, $\delta(0, Q) = 25$

$\delta(10, D) = 20$, $\delta(10, Q) = 35$, $\delta(25, D) = 35$

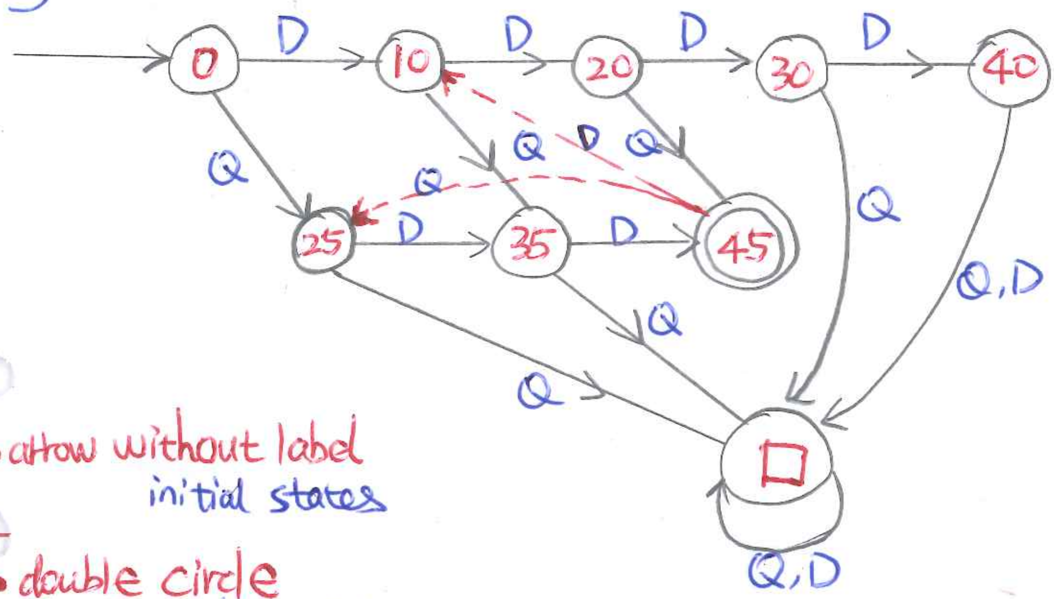
$\delta(25, Q) = \square$, $\delta(20, D) = 30$, $\delta(30, D) = 40$

$\delta(20, Q) = 45$, $\delta(35, D) = 45$, $\delta(35, Q) = \square$

$\delta(30, Q) = \square$, $\delta(40, D) = \square$, $\delta(40, Q) = \square$

$\delta(\square, Q) = \square$, $\delta(\square, D) = \square$ ⇐ typo. attention in the solution

diagram



$\delta(45, D) = 10$

$\delta(45, Q) = 25$

(I think alright)

- arrow without label
  initial states
- double circle
  marked states

(b) is it possible that the machine does not dispense soda?

sure. (might be should be asking ... does dispense ... :))

Livelock at the "overflow" state □   P14 of L8

formally, once state □ is reached, no transition is going out from (leaving) this state.

Ex 8.3

extra notes

(a) discribe formally the DES

$$A = (Q, E, \delta, q_0, Q_m)$$

where: states $Q = \{q_1, q_2\}$
events $E = \{0, 1\}$

transition δ: $\delta(q_1, 0) = q_2$, $\delta(q_2, 1) = q_1$
initial state q_0, marked state q_2

(b) marked language $L_m(A)$

P11 of L8

def: $L_m(A) = \{s \in E^* : \delta(q_0, s) \in Q_m\}$

(the set of finite strings ~~that~~ of events that drive the system to one of the marked states)

generated language $L(A)$

def: $L(A) = \{s \in E^* : \delta(q_0, s)$ is defined$\}$

(the set of finite string of events that drives the system starting from ~~one of th~~ the initial state)

In our case

$\mathcal{L}(A) = \{ \underbrace{0(10)^*}_{\text{ends in } q_2}, \underbrace{(01)^*}_{\text{ends in } q_1} \}$

$*$: asterisk
(finite repetition)

$\mathcal{L}_m(A) = \{ 0(10)^* \}$

Ex. 8.5

---

extra notes

typo in the diagram, $\delta(h, 0) = g$
(not bi-directional)

- given a discrete-event system
  determine the minimum state automaton

- equivalent states

States 'x' and 'y' are equivalent if $L_m(A(x)) = L_m(A(y))$

where $A(x)$ is the same $\boxed{\text{automaton as A with initial state } x}$

- two states are equivalent if they have the same behavior in terms of marked language. (in other words, given an input string L, $\delta(x,L)$ is a marked state if and only if $\delta(y,L)$ is a marked state)

- two equivalent states can be replaced by a single aggregate state

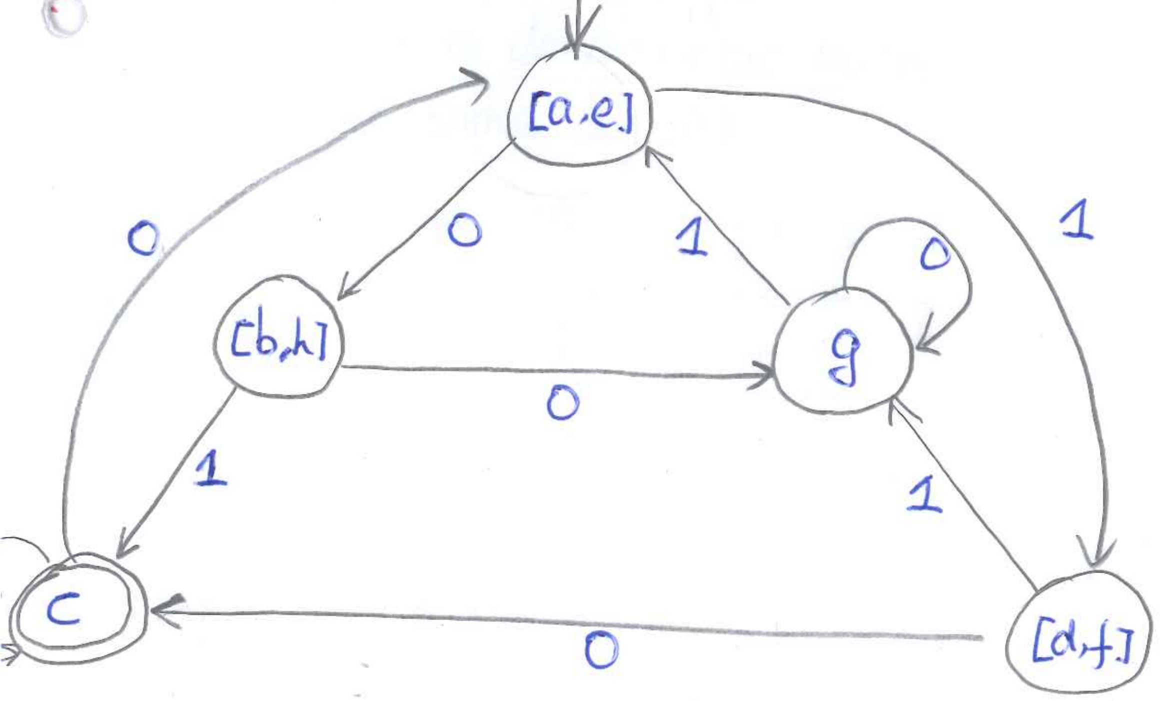
we bulid a table to indicate if its possible ~~to~~ for each pair of states to be equivalent

	a	b	c	d	e	f	g	h
h								
g								1
f							0	0
e						0	01	1
d					0	V	0	0
c				1	1	1	1	001
b			010	1	1	1	1	V
a		1	1	0	V	0	10	1

it turns out

$a \equiv e,\ b \equiv h,\ d \equiv f$

the minimaln automaton is



Ex 8.6

- Construct equivalent DFA from an NFA (not in the lecture)

  Subset Construction

- critical points
  - final states = all those with a member of F
  - transitions

$$\delta_D(\{[q_1, \dots q_k], a) = \underbrace{\bigcup_{i=1}^{k}}_{\text{union over all } q_i} \delta_N(q_i, a)$$

extra notes:

not taught in the lecture

- Construct DFA from NFA
- equivalent language

Given the nondeterministic automaton

$$A = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

where $\delta(q_0, 0) = \{q_0, q_1\}$, $\delta(q_0, 1) = \{q_1\}$

$\delta(q_1, 0) = \delta(q_1, 1) = \{q_0, q_1\}$

Prob: construct a deterministic automaton A', which accepts the same $L_m(A)$.

Idea: build / treat $\{q_0, q_1\}$ as a new states

Namely, build Q' as $\boxed{\text{all subsets of } Q}$

$Q' = \{[q_0], [q_1], [q_0, q_1]\}$ ← names, symbols

initial state $[q_0]$. final state $[q_1]$, $[q_0, q_1]$

events $\{0, 1\}$

transitions

$\delta'([q_0], 0) = [q_0, q_1]$

$\delta'([q_0], 1) = [q_1]$

$\delta'([q_1], 0) = \delta'([q_1], 1) = [q_0, q_1]$

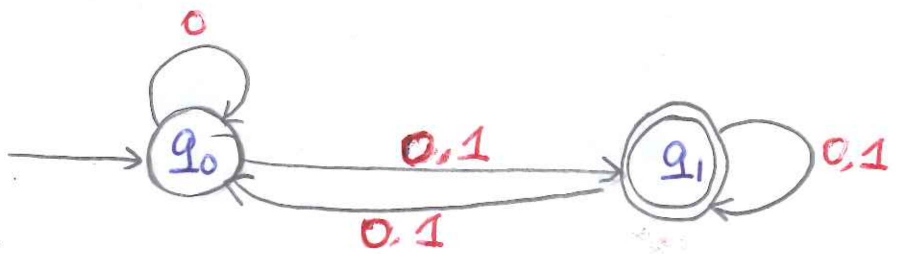
$\delta'([q_0, q_1], 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0, q_1\} = [q_0, q_1]$

$\delta'([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$

$= \{q_1\} \cup \{q_1, q_0\} = \{q_0, q_1\} = [q_0, q_1]$

original automaton A



deterministic automaton A'