Business Processes Modelling

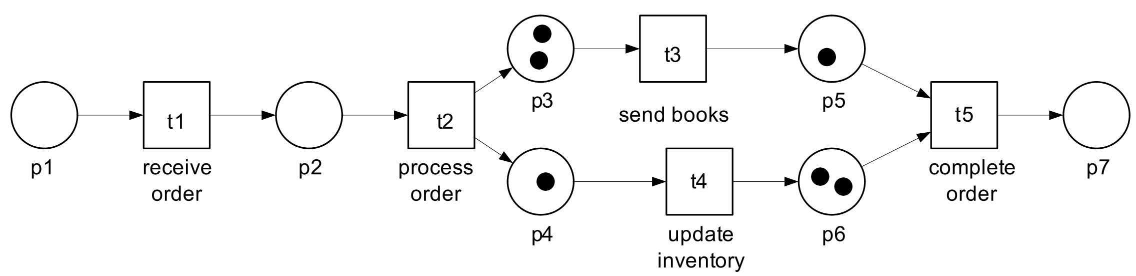
MPB (6 cfu, 295AA)

Roberto Bruni

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

08 - From automata to nets

Object



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

So far, which category for this course?

Subjects are divided in two categories:
1) *too difficult matters, that CANNOT be studied*
2) *easy matters, that DO NOT NEED to be studied*

\- back of a t-shirt

Let us take a U-turn!

Preliminaries

Set notation

8. Are you familiar with set notation?

Altri dettagli

●	Yes	54
●	No	5

Set notation

$\varnothing$	$A \cap B$	$A \cup B$	$A \setminus B$ $A - B$	$\overline{A}$
$a \in A$	$A = B$	$A \subseteq B$	$A \subset B$	$A \times B$
$a \notin A$	$A \neq B$	$A \not\subseteq B$	$\wp(A)$	$A \cap B = \emptyset$

$\mathbb{N}$	$\mathbb{Z}$	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{B}$

$N \subseteq \mathbb{N}$ $\quad N \in \wp(\mathbb{N})$ $\quad S \subseteq \wp(\mathbb{N})$

Functions, relations

9. Are you familiar with functions (f:A->B) and relations?

Altri dettagli

Dati analitici

Yes	52
No	7

10. Do you agree that a subset S of A can be seen as a function from A to the set of Booleans?

Altri dettagli

Dati analitici

Yes	36
No	3
I do not understand the question	20

Functions, relations

$f : A \to B$

$R \subseteq A \times B$

functions as relations

$$R_f \triangleq \{(a, f(a)) \mid a \in A\}$$

sets as functions
(characteristic function)

$$f_N : \mathbb{N} \to \mathbb{B}$$

$$f_N(n) \triangleq \begin{cases} 1 & n \in N \\ 0 & \text{otherwise} \end{cases}$$

$$N = \{n \mid f_N(n) = 1\}$$

First order logic

12. Are you familiar with propositional logic?

Altri dettagli

Dati analitici

●	Yes	50
●	No	9

First order logic

ff false 0 F tt 1 true T $P \wedge Q$ $P \vee Q$ $\neg P$

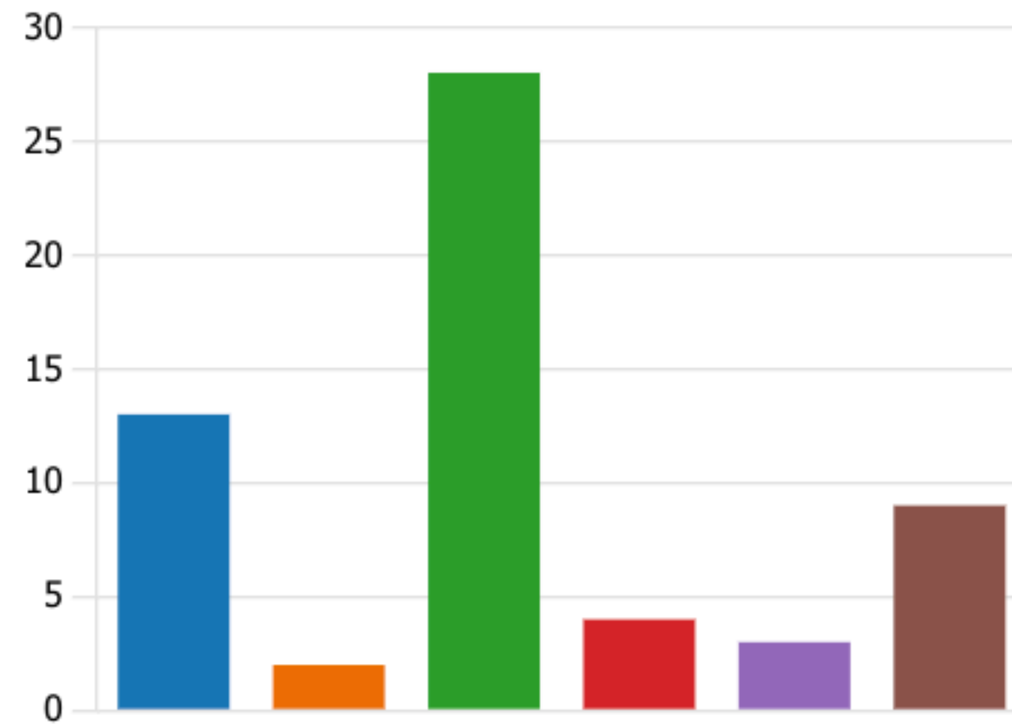
$\exists x.\ P(x)$ $\forall x.\ P(x)$ $P \Rightarrow Q$ $P \Leftrightarrow Q$

First order logic

3. Logical implication "P implies Q" (also written "P => Q") is equivalent to:

Altri dettagli   Dati analitici

P and Q	13
P or Q	2
(not P) or Q	28
P or (not Q)	4
(not P) or (not Q)	3
none of the above	9



4. Do you agree that "P implies Q" is logically equivalent to "(not Q) implies (not P)?

Altri dettagli   Dati analitici

Yes	37
No	22

First order logic

ff false 0 F

tt true 1 T

$P \wedge Q$ $P \vee Q$ $\neg P$

$\exists x.\ P(x)$ $\forall x.\ P(x)$ $P \Rightarrow Q$ $P \Leftrightarrow Q$

meaning of implication!

$$P \Rightarrow Q$$

$$Q \vee \neg P$$

$$\neg Q \Rightarrow \neg P$$

First order logic

15. Do you remember De Morgan's law about negation, conjunction and disjunction?

Altri dettagli

Dati analitici

Yes	31
No	28
I do not understand the question	0

16. Do you know what are the universal and existential quantifiers in predicate logic?

Altri dettagli

Dati analitici

Yes	38
No	21

First order logic

ff false 0 F tt true 1 T

$P \wedge Q$ $P \vee Q$ $\neg P$

$\exists x.\ P(x)$ $\forall x.\ P(x)$ $P \Rightarrow Q$ $P \Leftrightarrow Q$

meaning of implication!

$$P \Rightarrow Q$$

$$Q \vee \neg P$$

$$\neg Q \Rightarrow \neg P$$

order of quantifiers matters!

$$\forall n \in \mathbb{N}.\ \exists m \in \mathbb{N}.\ n < m$$

$$\exists m \in \mathbb{N}.\ \forall n \in \mathbb{N}.\ n < m$$

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

Strings

Alphabet A $\quad A^n \triangleq \underbrace{A \times \cdots \times A}_{n} \quad A^* \triangleq \bigcup_{n \in \mathbb{N}} A^n$

$$\begin{aligned}
\mathbb{B} &= \{0, 1\} \\
\mathbb{B}^0 &= \{\epsilon\} \\
\mathbb{B}^1 &= \{0, 1\} \\
\mathbb{B}^2 &= \{00, 01, 10, 11\} \\
\mathbb{B}^3 &= \{000, 001, 010, 011, 100, 101, 110, 111\} \\
&\ldots \\
\mathbb{B}^* &= \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}
\end{aligned}$$

Inductive definitions

17. Do you know what is an inductive definition?

Altri dettagli   Dati analitici

Yes	31
No	28

18. Do you know what is a recursive definition?

Altri dettagli   Dati analitici

Yes	41
No	18

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 k^n

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 k^n

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

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

Recursive definition

Inductively defined functions

Let us define the exponential function k^n

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

inductive case: for any $k>0,\ n\geq 0$ we set
$exp(k,\boxed{n+1}) \triangleq exp(k,\boxed{n})$ x k

More complex case **Simpler case**

Recursive definitions

$$\binom{n}{k} \triangleq \frac{n!}{k!\,(n-k)!}$$

$$\binom{n}{0} \triangleq 1$$

$$\binom{n+1}{k+1} \triangleq \frac{(n+1)\binom{n}{k}}{k+1}$$

$$f(n) \triangleq \begin{cases} 1 & \text{if } n \leq 1 \\ f(n/2) & \text{if } n > 1 \wedge n\%2 = 0 \\ f(3n+1) & \text{otherwise} \end{cases}$$

$$f(12) = f(6) = f(3) = f(10) = f(5) = f(16) = f(8) = f(4) = f(2) = f(1) = 1$$

Inductive definitions

$$
\begin{aligned}
0! &\triangleq 1 \\
(n+1)! &\triangleq n! \cdot (n+1)
\end{aligned}
$$

$$
\begin{aligned}
A^0 &\triangleq \{\epsilon\} \\
A^{(n+1)} &\triangleq A \times A^n
\end{aligned}
$$

$$
\begin{aligned}
|\epsilon| &\triangleq 0 \\
|w\ a| &\triangleq |w| + 1
\end{aligned}
$$

Finite automata examples

Finite state automaton

21. Do you know what is a Finite State Automata?

Altri dettagli   Dati analitici

Yes	18
No	41

22. Do you know what is the language recognized by an automata?

Altri dettagli   Dati analitici

Yes	14
No	41
I do not understand the question	4

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

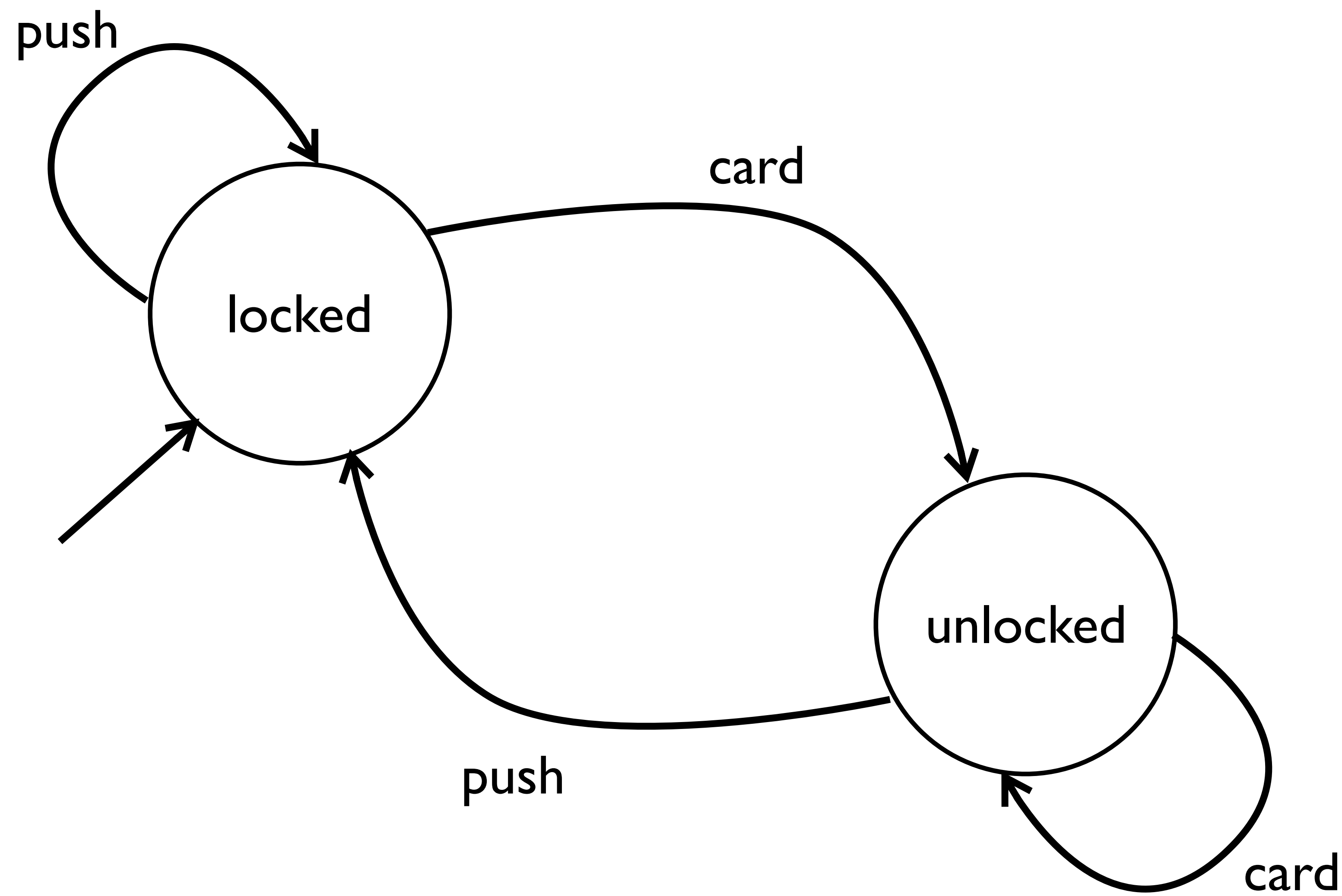
1. Identify the admissible **states** of the system
(Optional: mark some states as error states)

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

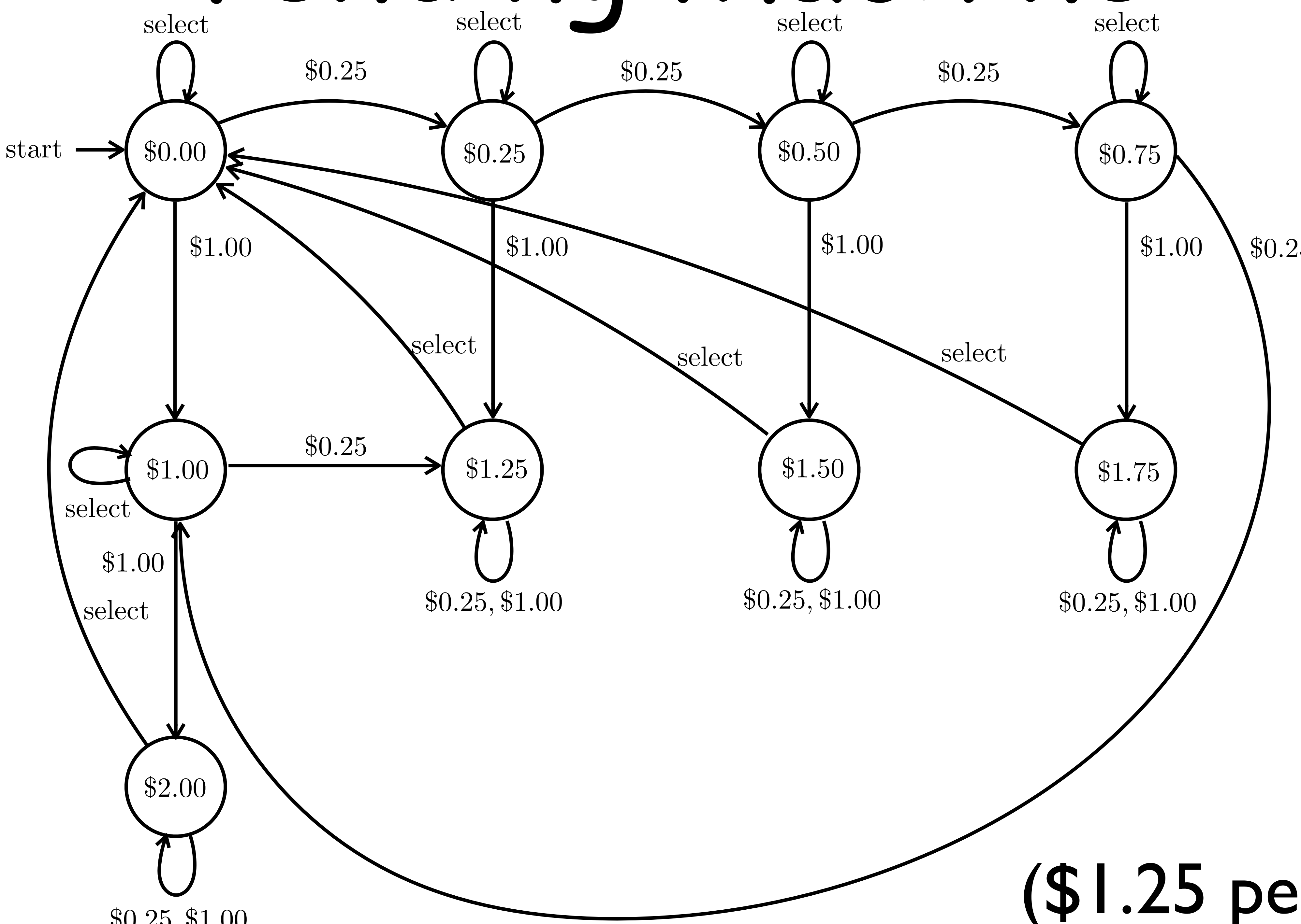
3. Set the **initial state**

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

Example: Turnstile

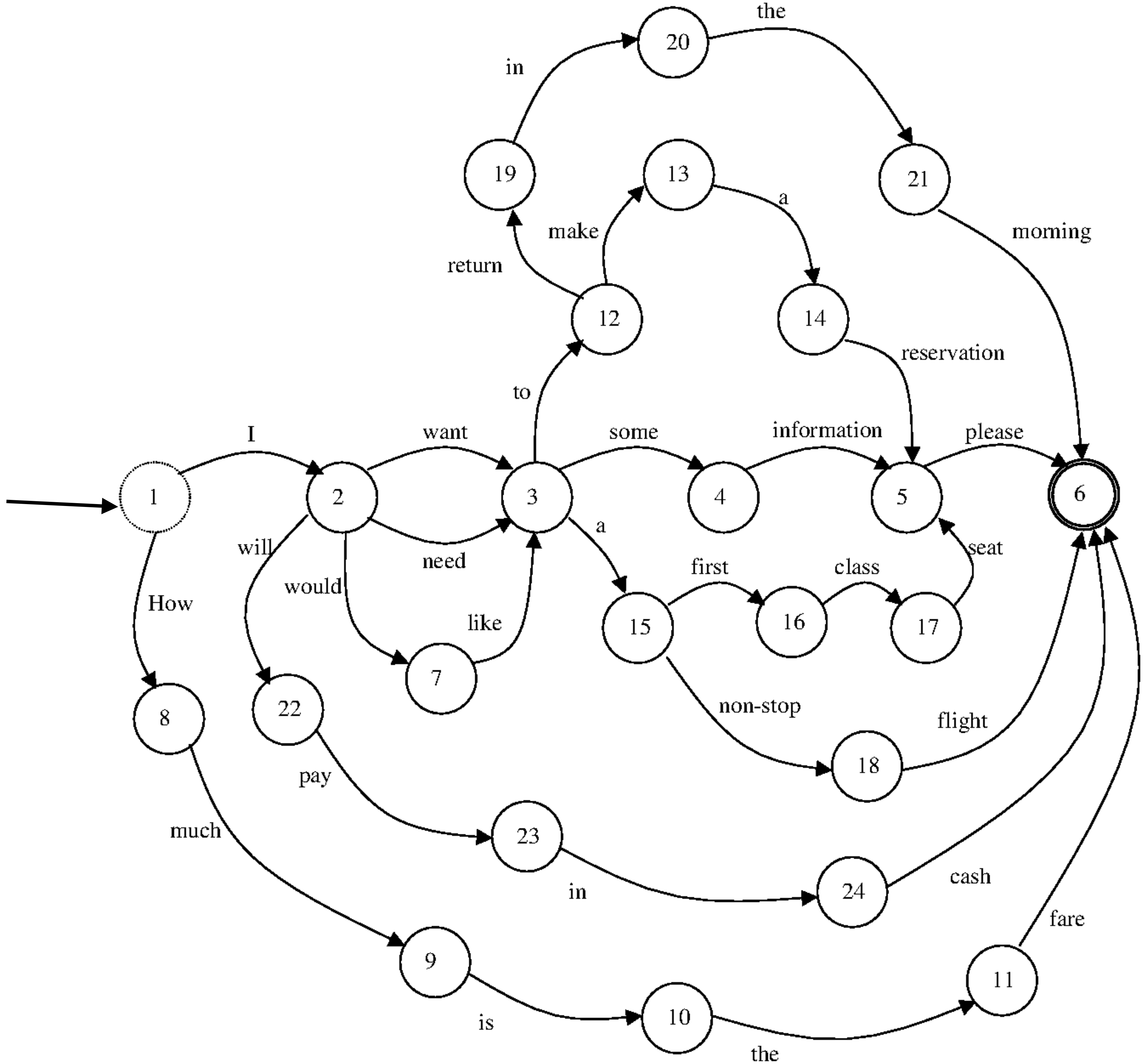


Example: Vending Machine

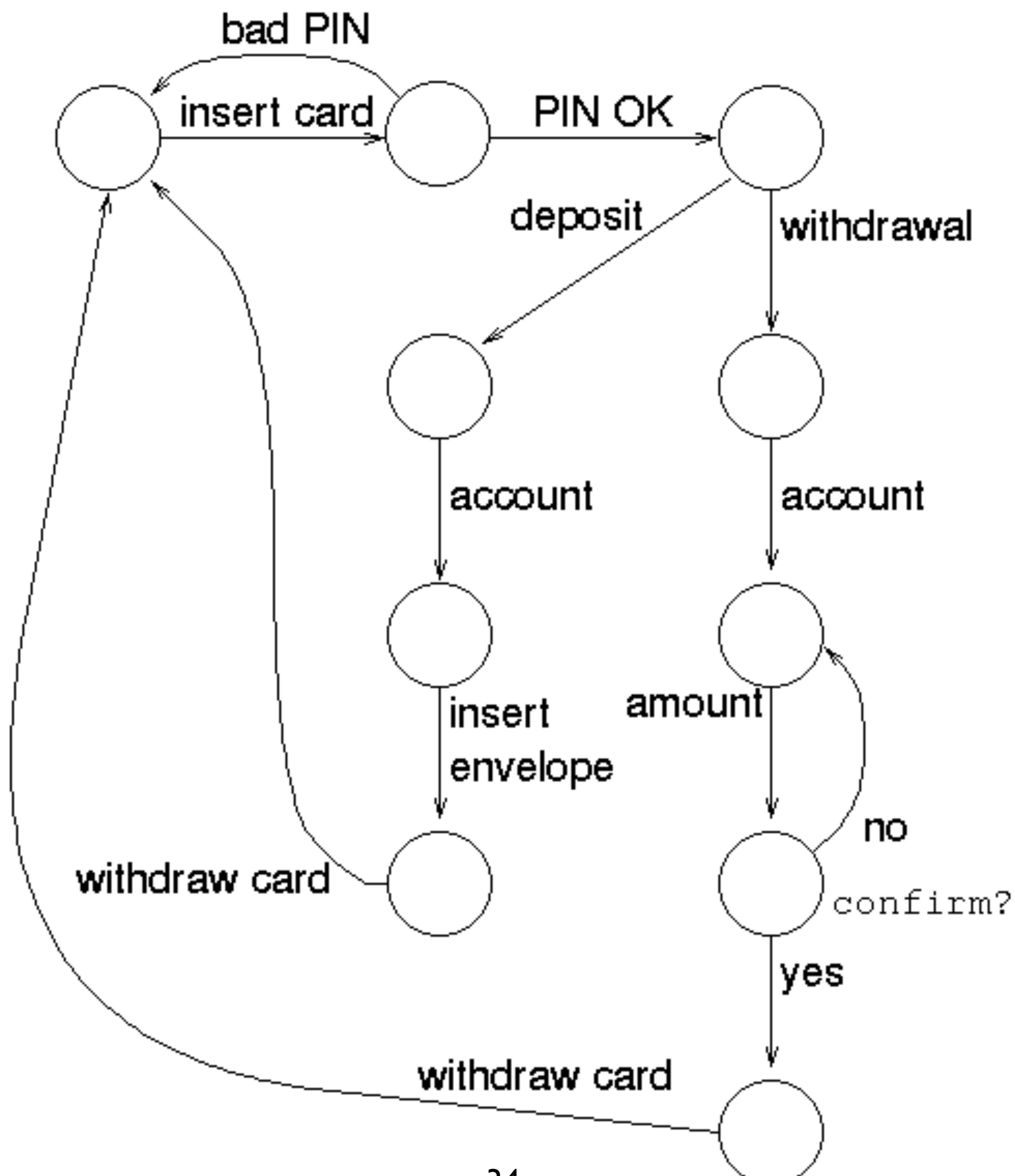


($1.25 per soda)

Example: Language Processing



Example: ATM

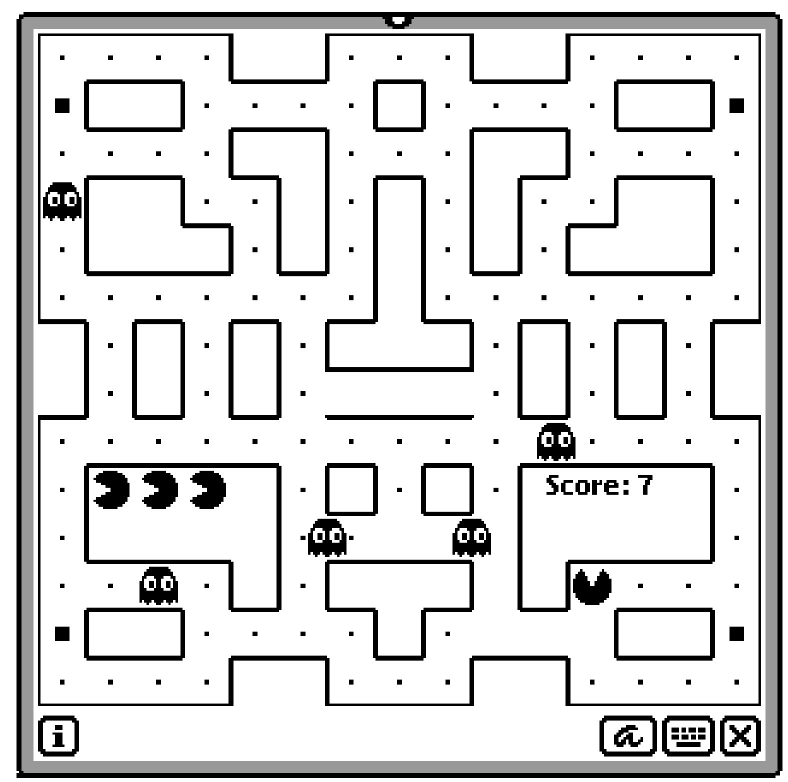


bad PIN
insert card
PIN OK
deposit
withdrawal
account
account
insert
envelope
amount
no
withdraw card
confirm?
yes
withdraw card

Computer controlled characters for games

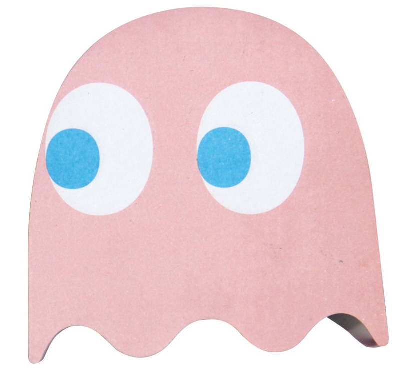
States = characters behaviours

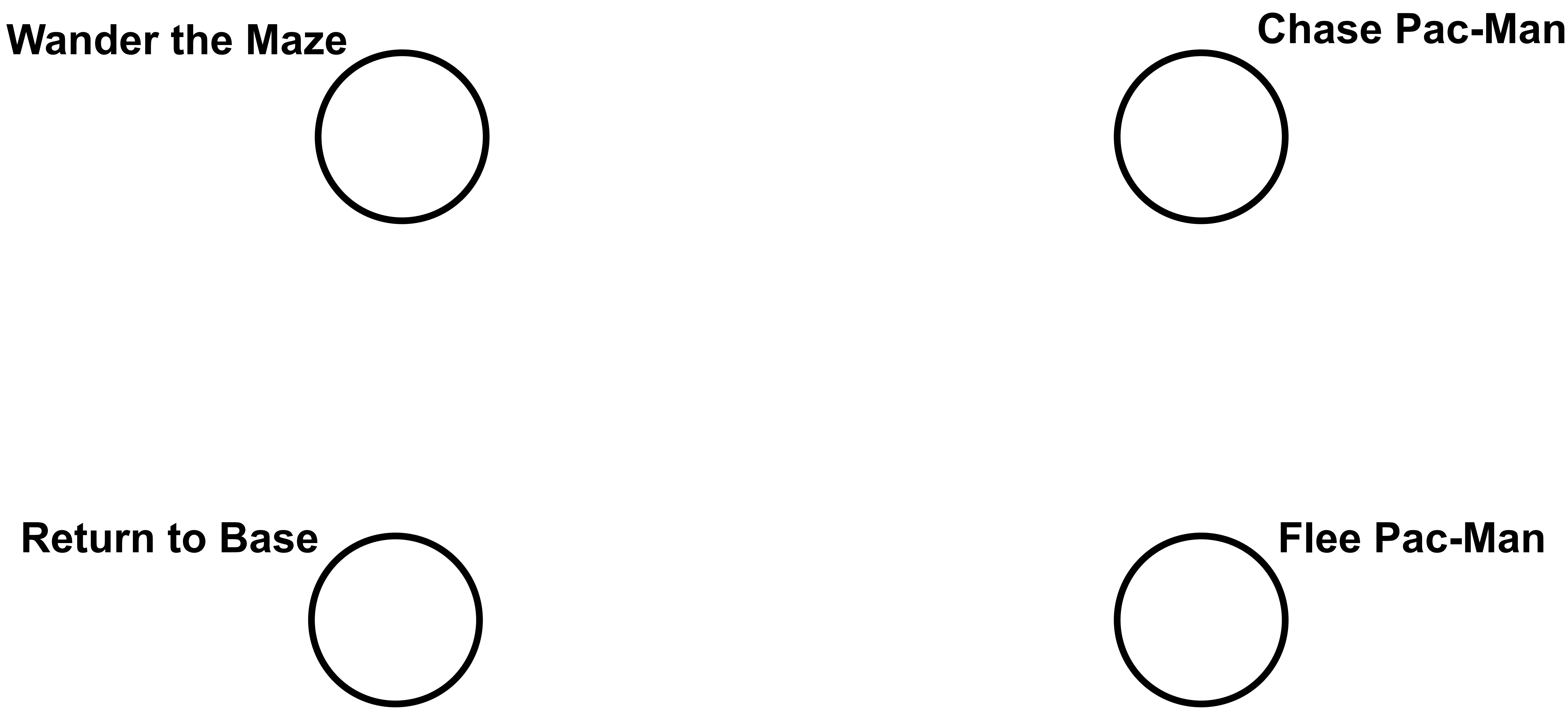
Transitions = events that cause a change in behaviour



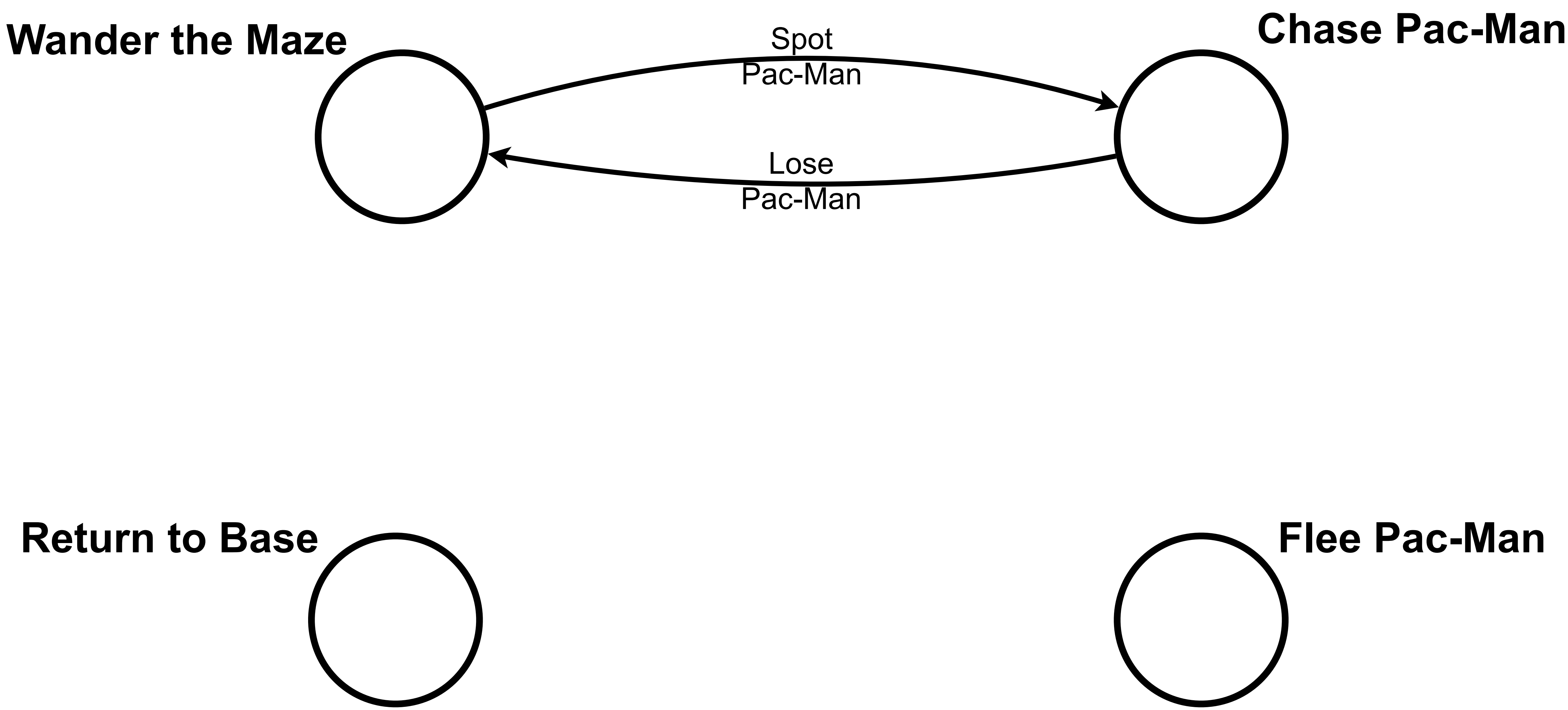
Example:
Pac-man moves in a maze
wants to eat pills
is chased by ghosts
by eating power pills, pac-man can defeat ghosts

Example: Pac-Man Ghosts

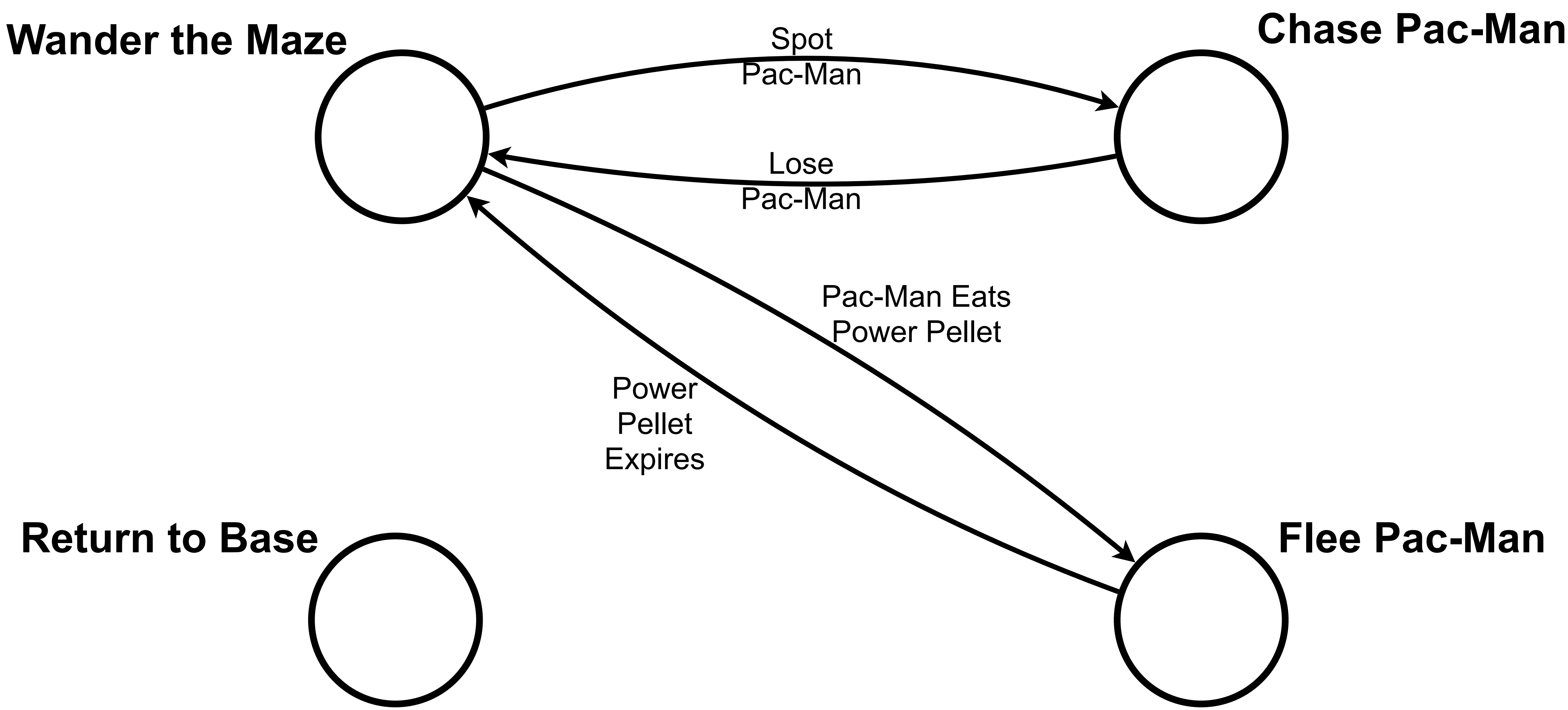




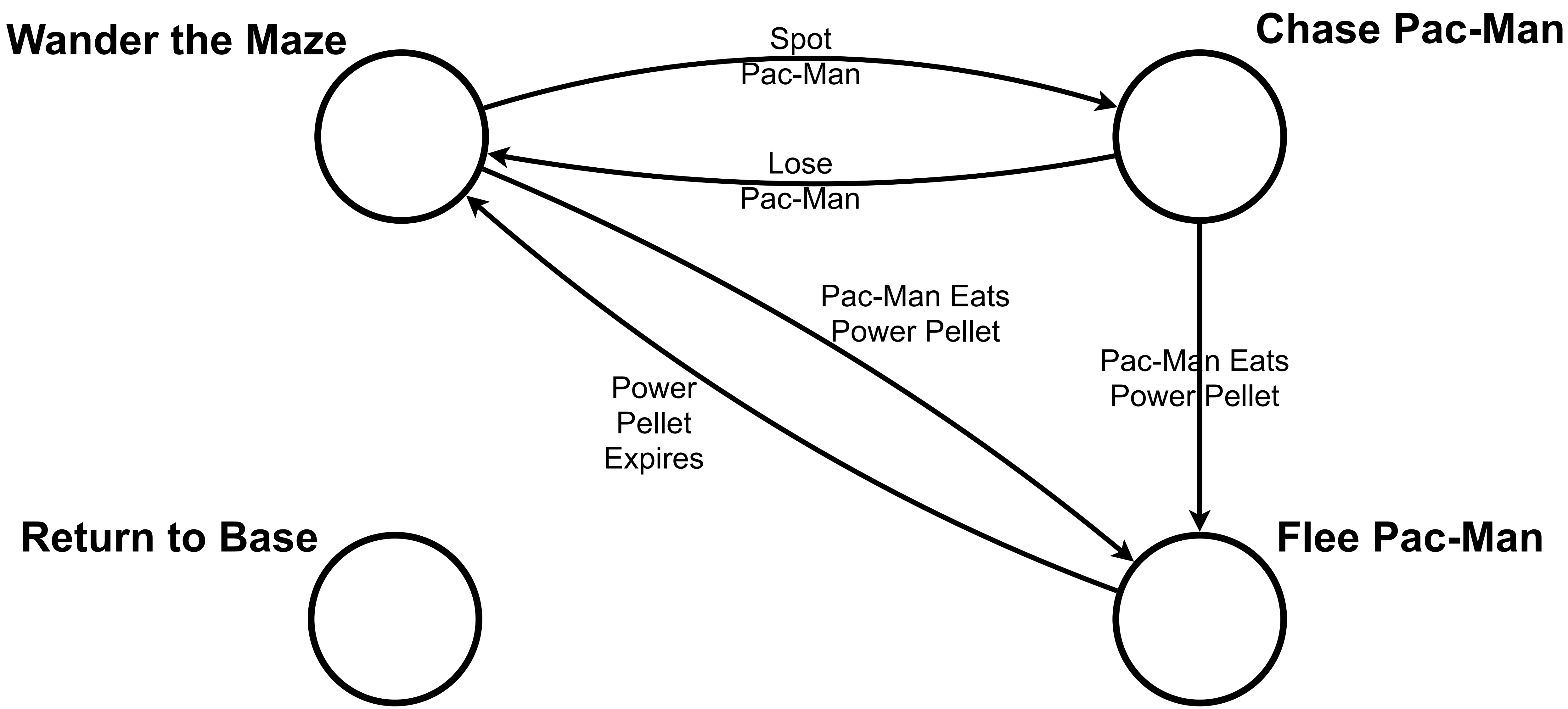
Example: Pac-Man Ghosts



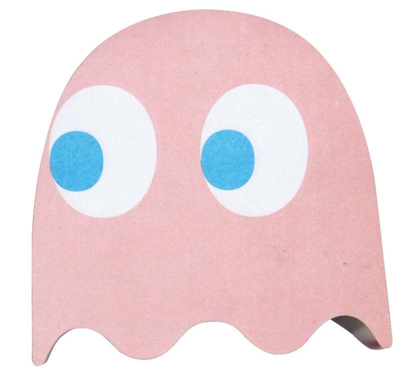
Example: Pac-Man Ghosts

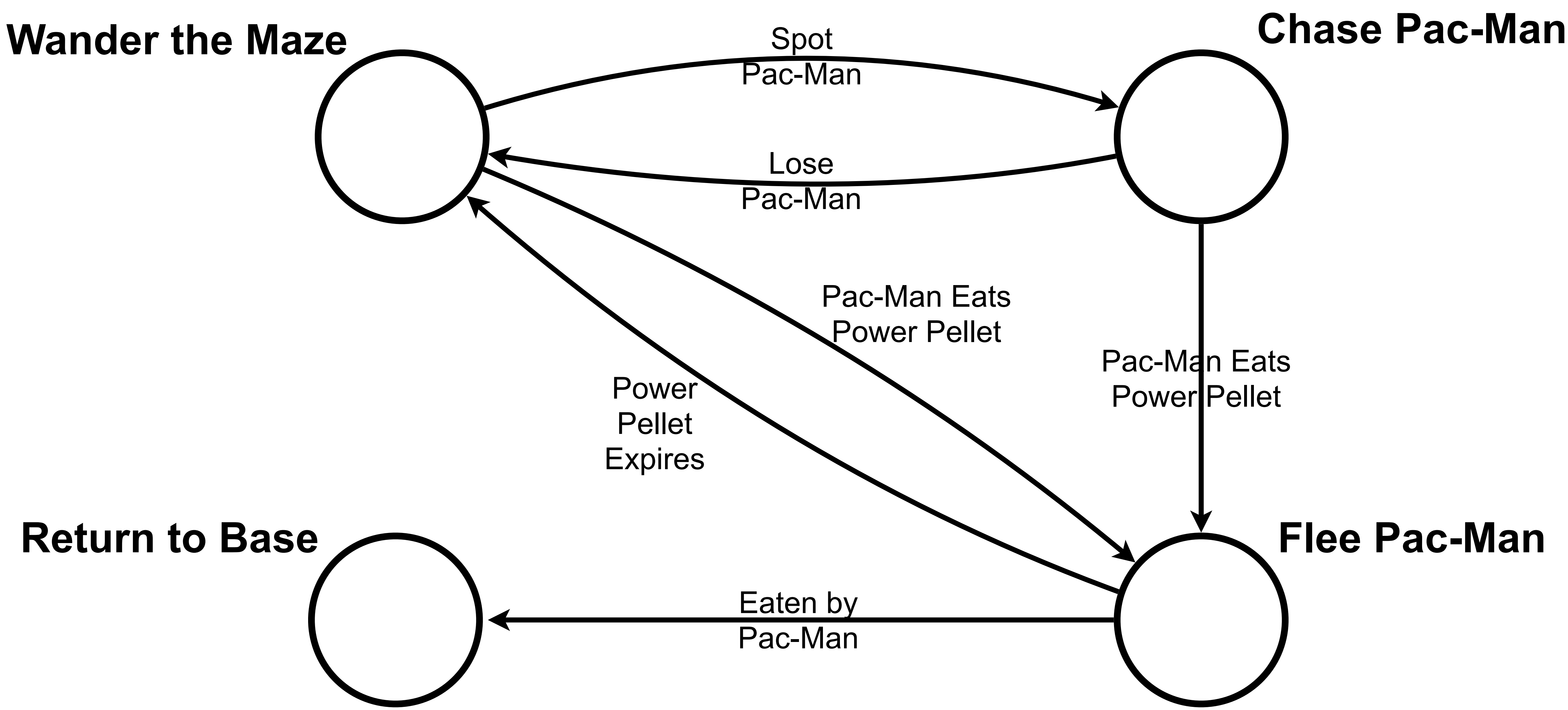


Example: Pac-Man Ghosts

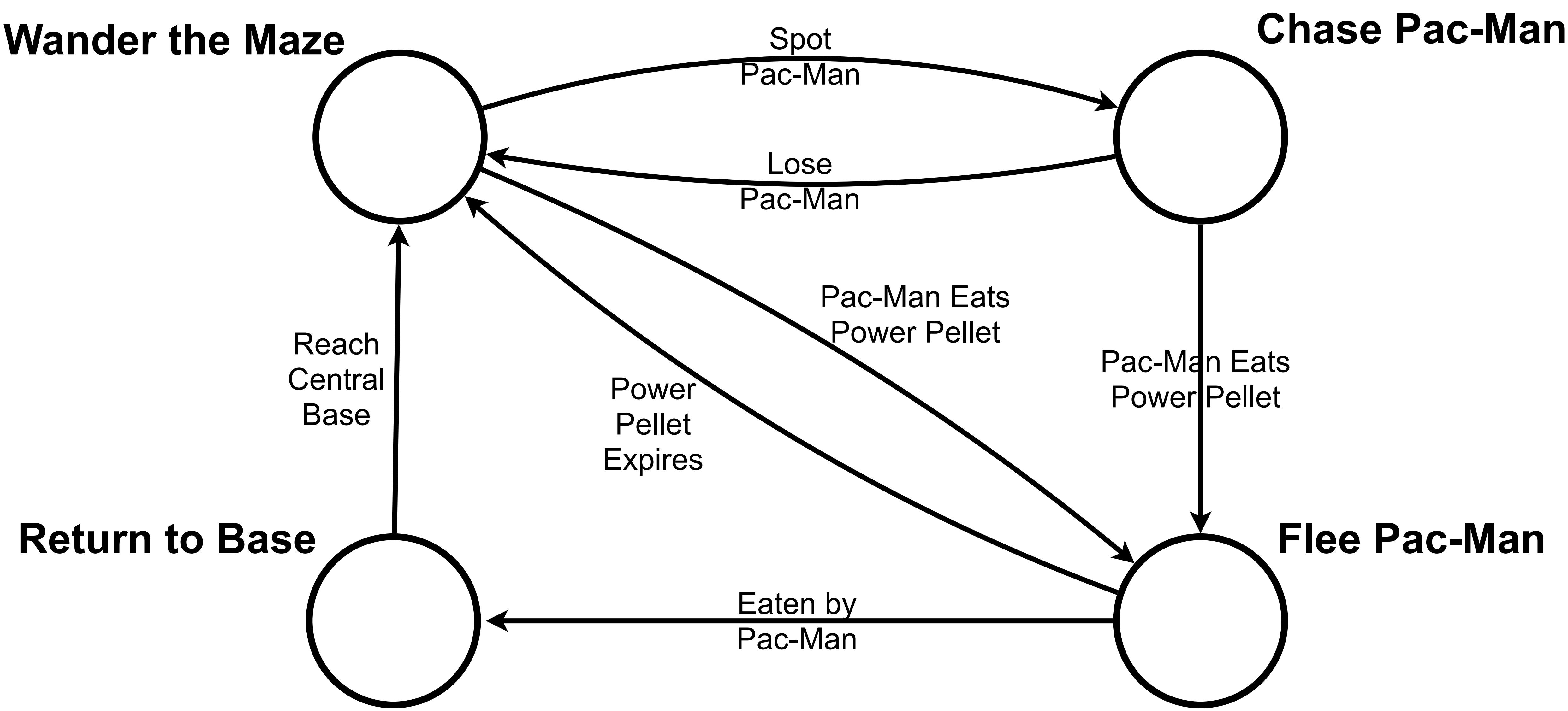


Example: Pac-Man Ghosts

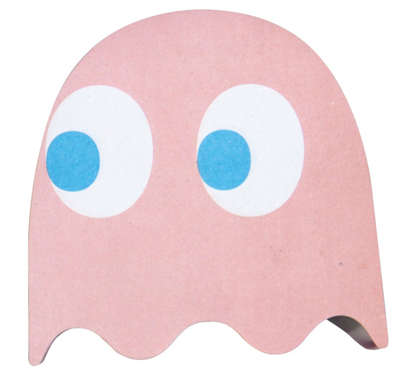


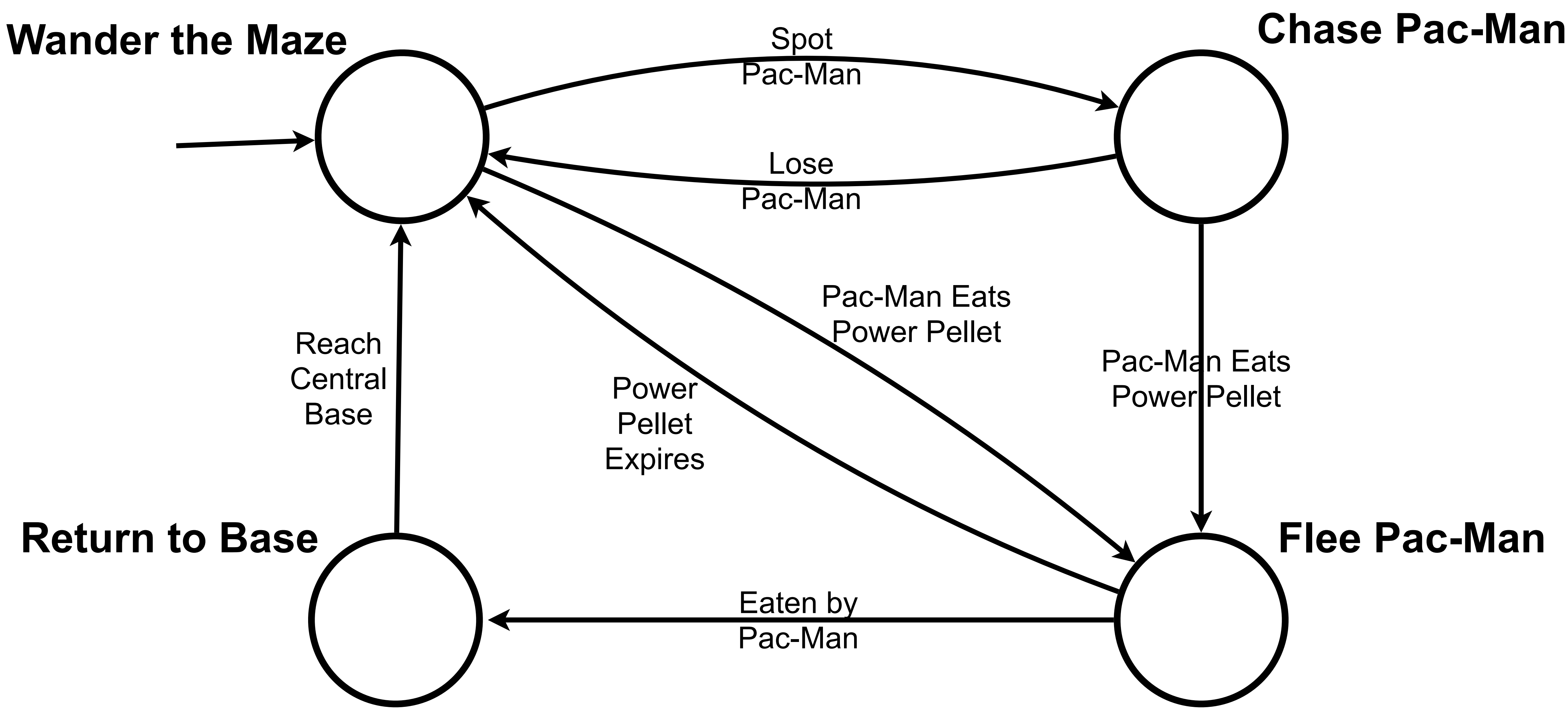


Example: Pac-Man Ghosts

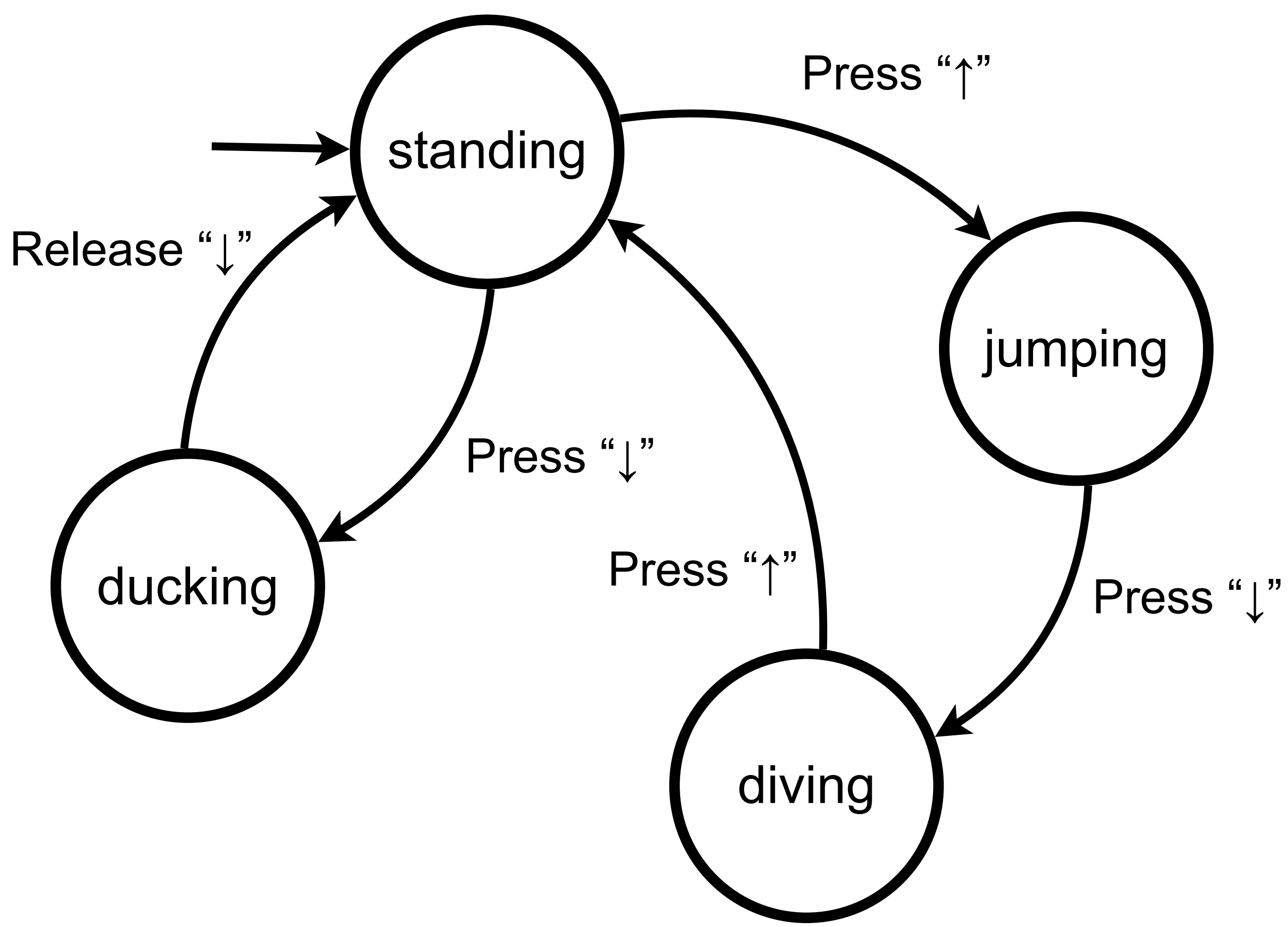


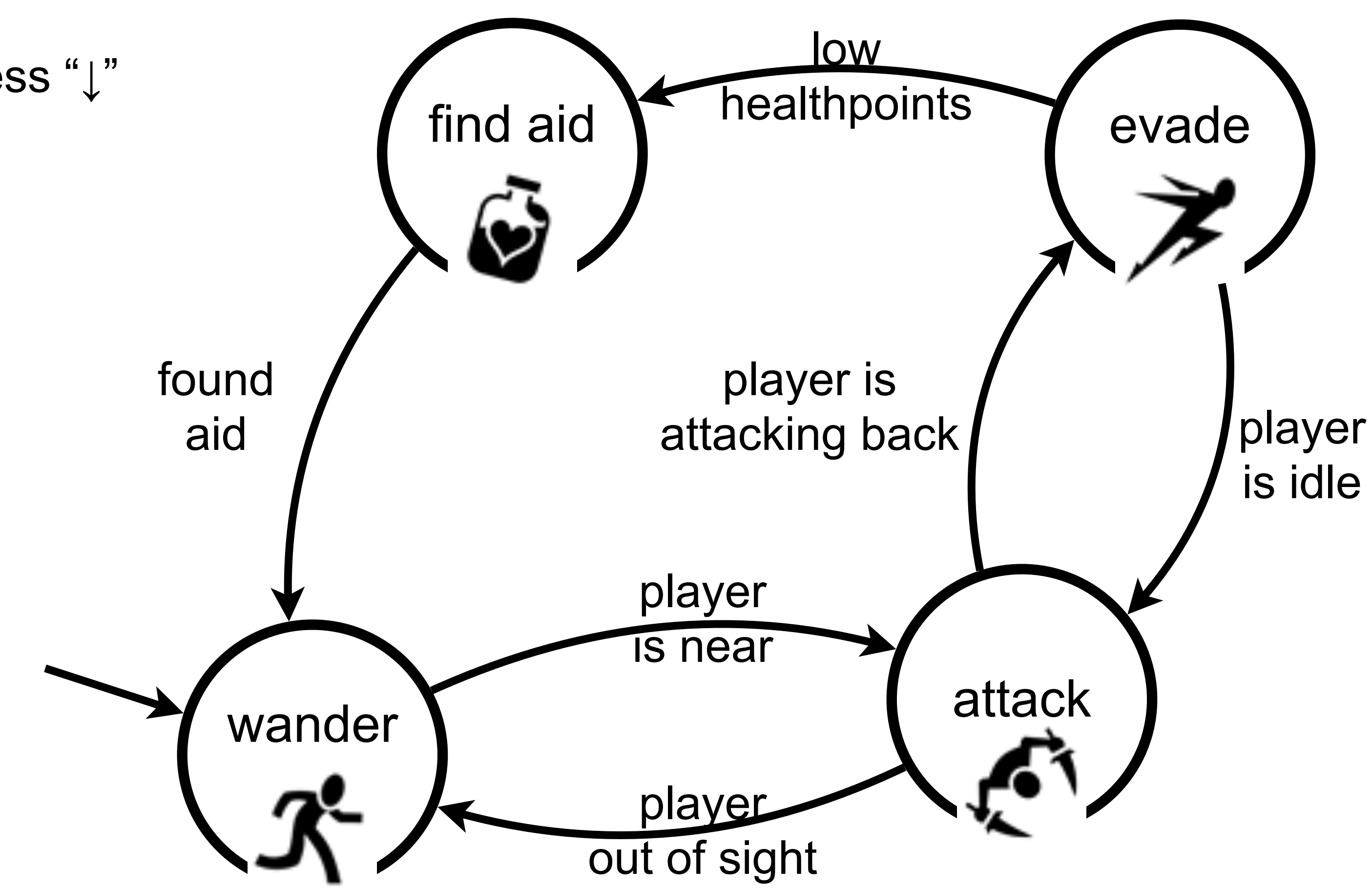
Example: Pac-Man Ghosts





Other examples





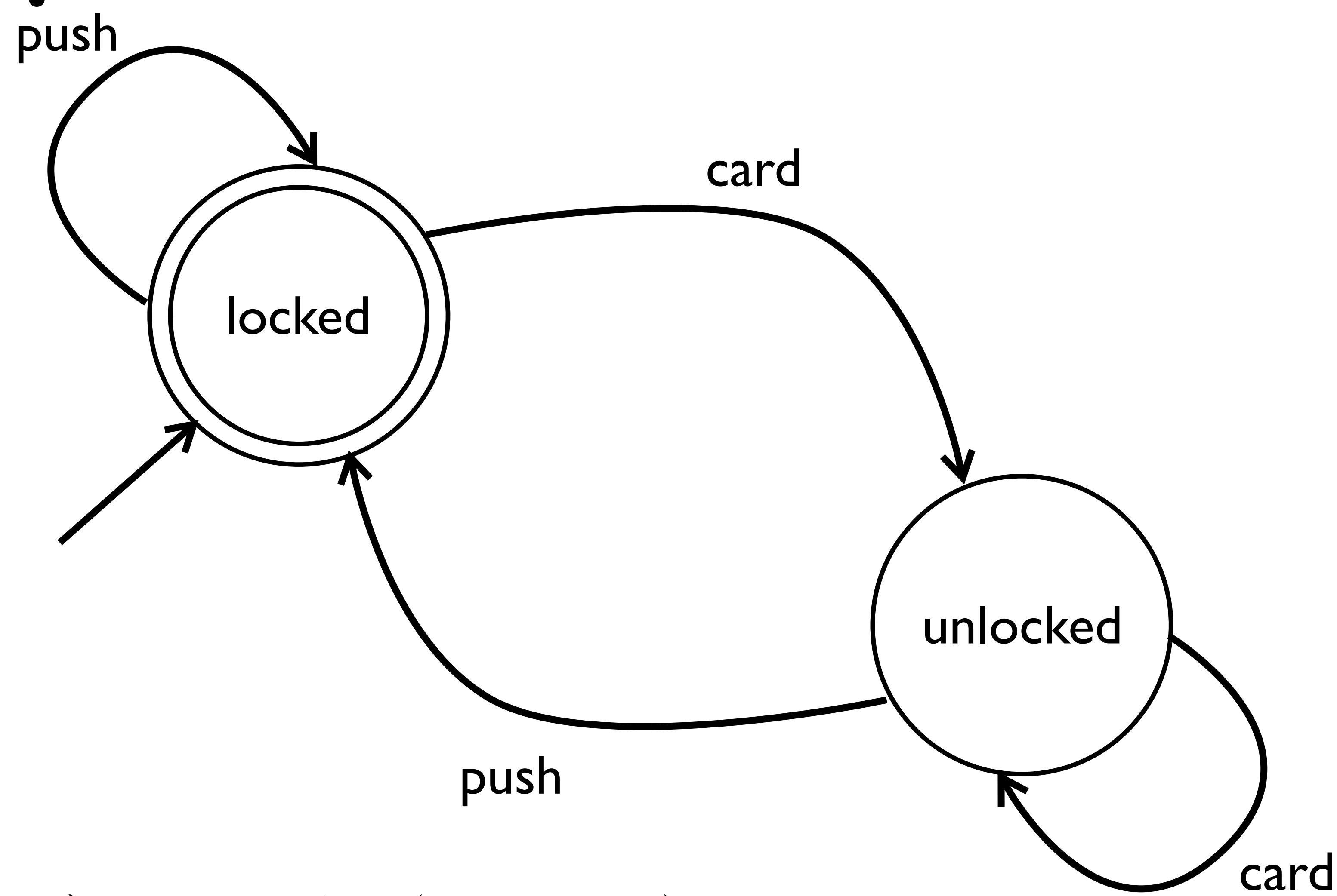
Finite state automata, formally

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)

Example: Turnstile



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)

$$Q = \{\text{locked}, \text{unlocked}\}$$

$$\Sigma = \{\text{push}, \text{card}\}$$

$$\delta(\text{locked}, \text{card}) = \text{unlocked}$$

$$q_0 = \text{locked}$$

$$F = \{\text{locked}\}$$

Extended transition function (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

$$\widehat{\delta}(q, \epsilon) \triangleq q$$

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

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

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

Extended transition function (destination function)

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

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

$$\widehat{\delta}(q, \epsilon) \triangleq q$$

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

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

Recursive definition

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

Extended transition function (destination function)

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

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

$$\widehat{\delta}(q, \epsilon) \triangleq q$$

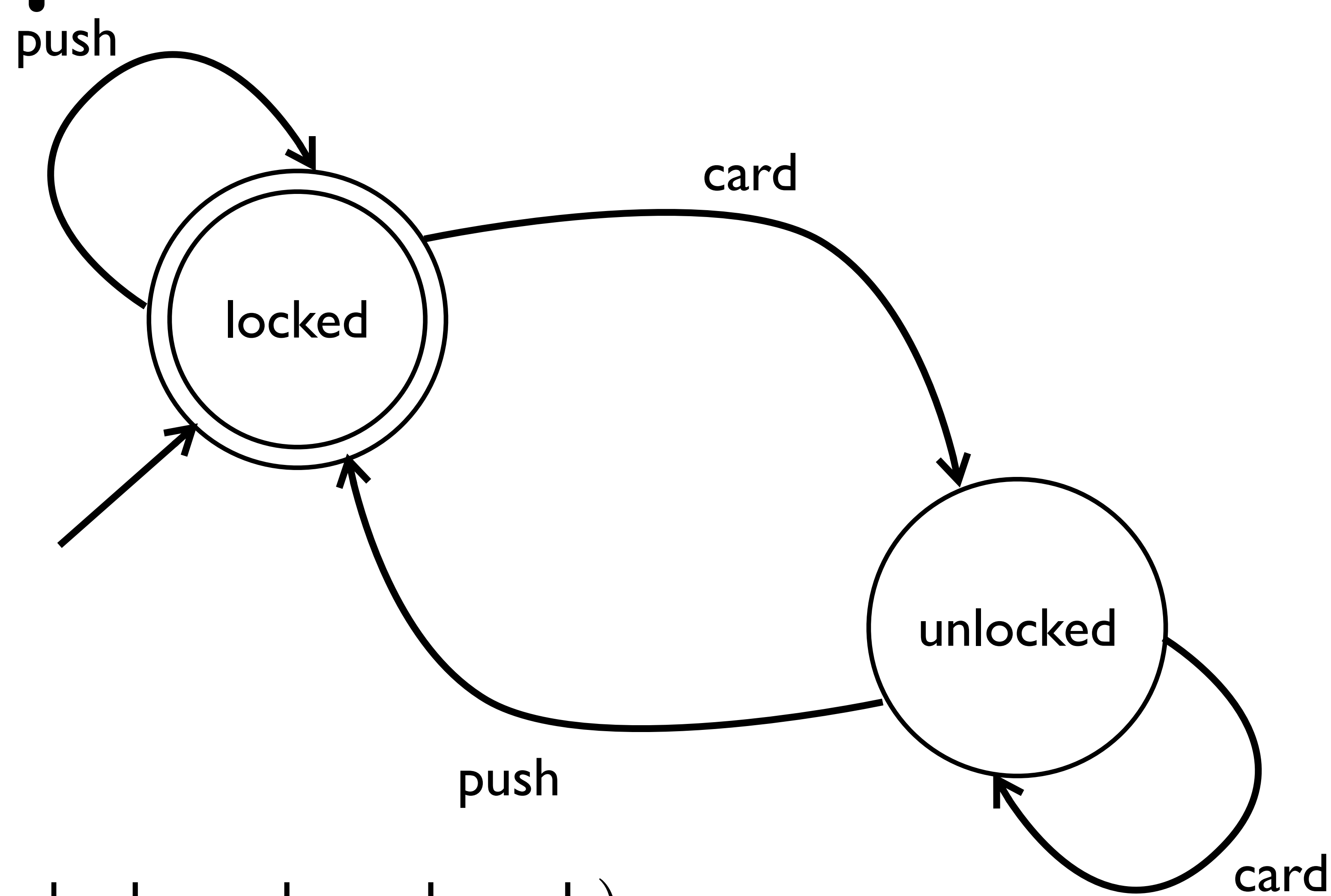
inductive case: For any $q \in Q, a \in \Sigma, w \in \Sigma^*$ we let

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

More complex case **Simpler case**

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

Example: Turnstile



$\delta(\text{locked}, \text{card}) = \text{unlocked}$

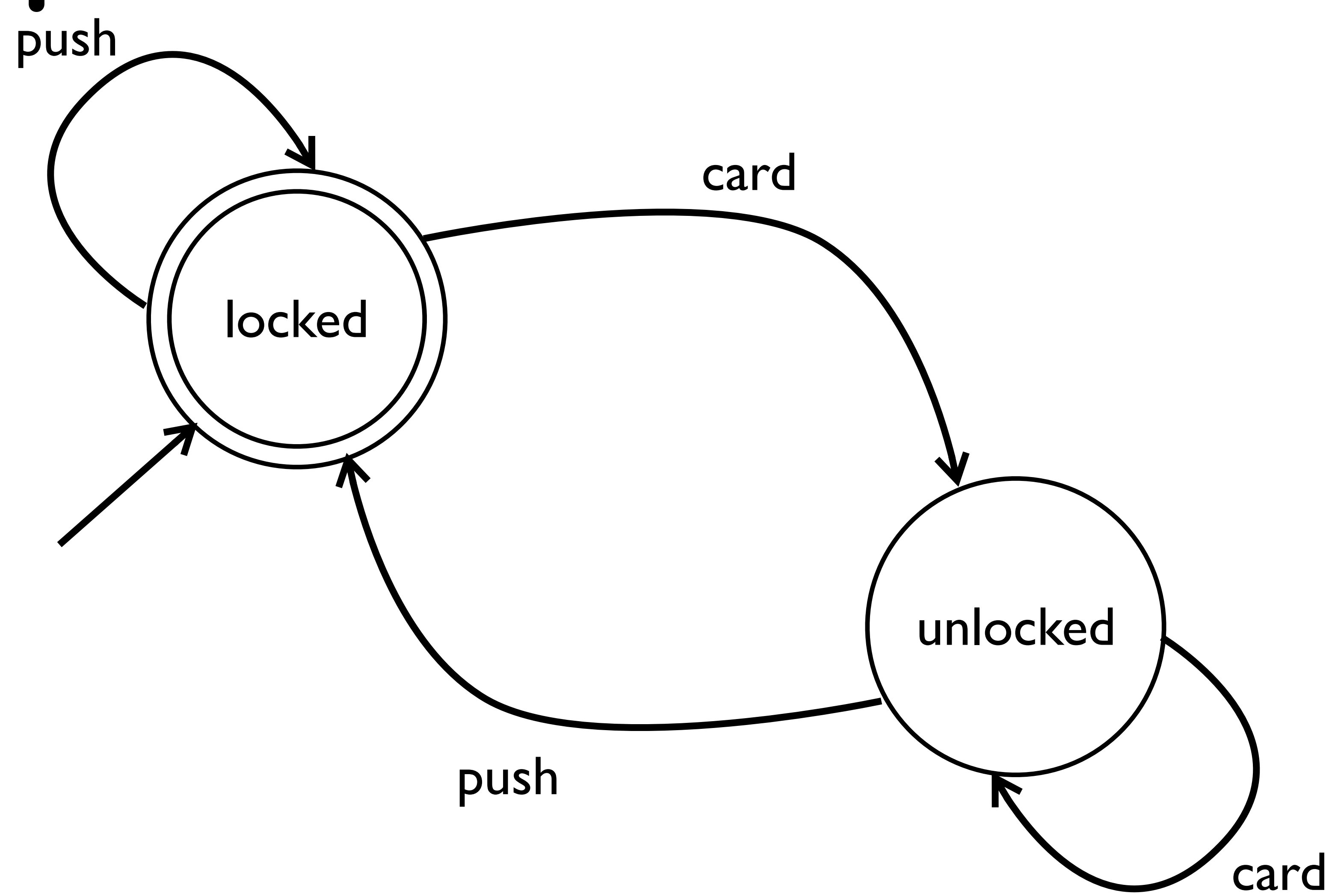
$\delta(\text{locked}, \text{push}) = \text{locked}$

$\delta(\text{unlocked}, \text{card}) = \text{unlocked}$

$\delta(\text{unlocked}, \text{push}) = \text{locked}$

$$
\begin{aligned}
&\widehat{\delta}(\text{locked}, \text{card card push})\\
&= \delta(\widehat{\delta}(\text{locked}, \text{card card}),\ \text{push})\\
&= \delta(\delta(\widehat{\delta}(\text{locked}, \text{card}),\ \text{card}),\ \text{push})\\
&= \delta(\delta(\delta(\widehat{\delta}(\text{locked}, \epsilon),\ \text{card}),\ \text{card}),\ \text{push})\\
&= \delta(\delta(\delta(\text{locked},\ \text{card}),\ \text{card}),\ \text{push})\\
&= \delta(\delta(\text{unlocked},\ \text{card}),\ \text{push})\\
&= \delta(\text{unlocked},\ \text{push}) = \text{locked}
\end{aligned}
$$

Example: Turnstile



$\delta(\text{locked}, \text{card}) = \text{unlocked}$

$\delta(\text{locked}, \text{push}) = \text{locked}$

$\delta(\text{unlocked}, \text{card}) = \text{unlocked}$

$\delta(\text{unlocked}, \text{push}) = \text{locked}$

$\widehat{\delta}(\text{locked}, \epsilon) = \text{locked}$

$\widehat{\delta}(\text{locked}, \text{card}) = \text{unlocked}$

$\widehat{\delta}(\text{locked}, \text{card card}) = \text{unlocked}$

$\widehat{\delta}(\text{locked}, \text{card card push}) = \text{locked}$

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

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

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

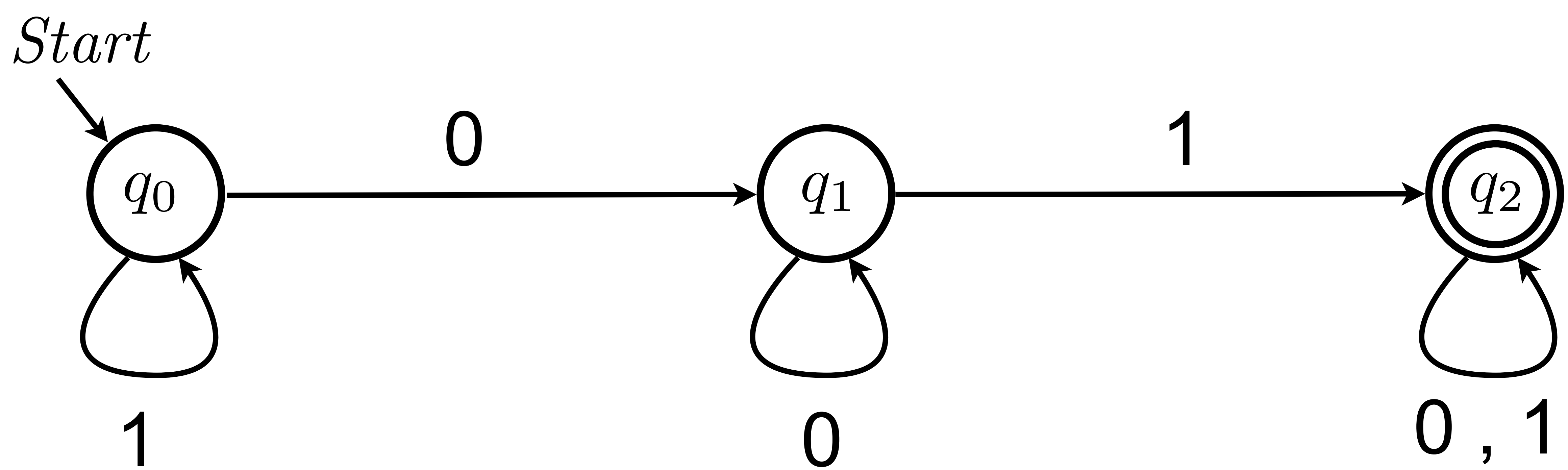
A DFA accepts a string w, if there is a path in its transition diagram such that:

it starts from the initial state

it ends in one final state

the sequence of labels in the path is exactly w

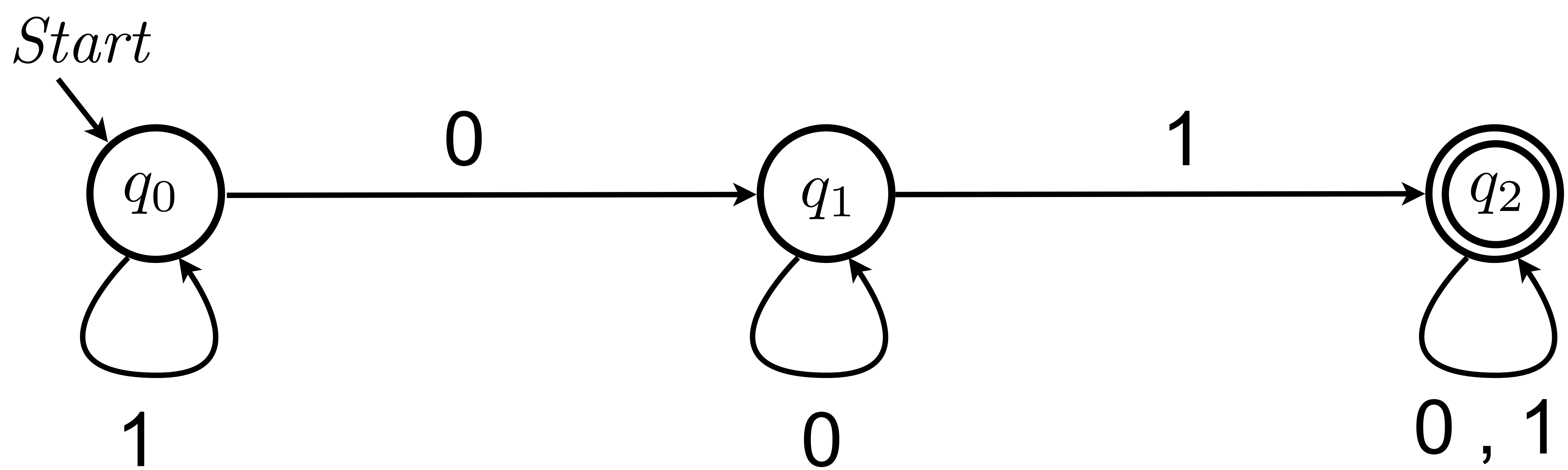
DFA: example



$$q_0 \; 1 \; q_0 \; 1 \; q_0 \; 1 \; q_0 \; 0 \; q_1 \; 0 \; q_1 \; 0 \; q_1 \notin F$$

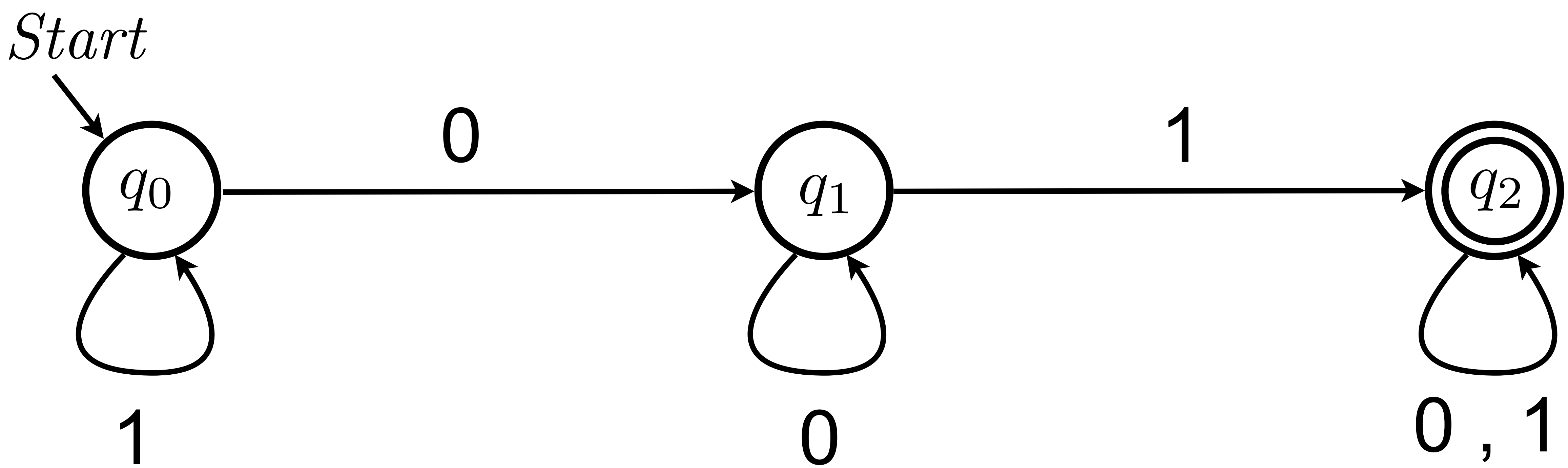
$$q_0 \; 1 \; q_0 \; 0 \; q_1 \; 0 \; q_1 \; 1 \; q_2 \; 1 \; q_2 \; 0 \; q_2 \in F$$

DFA: question time



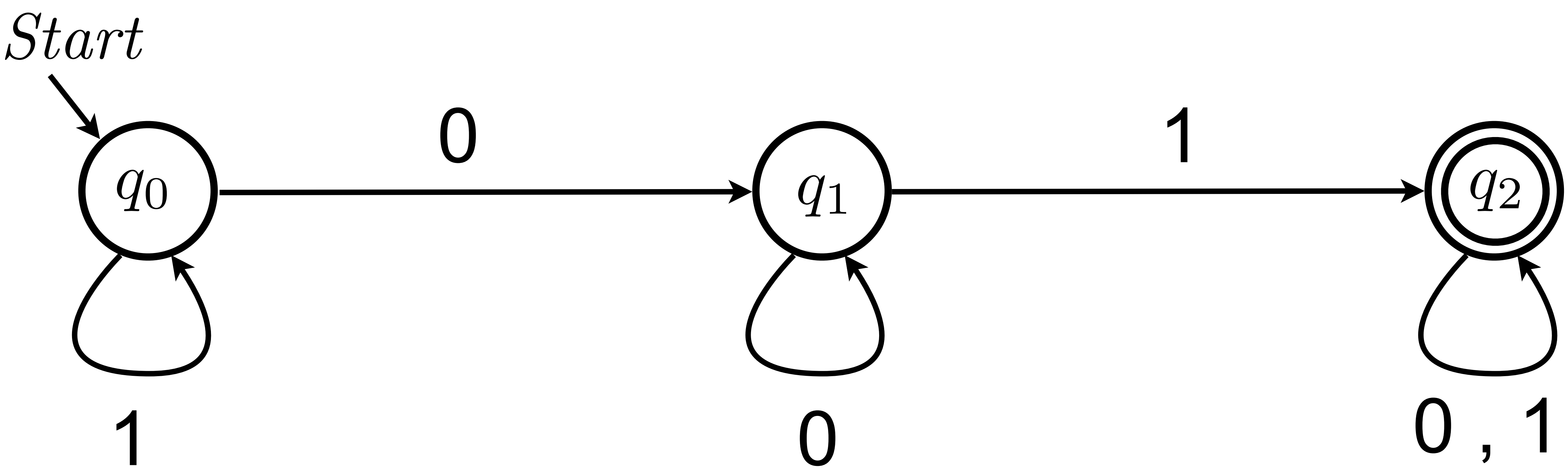
Does it accept 100 ?
Does it accept 011 ?
Does it accept 1010010 ?
What is L(A) ?

DFA: question time



Does it accept 100 ? NO
Does it accept 011 ?
Does it accept 1010010 ?
What is L(A) ?

DFA: question time



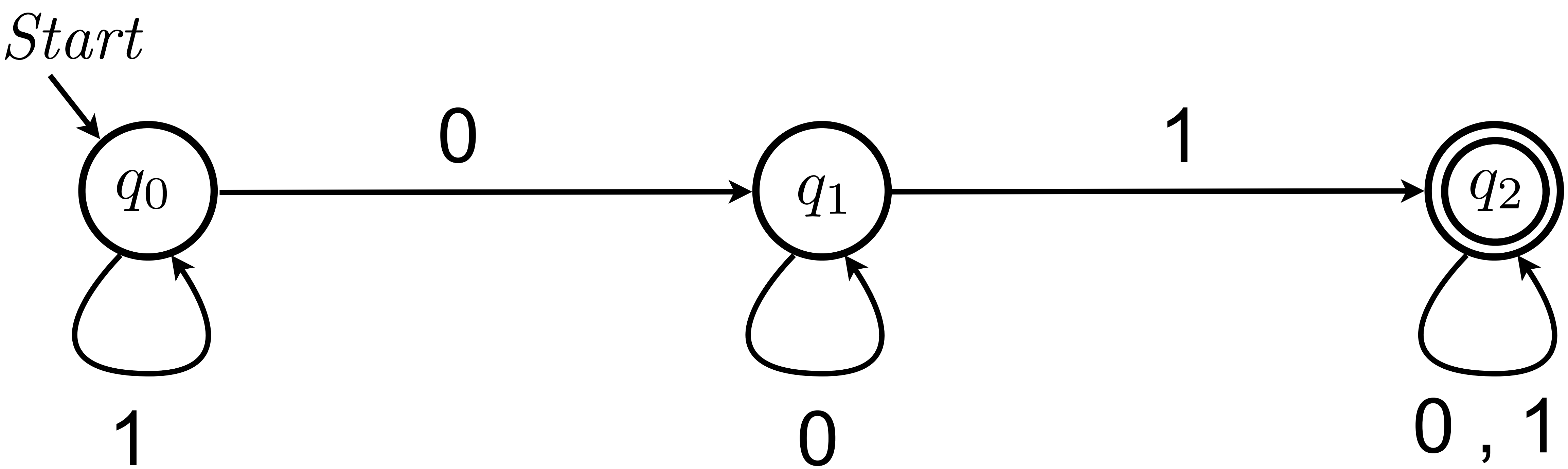
Does it accept 100 ? NO

Does it accept 011 ? YES

Does it accept 1010010 ?

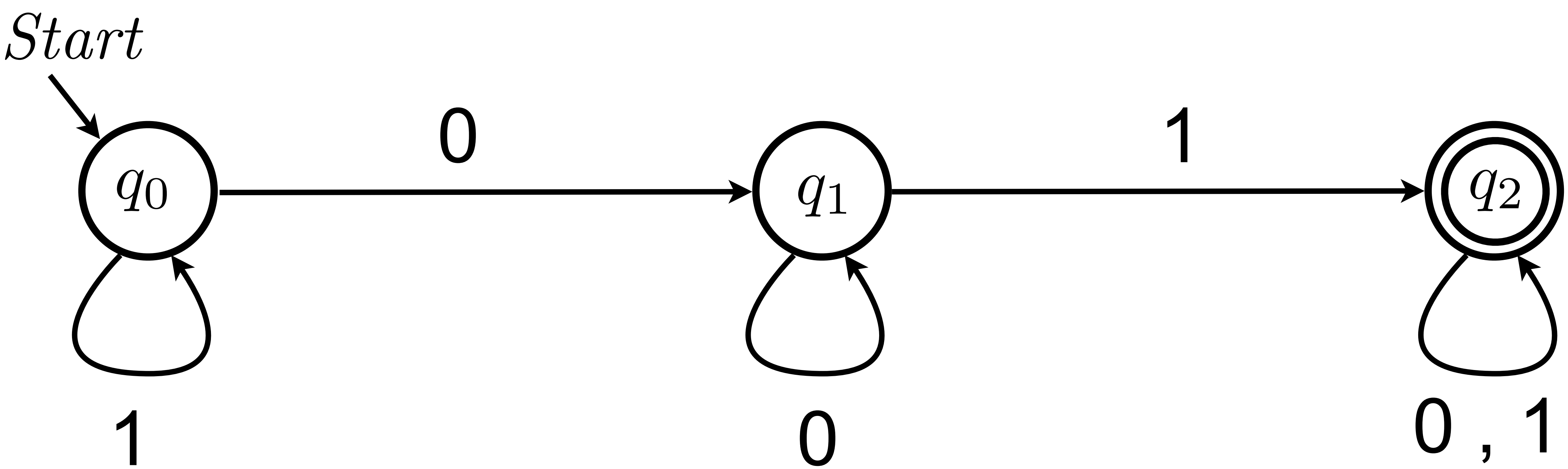
What is L(A) ?

DFA: question time



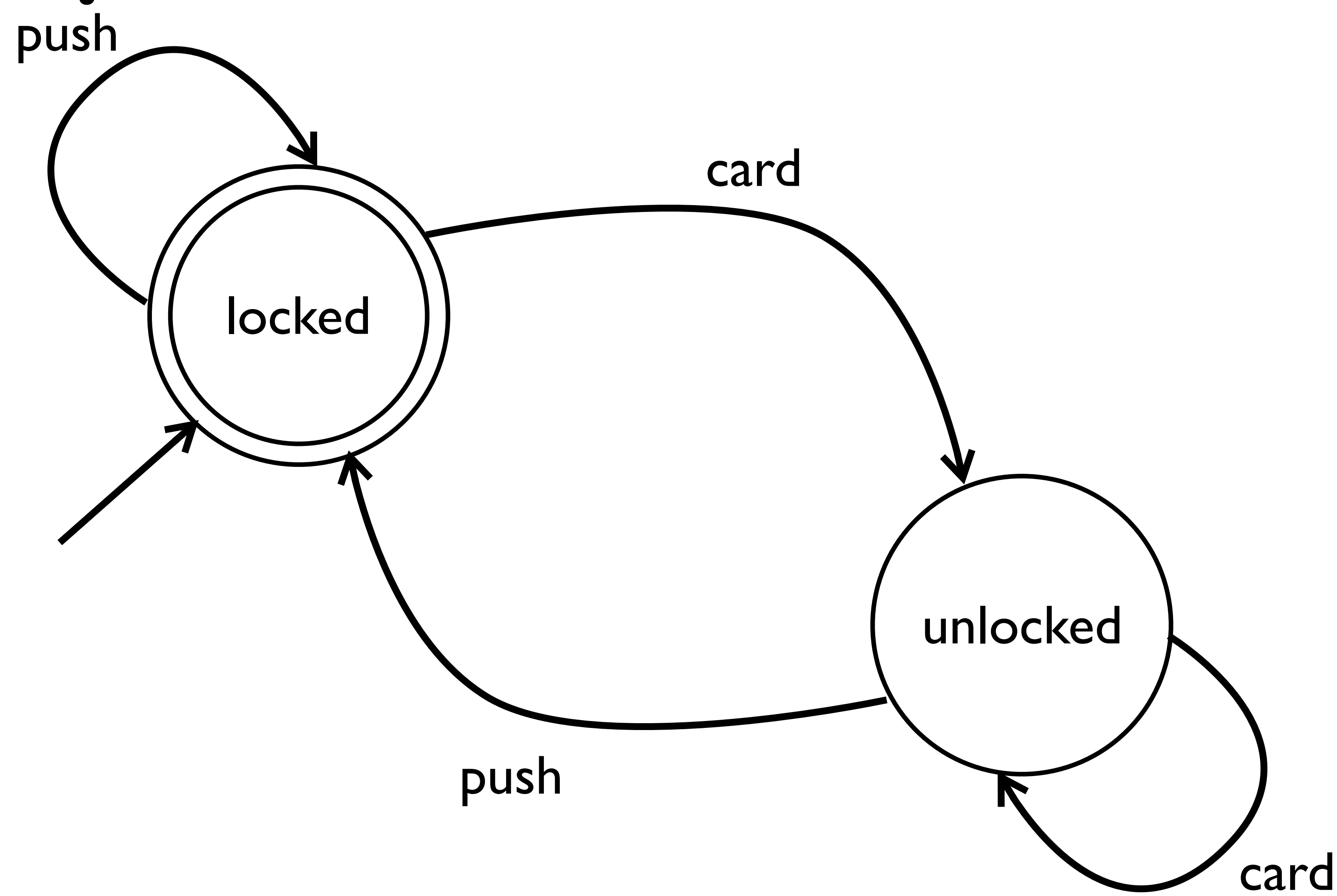
Does it accept 100 ? NO
Does it accept 011 ? YES
Does it accept 1010010 ? YES
What is L(A) ?

DFA: question time



Does it accept 100 ? NO
Does it accept 011 ? YES
Does it accept 1010010 ? YES
What is L(A) ? $\{ \ x01y \ | \ x, y \in \{0, 1\}^* \ \}$

DFA: question time



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

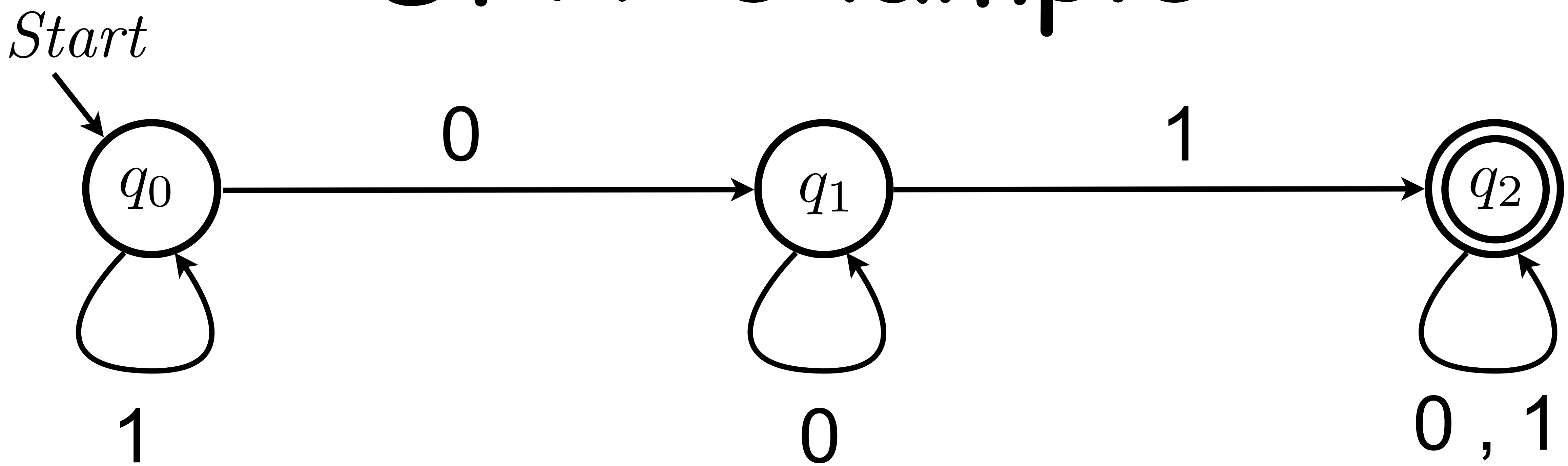
start state decorated with an arrow

all final states decorated with *

Transition table

					$\mathbf{a}$			
$\rightarrow$								
	$\mathbf{q}$				$\delta(q, a)$			
$*$								
$*$								

DFA: example



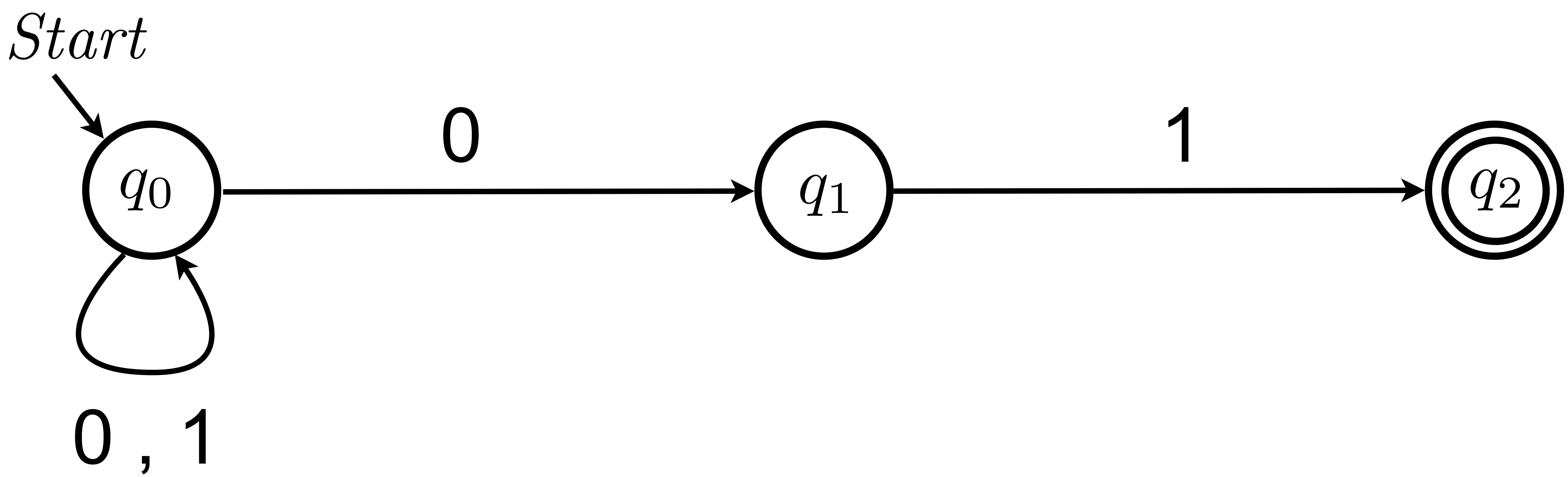
	$\mathbf{0}$	$\mathbf{1}$
$\rightarrow \mathbf{q_0}$		
$\mathbf{q_1}$		
$\star\ \mathbf{q_2}$		

NFA

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

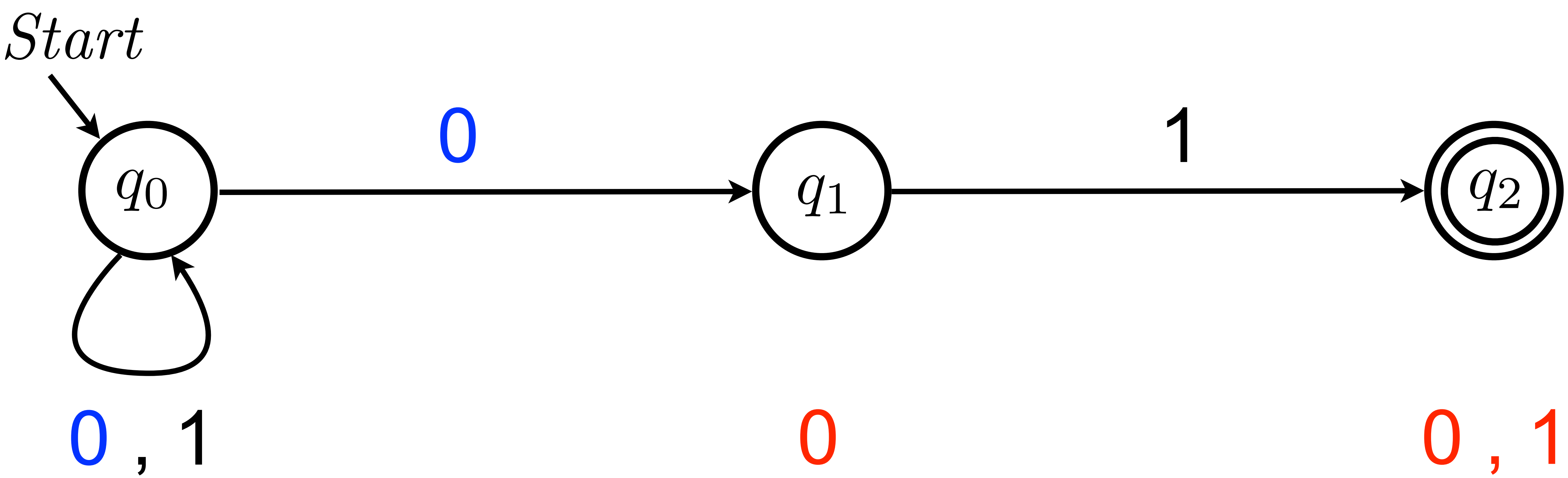
- Q is a finite set of states;
- Σ is a finite set of input symbols;
- $\delta : Q \times \Sigma \to \wp(Q)$ is the transition function; powerset of Q = set of sets over Q
- $q_0 \in Q$ is the initial state (also called start state);
- $F \subseteq Q$ is the set of final states (also accepting states)

NFA: example



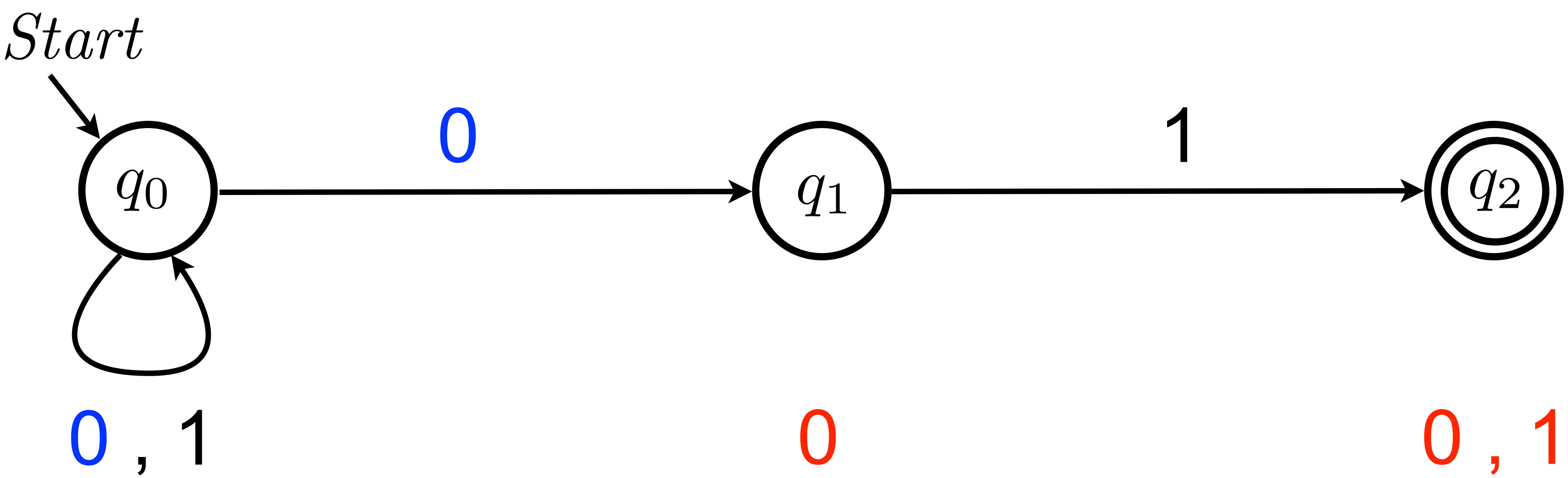
Can you explain why it is not a DFA?

NFA: example



Can you explain why it is not a DFA?

NFA: example



$$\delta(q_0, 0) = \{q_0, q_1\}$$
$$\delta(q_0, 1) = \{q_0\}$$

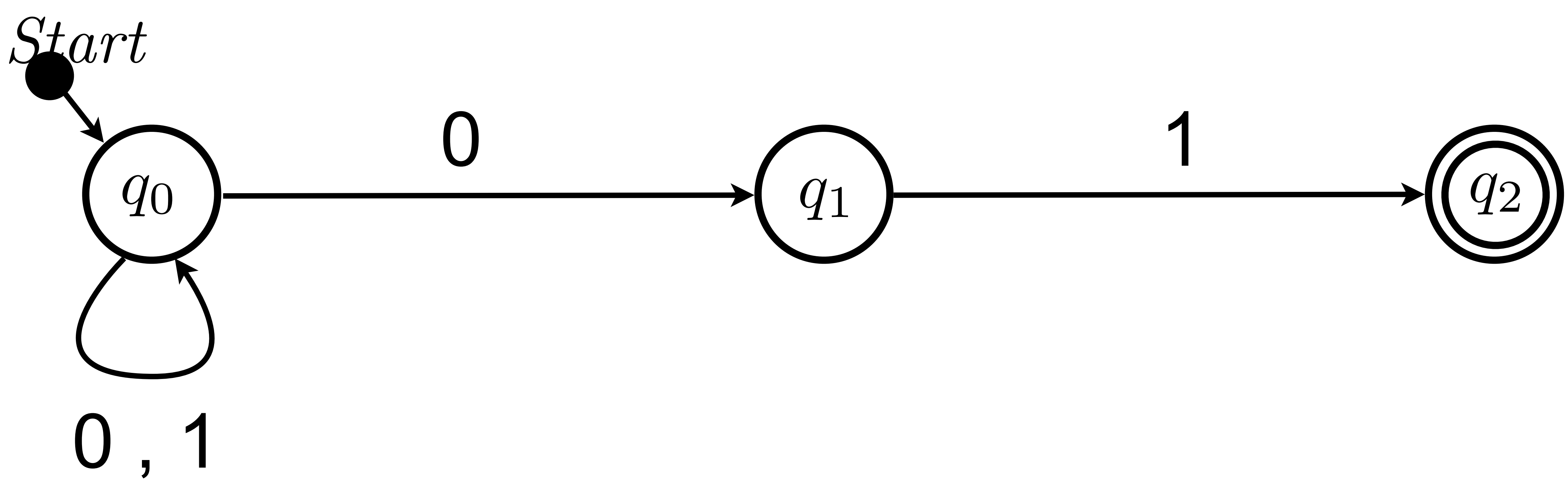
$$\delta(q_1, 0) = \emptyset$$
$$\delta(q_1, 1) = \{q_2\}$$

$$\delta(q_2, 0) = \delta(q_2, 1) = \emptyset$$

Can you explain why it is not a DFA?

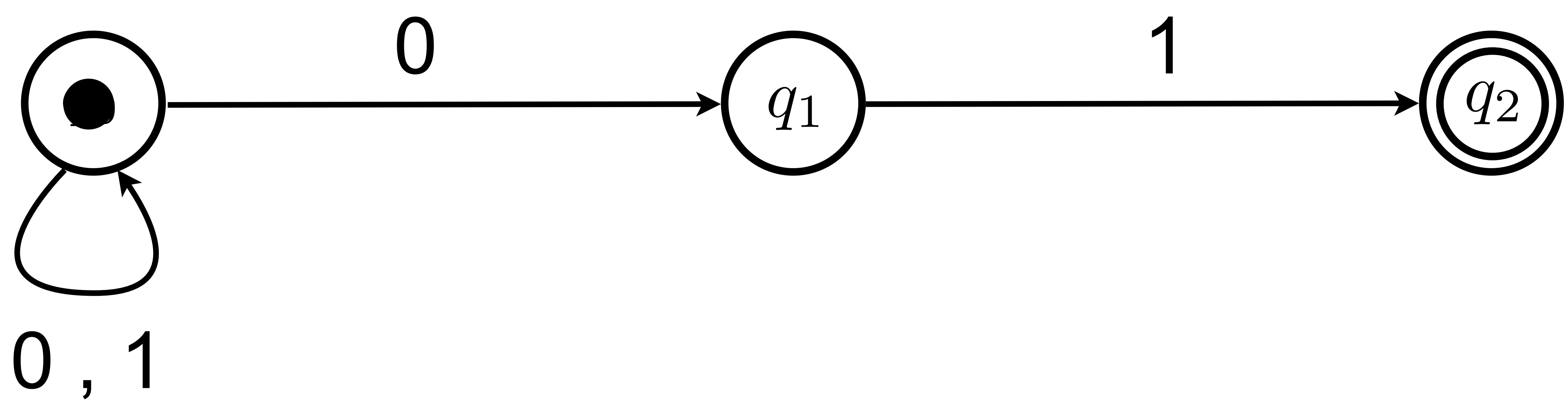
Reshaping

Step 1: get a token

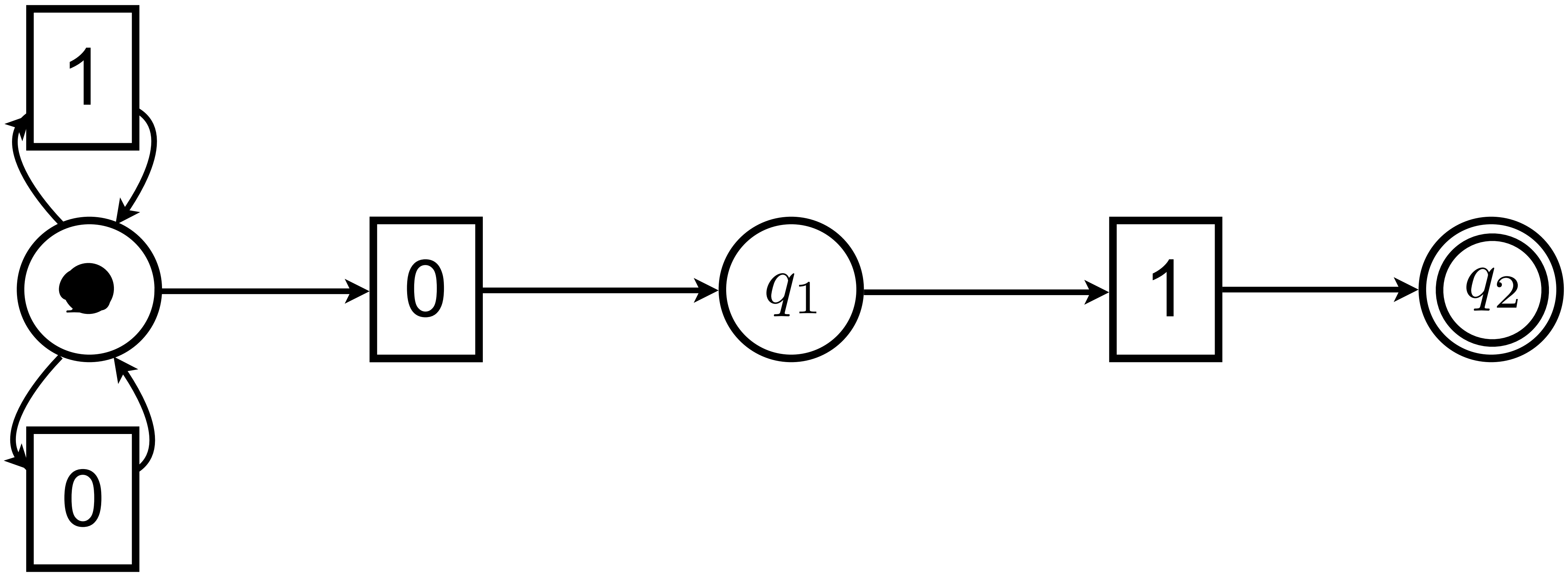


0	1	0	0	1

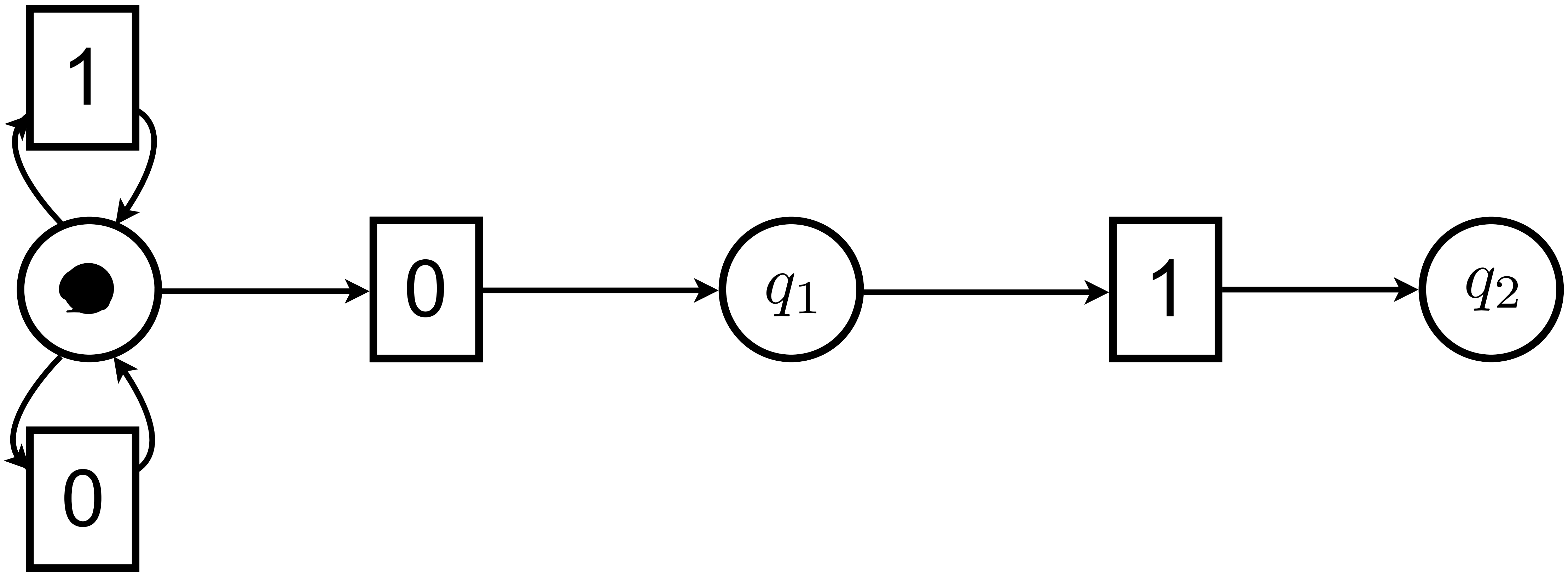
Step 2: forget initial state decoration



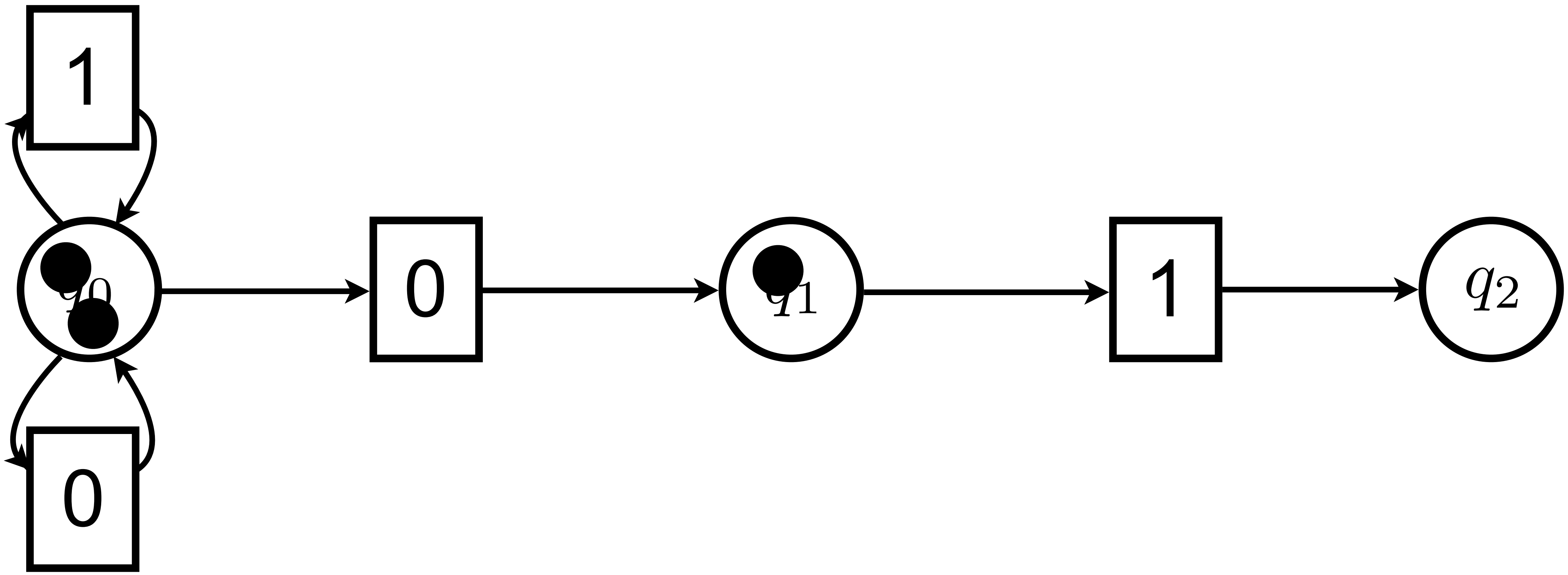
Step 3: transitions as boxes



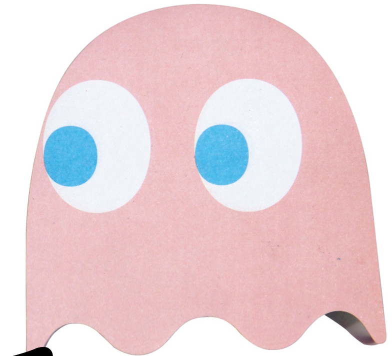
Step 4: forget final states

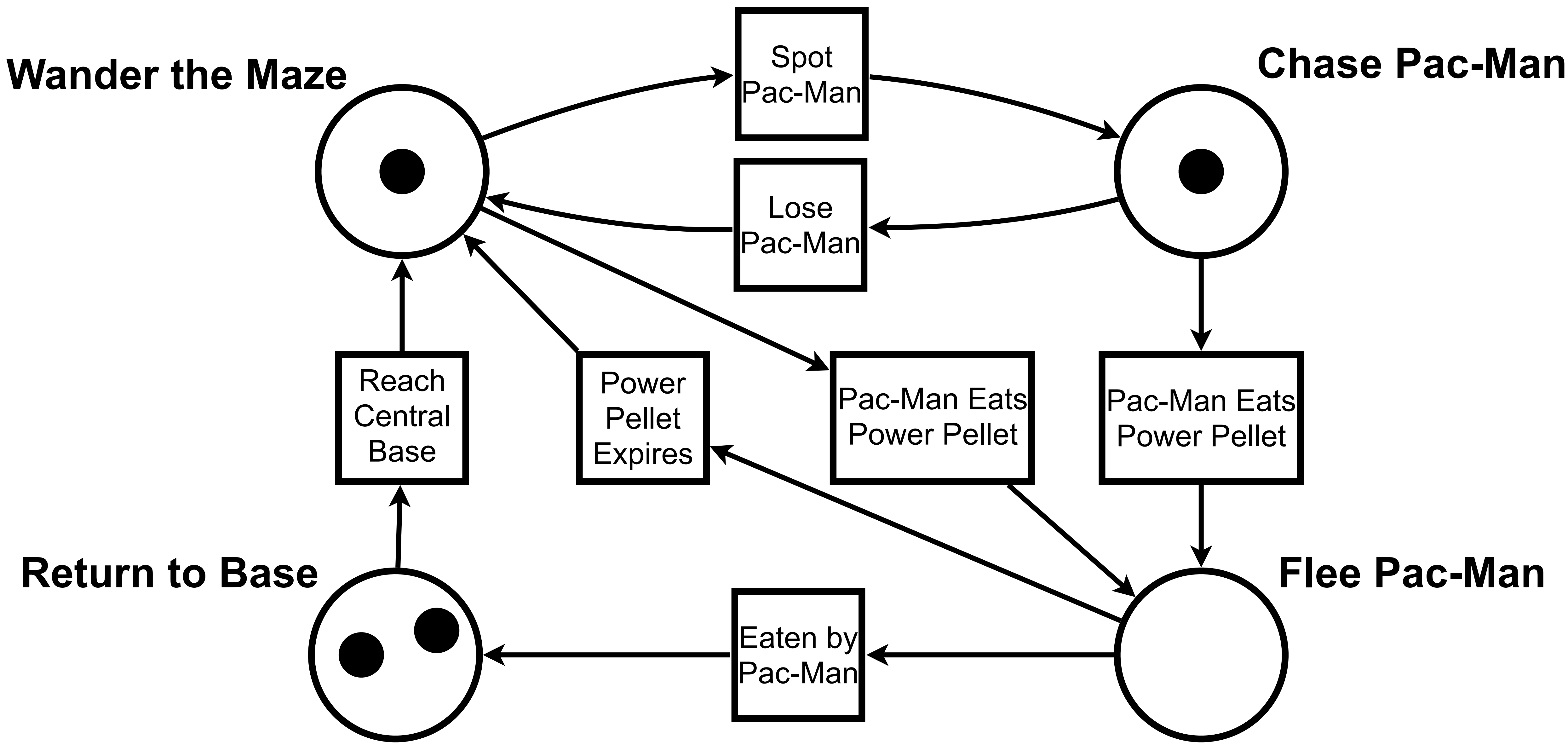


Step 5: allow for more tokens

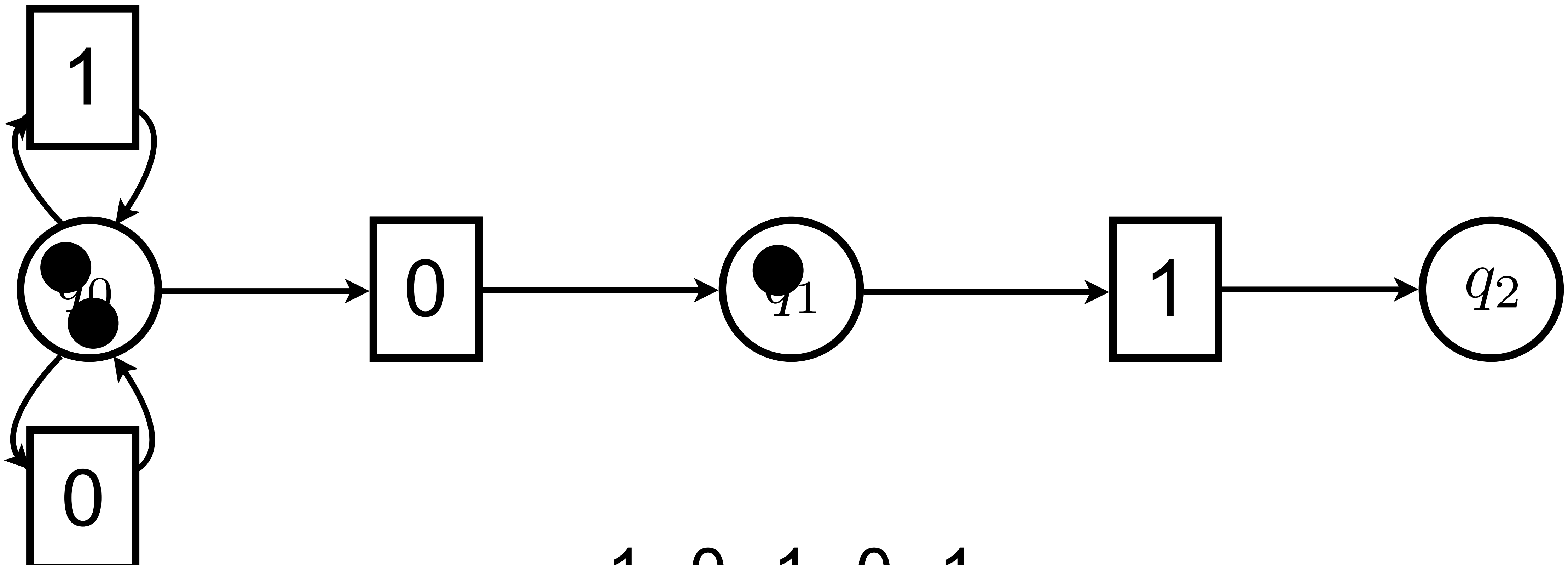


Example: Four Pac-Man Ghosts



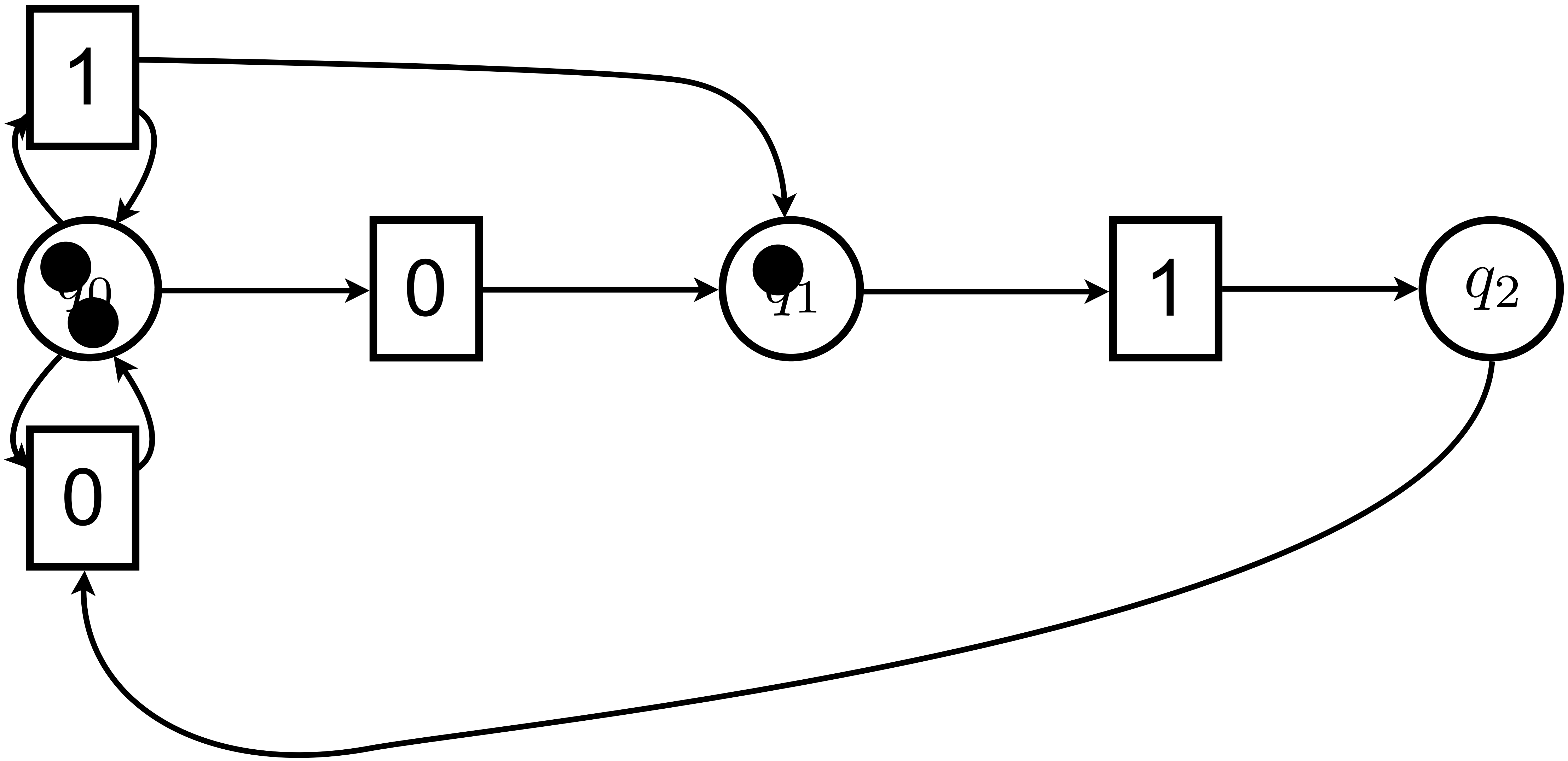


Example: token game

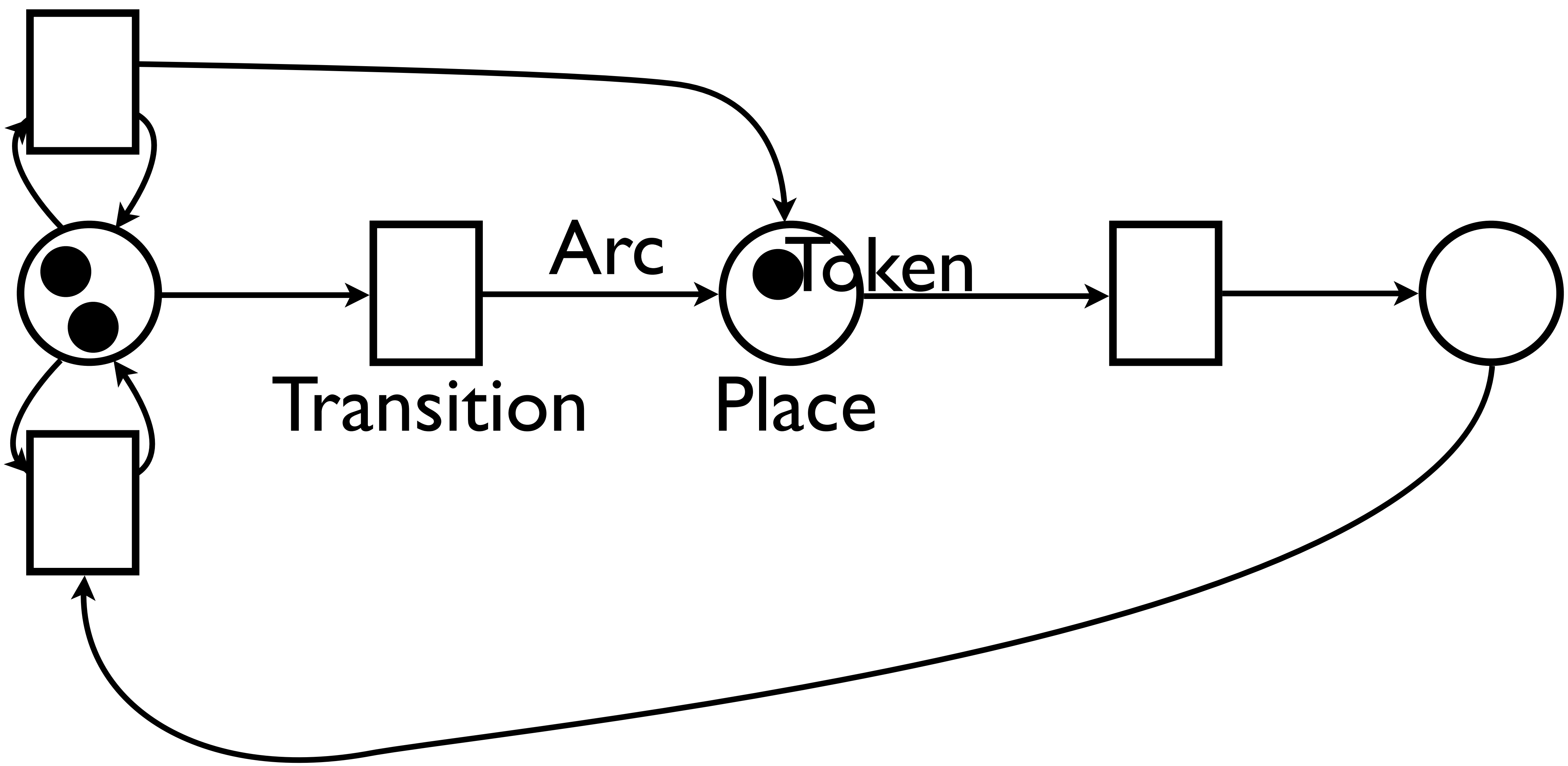


1 0 1 0 1

Step 6: allow for more arcs



Terminology



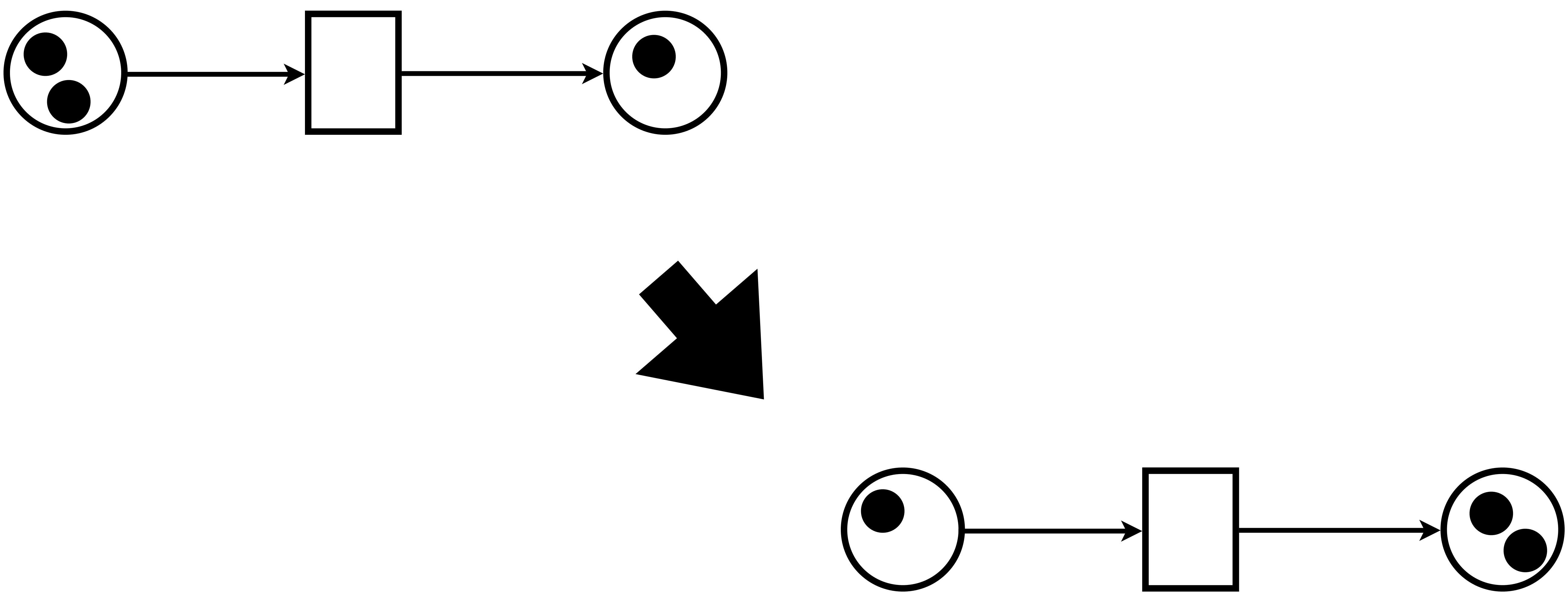
Some facts

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

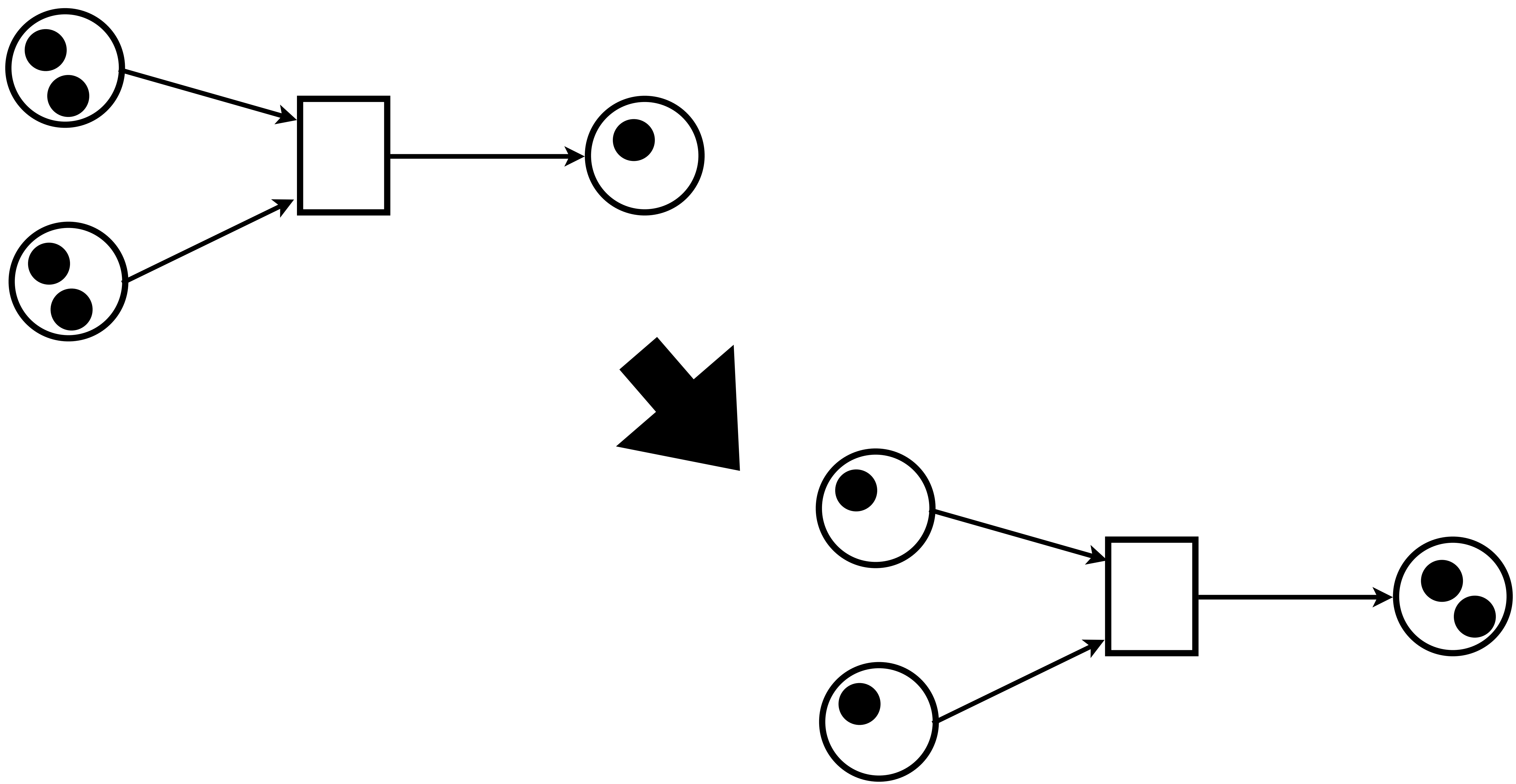
Static structure for dynamic systems:
places, transitions, arcs do not change
tokens move around places

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

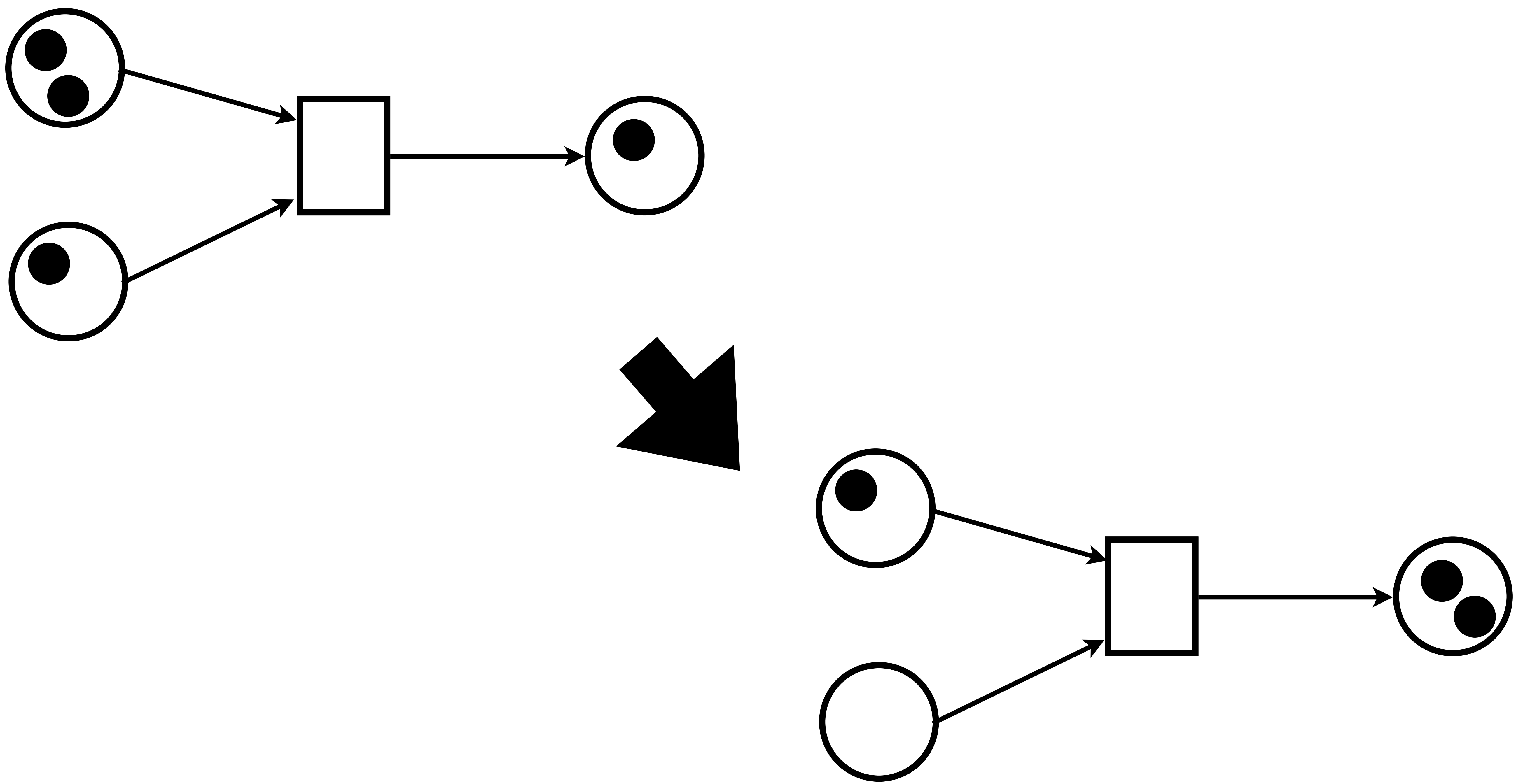
Token game: example



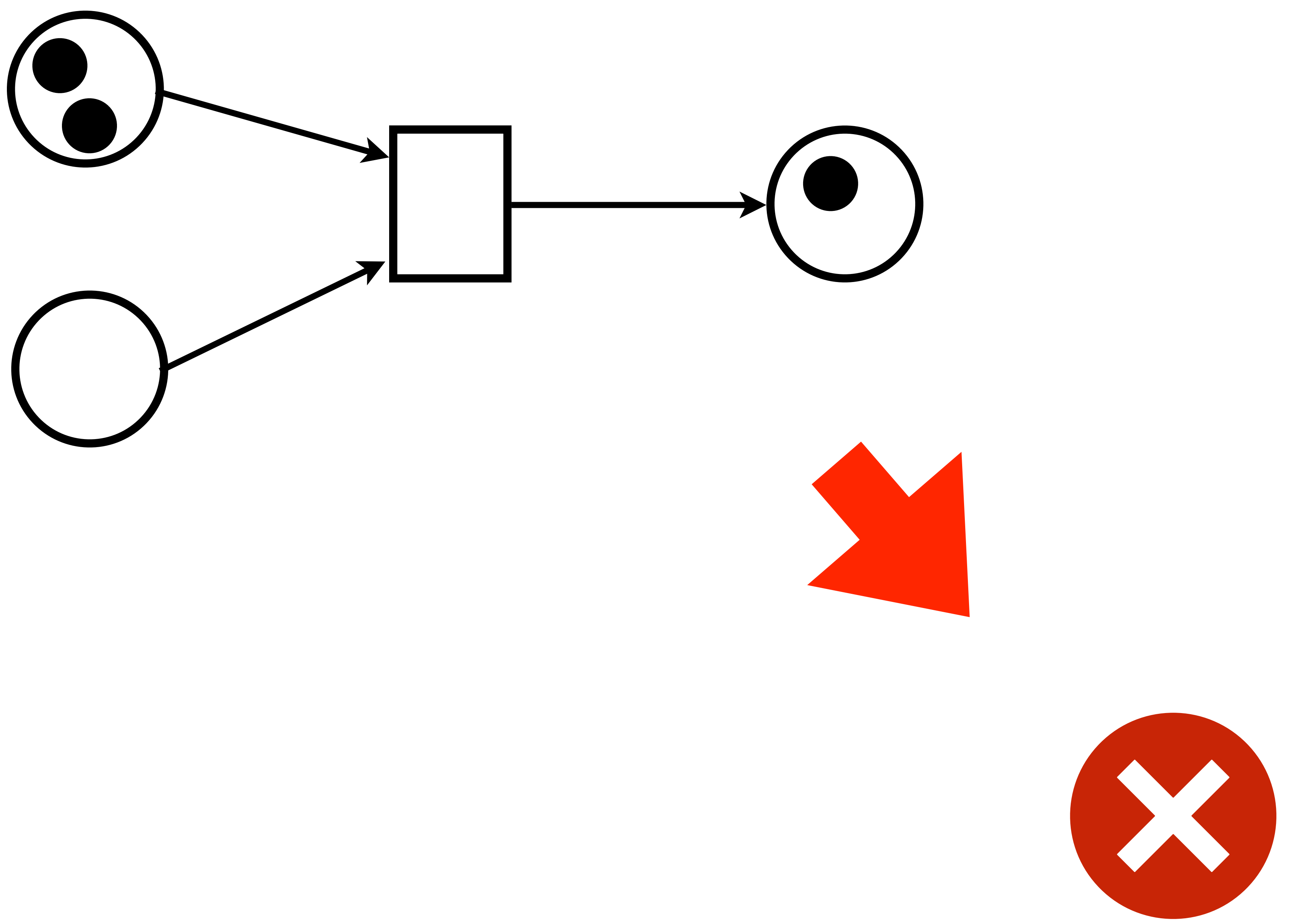
Token game: example



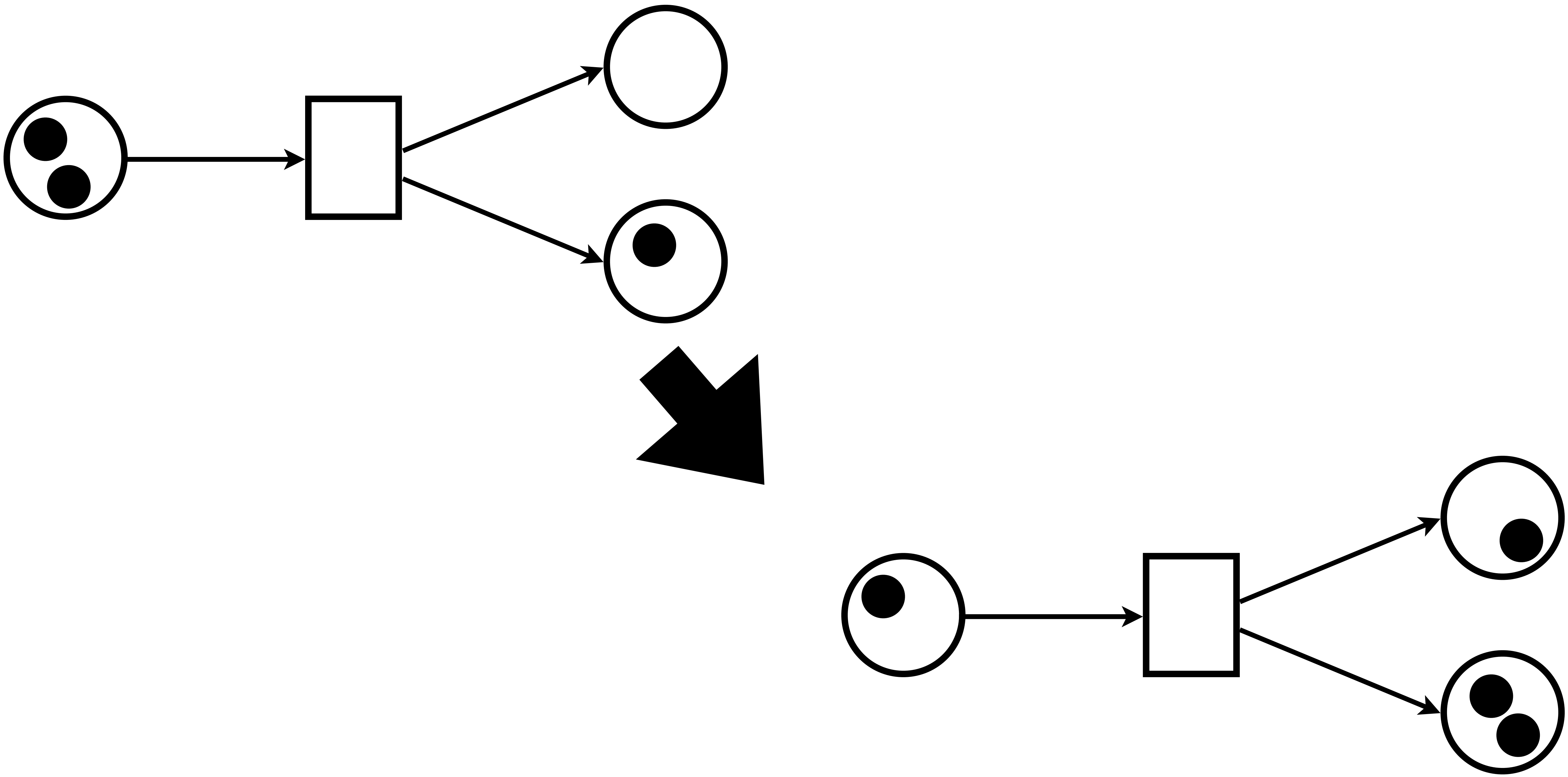
Token game: example



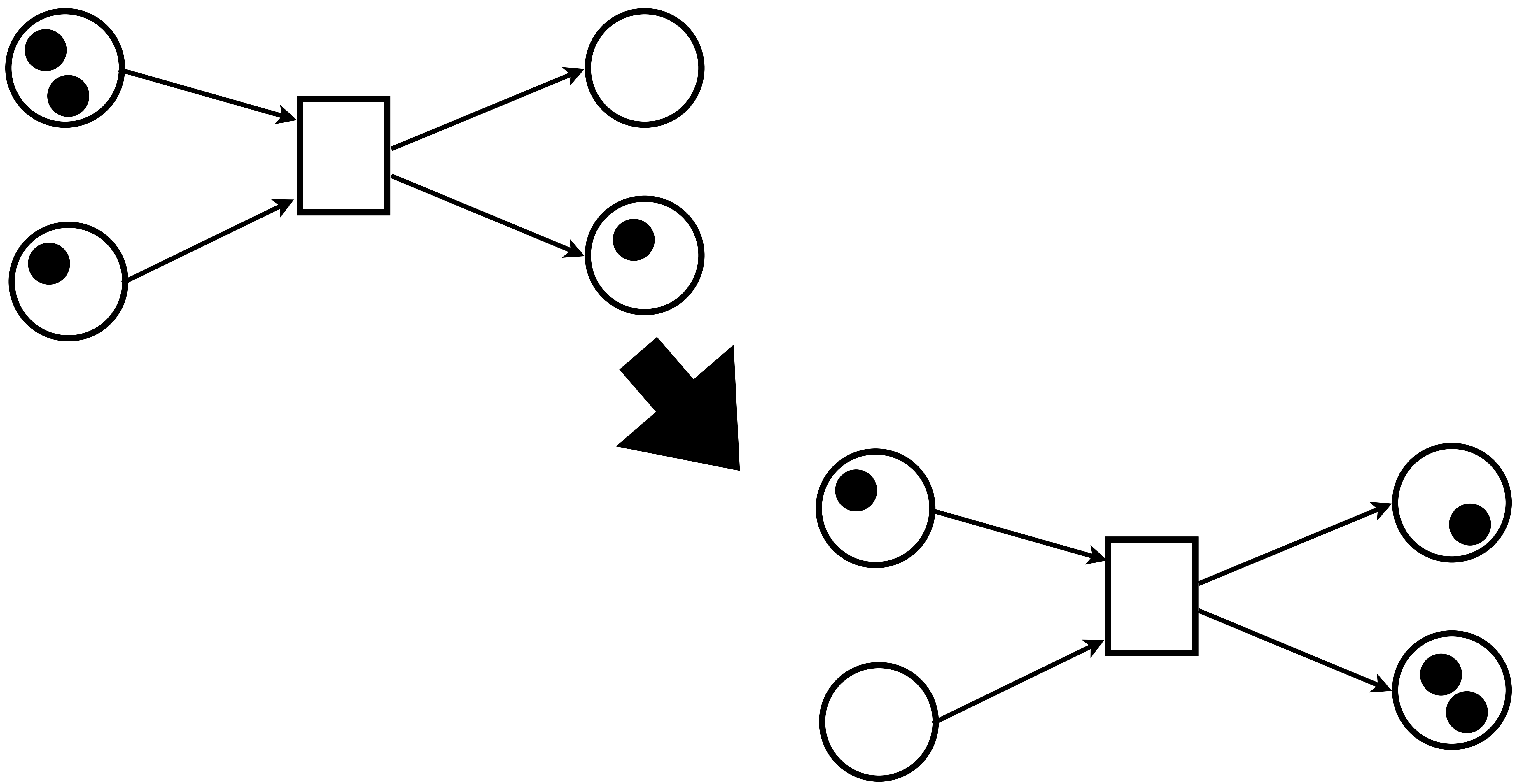
Token game: example



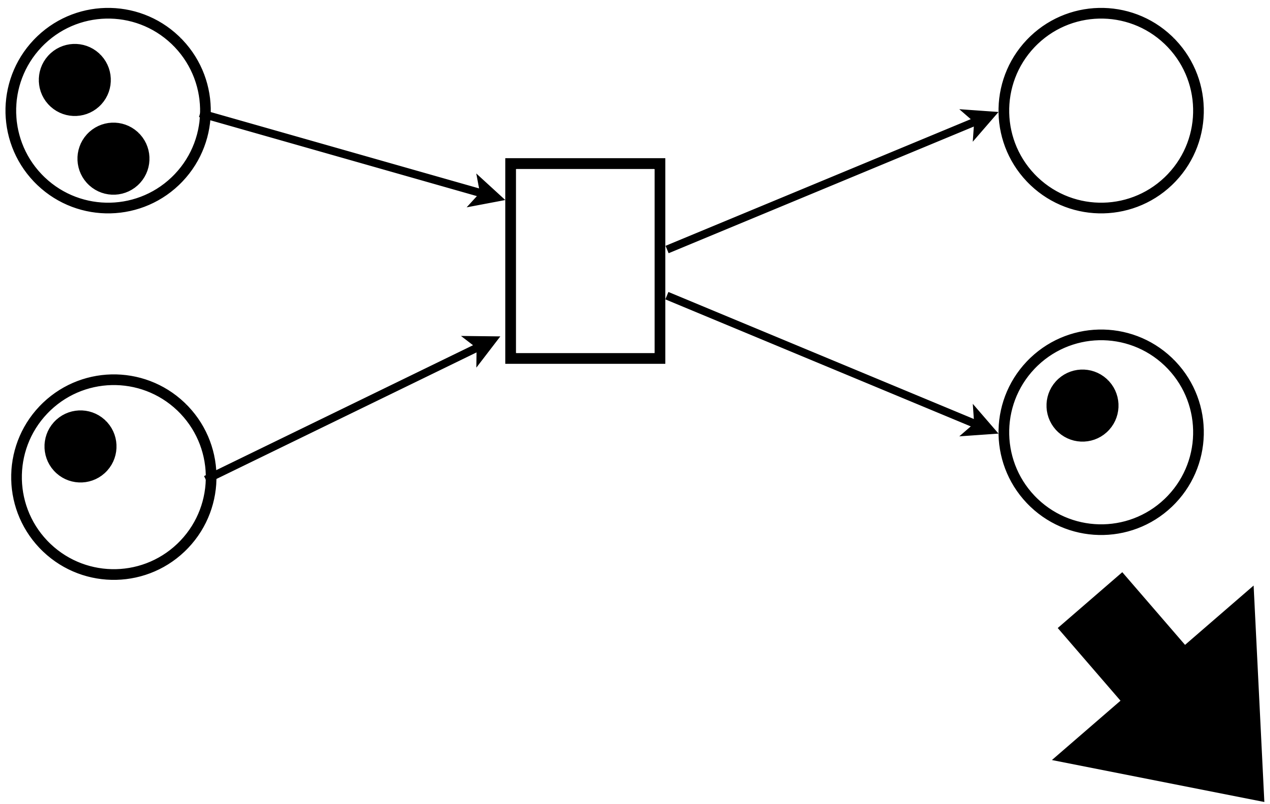
Token game: example



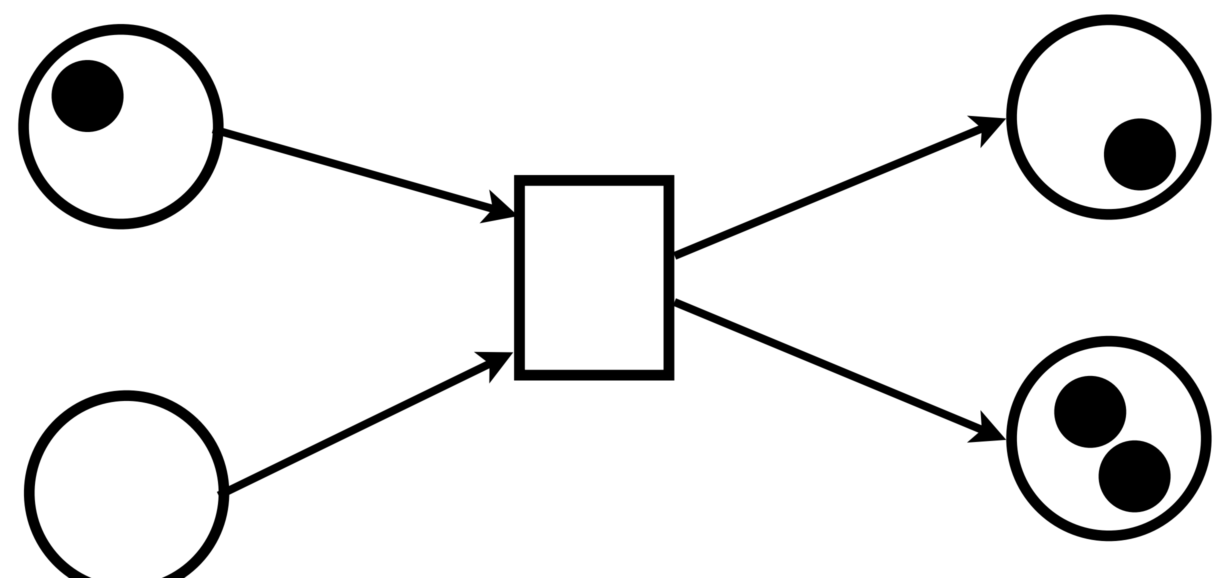
Token game: example



Token game: firing rule

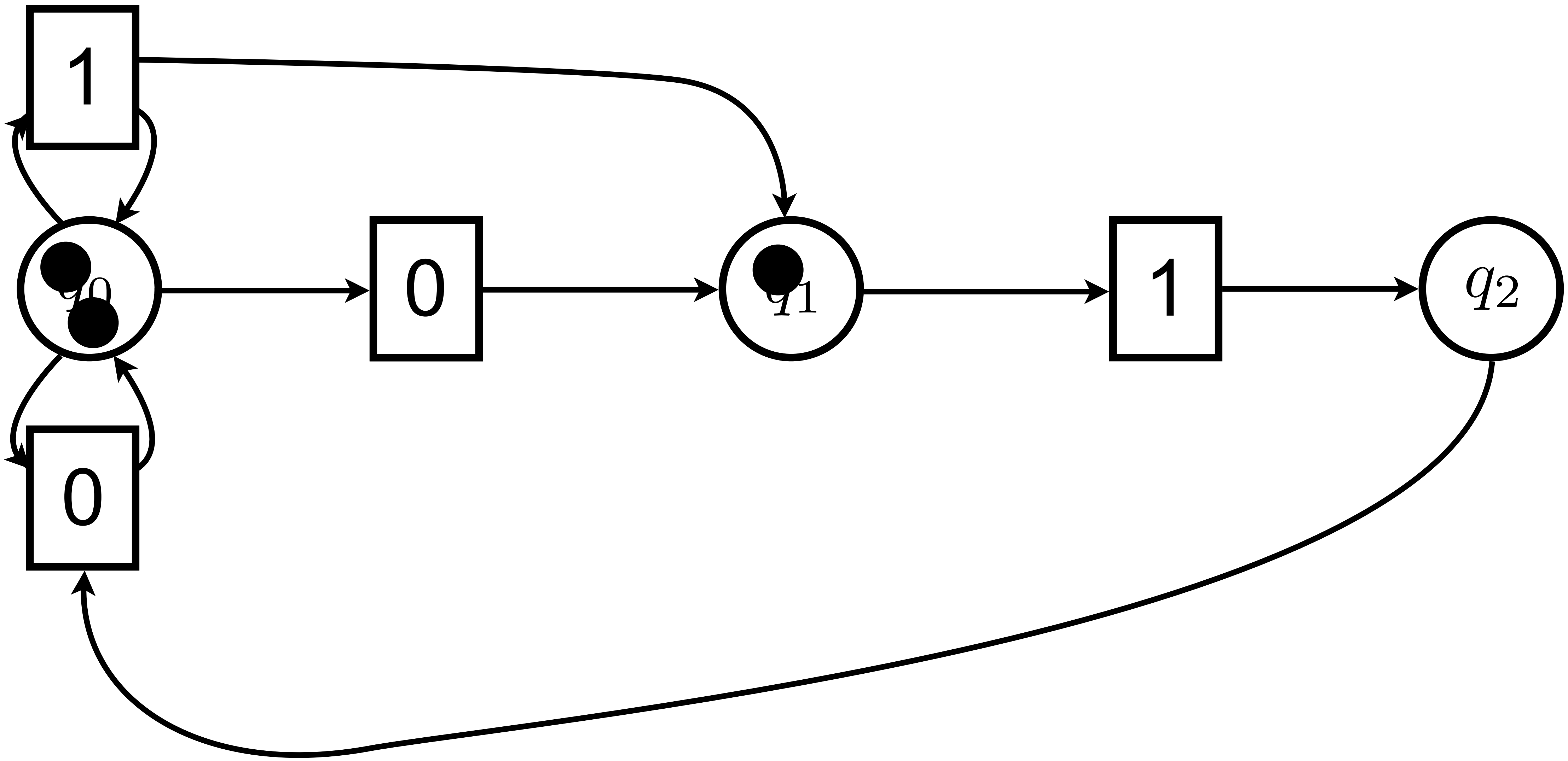


Collect one token from each “input” place

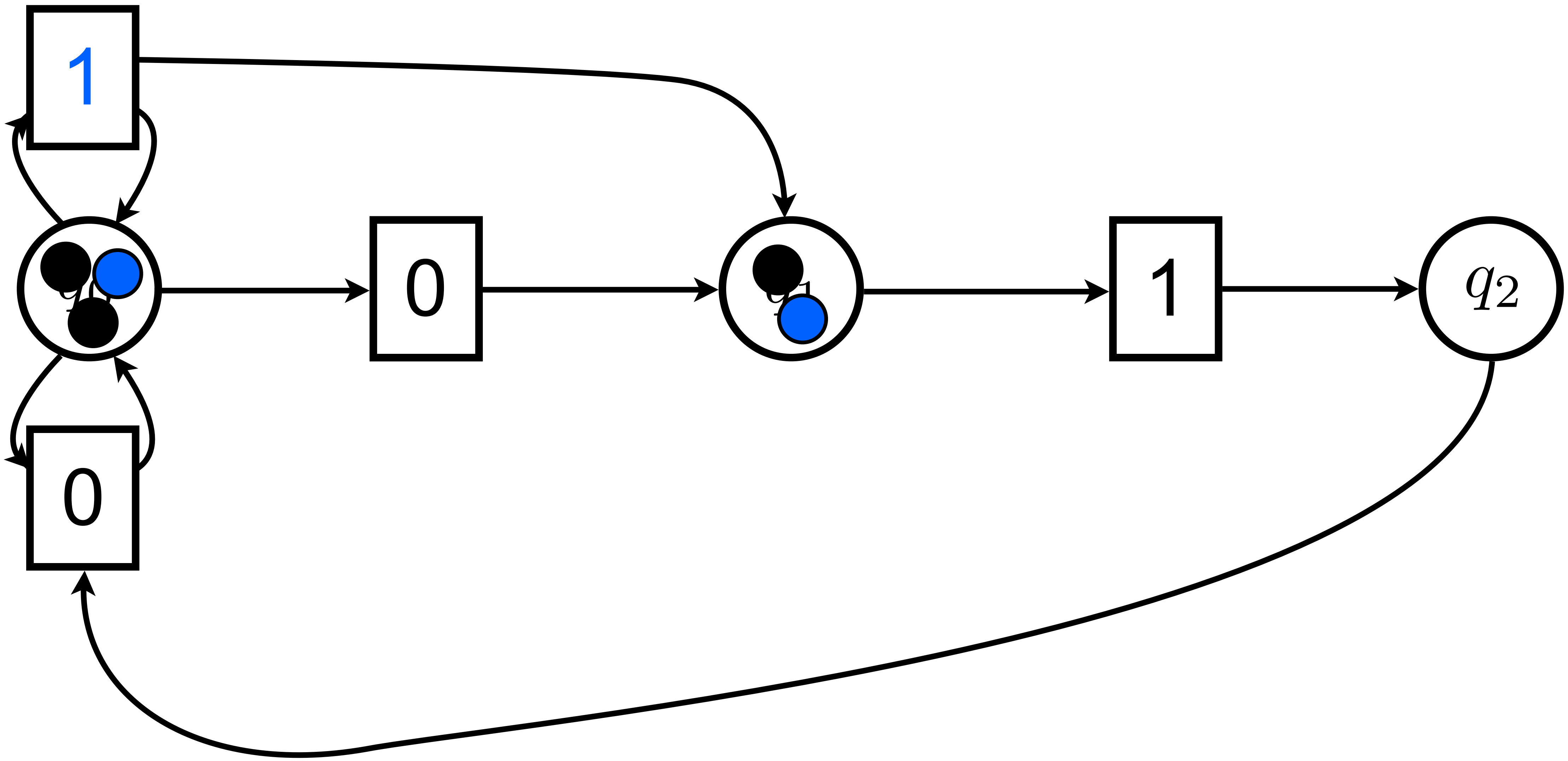


Produce one token into each “output” place

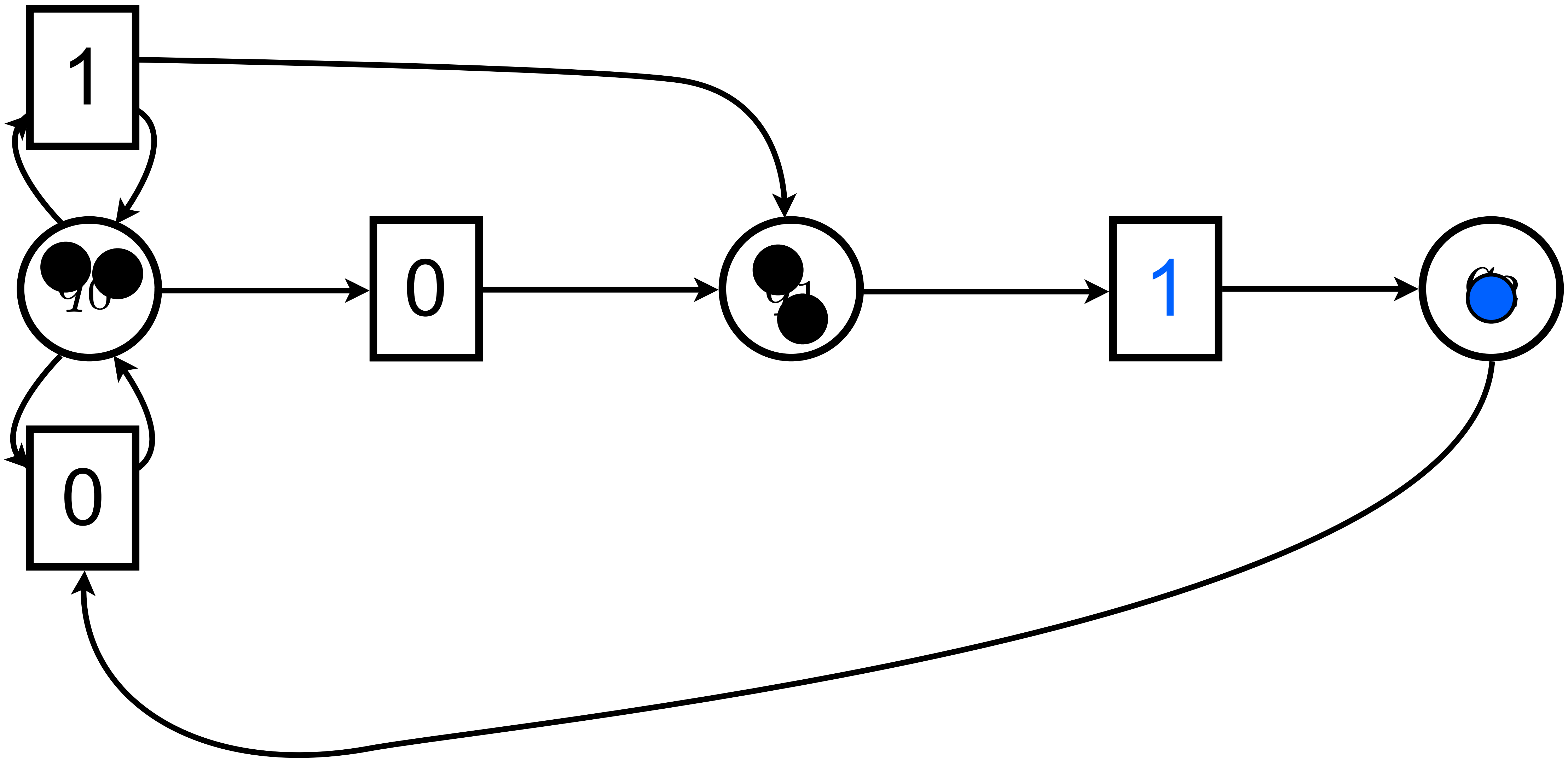
Example: token game



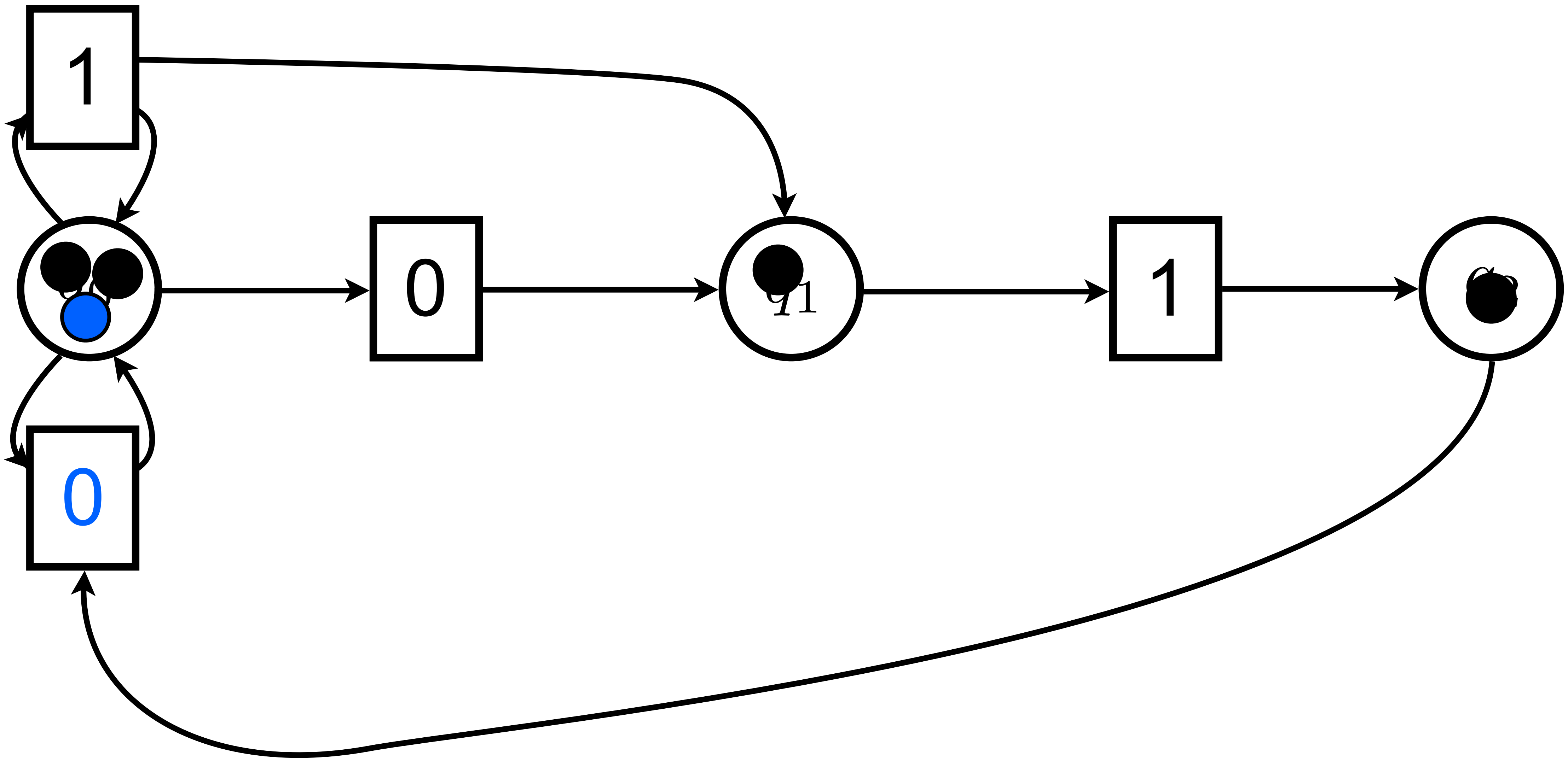
Example: token game



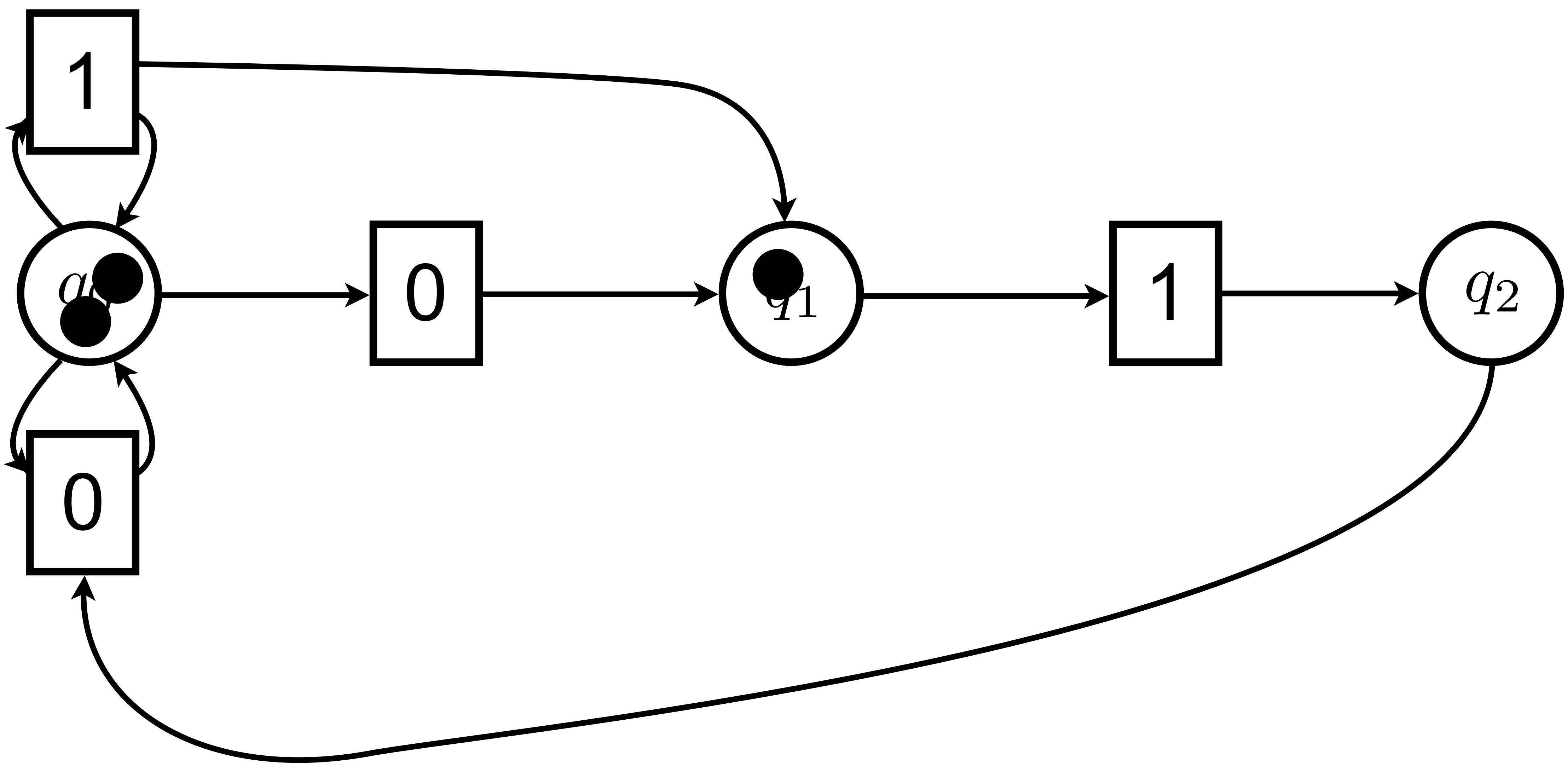
Example: token game



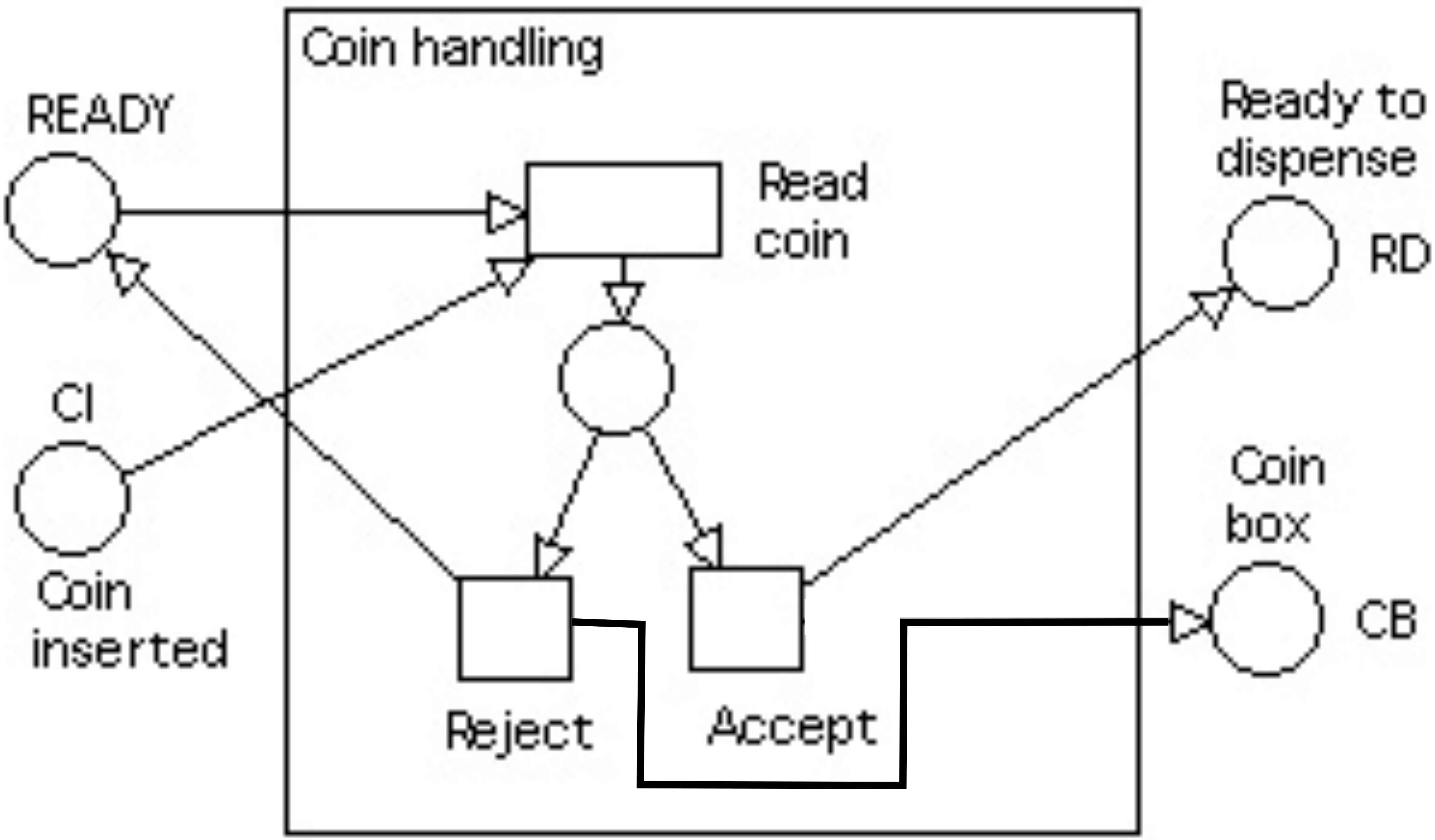
Example: token game



Example: token game



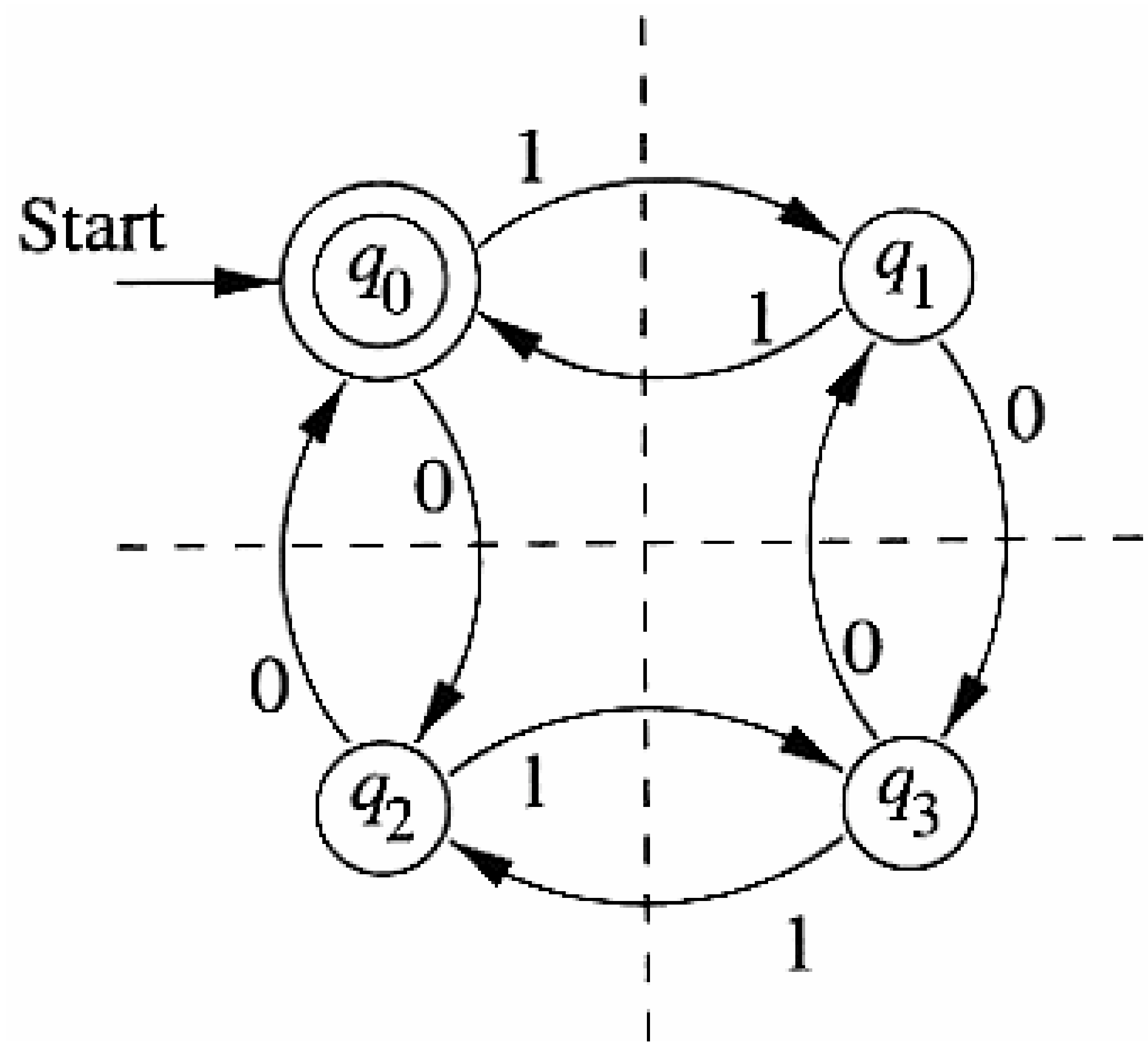
Example: Coin Handling



Exercises

Choose your favourite (video) game, and draw the finite state automaton for one of the characters in that game

DFA: exercise



Does it accept 100 ?

Write its transition table.

Does it accept 1010 ?

What is L(A) ?