Ricochet Robots with Infinite Horizontal Board is Turing-complete

Samuel Masseport, Tom Davot, Rodolphe Giroudeau

To cite this version:

Samuel Masseport, Tom Davot, Rodolphe Giroudeau. Ricochet Robots with Infinite Horizontal Board is Turing-complete. Journal of Information Processing, 2023, 31, pp.413-420. 10.2197/ipsjjip.31.413 . lirmm-04164381

HAL Id: lirmm-04164381
https://hal-lirmm.ccsd.cnrs.fr/lirmm-04164381

Submitted on 18 Jul 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Ricochet Robots with infinite horizontal board is Turing-complete*

Samuel Masseport, Tom Davot, and Rodolphe Giroudeau

[1] LIRMM, Univ Montpellier, CNRS, Montpellier, France surname.name@lirmm.fr
[2] Université de Technologie de Compiègne, CNRS, Heudiasyc, France
tom.davot@hds.utc.fr

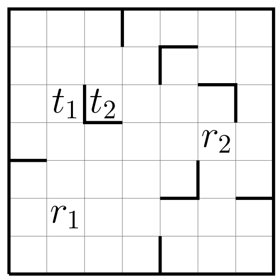
Abstract. This paper investigates the Ricochet Robots game problem from a complexity standpoint. The problem consists of moving robots in a rectangular square-tiled board, from initial tiles to reach specific target tiles. A robot can only move vertically or horizontally and when it starts to move in a given direction, the robot follows this direction, until being blocked by a wall or another robot. This paper proves that the corresponding decision problem to Ricochet Robots is Turing-complete for endless board game and an infinite number of robots. A reduction from a universal Turing machine to Ricochet Robots is exhibited.

1 Introduction

"Ricochet Robots" [1,4,5,6] is a puzzle board game designed by A. Randolph, in which a player moves pieces (robots) in a rectangular square-tiled board from an initial position to a given set of selected locations, with the fewest possible moves. The game-board is a rectangular square-tiled board that contains walls placed on the edges of some tiles. Robots can move horizontally or vertically on the board. A robot moves in a direction until being blocked by a wall or another robot. Each step consists of selecting both a robot and a direction that the robot will follow. To solve the puzzle, the player must reach a configuration where all target tiles are covered by robots of corresponding colors. It is often necessary to move a robots that serves as guide to stop the movement of another to an appropriate tile (see Figure 1 for an example). Several robots cannot move at the same time. In the original game, the board is a square (16×16 tiles) which contains four different colored robots and one colored target tile. The color of the target tile corresponds to the color of one of the robots. The player can move all robots and must reach the target tile with the robot of the same color.

Ricochet Robots game can be categorized as a sliding game like the PushPush game studied by Demaine et al. [2] or the Atomix game studied by Holzer and Schwoon [8] and Huffner et al. [10]. Icking et al. [11,12] considered the exploration problem of a grid polygon with or without obstacles inside it. Engels and

* Preprint version. The editor version is available at https://doi.org/https://doi.org/10.2197/ipsjjip.31.413



In this grid, r_1 cannot be placed on the target tile t_1 without the help of r_2. If the robot r_2 reaches the target tile t_2 with the two movements $\uparrow$ then $\leftarrow$ and creases to move, then r_1 will never reach t_1.
A possible solution for this instance is to execute the following moves: r_2: $\leftarrow$; r_1: $\uparrow, \leftarrow, \downarrow, \rightarrow$; r_2: $\uparrow, \rightarrow, \downarrow$.

Fig. 1. Example of an instance of Ricochet Robots. To solve it, r_1 must be placed on the target tile t_1 while r_2 must be on t_2 at the same time.

Kamphans [4] studied the solvability of Ricochet Robots with n uncolored robots and one target tile, and proved that this problem is NP-complete. Hesterberg and Kopinsky [7] studied the parameterized complexity of Ricochet Robots and Atomix. Gebser et al. [5,6] used Ricochet Robots game as a benchmark for answer set programming while Butko et al. [1] proposed to study how humans try to solve Ricochet Robots. The same authors reused the result of Holzer and Schwoon on Atomix to show that Ricochet Robots is also PSPACE-complete [3]. Another proof of the PSPACE-completeness of Ricochet Robots was given independently by Masseport et al. [13].

The contribution of this article is the construction presented in Section 4: any Turing machine can be simulated by a Ricochet Robots game with an endless board and an infinite number of robots. We can construct an instance of Ricochet Robots whose solvability depends on whether the Turing machine halts or not. This construction establishes the following theorem:

Theorem 1. *Ricochet Robots is Turing-complete.*

Section 2 provides background information on this work and several gadgets with specifics properties are presented in Section 3. These gadgets are used in Section 4 to prove that Ricochet Robots with an endless board and an infinite number of robots is Turing-complete.

2 Preliminaries

2.1 Ricochet Robots

The original game implies four different colored robots and one colored target tile. The decision problem of Generalized Reachability (GR) is defined as a generalization of Reachability Problem introduced by Engels and Kamphans [4].

A *board game* is a rectangular square-tiled board (all the tiles are the same size and are vertically and horizontally aligned) that contains horizontal and vertical walls between some tiles. In the following, we define a board B as a set of walls.Some tiles of the board game, called *target tiles* are colored. A *robot* is a colored token on a tile. A tile cannot contain more than one robot at the same time. An instance of Ricochet Robots $I = (G, R, T)$ is constituted by a board game B, a set of robots R and a set of target tiles T. A configuration is *winning* if each colored target tile is covered by a robot of the same color. To reach a winning configuration, at each step, the player can move a robot vertically or horizontally on the game board. When a robot starts to move, it follows this direction, until it hits an obstacle (*i.e.* another robot or a wall). Target tiles do not stop movement. The GENERALIZED REACHABILITY (GR) problem is defined as follows:

GENERALIZED REACHABILITY (GR)
Input: An instance of Ricochet Robots $I = (B, R, T)$
Question: Is there a reachable winning configuration?

The result of Turing-completeness presented in Section 4 uses a single color for target tiles and robots. For simplicity, the color is not specified. The construction depicted in the following creates an instance of Ricochet Robots of infinite size in the horizontal dimension. In order to show that the reduction is a many-one reduction, the instance resulting from it must be expressed in a finite way, otherwise the reduction would not consist in a computable function because the production of the instance would not terminate. Hence, in the following an infinite-size board is represented in a finite way by using something close to regular expressions to repeat some patterns indefinitely. Let $I_1 = (B_1, R_1, T_1)$ and $I_2 = (B_2, R_2, T_2)$ two instances of GR such that the horizontal size of B_1 is equal to x_1. The instance $I_1 \circ I_2$ is created by juxtaposing I_2 after I_1. That is, $I_1 \circ I_2 = (B_1 \cup B'_2, R_1 \cup R'_2, T_1 \cup T'_2)$ where (B'_2, R'_2, T'_2) is obtained by a horizontal translation of value x_1 applied to I_2 (*i.e.* the walls of the board, robots and target titles have been shifted to the right). The instance I^* corresponds to an infinite number of juxtapositions of the instance I, in other words: $I^* = I \circ I \circ \ldots$.

2.2 Turing machine

A Turing machine defines an abstract machine which manipulates symbols on a tape. This tape is divided into "cells" and the machine has a "head" positioned over a cell. This head reads and writes symbols on the tape. The cell on which the head is positioned is denoted as the *current cell.* A Turing machine has many definitions. Hopcroft [9] describes a Turing machine M as a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, Y_0, F)$ where:

- Q is a finite, non-empty set of states,
- Σ is a finite set of input symbols,
- Γ is the complete set of tape symbols, Σ is always a subset of Γ,
- $\delta(q, X) \rightarrow (p, Y, D)$ is a transition function which given a state q and a tape symbol X returns a triple containing, a next state p, a symbol Y written in

the cell being scanned and a direction D, either L ("left") or R ("right"), in which the head moves.
- $q_0 \in Q$ is the initial state of M,
- $Y_0 \in \Gamma \setminus \Sigma$ is a blank symbol, and
- $F \subseteq Q$ is a set of final or accepting states.

At each step, a Turing machine has a current state q. The head starts by reading the symbol X on the current cell. Then, according to the value returned by the transition function δ with input (q, X), the current state changes to $p \in Q$, the head writes a new symbol $Y \in \Gamma$ in the current cell and moves to the left or the right cell on the tape.

A universal Turing machine is a Turing machine that can simulate any arbitrary Turing machine on arbitrary input. The universal machine essentially achieves this by reading both the description of the machine to be simulated and the input to that machine from its own tape.

3 Description of constructions

3.1 Wires Representation

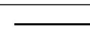
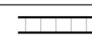
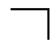
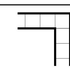
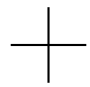
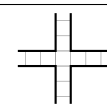
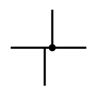
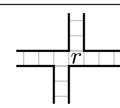
To simplify both our scheme and reader understanding's, gadgets are represented by rectangles that are connected by "wires". A wire corresponds to a line (possibly bent) of free spaces surrounded by walls on both sides. Two wires can cross each other to preserve planarity of the grid (see Table 1). Note that a robot passing through an intersection cannot change its way. In other words, a robot that gets in vertically cannot reach a horizontal output and vice versa. Wires, bent wires and crossed wires guarantee the planarity of the construction.

A helping robot wire is a particular connection of wires: if a robot is on the tile tagged by r (see Table 1), then it can "help" another robot coming from the left to go down. Notice the robot on the tile r cannot go down. In this paper, considering r_1 a helping robot and r_2 a robot needing help, "r_1 intercepts r_2 to reach the correct output (*i.e.* to go down)" means that r_1 goes to the position to help r_2 reach the correct output (*i.e.* the down output). Likewise, considering two gadgets G_1 and G_2 and a wire w, "G_1 intersects w to G_2" means that an output of G_1 and an input of G_2 are connected to w in such a way that a robot coming from G_1 can help a robot coming from w to reach G_2 by stopping it.

3.2 Basic Gadgets

This section presents some basic gadgets and their properties. These gadgets are used in the next section for construction of more sophisticated gadgets. A *k-router gadget* (Figure 2) is a gadget with k' inputs and k outputs (k' and $k > 0$) that has the following property:

Table 1. Wires representations.

	Representation	Corresponding grid
wire		
bent wire		
crossed wires		
helping robot wire		

Property 1 (Router Property) *When a robot reaches a router gadget (from an input or an output), it can reach any output.*

A *k-synchronizer gadget* is a gadget with k inputs and k outputs (with $k > 0$). A 2-synchronizer gadget (resp. a 3-synchronizer gadget) is depicted in Figure 2 (resp. Figure 3). A synchronizer gadget has the following property:

Property 2 (Synchronize Property) *Let S be a k-synchronizer gadget and suppose that k' robots enter S. If $k' < k$, then no robot can reach an output. If $k' = k$, then each robot can reach a distinct output (i.e. two robots cannot reach the same output).*

The case where the number of robots in a *k-synchronizer gadget* is strictly greater than k is not analyzed because it can not happen in the proposed construction. The next property is defined in order to clarify the construction:

Property 3 (k-No-Return Property) *If there are at most $k > 0$ robots in a gadget that has a k-No-Return Property and at least one of them has reached an output, then none of them can go back to any input.*

Obviously, a k-No-Return Property implies a $(k-1)$-No-Return Property. Note that all gadgets defined in this paper have a 1-No-Return Property. A k-synchronizer gadget has a k-No-Return Property.

4 Turing-completeness

This section is devoted to proving the Turing-completeness of Ricochet Robots game. In order to show this, a many-one reduction from a universal Turing machine to Ricochet Robots is exhibited. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Y_0, F)$ be an arbitrary Turing machine with m states ($Q = \{q_0, \ldots, q_{m-1}\}$) and n symbols ($\Gamma = \{Y_0, \ldots, Y_{n-1}\}$). The following construction is divided in two main gadgets:

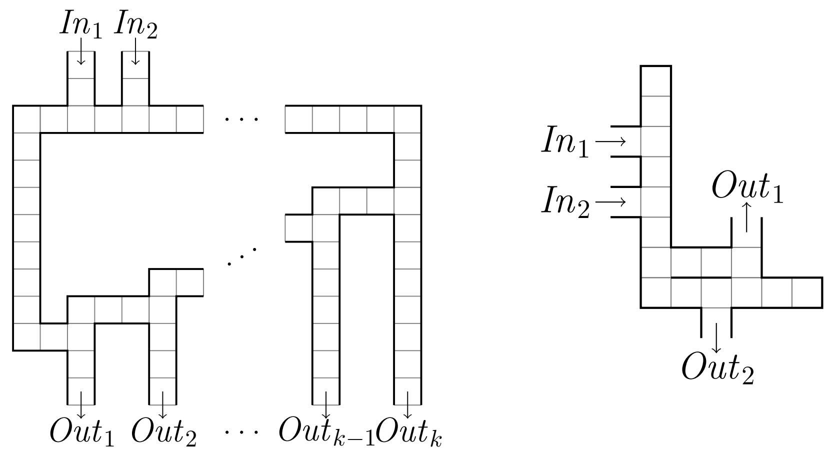


Fig. 2. Left: a *k-router gadget* with two inputs. In this gadget, a robot can reach any output (while traveling through counterclockwise). **Right:** A *2-synchronizer gadget*. Let r_1 (resp. r_2) be a robot that reaches the input In_1 (resp. In_2). In order to get r_1 and r_2 out of the gadget, one of them needs to reach the output Out_1 and the other the output Out_2 (they cannot reach the same one). Possible moves to cross this gadget after r_1 and r_2 have reached the input: r_2: $\downarrow$; r_1: $\downarrow, \rightarrow, \downarrow$; r_2: $\rightarrow, \downarrow$; r_1: $\uparrow$.

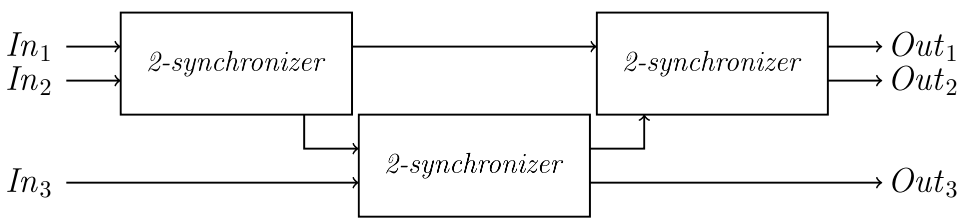


Fig. 3. A *3-synchronizer gadget*. See Section 3.1 for details on the representation system.

- **Tape gadget** used to encode the symbols written on the tape of M.
- **Controller gadget** used to encode the current state of M.

The next subsection describes how the tape gadget and the controller gadget communicate. The Section 4.2 presents the tape gadget and how the read and write operations are simulated. Further, the execution of the transition function by the controller gadget is defined in Section 4.3. Finally, Section 4.4 shows that Ricochet Robots game is Turing-complete.

4.1 Communication system

A *communication pipe* is an endless corridor (*i.e.* a line of free spaces surrounded by walls on both sides) with some helping robots connections. The communication system between tape gadget and controller gadget is composed of several communication pipes. We distinguish between two types of pipes: the “TCP”

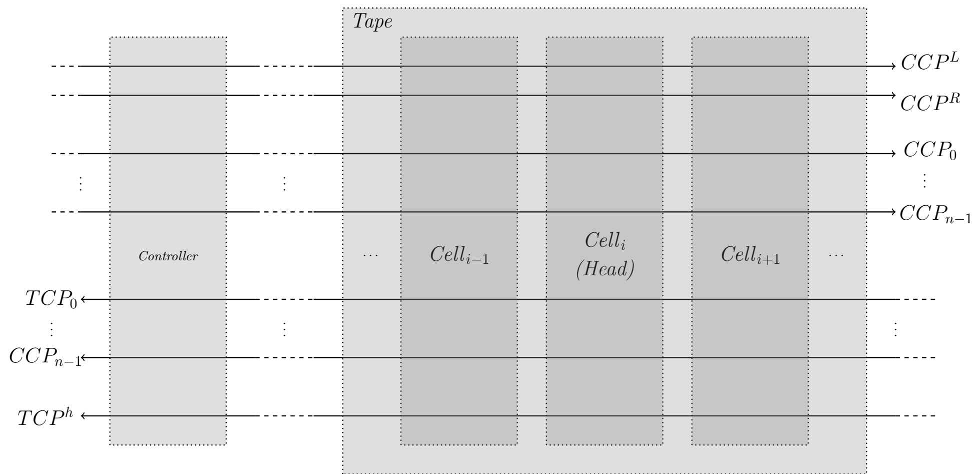


Fig. 4. The communication system. The arrows indicate the direction of communication of the lines and the value corresponds to the message. For example, the communication pipe TCP_0 is used to send the message "Y_0" from the tape to the controller. Note that a pipe cannot be used to communicate in both directions.

communication pipes are used by the tape gadget to send robots to the controller gadget while the "CCP" communication pipes are used by the controller gadget to send robots to the tape gadget.

Construction 1 (*Communication gadget*) *Create three communication pipes* TCP^h, CCP^L *and* CCP^R *and* $\forall Y_i \in \Gamma$ *create two communication pipes* TCP_i *and* CCP_i.

The TCP_i communication pipes are used by the tape gadget to transmit the symbol contained on the current cell whereas the CCP_i communication pipes are used by the controller gadget to transmit the new value of the cell (*i.e.* a pipe cannot be used to communicate in both directions). Precisely, the tape gadget (resp. controller gadget) sends a robot through TCP_i (resp. CCP_i) if the value (resp. the new value) on the current cell is Y_i. The controller indicates if the head has to move to the left or to the right by sending a robot through CCP^L or CCP^R, respectively. The communication pipe TCP^h is used to indicate the end of the reading action to the controller. See Figure 4 for the description of the communication system.

4.2 Head and Tape

This subsection introduces the gadget used to simulate both the head and the tape. The *tape gadget* consists of several *cell gadgets* defined below.

Construction 2 (Tape gadget) *Given a communication gadget produced by Construction 1, construct a* tape gadget *as follows. Let* c_i *be the* i^{th} *cell of the tape. For each cell* c_i*, construct a* cell gadget $Cell_i$ *(see Figure 5) as follows:*

- *construct one n-router gadget* TR_i,
- *construct one* $2(n+1)$*-router gadget* TR'_i,
- *construct two* 2*-synchronizer gadgets* $Left_i$ *and* $Right_i$*, and*
- *for each* $Y_j \in \Gamma$*, construct two* 2*-synchronizer gadgets,* Y_j^i *and* $Write(Y_j^i)$.

Inner gadgets of $Cell_i$ *are connected in the following way:*

- *intersect* CCP^L *with* TR'_i *to* $Left_i$,
- *intersect* CCP^R *with* TR'_i *to* $Right_i$,
- *for each* $Y_j \in \Gamma$*, connect an output of* $Write(Y_j^i)$ *and an output of* TR_i *with the inputs of* Y_j^i *and connect the outputs of* Y_j^i *with the communication pipes* TCP_j *and* TCP^h*, and*
- *for each* $Y_j \in \Gamma$ *intersect the communication pipe* CCP_j *with* TR'_i *to* $Write(Y_j^i)$*, connect an output of* TR'_i *with an input of* $Write(Y_j^i)$ *and connect the last output of* $Write(Y_j^i)$ *with inputs of* $Left_i$ *and* $Right_i$.

Finally, connect $Cell_i$ *with* $Cell_{i-1}$ *and* $Cell_{i+1}$*:*

- *connect the outputs of* $Left_i$ *with the inputs of* TR_{i-1} *and* TR'_{i-1}*, and*
- *connect the outputs of* $Right_i$ *with the inputs of* TR_{i+1} *and* TR'_{i+1}.

In a Turing machine, the head must be able to carry out three actions (if the transition function allows it):

- read the symbol of the current cell,
- write a symbol in the current cell,
- move on the tape to the cell on the right or left.

In each cell gadget $Cell_i$, a robot r_i is used to encode the symbol written in the corresponding cell c_i. The robot r_i is located in the 2-synchronizer gadget Y_j^i if and only if the i^{th} cell of the tape contains the symbol Y_j. If the robot r_i is in the gadget Y_j^i, $Cell_i$ is said to contain Y_j. Two robots h_1 and h_2 are used to simulate the head of M. The robot h_2 is located in the cell gadget $Cell_i$ if and only if the head is over the i^{th} cell. In that case, $Cell_i$ is the current cell gadget.

Lemma 1. *Let* $Cell_i$ *be a gadget produced by Construction 2.*

1. *If robots* r_i *and* h_1 *are located in gadgets* Y_j^i *and* TR_i*, then* r_i *and* h_1 *reach the communication pipes* TCP_j *and* TCP^h.
2. *If robots* r_i *and* h_1 *are coming respectively from communication pipe* CCP_j *and* CCP^L *(resp.* CCP^R*) and* h_2 *is located in* TR'_i*, then* r_i *reaches* Y_j^i*,* h_1 *reaches* TR_{i-1} *(resp.* TR_{i+1} *) and* h_2 *reaches* TR'_{i-1} *(resp.* TR'_{i+1}*).*

Proof. 1. If robot h_1 enters any gadget Y_k^i such that $k \neq j$, then by Property 2, h_1 is stuck in the gadget Y_k^i and r_i in Y_j^i. Thus, suppose that h_1 enters in Y_j^i. By Property 2, robots r_i and h_1 reach the communication pipes TCP_j and TCP^h, respectively.

2. By Property 1, h_2 can intercept successively both r_i and h_1 to $Write(Y_j^i)$ and $Left_i$ (resp. $Right_i$) respectively. If h_2 enters any gadget $Write(Y_k^i)$ such that

$k \neq j$, then by Property 2, r_i, h_1 and h_2 are stuck in their respective gadget. Thus, suppose that h_2 enters in $Write(Y_j^i)$. By Property 2, r_i and h_2 reach Y_j^i and $Left_i$ (resp. $Right_i$), respectively. Then, by Property 2, robots h_1 and h_2 reach gadgets TR_{i-1} and TR'_{i-1} (resp. TR_{i+1} and TR'_{i+1}) respectively.

When the head reaches the i^{th} cell, ($i.e.$ h_1 and h_2 enter in TR_i and TR'_i, respectively) three operations are simulated in the following way.

- **Reading operation.** This operation is executed by sending r_i with the help of h_1 to the controller gadget via TCP_j (Lemma 1(1)). Note that after this operation, h_1 is sent to the controller via TCP^h (this action indicates the end of the reading operation and is required to execute the transition function). Red paths in Figure 5 depict an example of a reading operation.
- **Writing operation.** The controller gadget indicates the new value Y_k to $Cell_i$ by sending back r_i via CCP_k. Thus, writing operation is simulated by intercepting r_i with h_2 in CCP_k and sending it to Y_k^i by crossing $Write(Y_k^i)$ (Lemma 1(2)). Blue paths in Figure 5 depict an example of such operation.
- **Moving operation.** The controller gadget indicates the direction in which the head has to move by sending back h_1 via CCP^L or CCP^R . Thus, a moving operation is simulated by intercepting h_1 with h_2 to $Left_i$ or $Right_i$. After the writing operation, h_2 can join h_1 in its gadget and they are sent to TR_{i-1} and TR'_{i-1}, if left, or TR_{i+1} and TR'_{i+1}, if right (Lemma 1(2)) Green and blue paths in Figure 5 depict an example of such operation.

4.3 Controller gadget

This subsection introduces the gadget used to simulate the transition function.

Construction 3 (Controller gadget) *Given a communication gadget produced by Construction 1, the* controller gadget *(see Figure 6) is constructed as follows. First, create a target tile t. Then, for each state $q_i \in Q$, construct a* state gadget *$State_i$ as follows:*

- *construct a $(2n+1)$-router gadget CR_i and an n-router gadget CR'_i,*
- *for each symbol $Y_j \in \Gamma$, construct a 3-synchronizer gadget Δ_j^i.*

For each state $q_i \in Q$ and for each symbol $Y_j \in \Gamma$, let (q_p, Y_ℓ, D) be the values returned by $\delta(q_i, Y_j)$ (with $D \in \{L, R\}$). State gadgets are connected as follows:

- *connect an output of CR'_i with an input of Δ_j^i,*
- *connect an output of CR_i with an input of Δ_j^i,*
- *intersect TCP^h with CR_i to CR'_i,*
- *intersect TCP_j with CR_i to Δ_j^i,*
- *connect the outputs of Δ_j^i with the inputs of CR_p, C'_{Y_ℓ} and C'_D respectively,*
- *if $\delta(q_i, Y_j)$ is a halting case for M, connect an output of Δ_j^i to the target tile t and close the two other outputs with a wall.*

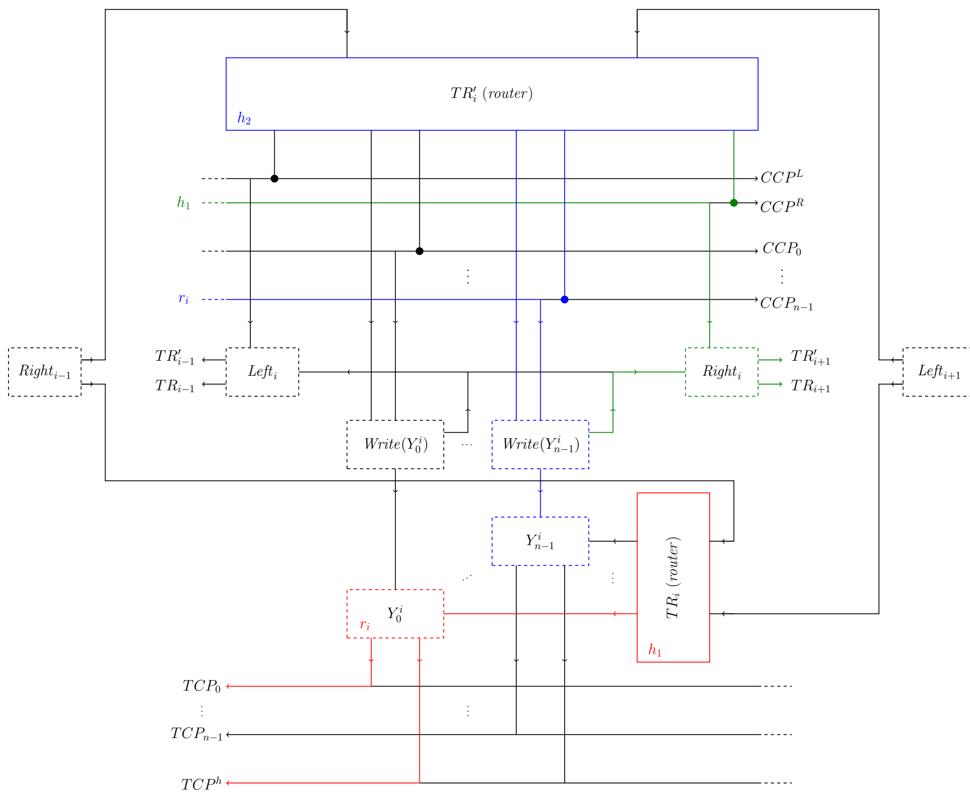


Fig. 5. The cell gadget $Cell_i$ representing the i^{th} cell of the tape of M. See Section 3.1 for details on the representation system. The output TR_{i-1} (resp. TR'_{i-1}) is connected to the input TR_{i-1} (resp. TR'_{i-1}) of the cell on the left. The output TR_{i+1} (resp. TR'_{i+1}) is connected to the input TR_{i+1} (resp. TR'_{i+1}) of the cell on the right.The gadgets $Right_{i-1}$ of the cell gadget $Cell_{i-1}$ and $Left_{i+1}$ of the cell gadget $Cell_{i+1}$ are also depicted. Suppose that there is a robot r_i in the gadget Y_0^i and a robot h_1 in TR_i. The two robots execute the reading operation by following red paths. In this example robots read the symbol Y_0. Now, suppose that there is a robot h_2 in the gadget TR'_i and two robots r_i and h_1 that come from the communication pipes CCP_{n-1} and CCP^R, respectively. The three robots execute the writing operation by following blue paths and the moving operation by following green and blue paths. In this example, robots write the symbol Y_{n-1} in the cell c_i and move the head to the right (i.e. to the cell c_{i+1}). These paths are detailed in Lemma 1.

The role of this gadget is twofold:

- it changes the state of the machine, and
- it transmits to the current gadget cell the symbol to write and the direction in which the head has to move.

A robot s is used to encode the current state of M. That is, s is located in CR_i if and only if the current state of M is q_i. In that case, the controller gadget is said to be in the state q_i.

Lemma 2. *Consider a controller gadget produced by Construction 3. Suppose that s is in CR_i. Suppose that two robots r_j and h_1 are coming from TCP_k and TCP^h respectively. Let (q_p, Y_ℓ, D) (with $D \in \{L, R\}$) be the values returned by $\delta(q_i, Y_k)$. Then, s, r_j and h_1 reach CR_p, CCP_ℓ and CCP^D respectively.*

Proof. By Property 1, s can intercept successively r_j and h_1 in order to help them to reach Δ^i_k and R'_{q_i}, respectively. By Construction 3, the robot r_j cannot reach another gadget than Δ^i_k. Later, by Property 1, h_1 and s can reach any Δ^i_t gadget ($\forall\ Y_t \in \Gamma$). If at least one of h_1 and r_s reaches a gadget Δ^i_t such that $t \neq k$, then by Property 2, s, r_{c_j} and h_1 are stuck in their gadget. Thus, suppose that both s and h_1 enter in Δ^i_k. By Property 2 and according to Construction 3, s, r_j and h_1 reach CR_p, CCP_ℓ and CCP^D respectively.

When the controller gadget receives the robots h_1 and r_j from the tape gadget, it performs a *transiting operation* defined as below.

- **Transiting operation.** Suppose that there is a robot s in the router gadget CR_i and two robots r_j and h_1 coming from TCP_ℓ and TCP^h respectively. According to Lemma 2, r_j and h_1 are sent to the tape gadget through CCP_ℓ and CCP^D (with $D \in \{L, R\}$) according to the transition function. Red paths in Figure 6 depict an example of transiting operation.

4.4 Full construction

An instance of Ricochet Robots that simulates a Turing machine M is presented in this subsection.

Construction 4 *Let M be a Turing machine with input σ. Let G be the board produced by Construction 1, Construction 2 and Construction 3. An instance of* Generalized Reachability (GR) *$I = (B, R, T)$ is created as follows. Robots composing R have the following starting positions:*

- *the robot s is placed in CR_0 (in State$_0$, itself in the controller gadget),*
- *let cell c_i be the initial current cell, place two robots h_1 and h_2 in gadgets TR_i and TR'_i respectively (in the cell gadget Cell$_i$, itself in the tape gadget),*
- *for all cell c_i of the tape of M, let Y_j be the initial symbol of c_i in the input T, a robot r_i starts in the gadget Y^i_j (in Cell$_i$, itself in the tape gadget).*

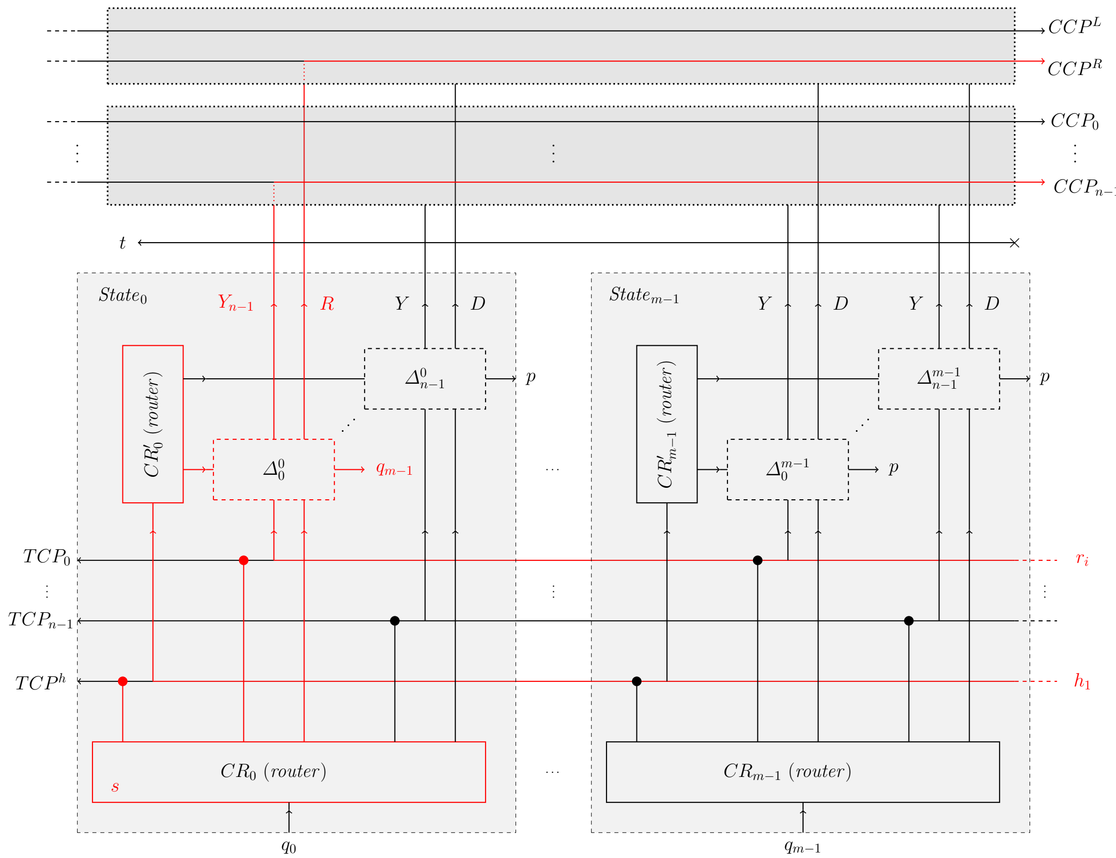


Fig. 6. The *controller gadget* that contains the unique target tile t. The connection on the dotted rectangles and the output p depend of the transition function δ. For example, if $\delta(q_0, Y_0) \to (q_{m-1}, Y_{n-1}, R)$ then the Δ_0^0 gadget is connected as follows: its output p is connected to the input q_{m-1} of the CR_{m-1} gadget, its output Y is connected to the communication pipe CCP_{n-1} and its output D to the pipe CCP^R. Suppose that there is a robot h_2 in the gadget CR_0 and two robots r_i and h_1 coming from the communication pipes TCP_0 and TCP^h, respectively. The three robots execute the transition function by following red paths (see Lemma 2 for more details).

The unique target tile t is located in the controller gadget (see Construction 3), hence $T = t$. A step in the instance produced by Construction 4 is composed of the following sequence of operations: (1) reading operation, (2) transitioning operation, (3) writing operation, and (4) moving operation. Lemma 1 and Lemma 2 ensure that the sequence order is respected and that no other operation is performed.

As said before, even if the instance produced by Construction 4 has an infinite horizontal dimension, it can be encoded in finite way as follows. For all $\ell \in \Gamma$, C^ℓ denotes a cell gadget $Cell_i$ with a robot in TR'_i and another one in Y_ℓ^i. Thus, given an input $\sigma \in \Sigma^* = \ell_1 \ell_2 \ldots \ell_{|\sigma|}$ to M, the tape can be encoded by the string "$(C^{Y_0})^* \circ C^{\ell_1} \circ \ldots \circ C^{\ell_{|\sigma|}} \circ (C^{Y_0})^*$" (Recall that Y_0 is the blank symbol). The controller gadget is then positioned above the first cell gadget that does not contain a blank character. Hence, the construction depicted here is a computational function taking a Turing machine as input and returning an instance of Ricochet Robots.

Theorem 2. *Considering an arbitrary Turing machine M and the corresponding instance I of* GR *obtained by Construction 4, for any k, at the k^{th} step in I and the k^{th} transition in M, the following conditions are respected:*

(a) the controller gadget is in the state q_i if and only if M is in state q_i,
(b) the current cell gadget is $Cell_j$ if and only if the head of M is over the cell c_j of the tape, and
(c) for all cell c_j, the cell gadgets $Cell_j$ contains Y_ℓ if and only if the cell c_j of the tape of M contains the symbol Y_ℓ.

Proof. Clearly (a), (b) and (c) are true if $k = 0$. Suppose that $k > 0$ steps and transitions have been completed in I and M and that (a), (b) and (c) are verified. Let q_i be the current state, c_j be the current cell of the tape and Y_ℓ be the symbol contained in c_j. Let $(p, Y_t\ , D)$ (with $D \in \{L, R\}$) be the values returned by $\delta(q_i, Y_\ell)$. The following shows that (a), (b) and (c) are still verified for the step $k+1$ in I and for the transition $k+1$ in M. First, by Lemma 1(1), a reading operation is performed in the current cell gadget, r_j and h_1 are sent to the controller gadget through TCP_ℓ and TCP^h, respectively. By Lemma 2, a transiting operation is then performed in the controller gadget. The robot s (that represents the current state) enters the state gadget $State_p$ in the gadget CR_p while r_j and h_1 are sent to the tape gadget through CCP_t and CCP^D, respectively. Further by Lemma 1, a writing operation and a moving operation are done in the tape gadget, that is, r_j reaches Y_t^j and h_1 and h_2 enter in gadgets TR_{j-1} and TR'_{j-1} (in the gadget $Cell_{j-1}$) if $D = L$, or in TR_{j+1} and TR'_{j+1} (in $Cell_{j+1}$) if $D = R$. Now, M is in state q_p and its head is over the cell c_{j-1} or c_{j+1} (depending on if $D = L$ or $D = R$). Hence, (a) and (b) are verified for $k+1$. Moreover, the cell c_j contains Y_t in M and the robot is in the Y_t^j (in the gadget $Cell_j$). Since only the cell c_j and the cell gadget $Cell_j$ have been modified by the previous transition and step, (c) is verified for $k+1$.

Theorem 3. *The instance $I = (B, R, T)$ of* Generalized Reachability (GR) *problem obtained by Construction 4 admits a winning configuration if and only if M reaches a halting state.*

Proof. Suppose that I admits a winning configuration (*i.e.* a robot reaches the target tile). Thus, robots reach a gadget Δ_j^i such that $\delta(q_i, Y_j)$ is a halting case for M. Hence, according to Theorem 2, robots have executed a sequence of transitioning operations such that the corresponding sequence of transitions for M is a transition from the initial configuration to a halting case.

Suppose that M reaches a halting case $\delta(q_i, Y_j)$. Thus, there is a sequence of transitions from the initial configuration to a halting case in M. According to

Theorem 2, the corresponding sequence of transitioning operations allows robots of I to reach the gadget Δ_j^i. Thus, one robot can reach the target tile t, then I admits a winning configuration.

By Theorem 3, the Generalized Reachability (GR) problem can simulate any Turing machine, then it can simulate a universal Turing machine, and then we prove the Theorem 1.

Acknowledgment

We would like to thank Tathagata Basu for his thorough re-reading of this article. The work of the author from Université de technologie de Compiègne was carried out in the framework of the Labex MS2T funded by the French Government through the program "Investments for the future" managed by the National Agency for Research.

References

1. Butko, N., Lehmann, K.A., Ramenzoni, V.: Ricochet robots—a case study for human complex problem solving. Proceedings of the Annual Santa Fe Institute Summer School on Complex Systems (CSSS'05) (2005)
2. Demaine, E.D., Hoffmann, M., Holzer, M.: Pushpush-k is pspace-complete. In: Proceedings of the 3rd International Conference on FUN with Algorithms. pp. 159–170. Citeseer (2004)
3. Engels, B., Kamphans, T.: Complexity of randolph's robot game technical. Tech. rep., Rheinische Friedrich-Wilhelms-Universität Bonn (2006)
4. Engels, B., Kamphans, T.: Randolphs robot game is NP-hard! Electronic Notes in Discrete Mathematics **25**, 49–53 (2006)
5. Gebser, M., Jost, H., Kaminski, R., Obermeier, P., Sabuncu, O., Schaub, T., Schneider, M.: Ricochet robots: A transverse asp benchmark. In: International Conference on Logic Programming and Nonmonotonic Reasoning. pp. 348–360. Springer (2013)
6. Gebser, M., Kaminski, R., Obermeier, P., Schaub, T.: Ricochet robots reloaded: A case-study in multi-shot asp solving. In: Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation, pp. 17–32. Springer (2015)
7. Hesterberg, A., Kopinsky, J.: The parameterized complexity of ricochet robots. Journal of Information Processing **25**, 716–723 (2017)
8. Holzer, M., Schwoon, S.: Assembling molecules in atomix is hard. Theoretical computer science **313**(3), 447–462 (2004)
9. Hopcroft, J.E.: Introduction to automata theory, languages, and computation. Pearson Education India (2008)
10. Hüffner, F., Edelkamp, S., Fernau, H., Niedermeier, R.: Finding optimal solutions to atomix. In: Annual Conference on Artificial Intelligence. pp. 229–243. Springer (2001)
11. Icking, C., Kamphans, T., Klein, R., Langetepe, E.: Exploring an unknown cellular environment. In: European Workshop on Computational Geometry. pp. 140–143 (2000)

12. Icking, C., Kamphans, T., Klein, R., Langetepe, E.: Exploring simple grid polygons. In: International Computing and Combinatorics Conference. pp. 524–533. Springer (2005)
13. Masseport, S., Darties, B., Giroudeau, R., Lartigau, J.: Ricochet robots game: complexity analysis. Tech. rep., LIRMM, Université de Montpellier (2019)