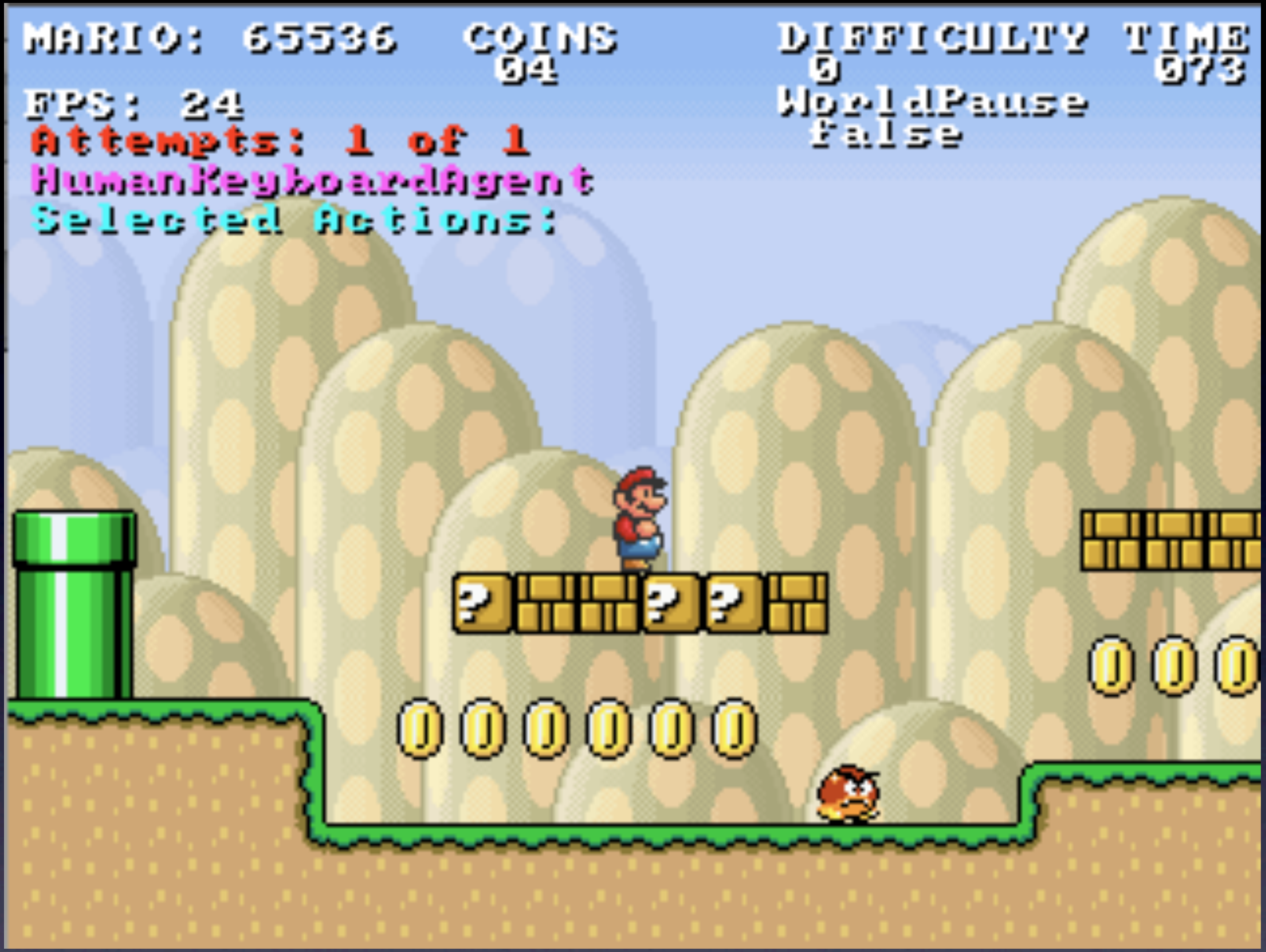
Mario AI Competition @ ICE-GIC 2009

Sergey Karakovskiy and Julian Togelius



Develop a controller/agent
(based on AI/machine learning?)
for "Super Mario Bros"

Infinite Mario Bros

- by Markus Persson
- quite faithful SMB 1/3 clone
- in Java
- random level generation
- open source

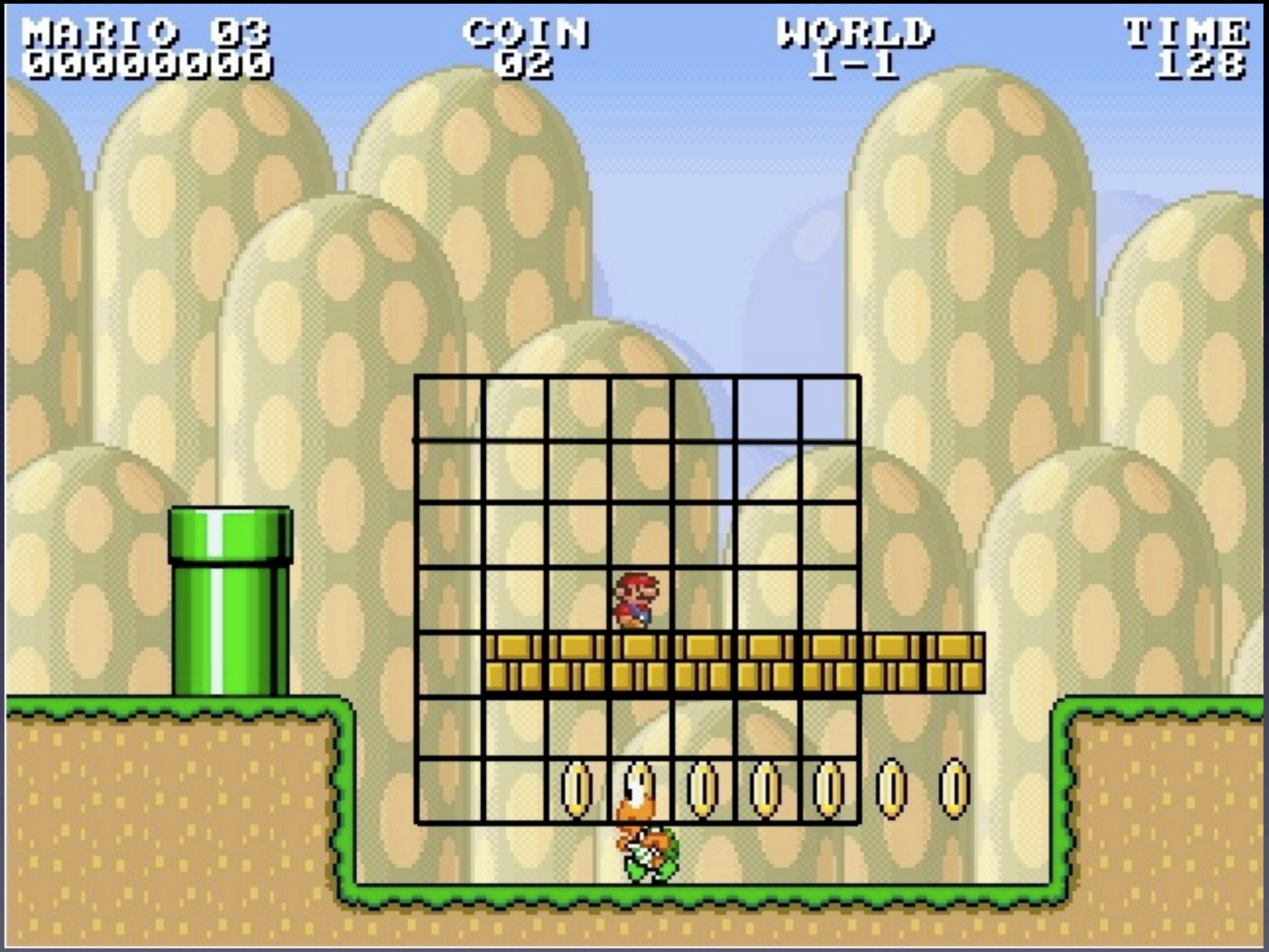
Our changes

- Rewrite the control loop
- Allow for 1000 times speed-up in headless mode
- Create an interface for controllers

Interface

- Each time step (24 fps), the agent gets a representation of the environment
  - Enemies and "blocks" around Mario
  - Fine position, jumping state
- And returns an action
  - 5 bits: left, right, down, A, B

Interface



Interface

Environment.java

```
// always the same dimensionality 22x22
// always centered on the agent
public byte[][] getCompleteObservation();
public byte[][] getEnemiesObservation();
public byte[][] getLevelSceneObservation();
public float[] getMarioFloatPos();
public float[] getEnemiesFloatPos();
public boolean isMarioOnGround();
public boolean mayMarioJump();
```

Agent.java

```
public enum AGENT_TYPE
       {AI, HUMAN, TCP_SERVER}
public void reset();
public boolean[] getAction
       (Environment observation);
public AGENT_TYPE getType();
public String getName();
public void setName(String name);
```

A very simple agent

```
public boolean[] getAction(Environment
observation) {

action[Mario.KEY_SPEED] =
action[Mario.KEY_JUMP] =
observation.mayMarioJump() || !
observation.isMarioOnGround();

return action;}
```

Neural network agent

```
for (int i = -3; i < 4; i++) {
            for (int j = -3; j < 4; j++) {
                inputs[which++] = probe(i, j, scene);}}
inputs[inputs.length - 3] =
    observation.isMarioOnGround() ? 1 : 0;
inputs[inputs.length - 2] =
    observation.mayMarioJump() ? 1 : 0;
inputs[inputs.length - 1] = 1;
double[] outputs = mlp.propagate (inputs);
for (int i = 0; i < action.length; i++) {
    action[i] = outputs[i] > 0;
return action;
```

Goal of the competition

- Develop an agent that gets as far as possible...
- ...on as many levels as possible...
- ...which are previously unseen
- Scoring: progress on 40 randomly generated levels

Main rules

- Implement the Agent interface (or connect to the TCPAgent)
- Use <u>only</u> information from the Environment interface
- Don't take more than 40 ms per time step
- Follow the submission instructions...

Challenges

- Handle a large state/observation space
- Handle very different situations (unlike e.g. car racing)
- Tactical tradeoffs (go back and get the power-up?)

What we thought would work

- Rule-based systems, with handcrafted complicated feature detectors
  - To handle the large observation space
- Tuned by e.g. artificial evolution
  - To handle the large parameter space
- Or TD-learning

Presentations of competitors

Robin Baumgarten

Using path-finding to find the optimal jump

AN A* MARIO AI

IDEA

- Analyse Mario's physics engine to obtain movement equations for all objects
- Create our own physics engine that can predict next world state
- Plug engine into an A* algorithm to evaluate fitness of each node
- Heuristic: How long before Mario reaches goal?
- Penalty for falling into gaps or being hurt
- Ignore coins, enemies, power-ups (for now!)

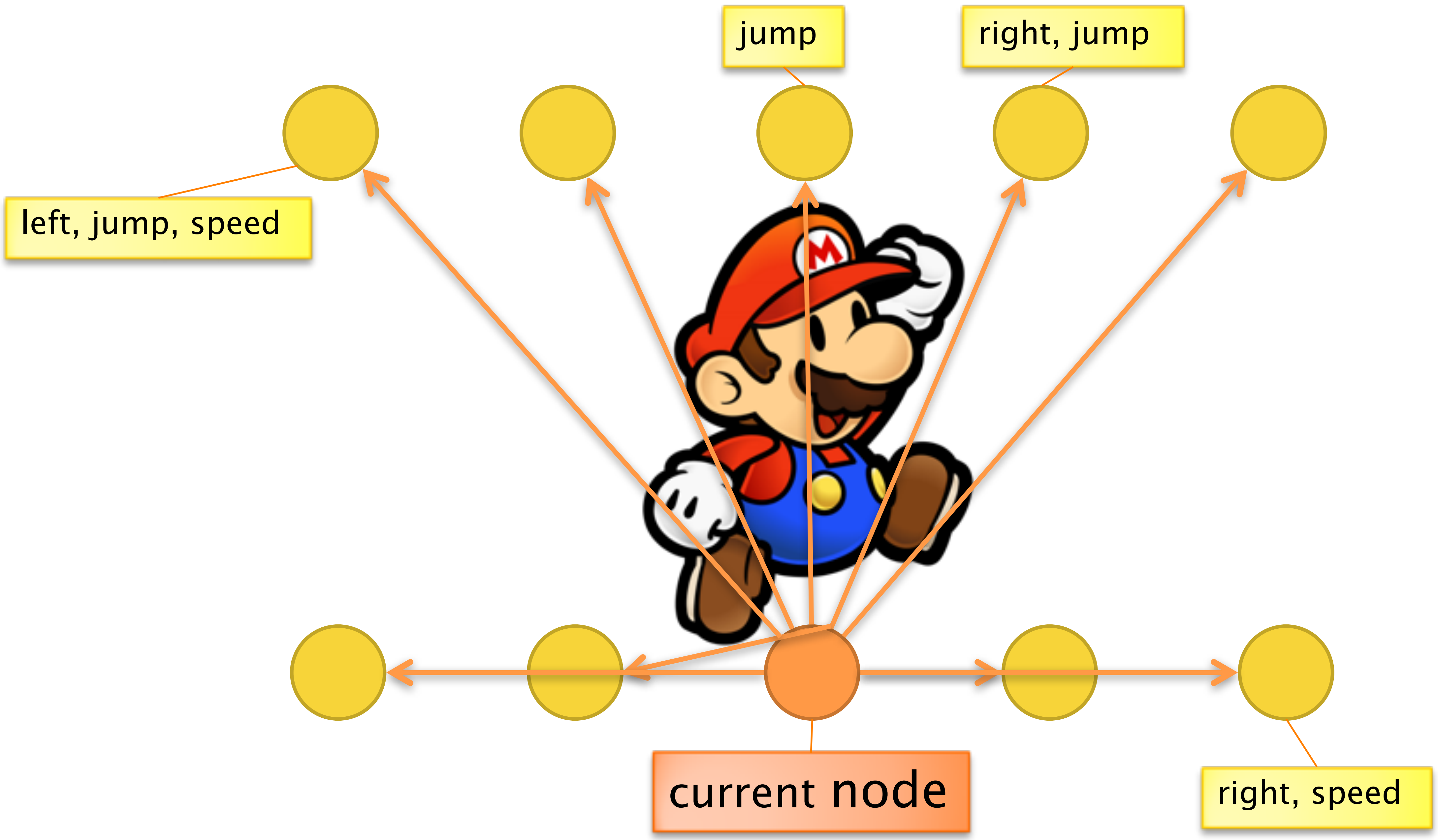
A* ALGORITHM

- Best-first graph search algorithm
- Need heuristic that estimates remaining distance
- Keep set of “open” nodes (initially: start node)
- While open set not empty:
  - Pick node in open set with **lowest estimated total distance from start to goal**
  - If node == goal: finish. Create path by backtracking through ancestors.
  - Generate child nodes, put them into open list (only if better than existing nodes for that location)
- If heuristic admissible (always underestimating), we then have the shortest path to goal.

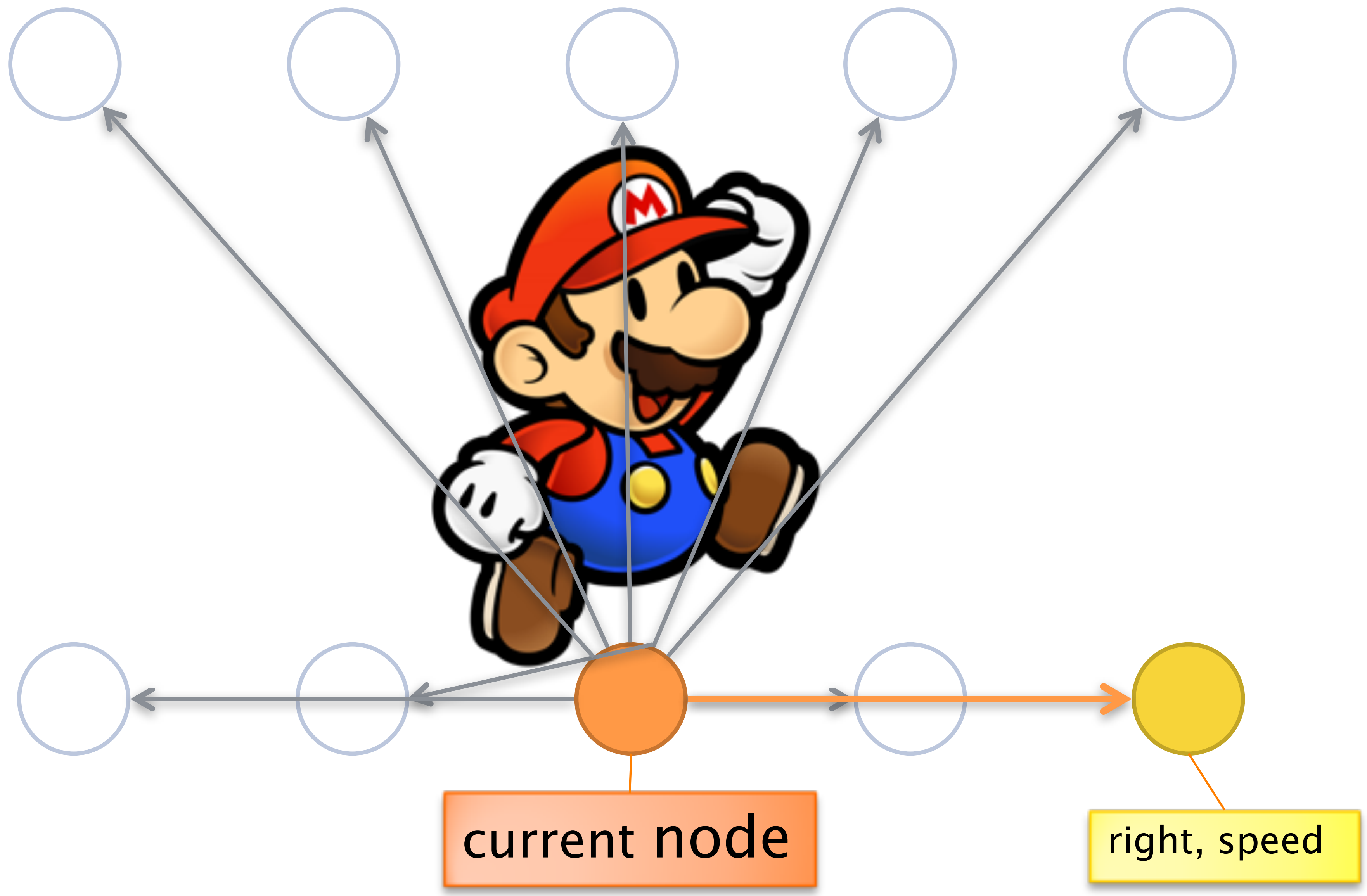
A* IN MARIO: CURRENT POSITION

Goal:
right border
of screen

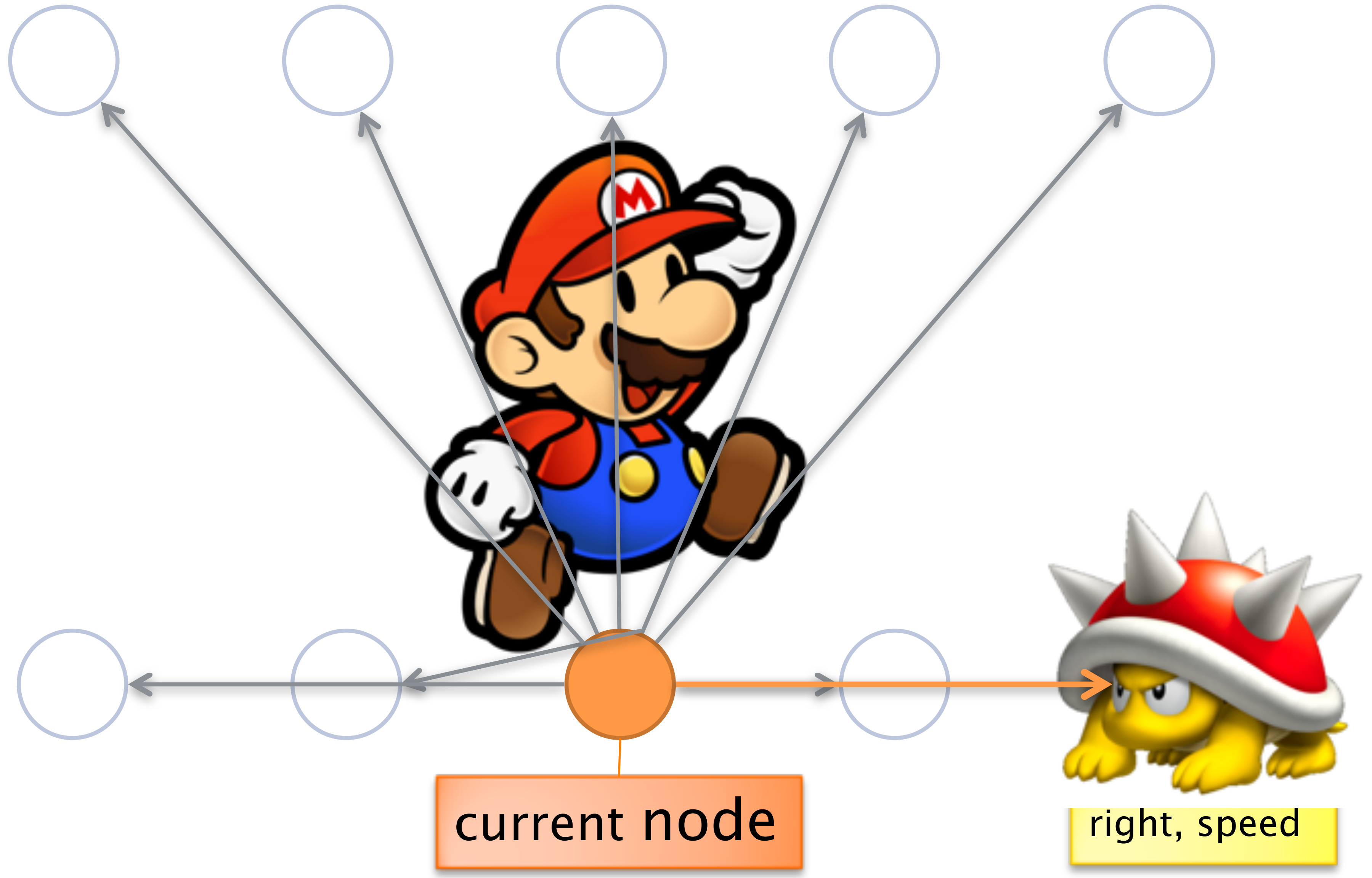
A* IN MARIO: CHILD NODES



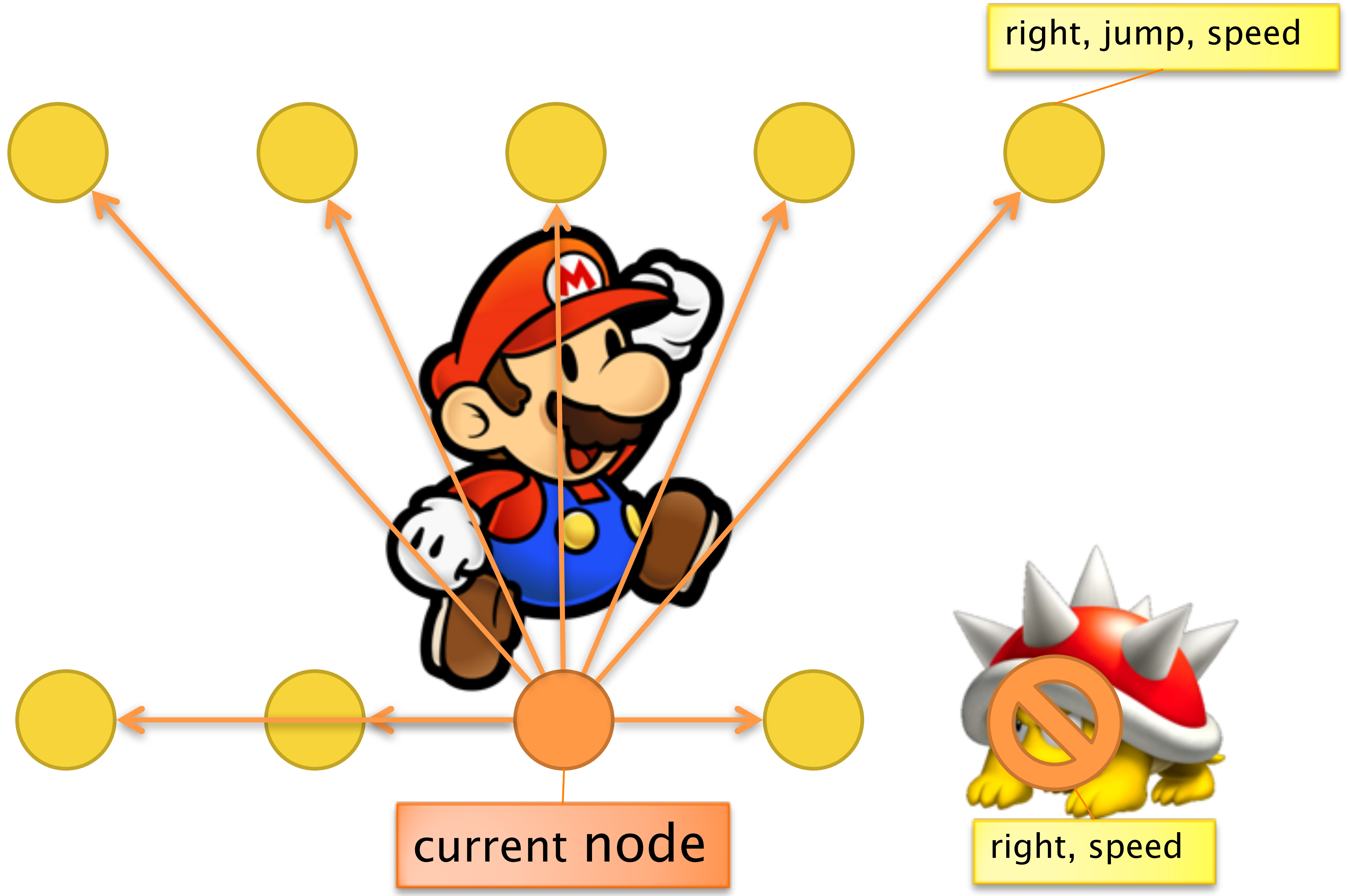
A* IN MARIO: BEST FIRST



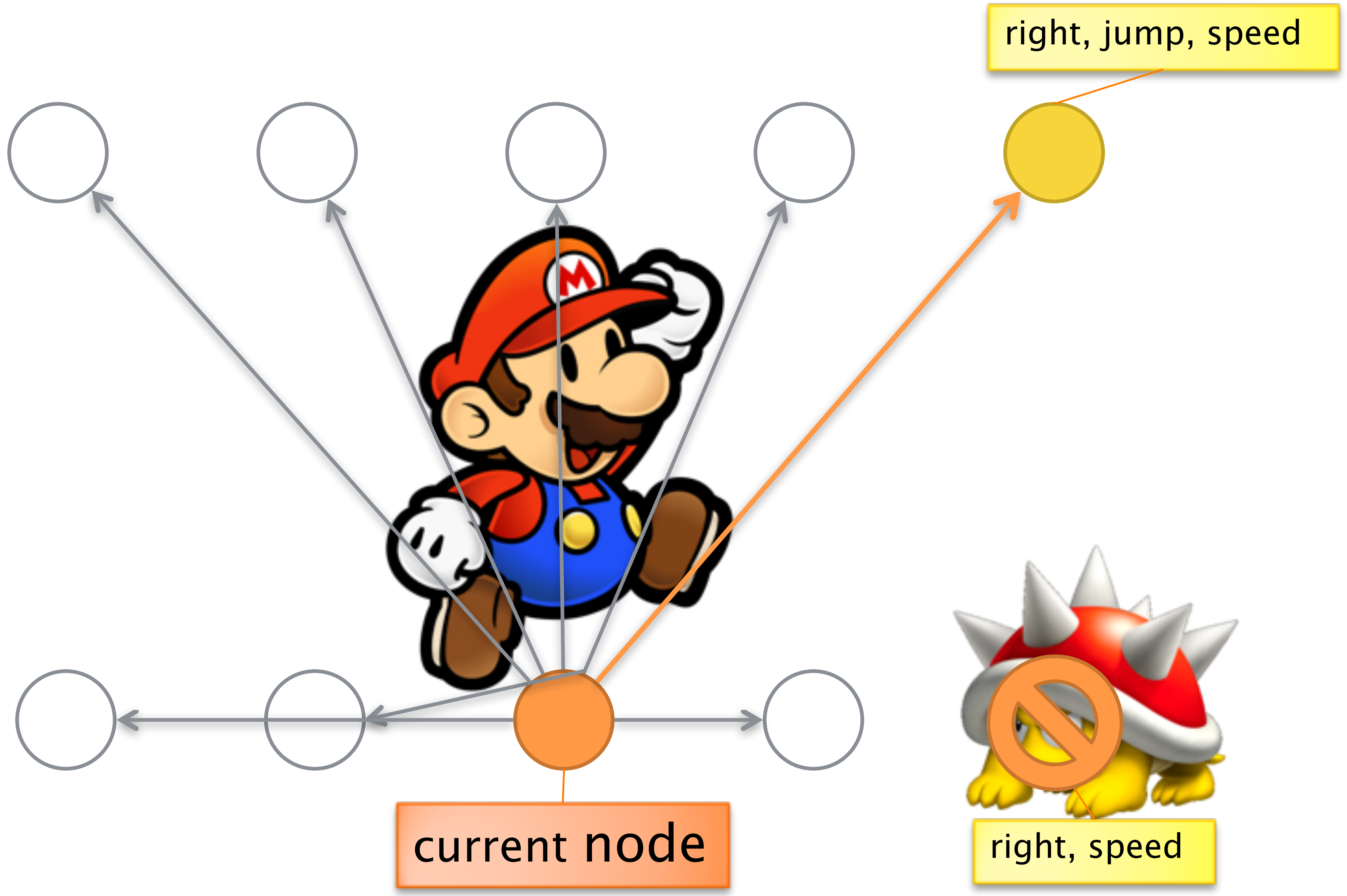
A* IN MARIO: EVALUATE NODE



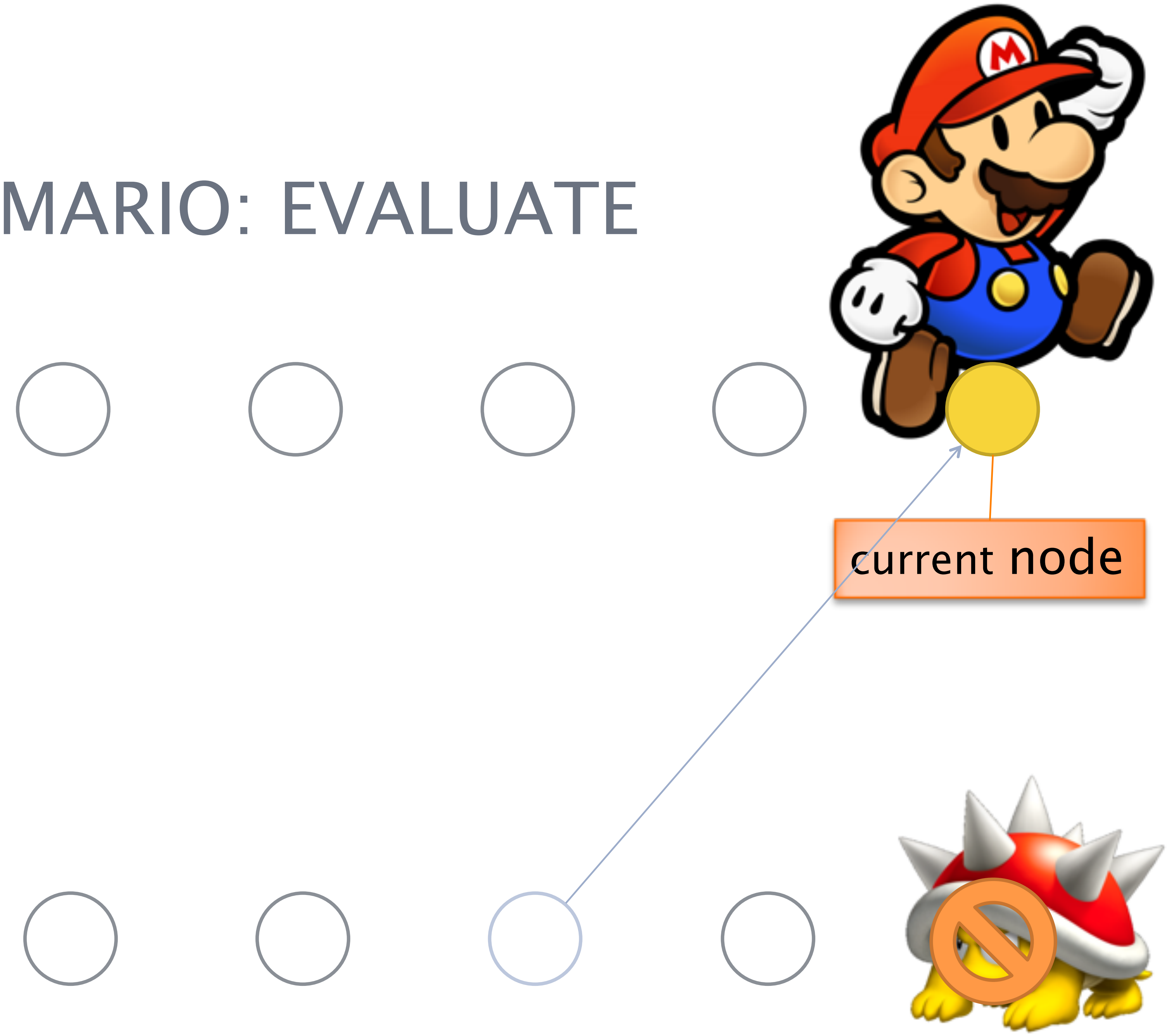
A* IN MARIO: BACKTRACK



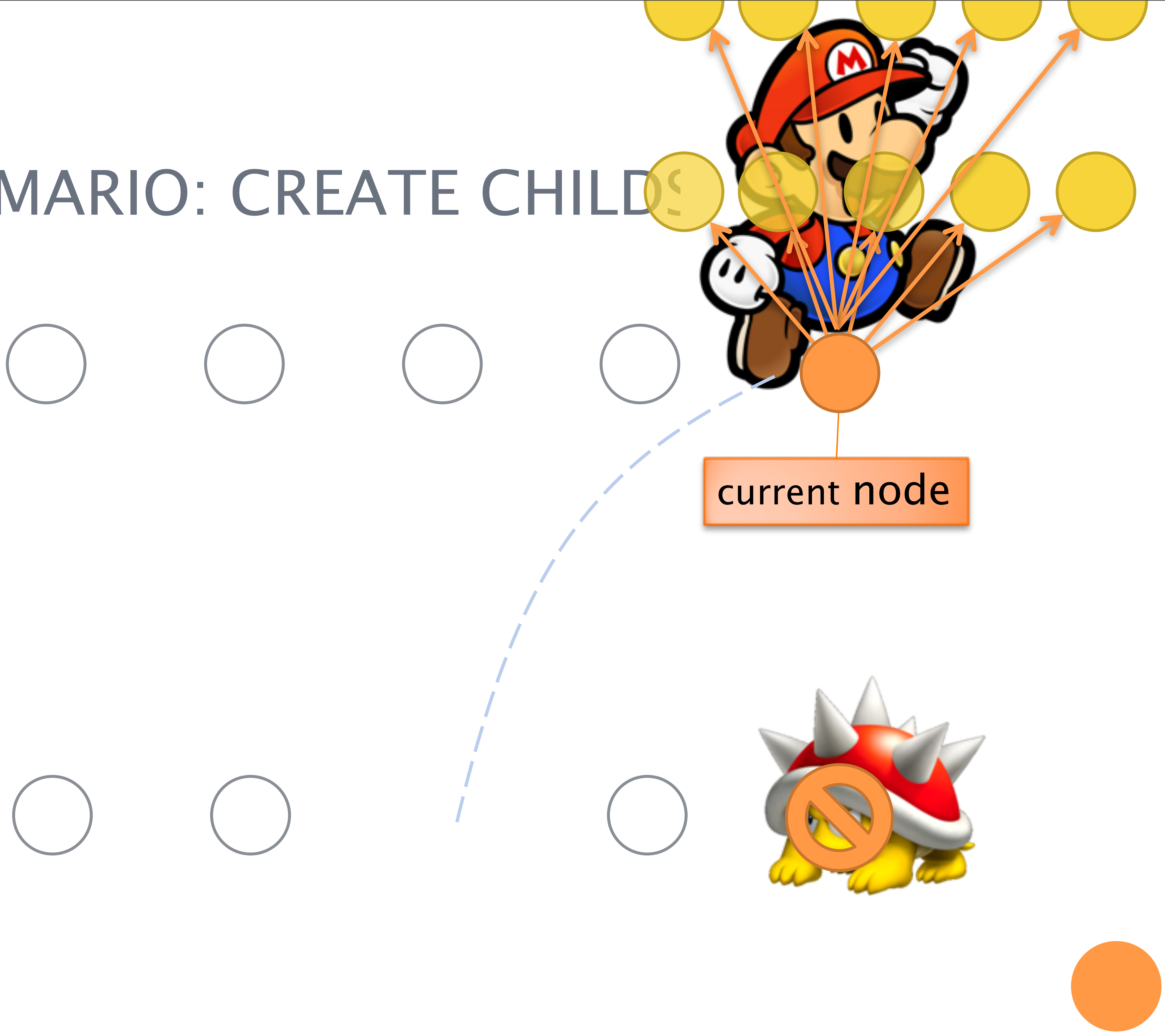
A* IN MARIO: BEST FIRST



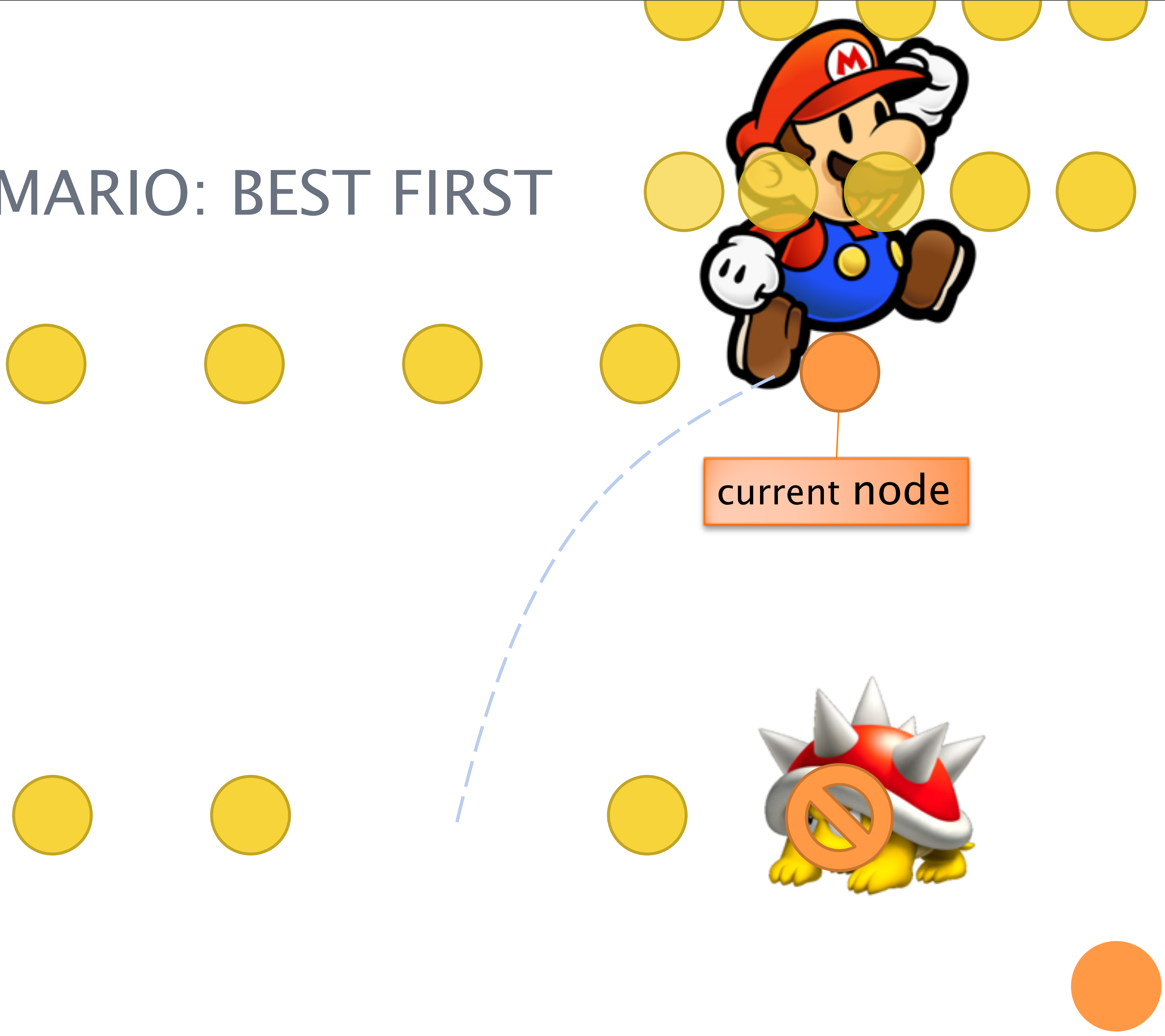
A* IN MARIO: EVALUATE



A* IN MARIO: CREATE CHILDS



A* IN MARIO: BEST FIRST



HEURISTIC

- Using Mario’s current speed and acceleration, how long does it take to reach the goal?
- Assume maximum acceleration and no obstacles (admissible heuristic!)

$$xa = xa+1.2$$
$$x = x+xa$$
$$xa = xa * 0.89$$

- Optimisation: Find a closed form for this.

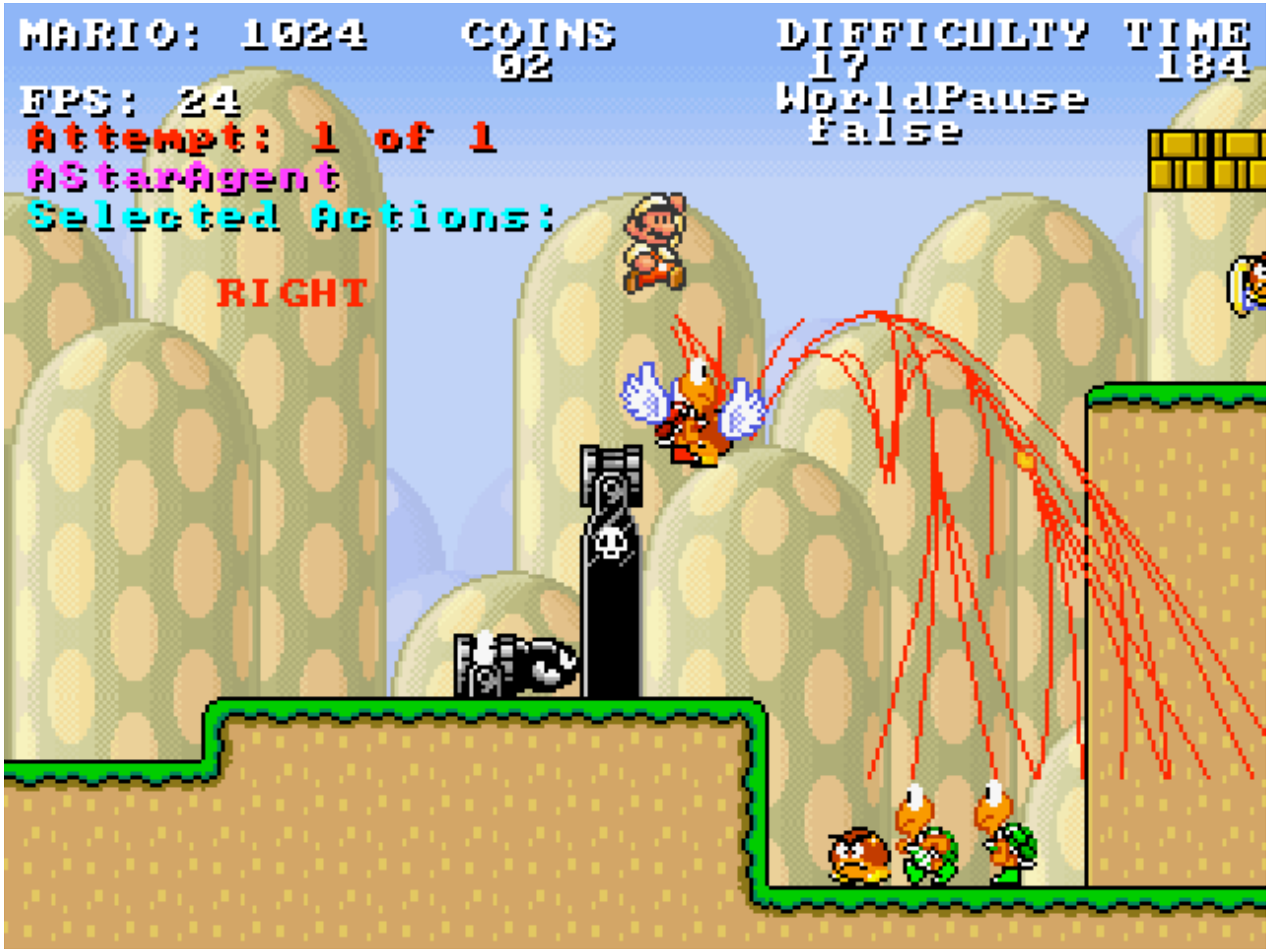
HANDLING NEW EVENTS

- Plan ahead for two ticks (=1/12 sec)
- Synchronise internal world-state with received enemies and object positions.

Possible Improvements:

- Keep & update old plan instead of starting from scratch each time
- Collect coins & power-ups (e.g., using a high-level planner that pans out the route between power-ups)

VIDEO


MARIO: 1024
COINS 02
DIFFICULTY 17
TIME 184
FPS: 24
WorldPause false
Attempt: 1 of 1
AStarAgent
Selected Actions:
RIGHT

Glenn Hartmann

- Modified version of one of the heuristic agents that came with the software
- Move forward
- Jump if in danger of falling
- Jump over enemies if safe
- Shoot continuously

Rafael Oliveira

- Did not submit any documentation
- Seems to be an elaborate heuristic

Peter Lawford

- A-star search to maximize x position
- Partial simulation to anticipate future positions (recalculated if simulation goes out of sync)
- Some pruning of search tree

Sergio Lopez

- Rule-based system, to answer 2 questions: “should I jump?” and “which type of jump?”
- Evaluates possible landing points based on environment info and heuristics (no simulation)
- Calculates “danger value” for each action, and “need to jump”
- Special situations, e.g. waiting for flowers and bullets to go away, climbing “stairs”

Mario Pérez

- Subsumption-type controller: later layers can override the action of earlier layers
- Each layer either a method or a state machine

- avanzar() -> makes Mario going forward
- saltarParedes() -> makes Mario jump when necessary for advance
- subirEscaleras() -> makes Mario climb "stairs" (these mean of rocks)
- saltarPozos() -> makes Mario jump over gaps
- saltarEnemigos() -> makes Mario jump over enemies
- dispararEnemigos() -> makes Mario shoot enemies
- evitarArrollarEnemigos() -> makes Mario going back to avoid enemies while in air

Andy Sloane

- Joint work with Caleb Anderson and Peter Burns
- Based on A star
- Separate simulation of the game physics (not using the game engine)
- (imperfect) prediction of enemies' movements
- Working towards propagating penalties in the tree

Erek Speed

- Rule-based system
- Maps the whole observation space to the action space
  - antecedent: 22x22 array, consequent: 6 bits action
  - put in hash table
- Evolved with a GA
  - Genome as > 100 Mb XML file!

Michal Tuláček

- State machine with 4 states: walk_forward, walk_backward, jump, jump_hole

Results

Name	Score	Time
Robin Baumgarten	17264	5.62
Peter Lawford	17261	6.99
Andy Sloane	16219	15.19
Sergio Lopez	12439	0.04
Mario Pérez	8952	0.03
Rafael Oliveira	8251	?
Michal Tuláček	6668	0.03
Erek Speed	2896	0.03
Glenn Hartmann	1170	0.06
our evolved neural net	*7805*	*0.04*
ForwardJumpingAgent	*9361*	*0.0007*

Observations

- The best-performing controllers take much longer time per time step (frame)
- This is because they use A star search!
    - ...and these work well because of the lack of blind alleys (should be fixed)
- But some heuristic controllers do very well
- Not a lot of learning/optimization techniques (though many competitors claim to be working on it)

Next phase: CIG 2009

- Milan, Italy, 7-11 September
- Submission deadline: 3 sept.
- Minor additions to the interface
- Fully backward-compatible: all agents submitted for this phase will work...
    - ...and will be automatically entered
- Still time for <u>you</u> to submit your agent!

After the competition

- Competition web page will remain, complete with competition software
  - ...which you can use in your teaching or research!
- Complete source code of all submitted controllers

The future of the Mario Competition

- Mario AI Championship 2010
- Run at 2 to 4 different conferences?
- More than one track, ideas include:
  - Agent time-budget track
  - Online learning of unseen level track
  - Personalized level generation track
- (your idea here)