# Reversible Computing Systems, Logic Circuits, and Cellular Automata

**Kenichi Morita**
**Hiroshima University**

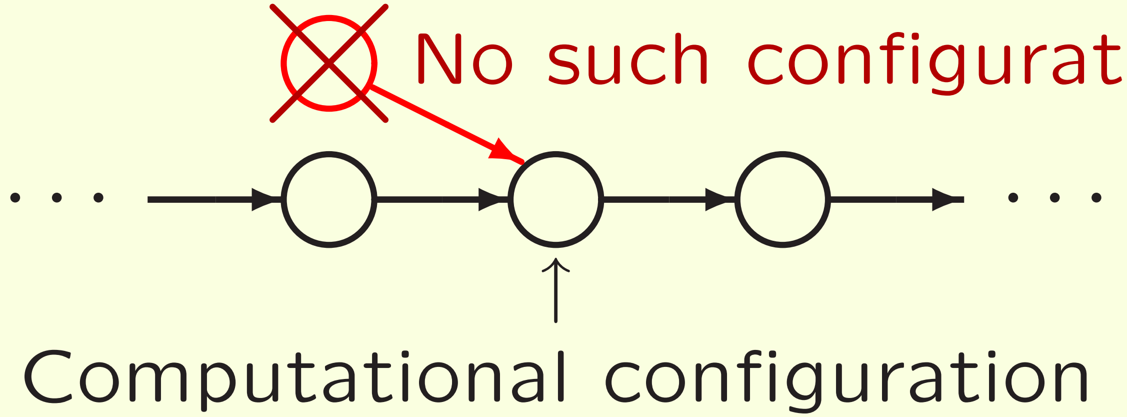
The Third International Conference on Networking and Computing (ICNC'12), Okinawa, December 5, 2012

## Contents

## 1. Introduction:
## What is reversible computing?

## Reversible computing

- Roughly speaking, it is a "backward deterministic" computing; i.e., every computational configuration has at most one predecessor.



- Though its definition is rather simple, it reflects physical reversibility well.

## Why reversible computing?

- Reversibility is one of the fundamental microscopic physical laws of Nature.
- It is important to investigate how computation can be carried out in a reversible system.
- It relates to energy consumption in computing, and to quantum computing.
- We shall see that in spite of the reversibility constraint, they have high capability of computing.

## Hierarchical structure in reversible computing

| Levels | Examples of computing models |
|---|---|
| Universal computing system | Reversible Turing machines<br>Reversible counter machines<br>Reversible cellular automata |
| Finite state machine | Reversible sequential machines |
| Logic circuit | Reversible logic circuits |
| Logic element | Reversible logic gates<br>Reversible logic elements with memory |
| Physical system | Billiard ball model |

- Any reversible system in the higher level can be realized by one in the lower level.
- Hereafter, we will show what these models are, and how they are related and constructed.

## 2. Reversible logic elements and circuits

## Reversible logic elements

A logic element whose function is described by a one-to-one mapping.

**(1) Reversible logic elements without memory (i.e., reversible logic gates):**

- Toffoli gate [Toffoli, 1980]
- Fredkin gate [Fredkin and Toffoli, 1982]
- etc.

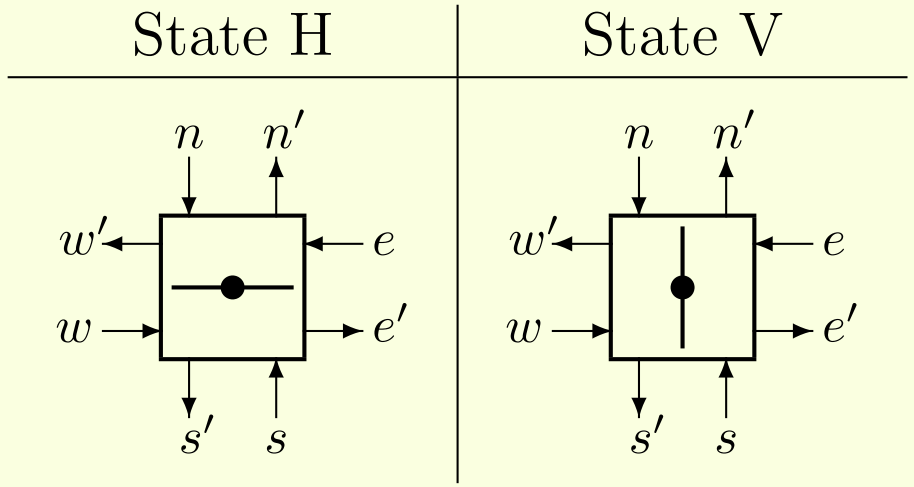
**(2) Reversible logic elements with memory (RLEMs):**

- Rotary element (RE) [Morita, 2001]
- $m$-state $n$-symbol RLEM (in general)

## Reversible logic elements with memory

**Rotary element (RE)** [Morita, 2001]

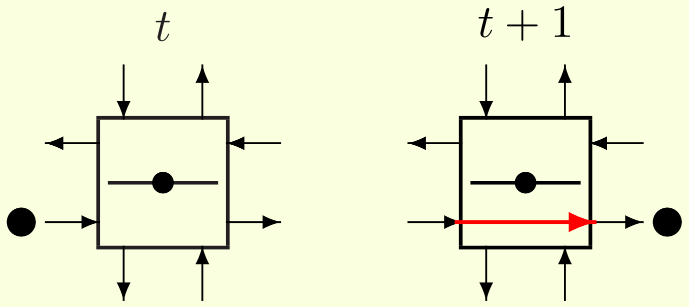
- A 2-state 4-input-line 4-output-line element.



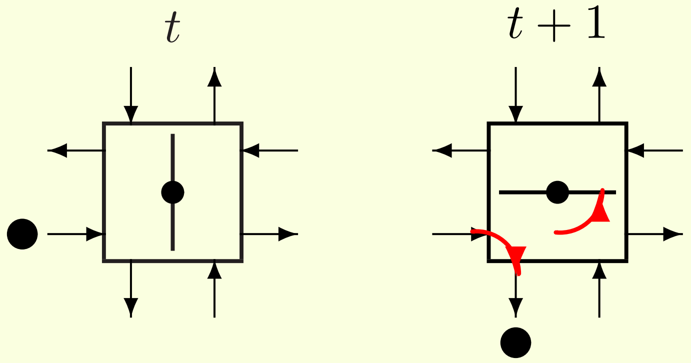
- Conceptually, it has a rotatable bar as above.
- Assume signal 1 is given at most one input line.

## Operations of a rotary element (RE)

- Its operations are very easily understood.
- The bar in the box controls an incoming signal.
- Parallel case:



- Orthogonal case:



## Reversible sequential machine (RSM)

$$M = (Q, \Sigma, \Gamma, \delta)$$

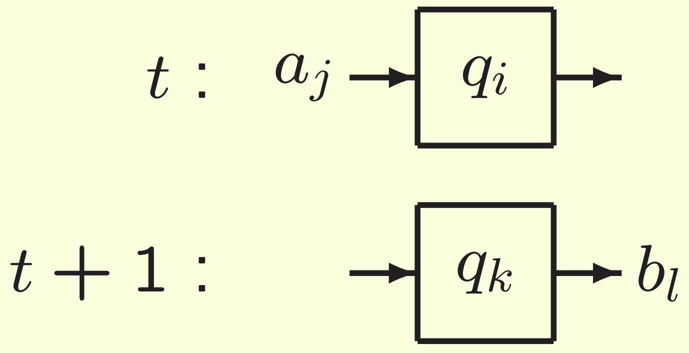
$Q$ : a set of states

$\Sigma$ : a set of input symbols

$\Gamma$ : a set of output symbols

$$\delta : \underbrace{Q}_{\text{present state}} \times \underbrace{\Sigma}_{\text{input}} \rightarrow \underbrace{Q}_{\text{next state}} \times \underbrace{\Gamma}_{\text{output}} : \text{a move function}$$

$$\delta(q_i, a_j) = (q_k, b_l)$$



- $M$ is called an RSM, if $\delta$ is one-to-one.

## RE is a 2-state 4-symbol RLEM

$$M_{\mathrm{RE}} = \Big(\ \underbrace{\{\text{⊟}, \text{◫}\}}_{\text{internal states}},\ \underbrace{\{n, e, s, w\}}_{\text{input symbols}},\ \underbrace{\{n', e', s', w'\}}_{\text{output symbols}},\ \underbrace{\delta_{\mathrm{RE}}}_{\text{move function}}\Big)$$

The move function $\delta_{\mathrm{RE}}$:

| Present state | Input | | | |
|---|---|---|---|---|
| | $n$ | $e$ | $s$ | $w$ |
| State H: ⊟ | ◫ $w'$ | ⊟ $w'$ | ◫ $e'$ | ⊟ $e'$ |
| State V: ◫ | ◫ $s'$ | ⊟ $n'$ | ◫ $n'$ | ⊟ $s'$ |

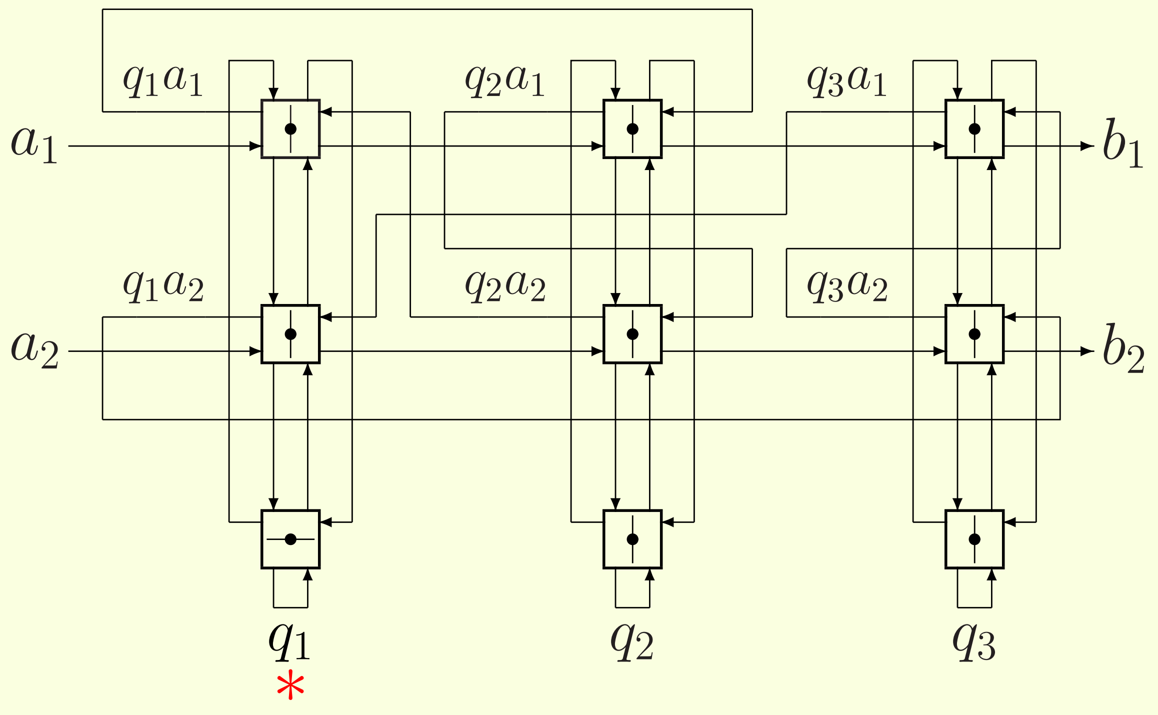
## Any RSM can be constructed by REs

**Theorem 1** [Morita, 2003] Any RSM can be simulated by a circuit composed only of REs. Thus, an RE is universal in this sense.

## An example of an RSM constructed by REs

**Example:**

| Input | State | | |
|---|---|---|---|
| | $q_1$ | $q_2$ | $q_3$ |
| $a_1$ | $q_2\ b_1$ | $q_2\ b_2$ | $q_1\ b_2$ |
| $a_2$ | $q_3\ b_2$ | $q_1\ b_1$ | $q_3\ b_1$ |

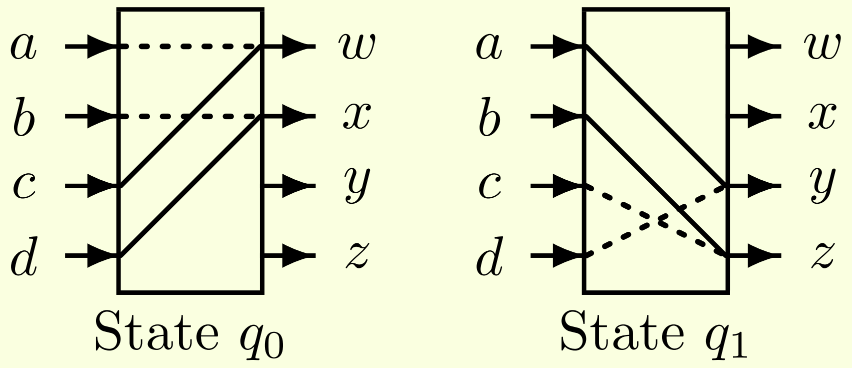


[Morita, 2003]

## We represent a 2-state RLEM graphically

An example: No. 4-289 (equivalent to an RE)

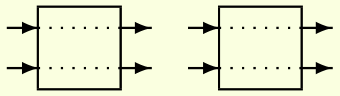
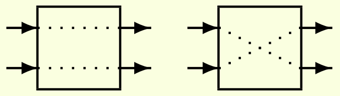
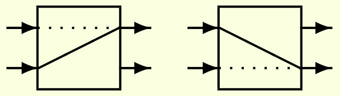
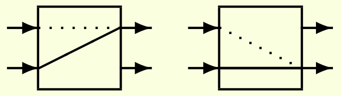
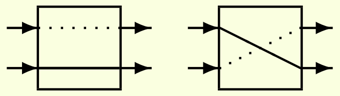
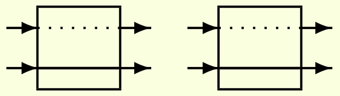
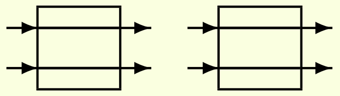
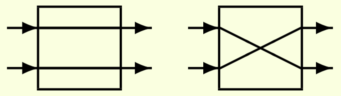
| Present state | Input | | | |
|---|---|---|---|---|
| | $a$ | $b$ | $c$ | $d$ |
| State $q_0$ | $q_0$ $w$ | $q_0$ $x$ | $q_1$ $w$ | $q_1$ $x$ |
| State $q_1$ | $q_0$ $y$ | $q_0$ $z$ | $q_1$ $z$ | $q_1$ $y$ |



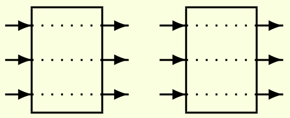
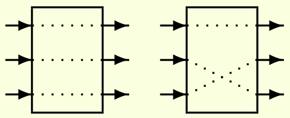
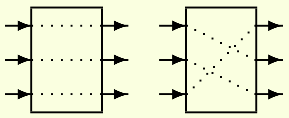
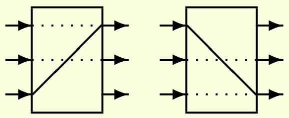
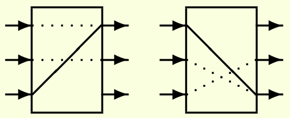
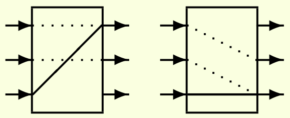
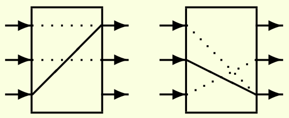
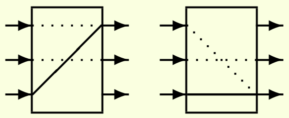
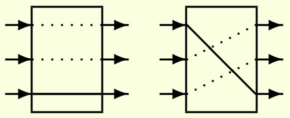
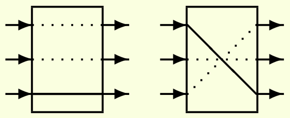
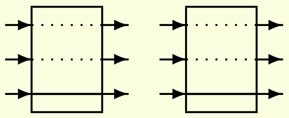
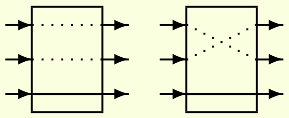
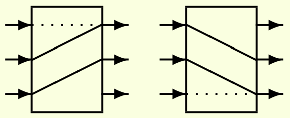
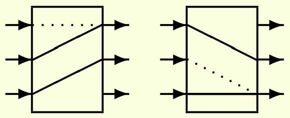
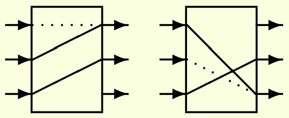
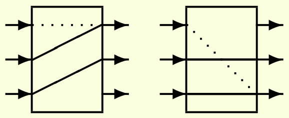
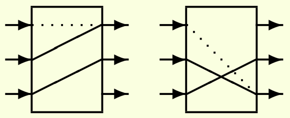
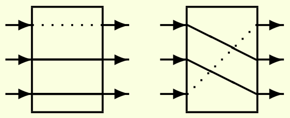
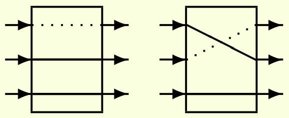
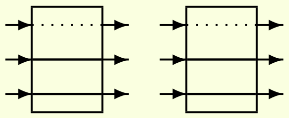
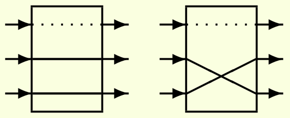
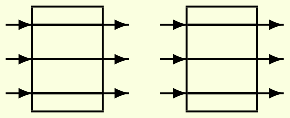
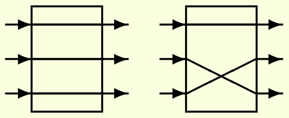
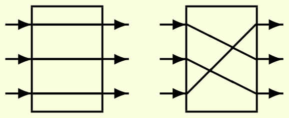
Solid edge: the state changes to another

Dotted edge: the state remains unchanged

## 8 representatives of 2-RLEMs

| | | | |
|---|---|---|---|
| 2-0 (eq. to wires)  | 2-1 (eq. to wires)  | 2-2  | 2-3  |
| 2-4  | 2-5 (eq. to wires)  | 2-16 (eq. to wires)  | 2-17  |

# 24 representatives of 3-RLEMs

| | | | |
|---|---|---|---|
| 3-0 (eq. to wires)  | 3-1 (eq. to wires)  | 3-3 (eq. to wires)  | 3-6 (eq. to 2-2)  |
| 3-7  | 3-9  | 3-10  | 3-11 (eq. to 2-3)  |
| 3-18  | 3-19 (eq. to 2-4)  | 3-21 (eq. to wires)  | 3-23  |
| 3-60  | 3-61  | 3-63  | 3-64  |
| 3-65  | 3-90  | 3-91  | 3-94 (eq. to wires)  |
| 3-95 (eq. to 2-17)  | 3-450 (eq. to wires)  | 3-451  | 3-453  |

## Numbers of representatives of degenerate and non-degenerate 2-, 3-, and 4-RLEMs

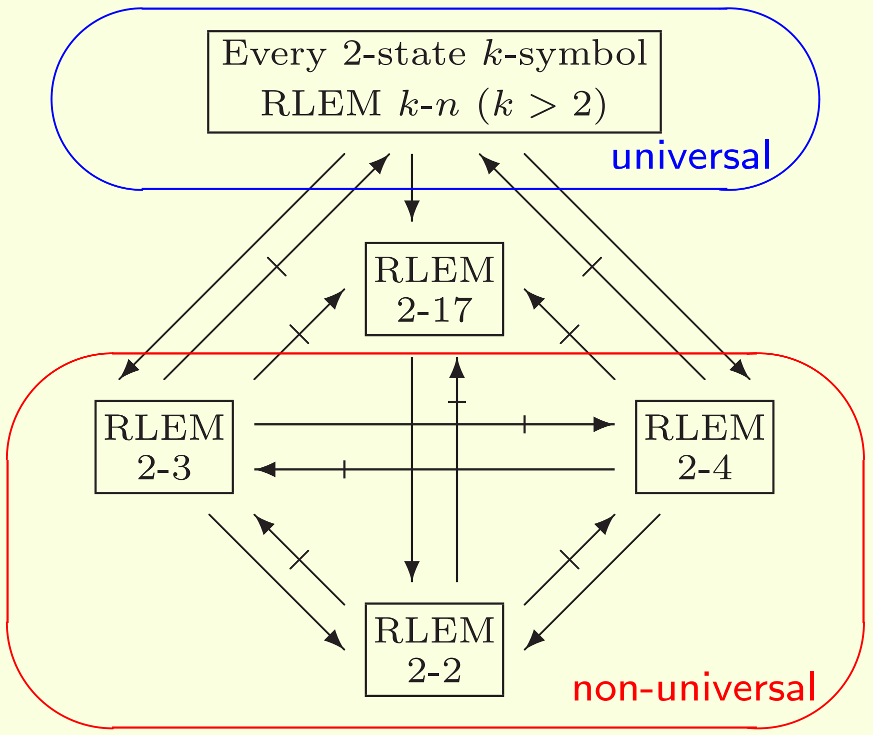
| | | Degenerate $k$-RLEMs | | | Non-degenerate |
|---|---|---|---|---|---|
| $k$ | Total | Type (i) | Type (ii) | Type (iii) | $k$-RLEMs |
| 2 | 8 | 2 | 2 | 0 | 4 |
| 3 | 24 | 3 | 3 | 4 | 14 |
| 4 | 82 | 5 | 4 | 18 | 55 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |

## Universality and non-universality of 2-state RLEMs

**Theorem 4** [Morita et al., 2012] Every non-degenerate 2-state $k$-symbol RLEM is universal if $k > 2$.

**Theorem 5** [Mukai, Morita, 2012] 2-state 2-symbol RLEMs 2-2, 2-3 and 2-4 are non-universal.

## Hierarchy of 2-state non-degenerate RLEMs



[Mukai, Morita, 2012]

- $A \rightarrow B$ ($A \not\rightarrow B$, respectively) represents that $A$ can (cannot) simulate $B$.
- Any two combinations from {2-3, 2-4, 2-17} is universal. [Lee, et al., 2008], [Mukai, Morita, 2011]

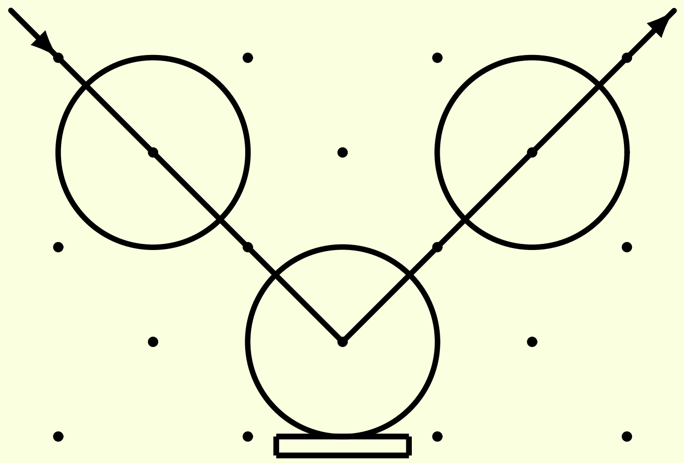
## 3. Realizing a reversible element with memory in the billiard ball model

## Billiard ball model (BBM)

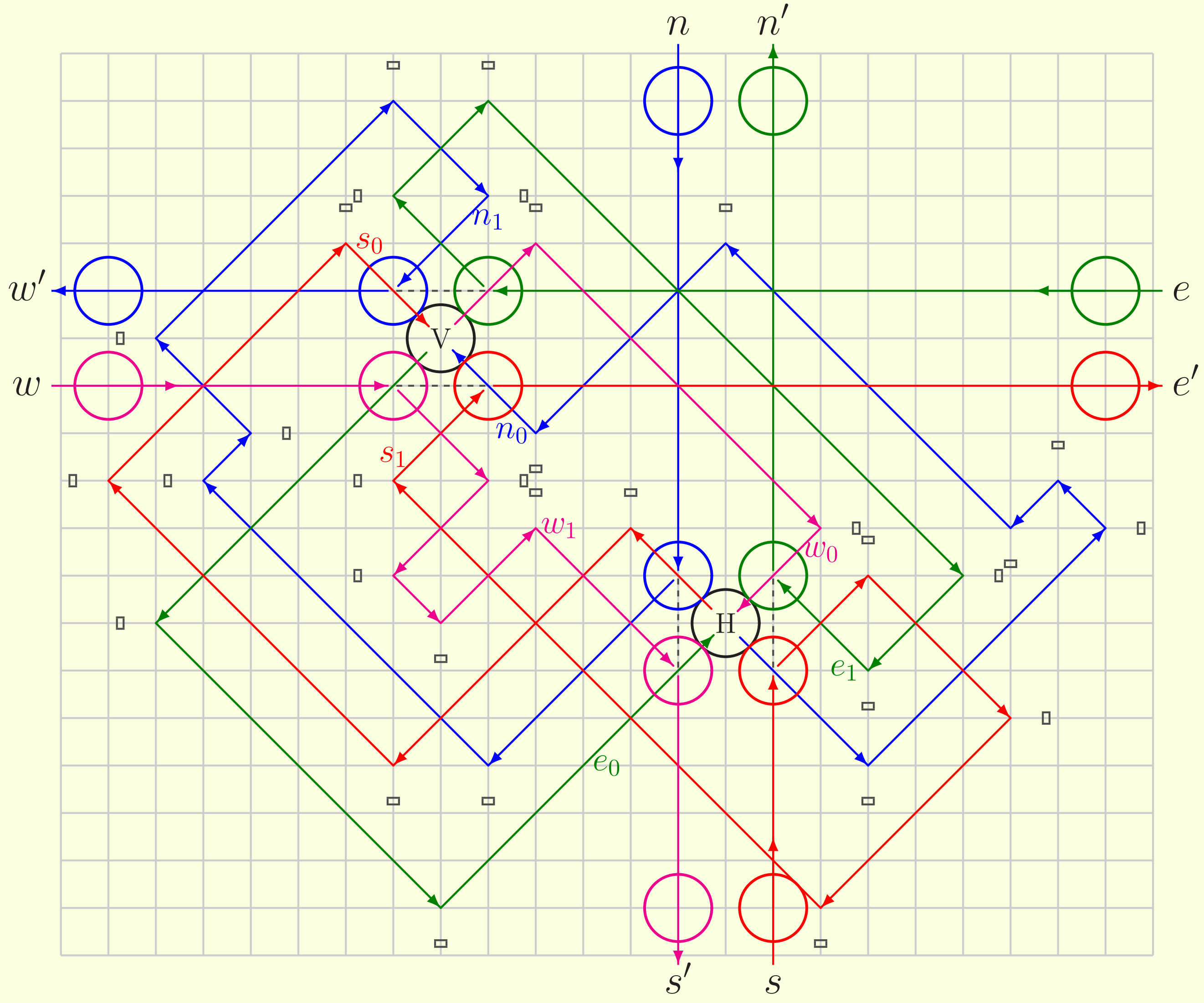
- A reversible physical model of computing.

  [Fredkin and Toffoli, 1982]

- It consists of ideal balls and reflectors.

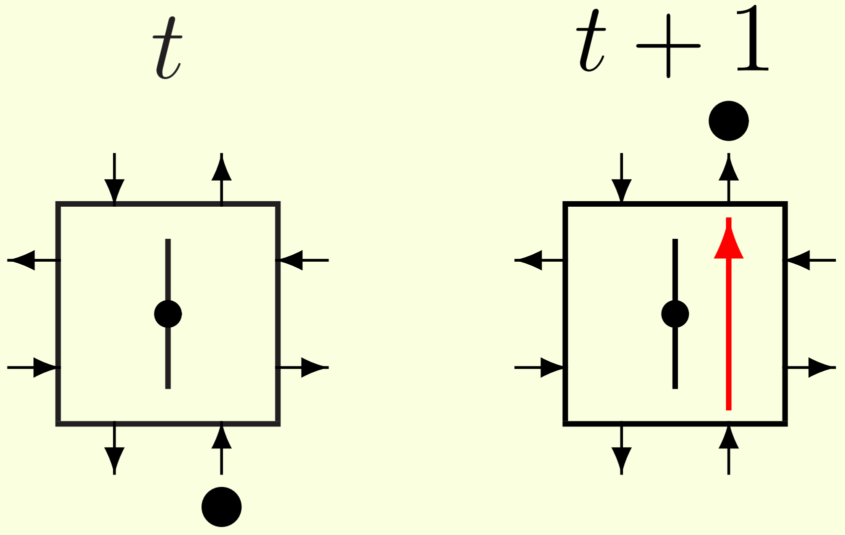


# Realization of an RE by BBM

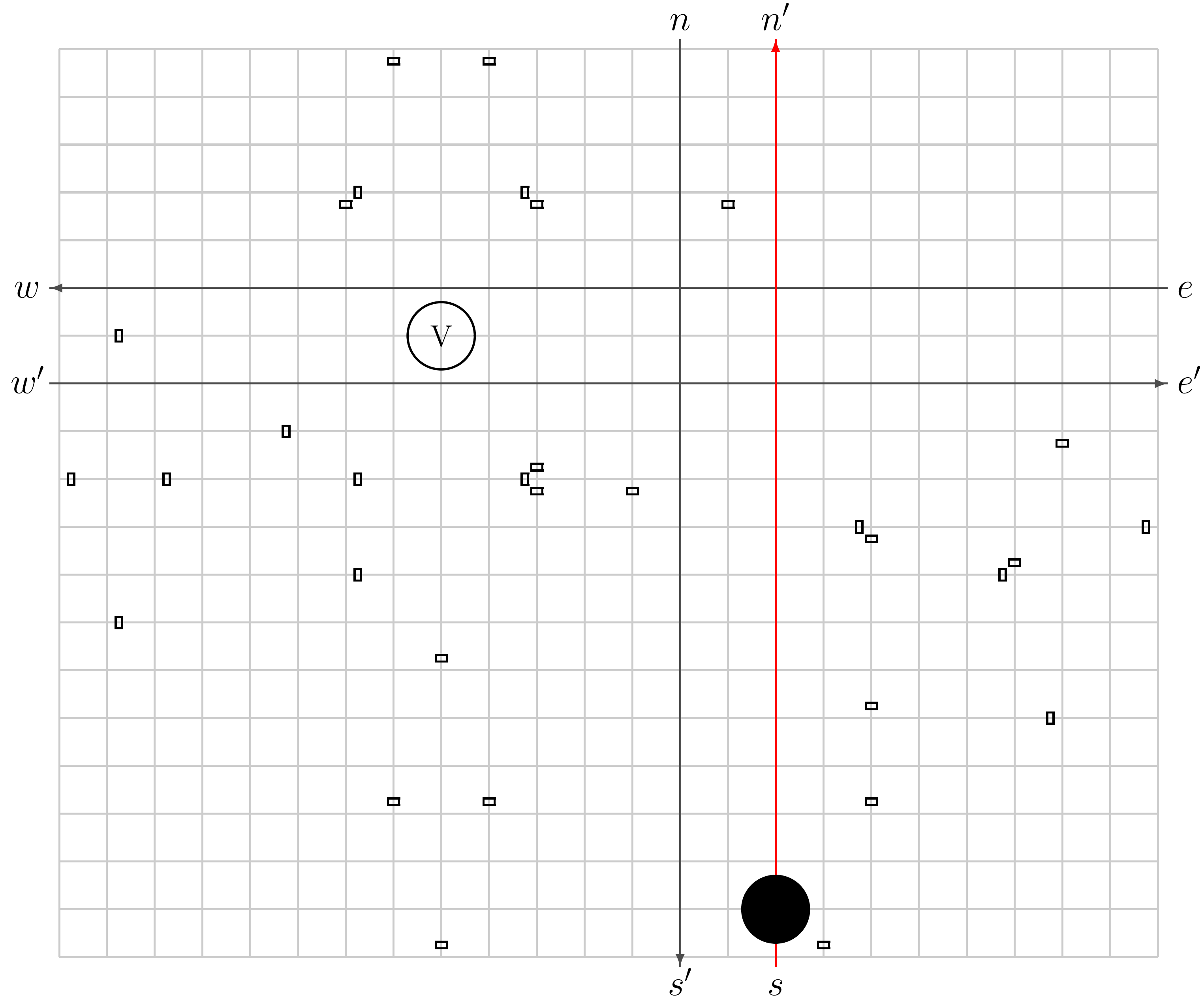


[Morita, 2008]

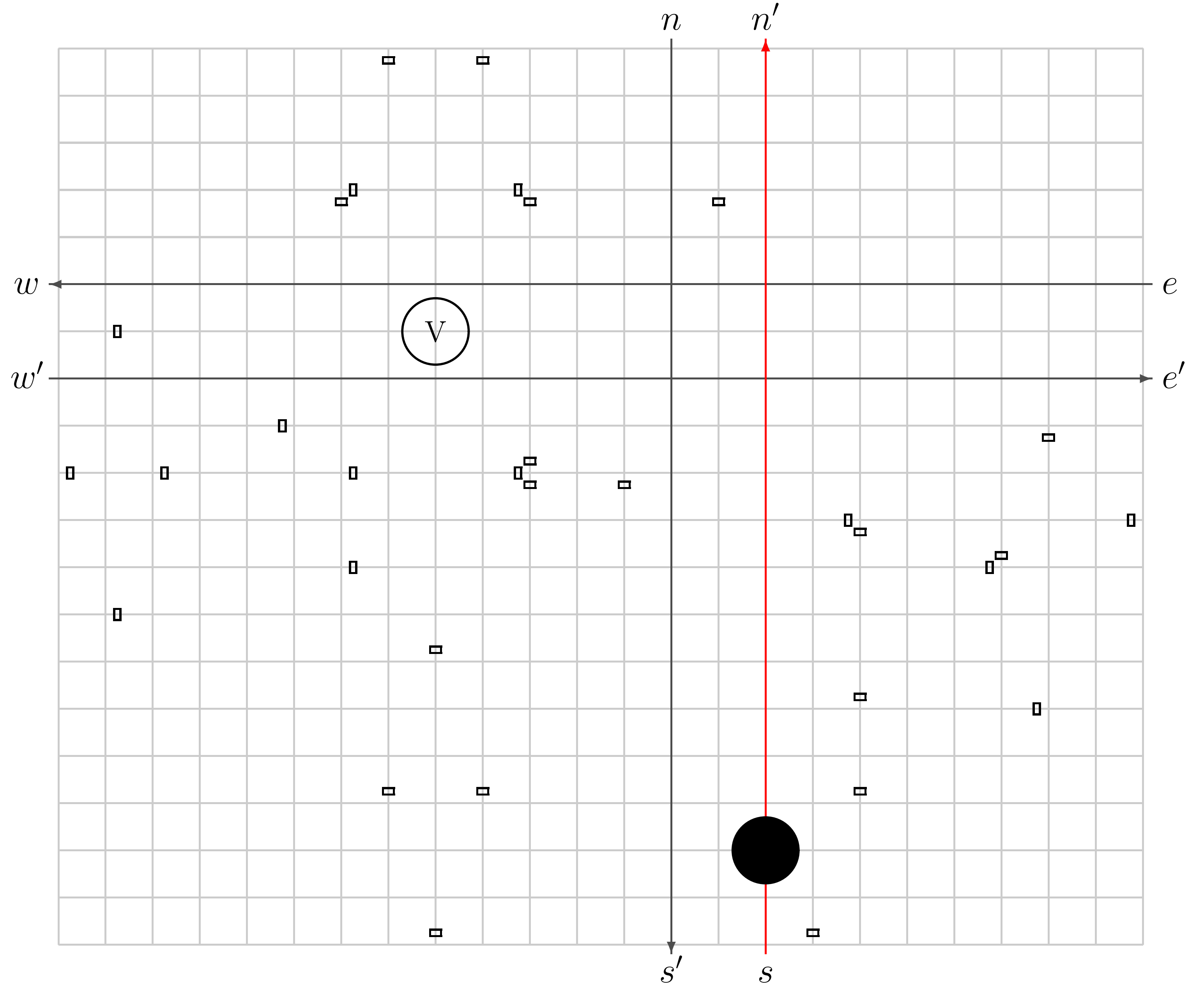
## Parallel case



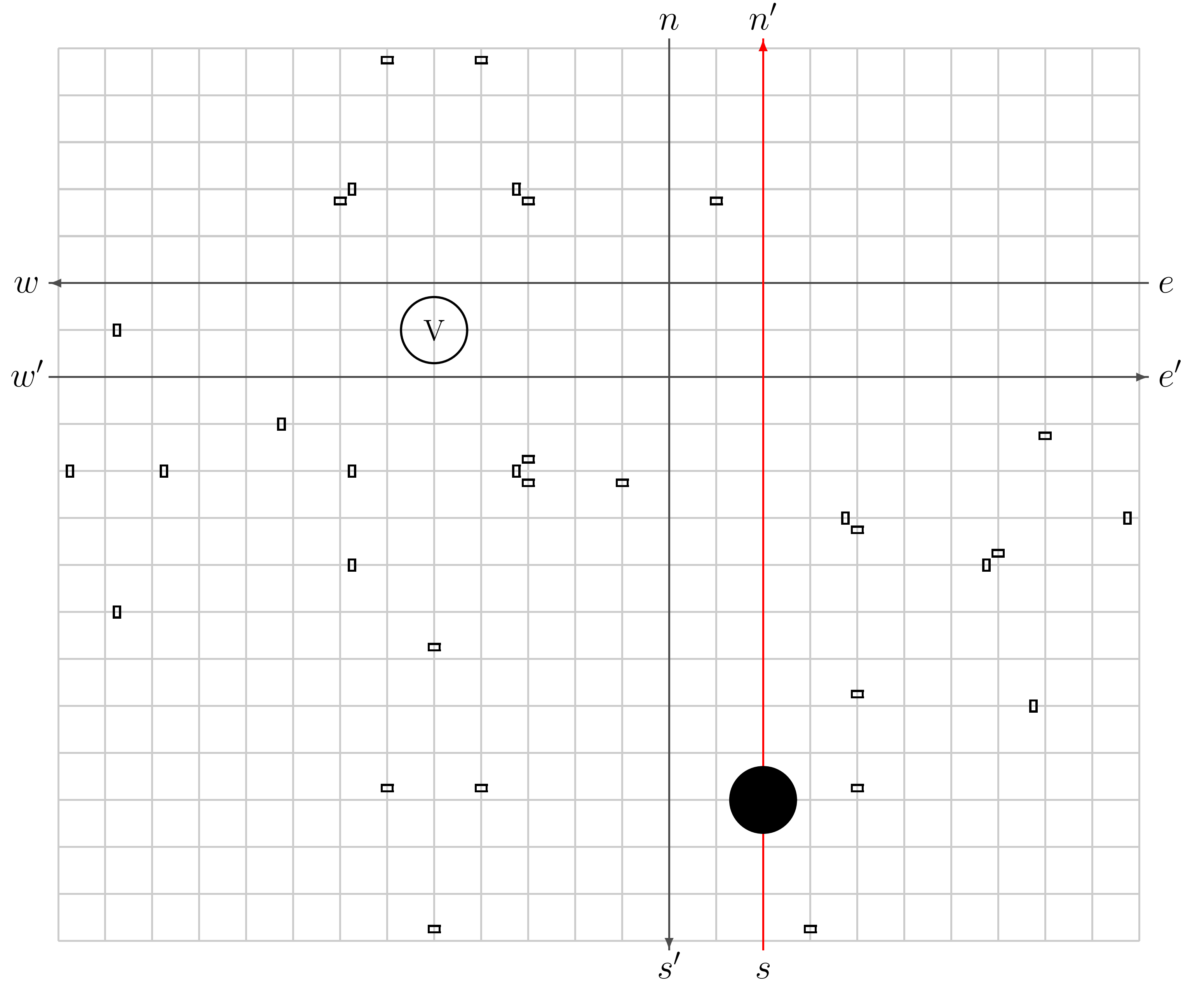
## Movements of Balls (State: $V$, Input: $s$)



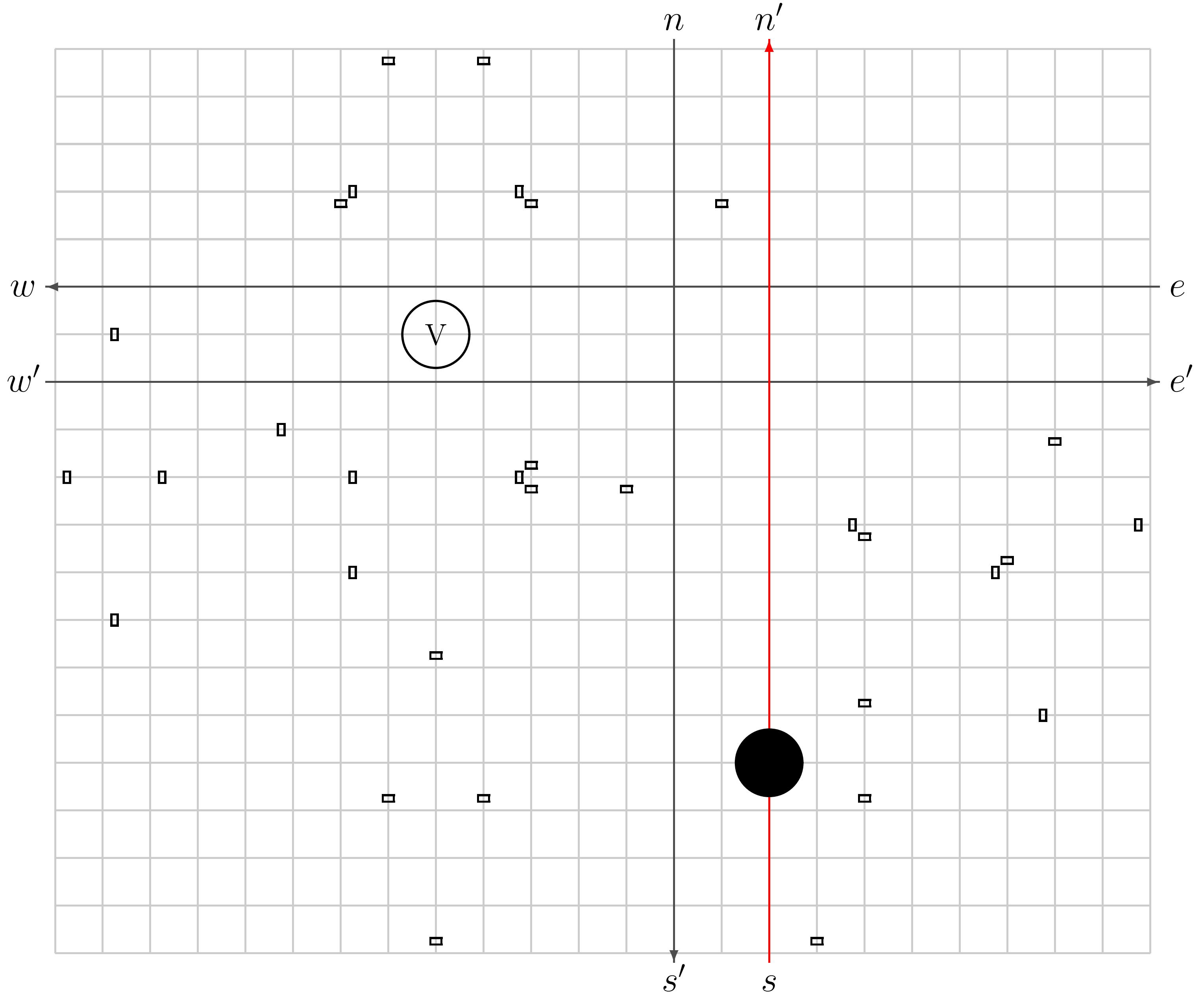
## Movements of Balls (State: $V$, Input: $s$)



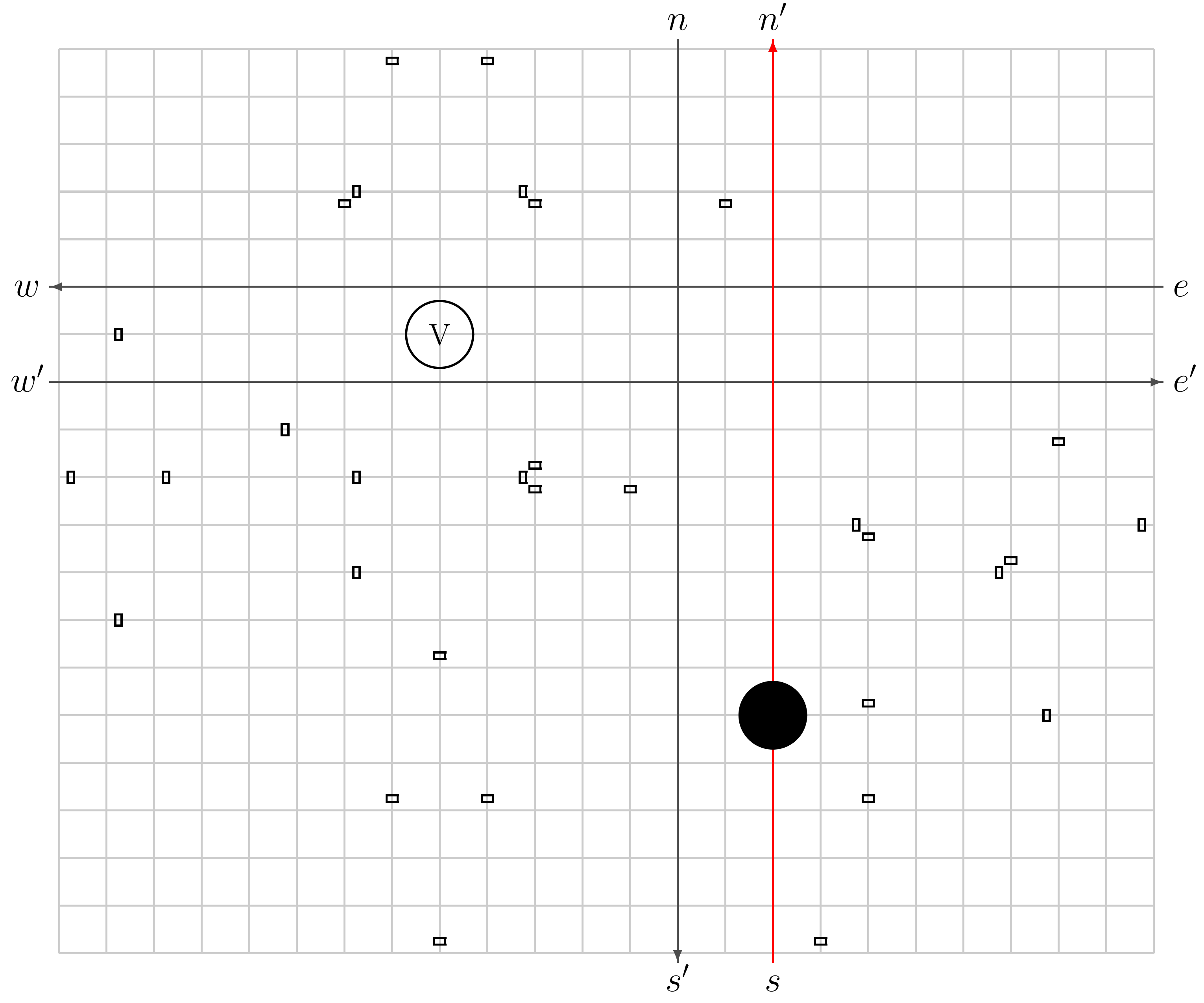
## Movements of Balls (State: $V$, Input: $s$)



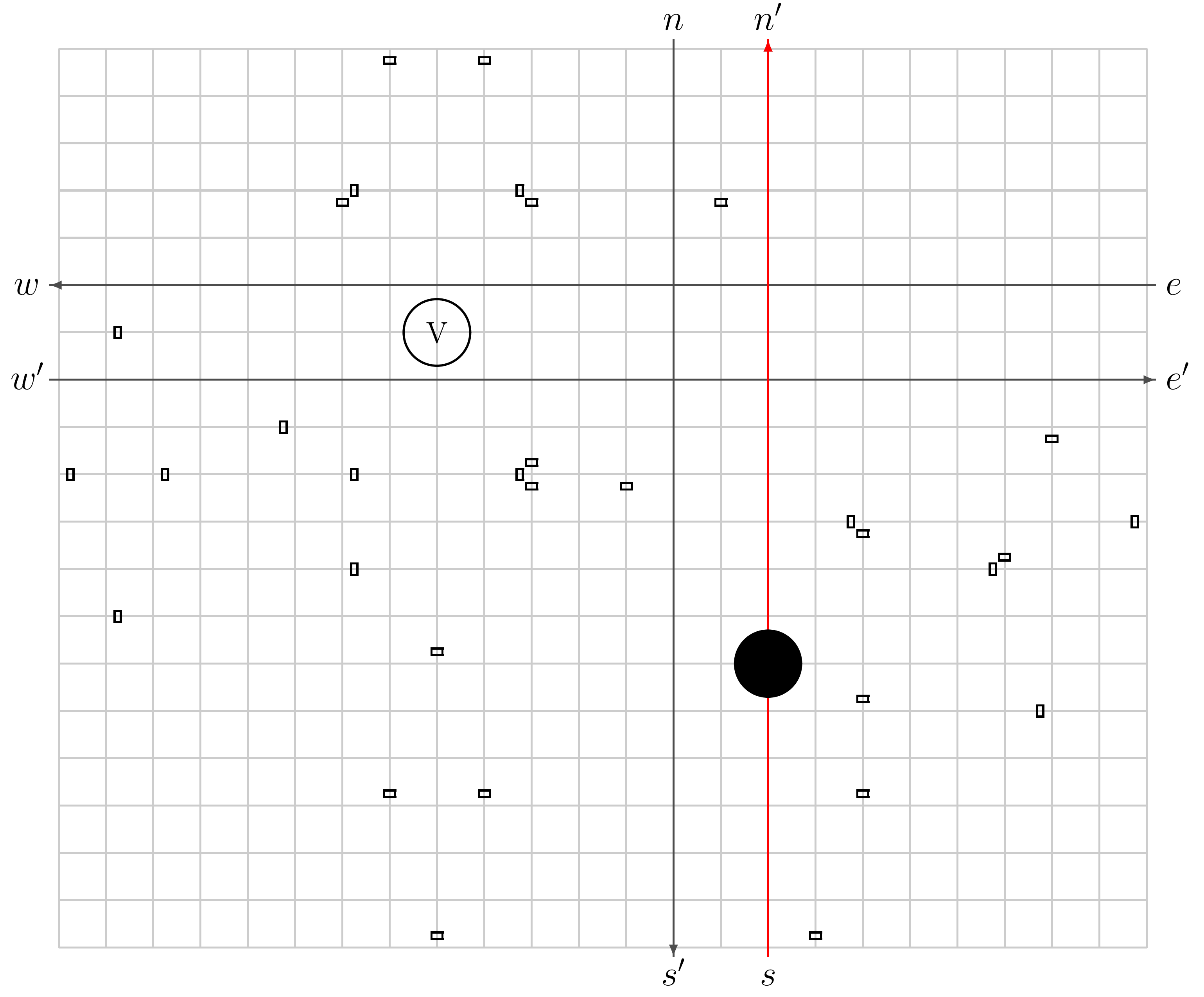
## Movements of Balls (State: $V$, Input: $s$)



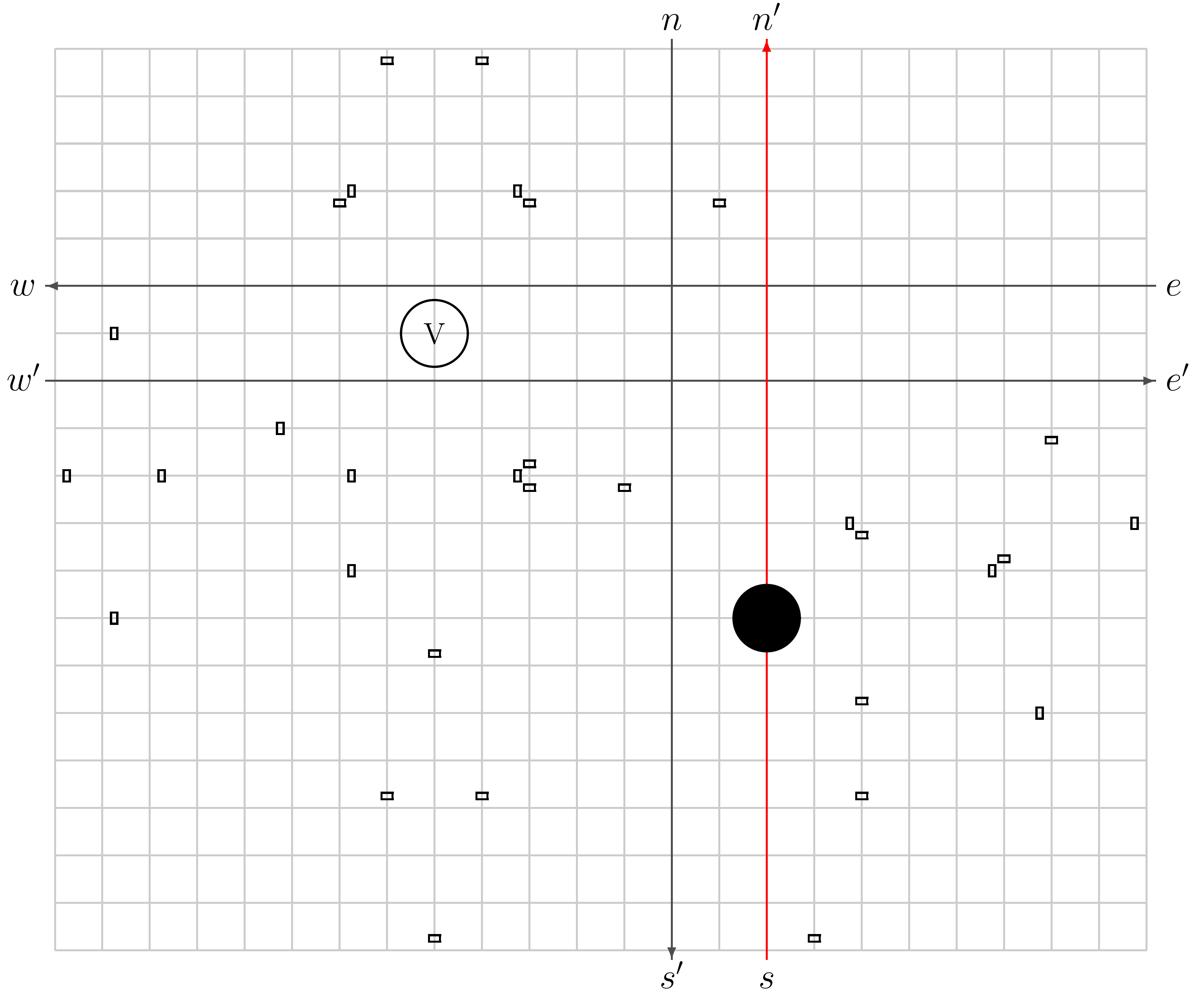
## Movements of Balls (State: $V$, Input: $s$)



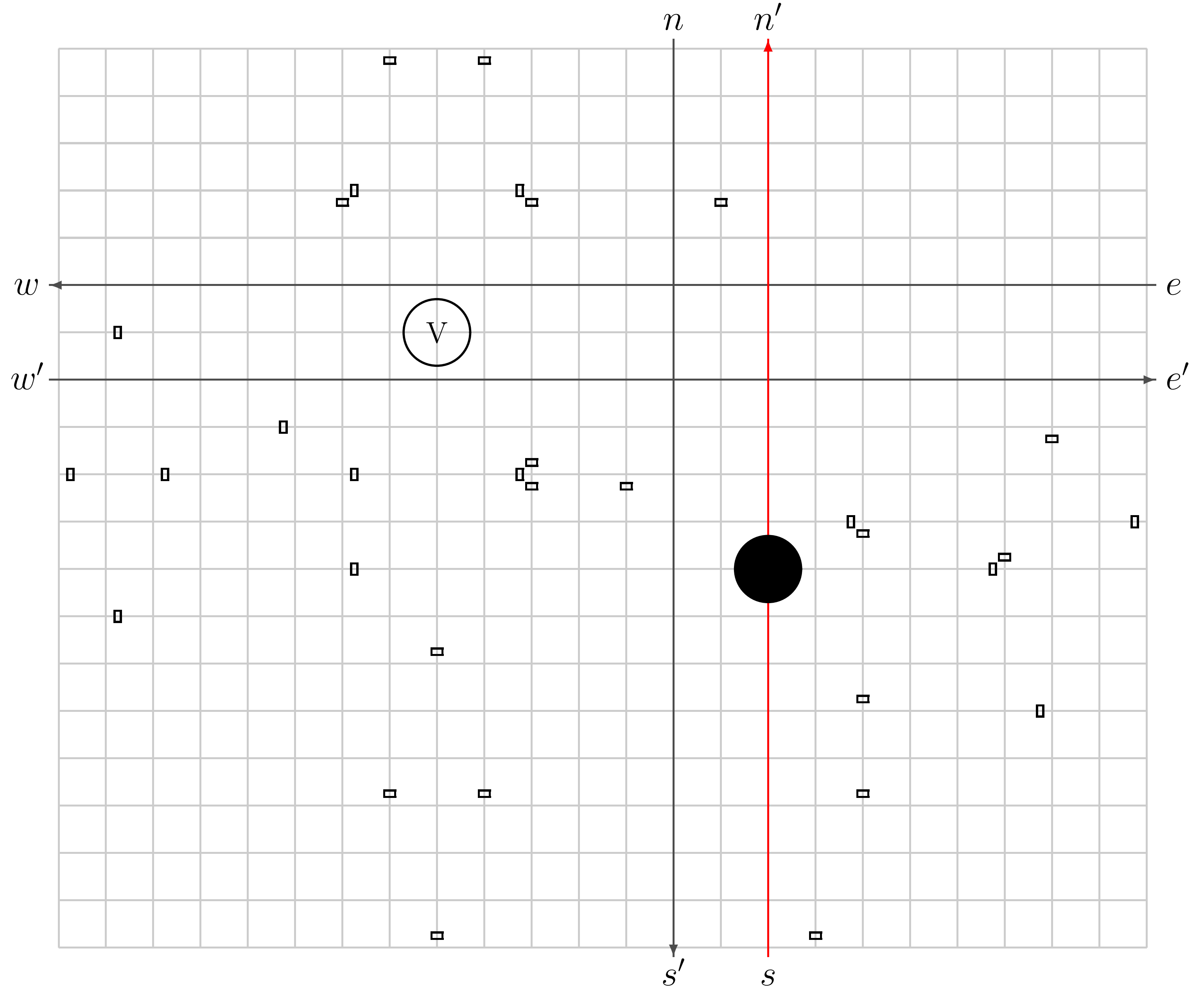
## Movements of Balls (State: $V$, Input: $s$)



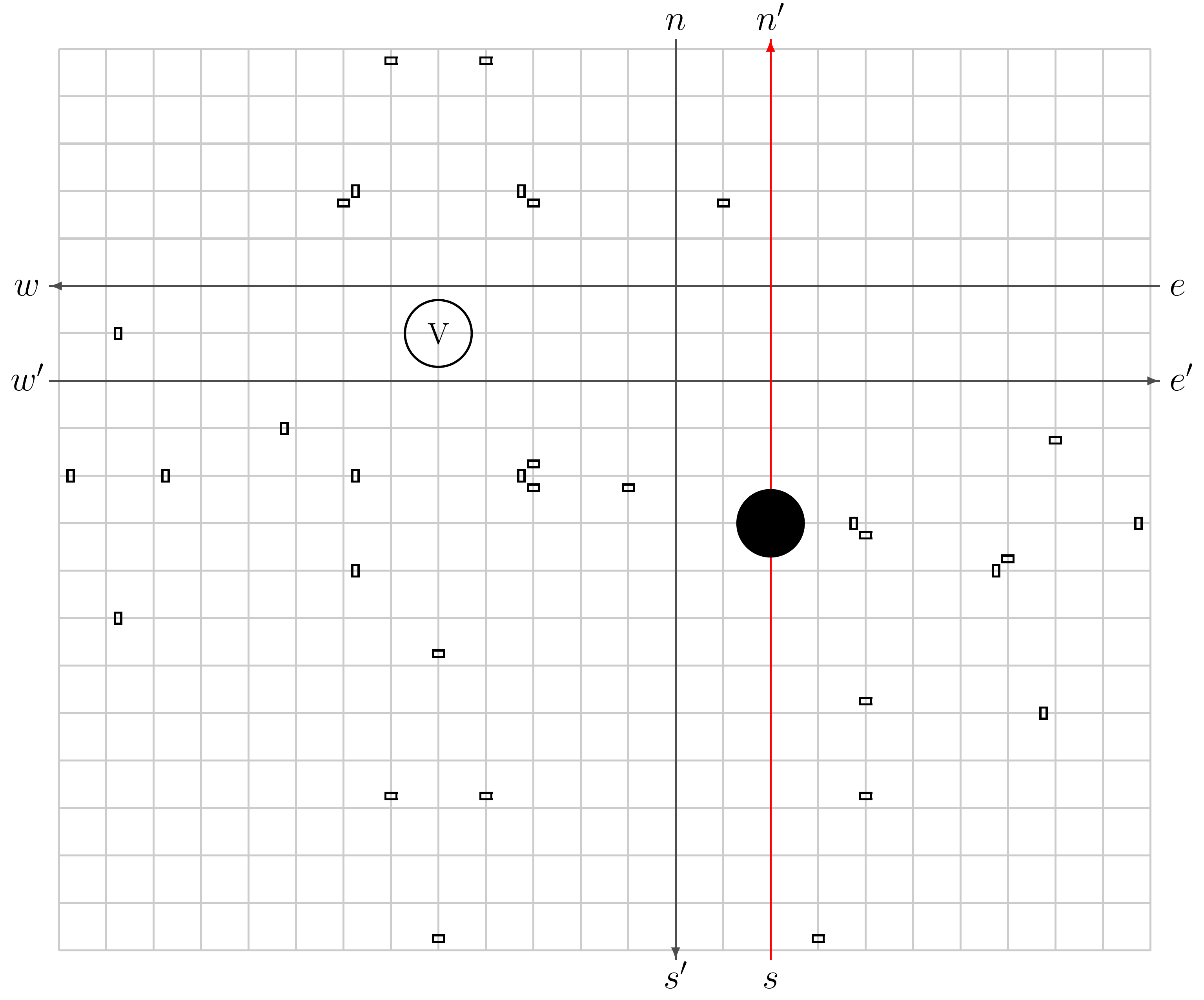
## Movements of Balls (State: $V$, Input: $s$)



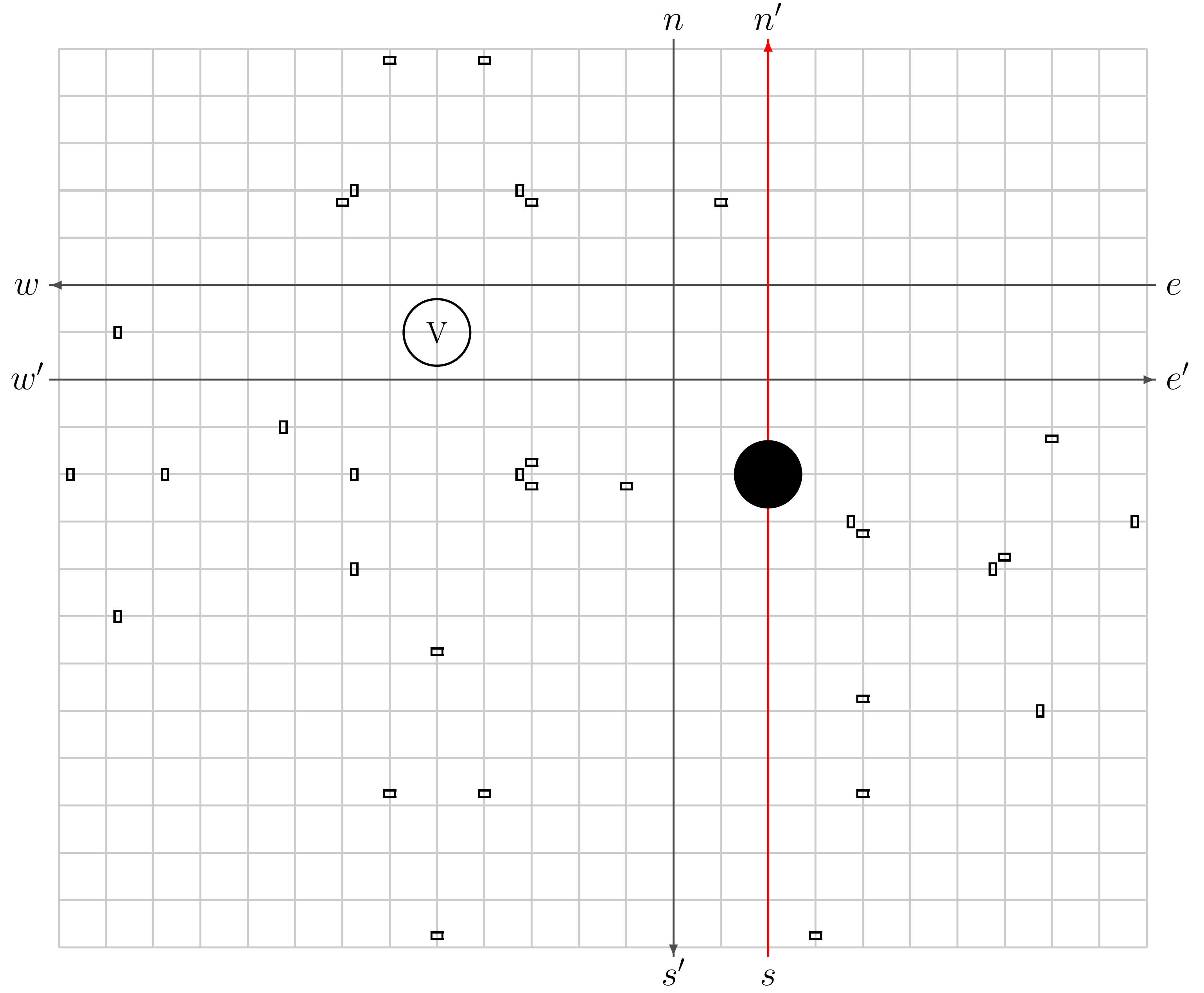
## Movements of Balls (State: $V$, Input: $s$)



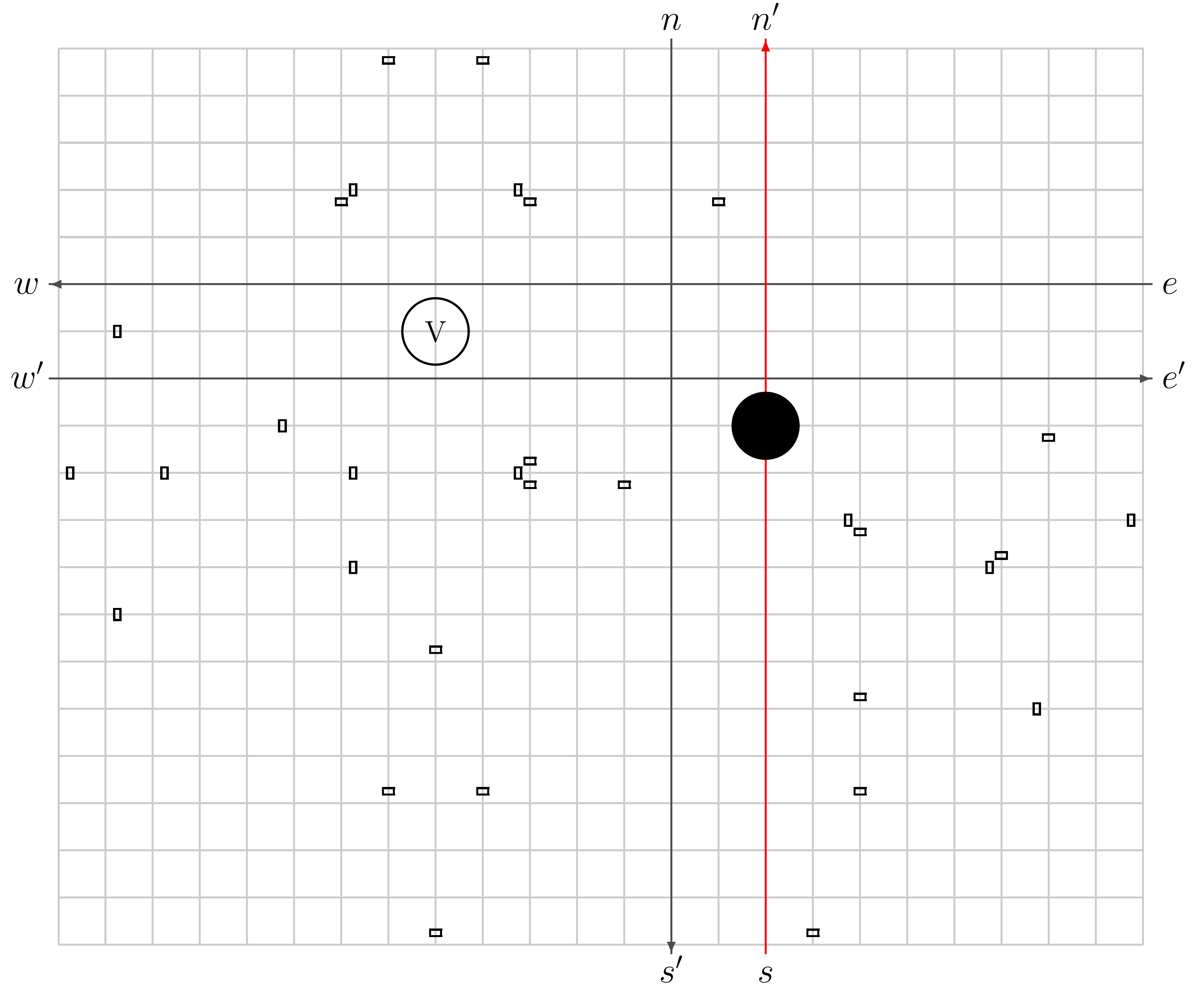
## Movements of Balls (State: $V$, Input: $s$)



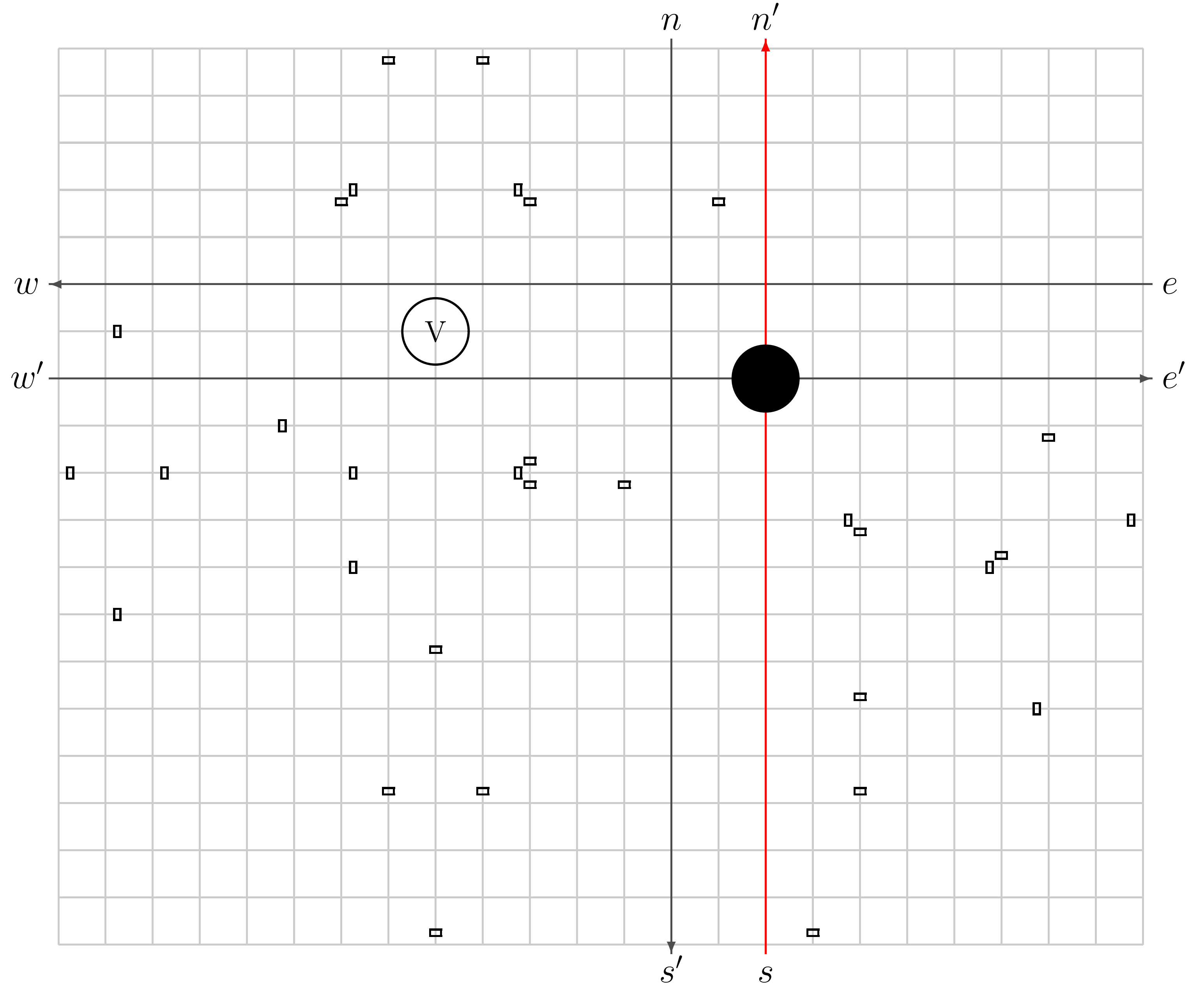
## Movements of Balls (State: $V$, Input: $s$)



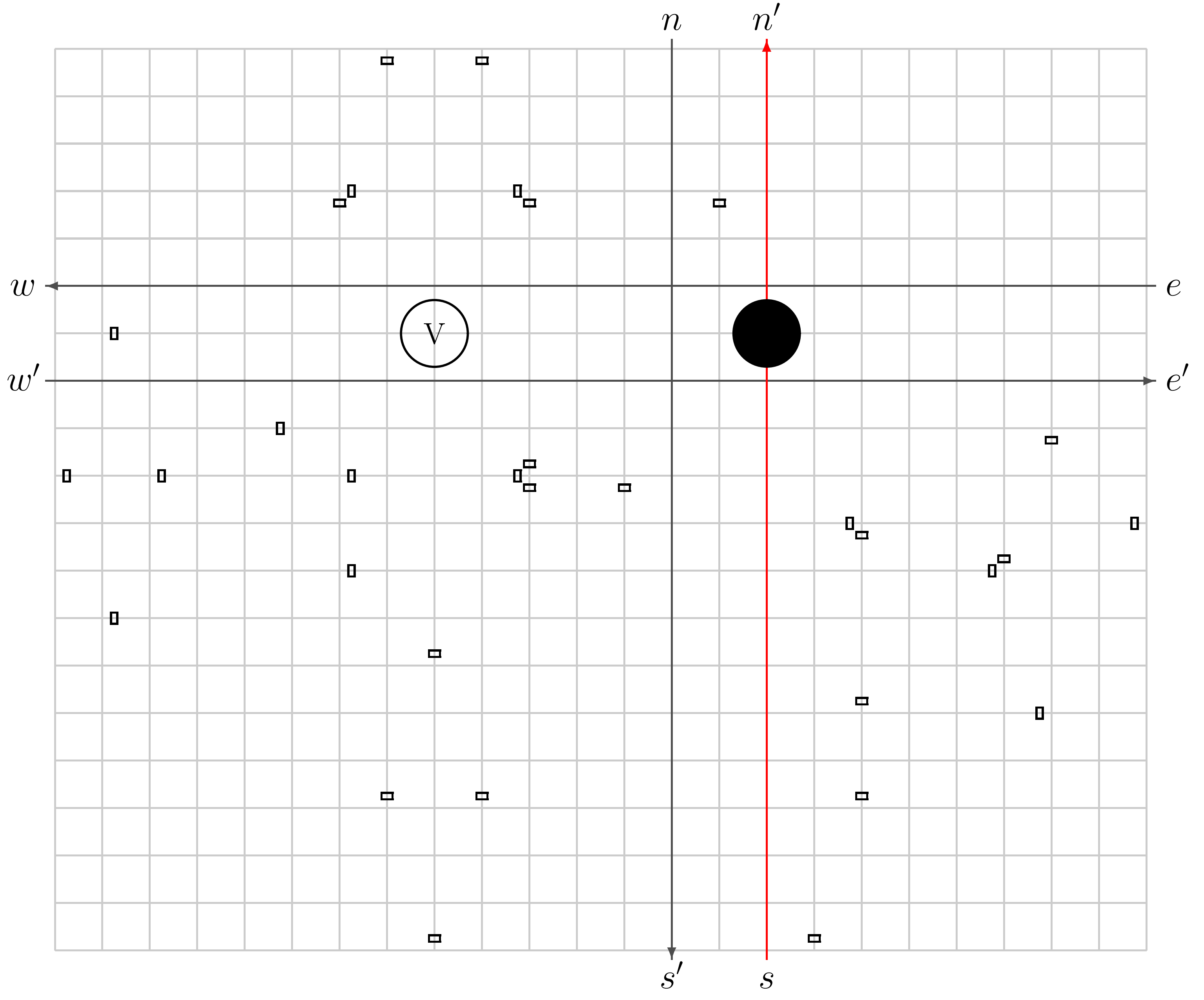
## Movements of Balls (State: $V$, Input: $s$)



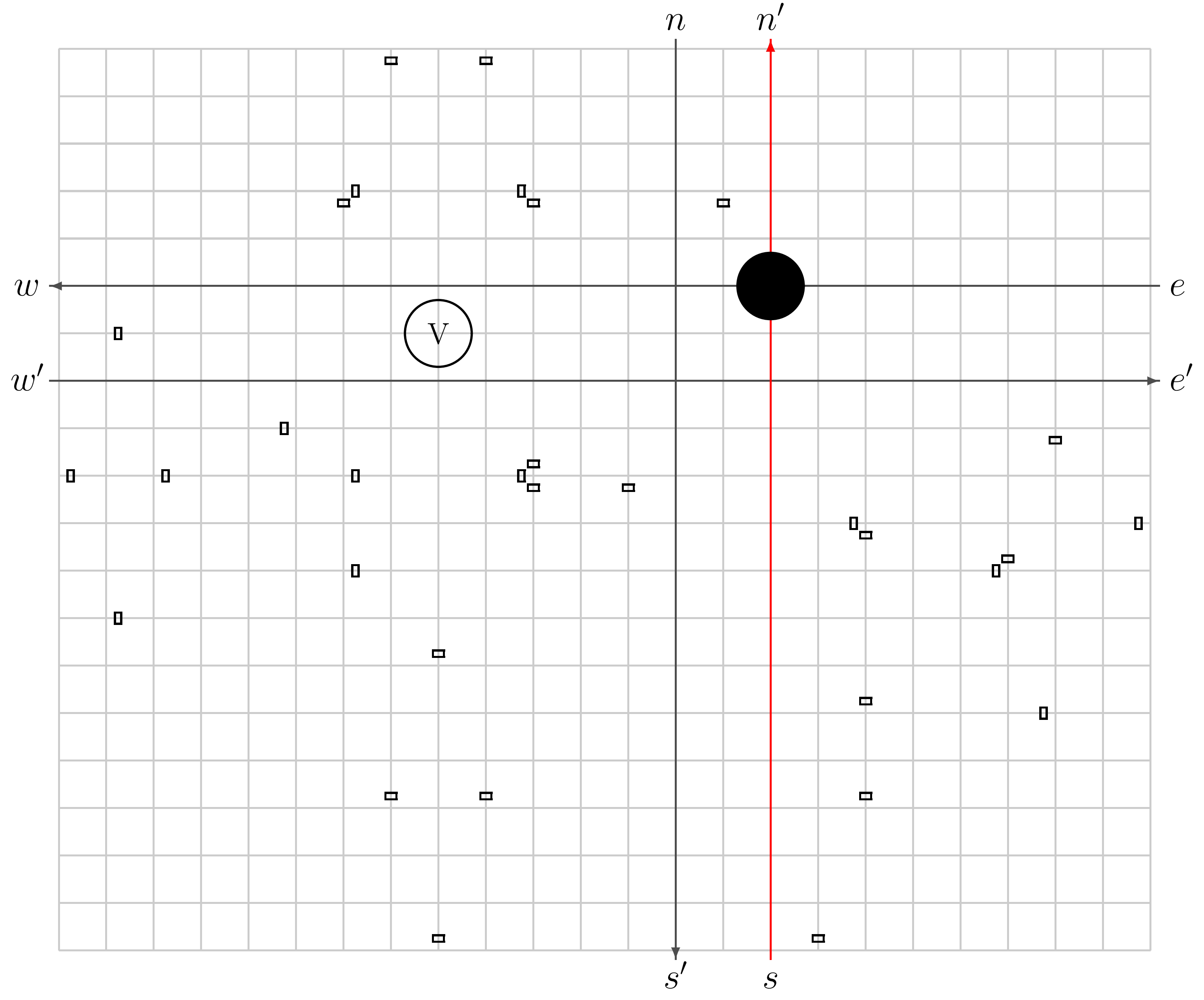
## Movements of Balls (State: $V$, Input: $s$)



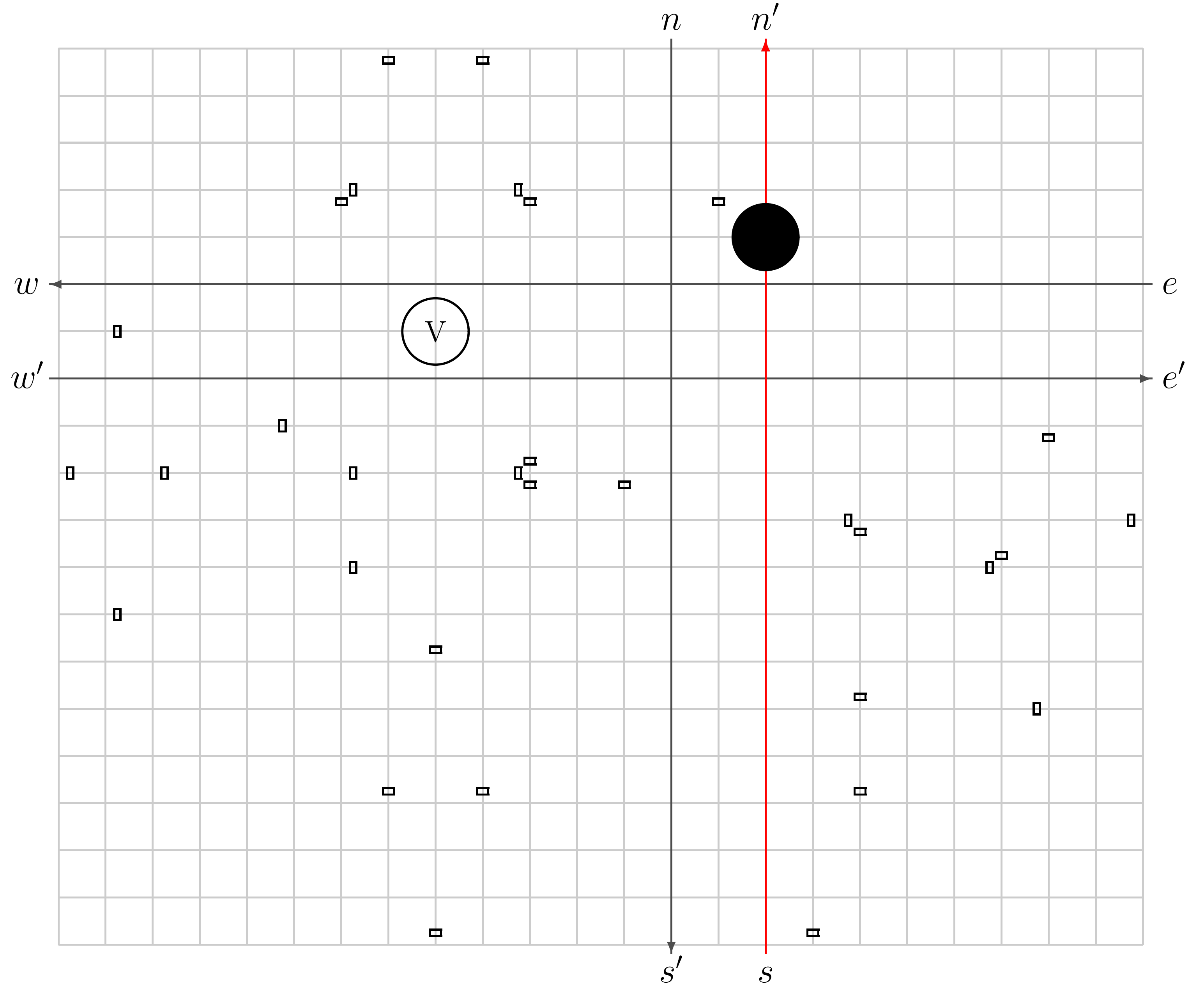
## Movements of Balls (State: $V$, Input: $s$)



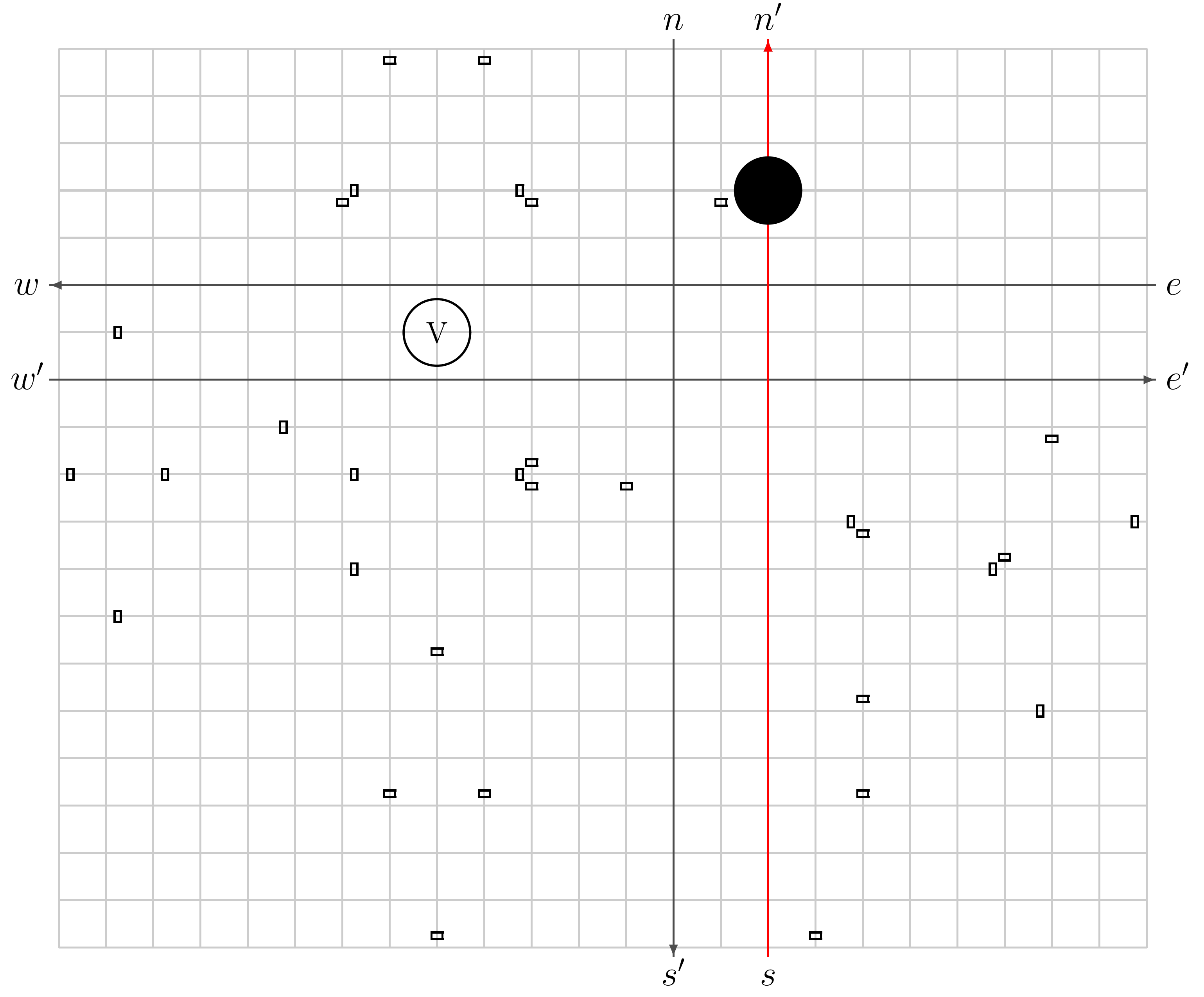
## Movements of Balls (State: $V$, Input: $s$)



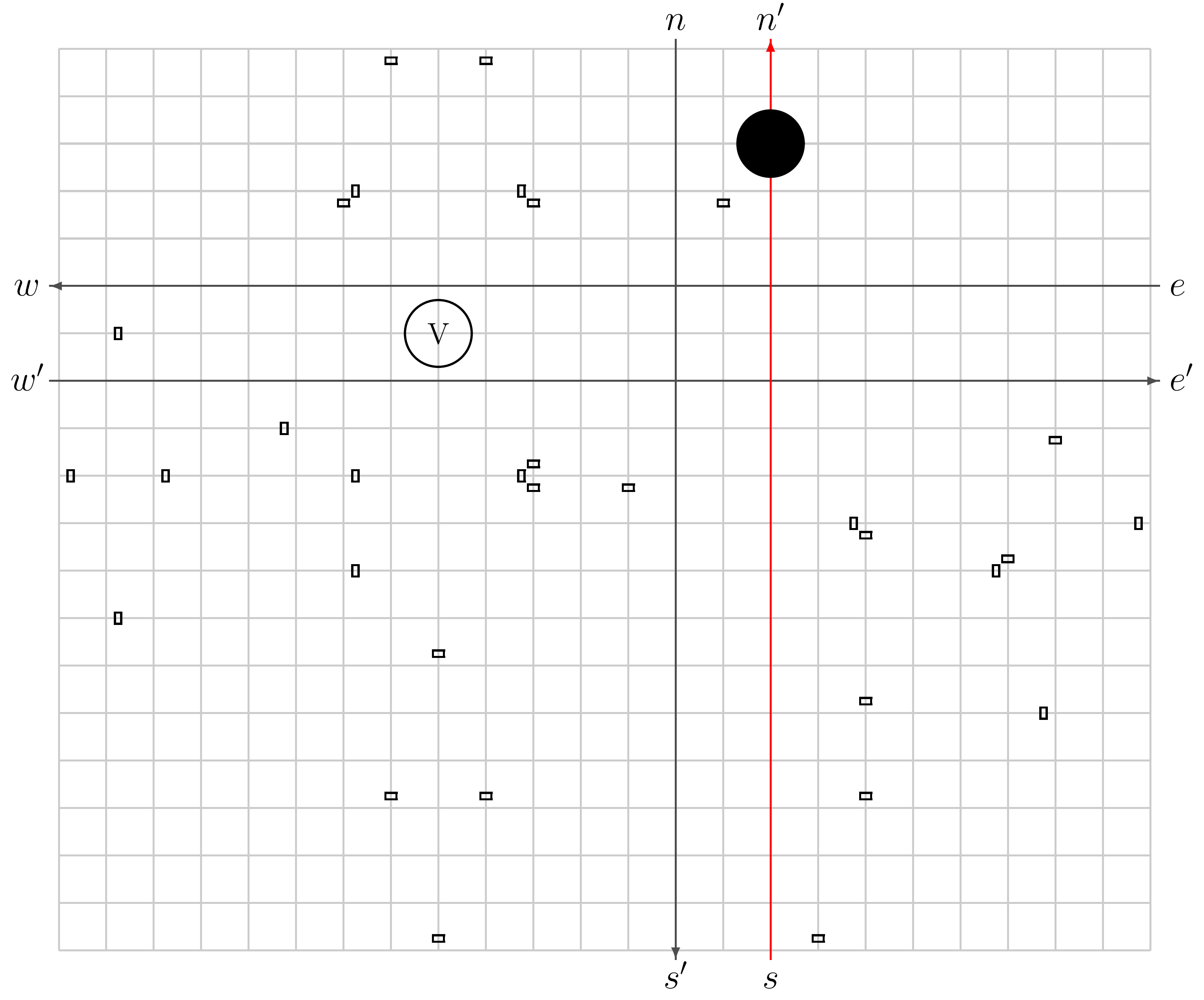
## Movements of Balls (State: $V$, Input: $s$)



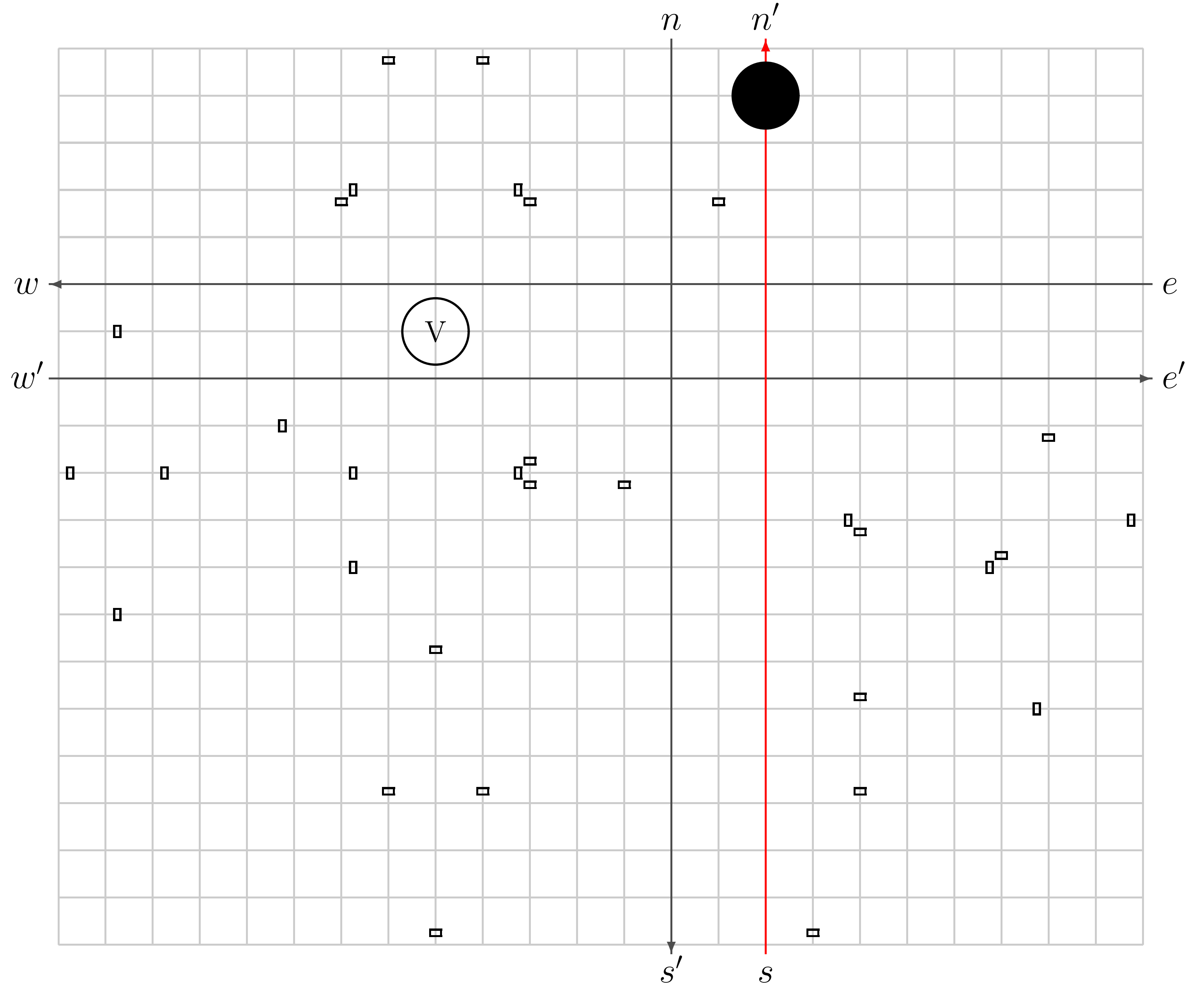
## Movements of Balls (State: $V$, Input: $s$)



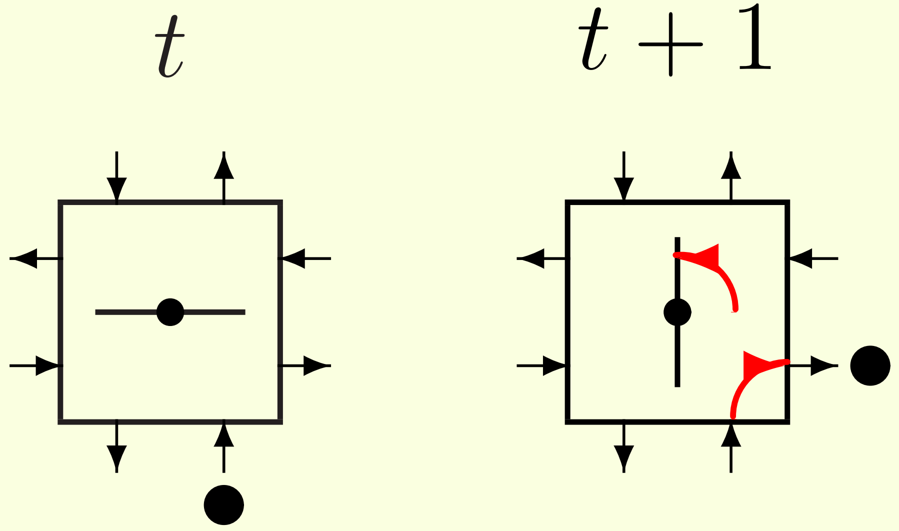
## Movements of Balls (State: $V$, Input: $s$)



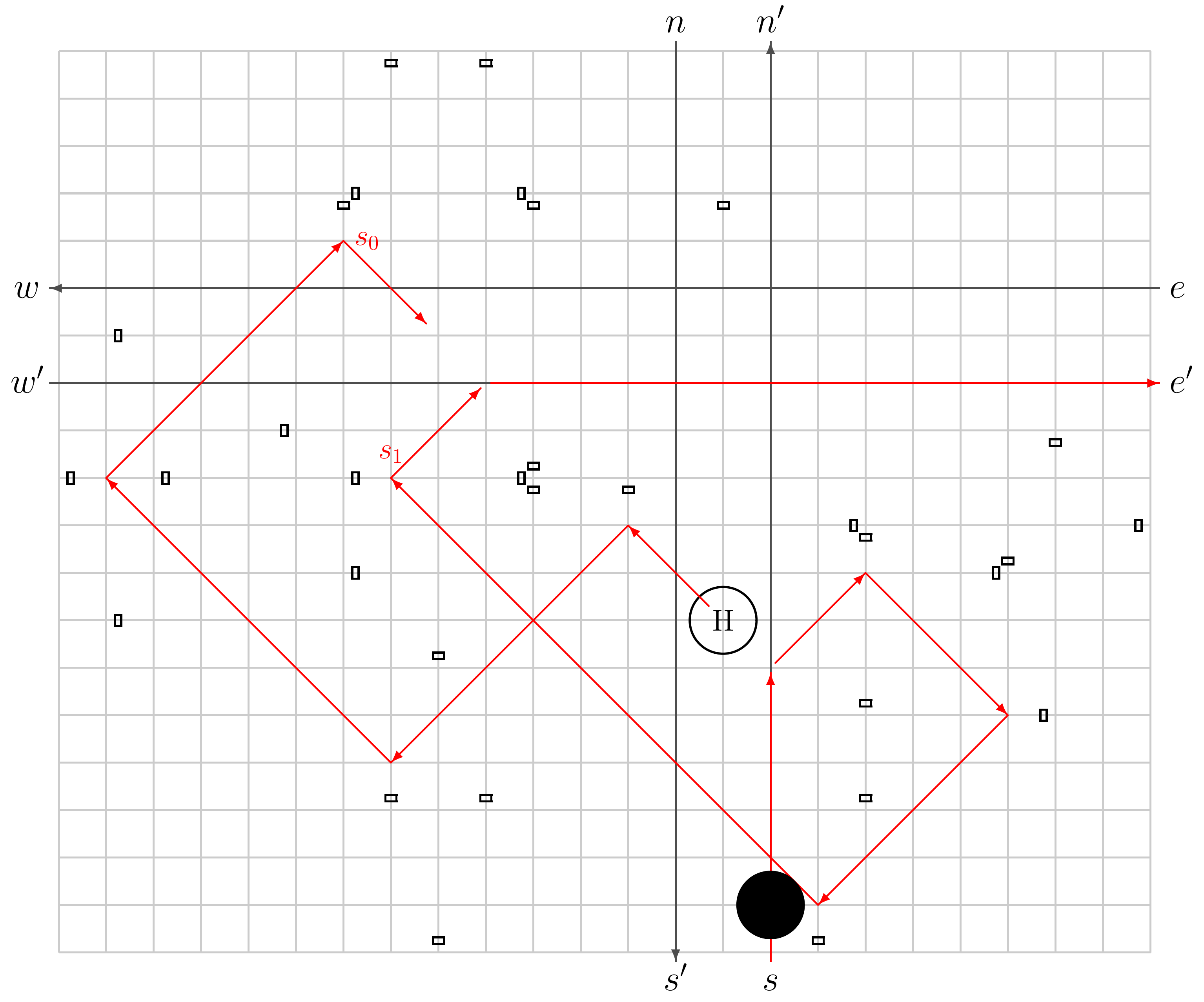
## Movements of Balls (State: $V$, Input: $s$)



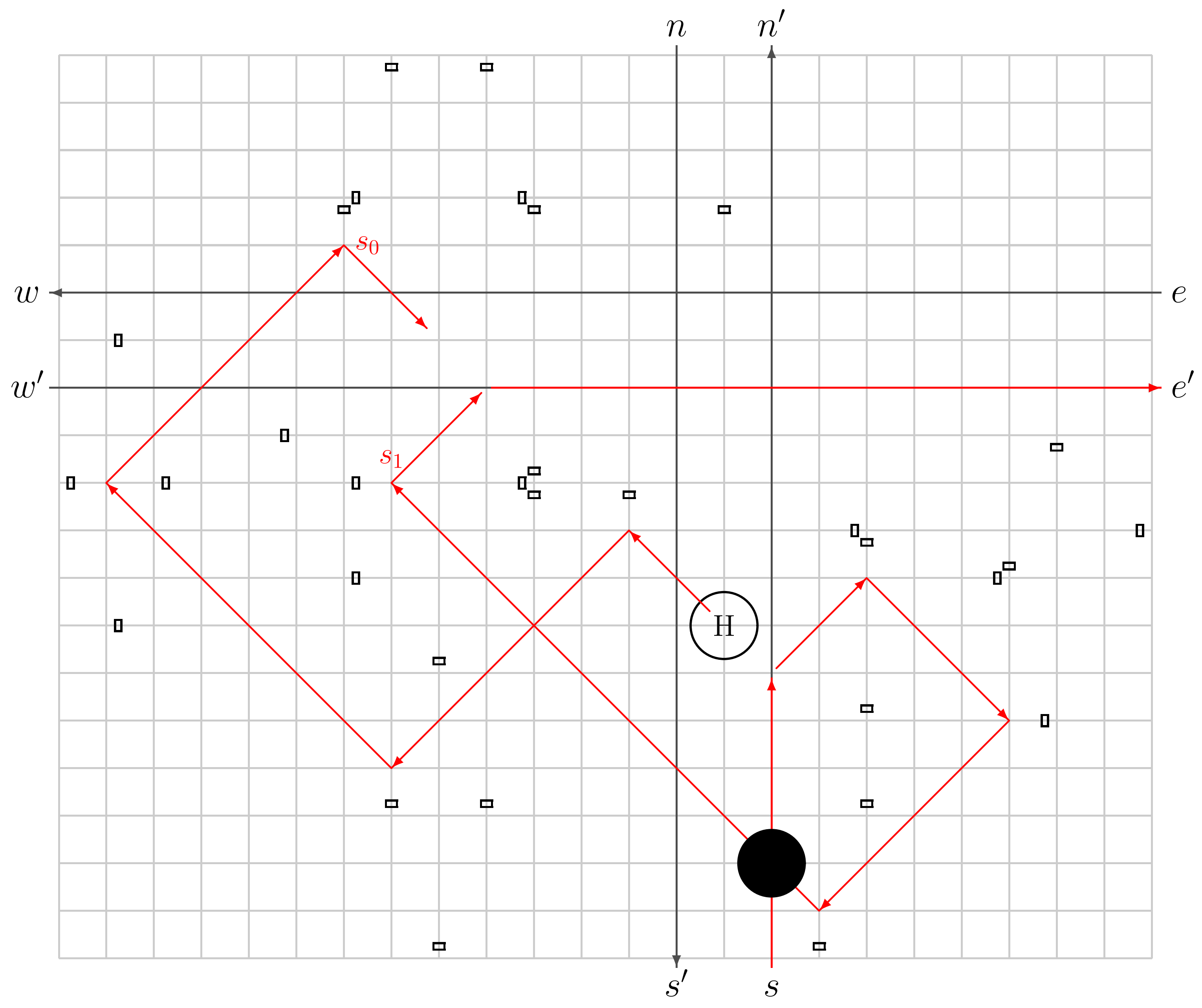
## Orthogonal case



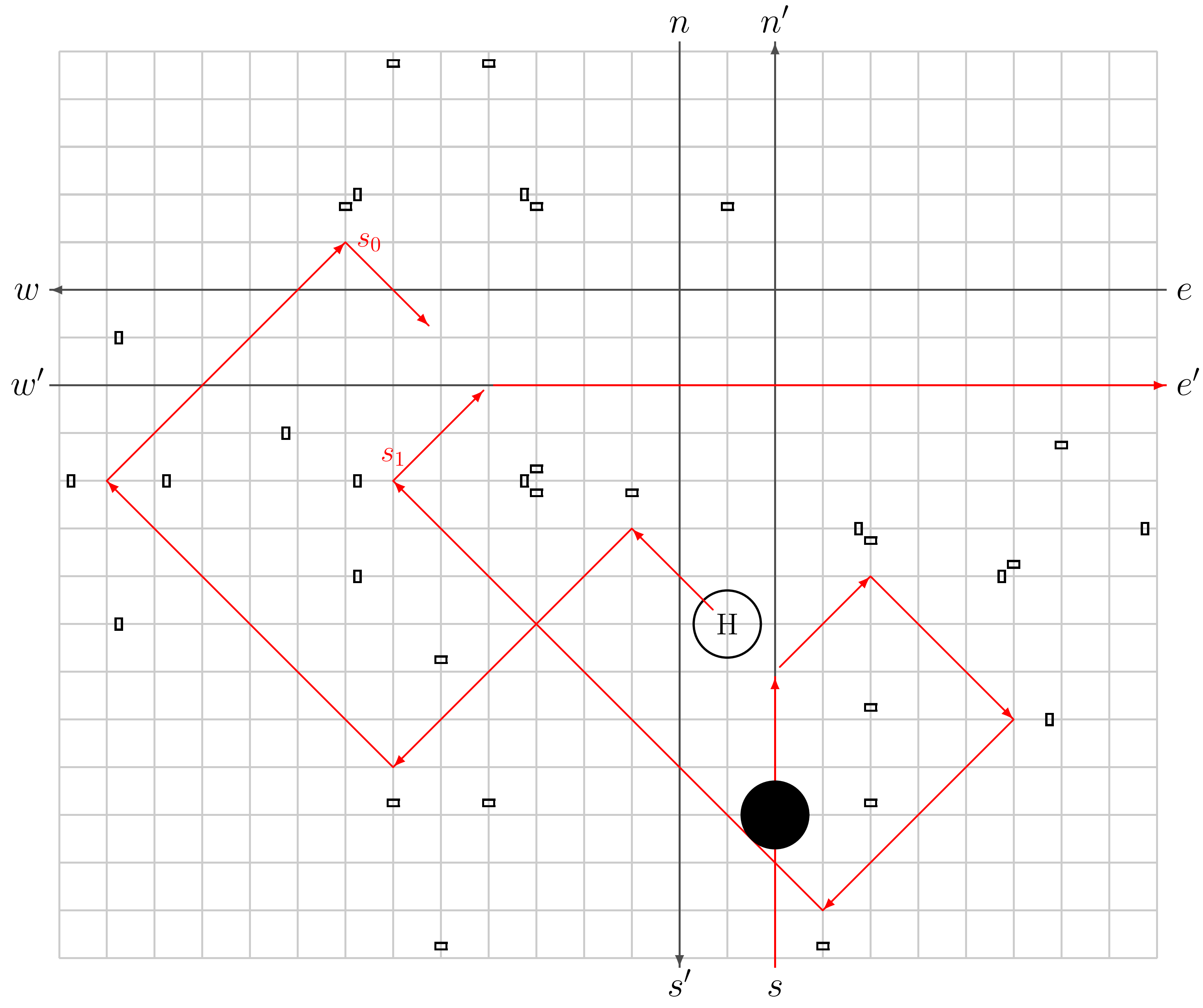
## Movements of Balls (State: $H$, Input: $s$)



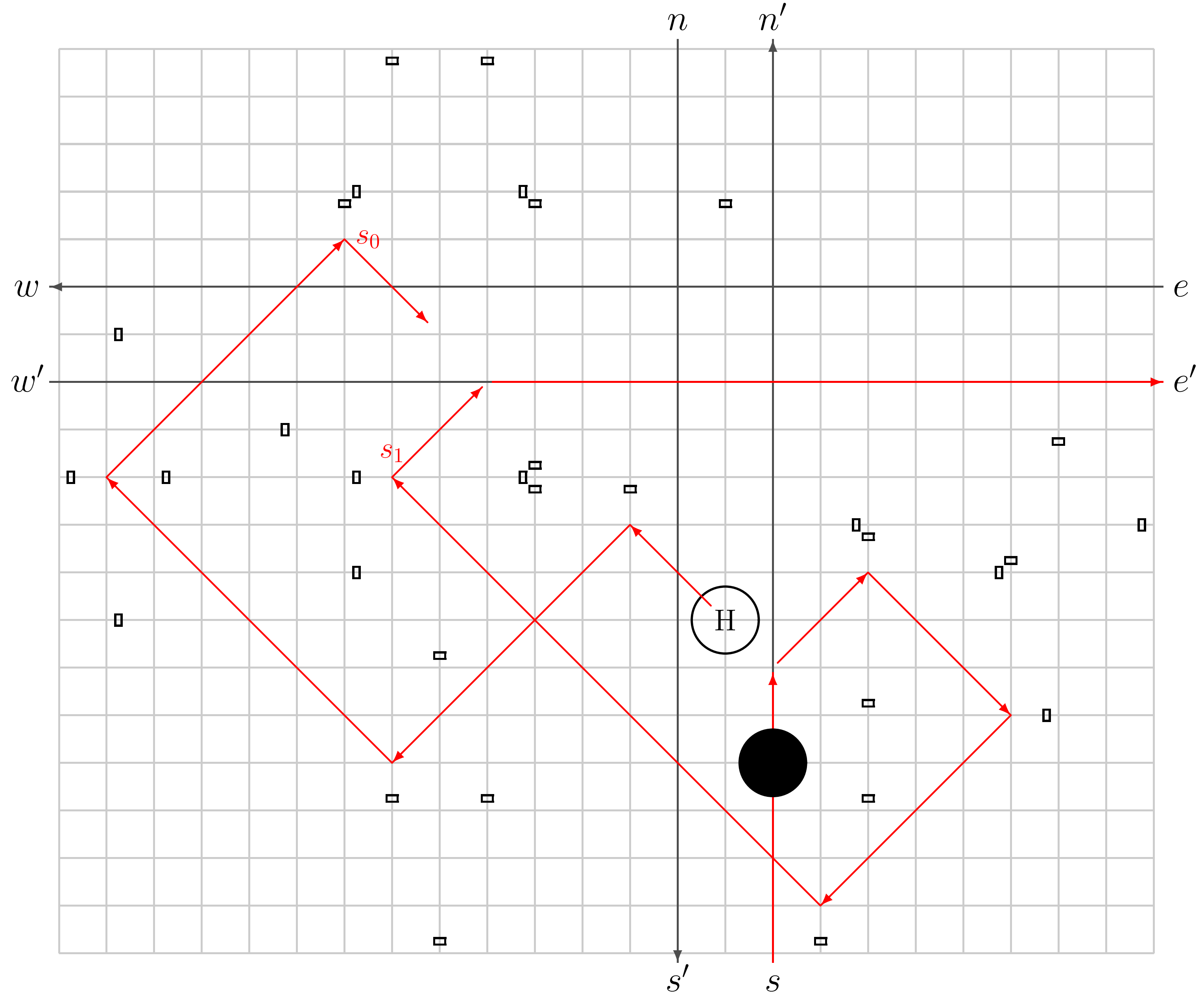
## Movements of Balls (State: $H$, Input: $s$)



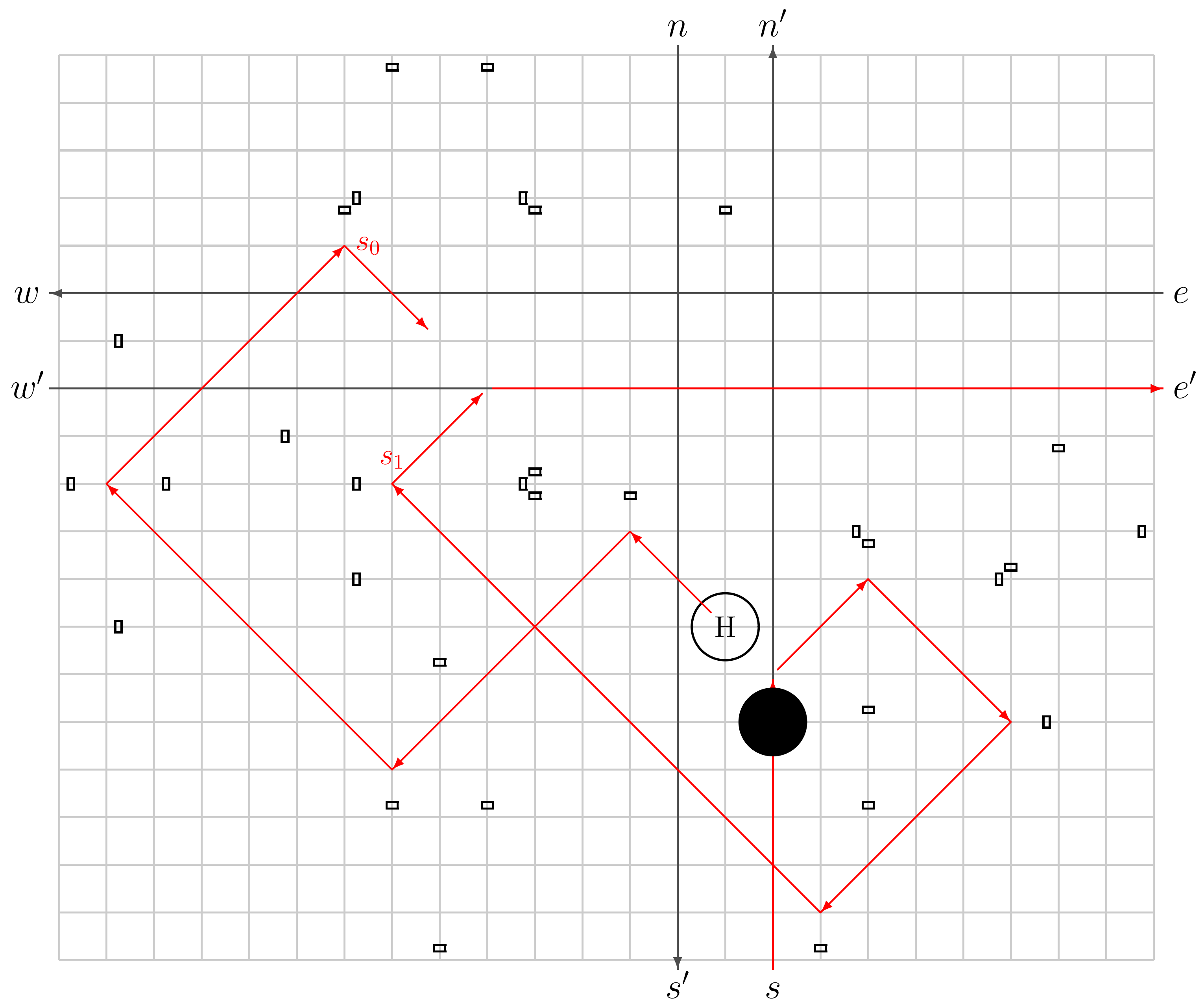
## Movements of Balls (State: $H$, Input: $s$)



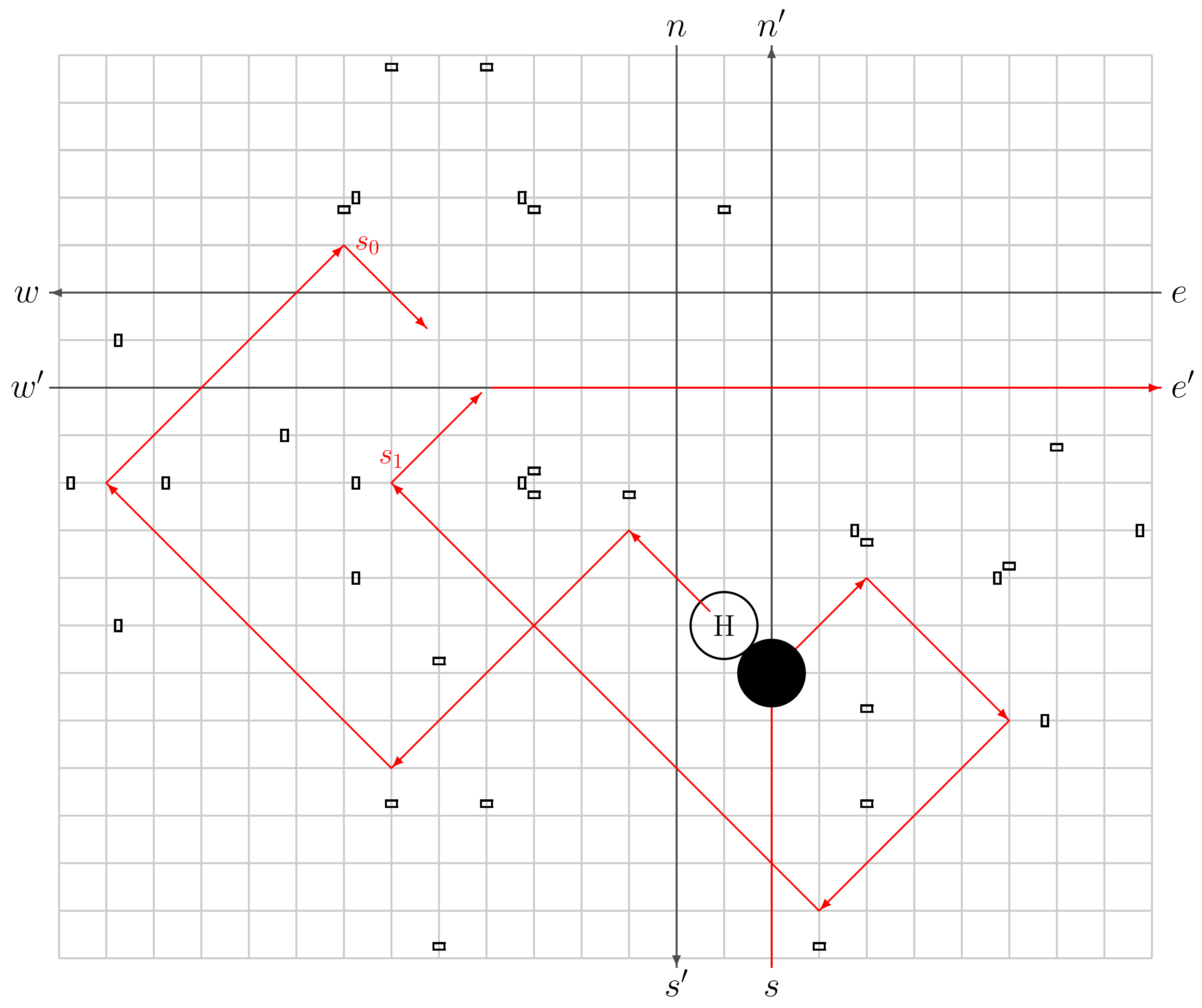
## Movements of Balls (State: $H$, Input: $s$)



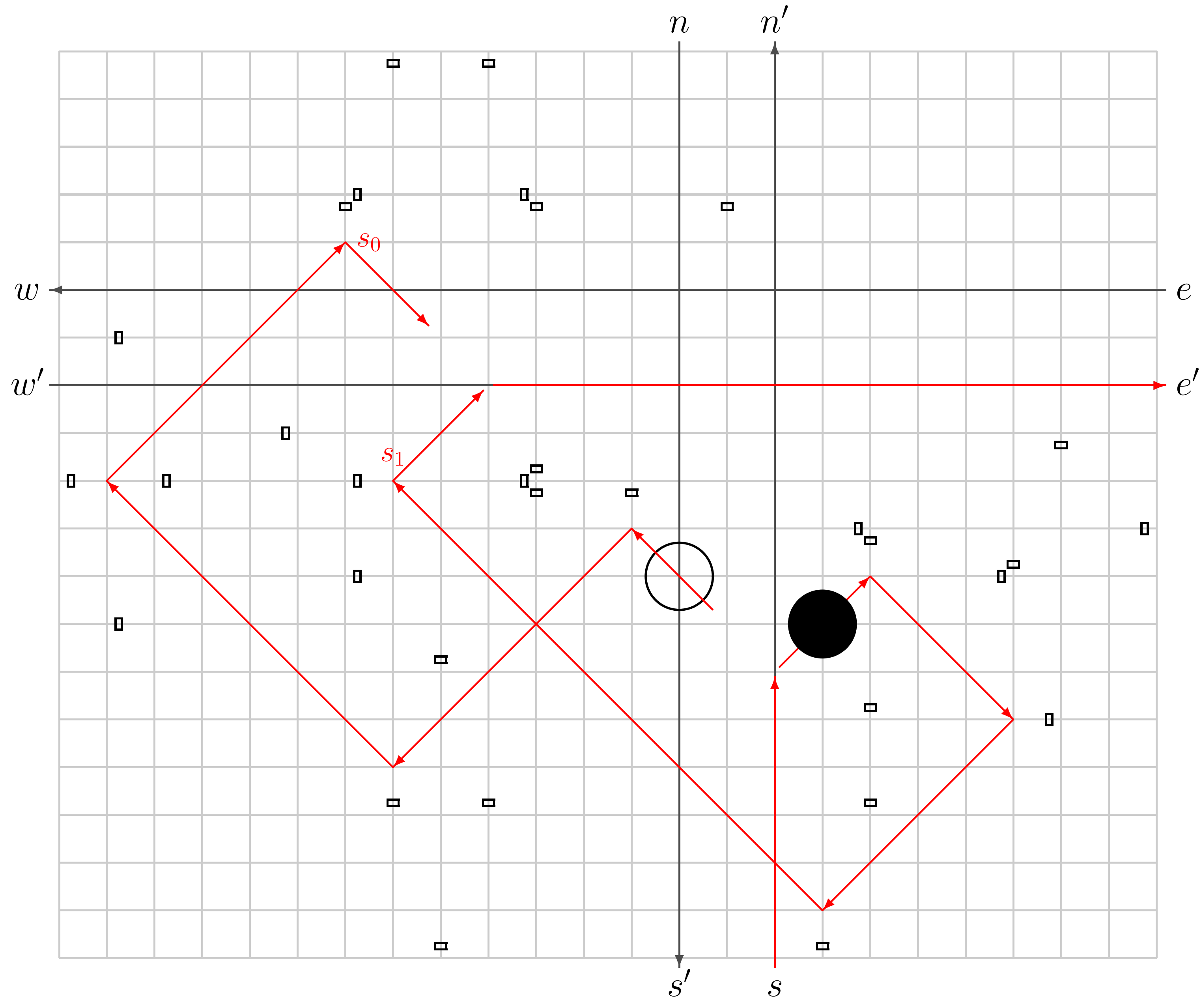
## Movements of Balls (State: $H$, Input: $s$)



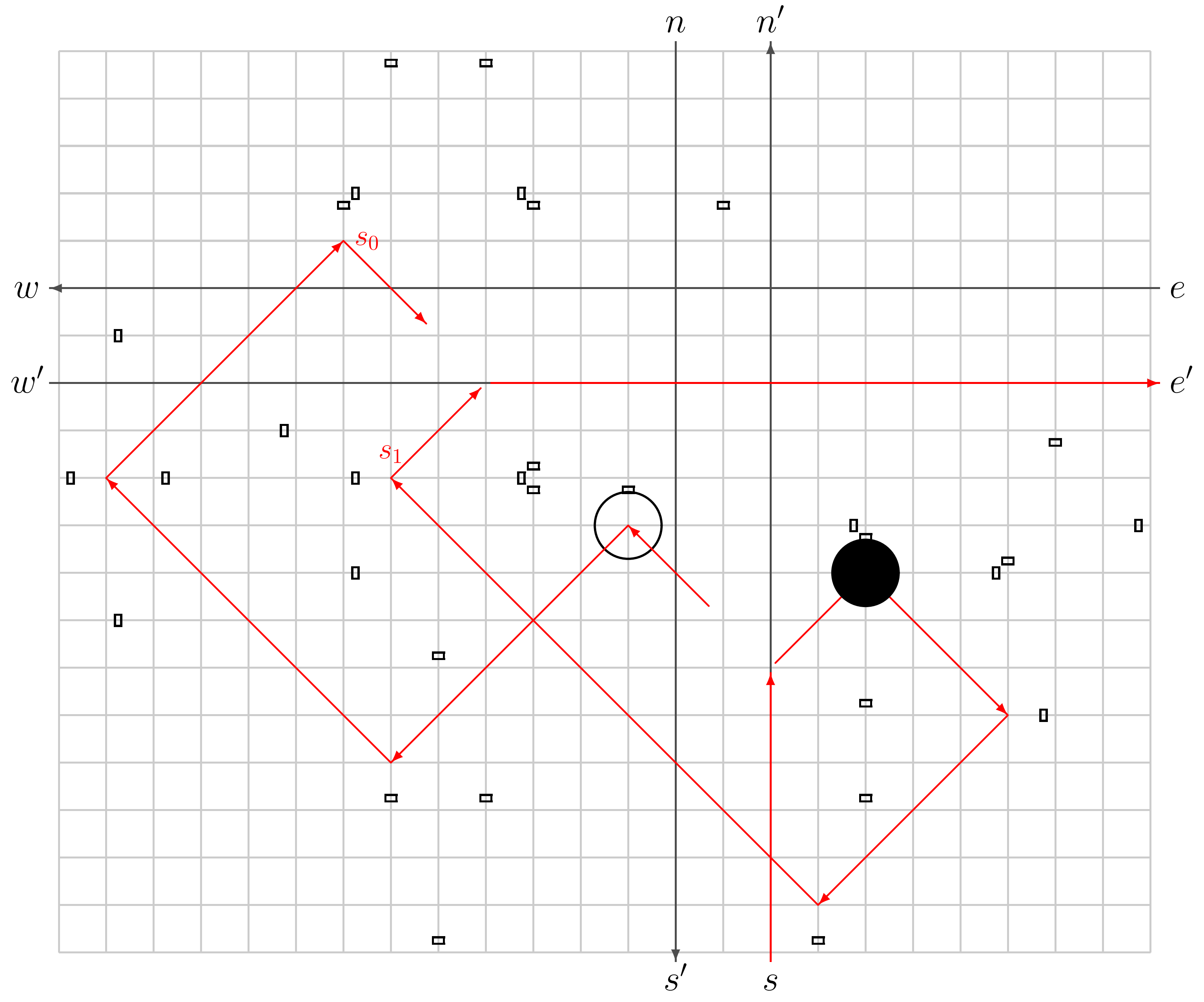
## Movements of Balls (State: $H$, Input: $s$)



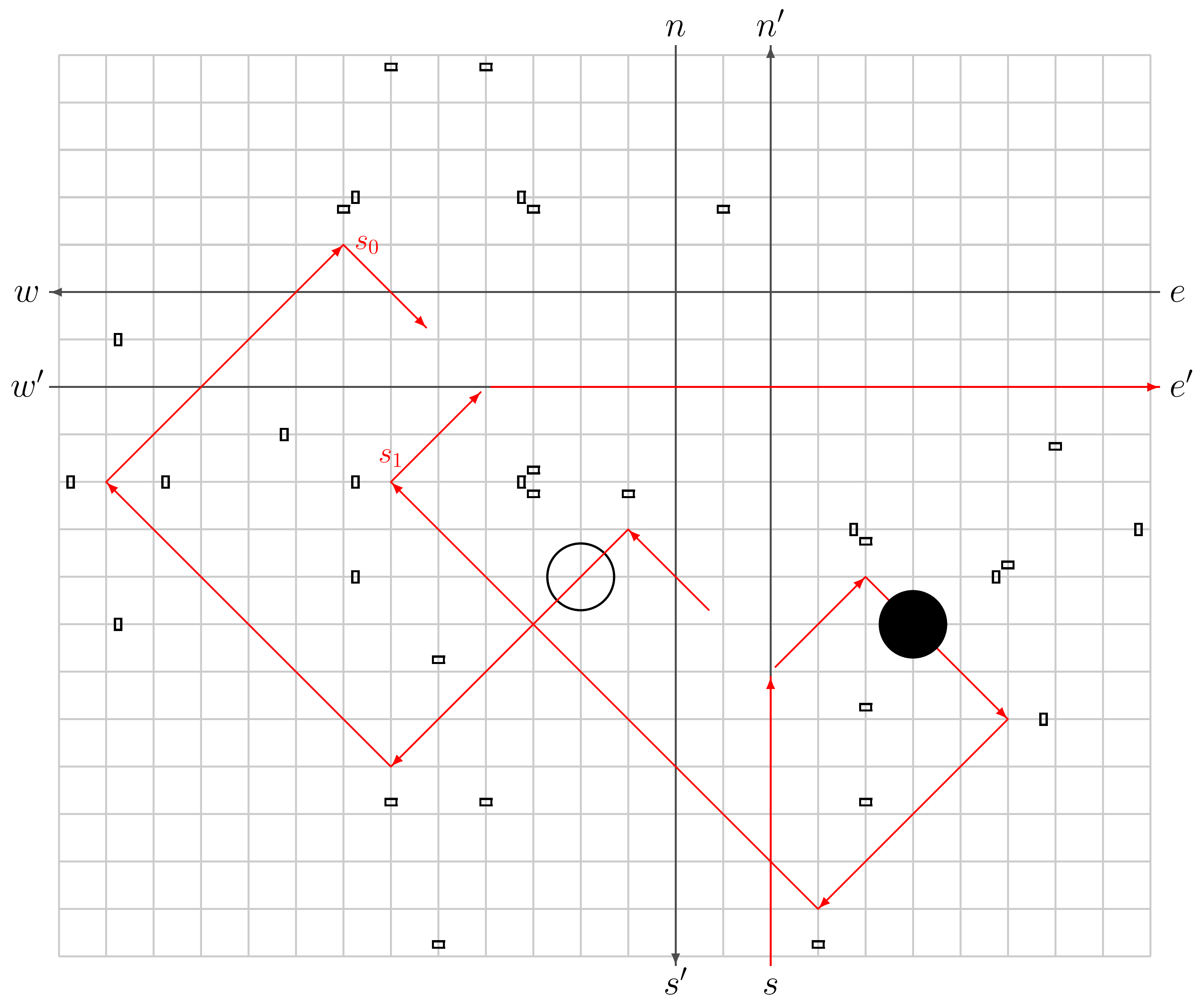
## Movements of Balls (State: $H$, Input: $s$)



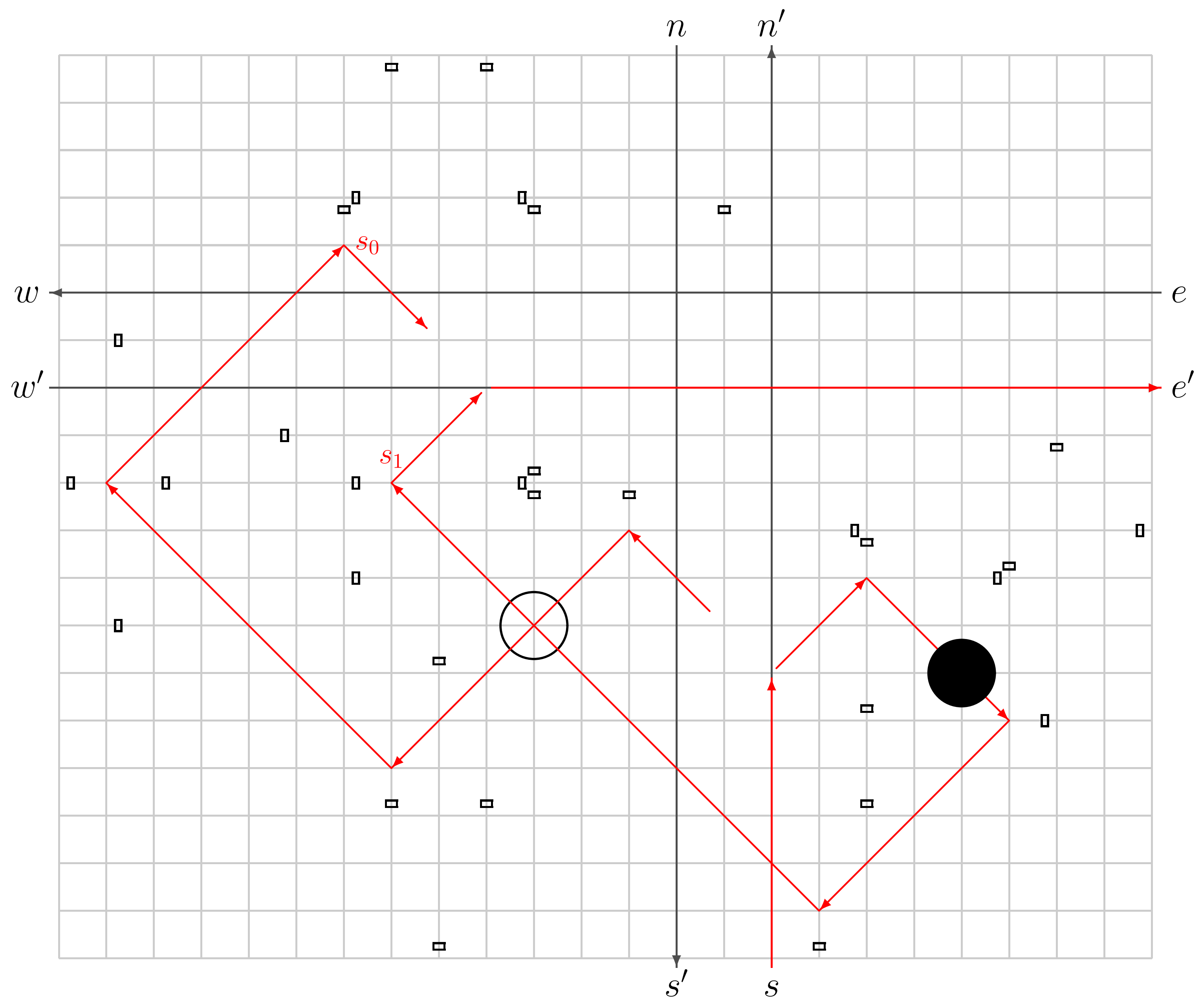
## Movements of Balls (State: $H$, Input: $s$)



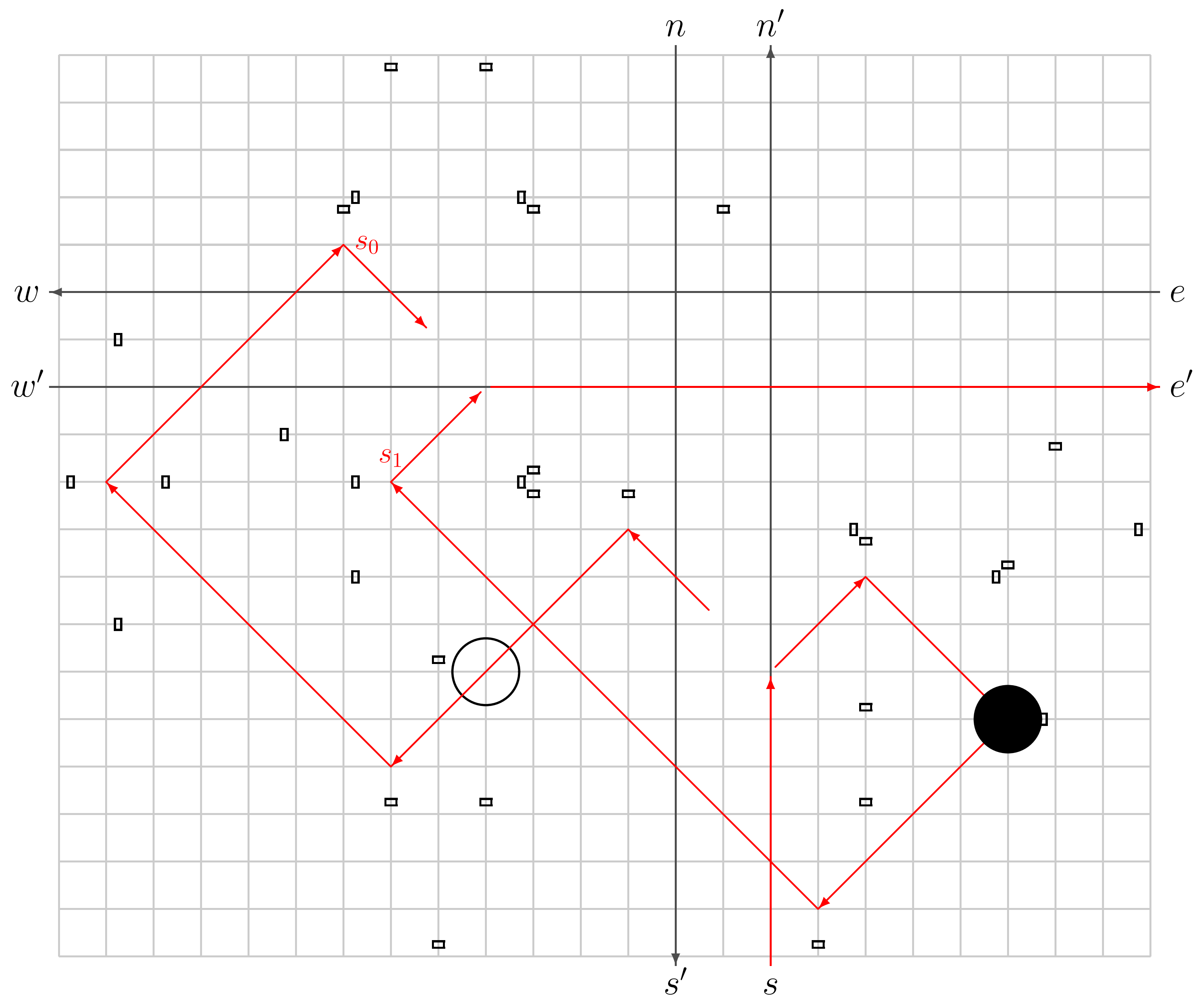
## Movements of Balls (State: $H$, Input: $s$)



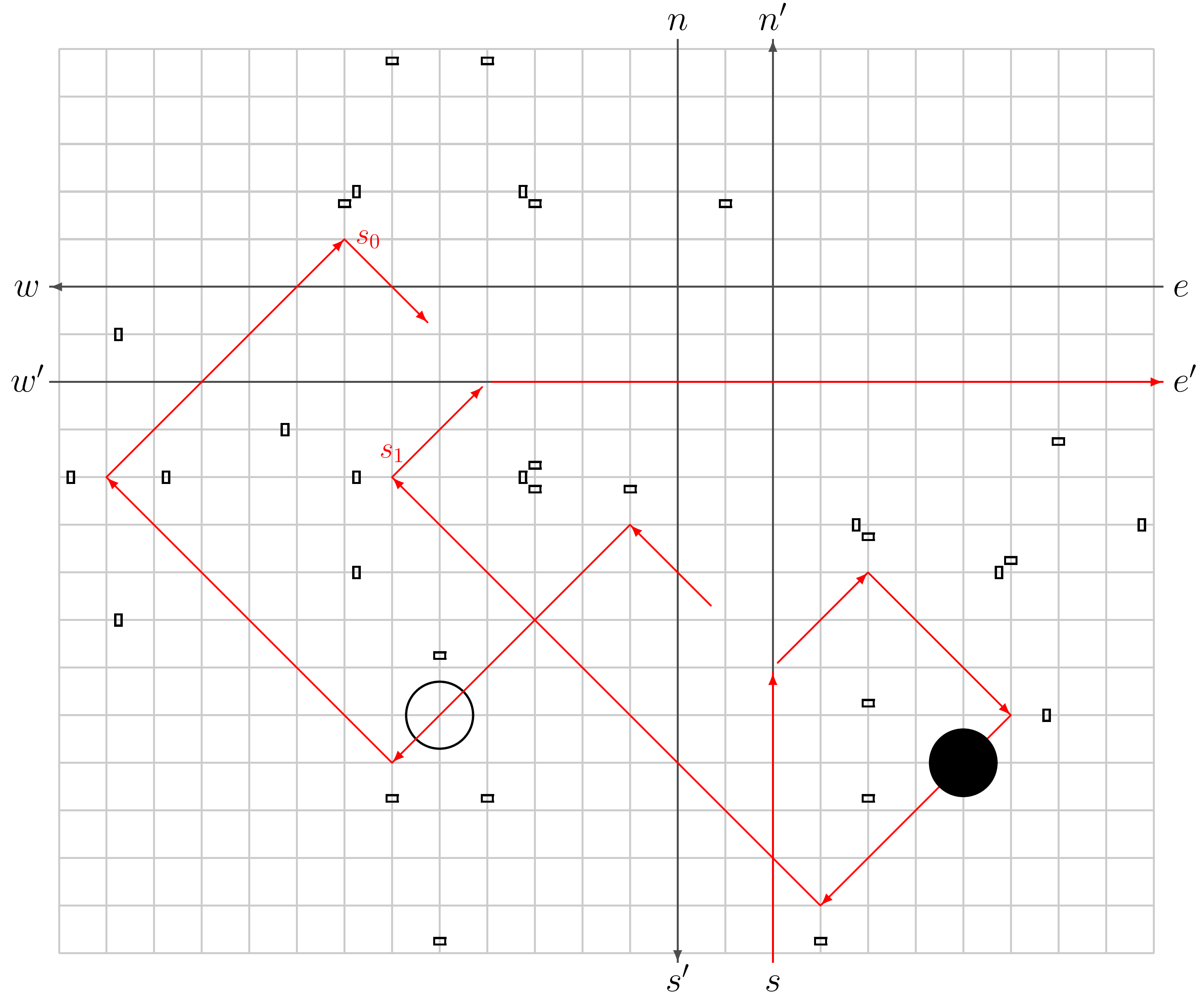
## Movements of Balls (State: $H$, Input: $s$)



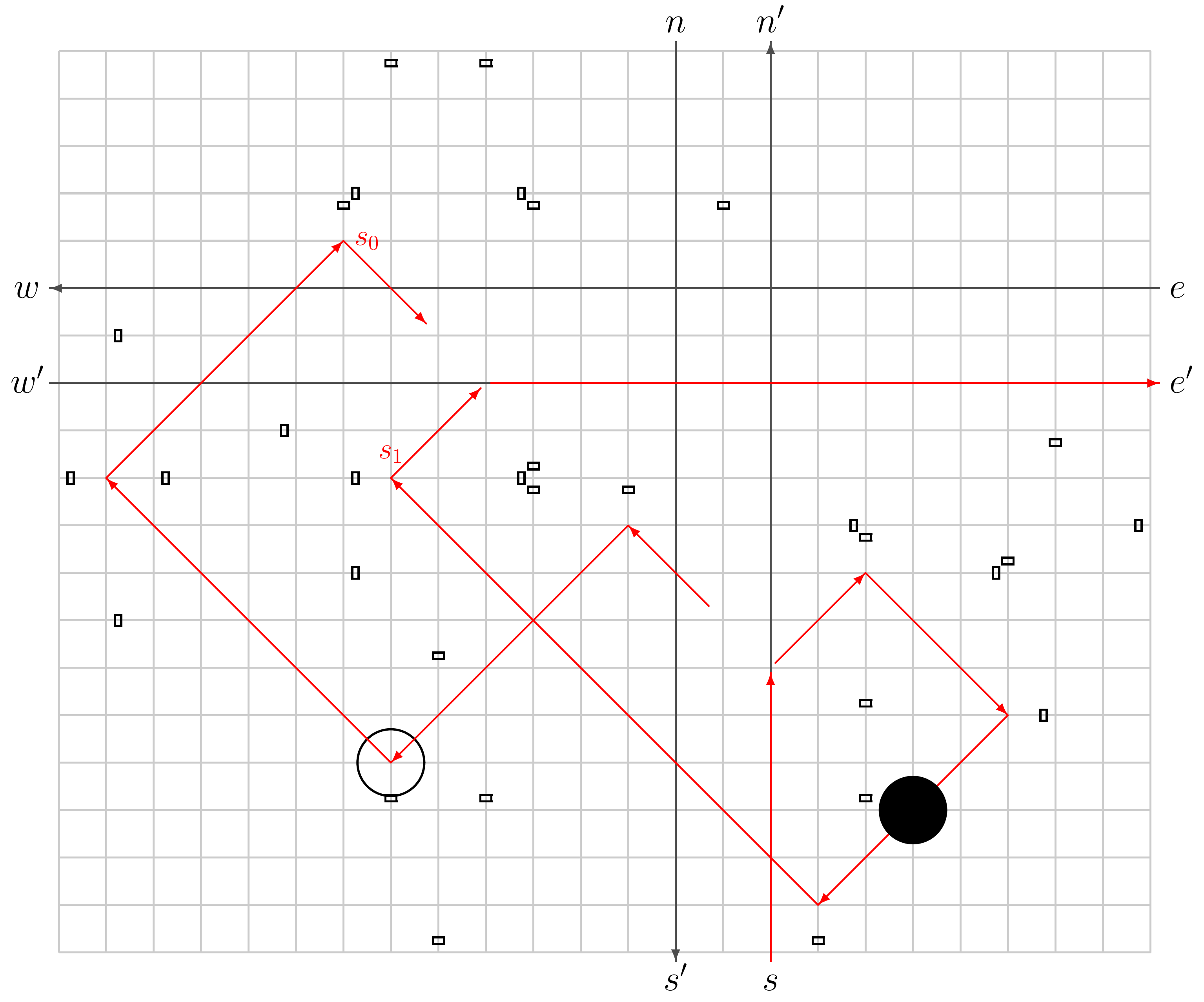
# Movements of Balls (State: $H$, Input: $s$)



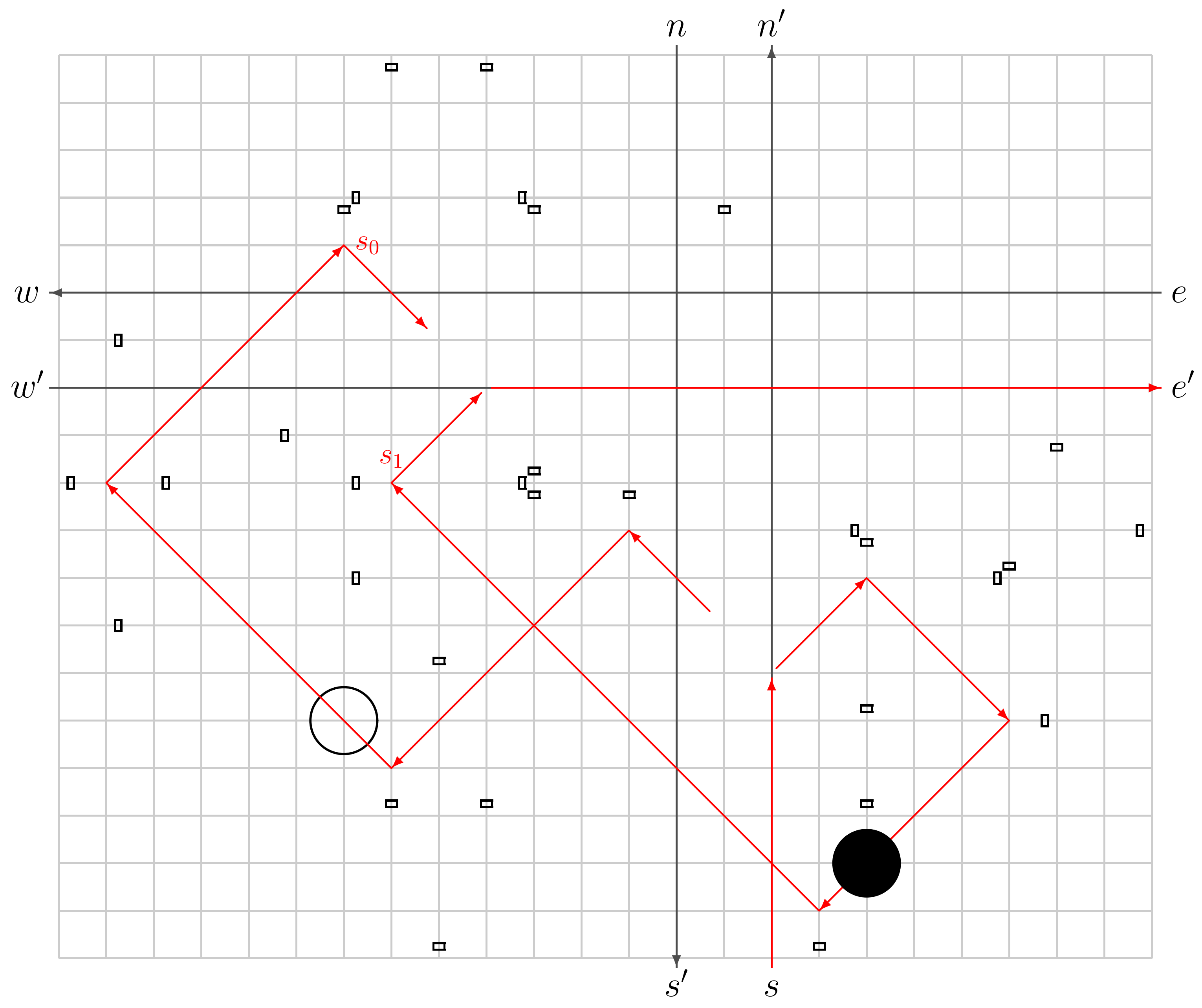
# Movements of Balls (State: $H$, Input: $s$)



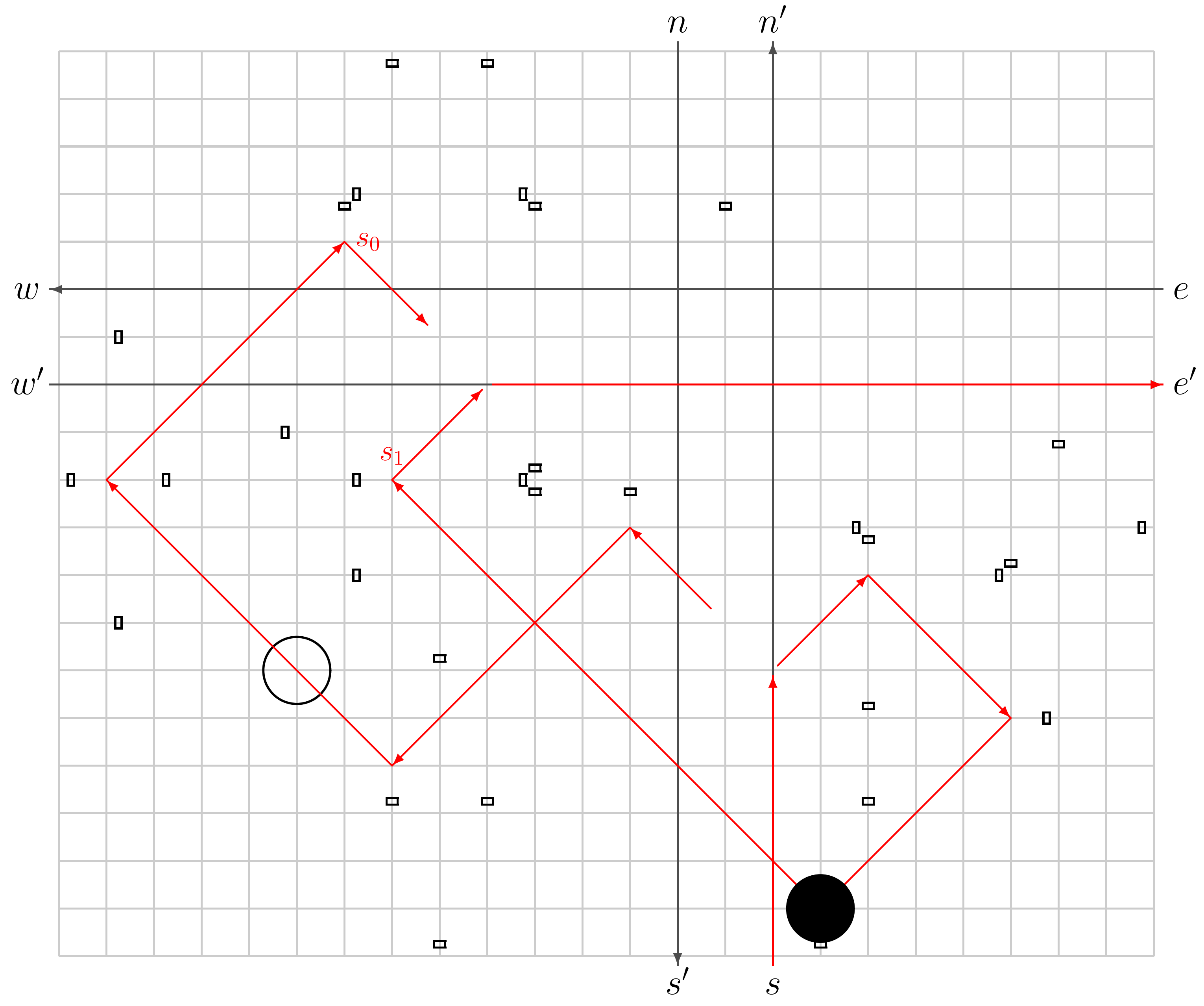
## Movements of Balls (State: $H$, Input: $s$)



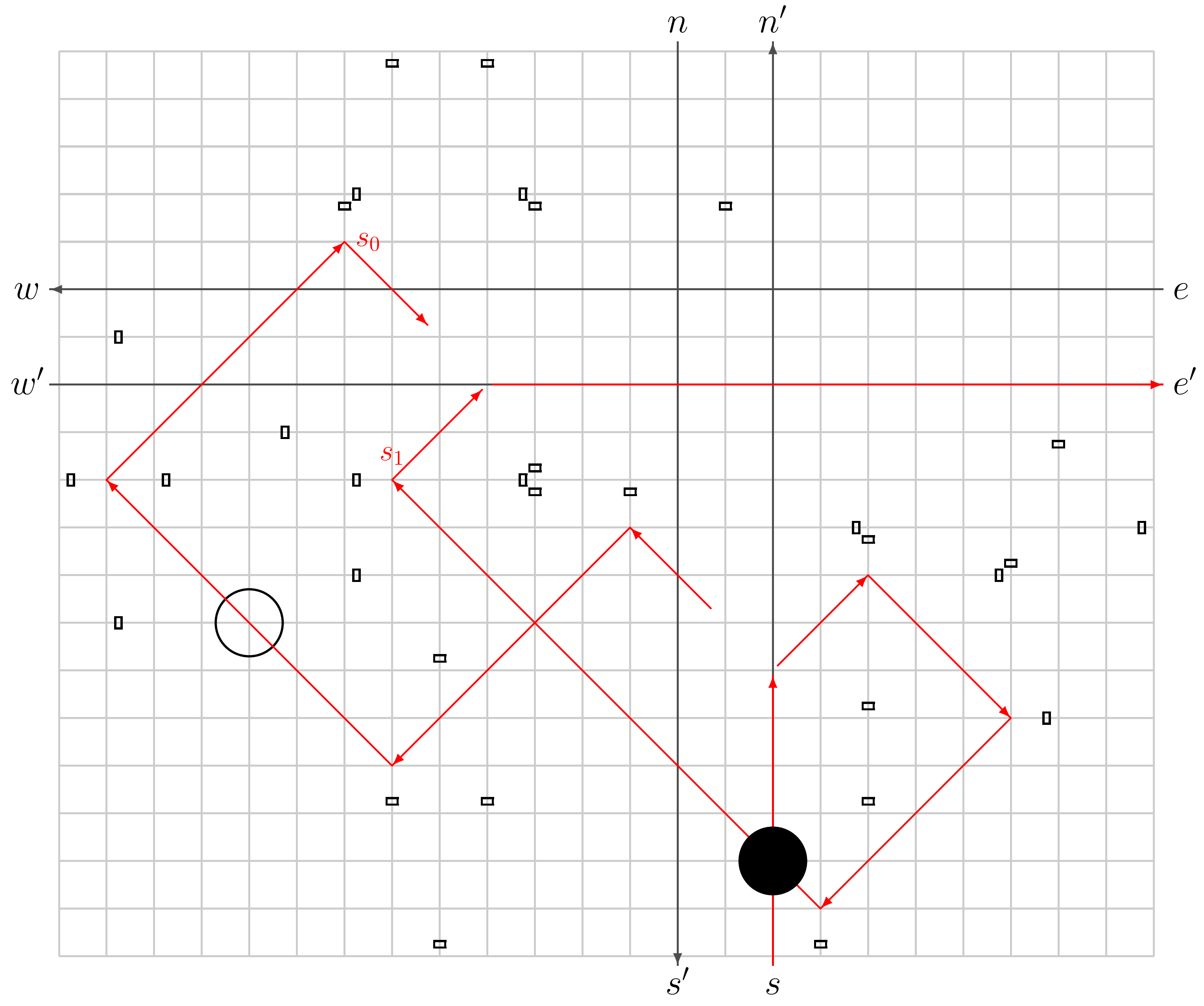
## Movements of Balls (State: $H$, Input: $s$)



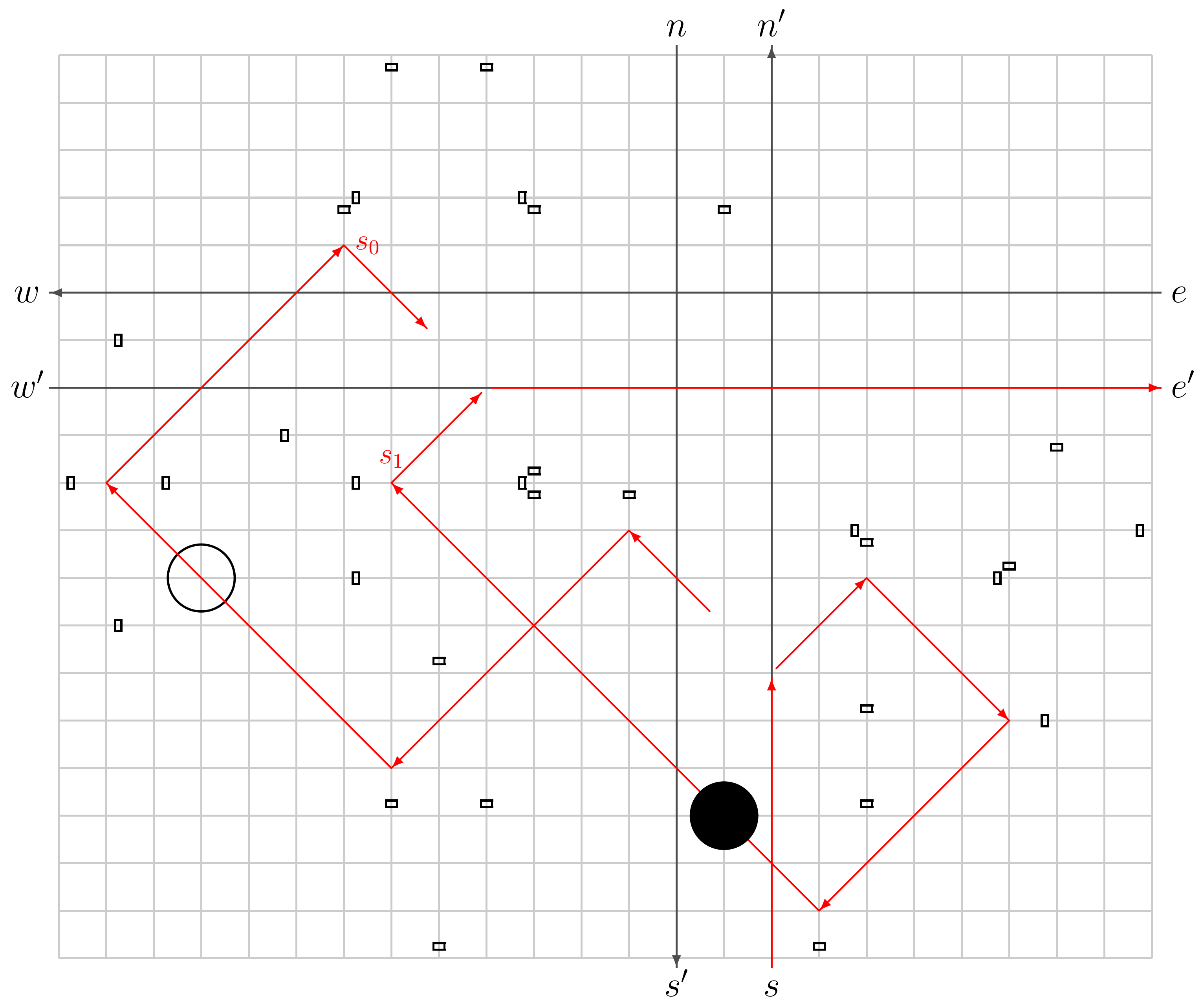
## Movements of Balls (State: $H$, Input: $s$)



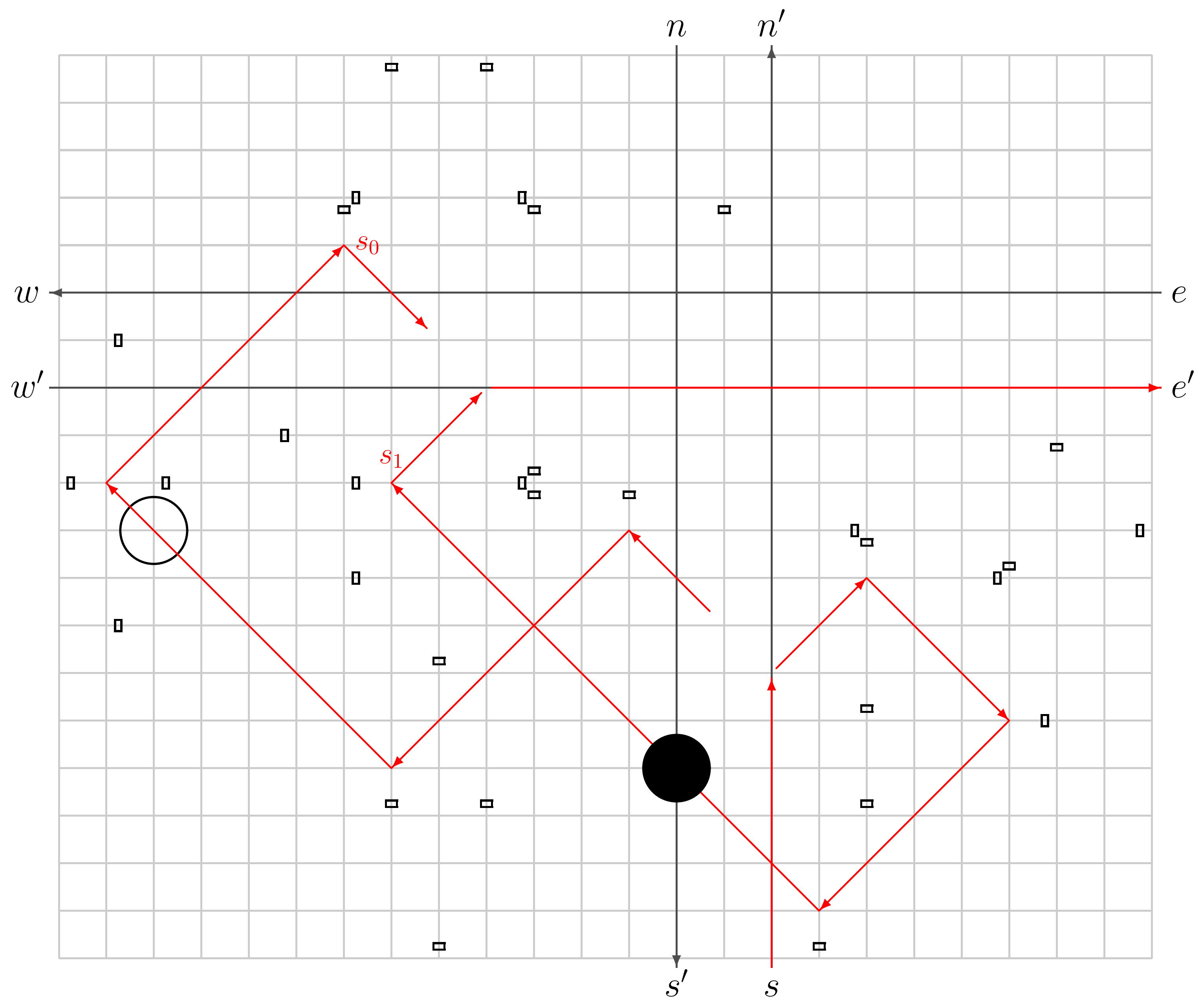
## Movements of Balls (State: $H$, Input: $s$)



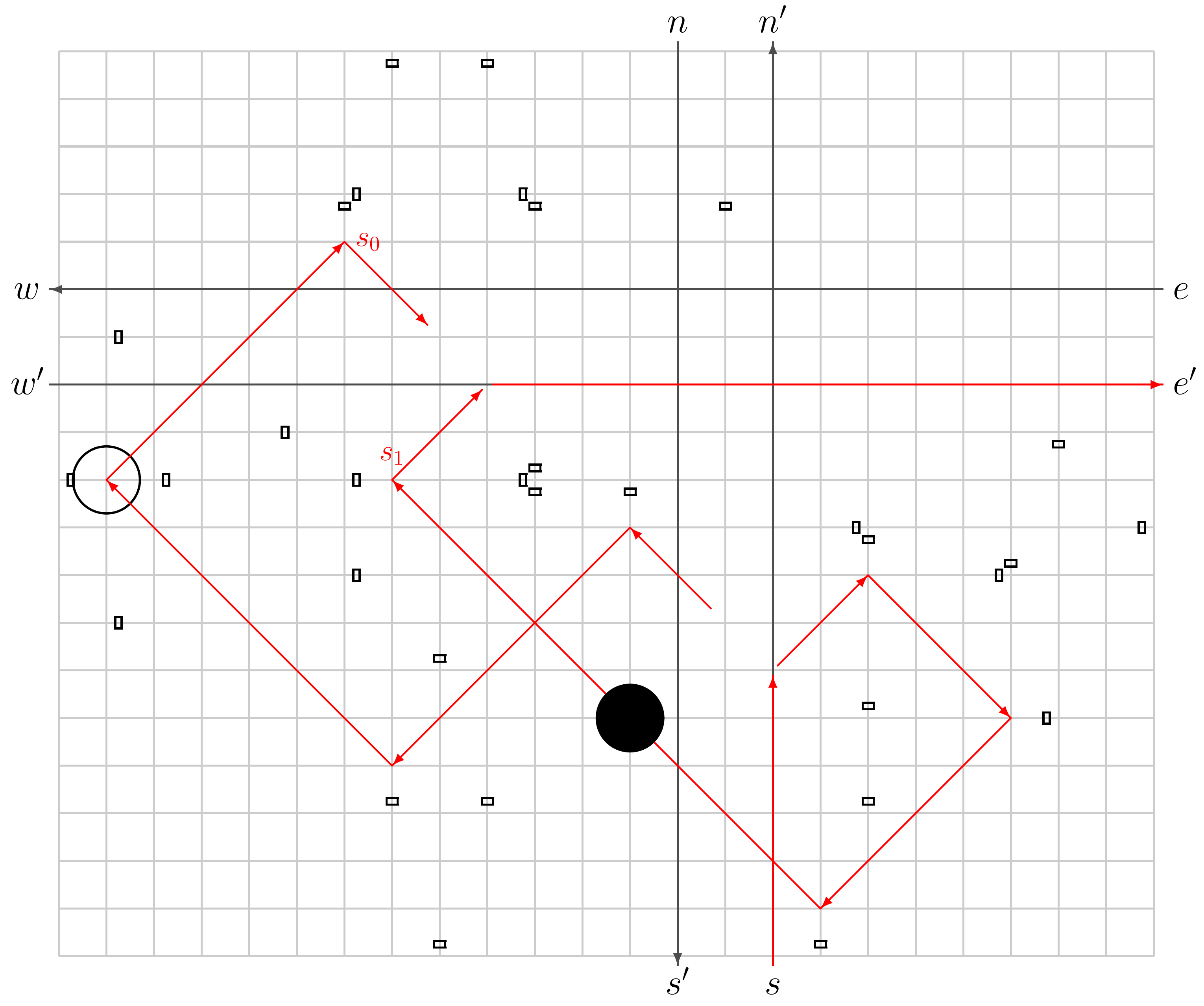
## Movements of Balls (State: $H$, Input: $s$)



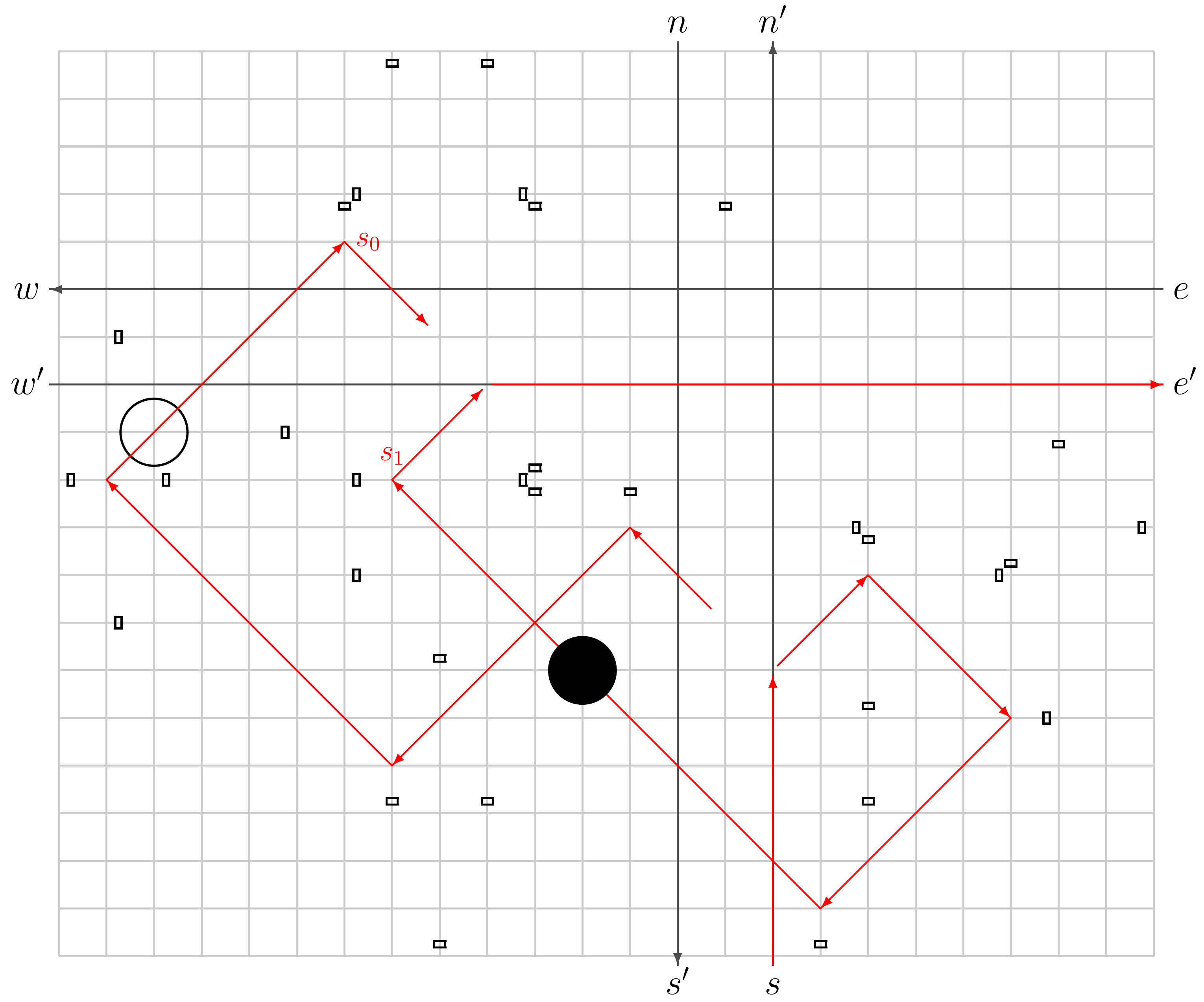
## Movements of Balls (State: $H$, Input: $s$)



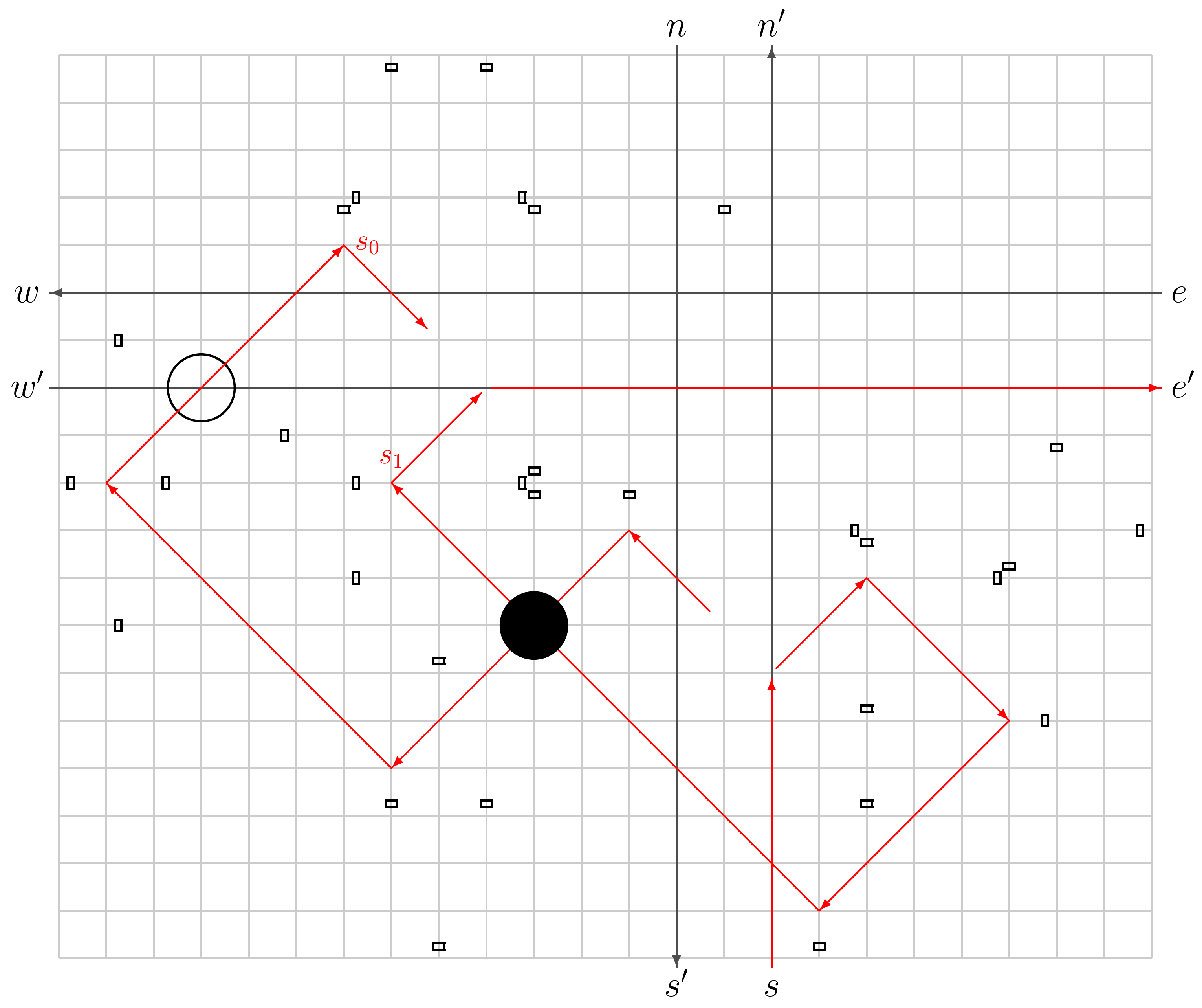
## Movements of Balls (State: $H$, Input: $s$)



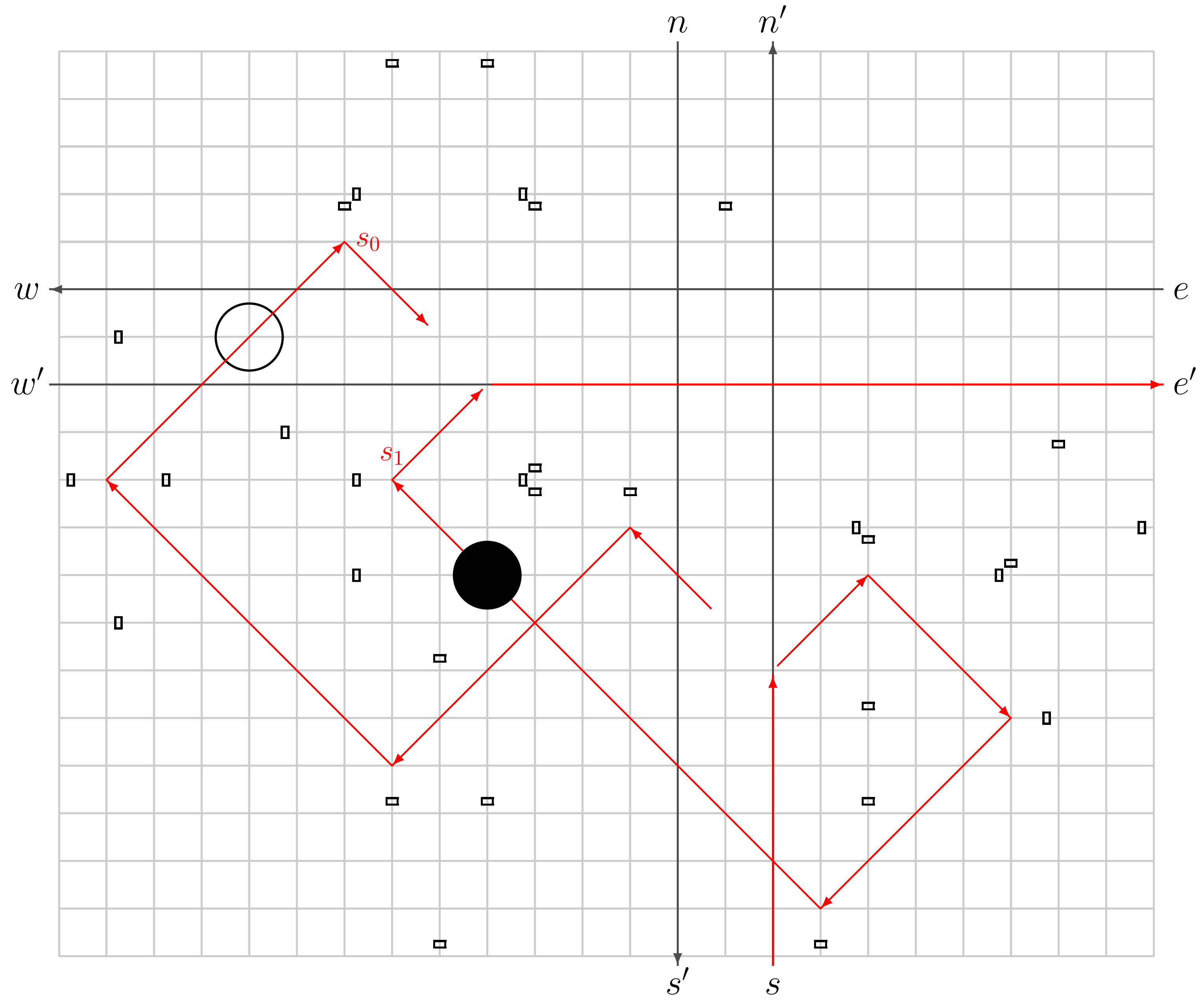
## Movements of Balls (State: $H$, Input: $s$)



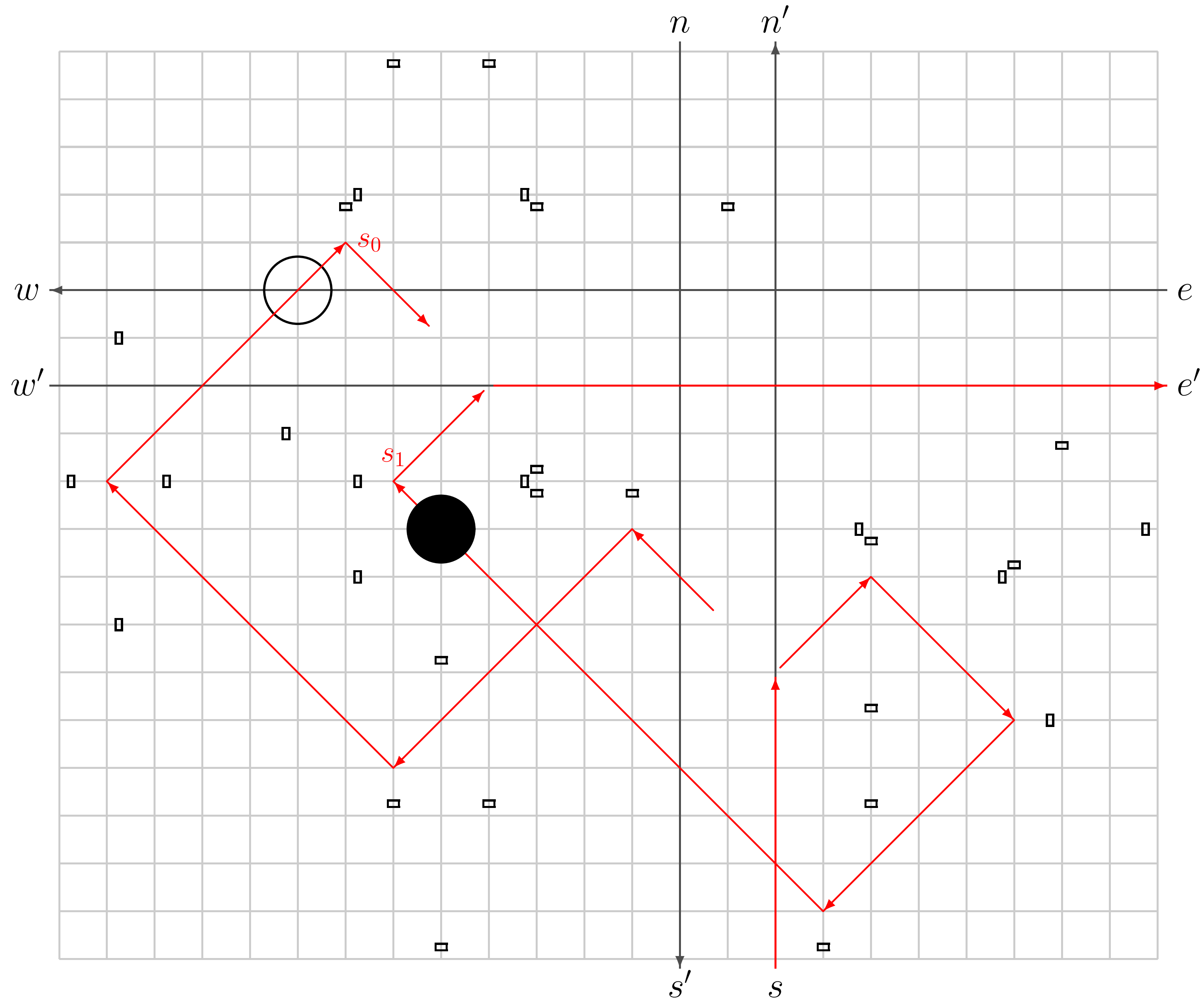
## Movements of Balls (State: $H$, Input: $s$)



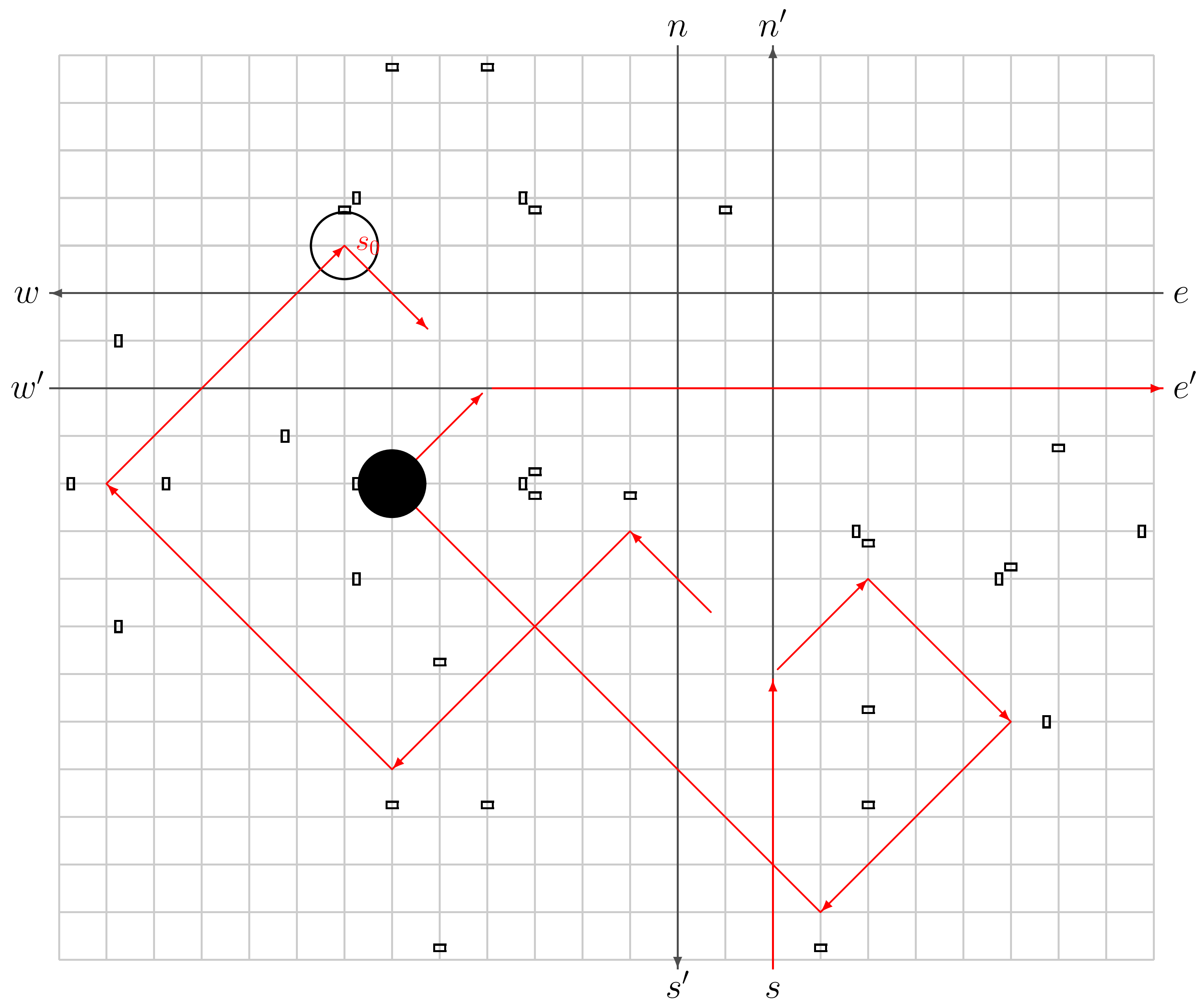
## Movements of Balls (State: $H$, Input: $s$)



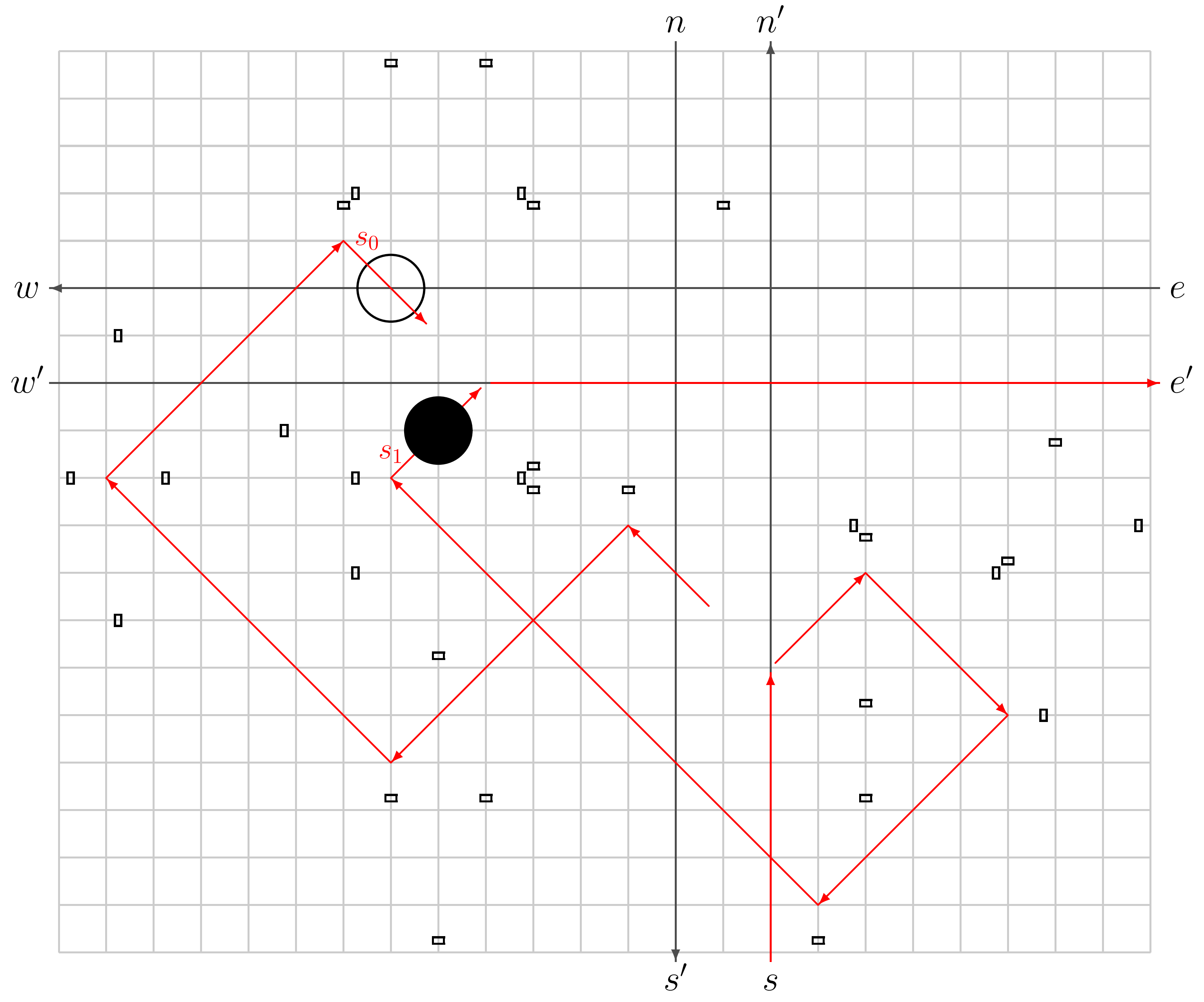
## Movements of Balls (State: $H$, Input: $s$)



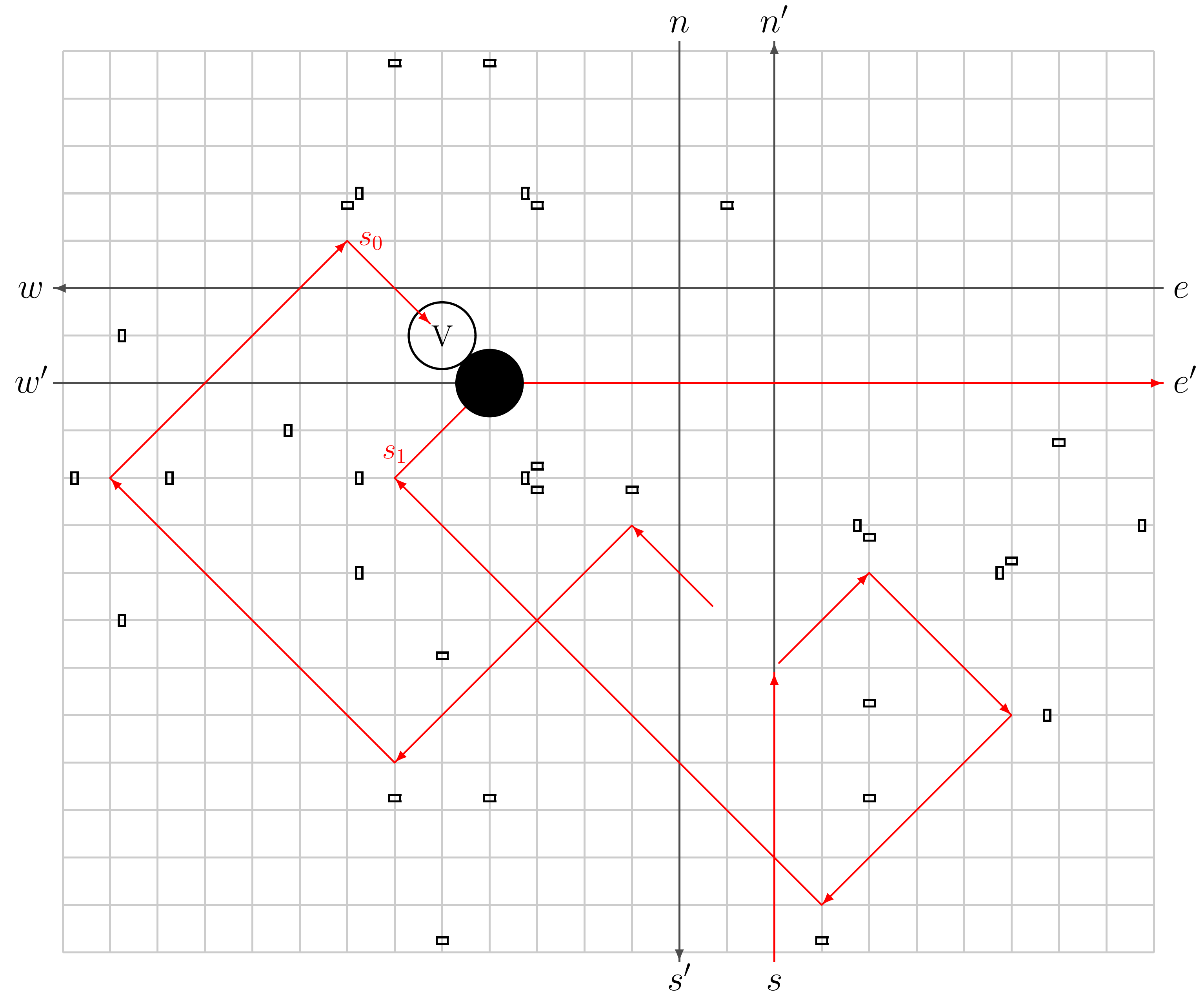
## Movements of Balls (State: $H$, Input: $s$)



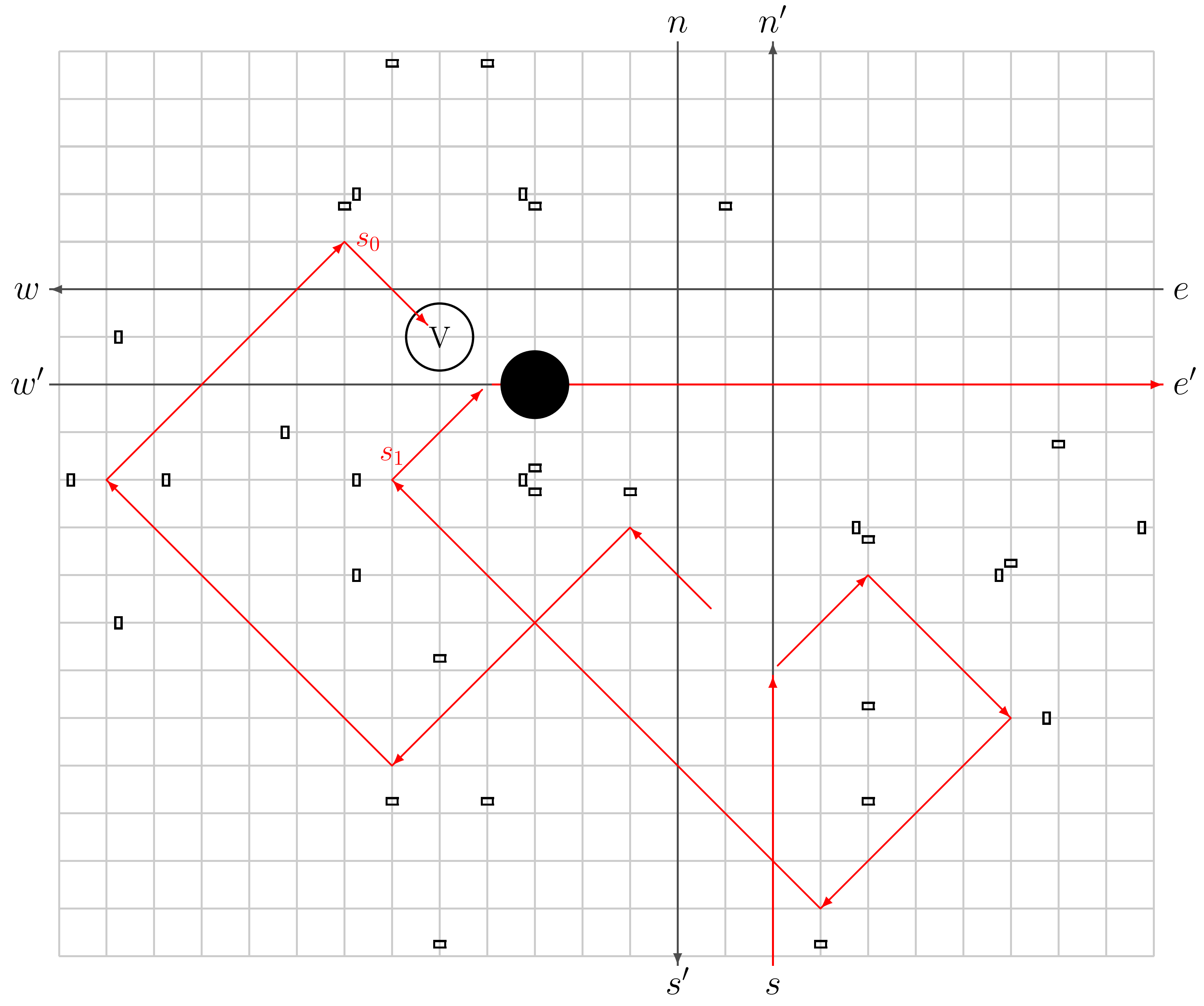
## Movements of Balls (State: $H$, Input: $s$)



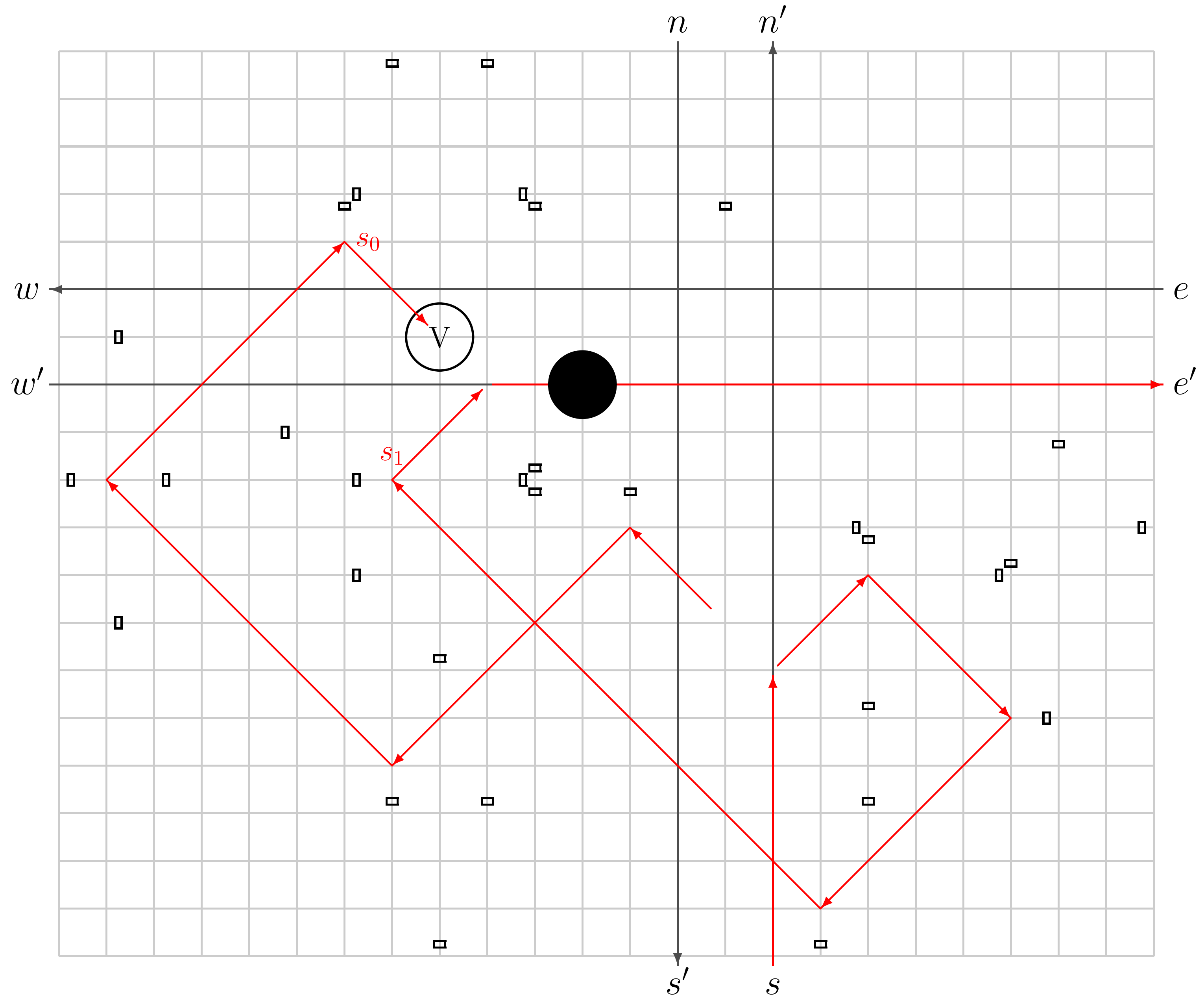
## Movements of Balls (State: $H$, Input: $s$)



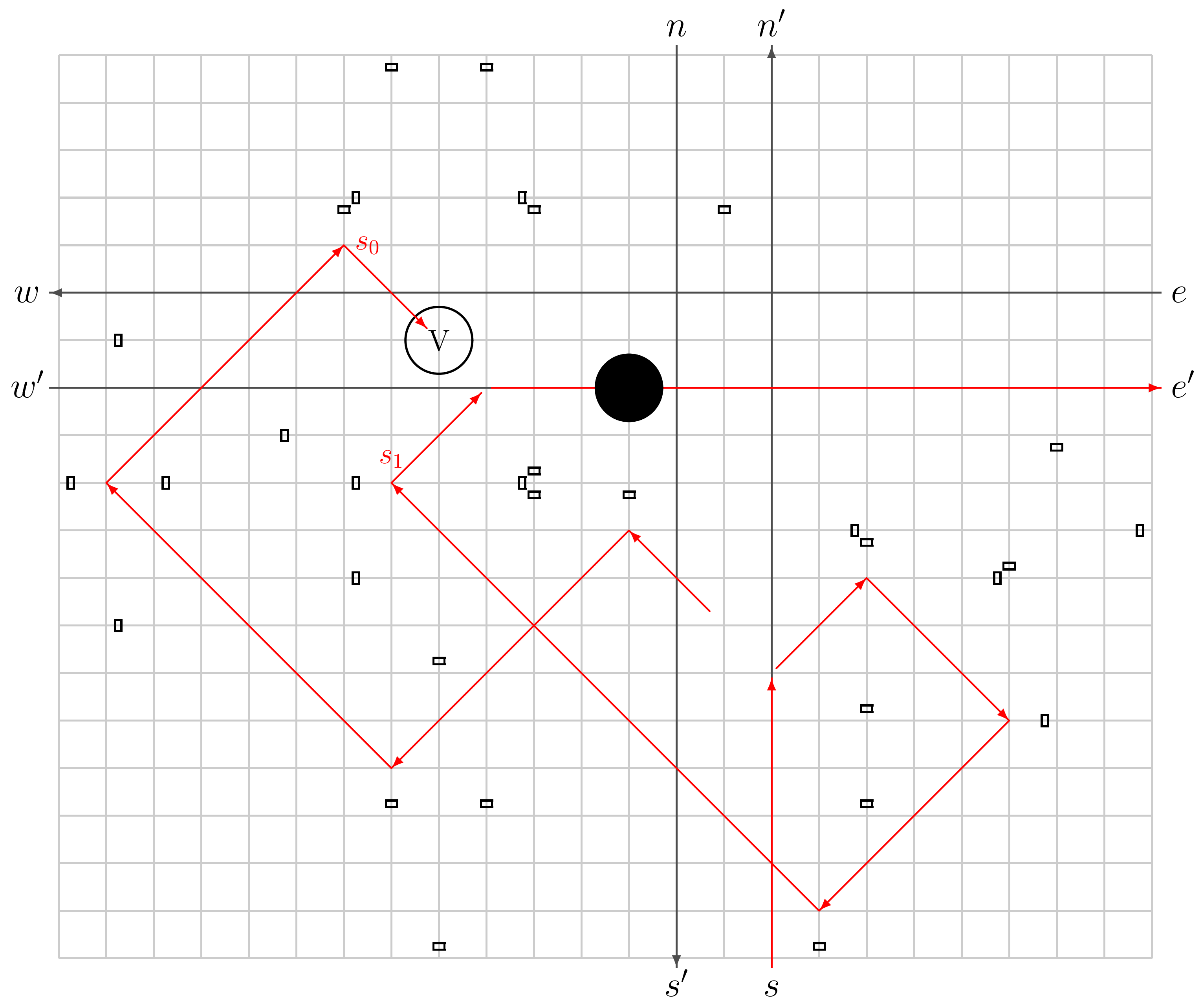
# Movements of Balls (State: $H$, Input: $s$)



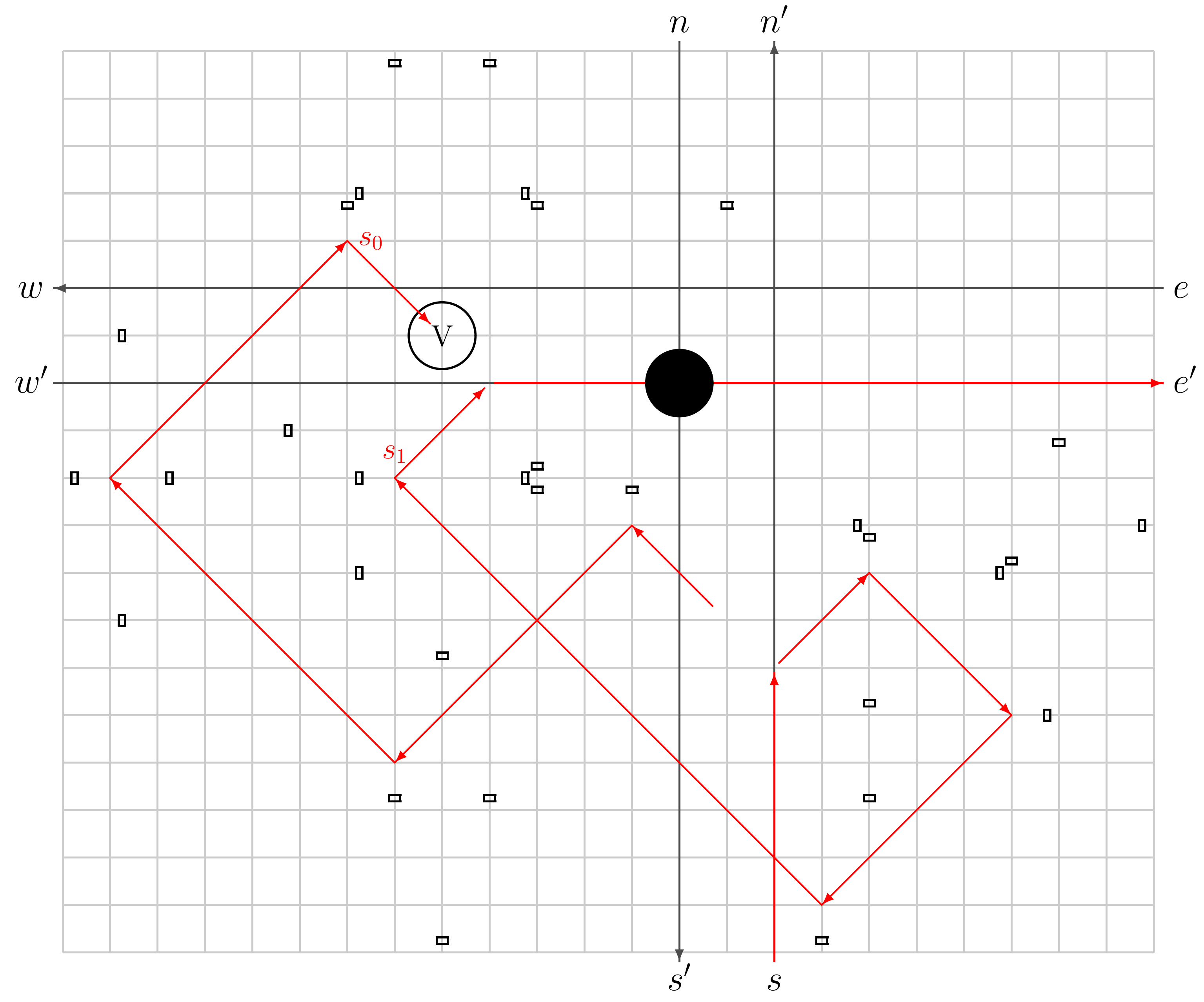
## Movements of Balls (State: $H$, Input: $s$)



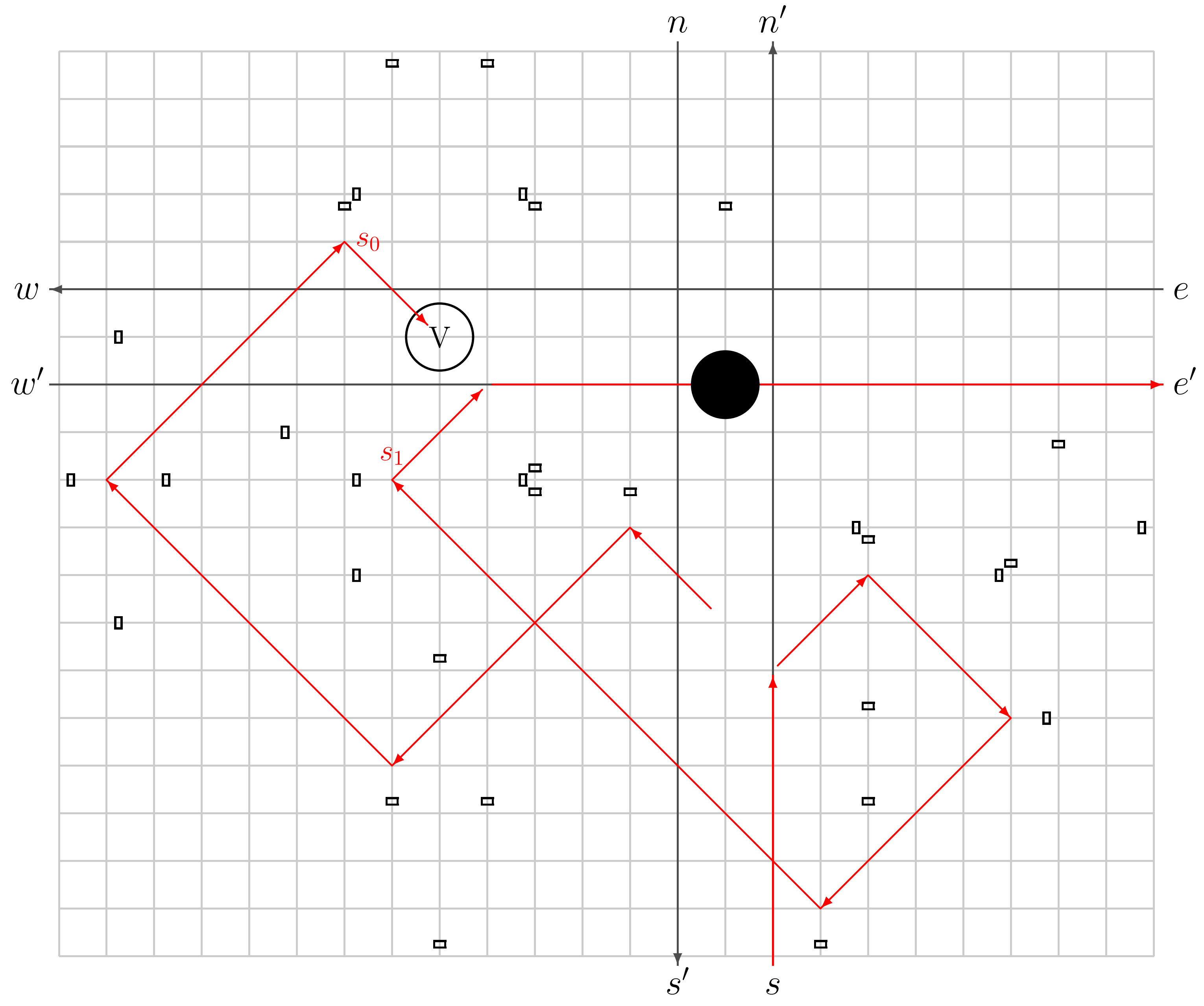
## Movements of Balls (State: $H$, Input: $s$)



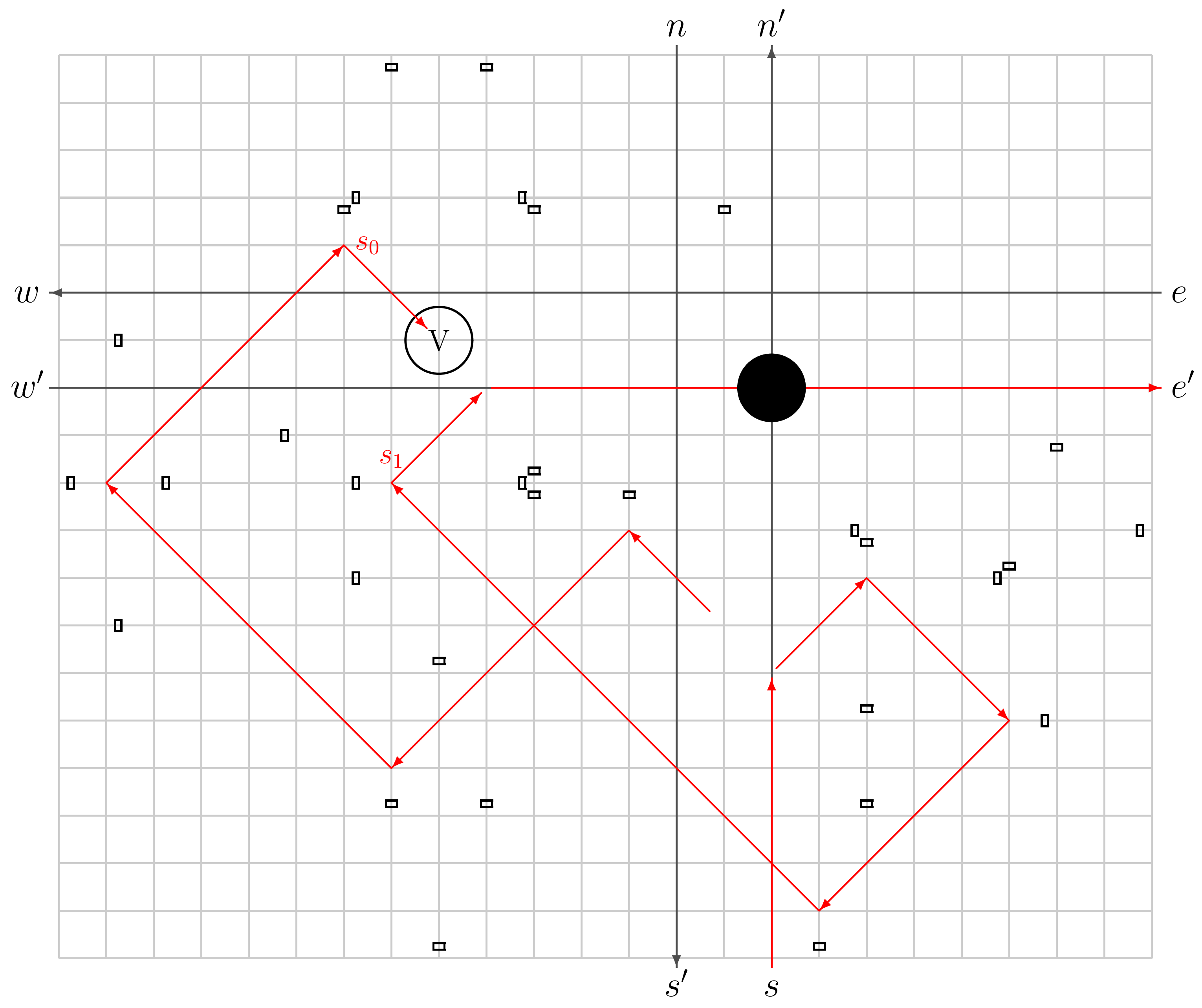
## Movements of Balls (State: $H$, Input: $s$)



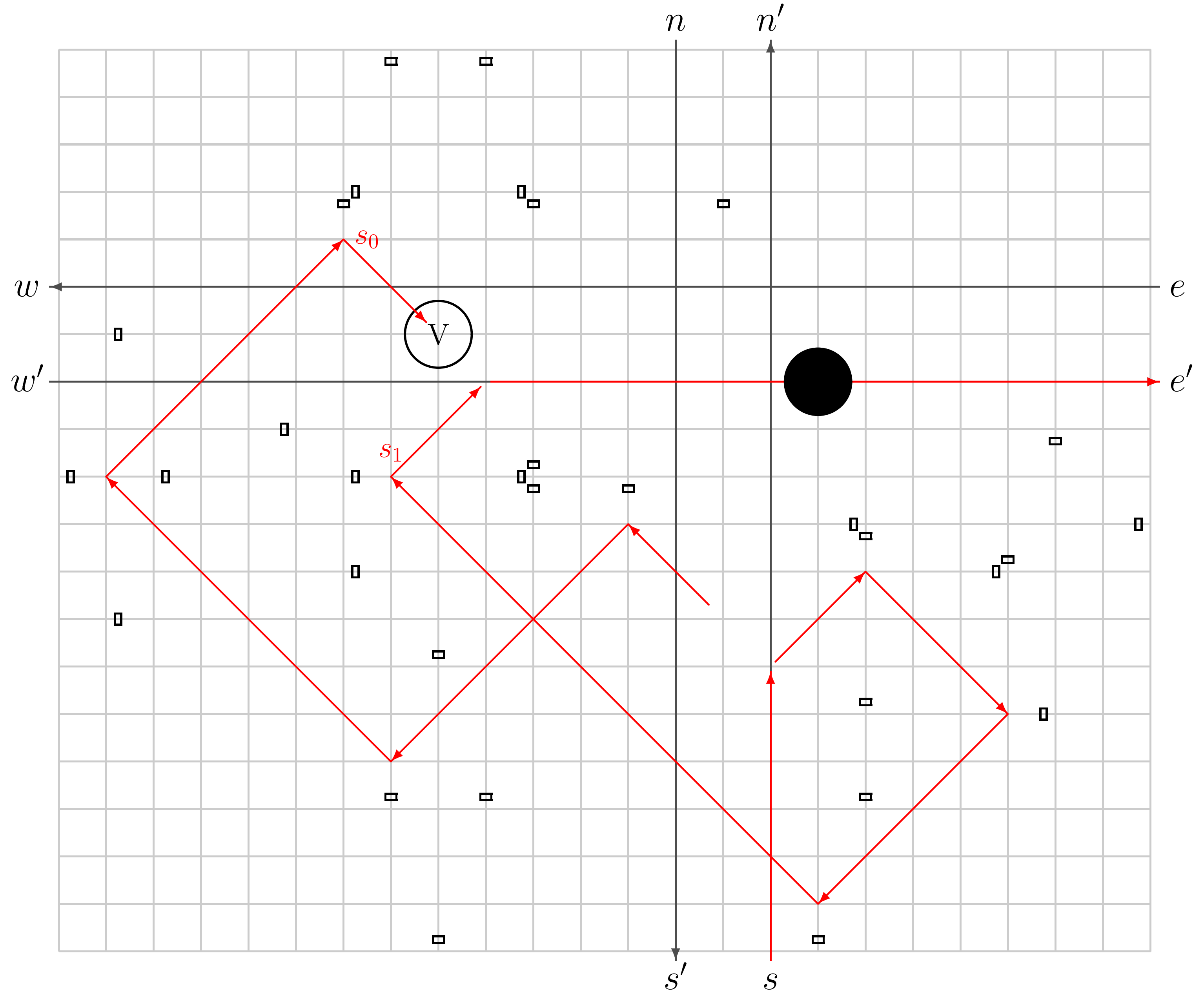
## Movements of Balls (State: $H$, Input: $s$)



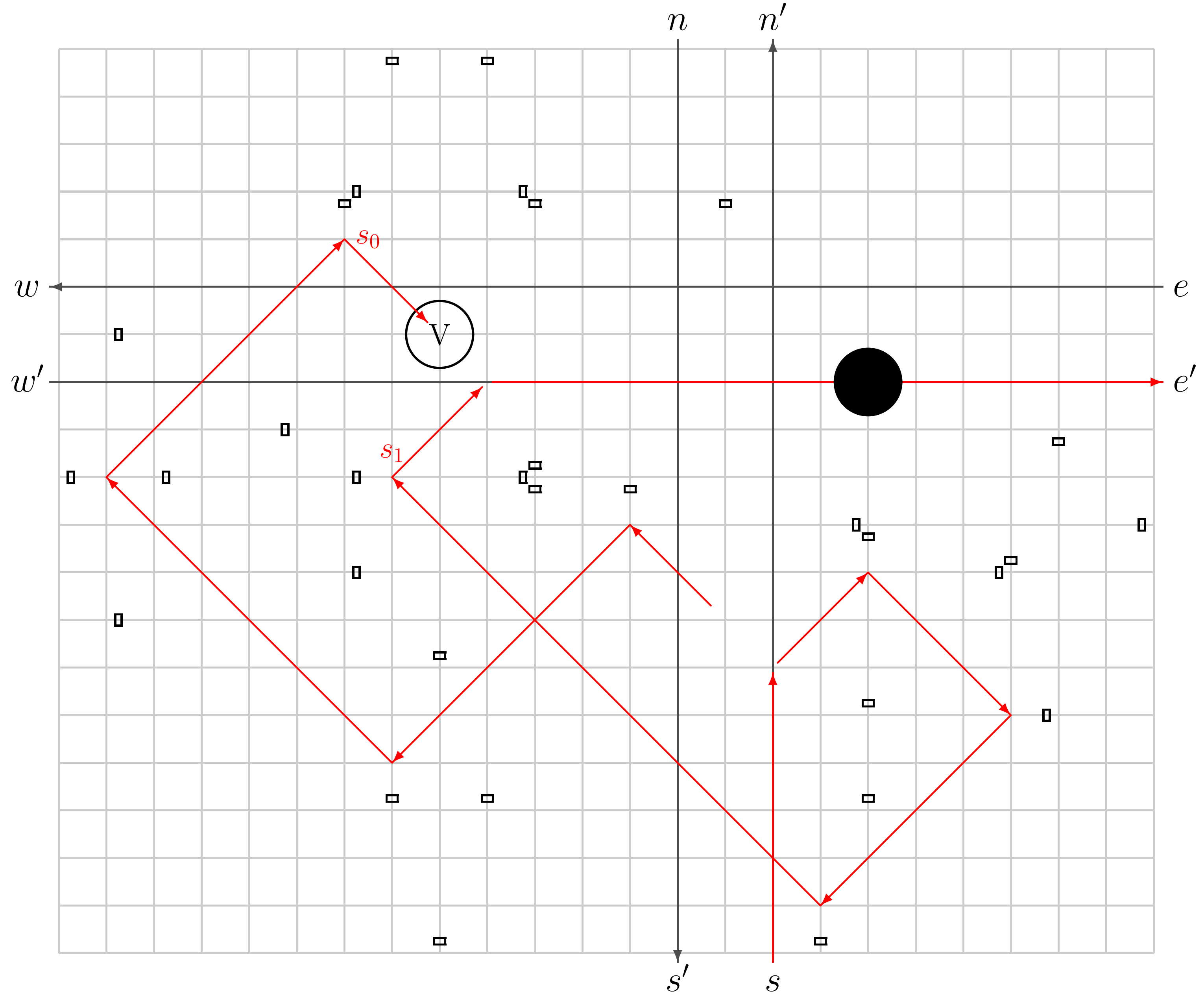
## Movements of Balls (State: $H$, Input: $s$)



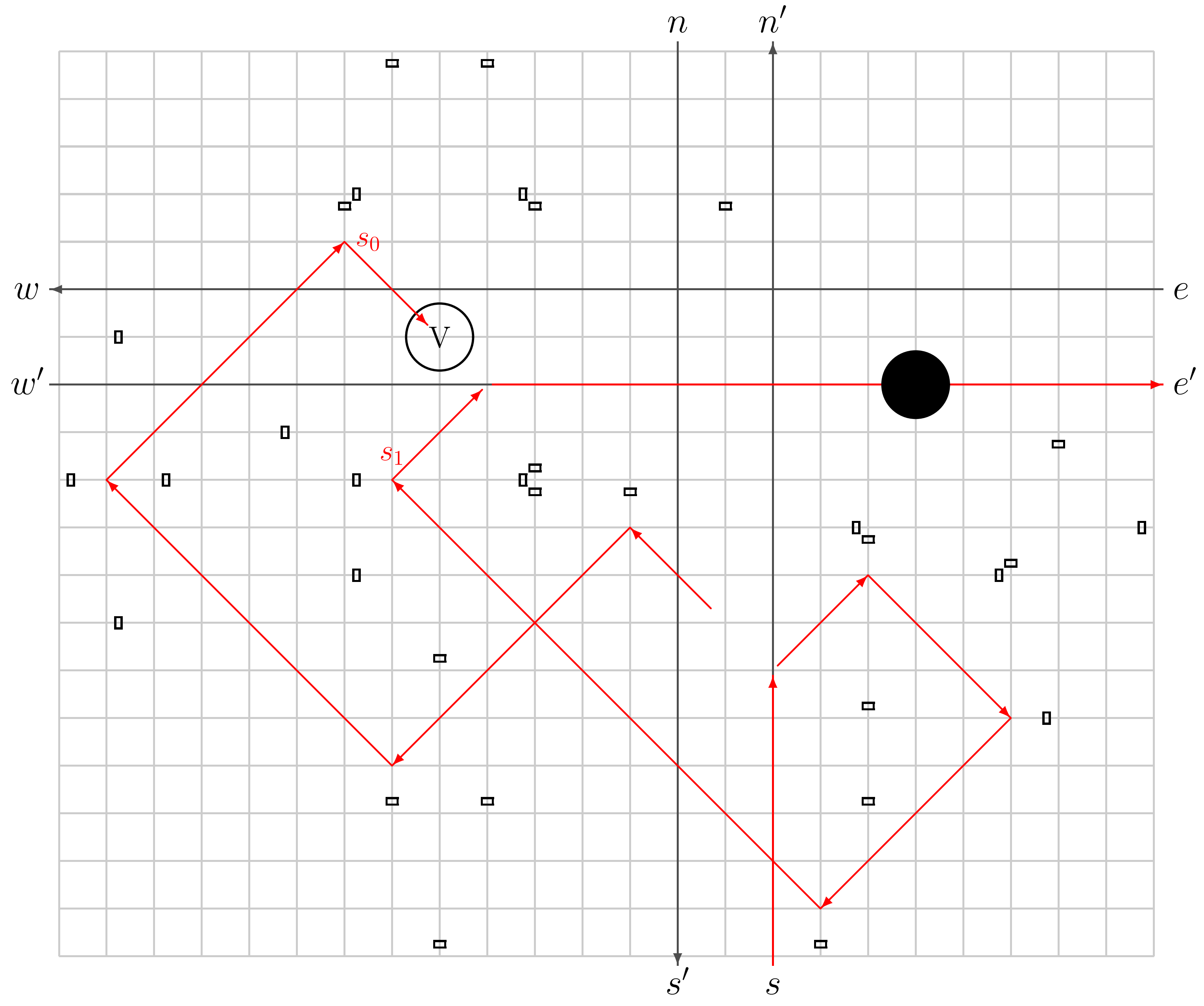
## Movements of Balls (State: $H$, Input: $s$)



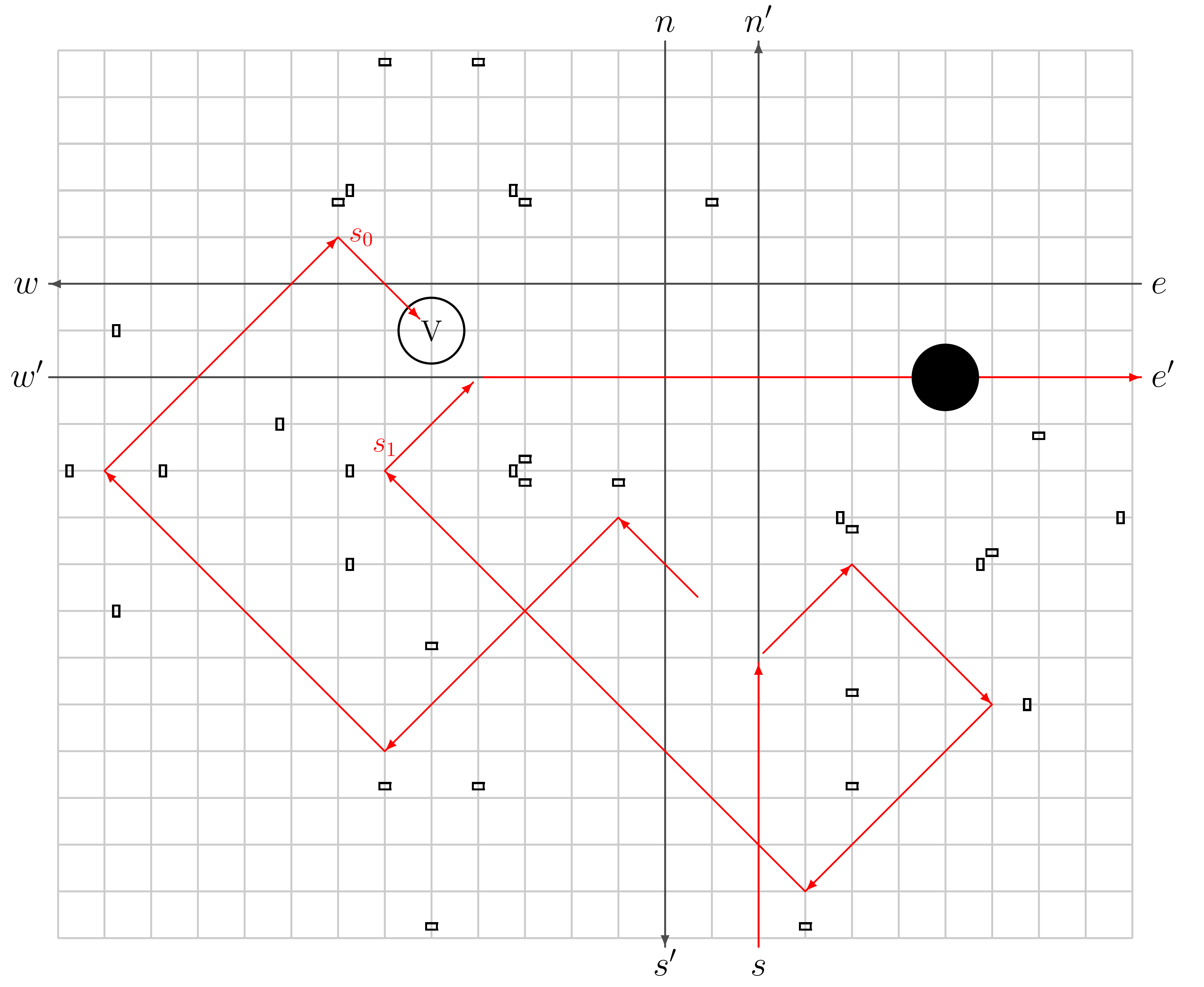
## Movements of Balls (State: $H$, Input: $s$)



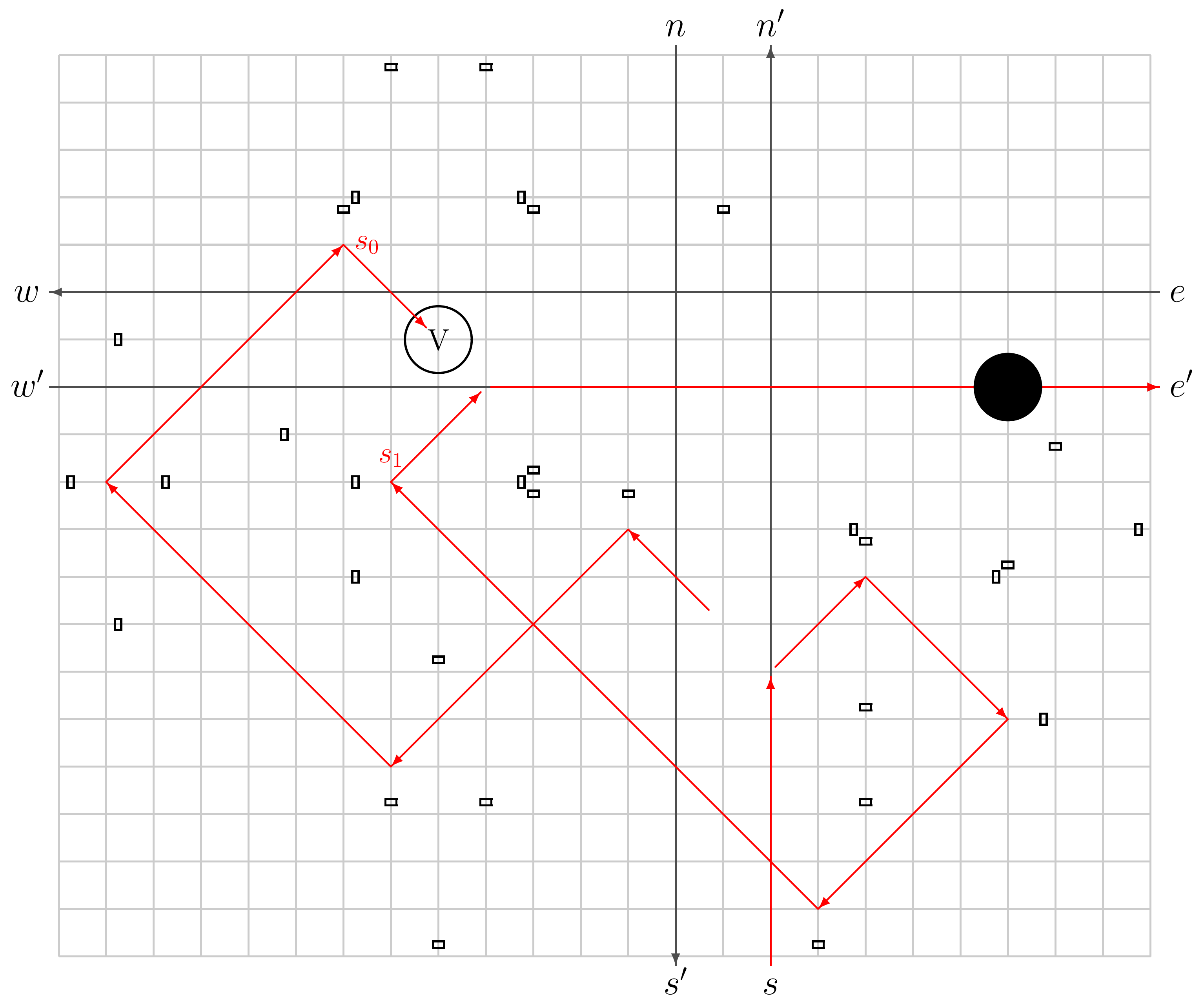
## Movements of Balls (State: $H$, Input: $s$)



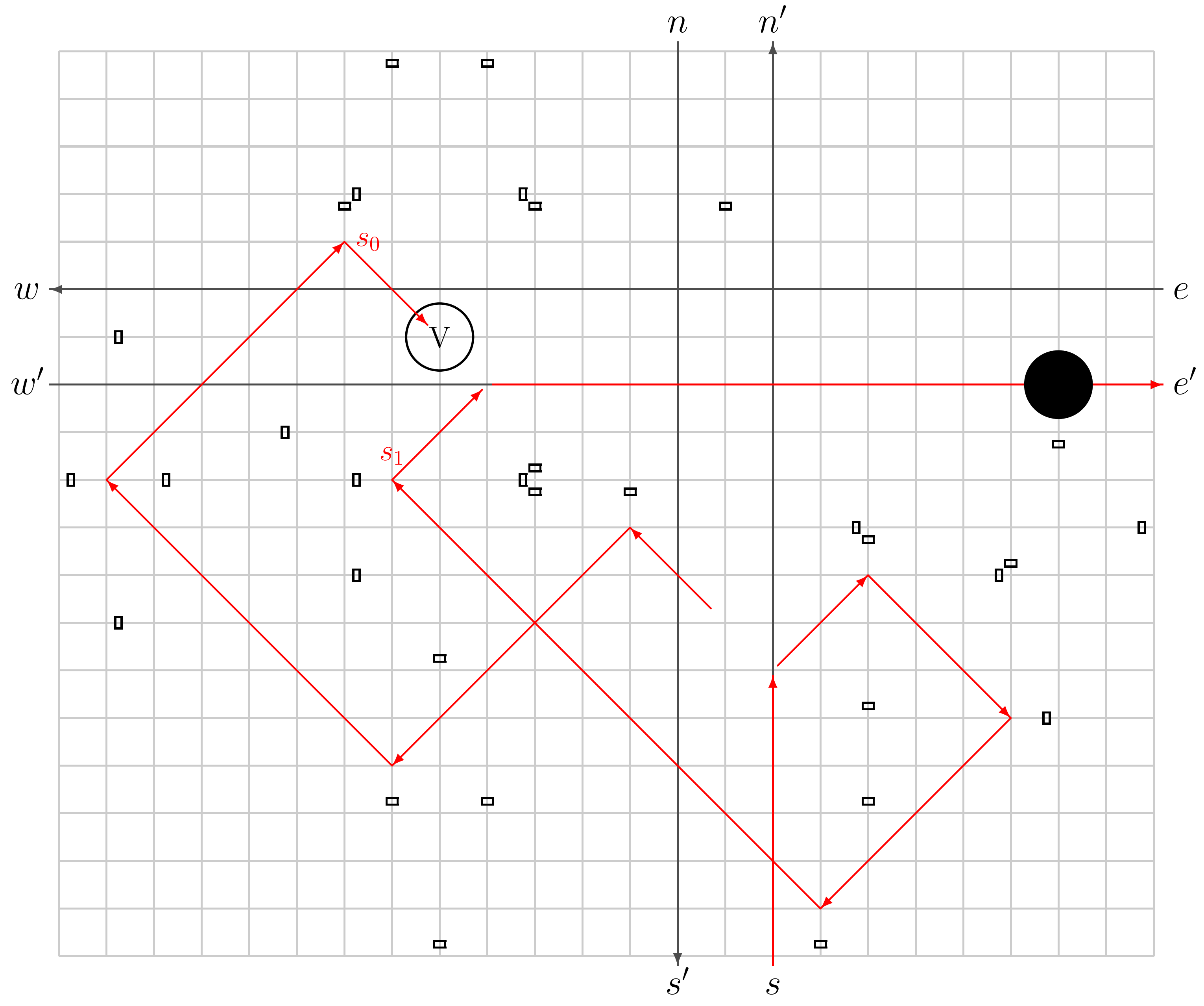
## Movements of Balls (State: $H$, Input: $s$)



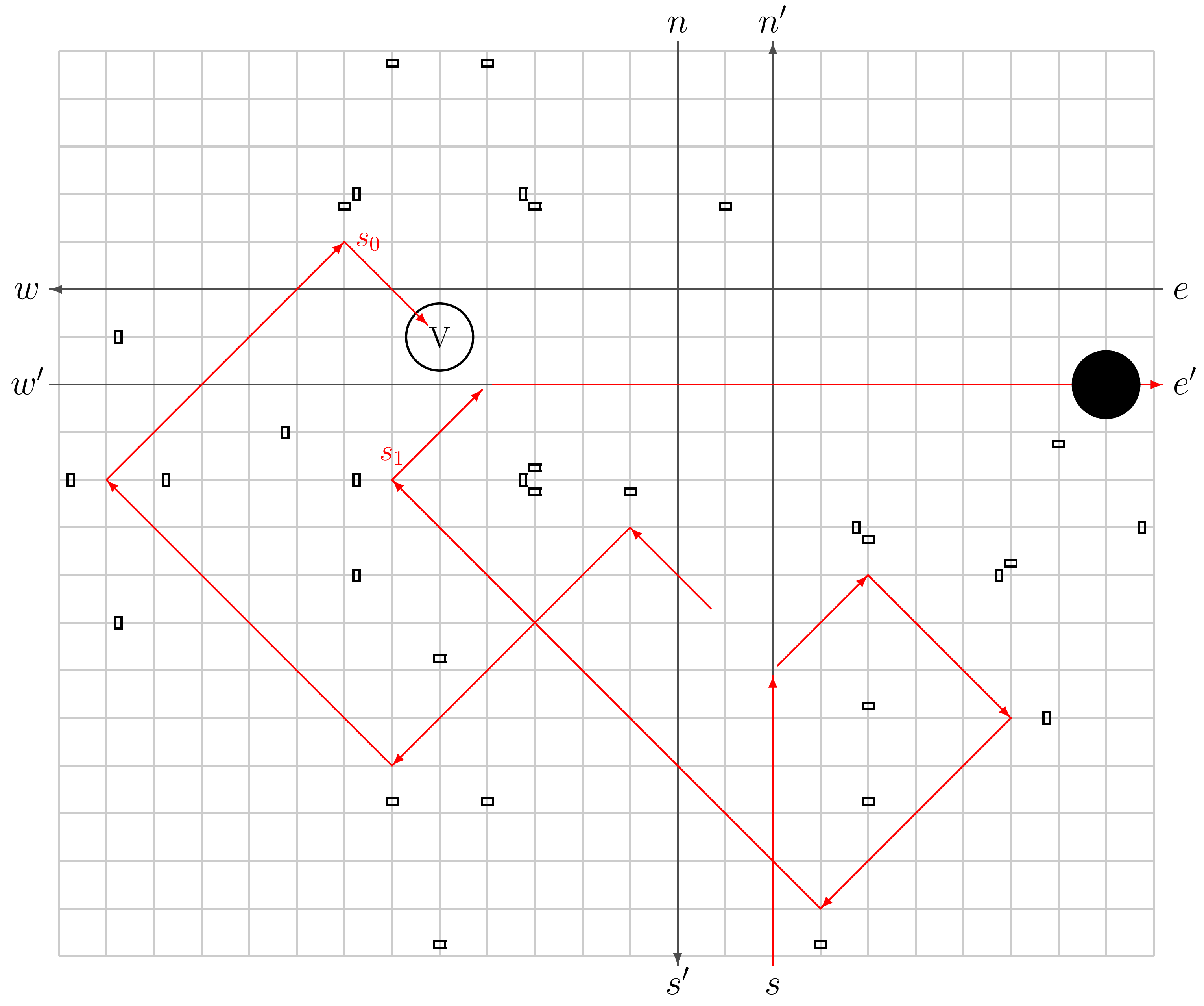
## Movements of Balls (State: $H$, Input: $s$)



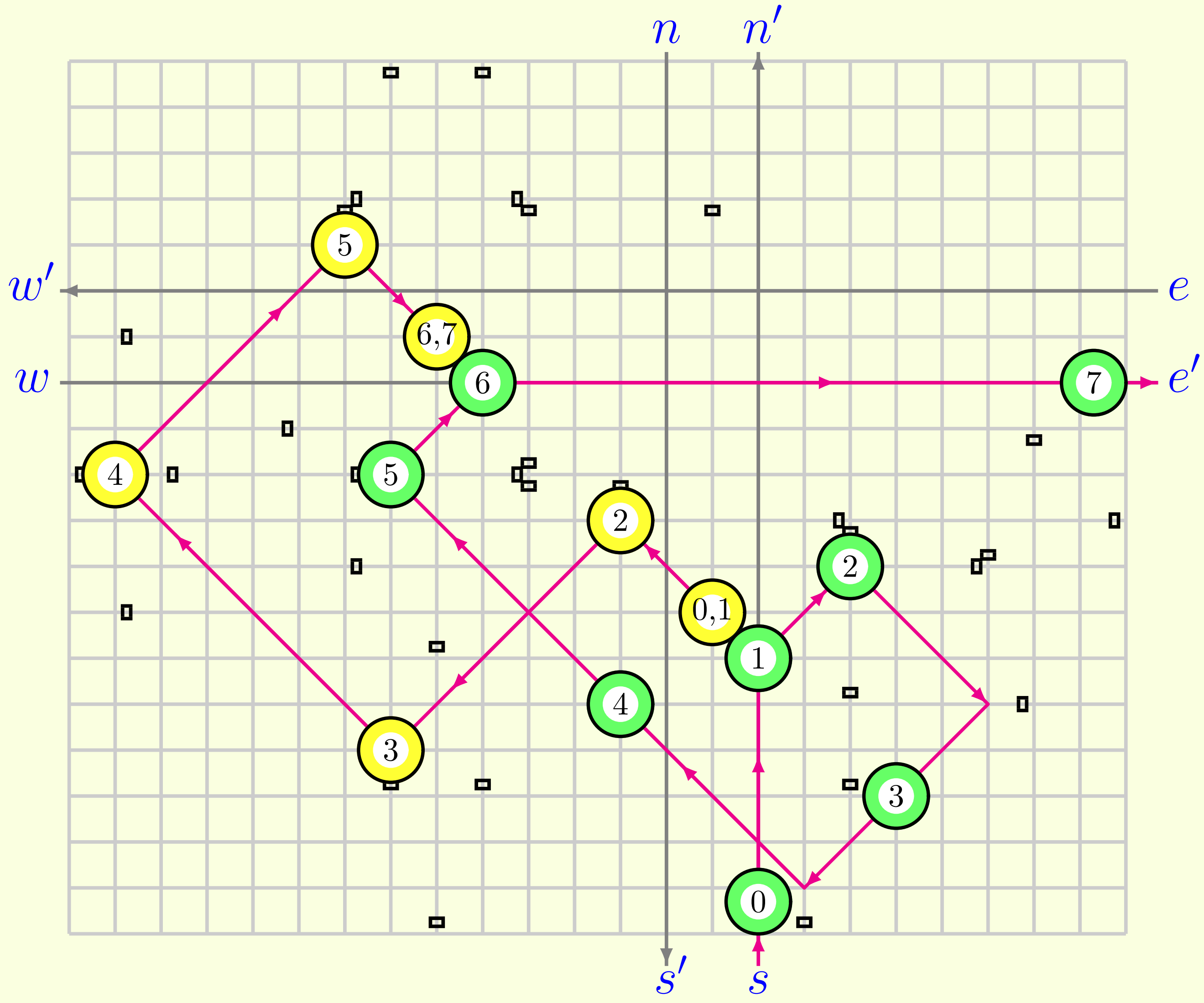
## Movements of Balls (State: $H$, Input: $s$)



# Movements of Balls (State: $H$, Input: $s$)



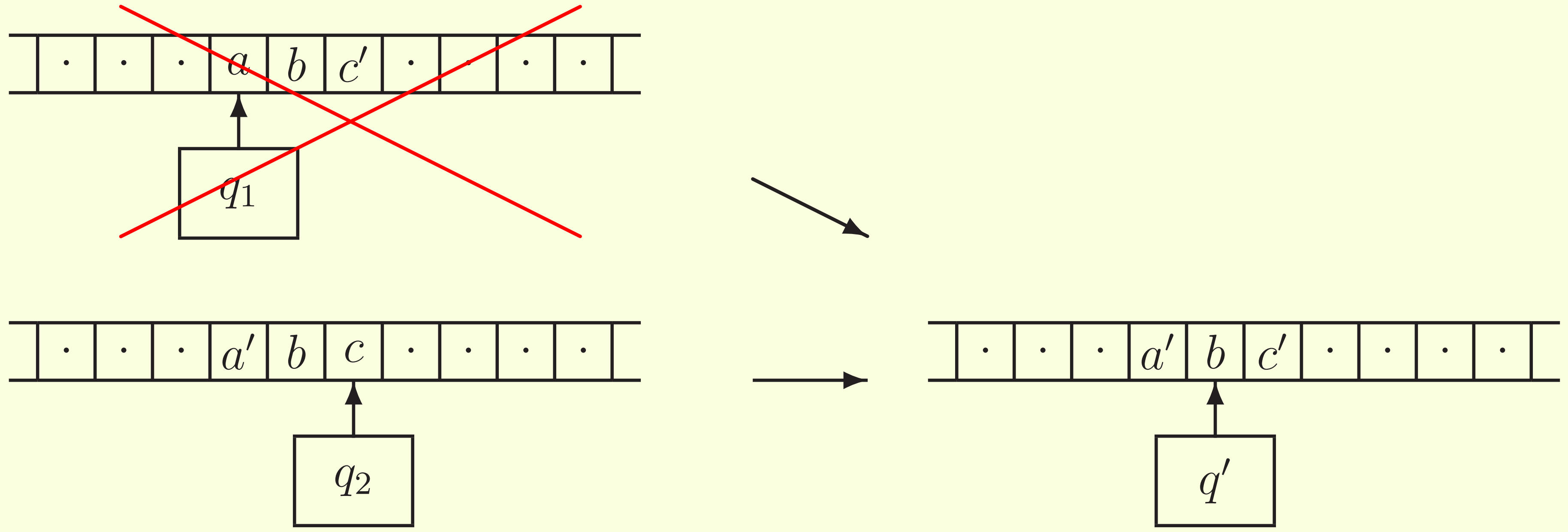
## BBM realization of an RE (orthogonal case)



## 4. Reversible models of computing

## Reversible Turing machine (RTM)

- A "backward deterministic" TM.



## Formal definition of a TM

$$T = (Q, S, q_0, F, s_0, \delta)$$

$Q$: a finite set of states.

$S$: a finite set of tape symbols.

$q_0$: an initial state $q_0 \in Q$.

$F$: a set of final states $F \subset Q$.

$s_0$: a blank symbol $s_0 \in S$.

$\delta$: a move relation given by a set of quintuples $[p, s, s', d, q] \in Q \times S \times S \times \{L, N, R\} \times Q$.
$p, q$: states, $s, s'$: symbols, $d$: shift direction.

## Reversibility of a TM

A TM $T = (Q, S, q_0, F, s_0, \delta)$ is called *reversible* iff the following condition holds for any pair of distinct quintuples $[p_1, s_1, s'_1, d_1, q_1]$ and $[p_2, s_2, s'_2, d_2, q_2]$.

If $q_1 = q_2$, then $s'_1 \neq s'_2 \wedge d_1 = d_2$

(If the next states are the same, then the written symbols must be different and the shift directions must be the same.)

## A simple example of an RTM: $T_{\text{parity}}$

$$T_{\text{parity}} = (Q, S, q_0, \{q_{\text{acc}}\}, 0, \delta)$$

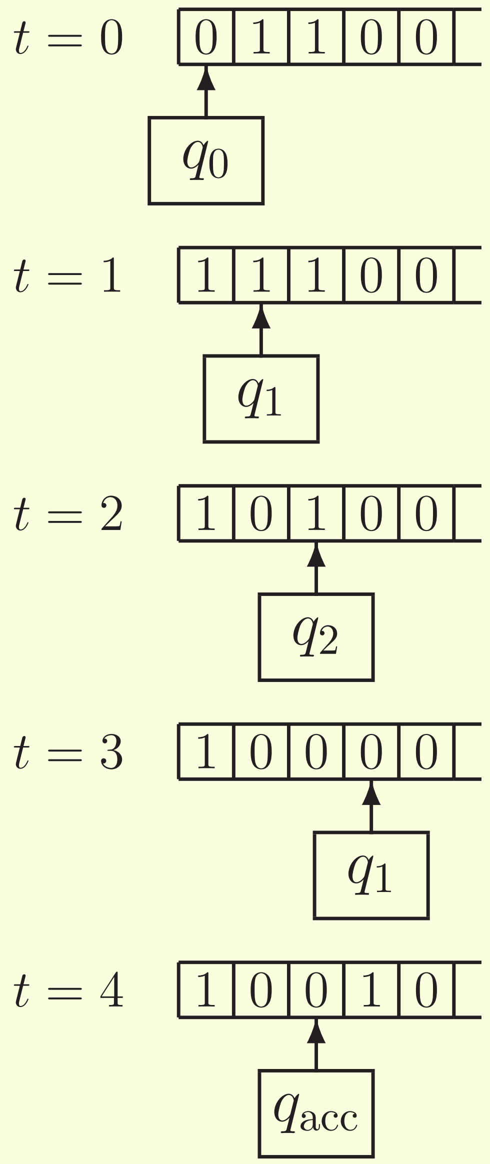
$Q = \{q_0, q_1, q_2, q_{\text{acc}}, q_{\text{rej}}\}$ : a set of states

$S = \{0, 1\}$ : a set of tape symbols

$\delta = \{[\ q_0, 0, 1, R, q_1\ ], [\ q_1, 0, 1, L, q_{\text{acc}}\ ],$
$[\ q_1, 1, 0, R, q_2\ ], [\ q_2, 0, 1, L, q_{\text{rej}}\ ],$
$[\ q_2, 1, 0, R, q_1\ ]\ \}$ : a move function

- A unary expression of an integer $n$ is given.
- If $n$ is even, $T_{\text{parity}}$ halts in the final state $q_{\text{acc}}$.
- If $n$ is odd, $T_{\text{parity}}$ halts in the state $q_{\text{rej}}$.
- All the symbols read by $T_{\text{parity}}$ are complemented.

## A computing process of $T_{\mathsf{parity}}$ for input "11"



$$
\begin{aligned}
\delta = \{ & [\ q_0, 0, 1, R, q_1\ ], \\
& [\ q_1, 0, 1, L, q_{\mathsf{acc}}\ ], \\
& [\ q_1, 1, 0, R, q_2\ ], \\
& [\ q_2, 0, 1, L, q_{\mathsf{rej}}\ ], \\
& [\ q_2, 1, 0, R, q_1\ ]\ \}
\end{aligned}
$$

## Any irreversible TM can be simulated by a time- or space-efficient reversible TM

**Theorem 7** [Bennett, 1973] For any irreversible TM, we can construct a garbage-less linear-time reversible TM that simulates the former.

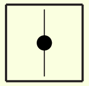
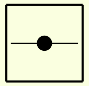
**Theorem 8** [Morita, 2012] For any irreversible TM, we can construct a garbage-less reversible TM that simulates the former, and uses exactly the same numbers of tape squares and tape symbols.

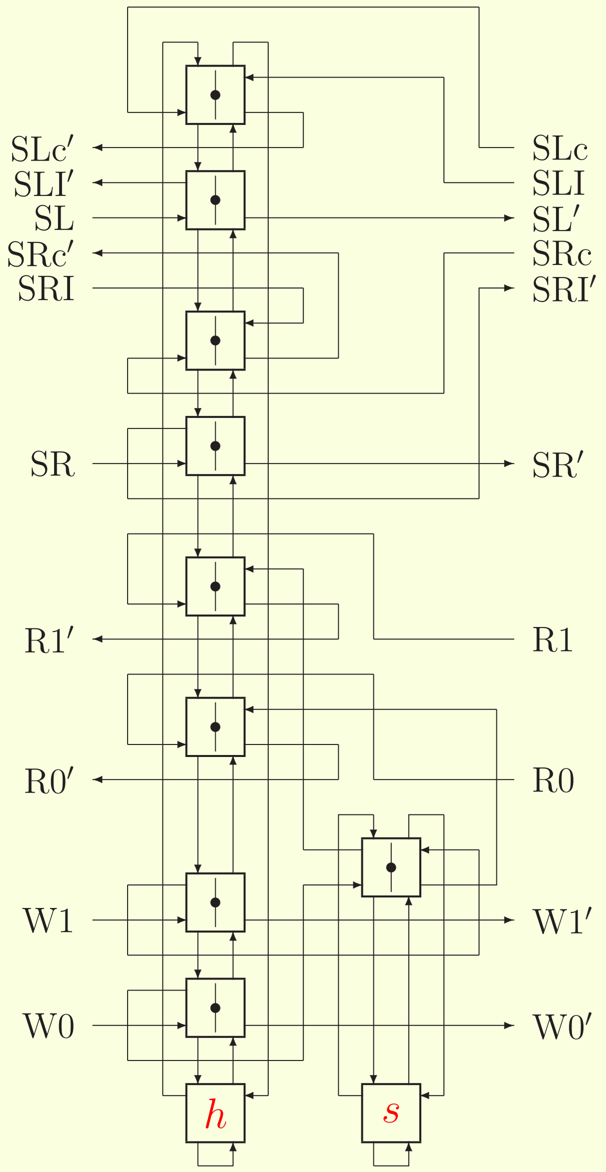
## 5. Realizing reversible Turing machines by reversible logic circuits

## Realizing RTMs by REs

- A memory cell (i.e., a tape square), and a finite-state control of an RTM are formalized as RSMs.

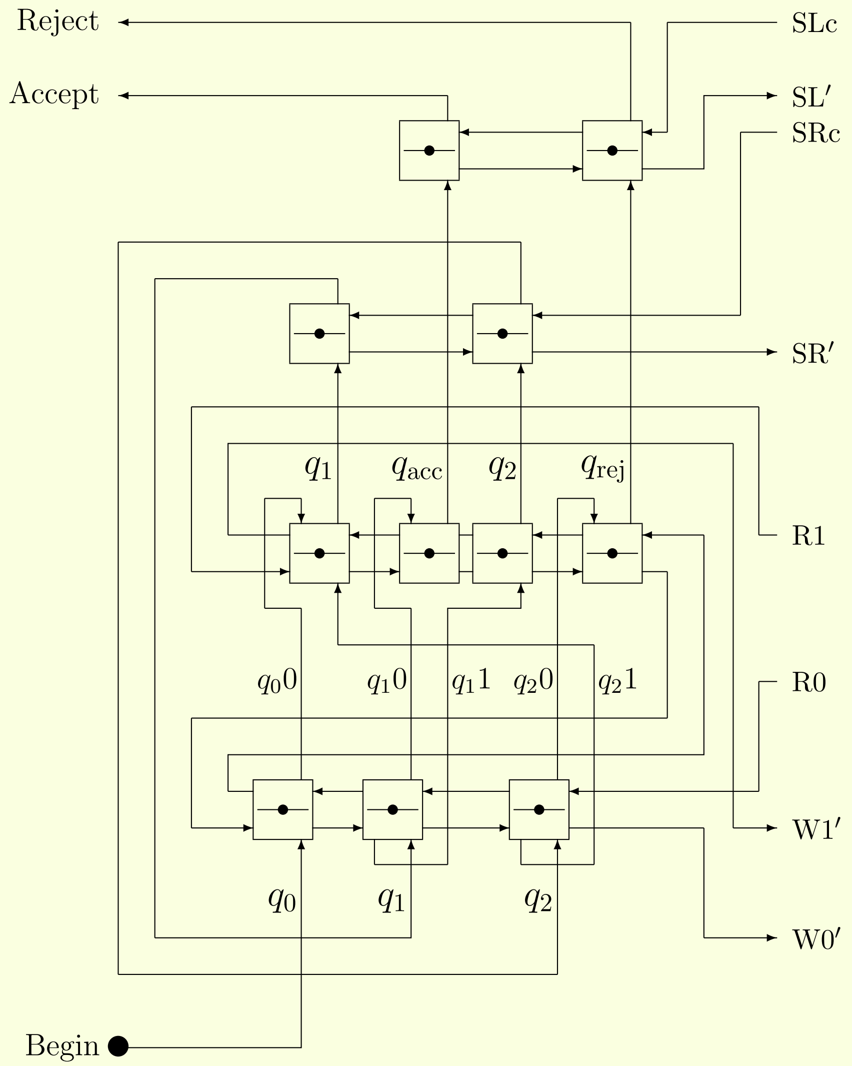
# Memory cell realized by REs

- It keeps a tape symbol $s \in \{0,1\}$.
- It also keeps the information $h$ whether the head is on this cell ($h = 1$) or not ($h = 0$).
- The REs at the positions $h$ and $s$ are set to  if the value is 0, and  if it is 1.

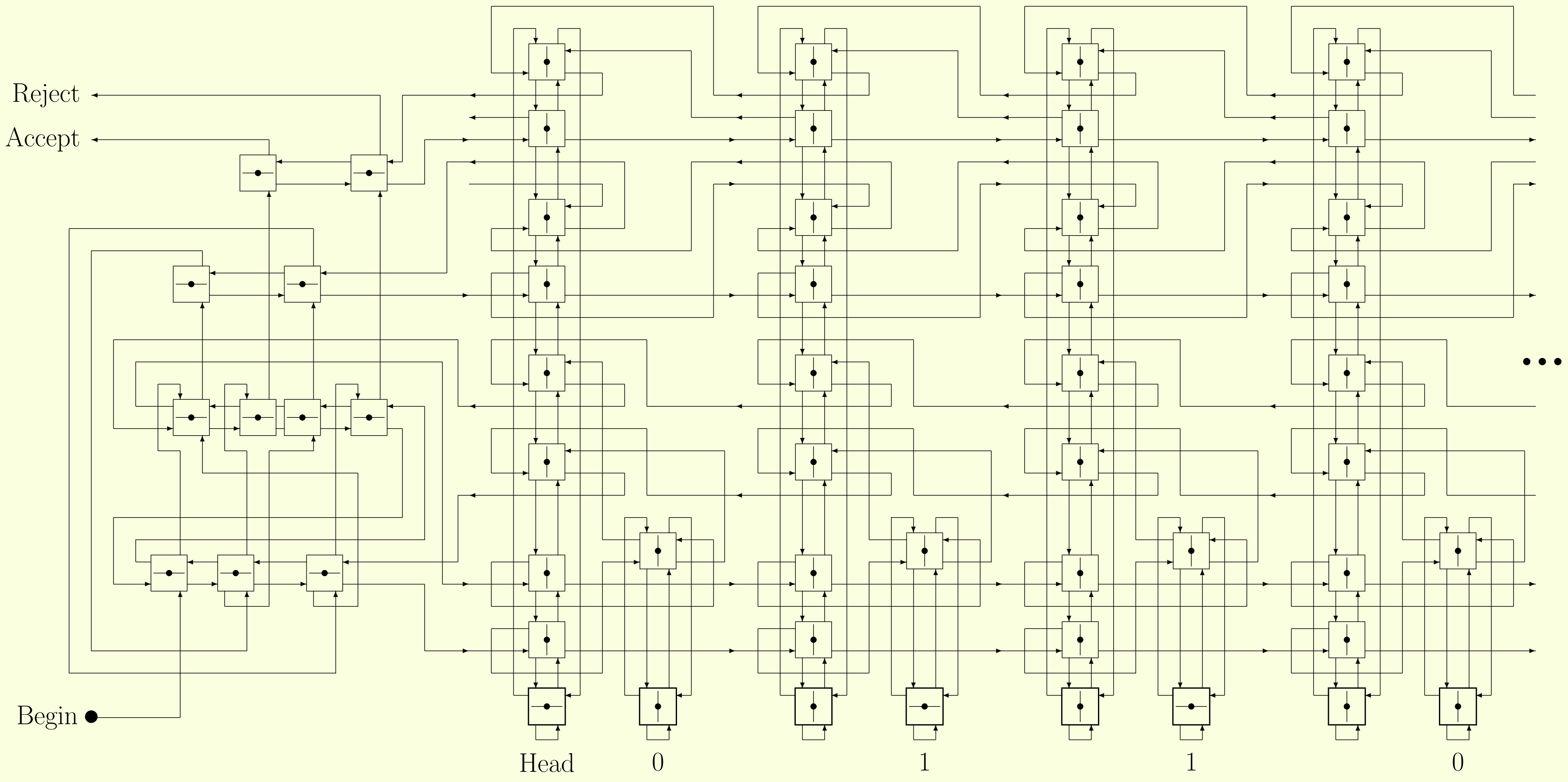


## Finite-state control of $T_{\mathrm{parity}}$ realized by REs

- The three REs in the 4th row correspond to $q_0, q_1$ and $q_2$. They read the symbol currently scanned.
- Then, the four REs of the 3rd row perform writing and state-change.
- Finally, the REs of the 1st or 2nd rows perform a head shift.



## The whole RE circuit that simulates $T_{\text{parity}}$



- Giving a particle to “Begin,” it starts to compute.
- It is also possible to further realize this circuit in BBM.

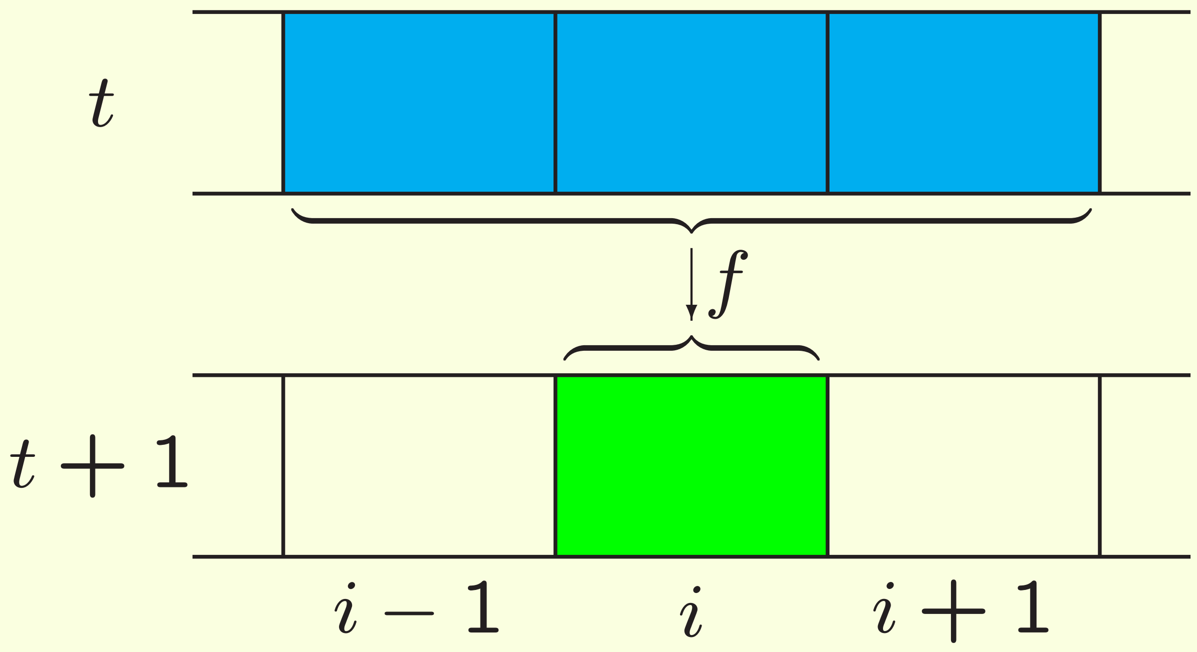
## 6. Reversible cellular automata (RCAs)

## Reversible cellular automaton (RCA)

- A cellular automaton (CA) is a discrete model for spatiotemporal phenomena.
- A reversible CA (RCA) is a one whose global function is one-to-one.
- RCAs have high capability of computing.
  - Computation-universality
  - Self-reproduction
  - etc.

## 1-dimensional cellular automaton (CA)

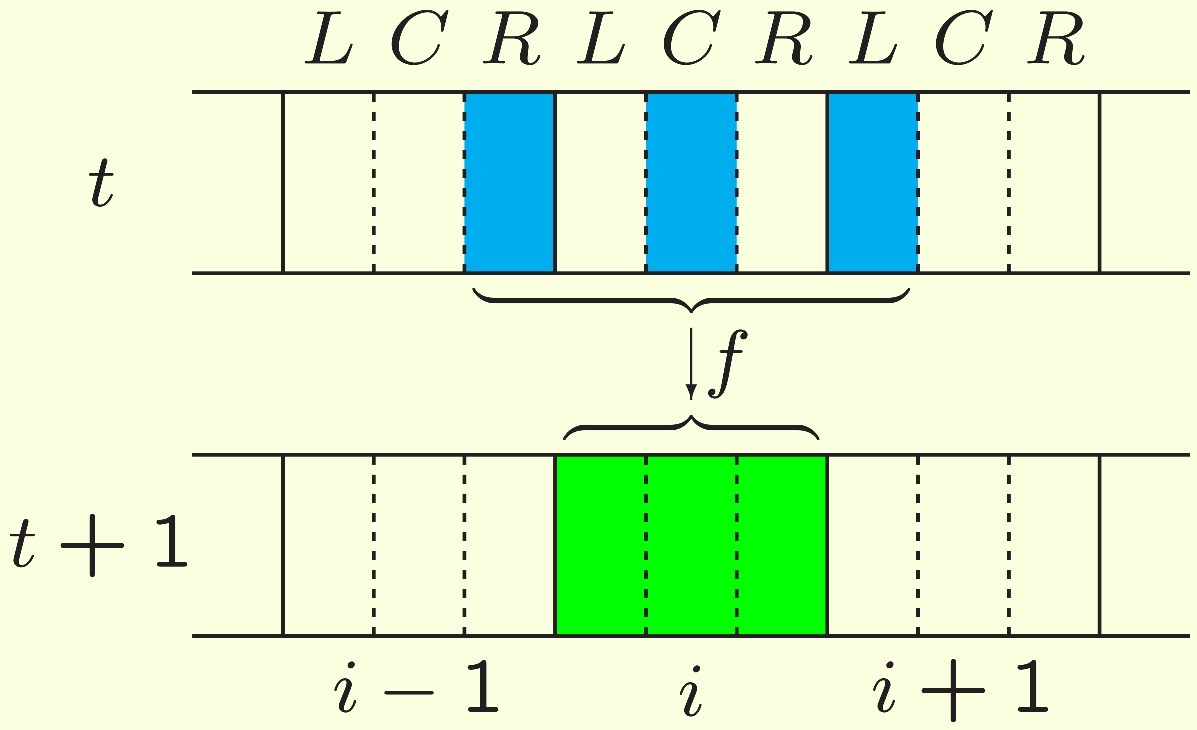
- A local function $f$ of a 1-D CA



- The local function is applied to all the cells.
- Difficult to design RCAs using this framework.

## 1-dimensional partitioned cellular automaton (PCA)

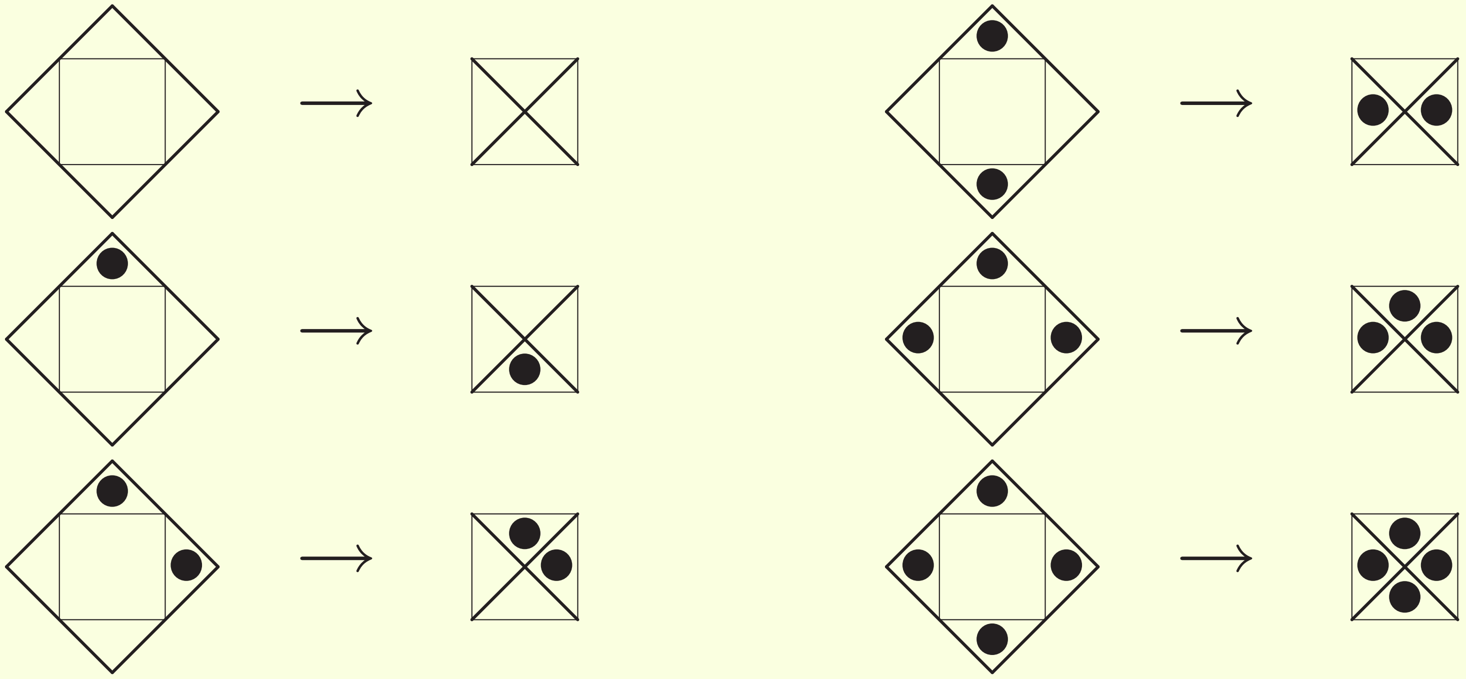
- A local function $f$ of a 1-D PCA.



- RCAs can be easily designed using PCAs.

## 2-dimensional partitioned cellular automaton

- Example:
  A local function of a 4-neighbor PCA $S_1$.



## Computationally universal reversible CAs — 1-D Case —

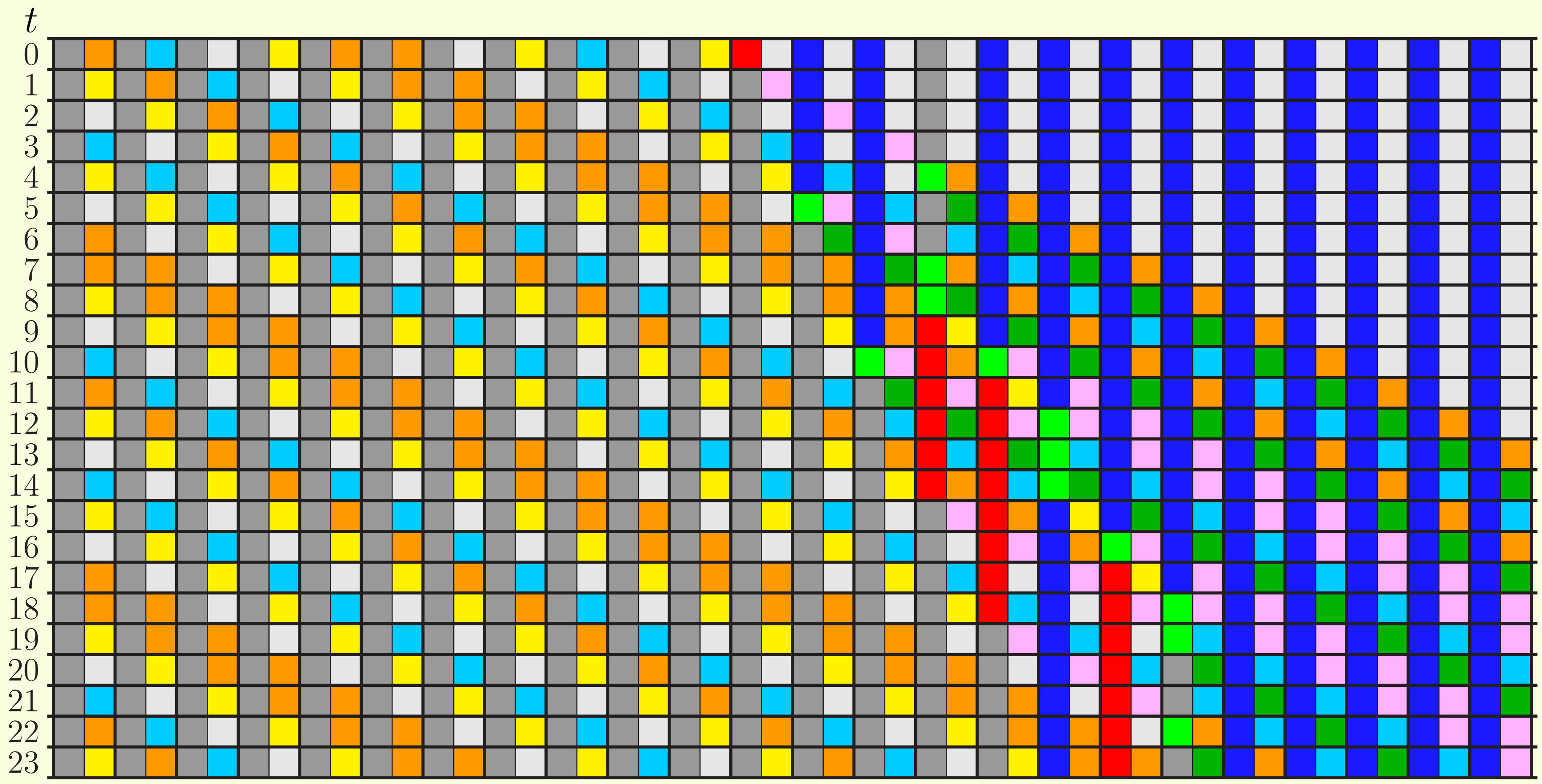
**Theorem 11**
(1) There is a 24-state universal reversible PCA with infinite configurations. [Morita, 2008]
(2) There is a 96-state universal reversible PCA with finite configurations. [Morita, 2007]

# A 24-state 1-D reversible PCA that simulates any cyclic tag system

| t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | N | − | Y | − | Y | − | − | − | Y | − | Y | − | Y | − | Y | − | Y | − |
| 1 | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | / | Y | − | Y | − | − | − | Y | − | Y | − | Y | − | Y | − | Y | − |
| 2 | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | Y | / | Y | − | − | − | Y | − | Y | − | Y | − | Y | − | Y | − |
| 3 | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | Y | − | Y | / | − | − | Y | − | Y | − | Y | − | Y | − | Y | − |
| 4 | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | Y | y | Y | − | + | n | Y | − | Y | − | Y | − | Y | − | Y | − |
| 5 | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | + | / | Y | y | − | + | Y | n | Y | − | Y | − | Y | − | Y | − |
| 6 | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | + | Y | / | − | y | Y | + | Y | n | Y | − | Y | − | Y | − |
| 7 | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | Y | + | + | n | Y | y | Y | + | Y | n | Y | − | Y | − |
| 8 | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | Y | n | + | + | Y | n | Y | y | Y | + | Y | n | Y | − |
| 9 | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | Y | n | N | * | Y | + | Y | n | Y | y | Y | + | Y | n |
| 10 | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | + | / | N | n | + | / | Y | + | Y | n | Y | y | Y | + |
| 11 | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | + | N | / | N | * | Y | / | Y | + | Y | n | Y | y |
| 12 | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | N | + | N | / | + | / | Y | / | Y | + | Y | n |
| 13 | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | N | y | N | + | + | y | Y | / | Y | / | Y | + |
| 14 | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | N | n | N | y | + | + | Y | y | Y | / | Y | / |
| 15 | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | / | N | n | Y | * | Y | + | Y | y | Y | / |
| 16 | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | N | / | Y | n | + | / | Y | + | Y | y |
| 17 | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | N | − | Y | / | N | * | Y | / | Y | + |
| 18 | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | N | y | Y | − | N | / | + | / | Y | / |
| 19 | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | / | Y | y | N | − | + | y | Y | / |
| 20 | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | Y | / | N | y | − | + | Y | y |
| 21 | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | − | n | Y | − | N | / | − | y | Y | + |
| 22 | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | − | n | Y | n | N | − | + | n | Y | y |
| 23 | − | * | − | n | − | y | − | − | − | * | − | n | − | n | − | − | − | * | − | y | − | − | − | * | − | n | − | y | − | − | − | * | Y | n | N | n | − | + | Y | n |

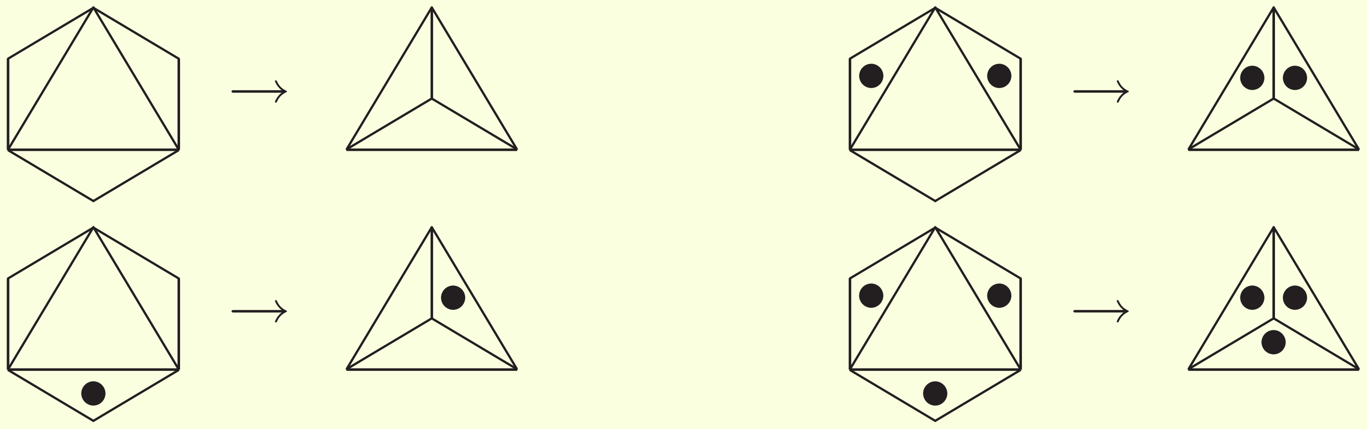
## A 24-state 1-D reversible PCA that simulates any cyclic tag system

(Color version)

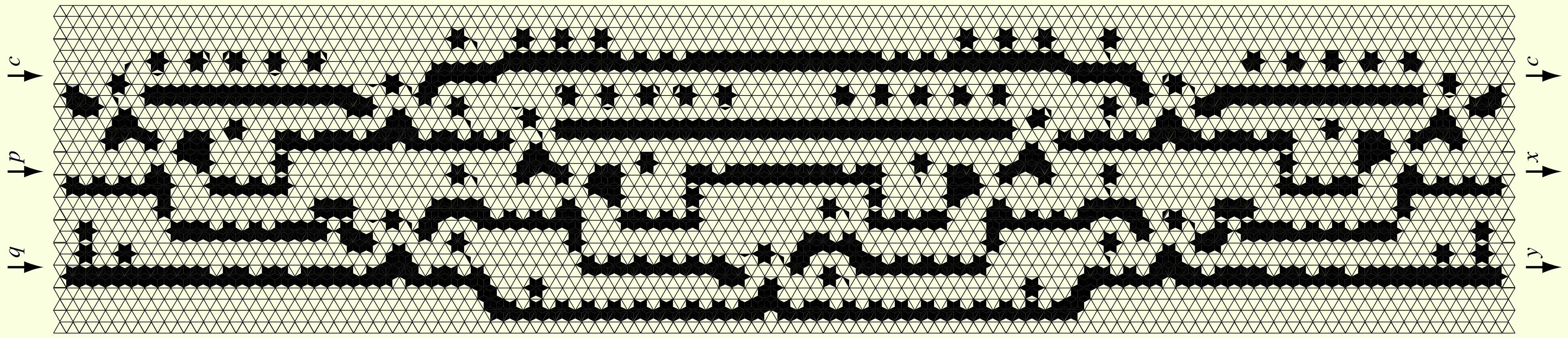


## Computationally universal reversible CAs — 2-D case (1) —

**Theorem 12** [Imai, Morita, 2000] The reversible 8-state PCA having the following local function with infinite configurations is universal.



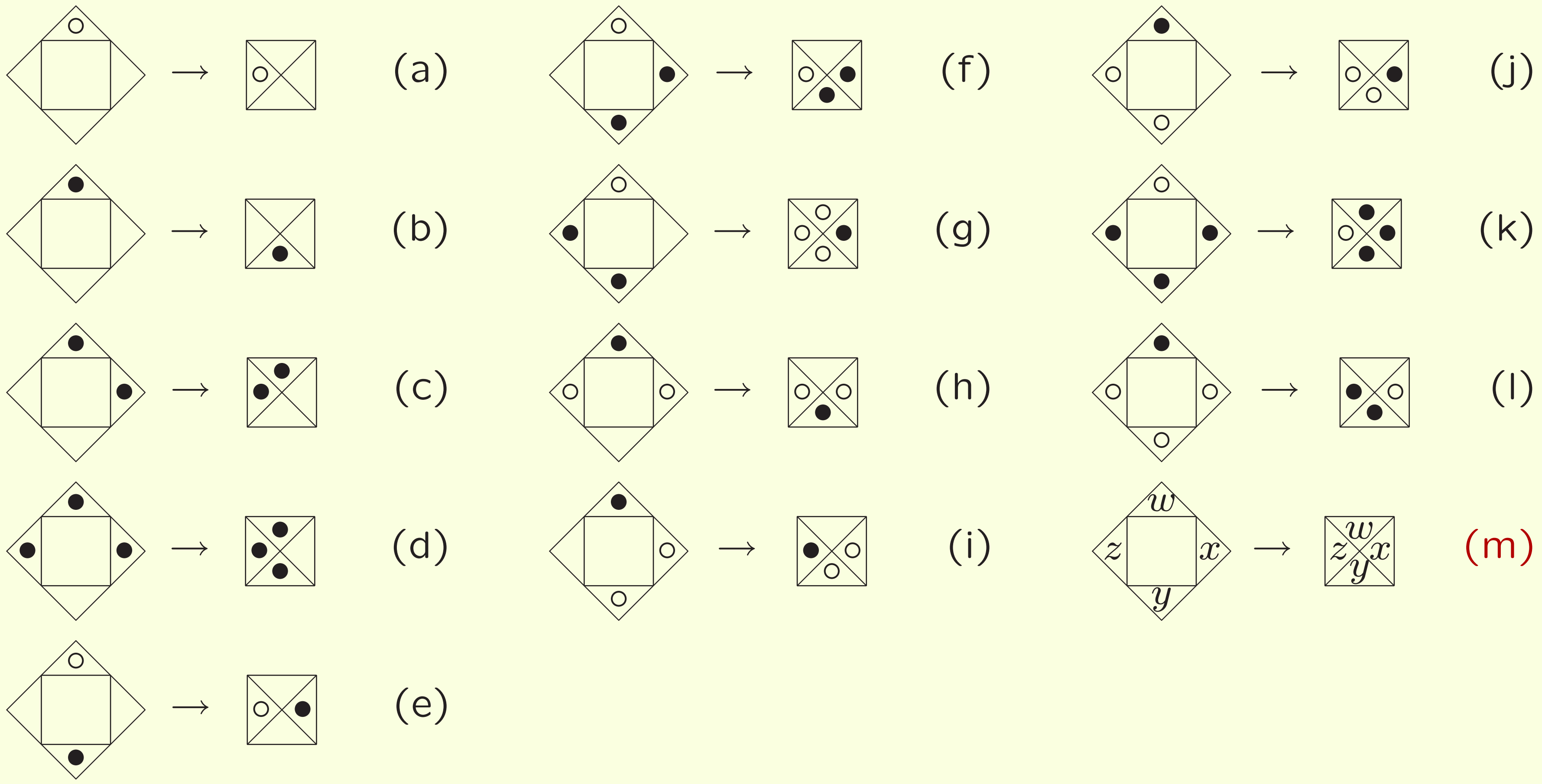
## A Fredkin gate in the 8-state reversible PCA



## Computationally universal reversible CAs — 2-D case (2) —

**Theorem 13** [Morita et al., 2002] There is a reversible 81-state PCA $P_3$ with finite configurations that can simulate any reversible counter machine. Hence, it is universal.

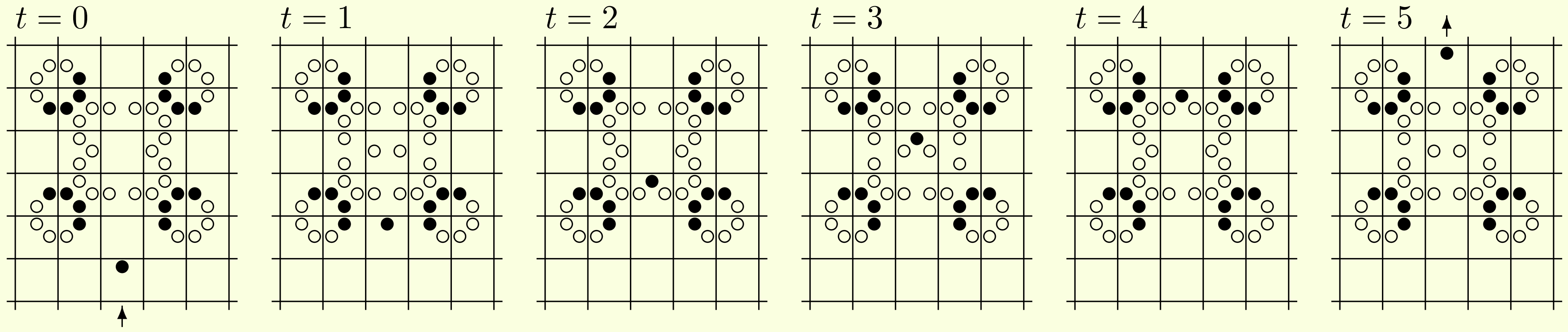
# The local function of the 81-state RPCA $P_3$



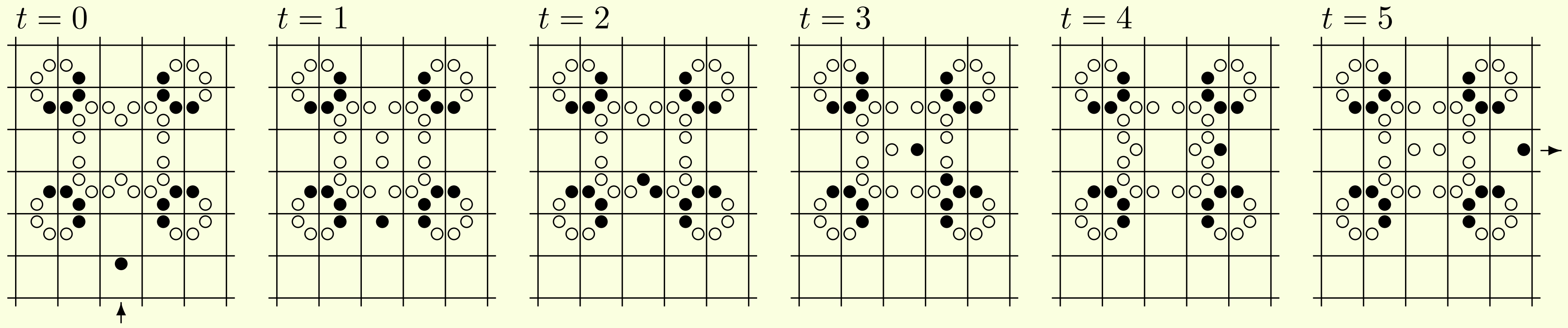
The rule scheme (m) represents 33 rules not specified by (a)–(l) $(w, x, y, z \in \{$ blank, ○, ● $\} = \{0, 1, 2\})$.

## A rotary element implemented in $P_3$

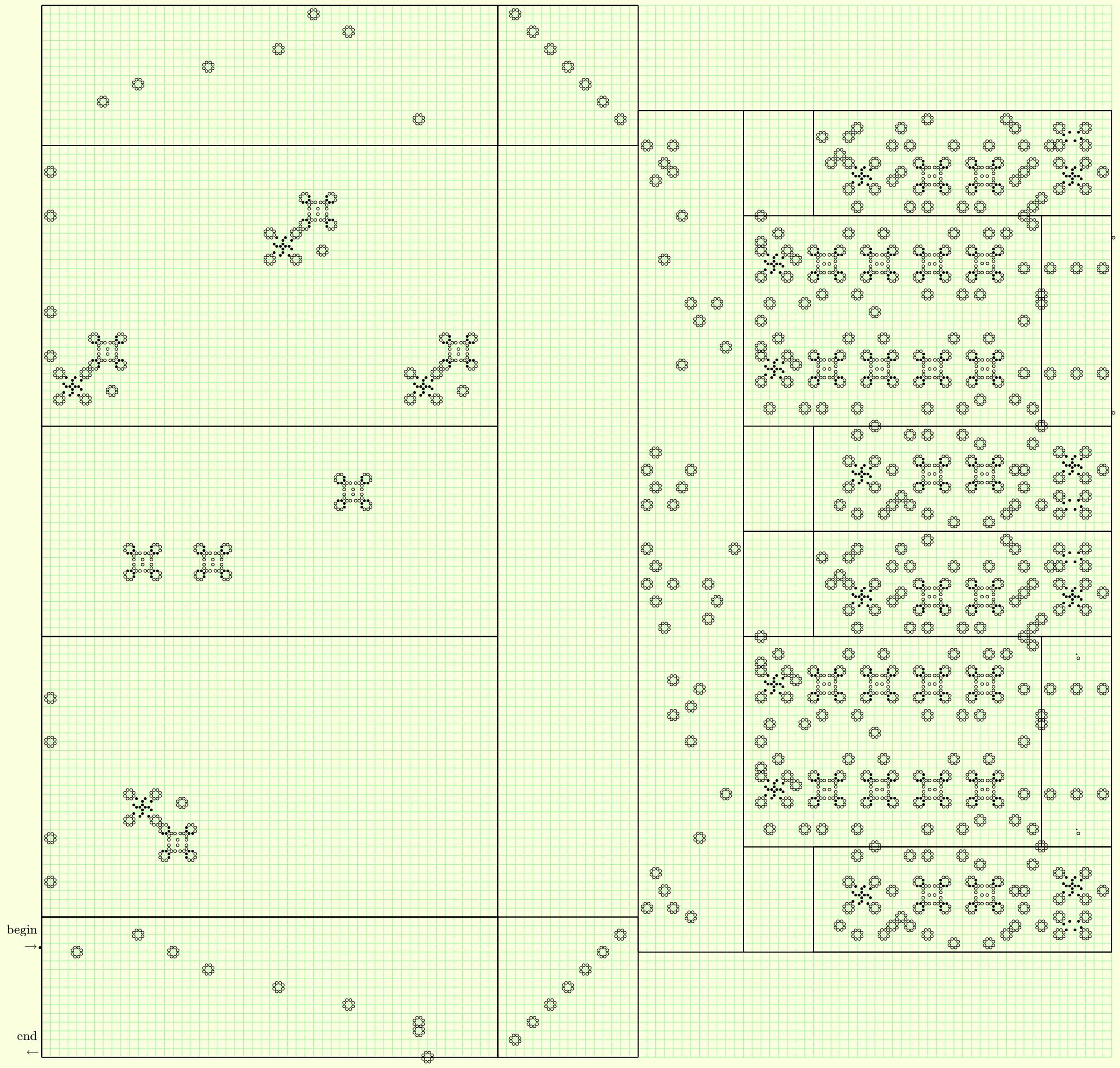
Parallel case:



Orthogonal case:



## Reversible counter machine (RCM) in $P_3$



## Concluding remarks

- We have seen even very simple reversible logic elements have universality.
- Reversible computers can be constructed in a very different way from that of conventional computers.
- Studies on reversible computing will give new insights for future computing.

## Some references

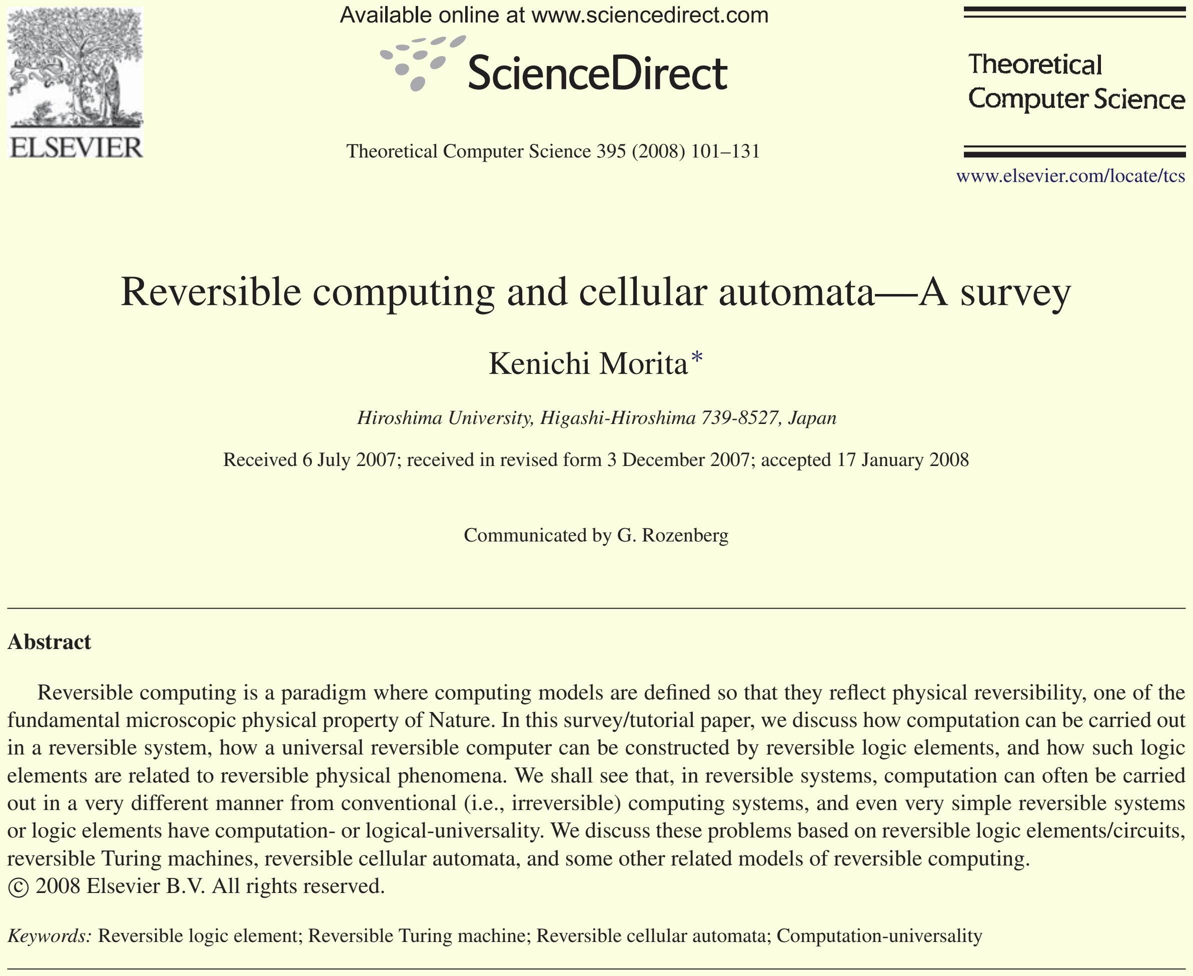
[20] Morita, K., Reversible computing and cellular automata — A survey, *Theoretical Computer Science*, **395**, 101–131 (2008). http://ir.lib.hiroshima-u.ac.jp/00025576 (2008).

[21] Morita, K., Constructing a reversible Turing machine by a rotary element, a reversible logic element with memory, http://ir.lib.hiroshima-u.ac.jp/00029224 (2010).

[25] Morita, K., Reversible computing — Can we construct a computer out of billiard balls? (in Japanese), *J. Information Proc. Soc. of Japan*, **53** pp.496–502 (2012).

[26] Morita, K., Reversible computing (in Japanese), Kindai Kagaku-sha Co., Ltd., Tokyo (2012).

[20] Morita, K., Reversible computing and cellular automata — A survey, *Theoretical Computer Science*, **395**, 101–131 (2008). http://ir.lib.hiroshima-u.ac.jp/00025576 (2008).



ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 395 (2008) 101–131

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Reversible computing and cellular automata—A survey

Kenichi Morita*

*Hiroshima University, Higashi-Hiroshima 739-8527, Japan*

Received 6 July 2007; received in revised form 3 December 2007; accepted 17 January 2008

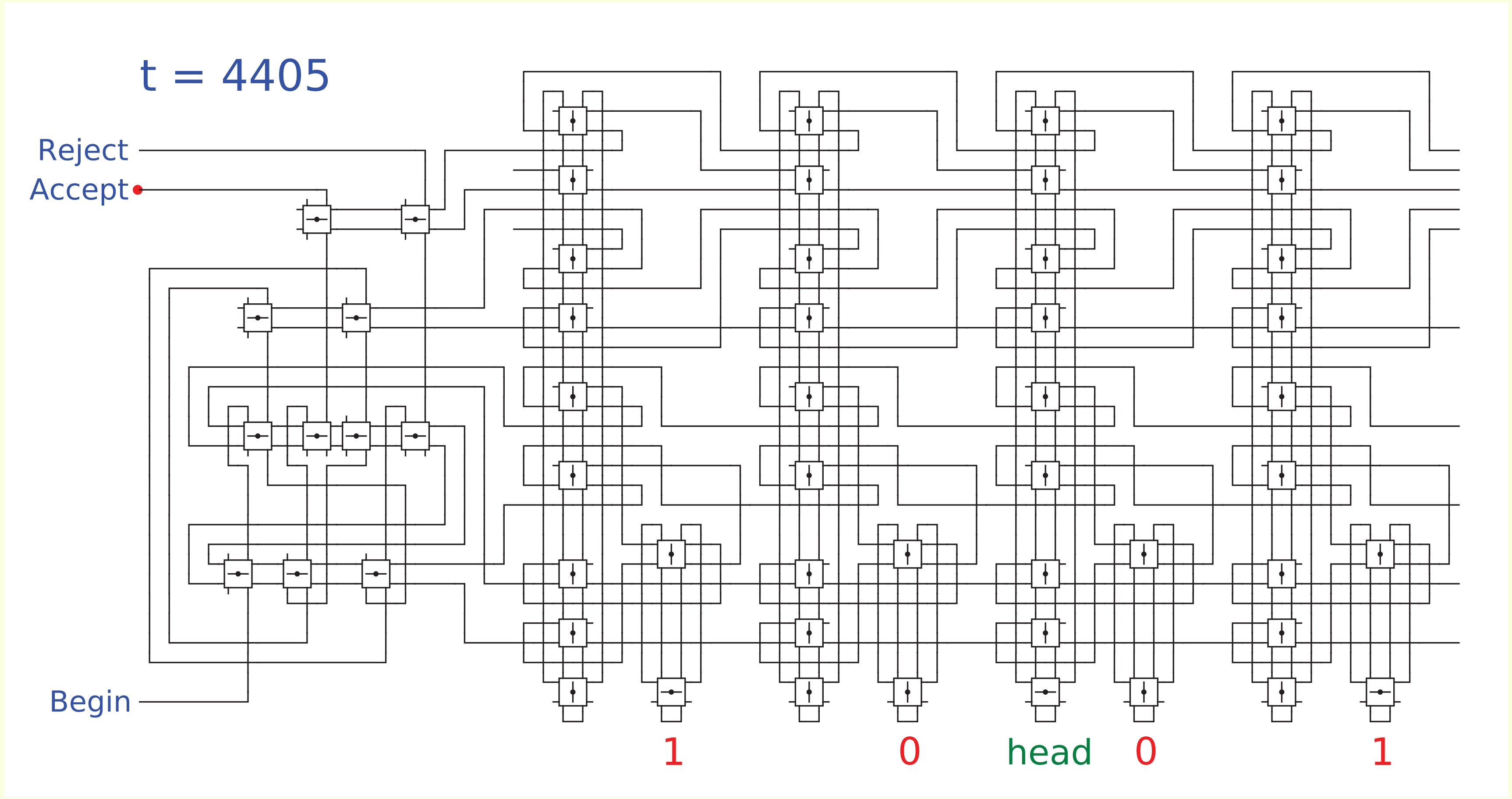
Communicated by G. Rozenberg

**Abstract**

Reversible computing is a paradigm where computing models are defined so that they reflect physical reversibility, one of the fundamental microscopic physical property of Nature. In this survey/tutorial paper, we discuss how computation can be carried out in a reversible system, how a universal reversible computer can be constructed by reversible logic elements, and how such logic elements are related to reversible physical phenomena. We shall see that, in reversible systems, computation can often be carried out in a very different manner from conventional (i.e., irreversible) computing systems, and even very simple reversible systems or logic elements have computation- or logical-universality. We discuss these problems based on reversible logic elements/circuits, reversible Turing machines, reversible cellular automata, and some other related models of reversible computing.
© 2008 Elsevier B.V. All rights reserved.

*Keywords:* Reversible logic element; Reversible Turing machine; Reversible cellular automata; Computation-universality

[21] Morita, K., Constructing a reversible Turing machine by a rotary element, a reversible logic element with memory, http://ir.lib.hiroshima-u.ac.jp/00029224 (2010).



An example of a whole computing process is shown.

[25] Morita, K., Reversible computing — Can we construct a computer out of billiard balls? (in Japanese), *J. Information Proc. Soc. of Japan*, Vol.53, No.5, pp.496–502 (2012).

1970年3月10日第三種郵便物認可　2012年4月10日印刷　2012年4月15日発行（毎月1回15日発行）
ISSN 0447-8053

情報処理

2012 5

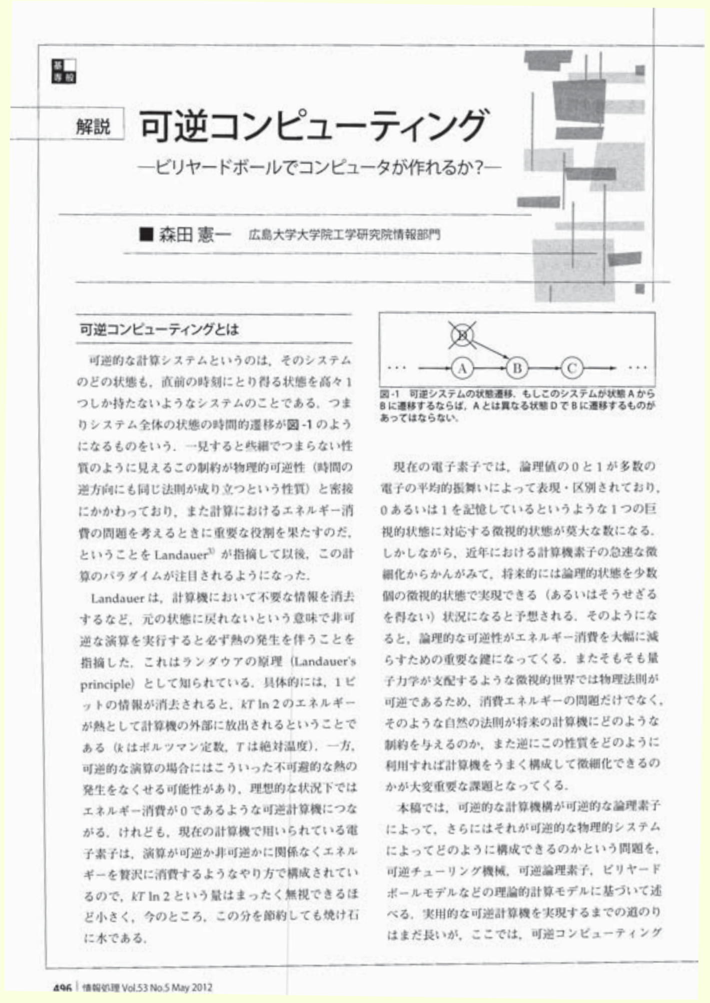
Vol.53 No.5 通巻566号

巻頭コラム　私たちがいかに恵まれているか
C. W. ニコル

特集　CGMの現在と未来
初音ミク, ニコニコ動画, ピアプロの切り拓いた世界

解説　可逆コンピューティング
　―ビリヤードボールでコンピュータが作れるか？―
圧縮したまま検索―Succinct Data Structure―
連載　細川茂樹 アプリ開発プロジェクト：
　第4回　目指せ情報処理学会発の業界標準アプリ
教育コーナー　ぺた語義：コラム：情報教育と情報技術教育
ぺた語義：産業技術大学院大学・情報アーキテクチャ専攻のPBL―社会人大学院におけるPBL学習―
ぺた語義：オープンソースを活用した専門職大学院大学におけるオブジェクト指向技術教育の紹介
ぺた語義：特別コラム：お大師様を訪ねて（4）君のひとみは10000ボルト
コラム　"I" 見聞録：ACM CoNEXT 2011
シニアコラム　IT好き放題：コンピュータの歴史的遺産の保存に取り組んで

一般社団法人 情報処理学会
Information Processing Society of Japan



解説　可逆コンピューティング
―ビリヤードボールでコンピュータが作れるか？―

■森田 憲一　広島大学大学院工学研究院情報部門

可逆コンピューティングとは

可逆的な計算システムというのは，そのシステムのどの状態も，直前の時刻にとり得る状態を高々1つしか持たないようなシステムのことである．つまりシステム全体の状態の時間的遷移が図-1のようになるものをいう．一見すると些細でつまらない性質のように見えるこの制約が物理的可逆性（時間の逆方向にも同じ法則が成り立つという性質）と密接にかかわっており，また計算におけるエネルギー消費の問題を考えるときに重要な役割を果たすのだ，ということをLandauer[3]が指摘して以後，この計算のパラダイムが注目されるようになった．

Landauerは，計算機において不要な情報を消去するなど，元の状態に戻れないという意味で非可逆な演算を実行すると必ず熱の発生を伴うことを指摘した．これはランダウアの原理（Landauer's principle）として知られている．具体的には，1ビットの情報が消去されると，$kT \ln 2$のエネルギーが熱として計算機の外部に放出されるということである（$k$はボルツマン定数，$T$は絶対温度）．一方，可逆的な演算の場合にはこういった不可避的な熱の発生をなくせる可能性があり，理想的な状況下ではエネルギー消費が0であるような可逆計算機につながる．けれども，現在の計算機で用いられている電子素子は，演算が可逆か非可逆かに関係なくエネルギーを贅沢に消費するようなやり方で構成されているので，$kT \ln 2$という量はまったく無視できるほど小さく，今のところ，この分を節約しても焼け石に水である．

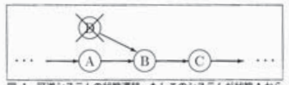


図-1　可逆システムの状態遷移．もしこのシステムが状態AからBに遷移するならば，Aとは異なる状態DでBに遷移するものがあってはならない．

現在の電子素子では，論理値の0と1が多数の電子の平均的振舞いによって表現・区別されており，0あるいは1を記憶しているというような1つの巨視的状態に対応する微視的状態が莫大な数になる．しかしながら，近年における計算機素子の急速な微細化からかんがみて，将来的には論理的状態を少数個の微視的状態で実現できる（あるいはそうせざるを得ない）状況になると予想される．そのようになると，論理的な可逆性がエネルギー消費を大幅に減らすための重要な鍵になってくる．またそもそも量子力学が支配するような微視的世界では物理法則が可逆であるため，消費エネルギーの問題だけでなく，そのような自然の法則が将来の計算機にどのような制約を与えるのか，また逆にこの性質をどのように利用すれば計算機をうまく構成して微細化できるのかが大変重要な課題となってくる．

本稿では，可逆的な計算機構が可逆的な論理素子によって，さらにはそれが可逆的な物理的システムによってどのように構成できるのかという問題を，可逆チューリング機械，可逆論理素子，ビリヤードボールモデルなどの理論的計算モデルに基づいて述べる．実用的な可逆計算機を実現するまでの道のりはまだ長いが，ここでは，可逆コンピューティング

496 | 情報処理 Vol.53 No.5 May 2012

[26] Morita, K., Reversible computing (in Japanese), Kindai Kagaku-sha Co., Ltd., Tokyo (2012).

Natural Computing
NC
ナチュラルコンピューティング・シリーズ
萩谷昌己・横森　貴　編
第5巻
可逆計算
森田憲一　著
近代科学社