Компютърни архитектури
CSCB008

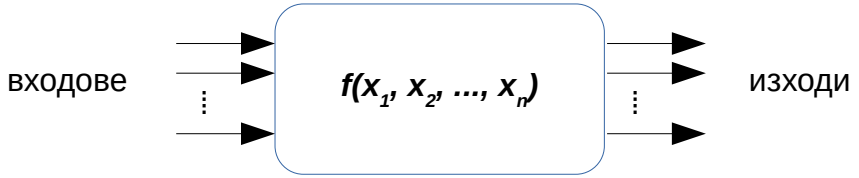
Последователностна логика. Автоматни модели

НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

доц. д-р Ясен Горбунов
2021

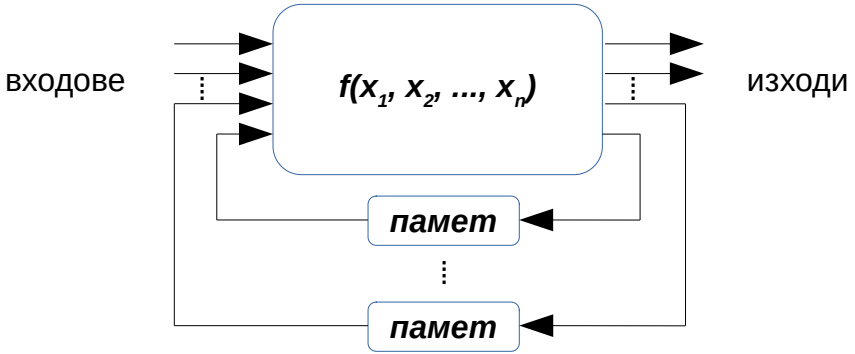
Автоматни модели

комбинационни логически схеми → схеми без обратни връзки – автомати без памет

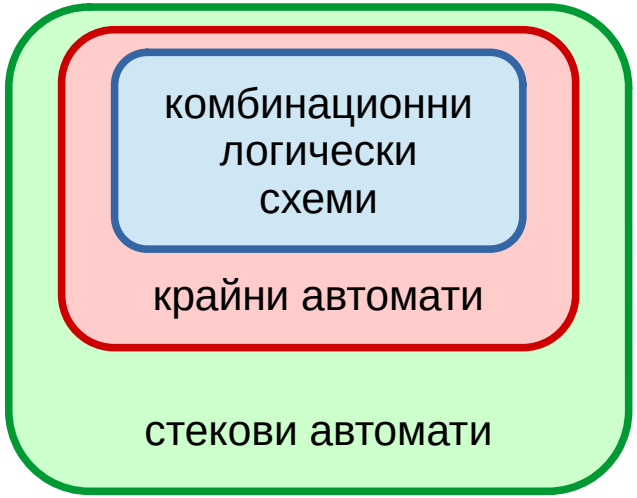


Автоматни модели

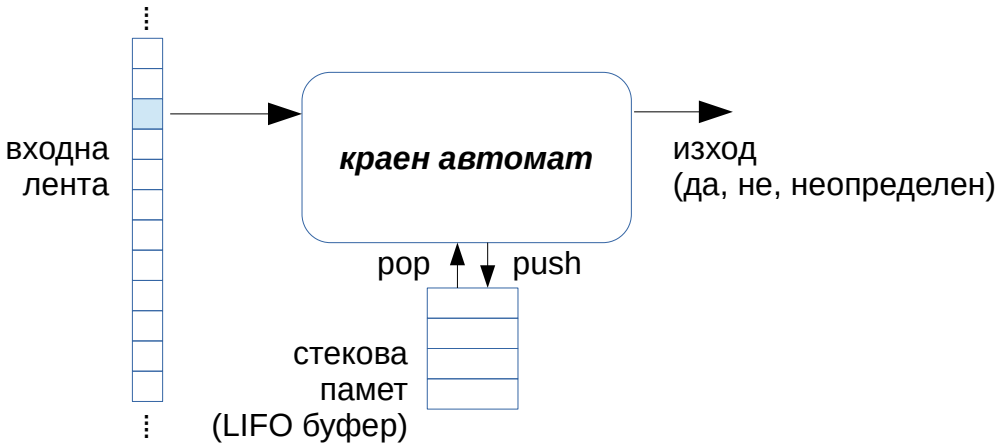
схеми, притежаващи обратни връзки и памет
представляват последователностни схеми
за тях се дефинират „състояния“



Автоматни модели

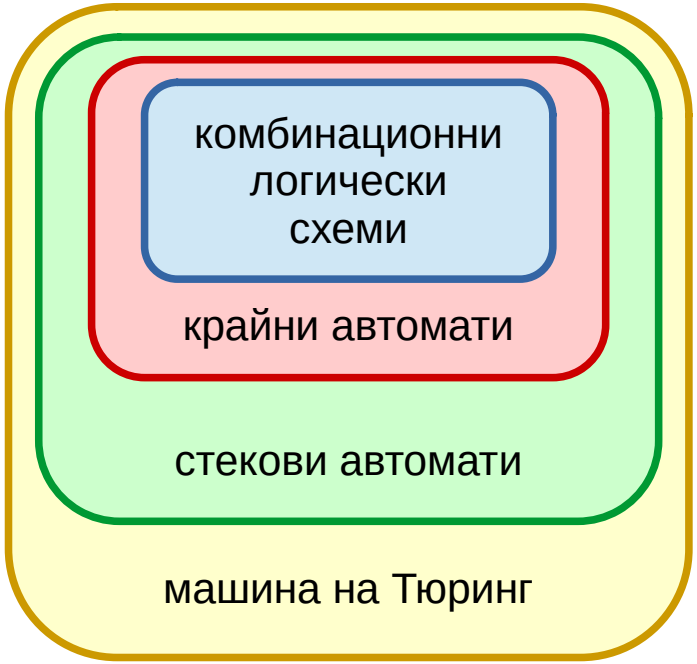


крайни автомати + стекова памет
„разпознаватели“

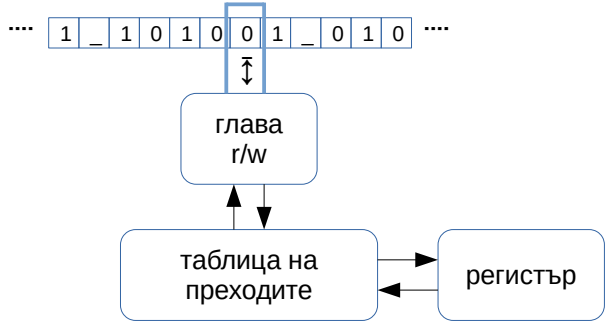


състоянието на автомата се определя от входните данни и от върха на стека, който може да бъде манипулиран като част от операцията

Автоматни модели



абстрактен изчислителен модел
с „безкрайна“ памет



автомати с „безкраен“ брой състояния, които притежават:

1. **Памет** – манипулират символ върху безкрайна поточна лента
2. **Глава** – при всеки такт главата прочита символа от клетката, над която се намира, записва нов символ и се премества
3. **Таблица на преходите** – краен списък от инструкции (програма)
4. **Регистър** – програмен брояч

Автоматни модели

Крайните автомати са важна концепция в проектирането на компютърни програми и цифрови схеми.

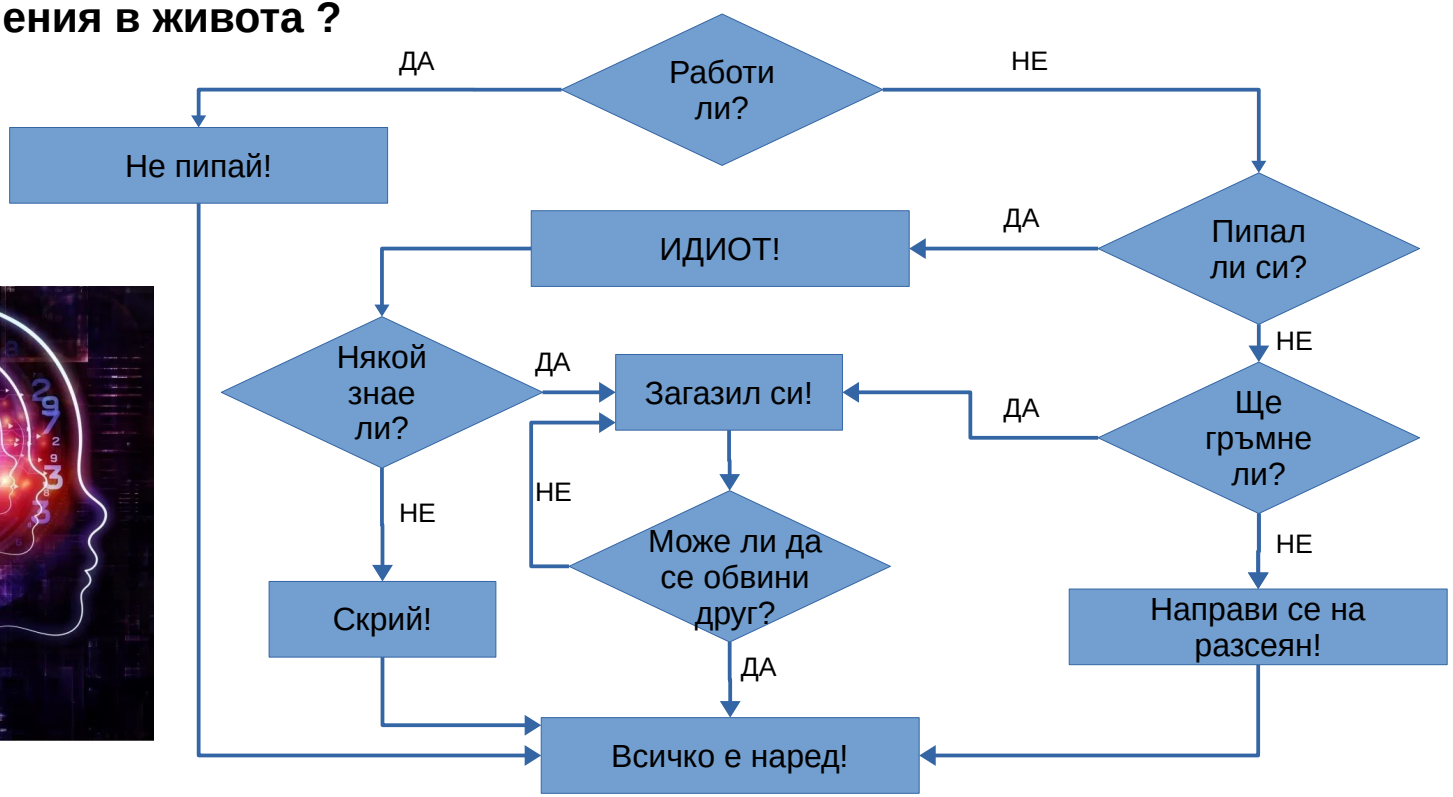
Крайните автомати са много по-прости от машините на Тюринг, защото могат да решават само ограничен кръг от задачи.

Крайните автомати представляват абстрактен поведенчески модел на машина с краен брой състояния и ограничена памет.

Всяка компютърна програма представлява краен автомат.

Приложения на крайните автомати

Как вземаме решения в живота ?



Приложения на крайните автомати

Технически приложения

Асансьор

Автомат за продажба

Машина за кафе

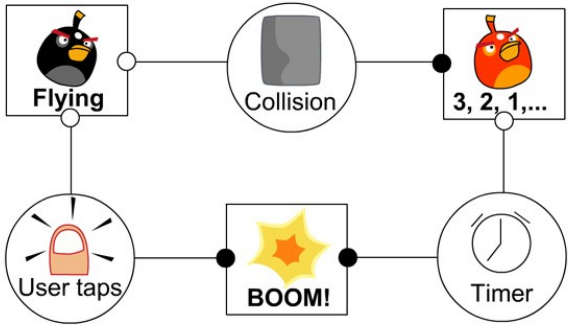
светофар

Кодова брава

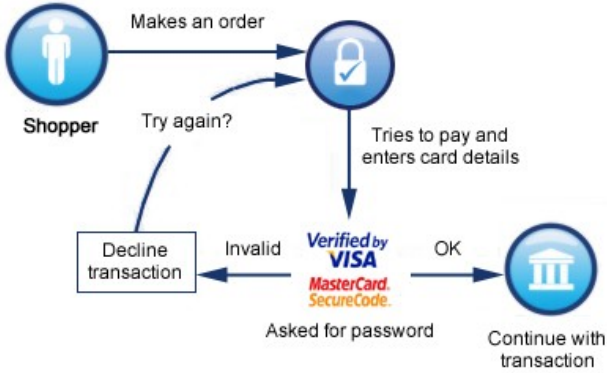
Приложения на крайните автомати

Технически приложения

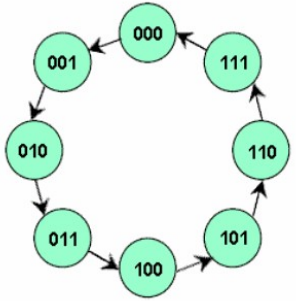
Компютърни игри



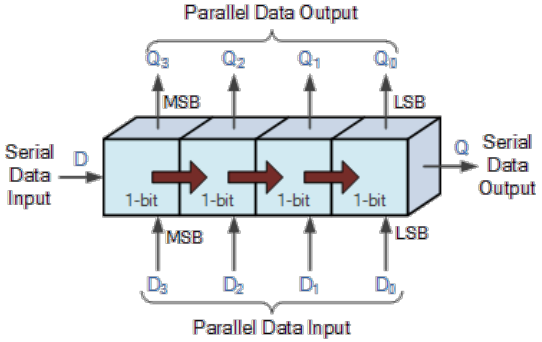
Валидиране на транзакции



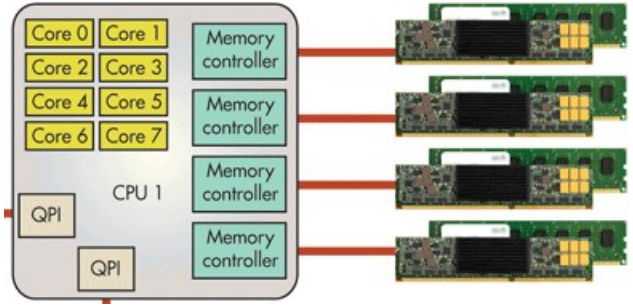
броячи



Премествящи регистри



DRAM контролер



Крайни автомати

Всяко (достатъчно) сложно цифрово устройство може да бъде разделено на две части

управляваща → управляващ автомат (УА)

управляема → операционен автомат (ОА)

Съществуват три подхода за реализация на алгоритми в компютърните архитектури:

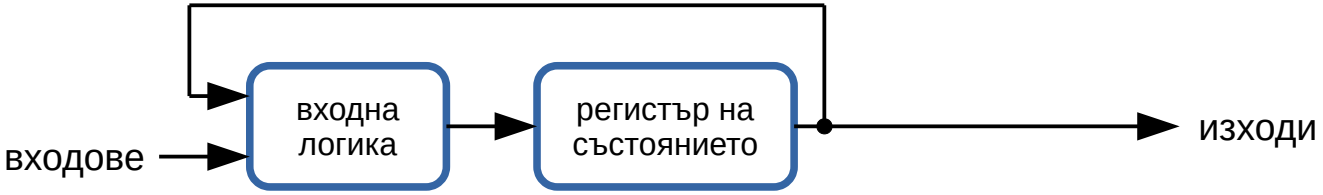
1. **Схемно управление** – алгоритмите представляват задание за проектиране на УА и са фиксирани в схема
   - \+ осигурен е паралелизъм и много високо бързодействие
   - \- всяка промяна изисква изменение в схемата

2. **Микропрограмно управление** – алгоритмите се кодират по определен начин и се записват в ROM / EEPROM
   - \+ всяка промяна се състои в препрограмиране, като структурата на УА не се променя
   - \- липсва паралелизъм и бързодействието е по-ниско

3. **Програмно управление** – алгоритмите са реализирани като подредени инструкции (програми)
   - \+ много висока степен на гъвкавост
   - \- най-ниско бързодействие

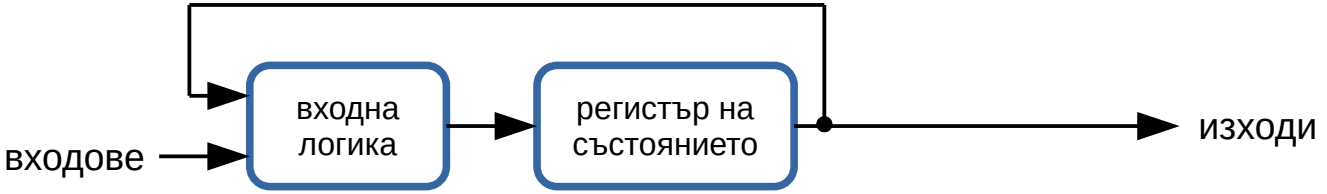
Автомат на Медведев

Автомат на Рабин-Скот

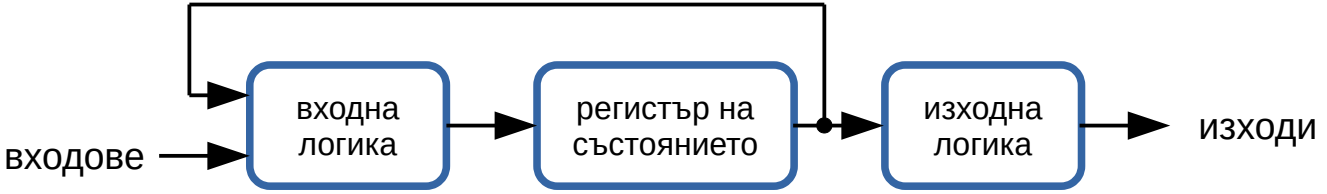


Автомат на Медведев

Автомат на Рабин-Скот

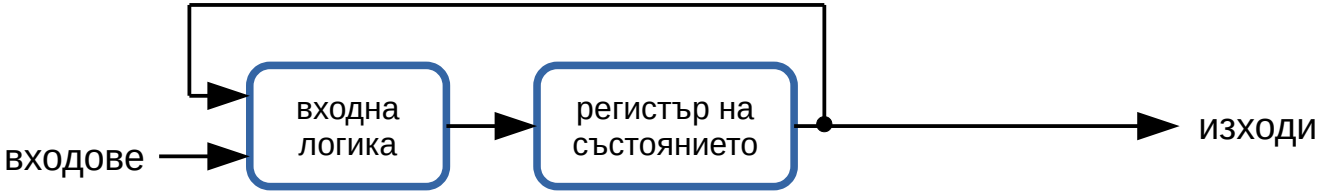


Автомат на Мур

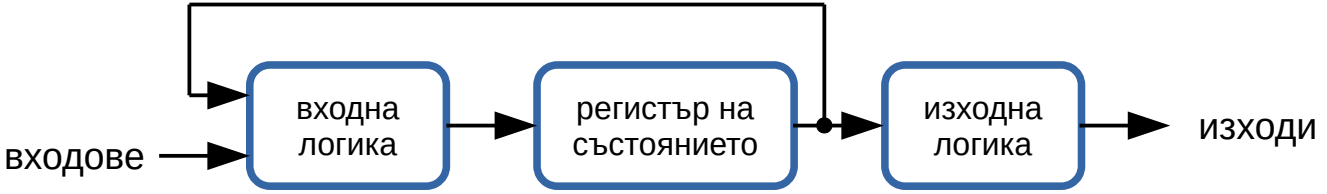


Крайни автомати

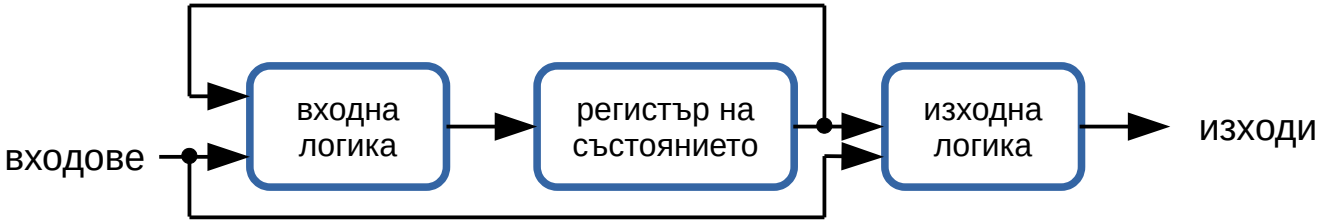
Автомат на Медведев
Автомат на Рабин-Скот



Автомат на Мур

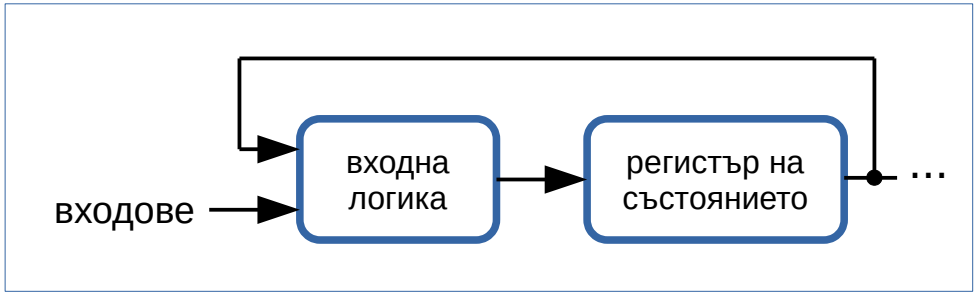


Автомат на Мили



асинхронни автомати

- промяната на поведението на схемата зависи само от момента на промяната на входните сигнали
- Асинхронните автомати са по-бързи, но липсата на синхронизиращи импулси може да доведе до неустойчива работа

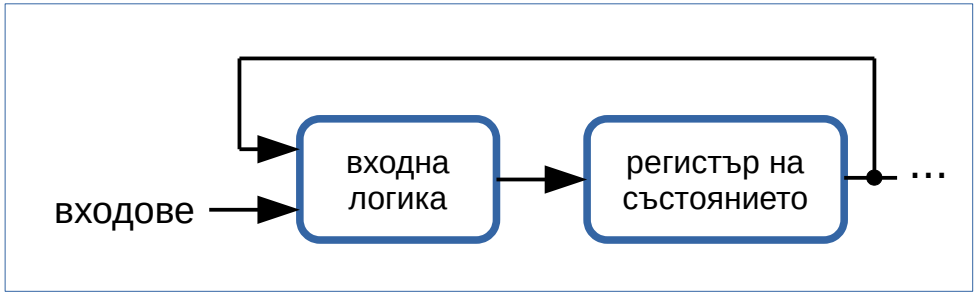


Крайни автомати

асинхронни автомати

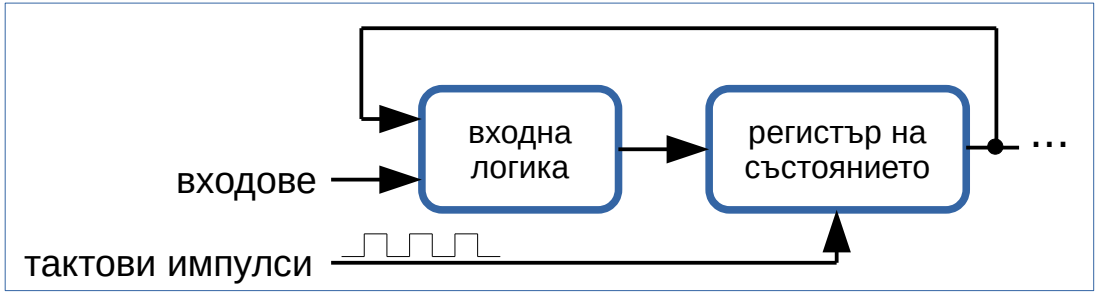
- промяната на поведението на схемата зависи само от момента на промяната на входните сигнали
- Асинхронните автомати са по-бързи,

но липсата на синхронизиращи импулси може да доведе до неустойчива работа



синхронни автомати

- поведението се дефинира чрез входните сигнали във всеки дискретен момент от времето



Крайни автомати

детерминирани крайни автомати (DFA)

- всеки техен преход е детерминиран от състоянието и буквата на входната азбука
- прочитането на входен символ е задължително за всеки преход

всяко състояние има най-много един преход при даден вход

недетерминирани крайни автомати (NFA)

- не се подчиняват на ограниченията на DFA
- в частност, всеки DFA е също и NFA

възможни са няколко различни прехода за един и същи вход

Регистър на състоянието

В последователностните схеми се въвеждат запомнящи елементи – мултивибратори.

бистабилни мултивибратори

- притежават две устойчиви изходни състояния – тригери (trigger-спусък)
- изходното им състояние остава непроменено до промяна на входа
  - тригери, управлявани по ниво – latch
  - тригери, управлявани по фронт – flip-flop

D-тригер (Delay) – управляван по преден фронт

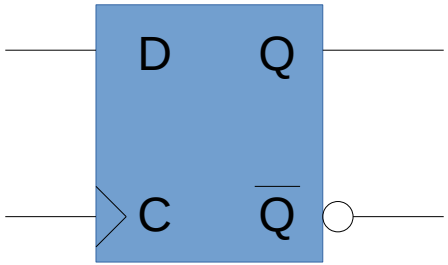


Таблица на преходите

clk	D	Q
↓	0	без промяна
↓	1	без промяна
↑	0	0
↑	1	1

Задаване на реалния автомат

1. краен брой входни сигнали $X=\{x_0, x_1, x_2, \ldots, x_n\}$ – входна азбука
2. краен брой изходни сигнали $Z=\{z_0, z_1, z_2, \ldots, z_m\}$ – изходна азбука
3. краен брой вътрешни променливи $A=\{a_0, a_1, a_2, \ldots, a_w\}$ – вътрешни състояния
4. начално състояние $a_0 \in A$
5. функция на преходите $A_{new}=\lambda(A_{old}, X)$ – A_{new} – ново състояние, A_{old} – старо състояние
6. функция на изходите $Z=\delta_1(A)$ или $Z=\delta_2(A, X)$

Автоматен модел на Мур

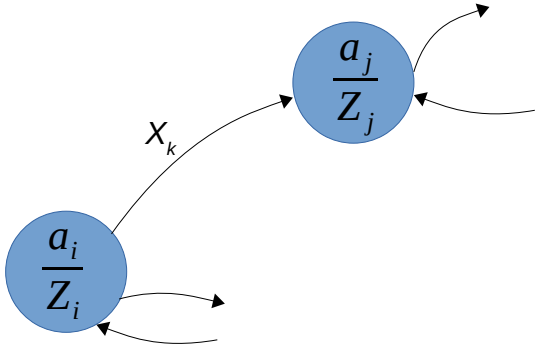
функция на преходите – старото състояние се заменя с ново (преход)

$$A_{new} = \lambda(A_{old}, X)$$

издава изходна реакция само в дадено състояние

$$Z = \delta_1(A)$$

представяне чрез насочен граф



Автоматен модел на Мили

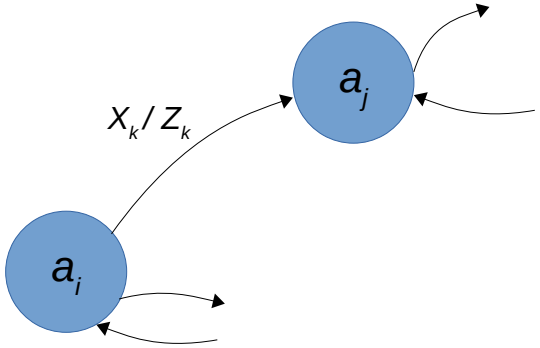
функция на преходите – старото състояние се заменя с ново (преход)

$$A_{new} = \lambda(A_{old}, X)$$

издава изходна реакция по време на преход

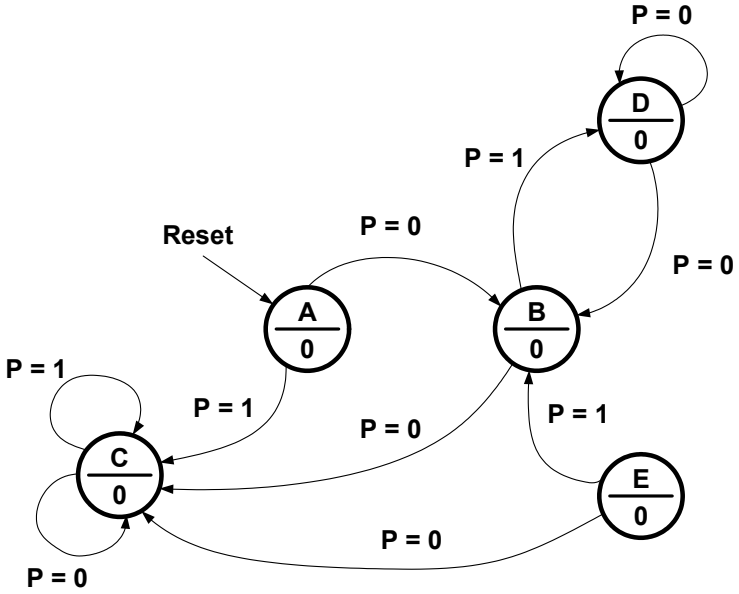
$$Z = \delta_2(A, X)$$

представяне чрез насочен граф



Грешки в диаграмите на състоянията

- всяко състояние, различно от началното, за което няма преход в него, трябва да бъде отстранено
- система с n входа, трябва да има точно 2^n прехода, водещи навън от всяко състояние
- липсващите преходи трябва да се добавят, а дублираните да се премахнат



Грешки в диаграмите на състоянията

- всяко състояние, различно от началното, за което няма преход в него, трябва да бъде отстранено
- система с n входа, трябва да има точно 2n прехода, водещи навън от всяко състояние
- липсващите преходи трябва да се добавят, а дублираните да се премахнат