Balancing Cooperation and Defection in a 2-agent grid-world game

The Malmö Collaborative AI Challenge

Adrià Garriga Alonso, Daniel Furelos Blanco, David Tena Cucala

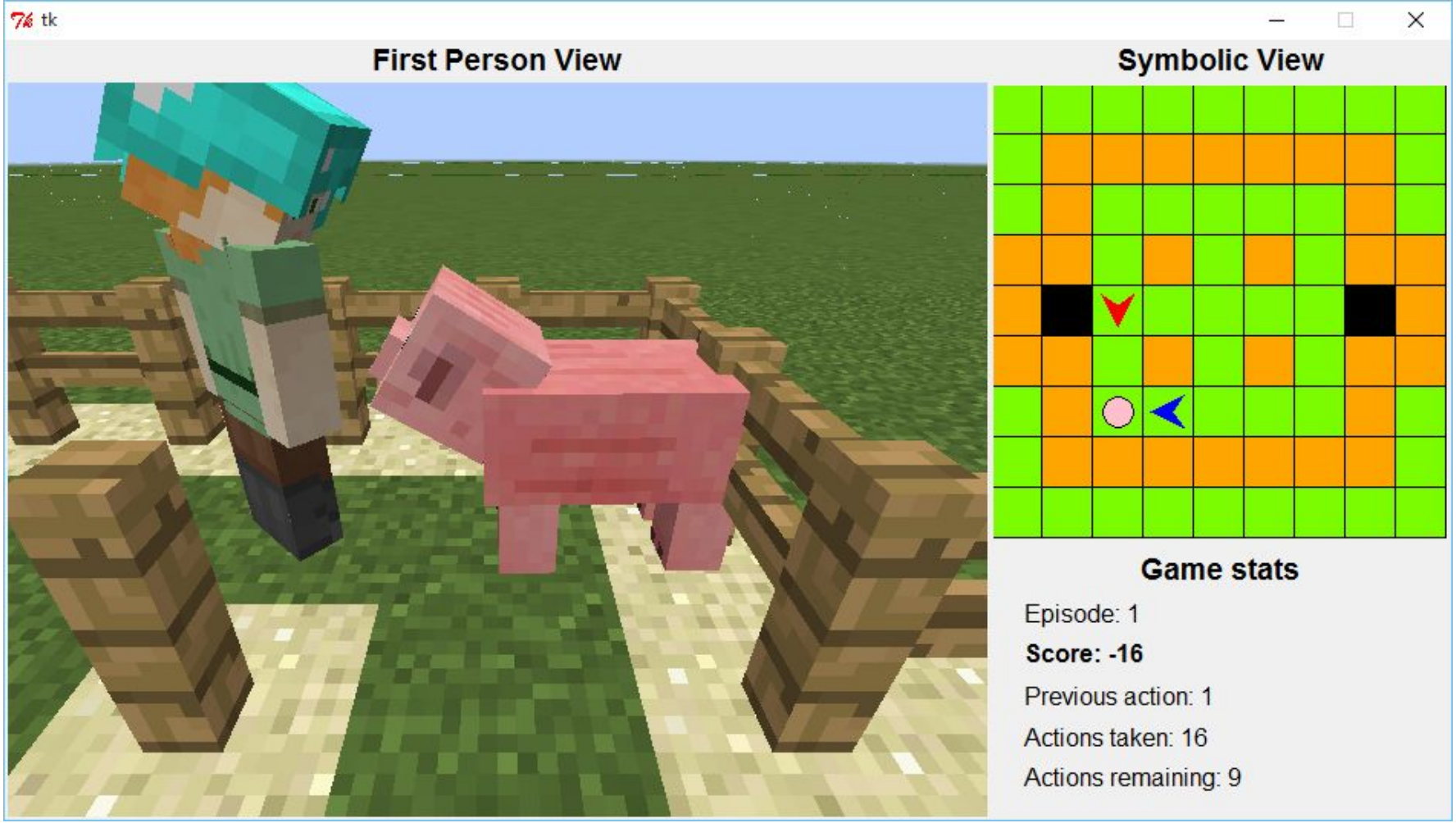


Image from the official challenge repository

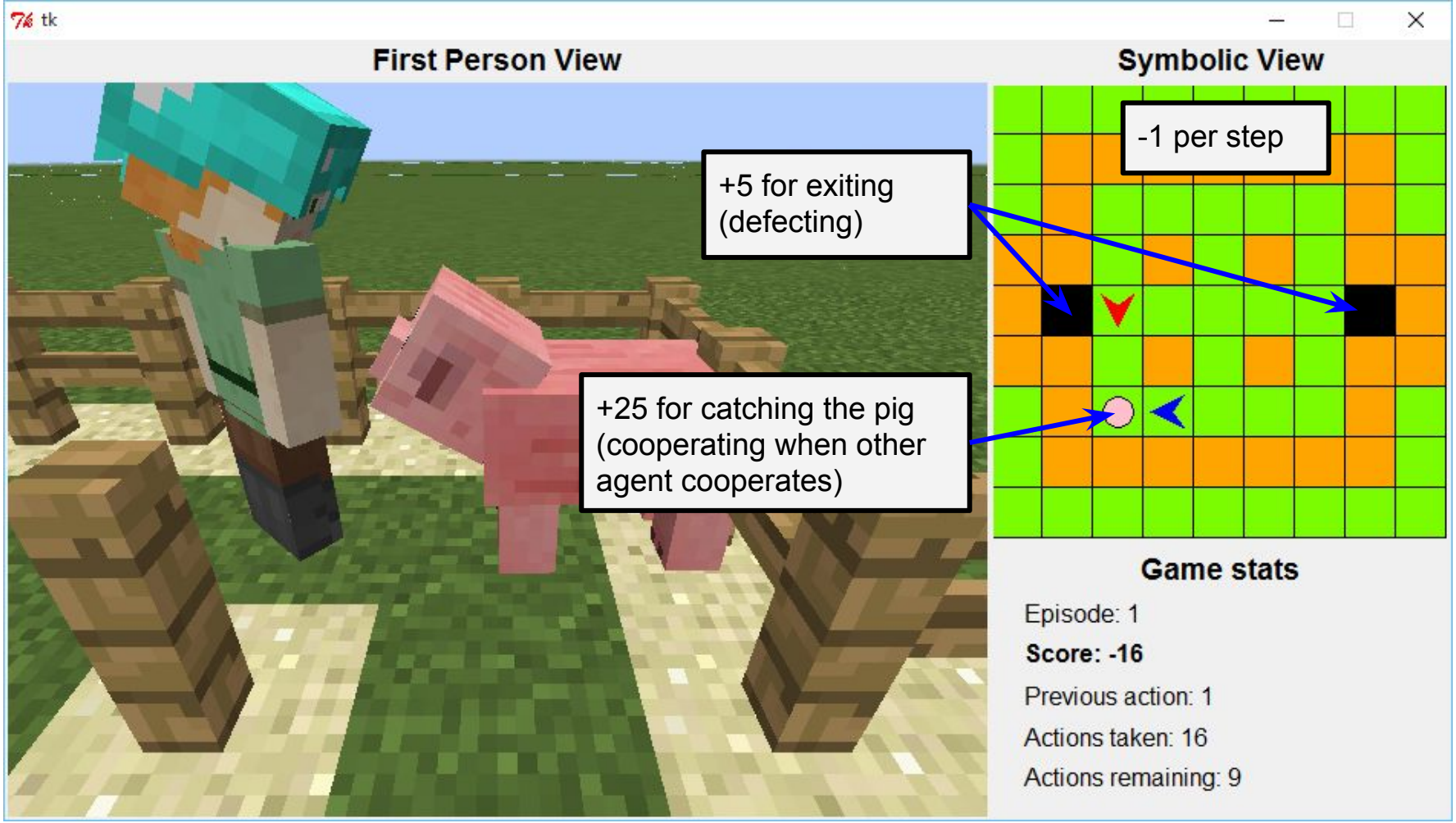


Image from the official challenge repository

Inferring other agent's strategy

- Define a set of basic strategies
  - Focused: go towards the pig as fast as possible
  - Random: Take an uniformly random action

Called "policies" in RL, they define probability distribution over actions given a state

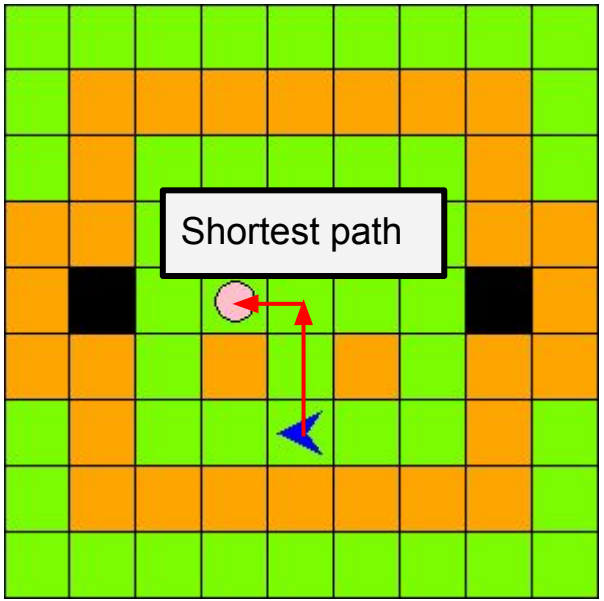
- Maintain a belief: probability of the other agent using each of the strategies

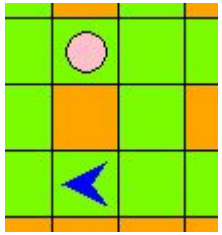
- Update using Bayes' theorem
  - Priors known

Specific case of Bayesian inference over a **parametrised family** of policies.

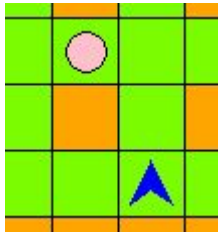
In this case there is only 1 parameter, which can have *n* discrete values for *n* base strategies.

Inferring other agent's strategy

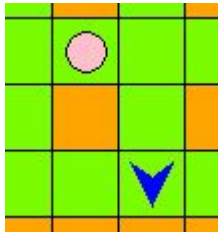




$$P(\uparrow \mid \text{focused}) = 0.01$$
$$P(\uparrow \mid \text{random}) = 1/3$$



$$P(\rightarrow \mid \text{focused}) = 0.98$$
$$P(\rightarrow \mid \text{random}) = 1/3$$



$$P(\leftarrow \mid \text{focused}) = 0.01$$
$$P(\leftarrow \mid \text{random}) = 1/3$$

Planning the optimal action

- UCT (Upper Confidence bounds for Trees), which is Monte-Carlo Tree Search with an exploration policy.
  - Default policy: if other agent cooperates, go to the place needed to catch pig; otherwise defect.

Other agent considered “environment”, its strategies sampled according to belief.

Could add stochastic model of pig moving but *we had no time.*

Exploration term

Avg. value

$$\bar{R}_j + 2c\sqrt{\frac{2\log n_j}{n_{\text{parent}(j)}}}$$

c=10 to compensate for R ∈ [-25, 25]

Evaluating metric: reward per step after 10 episodes.

Our algorithm: ~1.5 score, no training.

Tabular Q-learning baseline: ~0 score, 100k episodes training.

Takeaways

- Training not always needed
  - Especially in simple environments (where one can build model)
  - Especially in small-state-space environments

- Planning approaches easily adapt to new information as it comes
  - In this case, the last action of the other agent

- Learning in constrained domains very efficient
  - Learn other agent’s policy in 2-3 actions