State Machines

Peter Levinsky IT, Roskilde

17.09.2024

Where to use State Machines

State machines are a method of modelling systems whose output depends on the entire history of their inputs e.g.:

1. Synthetically (Robots)

2. Analytically (describe the behaviour)
   e.g. Text analysing, Network protocols, diff. network attack

3. Predictively (describe the way the environment works)
   e.g. Driverless car

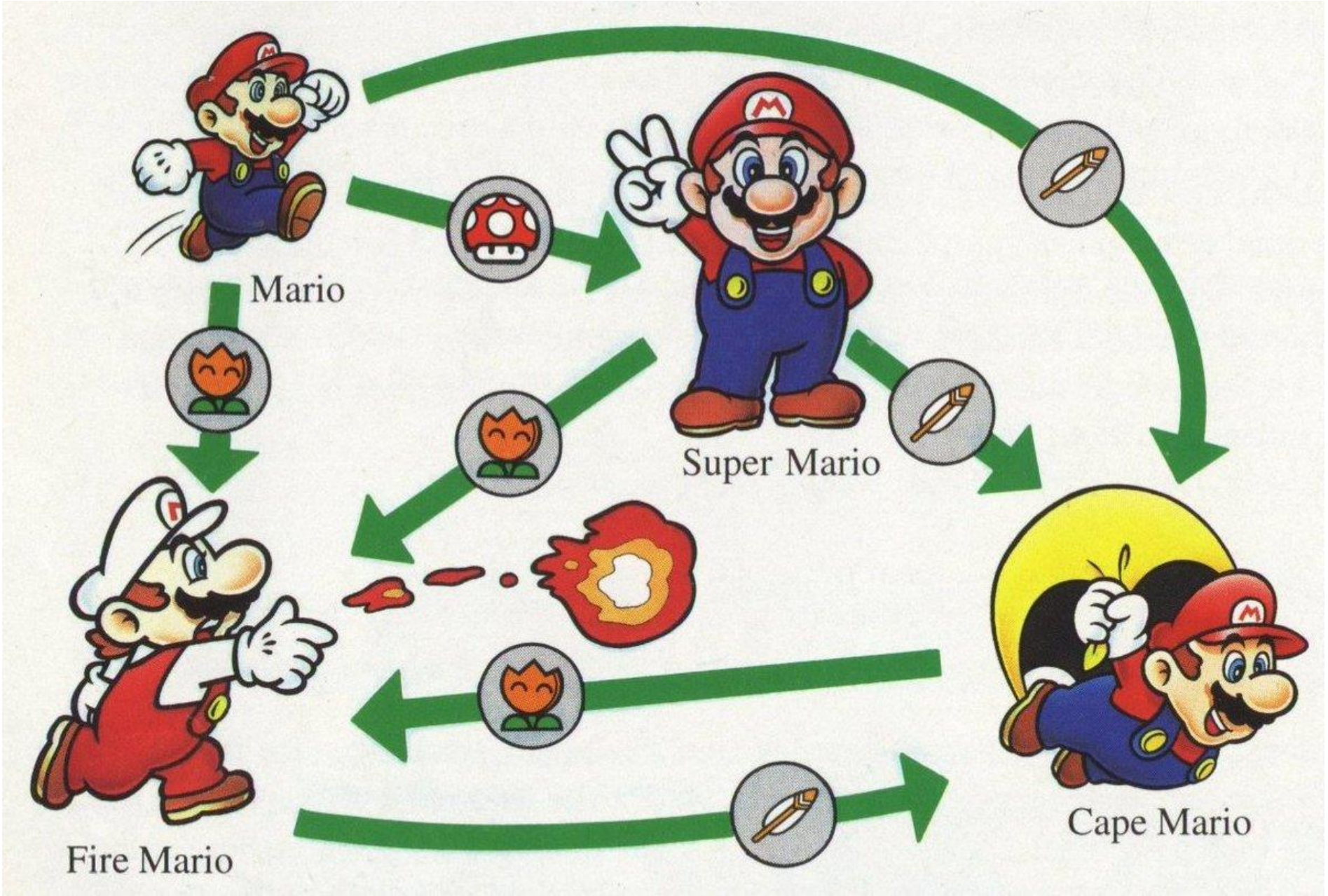
What is State Machines

From the theory of Finite State Machines – Automata Theory

In practice two forms

1. State Design Patterns
2. State-Event Tables
3. SDL State Machines (ITU i.e. Telephone Companies)

Fun example



Finite State Machines

Example

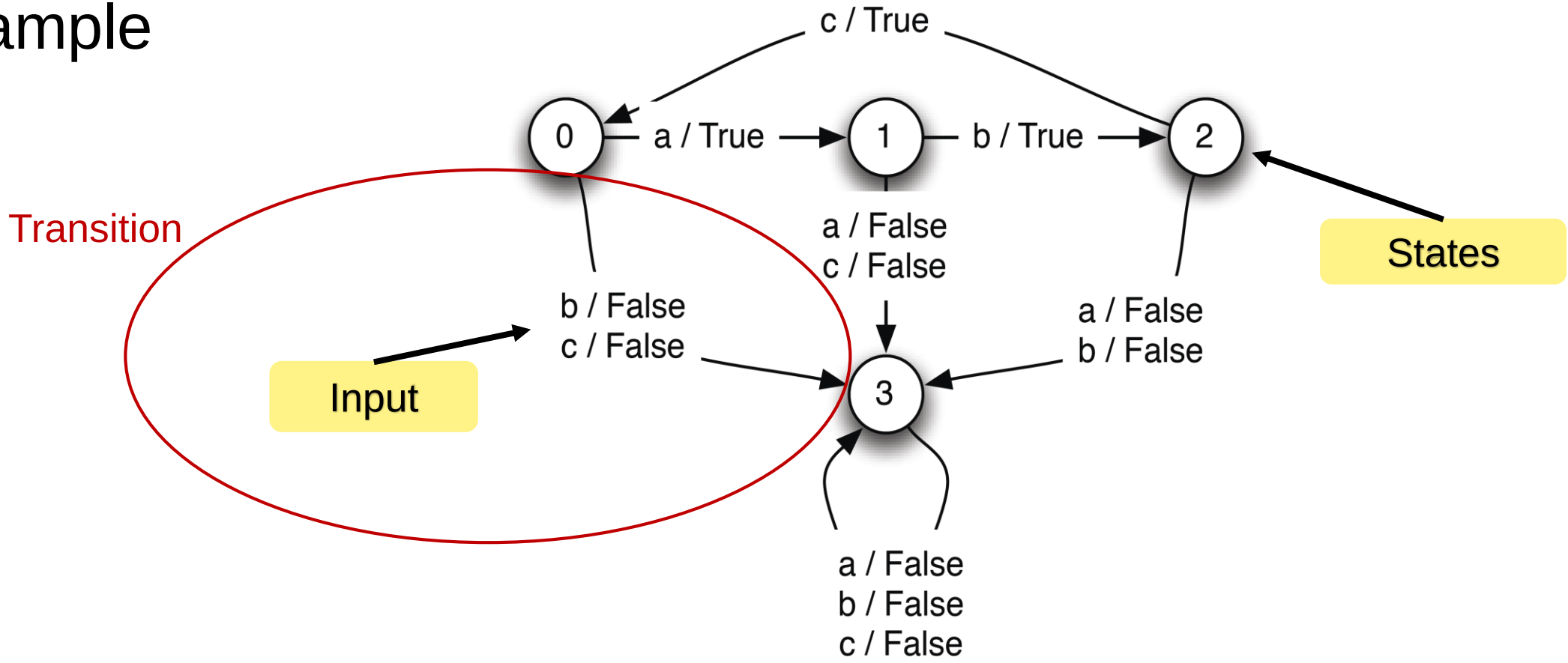


Figure 4.1 State transition diagram for language acceptor.

Theory Finite State Machines

- a set of states, S,
- a set of inputs, I, also called the input vocabulary,
- a set of outputs, O, also called the output vocabulary,
- an initial state, s0, which is the state at time 0.

- **a next-state function**, n(it, st) → st+1, that maps the input at time t and the state at time t to the state at time t + 1,

- **an output function**, o(it, st) → ot, that maps the input at time t and the state at time t to the output at time t

Example State Machines

Checking:

a, b, c, a, b, c, a, b, c

$$S = \{0, 1, 2, 3\}$$

$$I = \{a, b, c\}$$

$$O = \{true, false\}$$

$$n(s, i) = \begin{cases} 1 & \text{if } s = 0, i = a \\ 2 & \text{if } s = 1, i = b \\ 0 & \text{if } s = 2, i = c \\ 3 & \text{otherwise} \end{cases}$$

$$o(s, i) = \begin{cases} false & \text{if } n(s, i) = 3 \\ true & \text{otherwise} \end{cases}$$

$$s_0 = 0$$

Example State Machines #2

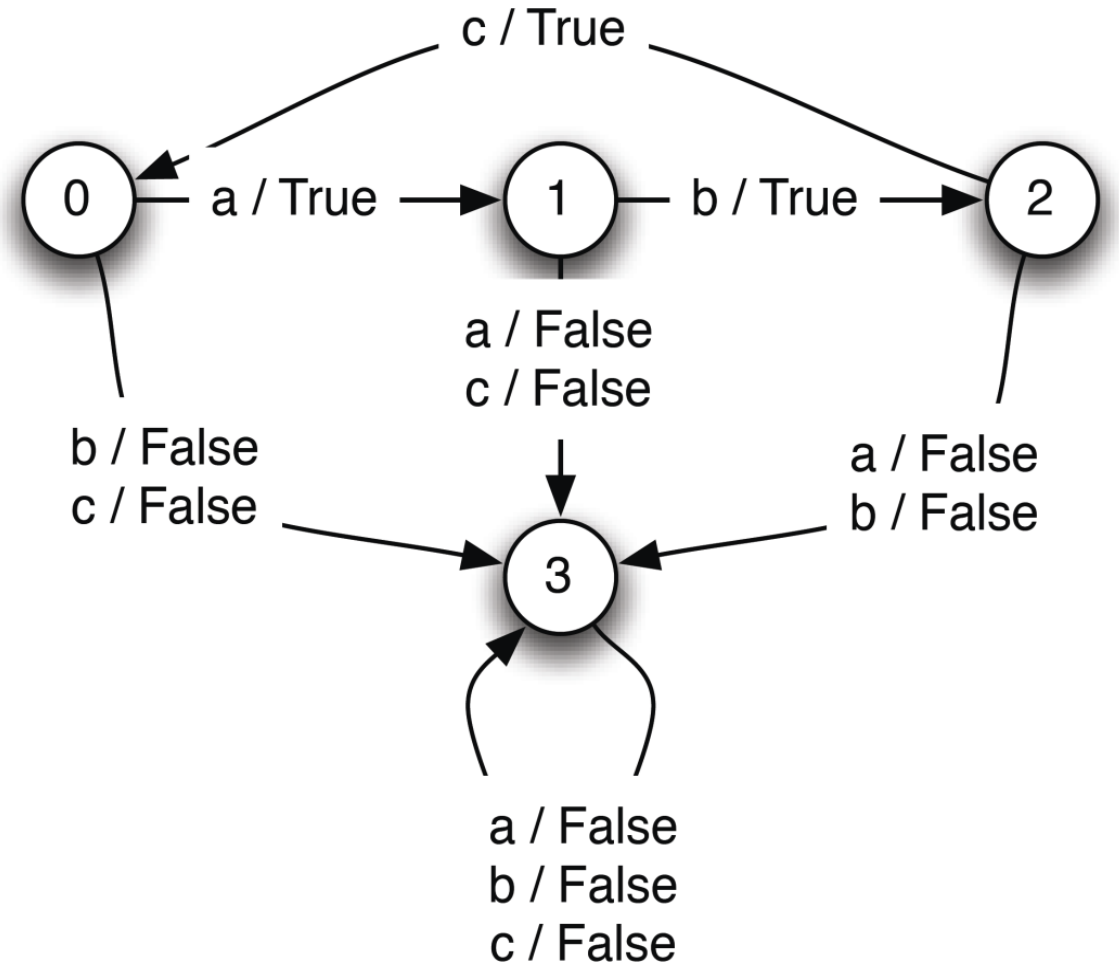
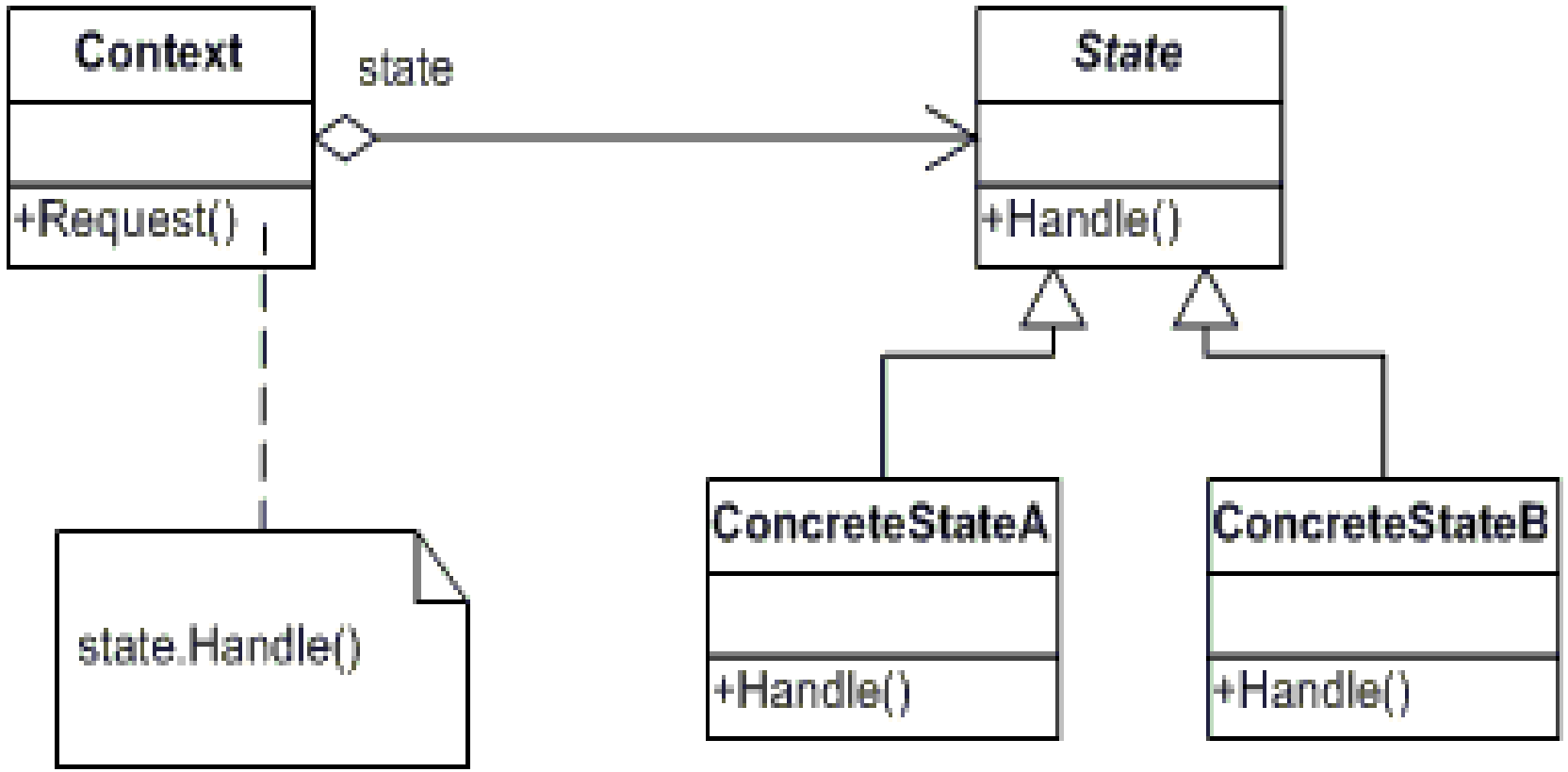


Figure 4.1 State transition diagram for language acceptor.

Exercise

Snake -- Assignment 1

State Machines – State Design Pattern



State Machines – State Design Pattern

Concrete Classes State0, State1, State2, and possible State3

Methods:

IState NextStateFunction(T input)
TOutput OutputFunction(T input)

In context:

IState currentState = new State0() // initial state
…
nextOutput = currentState.OutputFunction(someinput)
currentState = currentState.NextStateFunction(someInput)

State Machines – State Event Tables

(Table driven State machines)

Base on

1) Current state

2) Input

e.g. Current State B
+ Input Y =>
new Current State C

State-transition table

Current state / Input	State A	State B	State C
Input X	...	...	...
Input Y	...	State C	...
Input Z	...	...	...

Your turn