Self-stabilizing Spanning Tree

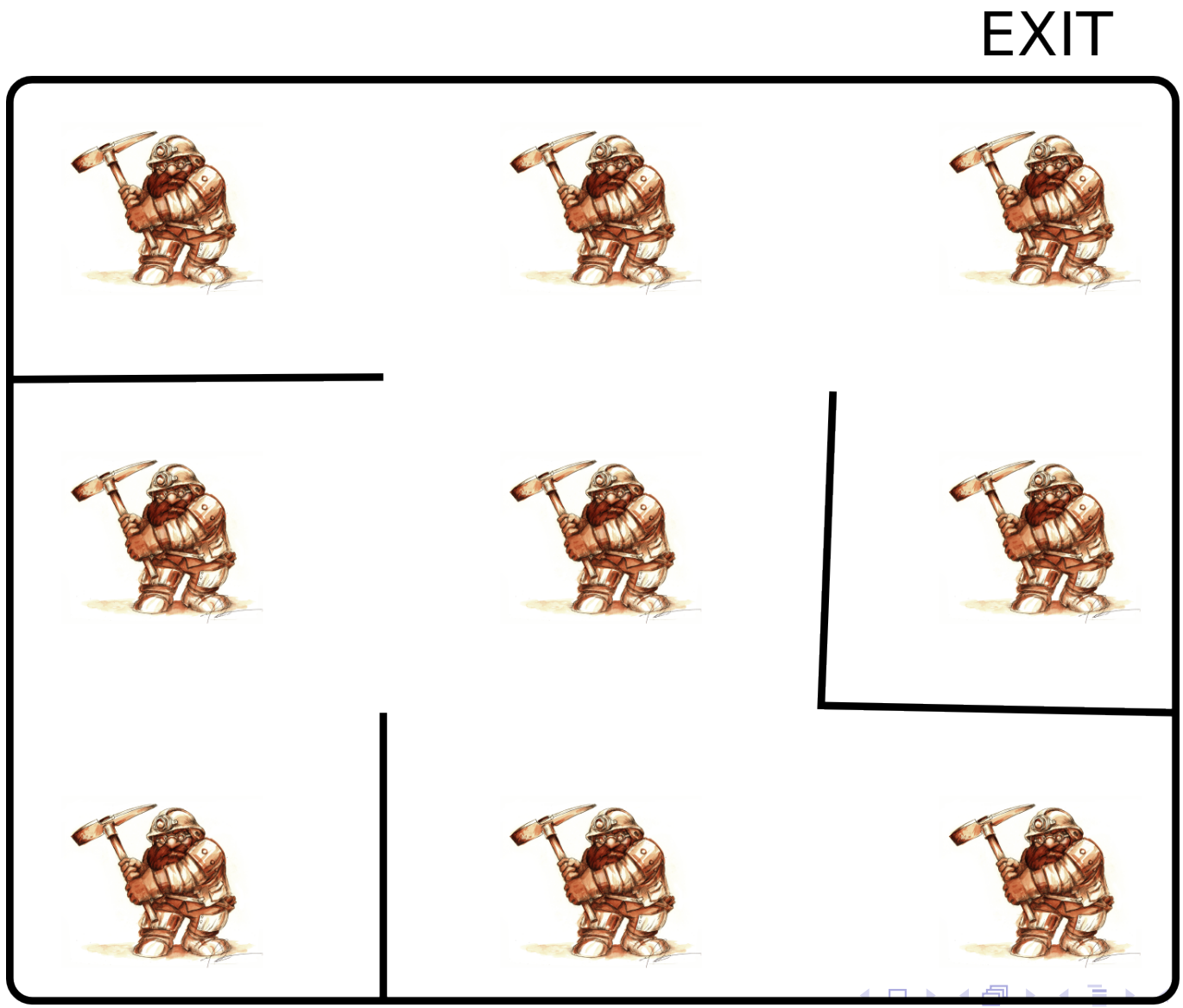
Peva BLANCHARD

- 1. The problem
- 2. The algorithm
- 3. Proof
  - a. Stability
  - b. Convergence
- 4. Conclusion

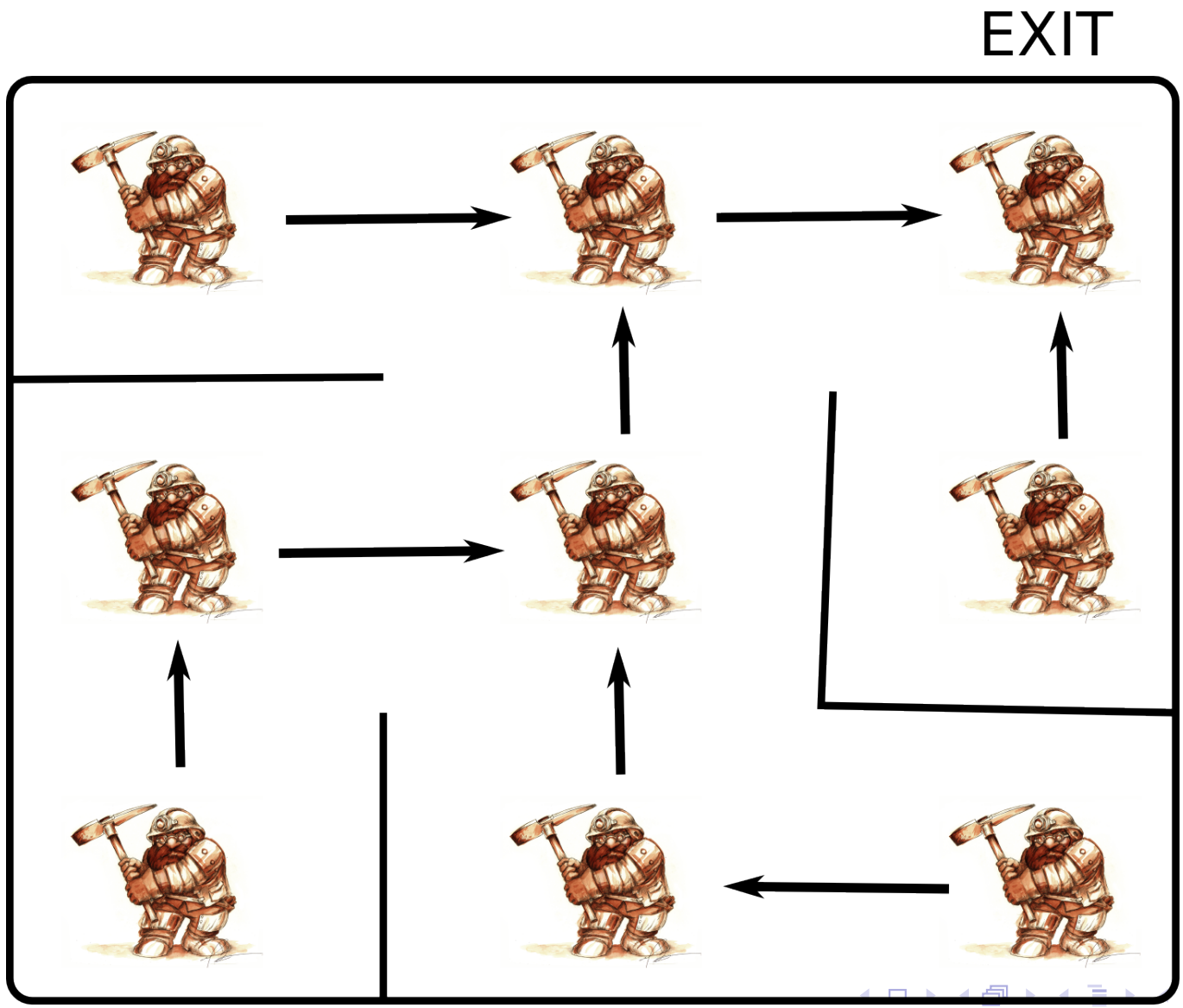
- 1. The problem
- 2. The algorithm
- 3. Proof
  - a. Stability
  - b. Convergence
- 4. Conclusion

Spanning-tree

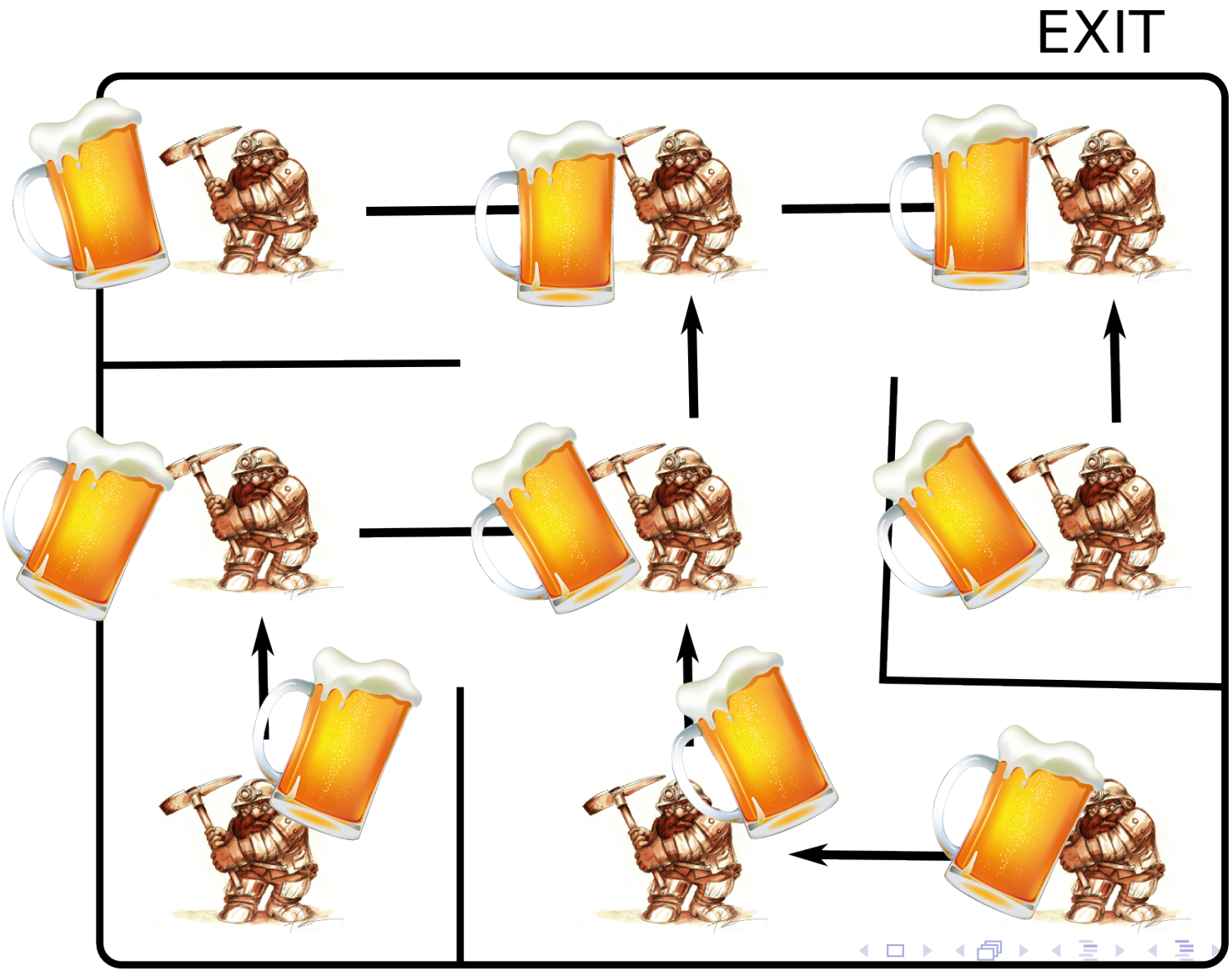
Remember ?



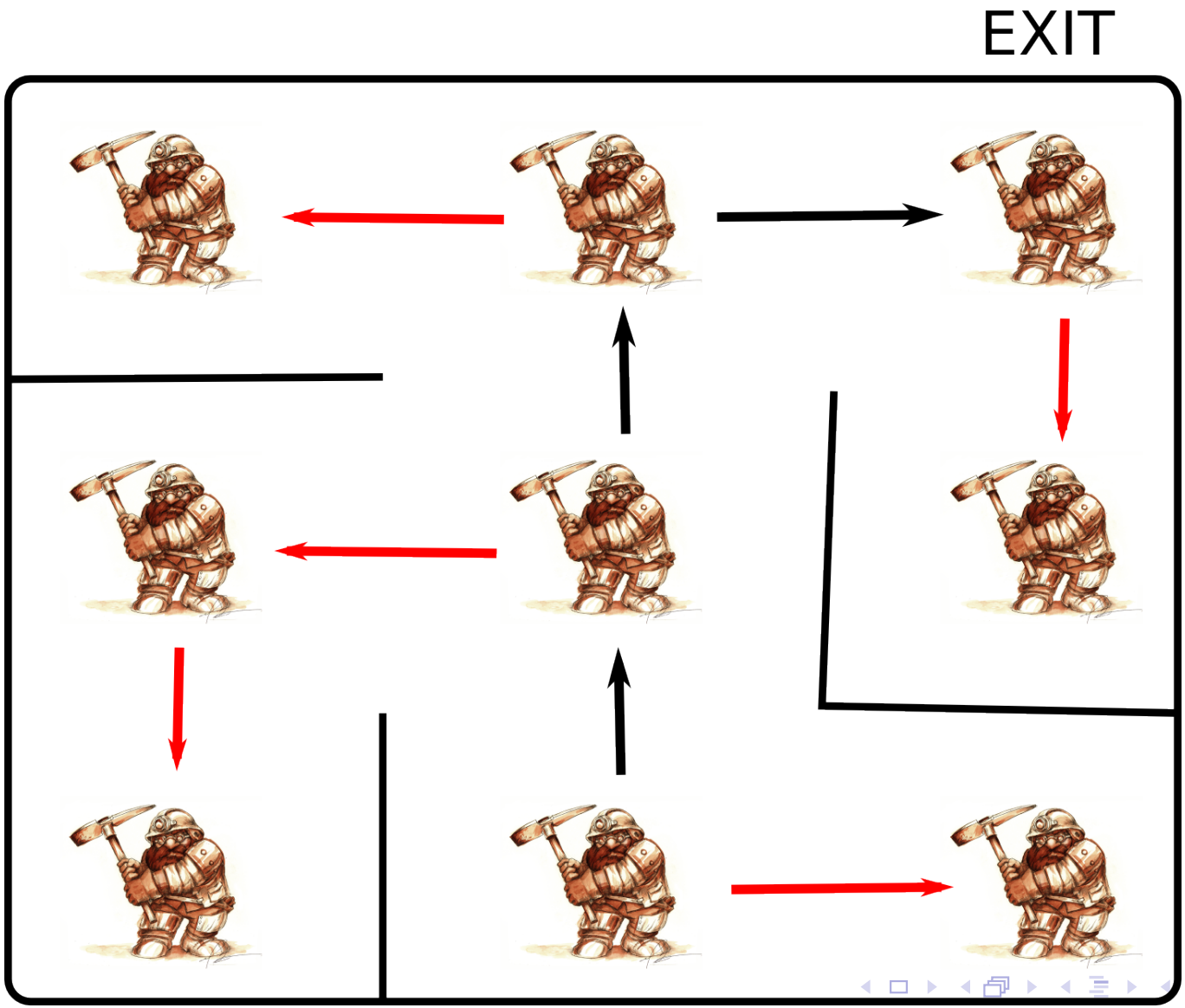
Remember ?



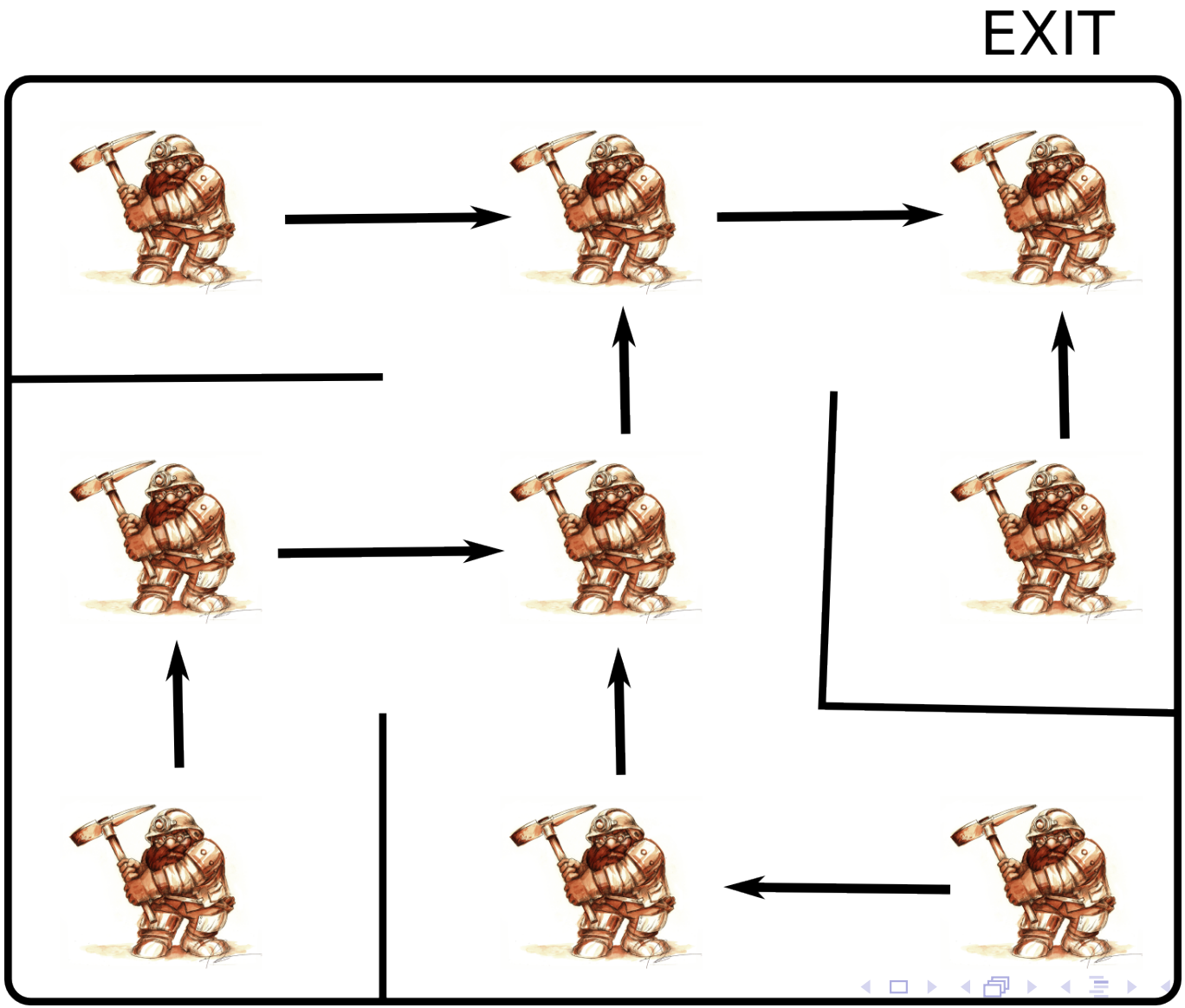
Self-stabilizing



Self-stabilizing

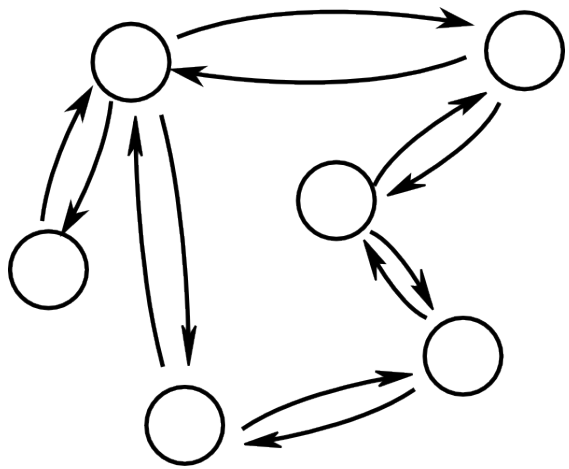


Self-stabilizing



Model

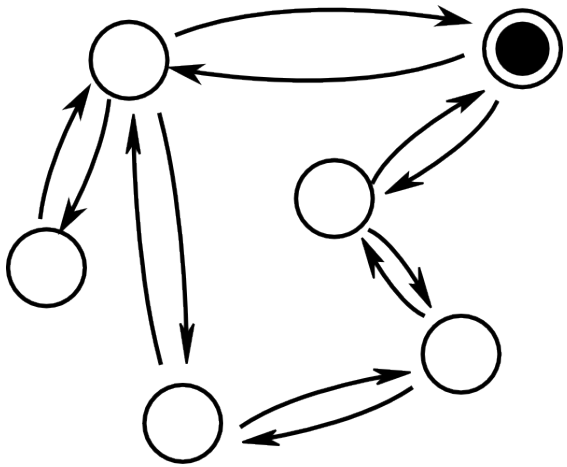
- nodes = processes $p_0, \ldots, p_{n-1}$
- *directed* edge (p_i, p_j) : atomic RW register r_{ij}
- p_i writes to r_{ij}
- p_i reads from r_{mi}



Model

Assumptions

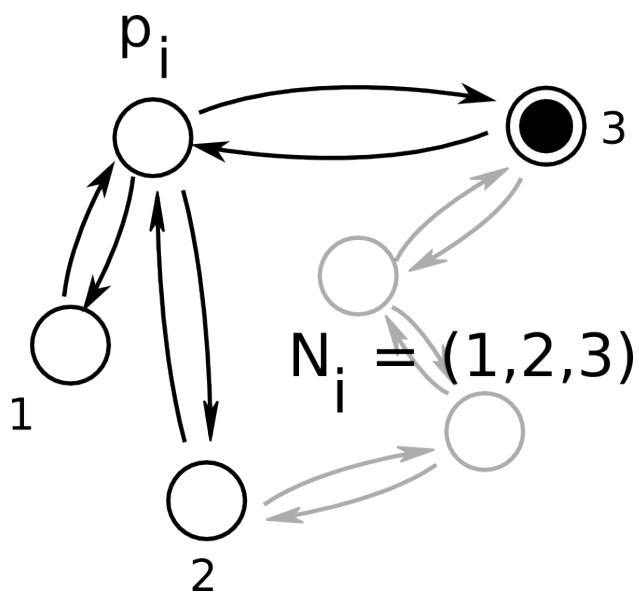
- p_0 is a distinguished processor (root) and knows it



Model

Assumptions

- p_0 is a distinguished processor (root) and knows it
- each p_i knows an ordered list N_i of its neighbours



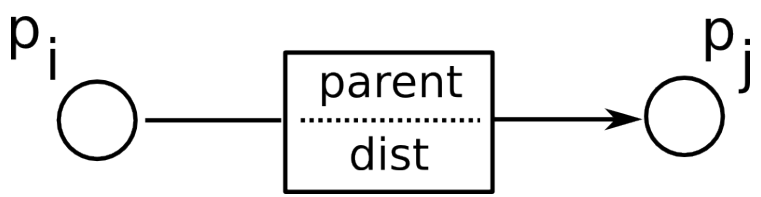
Model

Register r_{ij}

- boolean field *parent* :

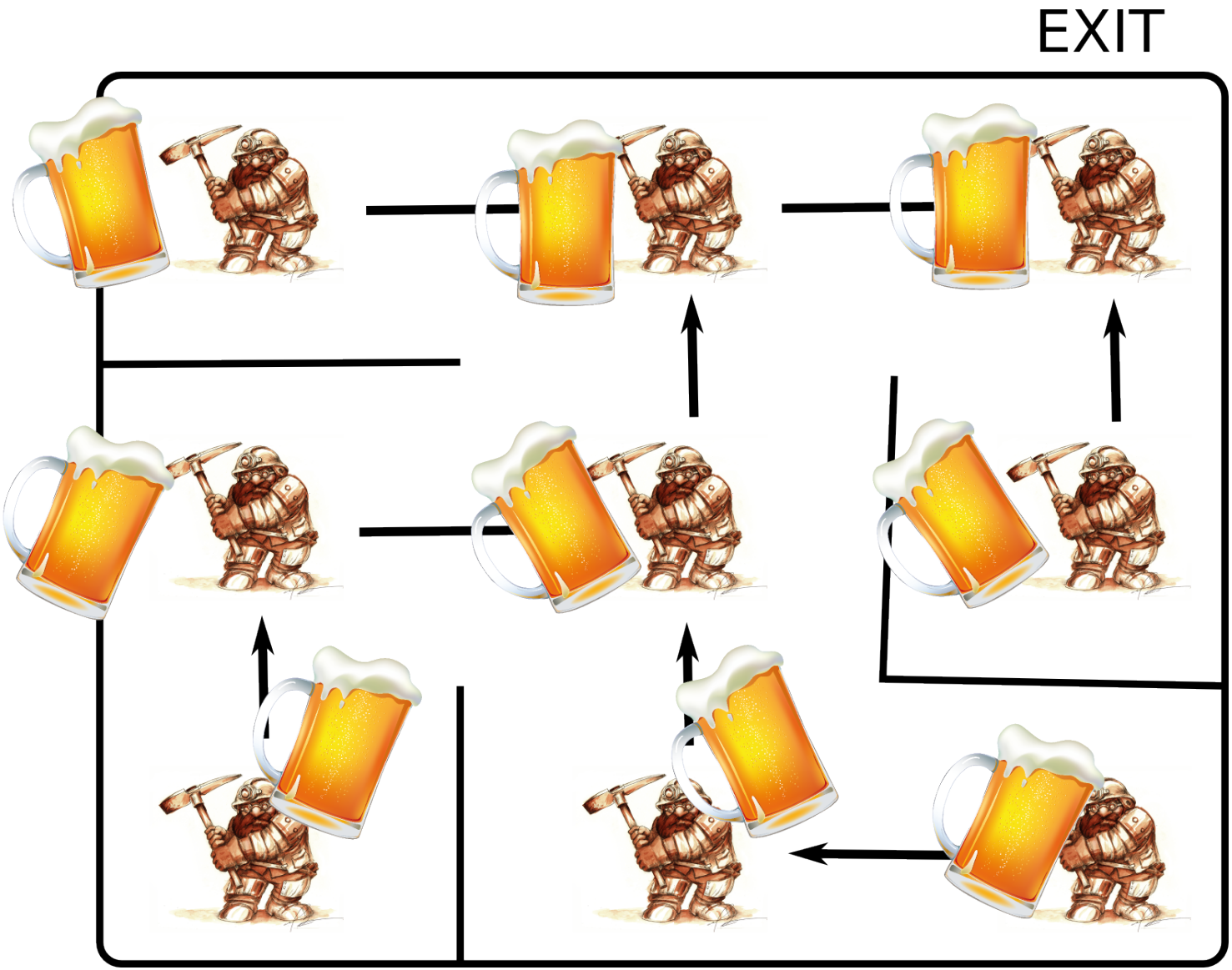
$$p_i \text{ points to } p_j \iff r_{ij}.parent = 1$$

- integer field *dist* : distance from the root p_0 to p_i.

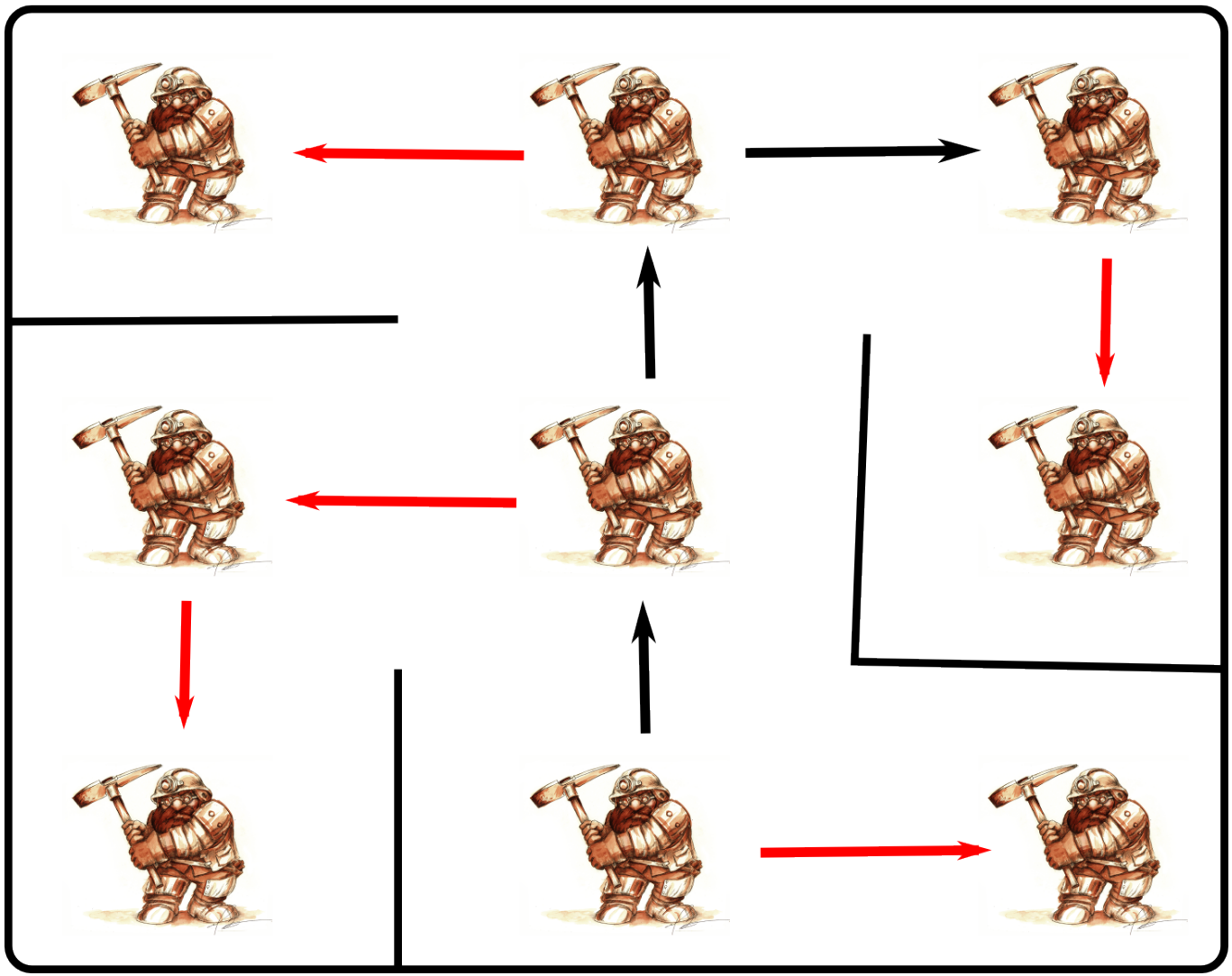


- 1. The problem
- 2. The algorithm
- 3. Proof
  - a. Stability
  - b. Convergence
- 4. Conclusion

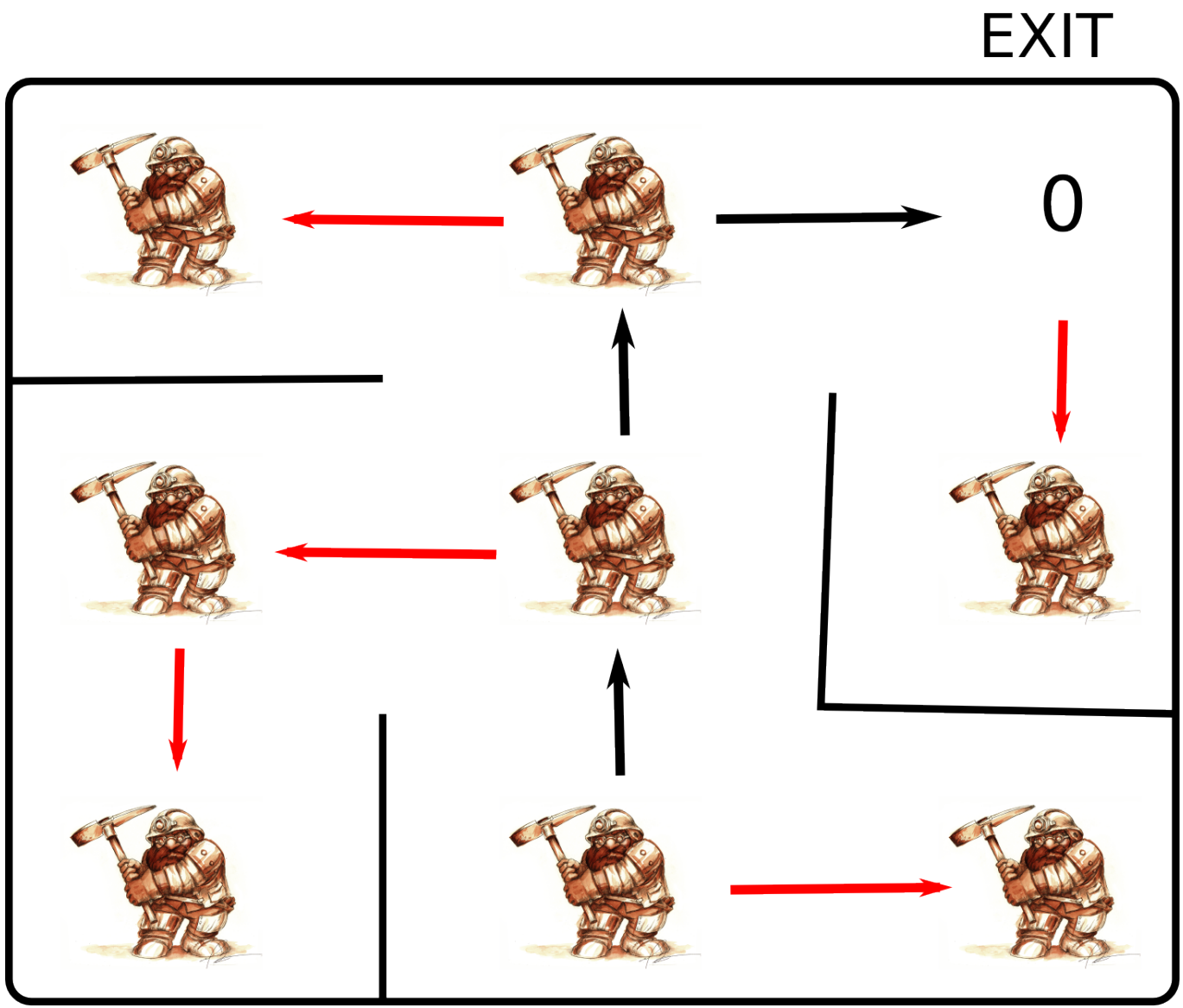
The algorithm



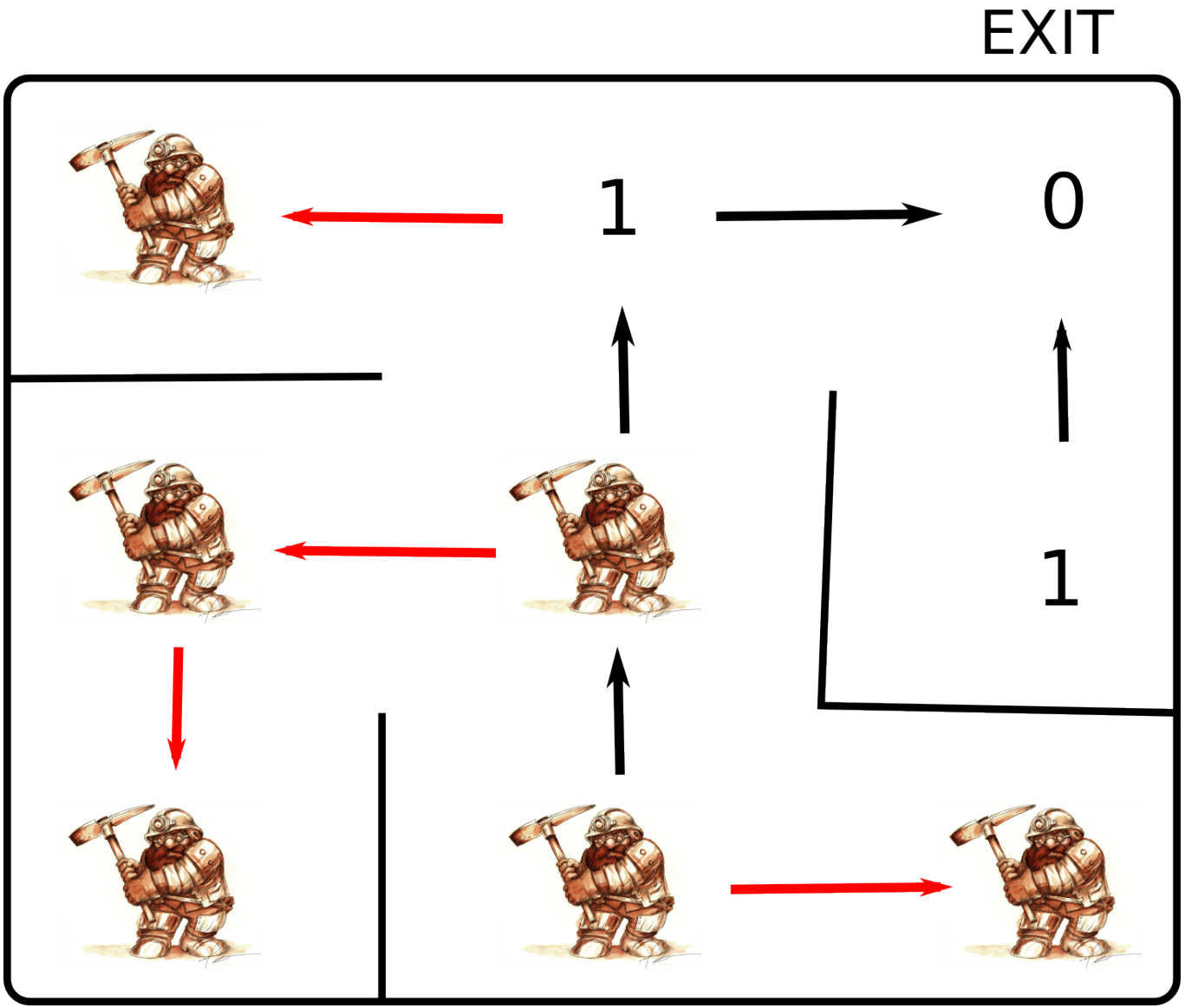
The algorithm



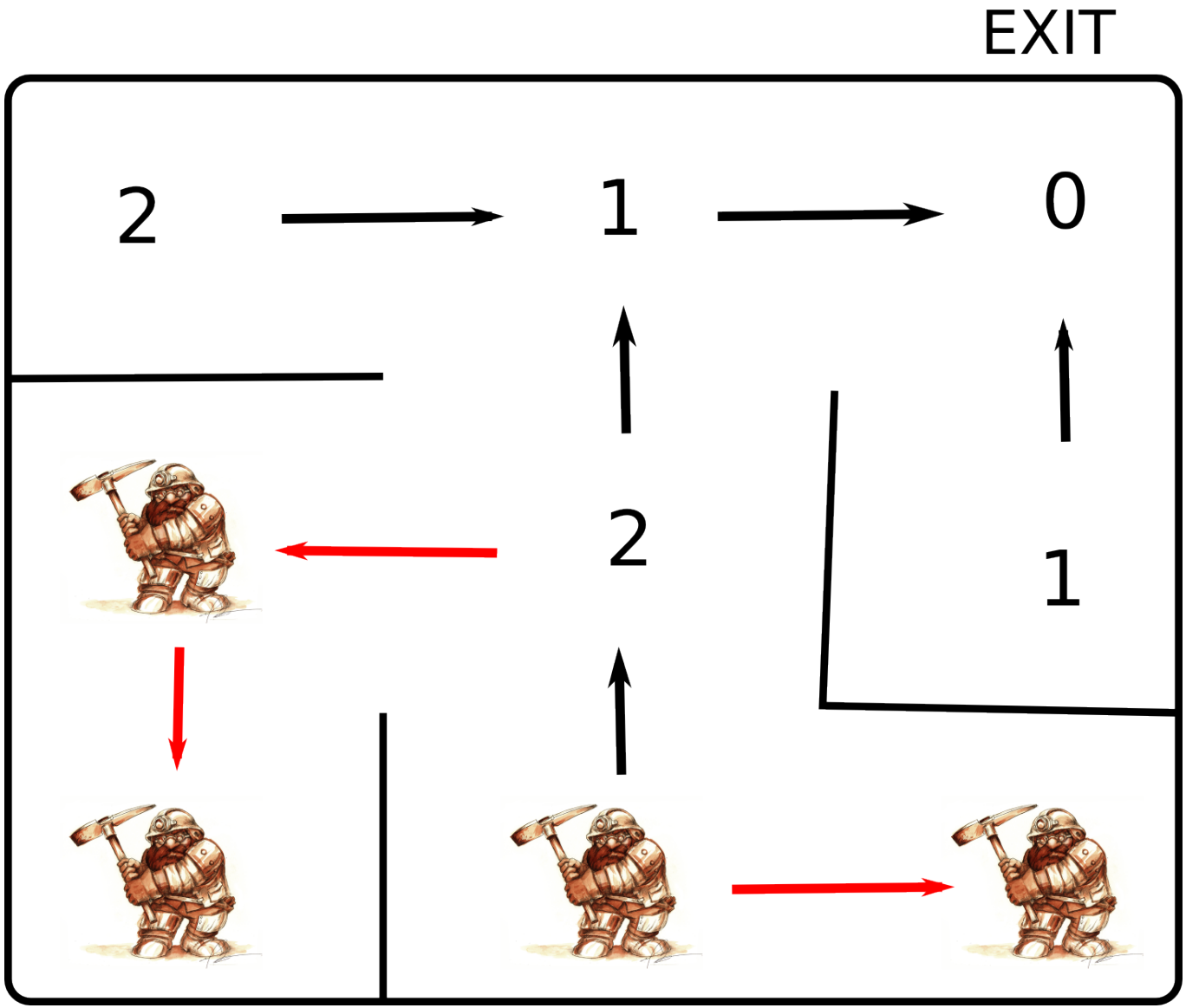
The algorithm



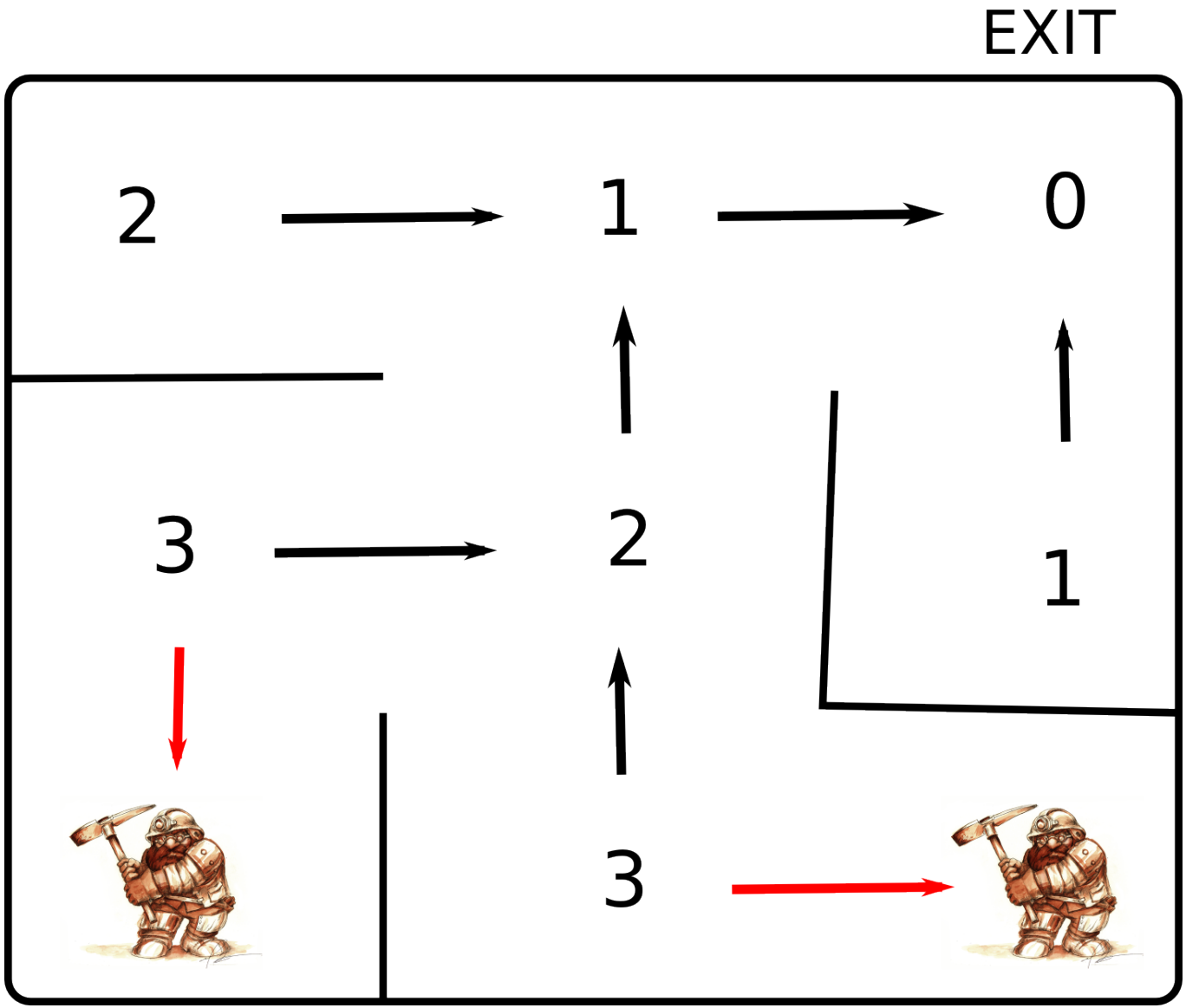
The algorithm



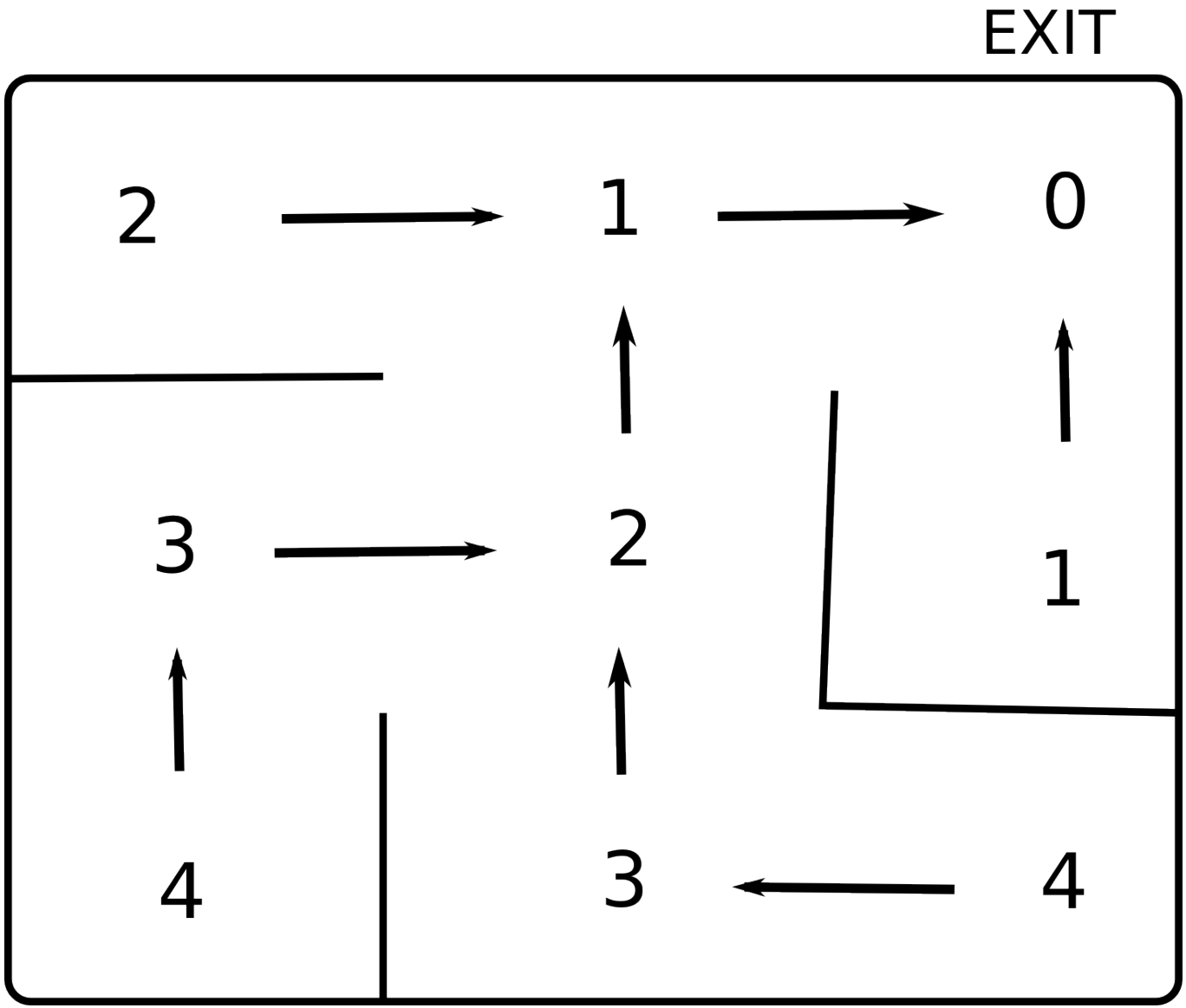
The algorithm



The algorithm



The algorithm

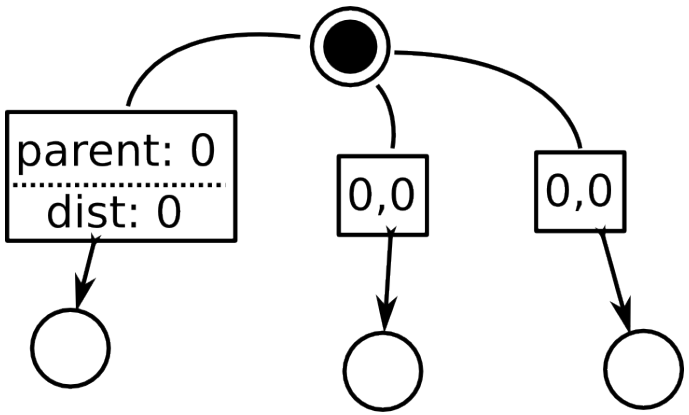


The algorithm

See blackboard (or exercice sheet)

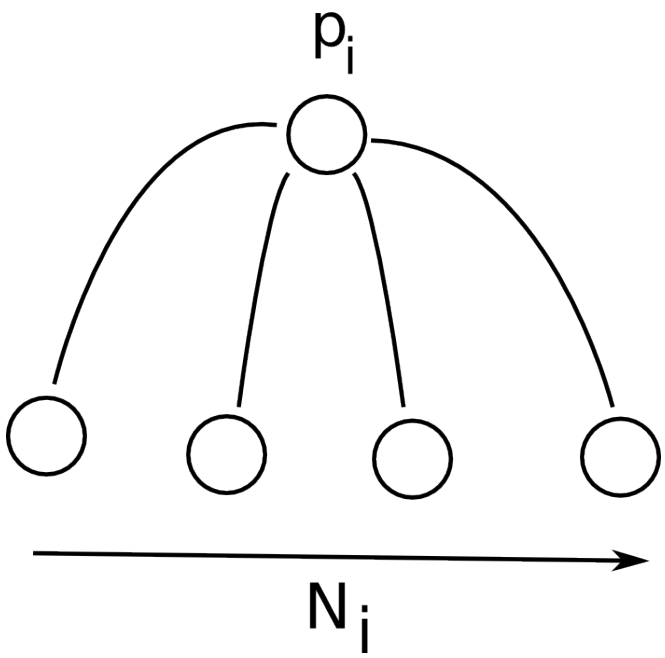
The algorithm

Root p_0



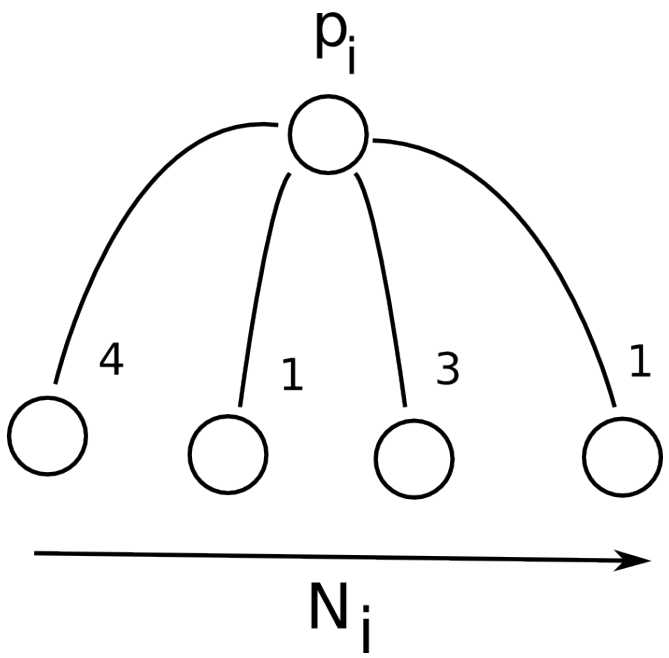
The algorithm

Non-root p_i



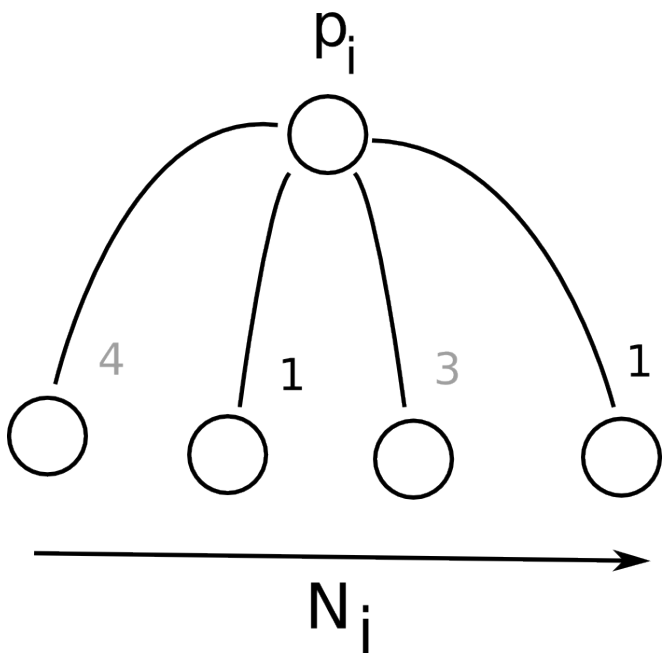
The algorithm

Non-root p_i



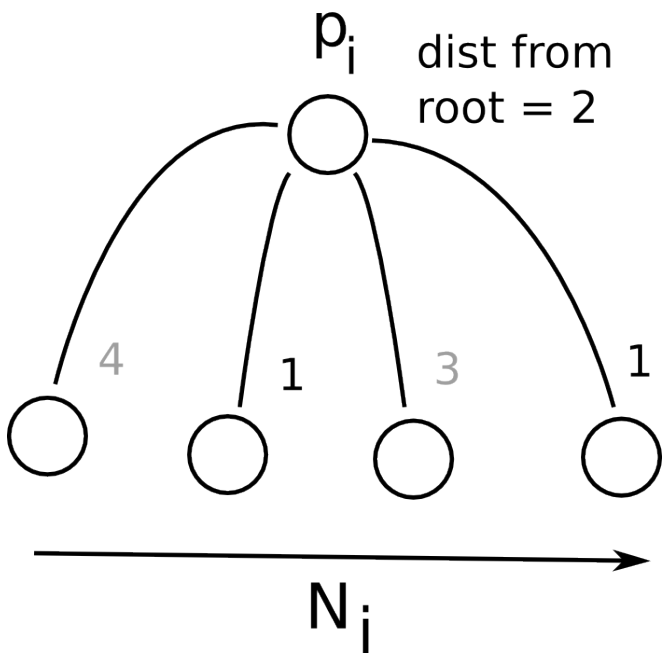
The algorithm

Non-root p_i



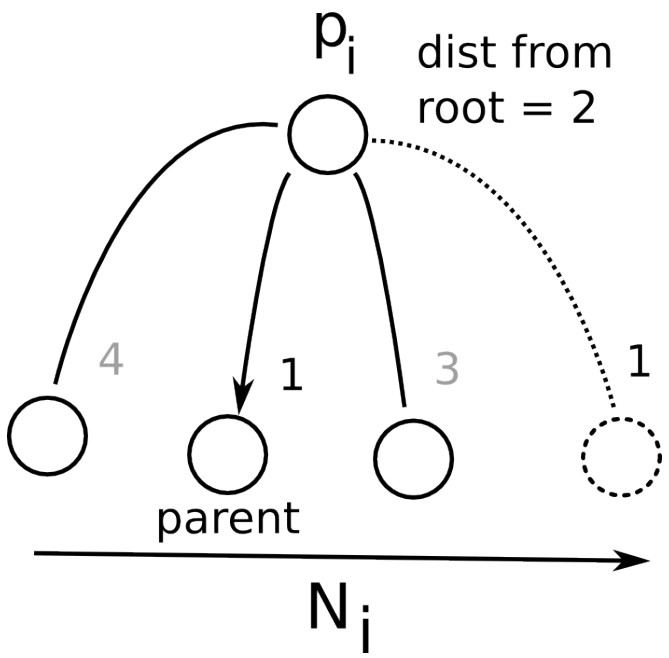
The algorithm

Non-root p_i



The algorithm

Non-root p_i

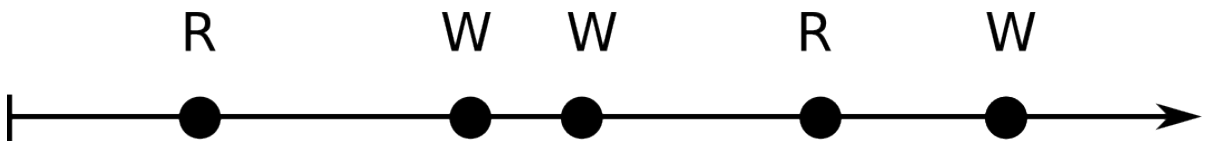


- 1. The problem
- 2. The algorithm
- 3. Proof
  - a. Stability
  - b. Convergence
- 4. Conclusion

Proof

Step at process p_i

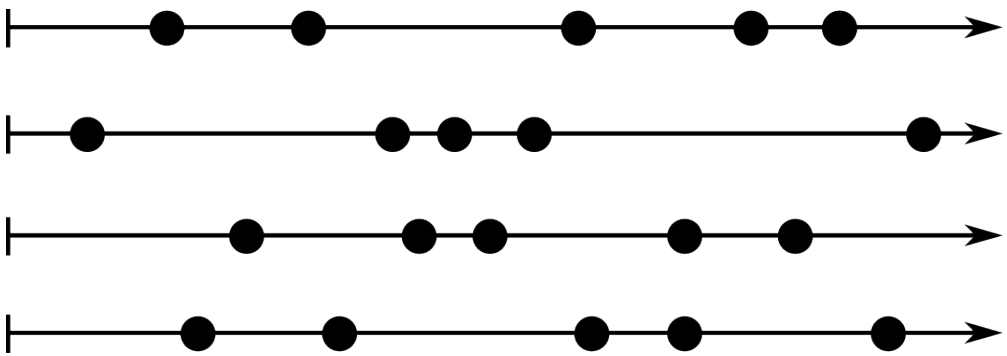
- sequence of instructions until the next access to some register (read or write)
- considered atomic here



Proof

Execution

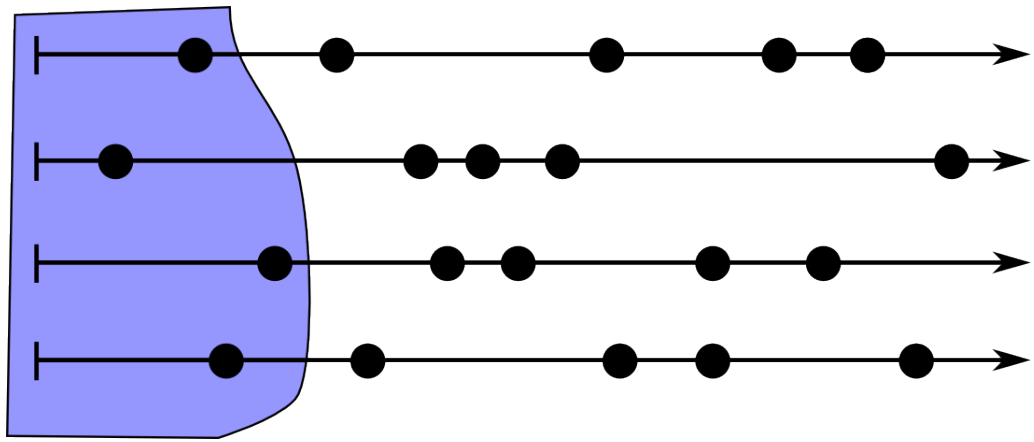
- (infinite) interleaving of steps of processes
- fairness assumption : every process takes infinitely many steps



Proof

Asynchronous round

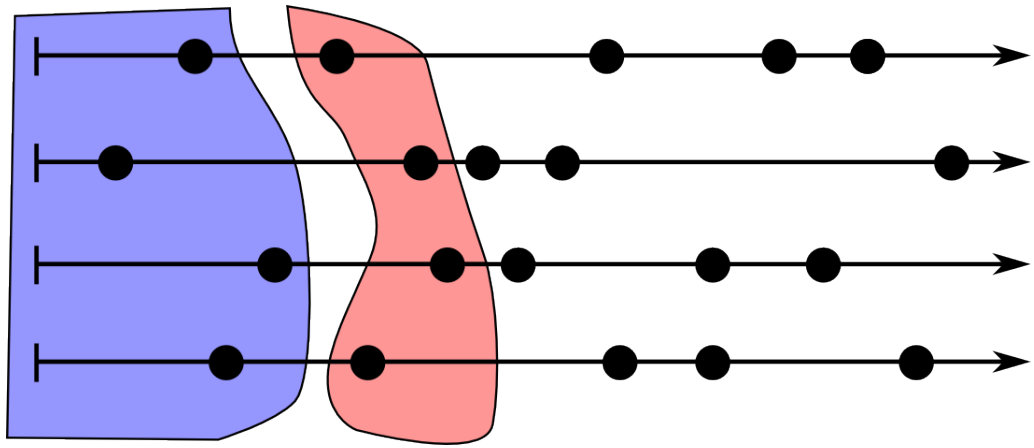
- round 1 : smallest prefix in which each process takes at least one step. and so on



Proof

Asynchronous round

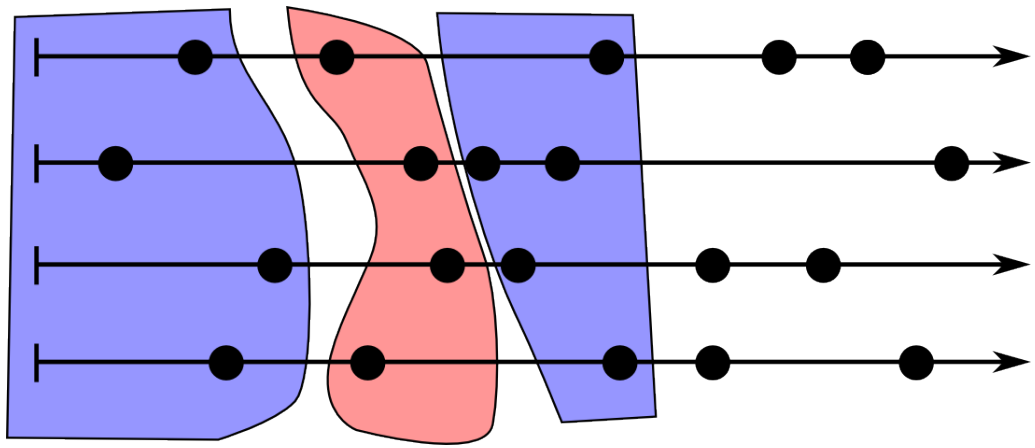
- round 1 : smallest prefix in which each process takes at least one step.
- round 2 : next segment in which each process takes at least one step and so on



Proof

Asynchronous round

- round 1 : smallest prefix in which each process takes at least one step.
- round 2 : next segment in which each process takes at least one step
- and so on



- 1. The problem
- 2. The algorithm
- 3. Proof
  - a. Stability
  - b. Convergence
- 4. Conclusion

Proof - Stability

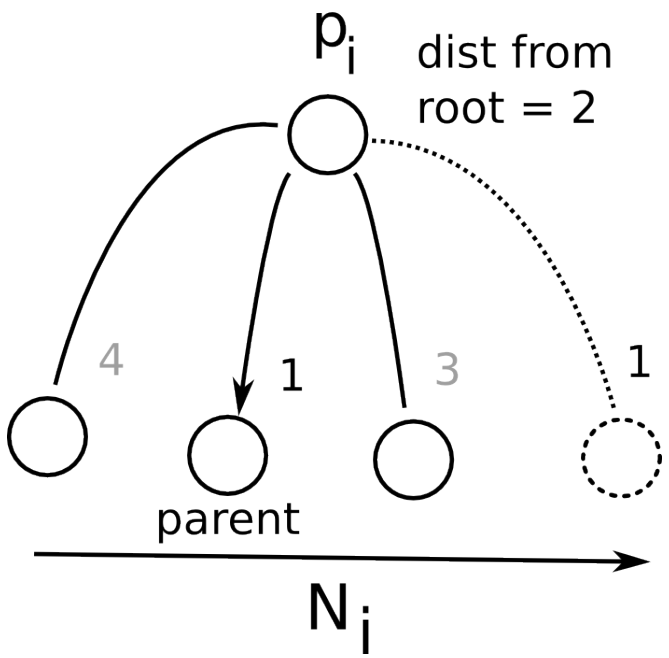
Configuration

- for each p_i, values of local variables lr_{mi}, F, $dist$
- register values r_{ij} for all p_i,p_j
- program counter : each process may not start at the beginning of the pseudo-code !

Proof - Stability

Legitimate configuration

- encodes a spanning tree rooted at p_0
- distance values in each r_{ij} is correct
- the parent p_j of p_i is the first process in N_i with minimal distance



Proof - Stability

- no process is inclined to change parent or distance
- the set of legitimate configurations is stable

- 1. The problem
- 2. The algorithm
- 3. Proof
  - a. Stability
  - b. Convergence
- 4. Conclusion

Proof - Convergence

Strategy : from an arbitrary initial configuration

- Each register r_{ij} eventually holds the correct distance from the root to p_i
- Then, each process p_i selects the first valid parent in $N_i \Rightarrow$ this yields a legitimate configuration

Proof - Convergence

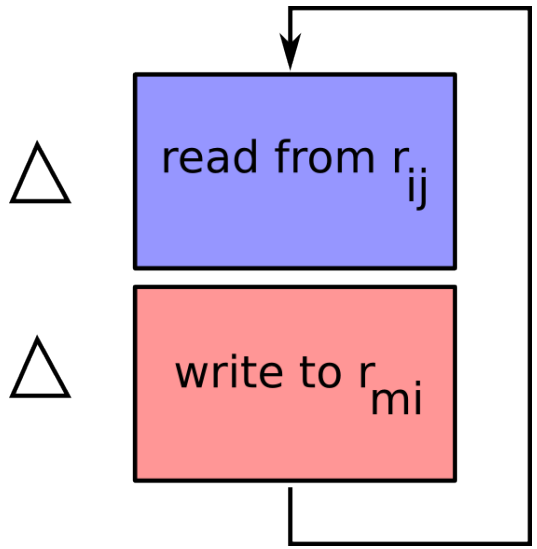
Strategy : from an arbitrary initial configuration

- Each register r_{ij} eventually holds the correct distance from the root to p_i
- Then, each process p_i selects the first valid parent in $N_i \Rightarrow$ this yields a legitimate configuration

Proof - Convergence

Lemma

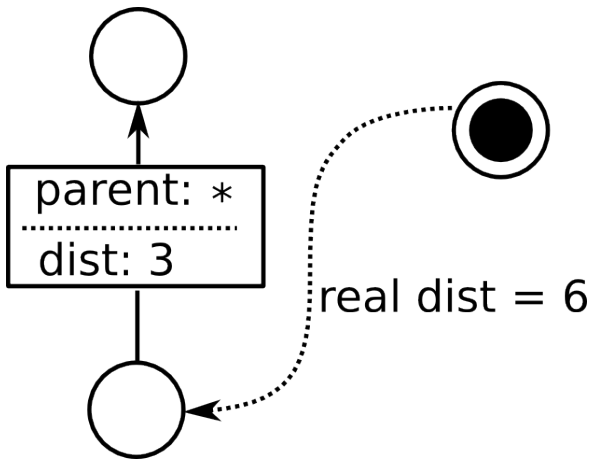
Let Δ = max. nb of neighbours. In every 2Δ successive rounds, each $p_i \neq p_0$ performs "one loop in the pseudo-code".



Proof - Convergence

Definition

A *floating distance* in configuration C is a value in some field $r_{ij}.dis$ that is smaller than the distance from the root to p_i.



Proof - Convergence

Lemma

Let E_k be the suffix of execution after the first $\Delta + 4k\Delta$ rounds.

- *(Small(k)) For any $C \in E_k$, if C has a floating distance, then the smallest floating distance in C is $\geq k$*
- *(Dist(k)) For any $C \in E_k$, the distance values in registers of processes within distance k from the root are correct*

Proof - Convergence

By induction : E_1 after $\Delta + 4\Delta$ rounds ($k = 1$)

- each distance field is ≥ 0 in the first (arbitrary) configuration
- Proc $p_i \neq p_0$
  - during the first 2Δ rounds, each non-root p_i computes *dist* : $dist \geq 1$
  - in the next 2Δ rounds, each non-root p_i writes *dist* to its registers r_{ij}

$\Rightarrow$ afterwards, always $r_{ij} \geq 1$ for all j. In particular, Small(1) holds

Proof - Convergence

By induction : E_1 after $\Delta + 4\Delta$ rounds ($k = 1$)

- each distance field is ≥ 0 in the first (arbitrary) configuration
- root p_0
  - first Δ rounds : p_0 writes 0 in every r_{0j}
  - in the next 2Δ rounds, each root's neighbour p_j reads 0 in r_{0j}
  - in the next 2Δ rounds, each root's neighbour p_j writes 1 to their registers distance fields.

  $\Rightarrow$ Dist(1) holds

Proof - Convergence

Assume $Small(k)$ and $Dist(k)$ hold in E_k.

- Let C_k first config of E_k
- $m \geq k$ smallest floating distance in C_k
- In the next 4Δ rounds after C_k, each proc that chooses m as the smallest value assigns $m+1$ to its distance
- Thus, afterwards, smallest floating distance $\geq m+1 \geq k+1$

$\Rightarrow$ Small(k+1) holds in E_{k+1}

Proof - Convergence

Assume $Small(k)$ and $Dist(k)$ hold in E_k.

- Dist(k) holds in E_k : every proc within distance k from the root have registers with correct distance
- Let p_i be proc at distance $k+1$
- In the next 4Δ rounds after C_k : p_i necessarily chooses value k, and assigns $k+1$ to its register distance fields.

$\Rightarrow$ Dist(k+1) holds in E_{k+1}

- 1. The problem
- 2. The algorithm
- 3. Proof
  - a. Stability
  - b. Convergence
- 4. Conclusion

Conclusion

Assumptions

- n processes, with a root p_0
- bidirectional communication with rw register r_{ij}
- each p_i knows a an ordered list of its neighbours
- each process takes infinitely many steps

$\Rightarrow$ the algorithm is self-stabilizing