Reversible Logic Elements with Memory

Kenichi Morita
Hiroshima University

Design of Reversible and Quantum Circuits
Dagstuhl Seminar 11502
Schloss Dagstuhl, Germany, 11-14 December 2011

Contents

1. What are reversible logic elements with memory (RLEMs)?
2. How can we construct reversible machines by RLEMs?
3. Can RLEMs be implemented in a reversible physical system efficiently?
4. Which RLEM is universal, and which is not?

1. What are reversible logic elements with memory (RLEMs)?

1. What are reversible logic elements with memory (RLEMs)?

Answer (at first, it is stated very shortly):

- They are very interesting logic elements as well as reversible logic gates (at least I think so).

Design methods of logic circuits/systems

- Traditional method:
  Elements for logical operation (logic gates), and those for memory (flip-flops) are separated.
  - Also in the case of reversible logic circuits, this method has been mainly employed.
- Method of using logic elements with memory:
  - In asynchronous logic circuits
    E.g., [Keller, 1974], [Büning, Priese, 1980], etc.
  - In cellular automata (CAs)
    Each cell is a logic element with memory
  - In reversible logic circuits
    E.g., [Morita, 2001]

Two kinds of reversible logic elements

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

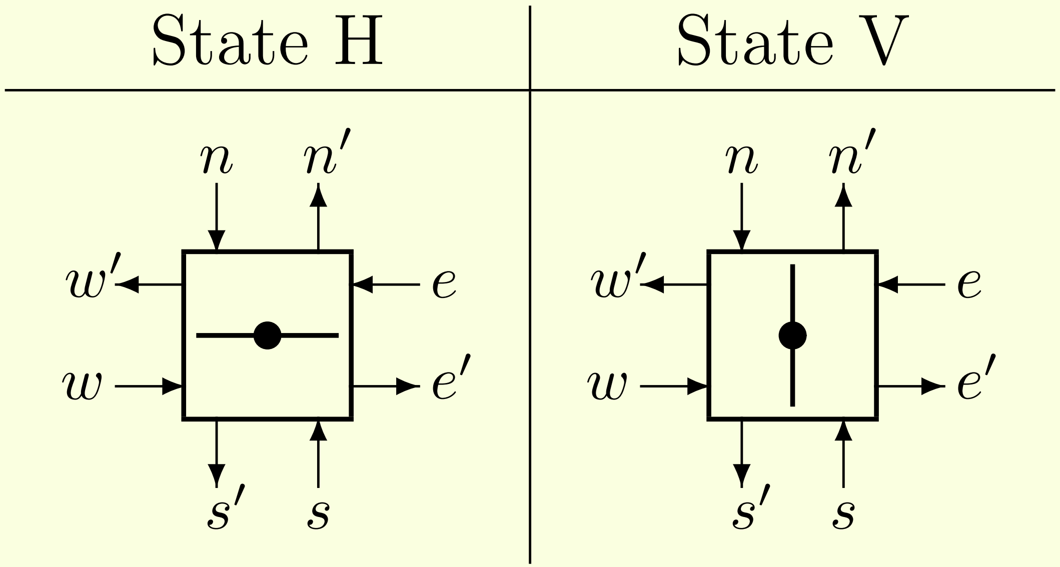
- Toffoli gate [Toffoli, 1980]
- Fredkin gate [Fredkin and Toffoli, 1982]
- etc.

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

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

Rotary element (RE) [Morita, 2001]

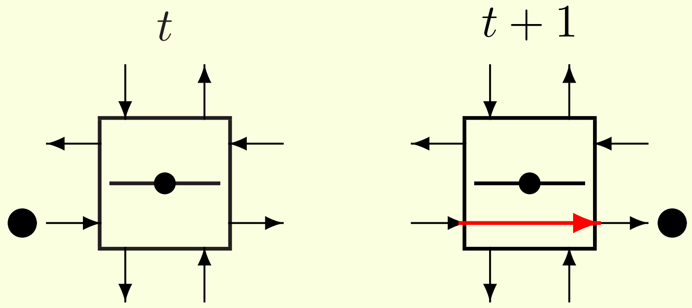
- A typical reversible logic element with memory.
- A 2-state 4-input-line 4-output-line element.
- Conceptually, it has a rotatable bar.



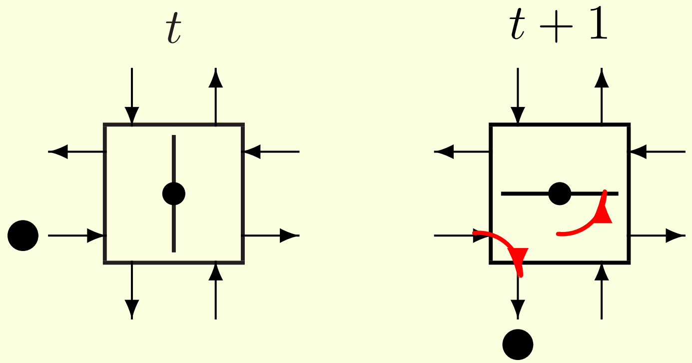
- Its operations are very easy to understand.

Operations of a rotary element (RE)

- The bar in the box controls an incoming signal.
- Parallel case:



- Orthogonal case:



- Assume signal 1 is given at most one input line.
- An RE is a kind of reversible sequential machine.

Reversible sequential machine (RSM)

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

Q : a set of states

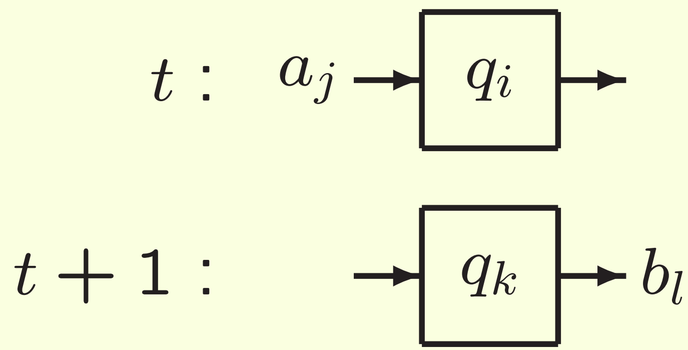
Σ : a set of input symbols

Γ : a set of output symbols

$\delta : Q \times \Sigma \rightarrow Q \times \Gamma$: a move function

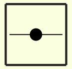
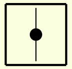
(Q: present state; Σ: input; Q: next state; Γ: output)

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



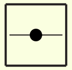
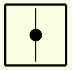
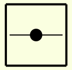
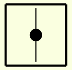
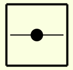
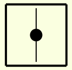
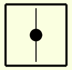
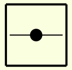
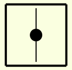
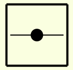
- M is called an RSM, if δ is one-to-one.

Rotary element is formalized as an RSM

$M_{\text{RE}} = ($ {, }, $\{n, e, s, w\}$, $\{n', e', s', w'\}$, $\delta_{\text{RE}})$

internal states — input symbols — output symbols — move function

- The move function δ_{RE}:

Present state	Input			
	n	e	s	w
State H: 	 w'	 w'	 e'	 e'
State V: 	 s'	 n'	 n'	 s'

- From the next state and the output, we can determine the previous state and the input uniquely.

2. How can we construct reversible machines by RLEMs?

2. How can we construct reversible machines by RLEMs?

Answer (though it is difficult to state in one line):

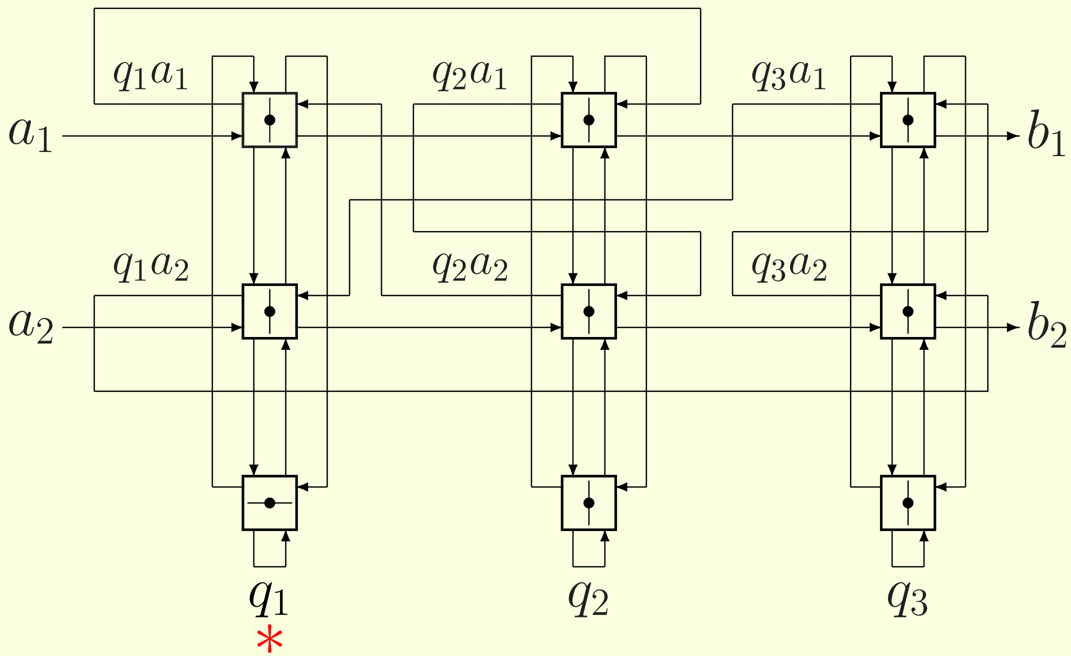
- They can be constructed very simply.

Reversible sequential machine (RSM)

- We can construct any RSM by REs easily.

Example:

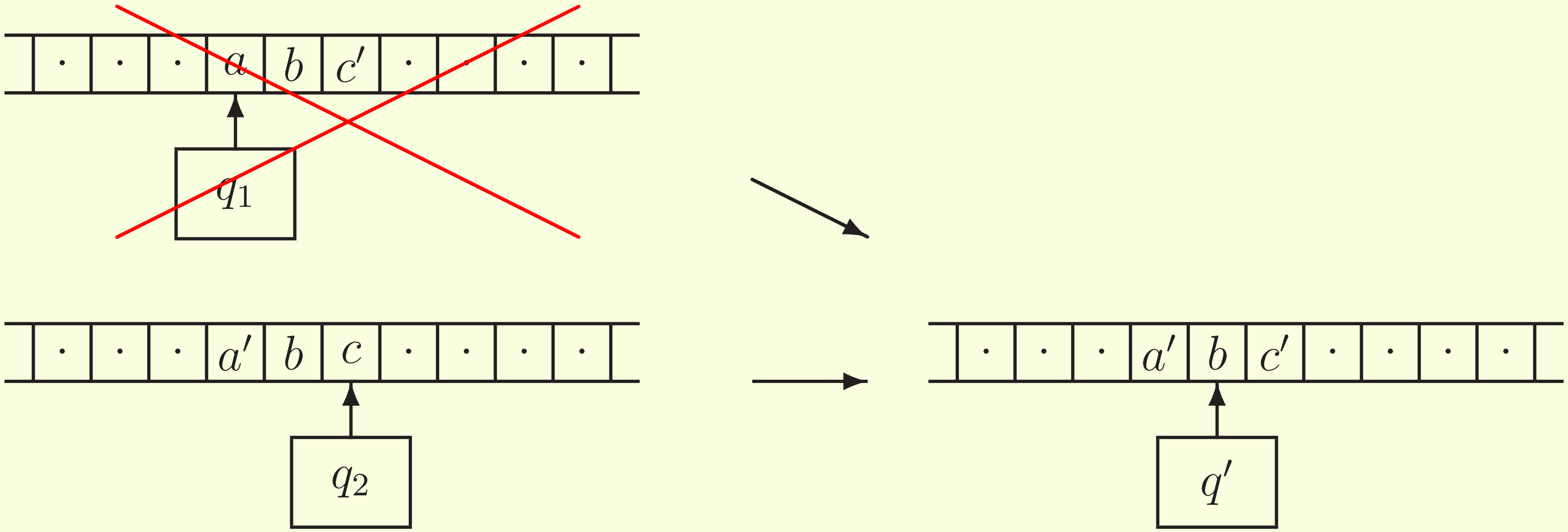
Input	State		
	q_1	q_2	q_3
a_1	$q_2\ b_1$	$q_2\ b_2$	$q_1\ b_2$
a_2	$q_3\ b_2$	$q_1\ b_1$	$q_3\ b_1$



[Morita, 2003]

Reversible Turing Machines (RTMs)

- A "backward deterministic" TM.



- We can also construct any RTM by REs simply.

A Simple Example of an RTM: T_{parity}

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

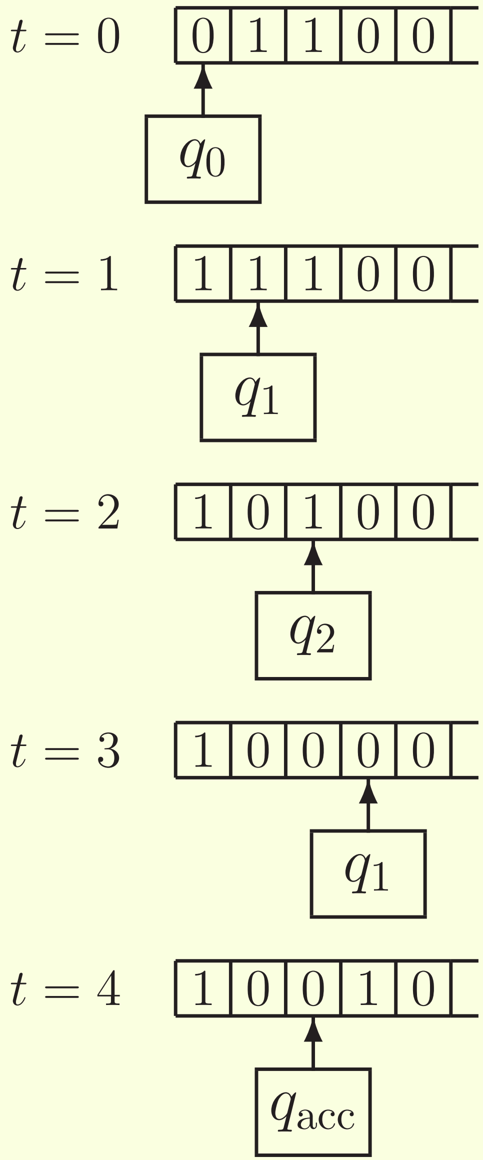
$Q = \{q_0, q_1, q_2, q_{\text{acc}}, q_{\text{rej}}\}$: a set of states

$S = \{0, 1\}$: 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_{parity} halts in the final state q_{acc}.
- If n is odd, T_{parity} halts in the state q_{rej}.
- All the symbols read by T_{parity} are complemented.

A computing process of T_{parity} for input "11"



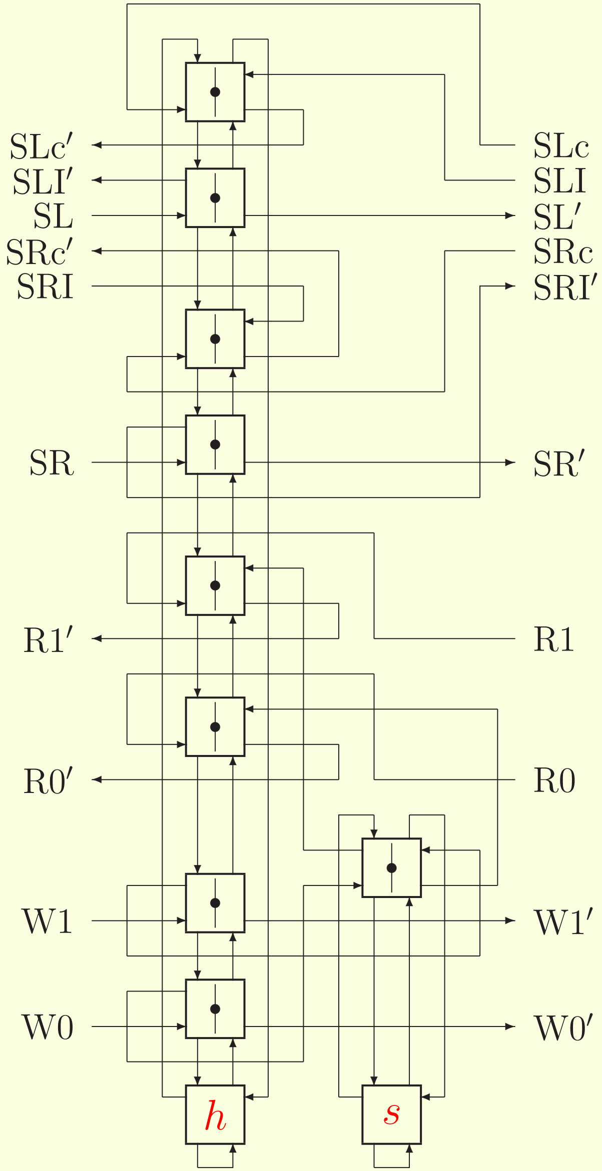
$$\delta = \{ [\, q_0, 0, 1, R, q_1 \,], \\ [\, q_1, 0, 1, L, q_{\mathsf{acc}} \,], \\ [\, q_1, 1, 0, R, q_2 \,], \\ [\, q_2, 0, 1, L, q_{\mathsf{rej}} \,], \\ [\, q_2, 1, 0, R, q_1 \,] \, \}$$

Realizing RTMs by REs

- A memory cell (i.e., a tape square), and a finite-state control of an RTM are formalized as RSMs.
- Therefore, they are constructed by REs systematically as shown above.
- Here we use additional techniques to reduce the number of REs in them. [Morita, 2001, 2010]

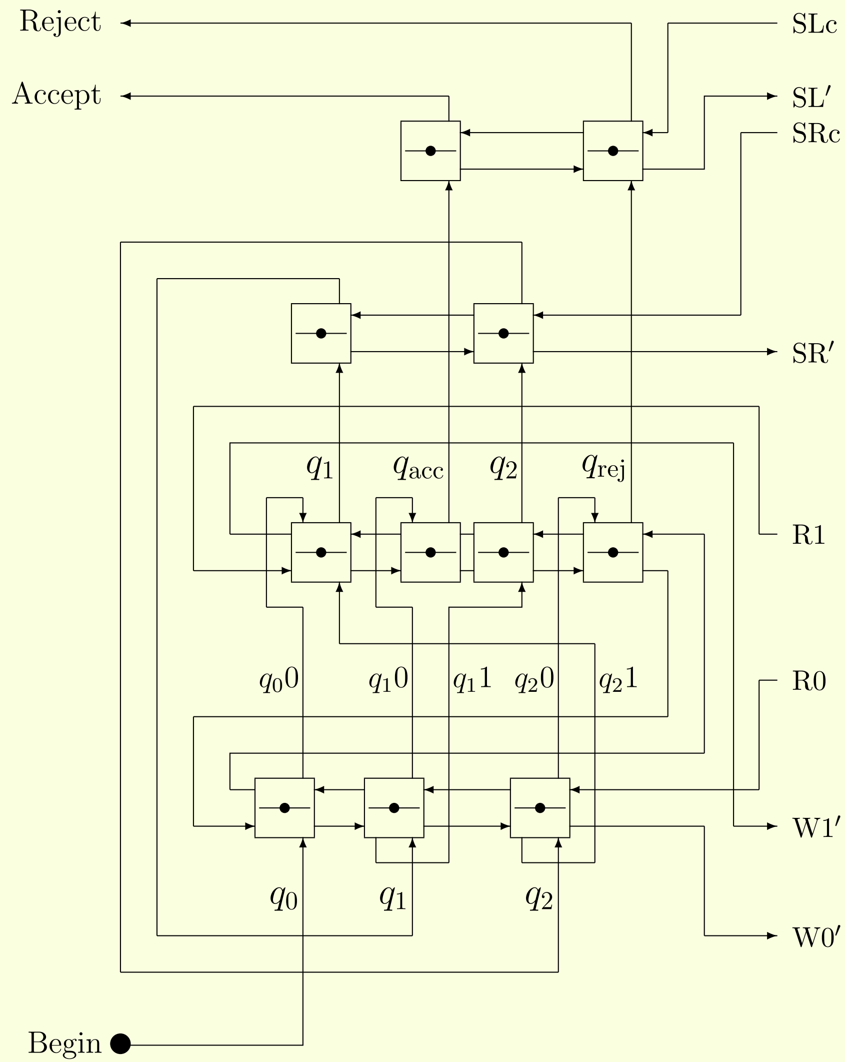
Memory cell realized by REs

- 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 [RE state 0 symbol] if the value is 0, and [RE state 1 symbol] if it is 1.

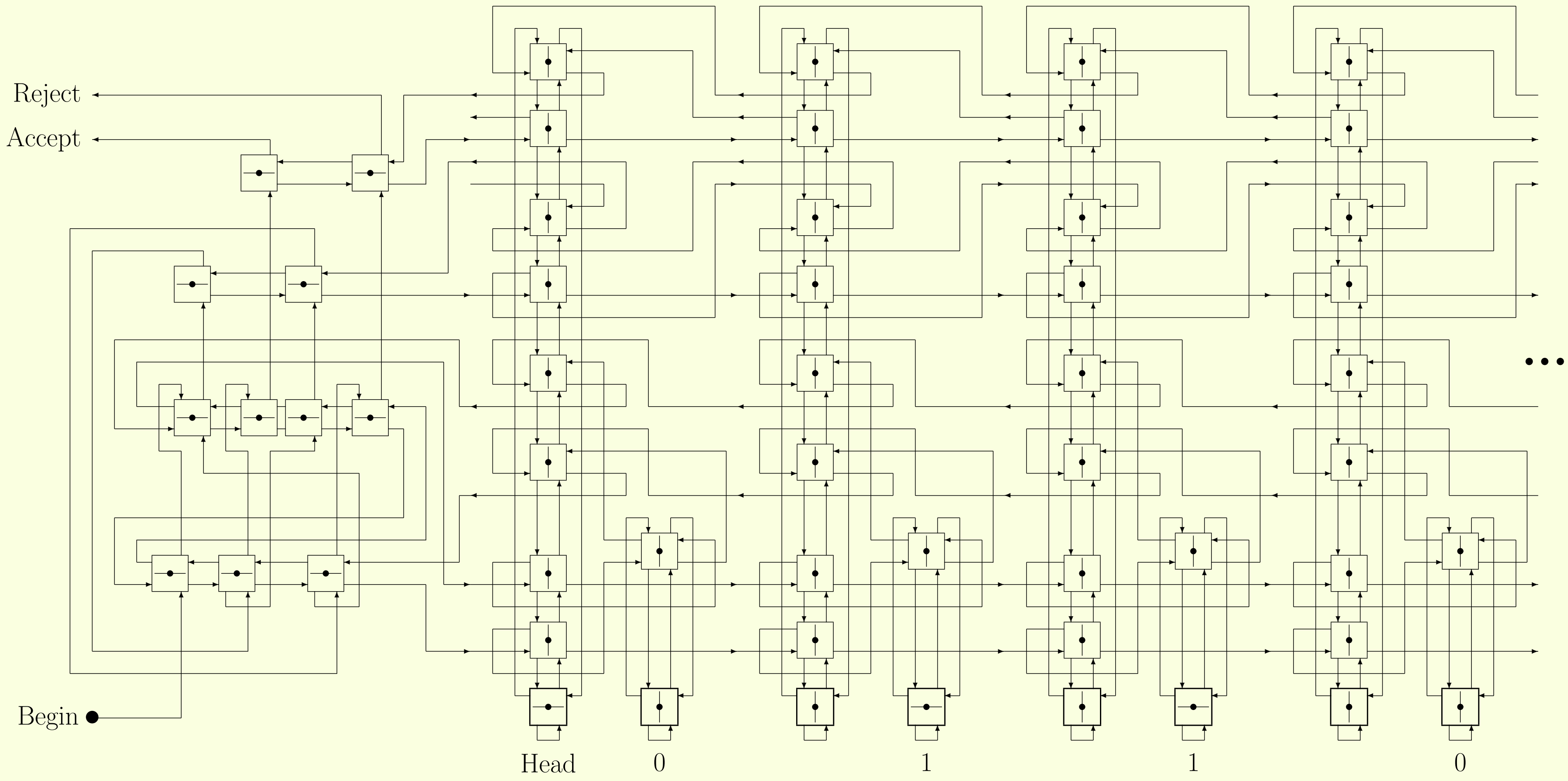


Finite-state control of T_{parity} realized by REs

- 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_{parity}



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

3. Can RLEMs be implemented in a reversible physical system efficiently?

3. Can RLEMs be implemented in a reversible physical system efficiently?

Answer (from my present knowledge):

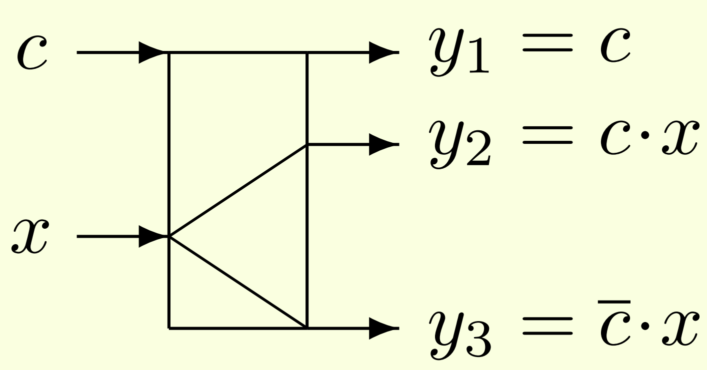
- "Yes," by a thought experiment.
- "Unknown," in a practical level.

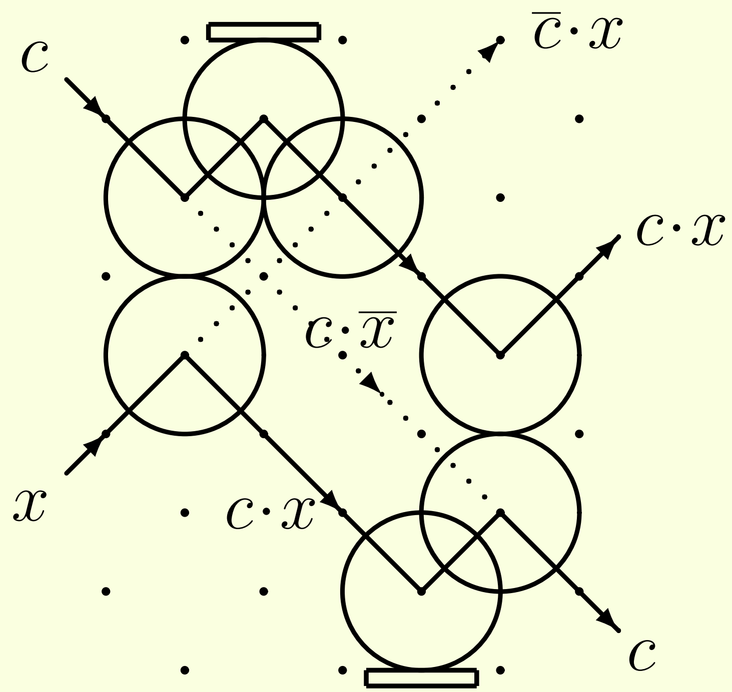
Billiard ball model (BBM)

- A reversible physical model of computing.

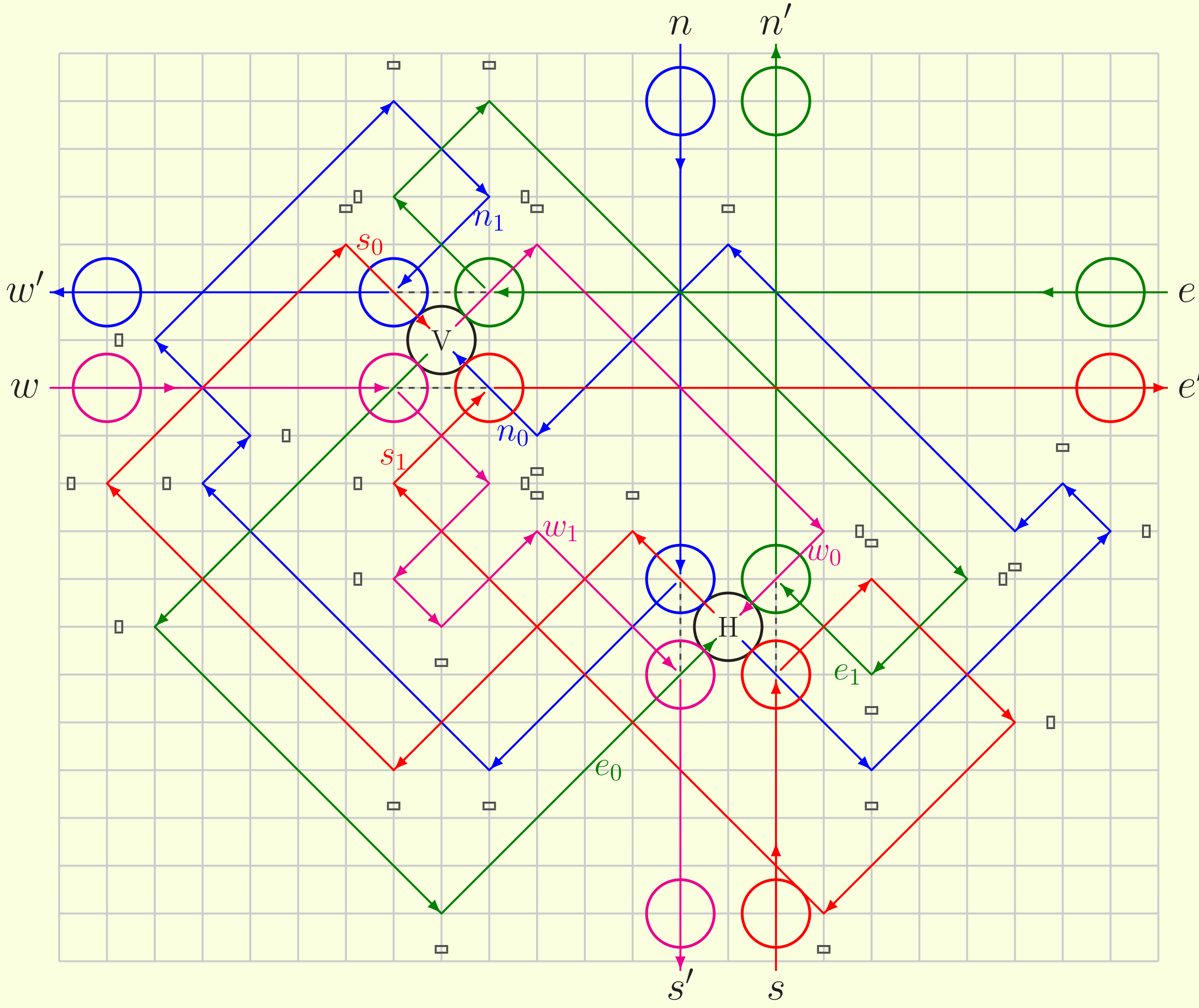
  [Fredkin and Toffoli, 1982]

- A switch gate is realized in the BBM.



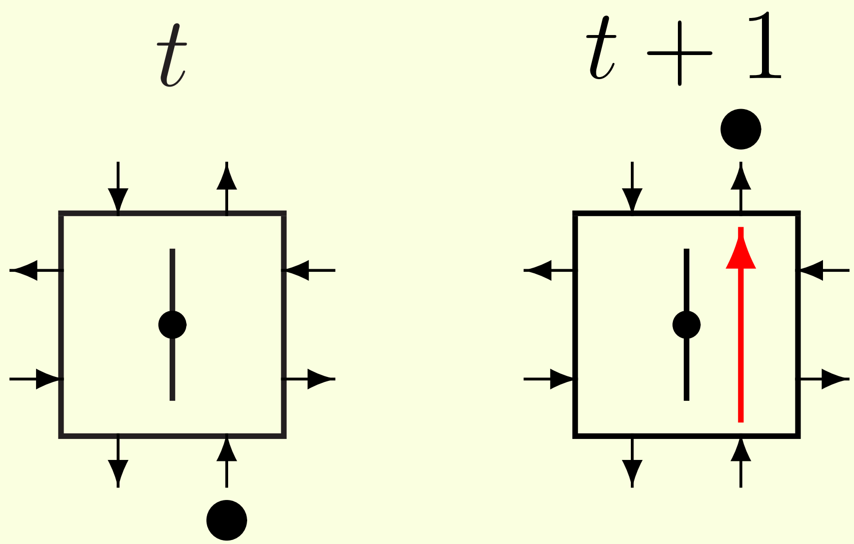


Realization of an RE by BBM

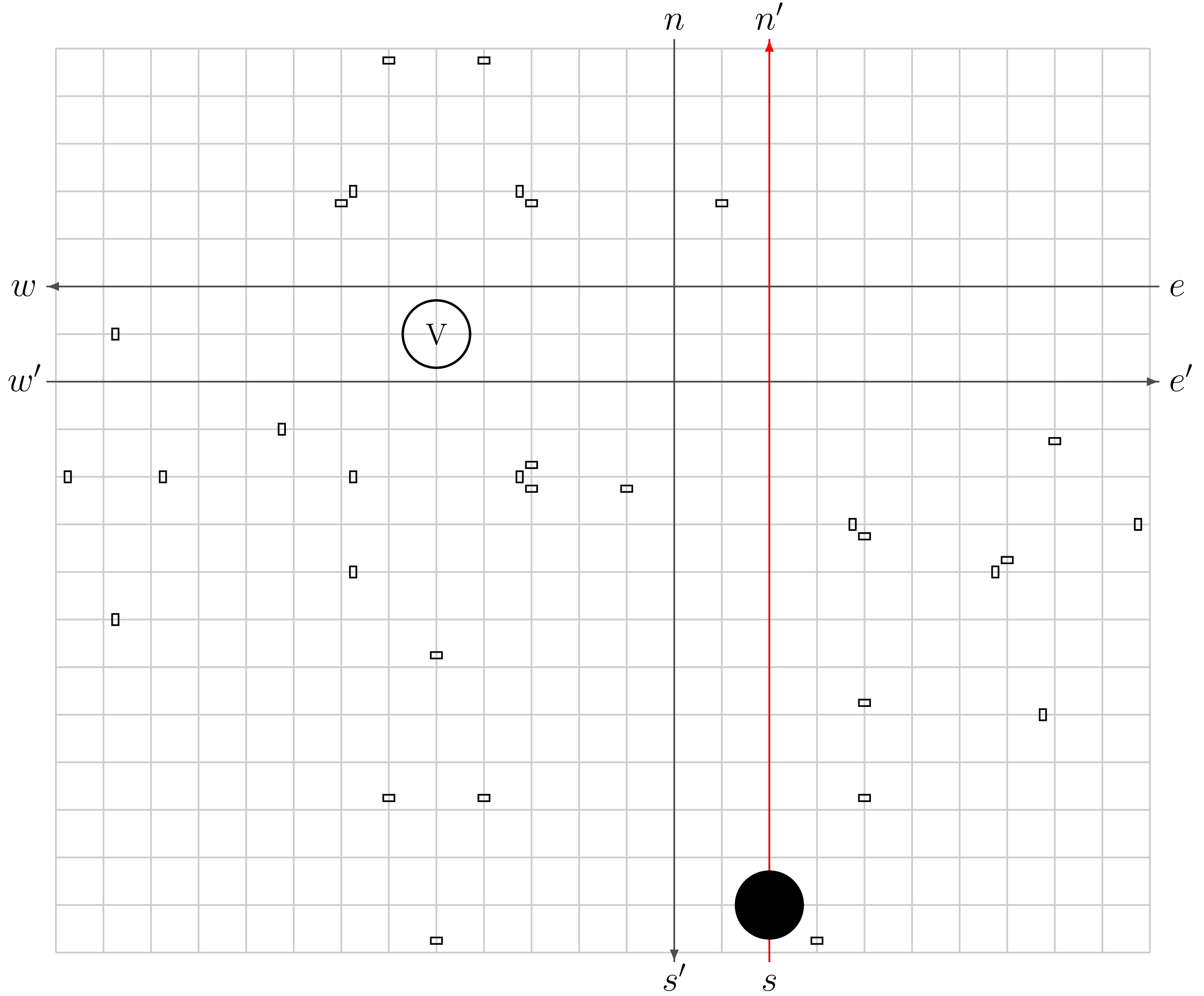


[Morita, 2008]

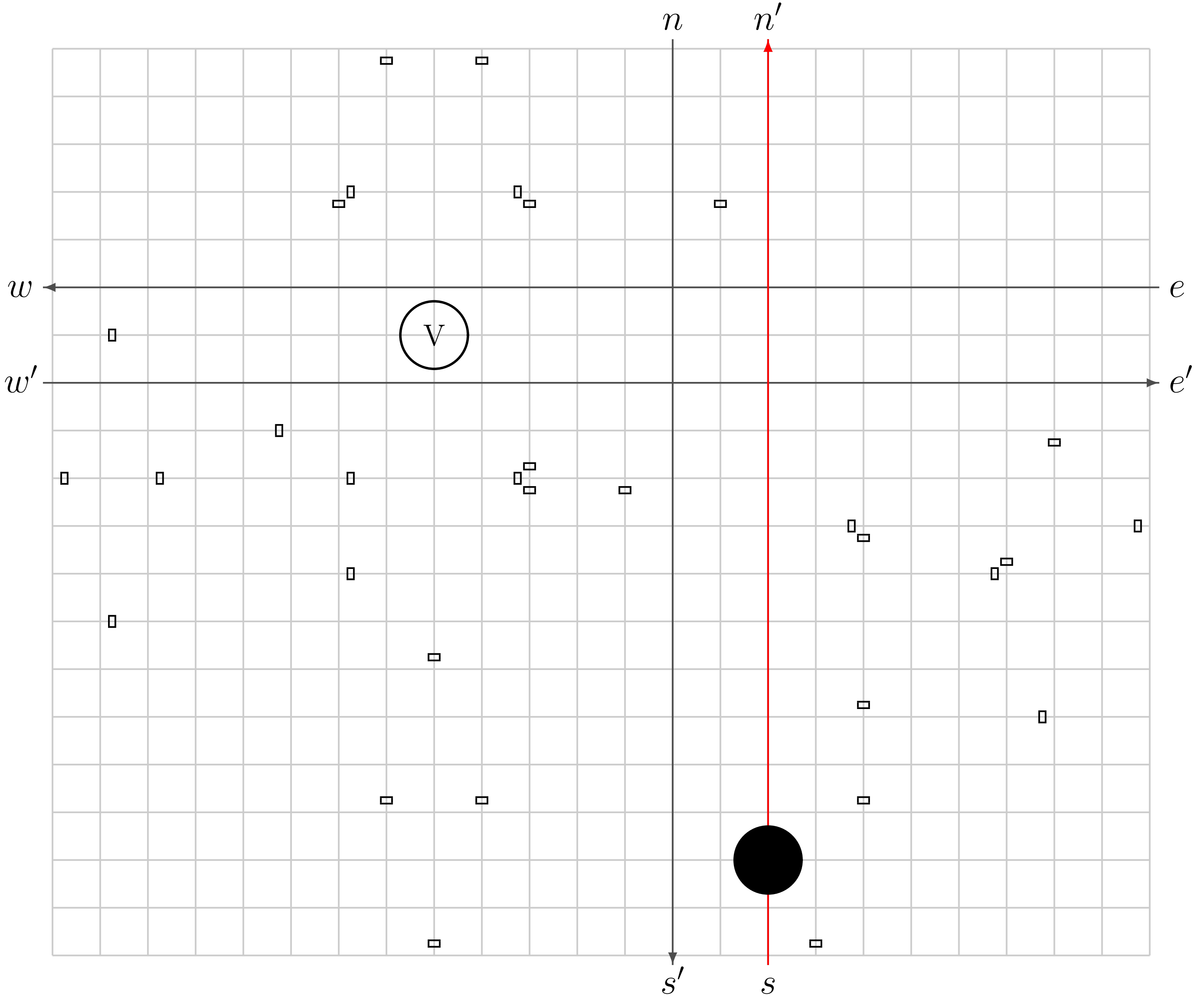
Parallel Case



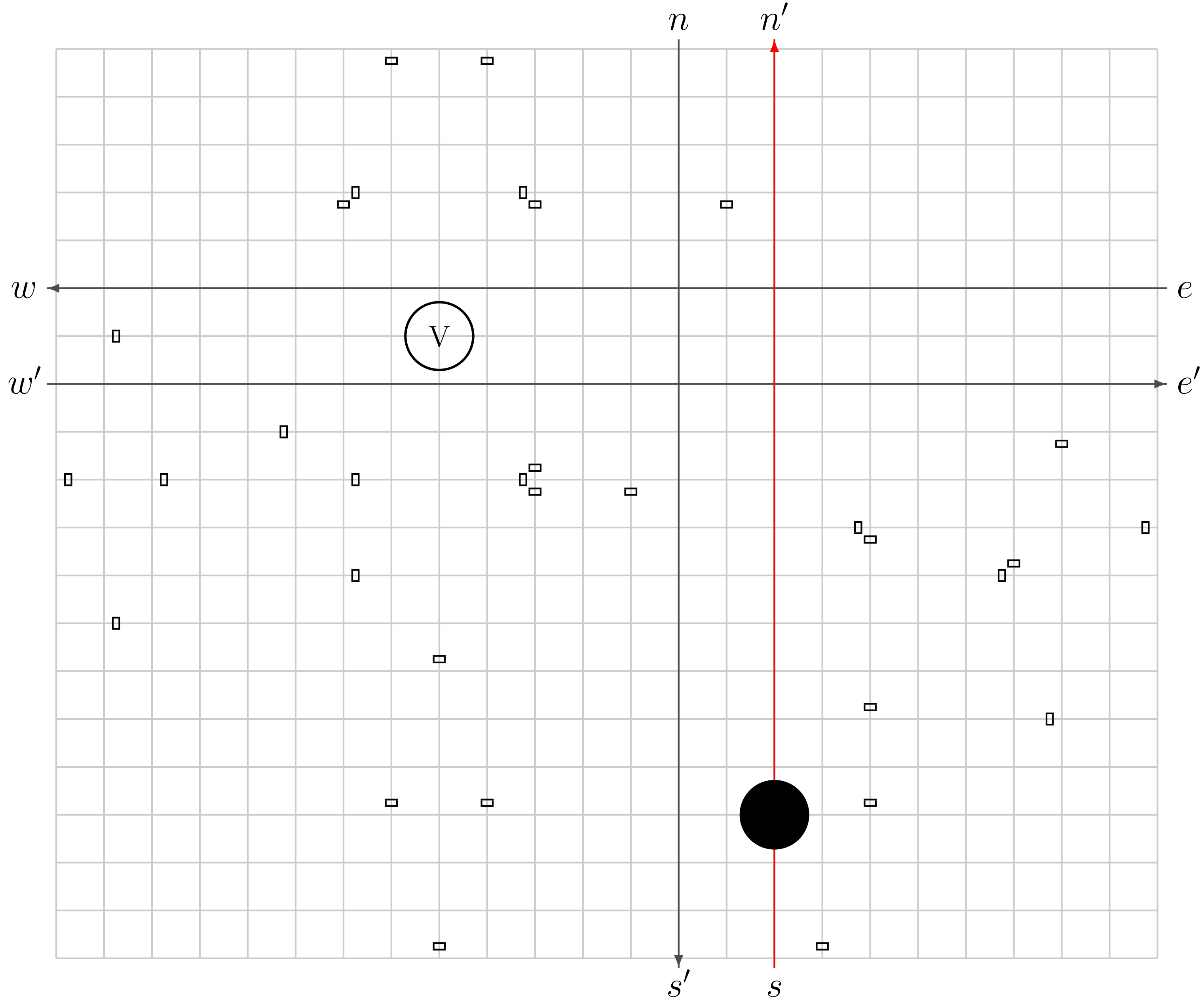
Movements of Balls (State: V, Input: s)



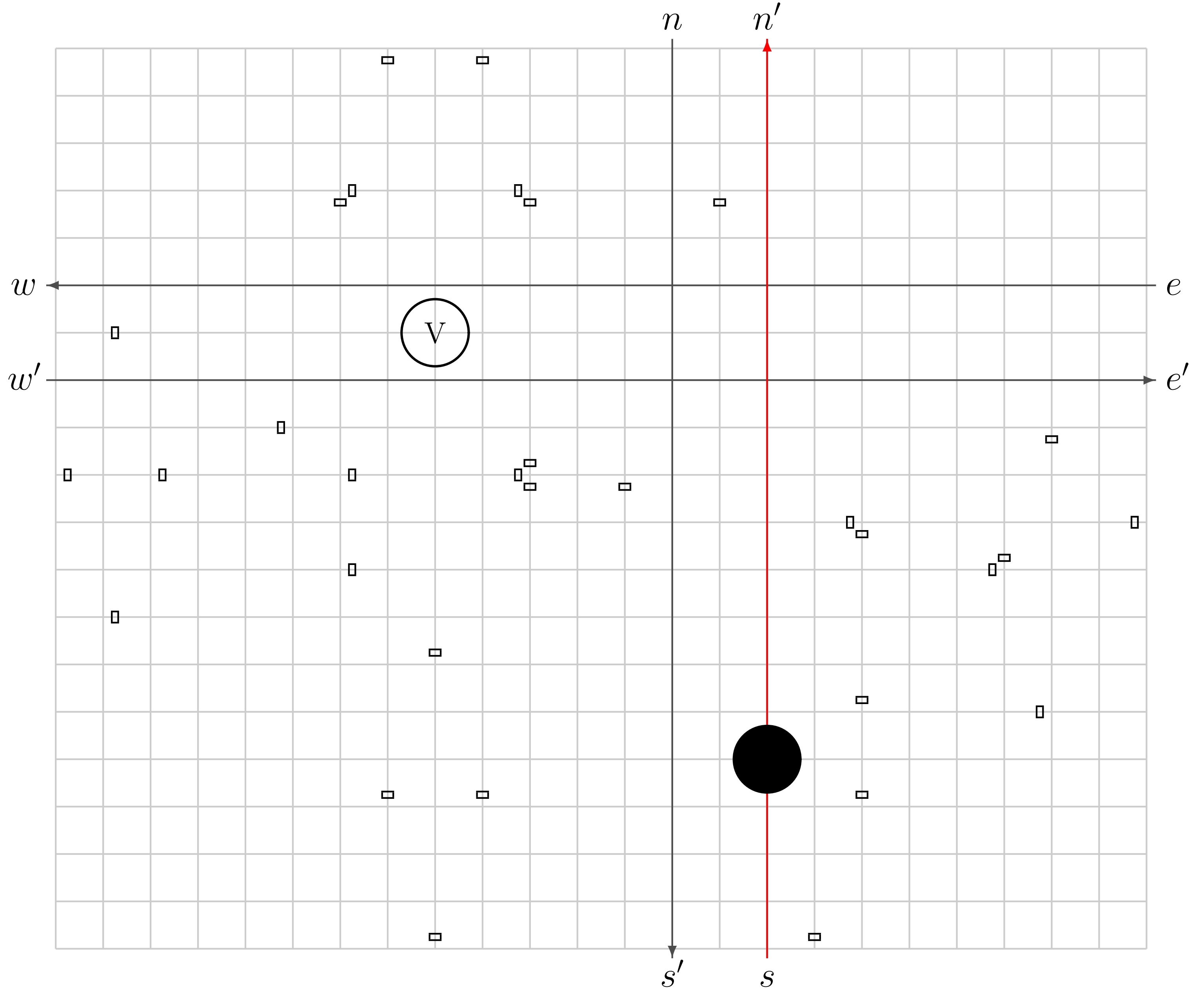
Movements of Balls (State: V, Input: s)



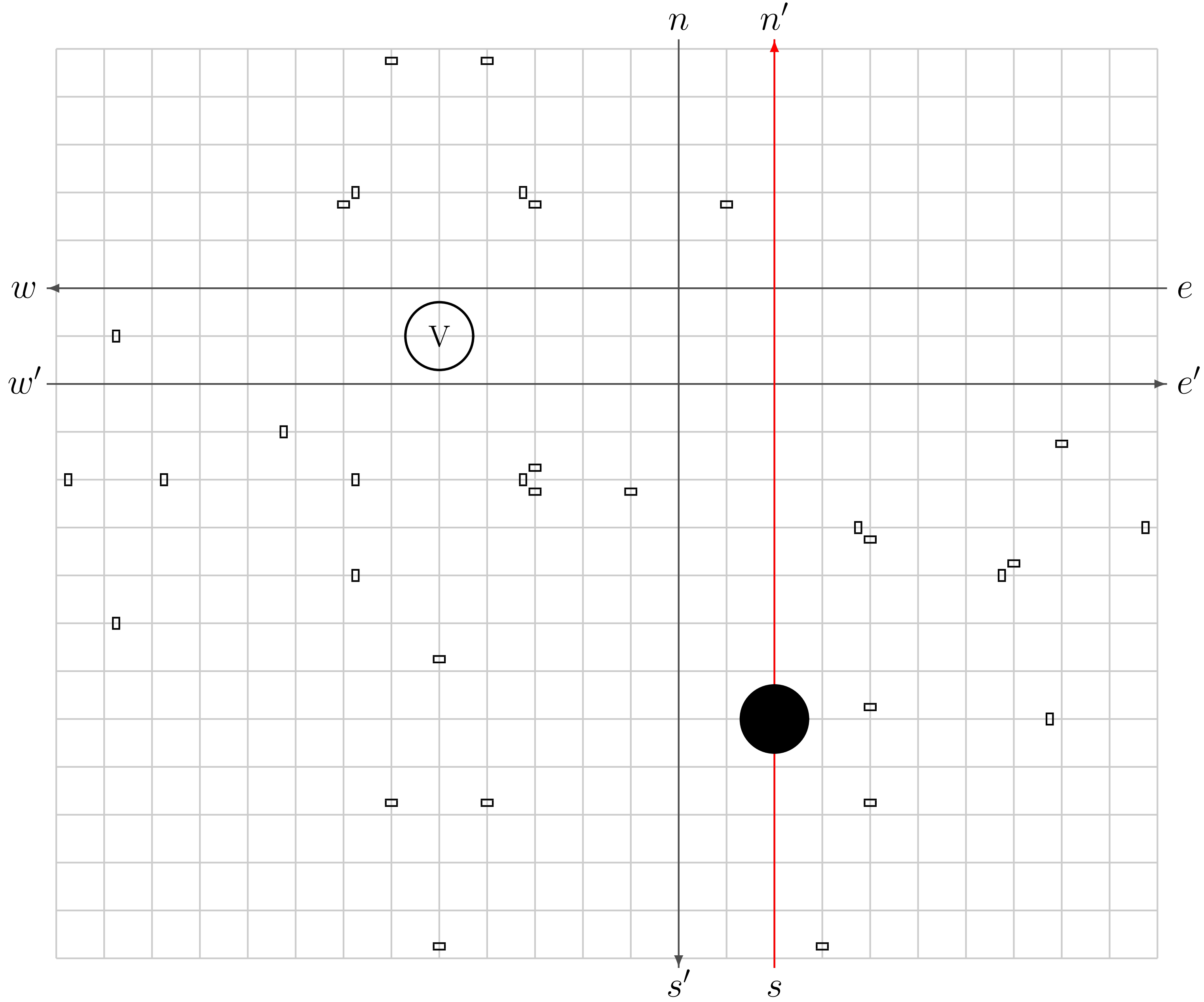
Movements of Balls (State: V, Input: s)



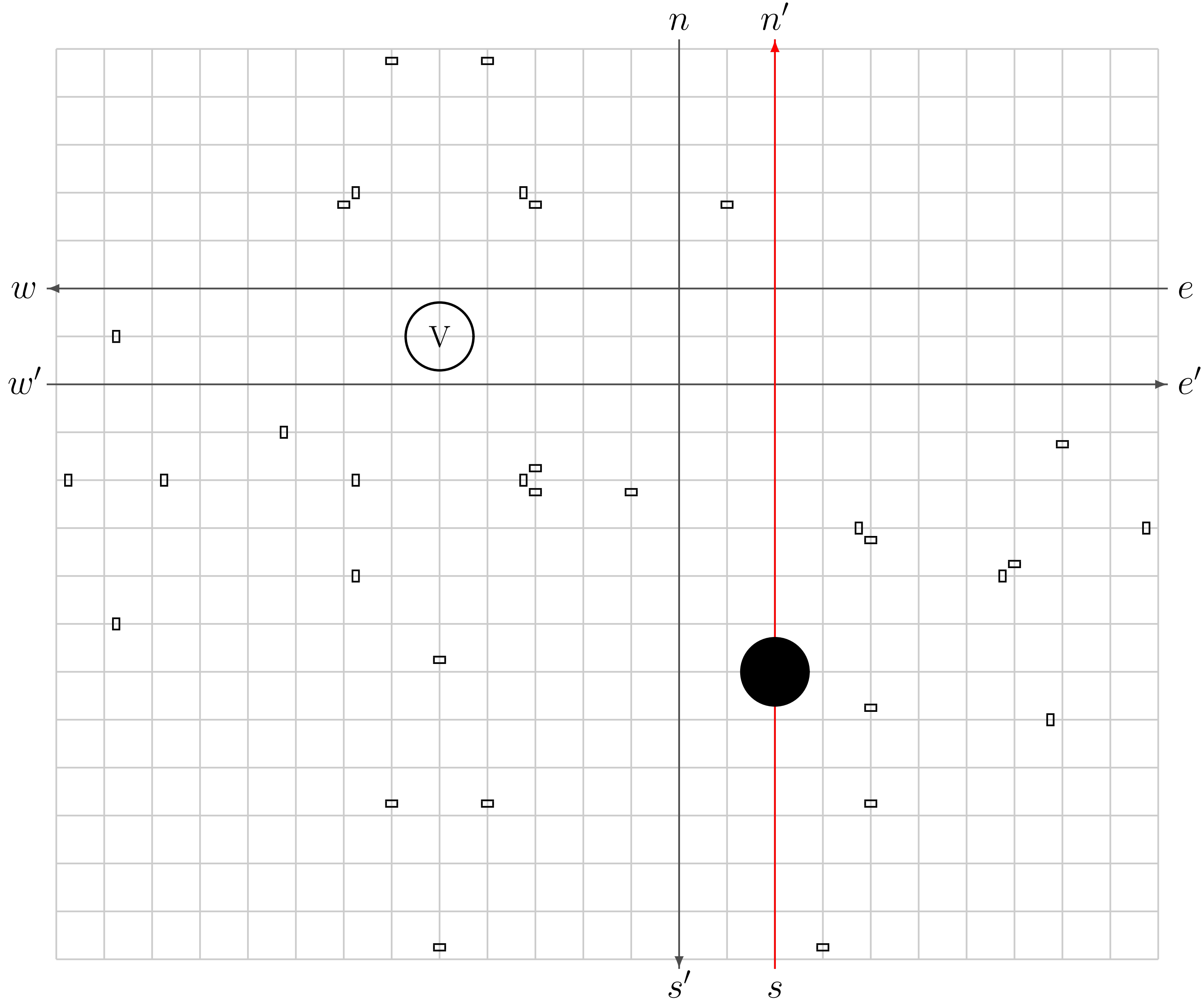
Movements of Balls (State: V, Input: s)



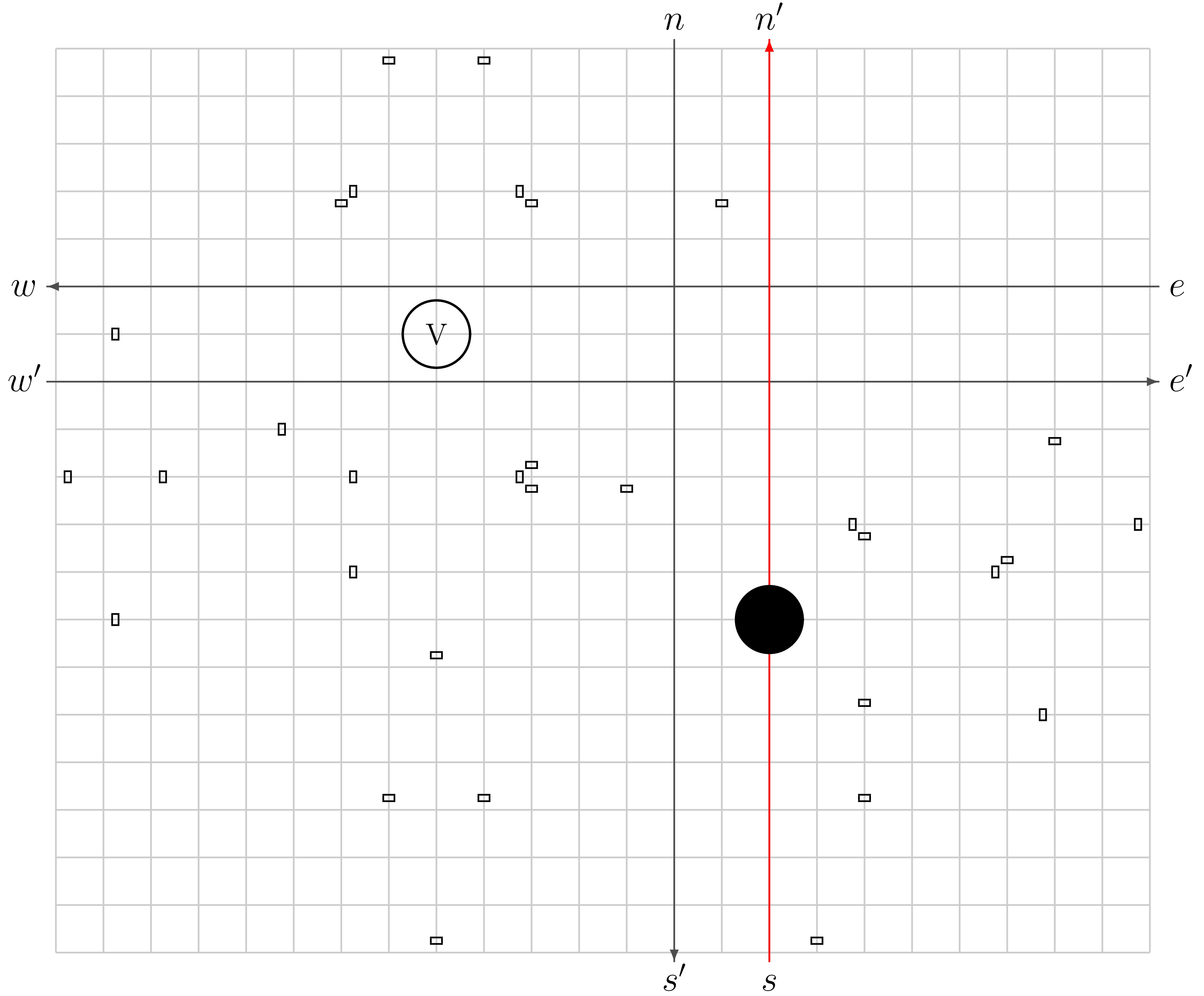
Movements of Balls (State: V, Input: s)



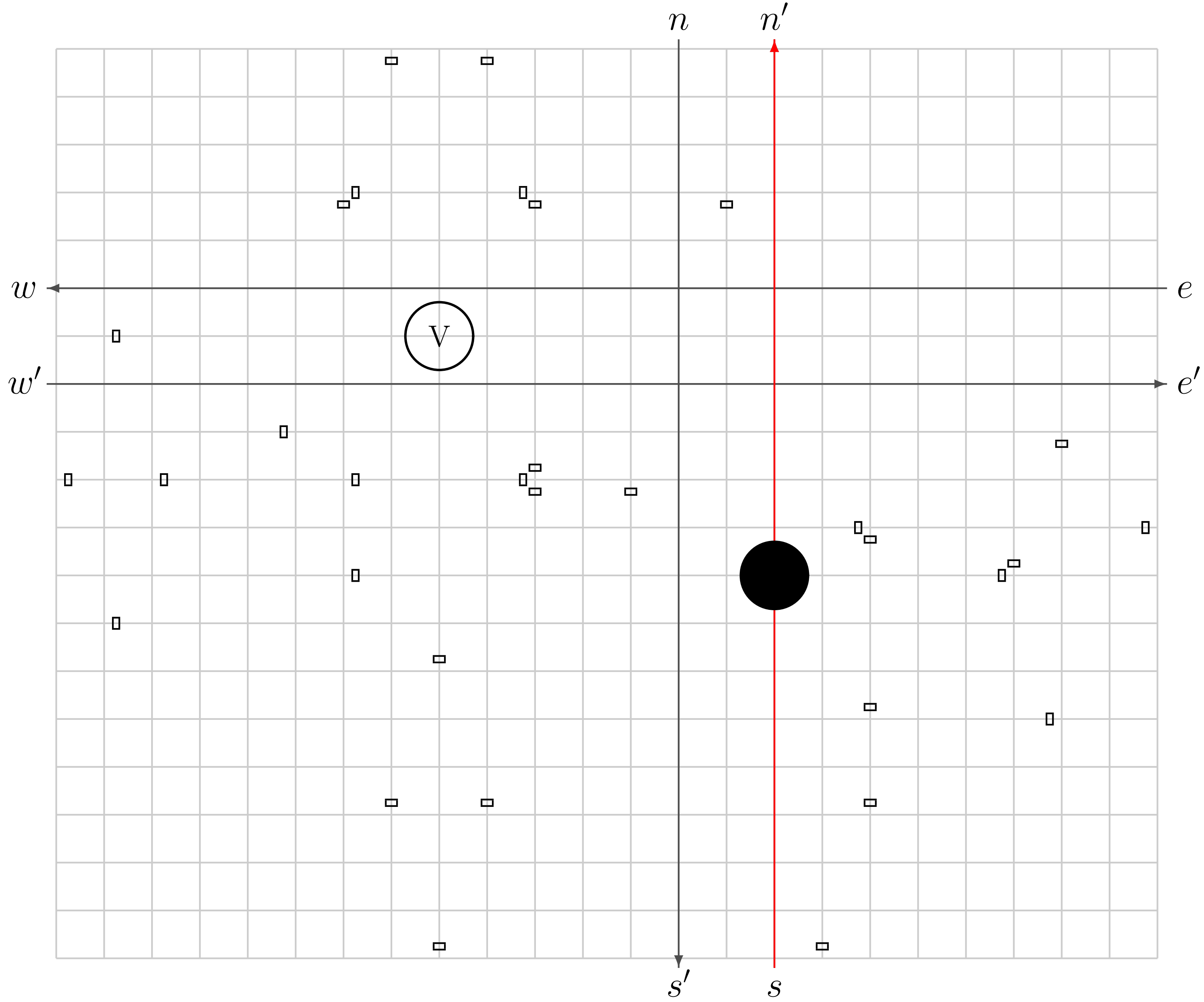
Movements of Balls (State: V, Input: s)



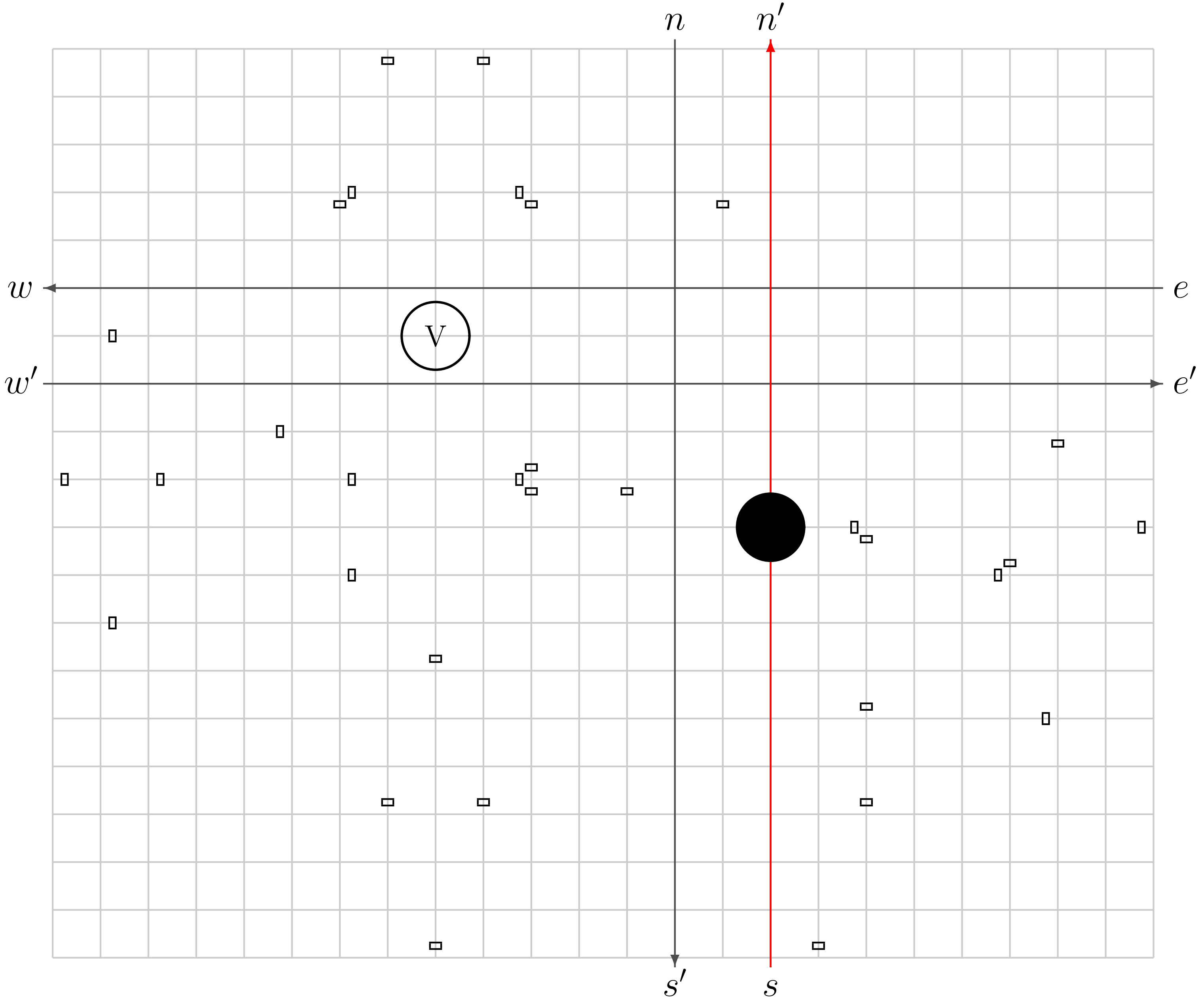
Movements of Balls (State: V, Input: s)



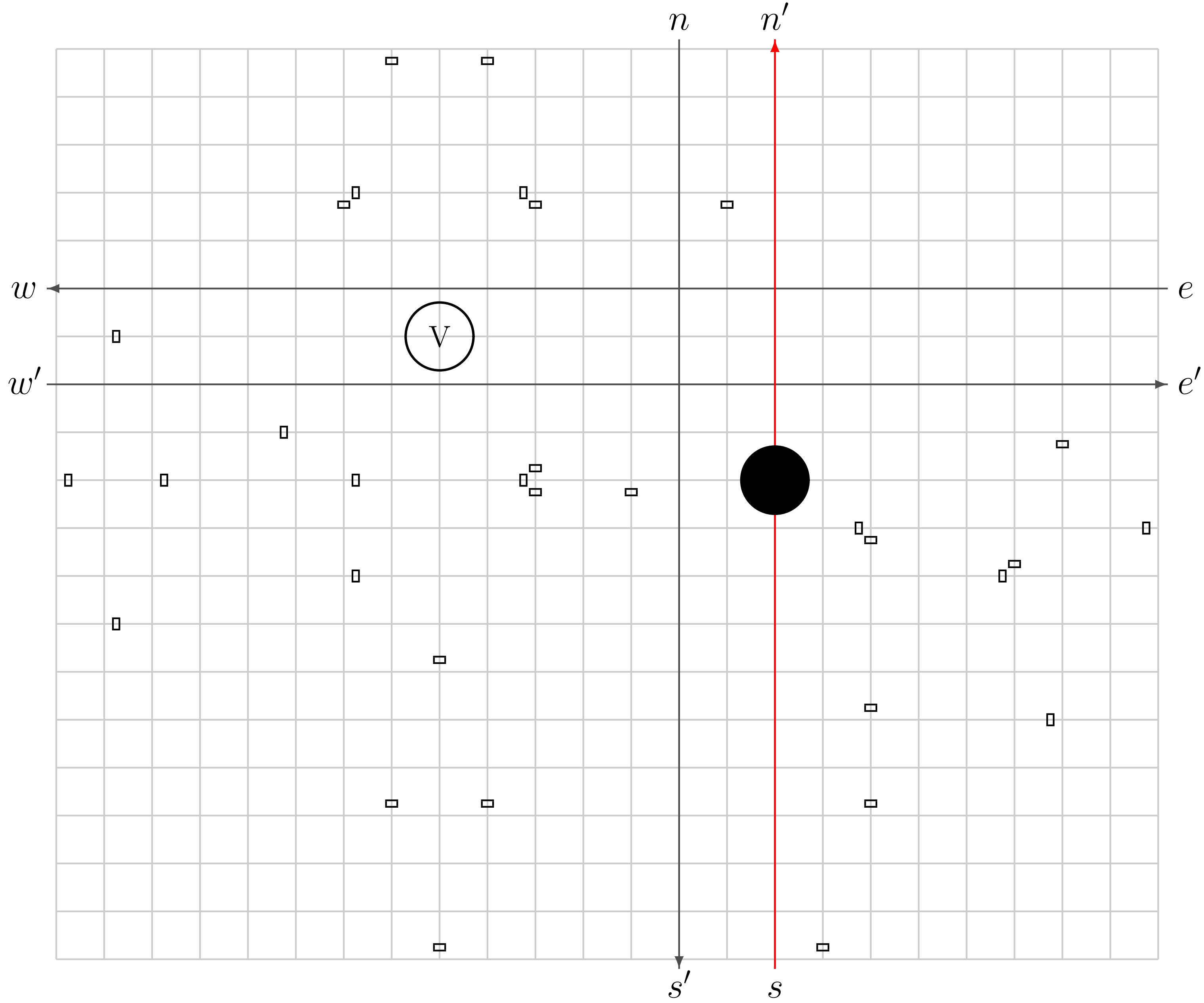
Movements of Balls (State: V, Input: s)



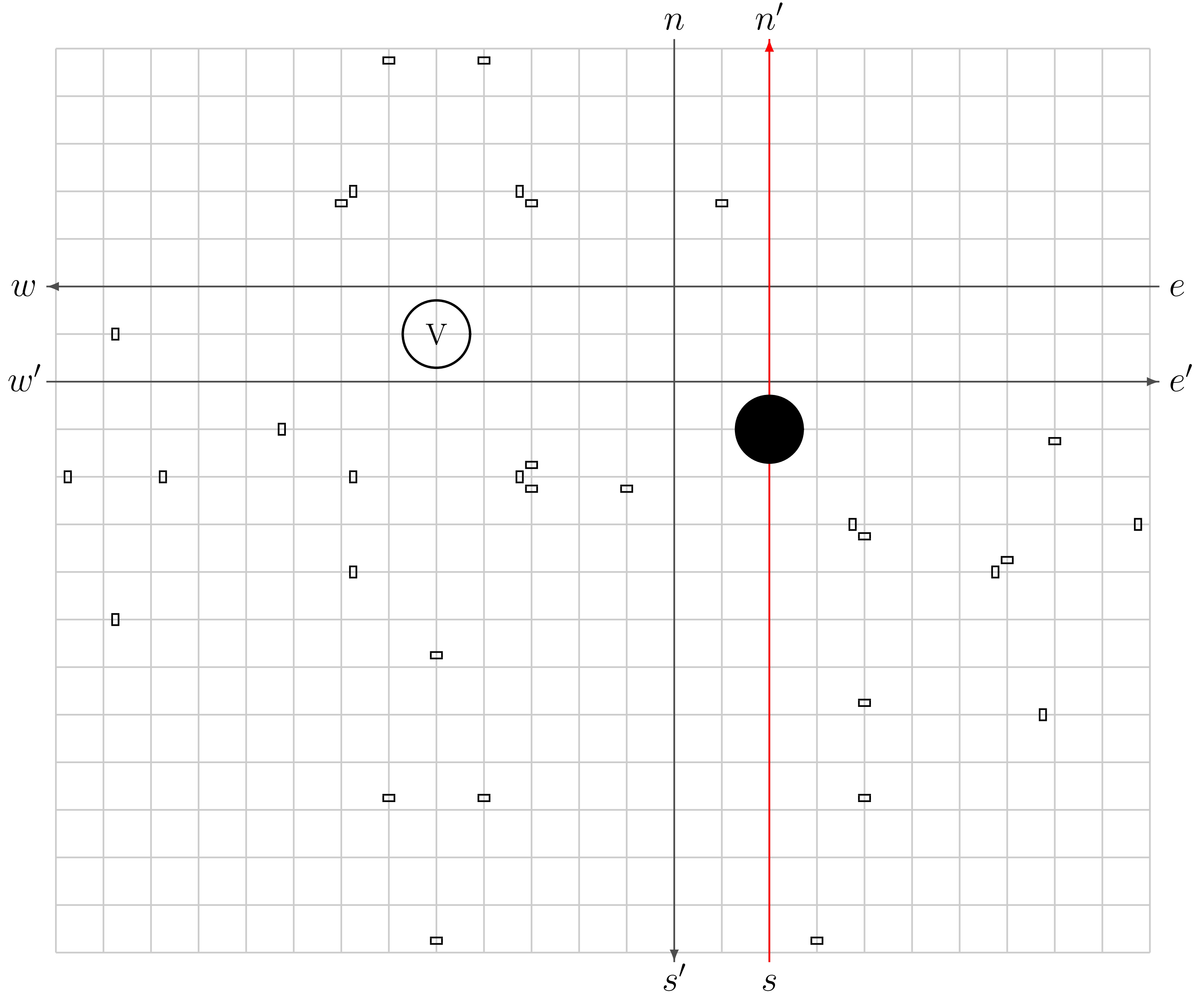
Movements of Balls (State: V, Input: s)



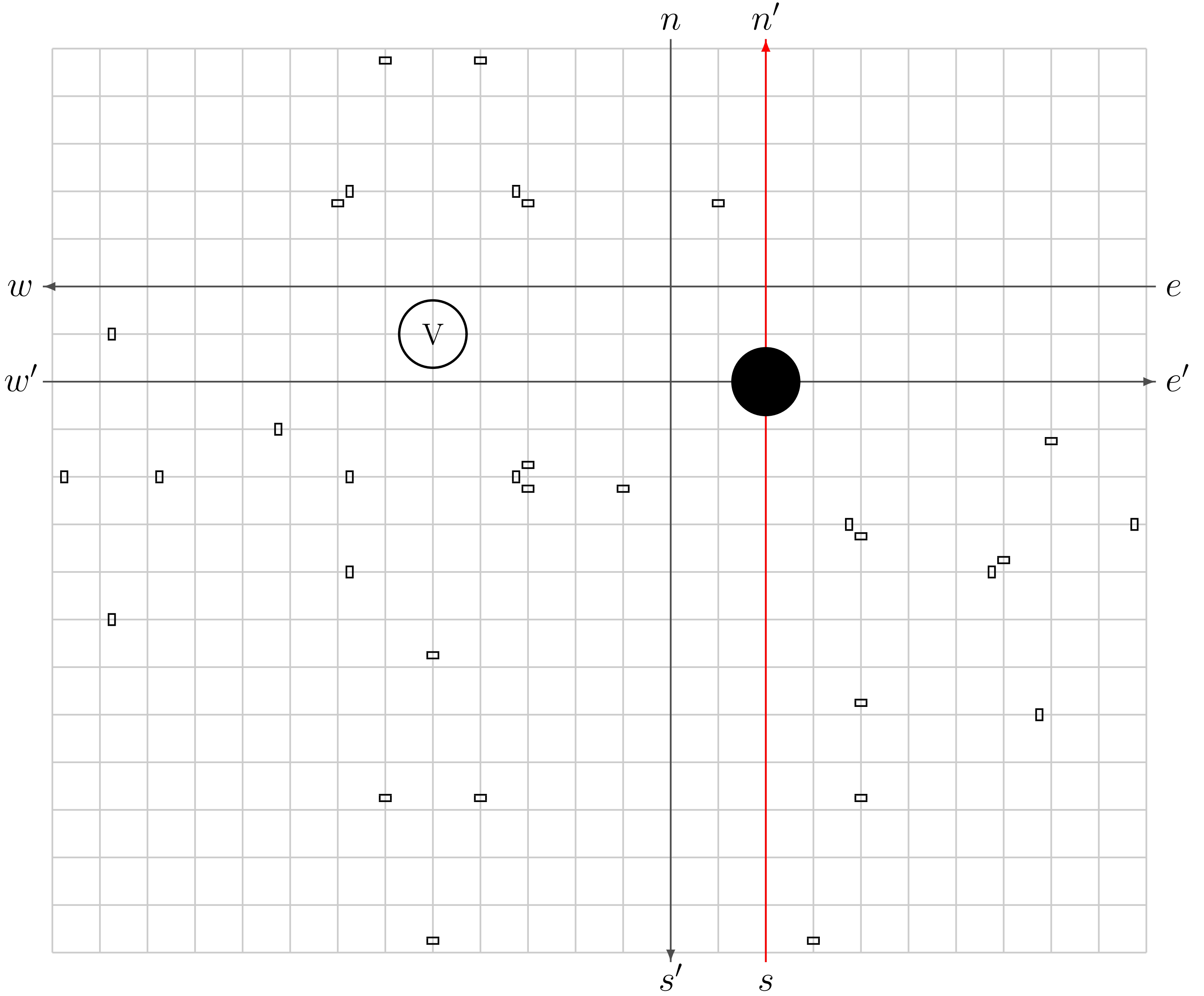
Movements of Balls (State: V, Input: s)



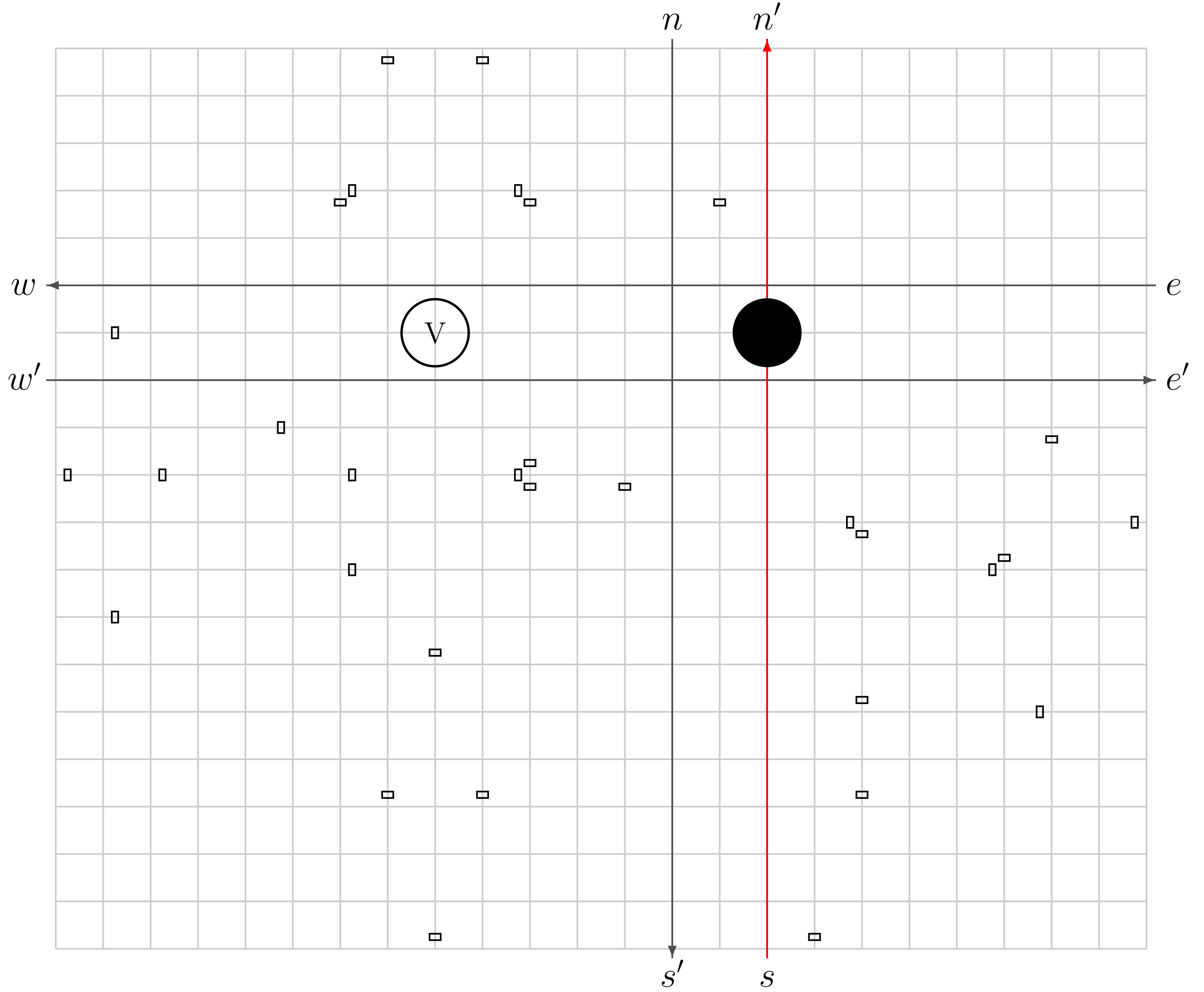
Movements of Balls (State: V, Input: s)



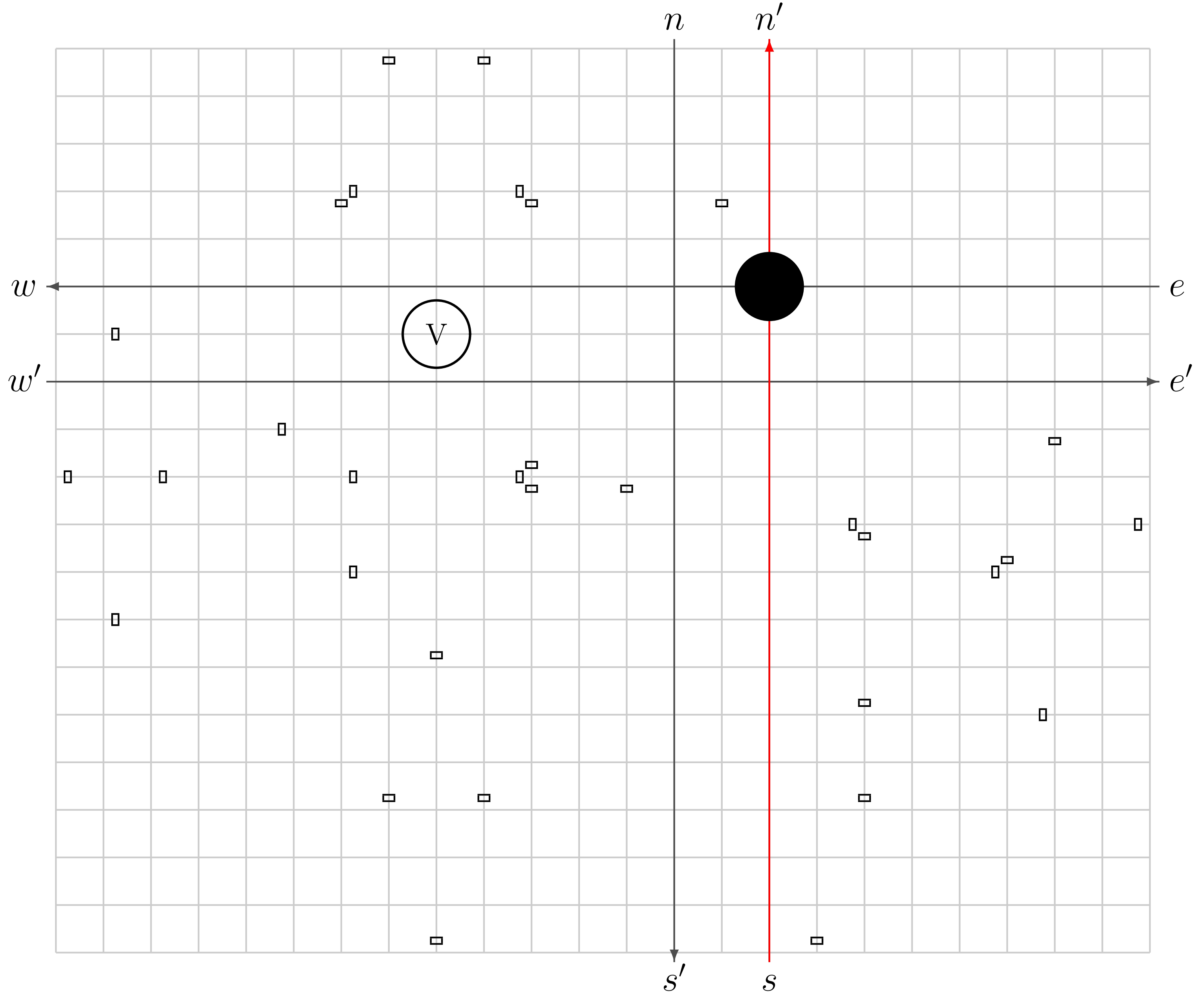
Movements of Balls (State: V, Input: s)



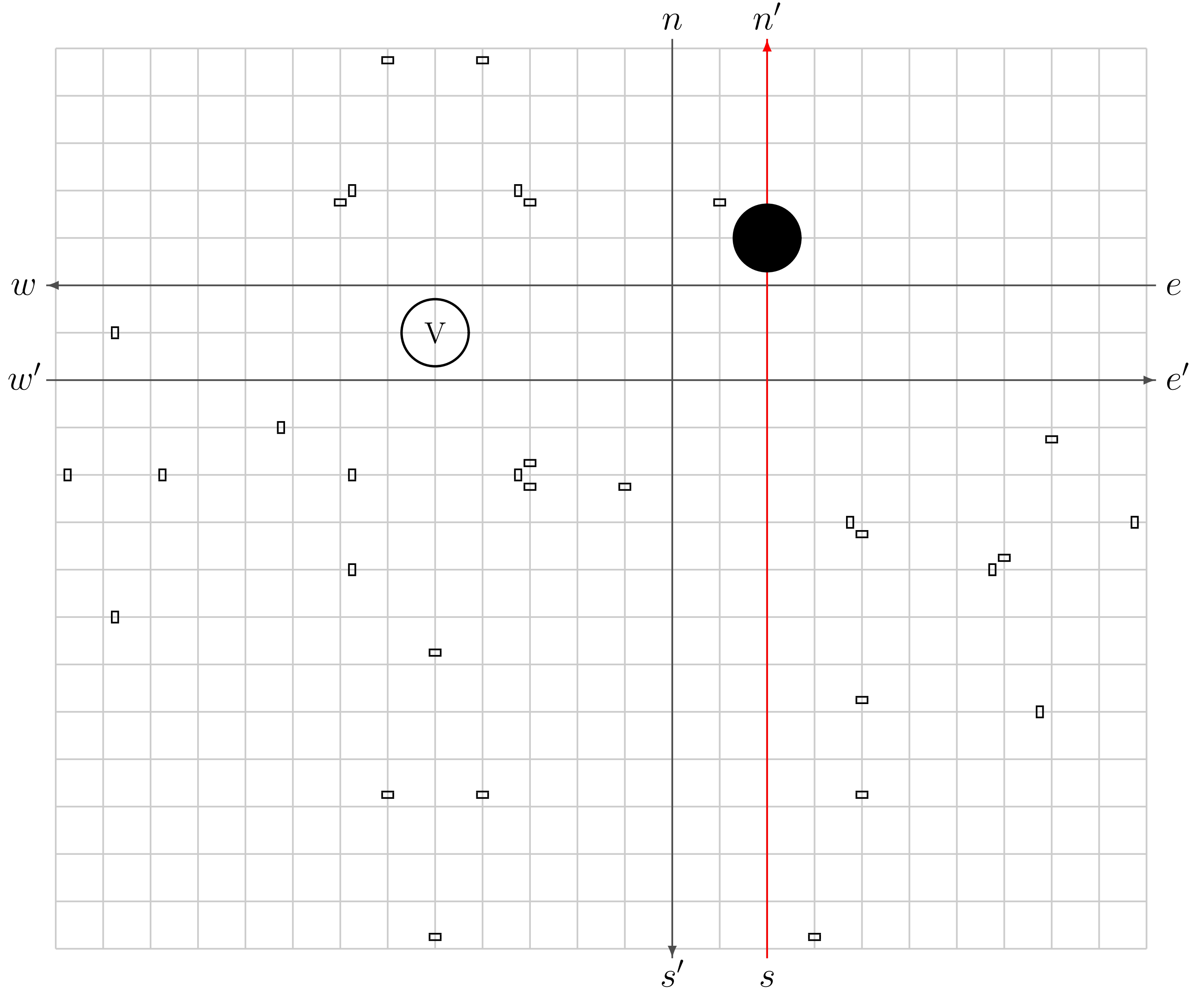
Movements of Balls (State: V, Input: s)



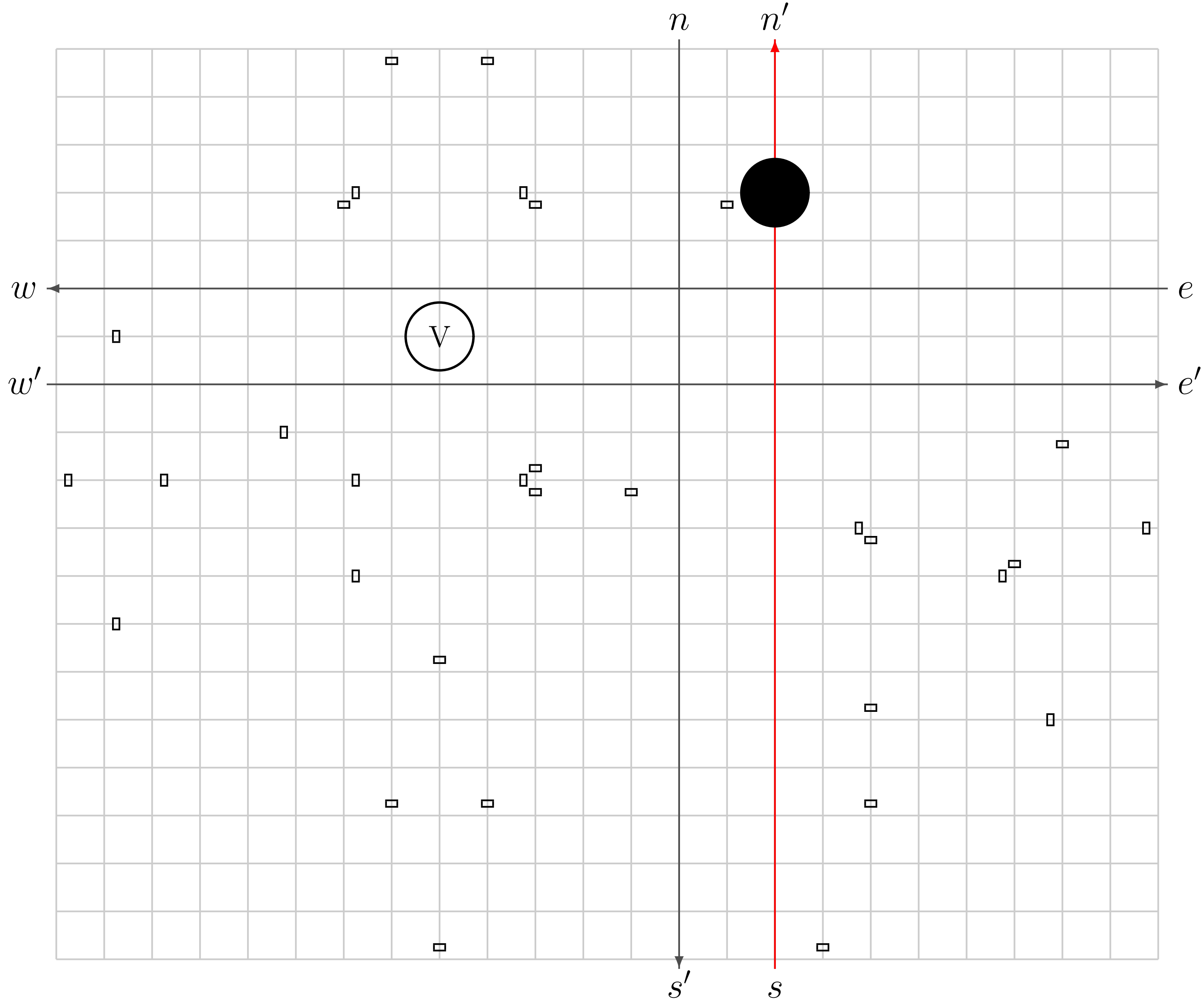
Movements of Balls (State: V, Input: s)



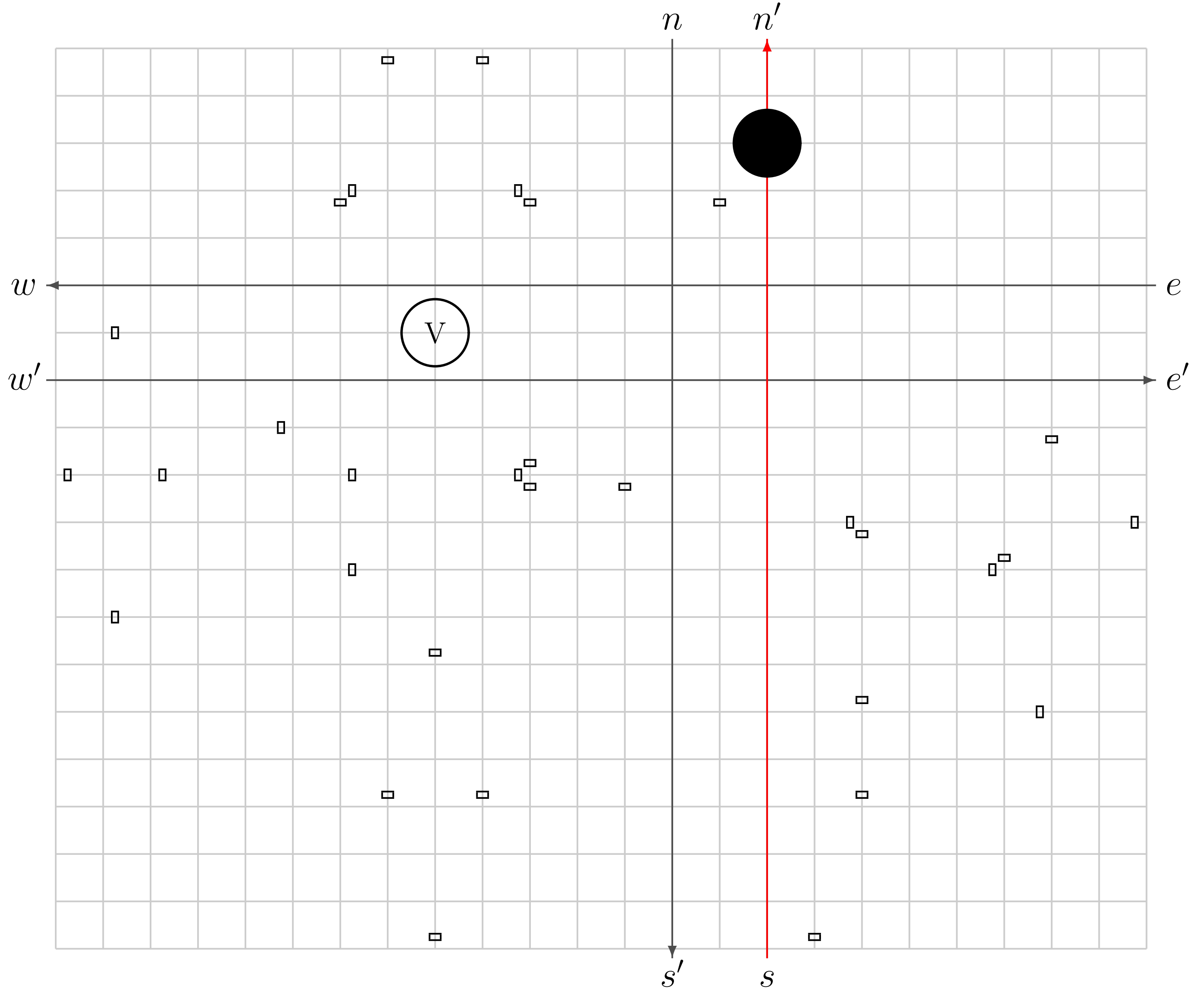
Movements of Balls (State: V, Input: s)



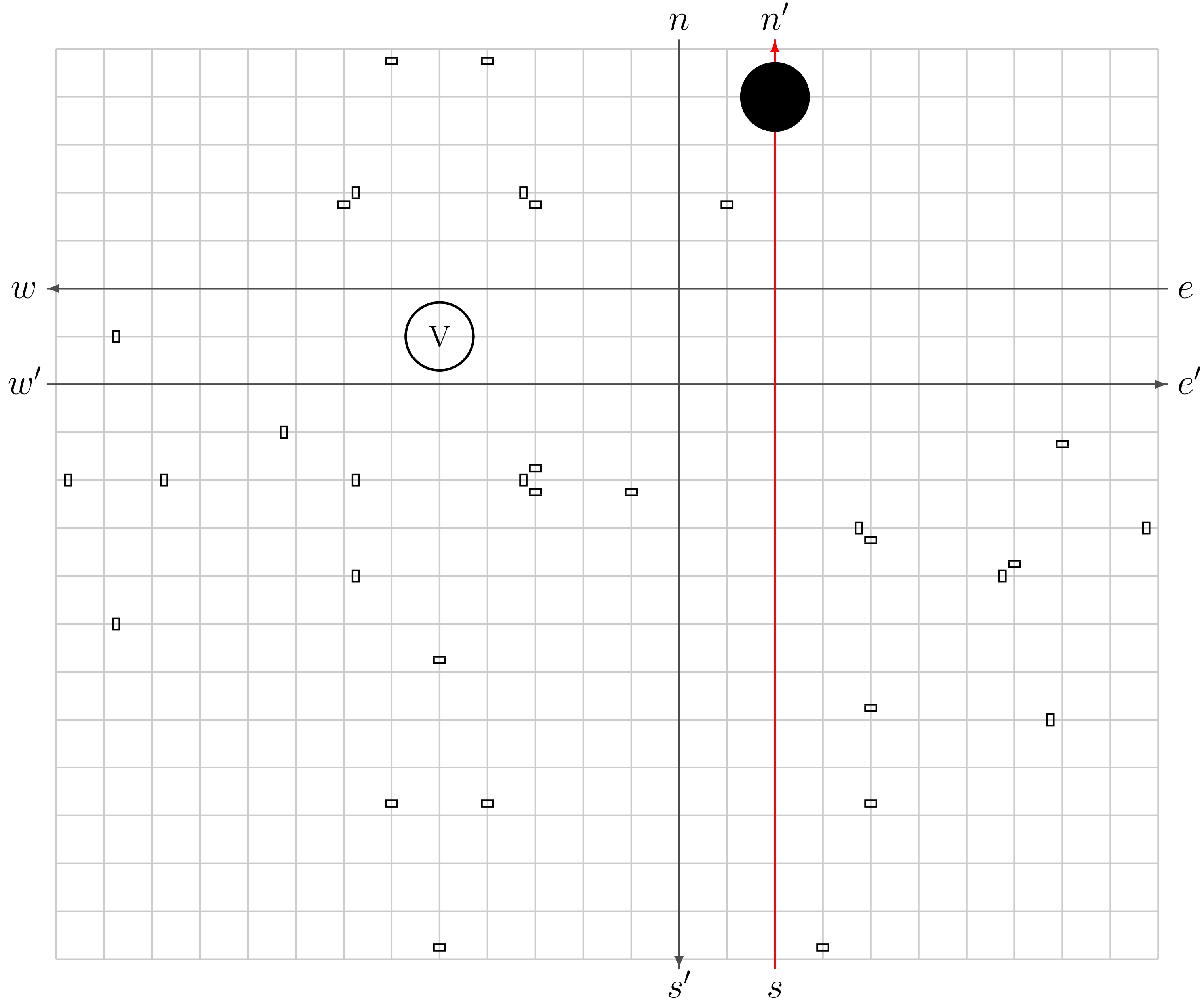
Movements of Balls (State: V, Input: s)



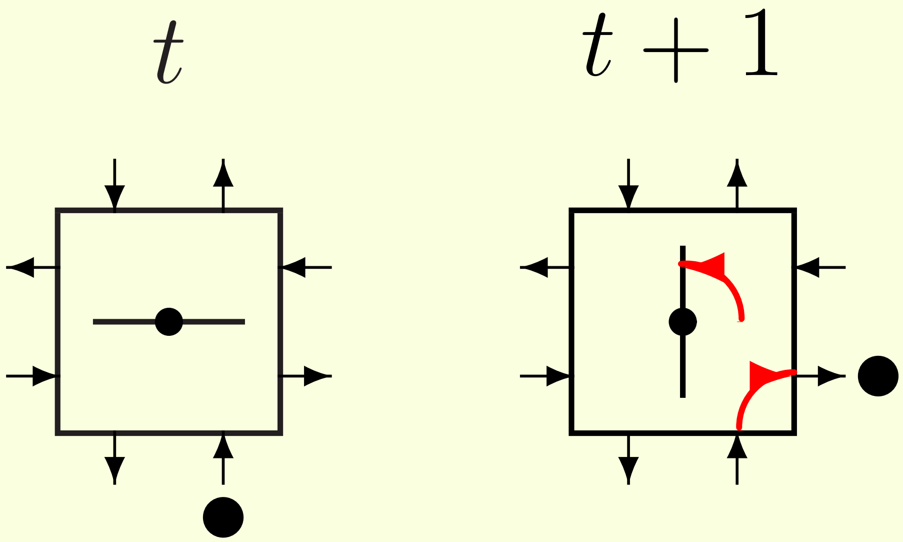
Movements of Balls (State: V, Input: s)



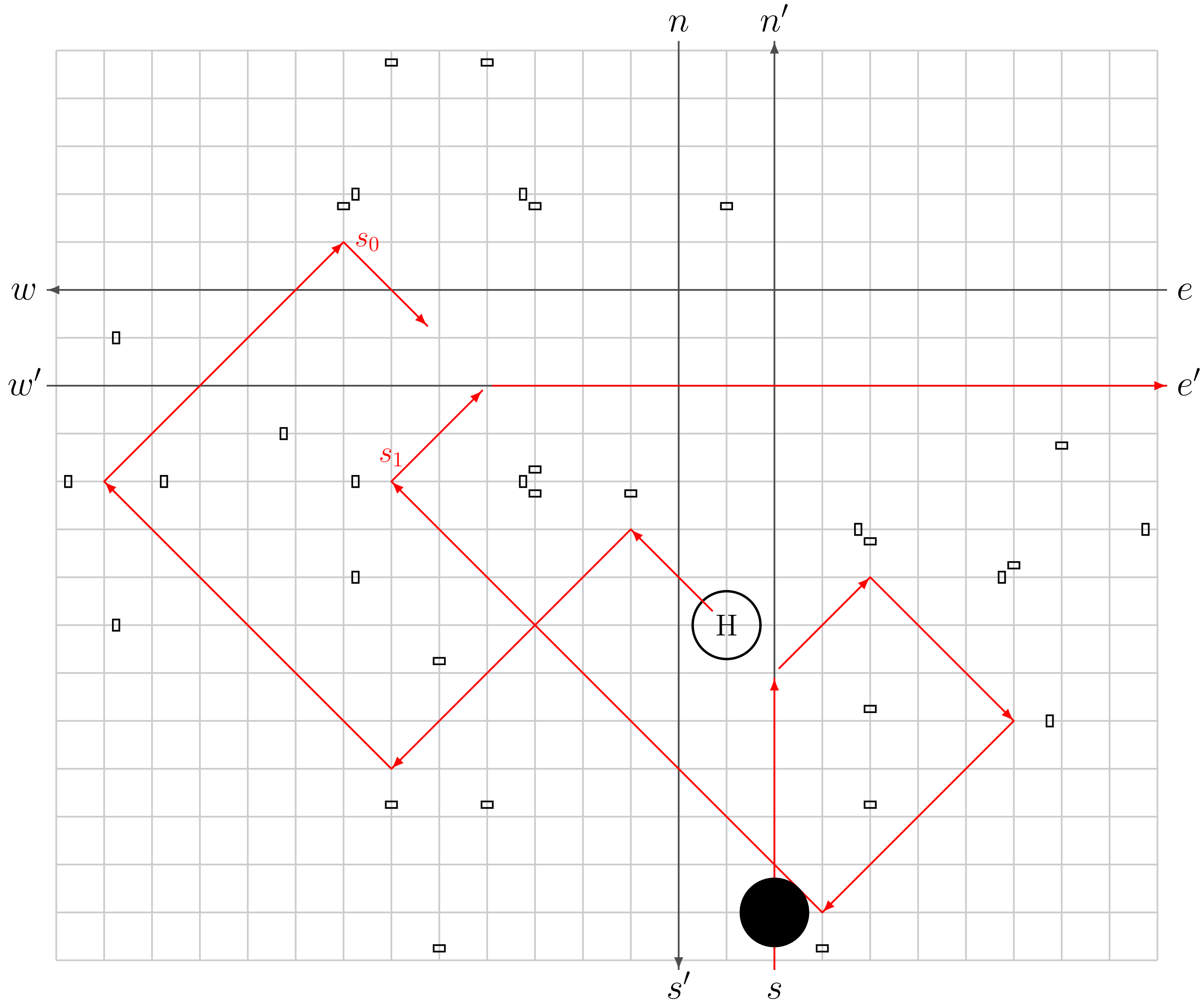
Movements of Balls (State: V, Input: s)



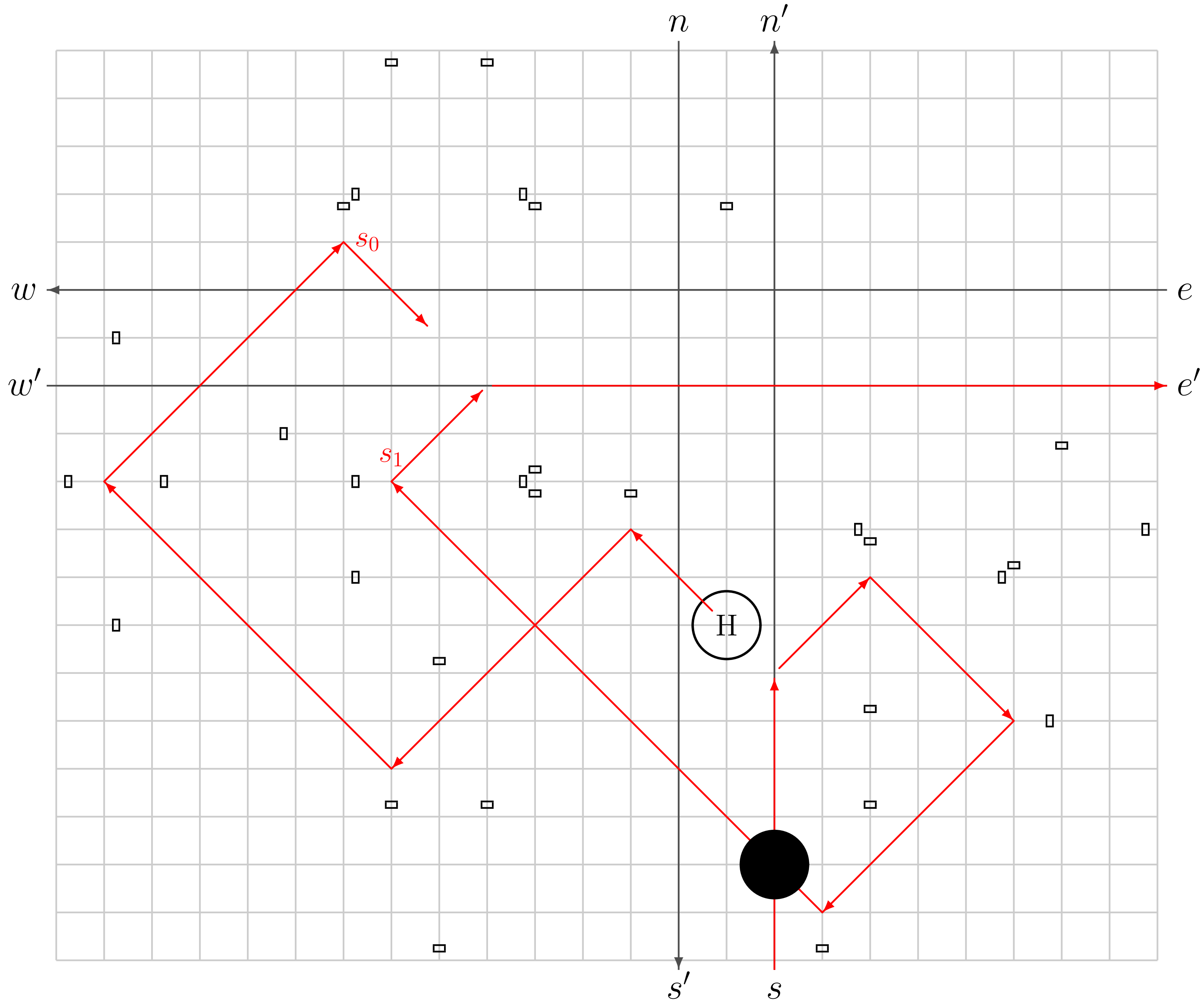
Orthogonal Case



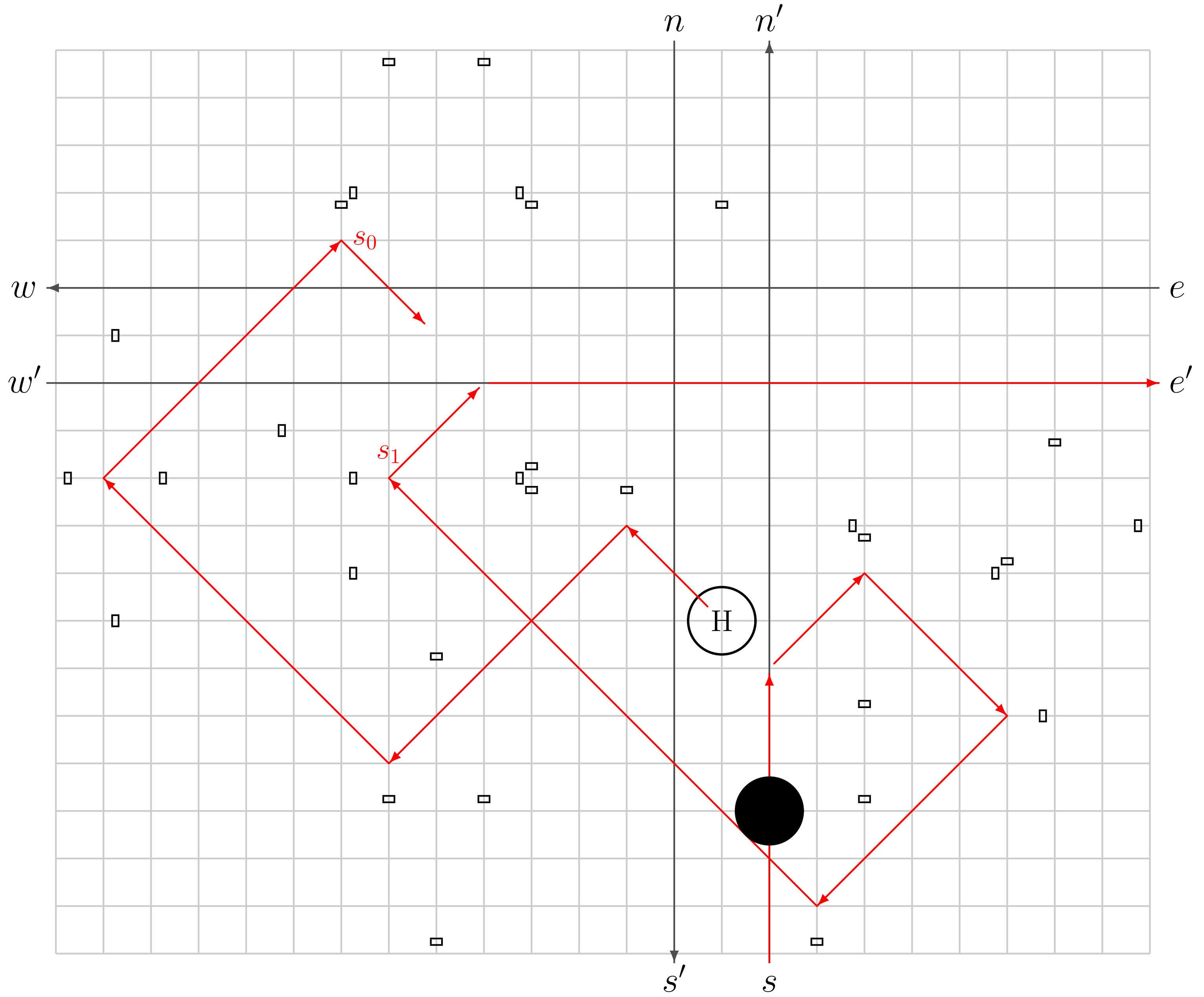
Movements of Balls (State: H, Input: s)



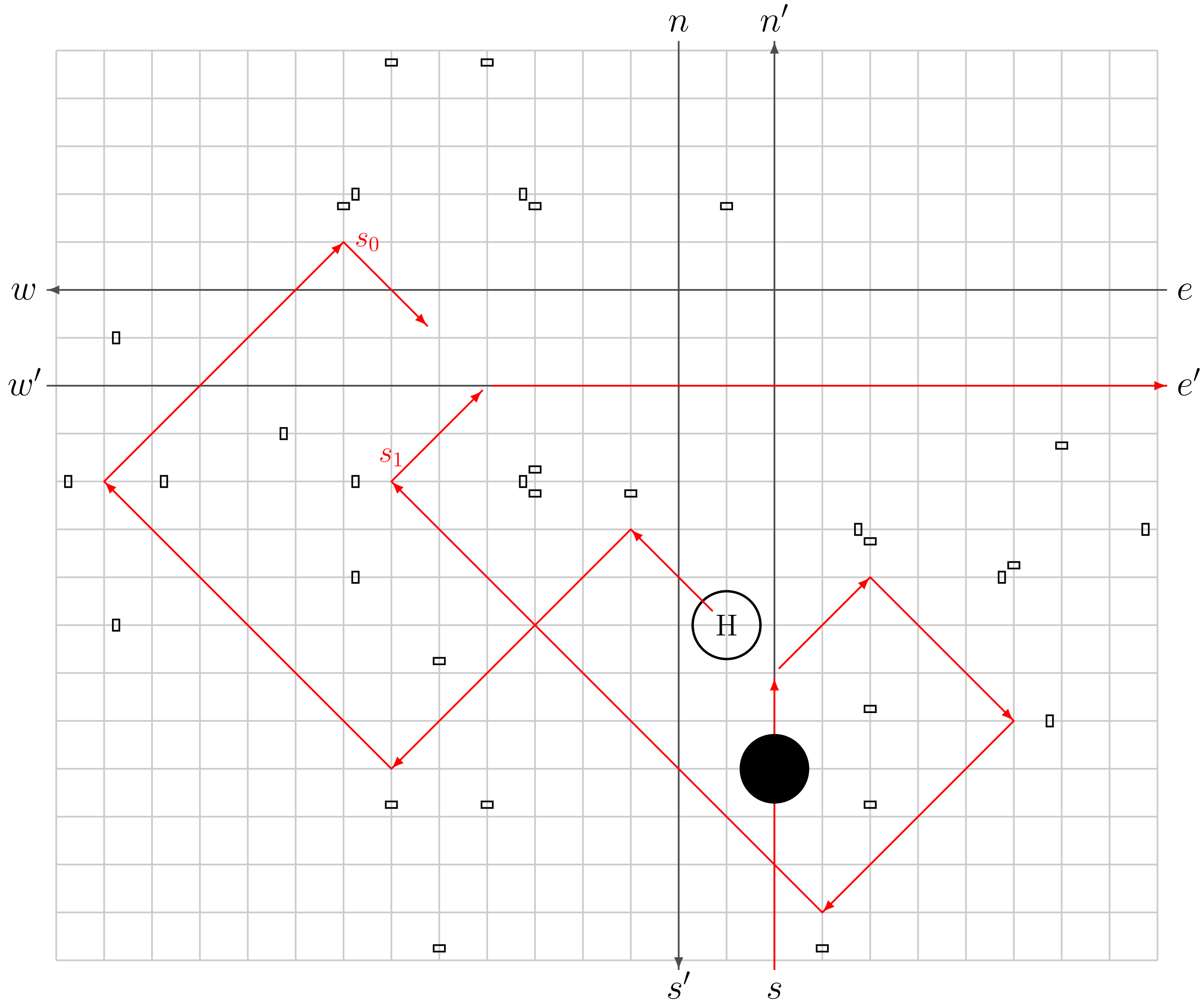
Movements of Balls (State: H, Input: s)



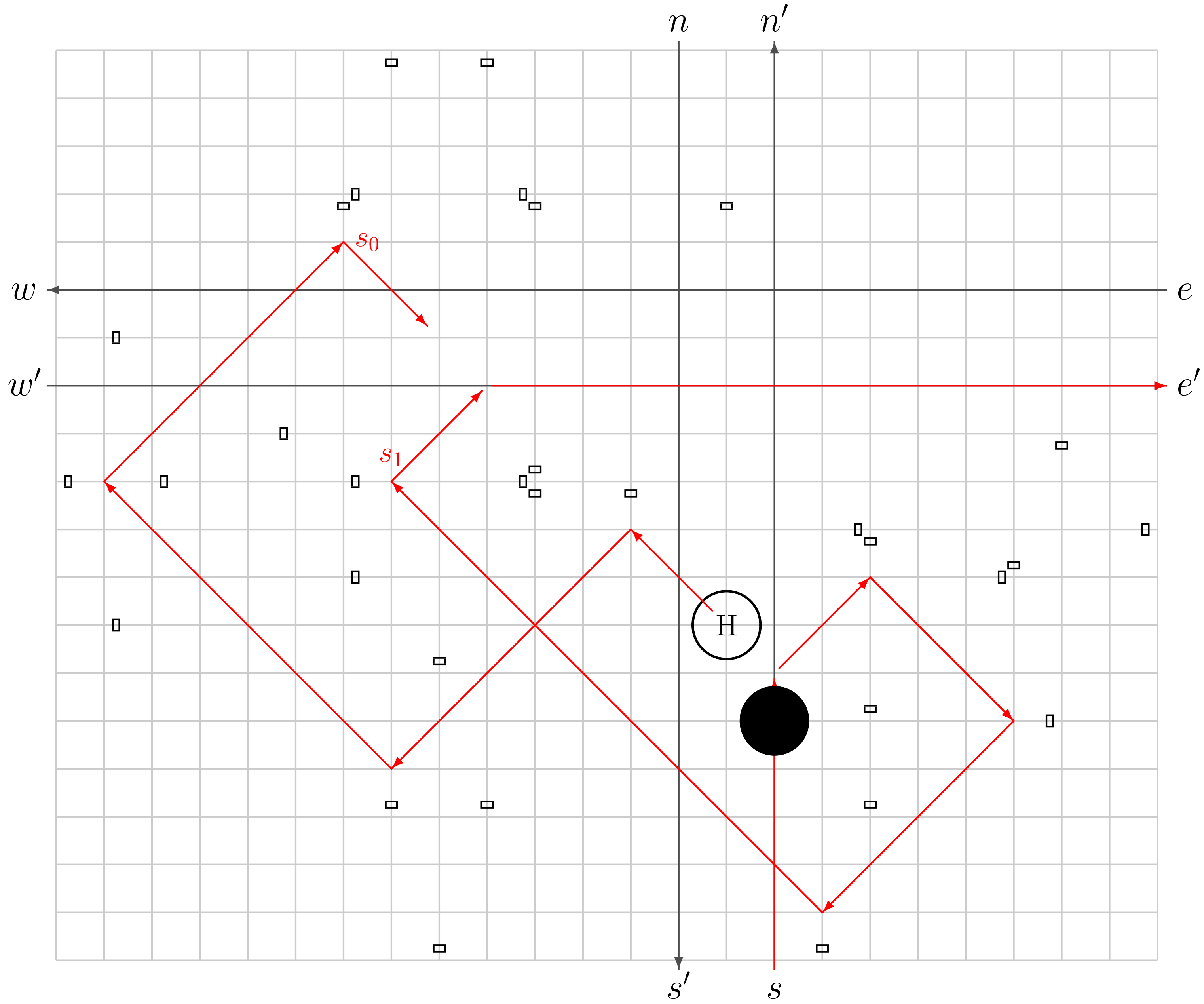
Movements of Balls (State: H, Input: s)



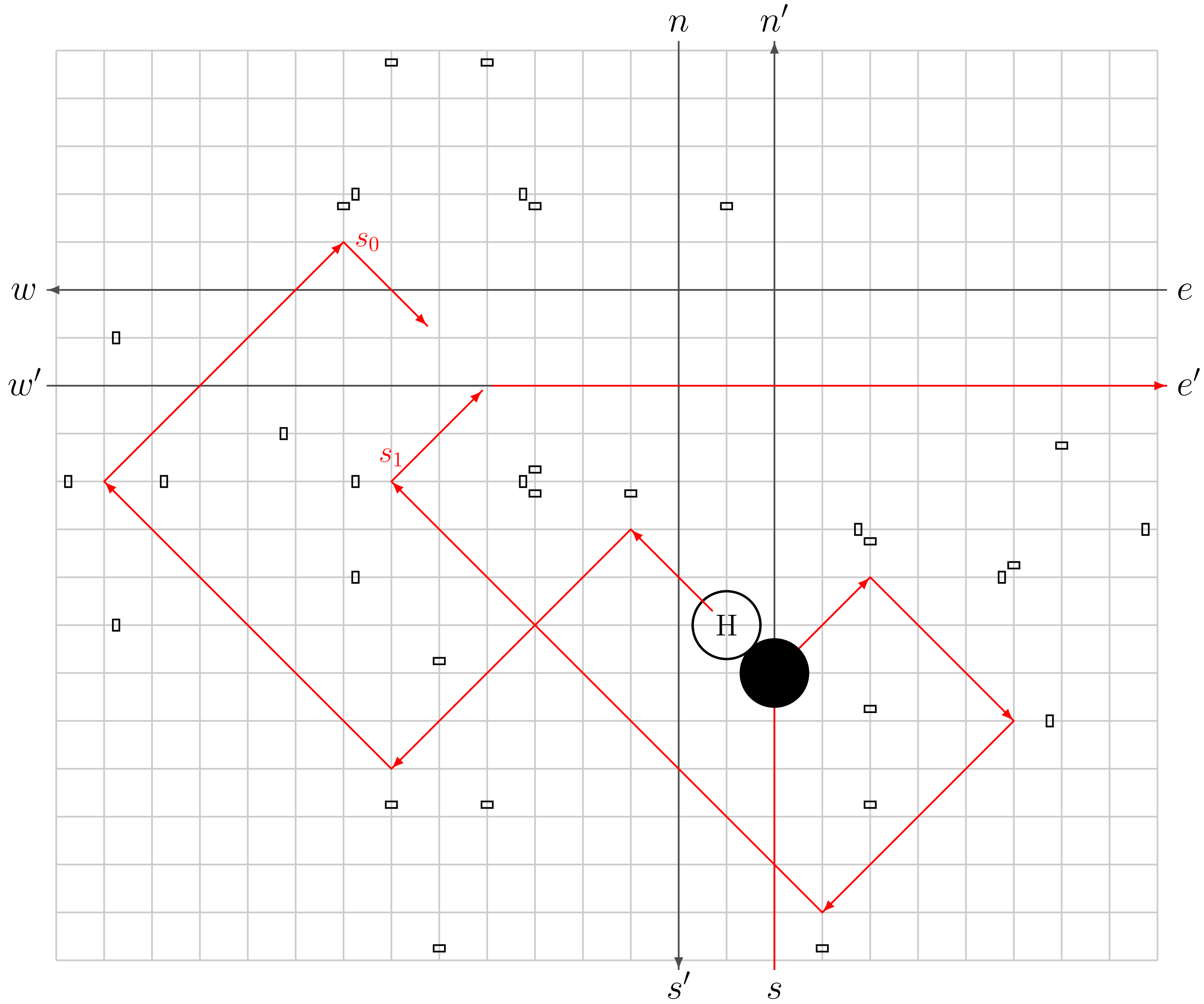
Movements of Balls (State: H, Input: s)



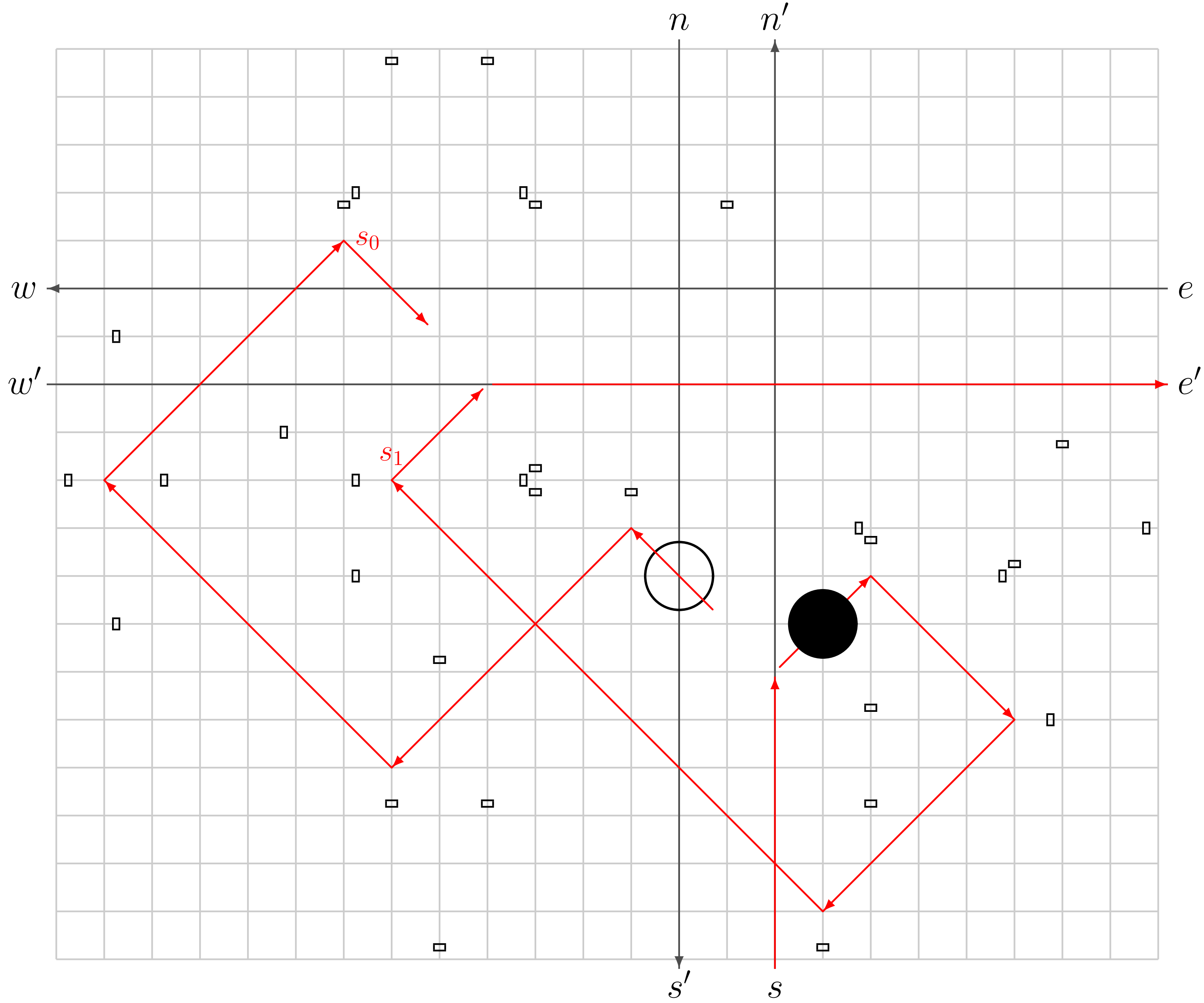
Movements of Balls (State: H, Input: s)



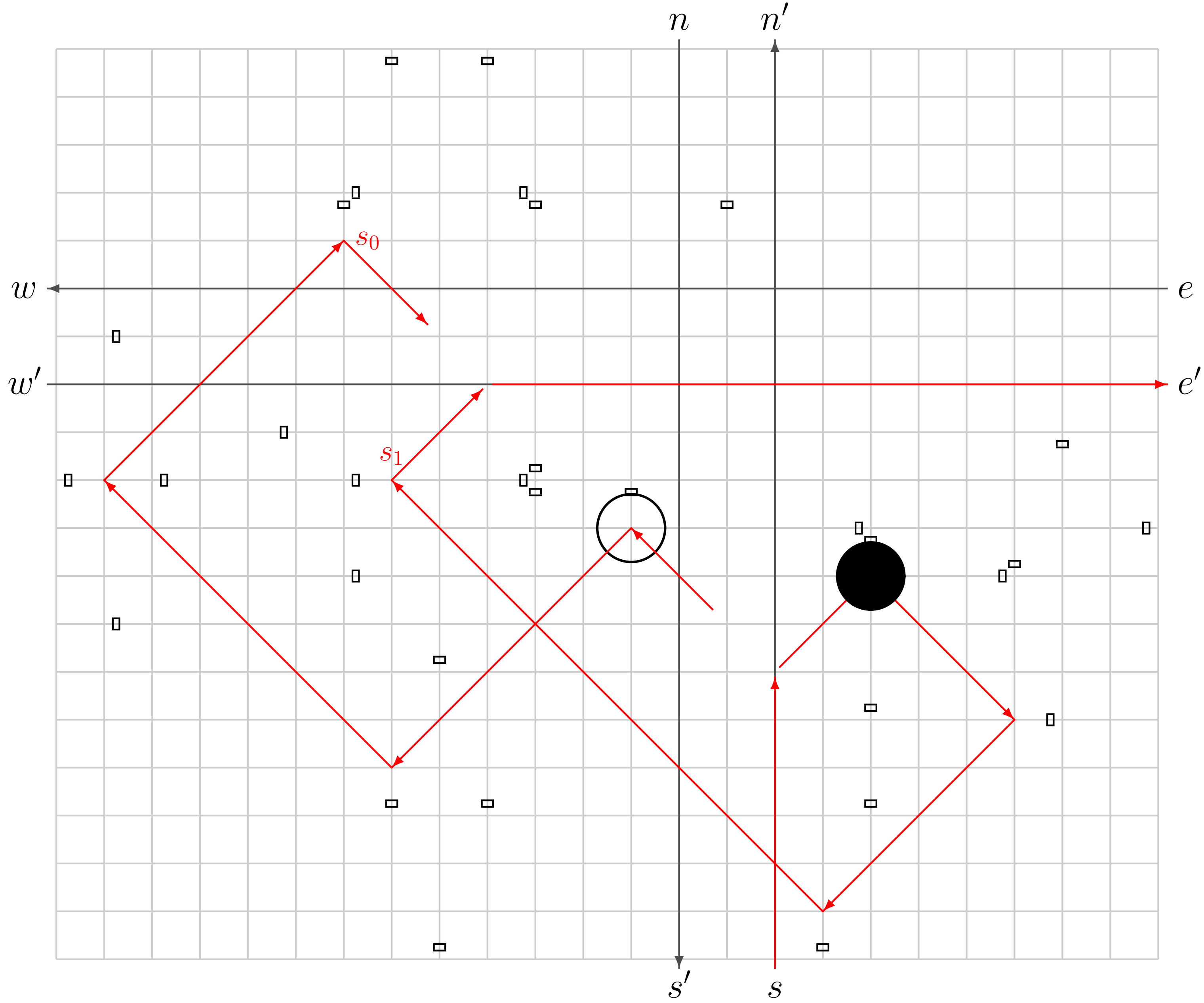
Movements of Balls (State: H, Input: s)



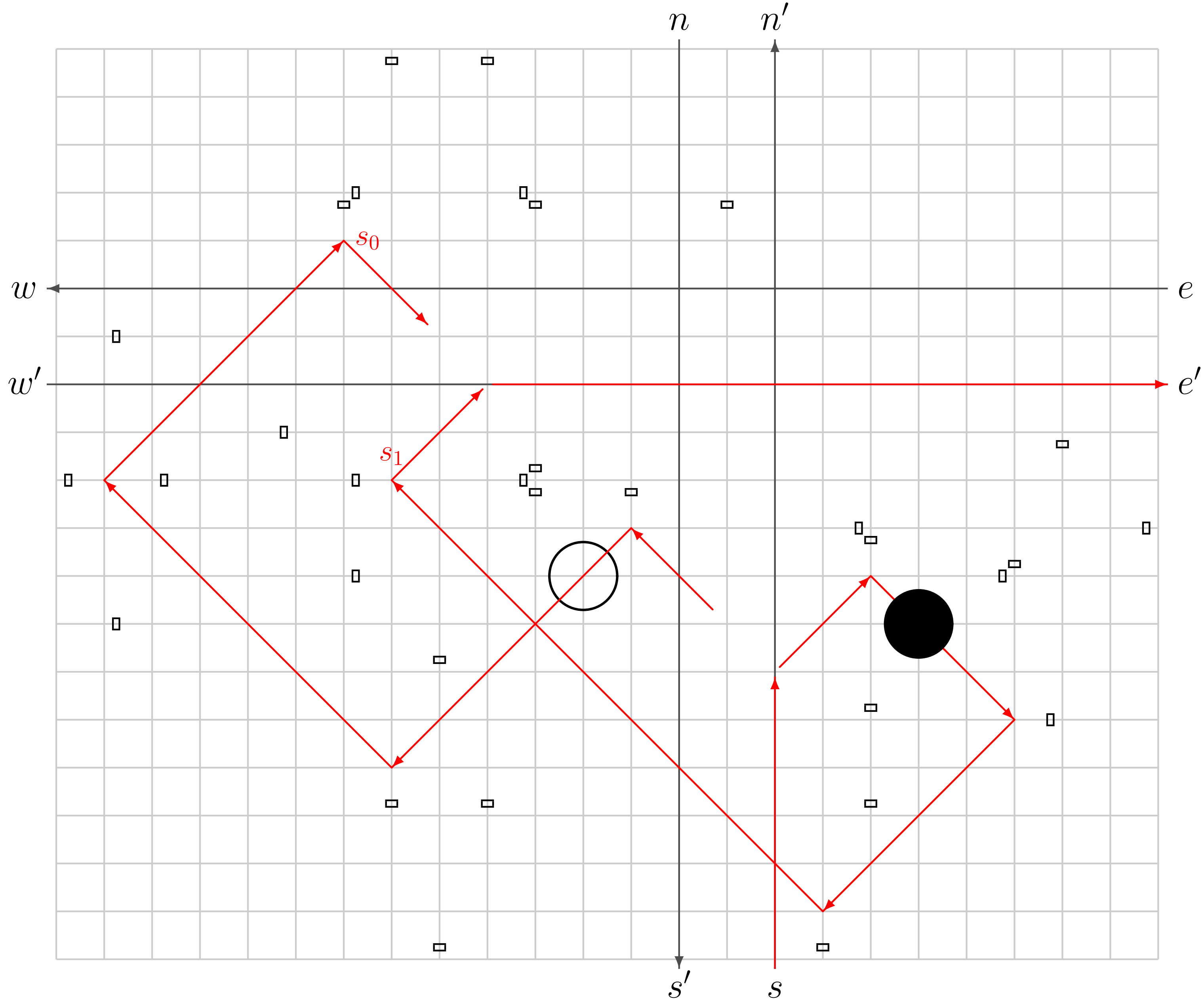
Movements of Balls (State: H, Input: s)



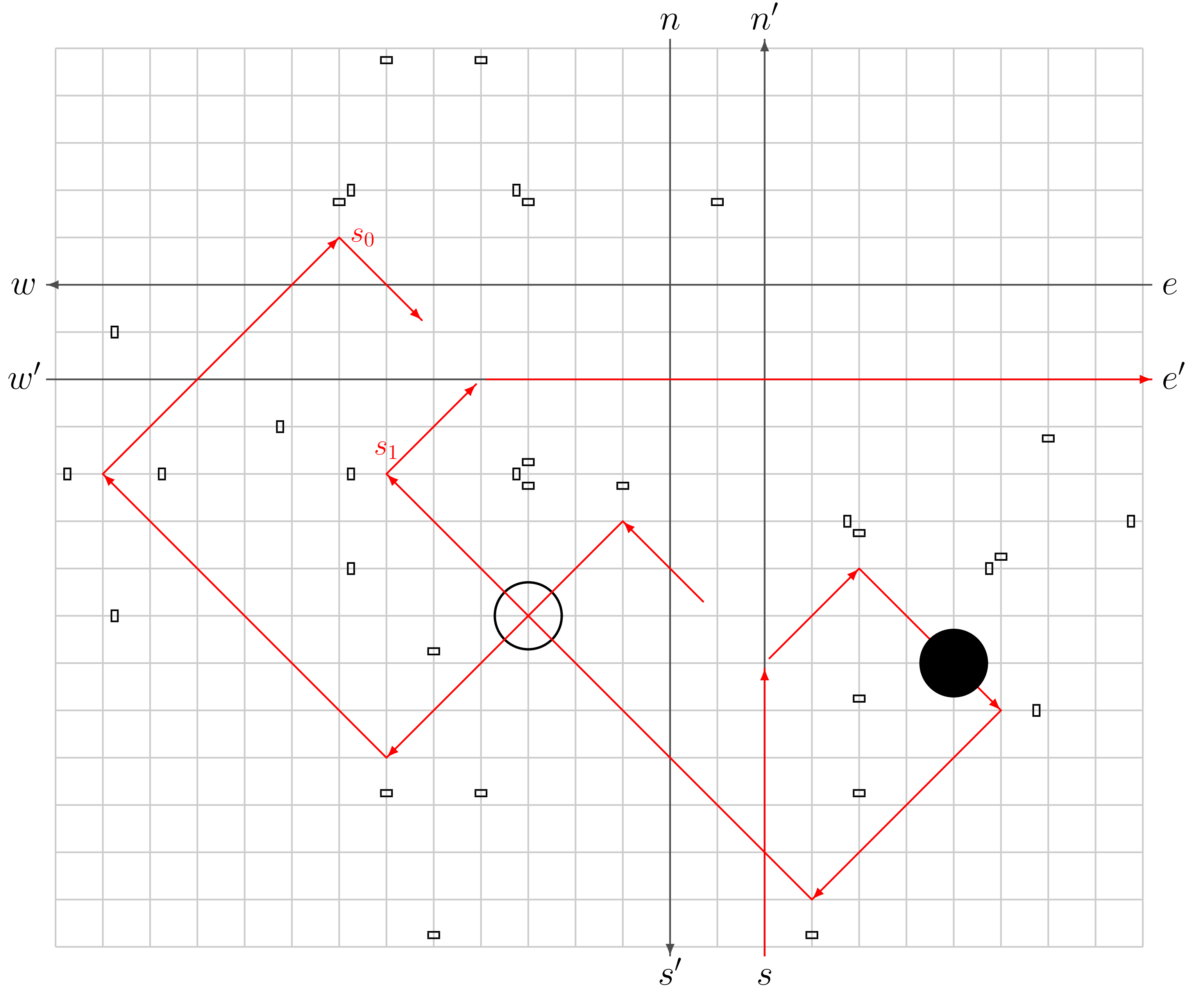
Movements of Balls (State: H, Input: s)



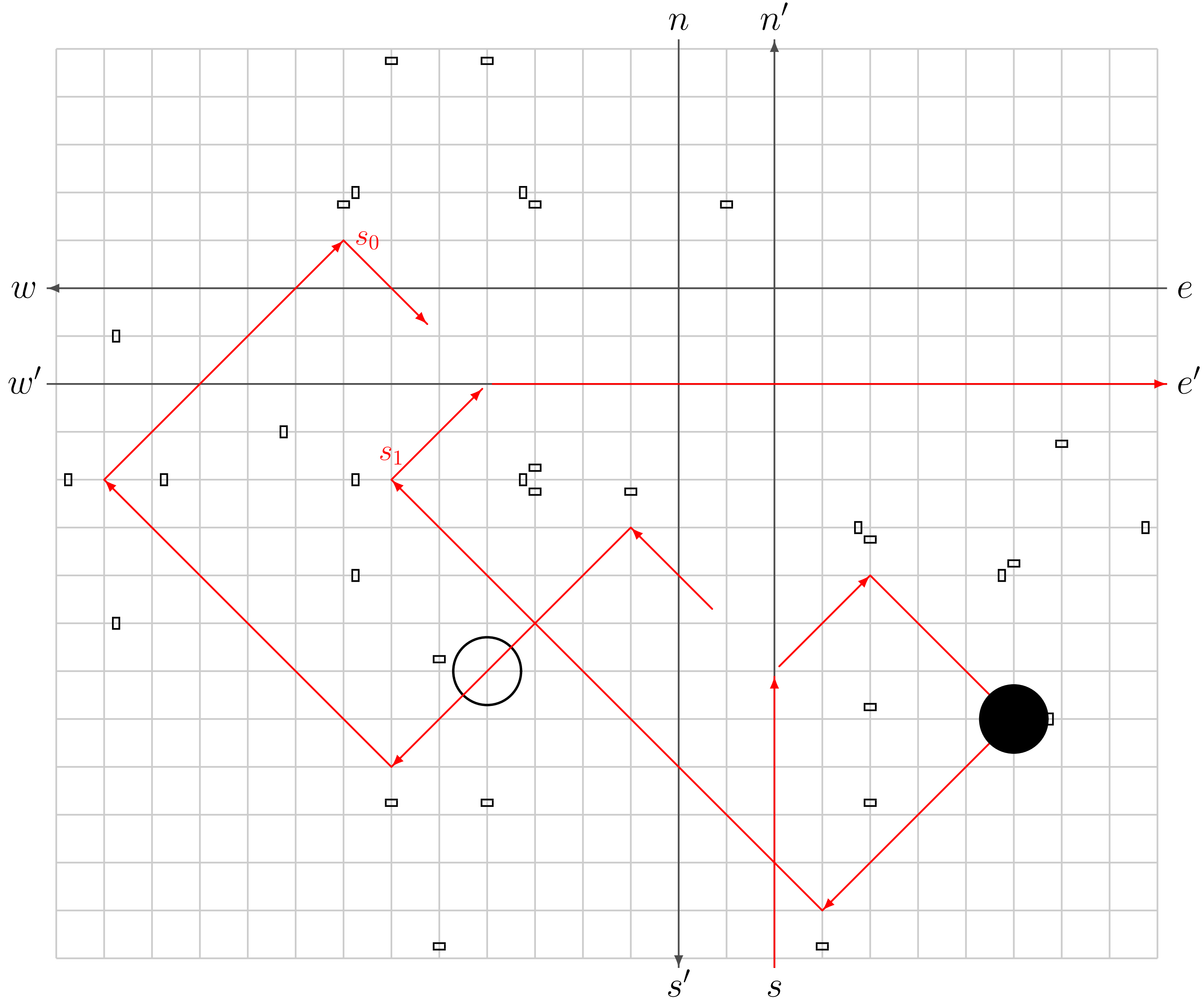
Movements of Balls (State: H, Input: s)



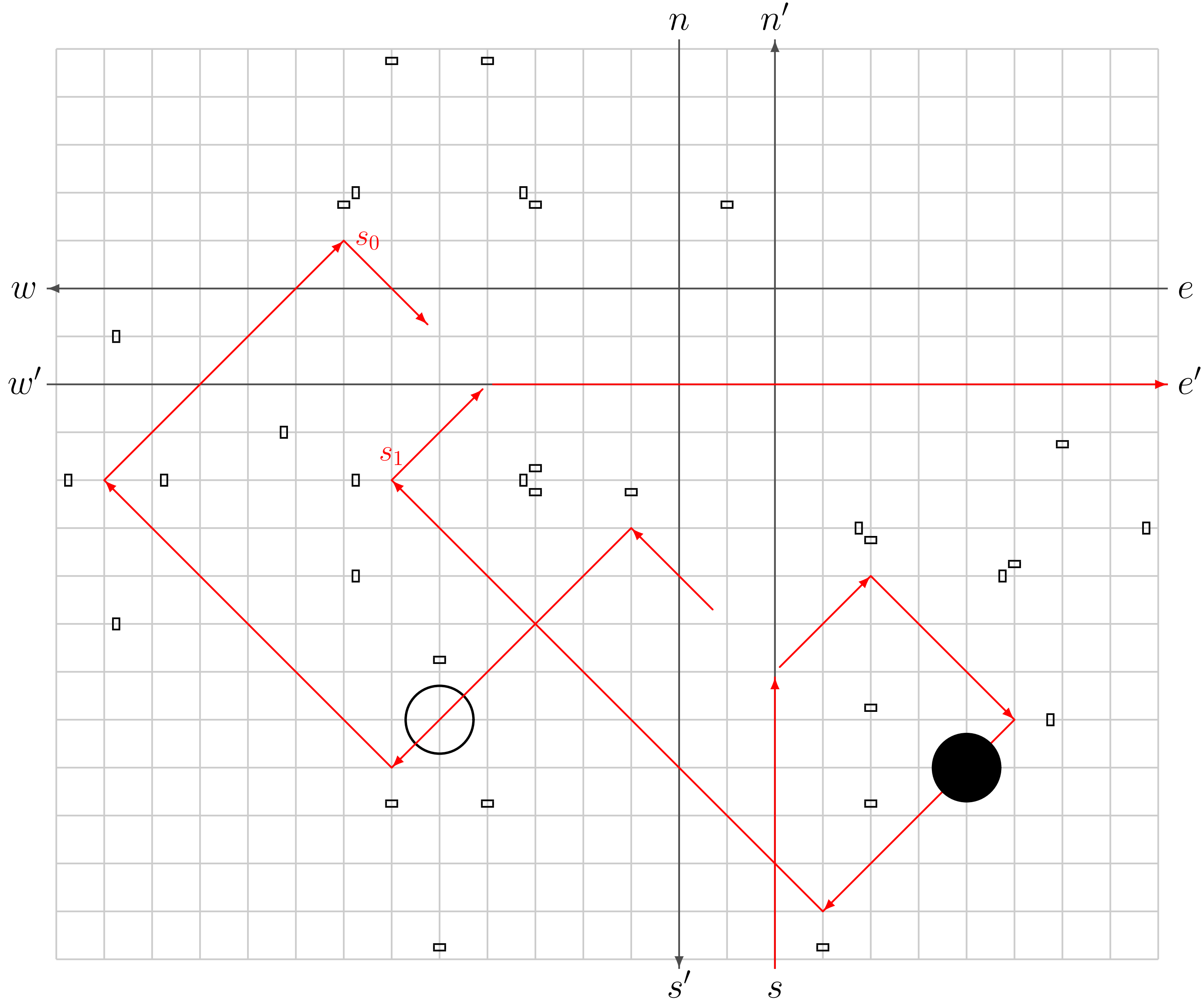
Movements of Balls (State: H, Input: s)



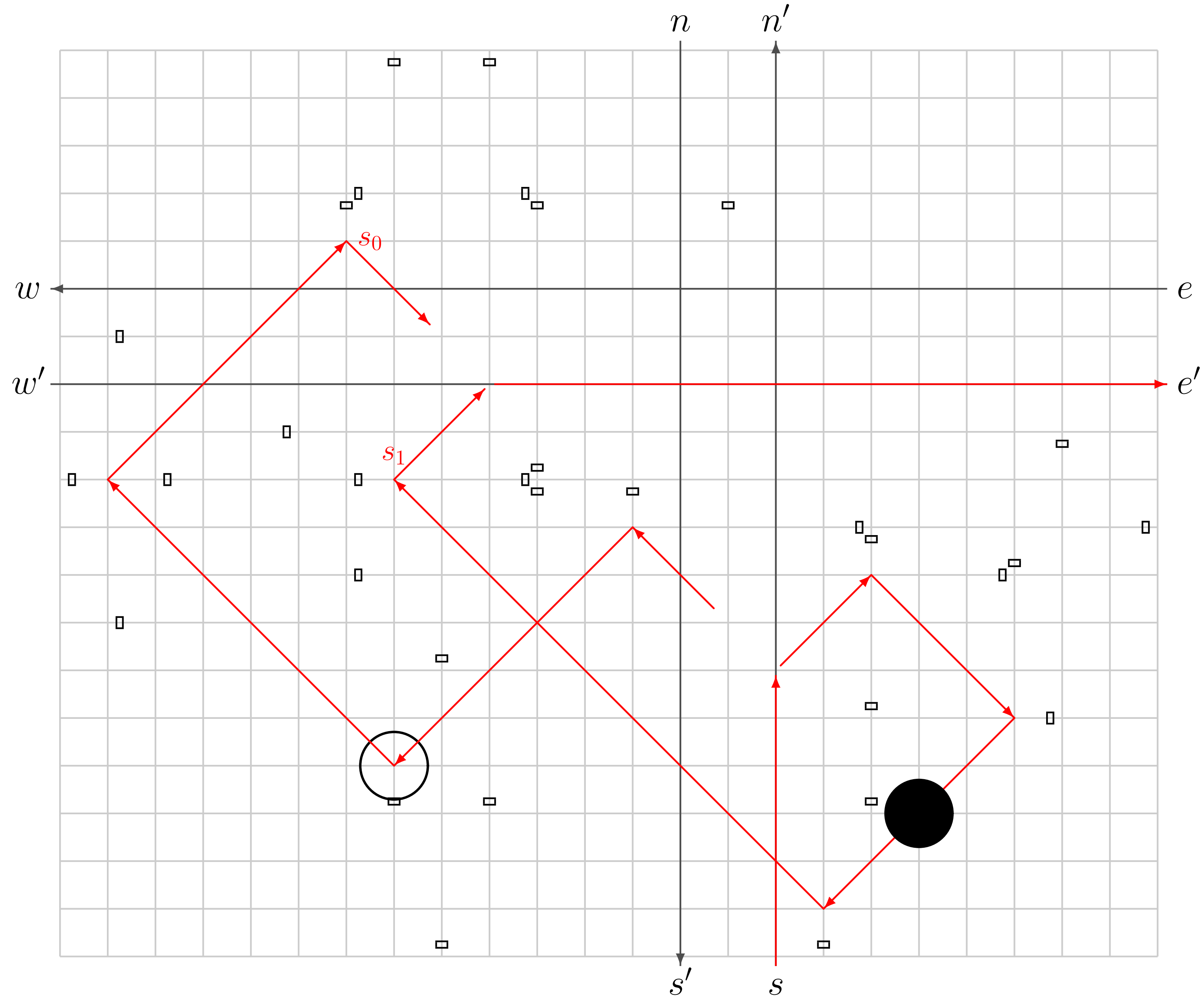
Movements of Balls (State: H, Input: s)



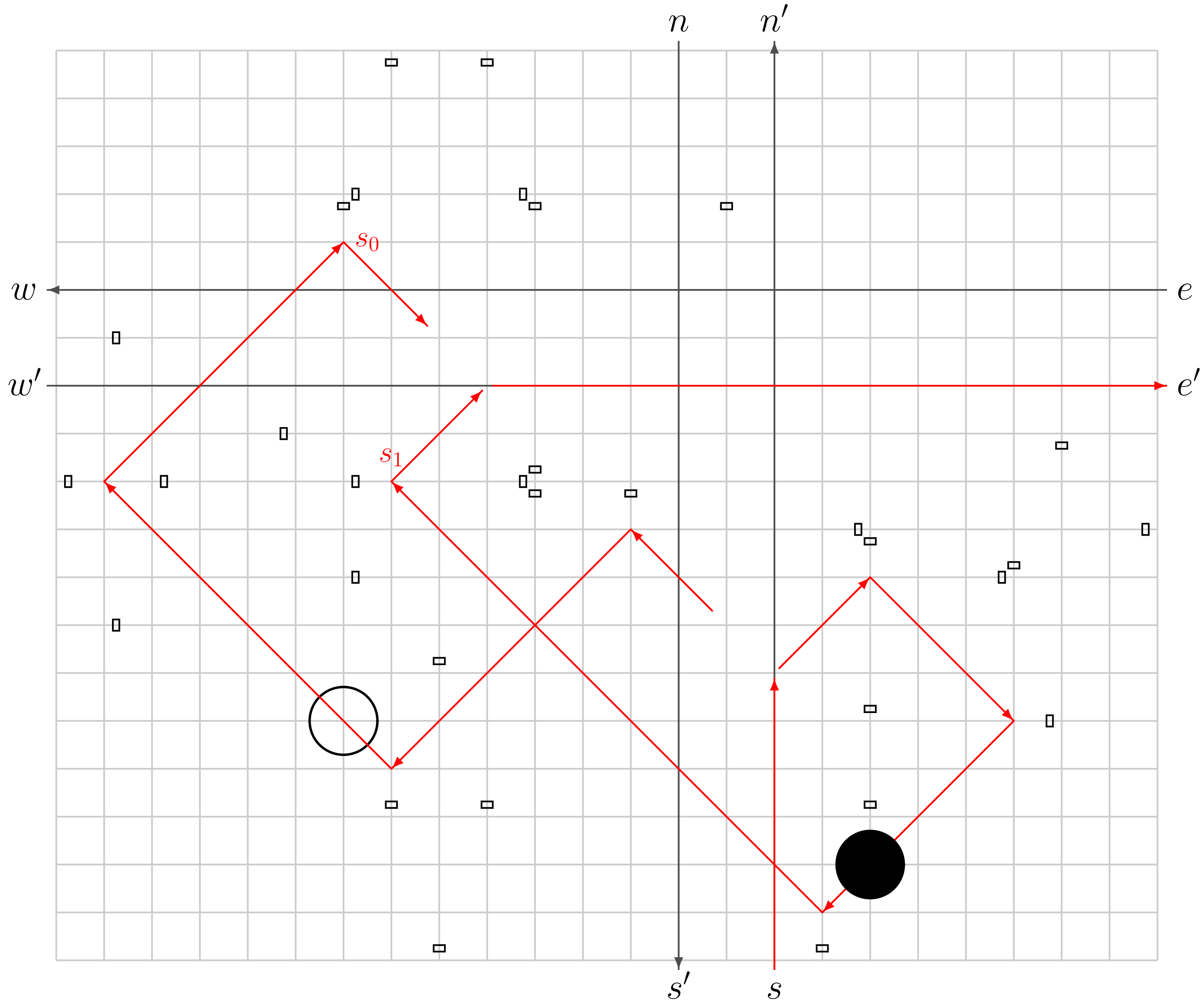
Movements of Balls (State: H, Input: s)



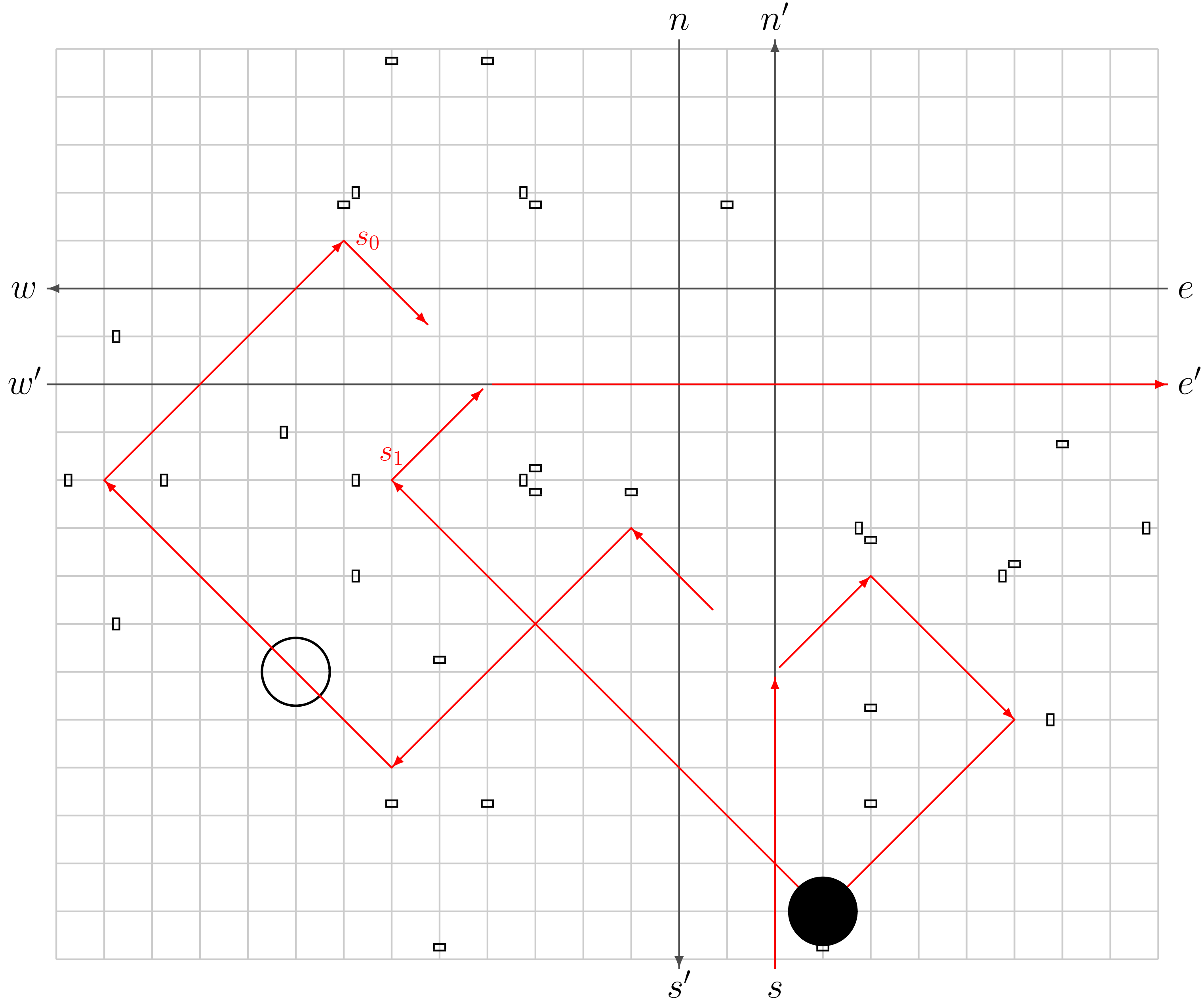
Movements of Balls (State: H, Input: s)



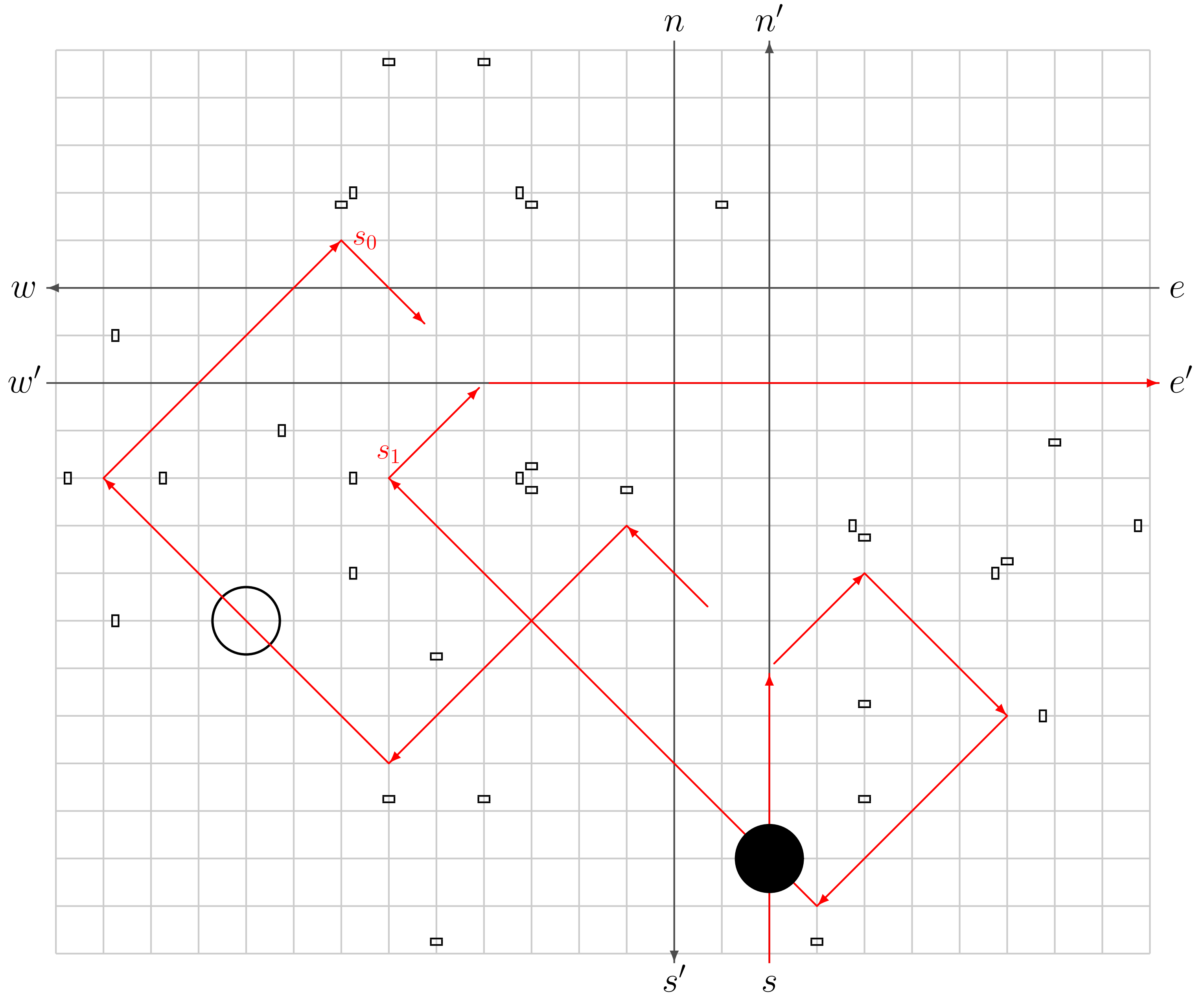
Movements of Balls (State: H, Input: s)



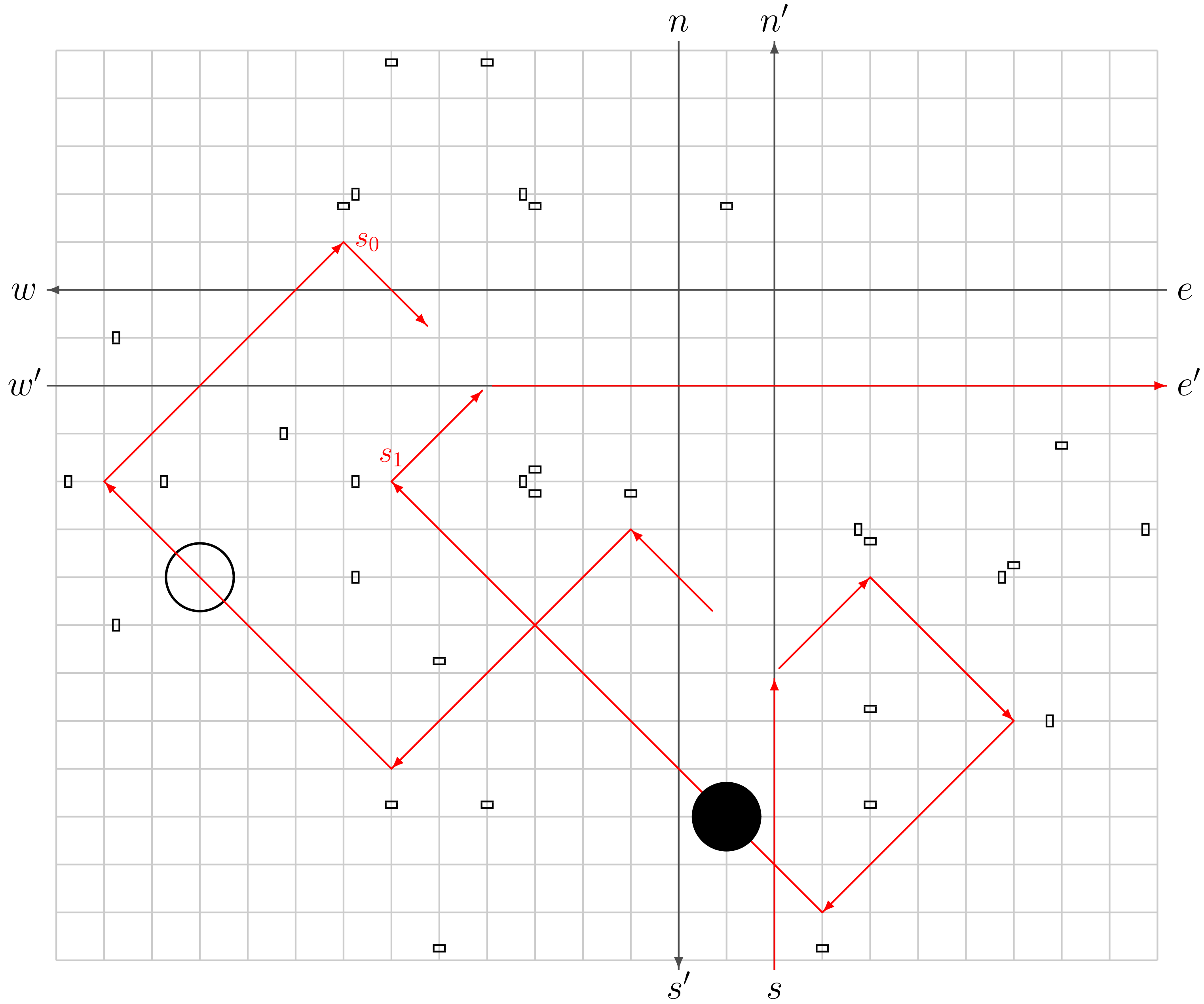
Movements of Balls (State: H, Input: s)



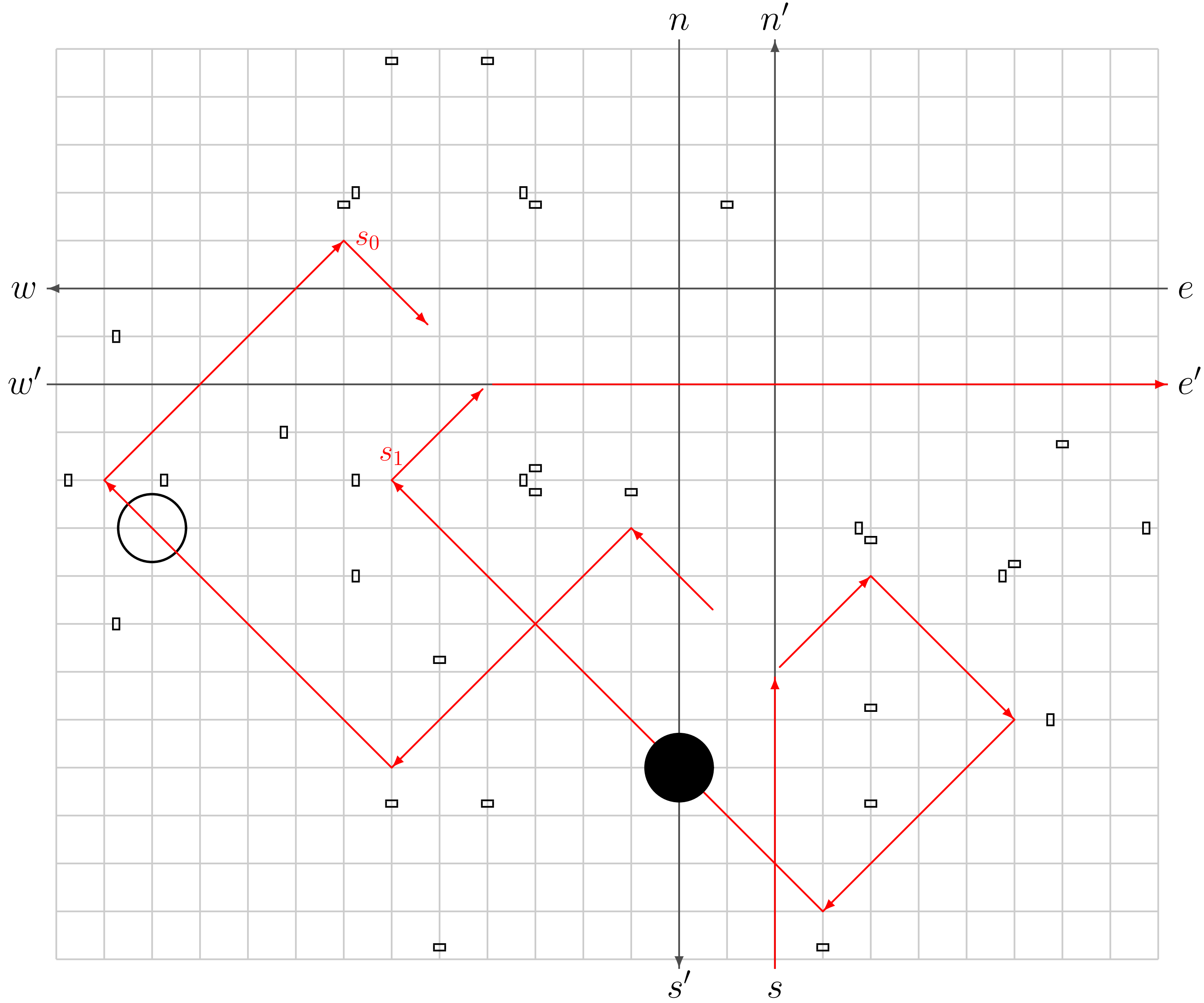
Movements of Balls (State: H, Input: s)



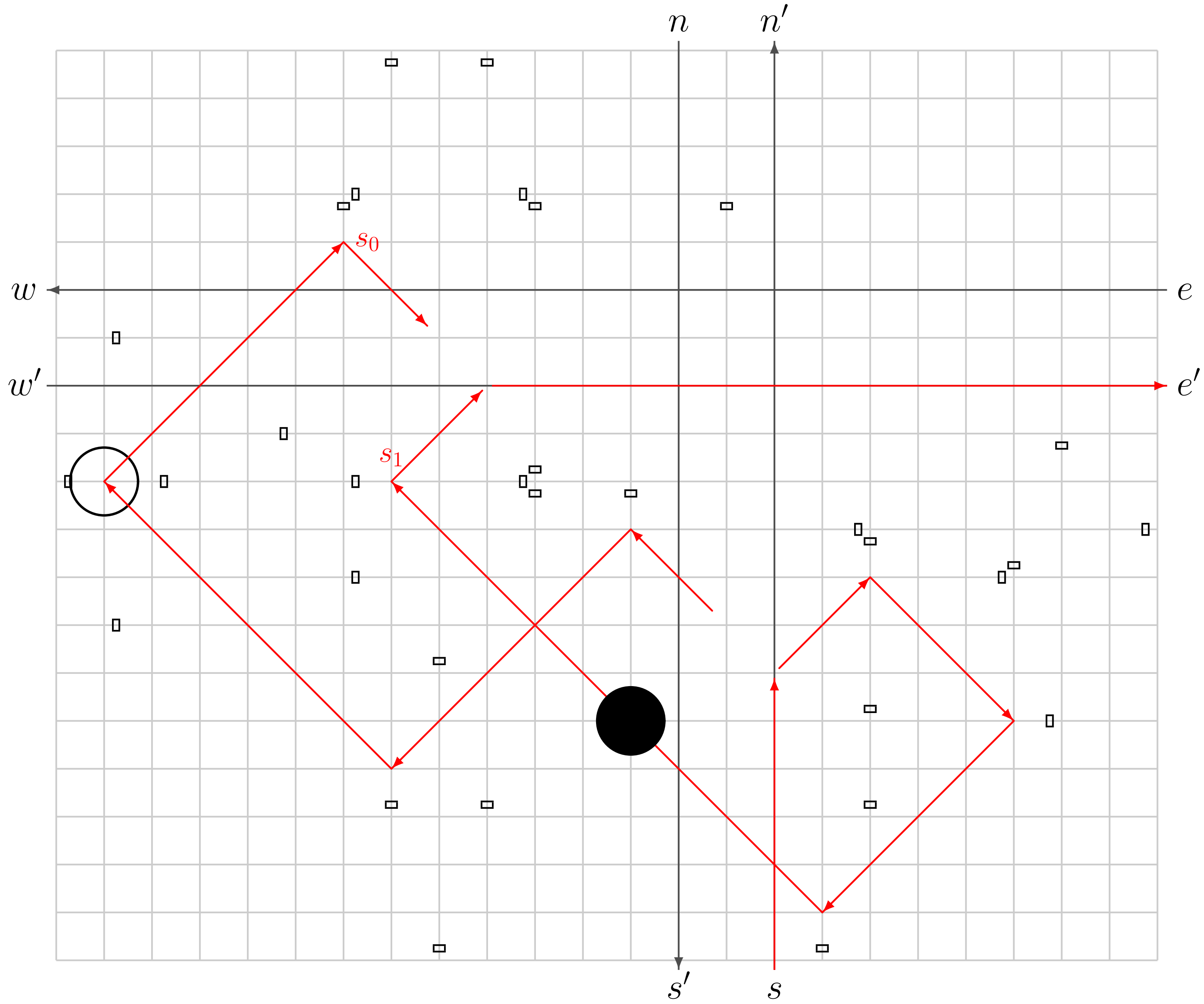
Movements of Balls (State: H, Input: s)



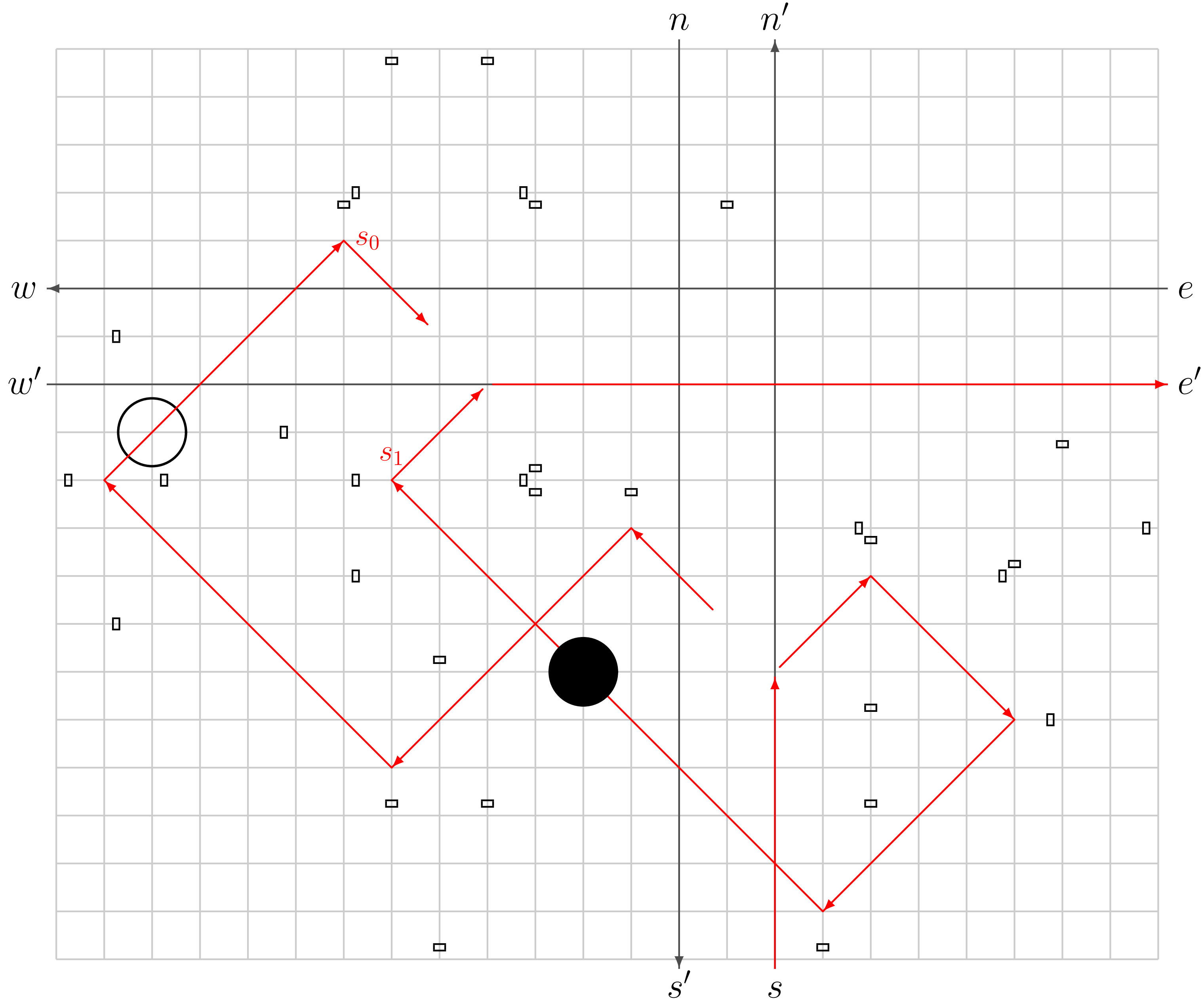
Movements of Balls (State: H, Input: s)



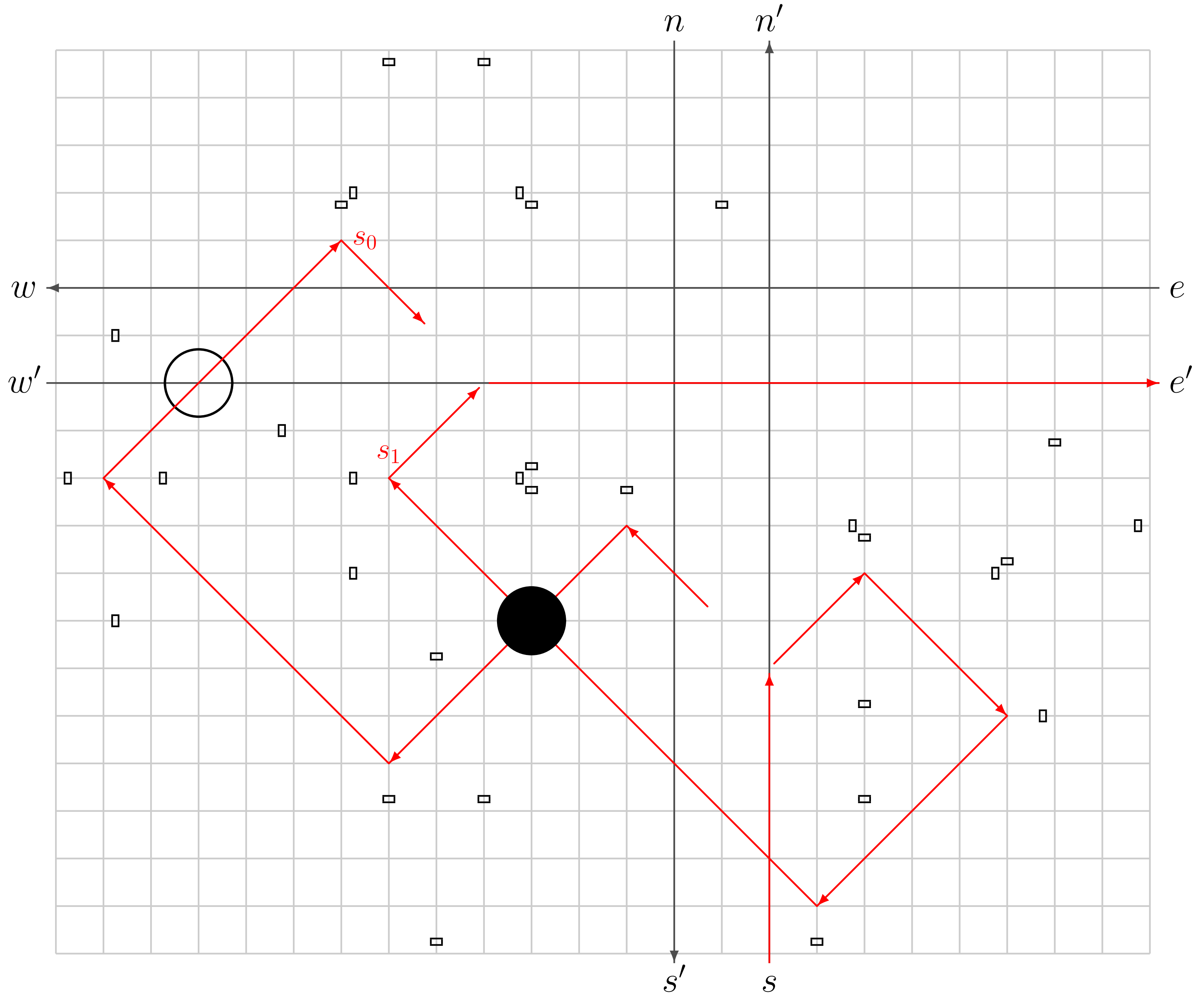
Movements of Balls (State: H, Input: s)



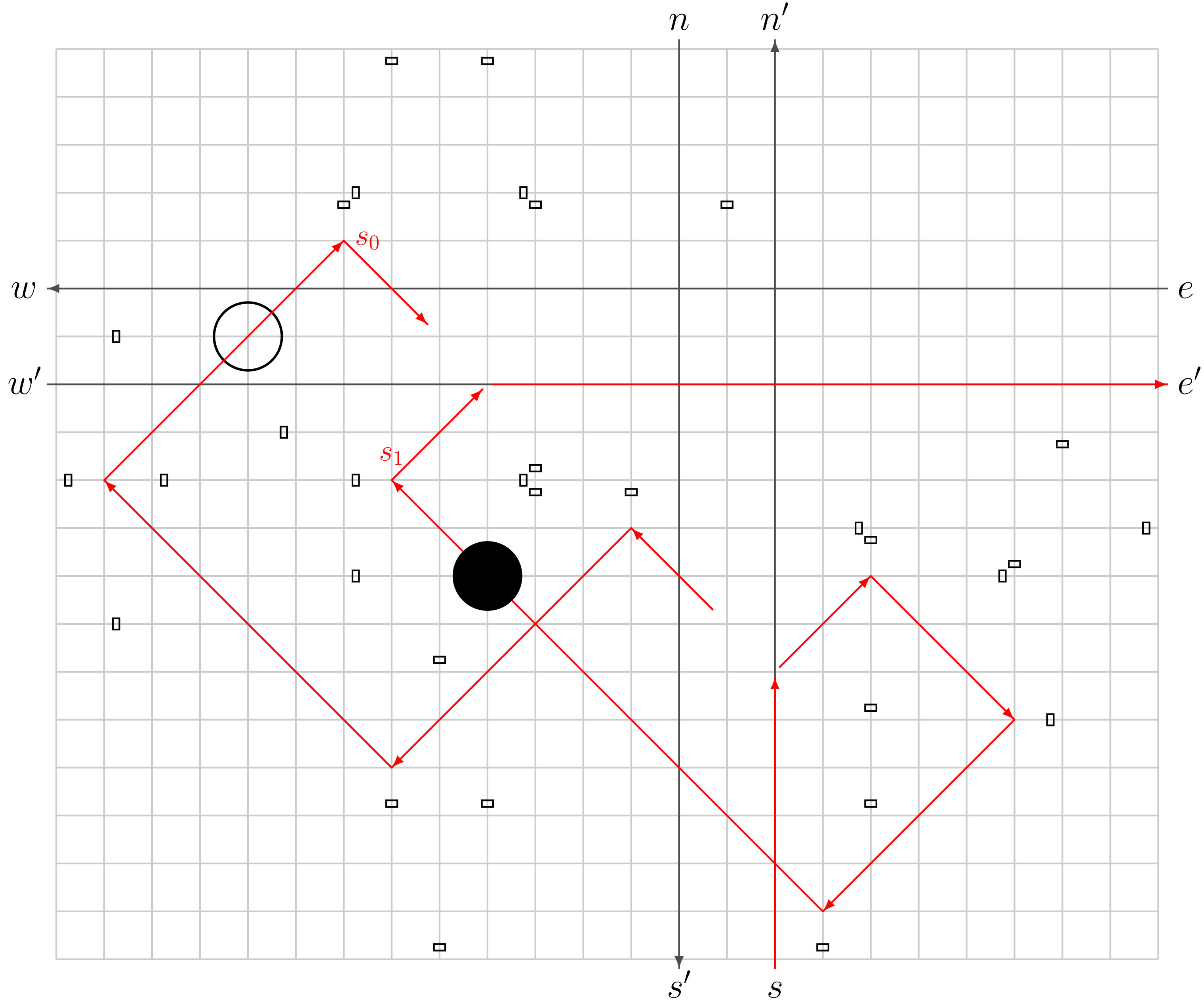
Movements of Balls (State: H, Input: s)



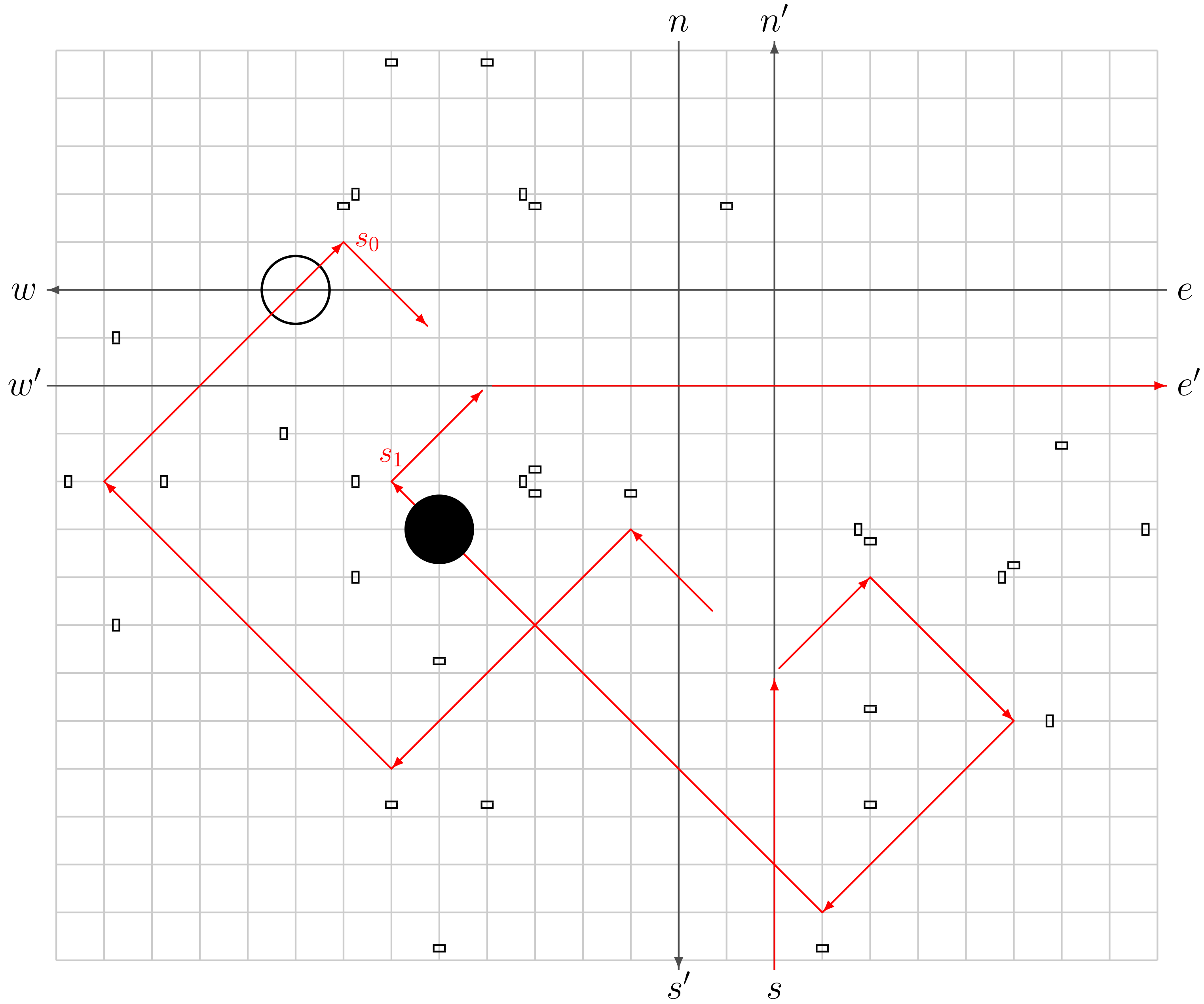
Movements of Balls (State: H, Input: s)



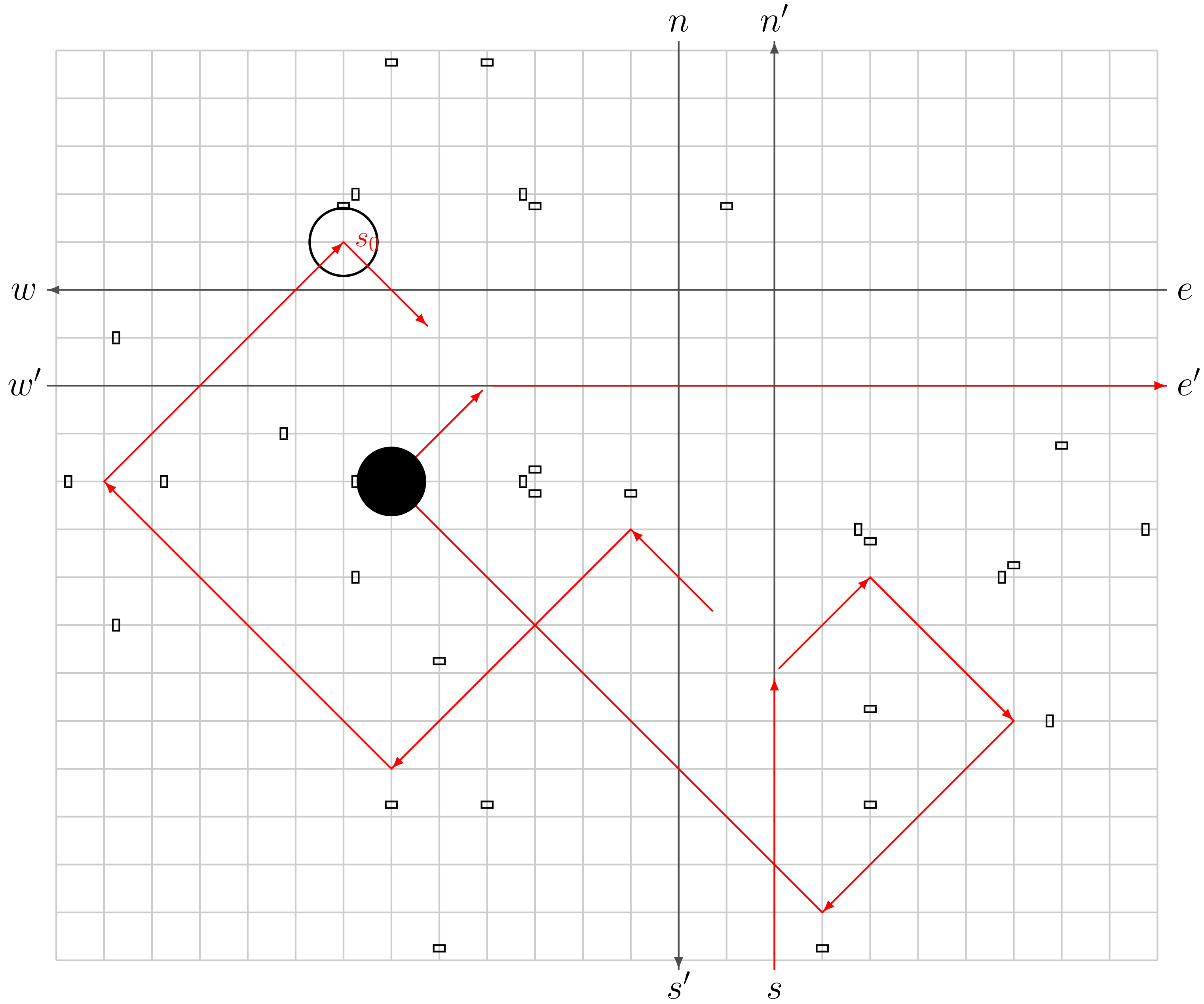
Movements of Balls (State: H, Input: s)



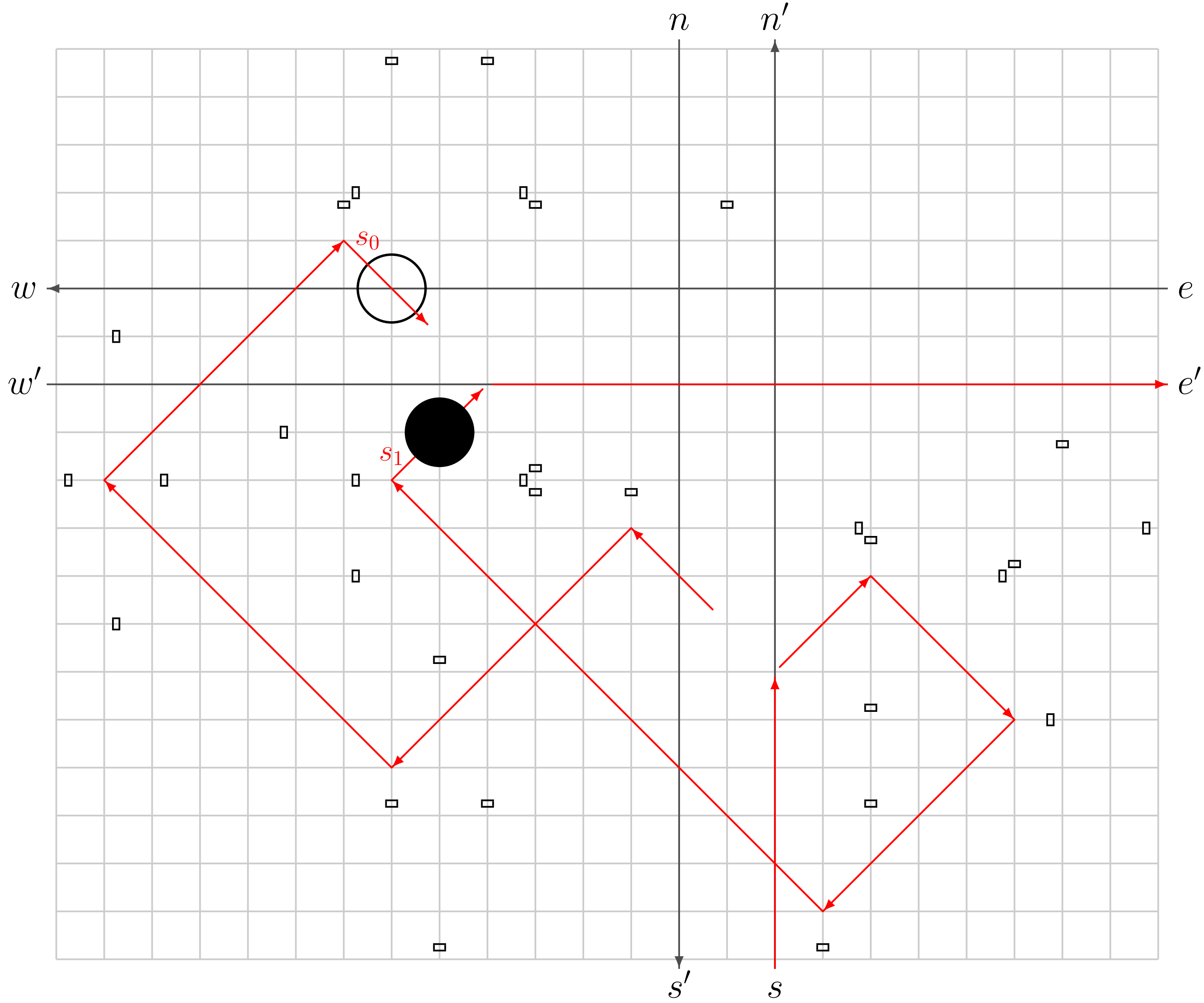
Movements of Balls (State: H, Input: s)



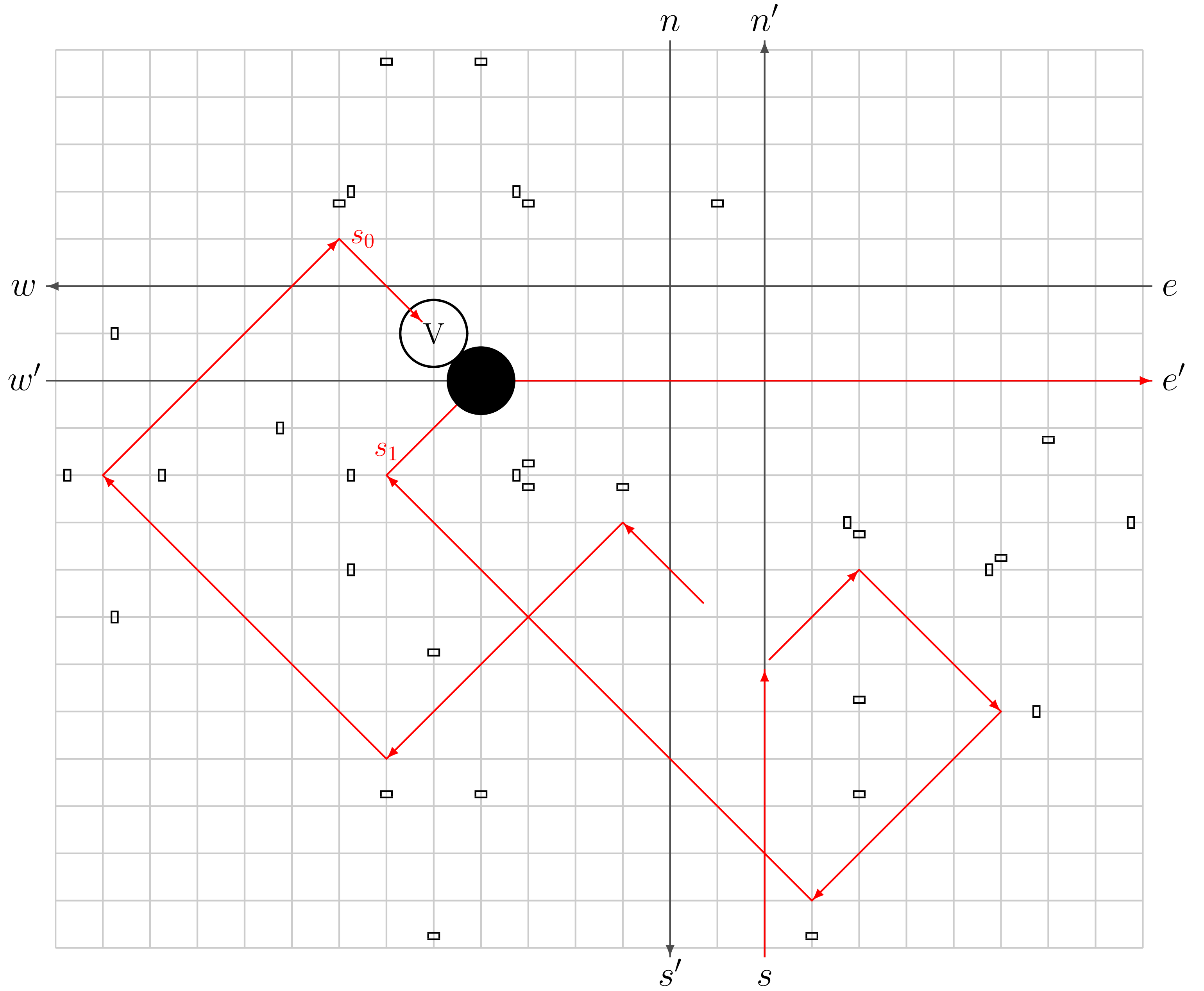
Movements of Balls (State: H, Input: s)



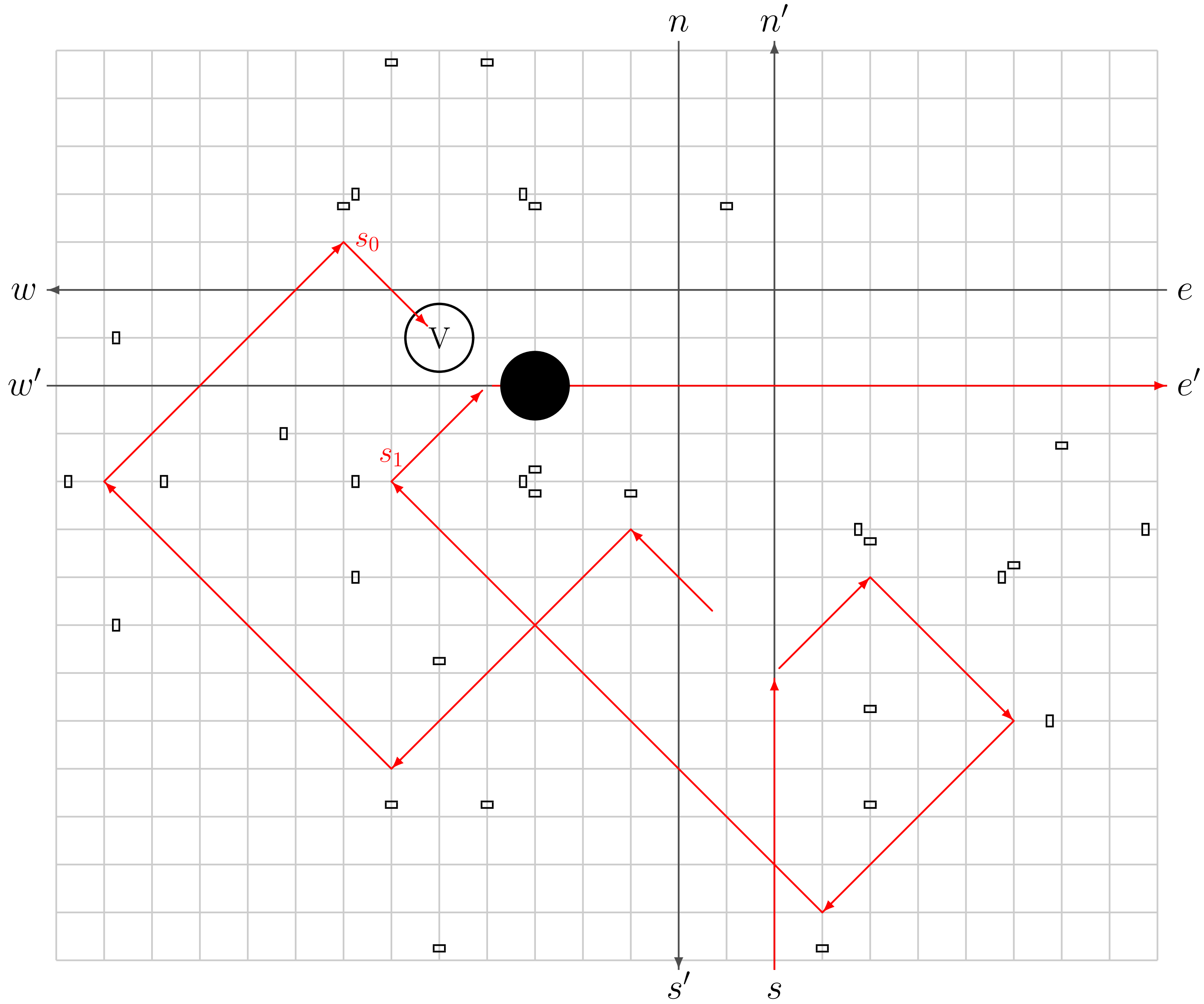
Movements of Balls (State: H, Input: s)



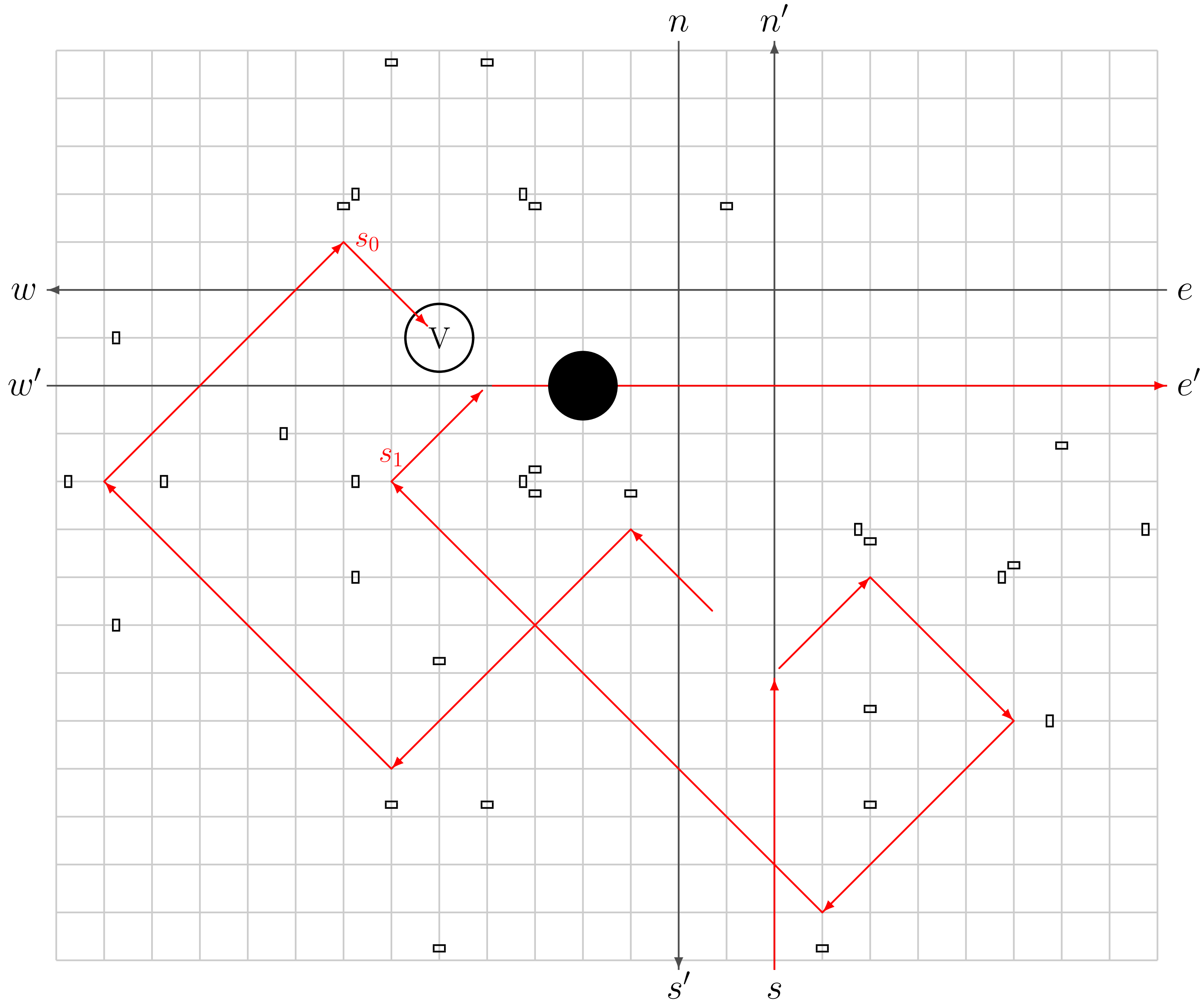
Movements of Balls (State: H, Input: s)



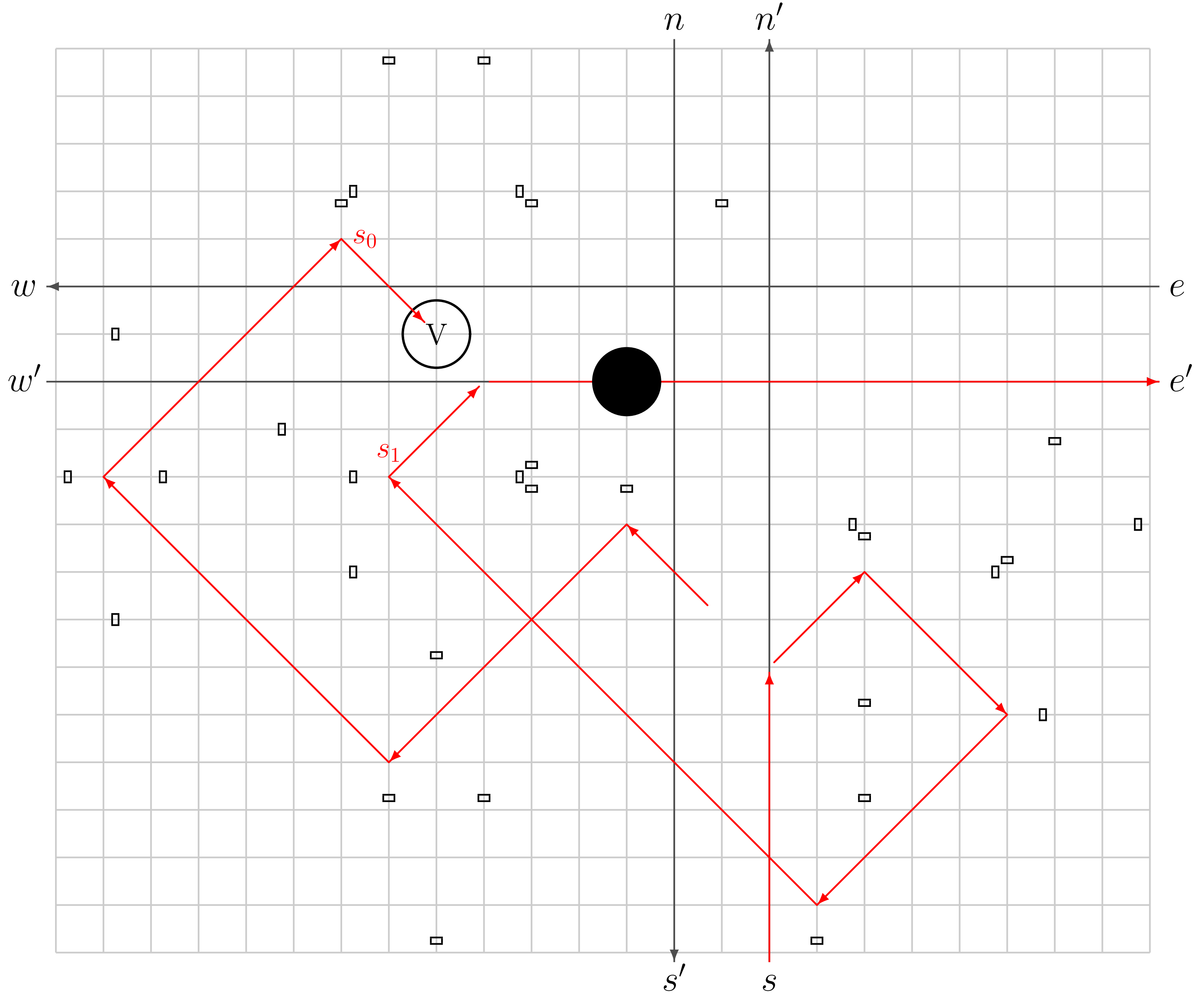
Movements of Balls (State: H, Input: s)



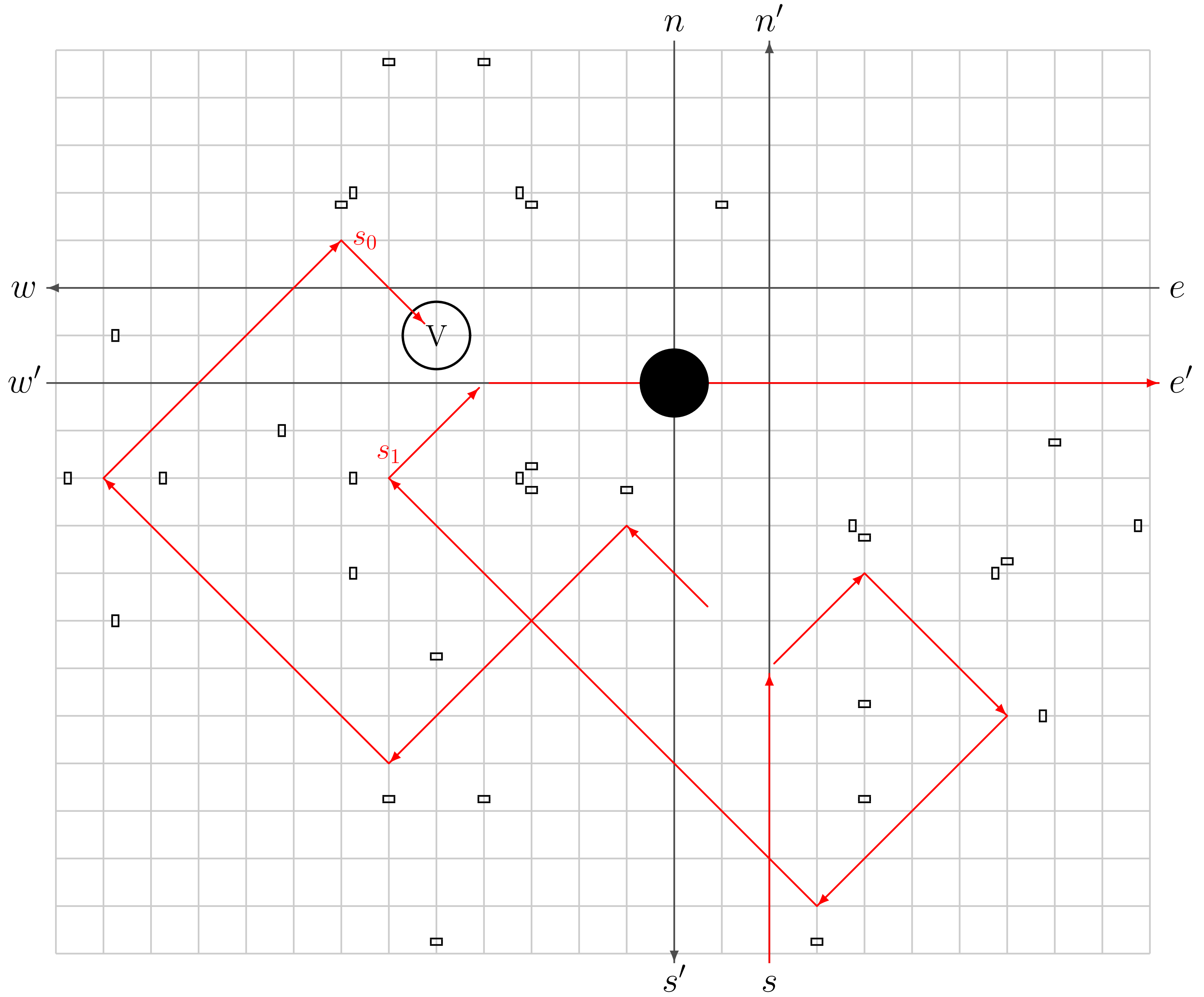
Movements of Balls (State: H, Input: s)



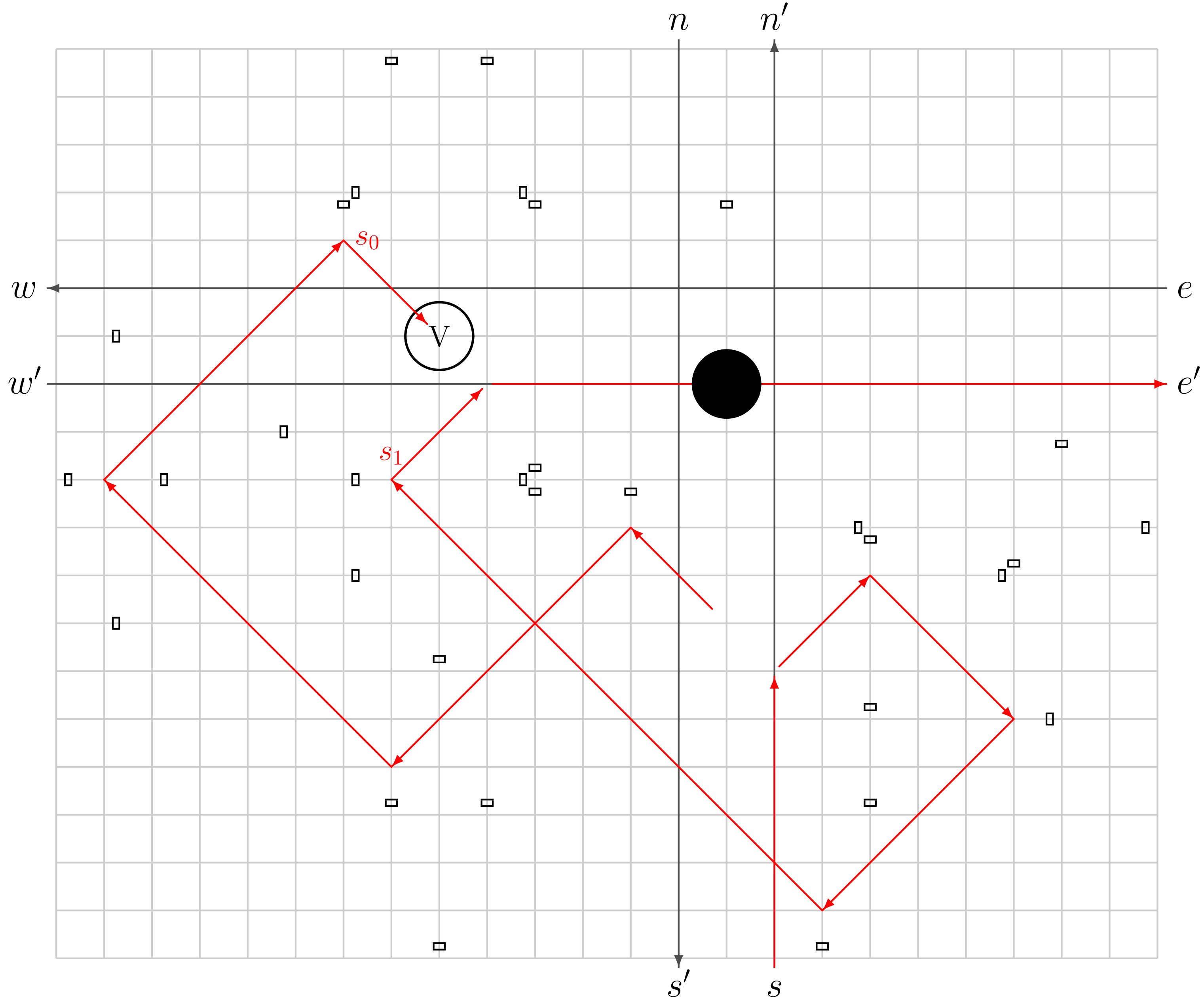
Movements of Balls (State: H, Input: s)



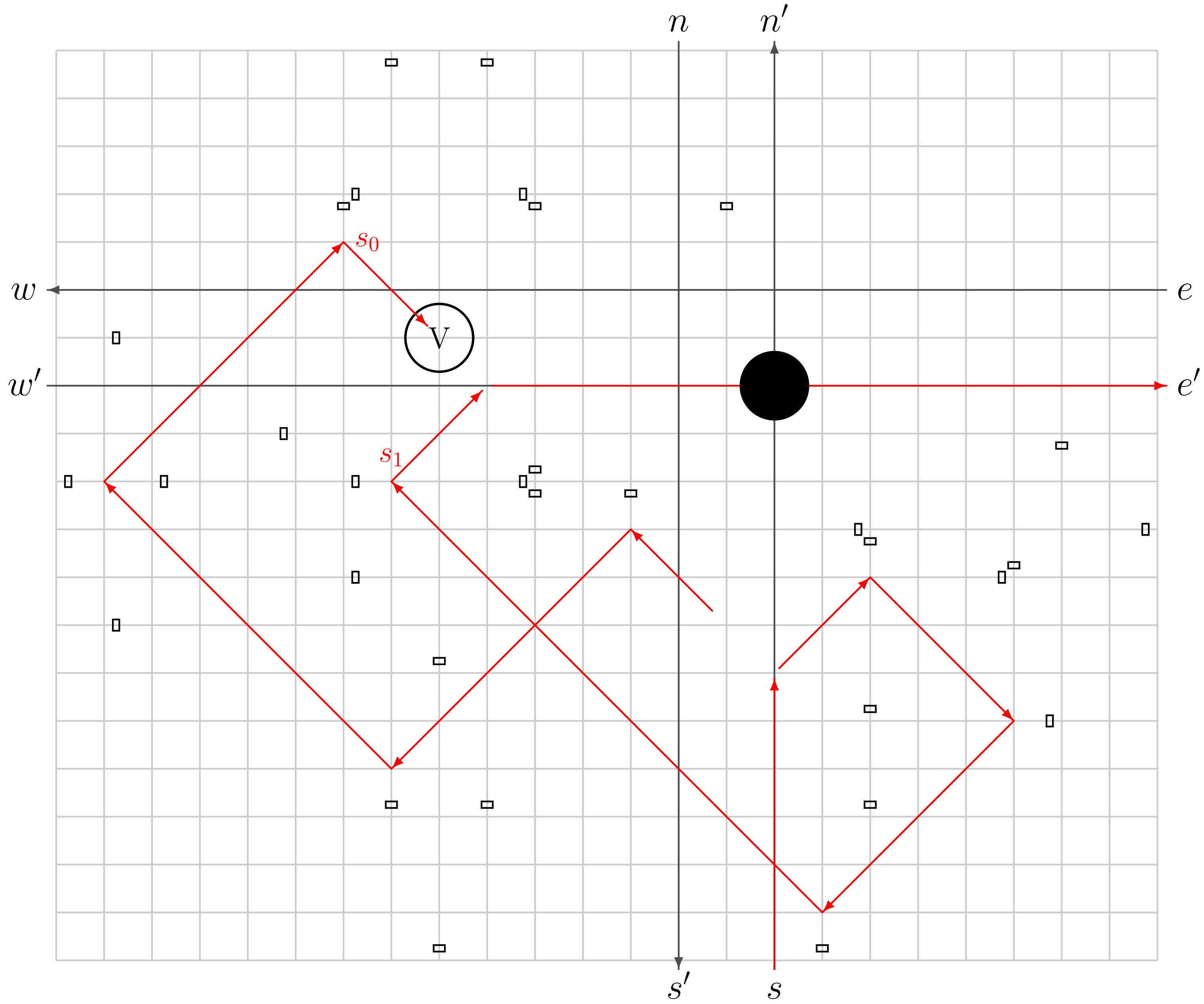
Movements of Balls (State: H, Input: s)



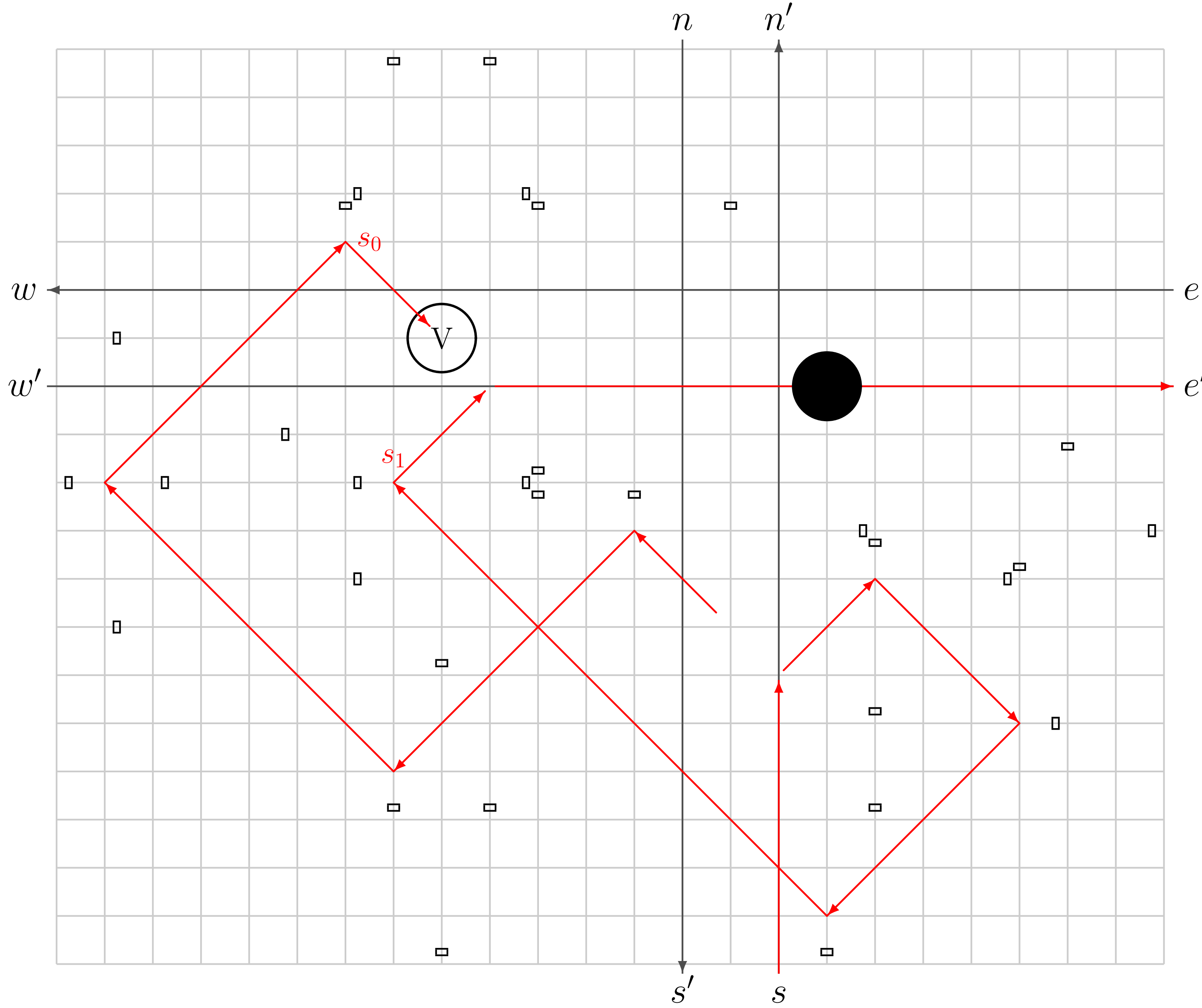
Movements of Balls (State: H, Input: s)



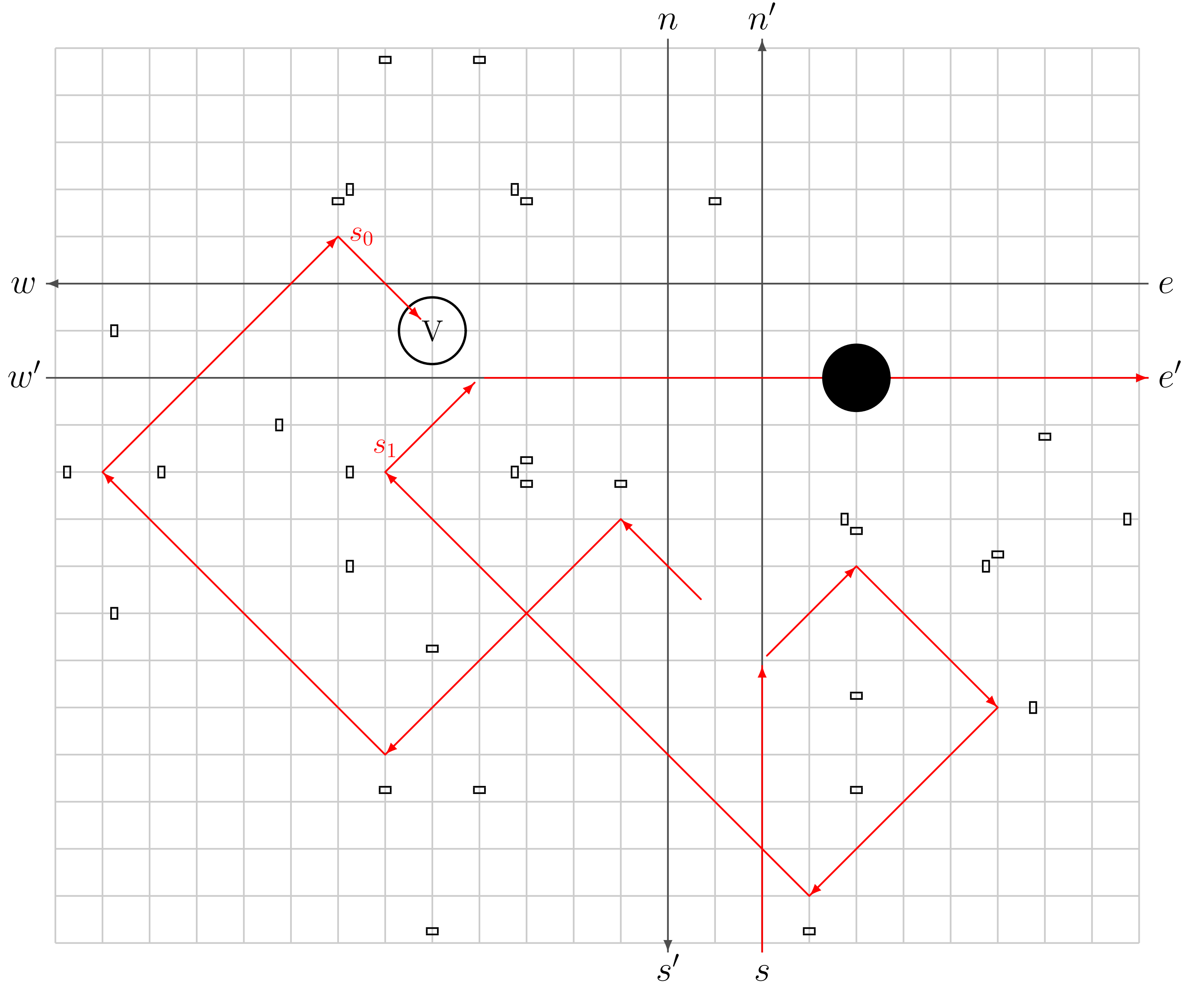
Movements of Balls (State: H, Input: s)



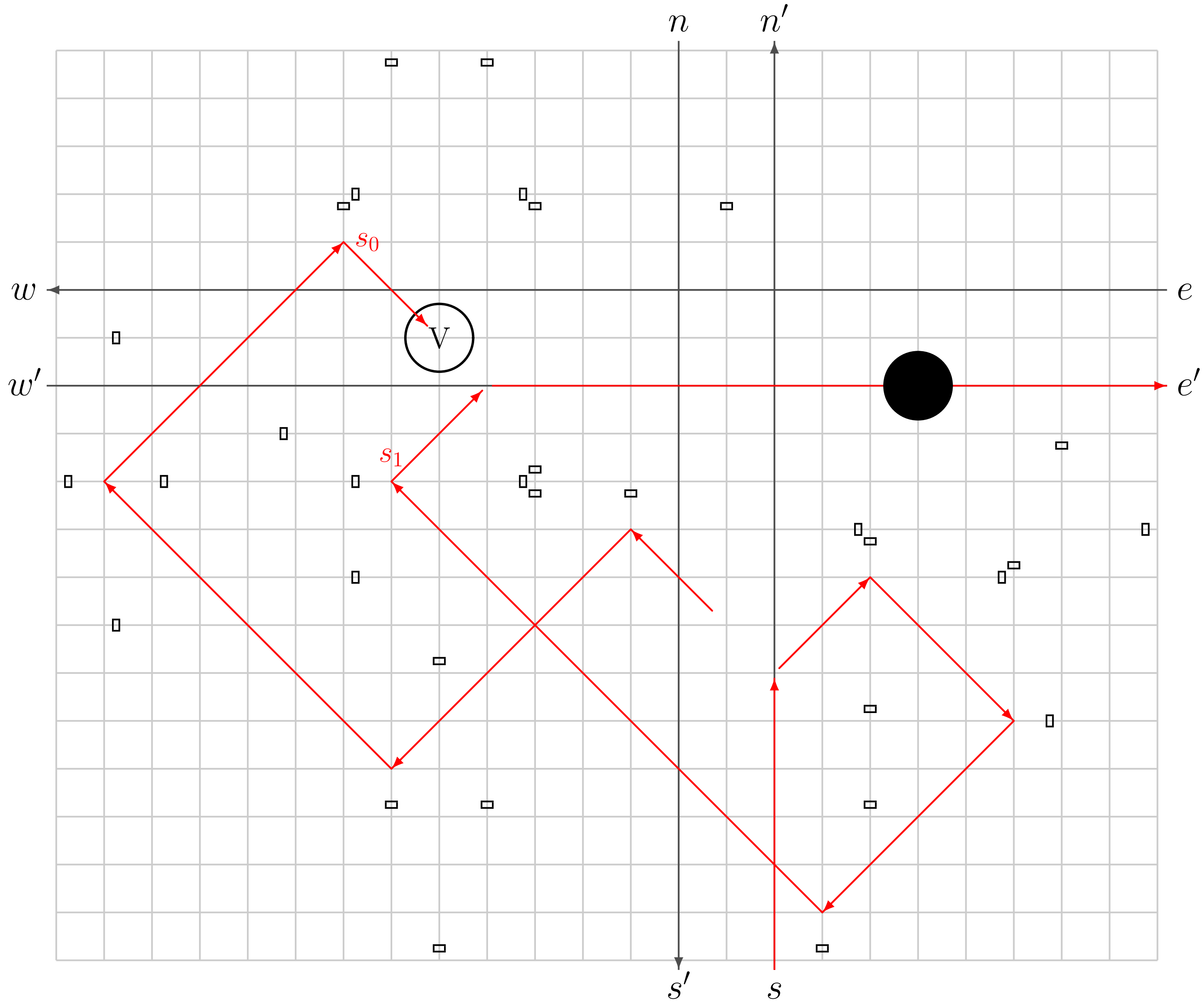
Movements of Balls (State: H, Input: s)



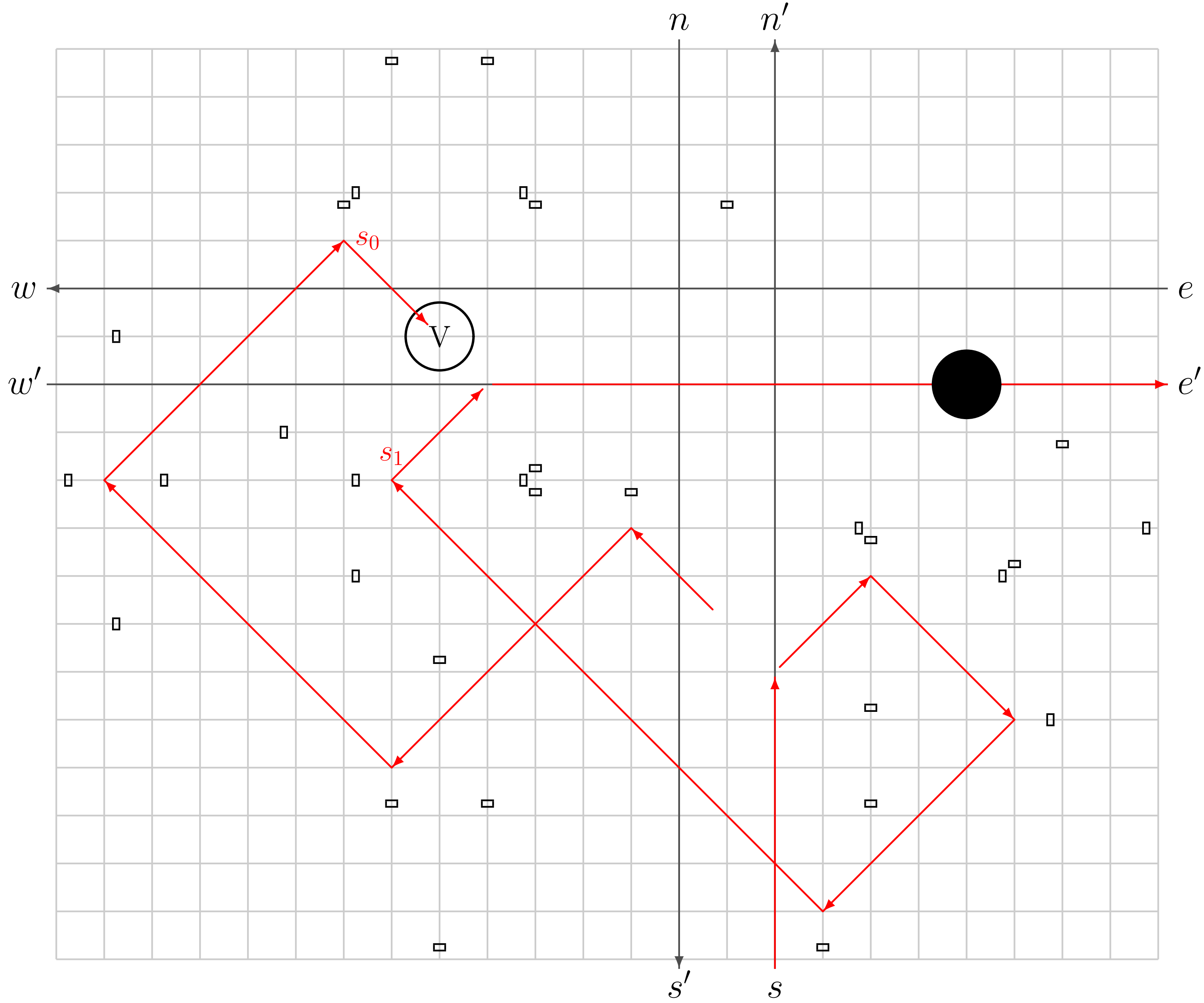
Movements of Balls (State: H, Input: s)



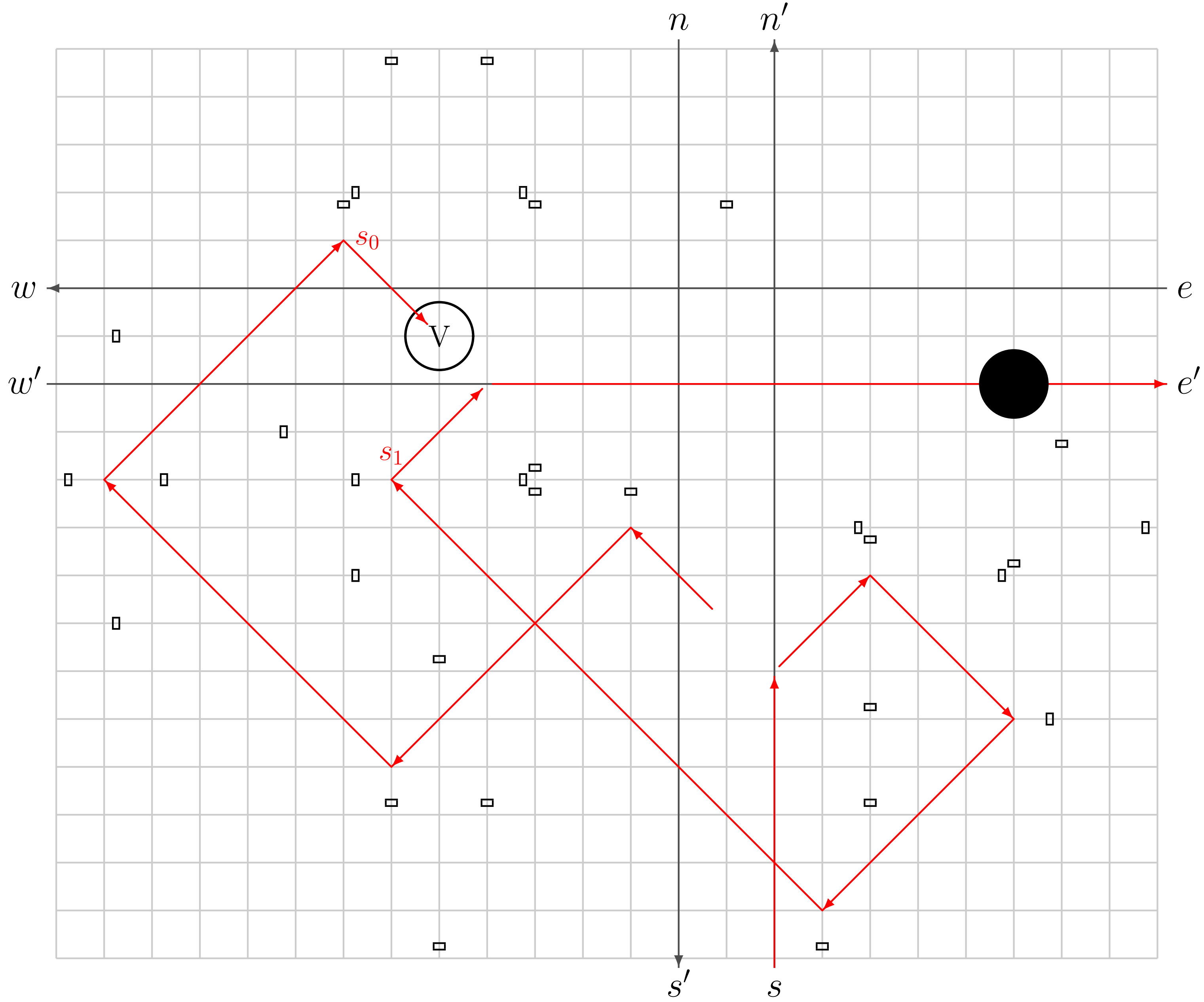
Movements of Balls (State: H, Input: s)



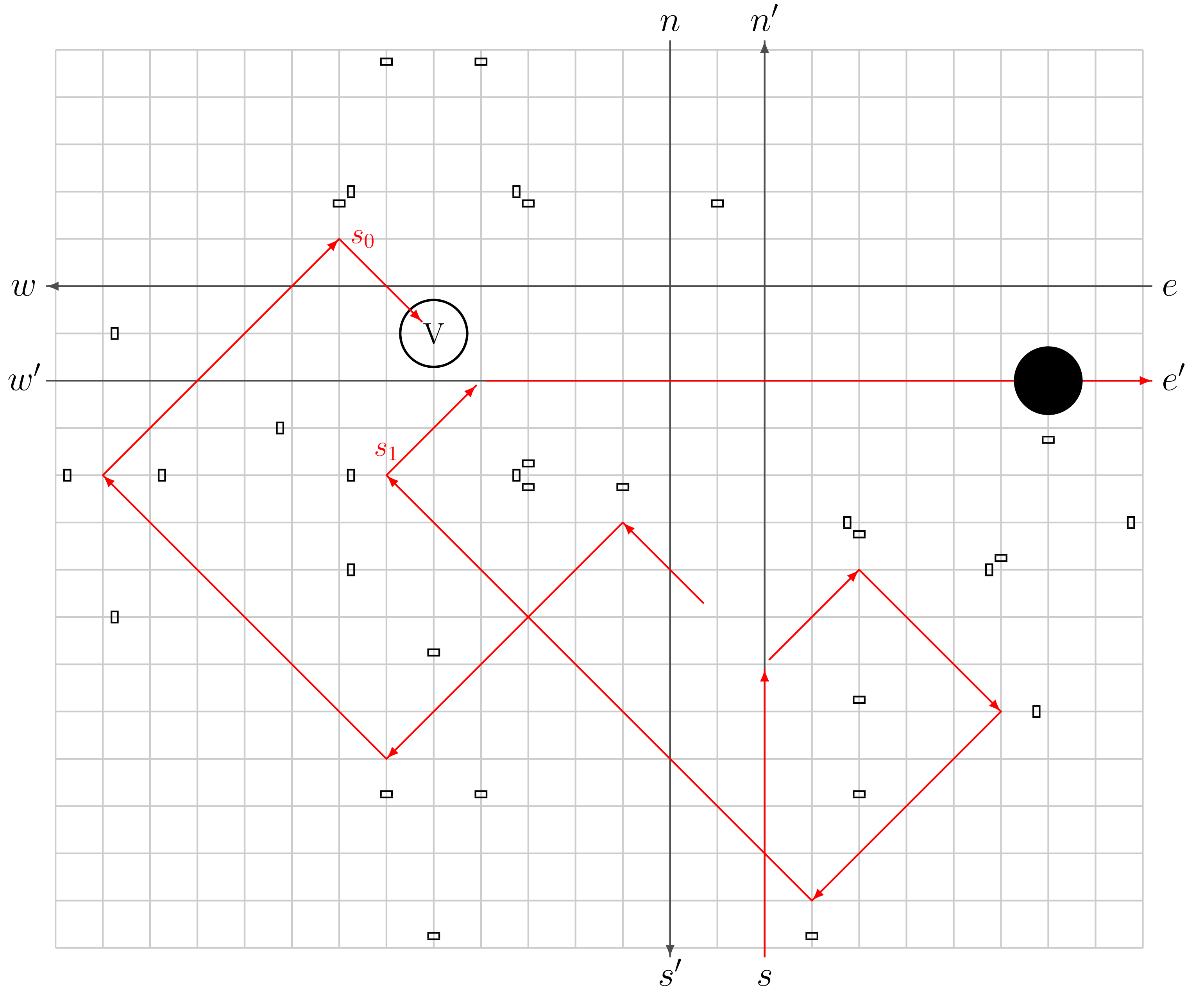
Movements of Balls (State: H, Input: s)



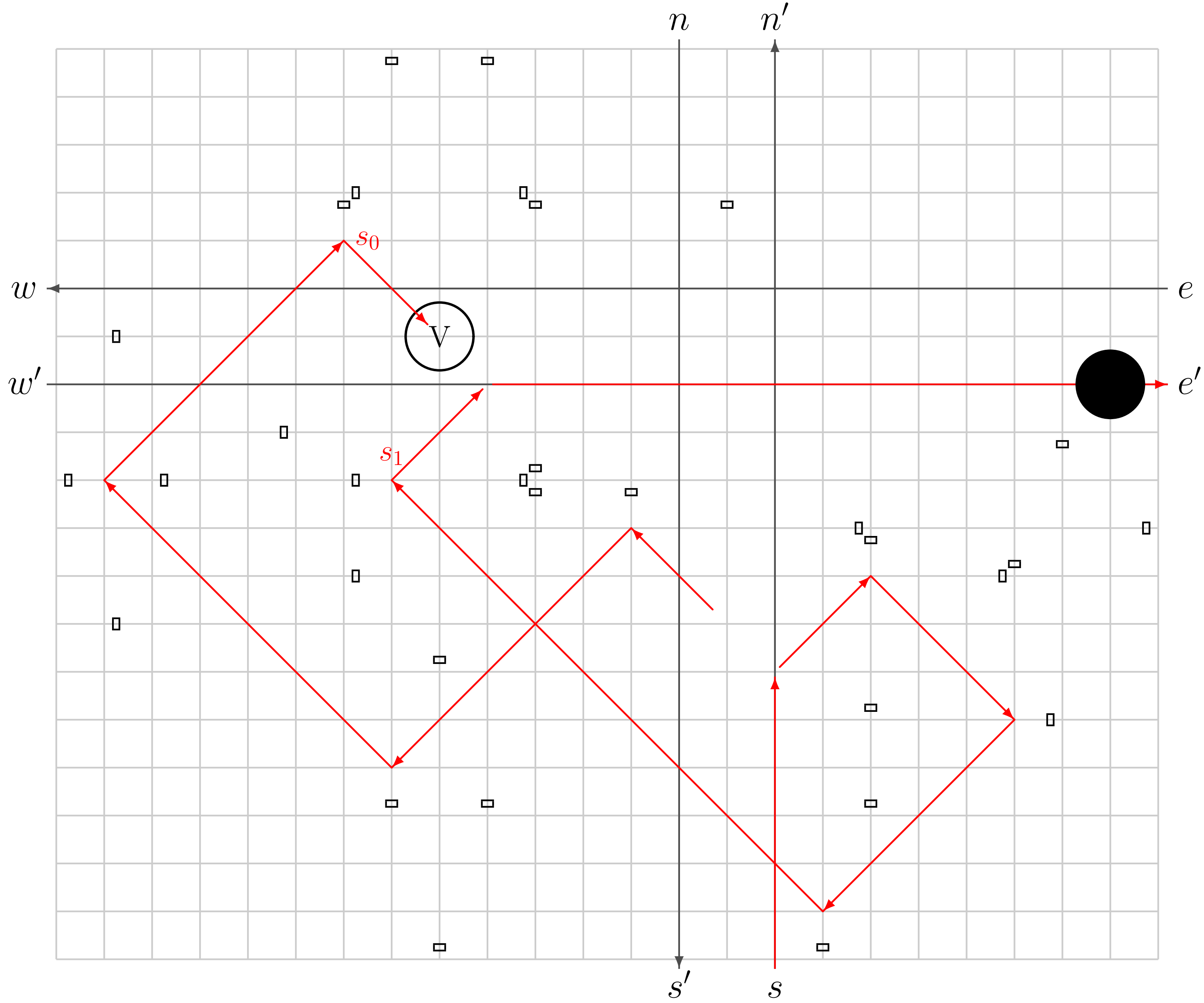
Movements of Balls (State: H, Input: s)



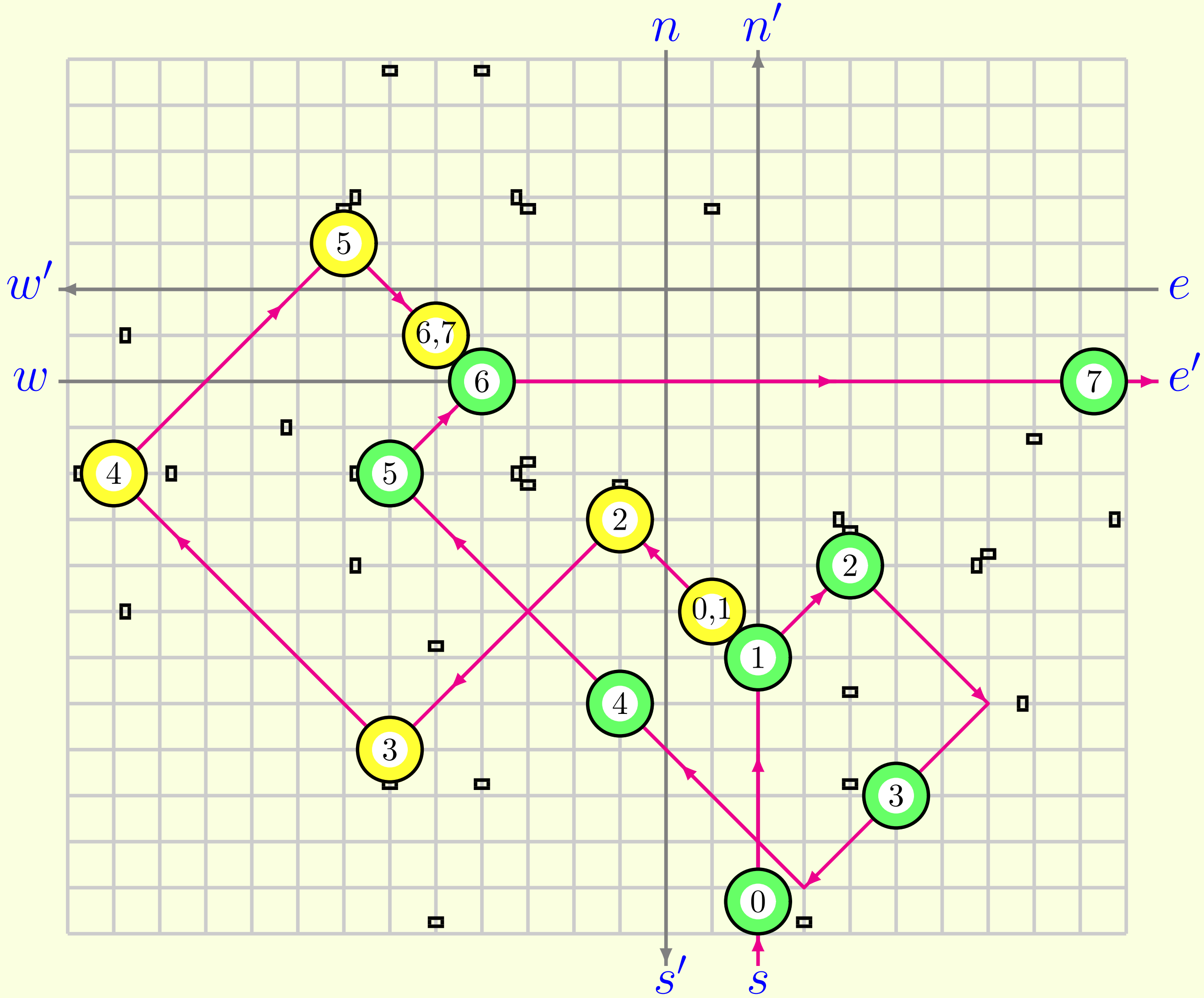
Movements of Balls (State: H, Input: s)



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



BBM realization of an RE (orthogonal case)



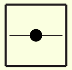
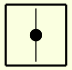
4. Which RLEM is universal, and which is not?

4. Which RLEM is universal, and which is not?

Answer:

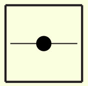
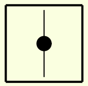
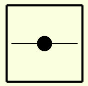
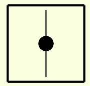
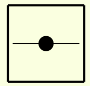
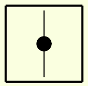
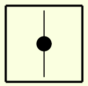
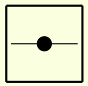
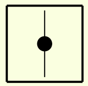
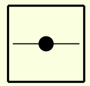
- There are infinitely many 2-state RLEMs, and "all" of them except only 4 are universal.

RE is a 2-state 4-symbol RLEM

$M_{\mathrm{RE}} = ($ {  ,  }, $\{n, e, s, w\}$, $\{n', e', s', w'\}$, $\delta_{\mathrm{RE}})$

internal states — input symbols — output symbols — move function

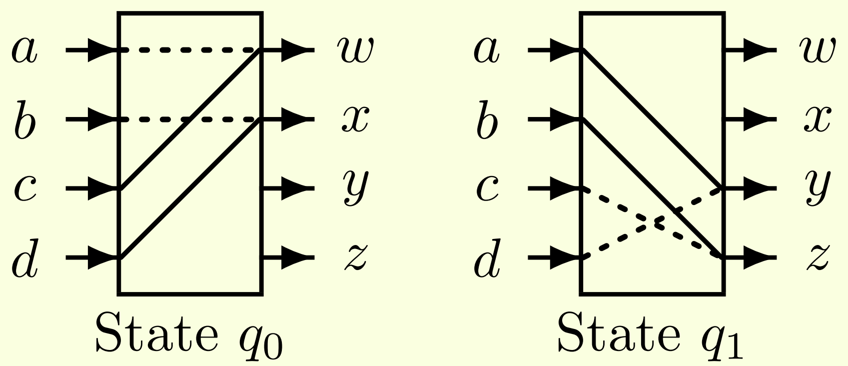
The move function δ_{RE}:

Present state	Input n	Input e	Input s	Input w
State H: 	 w'	 w'	 e'	 e'
State V: 	 s'	 n'	 n'	 s'

We represent a 2-state RLEM graphically

Example: RLEM 4-289 (equivalent to an RE)

Present state	Input a	Input b	Input c	Input 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

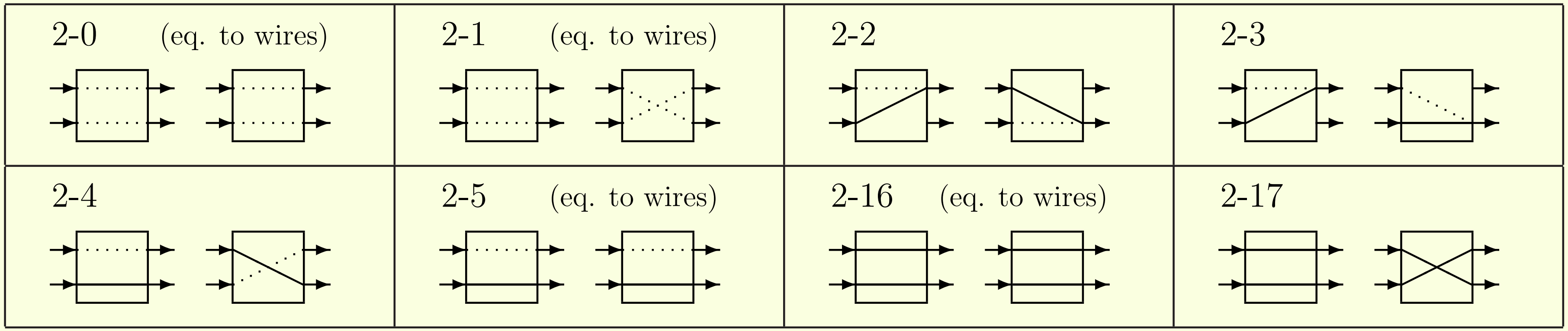
2-state k-symbol RLEMs (k-RLEMs)

1. Total numbers of RLEMs:
   - 2-state 2-symbol RLEMs: $4! = 24$
   - 2-state 3-symbol RLEMs: $6! = 720$
   - 2-state 4-symbol RLEMs: $8! = 40320$

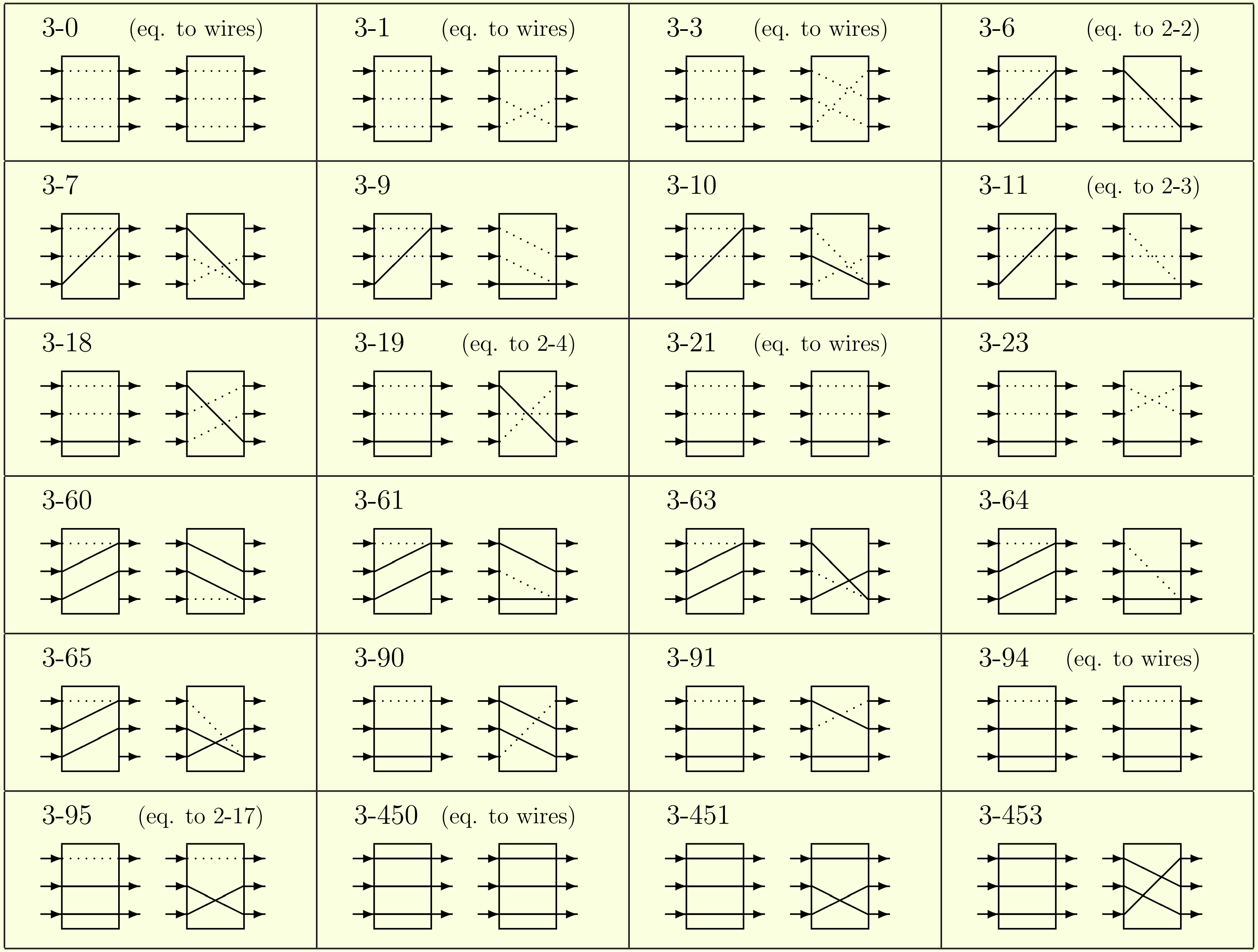
   ID numbers are given to RLEMs:
   e.g., No.2-17, No.3-453, etc.

2. Equivalence classes under the permutation of states and I/O symbols:
   - 2-state 2-symbol RLEMs: 8 classes
   - 2-state 3-symbol RLEMs: 24 classes
   - 2-state 4-symbol RLEMs: 82 classes

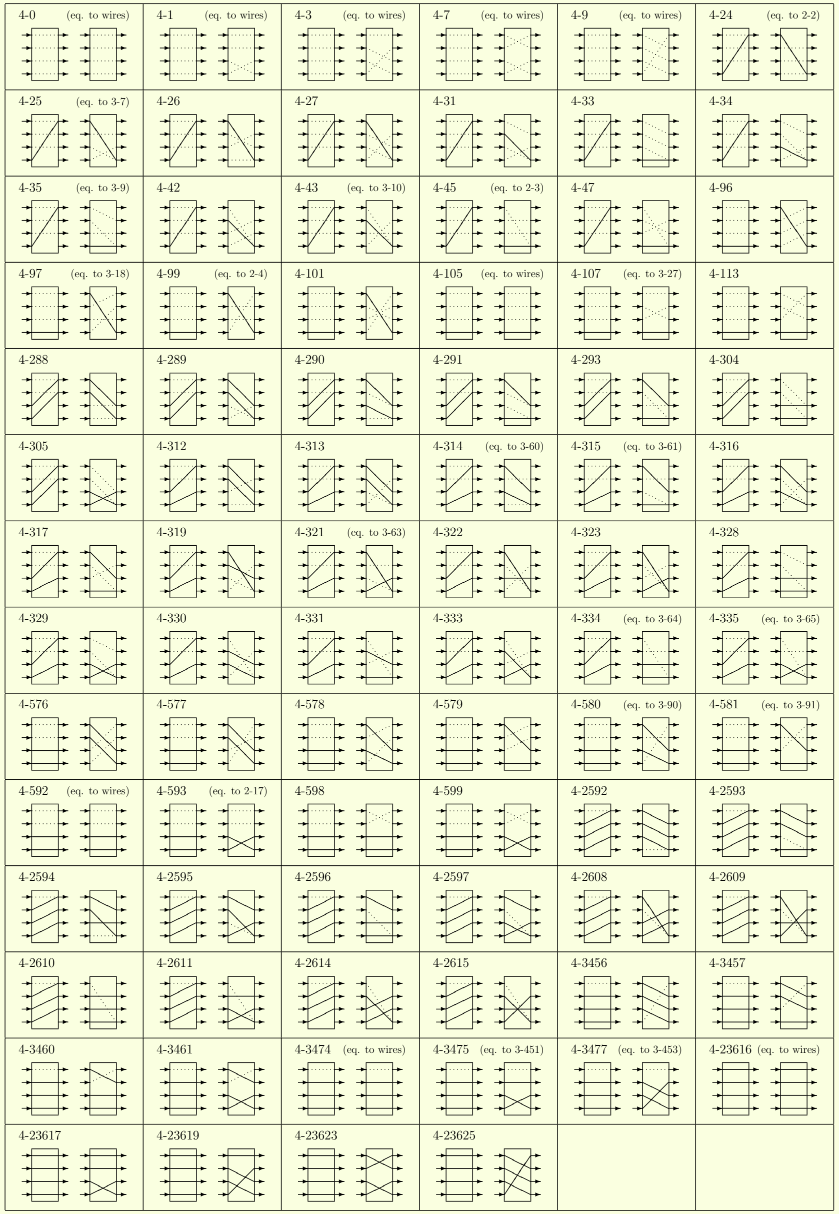
8 representatives of 2-RLEMs



24 Representatives of 3-RLEMs



82 Representatives of 4-RLEMs



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

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

Universality of an RE

From the previous discussion, we can see that an RE is universal in the following sense.

- Any reversible sequential machine can be realized by a garbage-less circuit made only of REs.
- Any reversible Turing machine can be realized by a garbage-less infinite circuit made of REs.

On the other hand, there are many kinds of RLEMs, and thus we have a question.

- Which RLEM is universal and which is not?

Non-degenerate k-RLEMs are ALL universal if $k > 2$

Lemma 1 [Lee et al., 2008] An RE is simulated by a circuit composed of RLEMs 2-3 and 2-4.

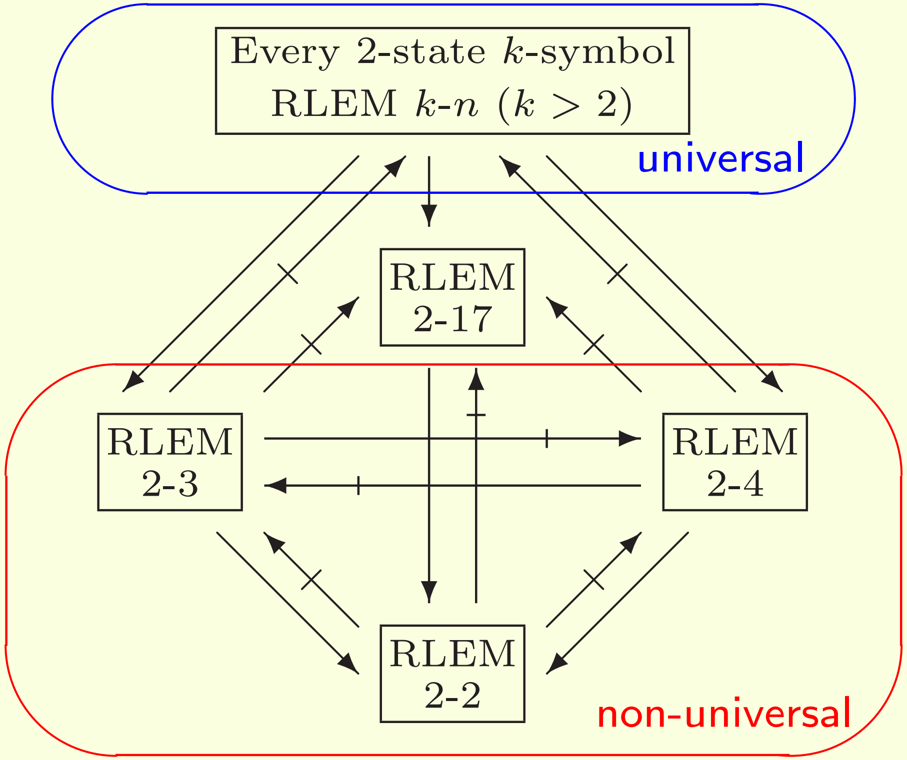
Lemma 2 RLEMs 2-3 and 2-4 can be constructed by any one of 14 non-degenerate 3-RLEMs.

Lemma 3 Any non-degenerate k-RLEM can simulate a non-degenerate $(k-1)$-RLEM, if $k > 2$.

The next theorem is derived from Lemmas 1–3.

Theorem [Morita et al., 2010] Every non-degenerate k-RLEM is universal, if $k > 2$.

Hierarchy of 2-state non-degenerate RLEMs



[Mukai, Morita, (in preparation)]

- $A \to B$ ($A \not\to B$, respectively) represents that A can (cannot) simulate B.
- Any two combination of {2-3, 2-4, 2-17} is universal. [Lee, et al., 2008], [Mukai, Morita, (in preparation)]

Concluding Remarks

Q1. What are RLEMs?

A: They look interesting objects to study.

Q2. How can we construct reversible machines by RLEMs?

A: There is an easy implementation method.

Q3. Can RLEMs be implemented in a reversible physical system efficiently?

A: Ideally yes, but practically unknown.

Q4. Which RLEM is universal, and which is not?

A: There are several simple and universal RLEMs.

References

[Bennett, 1973] Bennett, C.H., Logical reversibility of computation, *IBM J. Res. Dev.*, **17**, 525–532 (1973).

[Büning, Priese, 1980] Büning, H.K., Priese, L., Universal asynchronous iterative arrays of Mealy automata, *Acta Informatica*, **13**, 269–285 (1980).

[Fredkin, Toffoli, 1982] Fredkin, E., Toffoli, T., Conservative logic, *Int. J. Theoret. Phys.*, **21**, 219–253 (1982).

[Keller, 1974] Keller, R.M., Towards a theory of universal speed-independent models, *IEEE Trans. Computers*, **C-23**, 21–33 (1974).

[Lee et al., 2008] Lee, J., Peper, F., Adachi S., Morita, K., An asynchronous cellular automaton implementing 2-state 2-input 2-output reversed-twin reversible elements, *Proc. ACRI 2008*, LNCS 5191, Springer-Verlag, 67–76 (2008).

[Morita, 2001] Morita, K., A simple reversible logic element and cellular automata for reversible computing, *Proc. 3rd Int. Conf. on Machines, Computations, and Universality*, LNCS 2055, Springer-Verlag, 102–113 (2001).

[Morita, 2003] Morita, K.: A new universal logic element for reversible computing, *Grammars and Automata for String Processing* (C. Martin-Vide, V. Mitrana, eds.), Taylor & Francis, London, 285–294 (2003).

[Morita et al., 2005] Morita, K., Ogiro, T., Tanaka, K., Kato, H., Classification and universality of reversible logic elements with one-bit memory, *Proc. 4th Int. Conf. on Machines, Computations, and Universality*, LNCS 3354, Springer-Verlag, 245–256 (2005).

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

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

[Morita et al., 2010] Morita, K., Ogiro, T., Alhazov, A., Tanizawa, T., Non-degenerate 2-state reversible logic elements with three or more symbols are all universal, *Proc. RC 2010* (2010). (Final version: *J. Multiple-Valued Logic and Soft Computing*, **18**, 37–54 (2012)).

[Toffoli, 1980] Toffoli, T., Reversible computing, *Automata, Languages and Programming* LNCS 85, Springer-Verlag, 632–644 (1980).