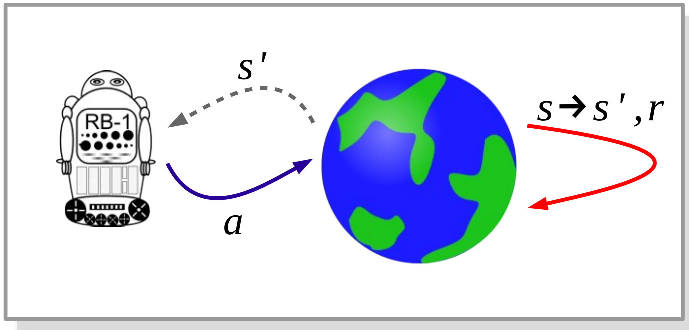
Probabilistic Artificial Intelligence

Lecture 10: (Model-Based) Reinforcement Learning

Slides, RN chap. 23

further reading: Sutton&Barto v2



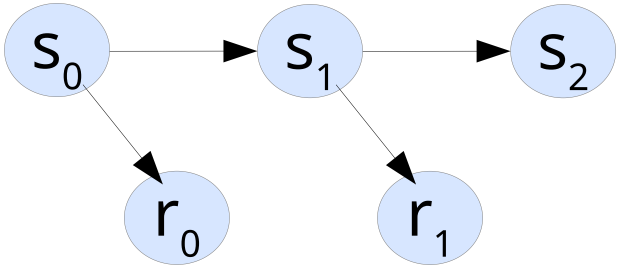
Dr. F. Oliehoek

TU Delft
Delft University of Technology

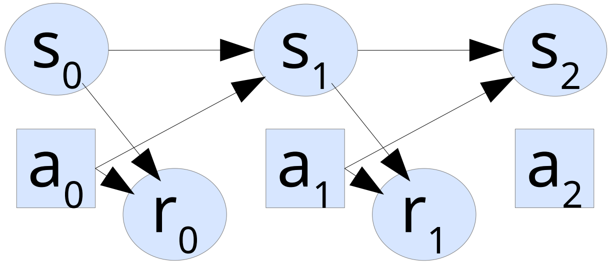
Recap

- RL: when we don't have a model
- Last lecture: value-based, model-free methods

- "Passive learning" – policy evaluation
  - Monte Carlo estimation,
  - TD-learning



- "Active learning" – learn which actions to take
  - Q-learning – off-policy
  - SARSA – on-policy
  - exploration...!



Updates…

- TD-learning: after (s,r,s') we update
  - $V(s) := V(s) + \alpha [r + \gamma V(s') - V(s)]$
- Q-learning: after (s,a,r,s') we update
  - $Q(s,a) := Q(s,a) + \alpha [r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
- SARSA: after (s,a,r,s',a') we update:
  - $Q(s,a) := Q(s,a) + \alpha [r + \gamma Q(s',a') - Q(s,a)]$
- But 'tabular' methods need to store values for all s,a
  - MDPs are huge… how to scale…?

→ function approximation

  - for Qnets (e.g., DQN)
  - for policies themselves: policy search methods

Policy Search

[RN 21.5]

Policy Gradient Methods

- main idea: do not bother with value functions Q/V
- Instead,
  - directly parametrize policy π(a|s ; w)
  - update these parameters based on the returns u(s) observed.

- REINFORCE:
  - $w_{t+1} := w_t + \alpha * u(s_t) * \nabla \log \pi(a_t \mid s_t ; w_t)$
  - See S&B v2 13.3

Policy Gradient Methods

- main idea: do not bother wit
- Instead,
  - directly parametrize policy π(
  - update these parameters bas observed.
- REINFORCE:
  - $w_{t+1} := w_t + \alpha * u(s_t) * \nabla \log \pi(a_t \mid s_t ; w_t)$
  - See S&B v2 13.3

Where does the log come from...?

- In the derivation of Reinforce, there appears a term:

  $\nabla\pi(a|s) / \pi(a|s)$

- since $\nabla \log x = 1/x * \nabla x$
- we can rewrite:

  $\nabla \log \pi(a|s) = 1/\pi(a|s) * \nabla\pi(a|s)$

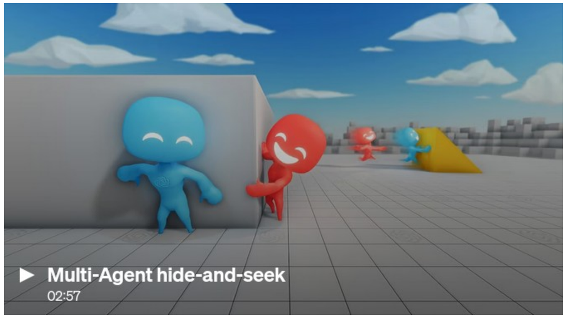
TUDelft

Actor-Critic Methods

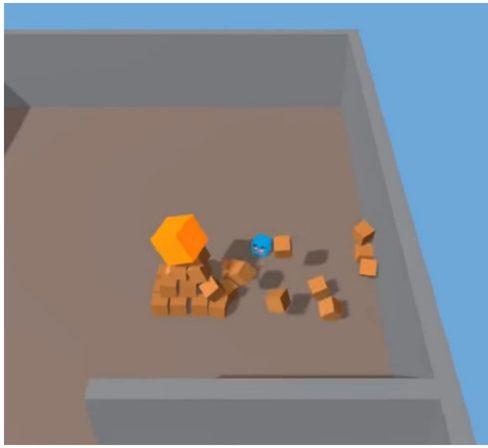
- Policy gradient methods can be combined with estimated Q-value functions:
  - policy = actor → just tries to take good actions
  - value function = critic → gives feedback to policy

- This addresses the high variance that PG methods (working directly on returns) otherwise have.

- Recent versions of these methods have led to state-of-the-art performance in many domains



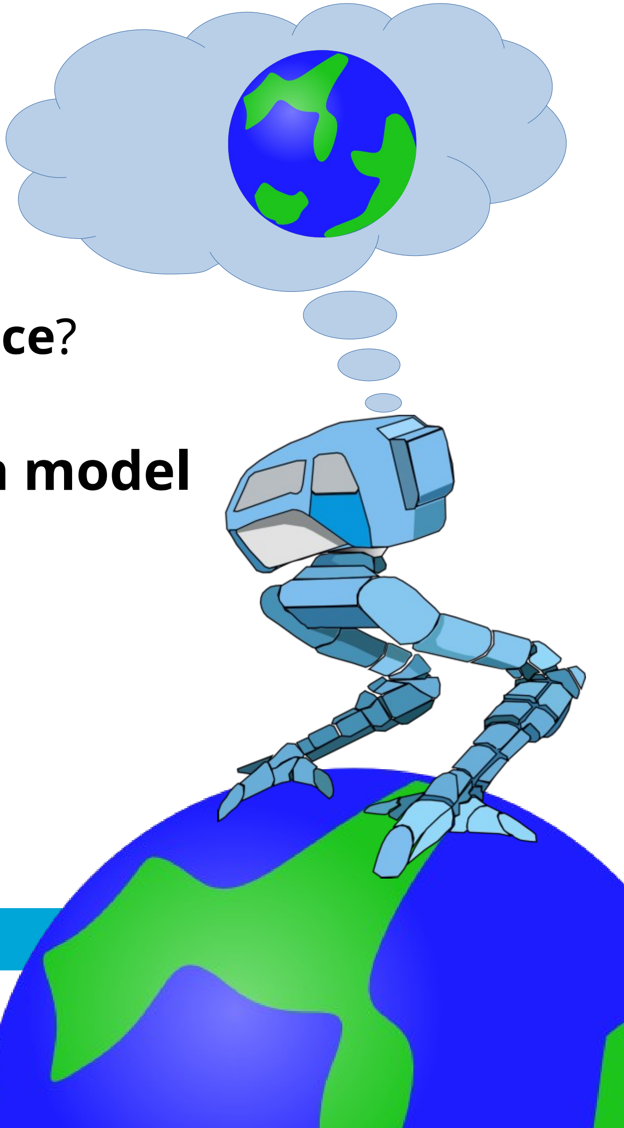
“emergent tool use”
https://openai.com/research/emergent-tool-use



Model-based RL

A Different Approach: Learning a Model

- So far ‘model-free’…
  - directly estimate Q(s,a) from samples
  - only update a little bit (learning rate α)
  - making the most of the **(precious!) experience**?
- Other approach: use the samples to **learn a model**
  - then plan
  - potentially more sample efficient
    - T(.|s,a) and R(s,a) could be easier to learn
    - e.g., are not subject to moving target
  - useful for exploration, too.

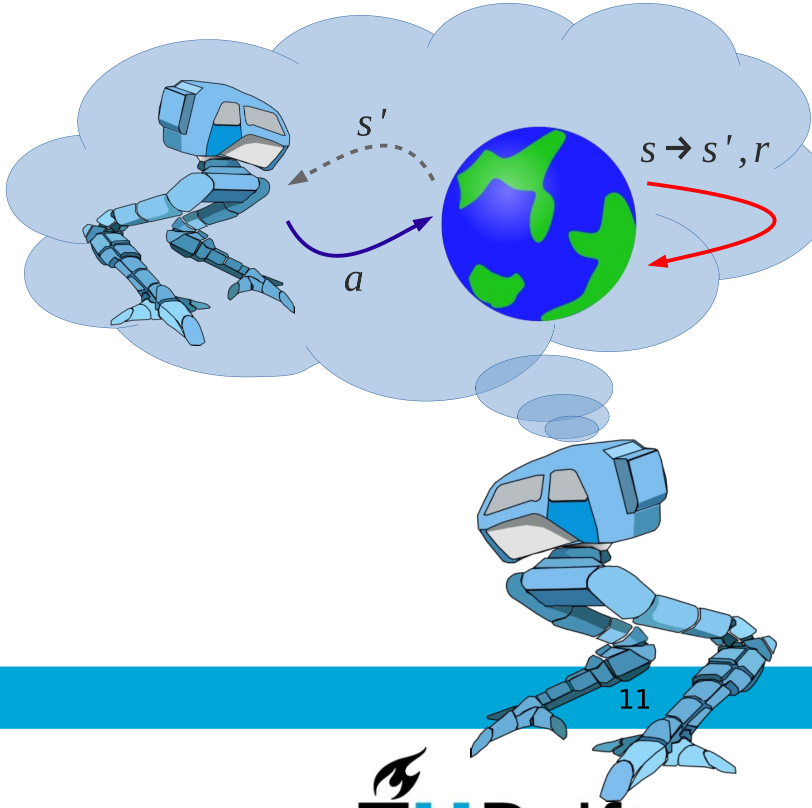


Adaptive dynamic programming–1
[RN 23.2.2]

- For now: passive learning (policy evaluation)
- Main idea, **use 'real experience' to learn models**:
  - store observed rewards
  - keep counts of transitions
- **After (s,r',s')**:
  - Store (deterministic) reward: $R(s,s') = r'$
  - $N[s]$ += 1
  - $N[s',s]$ += 1
- Induced transition function: $P(s'|s) \approx N[s',s]/N[s]$

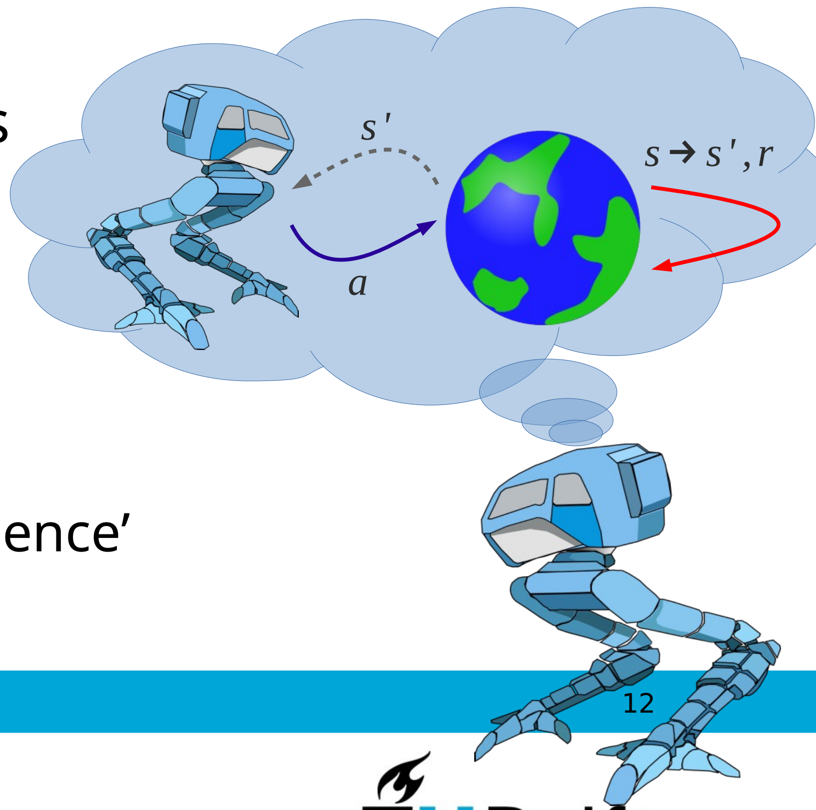
Adaptive dynamic programming–2

- Now **use the *estimated model*** R(s,s'), P(s'|s) to do policy evaluation
- Options…?



Adaptive dynamic programming–2

- Now **use the *estimated model*** R(s,s’), P(s’|s) to do policy evaluation
- Options:
  - solve system of linear equations
  - successive approximation:
    make sweeps
    $V(s) := R(s,s') + \gamma \sum_{s'} P(s'|s)\, V(s')$
  - or even TD learning!
    - but sampling ‘synthetic experience’ from the maintained model



TU Delft

How good is this...?

- Do you see any problems...?

How good is this...?

- Do you see any problems...?
  - Will it work with many data samples...?
  - With few...?

How good is this...?

- Do you see any problems...?
  - Will it work with many data samples...?
  - With few...?

- Prone to overfitting...!
  - chances of goal from this particular position?

Adding Model-Based Control
[RN: 23.3]

- Main idea unchanged: **use 'real experience' to learn models**, but now with actions
- **After (s,a,r',s')**:
  - Store (deterministic) reward: R(s,a,s') = r'
  - N[s,a] += 1
  - N[s',s,a] += 1
- Induced transition function: P(s'|s,a)≈N[s',s,a]/N[s,a]

- And then just use this model for planning…
  - value iteration, policy iteration
  - (or even SARSA, Q-learning, with *simulated* experience)

Acting using a model

- What could possibly go wrong...?

Acting using a model

- What could possibly go wrong…?
  - impact of incorrect model… can lead to loss in value (**estimation error**)
  - pure **greedy agent** might never find out that there is more reward to be found in different part of state space (**exploration**)

Acting using a model

using a **maximum likelihood** estimate of true model... prone to overfitting!
→ one solution use Bayesian estimation (of model) instead

- What could possibly go wrong...?
  - impact of incorrect model... can lead to loss in value (**estimation error**)
  - pure **greedy agent** might never find out that there is more reward to be found in different part of state space (**exploration**)

even with better (e.g., Bayesian) estimation of model...
can be states and actions that are tried insufficiently often to be confident about their effect
→ Need better exploration

TU Delft

Exploration

The Question of Exploration

- Basic question:
  - **Explore** how good things are you have not tried (often)

  or

  - **Exploit** by picking actions that were good before?

- Simplest setting: multi-armed bandits
  - n options (arms)
  - unknown means $<\mu_1,\ldots,\mu_n>$

The Question of Exploration

- Basic question:
  - **Explore** how go
    have not tried (

  or
  - **Exploit** by picki

- ►E.g., UCB uses **exploration bonus**
  - E.g., used in the "UCT" algorithm

$$U(a) = Q(a) + c\sqrt{\log(N)/N_a}$$

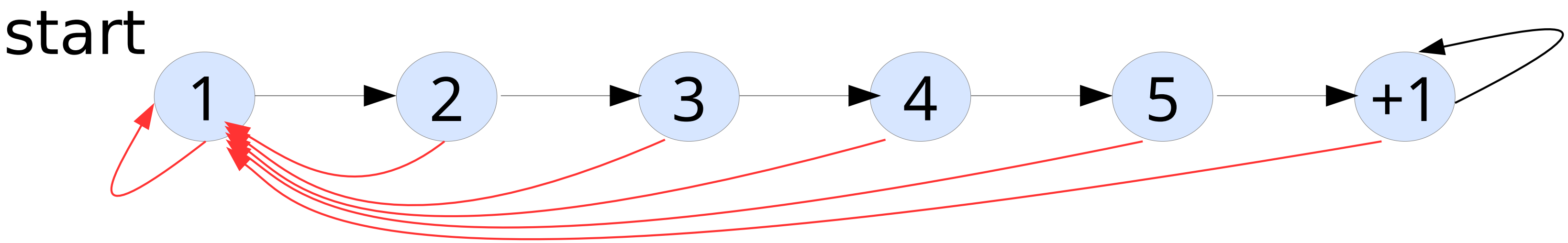
upper confidence bound　　mean return　　exploration bonus

- Simplest setting:
  multi-armed bandits
  - n options (arms)
  - unknown means $<\mu_1,\ldots,\mu_n>$

Exploration in Sequential Problems...

- We can not merely try out random (s,a) pairs...
  - need to get there first...!

- E.g., consider random exploration for a chain MDP:



Exploration in Sequential Problems: R-Max (like) methods

- Use maintained model to plan as before… but:
  - Initialize optimistically
    - R(s,a) = Rmax
    - P(s'|s,a) = **I**{ s=s' } ← assume we stay in state s
  - Mark *(s,a)* pairs 'known' IFF taken at least *m* times

- **After (s,a,r',s')**:
  - Store reward: Rset(s,a) = Rset(s,a) ∪ r'
  - Store transition: N[s',s,a] += 1, N[s,a] += 1
  - if N[s,a] == m:
    - R(s,a) := mean (Rset(s,a))
    - P(s'|s,a) := N[s',s,a] / N[s,a]
  - Plan next step with updated model

Exploration in Sequential Problems: R-Max (like) methods

- Use maintained model to plan as before… but:
  - Initialize optimistically
    - R(s,a) = Rmax
    - P(s'|s,a) = $\mathbf{I}${ s=s' } ← assume we stay in state s
  - Mark *(s,a)* pairs 'known' IFF taken at least *m* times

so we assume each unknown (s,a) pair corresponds to 'heaven'

- **After (s,a,r',s')**:
  - Store reward: Rset(s,a) = Rset(s,a) ∪ r'
  - Store transition: N[s',s,a] += 1, N[s,a] += 1
  - if N[s,a] == m:
    - R(s,a) := mean (Rset(s,a))
    - P(s'|s,a) := N[s',s,a] / N[s,a]
  - Plan next step with updated model

and we will plan to actually get to those unknown transitions

Exploration in Sequential Problems: R-Max (like) methods

- Use maintained model to plan as before… but:
  - Initialize optimistically
    - R(s,a) = Rmax
    - P(s’|s,a) = **I**{ s=s’ } ← assume we stay in state s
  - Mark *(s,a)* pairs ‘known’ IFF taken at least *m* times

so we assume each unknown (s,a) pair corresponds to ‘heaven’

- **After (s,a,r’,s’)**:
  - Store reward: Rset(s,a) = Rset(s,a) ∪ r’
  - Store transition: N[s’,s,a] += 1, N[s,a] += 1
  - if N[s,a] == m:

and we will plan to actually get to those unknown transitions

R&N formulate slightly differently, by modifying the way that the value is computed (eq. 23.5).

This boils down to pretty much the same thing.

- but R^+ needs to be an optimistic estimate of the **value** (not reward as stated)

TU Delft

Exploration Problems: R

- Use maintained model to
  - Initialize optimistically
    - R(s,a) = Rmax
    - P(s'|s,a) = **I**{ s=s' } ← a
  - Mark *(s,a)* pairs 'known' IF
- **After (s,a,r',s')**:
  - Store reward: Rset(s,a)
  - Store transition: N[s',s,a]
  - if N[s,a] == m:

R&N formulate slightly differen
way that the value is computed

This boils down to pretty much
- but R^+ needs to be an optim
  value (not reward as stated

whole algorithm:

Algorithm 1: R-MAX

Input: S, A, γ, m, ϵ_1, $R_{\max}$

$\bar{S} \leftarrow S \cup \{z\}$, where z is an arbitrary fictitious state
foreach $(s,a) \in \bar{S} \times A$ **do**
    $n(s,a) \leftarrow 0$
    $r(s,a) \leftarrow 0$
    $\bar{Q}(s,a) \leftarrow R_{\max}/(1-\gamma)$
    $\bar{R}(s,a) \leftarrow R_{\max}$
    foreach $s' \in S$ **do**
        $n(s,a,s') \leftarrow 0$
        $\bar{T}(s,a,s') \leftarrow 0$
    end
    $n(s,a,z) \leftarrow 0$
    $\bar{T}(s,a,z) \leftarrow 1$
end
for $t = 1,2,3,\dots$ **do**
    Observe current state s
    Execute action $a := \operatorname{argmax}_{a' \in A} \bar{Q}(s,a')$
    Observe immediate reward r and next state s'
    if $n(s,a) < m$ **then**
        $n(s,a) \leftarrow n(s,a)+1$
        $r(s,a) \leftarrow r(s,a)+r$
        $n(s,a,s') \leftarrow n(s,a,s')+1$
        if $n(s,a) = m$ **then**
            $\bar{R}(s,a) \leftarrow r(s,a)/m$
            foreach $s'' \in \bar{S}$ **do** $\bar{T}(s,a,s'') \leftarrow n(s,a,s'')/m$
            $\bar{Q} \leftarrow$ Solve $(\bar{S}, A, \bar{R}, \bar{T}, \gamma, \epsilon_1)$ using VI
        end
    end
end

From:

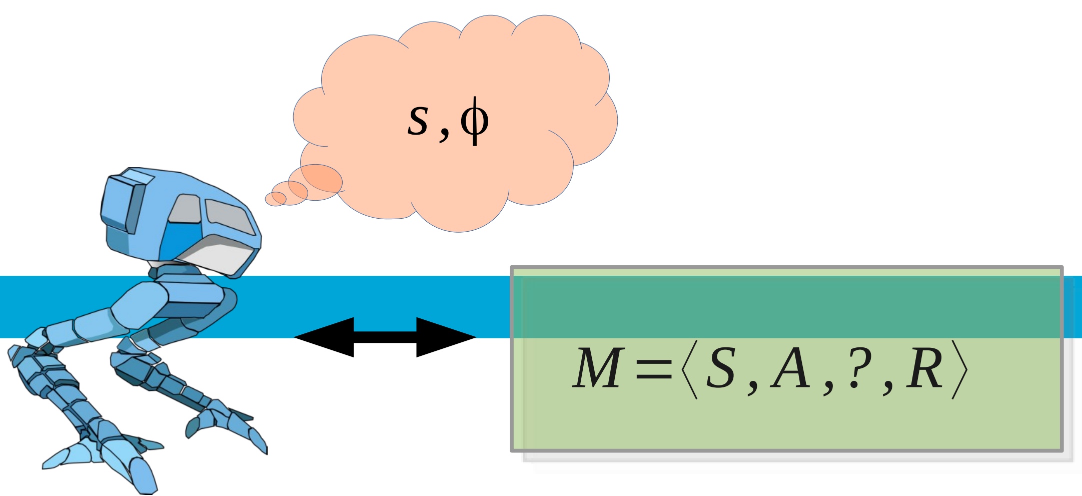
Karun Rao and Shimon Whiteson. 2012. V-MAX: tempered optimism for better PAC reinforcement learning. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12).

Bayesian RL

Beliefs over transition models
[For Dirichlet distribution, see RN 20.2.4]

- Using max. likelihood beliefs over transitions may lead to problems... → use a prior and Bayesian updating instead.

- **Bayes-Adaptive MDP** [Duff'02 PhD]
  - For each (s,a), the **distribution** over s' is uncertain...
  - Don't know the vector T_{sa} which specifies the probabilities P(. |s,a)
  - So want to maintain a belief (=prob. distr.) over T_{sa}
  - This can be represented using a Dirichlet distribution:
    - $T_{sa} \sim \text{Dir}(\langle s_1 \ldots s_{N-1} \rangle ; \alpha_1 \ldots \alpha_K)$
    - α_i indicates how often we have seen s_i' after (s,a)

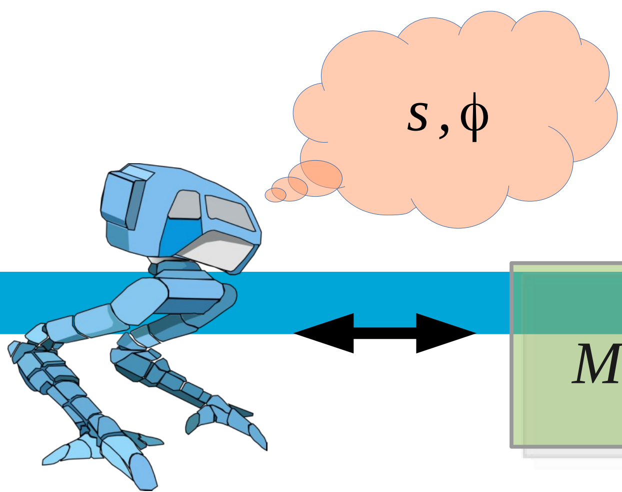


Beliefs over transition models
[For Dirichlet distribution, see RN 20.2.4]

- Using max. likelihood beliefs over transitions may lead to problems... → use a prior and Bayesian updating instead.

- **Bayes-Adaptive MDP** [Duff'02 PhD]
  - For each (s,a), the **distribution** over s' is uncertain...
  - Don’t know the vector T_{sa} which specifies the probabilities P(. |s,a)
  - So want to maintain a belief (=prob. distr.) over T_{sa}
  - This can be represented using a Dirichlet distribution:
    - $T_{sa} \sim \text{Dir}(\langle s_1 \ldots s_{N-1} \rangle ; \alpha_1 \ldots \alpha_K)$
    - α_i indicates how often we have seen s_i' after (s,a)

so just counts $N[s_i', s, a]$



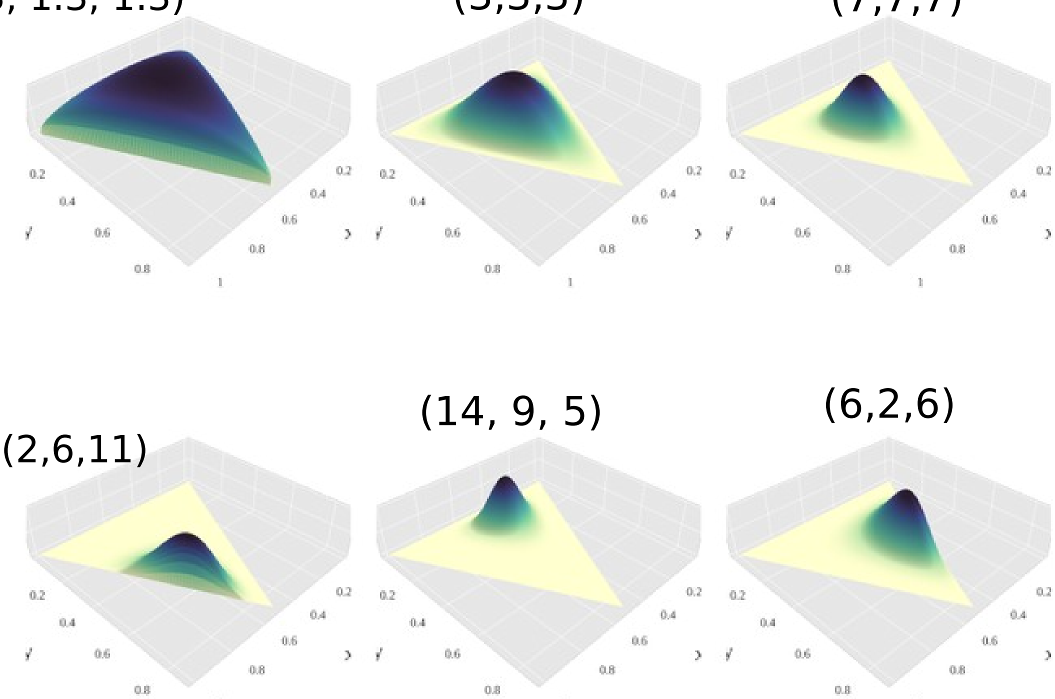
$$M = \langle S, A, ?, R \rangle$$

Beliefs over transition models

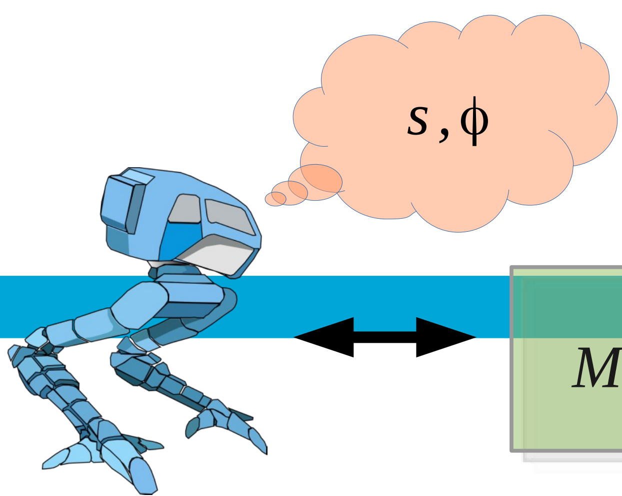
[For Dirichlet distribution, see RN 20.2.4]

- Using max. likelihood beli
  a prior and Bayesian upda
- **Bayes-Adaptive MDP** [Du
  - For each (s,a), the **distribu**
  - Don't know the vector T_{sa} v
  - So want to maintain a beli
  - This can be represented u
    - T_{sa} ~ Dir(<s_1...s_{N-1}> ; α_1..
    - α_i indicates how often w

Dirichlet distributions
- Over 2D simplex (so think of 3 next states)
- α vectors are indicated



(source: "Empetrisor" Wikipedia)



$$M = \langle S, A, ?, R \rangle$$

BA-MDPs

- Let $\alpha=(\alpha_1,\ldots,\alpha_Z)$ denote the collection of all count vectors
  - for all Z state action pairs
- Main idea: use 'augmented states' $<s,\alpha>$

BA-MDPs

- Let $\alpha=(\alpha_1,\ldots,\alpha_Z)$ denote the collection of all count vectors
  - for all *Z* state action pairs
- Main idea: use 'augmented states' $<s,\alpha>$
- Define transitions:
  - $P(<s',\alpha'> \mid <s,\alpha>, a) = P(s' \mid \alpha_{sa}) * I \{ \alpha_{sa}'(s') == \alpha_{sa}(s') +1 \}$

TUDelft

BA-MDPs

prob. of s' according to counts:

$P(s' \mid \alpha_{sa}) = \alpha_{sa}(s') / \Sigma_{s'} \alpha_{sa}(s')$

ection of all count vectors

- for all Z state action pairs

- Main idea: use 'augmented states' $\langle s,\alpha \rangle$
- Define transitions:
  - $P(\langle s',\alpha' \rangle \mid \langle s,\alpha \rangle, a) = P(s' \mid \alpha_{sa}) * I \{ \alpha_{sa}'(s') == \alpha_{sa}(s') +1 \}$

BA-MDPs

prob. of s’ according to counts:

$P(\, s' \mid \alpha_{sa}\,) = \alpha_{sa}(s') \,/\, \Sigma_{s'}\, \alpha_{sa}(s')$

ect

- for all Z state action pairs

‘indicator function’

- is 1 IFF we update counts correctly
- 0 otherwise

- Main idea: use ‘augmented state
- Define transitions:
  - $P(\langle s',\alpha'\rangle \mid \langle s,\alpha\rangle, a) = P(\, s' \mid \alpha_{sa}\,) * I\{\, \alpha_{sa}'(s') == \alpha_{sa}(s') + 1 \,\}$

BA-MDPs

- Let $\alpha=(\alpha_1,\ldots,\alpha_Z)$ denote the collection of all count vectors
  - for all *Z* state action pairs
- Main idea: use 'augmented states' $<s,\alpha>$
- Define transitions:
  - $P(<s',\alpha'> \mid <s,\alpha>, a) = P(s' \mid \alpha_{sa}) * I\{ \alpha_{sa}'(s') == \alpha_{sa}(s') + 1 \}$

- Together this defines a new MDP!

Learning as Planning

We have now casted the **learning** problem **as a planning** problem!

Learning as Planning

We have now casted the **learning** problem **as a planning** problem!

- Given the prior $b_0(\alpha)$
- the BA-MDP is fully specified
- optimal plan π^* would map <s,α> → *a*
  - **optimal** tradeoff: exploration vs. exploitation

Learning as Planning

We have now casted the **learning** problem **as a planning** problem!

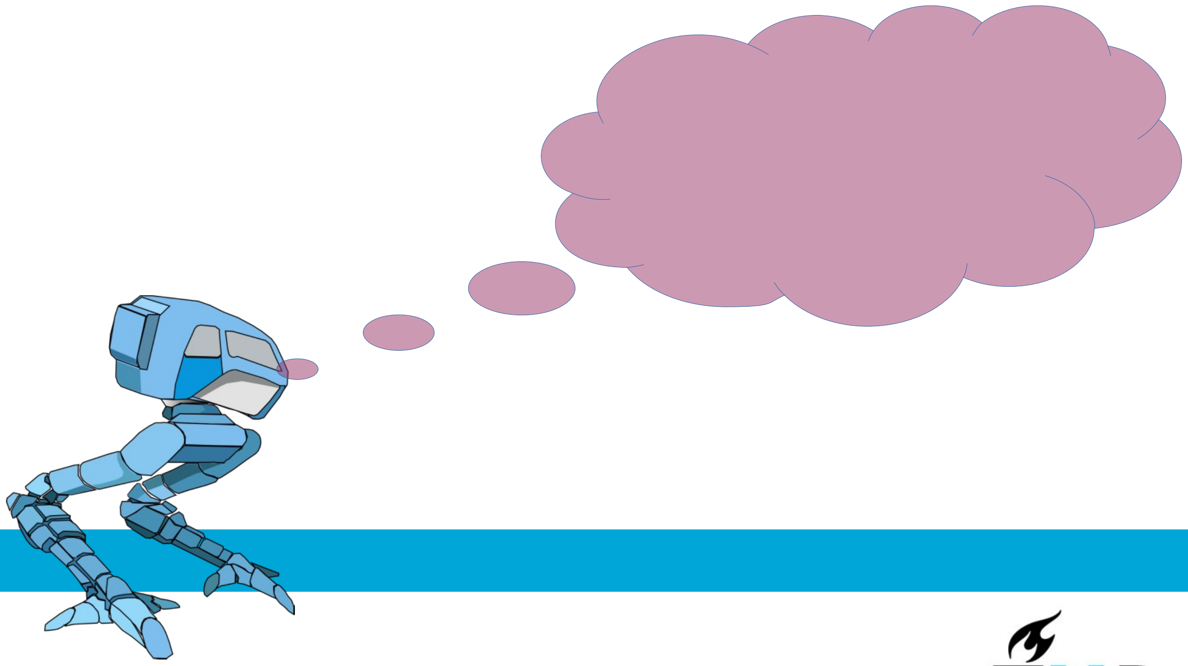
- Given the prior $b_0(\alpha)$
- the BA-MDP is fully specified
- optimal plan π^* would map $\langle s,\alpha \rangle \rightarrow a$
  - **optimal** tradeoff: exploration vs. exploitation

- Difficulty: infinite state space
  - (countably) infinite set of counts...
- But can do online planning
  - e.g., MCTS!

TU Delft

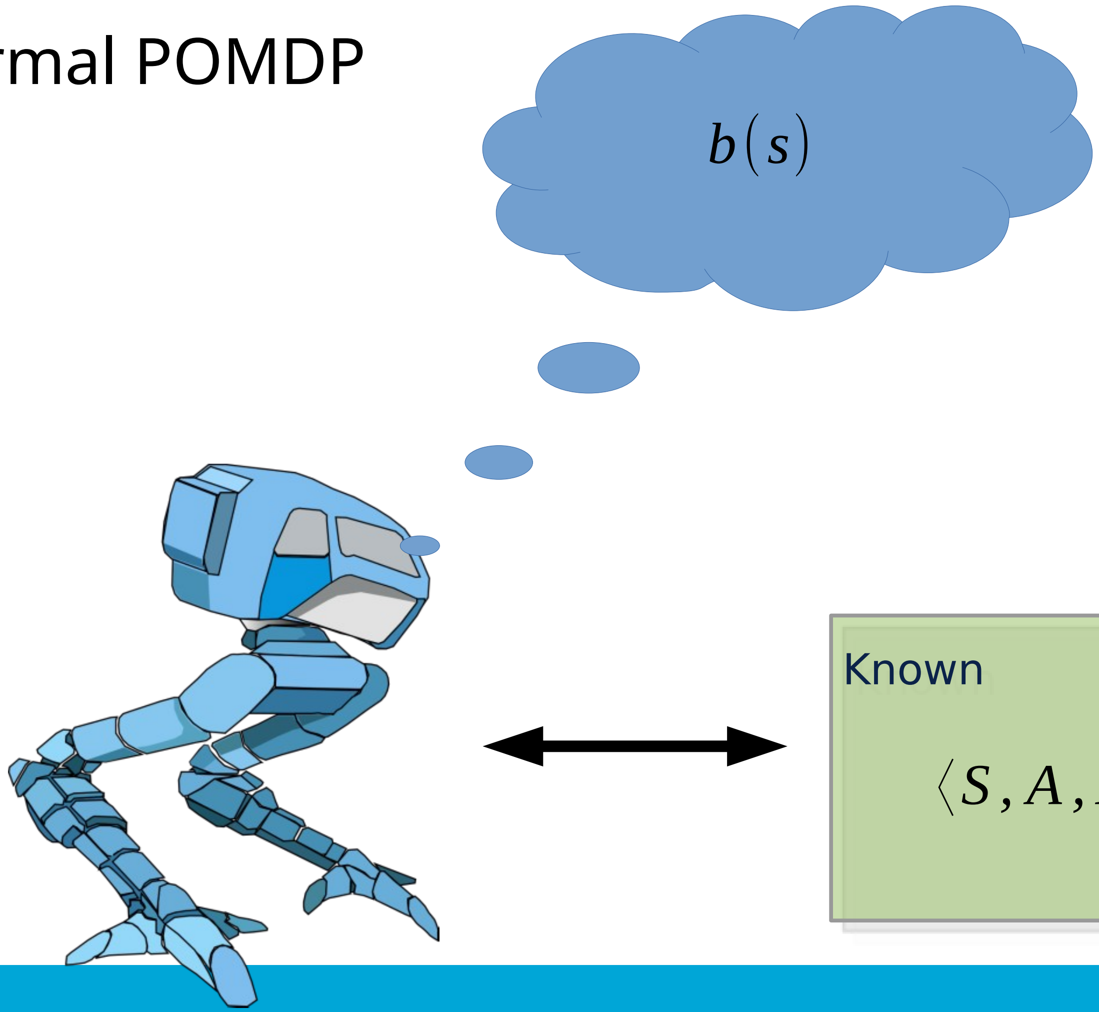
State & Model Uncertainty:

Bayesian RL for POMDPs via BA-POMDPs



POMDP

- Normal POMDP

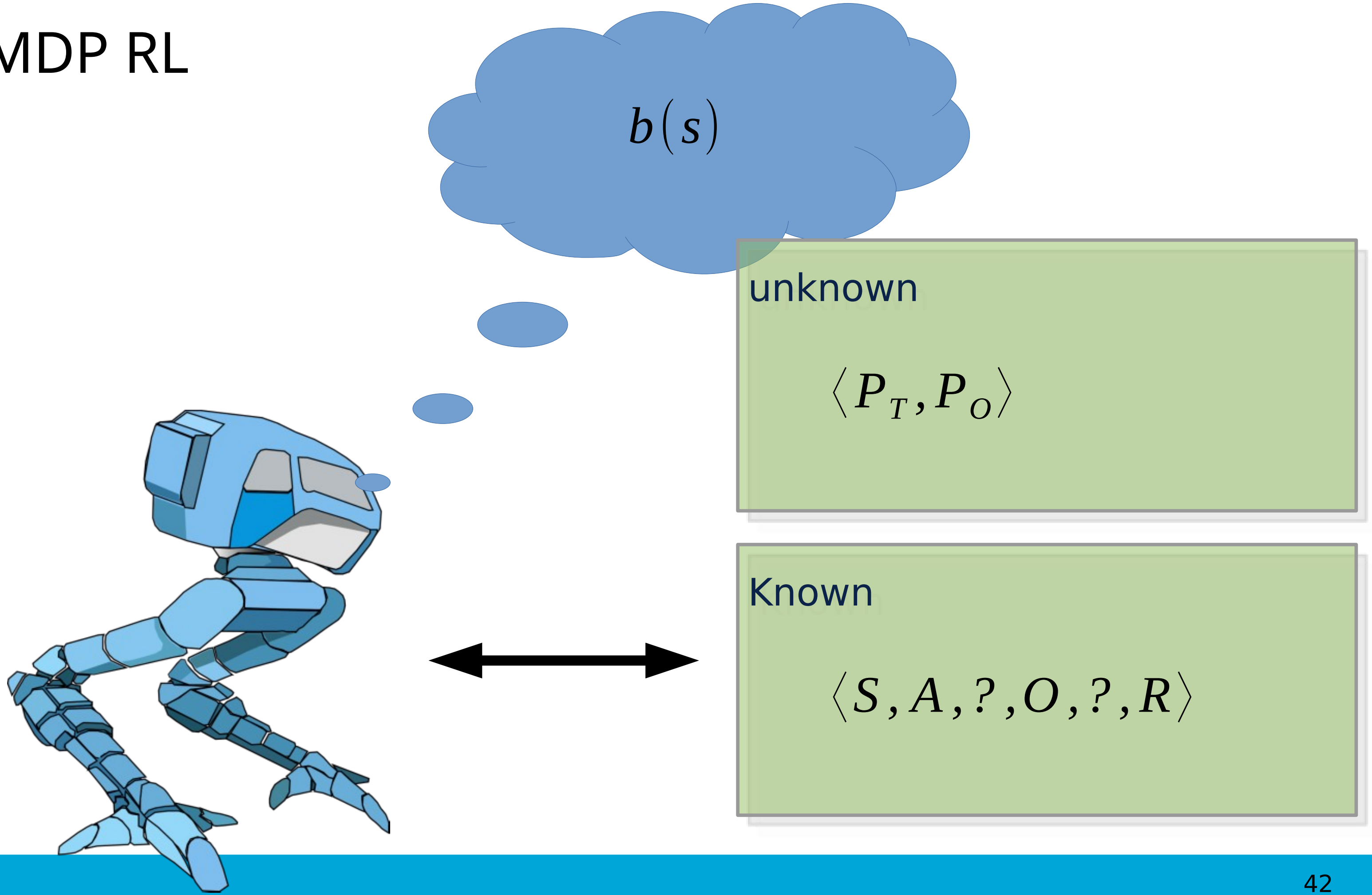


Known

$$\langle S, A, P_T, O, P_O, R \rangle$$

POMDP RL

- POMDP RL

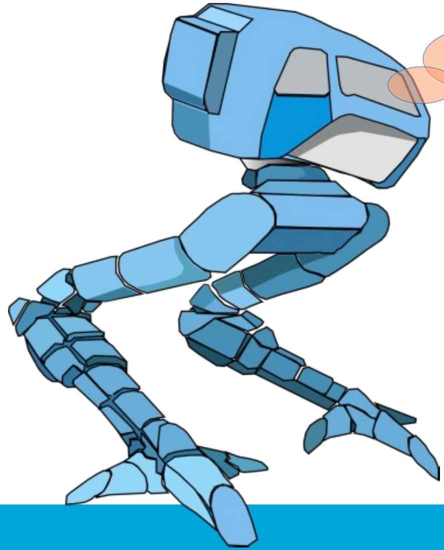


TU Delft

POMDP BRL

- Bayesian POMDP RL

$$b(s, P_T, P_O)$$

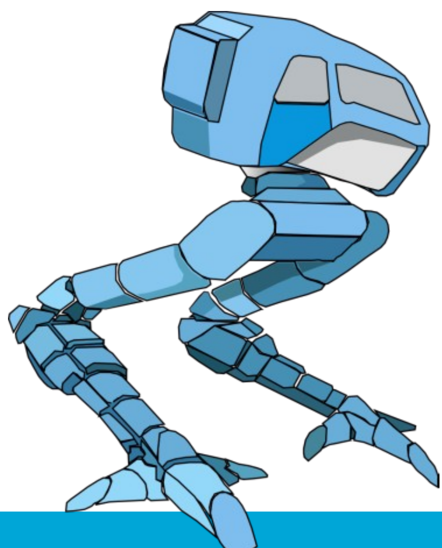


Known

$$\langle S, A, ?, O, ?, R \rangle$$

BA-POMDPs

- If we *could* observe states → could maintain counts
  - for transitions φ
  - for observations ψ



unknown

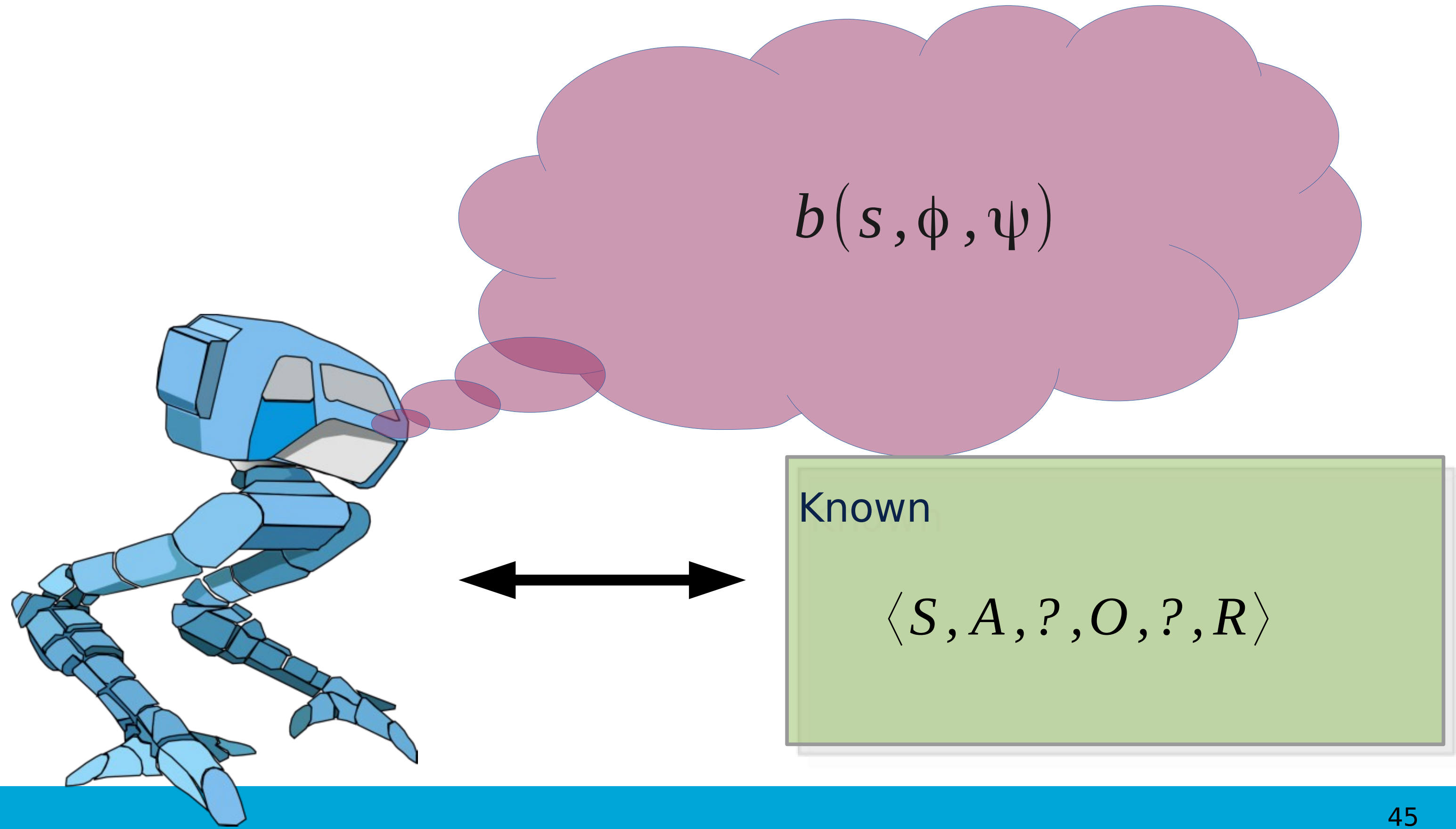
$$s, \phi, \psi$$

Known

$$\langle S, A, ?, O, ?, R \rangle$$

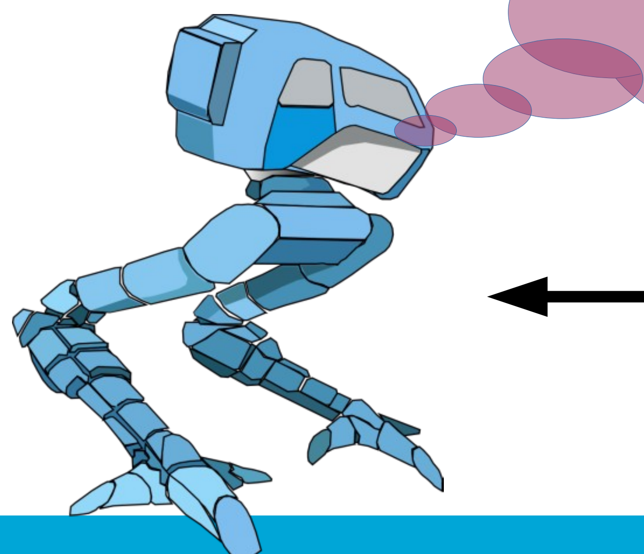
BA-POMDPs

- If we are Bayesian about it… maintain beliefs!



BA-POMDPs

- Can maintain beliefs... treat as a large POMDP!
  [Ross et al. JAIR'11]
- Augmented states:
  s_{BP}=<s,φ,ψ>



$$b(s,\phi,\psi)$$

Known: a **BA-POMDP**

$$M_{BP}=\langle S_{BP},A,T_{BP},O,O_{BP},R\rangle$$

$$T_{BP}(\langle s',\phi',\psi'\rangle|\langle s,\phi,\psi\rangle,a)$$

$$O_{BP}(o|a,\langle s',\phi',\psi'\rangle)$$

Bayesian RL… why do we care?

- In current day and age… why do we care?
  - Just throw into a neural network of some sorts…?
- Deep RL methods…
  - **sample inefficient**! no way to apply without simulator?
  - not clear how to use **prior knowledge**?
  - **exploration**?
- Bayesian RL methods (BA-MDPs) …
  - (in principle) **optimally** trade off exploration vs exploitation!
  - optimal w.r.t. prior knowledge
    → much more sample efficient!
- And of course, these ideas might be married! (“pseudo counts” etc.)

Bayesian RL… why do we care?

- In current day and age… why do we c
  - Just throw into a neural network of so

- Deep RL methods…
  - **sample inefficient**! no way to apply w
  - not clear how to use **prior knowledg**
  - **exploration**?

- Bayesian RL methods (BA-MDPs) …
  - (in principle) **optimally** trade off exploration vs exploitation!
  - optimal w.r.t. prior knowledge
    → much more sample efficient!

- And of course, these ideas might be married! ("pseudo counts" etc.)

Interesting? I do active research in this direction.

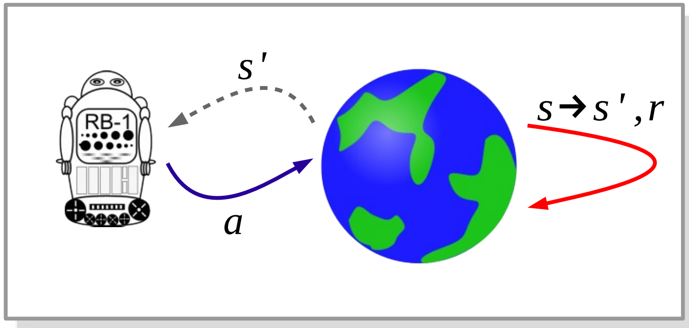
contact me if interested.

(or check out
https://www.fransoliehoek.net/wp/teaching/student-projects/
to get an idea of other projects.)

TU Delft

Summary

Summary



- RL: when we don't have a model
- Model-based RL
  - estimate a model and use to plan
  - passive and active learning settings
- Exploration
  - even for model-based methods
  - and can be more effective given a model
- Bayesian RL
  - "regularization" of ML estimates of models
  - but also: optimal trade off of exploration vs exploitation