Methods for the specification and verification of business processes
MPB (6 cfu, 295AA)
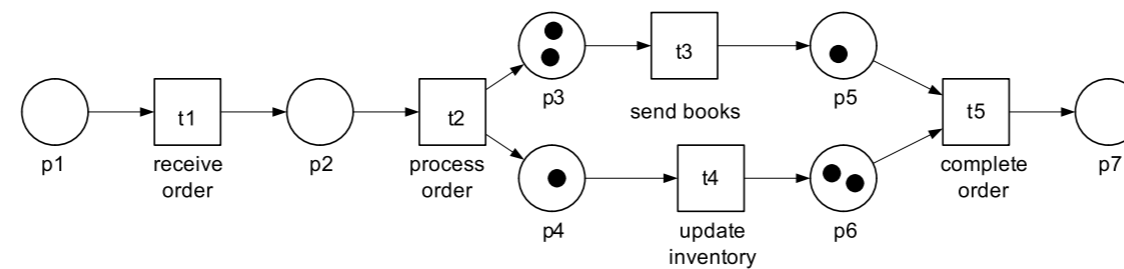
Roberto Bruni

http://www.di.unipi.it/~bruni

07 - Introduction to nets

1
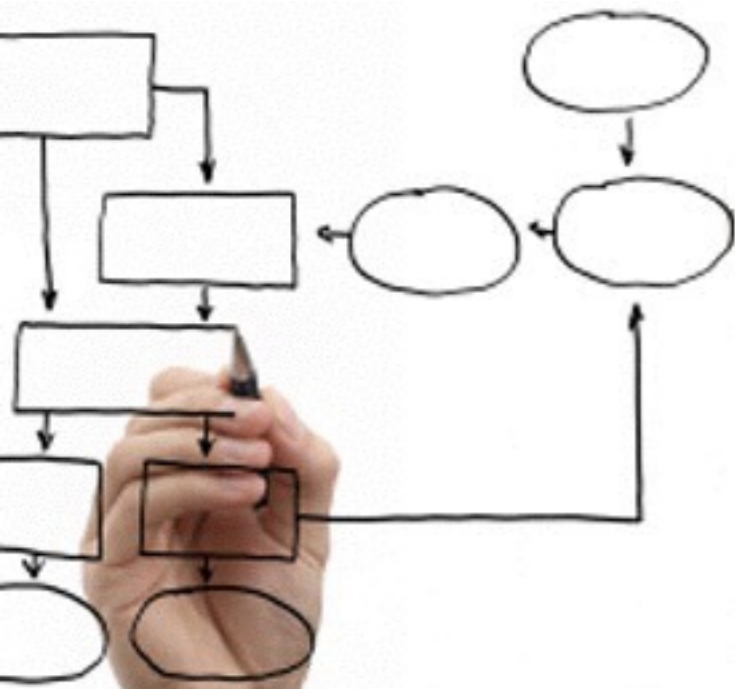
Object

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Overview of the basic concepts of Petri nets

Free Choice Nets (book, optional reading)
https://www7.in.tum.de/~esparza/bookfc.html

Why Petri nets

Business process analysis:
validation: testing correctness
verification: proving correctness
performance: planning and optimization

Use of Petri nets (or alike)
visual + formal
tool supported

Approaching Petri nets

Are you familiar with automata / transition systems?
They are fine for sequential protocols / systems
but do not capture concurrent behaviour directly

A Petri net is a mathematical model
of a parallel and concurrent system

in the same way that a finite automaton is a
mathematical model of a sequential system

Approaching Petri nets

Petri net theory can be studied
at several level of details

We study some basics aspects, relevant to the
analysis of business processes

Petri nets have a faithful and convenient graphical
representation, that we introduce and motivate next

Finite automata examples

Applications

Finite automata are widely used, e.g., in
protocol analysis,
text parsing,
video game character behavior,
security analysis,
CPU control units,
natural language processing,
speech recognition,
mechanical devices
(like elevators, vending machines, traffic lights)
and many more …

How to

Identify the admissible states of the system
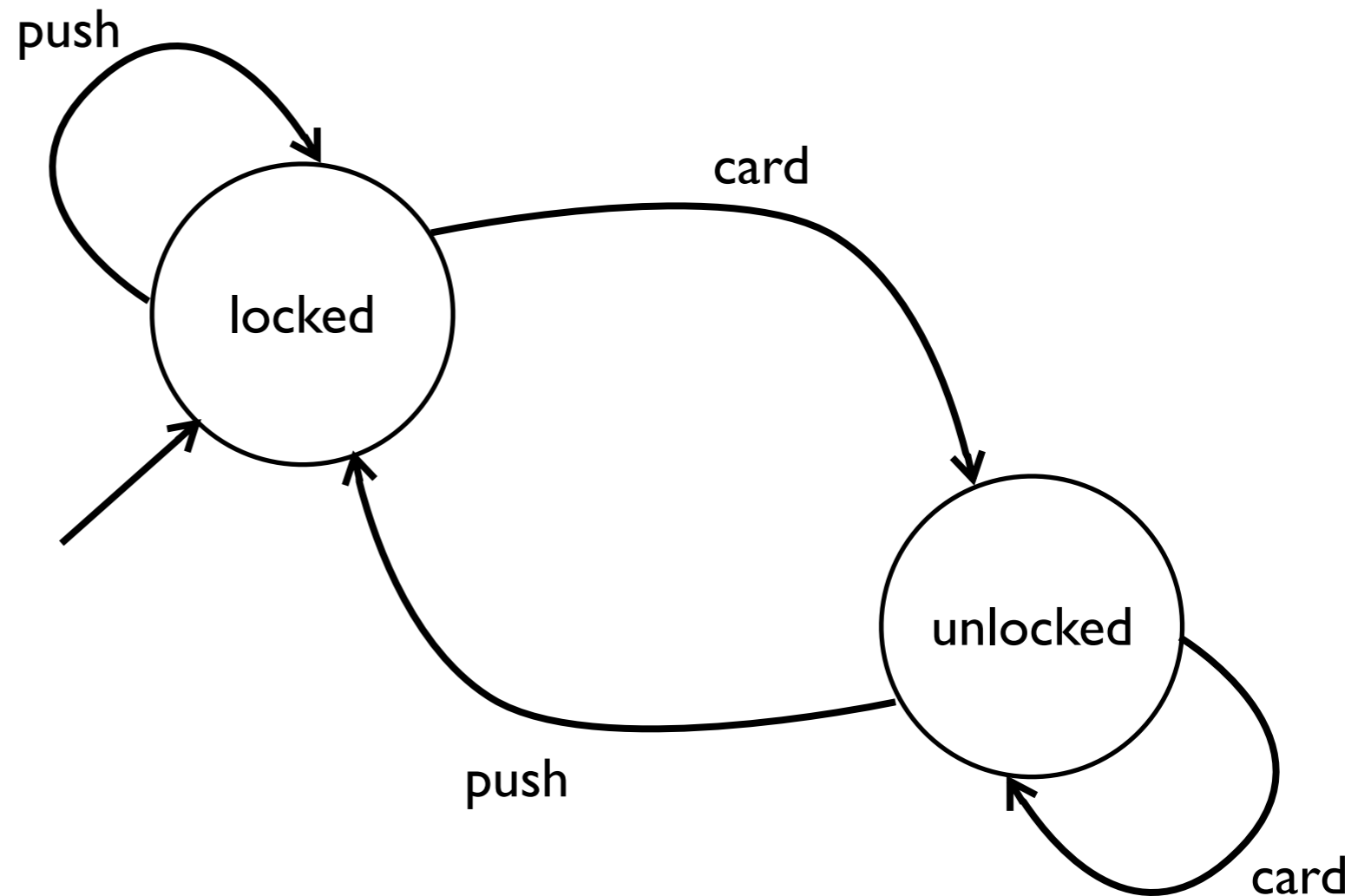Optional: Mark some states as error states

Add transitions
to move from one state to another
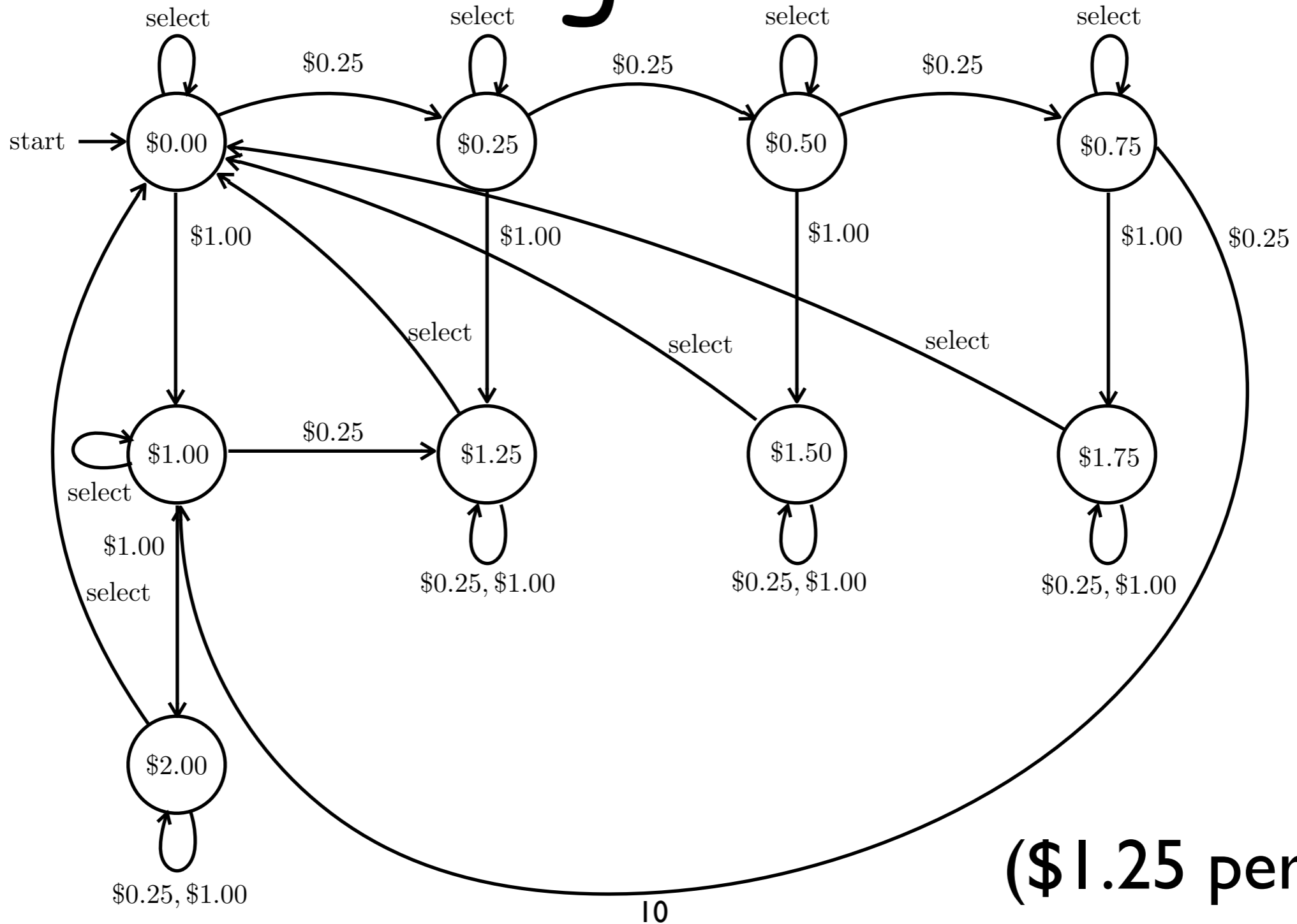(no transition to recover from error states)

Set the starting state

Optional: Mark some states as final

Example: Turnstile

push

card

locked

unlocked

push

card

Example: Vending Machine

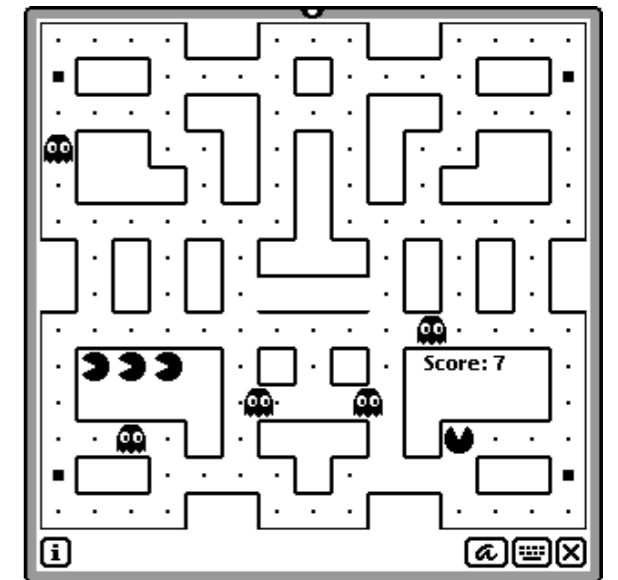
($1.25 per soda)

Computer controlled characters for games

States = characters behaviours

Transitions = labelled by events that cause a change in behaviour

Example:
Pac-man moves in a maze
wants to eat pills
is chased by ghosts

by eating power pills, pac-man can defeat ghosts
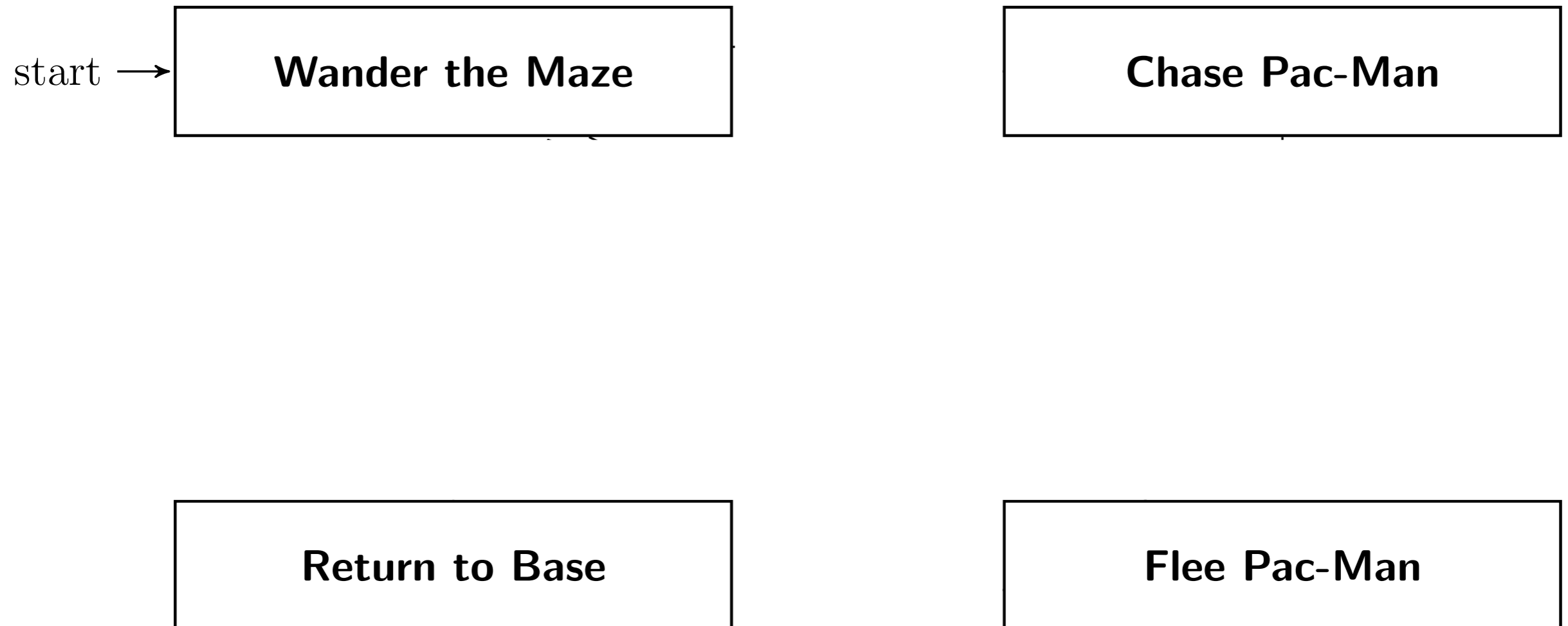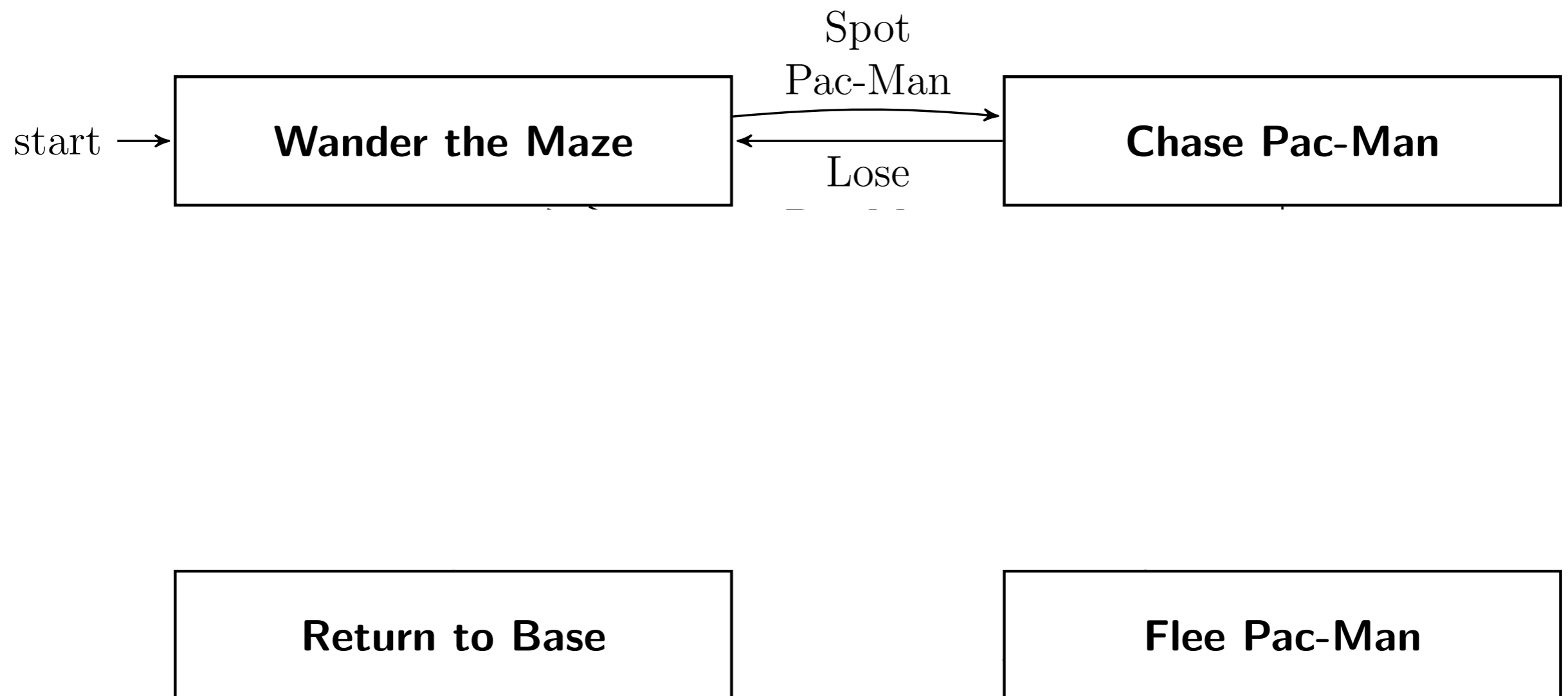
Example: Pac-Man Ghosts

Wander the Maze

Chase Pac-Man

Return to Base

Flee Pac-Man

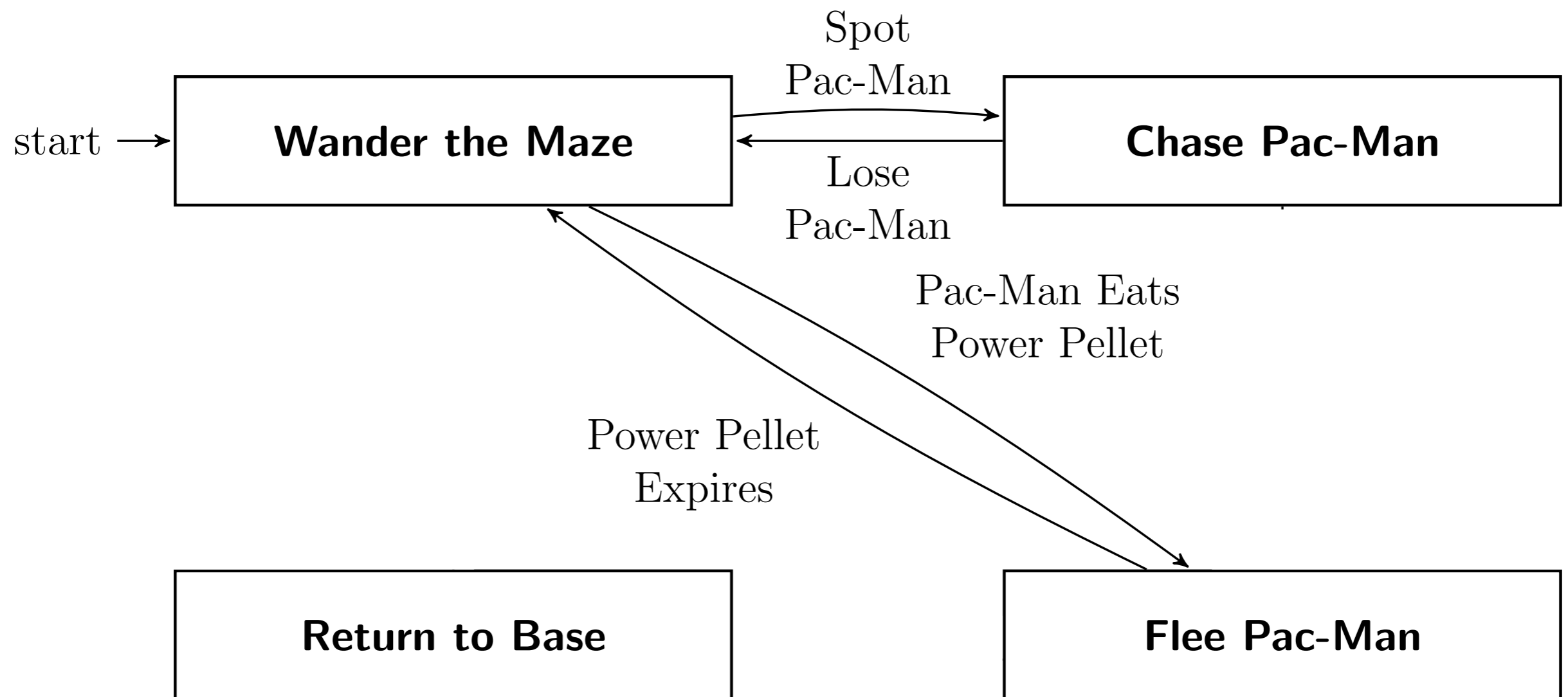# Example: Pac-Man Ghosts

start → **Wander the Maze**

Chase Pac-Man

Return to Base

Flee Pac-Man

Example:
Pac-Man Ghosts

start →

| Wander the Maze |

—— Spot Pac-Man ——→ | Chase Pac-Man |

←—— Lose ——

| Return to Base |

| Flee Pac-Man |

Example: Pac-Man Ghosts

start → **Wander the Maze** — Spot Pac-Man → **Chase Pac-Man**

Chase Pac-Man — Lose Pac-Man → **Wander the Maze**

Pac-Man Eats Power Pellet → **Flee Pac-Man**

Flee Pac-Man — Power Pellet Expires → **Wander the Maze**

Return to Base

Example: Pac-Man Ghosts

start → | **Wander the Maze** |

Spot Pac-Man →

Lose Pac-Man ←

| **Chase Pac-Man** |

Pac-Man Eats Power Pellet

Pac-Man Eats Power Pellet

Power Pellet Expires

| **Return to Base** |

| **Flee Pac-Man** |

Example:
Pac-Man Ghosts

start → **Wander the Maze**

Spot Pac-Man →

← Lose Pac-Man

Chase Pac-Man

Pac-Man Eats Power Pellet

Pac-Man Eats Power Pellet

Power Pellet Expires

Return to Base

Eaten by Pac-Man

Flee Pac-Man

Example:
Pac-Man Ghosts

start → **Wander the Maze**

Spot Pac-Man →

Lose Pac-Man ←

Chase Pac-Man

Reach Central Base

Pac-Man Eats Power Pellet

Power Pellet Expires

Pac-Man Eats Power Pellet

Return to Base

Eaten by Pac-Man

Flee Pac-Man

Exercises

Without adding states, draw the automata for a SuperGhost that can't be eaten. It chases Pac-Man when the power pill is eaten, but returns to base if Pac-Man eats a piece of fruit.

Choose a favourite (video) game, and try drawing the state automata for one of the computer controlled characters in that game.

From automata to Petri nets

DFA

A **Deterministic Finite Automaton** (**DFA**) is a tuple $A = (Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set of states;

- Σ is a finite set of input symbols;

- $\delta : Q \times \Sigma \to Q$ is the transition function;

- $q_0 \in Q$ is the initial state (also called start state);

- $F \subseteq Q$ is the set of final states (also accepting states)

Kleene-star notation A*

Given a set A we denote by A^*
the set of finite sequences of elements in A, i.e.:
$A^* = \{\, a_1 \cdots a_n \mid n \geq 0 \wedge a_1, ..., a_n \in A \,\}$
We denote the empty sequence by $\epsilon \in A^*$

For example:
$A = \{\, a, b \,\}$ $A^* = \{\, \epsilon, a, b, aa, ab, ba, bb, aaa, aab, ... \,\}$

Extended transit. func. (destination function)

Given $A = (Q, \Sigma, \delta, q_0, F)$, we define $\widehat{\delta} : Q \times \Sigma^* \to Q$ by induction:

base case: For any $q \in Q$ we let
$$\widehat{\delta}(q, \epsilon) = q$$

inductive case: For any $q \in Q, a \in \Sigma, w \in \Sigma^*$ we let

$$\widehat{\delta}(q, wa) = \delta(\, \widehat{\delta}(q, w)\, ,\, a\,)$$

$(\widehat{\delta}(q, w)$ returns the state reached from q by observing $w)$

String processing

Given $A = (Q, \Sigma, \delta, q_0, F)$ and $w \in \Sigma^*$ we say that A **accept** w iff

$$\widehat{\delta}(q_0, w) \in F$$

The **language** of $A = (Q, \Sigma, \delta, q_0, F)$ is

$$L(A) = \{ \ w \ | \ \widehat{\delta}(q_0, w) \in F \ \}$$

Transition diagram

We represent $A = (Q, \Sigma, \delta, q_0, F)$ as a graph s.t.

- Q is the set of nodes;

- $\{\, q \xrightarrow{a} q' \mid q' = \delta(q, a) \,\}$ is the set of arcs.

Plus some graphical conventions:

- there is one special arrow $Start$ with $\xrightarrow{Start} q_0$

- nodes in F are marked by double circles;

- nodes in $Q \setminus F$ are marked by single circles.

String processing as paths

A DFA accepts a string w, if there is a path in its transition diagram such that:

it starts from the initial state

it ends in one final state

the sequence of labels in the path is exactly w

DFA: example
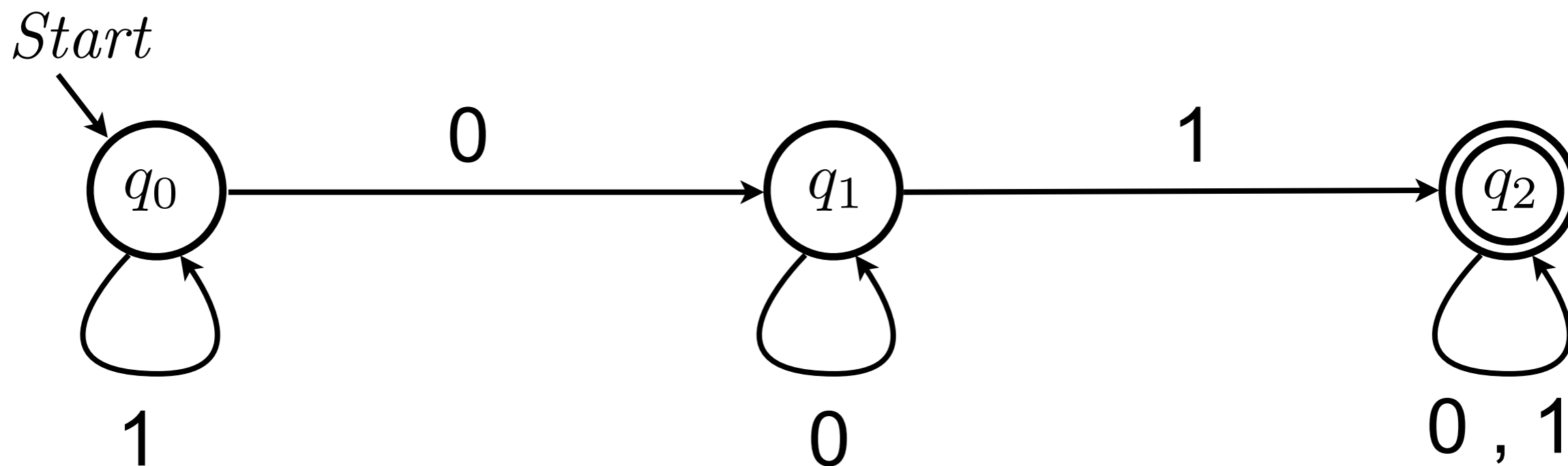
q_0	1	q_0	1	q_0	1	q_0	0	q_1	0	q_1	0	$q_1 \notin F$

q_0	1	q_0	0	q_1	0	q_1	1	q_2	1	q_2	0	$q_2 \in F$

DFA: question time
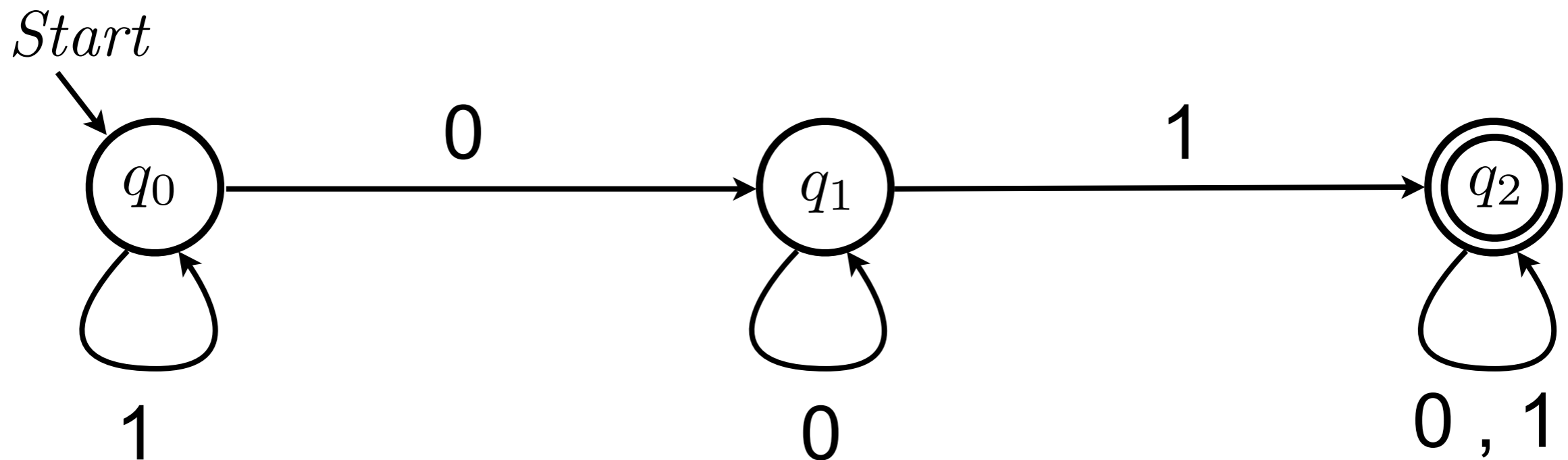
Does it accept 100 ?
Does it accept 011 ?
Does it accept 1010010 ?
What is L(A) ?

Transition table

Conventional tabular representation

its rows are in correspondence with states

its columns are in correspondence with input symbols

its entries are the states reached after the transition

Plus some decoration

start state decorated with an arrow

all final states decorated with *

Transition table

				a			
→							
q				$\delta(q,a)$			
*							
*							

DFA: example
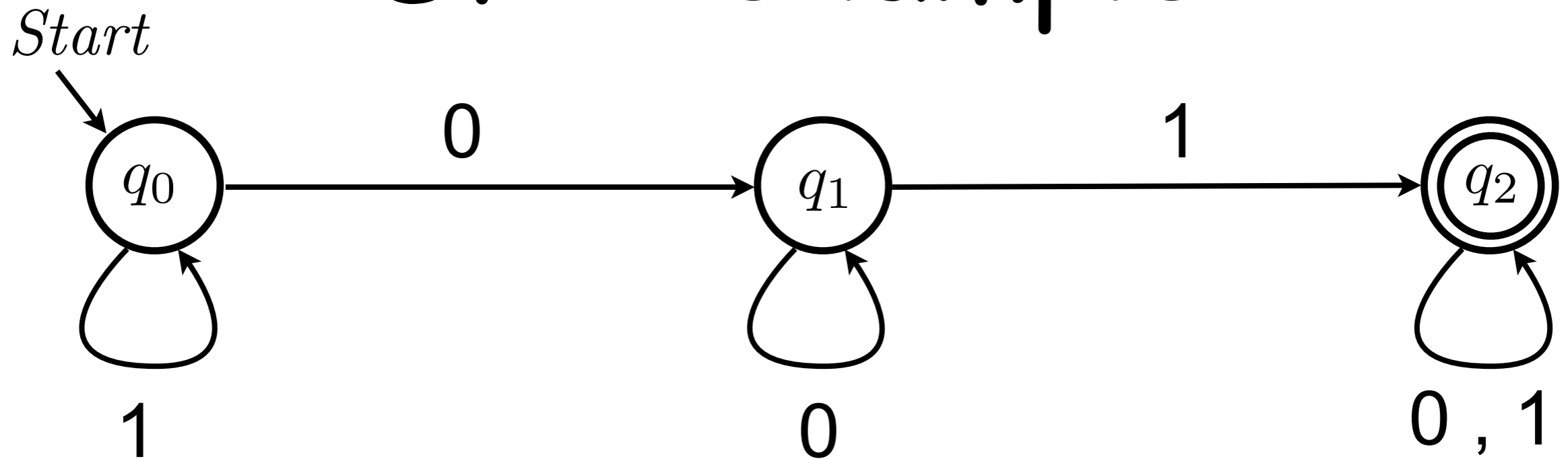
	0	**1**
→ q_0		
q_1		
⋆ q_2		

DFA: exercise
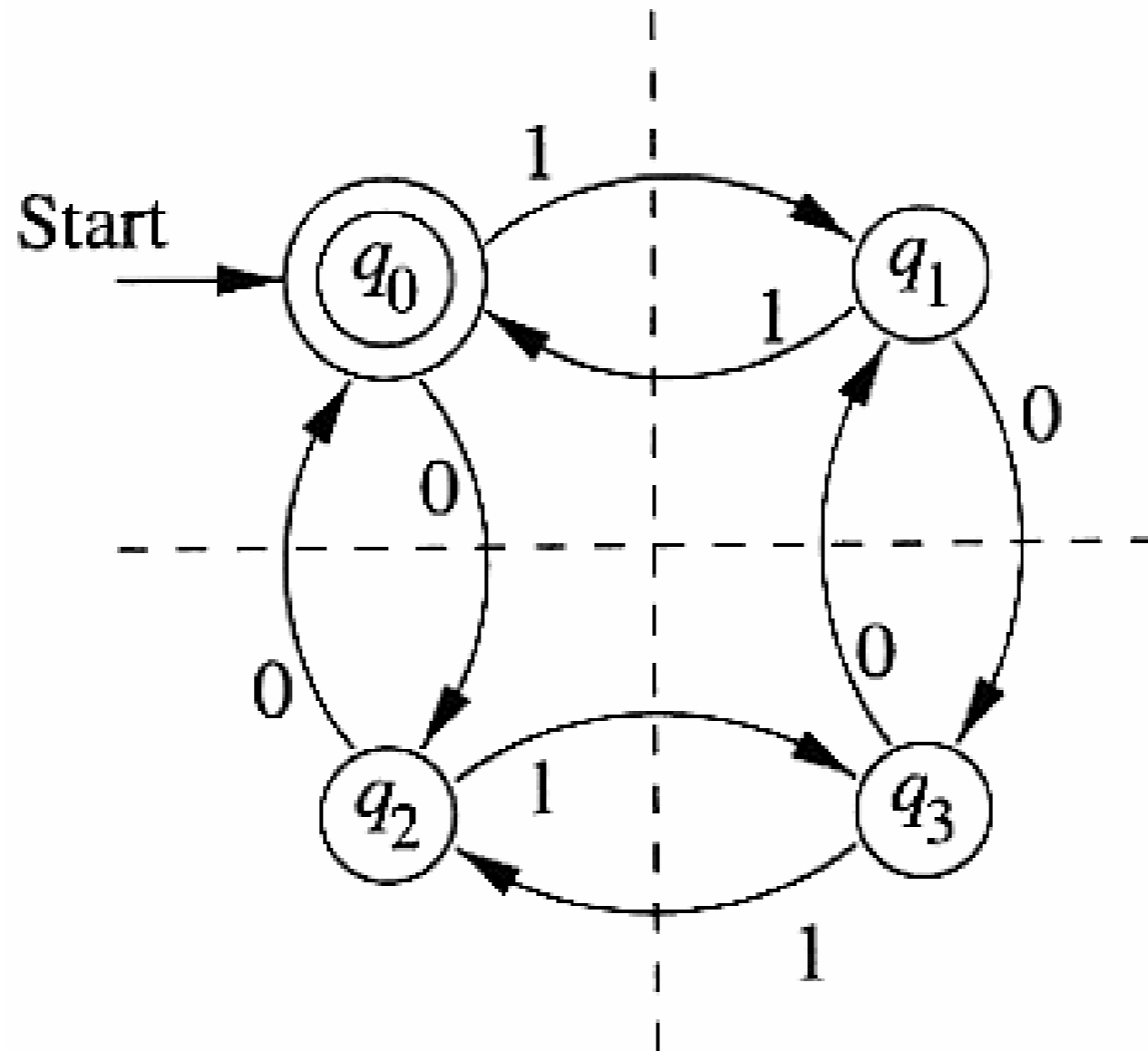
Does it accept 100 ?
Write its transition table.

Does it accept 1010 ?
What is L(A) ?

NFA

A **Non-deterministic Finite Automaton** (**NFA**) is a tuple $A = (Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set of states;

- Σ is a finite set of input symbols;

  powerset of Q = set of sets over Q

- $\delta : Q \times \Sigma \to \wp(Q)$ is the transition function;

- $q_0 \in Q$ is the initial state (also called start state);

- $F \subseteq Q$ is the set of final states (also accepting states)

NFA: example
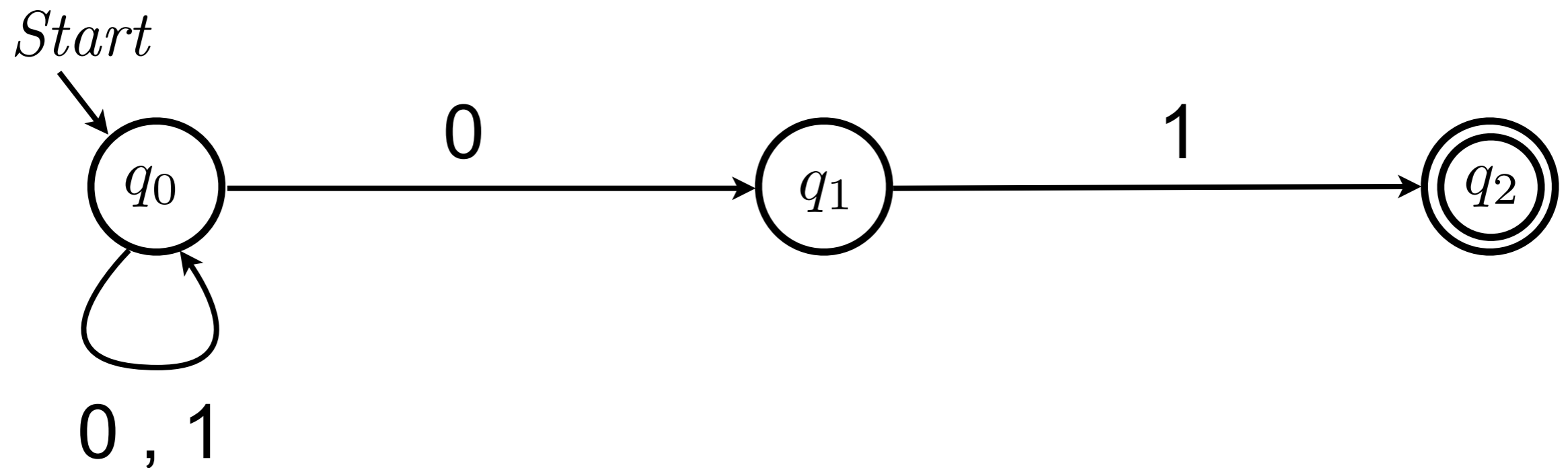
Can you explain why it is not a DFA?

NFA: example

Can you explain why it is not a DFA?

36
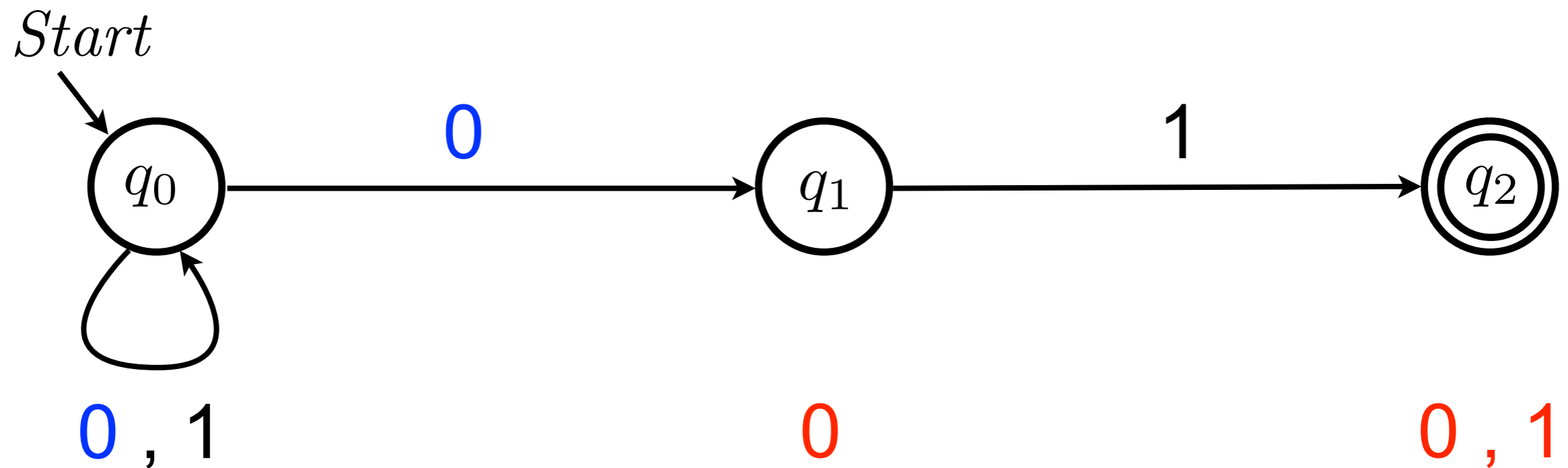
Reshaping

Step 1: get a token

Start

q_0 —0→ q_1 —1→ q_2

q_0 loop: 0 , 1

0	1	0	0	1
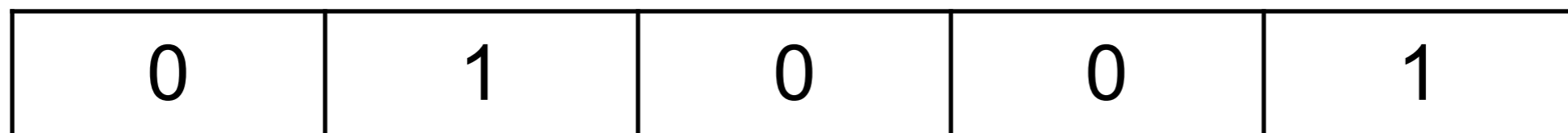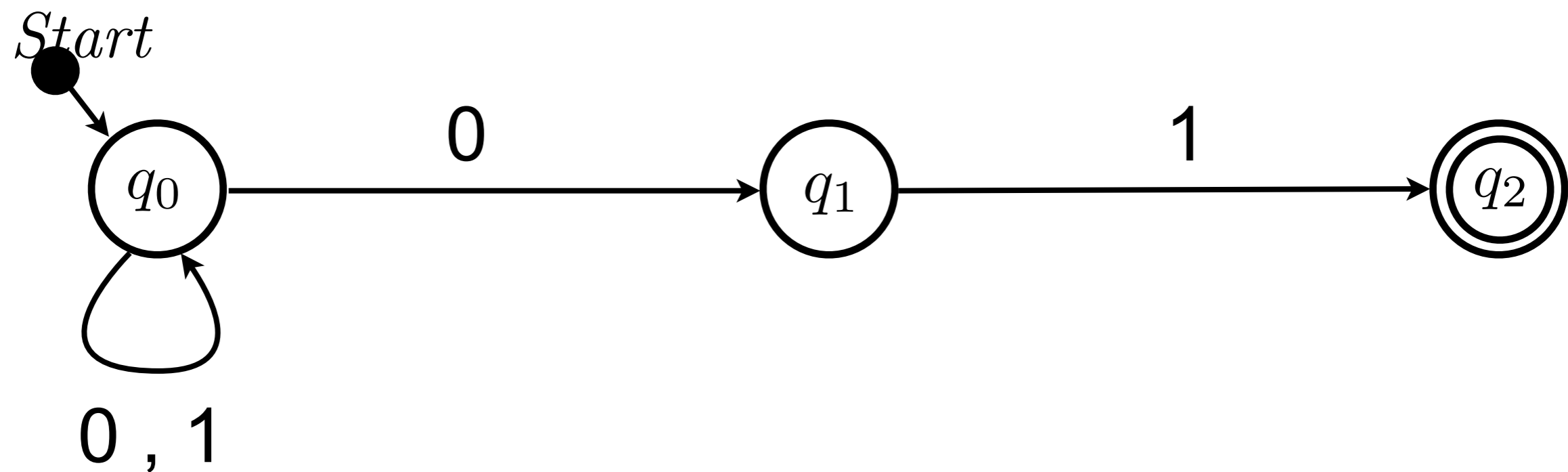
Step 2: forget initial state decoration
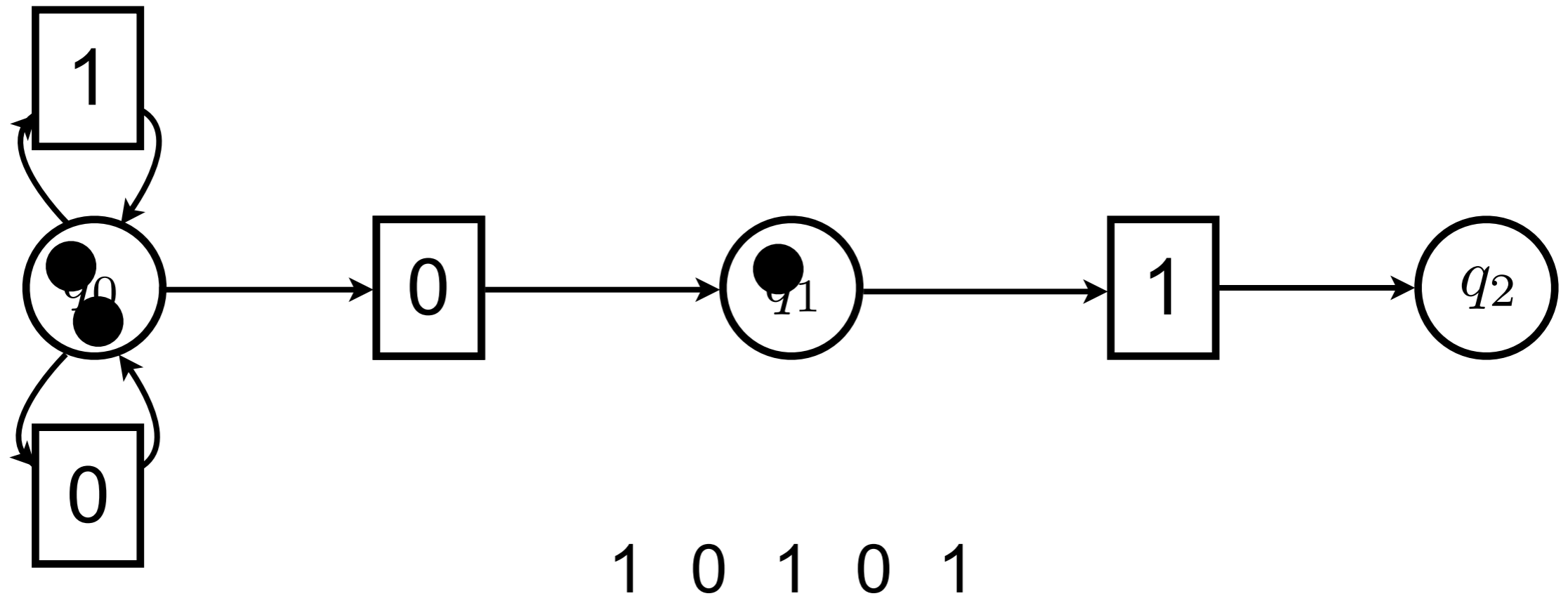

0 , 1
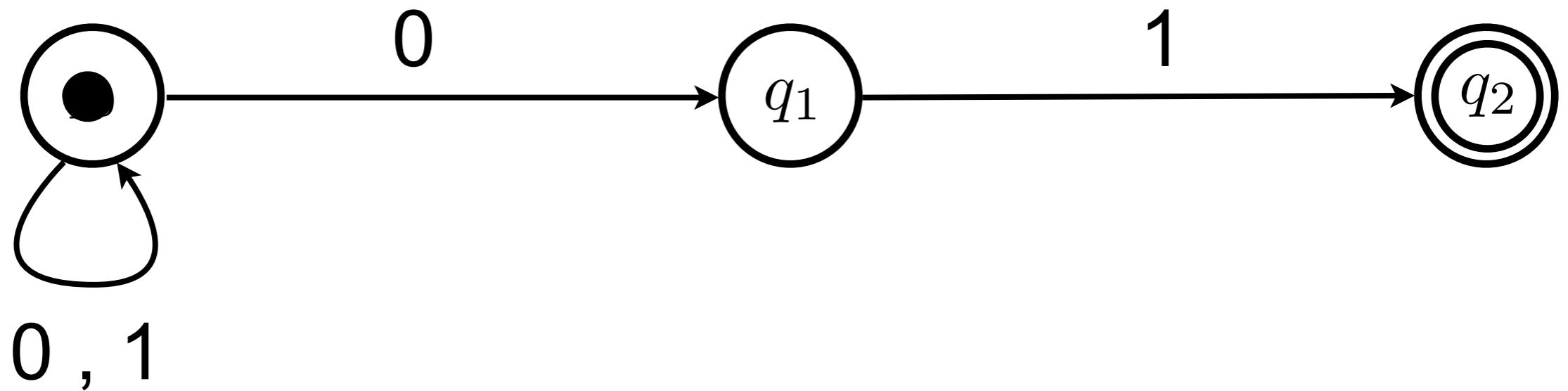
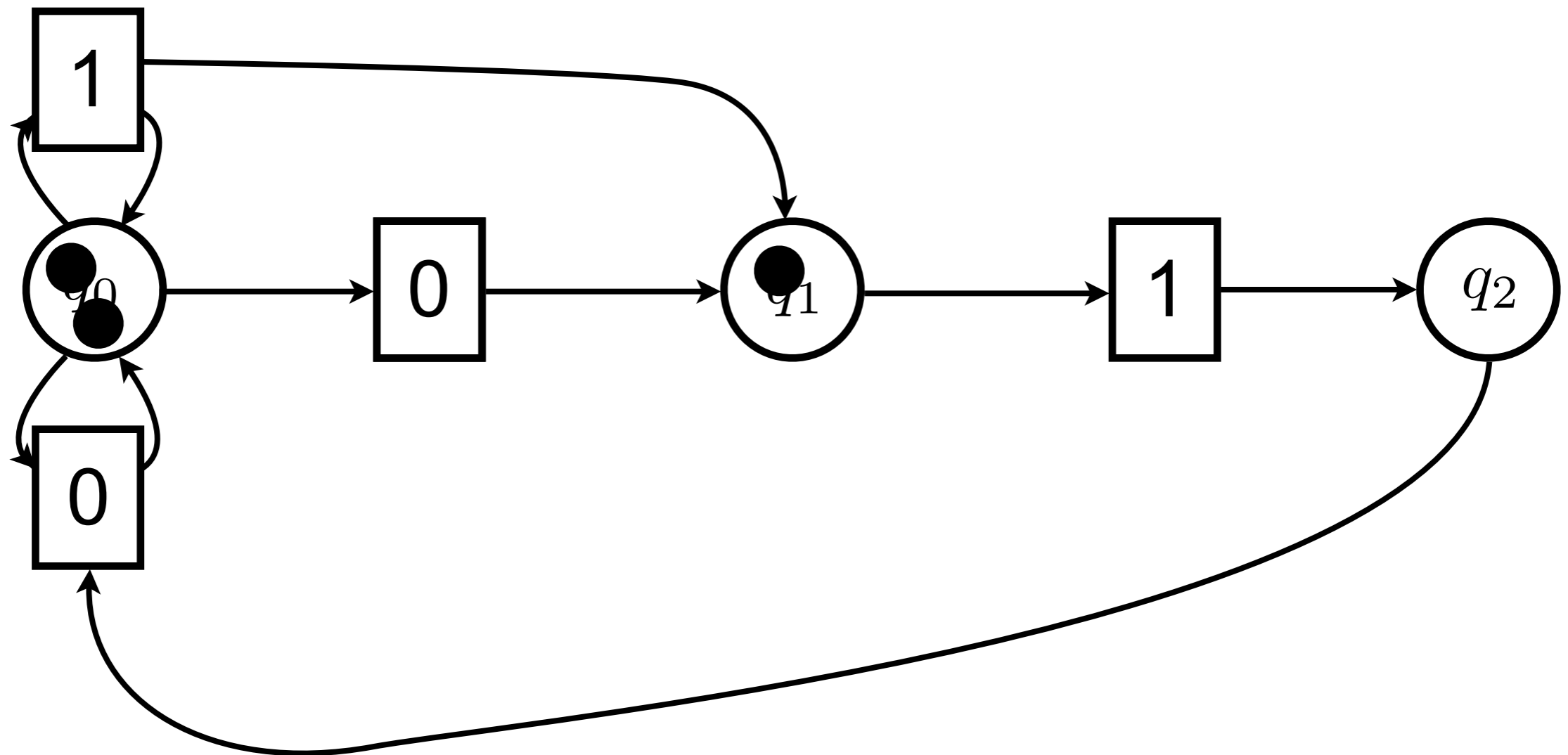# Step 3: transitions as boxes
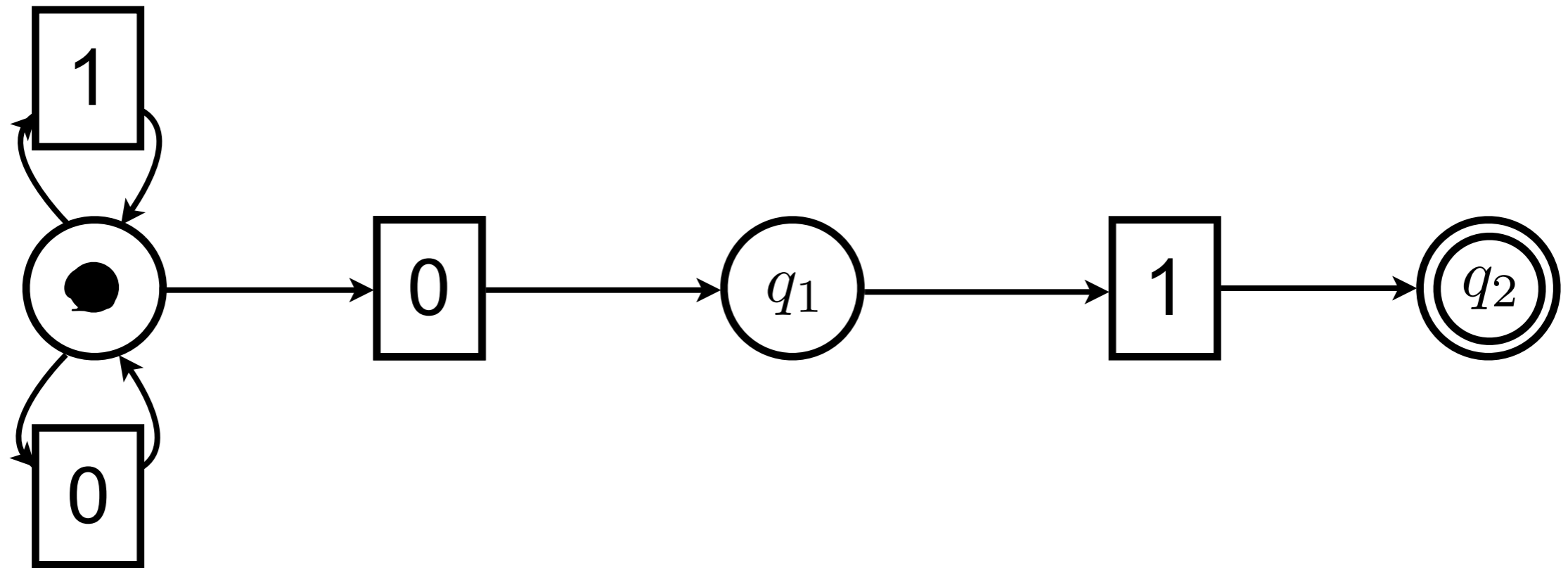
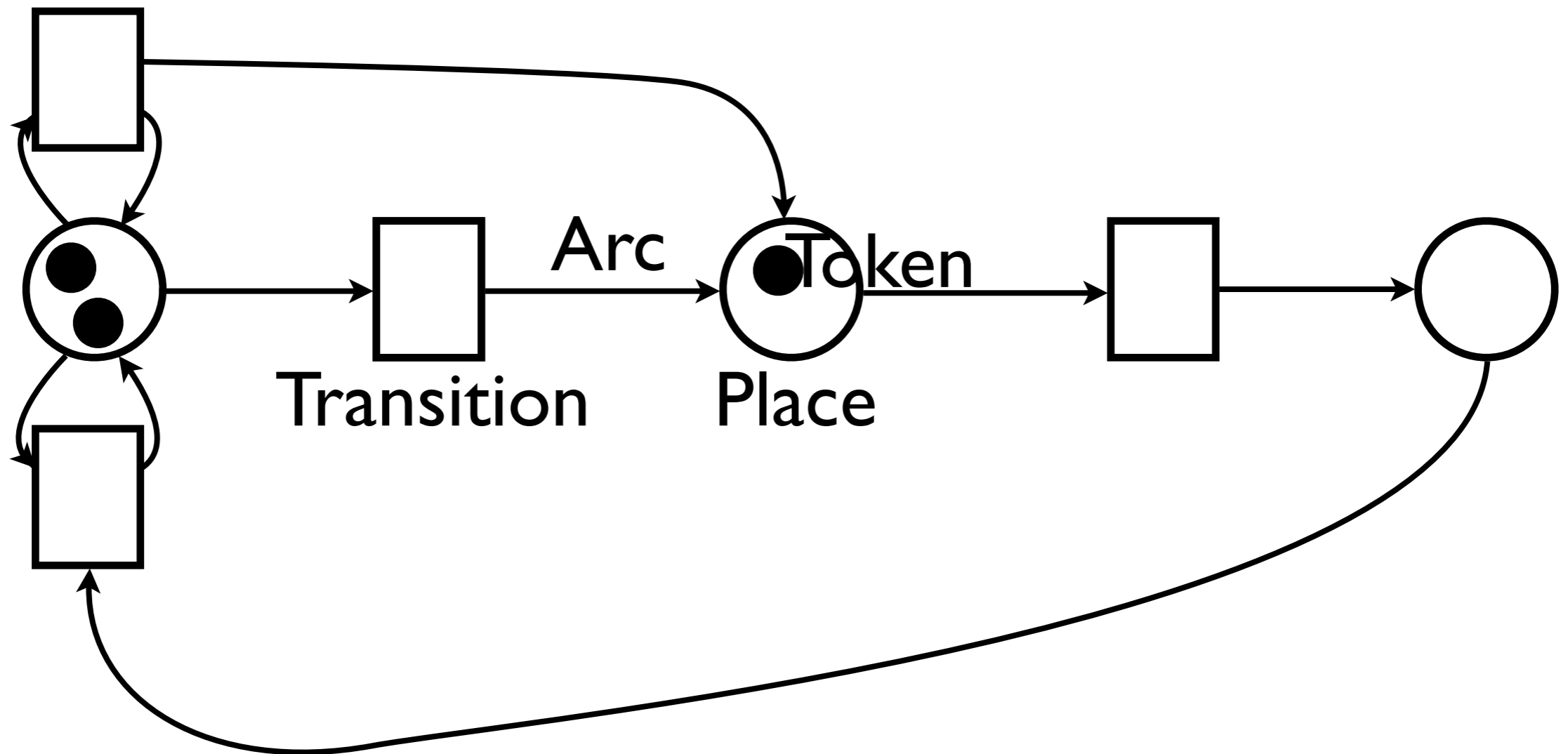# Step 4: forget final states
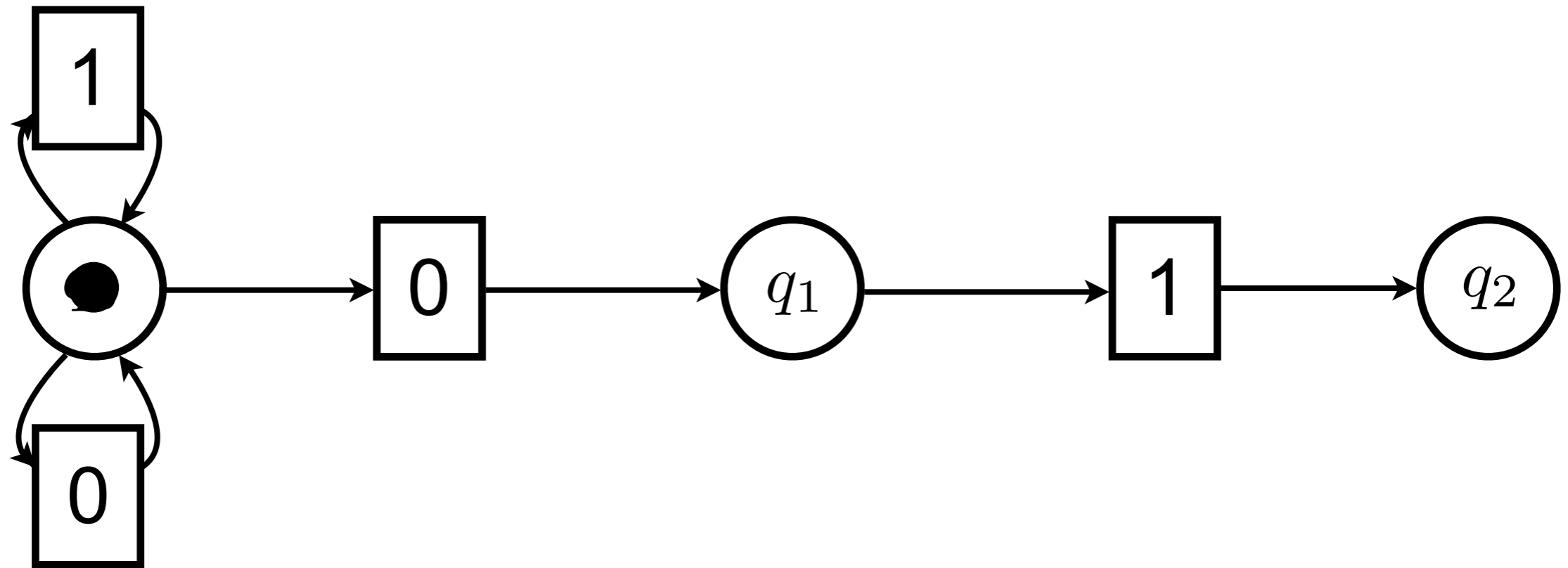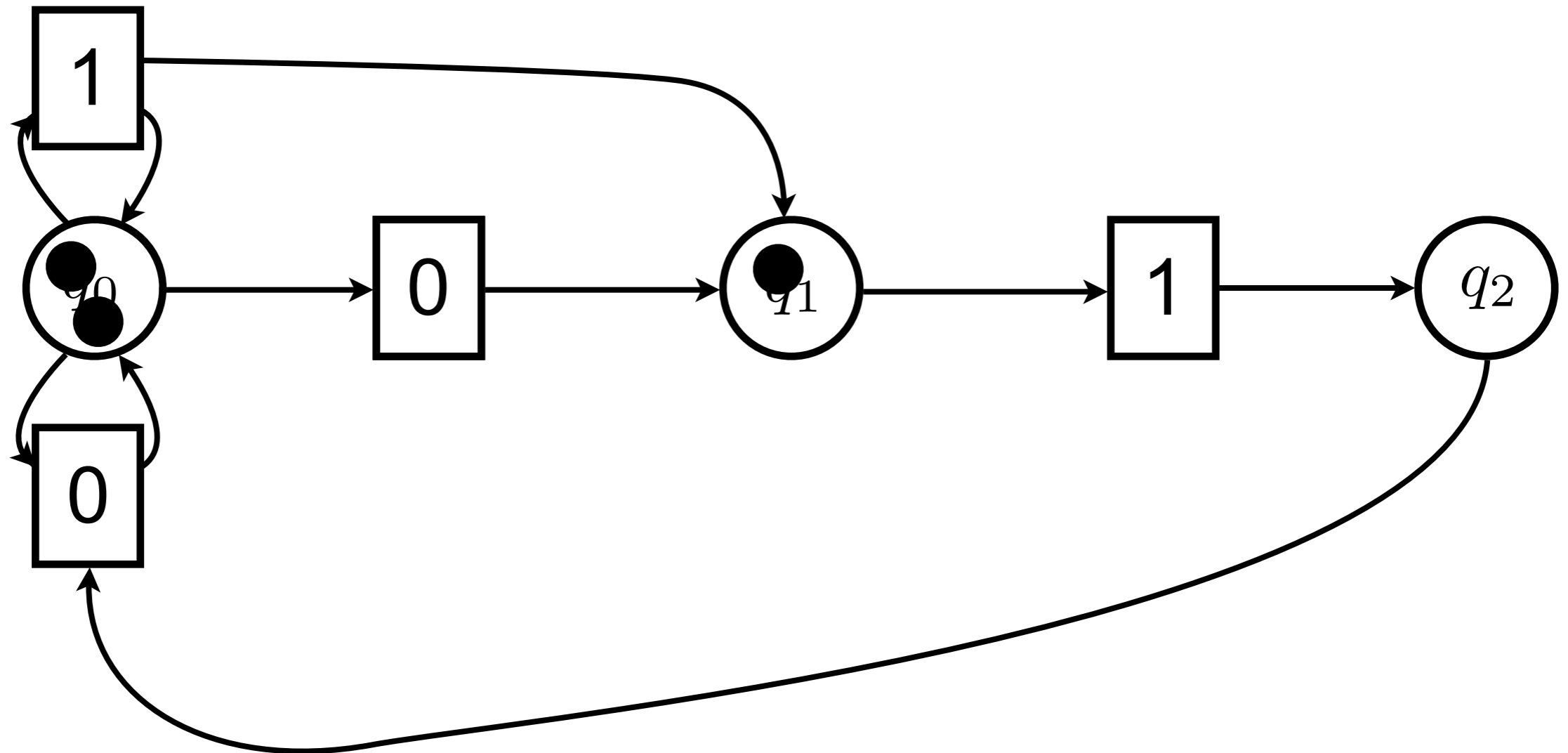
Step 5: allow for more tokens

Example: token game

1 0 1 0 1
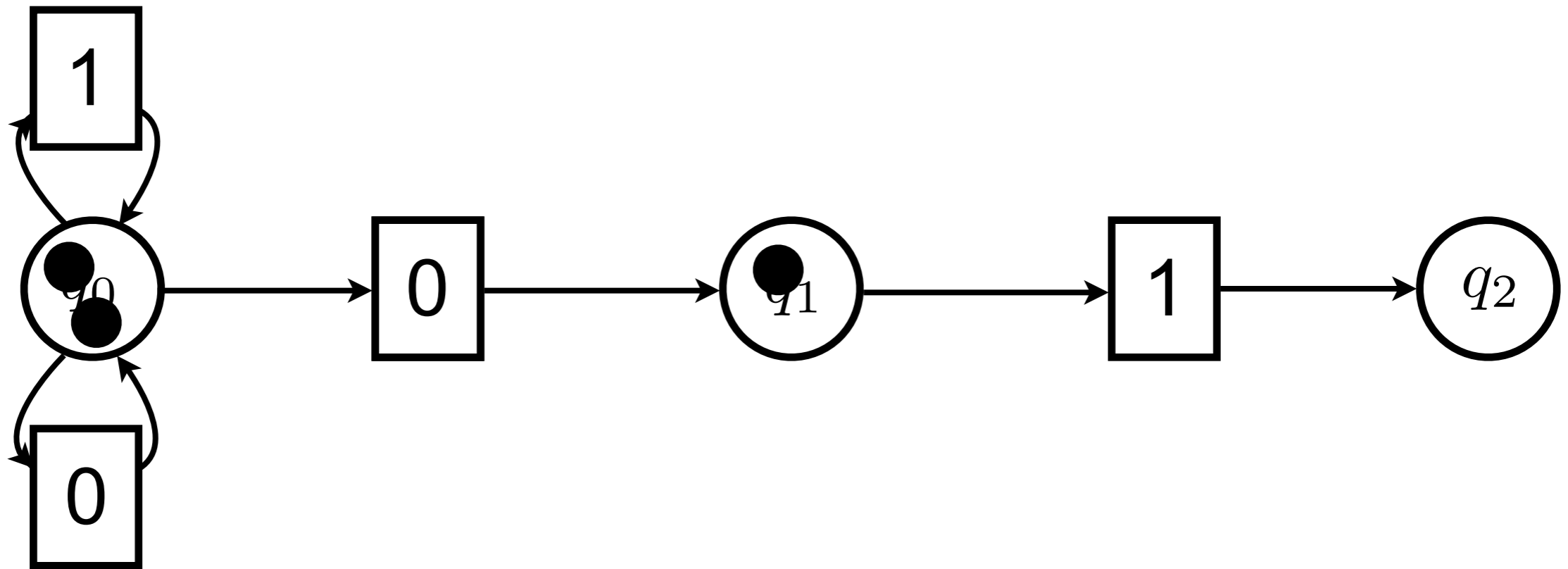
Step 6: allow for more arcs

Terminology
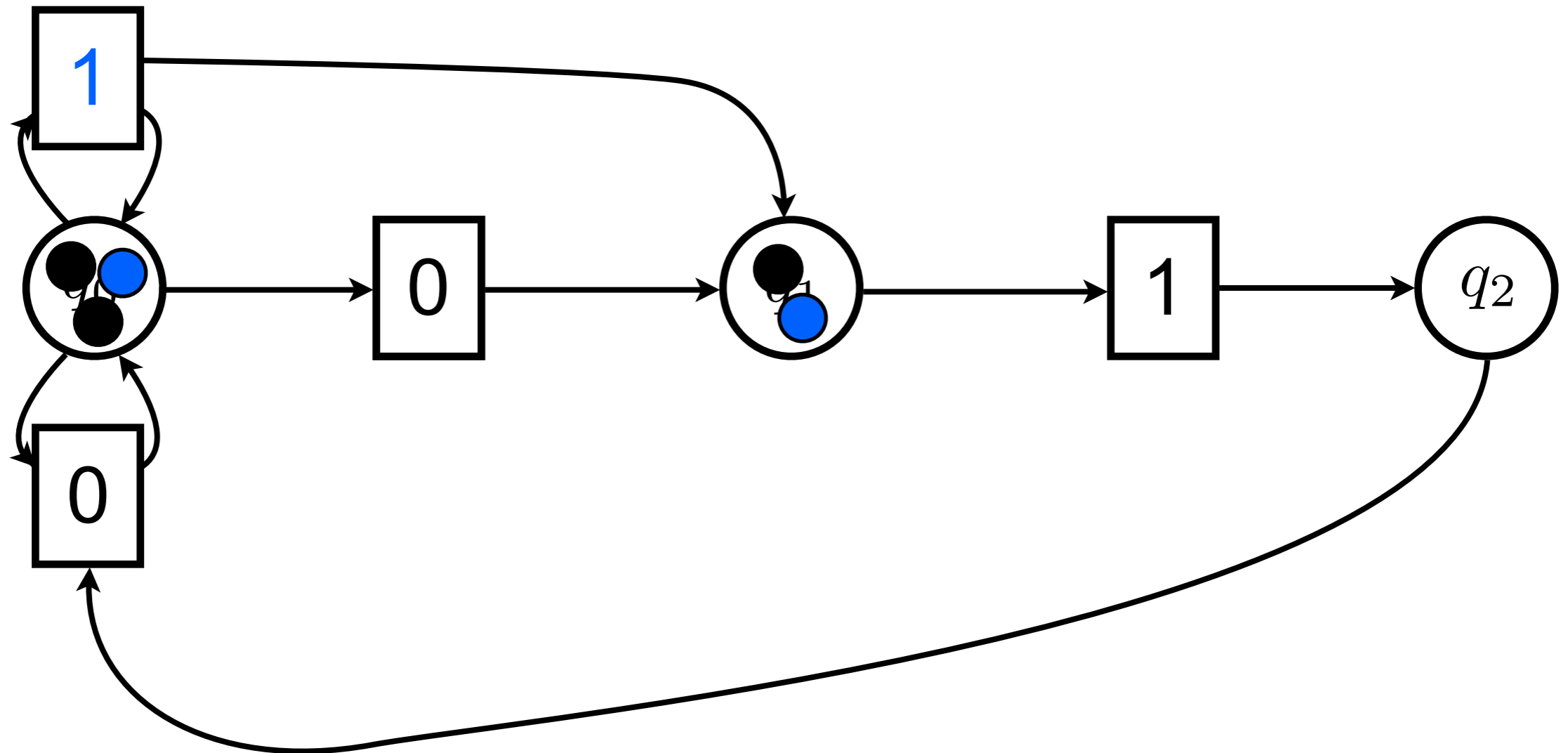

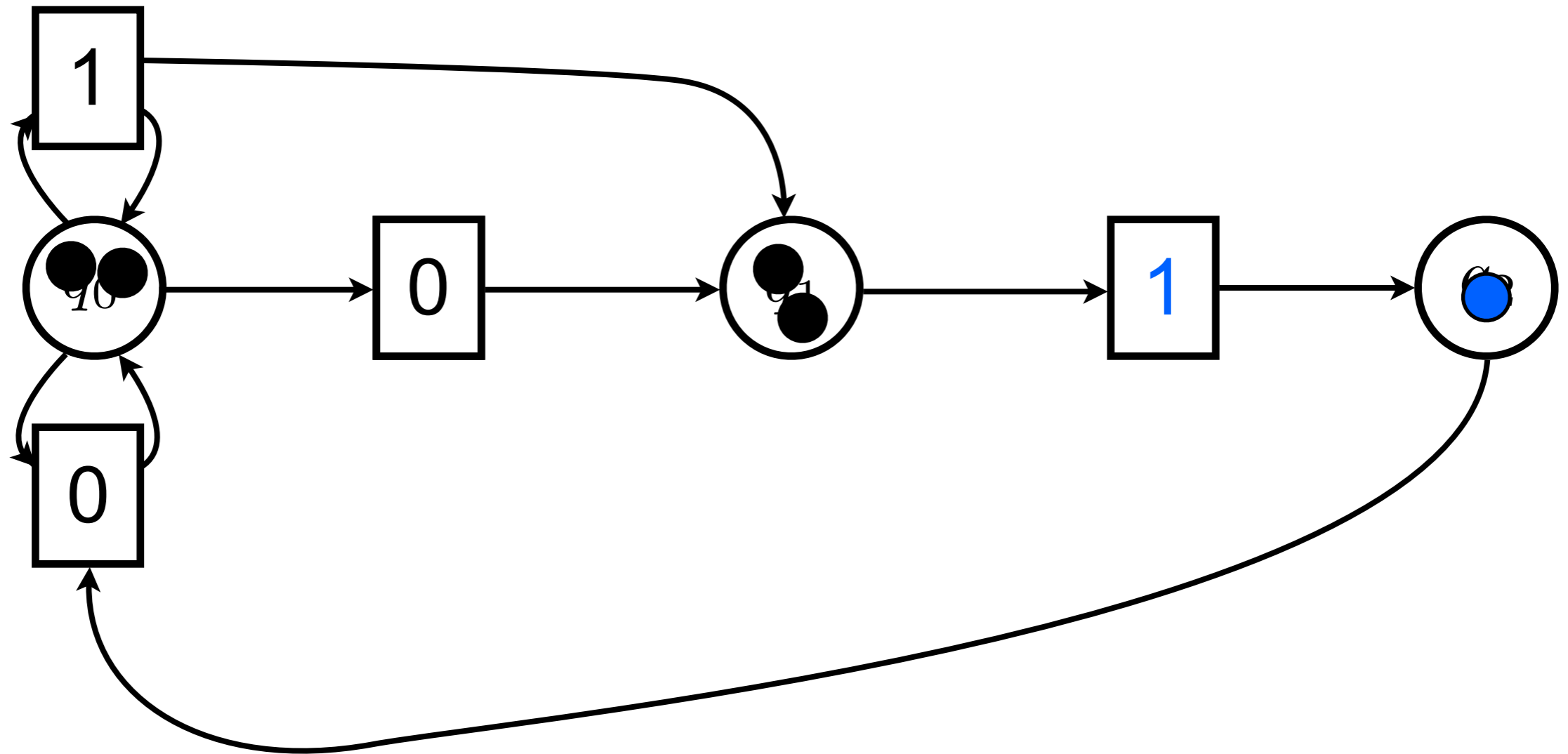
Arc   Token

Transition   Place
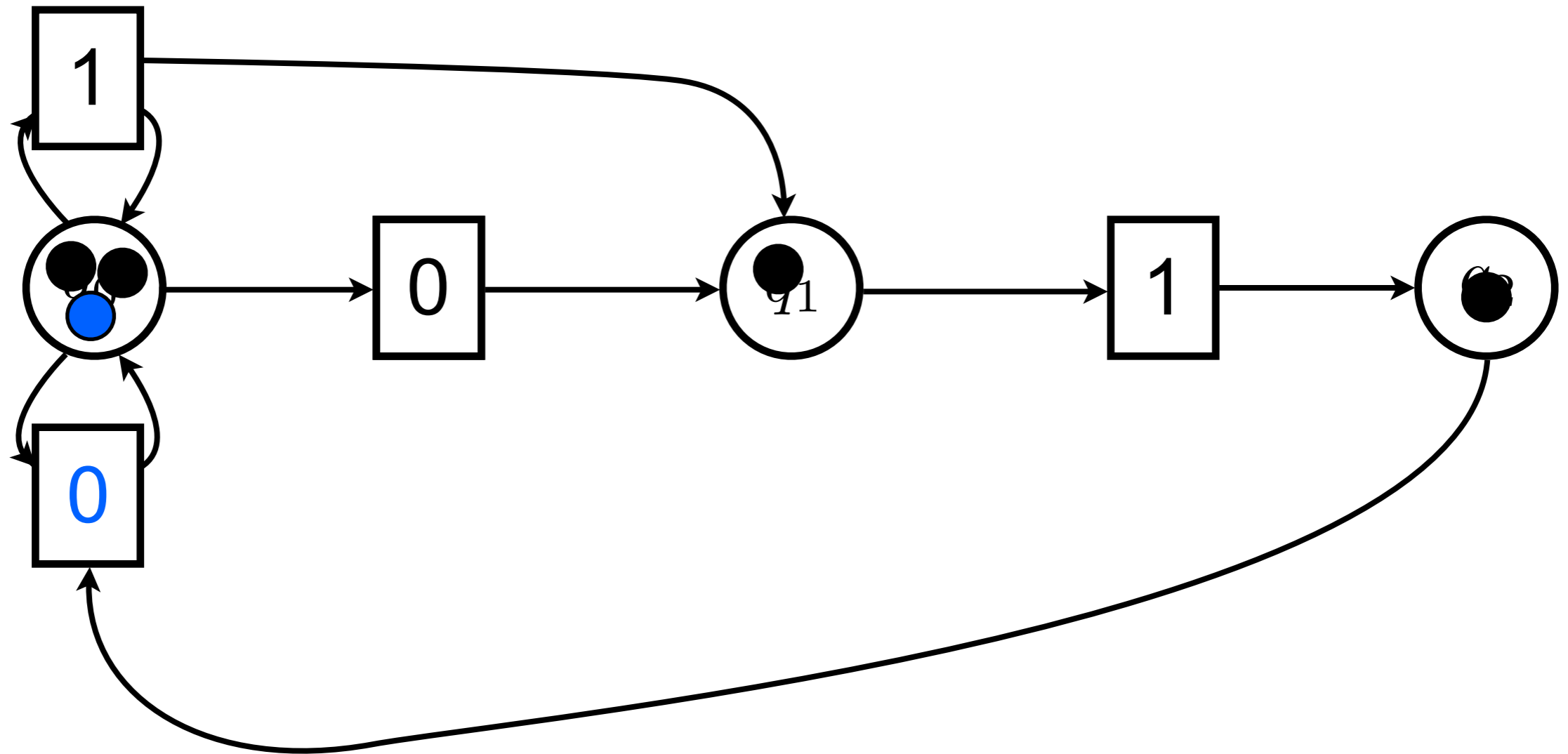
45

Example: token game

Example: token game
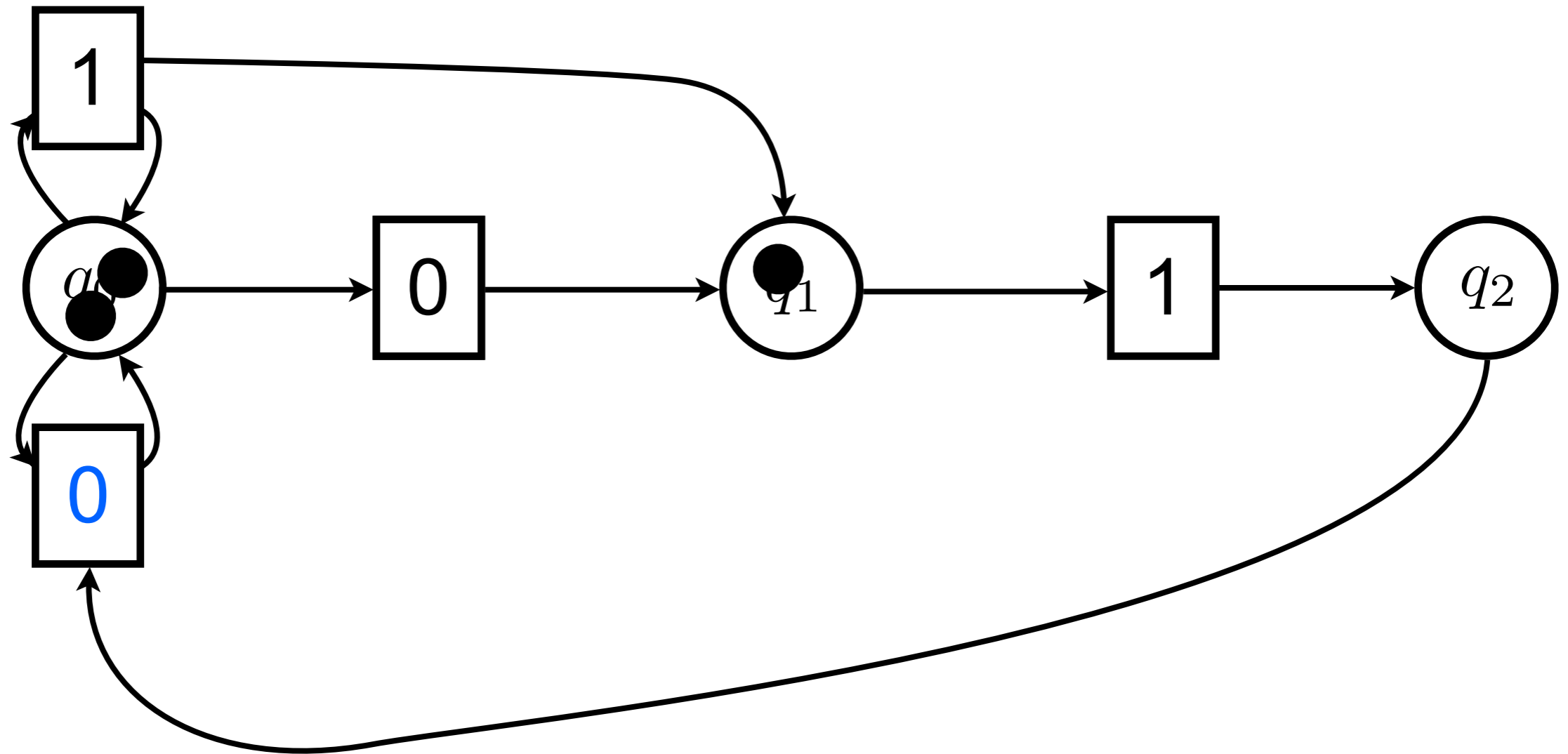
Example: token game

Example: token game

Example: token game

50

Some hints

Nets are **bipartite graphs**:
arcs never connect two places
arcs never connect two transitions

Static structure for dynamic systems:
places, transitions, arcs do not change
tokens move around places

Places are passive components
Transitions are active components:
tokens do not flow!
(they are removed or freshly created)