

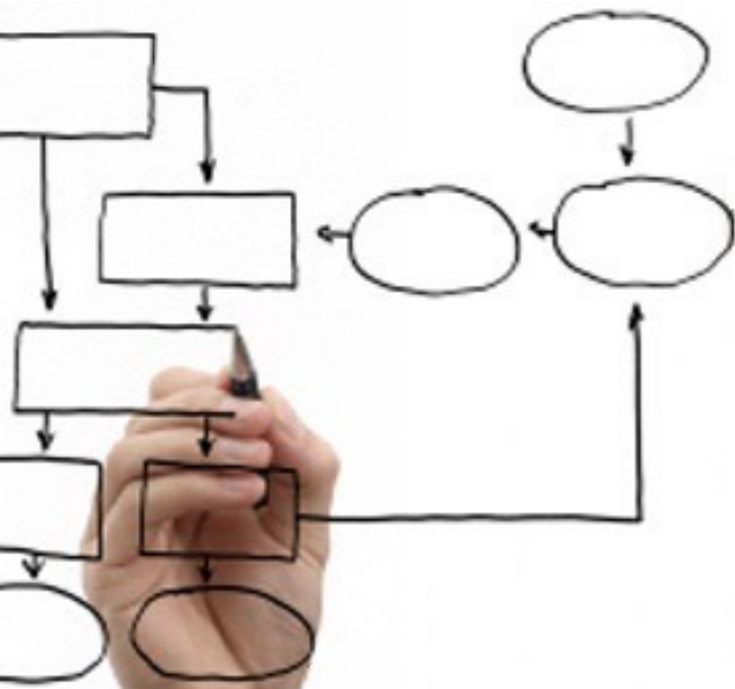
Business Processes Modelling

MPB (6 cfu, 295AA)

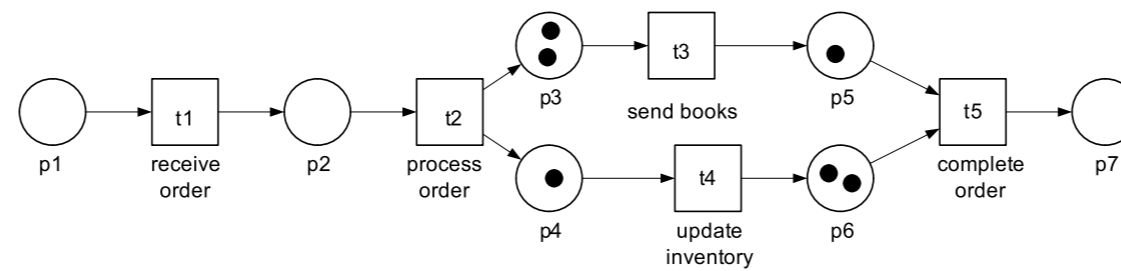
Roberto Bruni

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

07 - From automata to nets



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 define an automaton

1. Identify the admissible **states** of the system
(*Optional: mark some states as error states*)

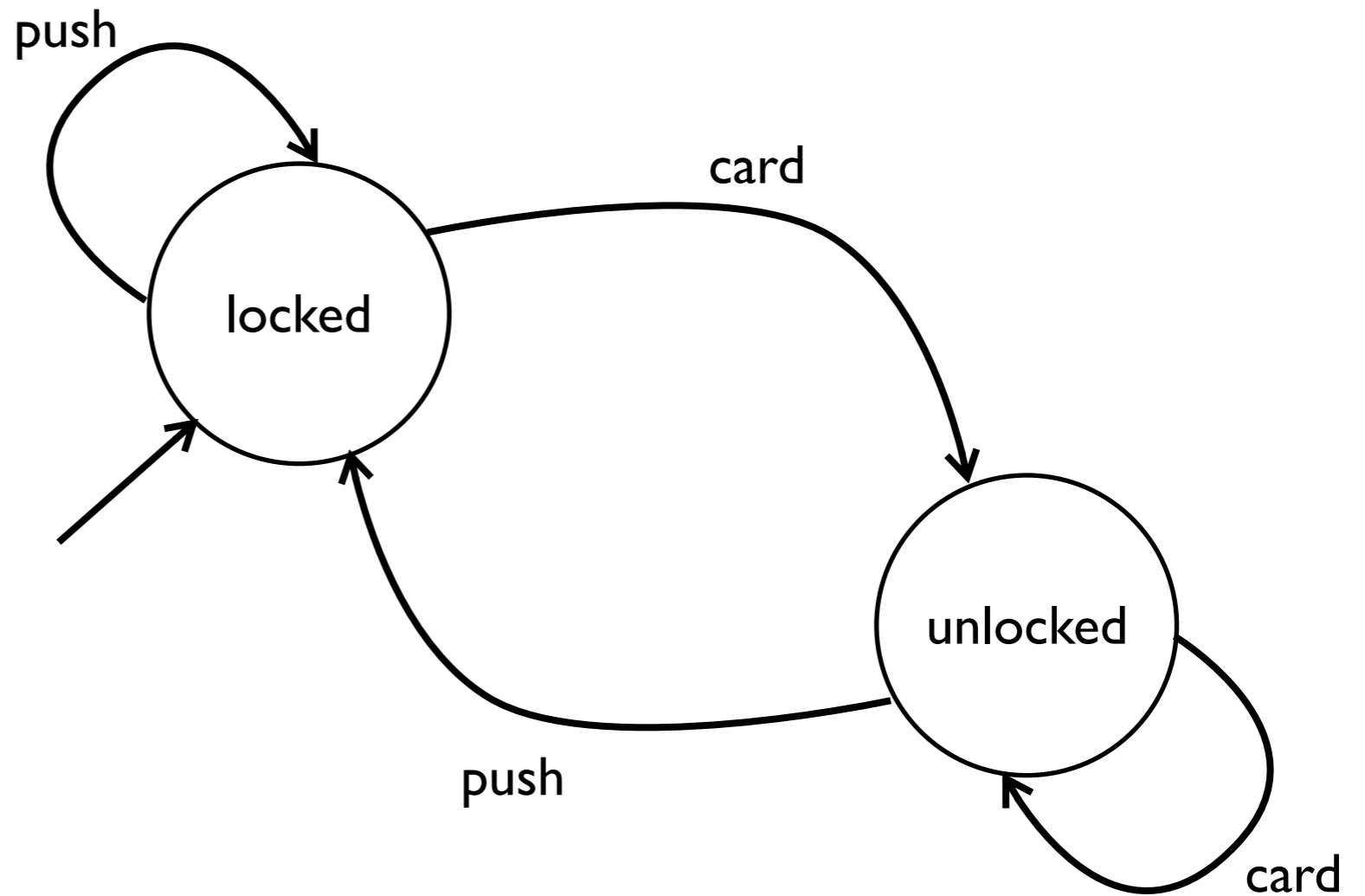
2. **Add transitions**

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

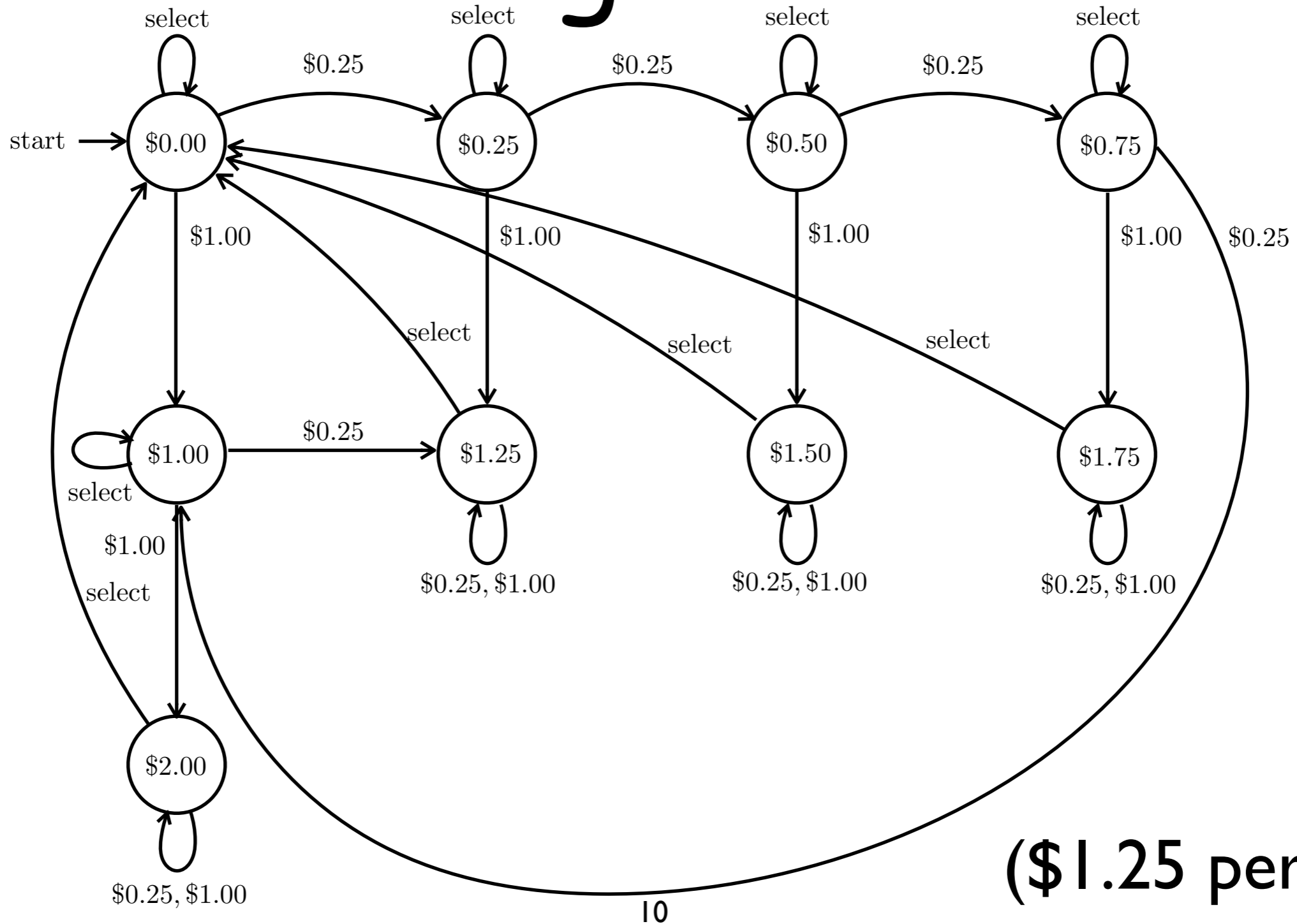
3. Set the **initial state**

4. (*Optional: mark some states as **final states***)

Example: Turnstile

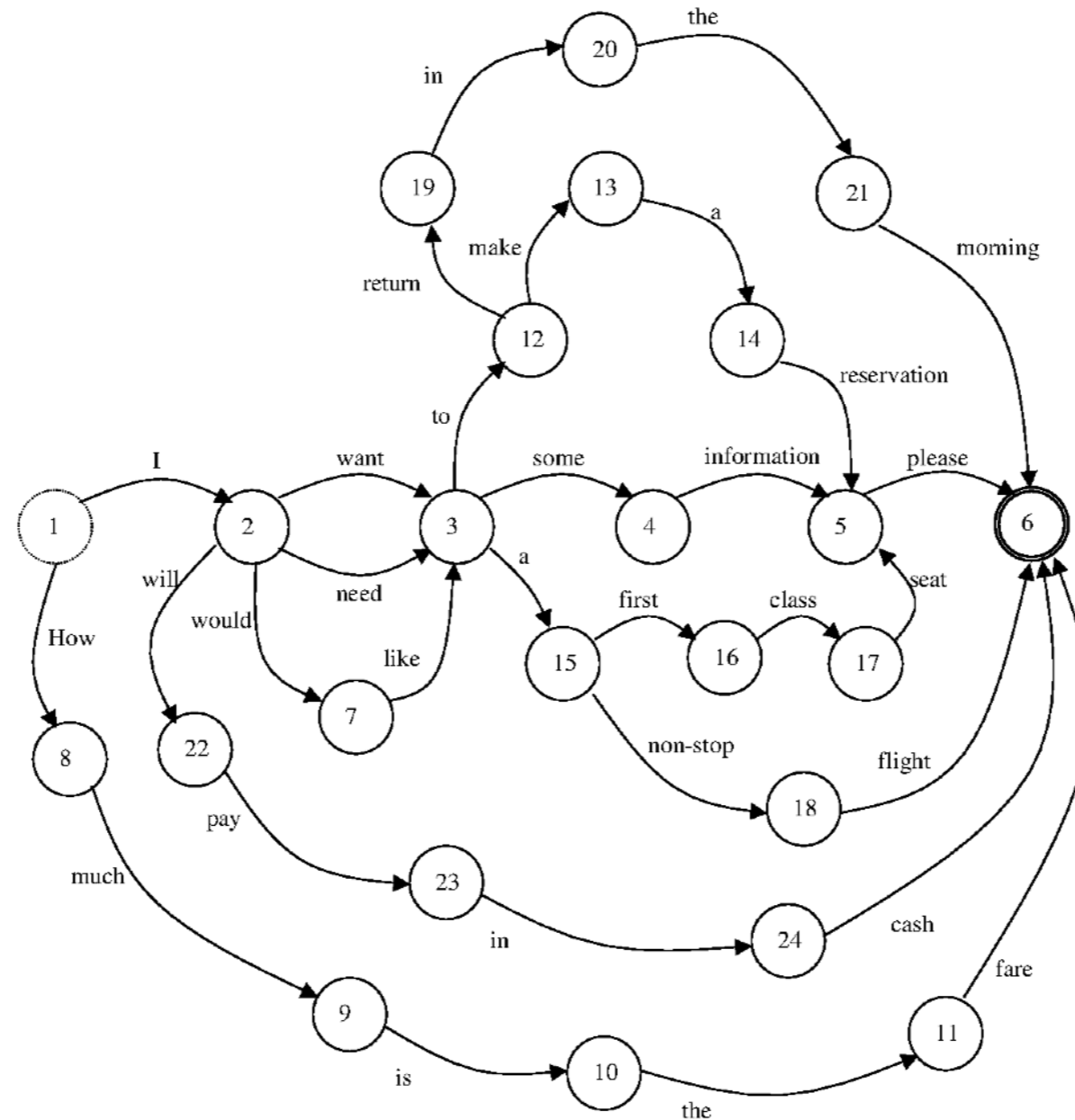


Example: Vending Machine

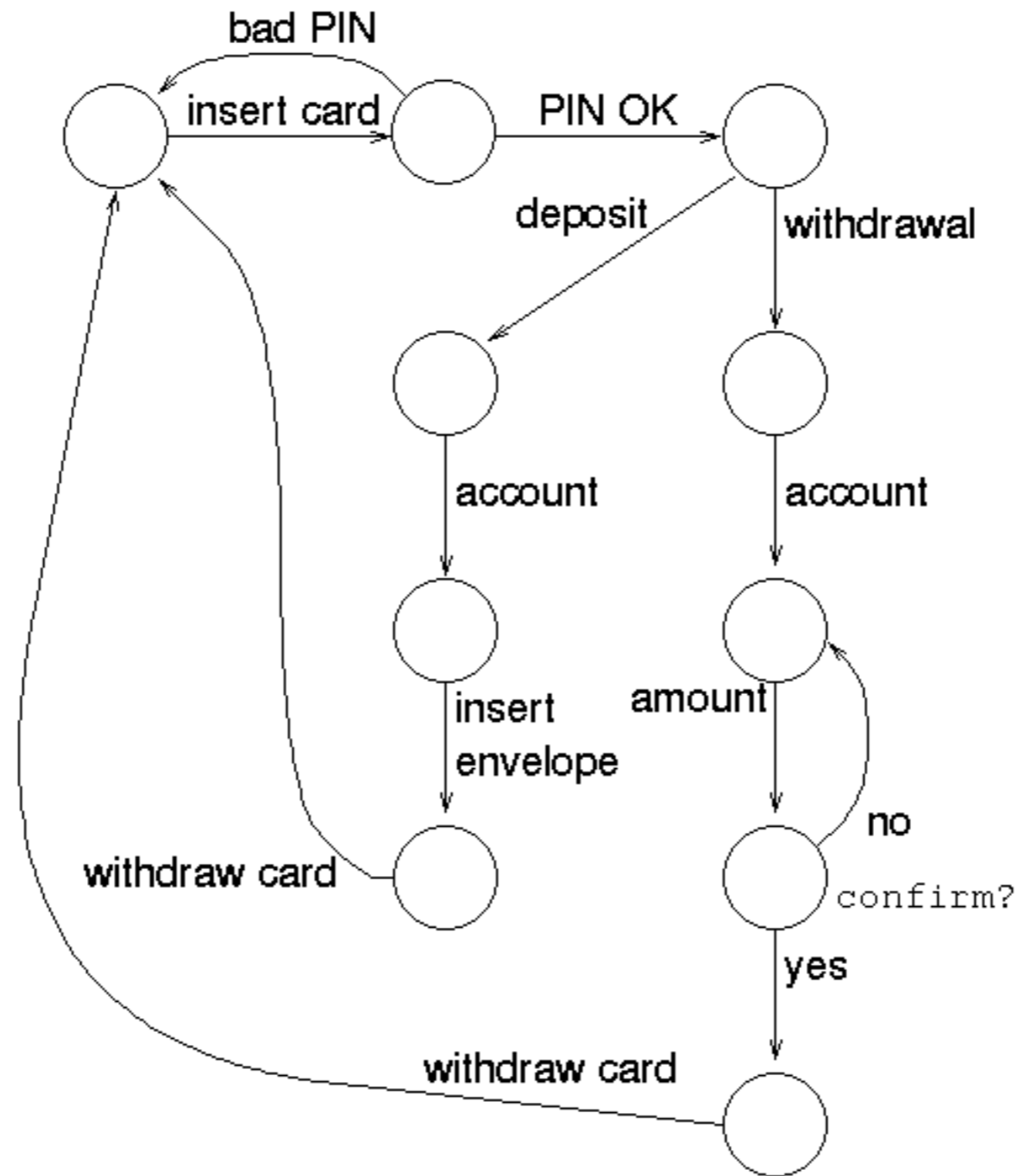


(\$1.25 per soda)

Example: Language Processing



Example: ATM



Computer controlled characters for games

States = characters behaviours

Transitions = events that cause a change in behaviour

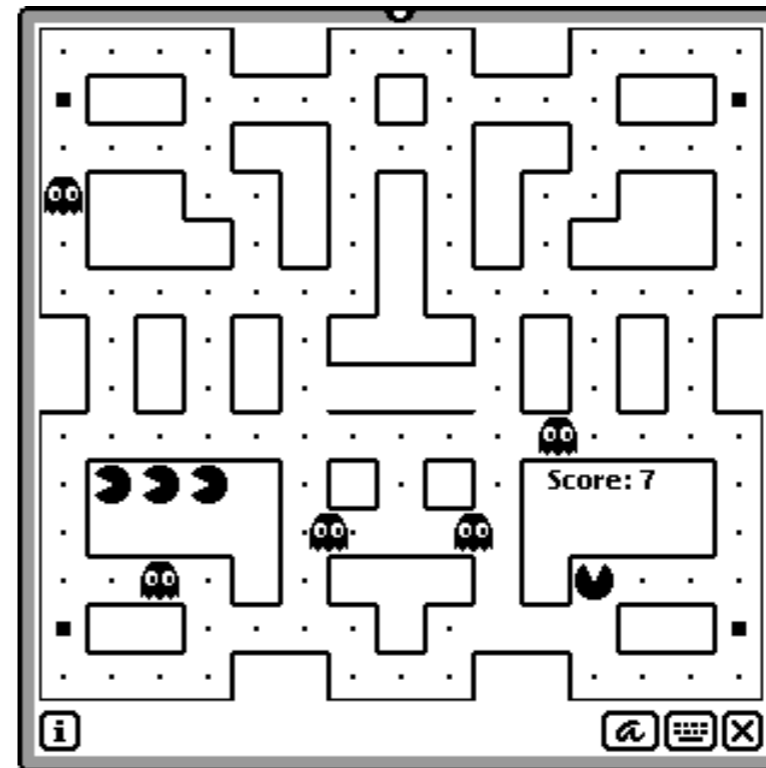
Example:

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



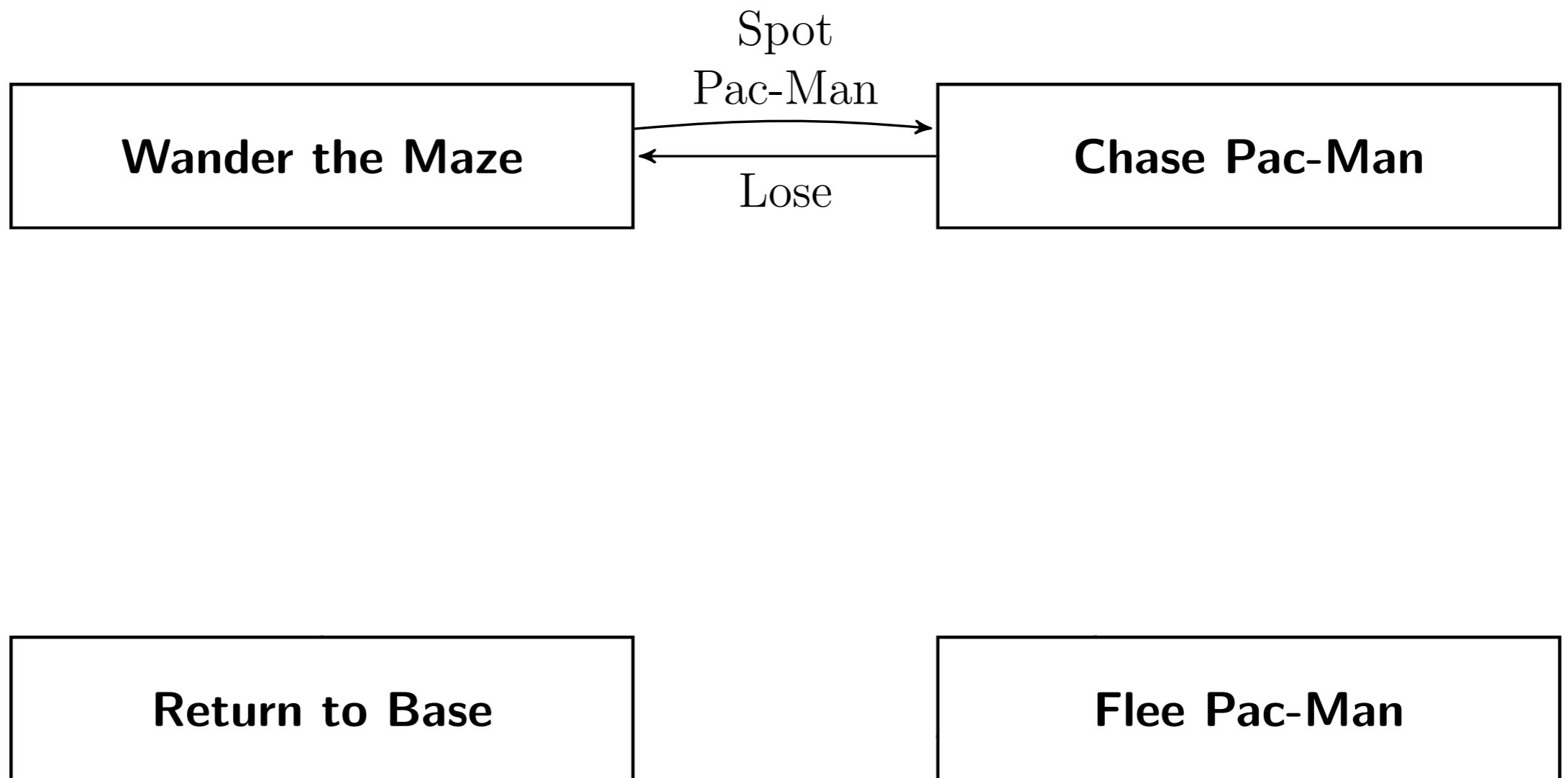
Wander the Maze

Chase Pac-Man

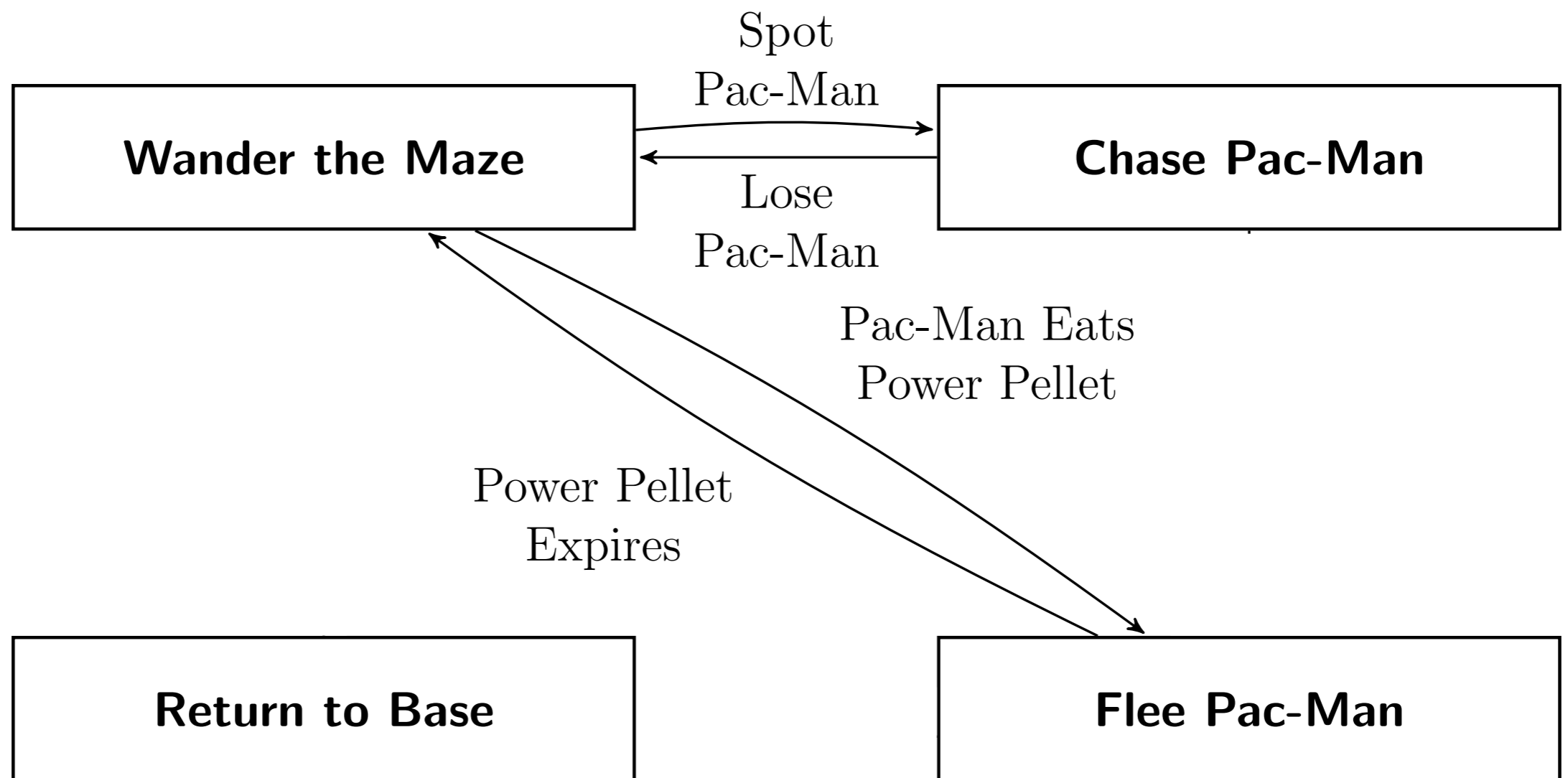
Return to Base

Flee Pac-Man

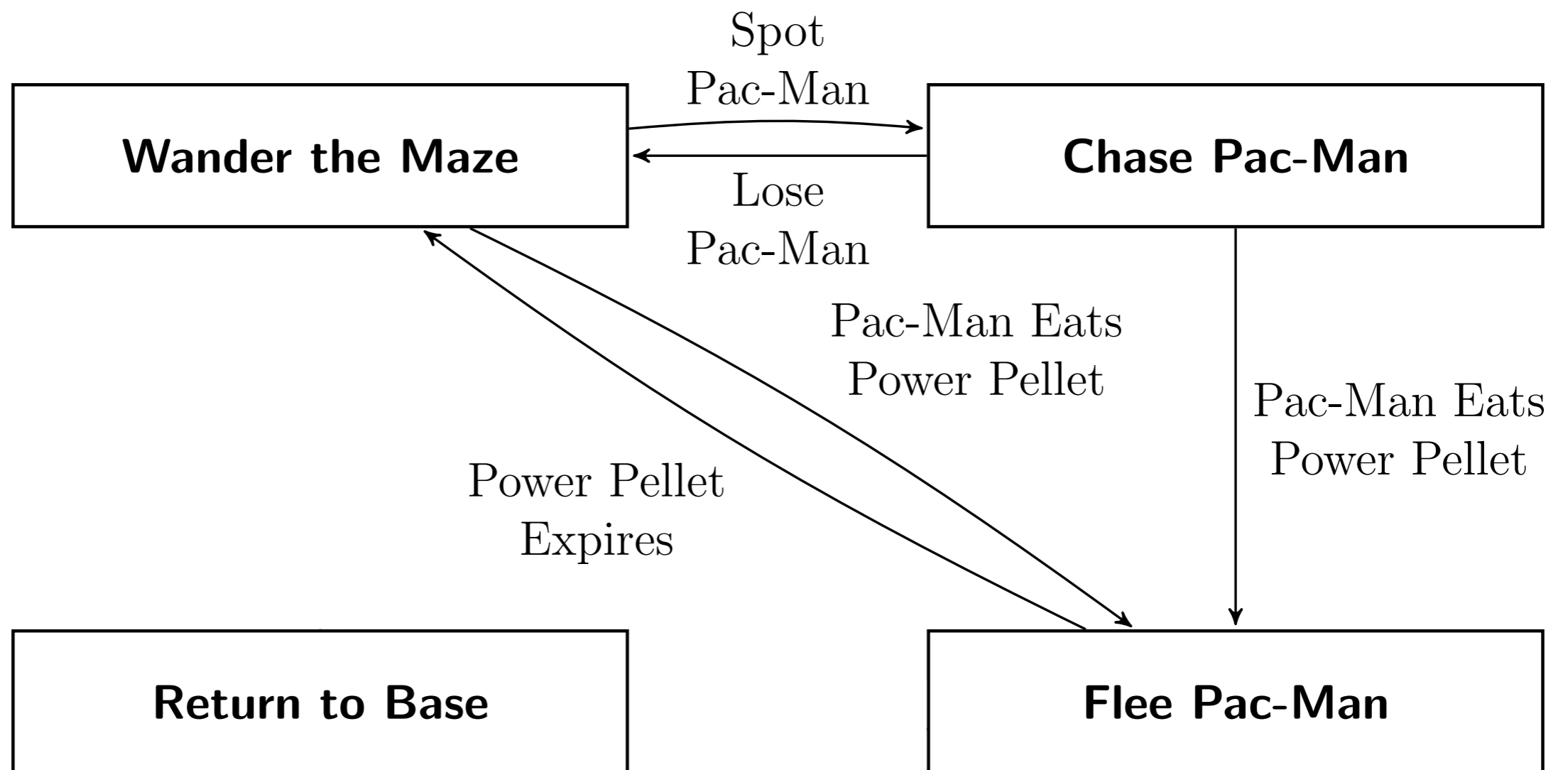
Example: Pac-Man Ghosts



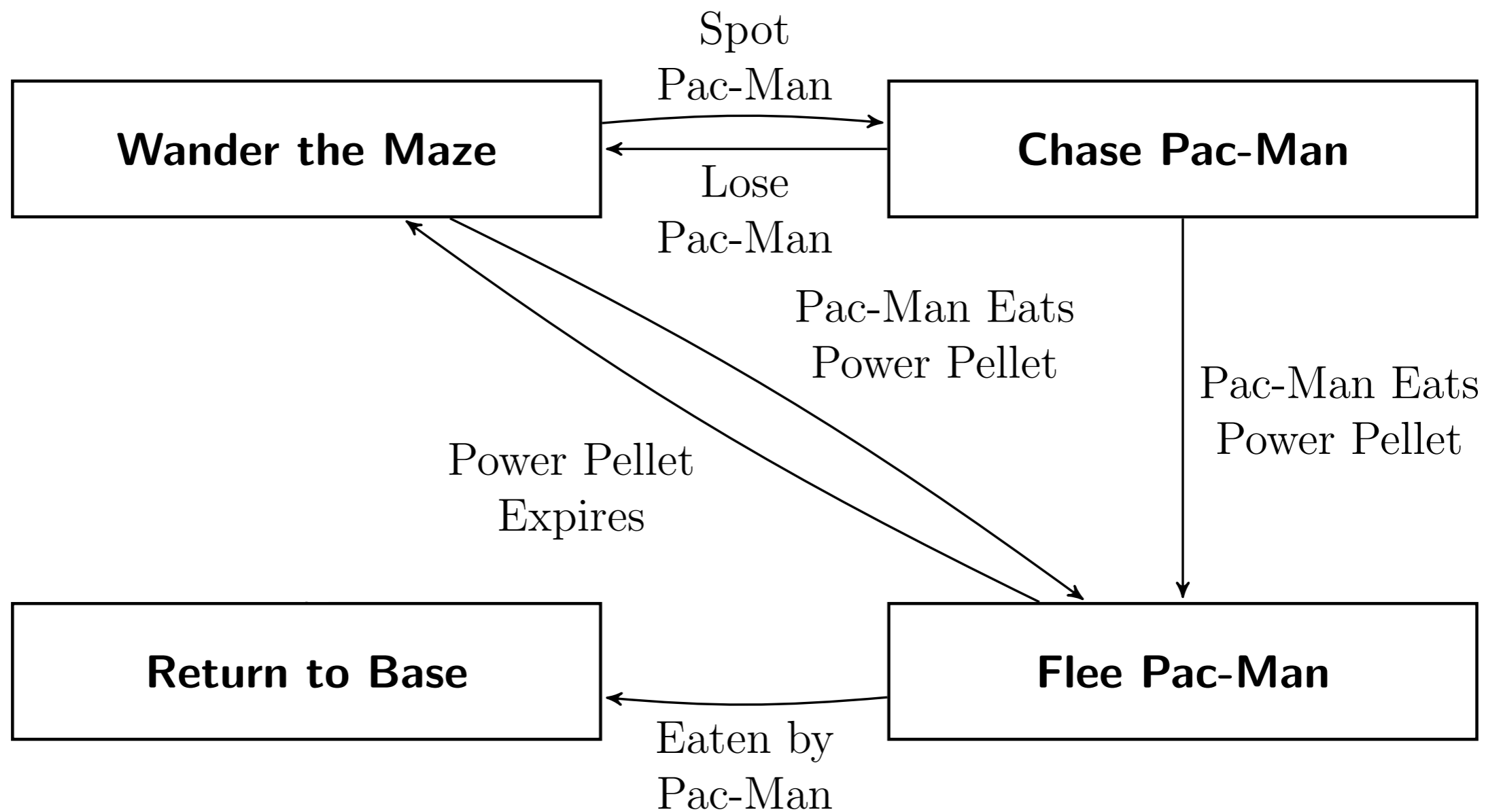
Example: Pac-Man Ghosts



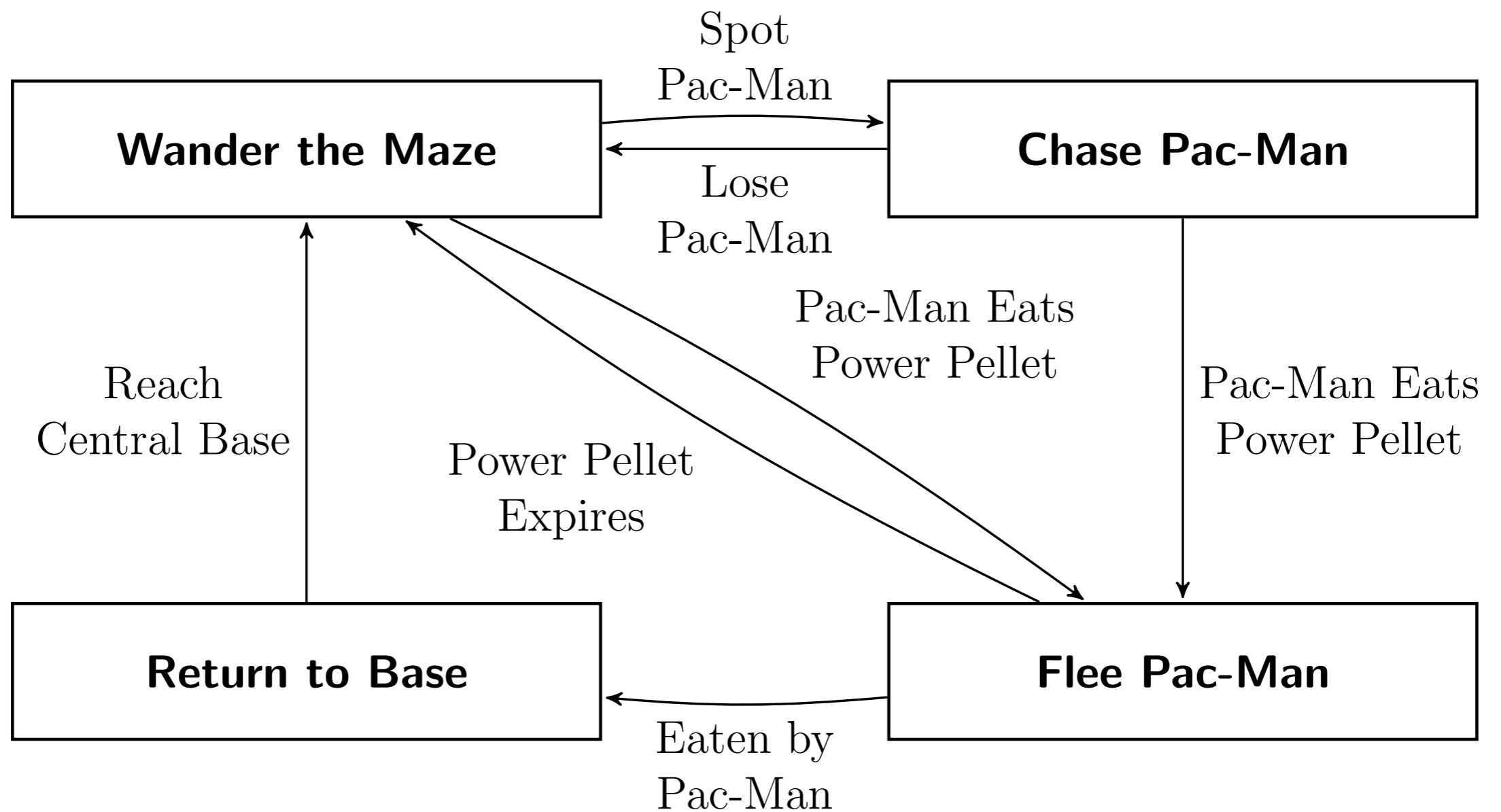
Example: Pac-Man Ghosts



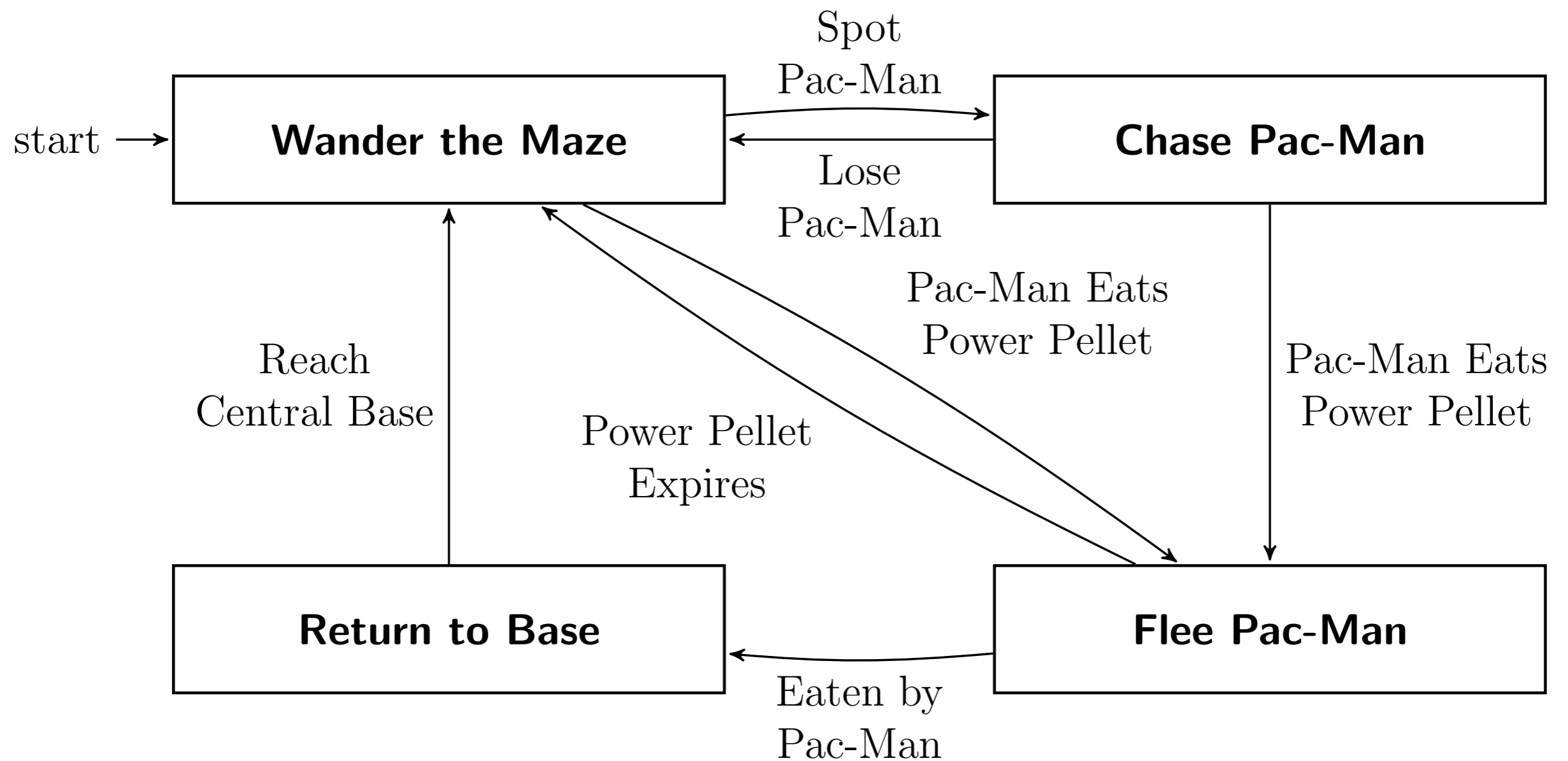
Example: Pac-Man Ghosts



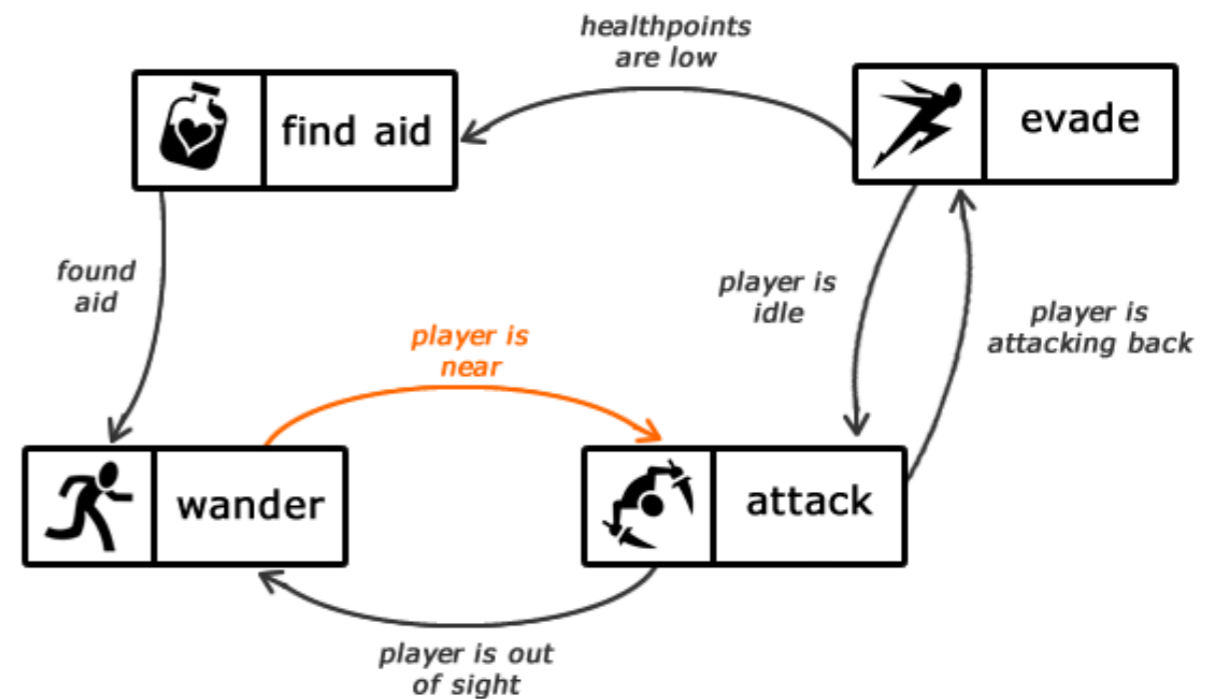
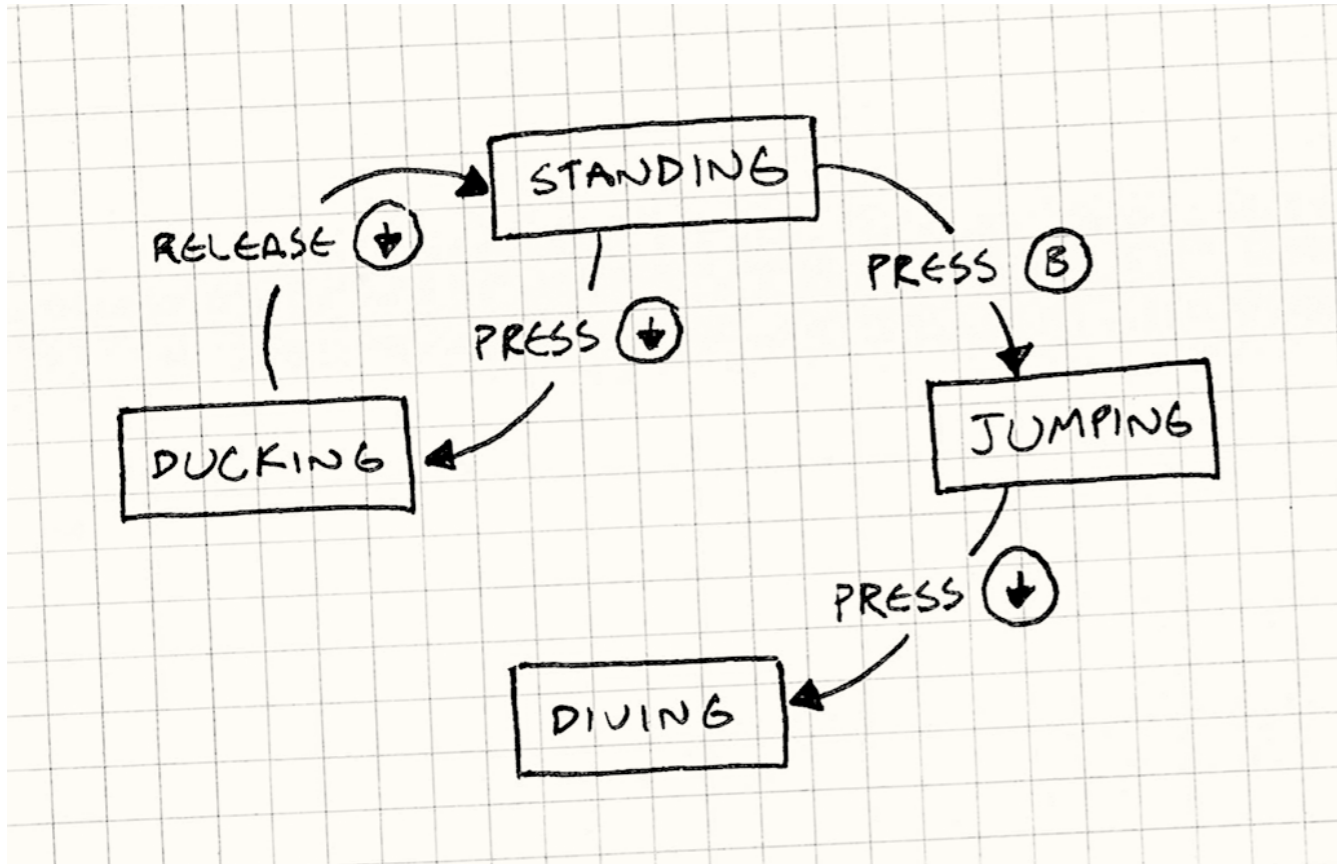
Example: Pac-Man Ghosts



Example: Pac-Man Ghosts



Other examples



Exercises

Choose your favourite (video) game, and draw the finite state automaton for one of the 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;
- $\delta : Q \times \Sigma \rightarrow Q$ 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)

Inductive definitions

A natural number is either:

- 0
- or the successor $n+1$ of a natural number n

A sequence over the alphabet A is either:

- the empty sequence ε
- or the juxtaposition wa of a sequence w with an element a of A

Inductively defined functions

Let us define the exponential function

base case: for any $k > 0$ we set
 $exp(k, 0) = 1$

inductive case: for any $k > 0, n \geq 0$ we set
 $exp(k, n+1) = exp(k, n) \times k$

Inductively defined functions

Let us define the exponential function

base case: for any $k > 0$ we set
 $exp(k, 0) = 1$

inductive case: for any $k > 0, n \geq 0$ we set
 $exp(k, n+1) = exp(k, n) \times k$

Recursive definition

Inductively defined functions

Let us define the exponential function

base case: for any $k > 0$ we set
 $exp(k, 0) = 1$

inductive case: for any $k > 0, n \geq 0$ we set
 $exp(k, n+1) = exp(k, n) \times k$

More complex
case

Simpler
case

Kleene-star notation A^*

Given a set A we denote by A^*

the set of finite sequences of elements in A , i.e.:

$$A^* = \{ a_1 \cdots a_n \mid n \geq 0 \wedge a_1, \dots, a_n \in A \}$$

We denote the empty sequence by $\epsilon \in A^*$

For example:

$$A = \{ a, b \} \quad A^* = \{ \epsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots \}$$

Extended transition function (destination function)

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

base case: For any $q \in Q$ we let

$$\hat{\delta}(q, \epsilon) = q$$

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

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

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

Extended transition function (destination function)

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

base case: For any $q \in Q$ we let

$$\hat{\delta}(q, \epsilon) = q$$

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

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

Recursive definition

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

Extended transition function (destination function)

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

base case: For any $q \in Q$ we let

$$\hat{\delta}(q, \epsilon) = q$$

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

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

More complex case **Simpler case**

$(\hat{\delta}(q, w))$ 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

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

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

$$L(A) = \{ w \mid \hat{\delta}(q_0, w) \in F \}$$

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

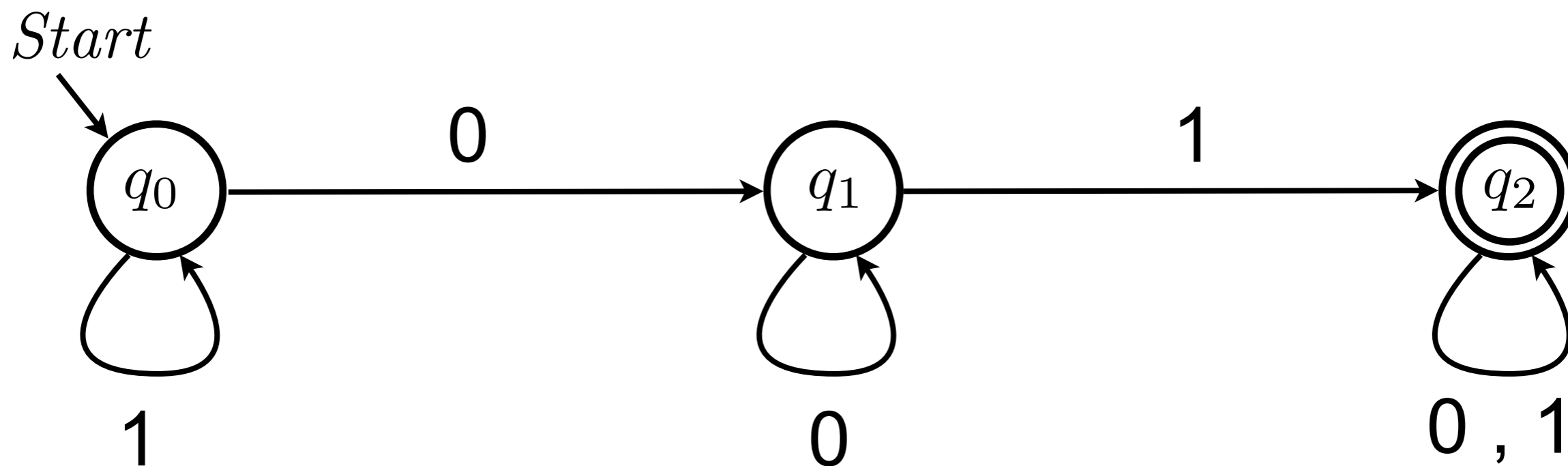
A DFA accepts a string w , if there is a path in its transition diagram such that:

it starts from the initial state

it ends in one final state

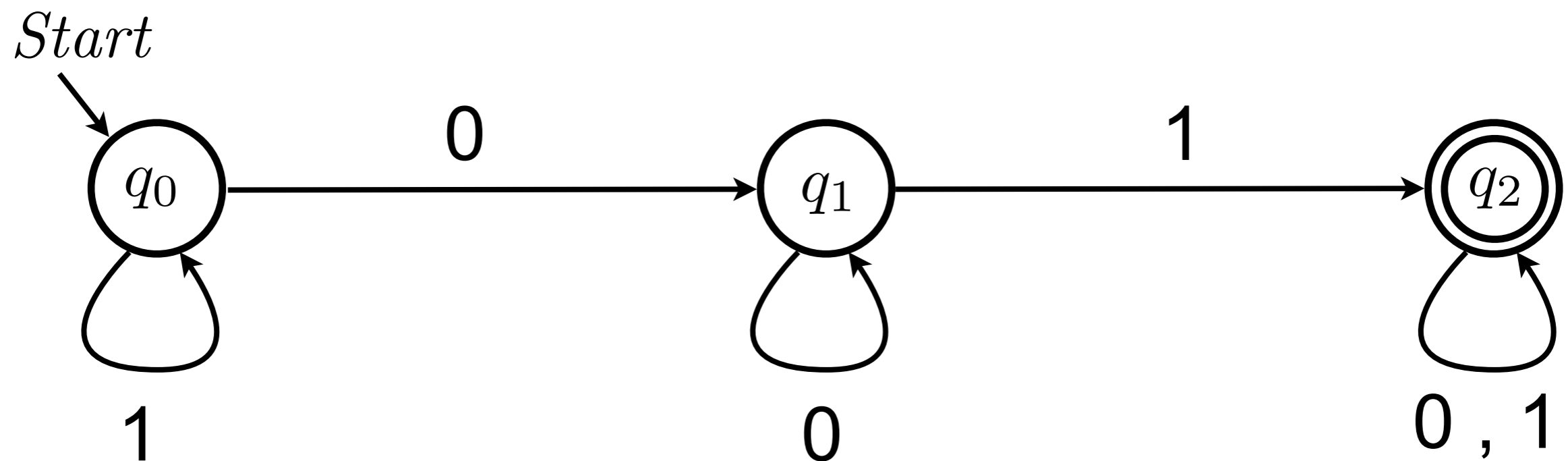
the sequence of labels in the path is exactly w

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



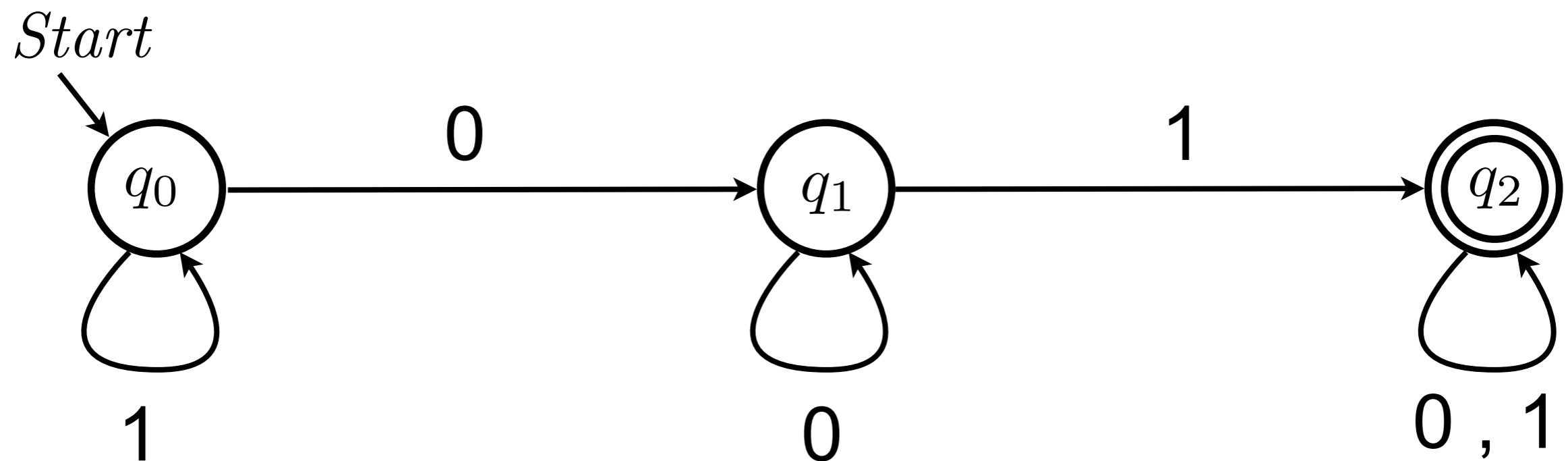
Does it accept 100 ?

Does it accept 011 ?

Does it accept 1010010 ?

What is $L(A)$?

DFA: question time



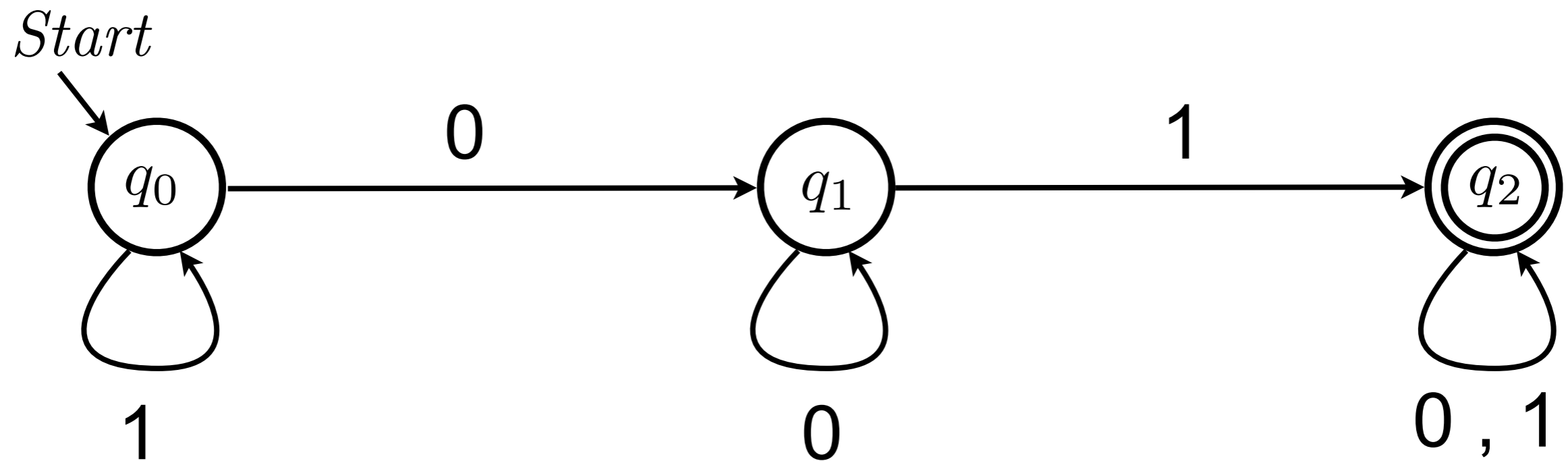
Does it accept 100 ? **NO**

Does it accept 011 ?

Does it accept 1010010 ?

What is $L(A)$?

DFA: question time



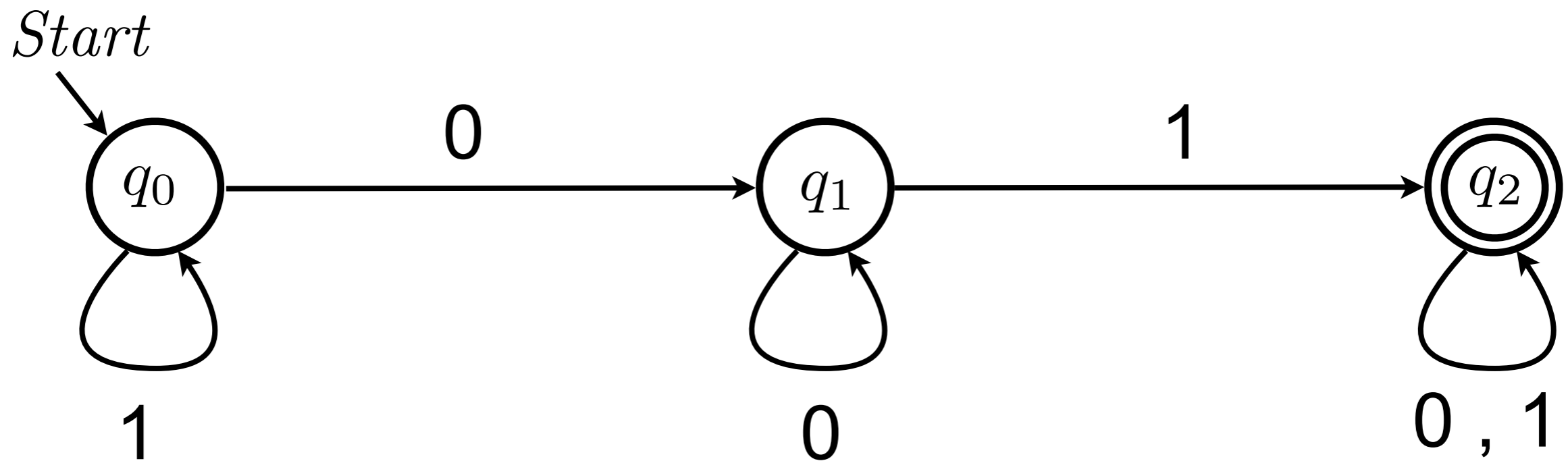
Does it accept 100 ? **NO**

Does it accept 011 ? **YES**

Does it accept 1010010 ?

What is $L(A)$?

DFA: question time



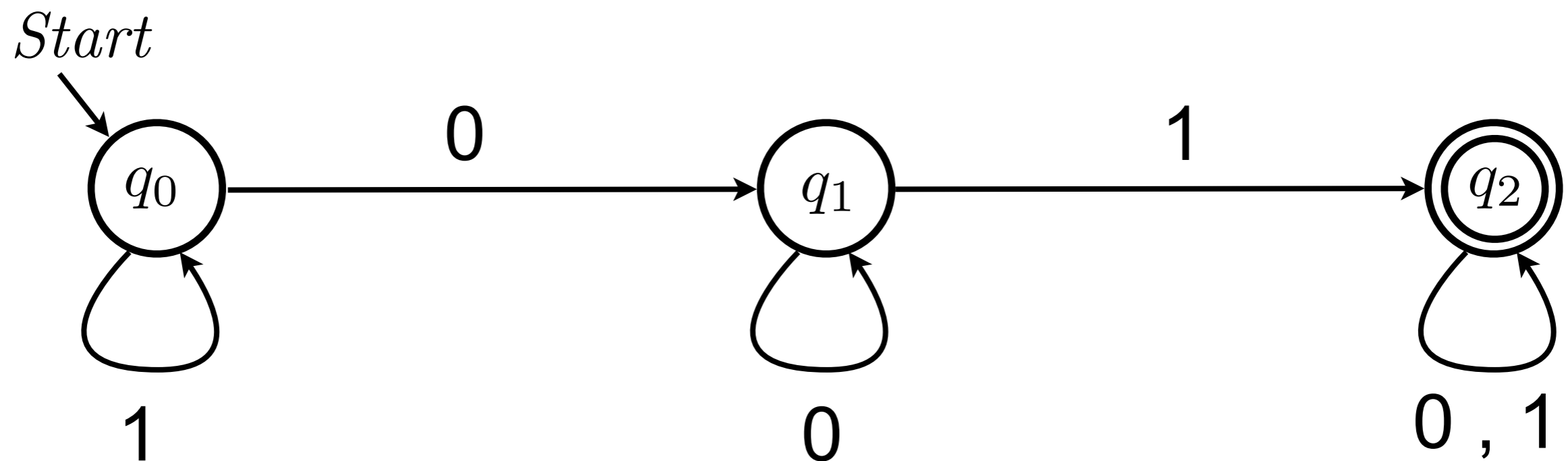
Does it accept 100 ? **NO**

Does it accept 011 ? **YES**

Does it accept 1010010 ? **YES**

What is $L(A)$?

DFA: question time



Does it accept 100 ? **NO**

Does it accept 011 ? **YES**

Does it accept 1010010 ? **YES**

What is $L(A)$? $\{ x01y \mid x, y \in \{0, 1\}^* \}$

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

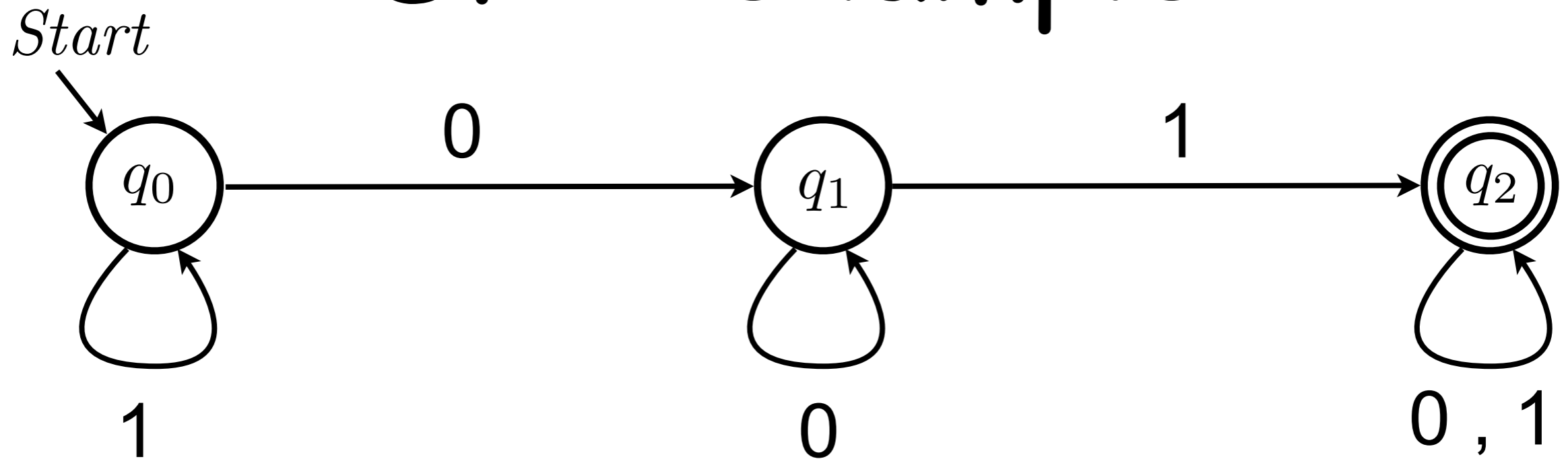
start state decorated with an arrow

all final states decorated with *

Transition table

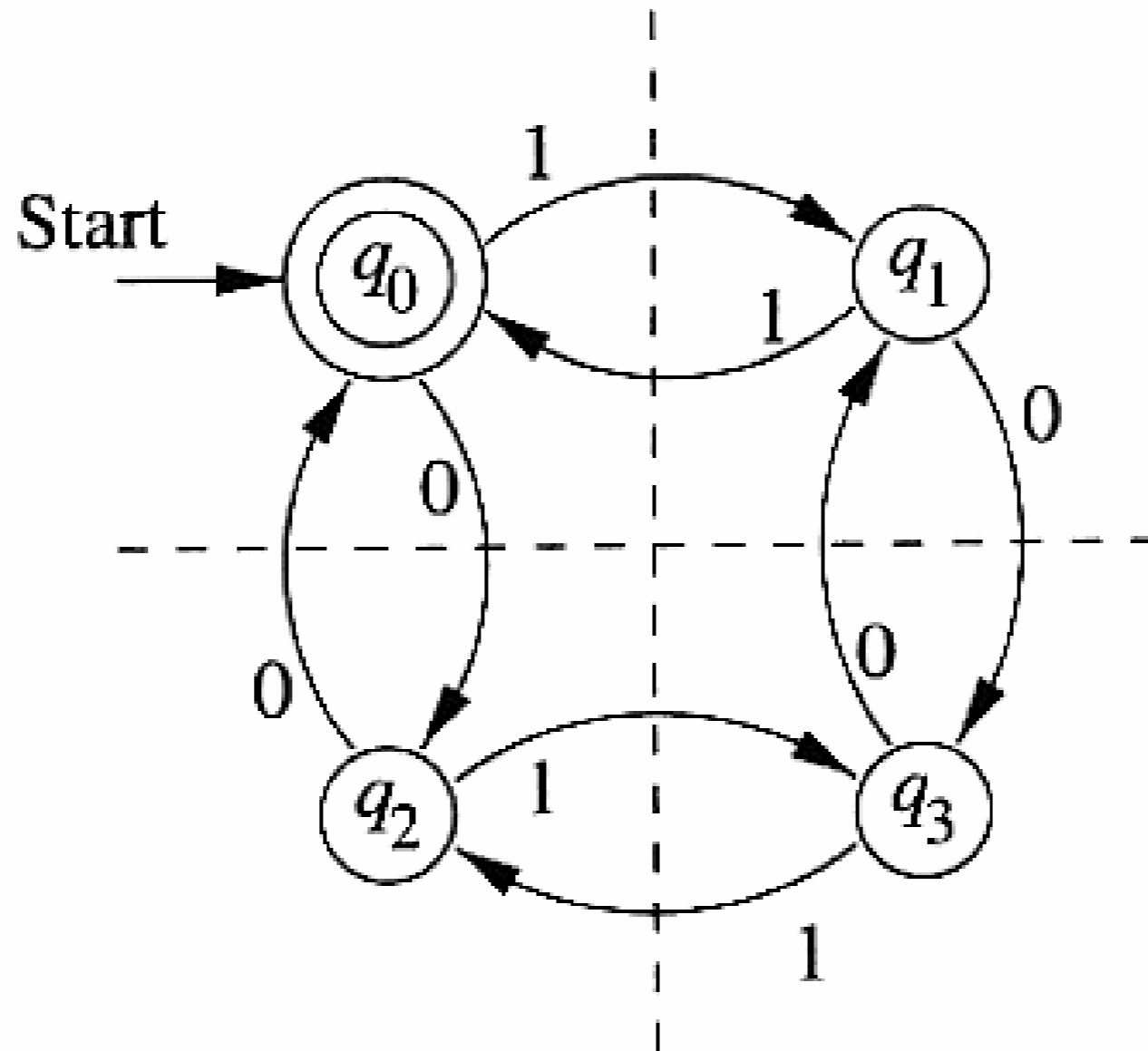
				a			
→							
q				$\delta(q, a)$			
*							
*							

DFA: example



	0	1
$\rightarrow q_0$		
q_1		
$\star q_2$		

DFA: exercise



Does it accept 100 ?
Write its transition table.

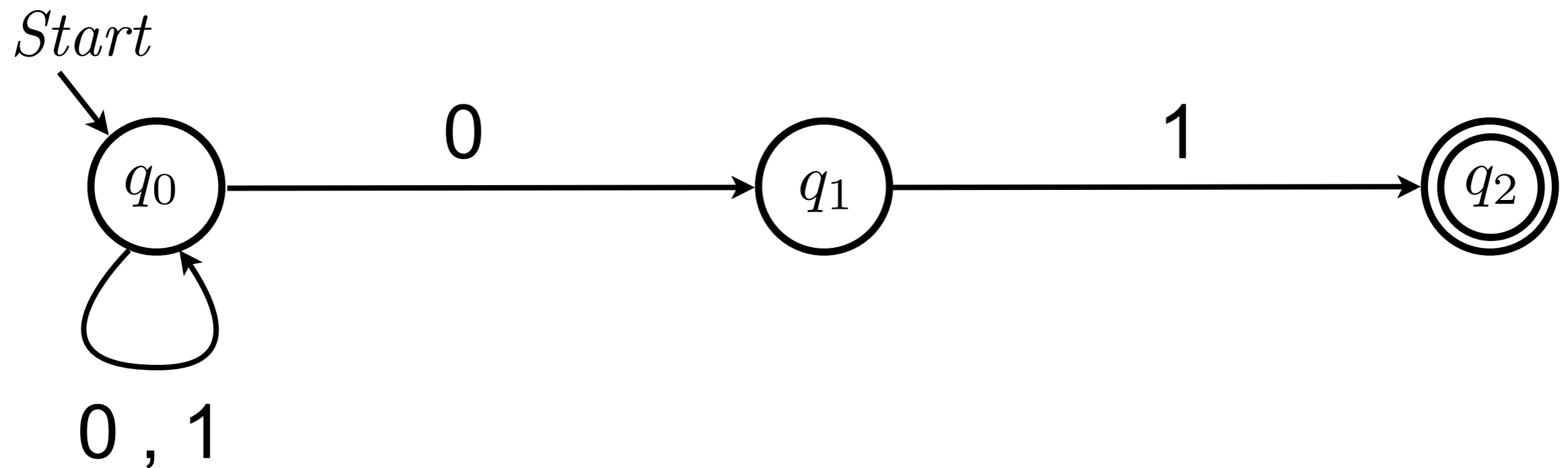
Does it accept 1010 ?
What is $L(A)$?

NFA

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

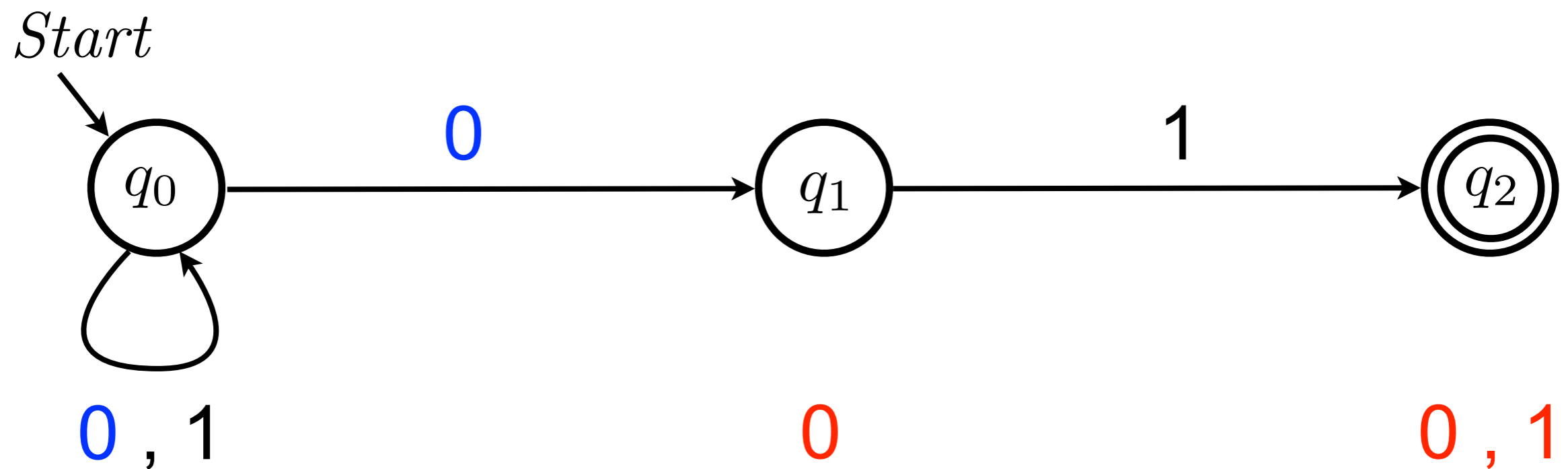
- Q is a finite set of states;
- Σ is a finite set of input symbols;
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function;
 powerset of Q = set of sets over Q
- $q_0 \in Q$ is the initial state (also called start state);
- $F \subseteq Q$ is the set of final states (also accepting states)

NFA: example



Can you explain why it is not a DFA?

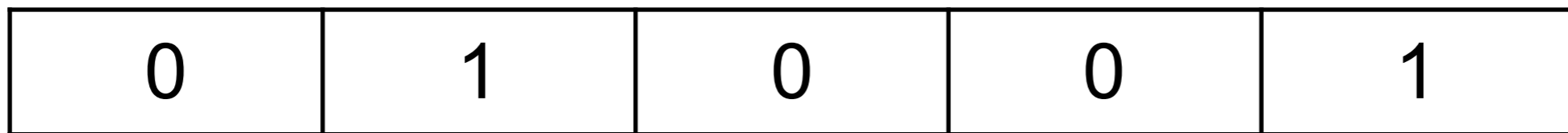
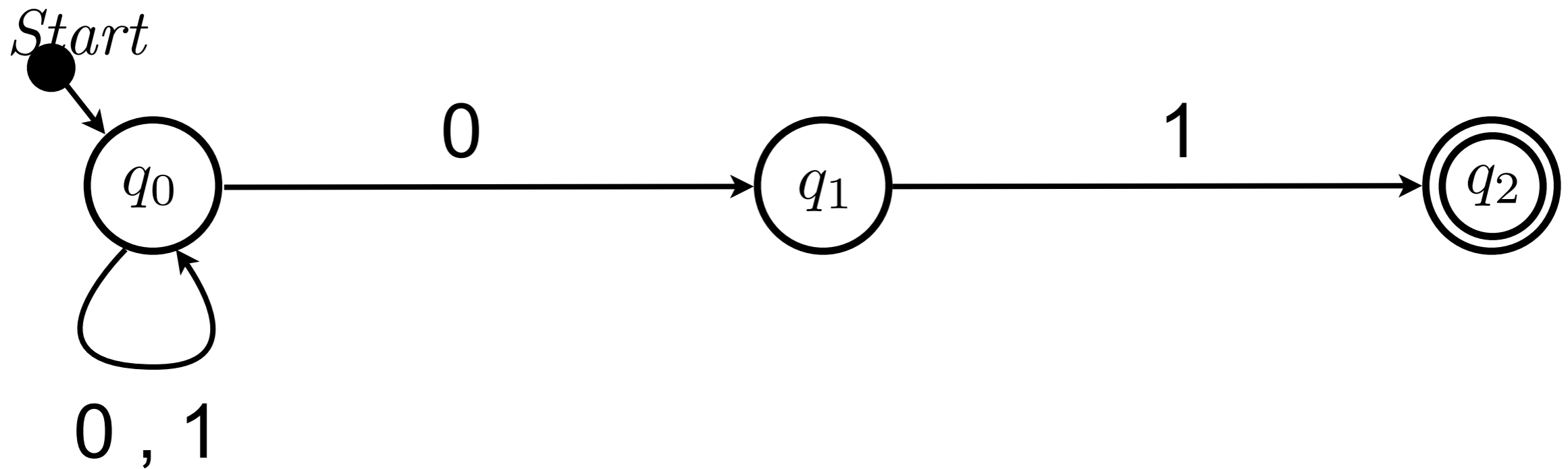
NFA: example



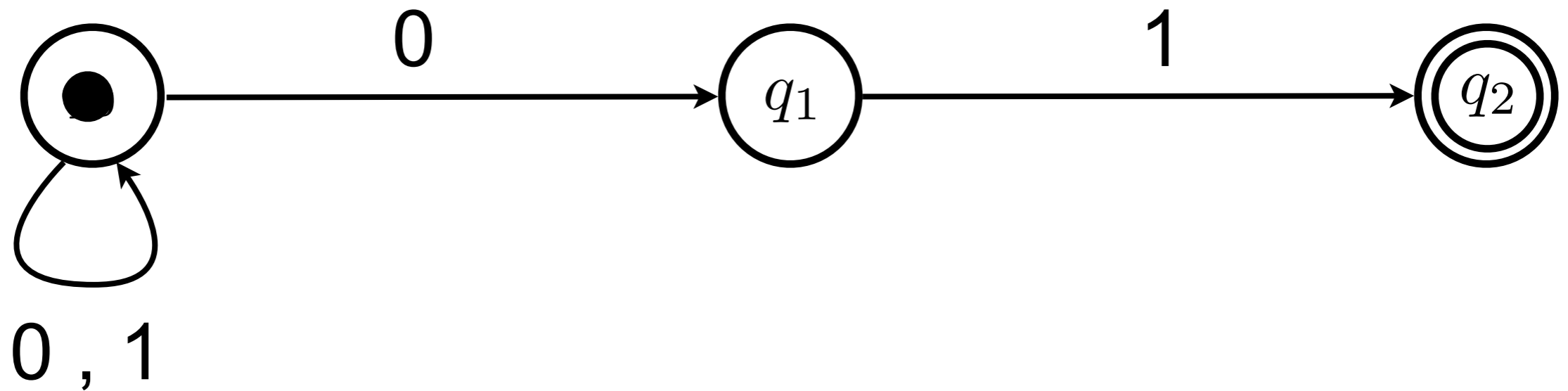
Can you explain why it is not a DFA?

Reshaping

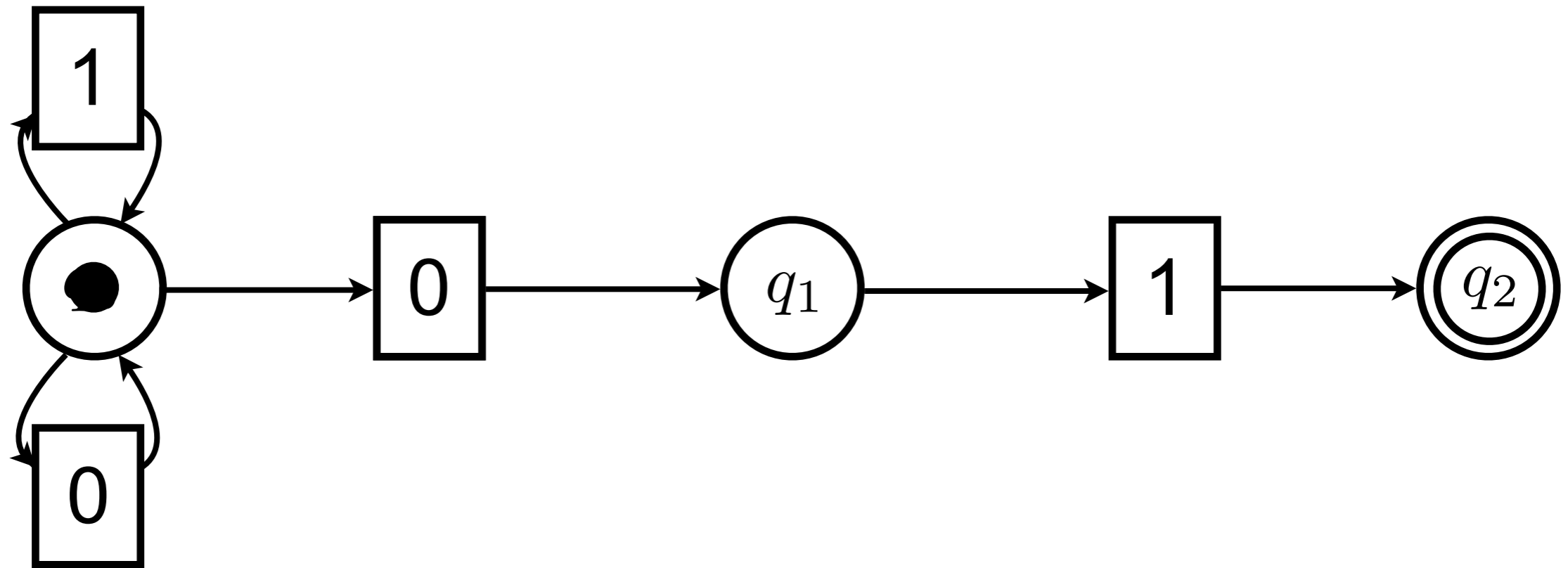
Step 1: get a token



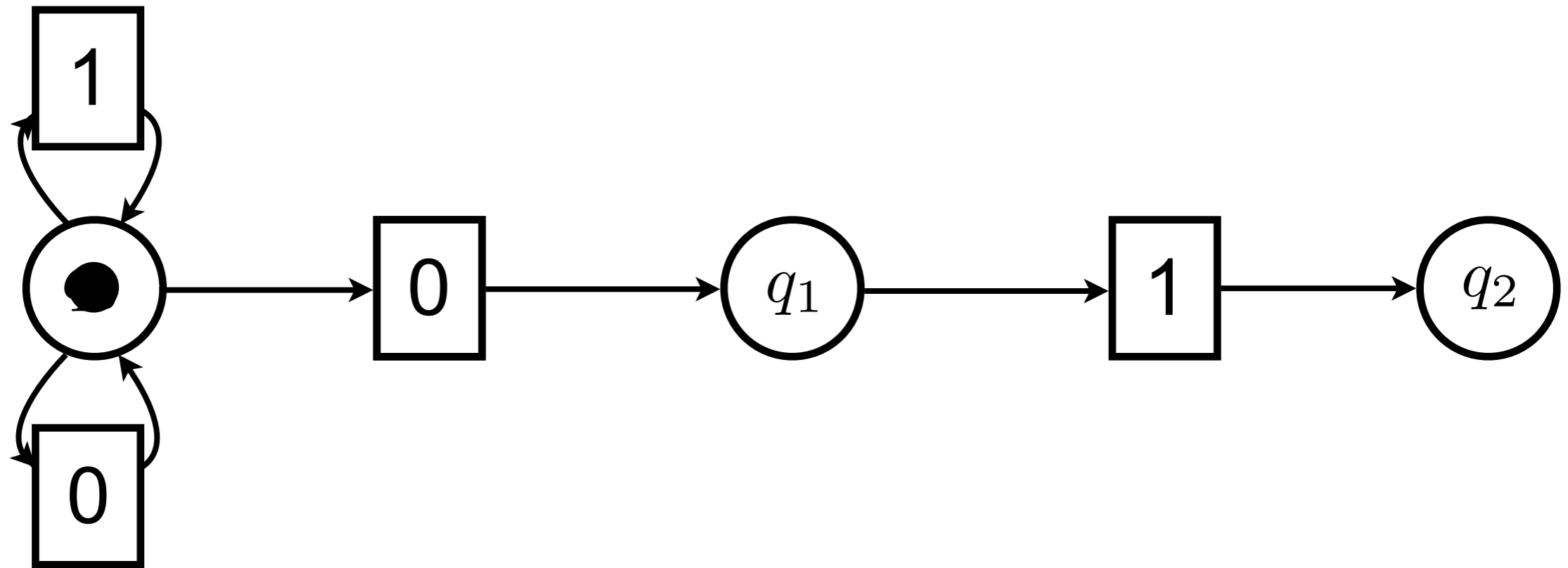
Step 2: forget initial state decoration



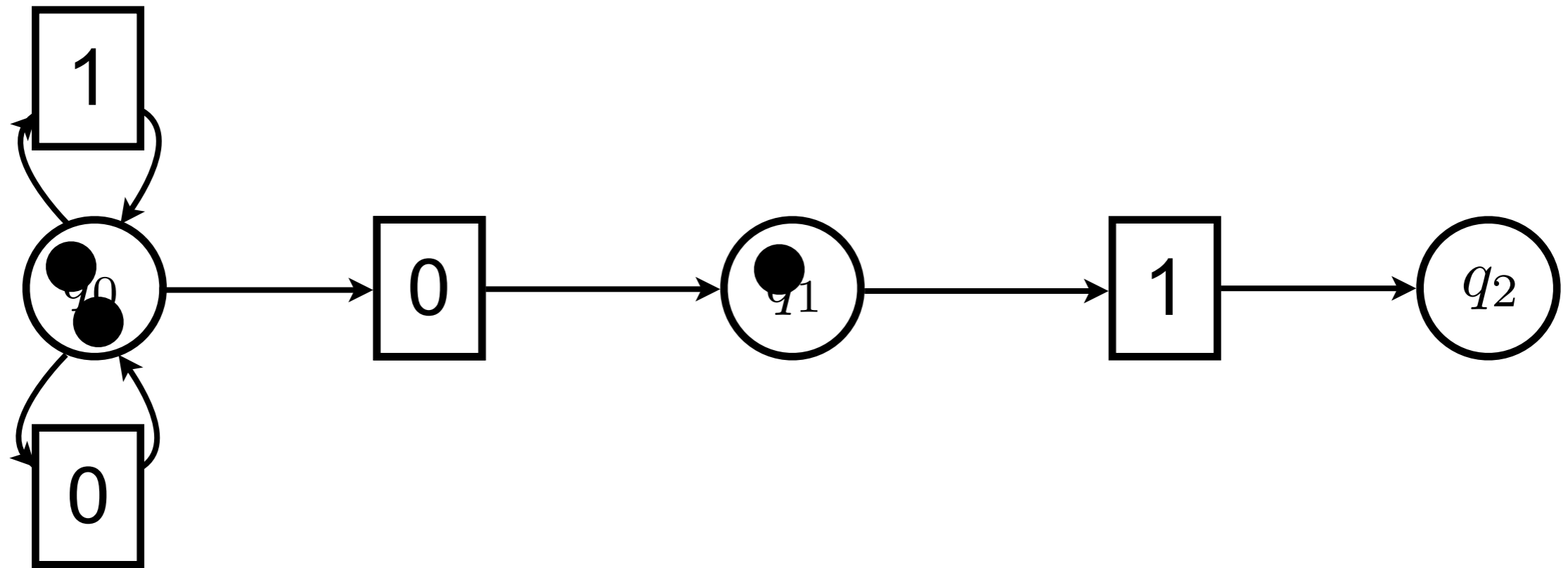
Step 3: transitions as boxes



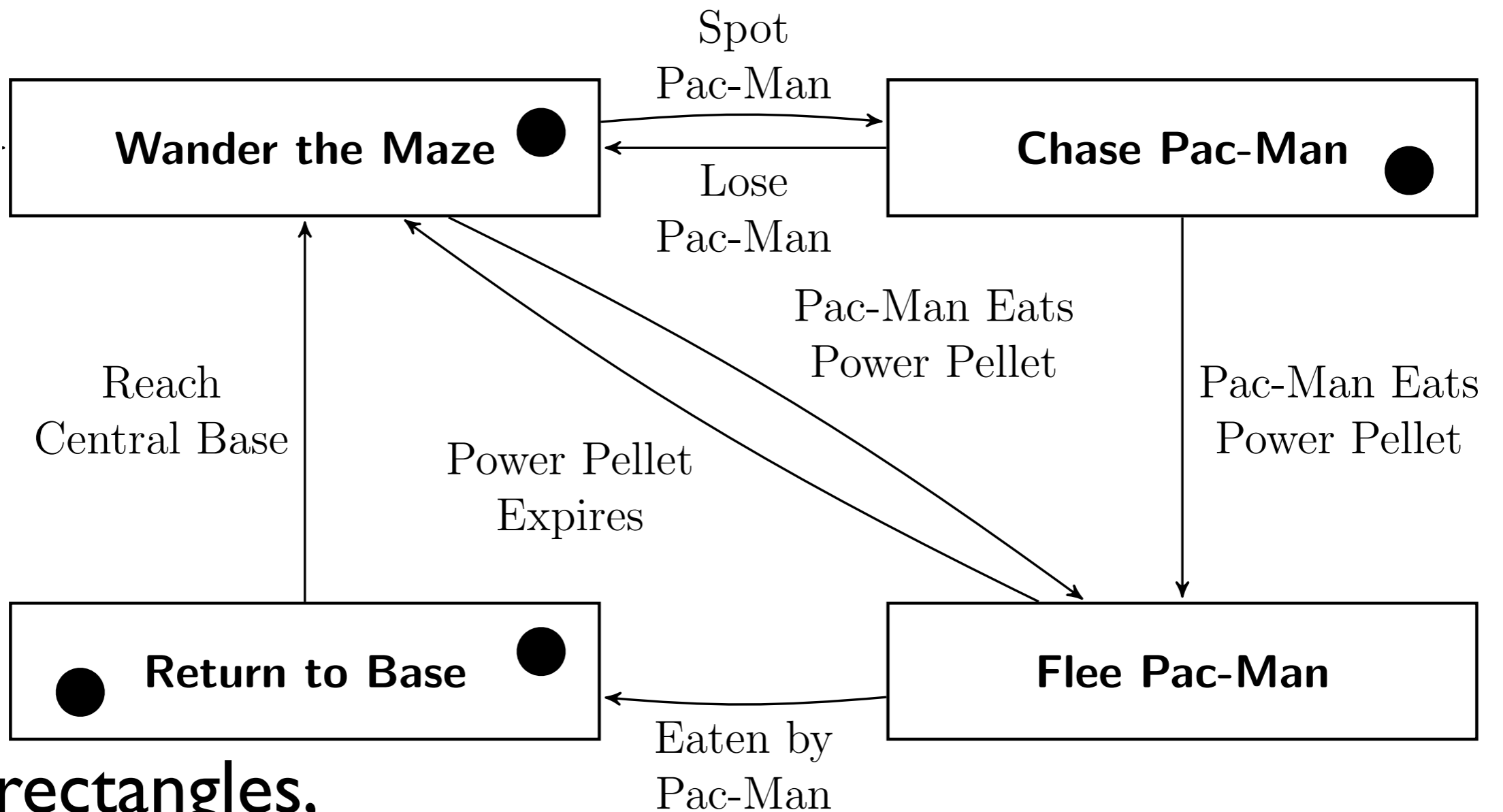
Step 4: forget final states



Step 5: allow for more tokens

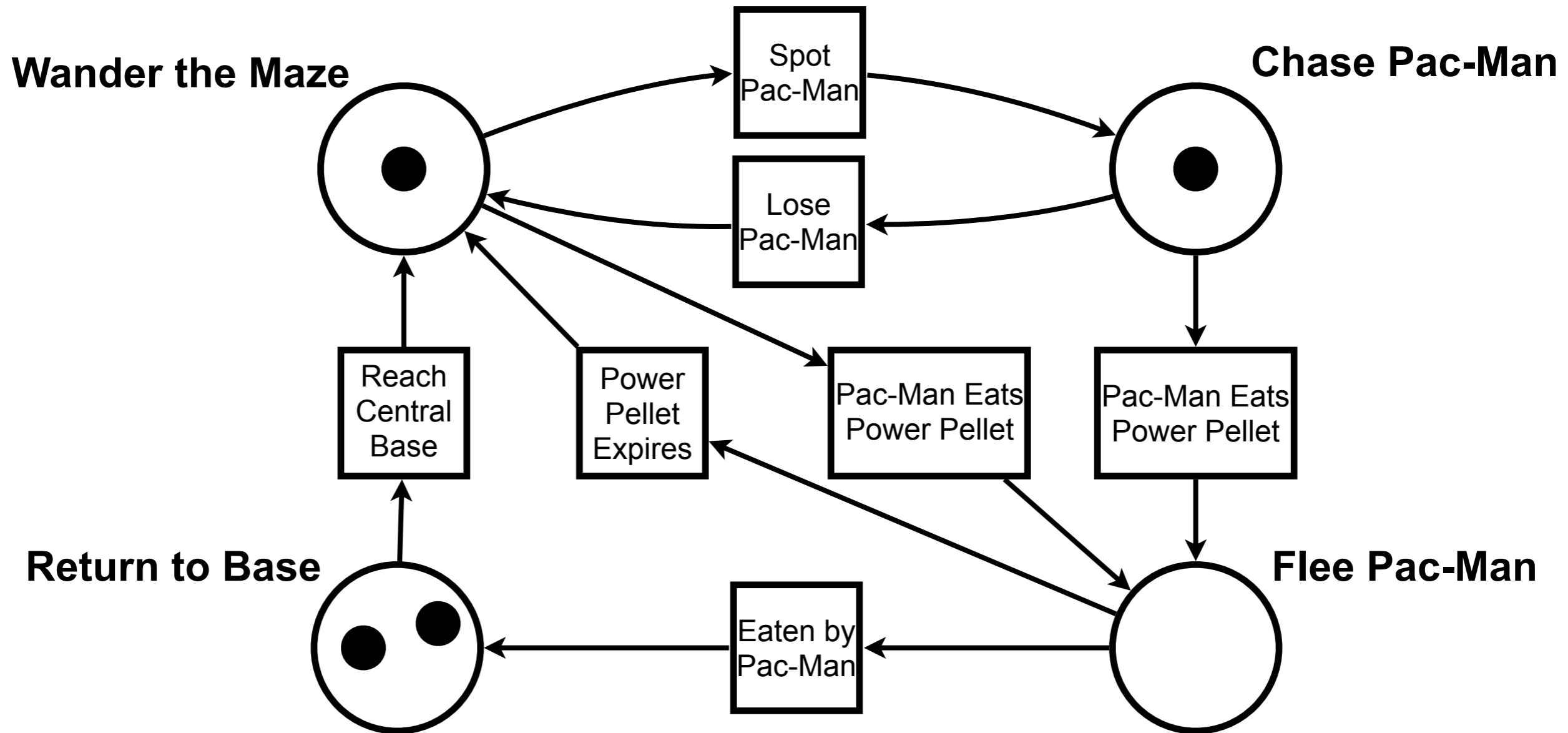


Example: Four Pac-Man Ghosts

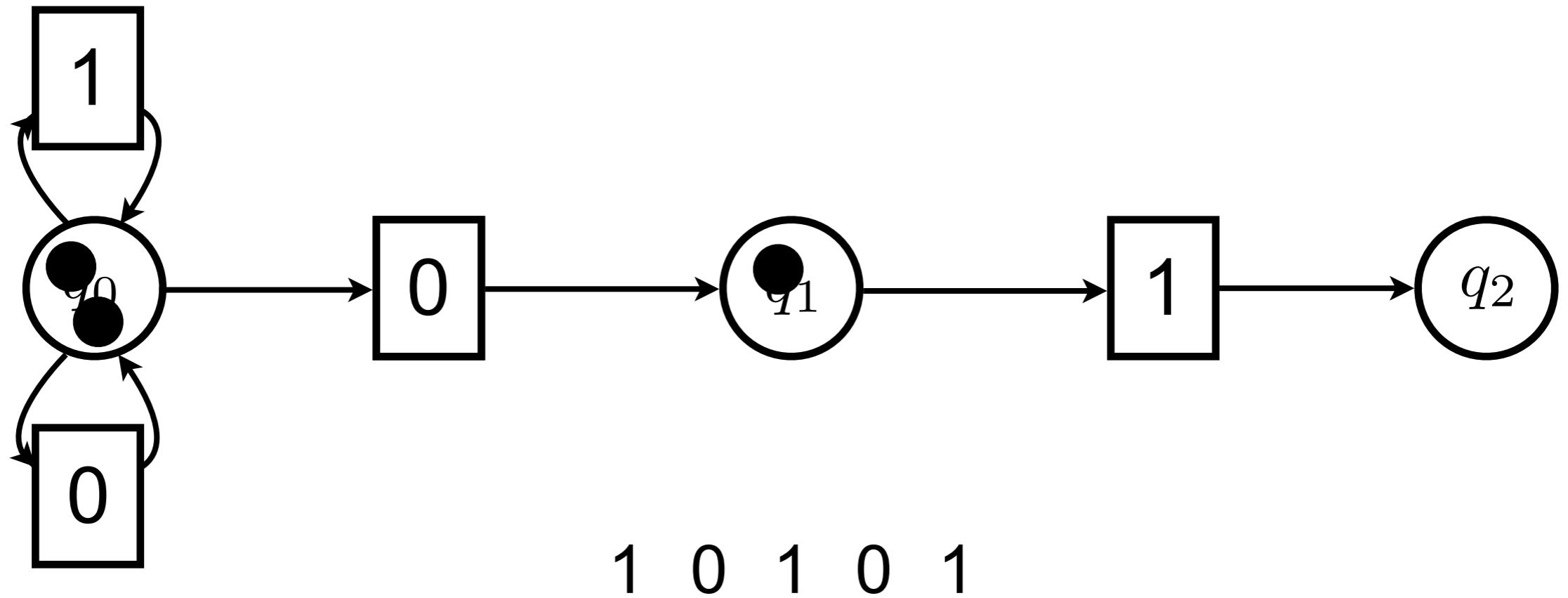


look as rectangles,
but are circles :-)

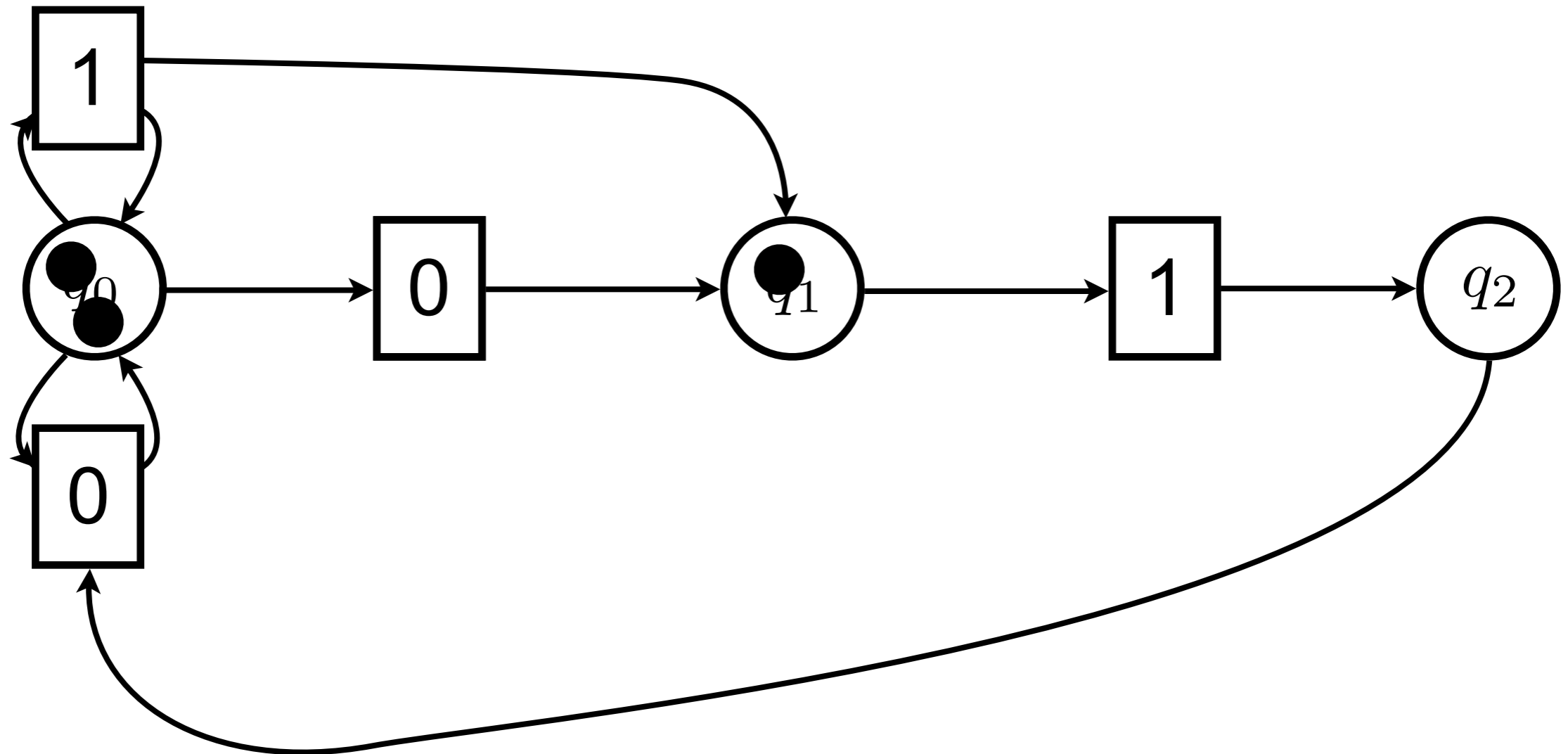
Example: Four Pac-Man Ghosts



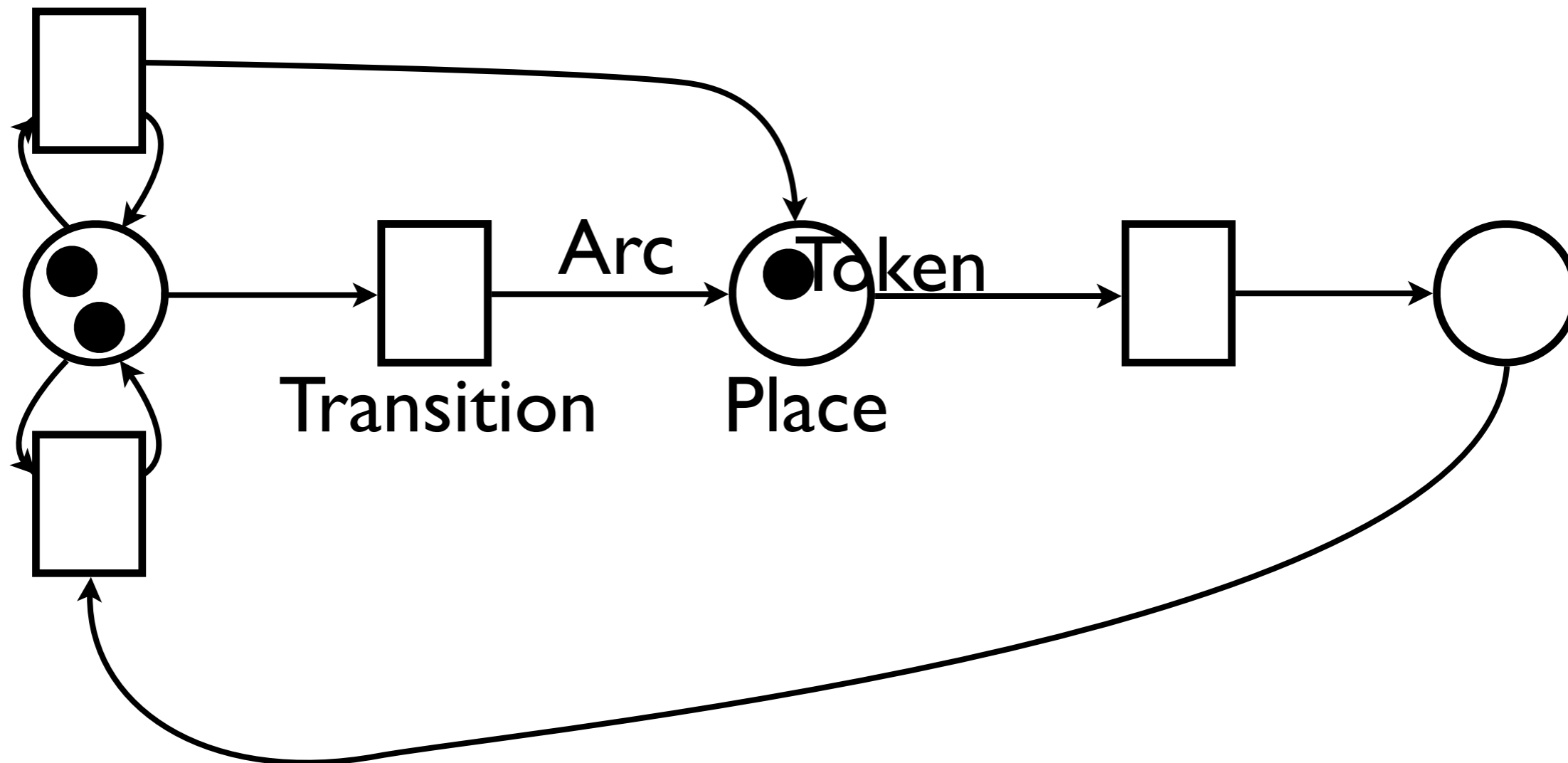
Example: token game



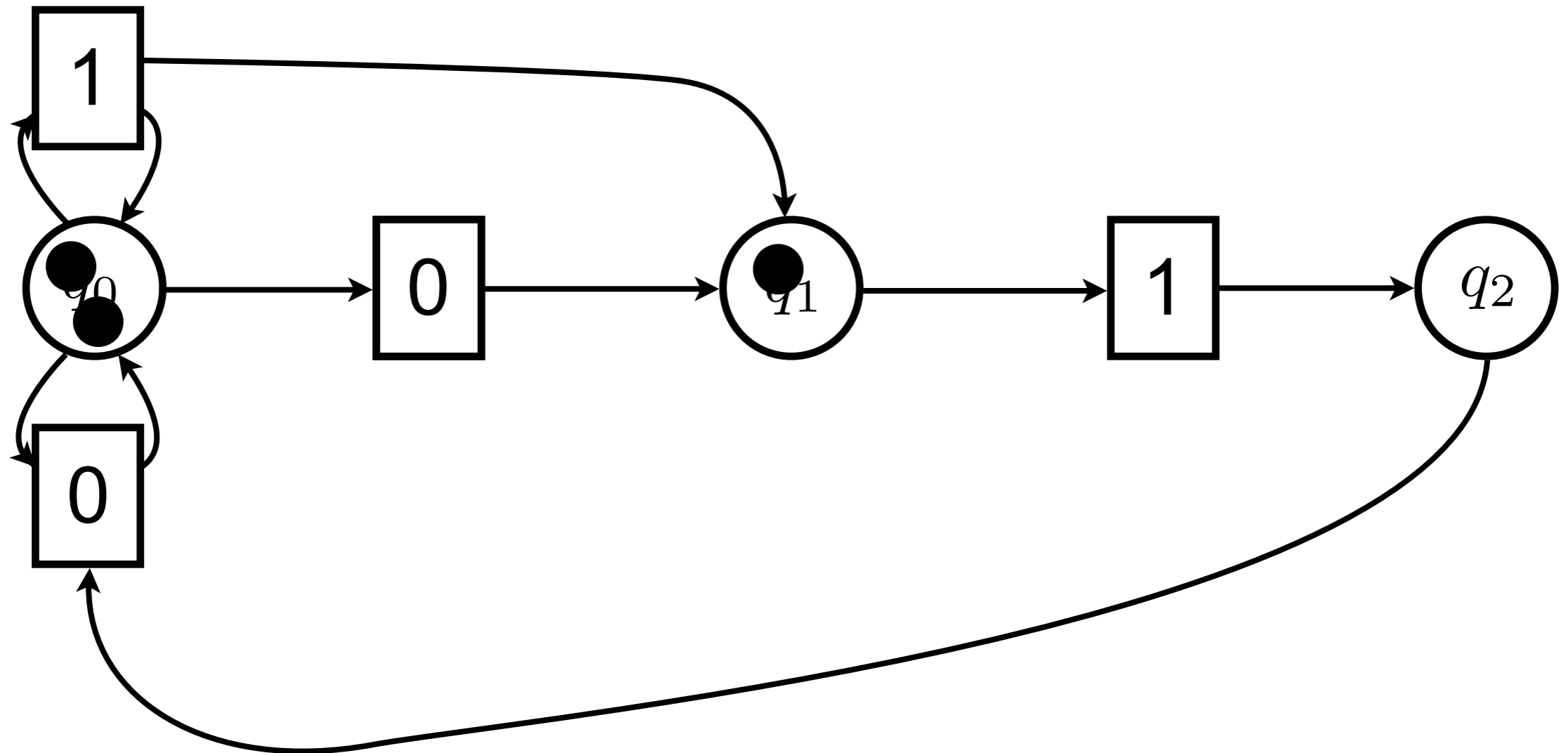
Step 6: allow for more arcs



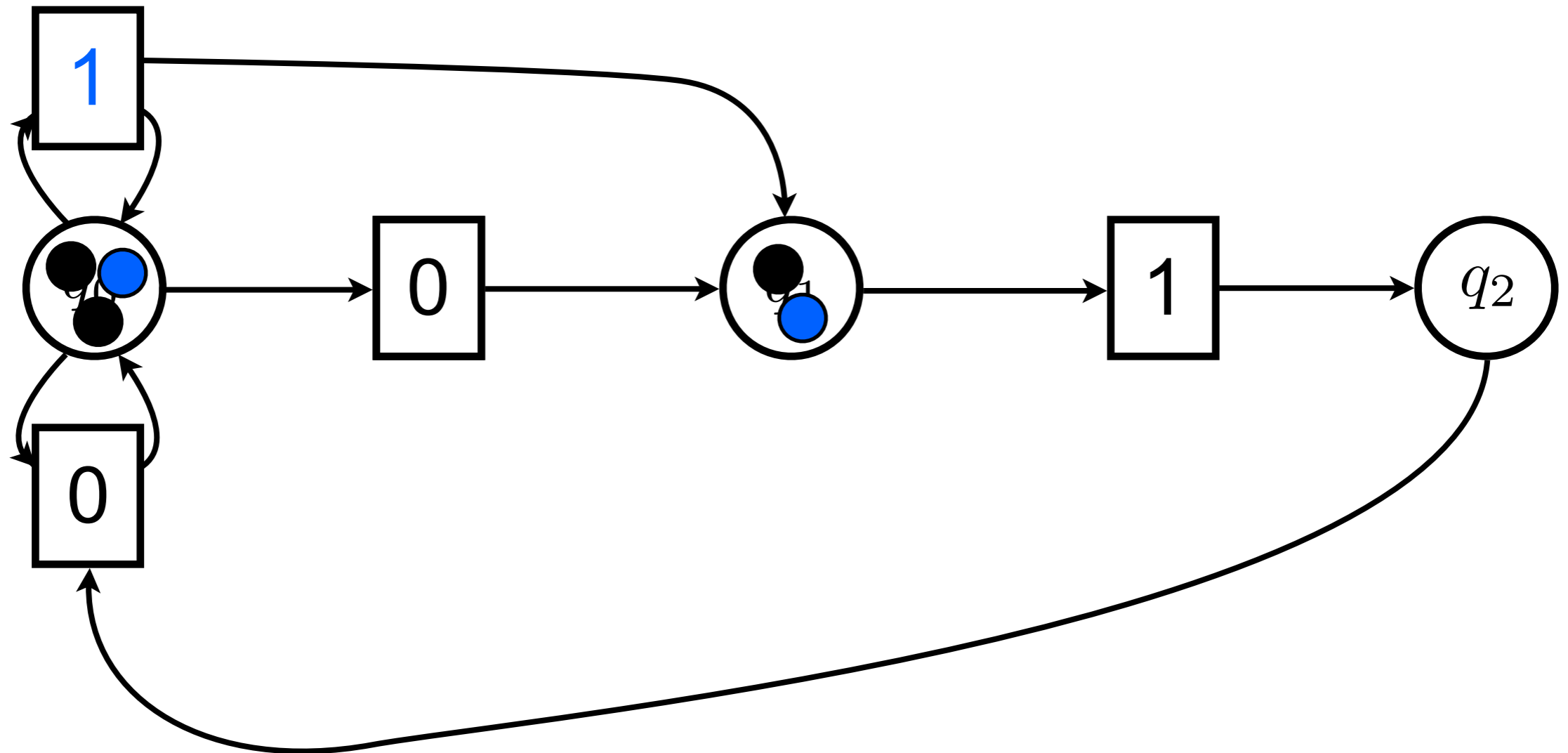
Terminology



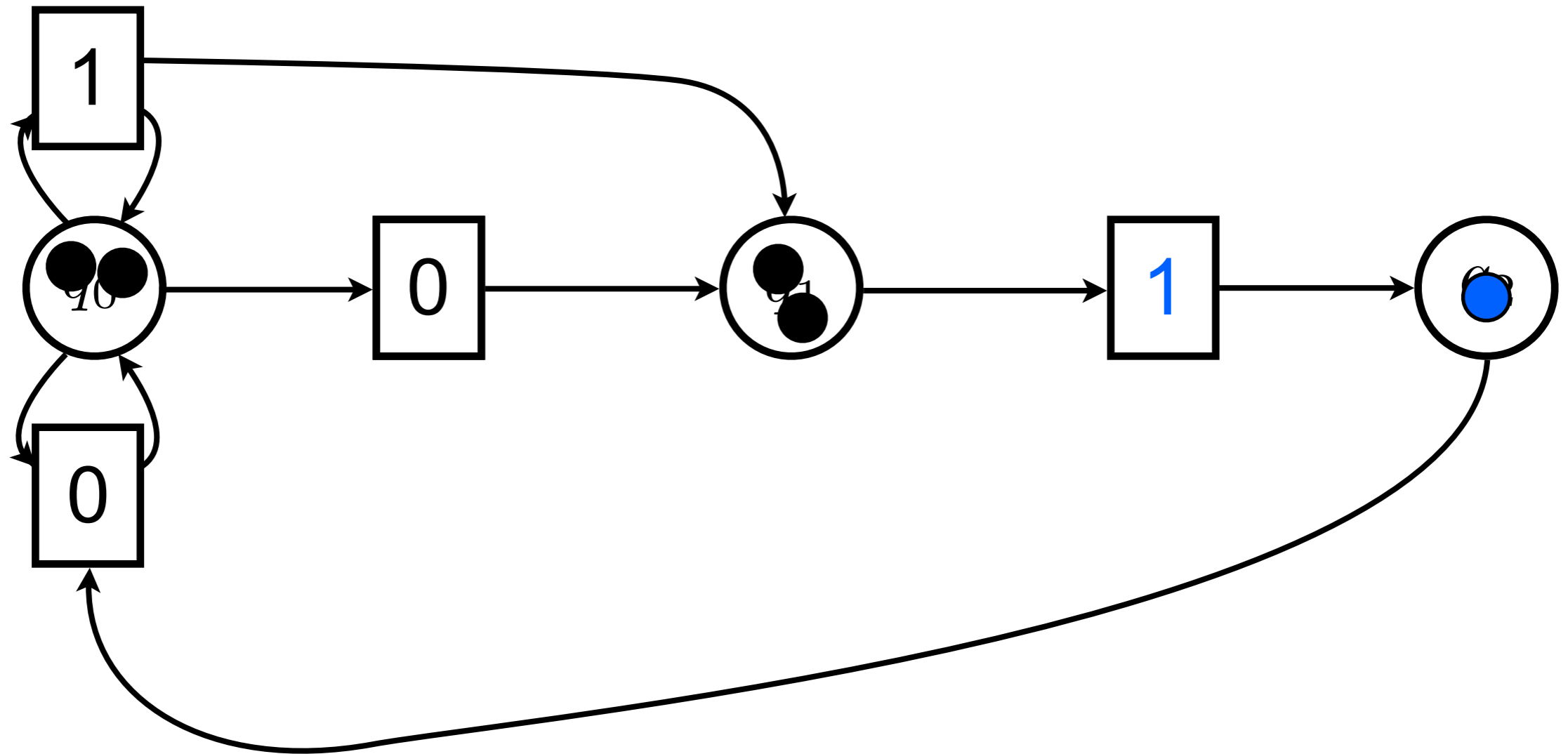
Example: token game



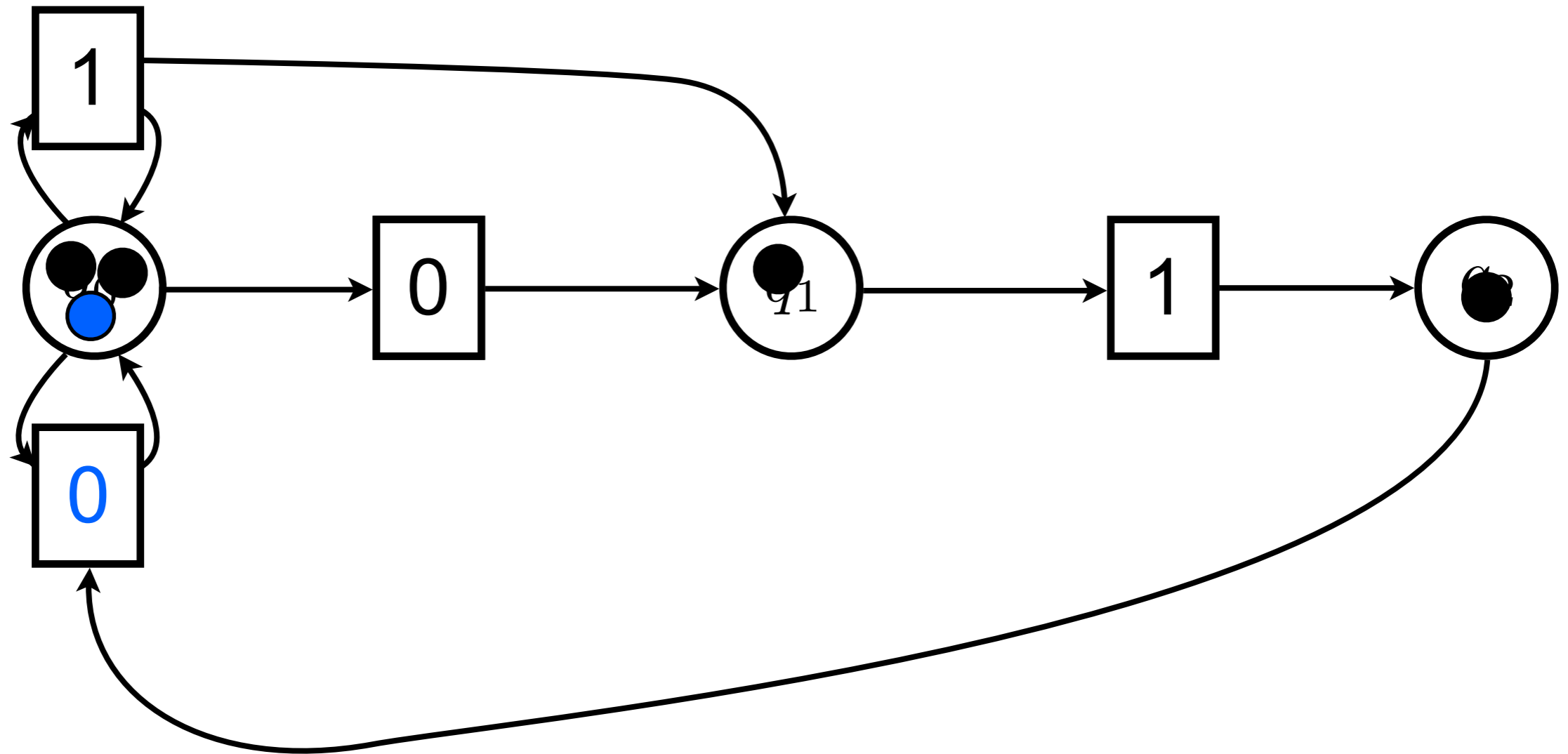
Example: token game



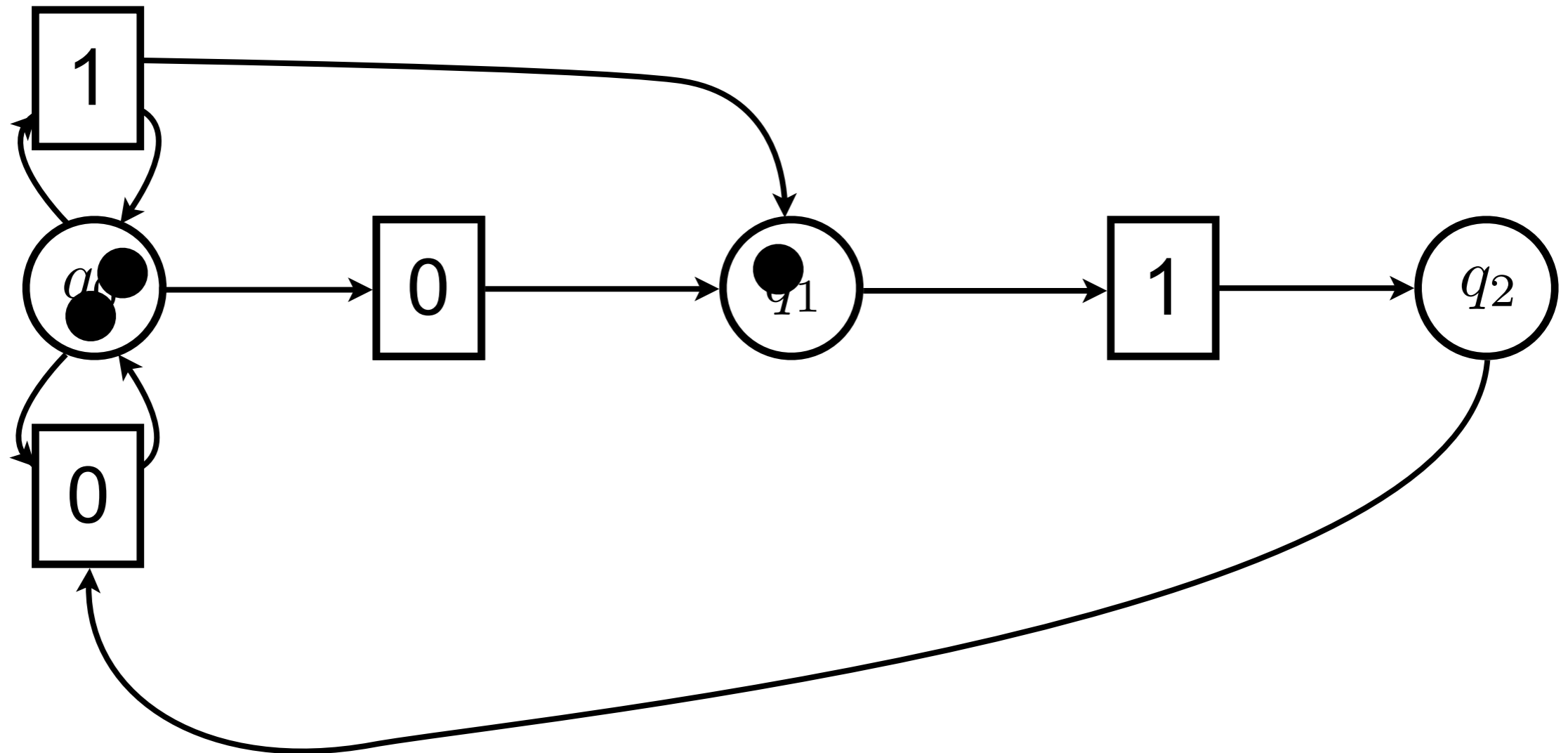
Example: token game



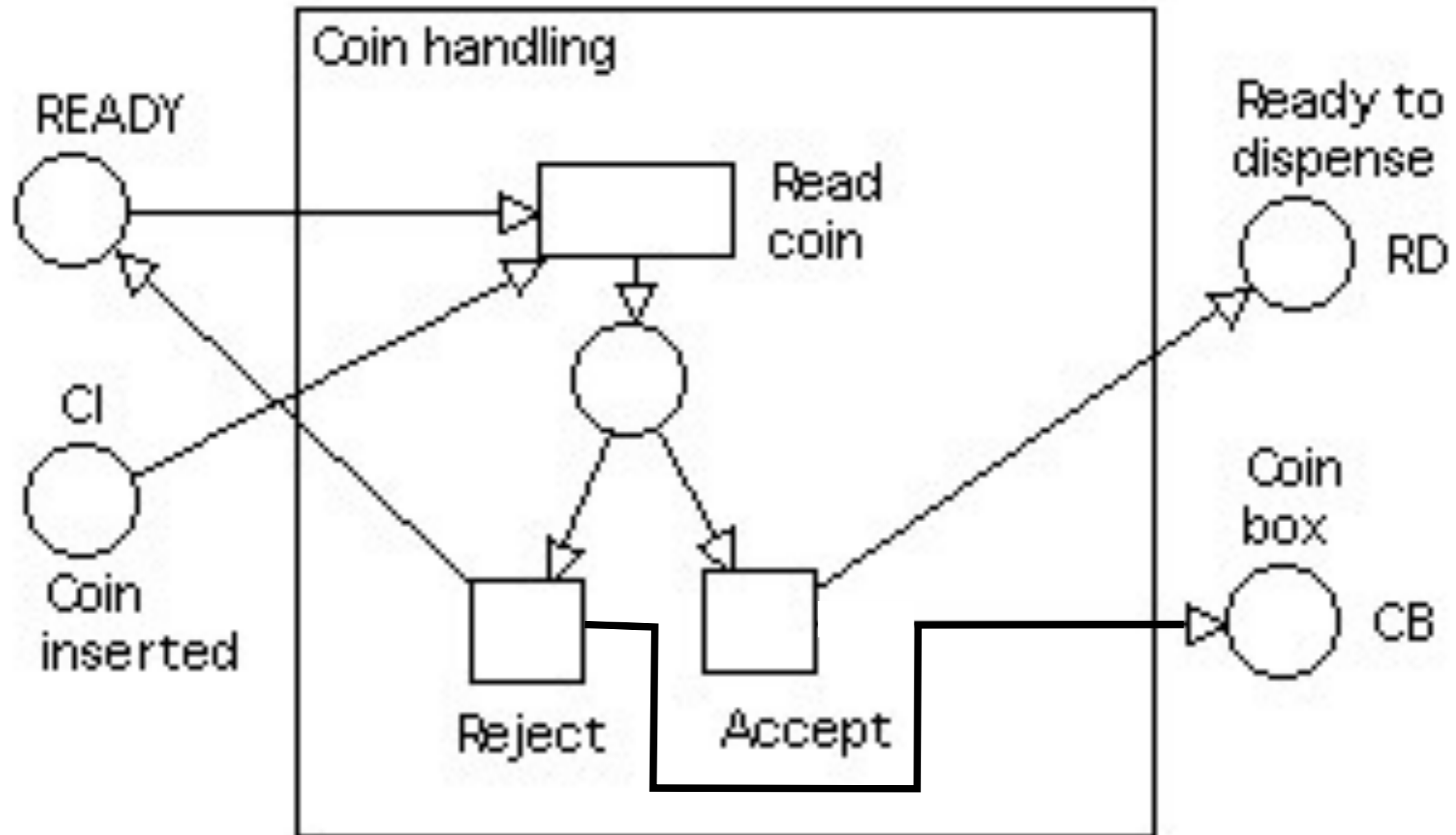
Example: token game



Example: token game



Example: Coin Handling



Some facts

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)