Logic-Based Encodings for Ricochet Robots

Filipe Gouveia, Pedro T. Monteiro, Vasco Manquinho,
Inês Lynce

{filipe.gouveia,pedro.tiago.monteiro,vasco.manquinho,ines.lynce}@tecnico.ulisboa.pt

INESC-ID / Instituto Superior Técnico
University of Lisbon
Lisbon, Portugal

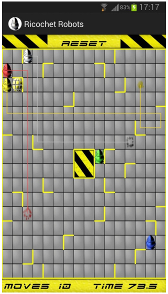
September 2017

Table of Contents

Table of Contents

Motivation

- Puzzles and games were always interesting for Artificial Intelligence
  - N-Queens, Towers of Hanoi, Sudoku, ...
- Answer Set Programming approaches exist
- Real board game and mobile apps



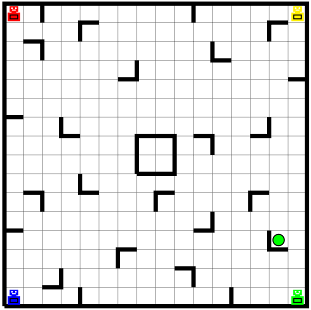
Ricochet Robots
Introduction

Simple board game, created by Alex Randolph in 1999

Also known as *Rasende Roboter* or *Randolph's Robots*

Content:

- 16 by 16 grid (256 positions), with some barriers
- 4 robots with different colors
- A goal position for a given robot



Ricochet Robots

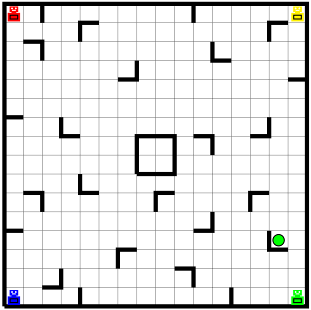
Introduction

Rules:

- Only one robot moves at a time
- Robots can only move horizontally or vertically
- Once a robot starts moving, it only stops when it reaches a barrier or another robot

Goal:

- Put the correspondent robot in target position



Ricochet Robots

Problem Specification

It is easy to find a solution for the game

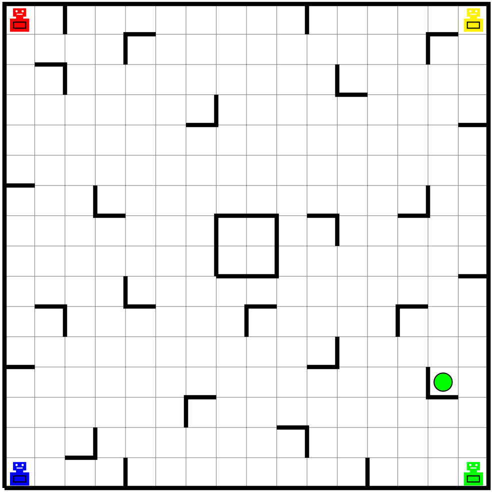
But it is not easy to find an **optimal** solution! (NP-Hard)

An optimal solution is one with the least amount of moves

Our goal: Find one optimal solution for a given starting configuration and a target position

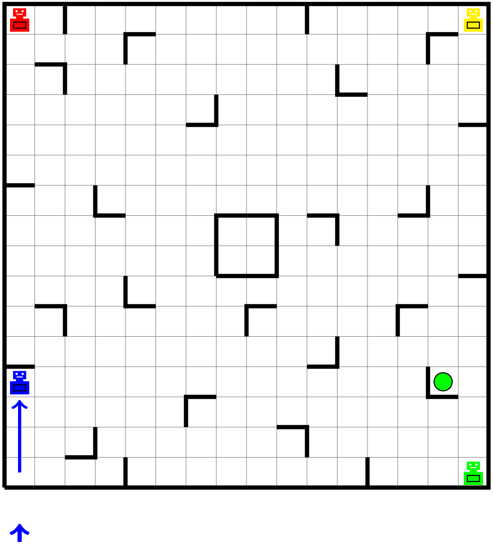
Ricochet Robots

Example



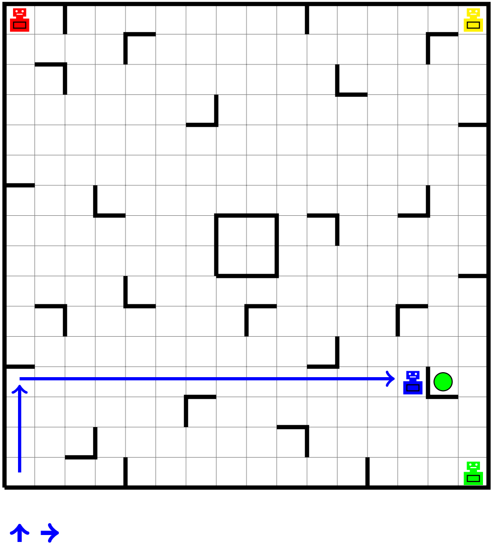
Ricochet Robots

Example



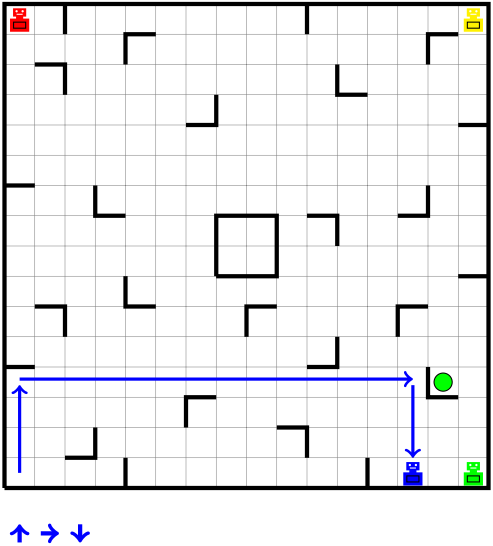
Ricochet Robots

Example



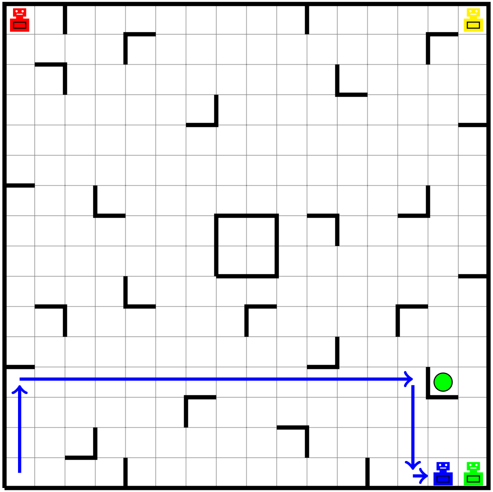
Ricochet Robots

Example



Ricochet Robots

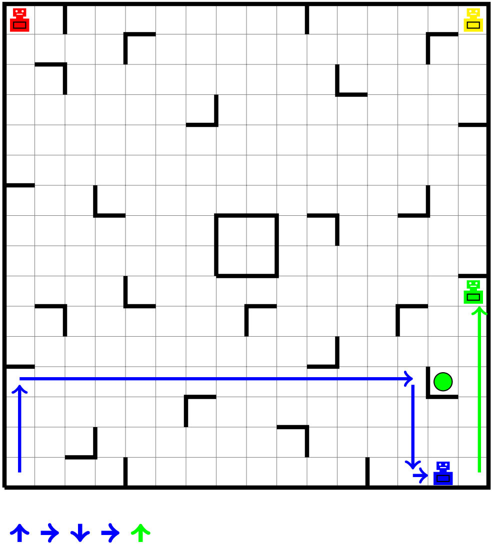
Example



↑ → ↓ →

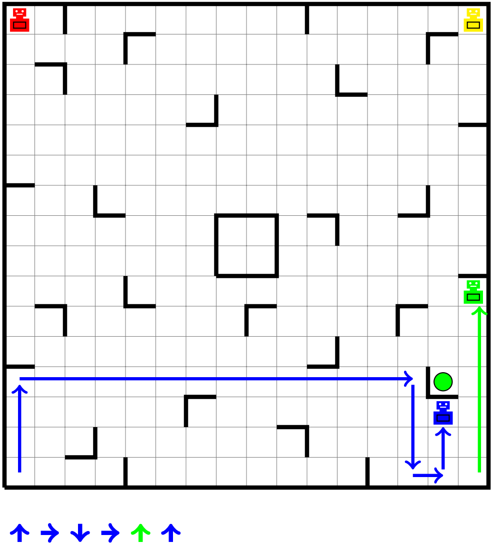
Ricochet Robots

Example



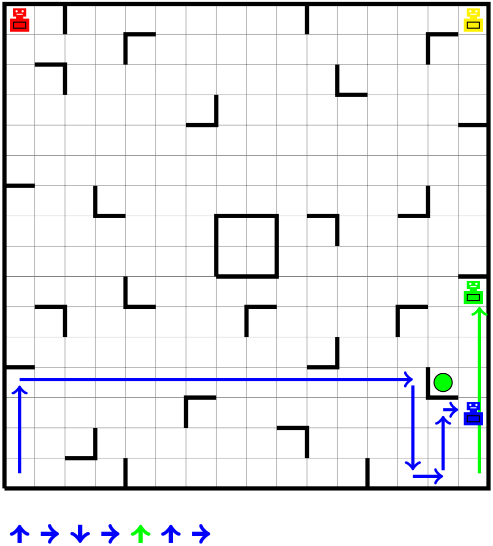
Ricochet Robots

Example



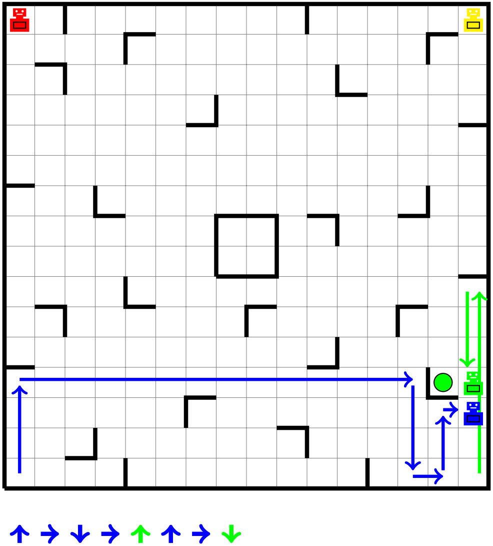
Ricochet Robots

Example



Ricochet Robots

Example



Ricochet Robots

Example

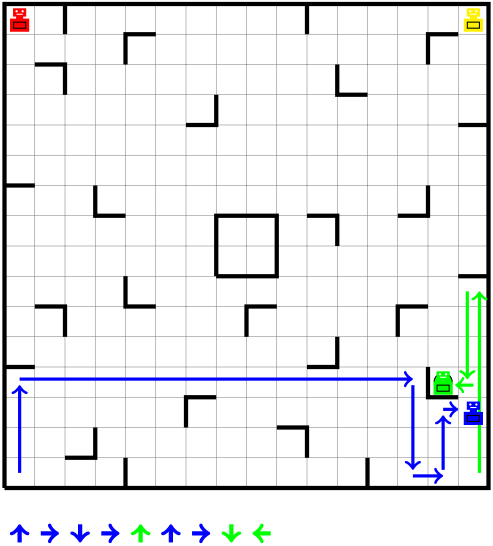
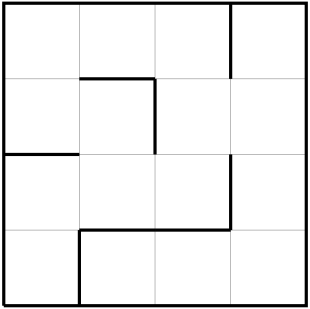


Table of Contents

Encodings

Base Model

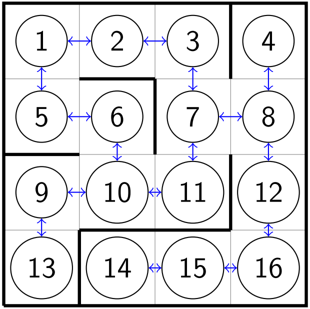
- Board represented as a graph



Encodings

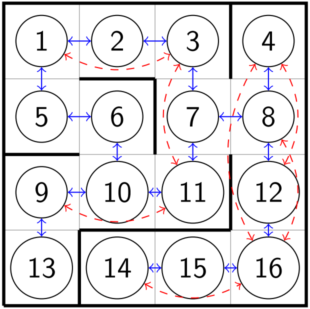
Base Model

- Board represented as a graph
- Each position is a vertex
- Adjacent positions with no barriers are connected by an edge



Encodings
Base Model

- Board represented as a graph
- Each position is a vertex
- Adjacent positions with no barriers are connected by an edge
- An *extended edge* is added between a position and each other position in the same row or column **iff** there are no barriers between them



Encodings

Boolean Encoding

The problem was encoded into Boolean Satisfiability (SAT) using conjunctive normal form (CNF)

- A CNF formula is a conjunction ($\wedge$) of clauses
- A clause is a disjunction ($\vee$) of literals
- A literal is a Boolean variable or its negation

A SAT problem is to decide whether there exists an assignment to the variables of a CNF formula that satisfies the formula

$$(x_1 \vee x_2) \wedge (x_3 \vee \neg x_1) \wedge \neg x_4$$

The problem was encoded into Boolean Satisfiability (SAT) using conjunctive normal form (CNF)

- A CNF **formula** is a conjunction ($\wedge$) of clauses
- A clause is a disjunction ($\vee$) of literals
- A literal is a Boolean variable or its negation

A SAT problem is to decide whether there exists an assignment to the variables of a CNF formula that satisfies the formula

$$\boxed{(x_1 \vee x_2) \wedge (x_3 \vee \neg x_1) \wedge \neg x_4}$$

Encodings

Boolean Encoding

The problem was encoded into Boolean Satisfiability (SAT) using conjunctive normal form (CNF)

- A CNF formula is a conjunction ($\wedge$) of clauses
- A **clause** is a disjunction ($\vee$) of literals
- A literal is a Boolean variable or its negation

A SAT problem is to decide whether there exists an assignment to the variables of a CNF formula that satisfies the formula

$$\boxed{(x_1 \vee x_2)} \wedge \boxed{(x_3 \vee \neg x_1)} \wedge \boxed{\neg x_4}$$

Encodings

Boolean Encoding

The problem was encoded into Boolean Satisfiability (SAT) using conjunctive normal form (CNF)

- A CNF formula is a conjunction ($\wedge$) of clauses
- A clause is a disjunction ($\vee$) of literals
- A **literal** is a Boolean variable or its negation

A SAT problem is to decide whether there exists an assignment to the variables of a CNF formula that satisfies the formula

$$(\boxed{x_1} \vee x_2) \wedge (x_3 \vee \boxed{\neg x_1}) \wedge \neg x_4$$

Boolean Encoding

Variables

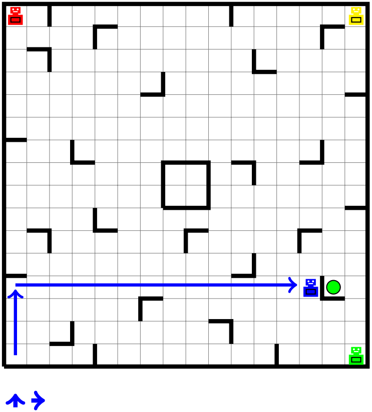
Consider a graph $G = (V, E)$ representing a $d \times d$ board:

- Set of vertexes $V = \{v_1, v_2, ..., v_n\}$, where $n = d \times d$
- E is the set of *extended edges*
- Set of μ robots $R = \{r_1, r_2, ..., r_\mu\}$
- Number of time steps $\eta \in \mathbb{N}_0$

Boolean Encoding
Variables

Propositional variables:

- $X^t_{j,k}$ - Position variable
  - $X^2_{206,blue}$ is true
- $Poss^t_{j,l}$ - Possible movement variable
  - $Poss^1_{193,206}$ is true
- M^t_k - Movement variable
  - M^1_{blue} is true



Boolean Encoding
Rules

- Represent the initial state of the board
  - Each robot k is in its initial position v_j at time 0
    $$X^0_{j,k}$$
  - The goal state of robot k is position v_j
    $$X^\eta_{j,k}$$

Boolean Encoding

Rules

- Robot placement at each time step
  - A robot must be in at least one vertex

$$\bigvee_{j=1}^{n} X_{j,k}^{t}$$

  - A robot cannot be in two vertexes

$$\bigwedge_{j=1}^{n} \bigwedge_{l=j+1}^{n} \neg X_{j,k}^{t} \vee \neg X_{l,k}^{t}$$

Boolean Encoding

Rules

- A robot either stays in the same vertex v_l or comes from a vertex v_j, such that $(v_j, v_l) \in E$, from which a movement is possible

$$X_{l,k}^{t+1} \implies X_{l,k}^{t} \vee \bigvee (X_{j,k}^{t} \wedge Poss_{j,l}^{t} \wedge M_k^t)$$

- A movement is possible if there are no robots along the way

$$Poss_{j,l}^{t} \implies \bigwedge_{h \in p(j,l)} \bigwedge_{k=1}^{\mu} \neg X_{h,k}^{t}$$

$p(j, l)$ denotes the vertexes in the path from v_j to v_l

Boolean Encoding

Rules

- A vertex is a *stop* vertex for a given direction if there is a robot in the following vertex

$$Poss^t_{j,l} \implies \bigvee_{k=1}^{\mu} X^t_{m,k}$$

  v_m represents the next vertex adjacent to v_l considering the direction from v_j to v_l

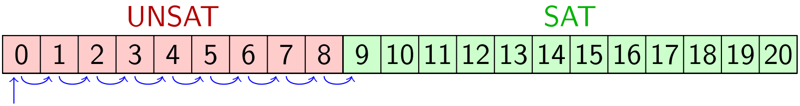
- Only one robot can move at each time step

$$\bigwedge_{k=1}^{\mu} \bigwedge_{h=k+1}^{\mu} \neg M^t_k \vee \neg M^t_h$$

Boolean Encoding

Optimal Solution

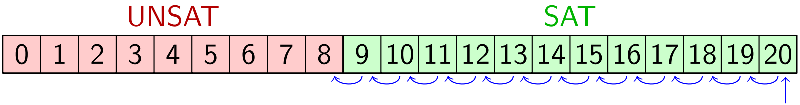
- Presented encoding allows to check if there is a solution for a given time step limit.
- It is possible to find, using a SAT solver, an optimal solution using an iterative approach:
  - UNSAT - SAT
    - Start with $\eta = 0$
    - If unsatisfiable, increment η by 1 and repeat
    - The first satisfiable solution is an optimal solution



Boolean Encoding

Optimal Solution

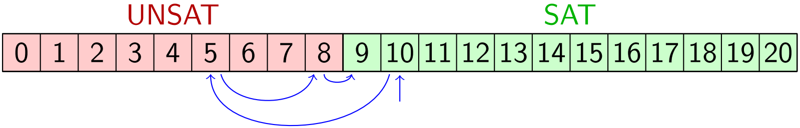
- Presented encoding allows to check if there is a solution for a given time step limit.
- It is possible to find, using a SAT solver, an optimal solution using an iterative approach:
  - SAT - UNSAT
    - Start with $\eta = 20$ (or another large value)
    - If satisfiable, decrement η by 1 and repeat
    - The last satisfiable solution is an optimal solution



Boolean Encoding

Optimal Solution

- Presented encoding allows to check if there is a solution for a given time step limit.
- It is possible to find, using a SAT solver, an optimal solution using an iterative approach:
  - Binary Search
    - Start with $LB = 0$, $UB = 20$, and average η_{avg}
    - If unsatisfiable, update $LB = \eta_{avg} + 1$, otherwise $UB = \eta_{avg}$
    - Stop when lower bound is equal to upper bound



Other logic-based tools were used:

- Satisfiability Modulo Theories (SMT)
- Constraint Programming (CP)

Other Logic-Based Encodings
SMT

- Implementation of the Boolean encoding presented
- Other encodings have been tried
  - The Boolean encoding was the best performing encoding

Other Logic-Based Encodings
CP

- Encoding developed with *MiniZinc* CP Solver

- Integer variables

  - Each position is represented by two integers
    - Row
    - Column

Table of Contents

Experimental Evaluation

- Proposed encodings were compared with previous proposed Answer Set Programming (ASP) approaches
- Instances:
  - Board 16×16 (256 positions)
  - 4 robots starting at each corner of the board
  - Each instance is one of the (256) possible positions for the target, for the same robot
- 600 seconds of CPU time limit

Experimental Evaluation

Tools and Approaches

- Tools:
  - *Glucose* SAT Solver
  - *z3* SMT Solver
  - *MiniZinc* with the Gecode solver provided
- Approaches:
  - Iterative algorithms:
    - Linear UNSAT-SAT search
    - Linear SAT-UNSAT search
    - Binary search
  - Incremental and non-incremental implementations were considered
  - Native optimization directive considered in CP encoding

Results

ASP vs. SAT

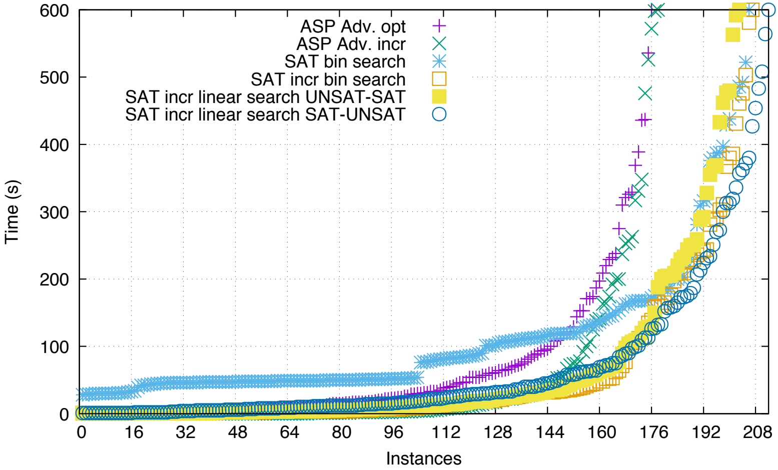


Figure: Time comparison (in seconds) between ASP Advanced encodings and SAT encoding

Results

Other Logic-Based Encodings

- Worse than ASP and SAT Encodings in terms of number of instances solved
- CP approach solves 4 to 5 times fewer instances than SAT and ASP approaches
  - Solved instances required less CPU time

Table of Contents

Conclusions

- New logic-based encodings proposed
- Proposed Boolean Encoding solved a larger set of instances than the previously published ASP encodings
  - Incremental linear search SAT-UNSAT is the most efficient
- SMT and CP approaches may still be improved
  - looks like there is no straightforward approach performing better than SAT or ASP encodings

Future Work

- Explore MaxSAT approaches
- Apply planning tools
- Extend the current encodings to similar problems

Thank you!

Questions?

Results

ASP vs. SAT - decision problem

Table: Average time results for decision problem in seconds, considering a limit of 20 time steps. The timeouts were not considered when computing the average time.

	Average Time (s)	#Timeouts
ASP Plain	252,38	51
ASP Advanced	34,95	20
SAT	42,98	3

Results

ASP vs. SAT

Table: Average time (in seconds) of solved instances.

	Time (s)	#Timeouts
ASP Advanced optimization	59,34	81
ASP Advanced incremental	41,84	79
SAT using binary search	108,37	51
SAT using linear search (UNSAT-SAT)	123,01	62
SAT using incremental binary search	56,79	50
SAT using incremental linear search (UNSAT-SAT)	57,43	54
SAT using incremental linear search (SAT-UNSAT)	66,27	45

Results

Other Logic-Based Encodings

Table: Average time (in seconds) of solved instances.

	Time (s)	#Timeouts
SMT incremental using linear search (UNSAT-SAT)	137,88	86
SMT incremental using linear search (SAT-UNSAT)	312,07	94
SMT incremental using binary search	144,60	87
CP using linear search (UNSAT-SAT)	24,06	239
CP using linear search (SAT-UNSAT)	27,38	251
CP using binary search	73,19	241
CP using optimization statement	27,95	251
ASP Advanced incremental	41,84	79
SAT using incremental linear search (SAT-UNSAT)	66,27	45

Other Logic-Based Encodings
CP

- Encoding developed with the language used by *MiniZinc* CP Solver.
- Input:
  - Dimension of the board;
  - List of robots;
  - Robots initial positions and goals;
  - List of barriers;
  - Number of time steps;
- Integer variables
  - Each position is represented by two integers
    - Column
    - Line

Other Logic-Based Encodings

CP

- Rules:
  - The position of a robot at time step 0 must be the initial position of the robot;
  - The position of a robot at the limit time step must be the goal position, if that robot has a goal;
  - At most one robot moves at each time step;
  - If a robot does not move then it stays in the same position;
  - If a robot moves, it moves to a valid position;
    - A predicate `valid_movement` was created to guarantee the validity of a robot movement.
  - If no robot moves in a given time step, then no robot moves in the next time step.

Encodings - CP

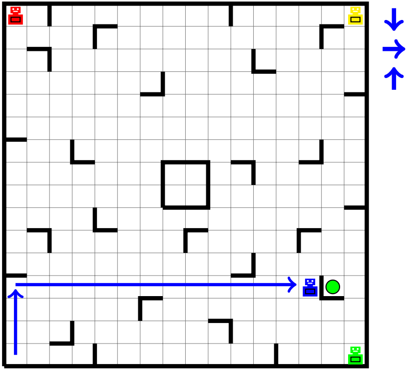
Alternative Encoding

- Inspired in the ASP encodings.
- The main difference from proposed CP encoding is the movement constraints
  - Predicate `valid_movement` is dropped
    - No longer considering all the possible combinations.
  - Movement of the robot is inferred recursively.

Encodings - CP

Alternative Encoding

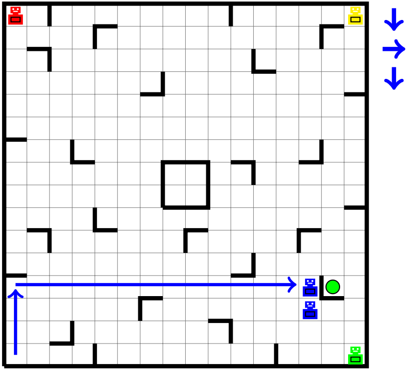
- Inspired in the ASP encodings.
- The main difference from proposed CP encoding is the movement constraints
  - Predicate `valid_movement` is dropped
    - No longer considering all the possible combinations.
  - Movement of the robot is inferred recursively.



Encodings - CP

Alternative Encoding

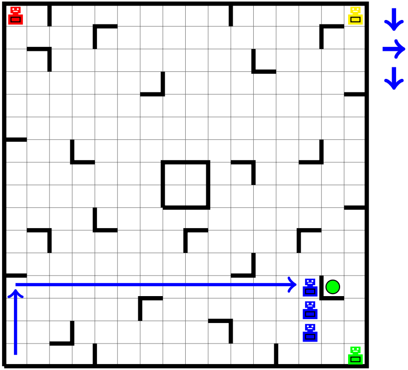
- Inspired in the ASP encodings.
- The main difference from proposed CP encoding is the movement constraints
  - Predicate `valid_movement` is dropped
    - No longer considering all the possible combinations.
  - Movement of the robot is inferred recursively.



Encodings - CP

Alternative Encoding

- Inspired in the ASP encodings.
- The main difference from proposed CP encoding is the movement constraints
  - Predicate `valid_movement` is dropped
    - No longer considering all the possible combinations.
  - Movement of the robot is inferred recursively.



Encodings - CP

Alternative Encoding

- Inspired in the ASP encodings.
- The main difference from proposed CP encoding is the movement constraints
  - Predicate `valid_movement` is dropped
    - No longer considering all the possible combinations.
  - Movement of the robot is inferred recursively.