<CAPS>

Session 3 – From Learning Architectures to Decision Profiles

Leo Klenner, Henry Fung, Cory Combs

From Deduction to Induction

Deduction

Learning architectures $\rightarrow$ decision profiles

Learning architectures $\overset{?}{\leftarrow}$ decision profiles

Induction

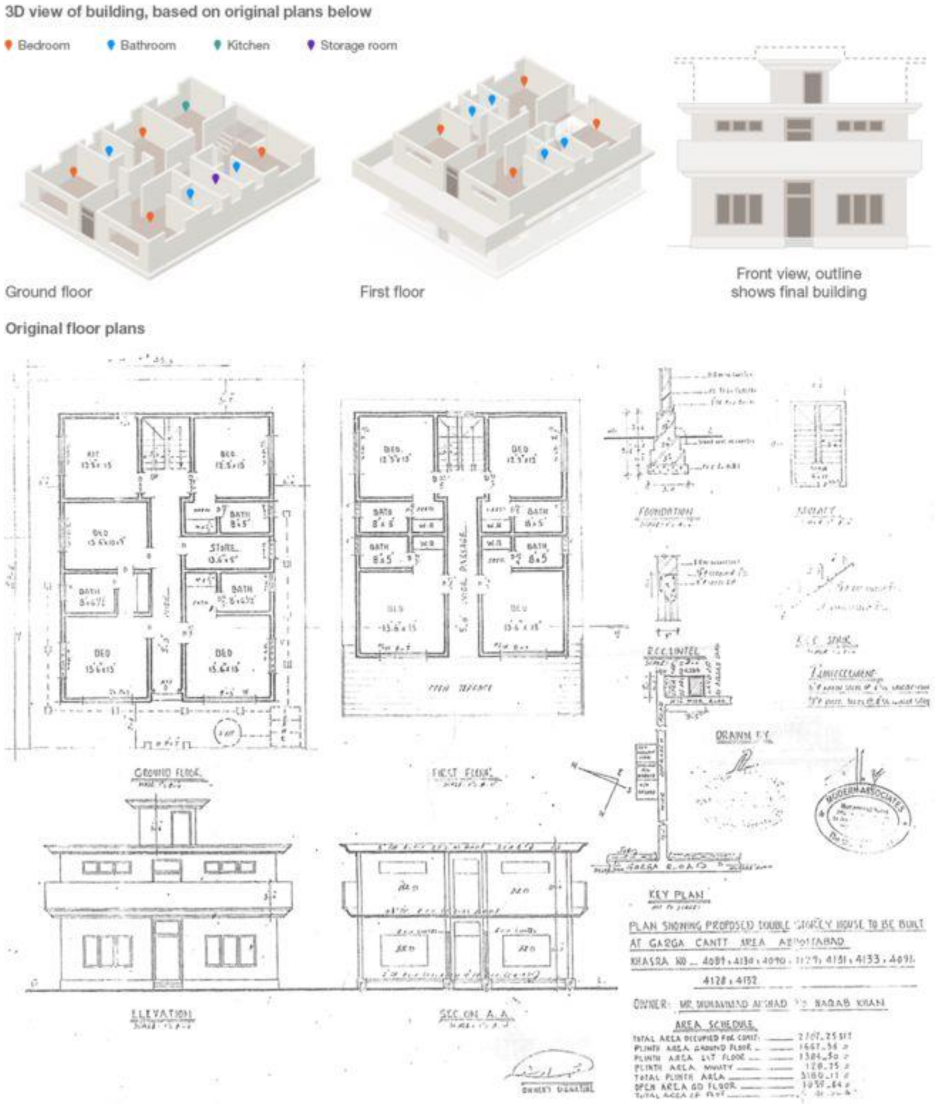
Review of Sessions 1 and 2

- Architectures for decision making
  - **Learning in human teams** (supervision, interaction)
  - **Reinforcement learning** (online-offline, learning with state-action pairs)
  - **Rule-based systems** (Finite State Machines, conditional logic)
  - **Multi-agent learning** (joint-separate, “Policy Geometry”)

Outline

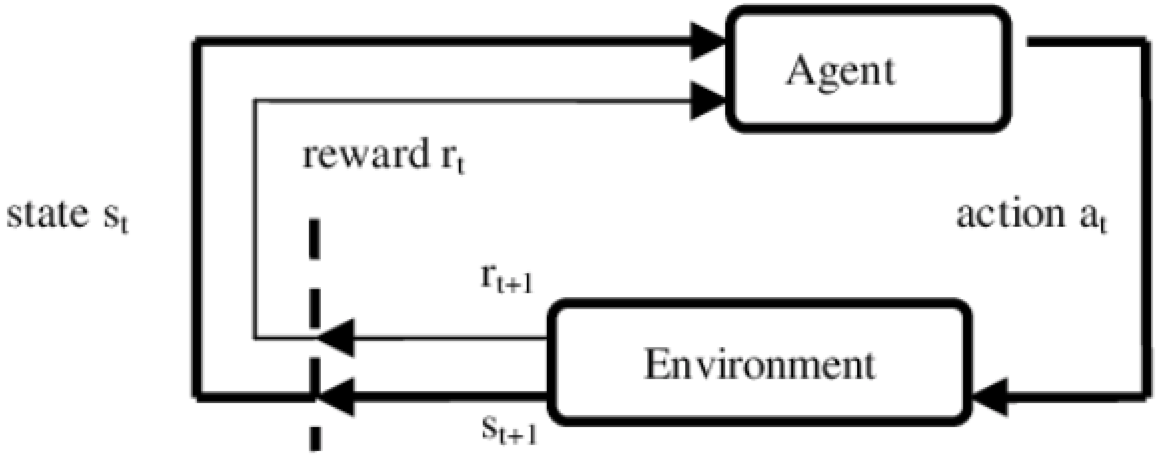
1. Exercise on Rule-Based Systems
2. Recap on Reinforcement Learning
3. Q-Learning
4. Case on Inferring Learning Architectures from Observed Behavior

Location X


3D view of building, based on original plans below
Bedroom
Bathroom
Kitchen
Storage room
Ground floor
First floor
Front view, outline shows final building
Original floor plans
GROUND FLOOR
FIRST FLOOR
ELEVATION
SEC. ON A.A.
FOUNDATION
DRAWN BY
KEY PLAN
AREA SCHEDULE

Review of Reinforcement Learning Basics

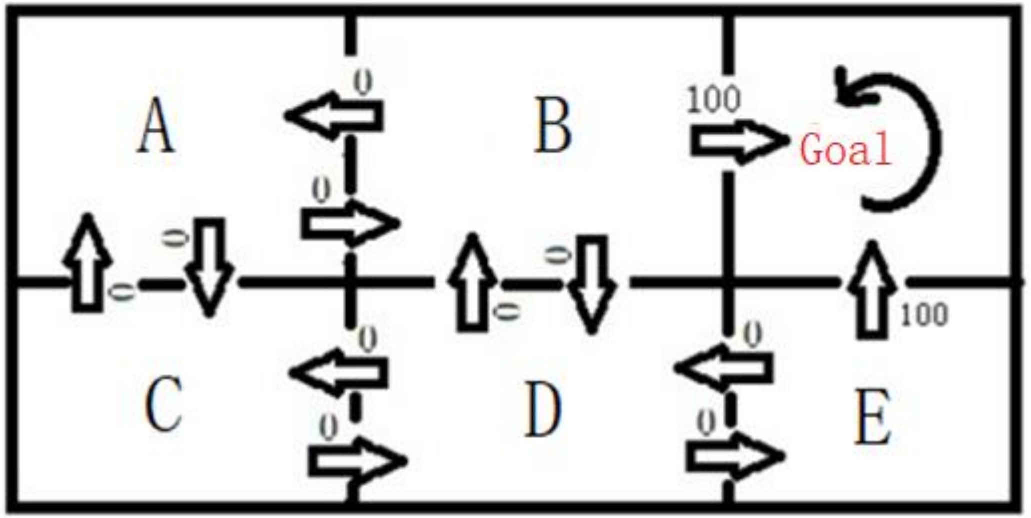
- RL is suitable to a class of problems known as the Markov Decision Process (MDP)
  - Outcome of applying action to any state depends only on this action *a* and state *s* (and not on preceding *a or s*)
  - Information available to the agent: rewards and subsequent state for each state-action transition



- *What are the components of a MDP?*

What is the Total (Cumulative) Reward?

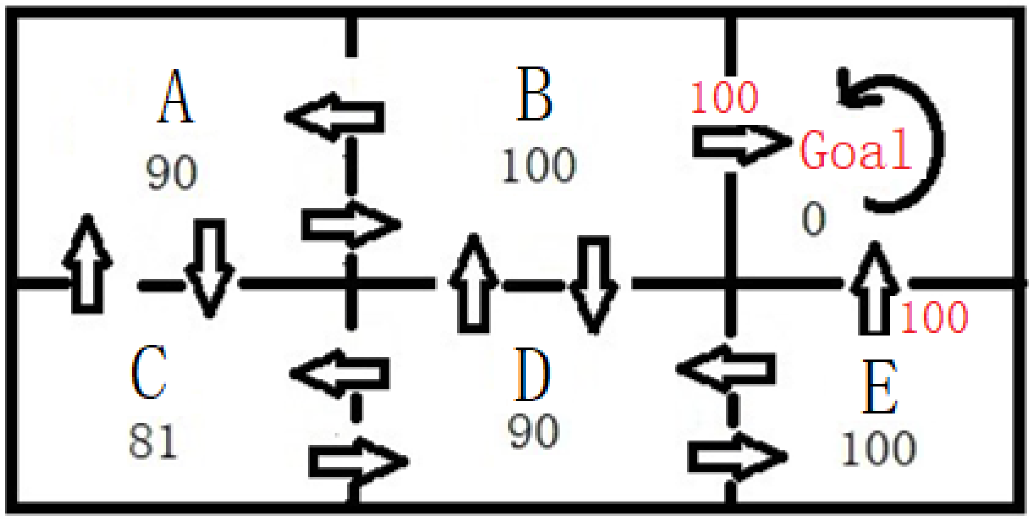
- After performing action a in state s (state action pair s, a):
  - The agent receives immediate reward of either 0 or 100
  - The agent goes to the next state
- The action that the agent chooses in every state s is dictated by a policy π
- $\pi(s_t) = a_t$
- The total reward that the agent gets (from π) is the sum of the sequence of rewards that it gets by following π, discounted by ϒ



$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + y^2 r_{t+2} + \cdots$$

What is the Agent's Goal?

- What is the goal of an agent that uses RL?
  - Pick the best policy that will maximize the total rewards that the agent receives
- How does it do that?
  - The agent needs to evaluate the "goodness" of each state. How good is it to be in Grid A, in Grid E?
  - This evaluation can be done if the agent knows the maximum total reward that it can achieve in each state
  - We call this the "optimal value function" V*(s)
  - If we have good estimates of V*(s), then we can choose the optimal policy

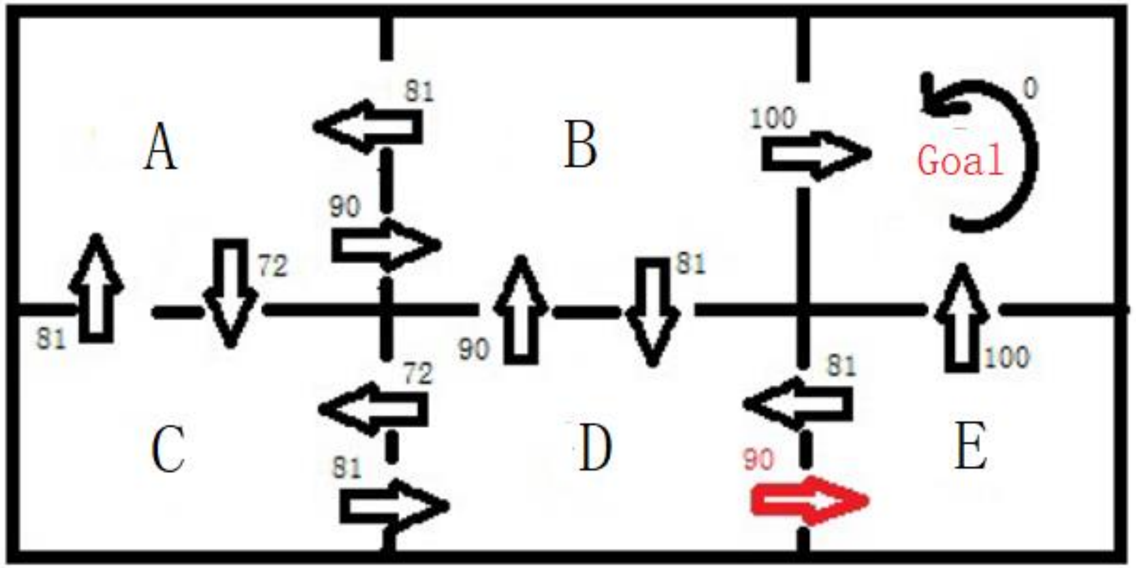


$$V^*(s = A)$$

$$\pi^*(s) = \underset{a}{argmax}[r(s,a) + \gamma V^*(\delta(s,a))]$$

The Q-Function

- Rather than evaluating the “goodness” of each state, we can also evaluate the goodness of each state-action pair
  - How good it is for me to perform RIGHT in state D (as oppose to RIGHT) in state B?
  - Compare the Q value of doing RIGHT in D and RIGHT in B
  - Q value is the maximum achievable total reward that the robot get by performing *a* in *s*, and then following optimal policy thereafter



$$Q(s,a) = r(s,a) + \gamma V^*\big(\delta(s,a)\big)$$

$$Q(s = D, a = right) = 0 + 0.9 * 100 = 90$$

The Bellman Equation

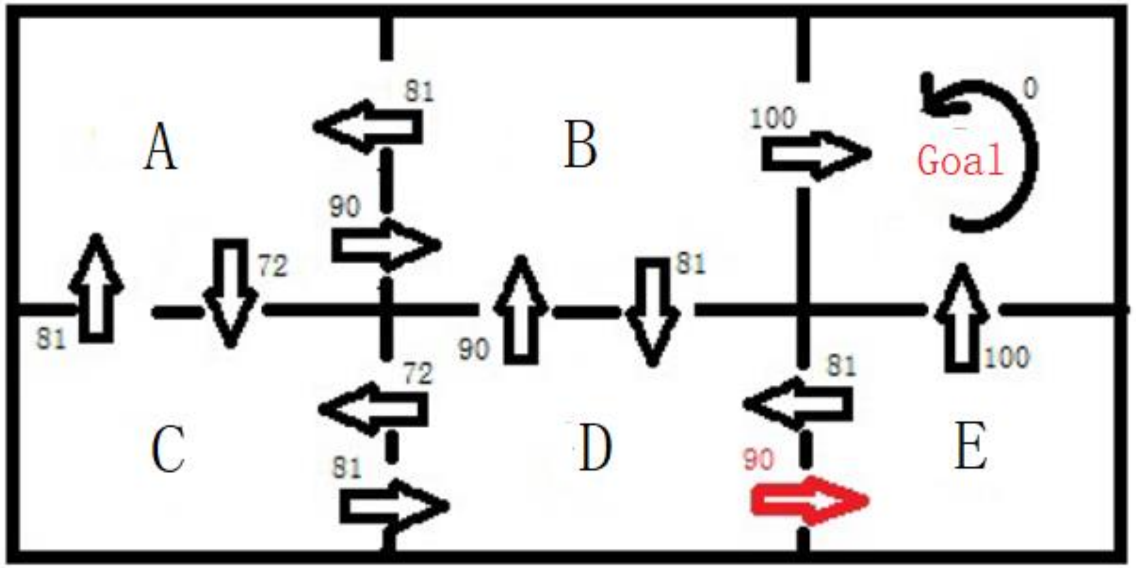
> The Q-function can be rewritten as the Bellmann Equation:

$$Q(s,a) = r(s,a) + \gamma \max Q\,(\delta(s,a), a')$$

> Learning Q corresponds to learning the optimal policy.

> What is Q(s=D, a=RIGHT) from the Bellmann Equation?

> Learning Q corresponds to learning the optimal policy. Assuming Q-values are known, what is the optimal strategy starting from A?



$$\pi^*(s) = \underset{a}{argmax}\; Q(s,a)$$

Learn V*(s) or Q(s, a)?

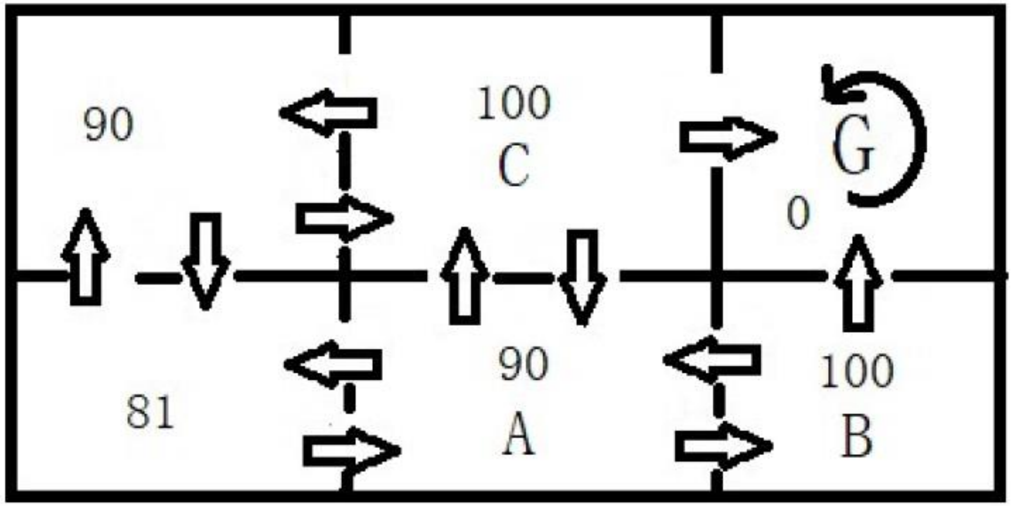
> **Approach 1:** If we know the true V*(s) values, we know the optimal policy:

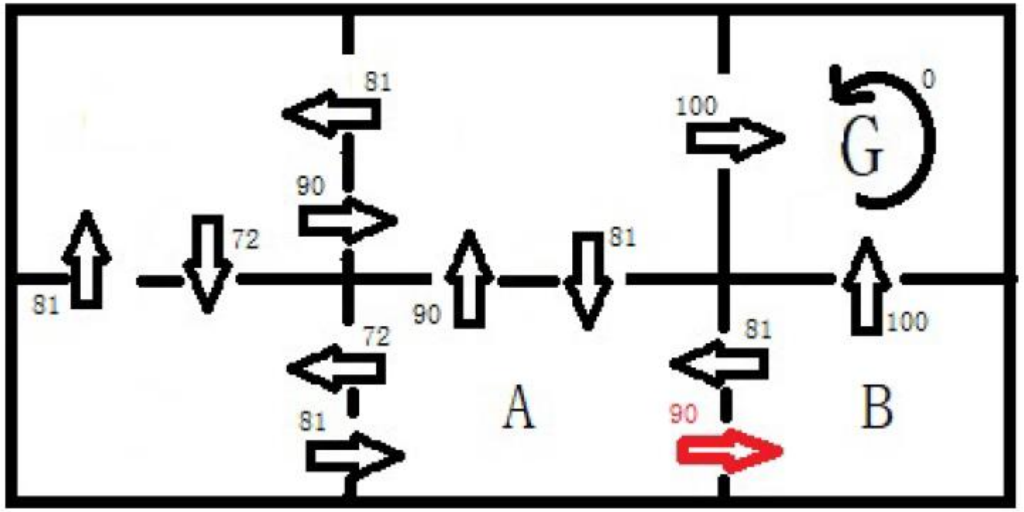
$$\pi^*(s) = \underset{a}{argmax}[r(s,a) + \gamma V^*(\delta(s,a))]$$

> **Approach 2:** If we know the true Q-values, we know the optimal policy:

$$\pi^*(s) = \underset{a}{argmax}\, Q(s,a)$$

> Which approach should we choose?





Model-based vs. Model-free Learning

Model-Based Learning

- Model of the environment : reward and state-transition need to be known
- The agent interacts with environment, and from the "history" of its interactions, the agent will approximate the reward and state transition
- After that, the agent learns the V*(s) using an algorithm called "Value iteration". Once the V*(s) values are known, the agent can find the optimal policy

$$\pi^*(s) = \underset{a}{argmax}[r(s,a) + \gamma V^*\big(\delta(s,a)\big)]$$

Model Free Learning

- The agent will not try to "guess" the reward and state transition functions. It will only "experience" them as it explores the environment
- By trail and error, the robot discovers what are the good and bad actions in each state
- Uses "Q learning" to approximate the Q(s, a) values and find optimal policy

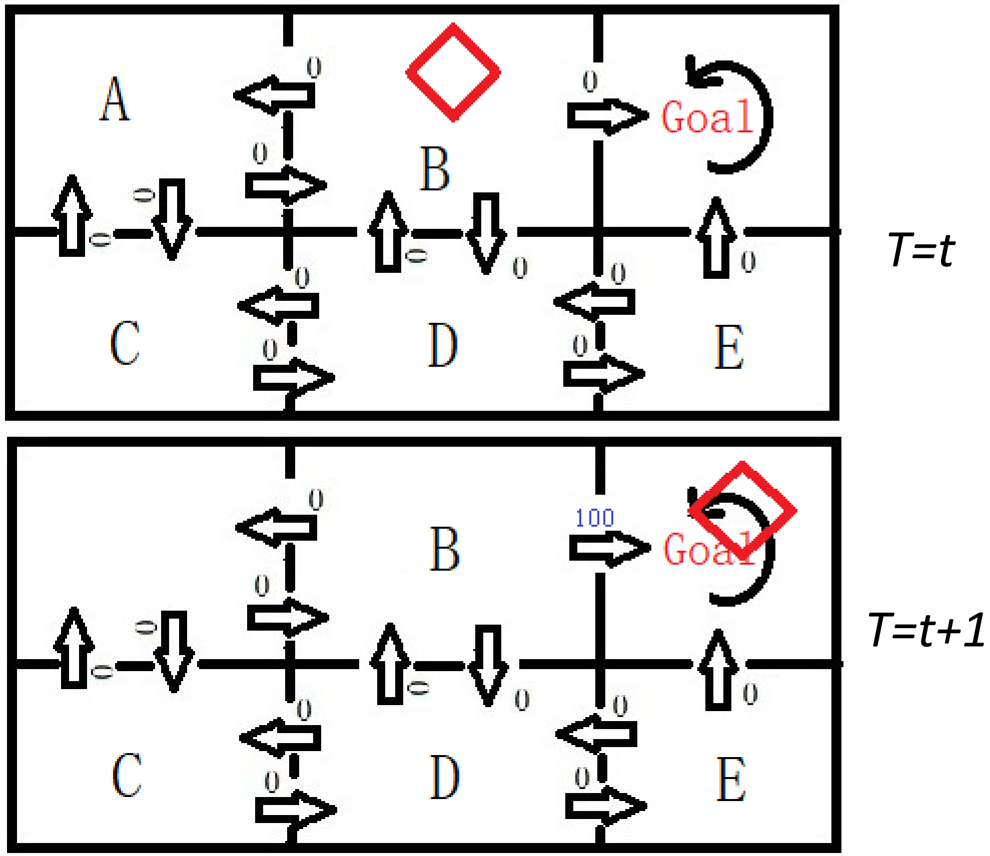
$$\pi^*(s) = \underset{a}{argmax}\, Q(s,a)$$

The Q-Learning Algorithm

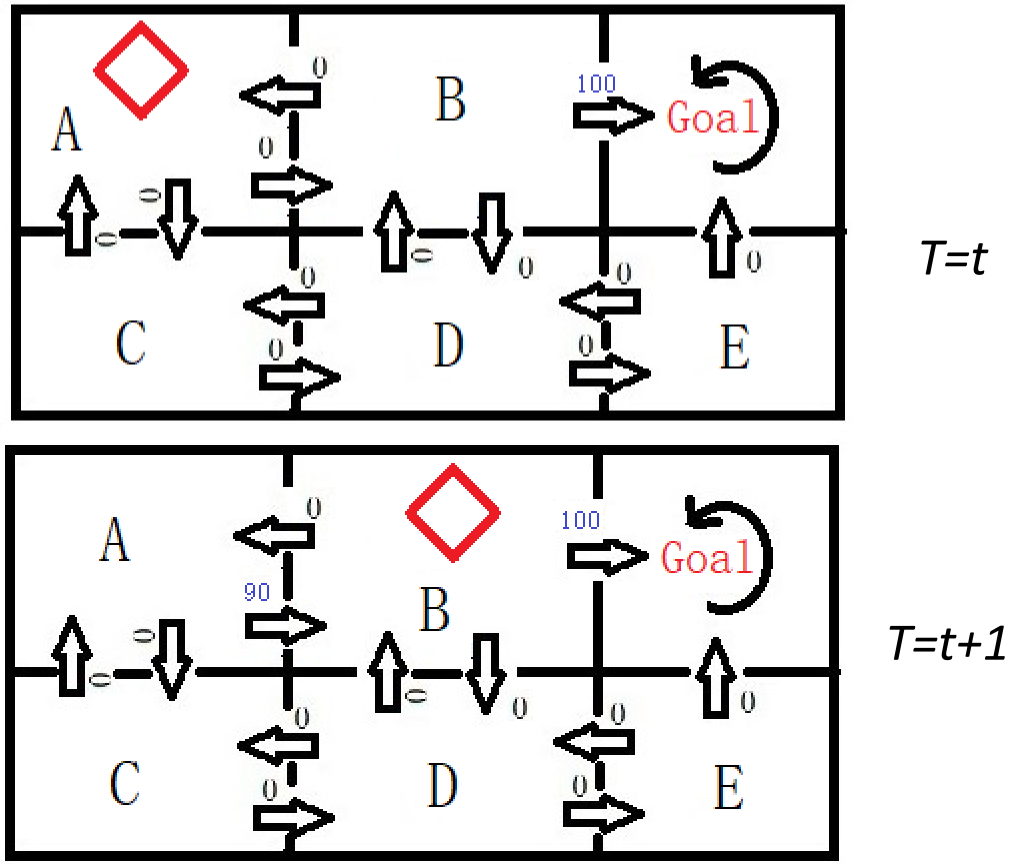
- Assumptions:
  - (A1) Reward and state-transitions functions are unknown but deterministic.
  - (A2) Robot visit every possible state with non-zero frequency overtime.
  - (A3) Rewards are bounded (not infinite).

- Algorithm:
  1. Initially fill the table of $\hat{Q}$ (estimated Q-values) with zeros
  2. Observe current state s
  3. Select and action a and executes it
  4. Receive immediate reward r
  5. Observe the new state s’
  6. Update the table entry for $\hat{Q}$(s, a) using the Bellman equation
  7. Set state s’ to s, repeat 2-6 until $\hat{Q}$ converges (new $\hat{Q}$ values do change much from old values)

Example: Information Propagation



Training Episode 1



Some Training Episode after 1

Strategies for Experimentation

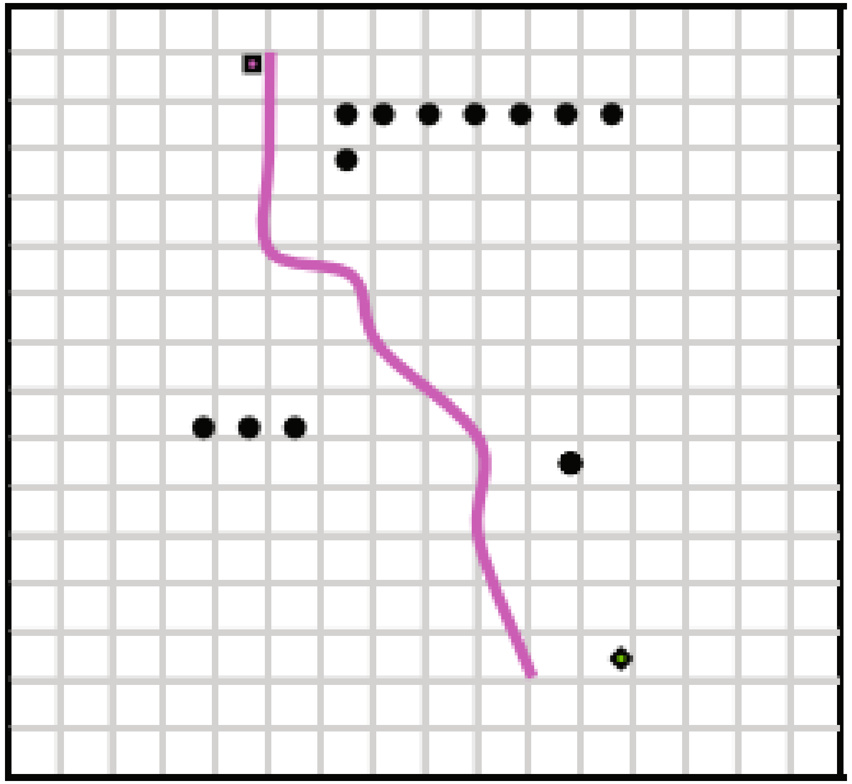
- > The Q-learning algorithm does not specify how the agent chooses its actions
- > If the agent *always* selects the state-action pair with the highest estimated Q-values, then it will "overcommit" to its actions found in the early training sessions, while failing to explore other state-action pairs that might have even higher Q-values
- > Solution to Exploration vs. Exploitation dilemma:
  - > During early training episodes: All actions have a chance of being executed
  - > During the latter training episodes: Actions with higher $\hat{Q}$ will be assigned with increasingly higher probabilities, so overtime, the agent will do more exploitation and less exploration

Decision Profiles in Unbounded Environments

Learning architectures $\overset{?}{\leftarrow}$ decision profiles

- Does each architecture correspond to a distinctive decision profile?
- How can we discern this decision profile?
- What are the consequences, if we answer the first or second question negatively?

AV Flight Trajectory



> Goerzen, et al. 2009. A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance

Toy Model of Learning Architectures

> Does each architecture correspond to a distinctive decision profile?

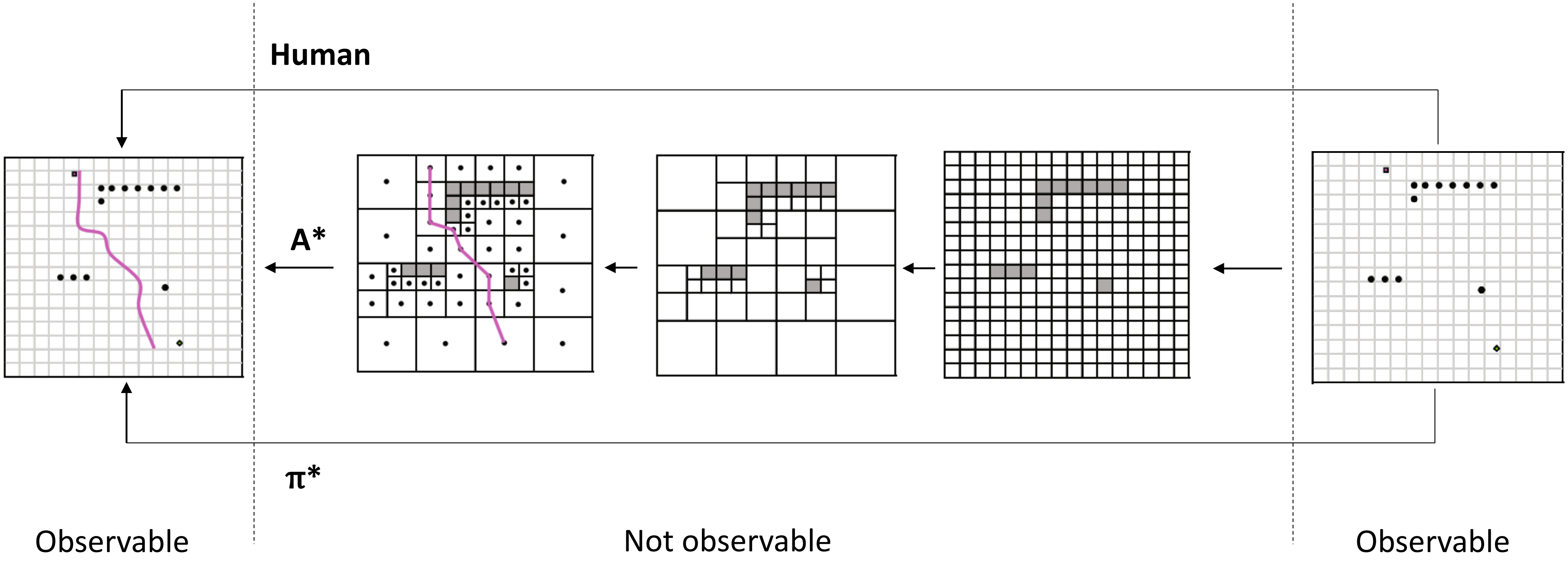
Rule-based → precise solution, no adaptation

Reinforcement learner → multiple degrees of adaptation

Human → full adaptation and rule generation

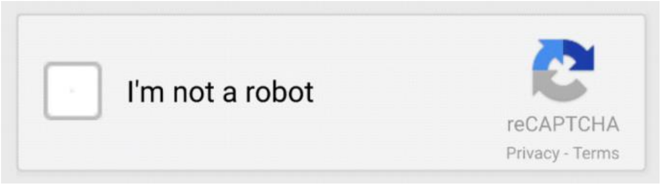
> Assumes we have perfect information about the environment
> Real-world is often too complex to justify such an assumption

Possible Architectures for Trajectory



Why Care about Verification?

Arms control, countermeasures



uncertain

What are appropriate criteria?

Differentiating Human from Machine

Turing Test +
its variations

→ Is the machine indiscernible from a human?

↑

Test asssumes stable and constrained test environment

↑

Real world is unstable, partially observable

> **Hypothesis**: A system that would not pass the Turing Test in theory might pass it in practice, because the test environment is fuzzy

Other Mechanisms of Verifcation

	Worse than human	Human-level	Better than human
Performance	Worse than human	Human-level	Better than human
Verifcation	Easy	Hard	Easy; depends
	↓	↓	↓
	Clear failure modes	**Internal factors:** Unexpected decision patterns Unexpected failure modes **External factors:** Change in force structure Change in strategy **Introduce disturbances in env.**	Unparalleled performance

Petrov Verification

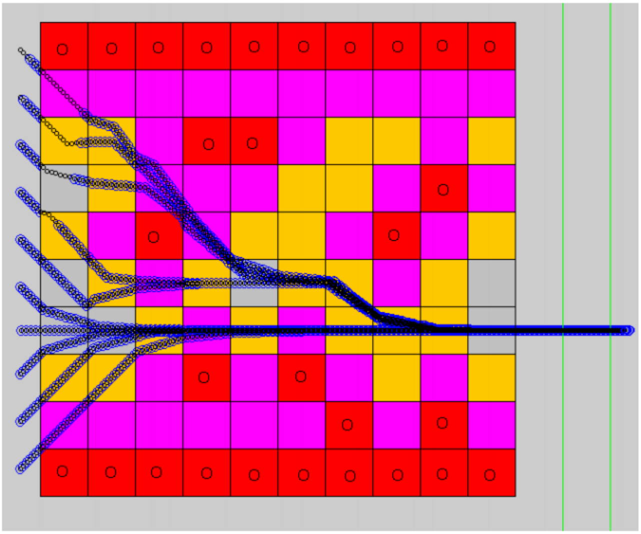
- > September 26, 1983
- > Five U.S. ICBMs surface on Soviet radar, station monitored by Stanislav Petrov
- > Petrov reasons that this is a false alarm, decides not to launch counter-attack

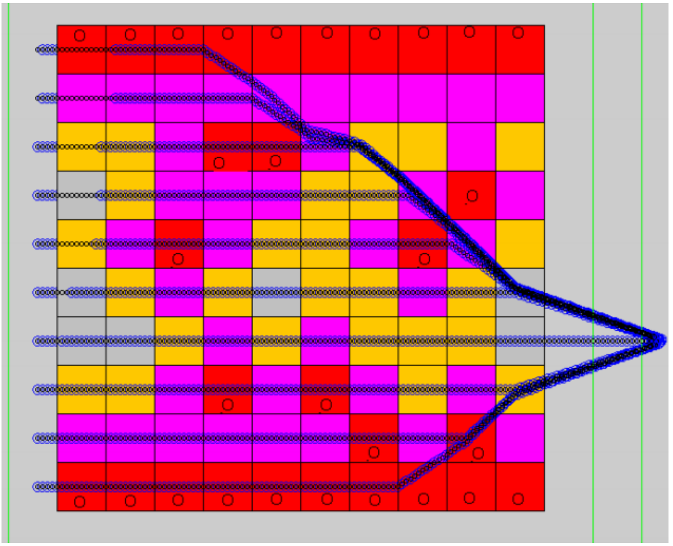
Soviet warning systems *technologically* limited

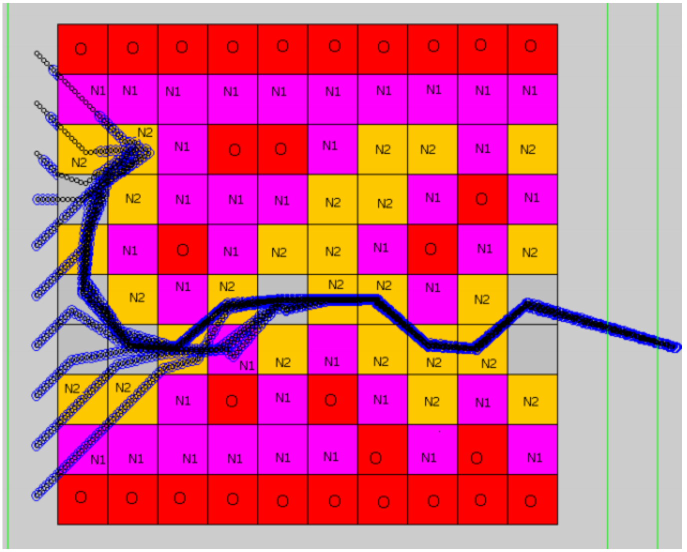
Contained first strike *strategically* nonsensical

= Petrov Verification: combination of internal and external factors; educated guess *across* domains on limited information

Appendix on Countermeasures







> Krishna et al. 2005. Parametric Control of Multiple Unmanned Air Vehicles over an Unknown Hostile Territory