Masini cu stari finite

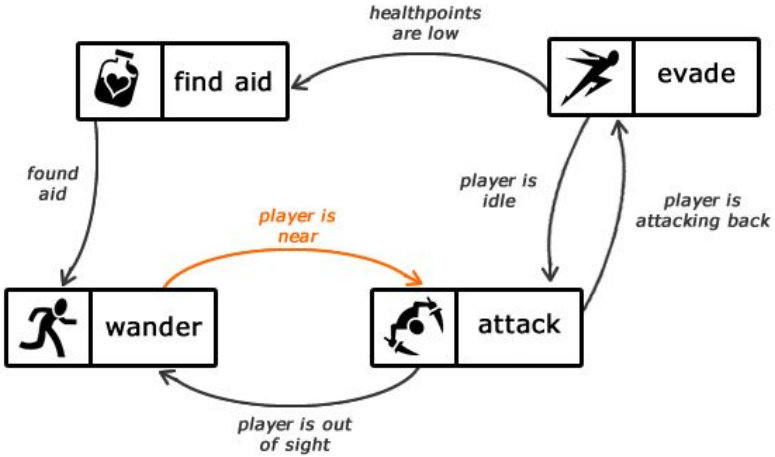
Alexandru ADEACONITEI
Claudiu OROSANU
Cosmin REZMERITA

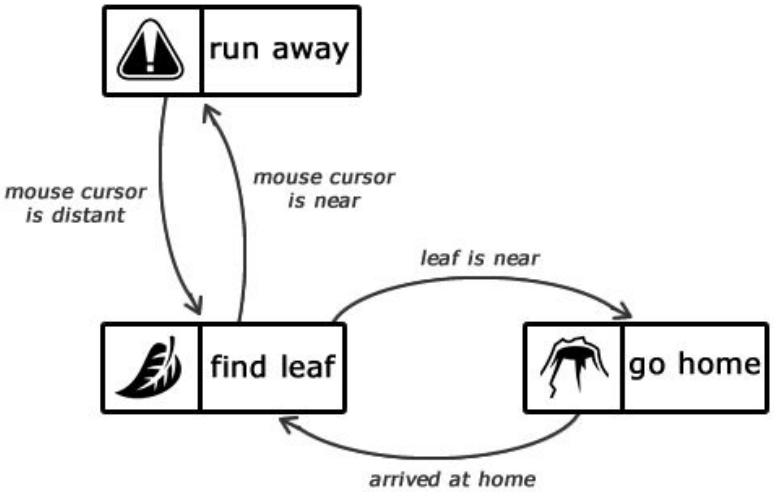
Definitie

- Model computational care poate fi implementat hardware sau software si poate fi folosit pentru a simula **logica secventiala**.
- Masinile cu stari finite sunt folosite pentru a modela probleme in multe domenii, cum ar fi matematica, inteligenta artificiala, jocuri si lingvistica.
- Exista doua tipuri de masini cu stari finite: **deterministe** (DFA - deterministic finite automaton) si **non-deterministe** (NDFA - nondeterministic finite automaton)
- Are o putere de calcul mai mica fata de alte modele computationale (ex. Masina Turing)

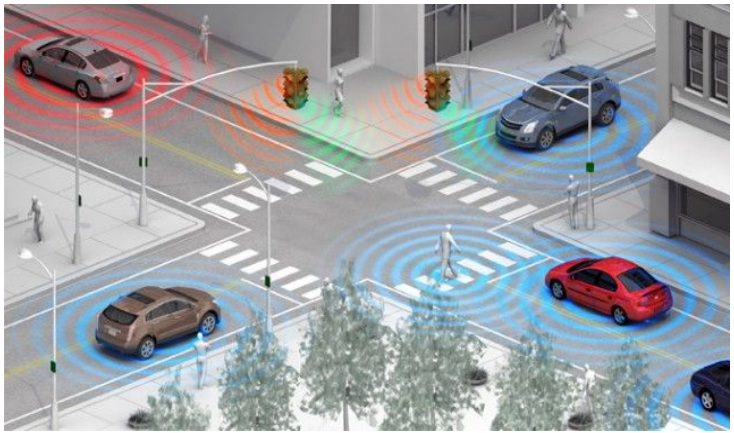
Domenii de aplicare

- Masinile cu stari finite descriu, in definitiv, **un sir de evenimente** (un fir de executie); asadar, acestea pot fi folosite pentru a descrie evenimentele dintr-un joc video ce foloseste concepte de inteligenta artificiala. **“Creierul”** inamicului din joc, spre exemplu, poate fi descris folosind modelul unei masini cu stari finite: fiecare stare va reprezenta o actiune (ex. Ataca, Evadeaza).
- Un fir de executie poate fi reprezentat ca un **graf** in care nodurile reprezinta **starile** masinii, iar muchiile sunt **tranzitiile**.


healthpoints
are low
find aid
evade
found
aid
player is
idle
player is
attacking back
player is
near
wander
attack
player is out
of sight


run away
mouse cursor
is distant
mouse cursor
is near
leaf is near
find leaf
go home
arrived at home

- Alte exemple de folosire a masinilor cu stari finite:
  - Aparatele de vanzare - ofera produsul comandat de catre cumparator numai atunci cand este depozitata suma potrivita de bani
  - Ascensoare - secventa lor de opriri este determinata de sirul de etaje la care vor sa ajunga utilizatorii lor
  - Semafoarele inteligente - isi schimba culorile atunci cand sunt masini care asteapta
  - Seifurile - se deschid numai atunci cand este introdusa combinatia potrivita de cifre

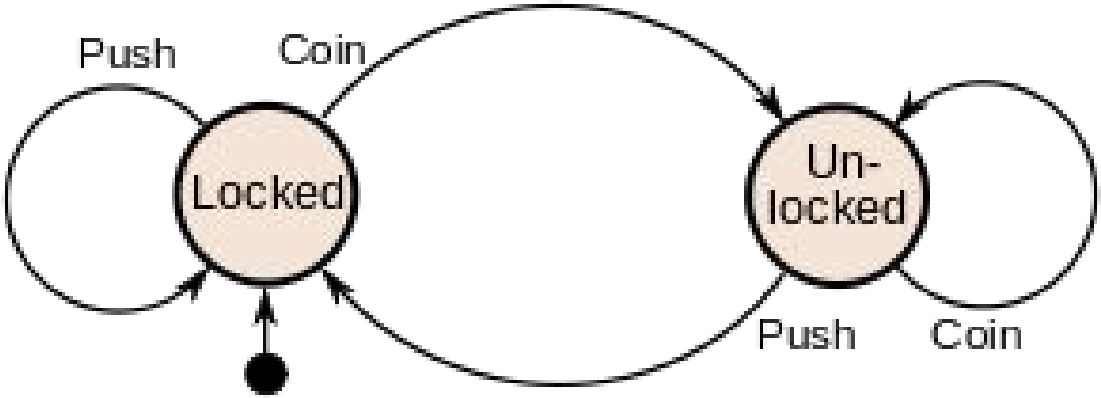


Caracteristicile unei masini cu stari finite

O masina cu stari finite M este descrisa de urmatoarele caracteristici:

- Q - set finit de stari
- Σ - alfabetul; o multime finita de simboluri
- δ - functia de tranzitie
- q_0 - starea initiala
- F - set de stari finale acceptate

- Trecerea dintr-o stare in alta reprezinta un raspuns la anumiti stimuli.
- Memoria unei masini de stare rezida din numarul limitat de stari pe care le are
- Urmatoarea stare depinde de vechea stare si de input



Current State	Input	Next State	Output
Locked	coin	Unlocked	Unlocks the turnstile so that the customer can push through.
	push	Locked	None
Unlocked	coin	Unlocked	None
	push	Locked	When the customer has pushed through, locks the turnstile.

$Q = \{s_1, s_2\}$

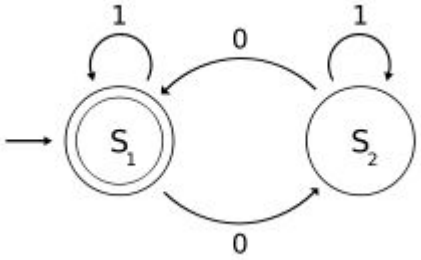
$\Sigma = \{0, 1\}$

The following table describes δ:

current state	input symbol	new state
s_1	1	s_1
s_1	0	s_2
s_2	1	s_2
s_2	0	s_1

$q_0 = s_1$

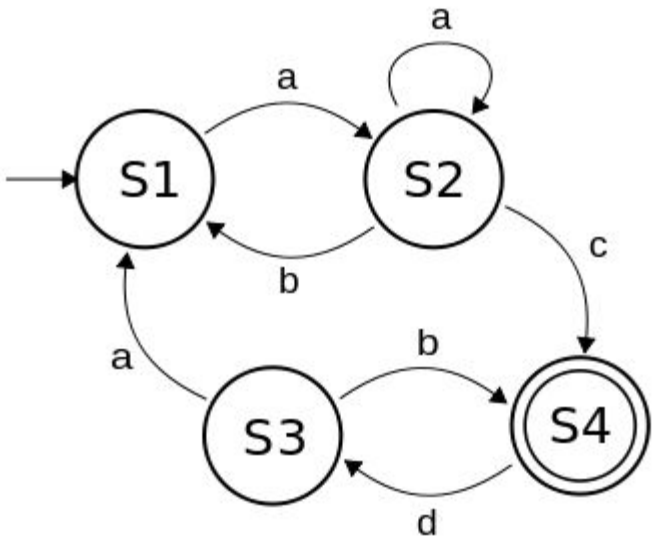
$F = s_1$



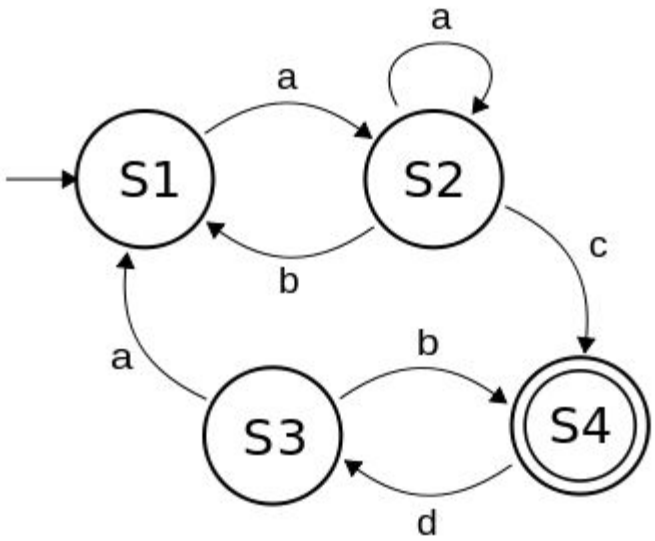
Acest **DFA** recunoaste orice string care are un numar par de 0 si oricate 1.

Asta inseamna ca poti pune, ca input, orice sir de caractere care contine un numar par de 0, iar masina cu stari finite il va accepta.

Daca input-ul este un sir de caractere cu un numar impar de 0, atunci se va ajunge in starea s0, care nu e o stare acceptata.



- abac
- abacdaac
- aaaaac
- aaaacd



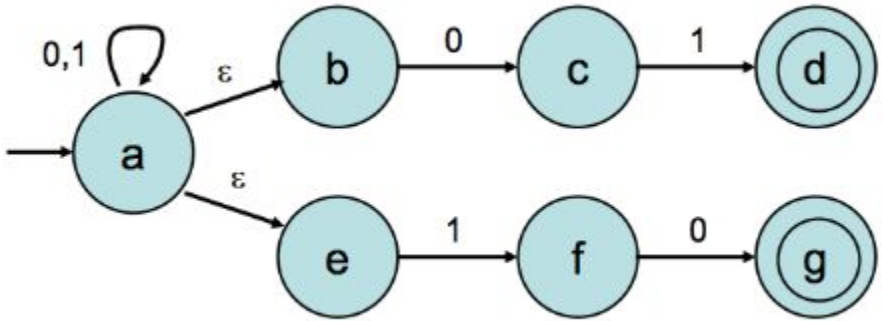
abac

abacdaac

aaaaac

aaaacd

NDFA



Siruri de caractere recunoscute:

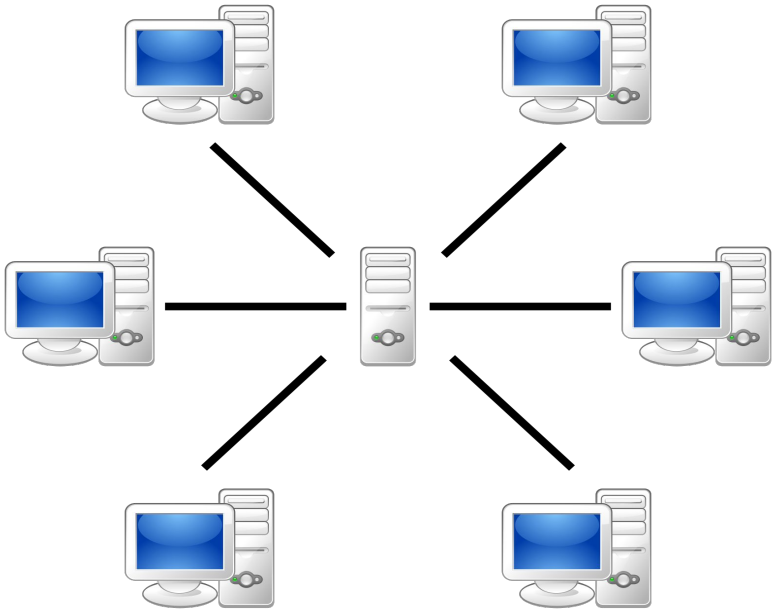
- 00000000010
- 10
- 01
- 1111101

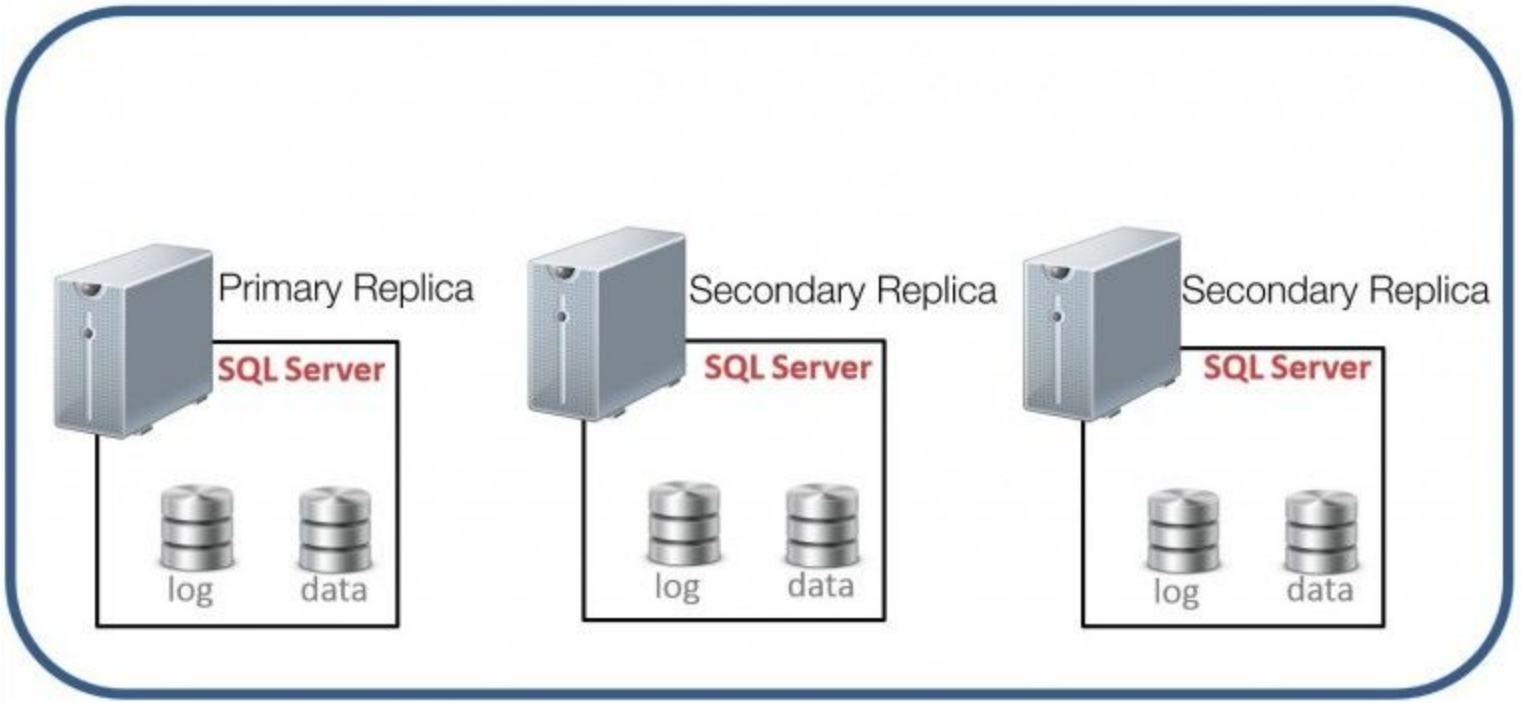
Practic, orice sir care se termina in 01 sau 10.

Replicarea masinilor de stare

- Metoda generala pentru implementarea serviciilor **fault-tolerant** prin replicarea serverelor si coordonarea interactiunii clientilor cu acestea
- Sistemele distribuite sunt adesea structurate in jurul tuplului client - serviciu; un serviciu reprezinta un set de operatii intreprinse de catre servere drept raspuns la cererea clientului.
- Desi este simplu sa fie folosit **un singur server** pentru a consuma servicii, in acest mod nu vom obtine o toleranta la esec mai mare decat a procesorului de pe acel server. Solutia pentru a obtine o toleranta la esec cat mai mare este o **retea de servere** care pot cadea independent.
- De obicei, **replicile** unui server sunt executate pe **procesoare separate** din sistemul nostru distribuit.
- **Izolarea** fizica si electrica a procesoarelor dintr-un sistem distribuit ne da siguranta ca esecurile serverelor sunt independente unul fata de celelalt.

Client-server





Windows Server Failover Cluster

Toleranta la esec

Pentru a asigura toleranta la esec, o masina cu stari finite trebuie sa fie **determinista**, adica pentru **acelasi set de inputuri**, multiple copii ale aceleiasi masini vor produce **acelasi set de outputuri**.

Aceasta caracteristica este extrem de importanta, deoarece, pe baza ei, ne putem da seama ca o replica a masinii produce rezultate gresite, comparand **starea** sau **output-ul** acelei replici cu restul replicilor.

Pentru a ne da seama ca o replica produce rezultate eronate, avem nevoie de cel putin **trei** replici ale masinii cu stari finite. Dezavantajul acestui sistem cu trei replici este ca suporta doar un singur esec.

In general, un sistem care suporta **F** esecuri, are nevoie de **2F+1** replici ale masinii. Copiile suplimentare sunt necesare pentru a afla care sunt replicile cu probleme.

Toleranta la esec

Tipuri de situatii in care o componenta este considerata ***faulty*** (comportamentul sau nu mai respecta specificatiile initiale):

- Esecuri de tip **bizantin** - componenta expune un comportament malitios si arbitrar si poate interactiona cu alte componente cu un comportament similar. In acest caz cel putin 2F + 1 replici sunt necesare pentru a asigura integritatea sistemului.
- Esecuri de tip **fail-stop** - ca raspuns la un astfel de tip de esec componenta isi schimba starea astfel incat celelalte componente sa detecteze eroarea si apoi se opreste. Pentru asigurarea functionarii obisnuite sunt necesare F + 1 copii, iar outputul ansamblului poate fi outputul oricaruia dintre membri.

Va multumim!