# Reinforcement Learning: State of the Art & Challenges


Frans A. Oliehoek

Dept. of Intelligent Systems


ML Bootcamp, 14-2-2023, CWI

# Reinforcement Learning

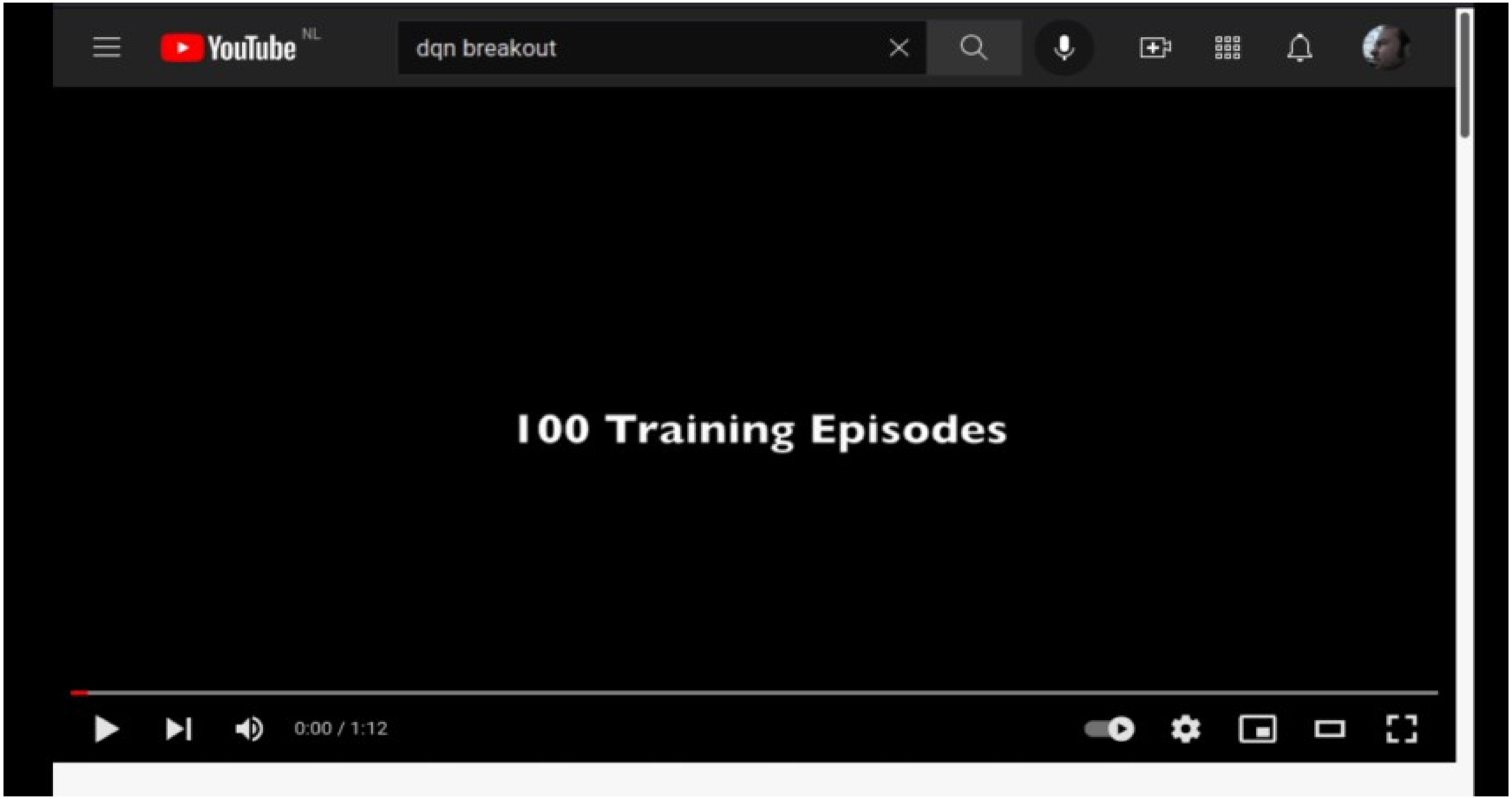
- Who knows...
  - what an MDP is?
  - what RL is?
  - what DQN is?
  - how DQN works?
  - how MCTS works?
  - how alpha Go works?

# Reinforcement Learning

- Who knows...
  - what an MDP is?
  - what RL is?
  - what DQN is?
  - how DQN works?
  - how MCTS works?
  - how alpha Go works?

Let’s start with a glimpse...

# Breakout: DQN (2013)



# Alpha Go – Deepmind (2016)

The Telegraph

HOME | NEWS | S

Science

Science

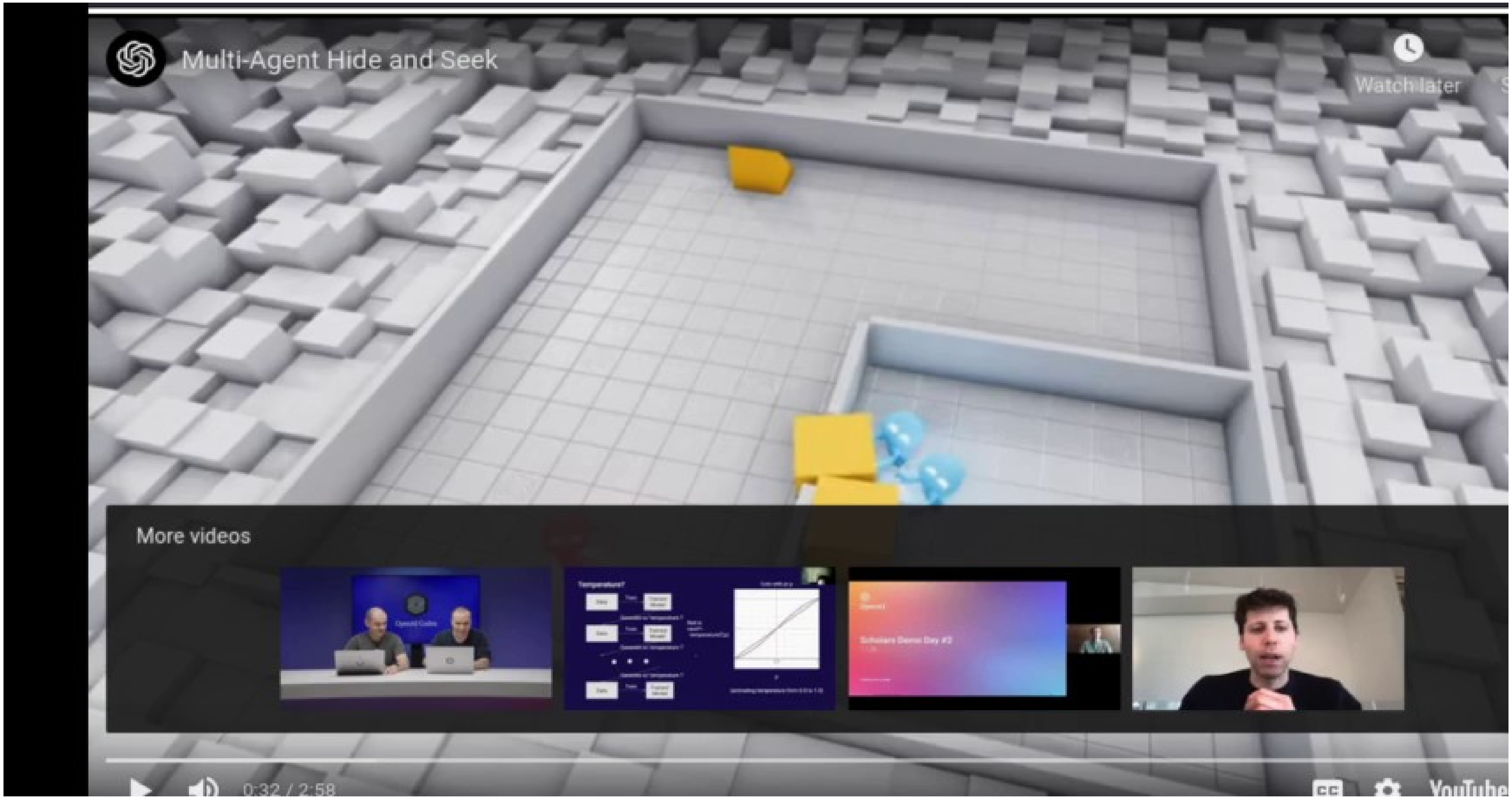
## AlphaGo Zero: Google DeepMind supercomputer learns 3,000 years of human knowledge in 40 days

share

16


Chinese Go player Ke Jie competes against Google's artificial intelligence (AI) program, AlphaGo CREDIT: IMAGINECHINA/REX/SHUTTERSTOCK

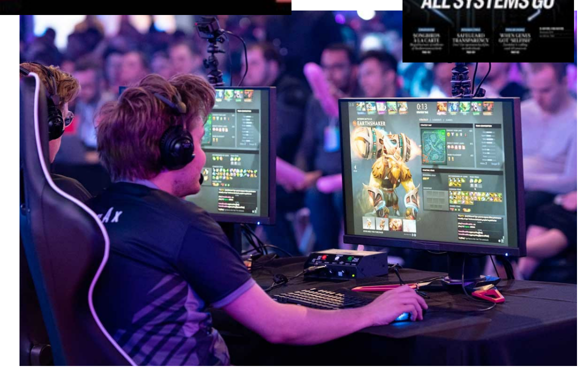
# Hide and Seek – OpenAI (2019)

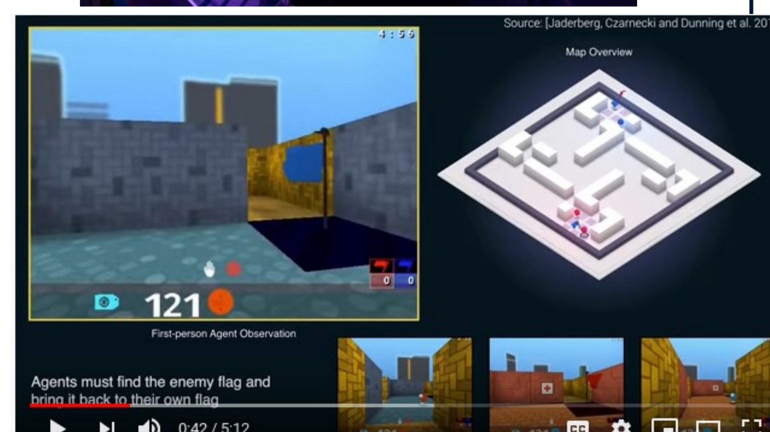


# Glimpse of the State of the Art...

- “Deep RL”: Combination of RL techniques with deep neural networks

- Many recent results:
  - Atari Breakout
  - Go, Poker
  - Dota 2 / Starcraft
  - Simulated Robotics/Locomotion
  - Hide and Seek
  - Capture the flag
  - Chip Design
  - Summarizing books & ChatGPT
  - ‘Generally capable agents’

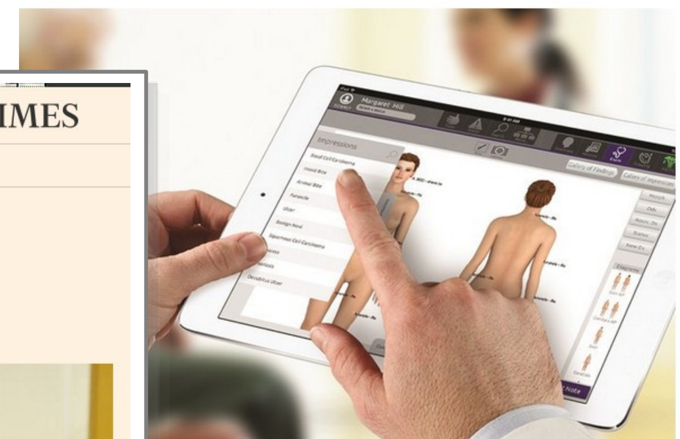


# Vision: improving the world

Brussels traffic jams, the biggest cause of air pollution, carry a yearly cost of 511 million Euros

POSTED ON JULY 23, 2015 CATEGORIES: ECONOMIE/ECONOMY, UNCATEGORIZED NO COMMENTS YET

ARTIFICIAL INTELLIGENCE IS NOW TELLING DOCTORS HOW TO TREAT YOU



...SY OF MODERNIZING MEDICINE

...LAND DERMATOLOGIST Kavita Mariwalla knows how ... acne, burns, and rashes. But when a patient came in

## Rangers Use Artificial Intelligen... to Fight Poachers

Emerging technology may help wildlife officials beat back traffickers.

Antipoaching patrols like this team at the Lewa Wildlife Conservancy in Kenya may soon use AI technology to stay one step ahead of criminals.

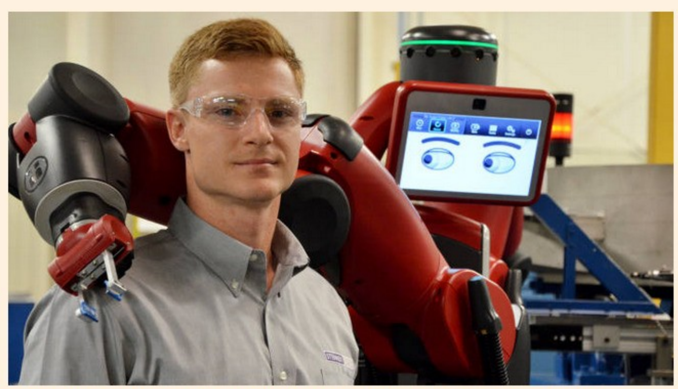
PHOTOGRAPH BY AMI VITALE, NATIONAL GEOGRAPHIC CREATIVCE

FINANCIAL TIMES

...D US COMPANIES MARKETS OPINION WORK & CAREERS LIFE & ARTS

Artificial Intelligence and Robotics + Add to myFT

## Meet the cobots: humans and robots together on the factory floor



Lending hand: mechanical engineer Jesse Rochelle works with Baxter at the Stenner Pumps factory in Jacksonville, Florida © FT

13 Save

MAY 5, 2016 by: Peggy Hollinger, Industry Editor

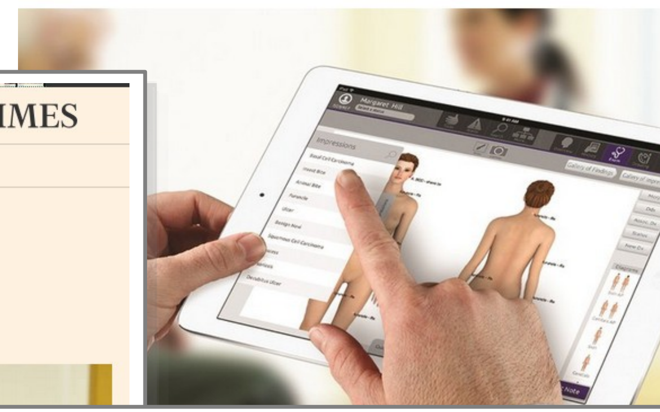
Walking across the floor of SEW-Eurodrive's factory in Baden-Württemberg is

# Vision: improving the world

Brussels traffic jams, the biggest cause of air pollution, carry a yearly cost of 511 million Euros

POSTED ON JULY 23, 2015 CATEGORIES: ECONOMIE/ECONOMY, UNCATEGORIZED NO COMMENTS YET

ARTIFICIAL INTELLIGENCE IS NOW TELLING DOCTORS HOW TO TREAT YOU

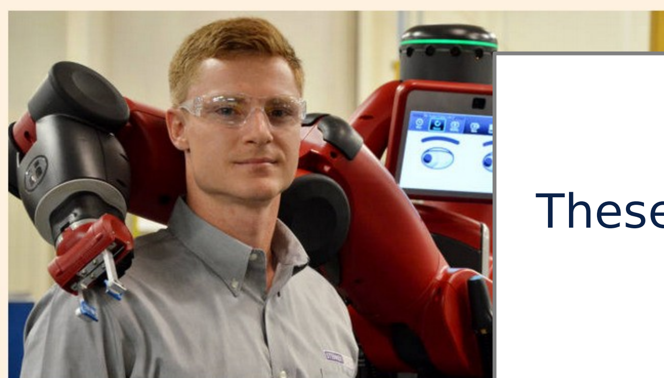


FINANCIAL TIMES

.D US COMPANIES MARKETS OPINION WORK & CAREERS LIFE & ARTS

Artificial Intelligence and Robotics + Add to myFT

Meet the cobots: humans and robots together on the factory floor



Lending hand: mechanical engineer Jesse Rochelle works with Baxter at the Stenne
Jacksonville, Florida © FT

MAY 5, 2016 by: Peggy Hollinger, Industry Editor

Walking across the floor of SEW-Eurodrive's factory in Bade

Rangers Use Artificial Intelligen
to Fight Poachers

Emerging technology may help wildlife officials beat back traffickers.

Antipoaching patrols like this team at the Lewa Wildlife Conservancy in Kenya may soon use AI technology to stay one step ahead of criminals.
PHOTOGRAPH BY AMI VITALE, NATIONAL GEOGRAPHIC CREATIVCE

These are all tasks of a sequential nature…

→ reinforcement learning can potentially make a big impact

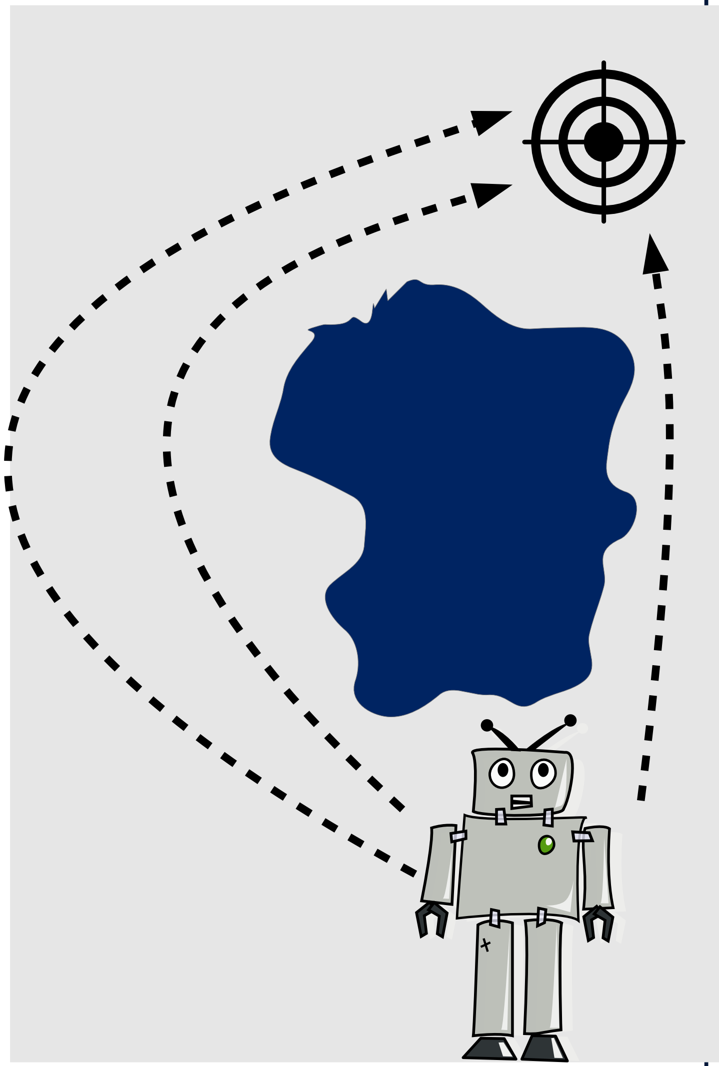
# Outline for Today

- ~~Teaser~~
- **Foundations of RL**
- Intuition behind state of the art
- Challenges**

**Disclaimer: I took many examples from my own research, but this is only a very small sample.

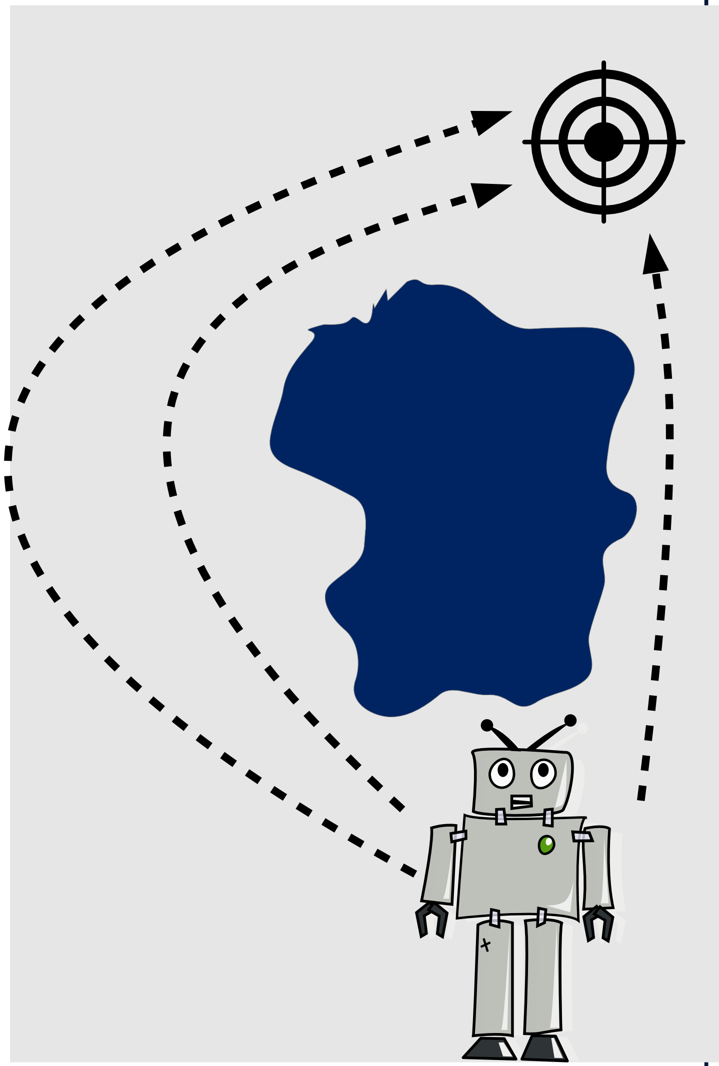
# Sequential Decision Making (SDM)

- Actions over multiple time steps
- SDM problems are complex...
  - **immediate** vs **long-term benefits**
  - deal with **uncertainties** (stochasticity, partial information)



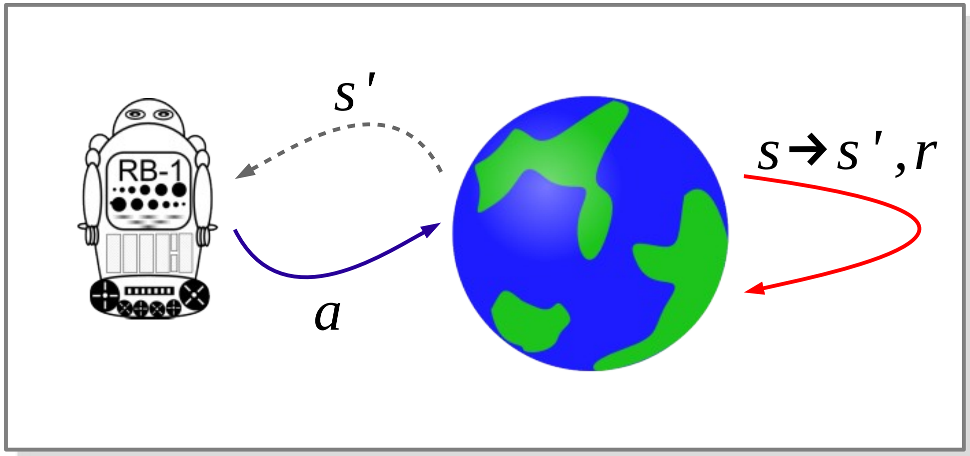
# Sequential Decision Making (SDM)

- Actions over multiple time steps
- SDM problems are complex...
  - **immediate** vs **long-term benefits**
  - deal with **uncertainties** (stochasticity, partial information)
- Manual programming is difficult
  - Instead: “programming via rewards”
  - planning / reinforcement learning



# Complex decisions over time

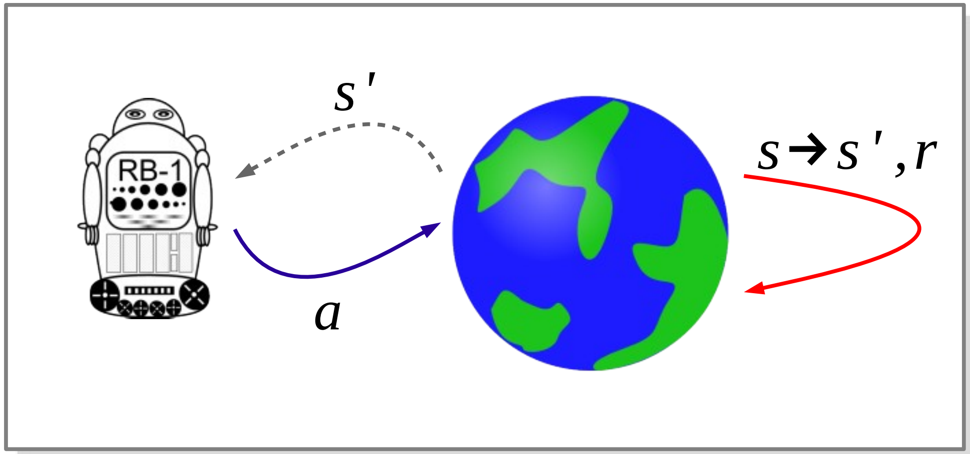
- Formalized as **Markov decision process (MDP)**
  - states (s), actions (a), rewards (r)




  - states are observed
  - but transitions are stochastic: P(s’ | s, a)
  - and rewards could be too: r ~ R(s,a)

# Complex decisions over time

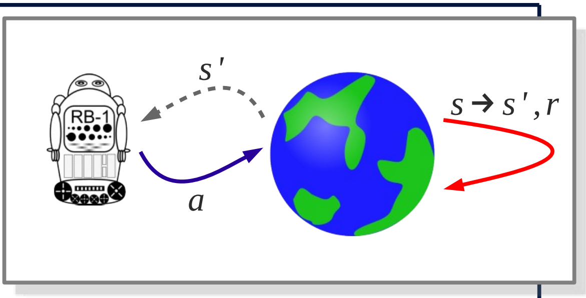
- Formalized as **Markov decision process (MDP)**
  - states (s), actions (a), rewards (r)




  - states are observed
  - but transitions are stochastic: P(s' | s, a)
  - and rewards could be too: r ~ R(s,a)

  - OK, so how to
    - **balance short-term vs long-term rewards**
    - taking into account the **uncertainty**
      ?

# MDP Objective

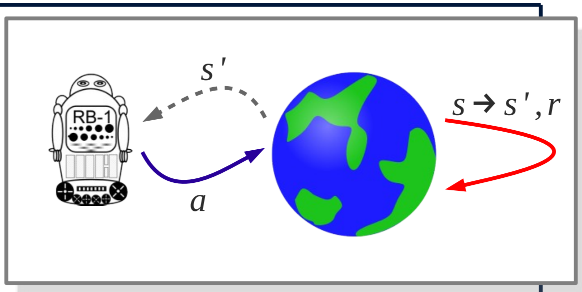



- Goal: optimize the '**value**' of a policy π:
  - i.e., expected (discounted) sum of rewards

$$V(\pi) = E[\ \Sigma_t\ \gamma^t * R(s,a)\ \mid \pi\ ]$$

- Task is **planning:**
  - compute a good/optimal policy π
  - given the model (or a simulator)

# MDP Objective




- Goal: optimize the '**value**' of a policy π:
  - i.e., expected (discounted) sum of rewards

    $V(\pi) = E[\ \Sigma_t\ \gamma^t * R(s,a)\ \ |\ \pi\ ]$

- Task is **planning:**
  - compute a good/optimal policy π
  - given the model (or a simulator)

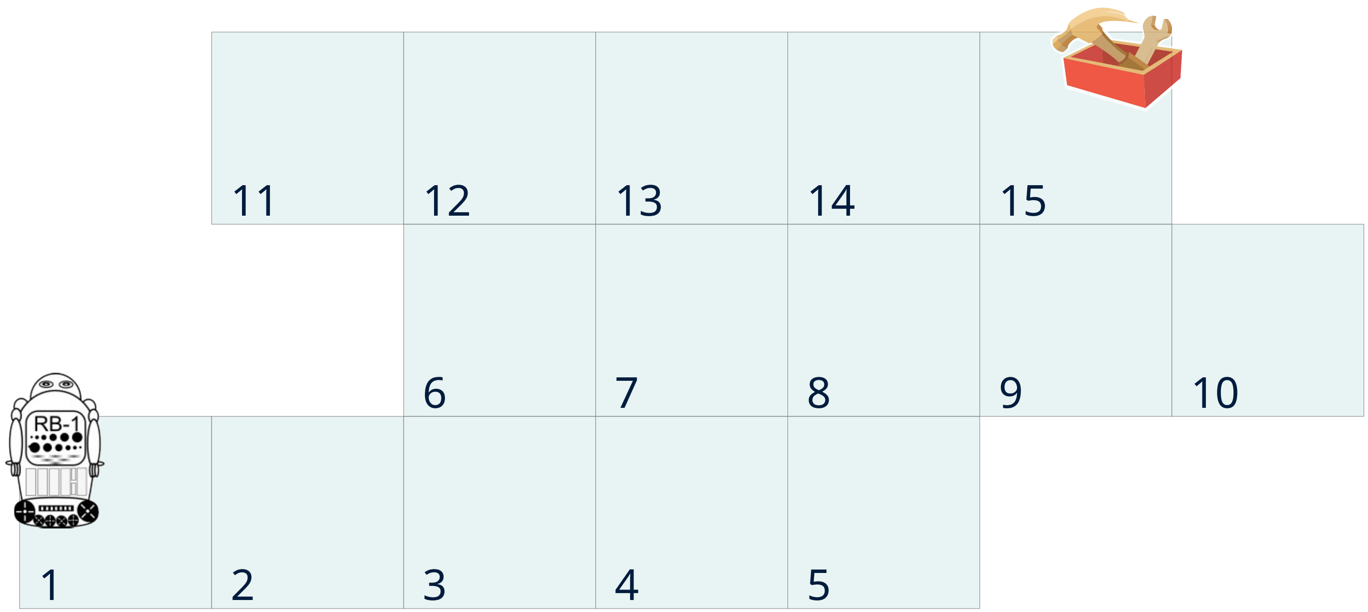
- Typical approach:
  compute 'optimal Q-value function' Q*(s,a)
  - expresses expected value given s,a
  - Bellman optimality equation:

    $$Q^*(s,a) = R(s,a) + \gamma\ \Sigma_{s'}\ P(s'|s,a) V^*(s')$$

    - where

      $V^*(s) = \max_a Q^*(s,a)$

# Example: pick up the toolbox



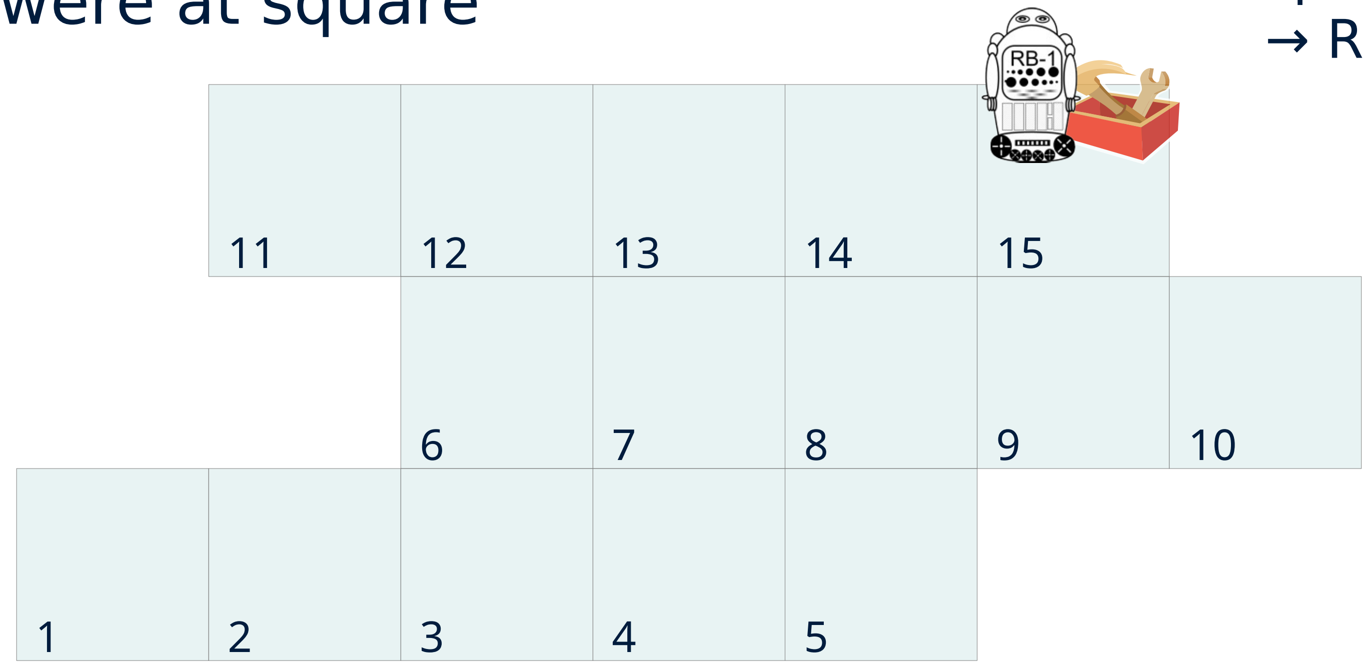

- $Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$
- $V^*(s) = \max_a Q^*(s,a)$

Robot needs to go to toolbox, and pick it up.
→ reward: +1
→ let's assume $\gamma=0.9$
→ and deterministic movement

# Example: pick up the toolbox

if we were at square 15...

“pickup”
→ R=+1

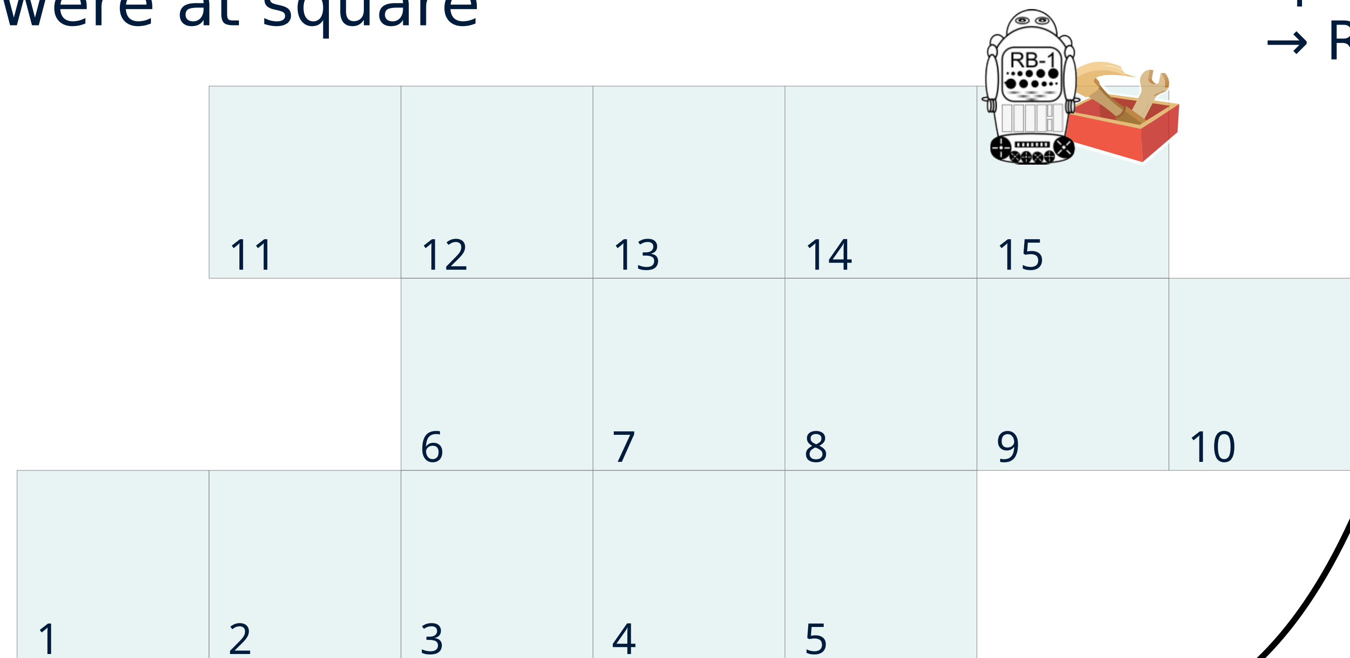



- $Q^*(s,a) = R(s,a)+\gamma \sum_{s'} P(s'|s,a)V^*(s')$
- $V^*(s) = \max_a Q^*(s,a)$

# Example: pick up the toolbox

if we were at square 15...

“pickup” → R=+1



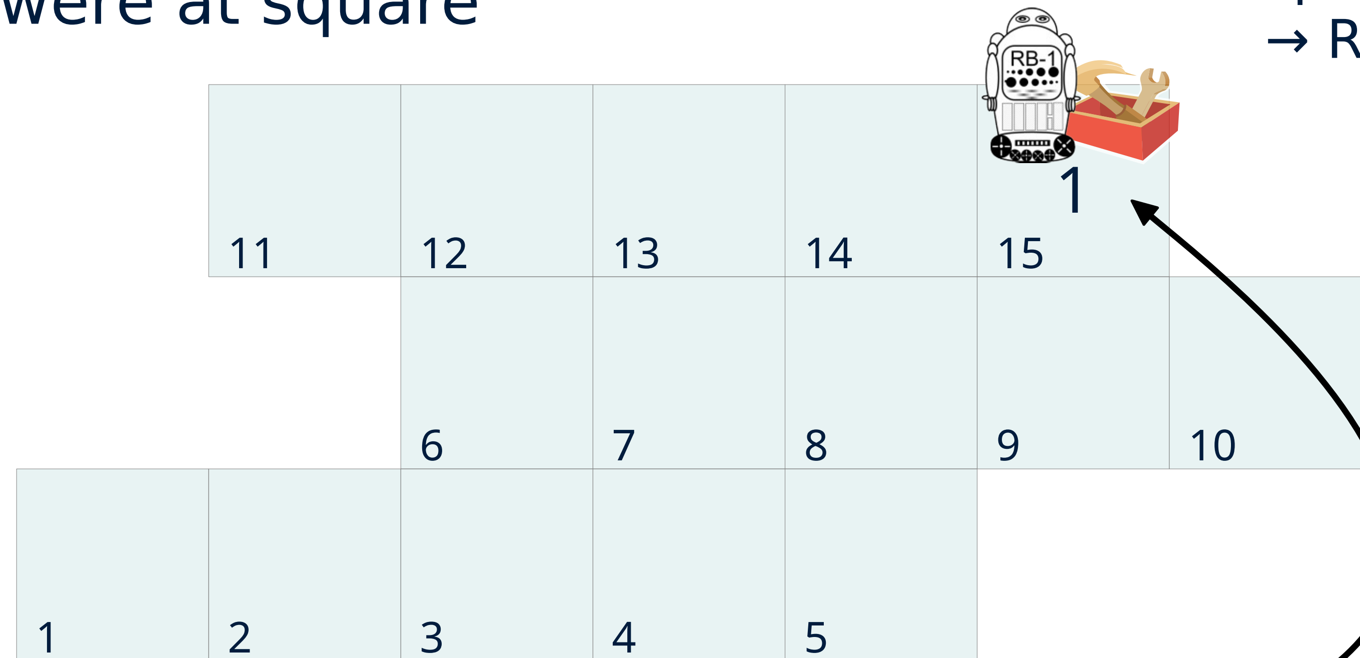

Q*(s=15, a=pickup) = 1

- Q*(s,a) = R(s,a)+γ $\Sigma_{s'}$ P(s’|s,a)V*(s’)
- V*(s) = $\max_a$ Q*(s,a)

# Example: pick up the toolbox

if we were at square 15…

“pickup” → R=+1




$Q^*(s=15, a=\text{pickup}) = 1$

$V^*(s=15) = 1$

- $Q^*(s,a) = R(s,a)+\gamma \sum_{s'} P(s'|s,a)V^*(s')$
- $V^*(s) = \max_a Q^*(s,a)$

# Example: pick up the toolbox

if we were at 14?

“right” → R=0


- Q*(s,a) = R(s,a)+γ $\Sigma_{s'}$ P(s’|s,a)V*(s’)
- V*(s) = $\max_a$ Q*(s,a)

Q*(s=15, a=pickup) = 1
V*(s=15) = 1

# Example: pick up the toolbox

if we were at 14?

“right” → R=0


- $Q^*(s,a) = R(s,a)+\gamma \sum_{s'} P(s'|s,a)V^*(s')$
- $V^*(s) = \max_a Q^*(s,a)$

$Q^*(s=15, a=pickup) = 1$

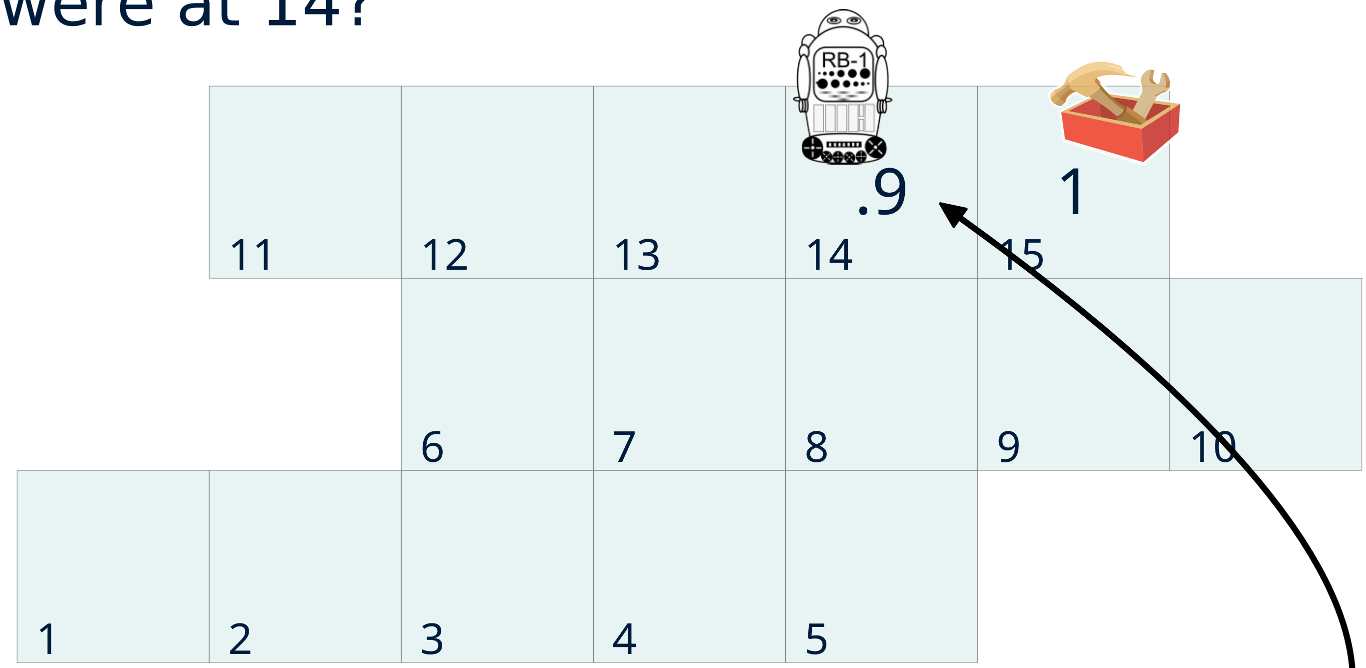
$V^*(s=15) = 1$

$Q^*(s=14, a=right) = 0 + 0.9 * 1 = 0.9$

# Example: pick up the toolbox

“right” → R=0

if we were at 14?




- $Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$
- $V^*(s) = \max_a Q^*(s,a)$

$Q^*(s=15, a=\text{pickup}) = 1$

$V^*(s=15) = 1$

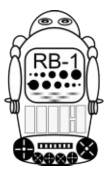
$Q^*(s=14, a=\text{right}) = 0 + 0.9 * 1 = 0.9$

$V^*(s=14) = 0.9$

# Example: pick up the toolbox

compute value of any location

| | | | | | | |
|---|---|---|---|---|---|---|
| | .66 (11) | .73 (12) | .81 (13) | .9 (14) | 1 (15) | |
| | | .66 (6) | .73 (7) | .81 (8) | .9 (9) | .81 (10) |
| .48 (1) | .53 (2) | .59 (3) | .66 (4) | .73 (5) | | |



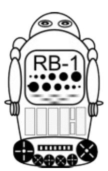

- $Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$
- $V^*(s) = \max_a Q^*(s,a)$

Q*(s=15, a=pickup) = 1
V*(s=15) = 1
Q*(s=14, a=right) = 0 + 0.9 * 1 = 0.9
V*(s=14) = 0.9
...
...
Q*(s=1, a=right) = 0 + 0.9 * .53 = 0.48
V*(s=1) = 0.48

# e toolbox

This (form of planning) is also called
**“dynamic programming”**

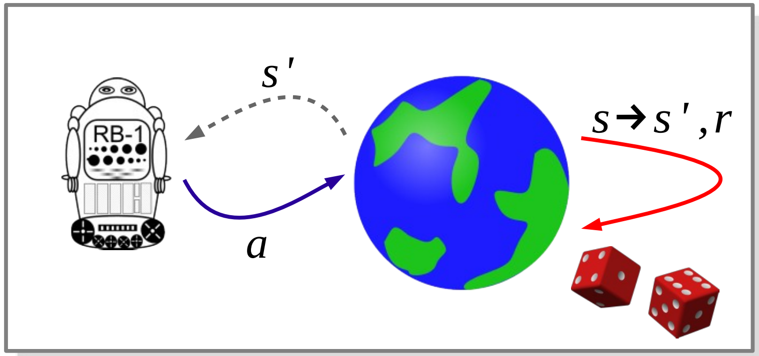
| | | | | | | |
|---|---|---|---|---|---|---|
| | .66 11 | .73 12 | .81 13 | .9 14 | 1 15 | |
| | | .66 6 | .73 7 | .81 8 | .9 9 | .81 10 |
| .48 1 | .53 2 | .59 3 | .66 4 | .73 5 | | |




- $Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$
- $V^*(s) = \max_a Q^*(s,a)$

Q*(s=15, a=pickup) = 1
V*(s=15) = 1
Q*(s=14, a=right) = 0 + 0.9 * 1 = 0.9
V*(s=14) = 0.9
…
…
Q*(s=1, a=right) = 0 + 0.9 * .53 = 0.48
V*(s=1) = 0.48

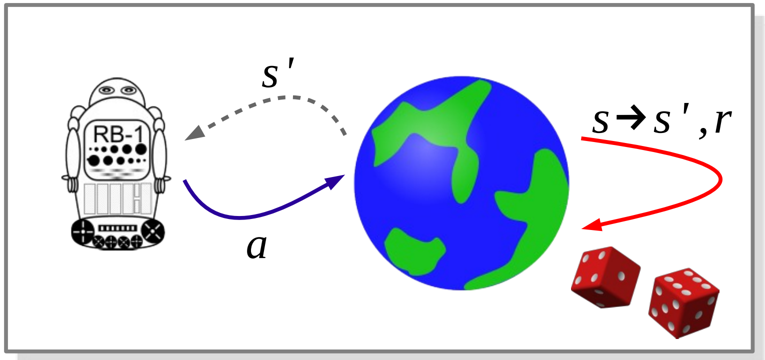
# MDP Planning




- Given an MDP:
  - *S* – set of states
  - *A* – set of actions
  - transition model: *P(s'|s,a)*
  - rewards: *R(s,a)*
- Goal:
  - **compute** a policy *π*
  - that optimizes value V(π)

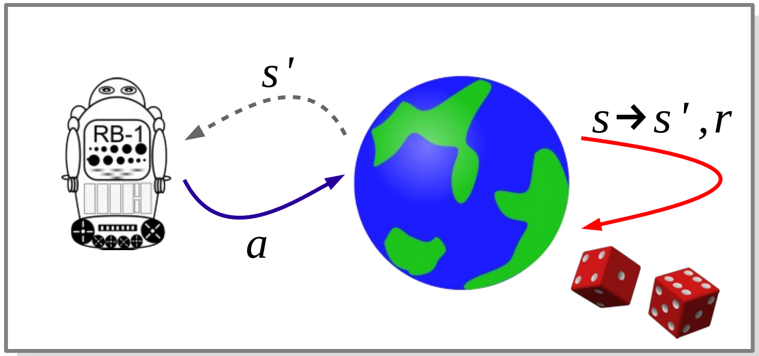
# ~~MDP Planning~~
# Reinforcement Learning

- Given ~~an MDP~~:
  - *S* – set of states
  - *A* – set of actions
  - ~~transition model: *p(s'|s,a)*~~
  - ~~rewards: *R(s,a)*~~
- Goal:
  - ~~**compute**~~ **learn** a policy *π*
  - that optimizes value V(π)



# ~~MDP Planning~~
# Reinforcement Learning

- Given ~~an MDP~~:
  - *S* – set of states
  - *A* – set of actions
  - ~~transition model: *p(s'|s,a)*~~
  - ~~rewards: *R(s,a)*~~
- Goal:
  - ~~**compute**~~ **learn** a policy $\pi$
  - that optimizes value V(π)

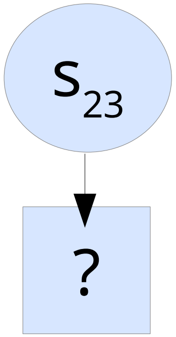



Reinforcement learning is a **problem**

(not a particular technique)

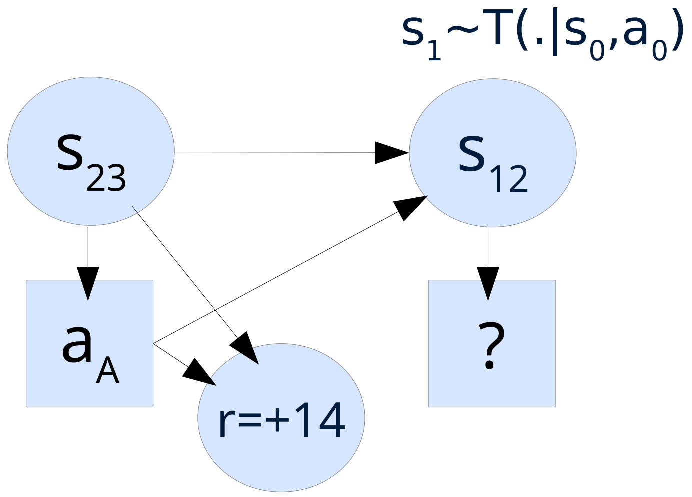
# Example...

- You are in state 23
  - what do you want to do? (A or B)



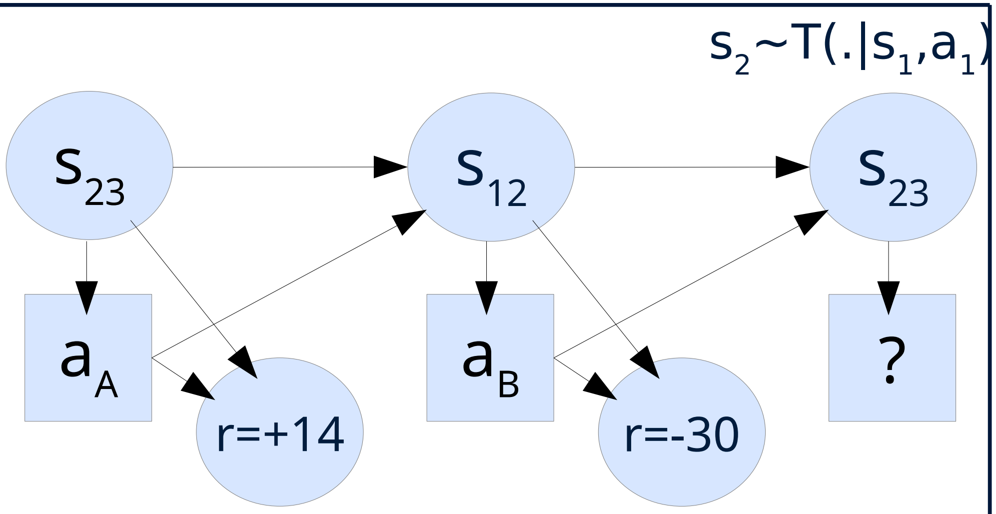
# Example...

- You are in state 23
  - what do you want to do? (A or B)
  - +14
- You are now in state 12
  - what do you want to do? (A or B)



# Example...

- You are in state 23
  - what do you want to do? (A or B)
  - +14
- You are now in state 12
  - what do you want to do? (A or B)
  - -30
- You are in state 23 again
  - what do you want to do? (A or B)



# Foundations:
# Q-Learning

# Q-learning

- Takes Bellman equations
  → turns into an update equation that learns from **sampled experience**

- After a transition (s,a,r,s') we update
  - $Q(s,a) := (1-\alpha)\, Q(s,a) + \alpha\, [\, r + \gamma \max_{a'} Q(s',a')\, ]$

update target

- Need to sufficiently **explore** the environment
  - But then will converge to Q*

# RL Nomenclature

Terminology in RL sometimes confusing…

- model available → ‘planning’
  - small problems: exact planning (DP, VI, PI, etc.)
  - large problems: simulation-based planning
    (aka approximate DP, neurodynamic programming, ... etc.)

- model not available → ‘reinforcement learning’
  - model-based RL: learns a model
  - model-free RL: does not learn a model
    - value-based: directly learn value function
    - policy search: directly learn policy

# RL Nomenclature

Terminology in RL sometimes confusing…

- model available → ‘planning’
  - small problems: exact planning (DP, VI, PI, etc.)
  - large problems: simulation-based planning

    (aka approximate DP, neurodynami

**Common confusion #1: mixing up these**

- Of course: MBRL typically **uses** planning

- model not available → ‘reinforcement learning’
  - model-based RL: learns a model
  - model-free RL: does not learn a model
    - value-based: directly learn value function
    - policy search: directly learn policy

# RL Nomenclature

Terminology in RL sometim

Common confusion #2: mixing up these
- indeed, can use Q-learning for both!
- but:
  - in former we care about **computational cost**
  - in latter we learn **online**: care about the rewards during learning ('regret')

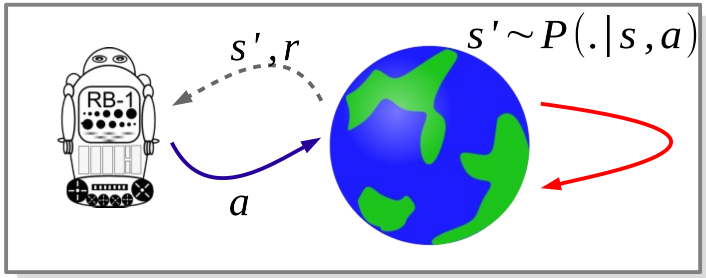
- model available →
  - small problems: exact planning (DP, VI, PI, etc.)
  - large problems: simulation-based planning (aka approximate DP, neurodynamic programming, ... etc.)

- model not available → 'reinforcement learning'
  - model-based RL: learns a model
  - model-free RL: does not learn a model
    - value-based: directly learn value function
    - policy search: directly learn policy

# Foundations:
# Monte-Carlo Tree Search

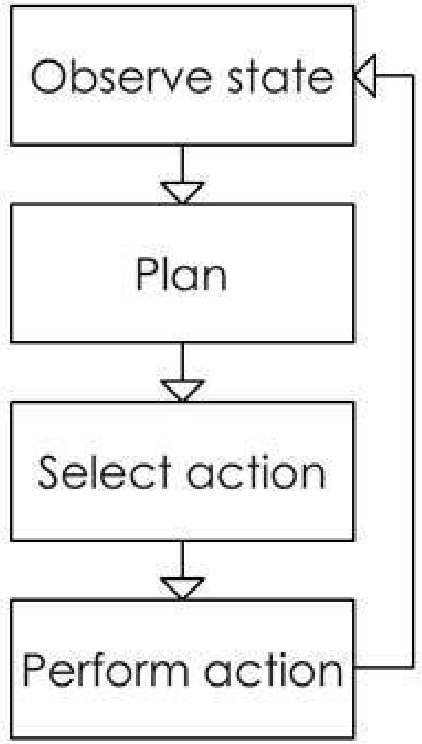
# Online Planning

- Pre-planning for all states could be infeasible...



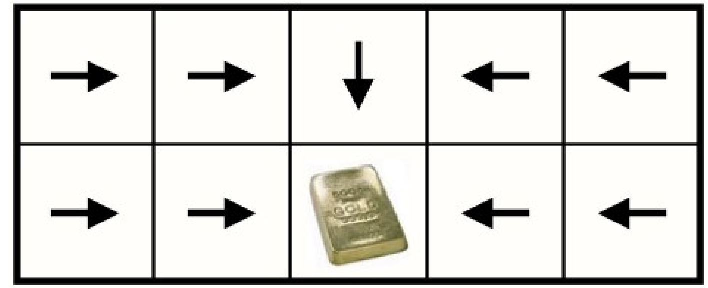

- 1 possible solution:
  Interleave planning and execution

→ focuses computational effort
on states reachable in near-future

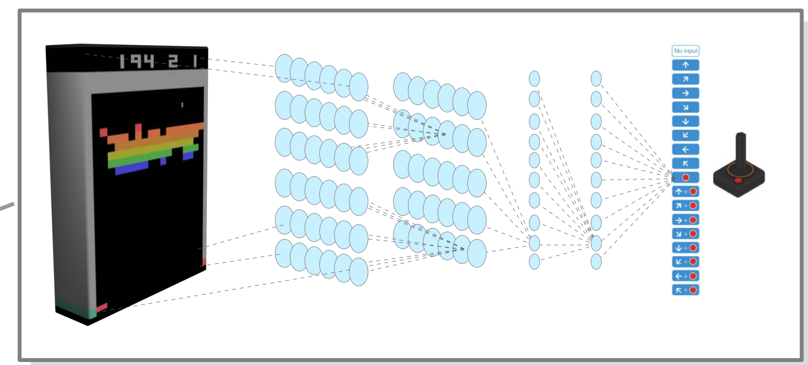


# <Intermezzo: Policy Representations>

- A policy $\pi$, in an MDP: states to actions $\pi{:}S \rightarrow A$



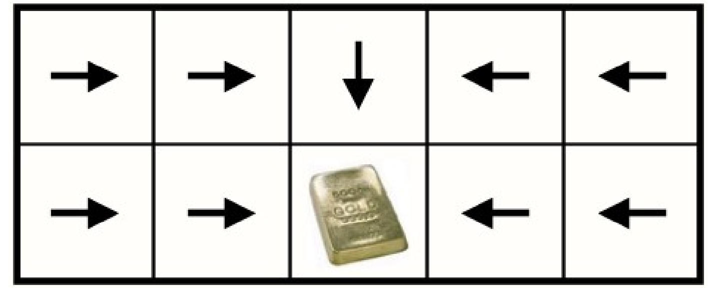
- How **represented**?
  - Lookup table
  - (More) computation
    - e.g., neural network
    - entire planning algorithm



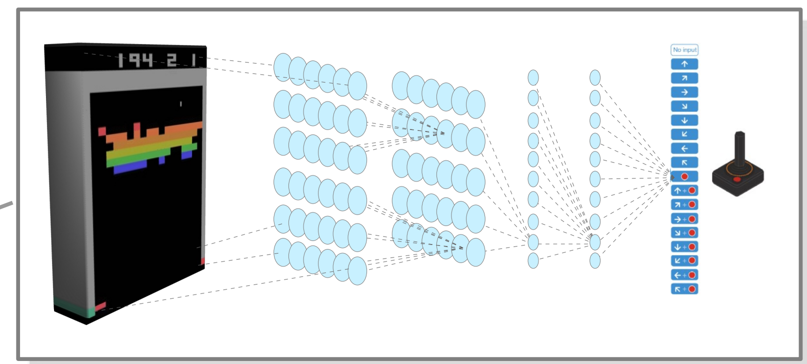
```
function MonteCarloPlanning(state)
repeat
    search(state, 0)
until Timeout
return bestAction(state,0)

function search(state, depth)
if Terminal(state) then return 0
if Leaf(state, d) then return Evaluate(state)
action := selectAction(state, depth)
(nextstate, reward) := simulateAction(state, action)
q := reward + γ search(nextstate, depth + 1)
UpdateValue(state, action, q, depth)
return q
```

# <Intermezzo: Policy Representations>

- A policy $\pi$, in an MDP: states to actions $\pi{:}S \rightarrow A$



- How **represented**?
  - Lookup table
  - (More) computation
    - e.g., neural network
    - entire planning algorithm



```
function MonteCarloPlanning(state)
repeat
    search(state, 0)
til Timeout
urn bestAction(state,0)

nction search(state, depth)
Terminal(state) then return 0
Leaf(state, d) then return Evaluate(state)
ion := selectAction(state, depth)
extstate, reward) := simulateAction(state, action)
= reward + γ search(nextstate, depth + 1)
odateValue(state, action, q, depth)
turn q
```

one perspective:

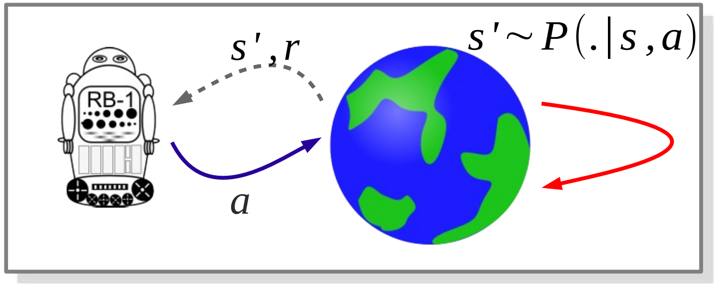
▶ online planning is a (pre-specified) policy

Implication:

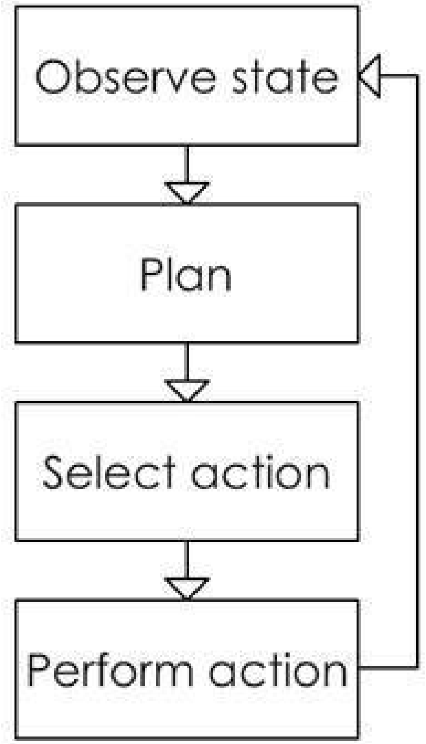
▶ learning to plan not fundamentally different than learning a policy with a different parametrization?

# Online Planning

- Pre-planning for all states could be infeasible…

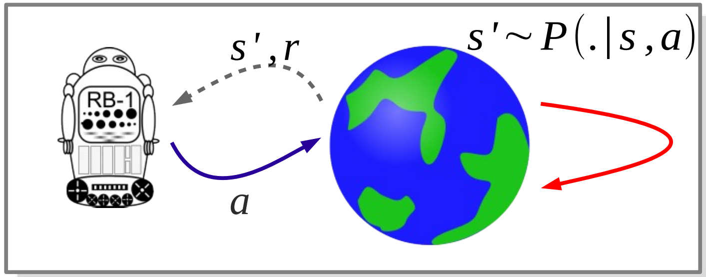



- 1 possible solution:
  Interleave planning and execution




→ focuses computational effort
on states reachable in near-future

# Online Planning



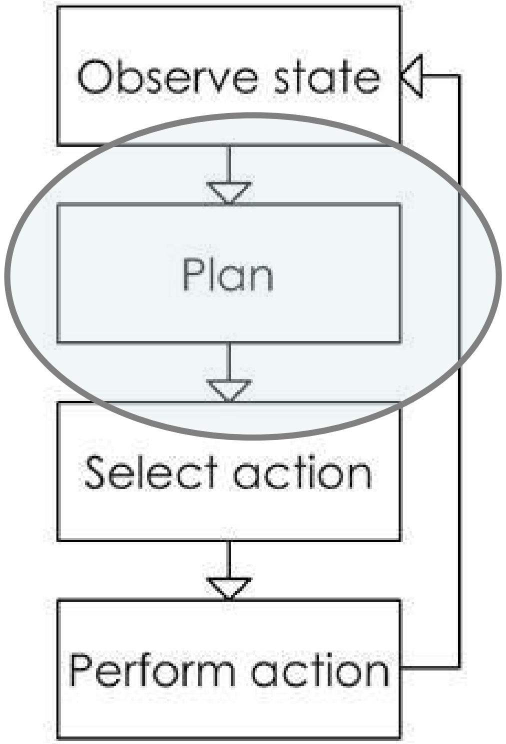

- Pre-planning for all states could be infeasible...

- 1 possible solution:
  Interleave planning and execution

**What planning methods?**

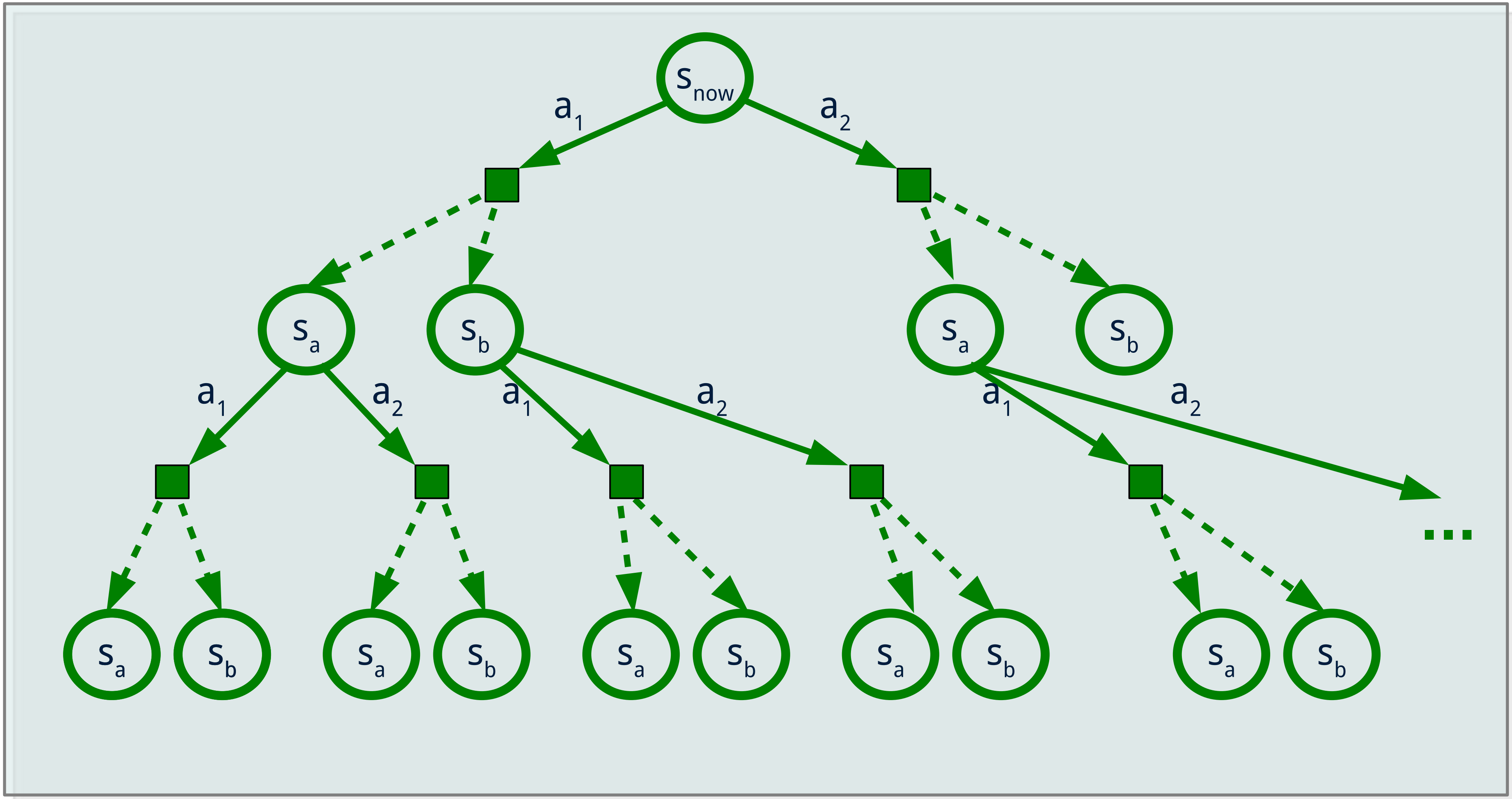
Build a lookahead tree (“search”)

- ►over which we can do dynamic programming
- ►E.g., chess



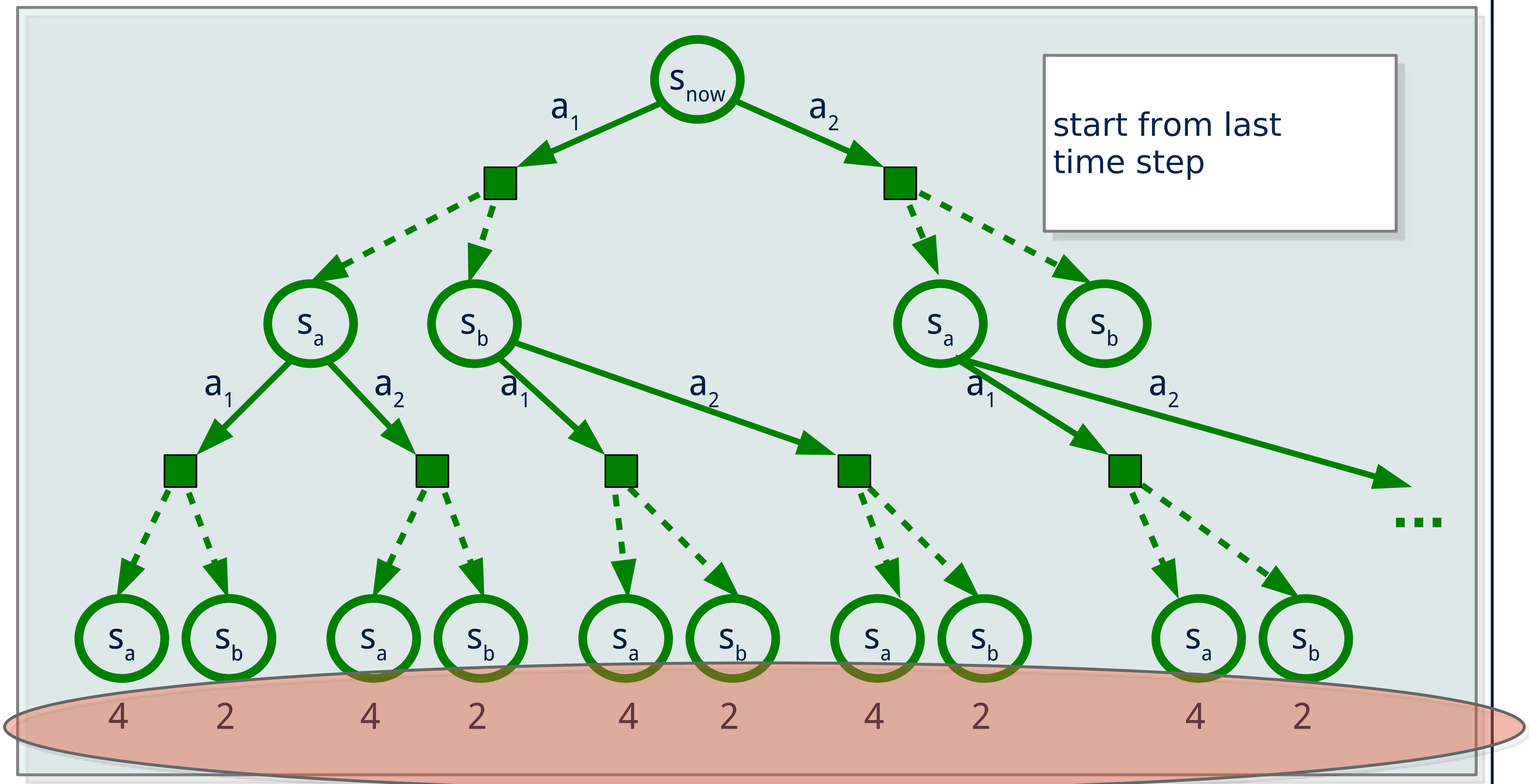
# Dynamic Programming

- Construct a plan for *T* time steps into the future



# Dynamic Programming

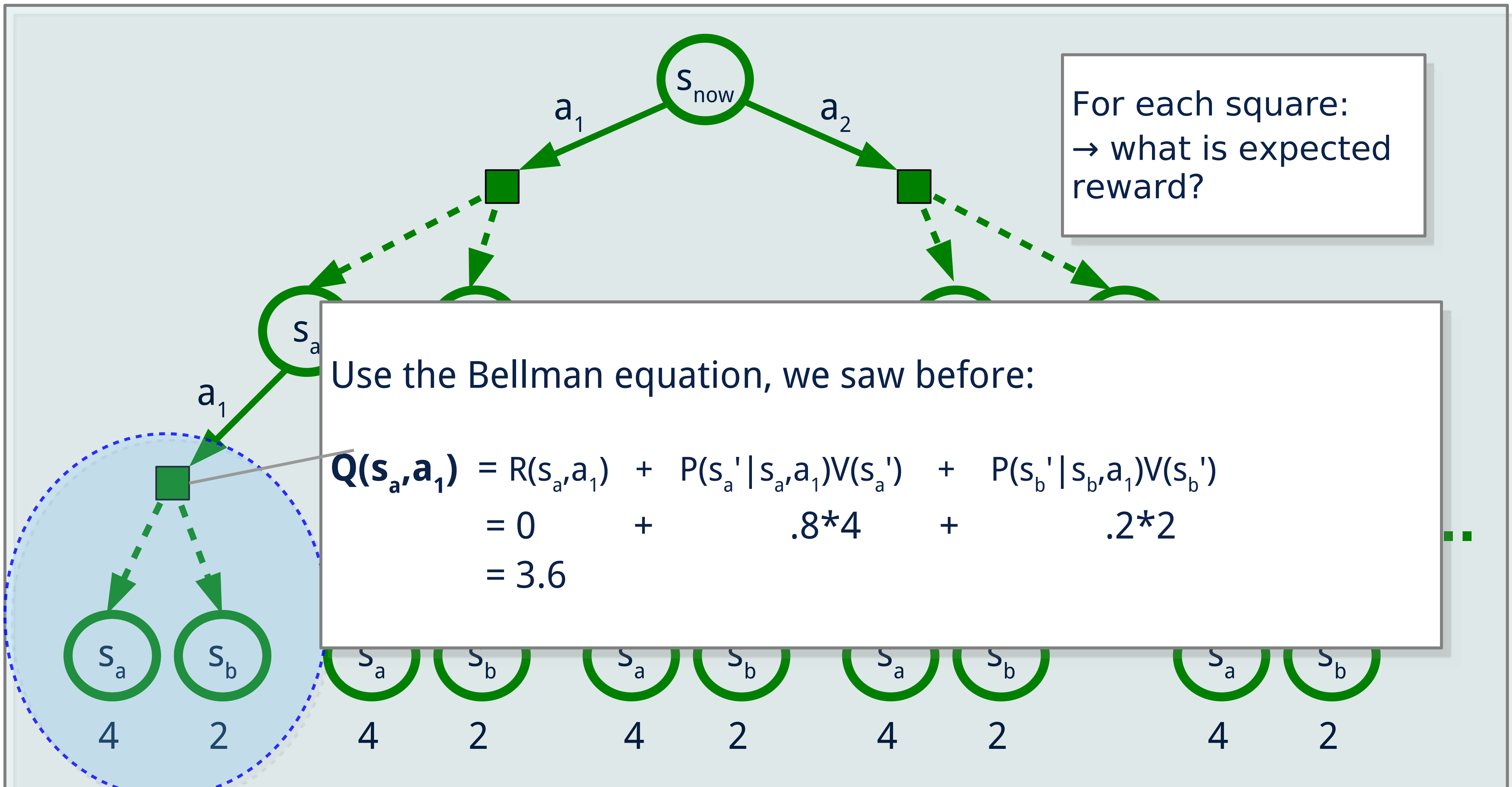
- Construct a plan for *T* time steps into the future




**use heuristic to provide values, V(s), for the leaves**

# Dynamic Programming

- Construct a plan for *T* time steps into the future



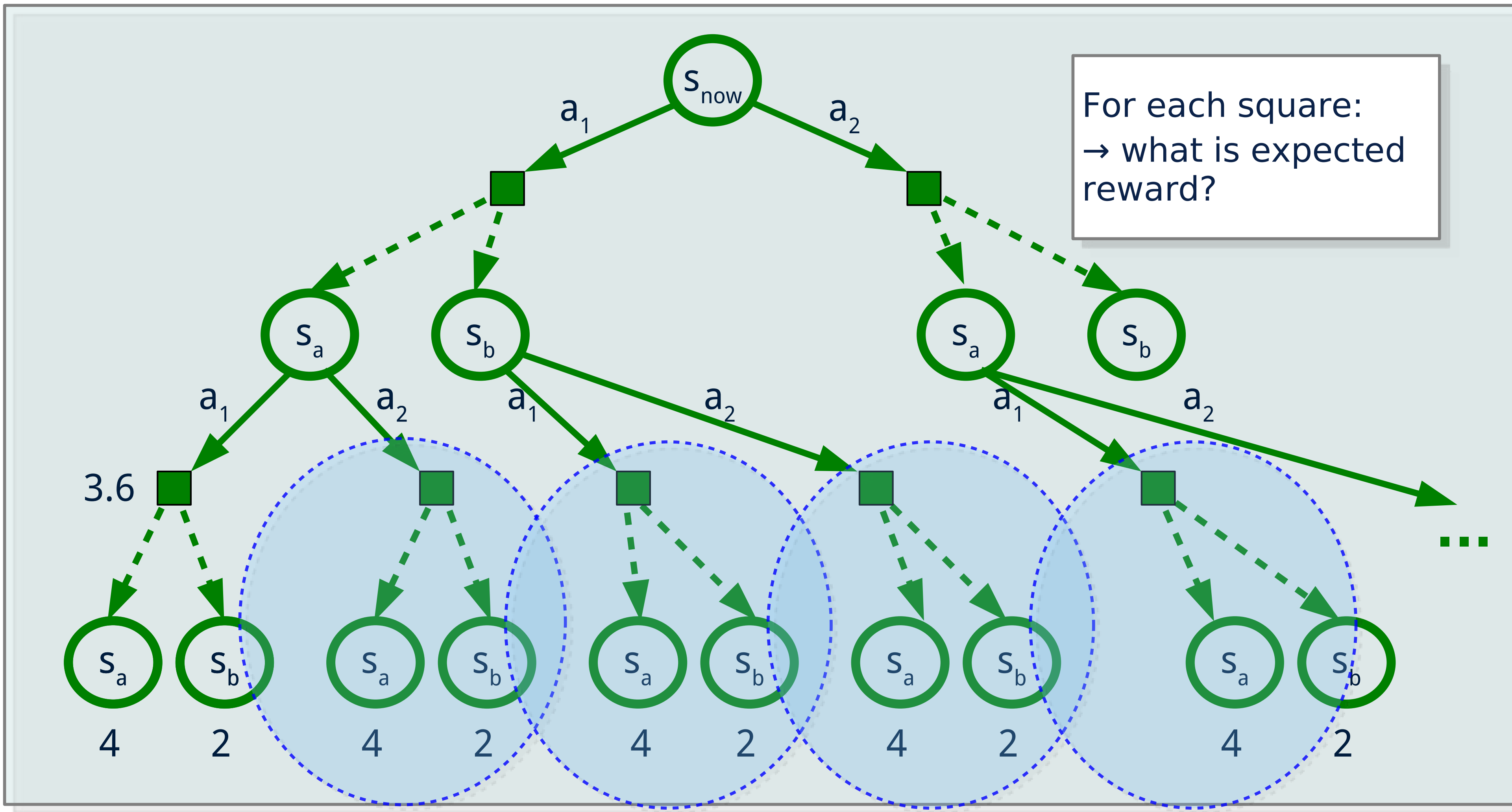

For each square:
→ what is expected reward?

Use the Bellman equation, we saw before:

$$\mathbf{Q(s_a,a_1)} = R(s_a,a_1) + P(s_a'|s_a,a_1)V(s_a') + P(s_b'|s_b,a_1)V(s_b')$$
$$= 0 + .8*4 + .2*2$$
$$= 3.6$$

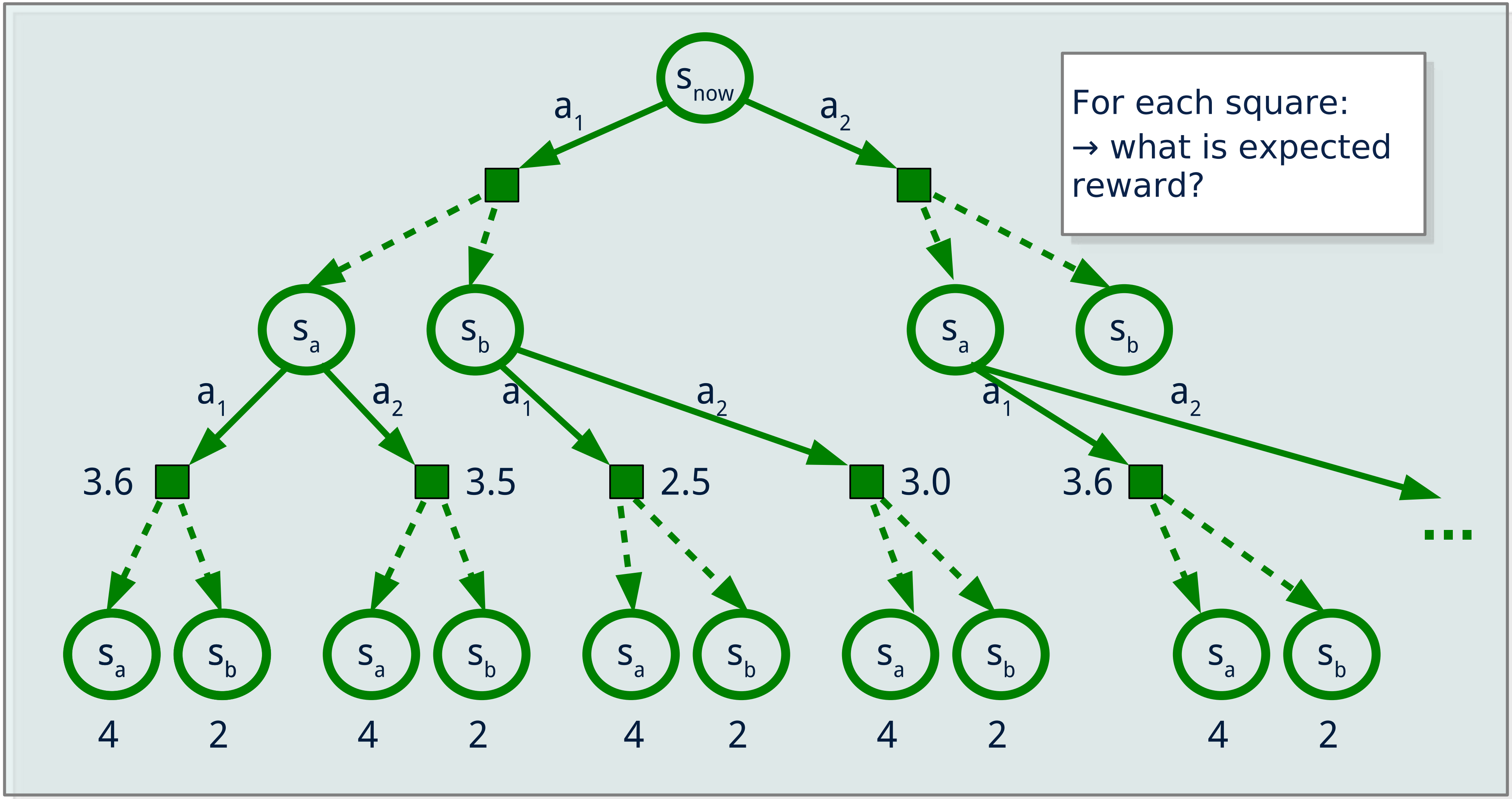
# Dynamic Programming

- Construct a plan for $T$ time steps into the future



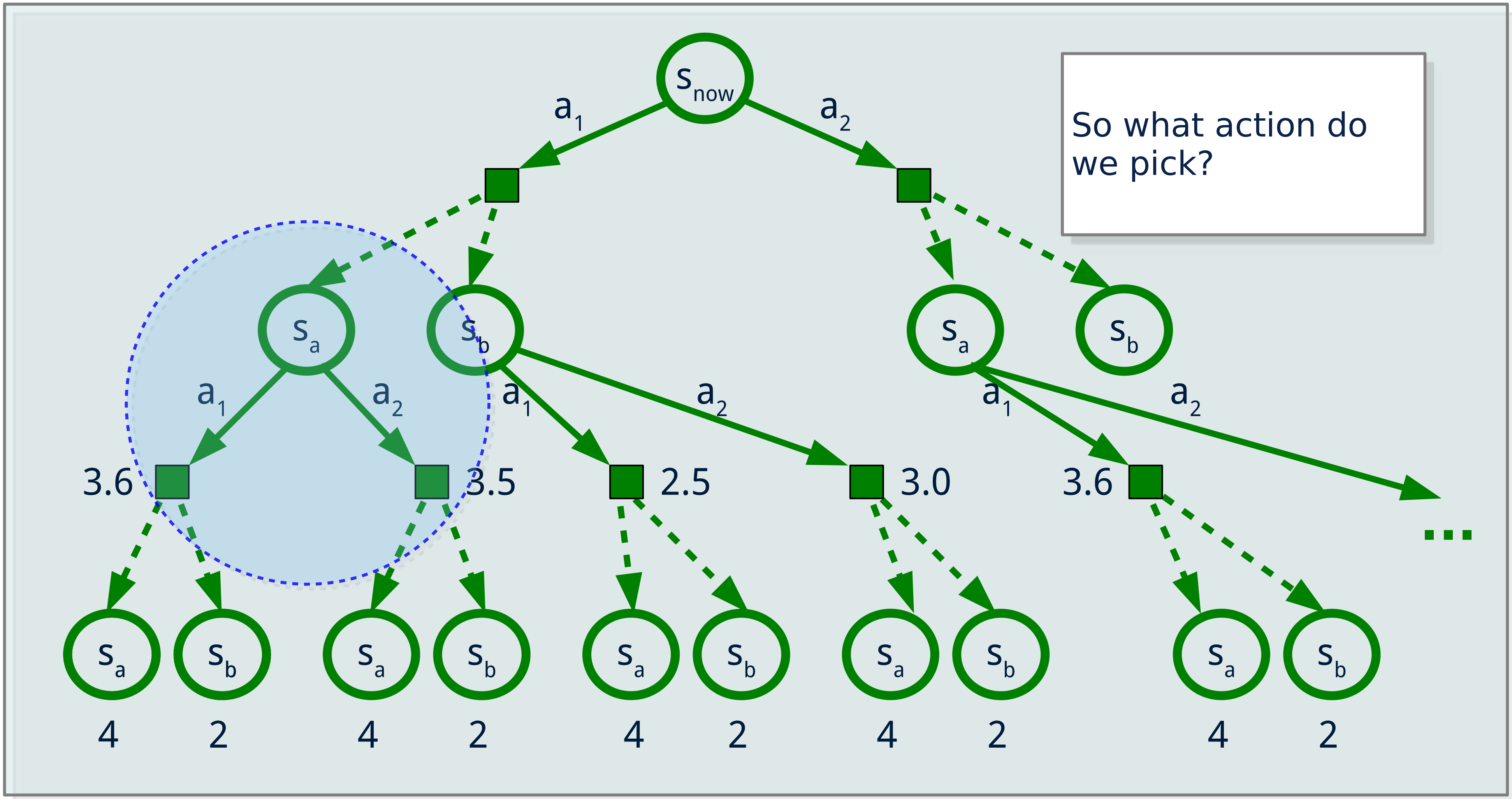
# Dynamic Programming

- Construct a plan for *T* time steps into the future



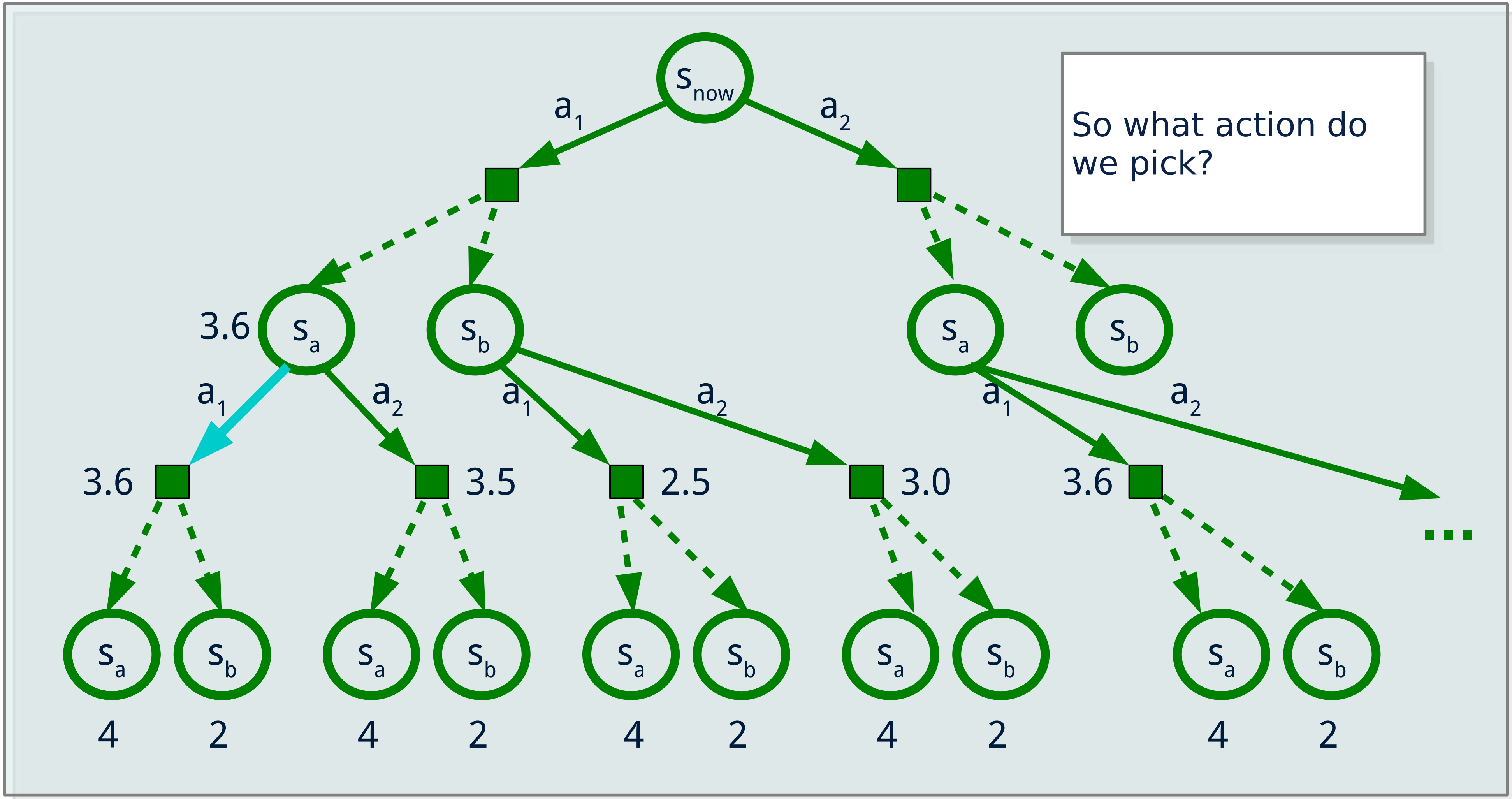
# Dynamic Programming

- Construct a plan for *T* time steps into the future



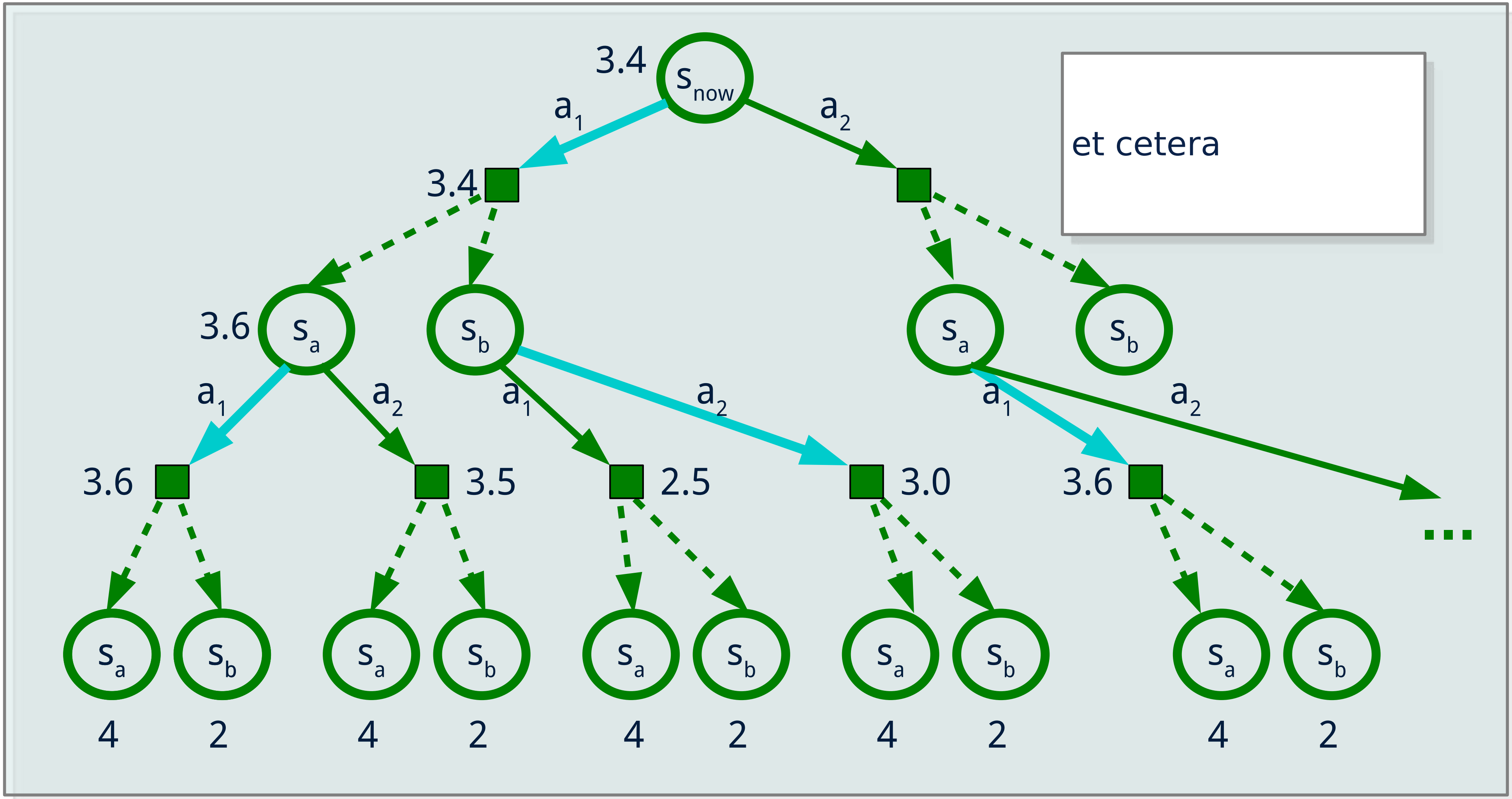
# Dynamic Programming

- Construct a plan for $T$ time steps into the future



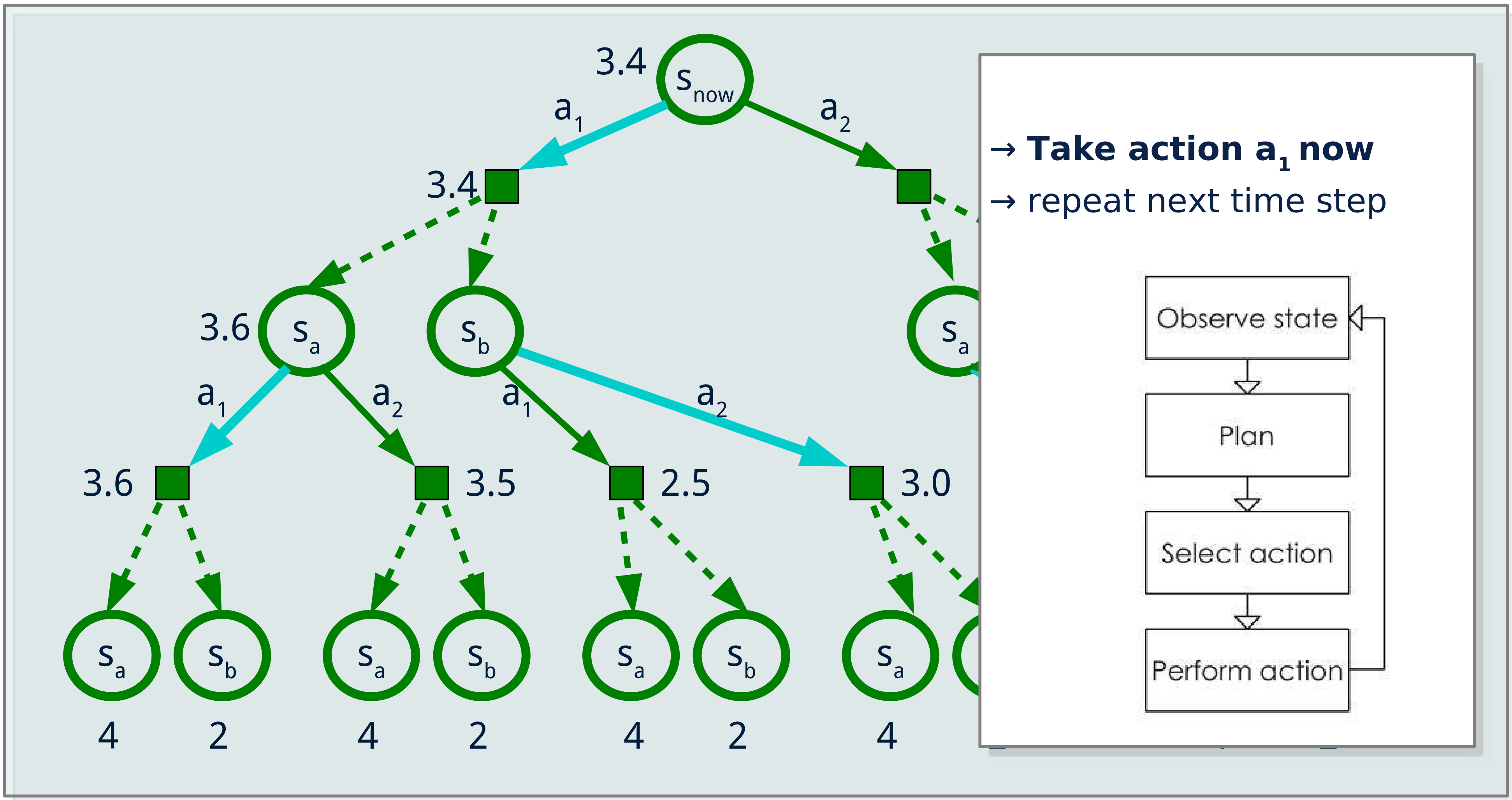
# Dynamic Programming

- Construct a plan for *T* time steps into the future



# Dynamic Programming

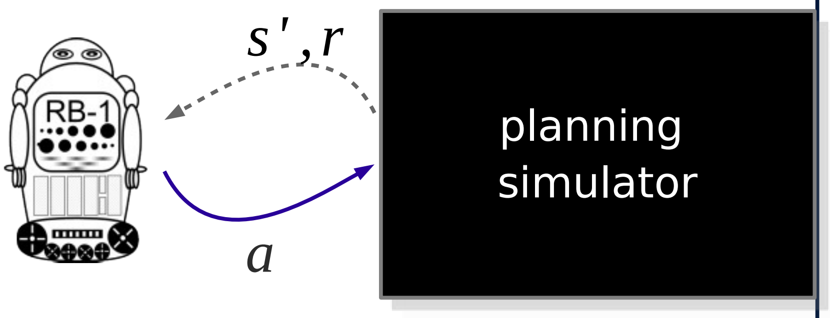
- Construct a plan for *T* time steps into the future



# Monte Carlo Tree Search (MCTS)

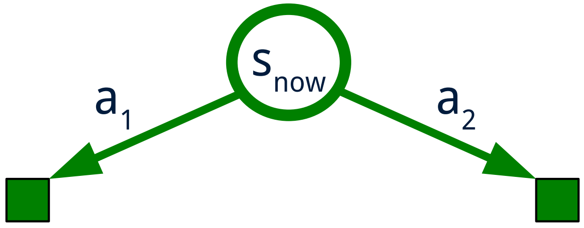
- Problem: trees get huge...!

- MCTS provides leverage by:
  - **incrementally** constructing a **sampled version** of the tree
  - focusing on **promising regions**



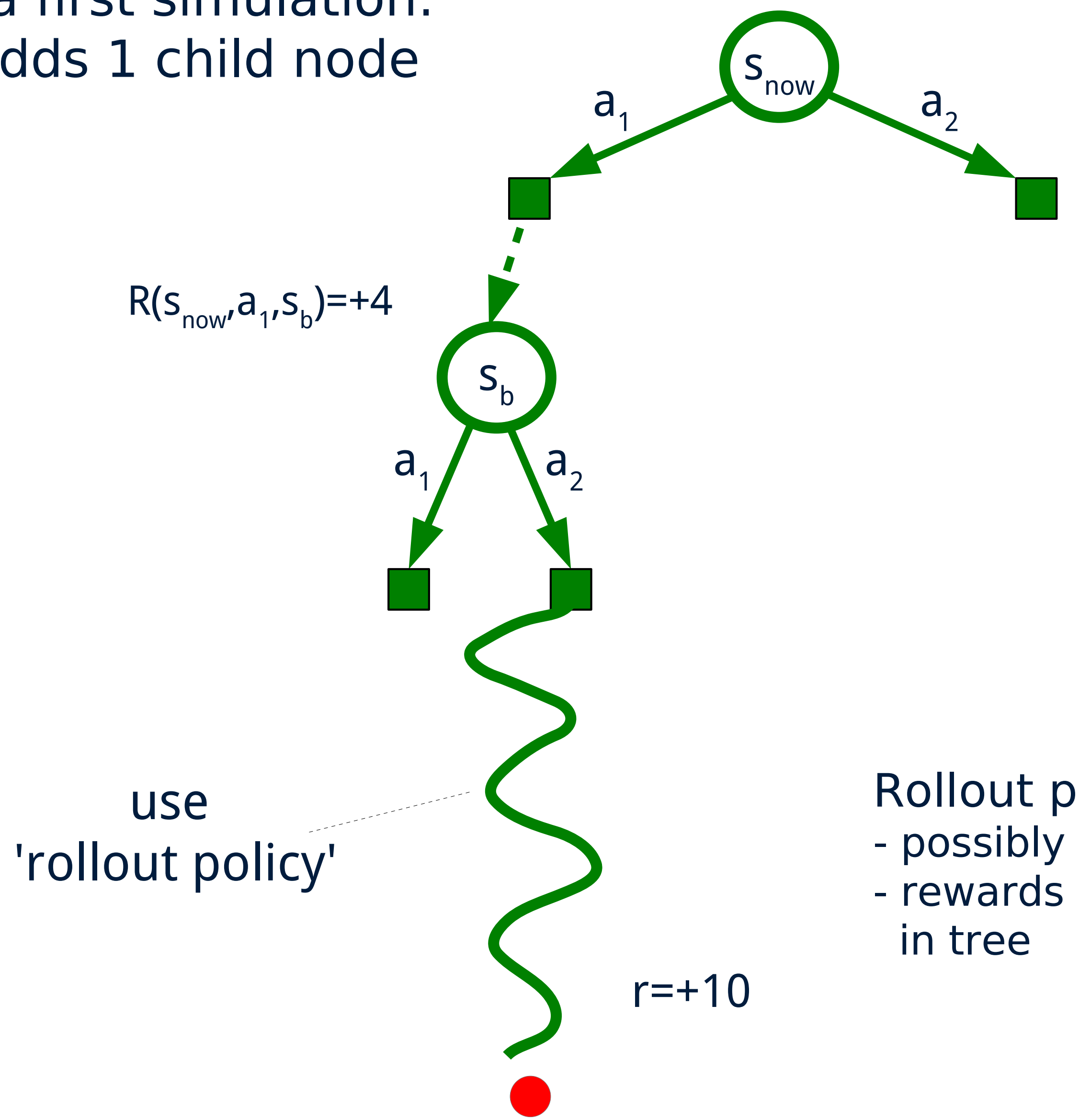
# MCTS – Example

starting with only a root node



# MCTS – Example

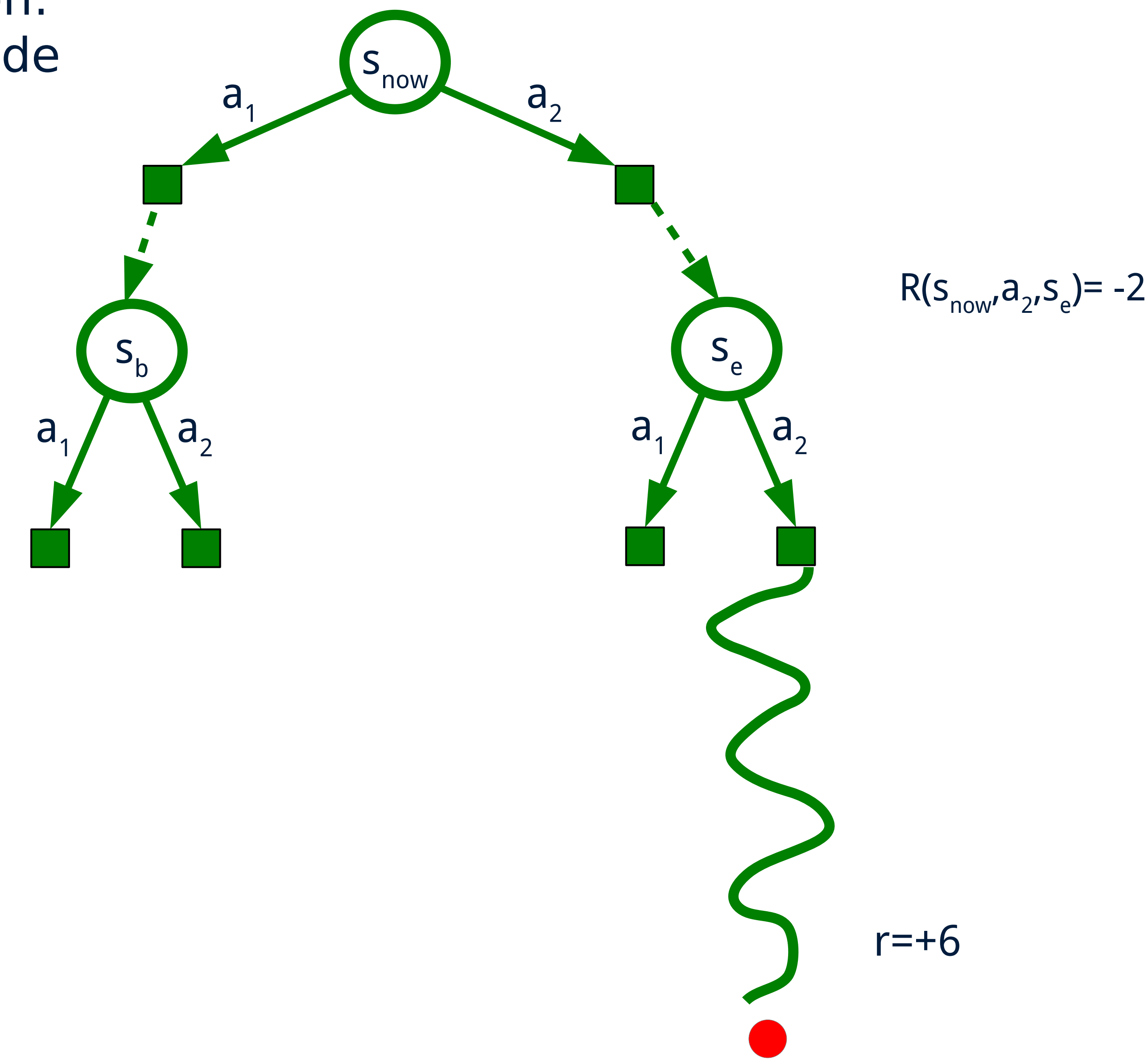
do a first simulation:
→ adds 1 child node



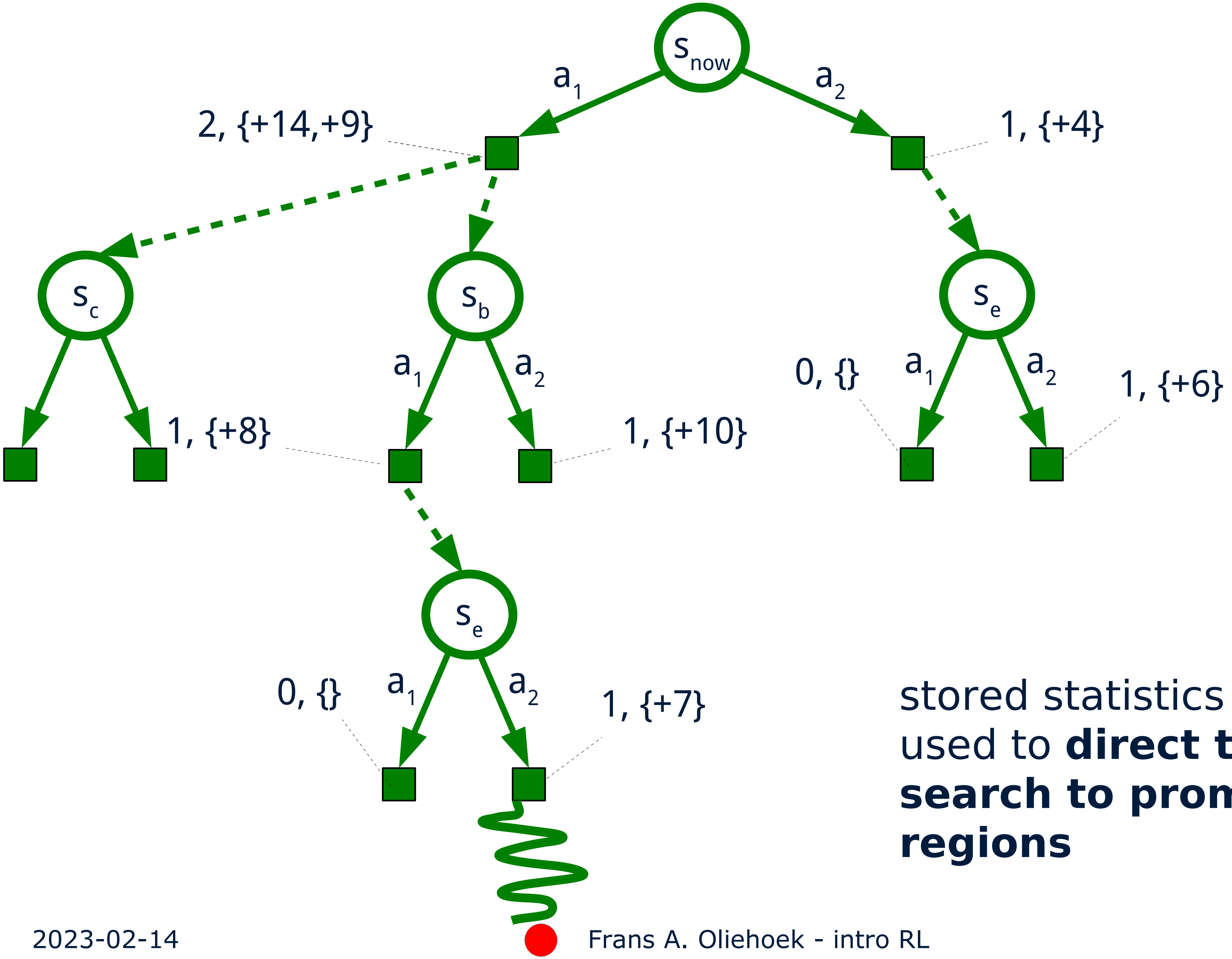

Rollout policy:
- possibly till end of problem
- rewards used to update statistics
  in tree

# MCTS – Example

another simulation:
→ adds 1 child node



# MCTS – Example




stored statistics are used to **direct the search to promising regions**

# Rollout Policies

- Another important component: which rollout policy?
  - In theory: as long as it gives positive probability to any action
  - In practice: huge effect → use domain knowledge.

- Perspective: MCTS as a **policy improvement operator**
  - given a policy, MCTS improves it
    by applying additional search
  - How AlphaGo improves its policies...

# MCTS Pros/Cons

- Pros:
  - rapidly zooms in on promising regions
  - can be used to improve policies
  - basis of many successful application

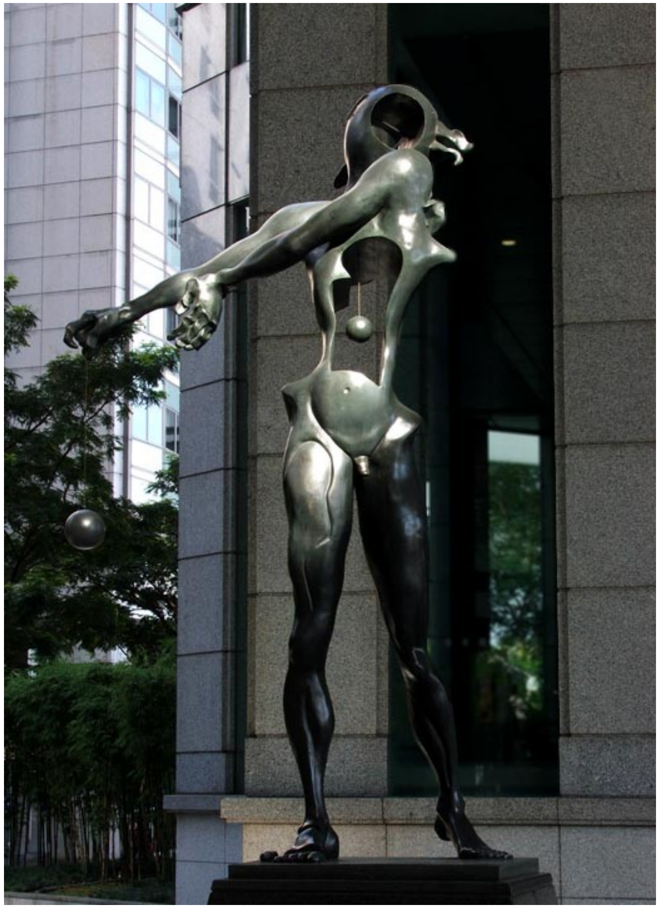

- Limitations:
  - needle in the hay-stack problems
  - problems with high branching factor

# Foundations: Summary

- MDPs formalize decision making in stochastic environment
  - Model available → planning
  - Model not available → reinforcement learning (RL)
- RL is a **problem**
  - Q-learning is one of the most popular **techniques**
- For complex problems:
  - representing a policy as a table not feasible
  - online planning can help
- Given infinite computation:
  - dynamic programming on tree of trajectories
- Monte Carlo tree search (MCTS):
  - avoid creating the entire tree; focus on promising parts
  - by selecting actions in a smart way
  - use domain knowledge via rollout policies

# Outline for Today

- ~~Teaser~~
- ~~Foundations of RL~~
- **Intuition behind state of the art**
  - DQN
  - AlphaGo
- Challenges



Homage to Newton by Salvador Dali

# Deep RL:
# Deep Q-Networks (DQN)

# Deep RL: Scaling up via deep learning

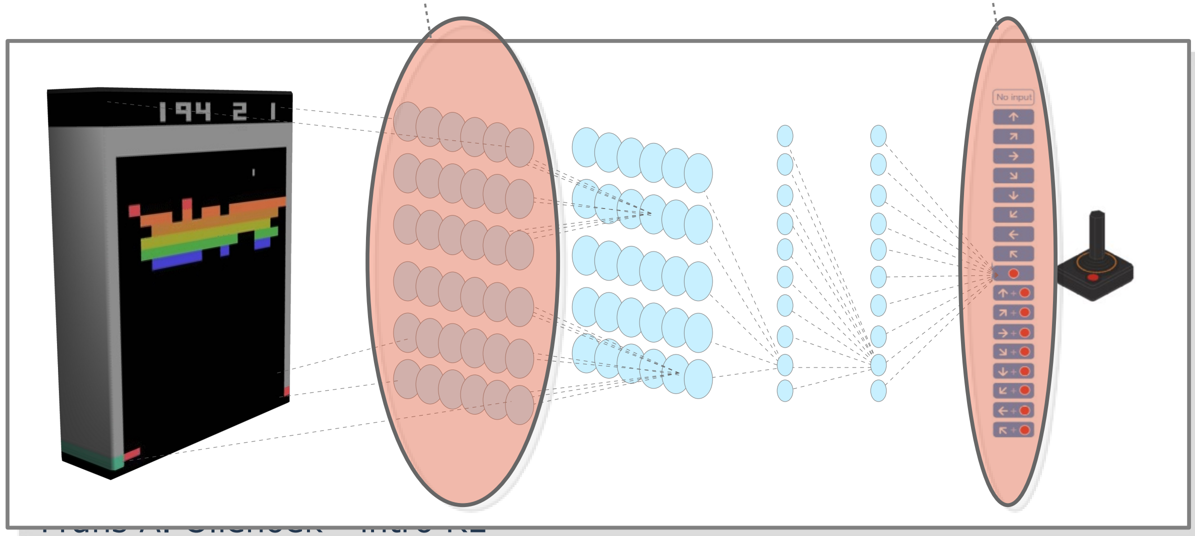
- Methods covered so far are **tabular**
  - Q(s,a) values for each (s,a) in a table

- But MDPs are huge...!
  (e.g., number of possible screens in Atari?)

- use function approximation
  to scale up!
  - e.g., represent Q(s,a) with
    a deep neural network

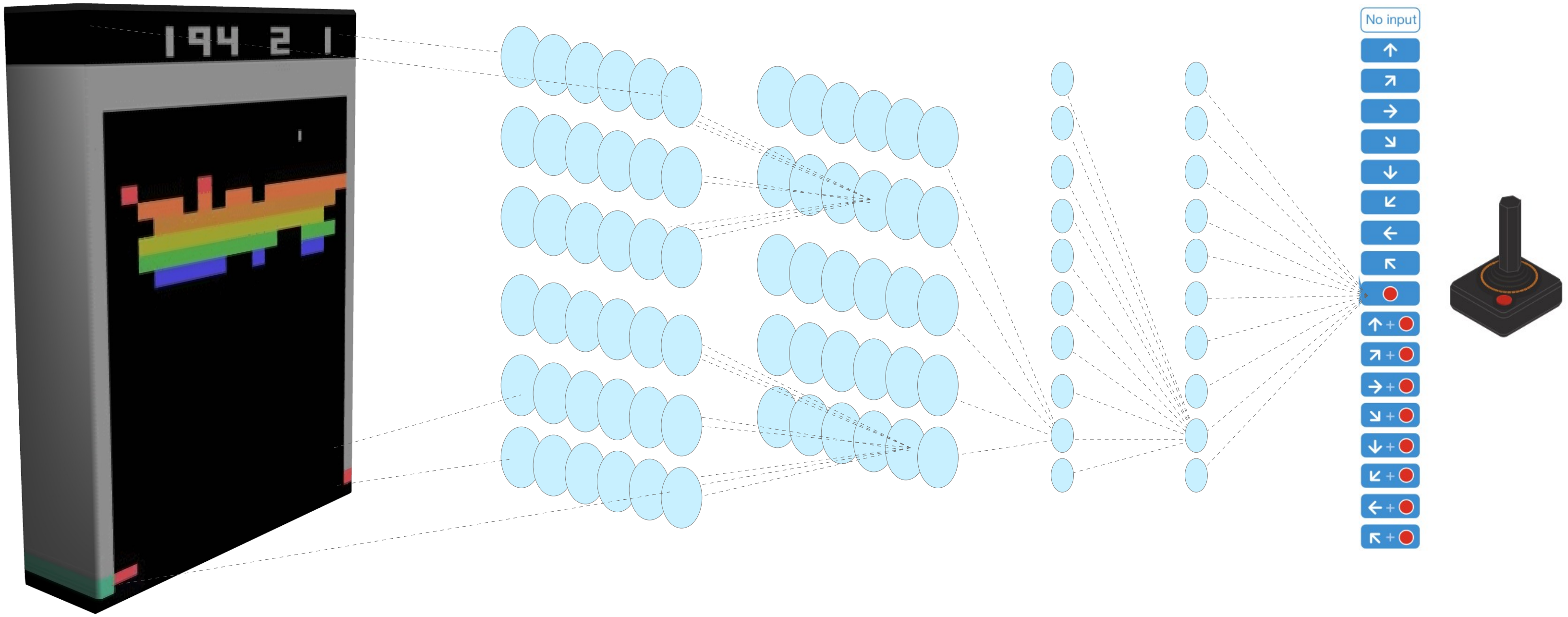
map from
'state
features'

to values for
each action



# Deep Q-Networks


- Prototypical example: DQN [Mnih et al. 2015]
- does precisely this:
  - Q-network: 84x84 image → 'action values'
  - Train with Q-learning




- only: Q-learning with neural networks might diverge...

Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529.

# Tricks for Convergence

- So need to stabilize…

- DQN uses a number of techniques:
  - experience replay
  - ‘target network’
  - gradient clipping

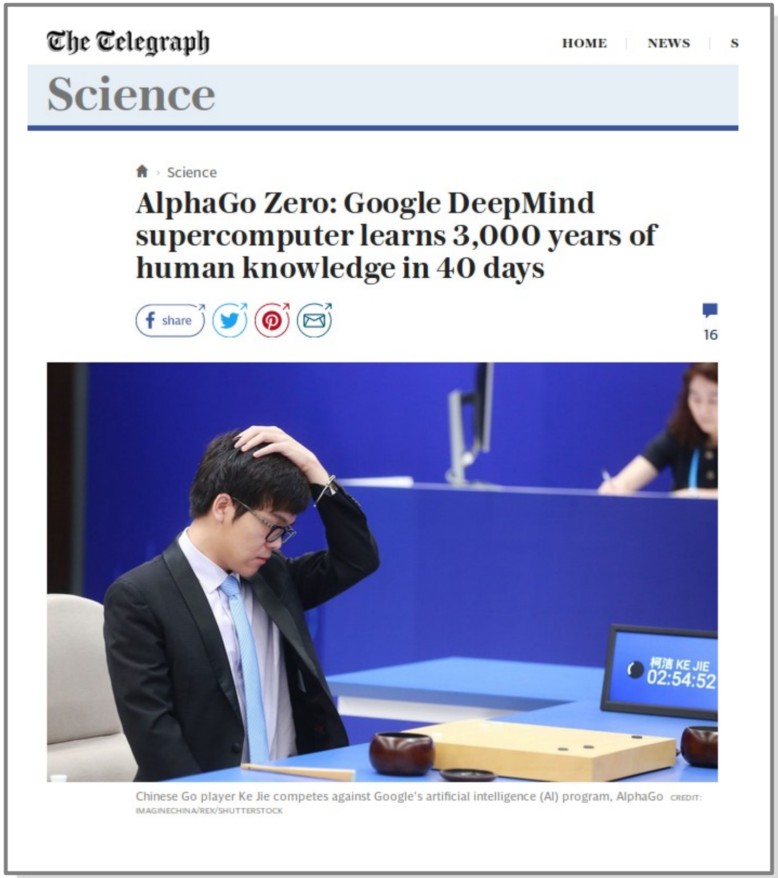
- together, they lead to sufficient stability


**Algorithm 1: deep Q-learning with experience replay.**
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
**For** episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  **For** $t = 1$,T **do**
    With probability $\varepsilon$ select a random action $a_t$
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters $\theta$
    Every $C$ steps reset $\hat{Q} = Q$
  **End For**
**End For**

# Deep RL:
# Alpha-Go

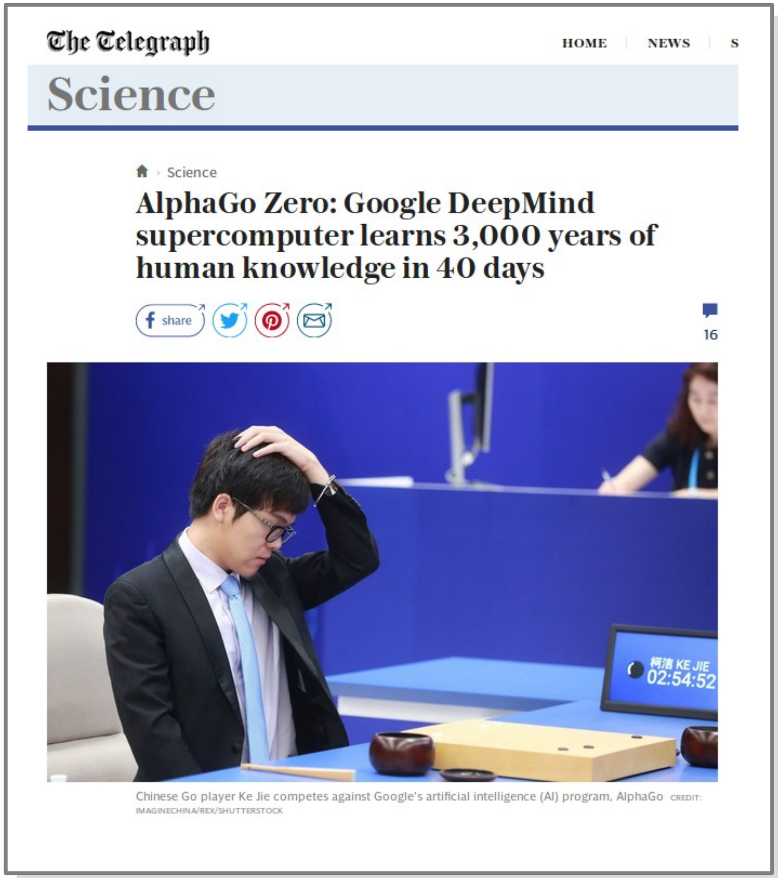
# AlphaGo

- Combines neural networks and MCTS

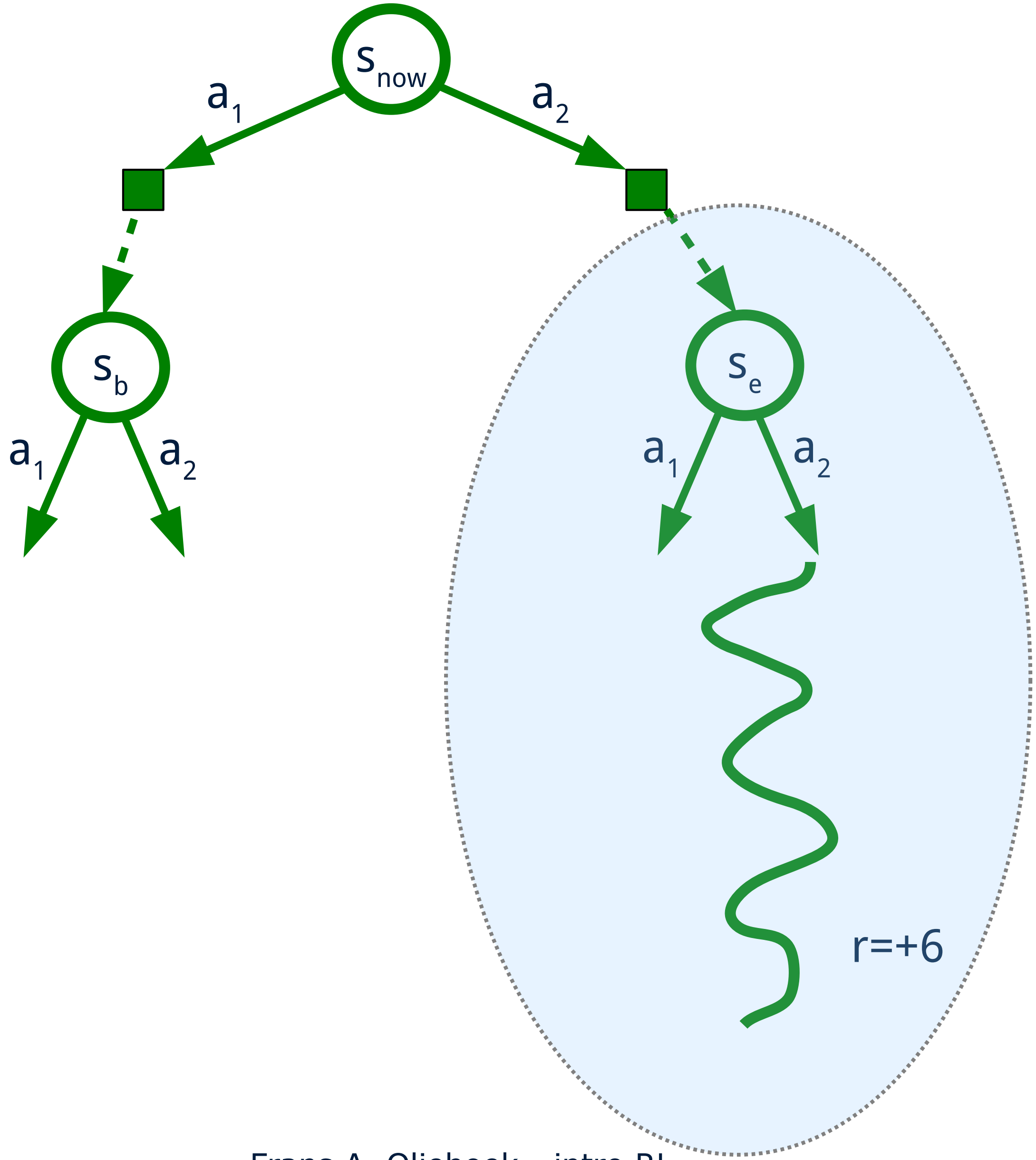


# AlphaGo

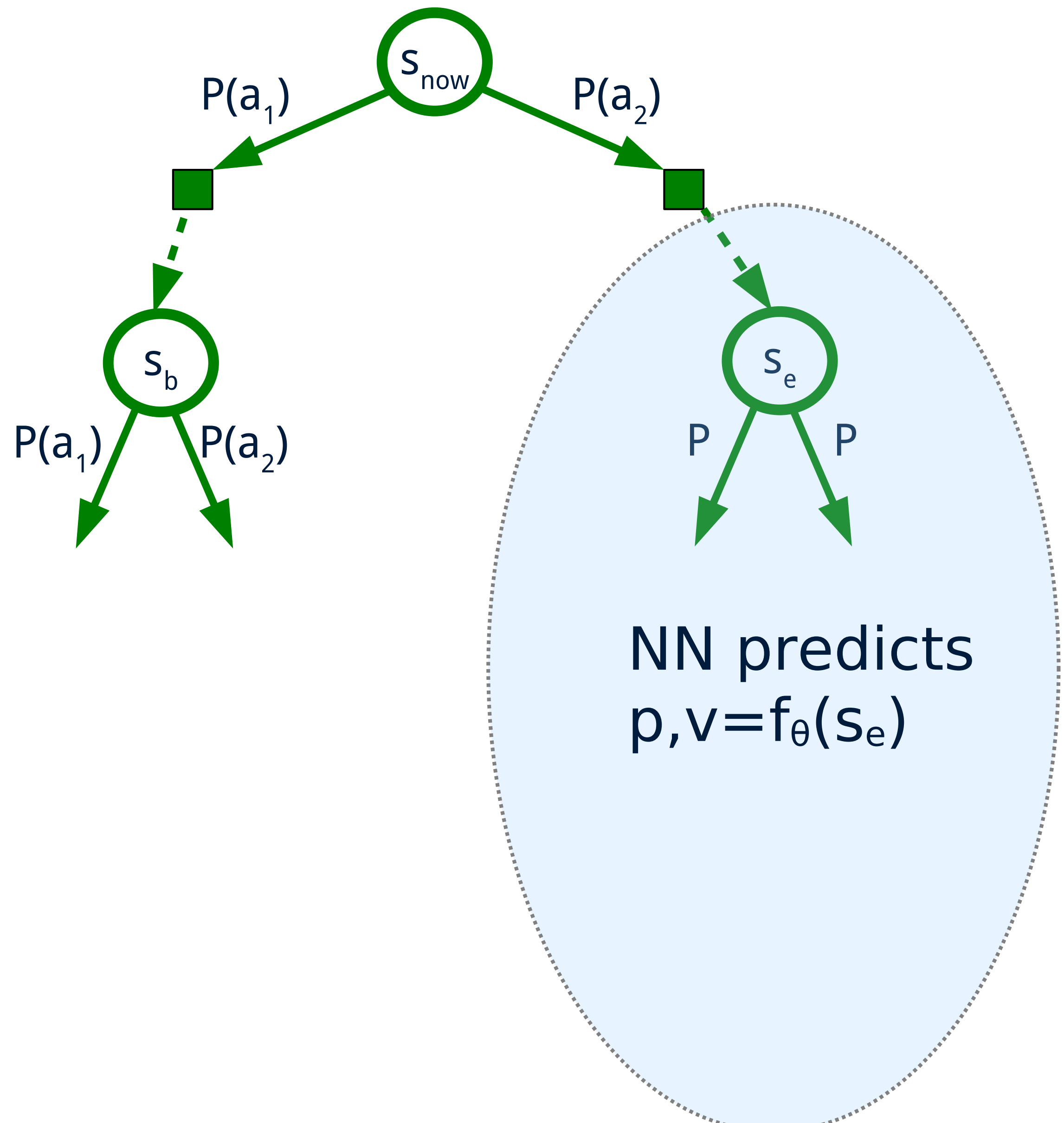
- Combines neural networks and MCTS
- Main challenges:
  - many actions
    → learn a policy network
  - deep trees, long rollouts
    → value network


The Telegraph
HOME | NEWS | S
Science
Science
**AlphaGo Zero: Google DeepMind supercomputer learns 3,000 years of human knowledge in 40 days**
share
16
Chinese Go player Ke Jie competes against Google's artificial intelligence (AI) program, AlphaGo CREDIT: IMAGINECHINA/REX/SHUTTERSTOCK

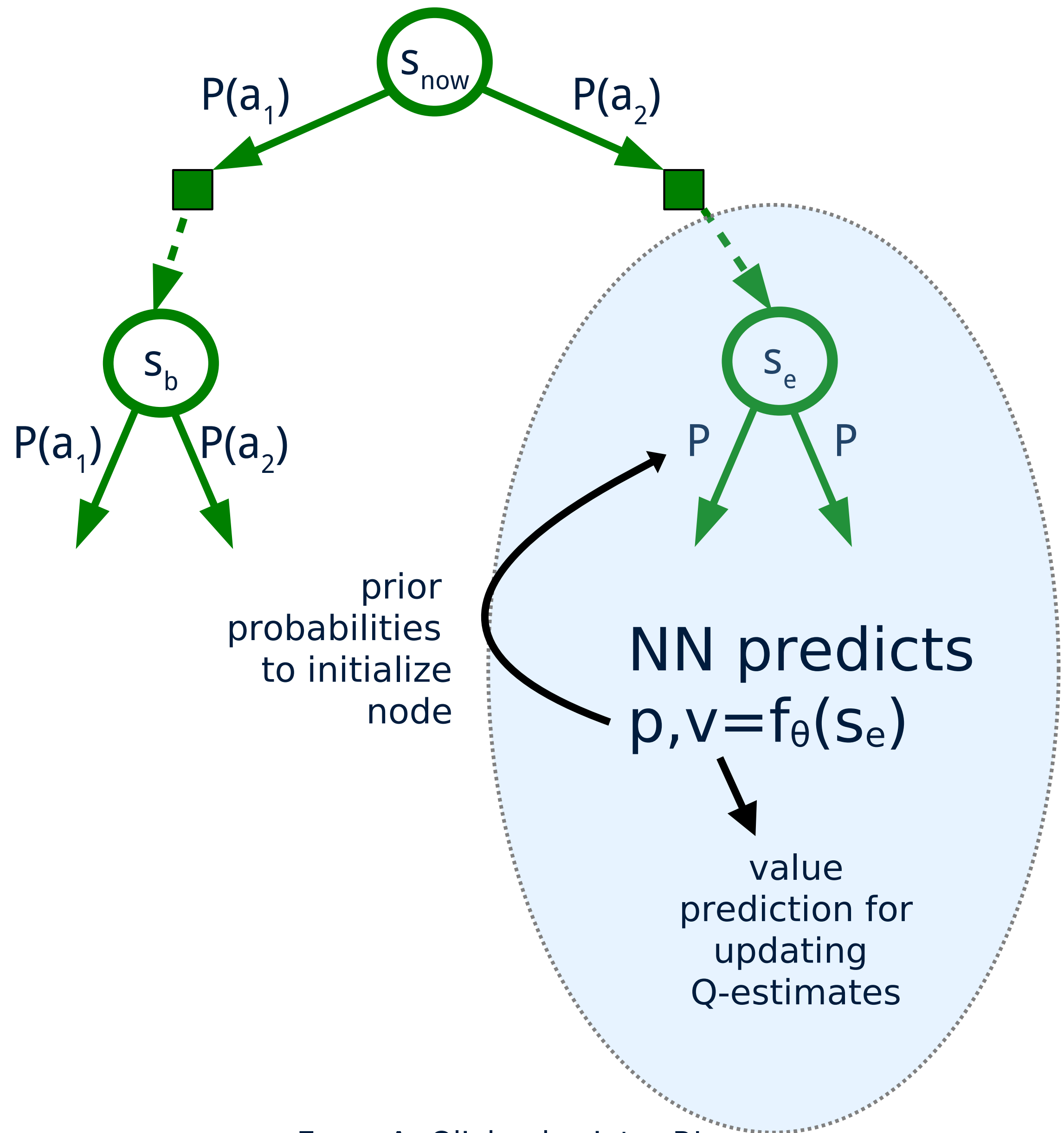
# AlphaGo



# AlphaGo



# AlphaGo

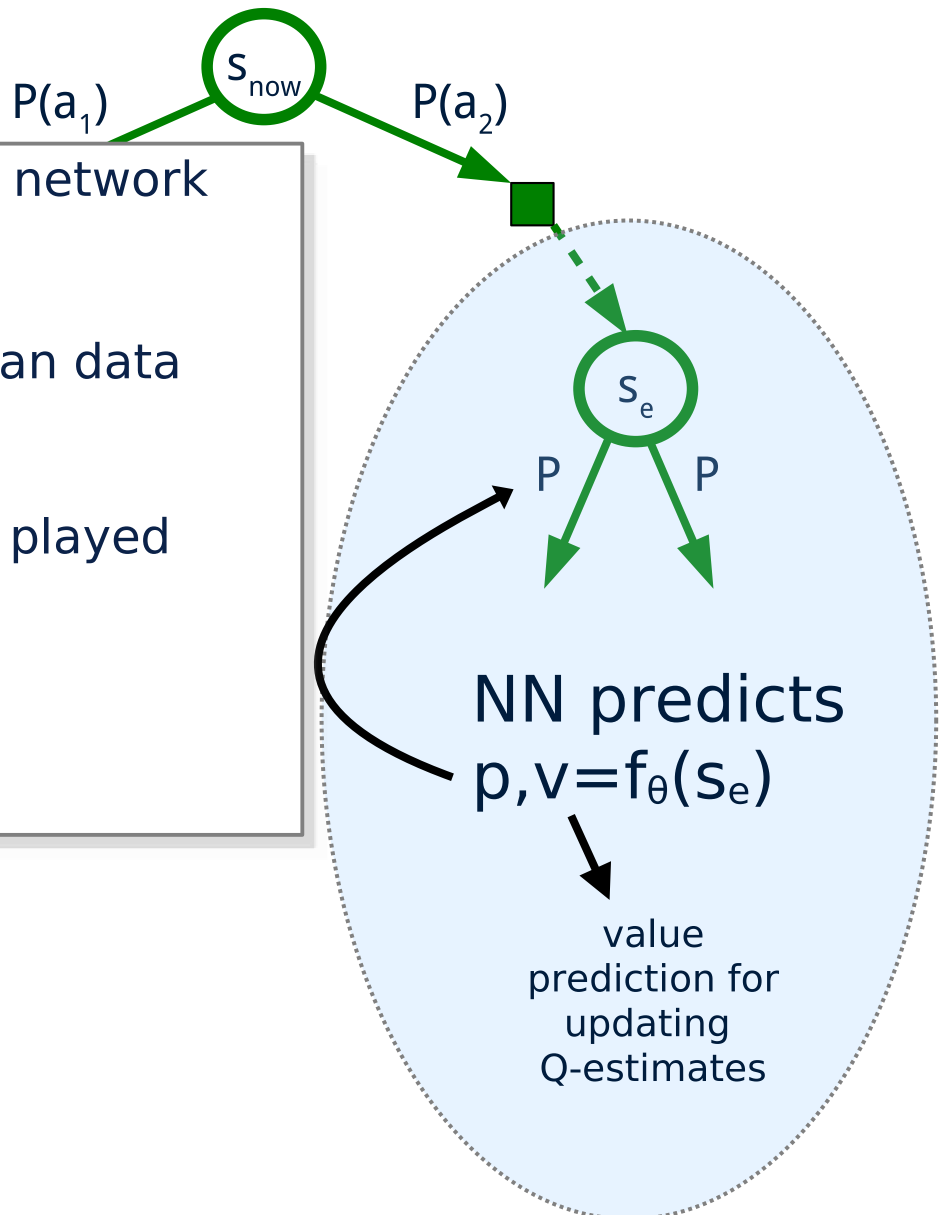


$s_{now}$
$P(a_1)$
$P(a_2)$
$s_b$
$P(a_1)$
$P(a_2)$
$s_e$
P
P
prior probabilities to initialize node
NN predicts $p,v=f_\theta(s_e)$
value prediction for updating Q-estimates

# AlphaGo

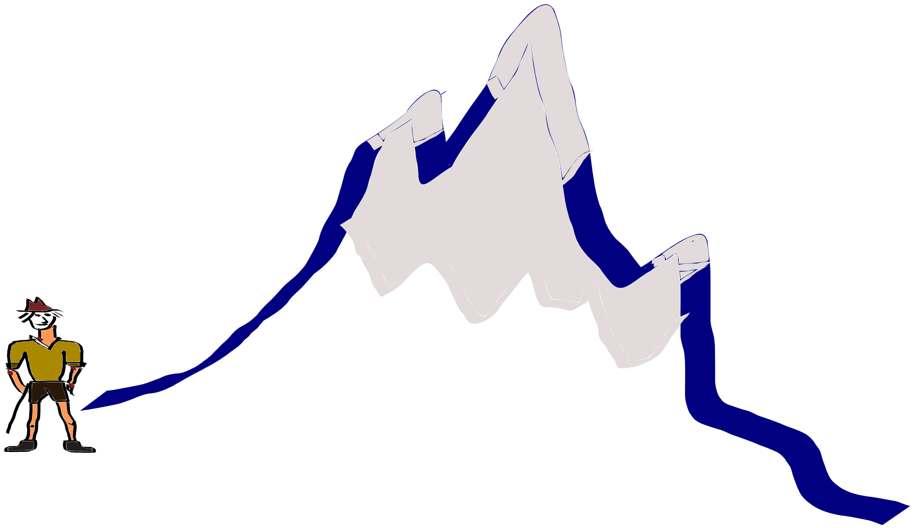
Where does the neural network come from?

- initialized from human data
- self-play:
  learned from games played against itself



# Outline for Today

- ~~Teaser~~
- ~~Foundations of RL~~
- ~~Intuition behind state of the art~~
- **Challenges**
  - sample complexity
  - learning models
  - partial observability
  - scaling & need for abstraction
  - multiagent systems
  - generalization



# Sample Complexity

# Deep RL methods are data hungry

- Atari: DQN was using
  **50 million frames**** per game
  (38 days of play by a human)

- Dota 2 ***:
  1–3M steps per batch
  estimated **9.7 trillion steps**

- XLand
  - `fine tuning' --- **100M steps**
  - training of last (5$^{th}$)
    generation
    **> 100 billion steps**

** training used 'frame skipping' so 200M frames from environment needed
*** also 'frameskipping' so almost x4 frames from environment

# Deep RL methods are data hungry

- Atari: DQN was using
  **50 million frames**** per game
  (38 days of play by a human)
- 1-7 days on 1 GPU

- Dota 2 ***:
  1–3M steps per batch
  estimated **9.7 trillion steps**
- 10 months
  80k—173k CPUs: 7.5 steps/s
  1000s of GPUs

- XLand
  - \`fine tuning' --- **100M steps**
  - training of last (5th) generation
    **> 100 billion steps**
- 8 TPUv3s
  - 30mins
  - 23 days

** training used 'frame skipping' so 200M frames from environment needed
*** also 'frameskipping' so almost x4 frames from environment

# Ideas for improving sample efficiency

- Use models... (later)

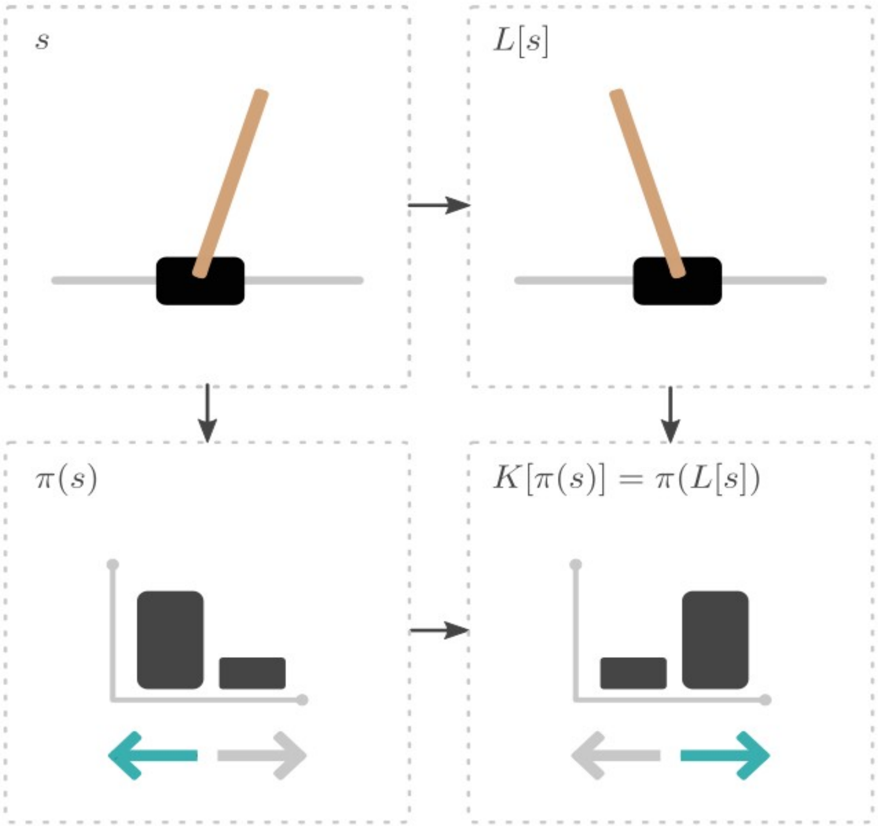
- Store as much as data as we can!
  - E.g., Replay memory
  - "Do we need a parametric model?"
    [Van Hasselt et al. 2019 NeurIPS]

- Data augmentation: exploit invariance

- Exploit symmetries...

# 1 Idea: Exploiting Symmetries

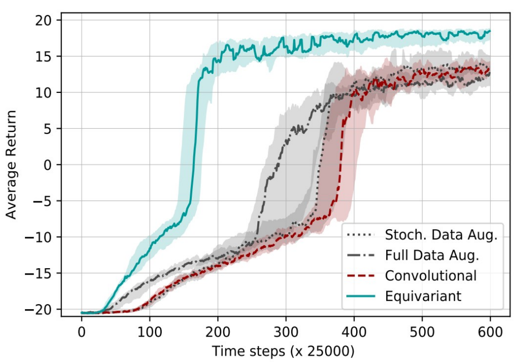
- Many RL problems exhibit symmetries
  - Symmetric (s, a) pairs
    should have the same policy




- Idea:
  → put constraints on network weights
  → enables more efficient learning

- **MDP homomorphic networks**
  for data-efficient RL
  - use a `symmetrizer' to construct equivariant weights
  
  → Fewer interactions with the world needed!

[van der Pol, Worrall, van Hoof, Oliehoek & Welling, NeurIPS, 2020]




reward in Pong
(as function of steps)

# Learning Models

# Learning models

- If we can a **learn model of the environment**
  → can generate new training data
  → and/or directly use in (online) planning (e.g. Alpha Go)

- In small problems, sure! Learn tables for
  - empirical transition probabilities
  - empirical rewards

- But when learning from sensor data...?


‘world models’
[Ha&Schmidhuber’18 NeurIPS]

# Constraints on the Latent Space

- Much recent work: learn latent representation
  (Deep MDP, Word Models, MuZero, etc. etc.)

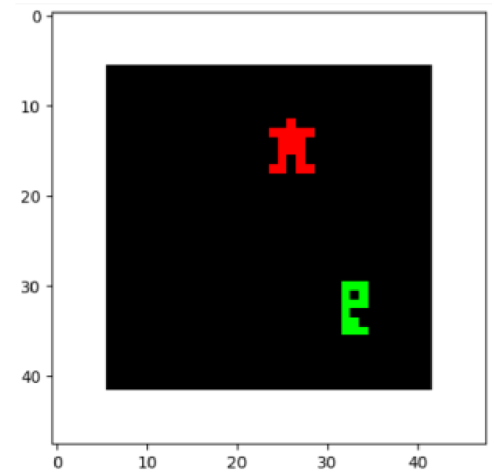
  - But also proved to be difficult...
    → need appropriate constraints!

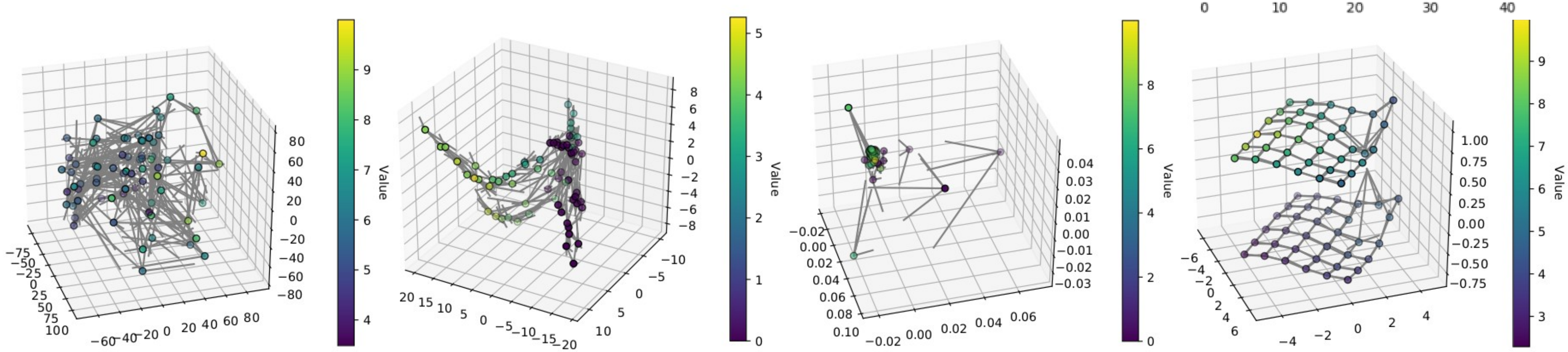
# Constraints on the Latent Space

- Much recent work: learn latent representation

  (Deep MDP, Word Models, MuZero, etc. etc.)

  - But also proved to be difficult...
    → need appropriate constraints!






(a) WM-AE Baseline (b) LD-AE Baseline (c) DMDP-HOM Baseline (d) This paper

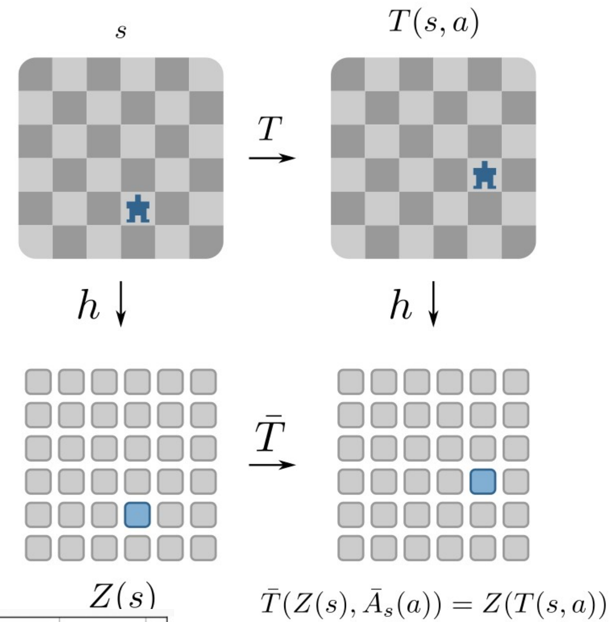
Abstract MDP. Nodes: abstract states, edges: abstract transitions, color: predicted value.

[van der Pol, Kipf, Oliehoek & Welling, AAMAS, 2020]

# Put constraints on the Latent Space

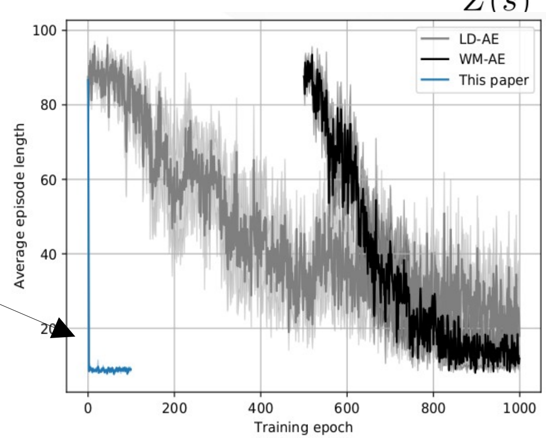
How?

- ▶ "MDP homomorpism metrics"
  - latent states need to predict the reward well
  - latent states need to predict the transitions well
- ▶ Enforce consistency (with 'contrastive learning')




The models support planning

- ▶ discretize
- ▶ apply standard planning (value iteration)
- ▶ much better sample complexity

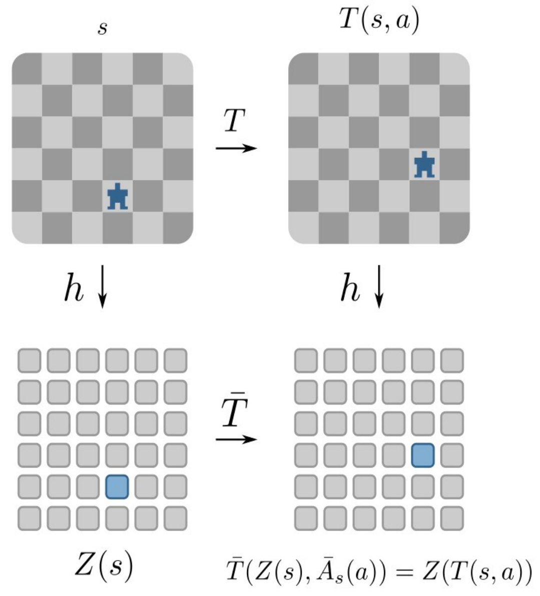



Abstract MDP. Nodes: abstract states, edges: abstract transitions, color: predicted value.

[van der Pol, Kipf, Oliehoek & Welling, AAMAS, 2020]

# Efficient-Zero [Ye e.a. 2021 NeurIPS]

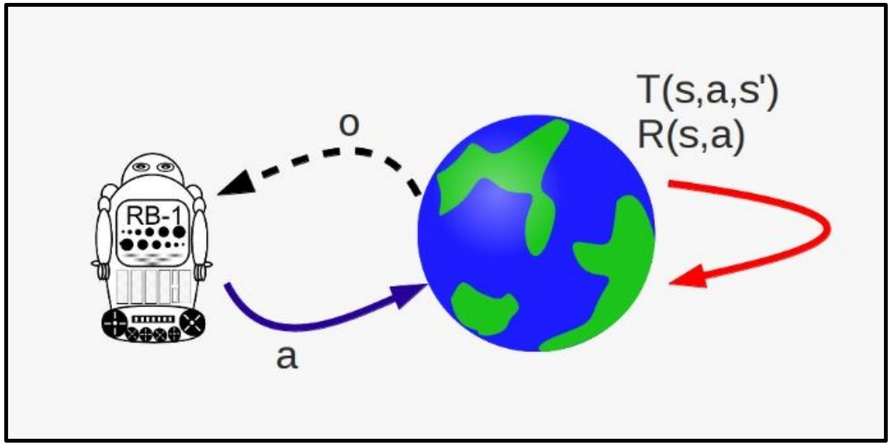
- MuZero: version of AlphaGo that learns a model
- Efficient-Zero improves sample complexity
  - 3 modifications
  - most impact: enforce the temporal consistency of the latent transition model
- Outperforms humans with **just 2h of ‘play-time’** per game.



# Partial Observability

# Partially observable RL (PORL)

- Rare for agent to see the Markov state
  → more often: just an **observation**



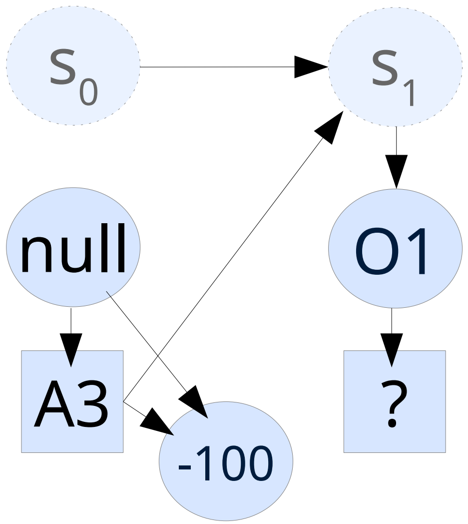

- But… also difficult… let's give it a try…

# Example

- your action: A1, A2, or A3?   → A3

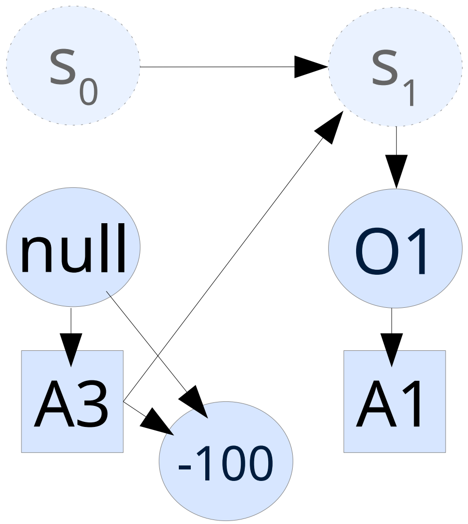
# Example

- your action: A1, A2, or A3? → A3
- you observed: -100, O1



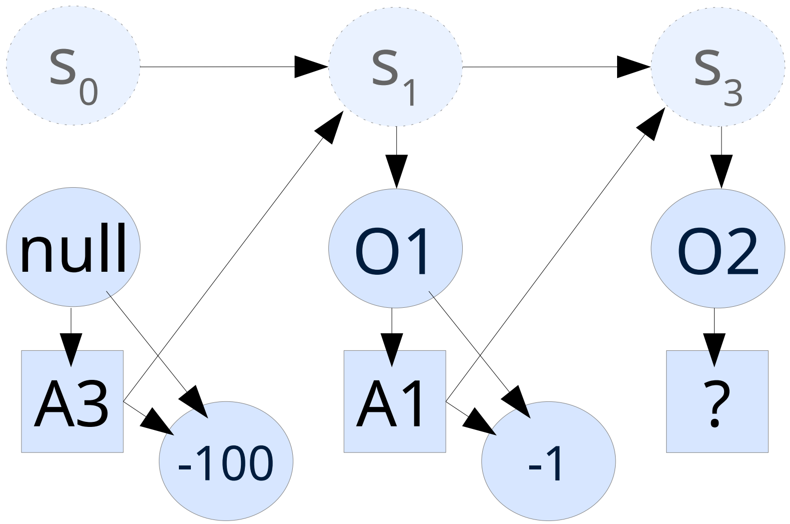
# Example

- your action: A1, A2, or A3?   → A3
- you observed:      -100, O1
- your action: A1, A2, or A3?    → A1



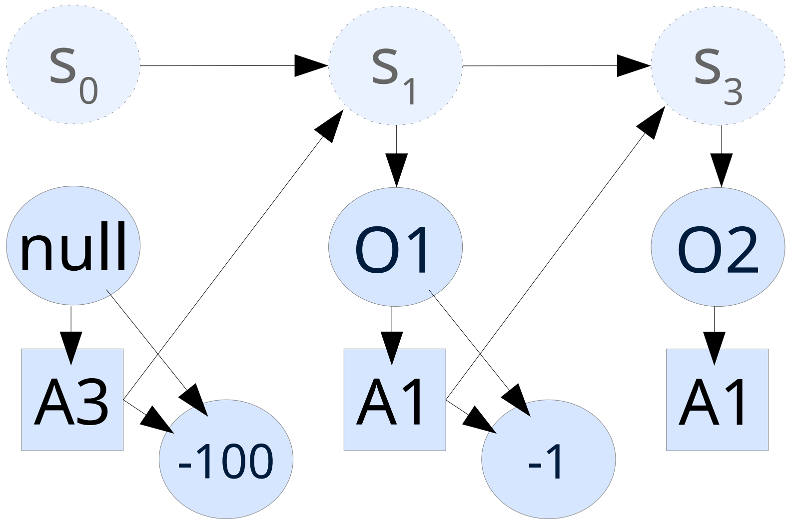
# Example

- your action: A1, A2, or A3? → A3
- you observed: -100, O1
- your action: A1, A2, or A3? → A1
- you observed: -1, O2



# Example

- your action: A1, A2, or A3? → A3
- you observed: -100, O1
- your action: A1, A2, or A3? → A1
- you observed: -1, O2
- your action: A1, A2, or A3? → A1



# Example

- your action: A1, A2, or A3? → A3
- you observed: -100, O1
- your action: A1, A2, or A3? → A1
- you observed: -1, O2
- your action: A1, A2, or A3? → A1
- you observed: -1, O2

# Example

- your action: A1, A2, or A3? → A3
- you observed: -100, O1
- your action: A1, A2, or A3? → A1
- you observed: -1, O2
- your action: A1, A2, or A3? → A1
- you observed: -1, O2
- your action: A1, A2, or A3? → A2
- you observed: +10, O1

# Example

- your action: A1, A2, or A3? → A3
- you observed: -100, O1
- your action: A1, A2, or A3? → A1
- you observed: -1, O2
- your action: A1, A2, or A3? → A1
- you observed: -1, O2
- your action: A1, A2, or A3? → A2
- you observed: +10, O1
- your action: A1, A2, or A3? → A2
- you observed: -100, O1

# Example

- your action: A1, A2, or A3? → A3
- you observed: -100, O1
- your action: A1, A2, or A3? → A1
- you observed: -1, O2
- your action: A1, A2, or A3? → A1
- you observed: -1, O2
- your action: A1, A2, or A3? → A2
- you observed: +10, O1
- your action: A1, A2, or A3? → A2
- you observed: -100, O1

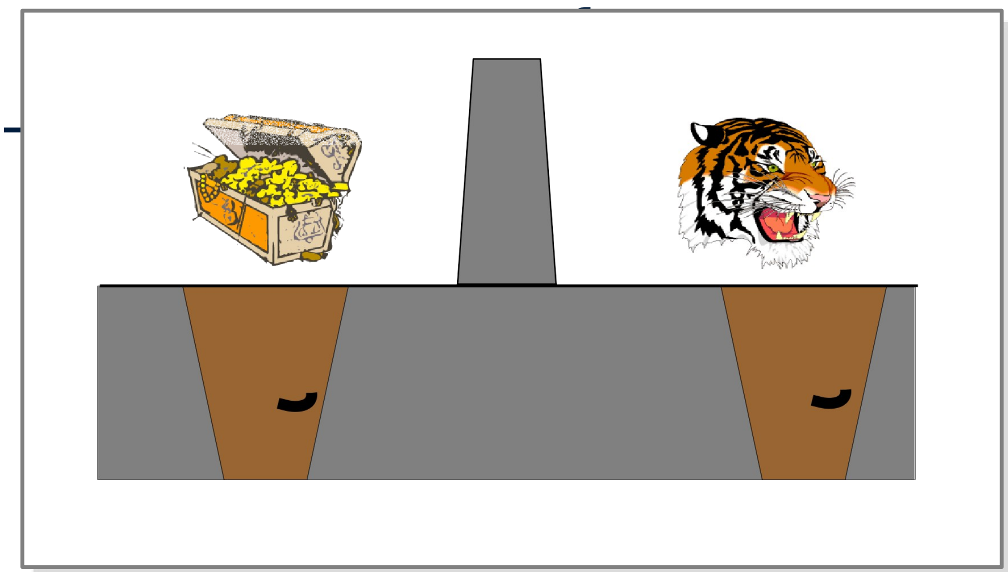
Fun, eh...?

# Example

- your action: A1, A2, or A3? → A3 =OpenRight
- you observed: -100, O1 =HearLeft
- your action: A1, A2, or A3? → A1 =Listen
- you observed: -1, O2 =HearRight
- your action: A1, A2, or A3? → A1 =Listen
- you observed: -1, O2 =HearRight
- your action: A1, A2, or A3? → A2 =OpenLeft
- you observed: +10, O1 =HearLeft
- your action: A1, A2, or A3? → A2 =OpenLeft
- you observed: -100, O1 =...

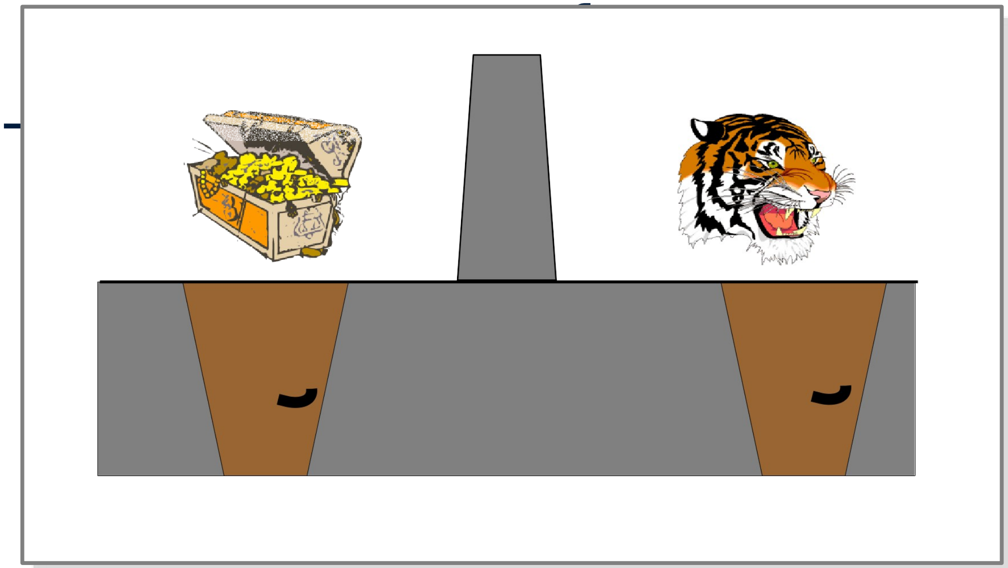
# Example

- your action: A1, A2, or A3? → A3 =OpenRight
- you observed: -100, O1 =HearLeft
- your action: A1, A2, or A3? → A1 =Listen
- you observed: -1, O2 =HearRight
- your action: A1, A2, or A3? → A1 =Listen
- you observed: -1, O2 =HearRight
- your action: A1, A2, or A3? → A2 =OpenLeft
- you observed: +10, O1
- your action: A1, A2, or A3?
- you observed: -100, O1



# Example

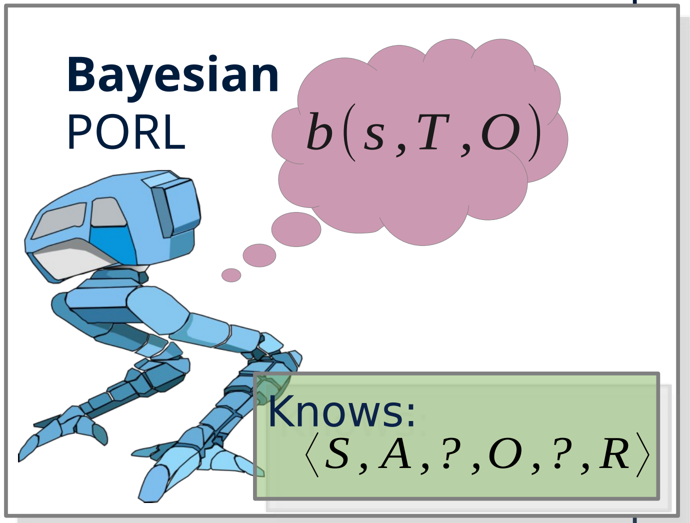
- your action: A1, A2, or A3? → A3 =OpenRight
- you observed: -100, O1 =HearLeft
- your action: A1, A2, or A3? → A1 =Listen
- you observed: -1, O2 =HearRight
- your action: A1, A2, or A3? → A1 =Listen
- you observed: -1, O2 =HearRight
- your action: A1, A2, or A3? → A2 =OpenLeft
- you observed: +10, O1
- your action: A1, A2, or A3?
- you observed: -100, O1



**Key points:**
- very hard problem
- but can get a lot of of prior knowledge

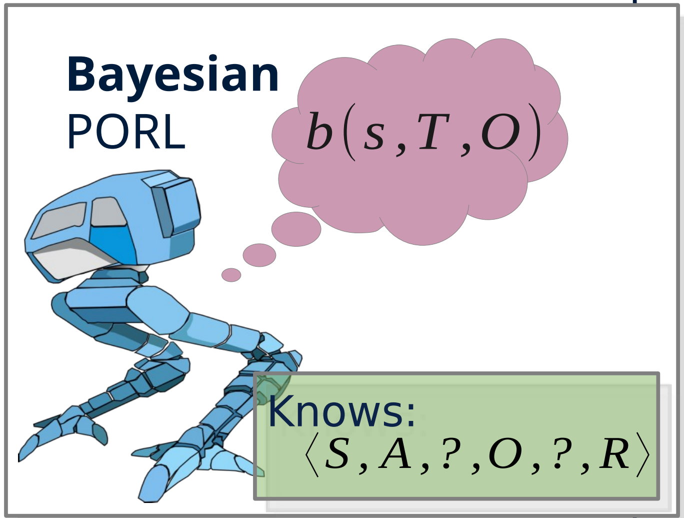
# Approaches to PORL

- Use prior knowledge
  → Bayesian RL: maintains
  belief over possible models
  - hard – even in tabular settings!
  - But there is progress
    [Katt e.a. 2019 AAMAS, Katt e.a. 2017 ICML]

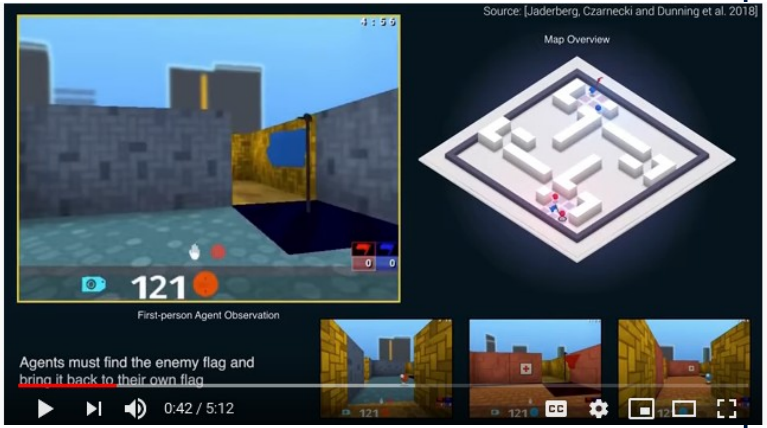


# Approaches to PORL

- Use prior knowledge
  → Bayesian RL: maintains
  belief over possible models
  - hard – even in tabular settings!
  - But there is progress
    [Katt e.a. 2019 AAMAS, Katt e.a. 2017 ICML]



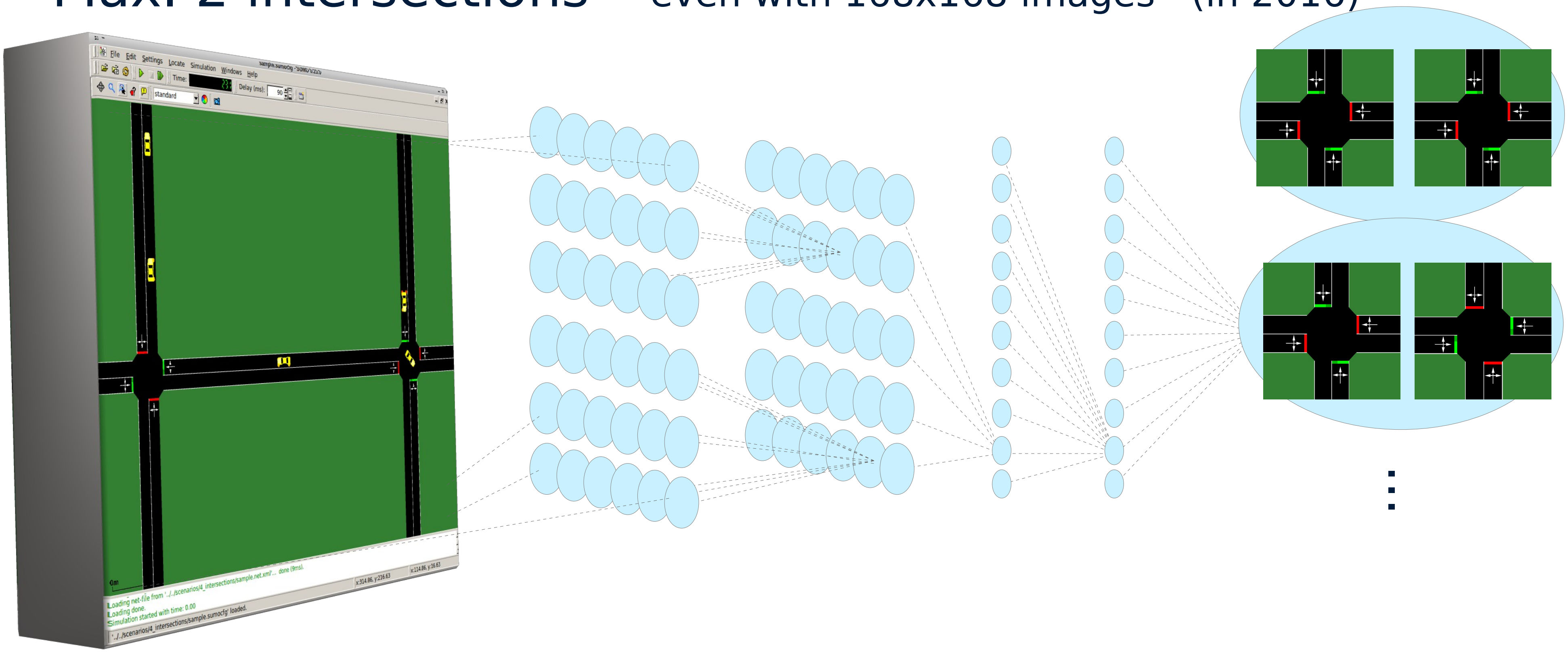

- Other approaches:
  - use recurrent neural networks or other deep learning
    [Schmidhuber et al since 1990s]
  - e.g., hierarchical LSTM for capture the flag [Jaderberg e.a. 2019 Science]



# Scalability & (more explicit forms of) Abstraction

# Only deep learning is not enough

- Max. 2 intersections – even with 168x168 images (in 2016)



**Problems:**

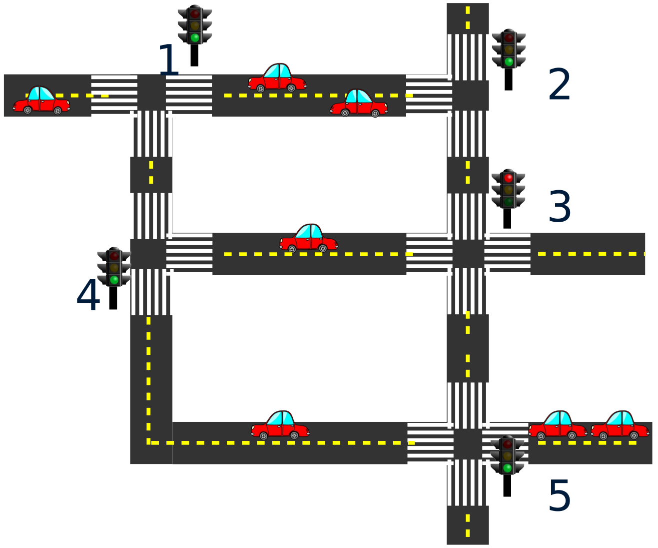
- ►inherent limitations on size of neural networks (e.g., GPU memory)
- ►training time prohibitive

# So how would we...

- Process information of an entire city?

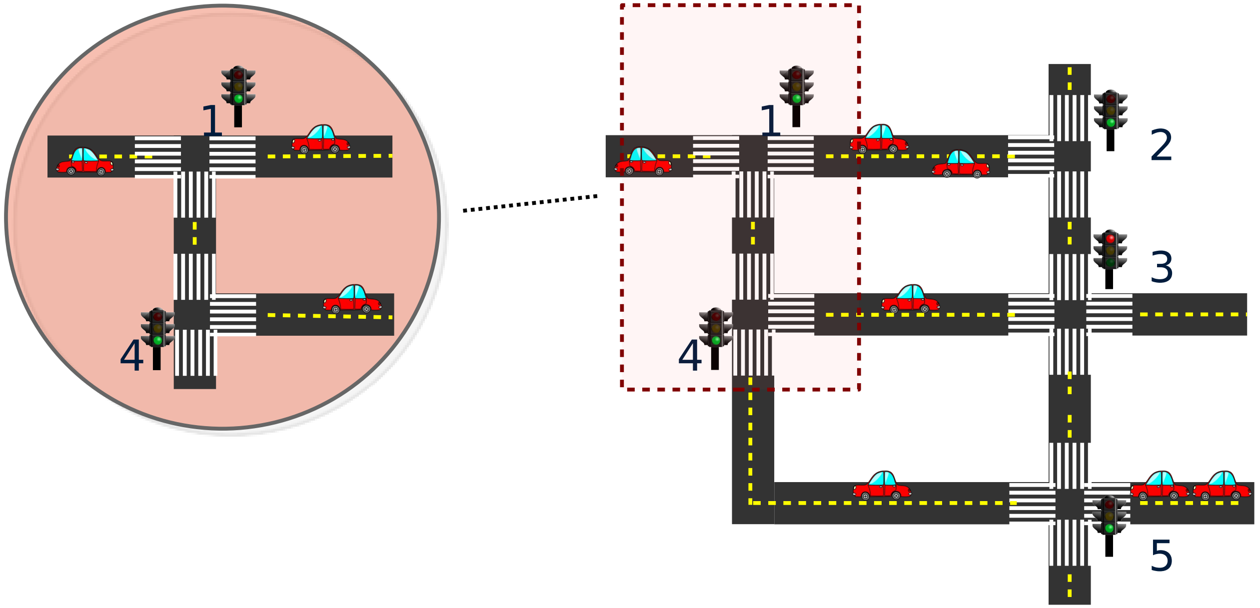
# The Intuition behind Abstraction

- Alternative: reason only about a small part of the problem



# The Intuition behind Abstraction

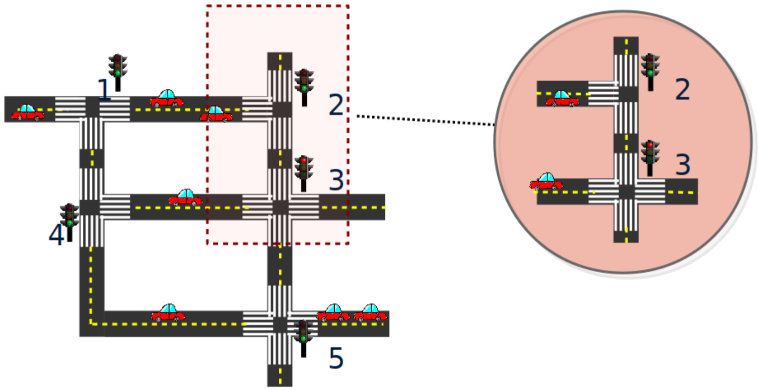
- Alternative: reason only about a small part of the problem



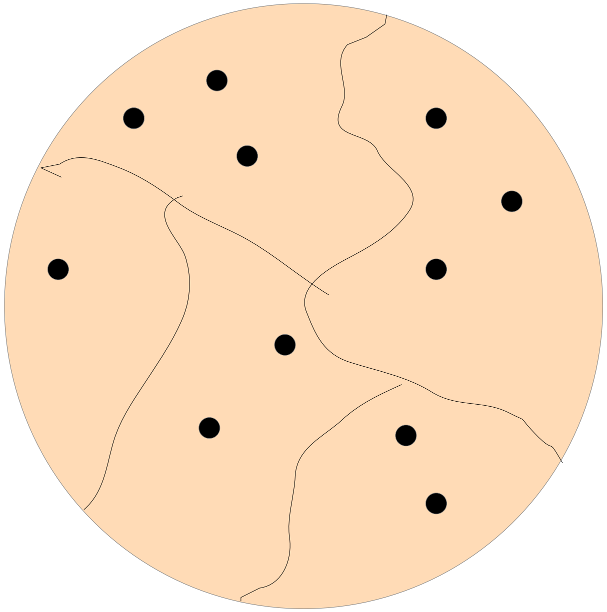
# How to do Abstraction?

- Can we reason over part of the system?

**“Well certainly not for all systems… they could be arbitrarily coupled…?”**



# Abstraction



- Number of states s is huge…
- Use an abstraction function $\varphi(s)$

- Given an MDP:
  construct an abstract MDP

  $T(\varphi'|\varphi,a) = \Sigma_{s' \in \varphi'} \Sigma_{s \in \varphi} T(s'|s,a)\ \omega_{\varphi}(s)$

  - For each abstract state $\varphi$, weighting function $\omega_{\varphi}(s)$ specifies the assumed state probabilities (link POMDPs)
  - Similar for rewards

- Under some assumptions ('ε-model similarity abstraction'): value loss bounded.

# Combining RL and Abstraction

- When the MDP is not known…

  → learn about abstract states directly?

- E.g., directly learn T(φ'|φ,a), R(φ,a) using model-based RL?

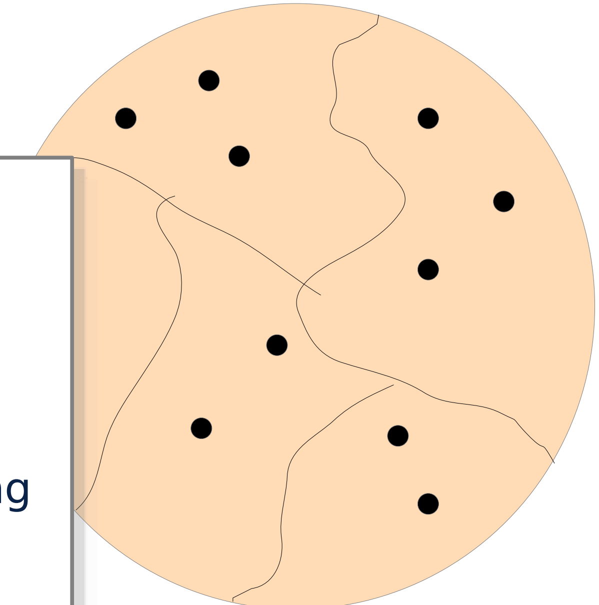
- Sure…

  …but guarantees for MBRL method may not hold!
  - These proofs are typically based on independence of samples
  - In some cases it is possible to fix by resorting to Martingale bounds

  Starre et al. 2022 arxiv "An Analysis of Abstracted Model-Based Reinforcement Learning"

# Abstraction

- Number of states s is huge...
- Use an abstraction function φ(s)



- Given an MDP:
  construct an abs

  $T(\varphi'|\varphi,a) = \Sigma_{s' \in \varphi'}$

Also...

these assumptions are rather strict...
→ typically do not hold when abstracting
away entire state variables...

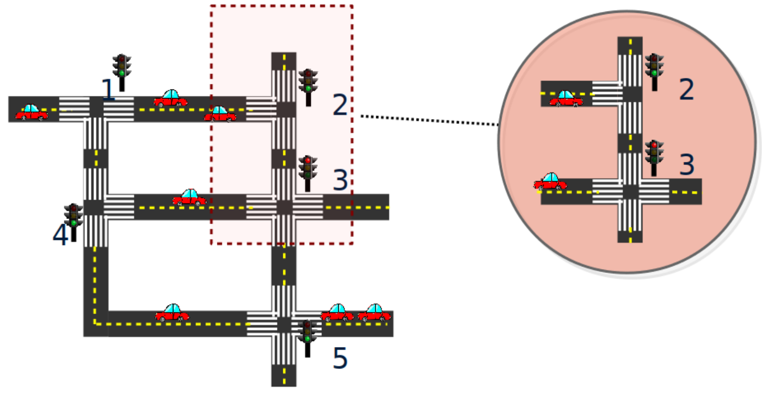
  - For each abstra $_{\varphi}$(s) specifies the assumed state probabilities (link POMDPs)
  - Similar for rewards

- Under some assumptions ('ε-model similarity abstraction'): value loss bounded.

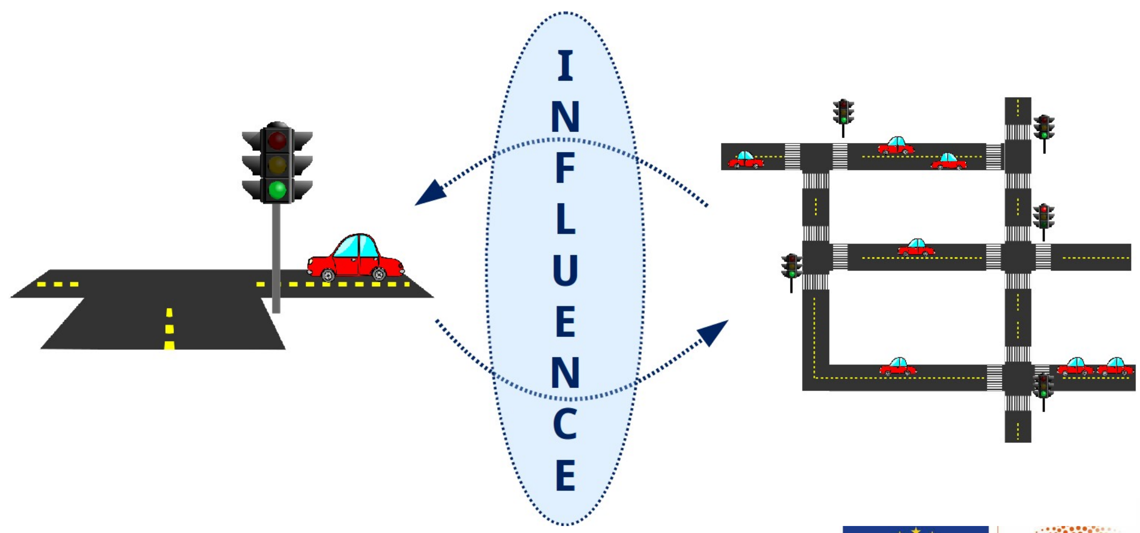
# How to do Abstraction?

- Can we reason over part of the system?

**“Well certainly not for all systems… they could be arbitrarily coupled…?”**




- INFLUENCE project:
  - Consider from perspective of local problem
  - Exploit accurate representations of **influence?**



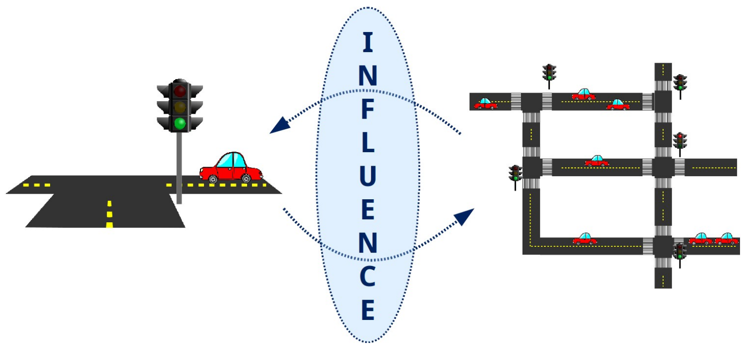

https://www.fransoliehoek.net/wp/2022/02/03/a-blog-about-influence/

European Research Council
Established by the European Commission

# INFLUENCE results

- Exploring
  **“Approximate influence points”**

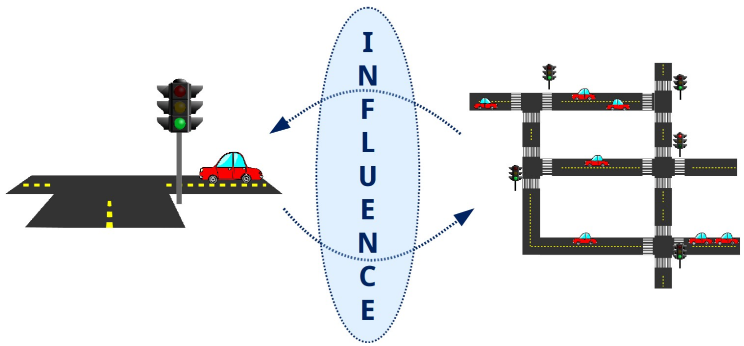
  - allows for decoupling local problem from rest of system



# INFLUENCE results

- Exploring
  **“Approximate influence points”**

  - allows for decoupling local problem from rest of system




how to learn?
→ ‘normal’ CE loss

**Theorem 2.** Consider an IALM $\mathcal{M} = (\mathcal{S}, A, \mathcal{T}, R, h, b^0)$ and an AIP $\hat{I}$ inducing $\hat{\mathcal{M}} = (\mathcal{S}, A, \hat{\mathcal{T}}, R, h, b^0)$. Then, a value loss bound in terms of the 1-norm error is given by

$$||\mathcal{V}_h^* - \mathcal{V}_h^{\hat{\pi}^*}||_\infty \leq 2h^2|R| \max_{t,\,d^t} ||(I^t(\cdot|\,d^t) - \hat{I}^t(\cdot|\,d^t))||_1. \qquad (4)$$

Elena Congeduti, Alexander Mey, and Frans A. Oliehoek. Loss Bounds for Approximate Influence-Based Abstraction. AAMAS’21

# INFLUENCE results

- Exploring
  **“Approximate influence points”**

  - allows for decoupling local problem from rest of system

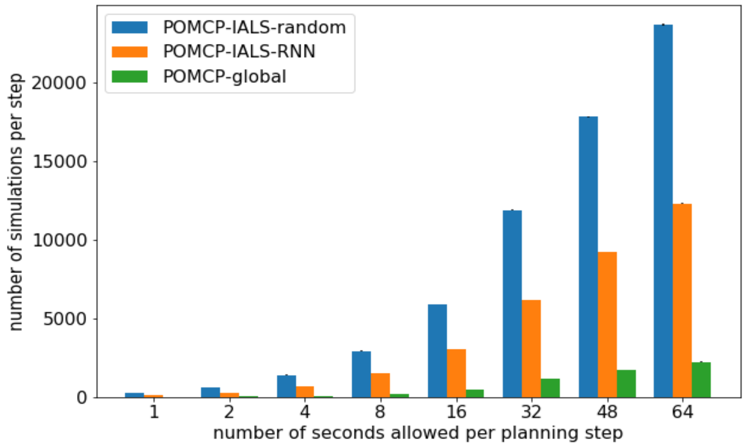
how to learn?
→ ‘normal’ CE loss

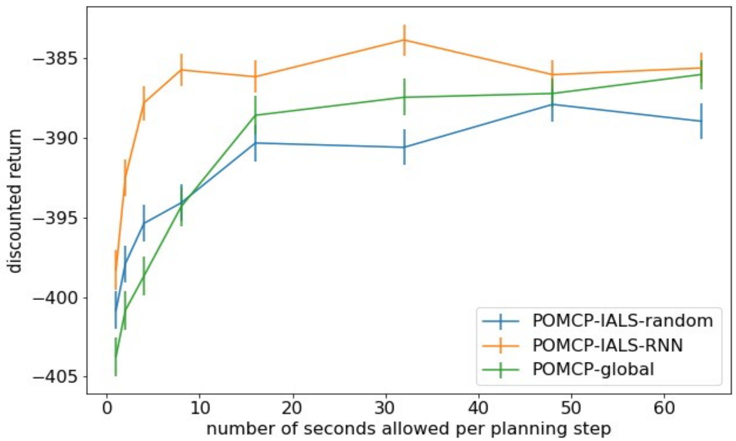
**Theorem 2.** Consider an IALM $\mathcal{M} = (\mathcal{S}, A, \mathcal{T}, R, h, b^0)$ and an AIP $\hat{I}$ inducing $\hat{\mathcal{M}} = (\mathcal{S}, A, \hat{\mathcal{T}}, R, h, b^0)$. Then, a value loss bound in terms of the 1-norm error is given by

$$||\mathcal{V}_h^* - \mathcal{V}_h^{\hat{\pi}^*}||_\infty \leq 2h^2|R| \max_{t, d^t} ||(I^t(\cdot|\, d^t) - \hat{I}^t(\cdot|\, d^t)||_1. \quad (4)$$

Elena Congeduti, Alexander Mey, and Frans A. Oliehoek. Loss Bounds for Approximate Influence-Based Abstraction. AAMAS’21

how to use in MCTS?
→ construct a ‘local simulator’




Jinke He, Miguel Suau, and Frans A. Oliehoek. Influence-Augmented Online Planning for Complex Environments. In NeurIPS, 2020.

# Multiagent Systems

# So how would we...

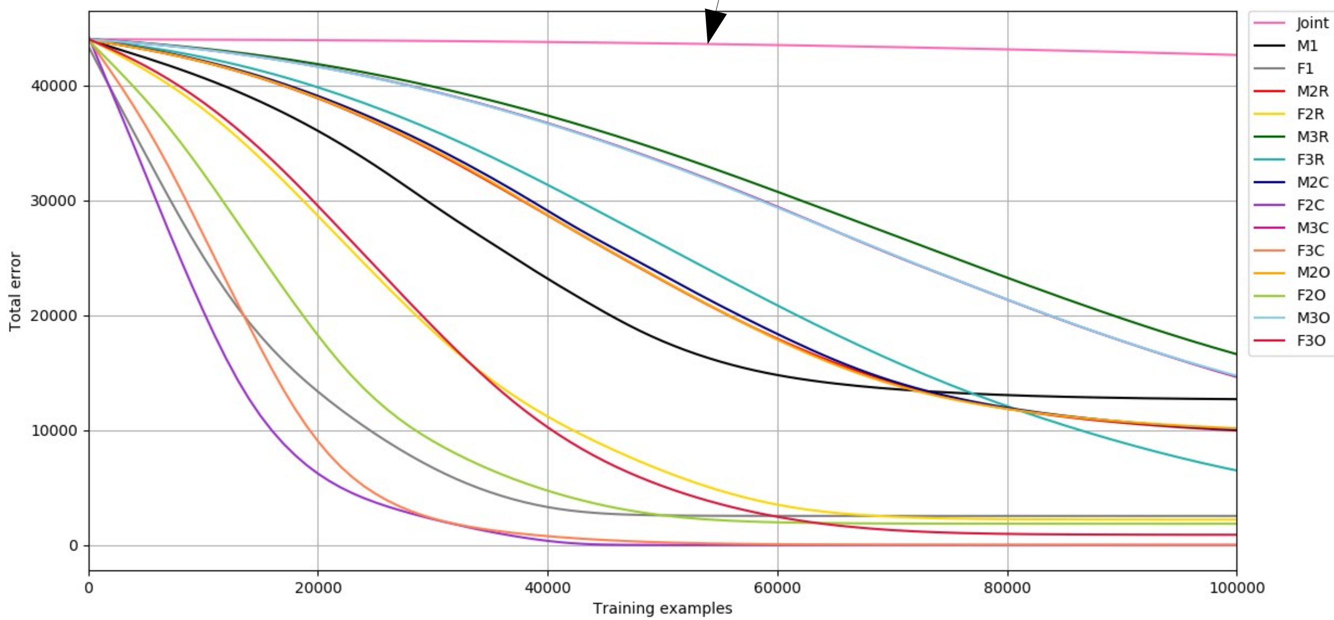
- coordinate traffic control in an entire city?

# But only deep learning is not enough

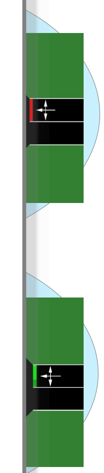
- Max

E.g., learning Q(s,a) for a large (multiagent) problem

“plain deep learning”




Jacopo Castellini, Frans A. Oliehoek, Rahul Savani, and Shimon Whiteson. The Representational Capacity of Action-Value Networks for Multi-Agent Reinforcement Learning. In Proceedings of the Eighteenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2019

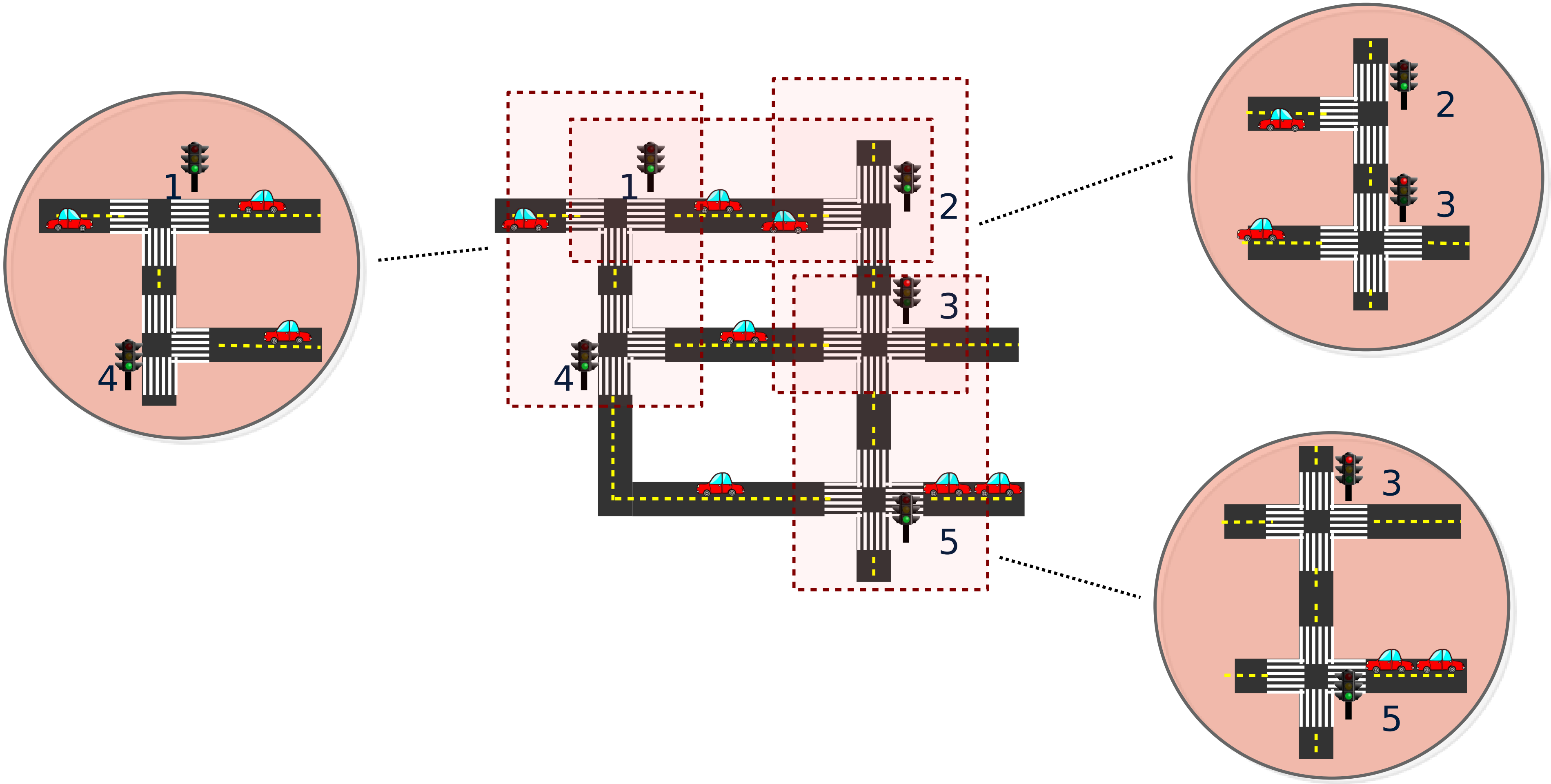


**Problems:**

- inherent limitations on size of neural networks (e.g., GPU memory)
- training time prohibitive
- **joint action spaces scale exponentially**

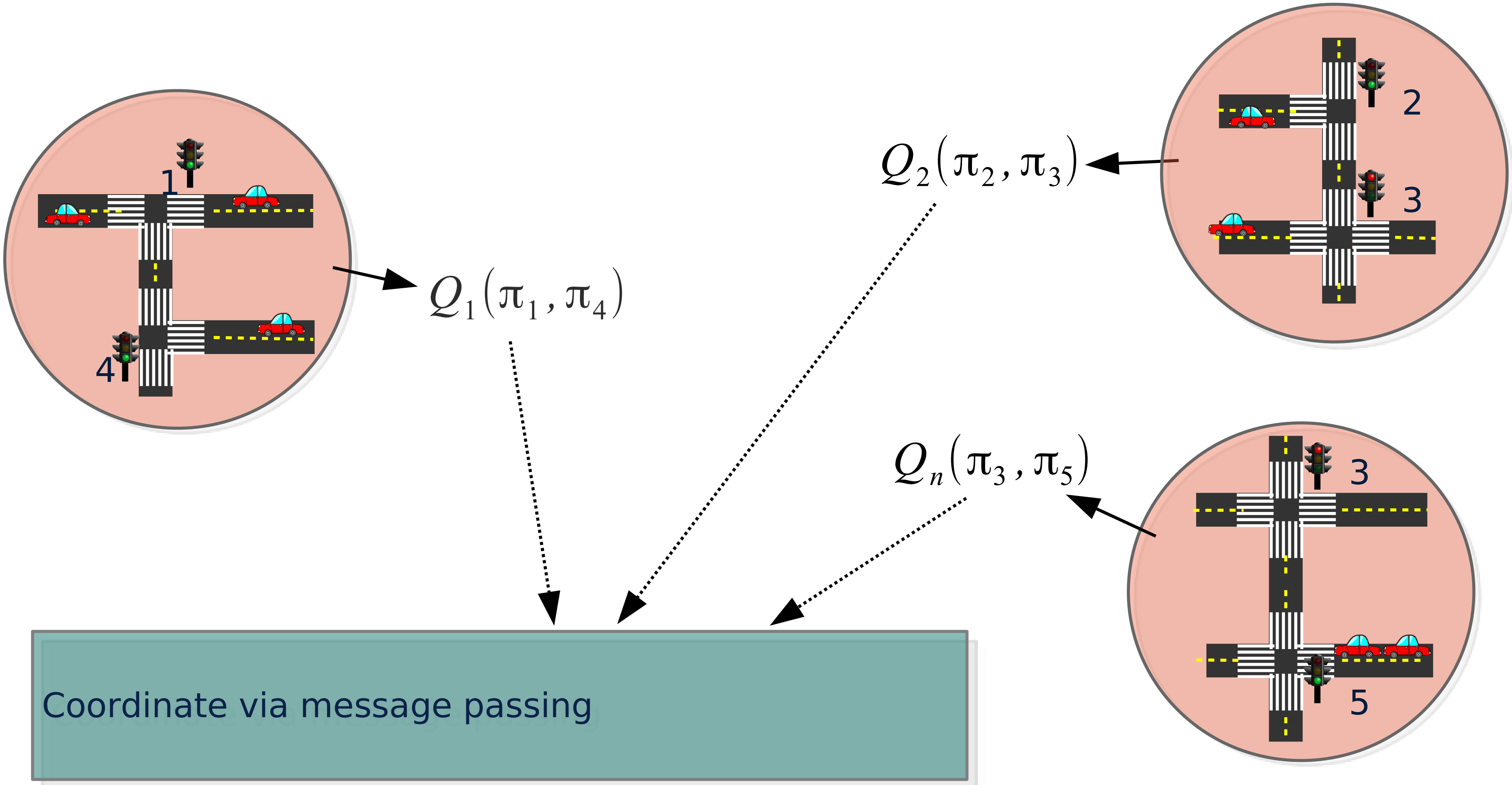
# Abstraction for Multiagent Problems

- Reason about multiple sub-problems



# E.g.: Transfer Planning

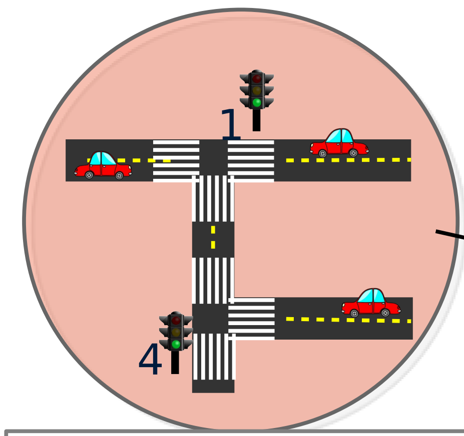
- Solve source problems independently
  - Also “factored value functions”




Frans A. Oliehoek, Shimon Whiteson, and Matthijs T. J. Spaan. Approximate Solutions for Factored Dec-POMDPs with Many Agents. In Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems, pp. 563–570, 2013.

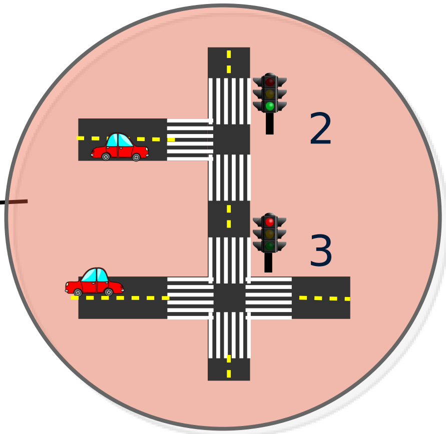
# E.g.: Transfer Planning

- Solve source problems independently
  - Also “factored value functions”

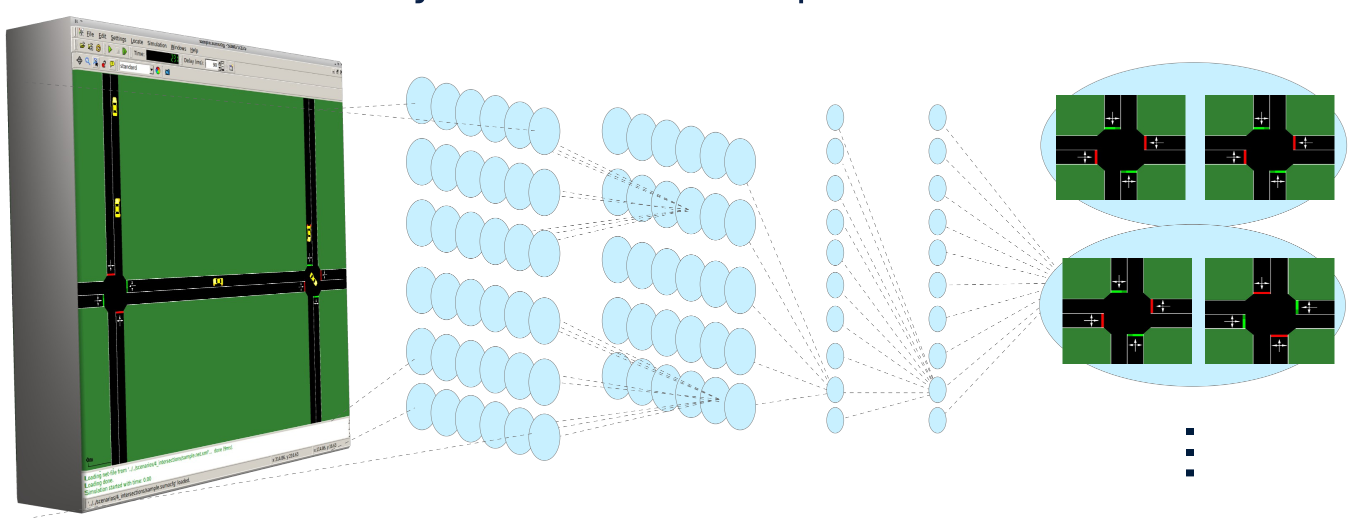


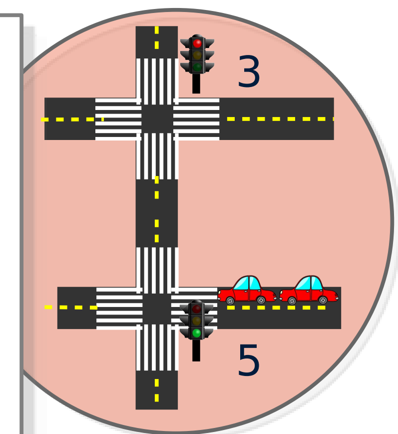

$Q_1(\pi_1, \pi_4)$

$Q_2(\pi_2, \pi_3)$




And free to choose way in which source problems are solved!

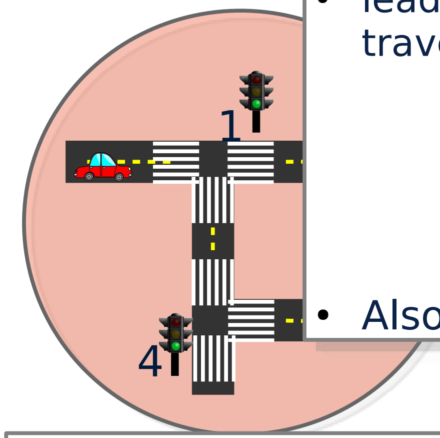





y Agents. In Proceedings of the

# E.g.: T

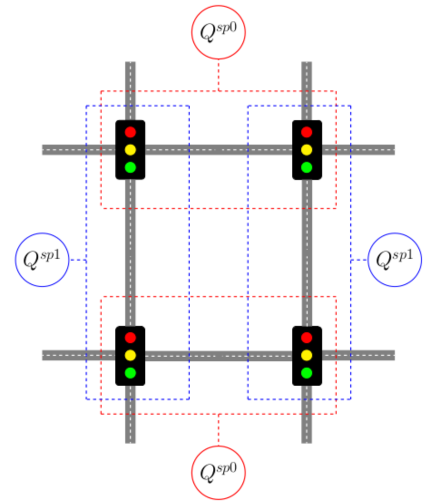
- Solve s
  - Also “fac

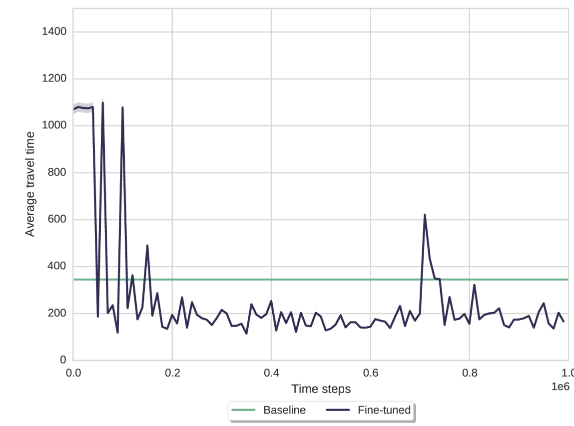



## Coordinated Deep Reinforcement Learning

[Van der Pol & Oliehoek, 2016]

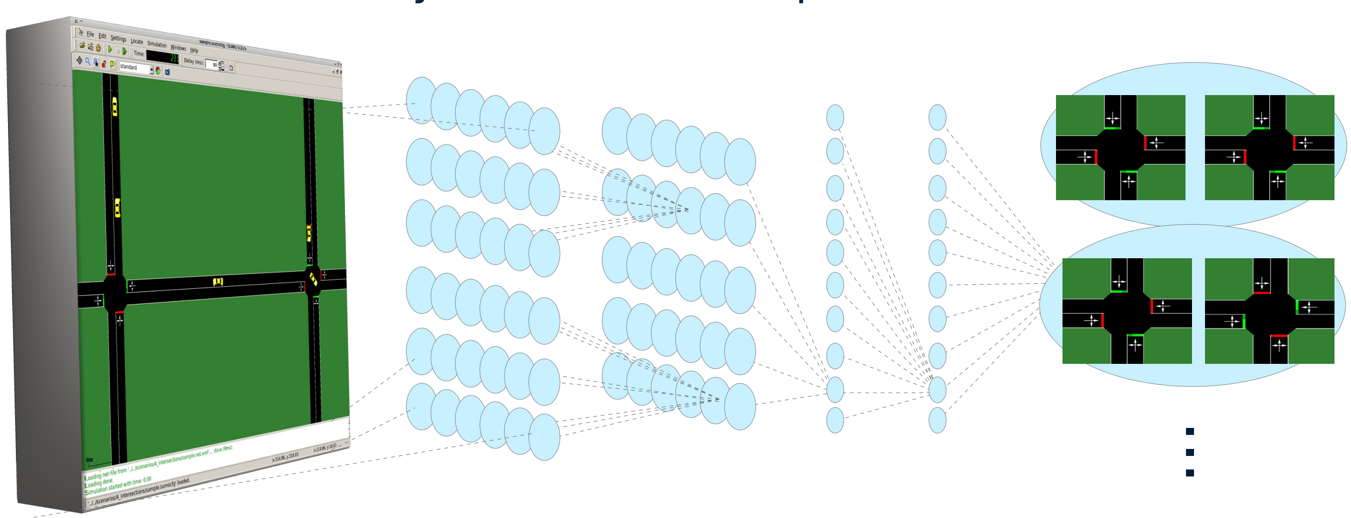
- DQN learners
- on different SUMO configurations
- leading to lower average travel time

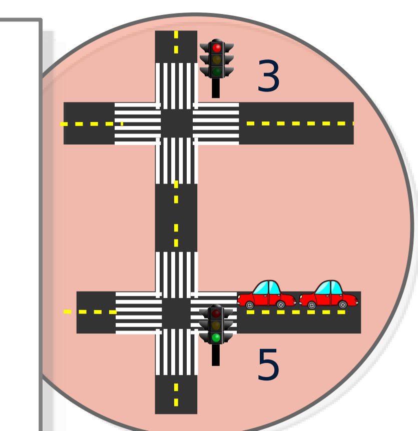



- Also: http://www.fransoliehoek.net/trafficvideo

And free to choose way in which source problems are solved!

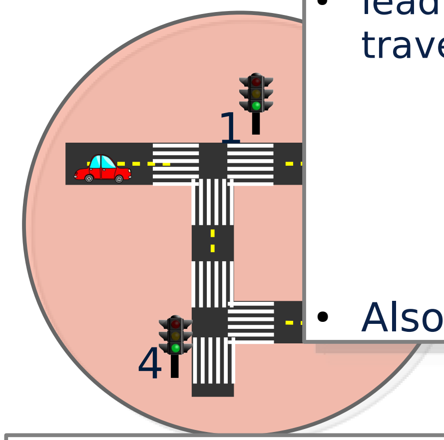





y Agents. In Proceedings of the

# E.g.: T

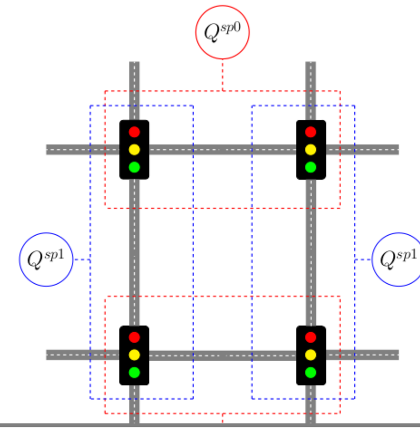
- Solve s
  - Also “fac

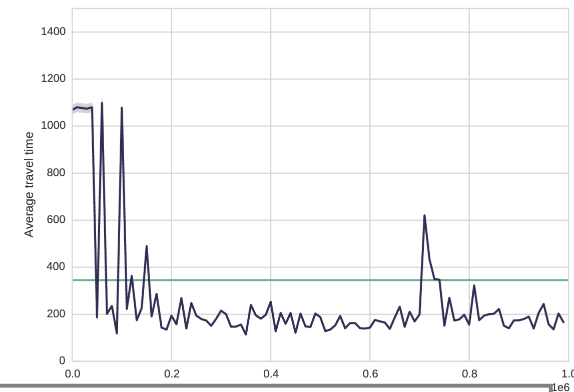



## Coordinated Deep Reinforcement Learning

[Van der Pol & Oliehoek, 2016]

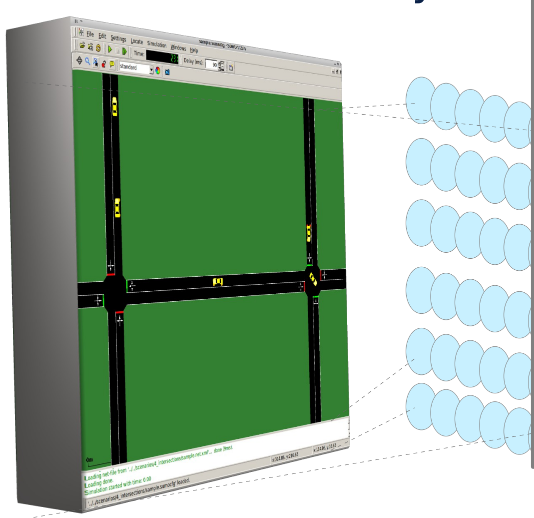
- DQN learners
- on different SUMO configurations
- leading to lower average travel time




- Also: http://w

And free to choose way in w



## Value factorization has become a major technique in multiagent RL

- ►Guestrin'02 (NIPS'01) - ‘factored value functions’
- ►learned message passing [Sukhbaatar et al. 2016]
- ►Value Decomposition networks [Sunehag et al. 2018]
- ►Q-Mix [Rashid et al. 2018]
- ►Deep Coordination Graphs [Böhmer et al. [2019]
- ►Capacity of factored value functions [Castellini et al. 2021]
- ►etc.

# Multiagent MCTS

MCTS in multiagent settings?

- developed for games
- but **does not scale well with number of agents**...

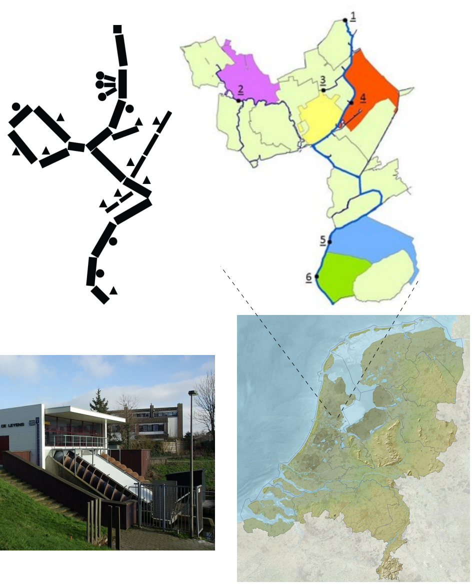
# Multiagent MCTS

2 main ideas:

- Apply value factorization inside MCTS tree

  [Amato&Oliehoek'15 AAAI]

**Coordinated control of pumping stations**



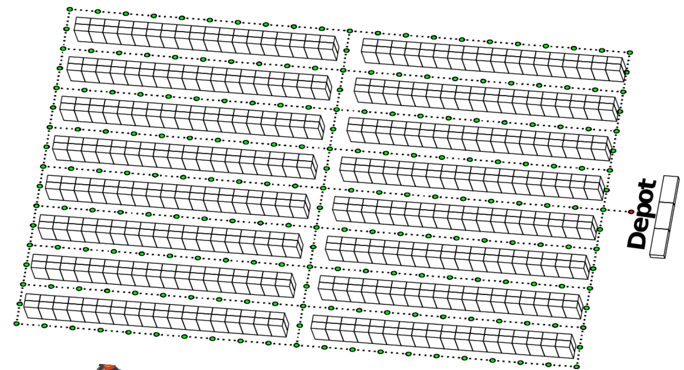
# Multiagent MCTS

2 main ideas:

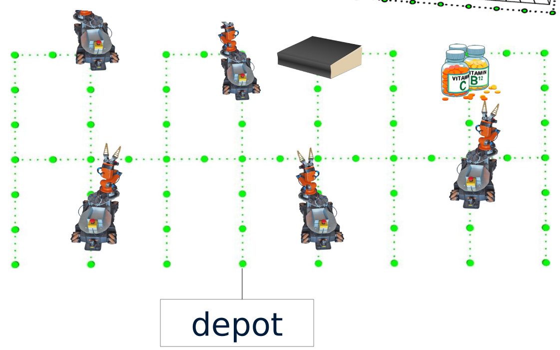
- Apply value factorization inside MCTS tree
  [Amato&Oliehoek'15 AAAI]
- Decentralized MCTS: predict teammates

**Coordinated control of pumping stations**

**Warehouse Commissioning**
[Claes et al.'17 AAMAS]




Toru Robot
(Magazino GmbH)



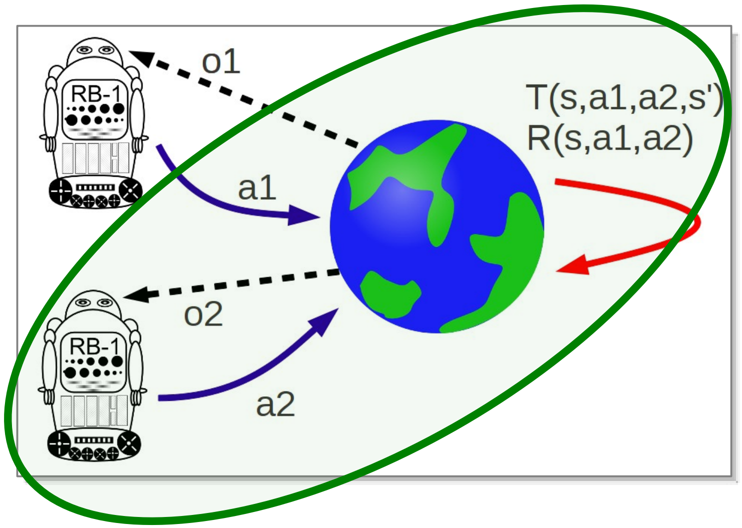

modeled as a graph

Live demo
at swarmlab,
Univ. Liverpool

# Other Agents & Nonstationarity

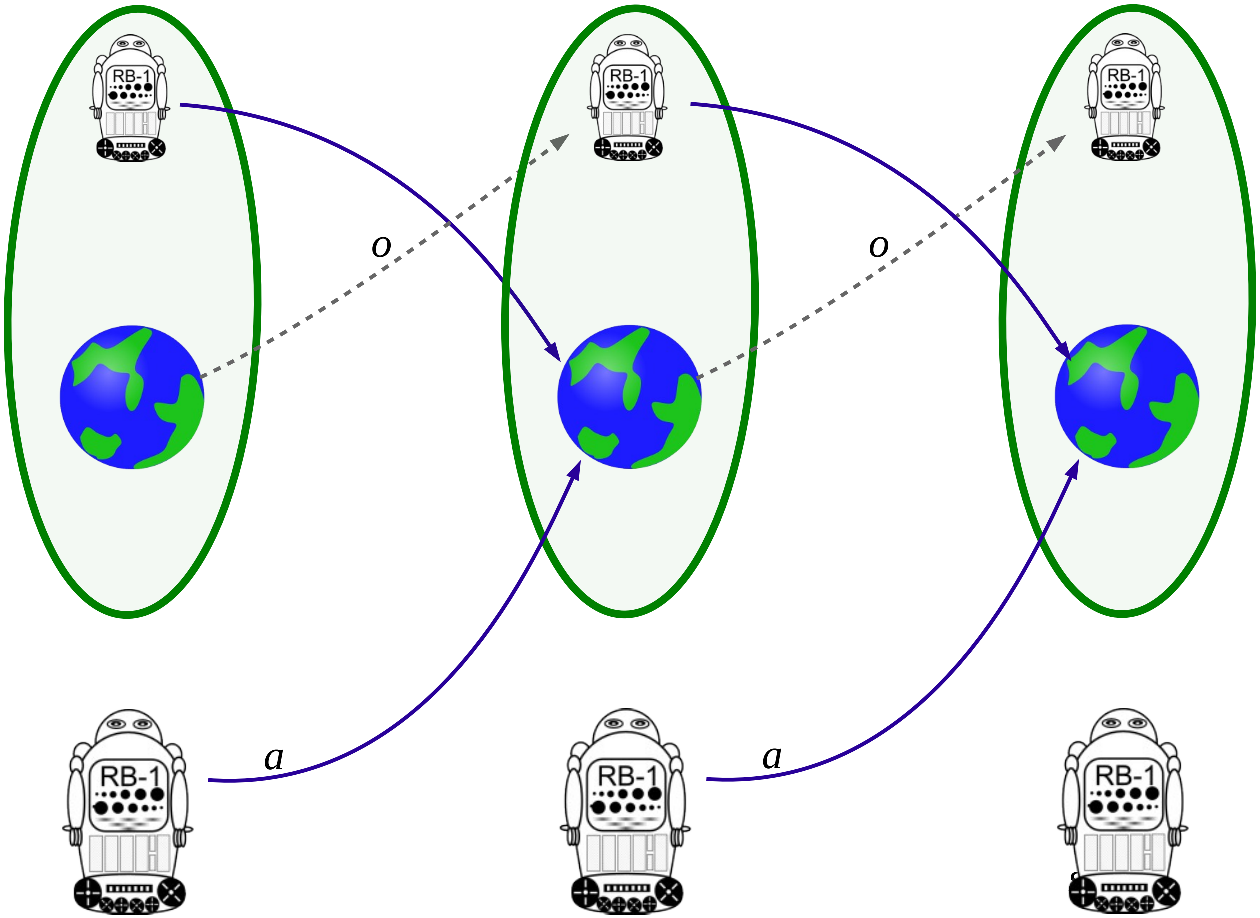
- If other agents change...
  the world is non-stationary!

- I.e., it is part of the environment



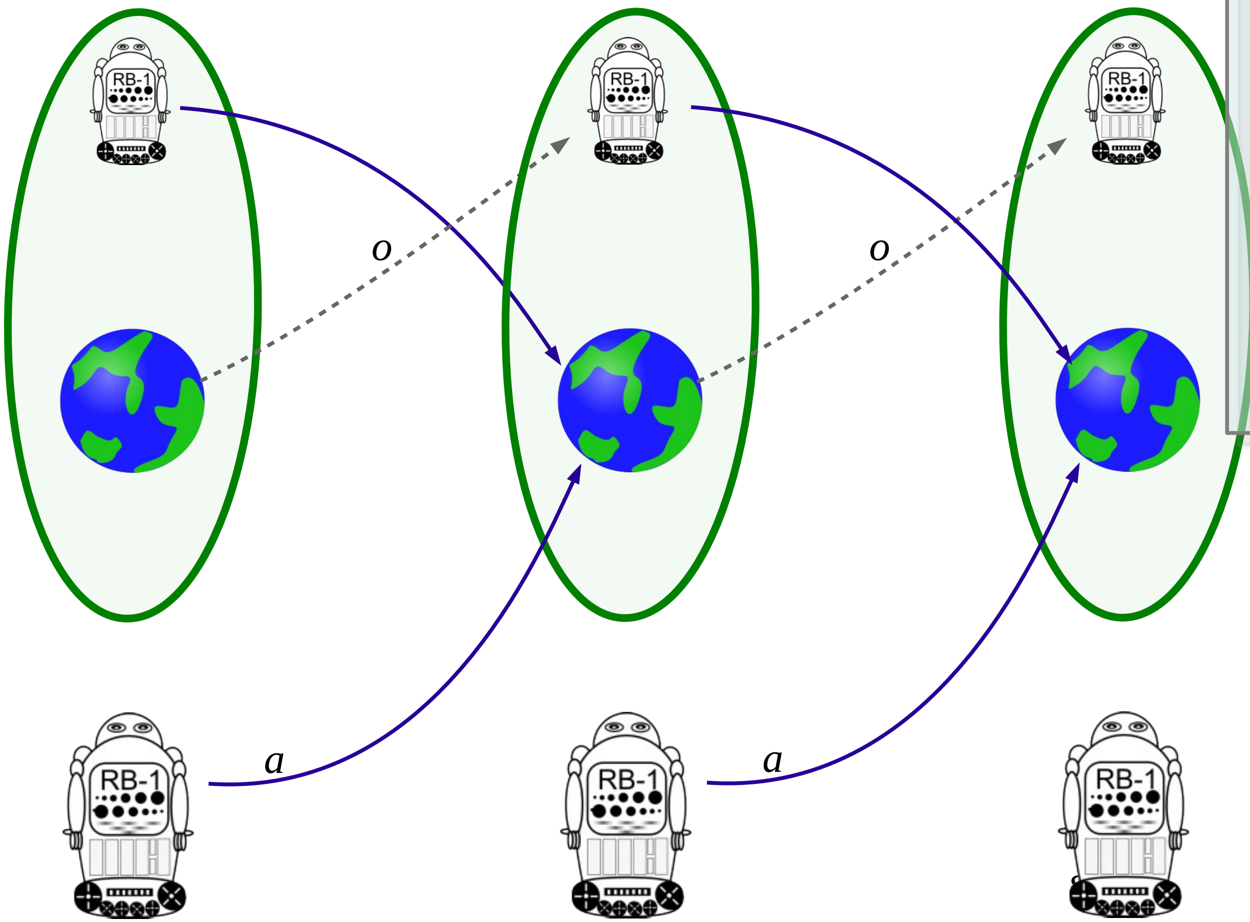
# Other Agents & Nonstationarity

- One idea: try to model it!



# Other Agents & Nonstationarity

- One idea: try to model it!

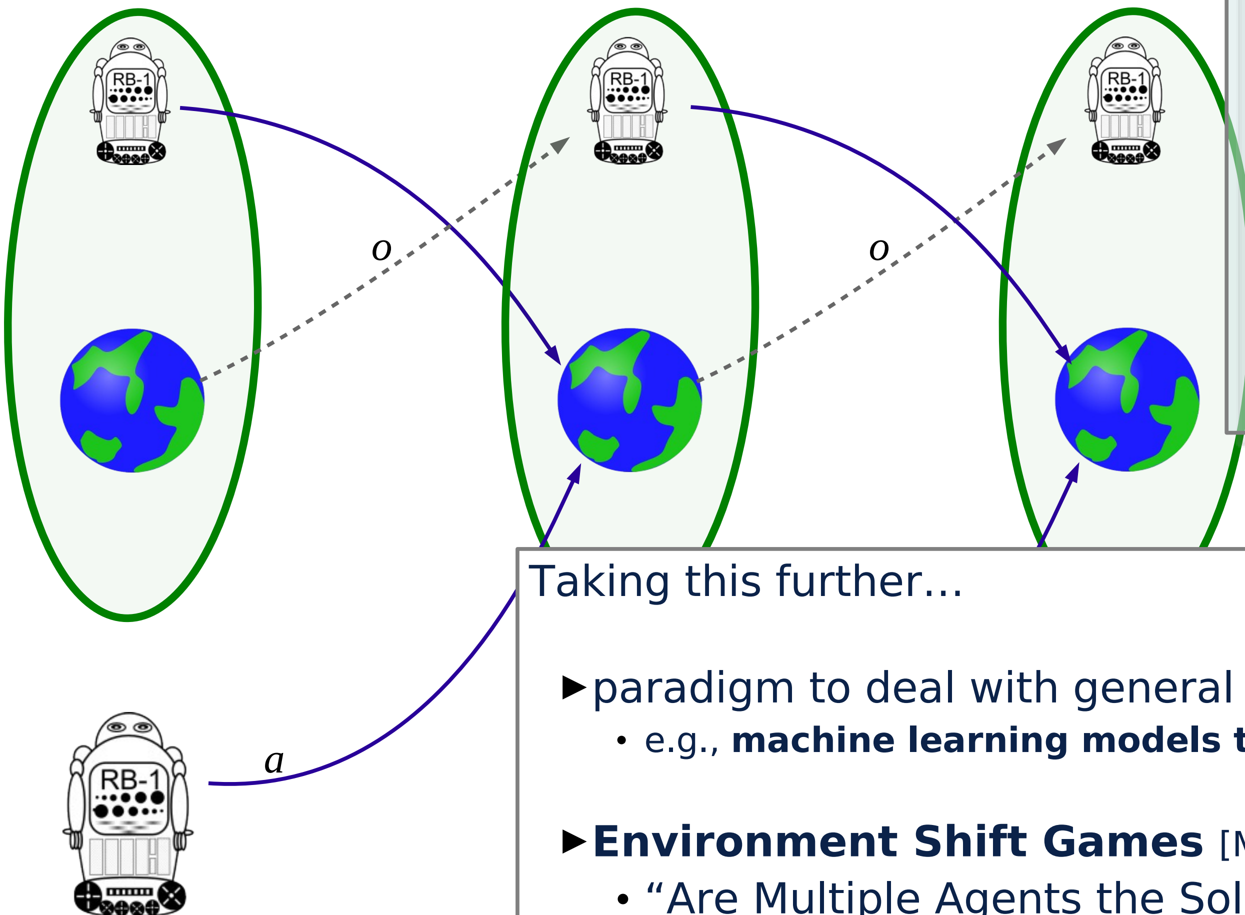



INFLUENCE suggest this might be possible

- certainly if world is structured
- and 'influence strength' of other agent limited

# Other Agents & Nonstationarity

- One idea: try to model it!




INFLUENCE suggest this might be possible

- certainly if world is structured
- and ‘influence strength’ of other agent limited

Taking this further...

- paradigm to deal with general sources of non-stationarity
  - e.g., **machine learning models that affect their own data.**
- **Environment Shift Games** [Mey & O. 2021 AAMAS Blue Sky]
  - “Are Multiple Agents the Solution, and not the Problem, to Non-Stationarity?”

# Learning Reward Functions
# or Dealing with Unknown Rewards

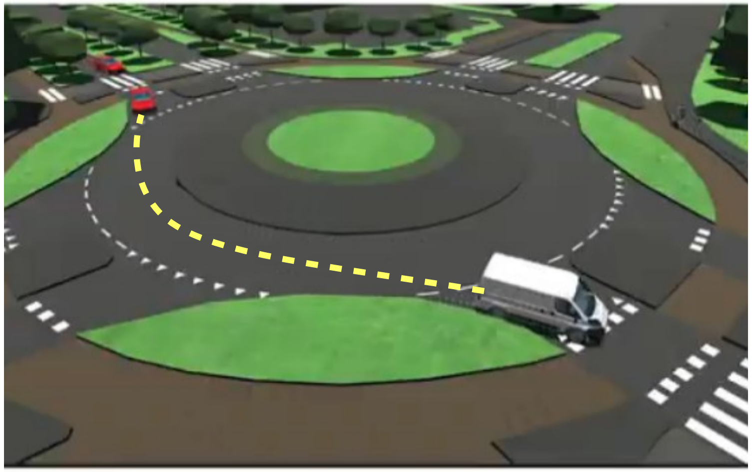
# Dealing with unknown rewards

- Specifying rewards can be tricky
- E.g., when dealing with humans:
  - how much distance should a robot keep?
  - how to approach an intersection like a human driver?
    [Neumeyer'21 IEEE RAL]
- Learning from demonstration

# Dealing with unknown rewards

- Specifying rewards can be tricky
- E.g., when dealing with humans:
  - how much distance should a robot keep?
  - how to approach an intersection like a human driver? [Neumeyer'21 IEEE RAL]
- Learning from demonstration

Learning from demonstration
- ►models of car driving behavior
- ►from camera data
- ►using GAIL [Ho&Ermon 2016 NeurIPS]: learns a classifier punish learning agent when not human-like

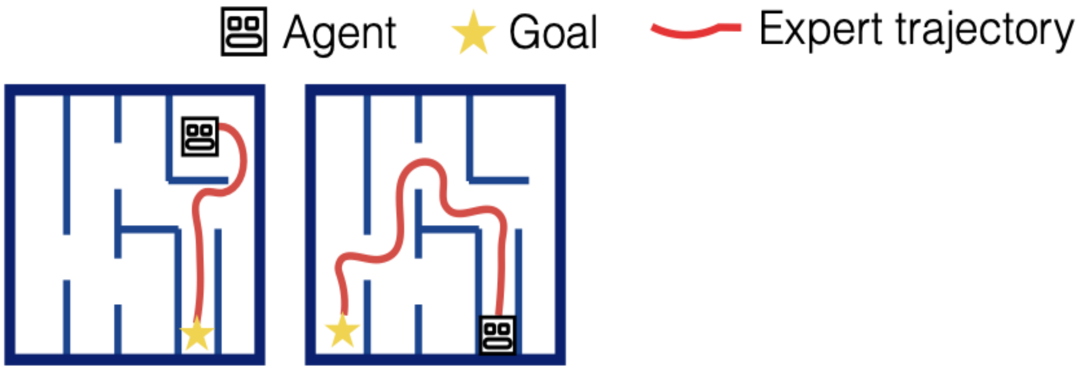


Learned car behaviors
[Behbahani et al. 2019]

# Dealing with unknown rewards

## Abstraction-Guided Policy Recovery from Expert Demonstrations (RECO)

- Expert data is sparse...

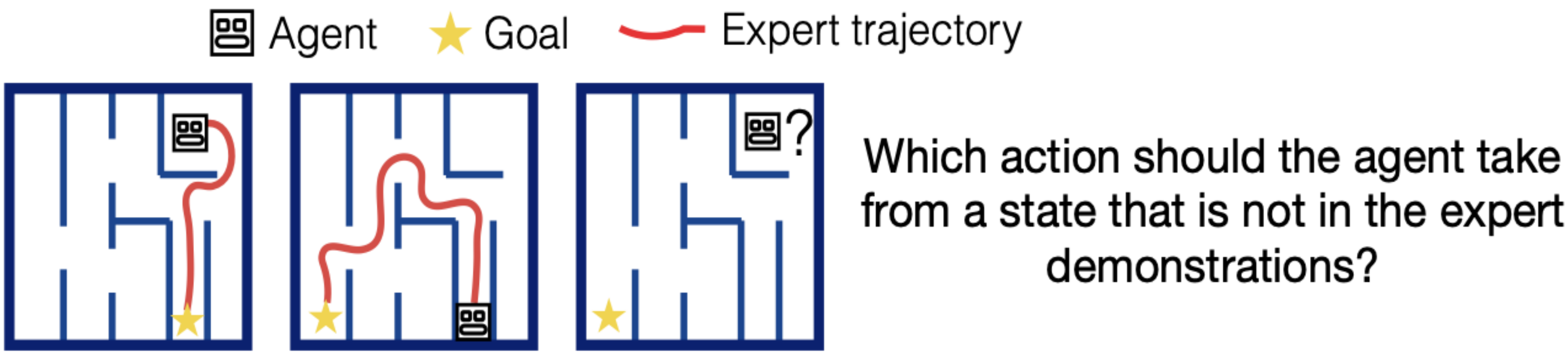



Ponnambalam, C.T., Oliehoek F.A., Spaan, M.T.J. (2021). Thirty-First International Conference on Automated Planning and Scheduling (ICAPS).

# Dealing with unknown rewards

**Abstraction-Guided Policy Recovery from Expert Demonstrations (RECO)**

- Expert data is sparse...

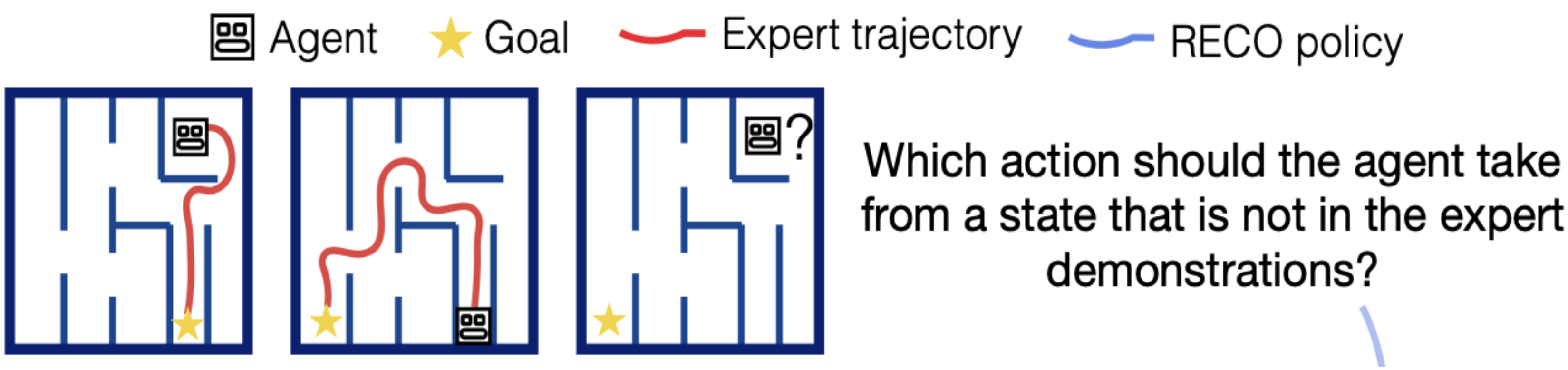



Ponnambalam, C.T., Oliehoek F.A., Spaan, M.T.J. (2021). Thirty-First International Conference on Automated Planning and Scheduling (ICAPS).

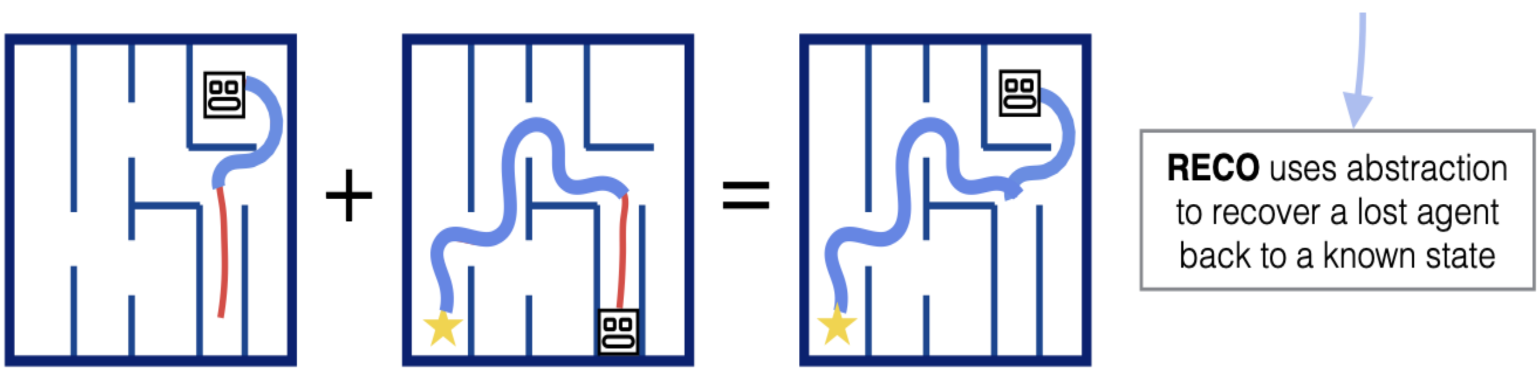
# Dealing with unknown rewards

## Abstraction-Guided Policy Recovery from Expert Demonstrations (RECO)

- Expert data is sparse...




- ...use appropriate abstractions to piece together expert demonstrations:



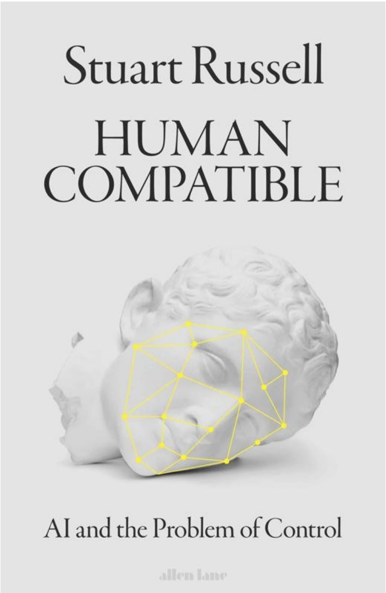

Ponnambalam, C.T., Oliehoek F.A., Spaan, M.T.J. (2021). Thirty-First International Conference on Automated Planning and Scheduling (ICAPS).

# Value Alignment

- More generally:
  how do we get AI to do
  what we really want?


https://openai.com/blog/faulty-reward-functions/

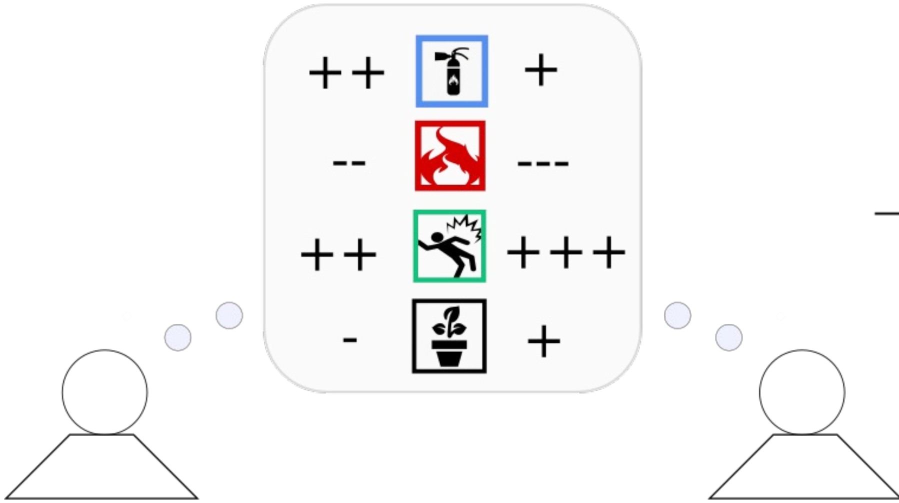
- how to incentivize AI?
  - prevent the terminator scenario.



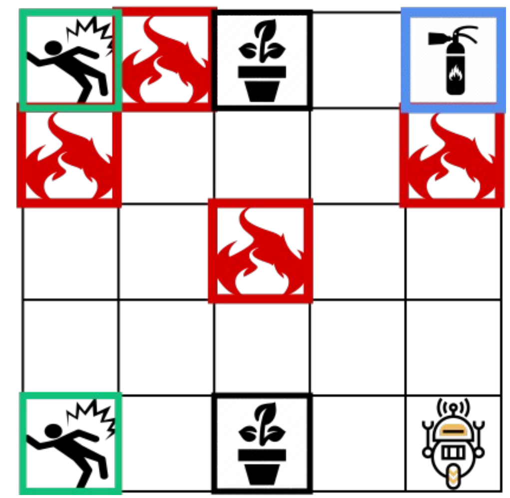

- We may not have the best track record so far…

# Learning Rewards from Multiple Sources

- Learning human-aligned reward functions: requires multiple objectives




$$r(s,a) = ?$$



- This will likely require the combination of different sources of feedback:

- Multi-Objective Reinforced Active Learning **combines demonstrations and preferences** to actively learn a reward for trading off objectives.


Peschl, M., Zgonnikov, A., Oliehoek, F., Siebert, L. C. (2021) MORAL: Aligning AI with Human Norms through Multi-Objective Reinforced Active Learning.

# Learning from data
# & how to generalize?

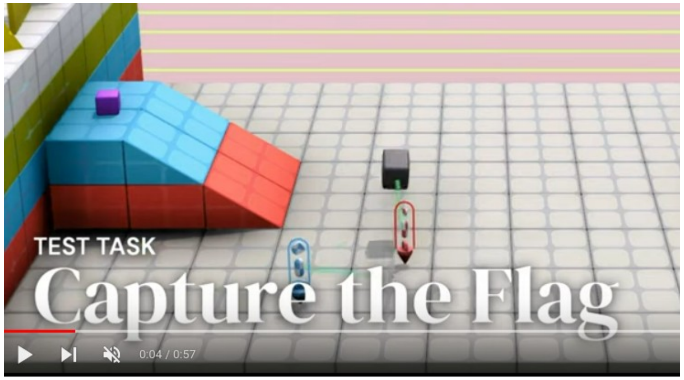
# Generalization

- "Agent, please do the right thing...

  ...also if the environment
  looks slightly different"

OpenAI's CoinRun environment

- Try to reduce overfitting to irrelevant features
- Data augmentation
- Meta-learning: train on a set of tasks
  e.g., "generally capable agents"
- Statistical models, might be limited in learning "out of distribution"
  → ideas from causal inference [Bareinboim'20 ICML tutorial]



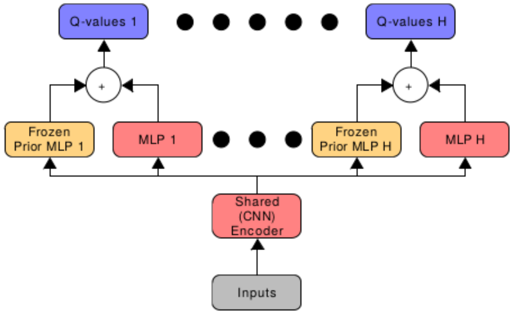
# Off-line RL: Learning from data sets

- When only learning offline from data...
  ...even more critical to not overfit:

  **actions that look good due to chance will not be corrected!**

PEBL: Pessimistic Ensembles for Offline Deep Reinforcement Learning

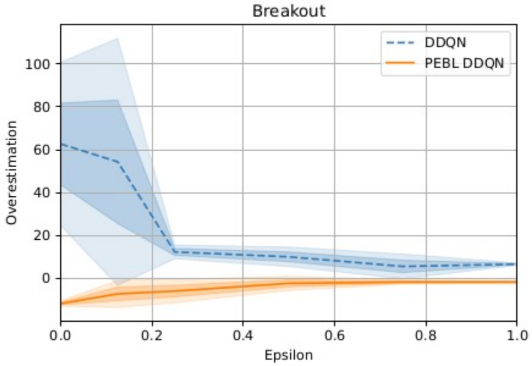
- ensemble of neural networks:
  represents uncertainty about Q-values




- penalize uncertainty:

  replace Q-values with a
  one standard deviation lower bound

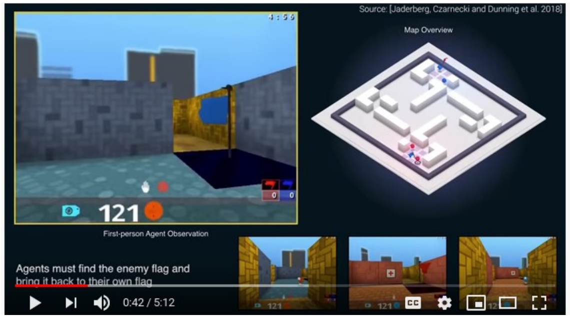
- Leads to lower over-estimation, also
  with less diverse training set

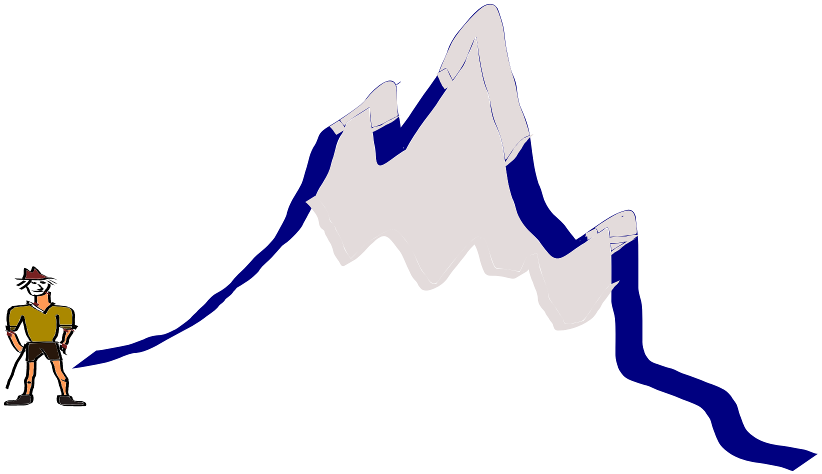



Smit, J., Ponnambalam, C.T., Spaan, M.T.J., Oliehoek F.A. (2021). Robust and Reliable Autonomy in the Wild Workshop (IJCAI).

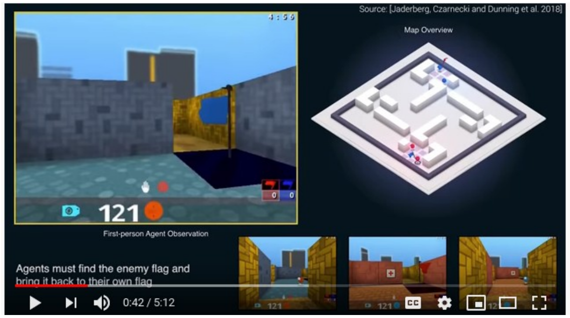
# Conclusions

- RL can do some cool things
- Foundations & state of the art
- Challenges are big
  - sample complexity
  - learning models
  - partial observability
  - scaling & the need for abstraction
  - multiagent systems
  - generalization

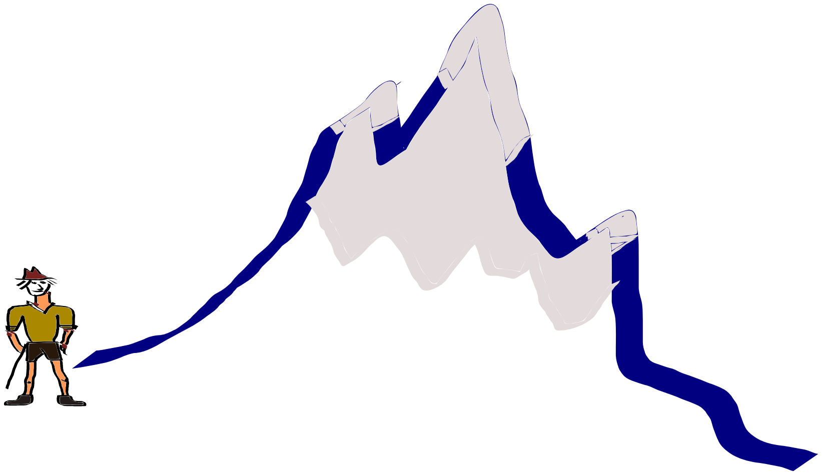


# Conclusions




- RL can do some cool things
- Foundations & state of the art
- Challenges are big
  - sample complexity
  - learning models
  - partial observability
  - scaling & the need for abstraction
  - multiagent systems
  - generalization
- ...but so might the future benefits be:

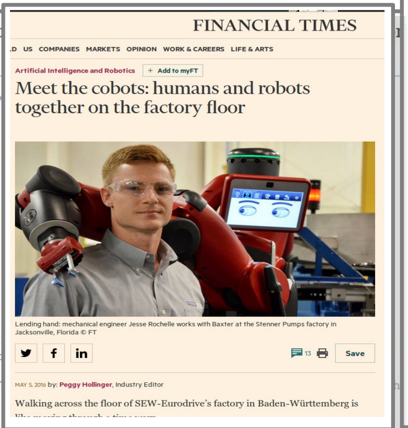


Brussels traffic jams, the biggest cause of ai
Euros

POSTED ON JULY 23, 2015 CATEGORIES: ECONOMIE/ECONOMY, UNCATEGORIZED

Every year, cars are stuck for 32 million hours in jams in the Brussels region.
carry a total cost of 511 million Euro, of which 70 million would be due to air po

Yearly 31 people die in traffic accidents, but 632 die early due to air pollution.
guidelines of the WHO would be followed.



FINANCIAL TIMES

Artificial Intelligence and Robotics

Meet the cobots: humans and robots together on the factory floor

Lending hand: mechanical engineer Jesse Rochelle works with Baxter at the Stenner Pumps factory in Jacksonville, Florida © FT

MAY 5, 2016 by: Peggy Hollinger, Industry Editor

Walking across the floor of SEW-Eurodrive's factory in Baden-Württemberg is

ARTIFICIAL INTELLIGENCE IS NOW TELLING DOCTORS HOW TO TREAT YOU

IMAGE: COURTESY OF MODERNIZING MEDICINE

LONG ISLAND DERMATOLOGIST Kavita Mariwalla knows how to treat acne, burns, and rashes. But when a patient came in

# Acknowledgments


- Students/collaborators:

  Elena Congeduti, Aleksander Czechowski, Alexander Mey, Jinke Hey, Rolf Starre, Miguel Suau, Canmanie Ponnambalam, Jordi Smit, Matthijs Spaan, Nele Albers, Sammie Katt, Christopher Amato, Shimon Whiteson, Elise Van der Pol, Elise van der Pol, Thomas Kipf, Max Welling Daniel Claes, Hendrik Baier,  Daniel Hennes, Karl Tuyls, Feryal Behbahani, Kyriacos Shiarlis, Xi Chen, Vitaly Kurin, Sudhanshu Kasewa, Ciprian Stirbu, João Gomes, Supratik Paul, João Messias, Timon Kanters, Michael Kaisers, Nikos Vlassis, Frans Groen

- Funding agencies:

  NWO Netherlands Organisation for Scientific Research

  EPSRC Engineering and Physical Sciences Research Council

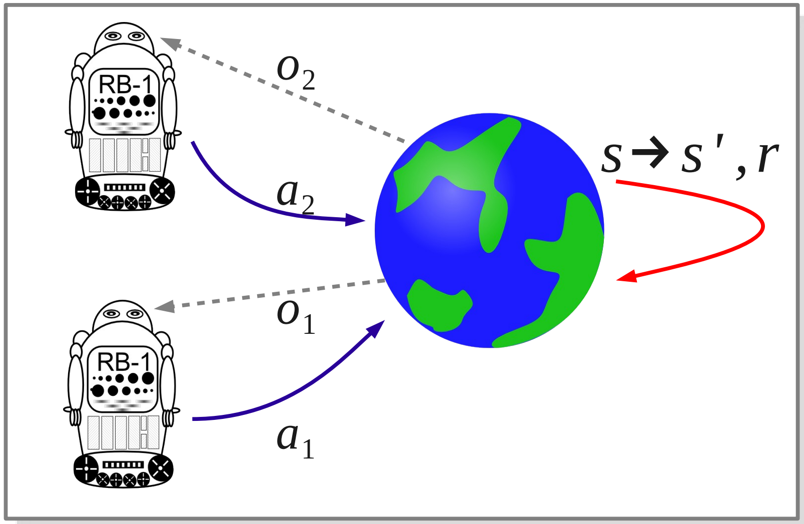
  erc European Research Council Established by the European Commission


- **Questions?**

# Backup Slides

# Decentralized POMDPs

- A minimal framework for
  - multiple cooperative agents
  - stochastic environments
  - state uncertainty




- A Dec-POMDP $\langle S, A, P_T, O, P_O, R \rangle$
  - $n$ agents
  - $S$ – set of states
  - $A$ – set of **joint** actions $\quad a = \langle a_1, a_2, \ldots, a_n \rangle$
  - $P_T$ – transition function $\quad P(s'|s,a)$
  - $O$ – set of **joint** observations $\quad o = \langle o_1, o_2, \ldots, o_n \rangle$
  - $P_O$ – observation function $\quad P(o|a,s')$
  - $R$ – reward function $\quad R(s,a)$

- Act based on **individual** observations

# What does it buy us?

- Optimal plans need to trade-off:
  - immediate vs long-term reward (as in MDPs)
  - knowledge gathering vs exploitation (as in POMDPs and/or RL)
  - exploiting individual knowledge vs being predictable

- Using Dec-POMDPs (and similar models) we can study quantitatively and qualitatively the effect of interaction.

# Decentralized POMDPs

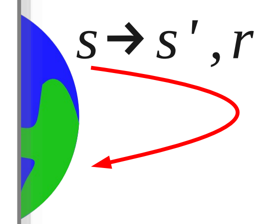
Yes, these are horribly complex to solve optimally...

- NEXP-complete [Bernstein et al. 2000]
- but no easy way out - this is a minimal model.

...but we are making steady progress

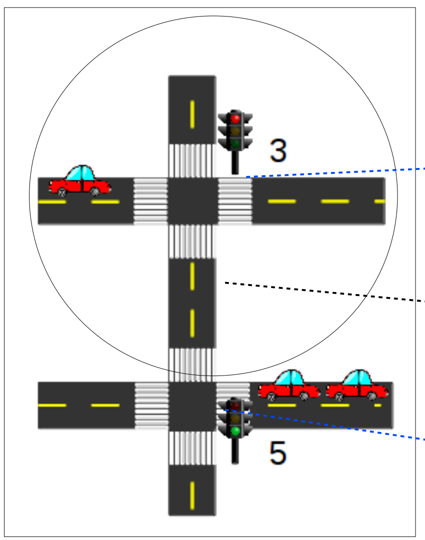
- E.g., multi-robot systems - Christopher Amato et al.

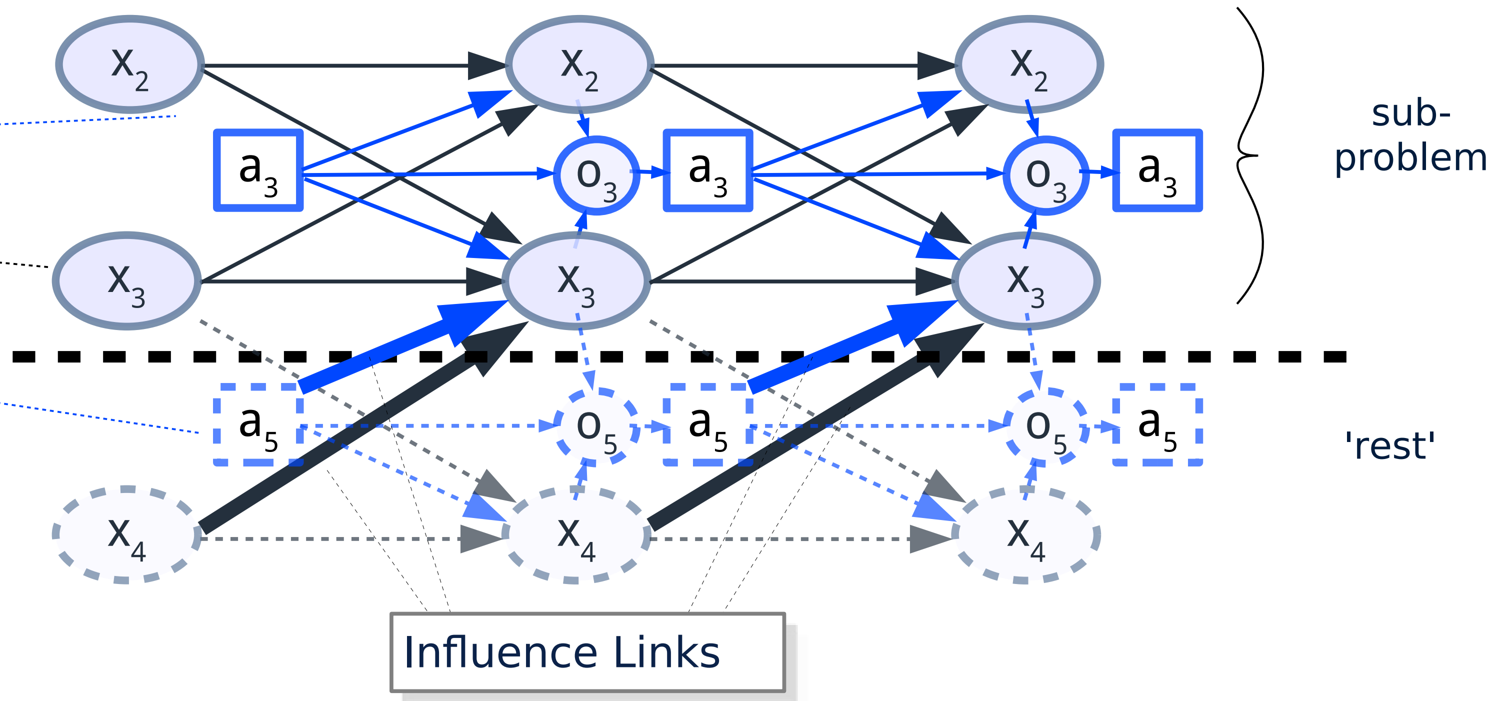

- PR2 + 2 turtlebots for faster drinks delivery!



# A Definition of Influence

- Restrict attention to part of state space

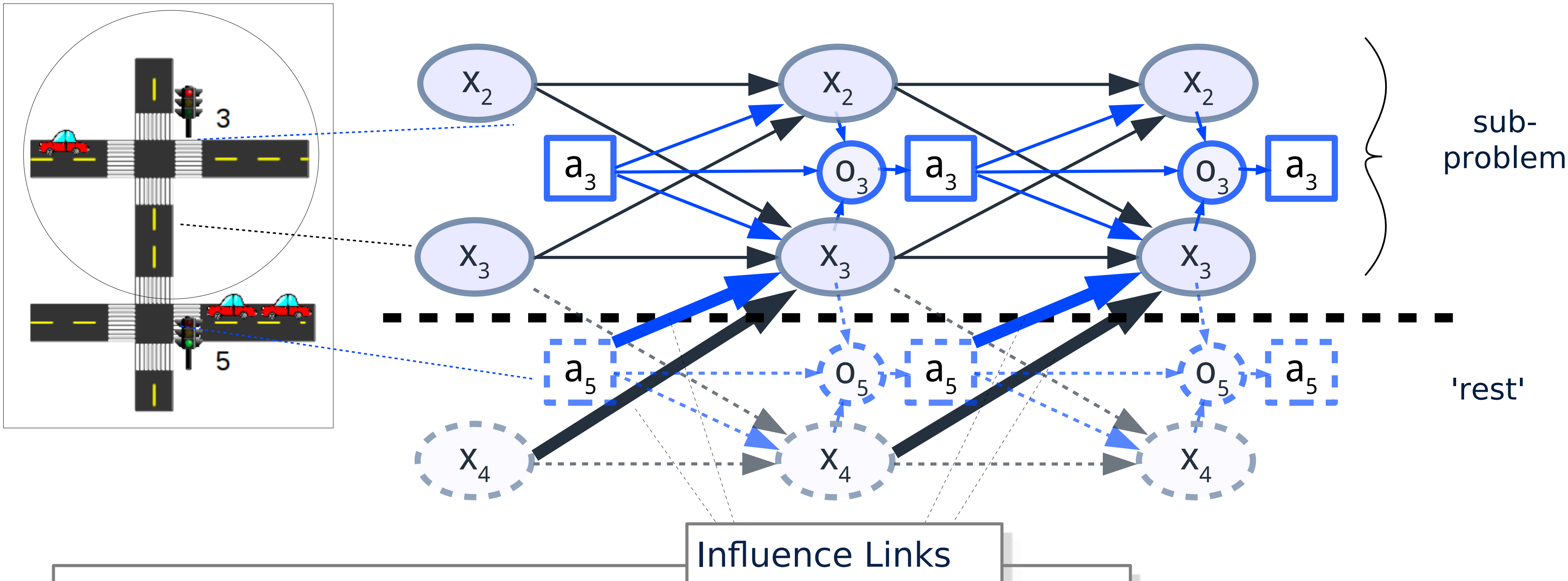



Goal:
predict ‘influence sources’

# A Definition of Influence

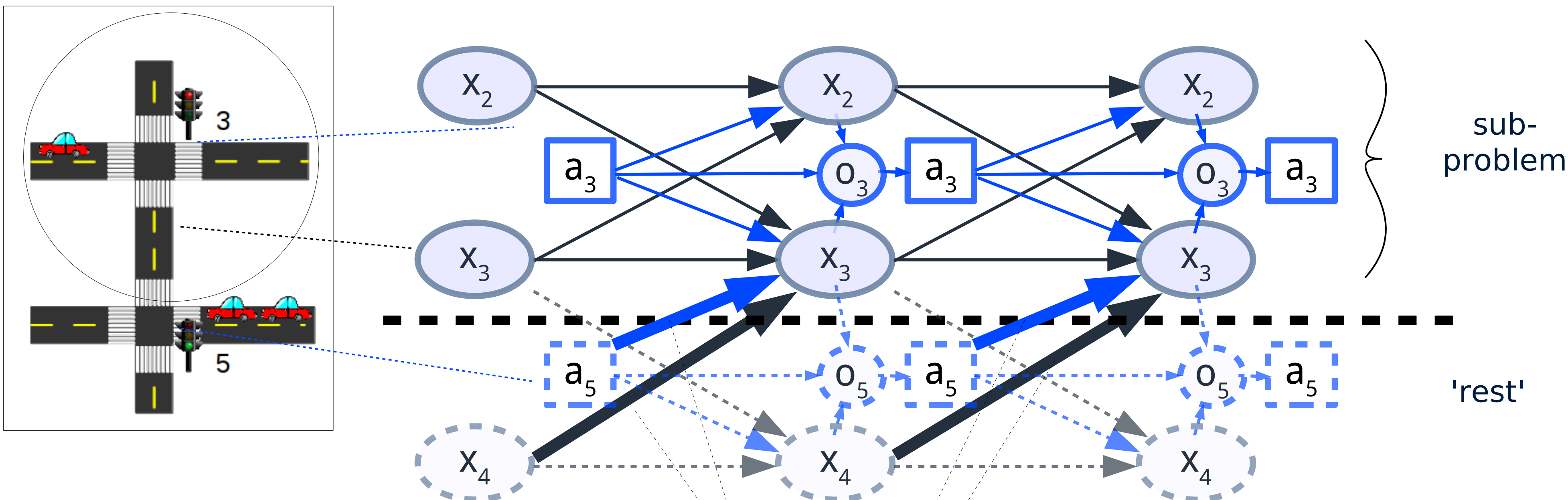
- Restrict attention to part of state space




- If we new the values of **influence sources** in advance...
- Of course we don't, but we can:
  - compute a distribution over them
  - need to condition on some stuff, D, in the local problem
- So, an **influence point** is a collection { $P(x_{sources} \mid D)$ }

# A Definition of Influence

- Restrict attention to part of state space




Influence Links

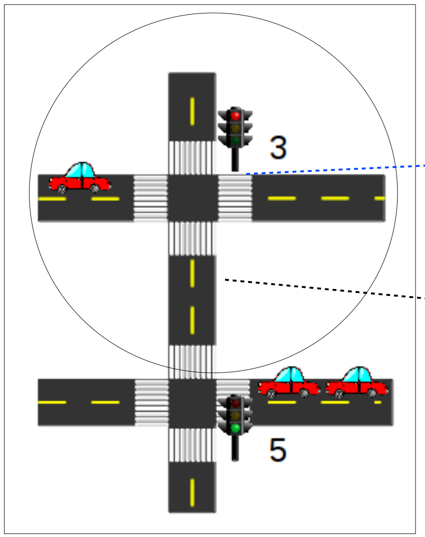
D-separating set

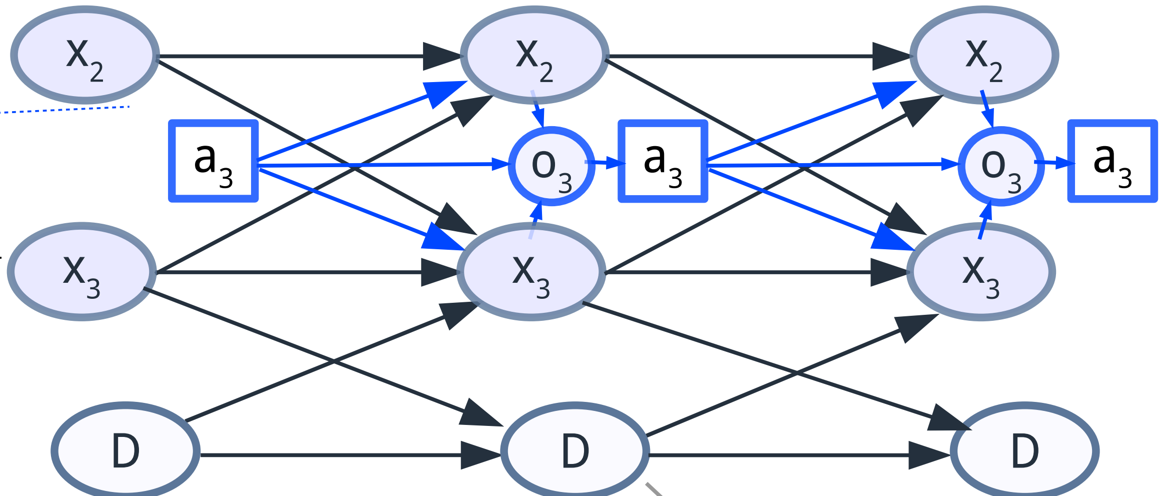
An inference task

- If we new the values of **influence sources** in advance...
- Of course we don't, but we can:
  - compute a distribution over them
  - need to condition on some stuff, D, in the local problem
- So, an **influence point** is a collection { $P(x_{sources} \mid D)$ }

# A Definition of Influence

- Enables **lossless** abstraction:




Catch **intractable**:

- 'D' might be huge...
- but we know many classes for which it is compact

# The Procedure of IBA

- Define the **local model**
  - reward-relevant and observation-relevant variables must be included

- Determine a d-separating set

- Construct the **influence-augmented local model (IALM)**
  - With states <s,D>
  - Compute influence point: $\{ P(x_{sources} \mid D) \}$
  - Compute transitions $P(\langle s',D'\rangle \mid \langle s,D\rangle, a)$

# An Example: Planetary Rover

- Satellite (agent 1) can send a plan to rover (agent 2)

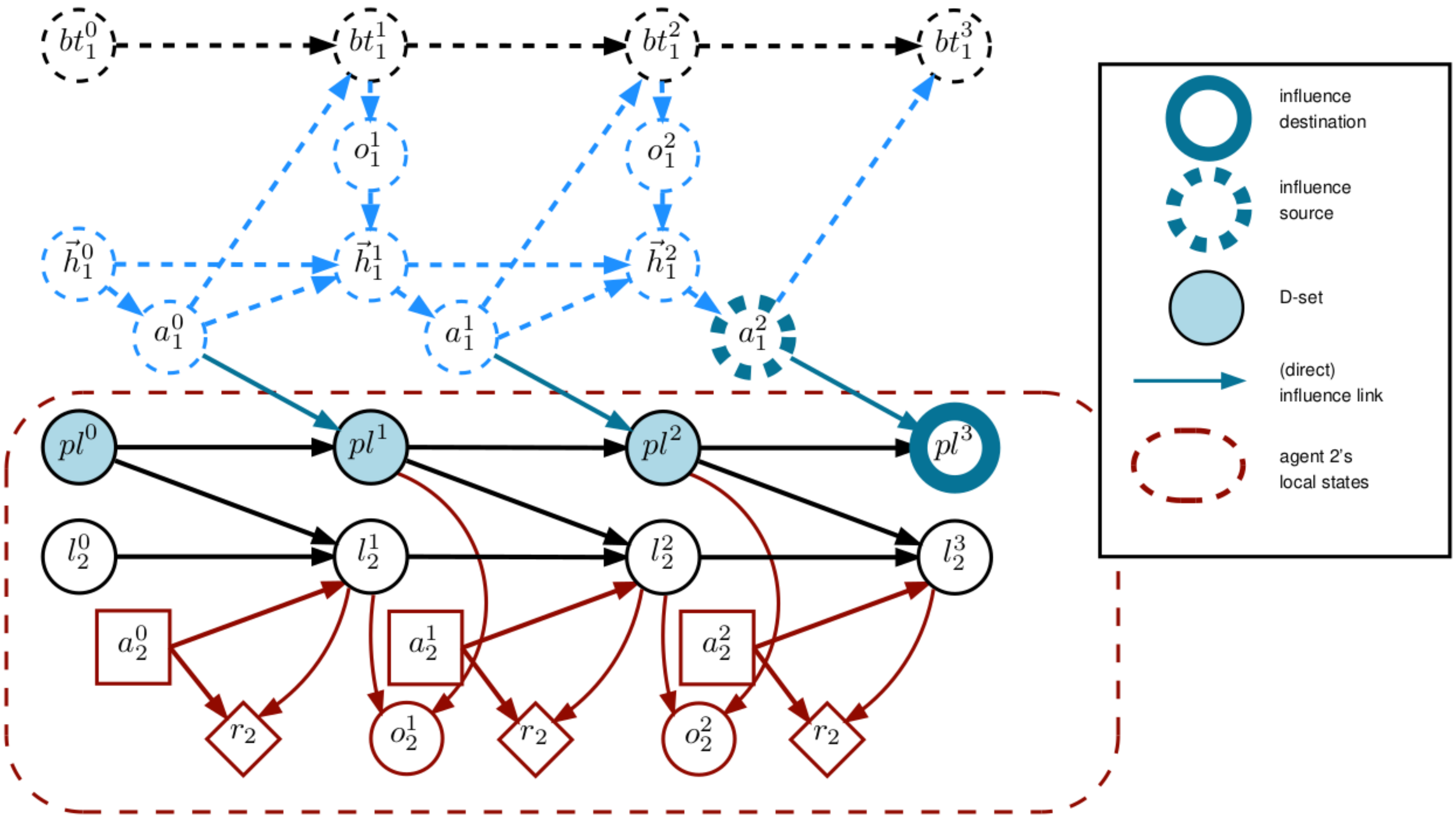



Figure 7: Illustration of the influence experienced by the mars rover (agent $i = 2$) at stage $t = 3$ in the PLANETARY EXPLORATION domain. If the satellite (agent 1) computes and transmits a plan ($pl$), the rover can more effectively navigate from that point onward.

# An Example: Planetary Rover

- Satellite (agent 1) can send a plan to rover (agent 2)

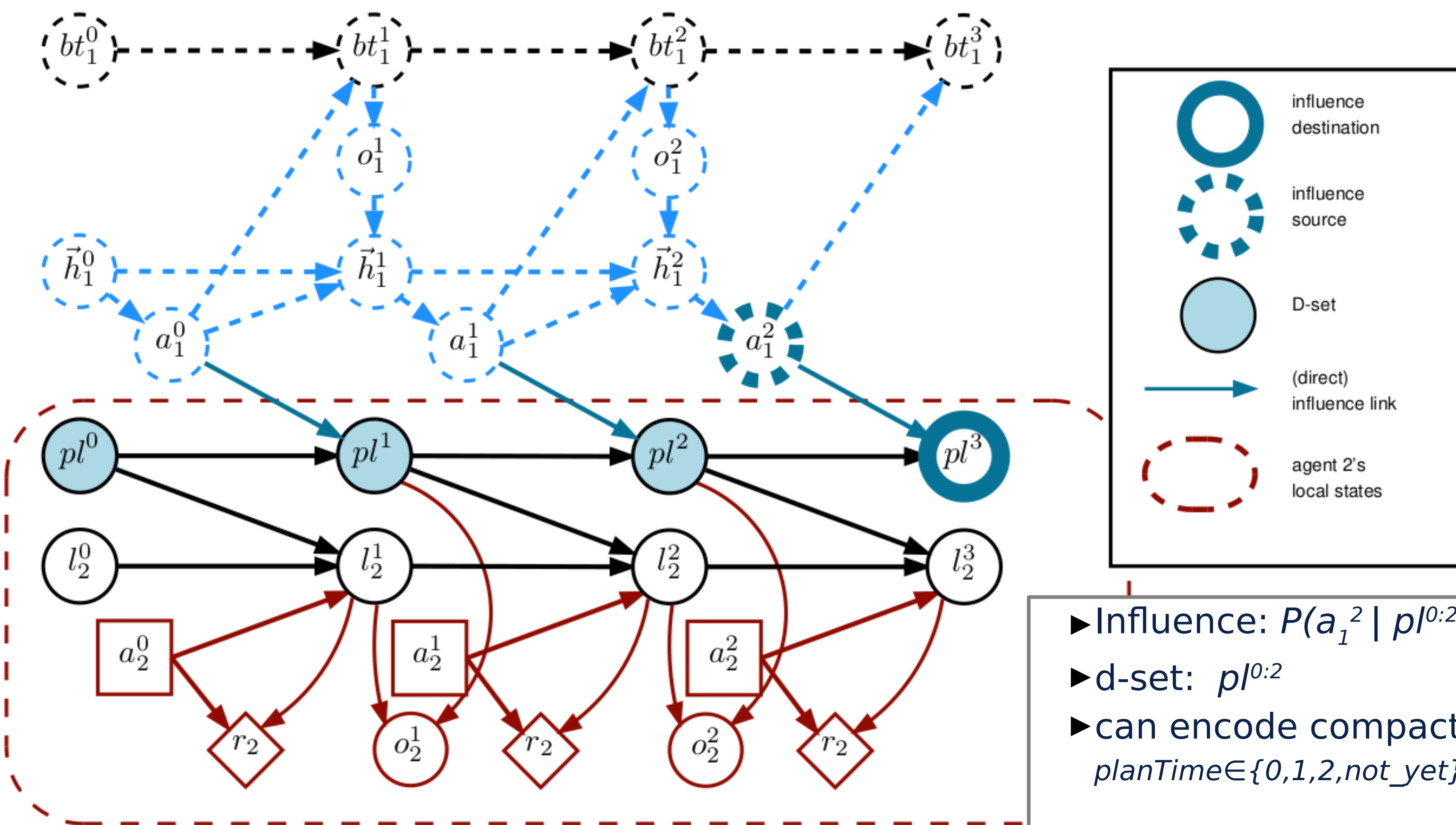
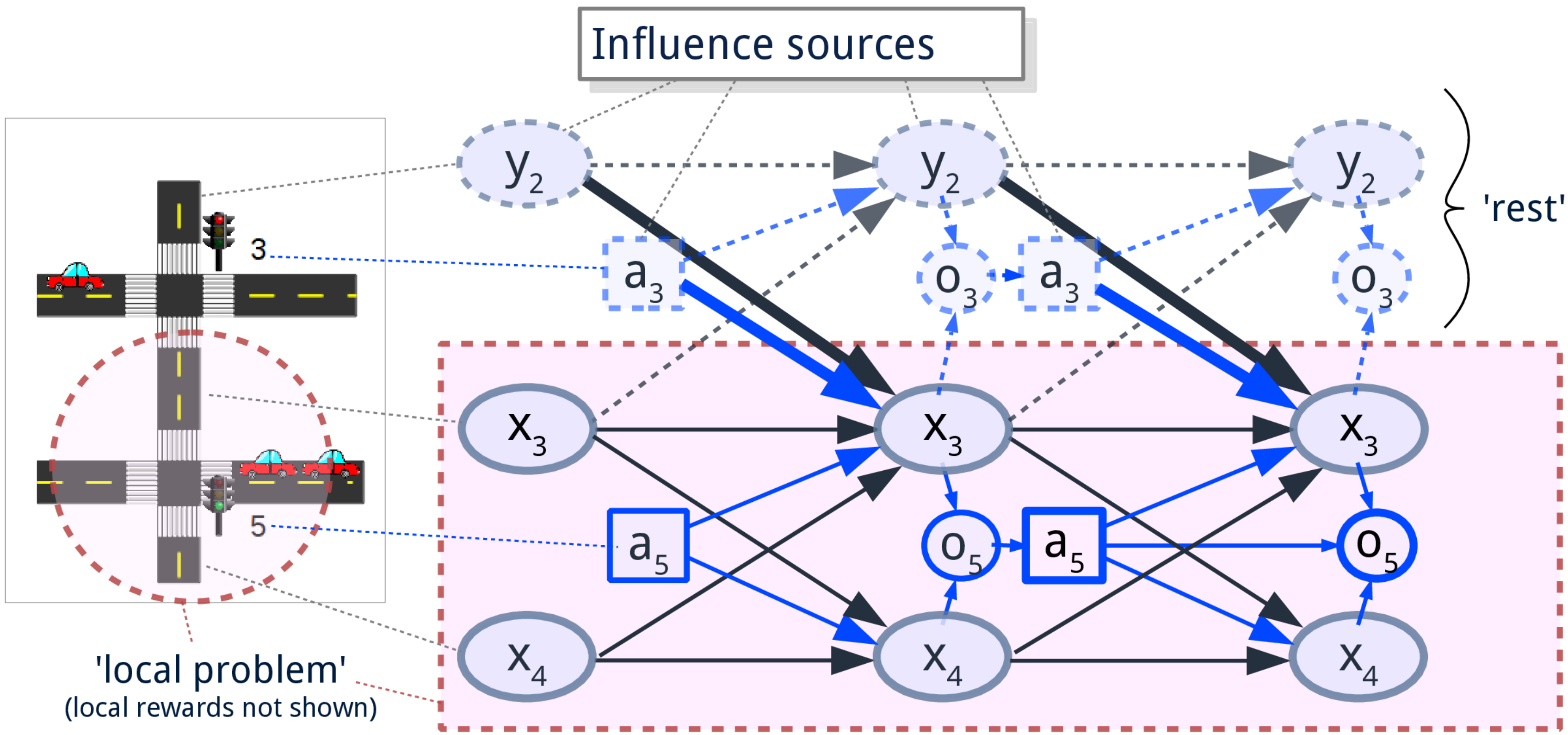



Figure 7: Illustration of the influence experienced by the mars rover (a
in the PLANETARY EXPLORATION domain. If the satellite (agent 1) com
(*pl*), the rover can more effectively navigate from that point onward.

- Influence: $P(a_1^2 \mid pl^{0:2})$
- d-set: $pl^{0:2}$
- can encode compactly: *planTime∈{0,1,2,not_yet}*
- IALM states: $<l_2, planTime>$

# Approximate Influences

# Exact Influence Points




If you can compute them, exact influence points are great!

- ...but in general intractable.

# Exact Influence Points (EIPs)

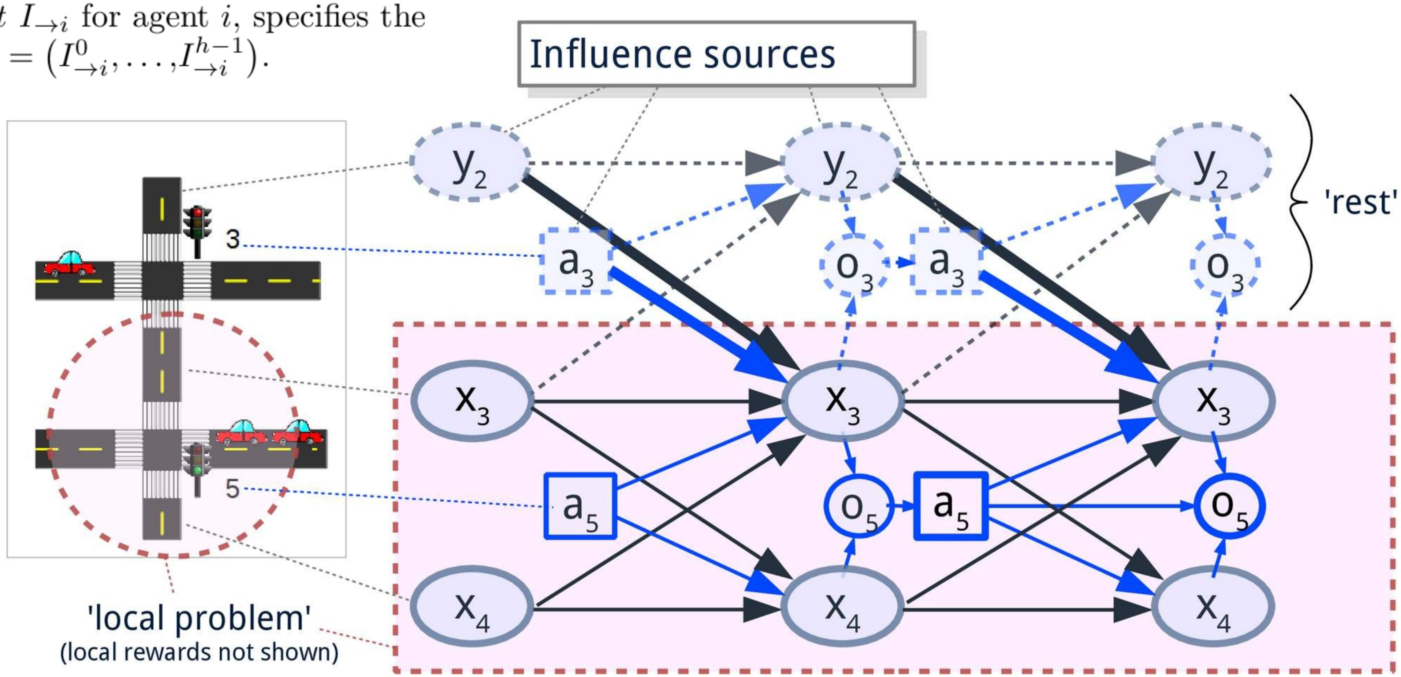
- An exact representation of influence *exist*: **Exact Influence Point (EIP)** [Oliehoek et al. 2012]

**Definition.** The *incoming influence* for agent $i$, for a stage $t$ is the conditional probability distribution of the influence sources:

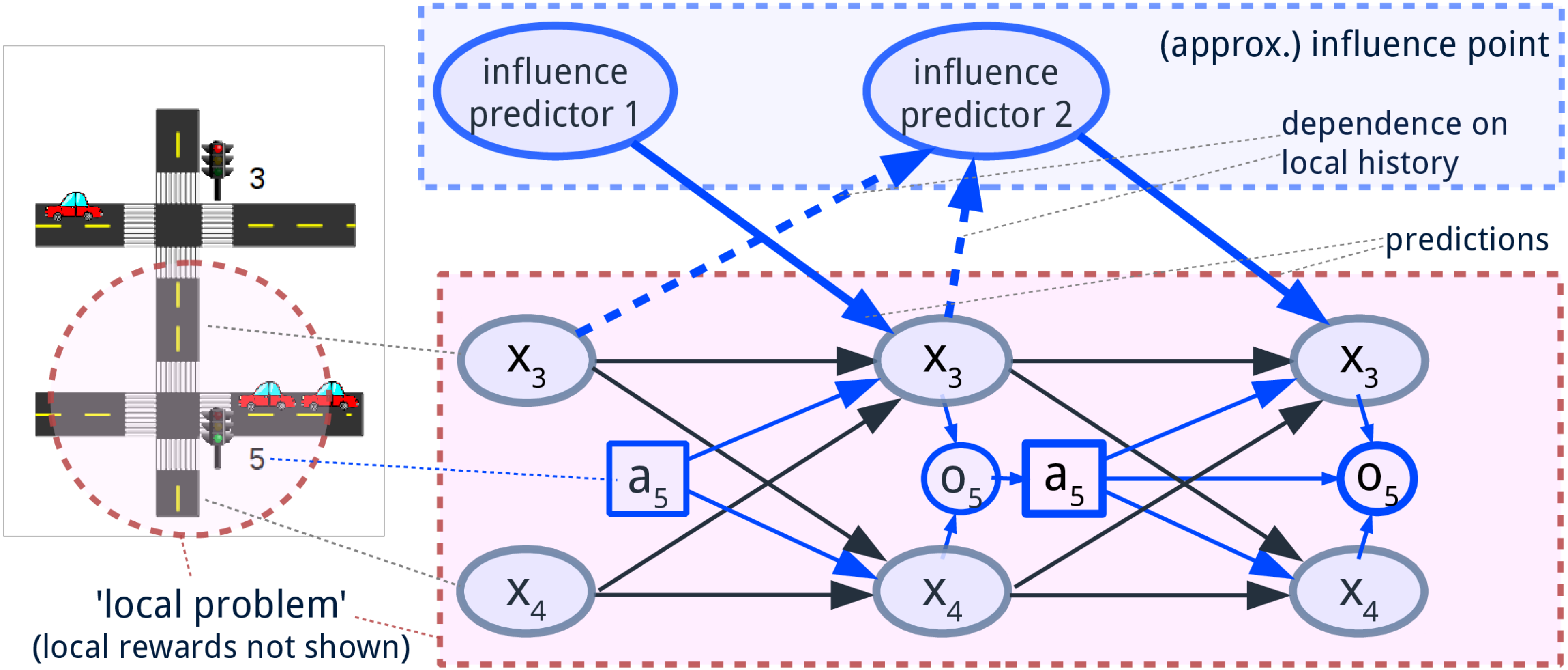
$$I^t_{\rightarrow i} = \Pr(u^t_{\rightarrow i} | D^t_i),$$

given enough local history $D^t_i$ to d-separate $u^t_{\rightarrow i}$ from the (other) local states and observations.

**Definition.** An *exact influence point* $I_{\rightarrow i}$ for agent $i$, specifies the incoming influences for all stages $I_{\rightarrow i} = (I^0_{\rightarrow i}, \ldots, I^{h-1}_{\rightarrow i})$.



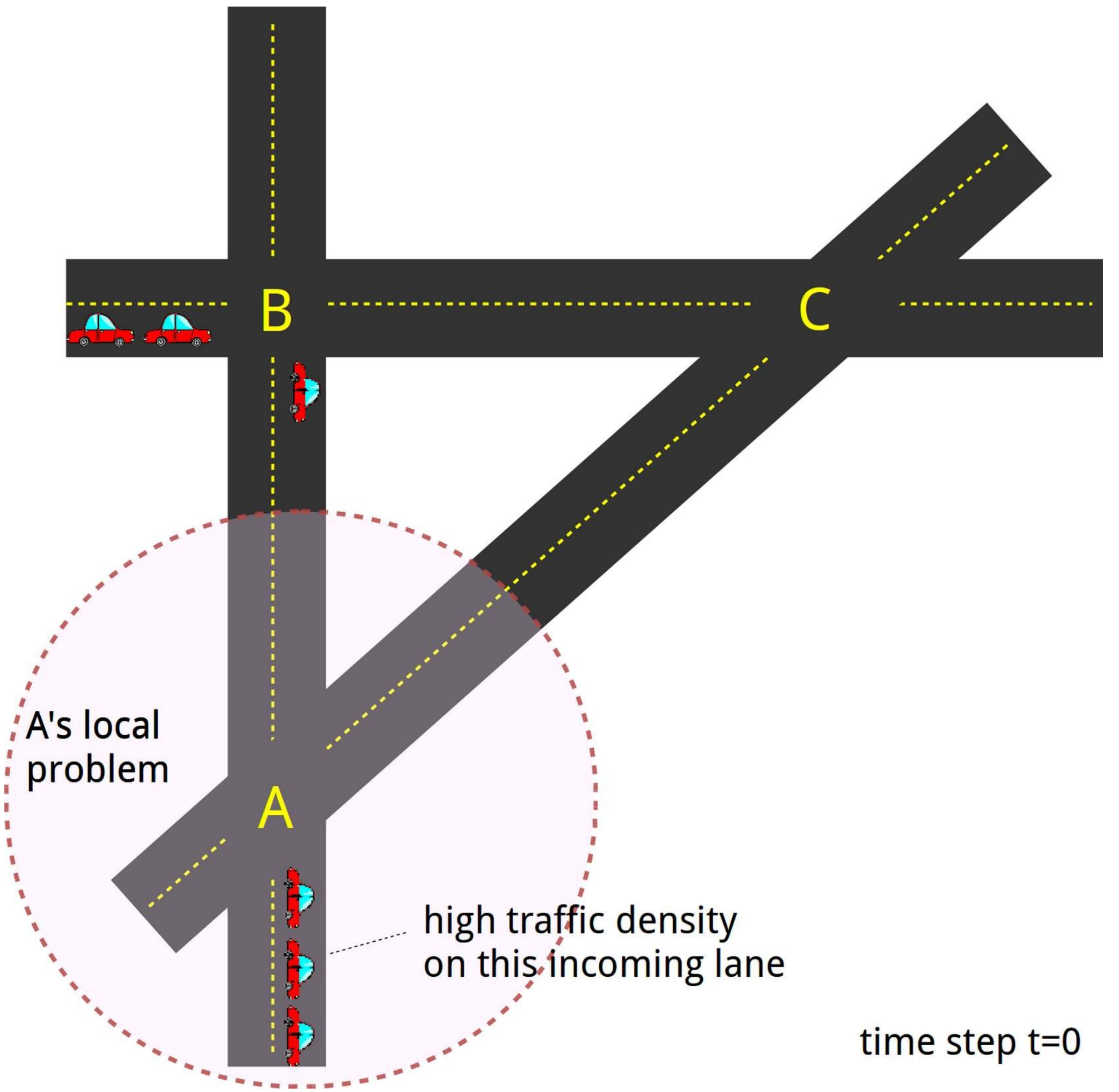
# Approximate Influence Points (AIPs)



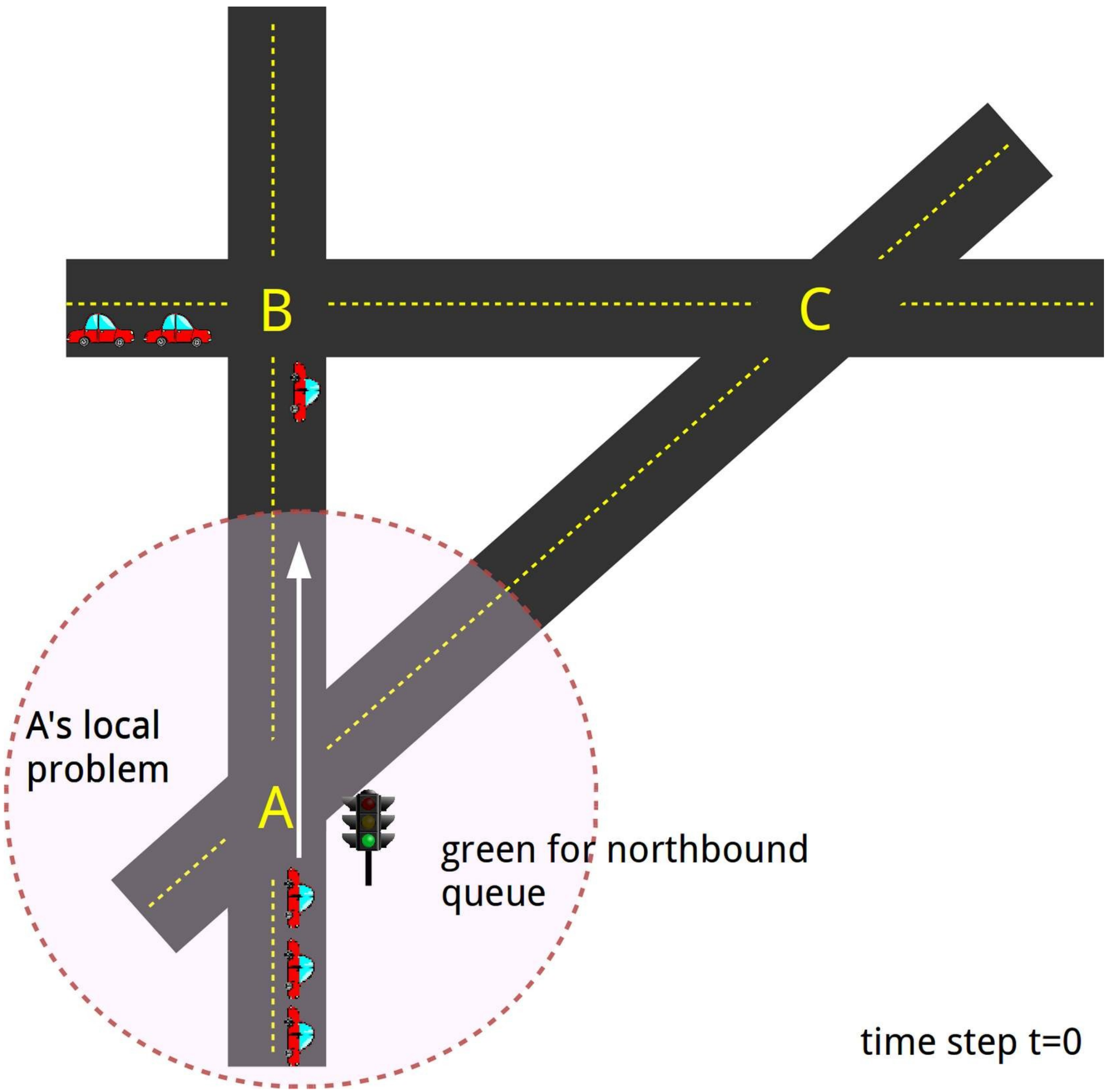

May need to resort to **approximate** influence points (AIPs) to predict $P(x_{sources}|D)$

- ►Form of sequence prediction: supervised learning.
- ►E.g., build on deep learning

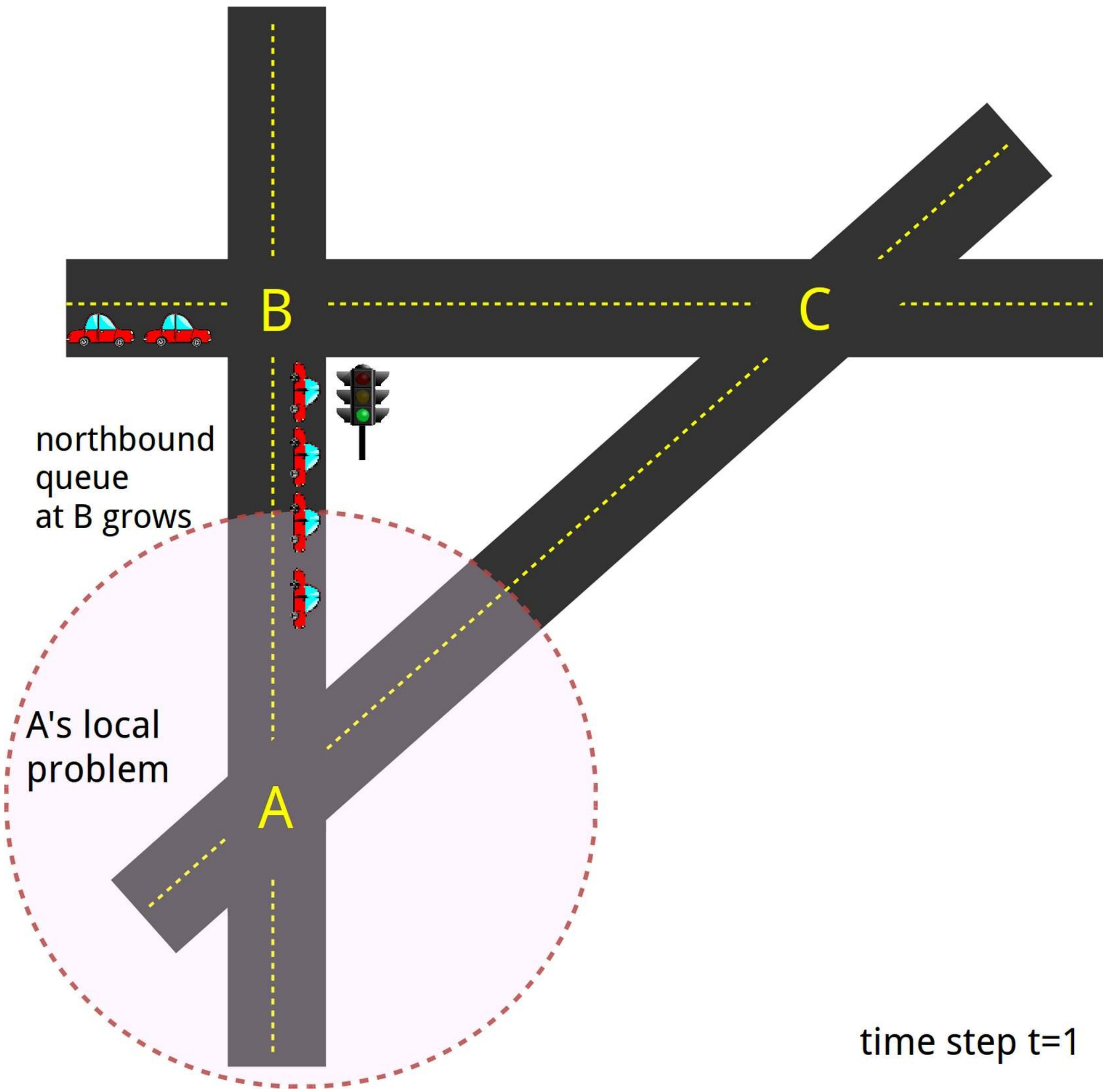
# An Example


B
C
A
A's local problem
high traffic density on this incoming lane
time step t=0

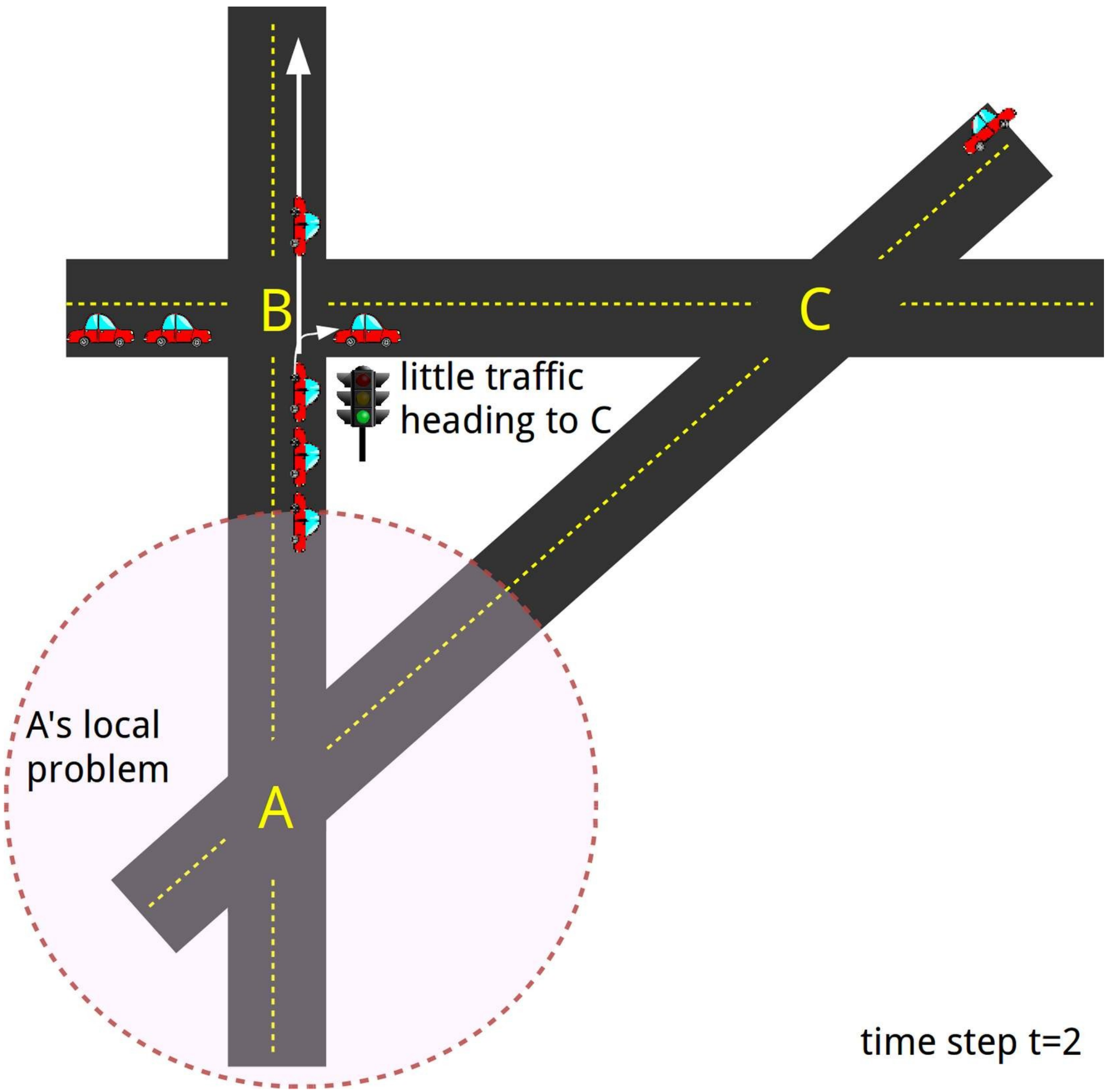
# An Example


B
C
A's local
problem
A
green for northbound
queue
time step t=0

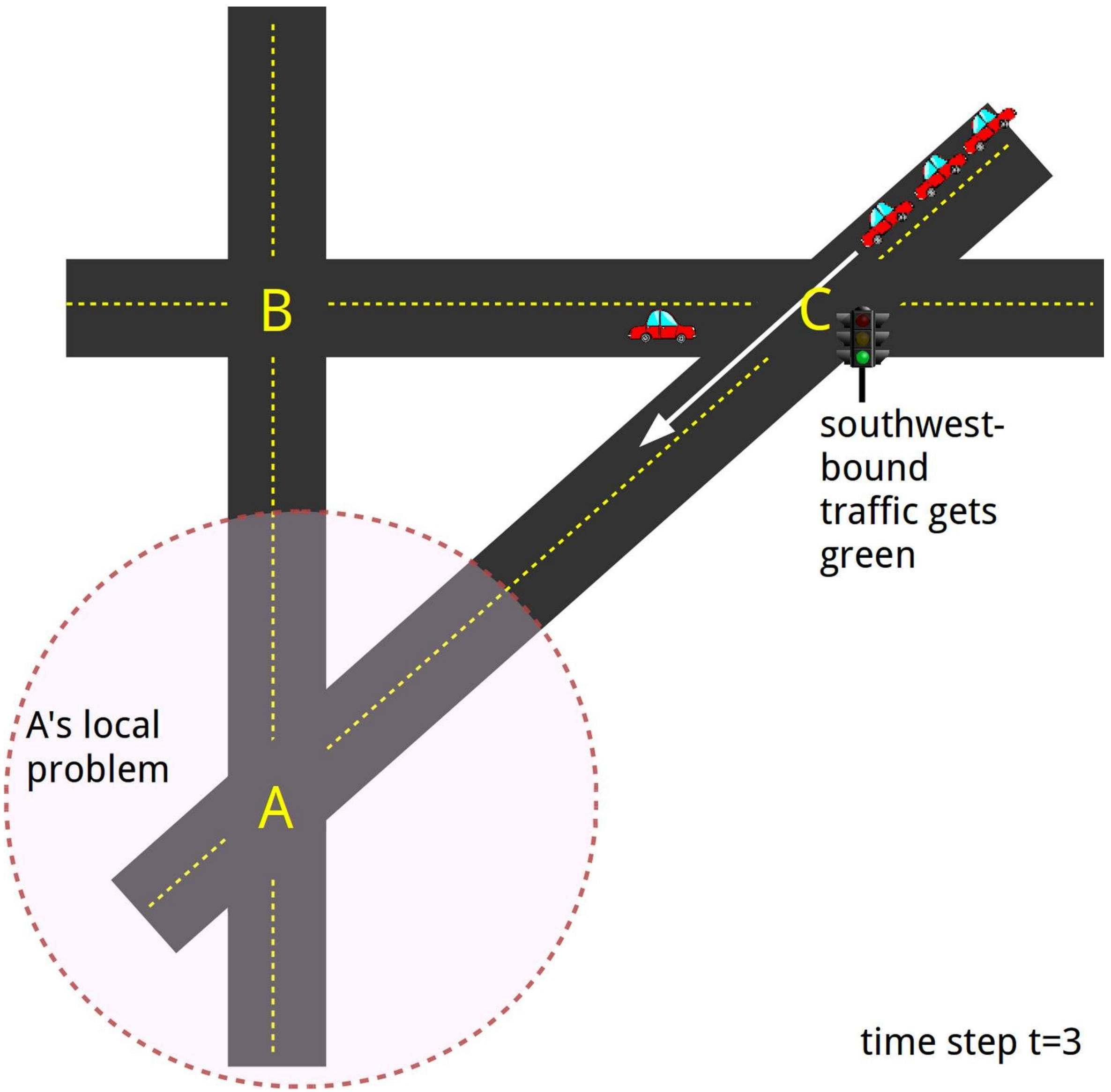
# An Example


B
C
northbound
queue
at B grows
A's local
problem
A
time step t=1

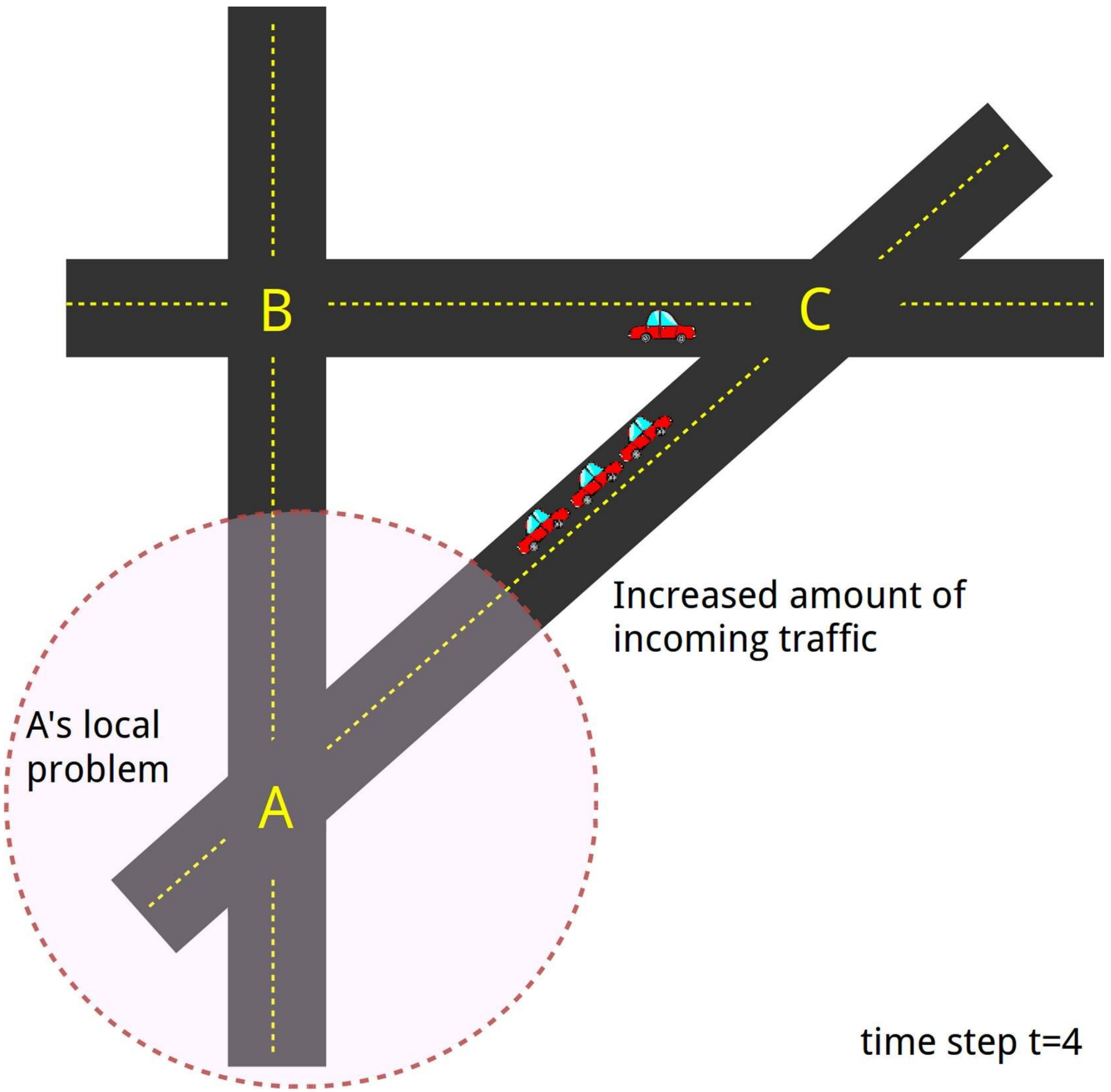
# An Example


B
C
little traffic
heading to C
A's local
problem
A
time step t=2

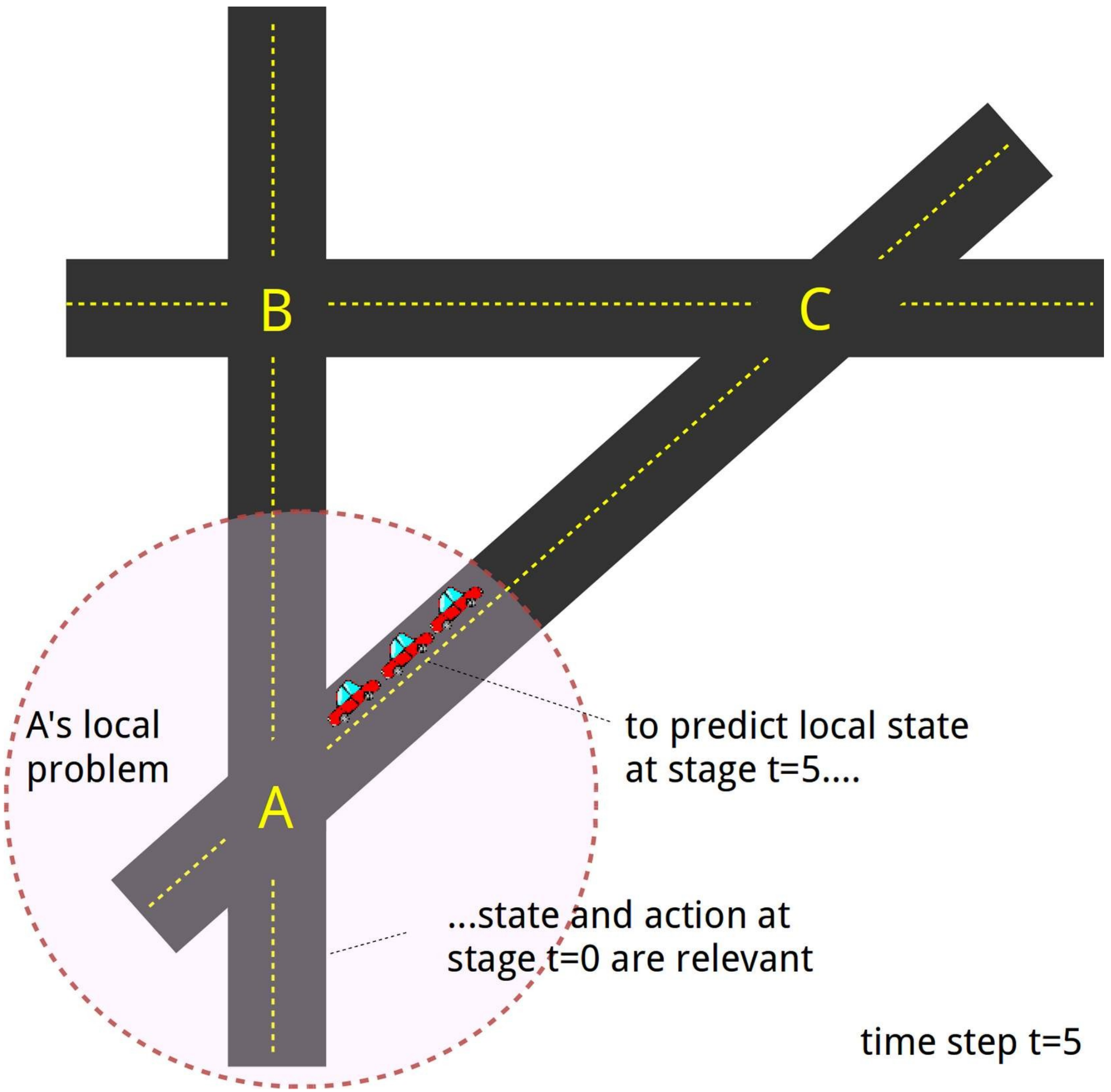
# An Example


B
C
southwest-bound traffic gets green
A's local problem
A
time step t=3

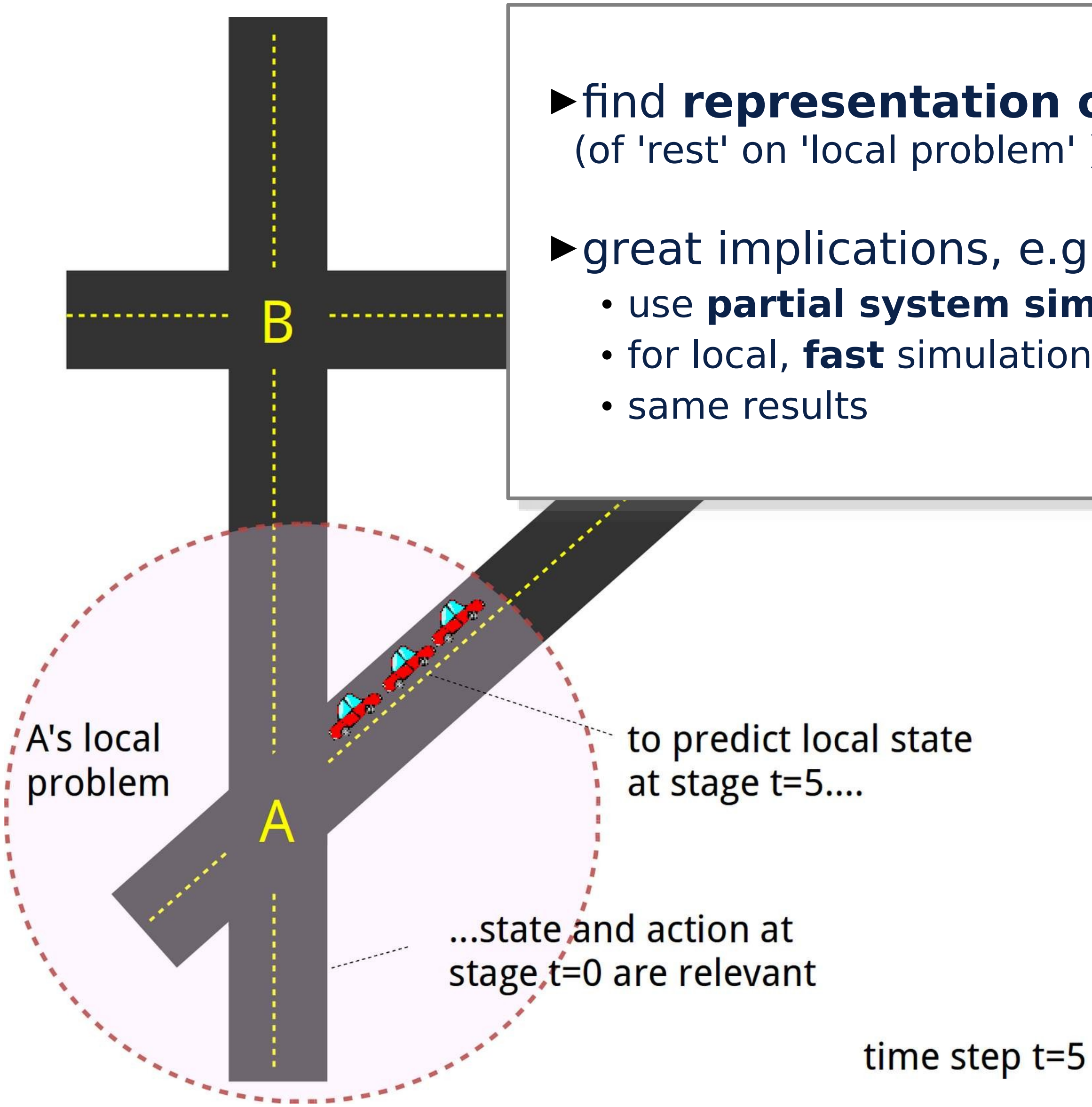
# An Example


B
C
Increased amount of incoming traffic
A's local problem
A
time step t=4

# An Example

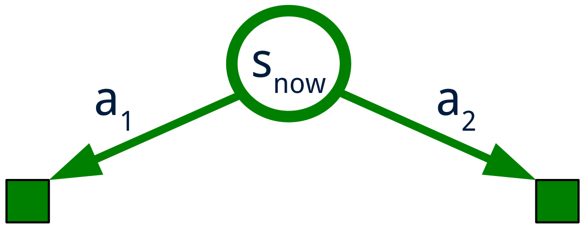

B
C
A
A's local
problem
to predict local state
at stage t=5....
...state and action at
stage t=0 are relevant
time step t=5

# An Example

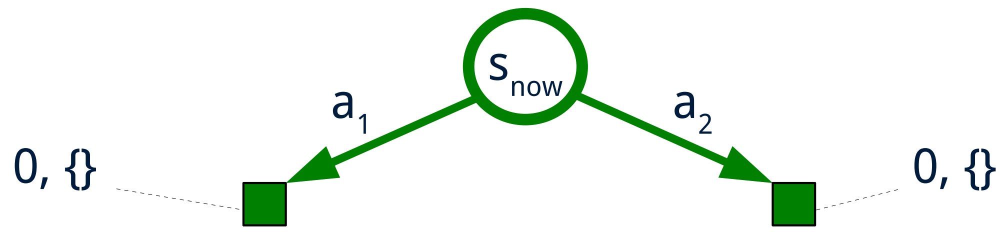



- find **representation of influence** (of 'rest' on 'local problem' )
- great implications, e.g.:
  - use **partial system simulator**
  - for local, **fast** simulation
  - same results

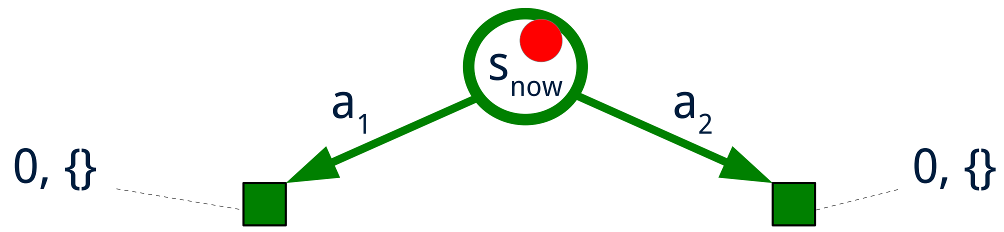
# MCTS – Example



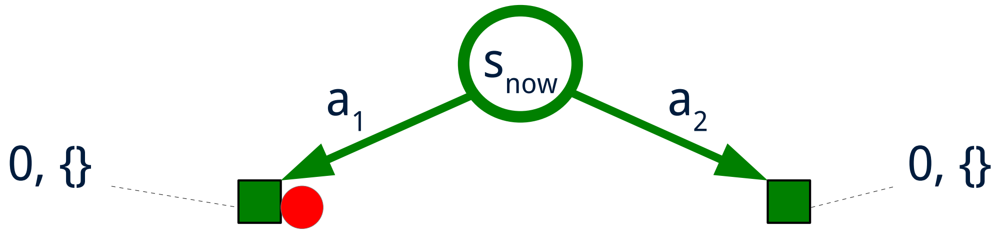
# MCTS – Example



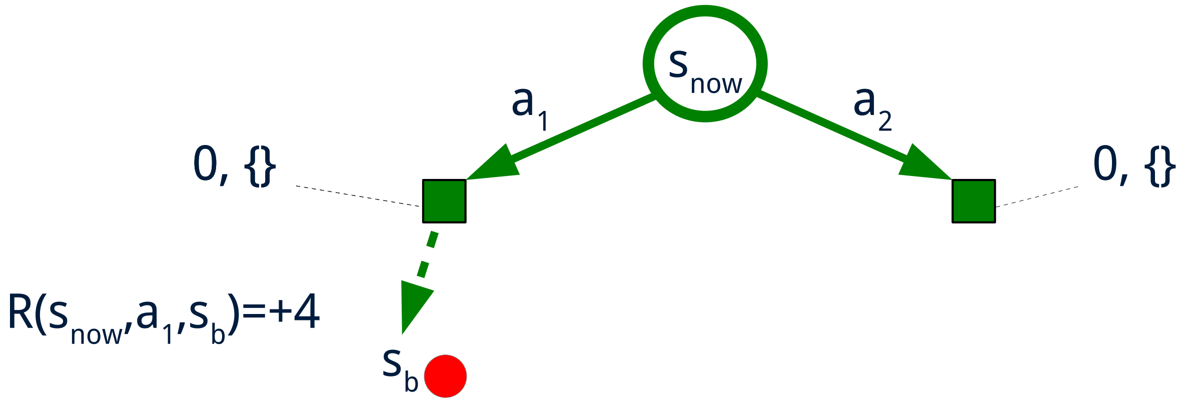
# MCTS – Example



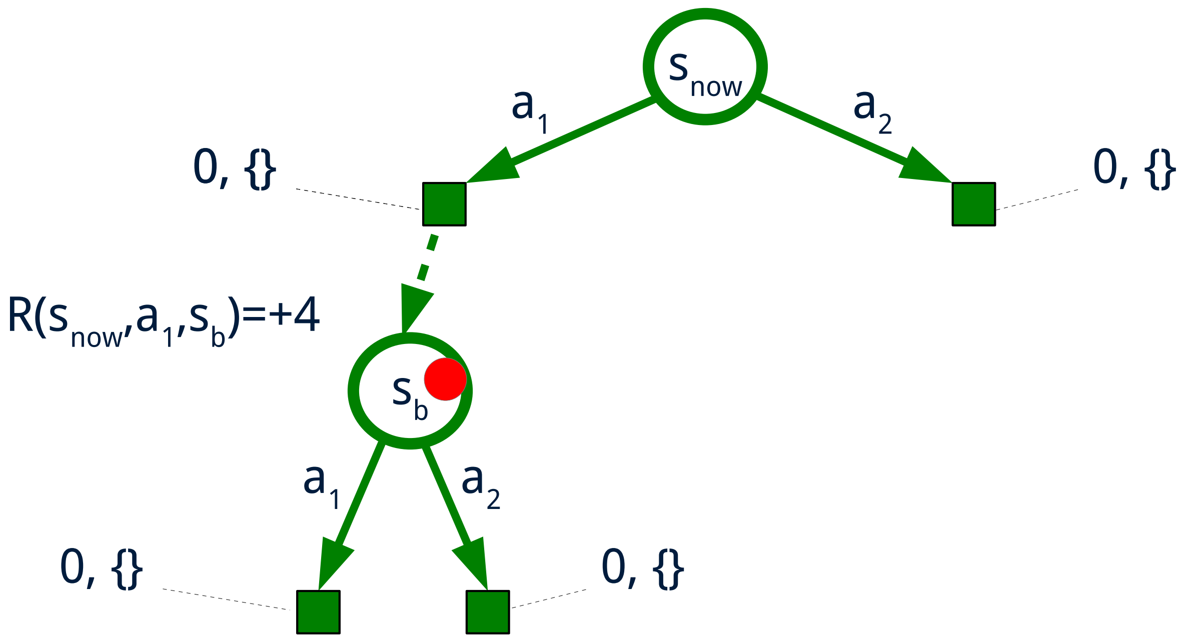
# MCTS – Example



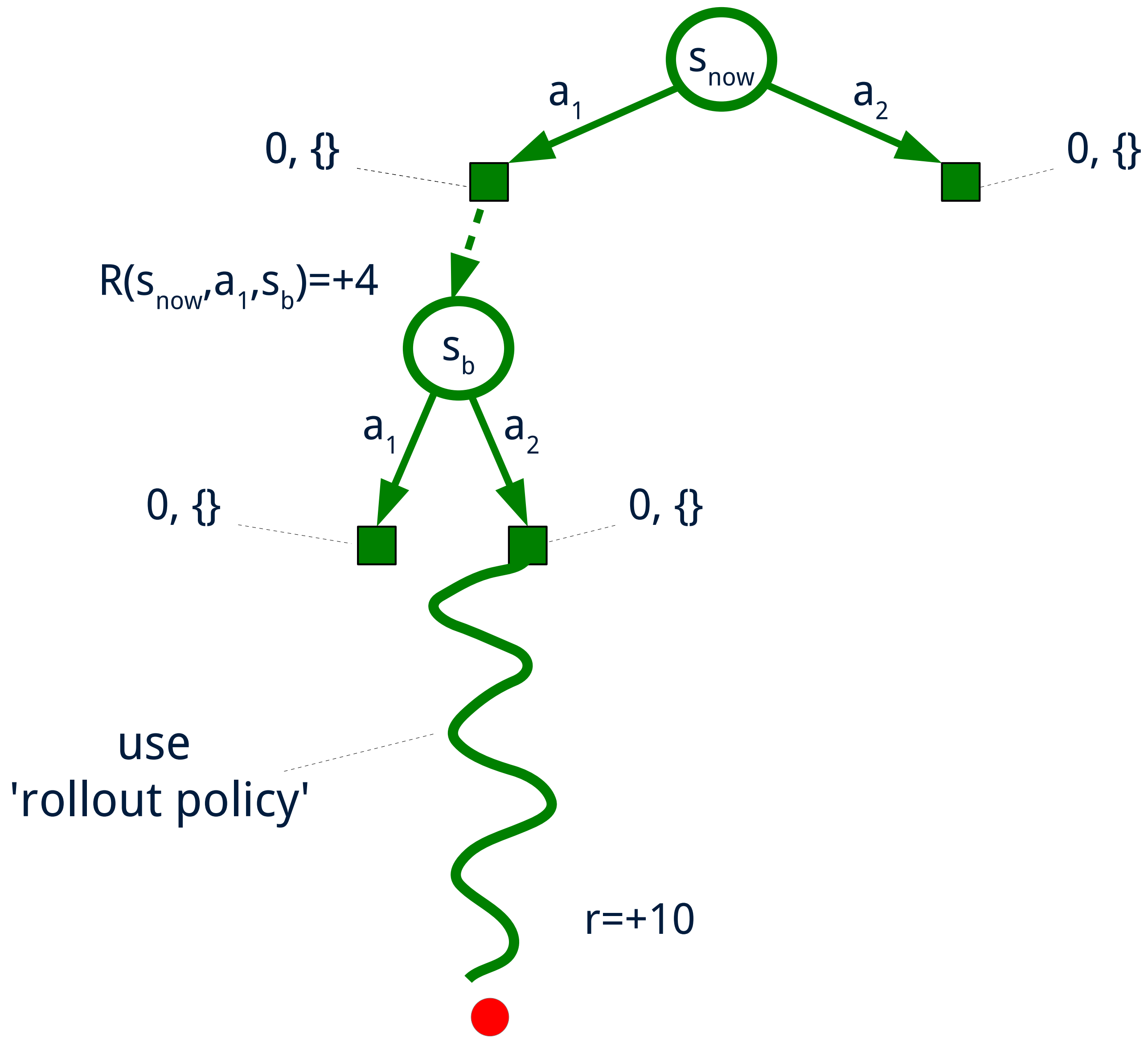
# MCTS – Example



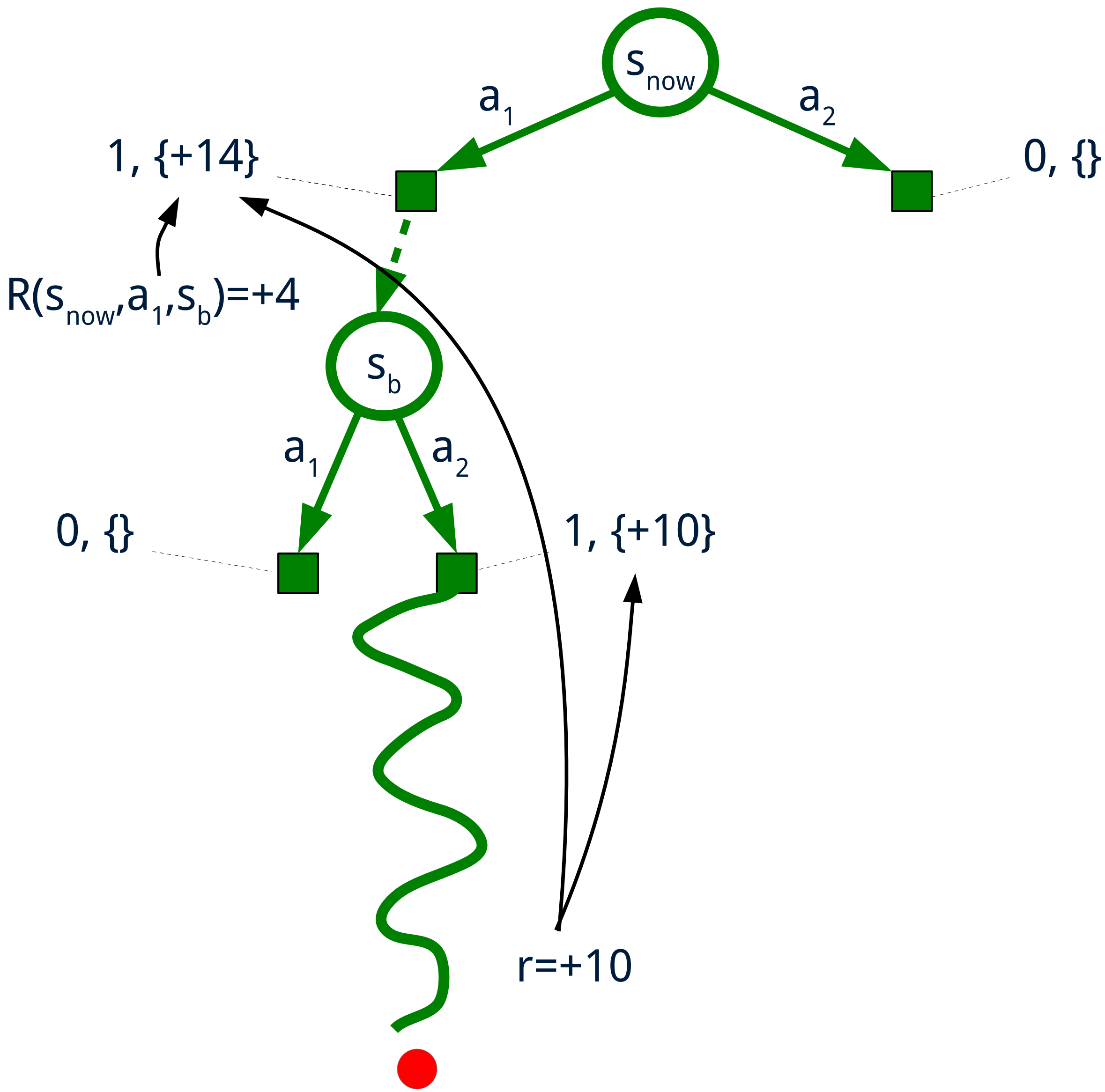
# MCTS – Example



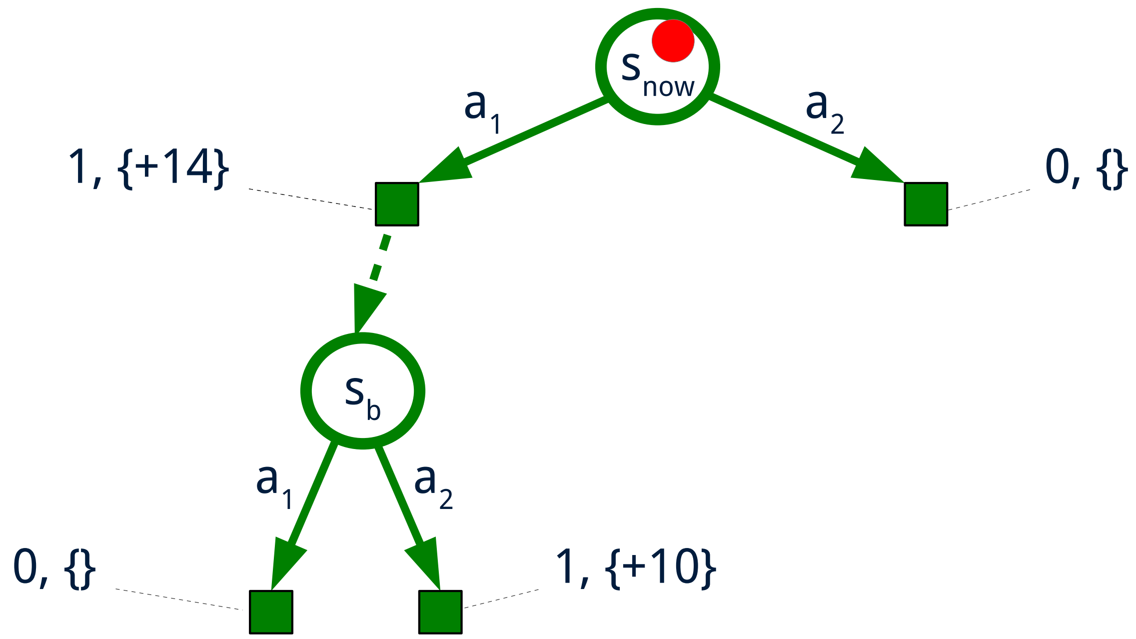
# MCTS – Example



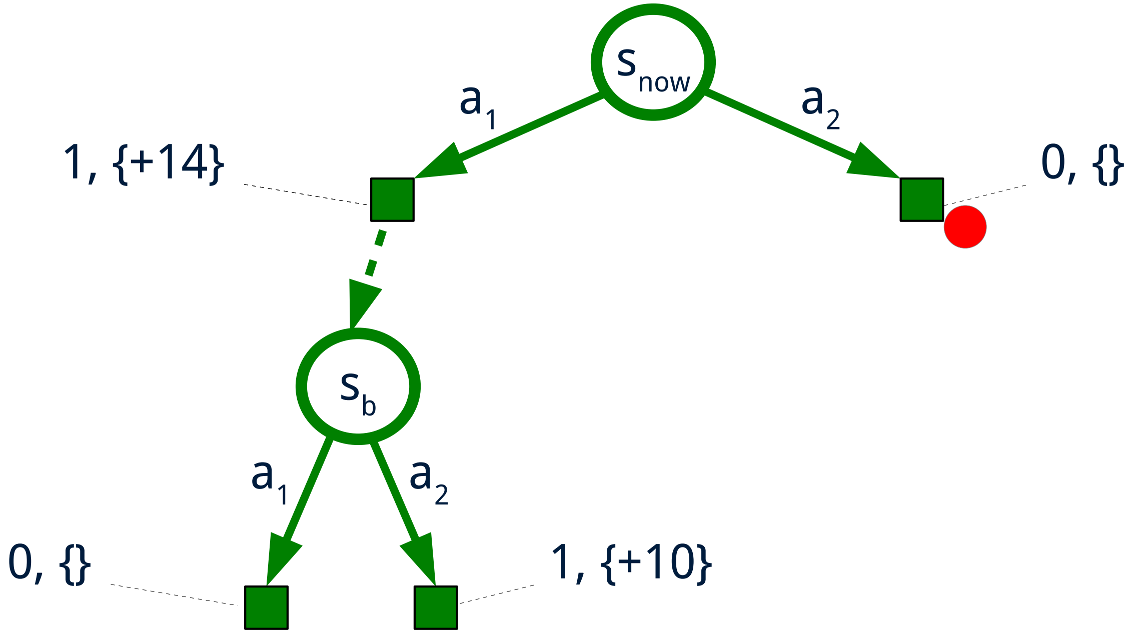
# MCTS – Example



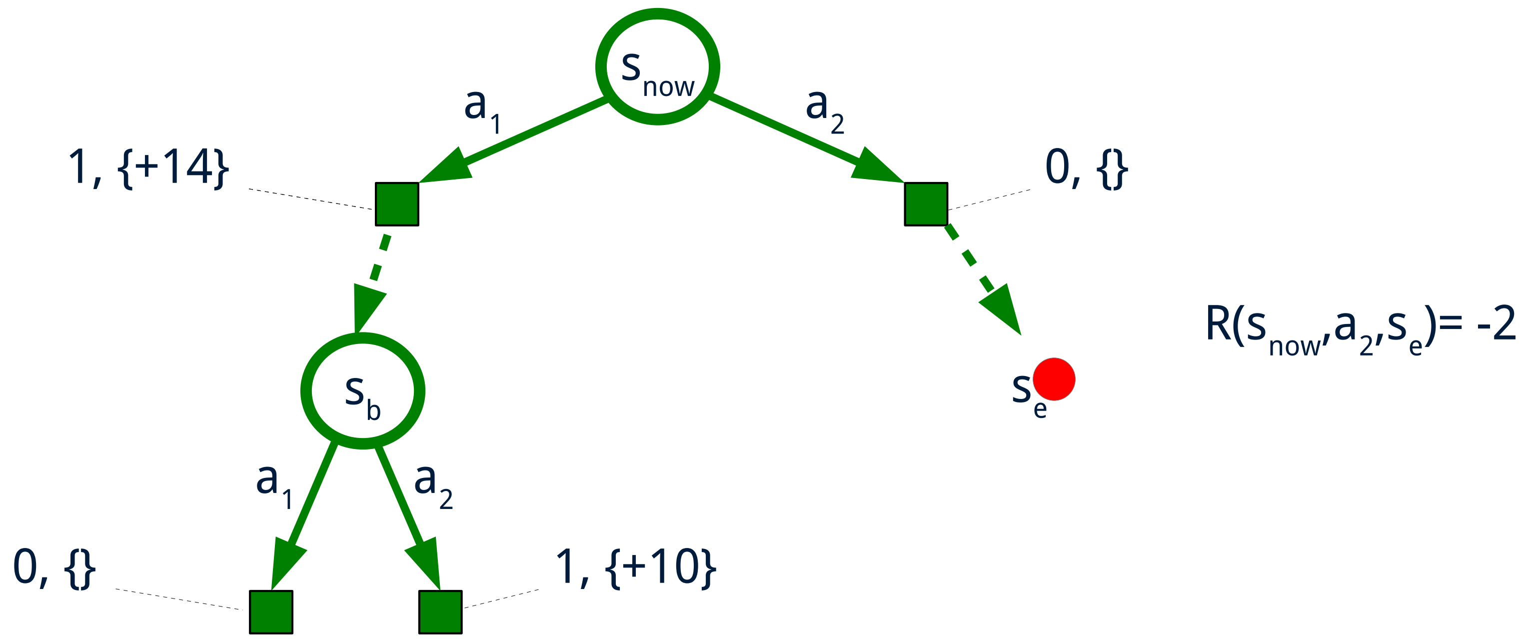
# MCTS – Example



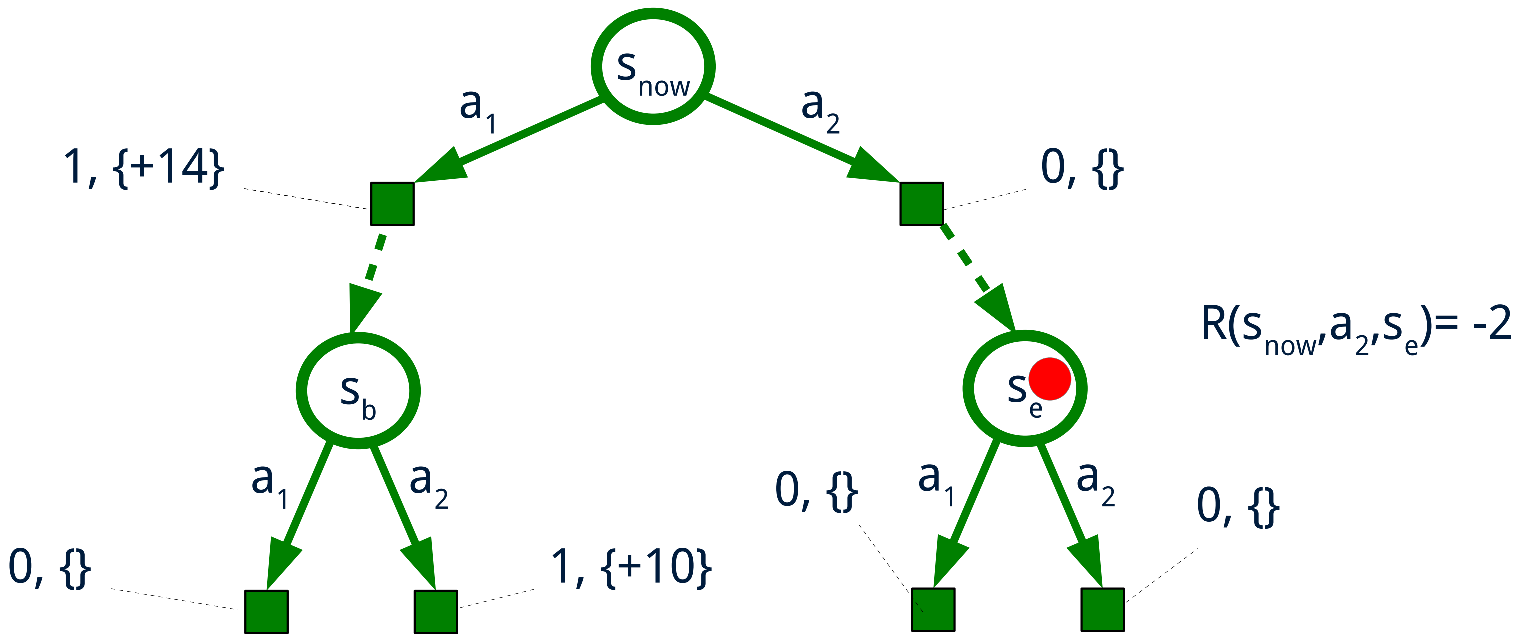
# MCTS – Example



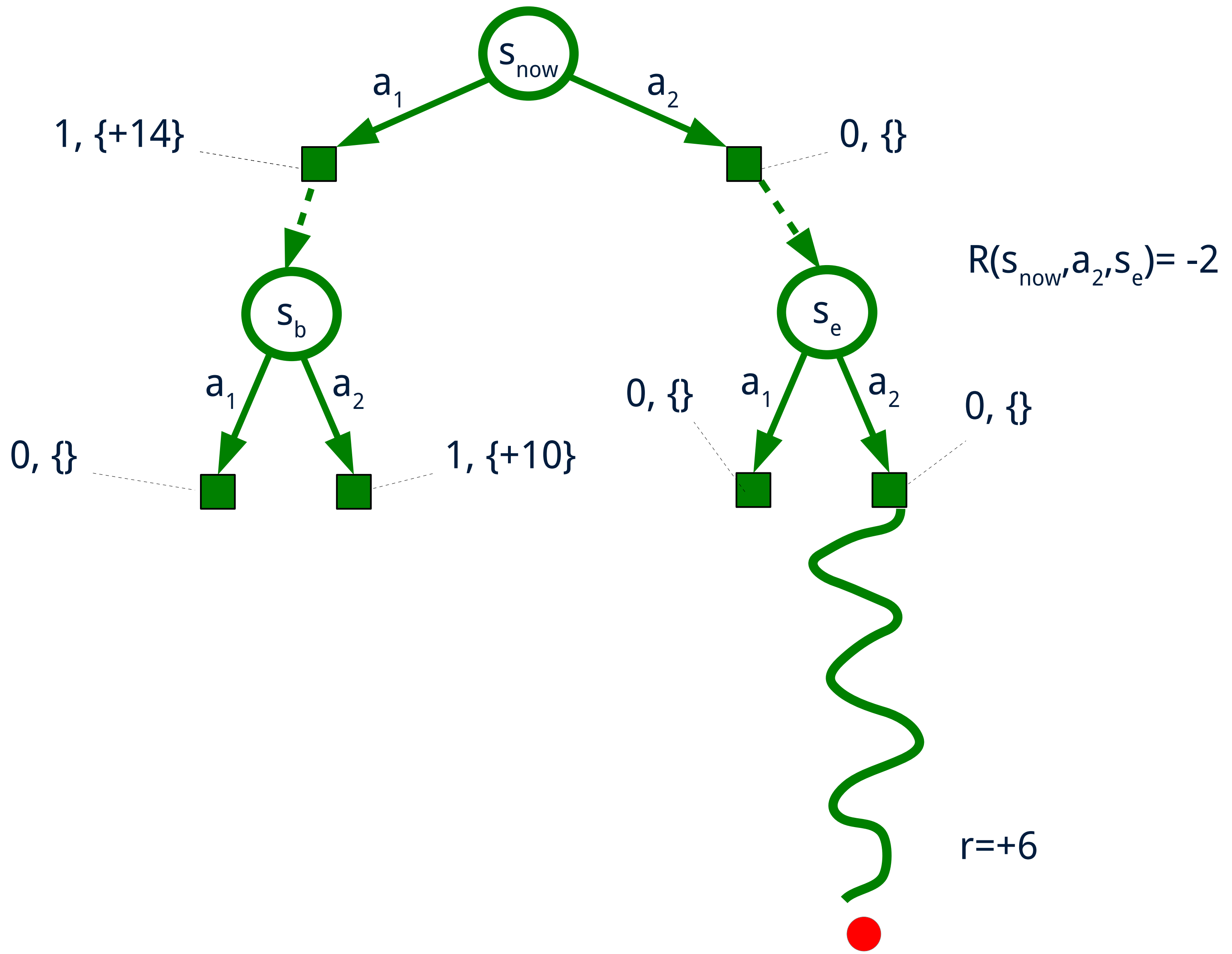
# MCTS – Example



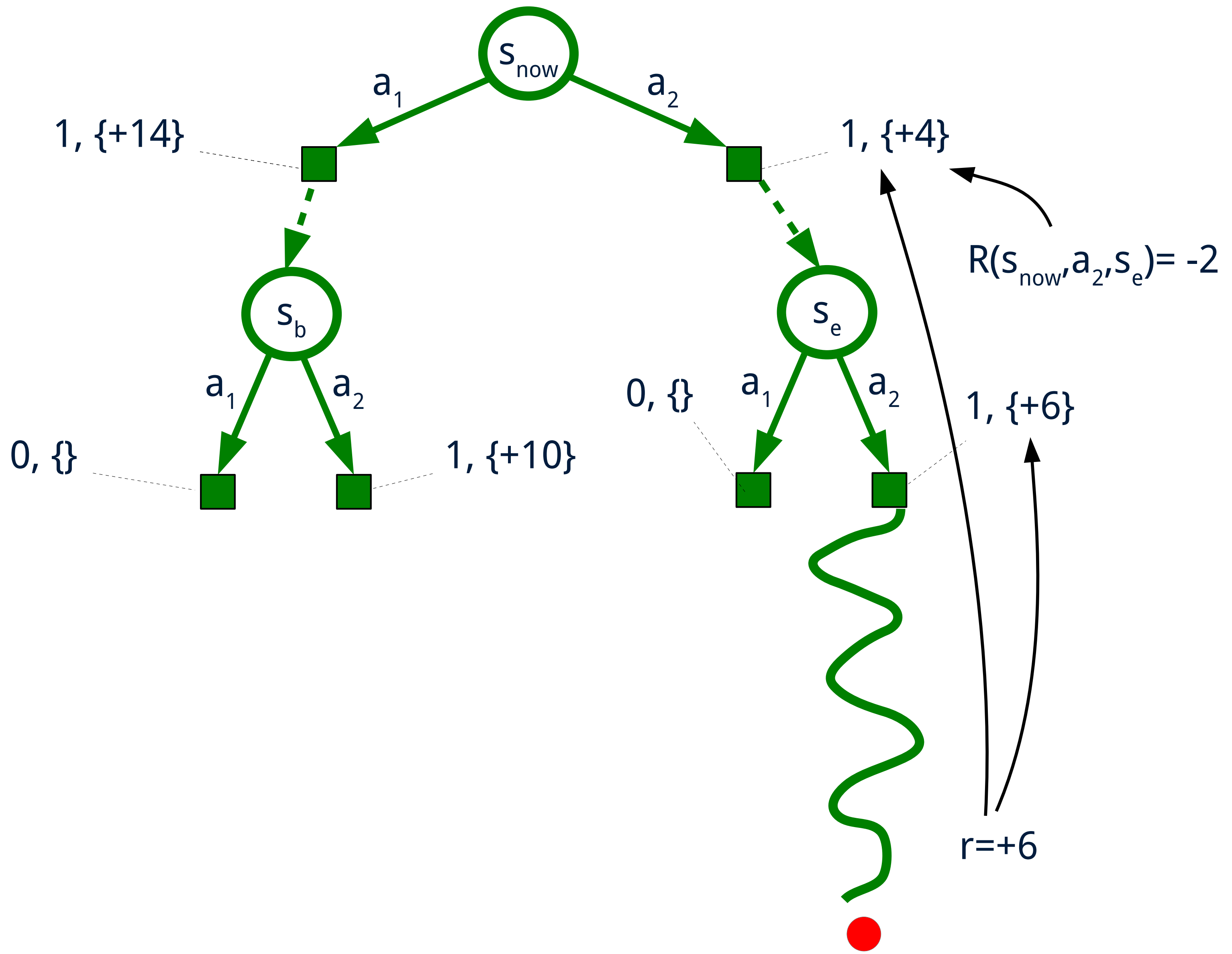
# MCTS – Example



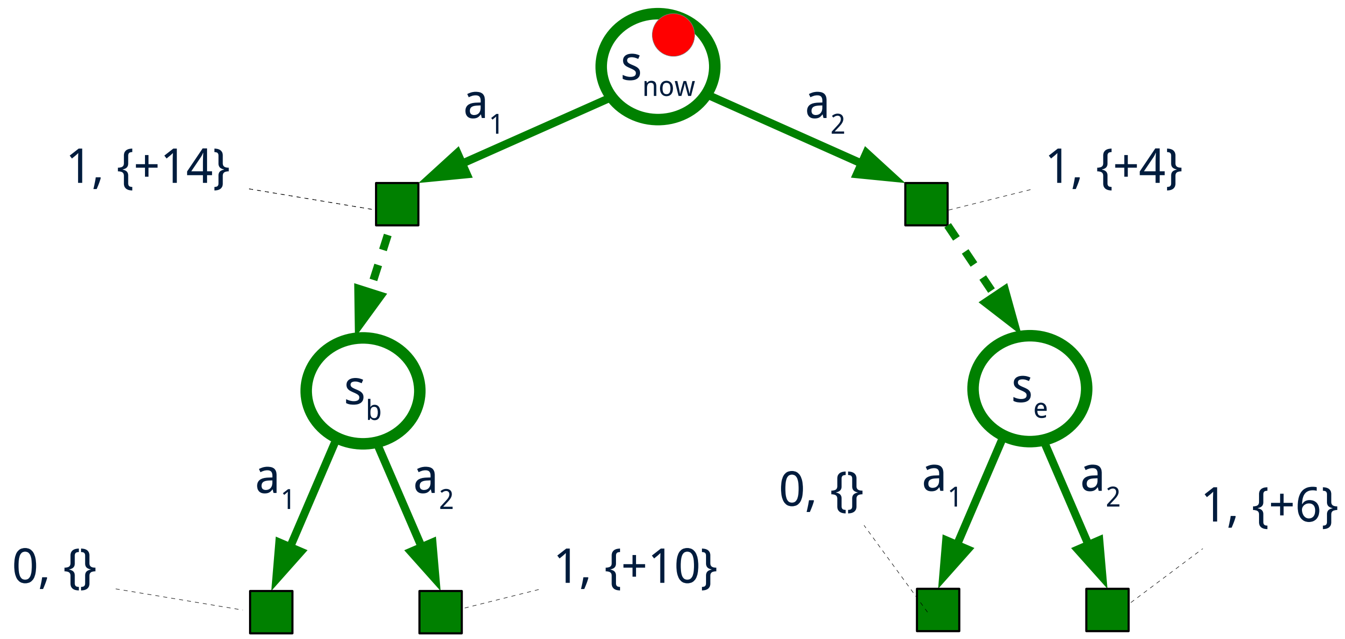
# MCTS – Example



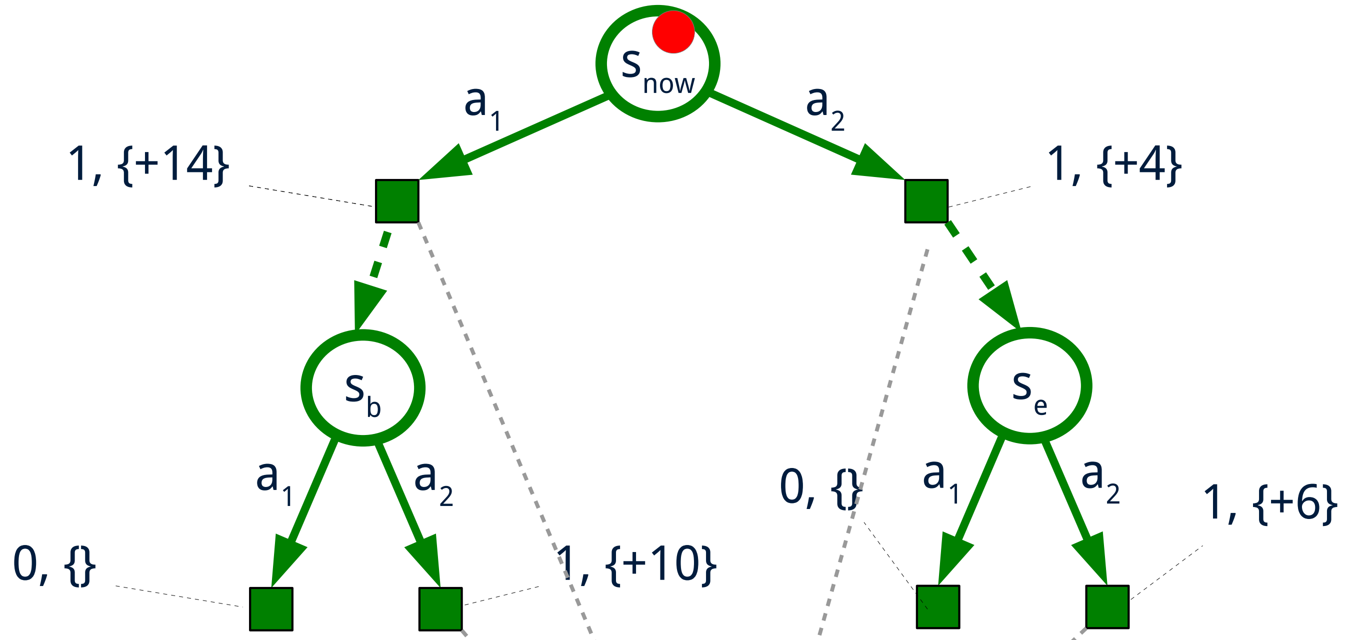
# MCTS – Example



# MCTS – Example

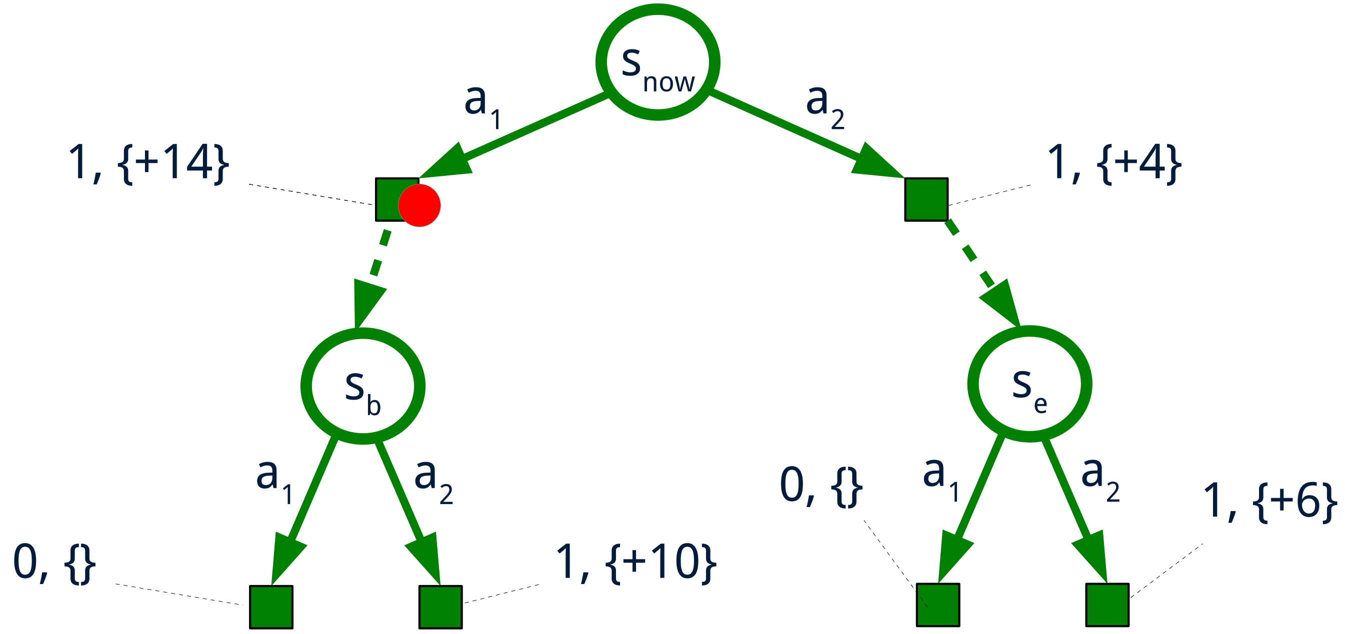


# MCTS – Example

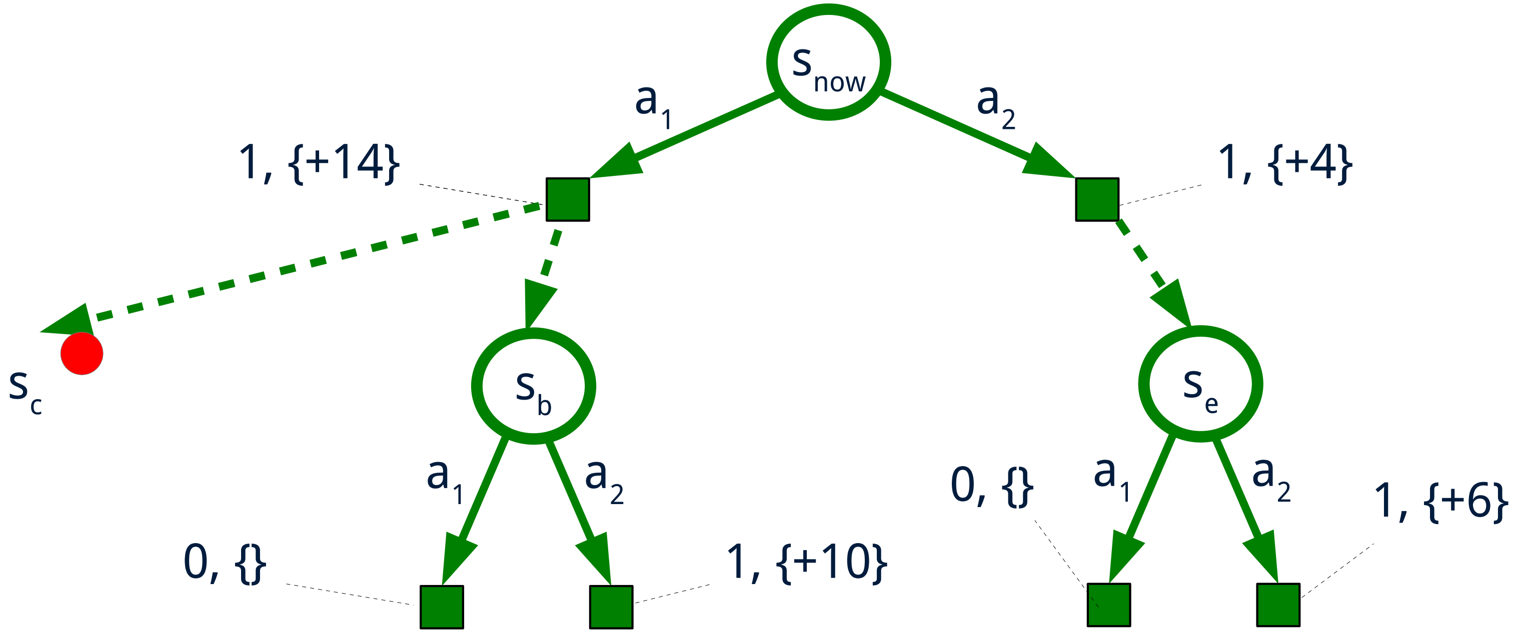



NOTE: the statistics maintained, represent an estimate of **Q(s,a)**

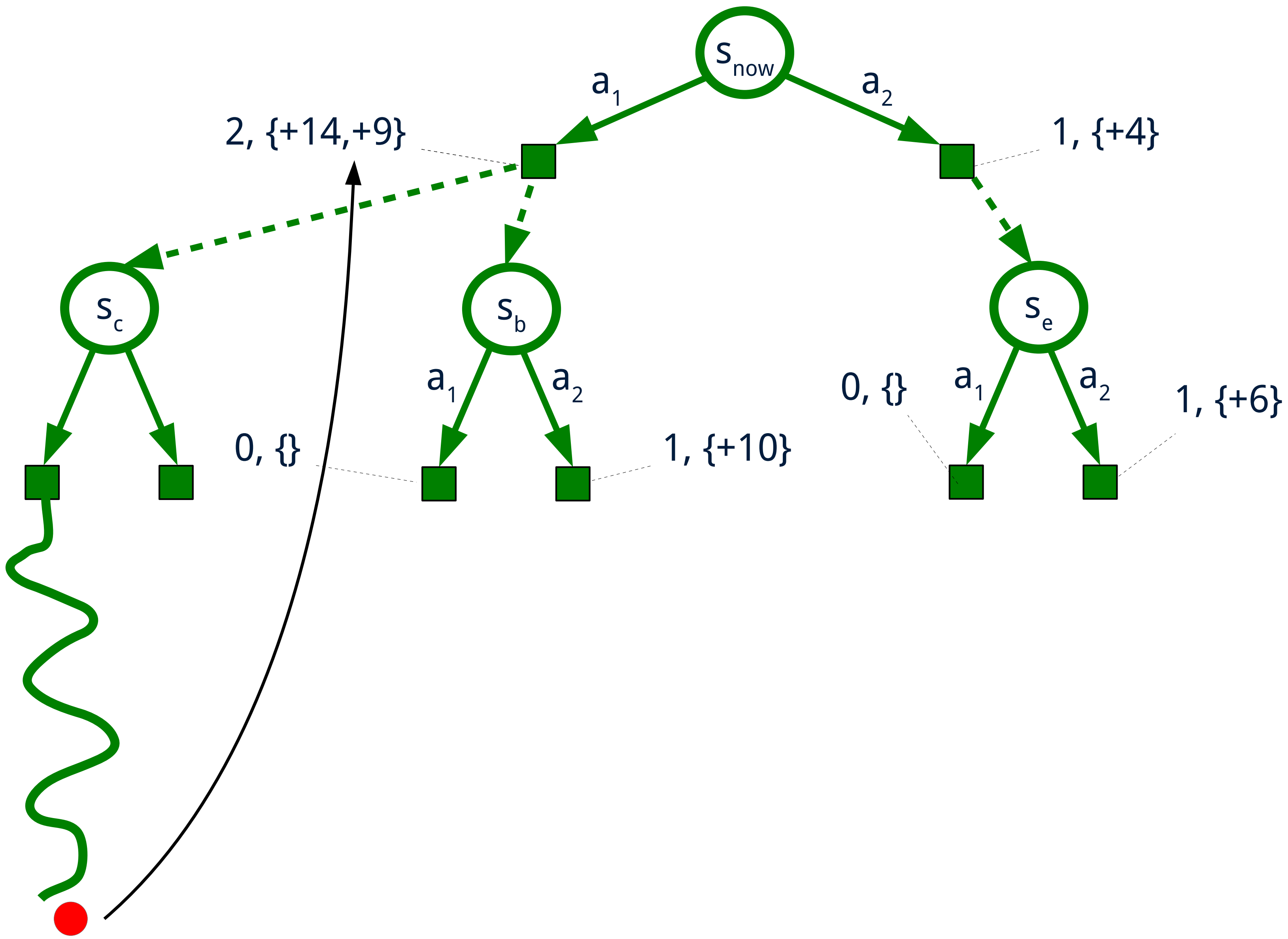
# MCTS – Example



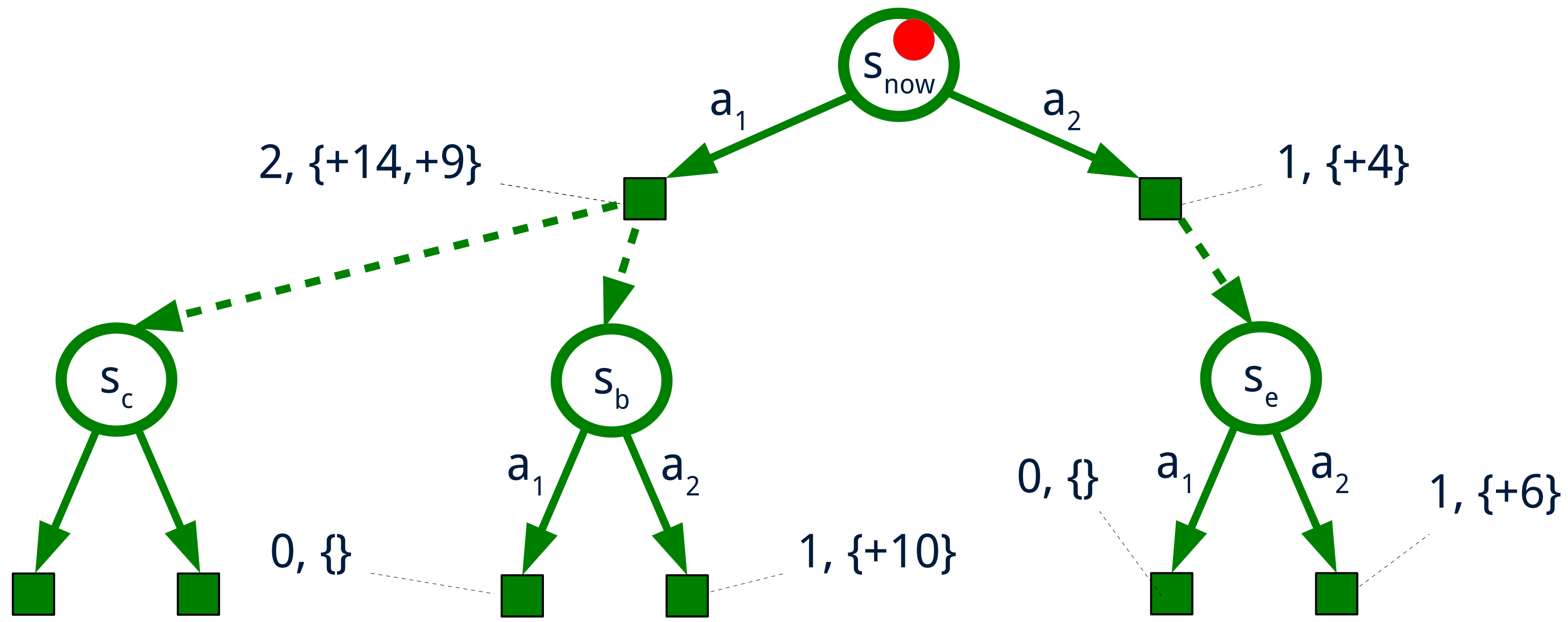
# MCTS – Example



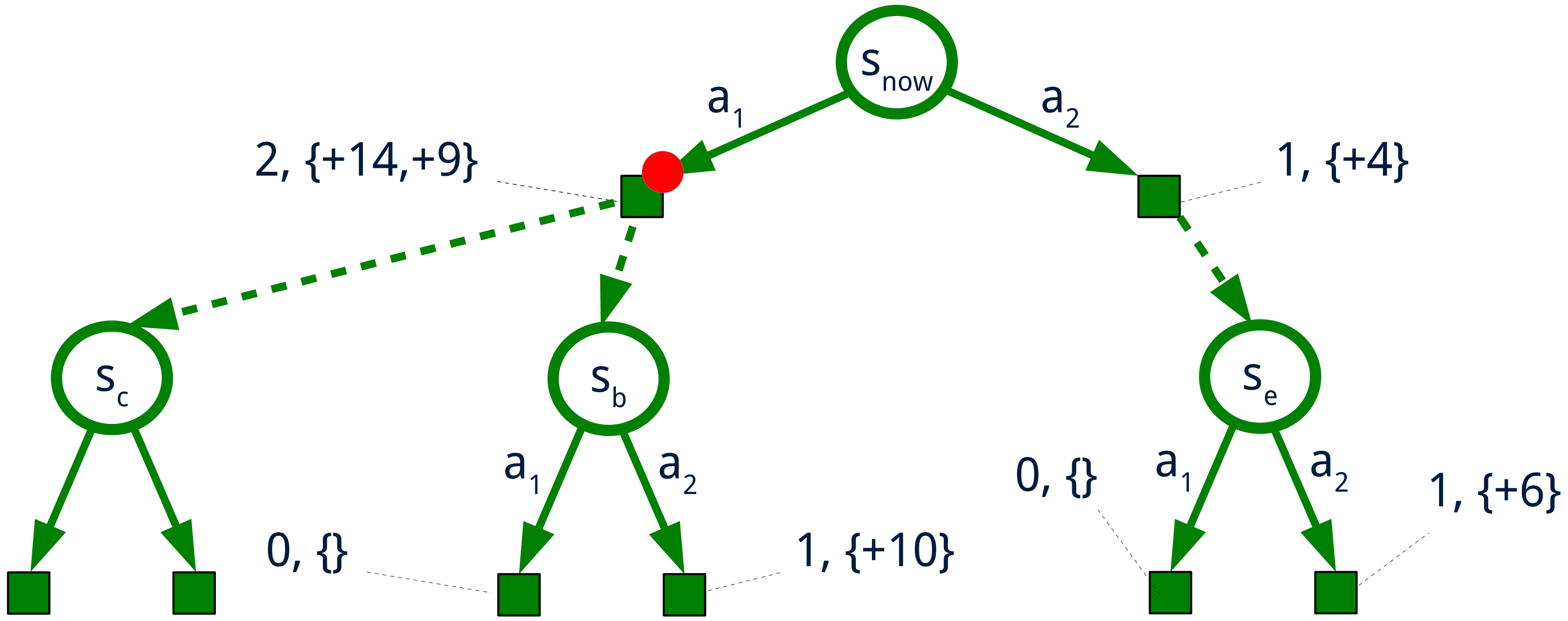
# MCTS – Example



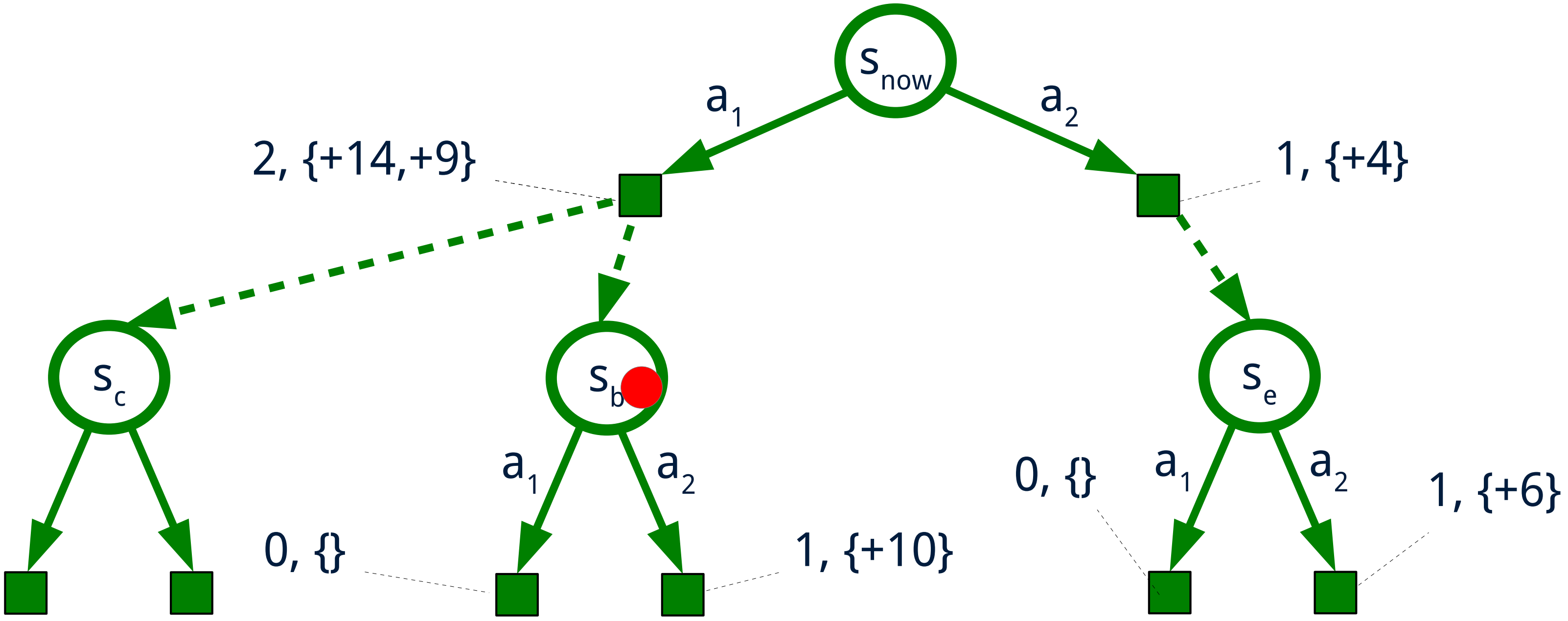
# MCTS – Example



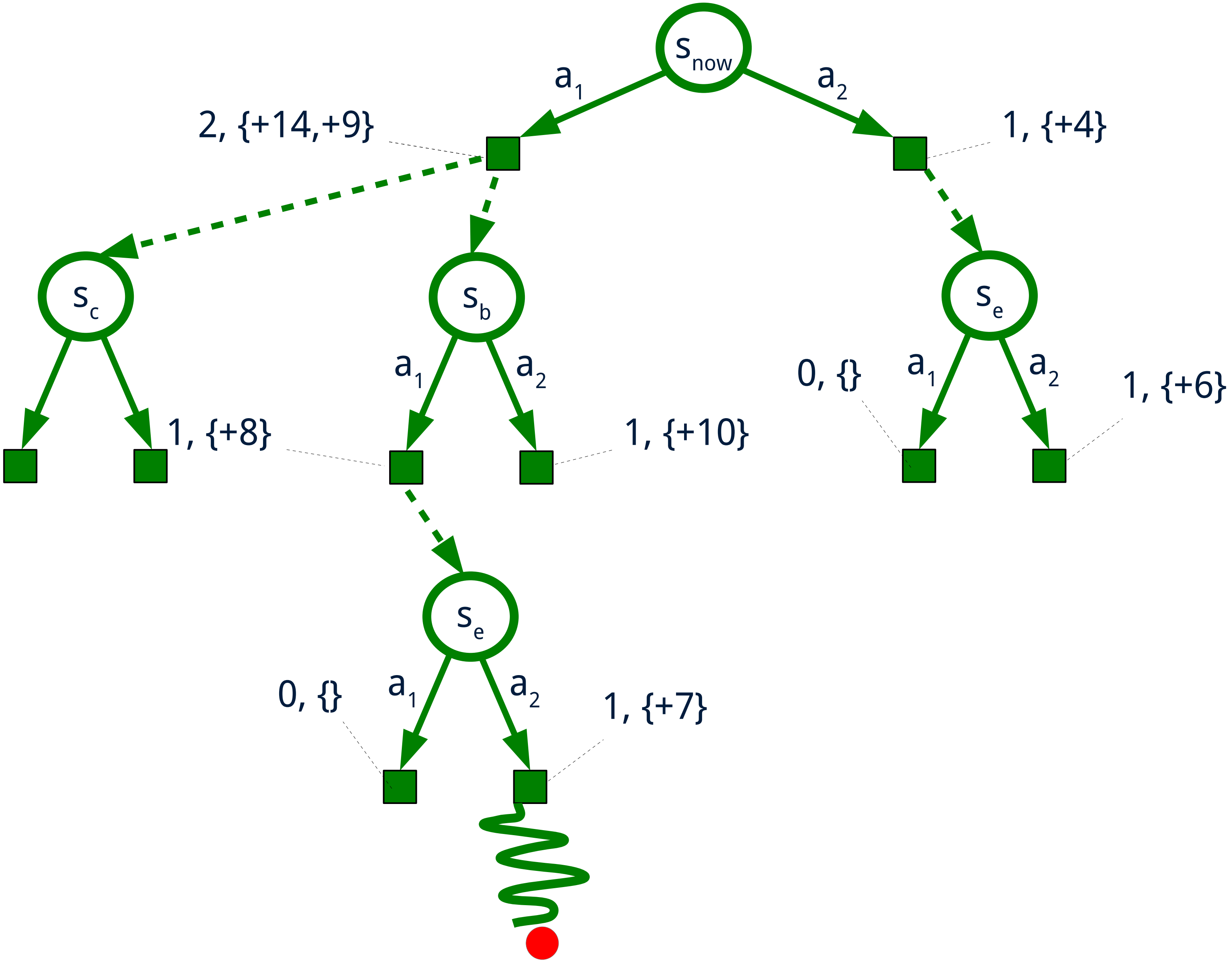
# MCTS – Example



# MCTS – Example

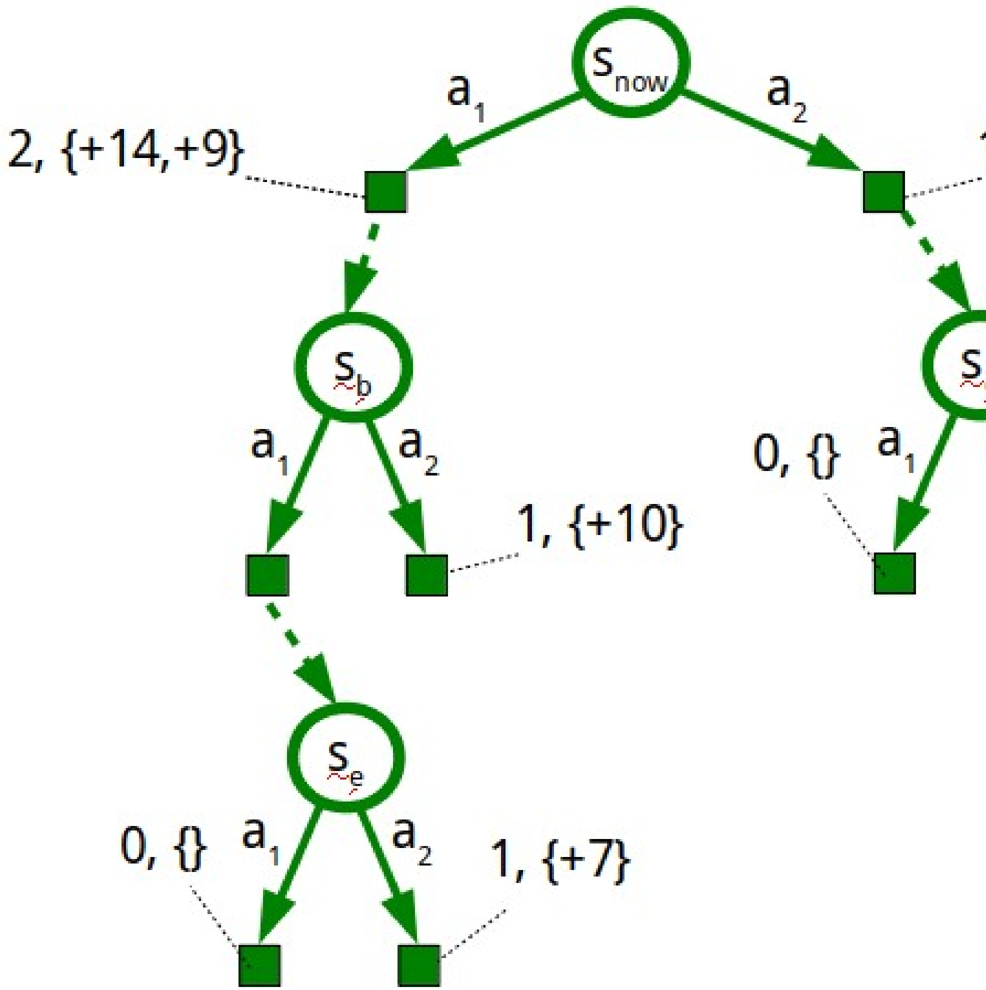


# MCTS – Example



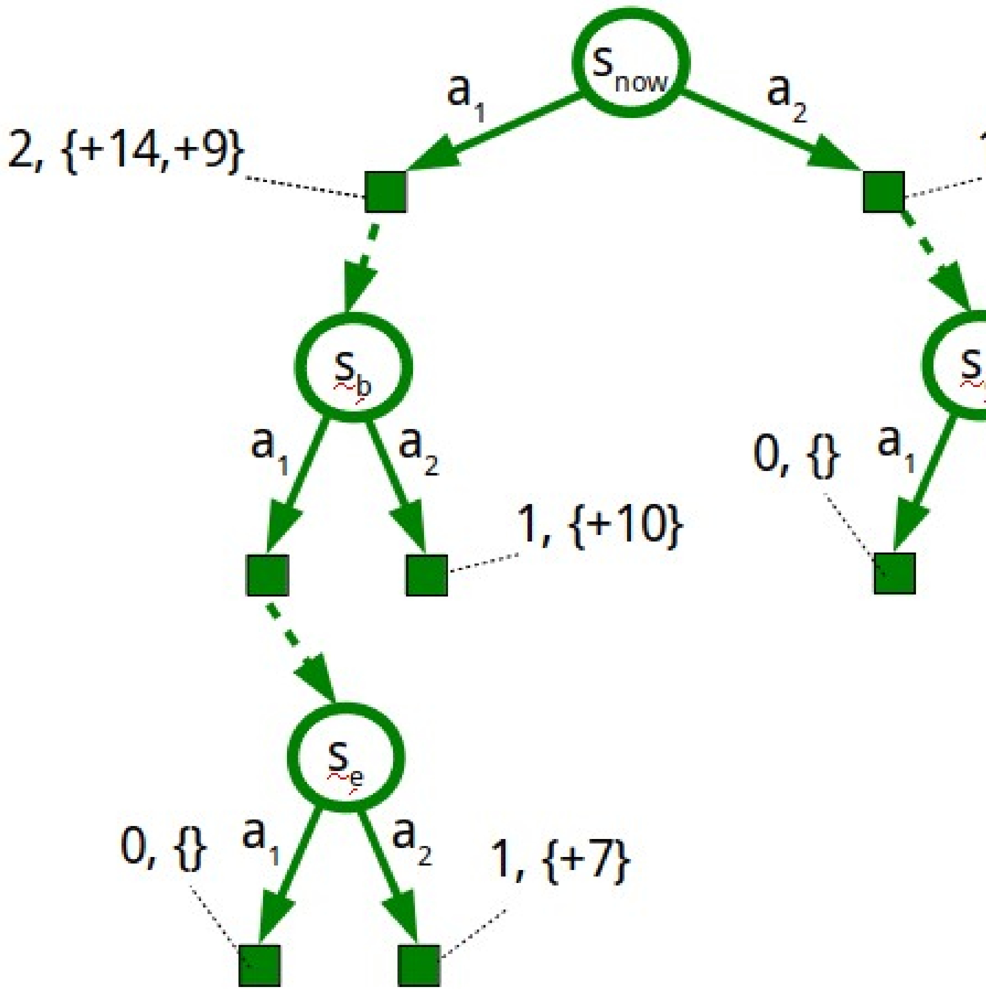
# Convergence...?

- Does this converge...?



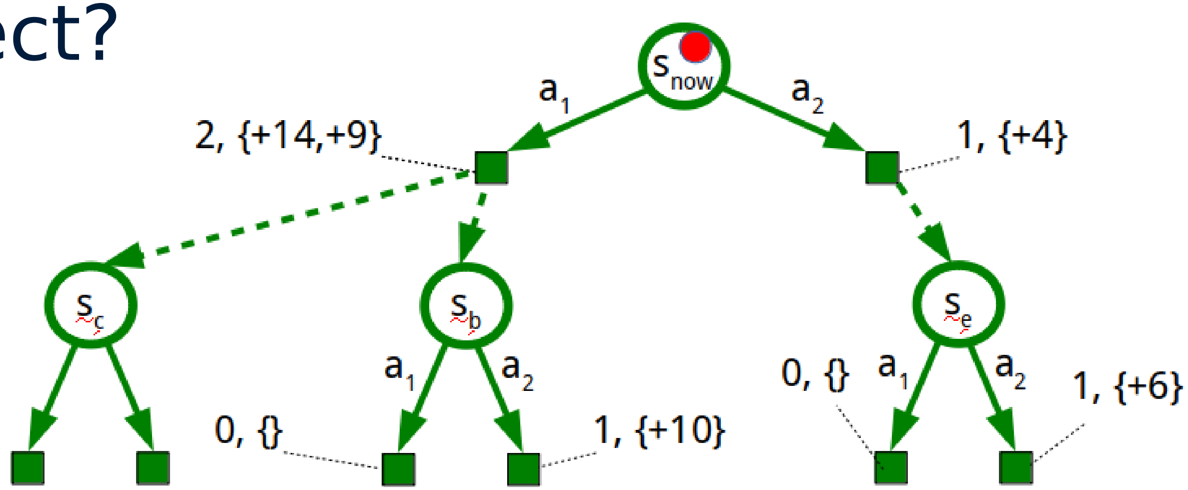
# Convergence...?

- Does this converge...?
- Yes... but not trivial... conflicting requirements:
  - accurate value estimates:
    → try all actions infinitely often
  - estimates of an optimal policy
    → be greedy in sub-tree



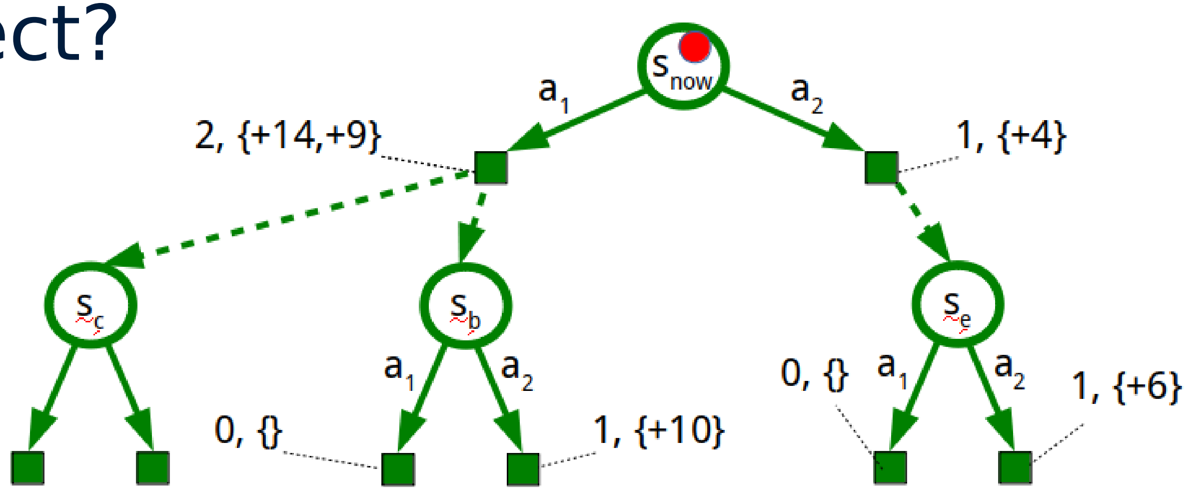
# Action Selection in the tree?

- What actions to select?



# Action Selection in the tree?

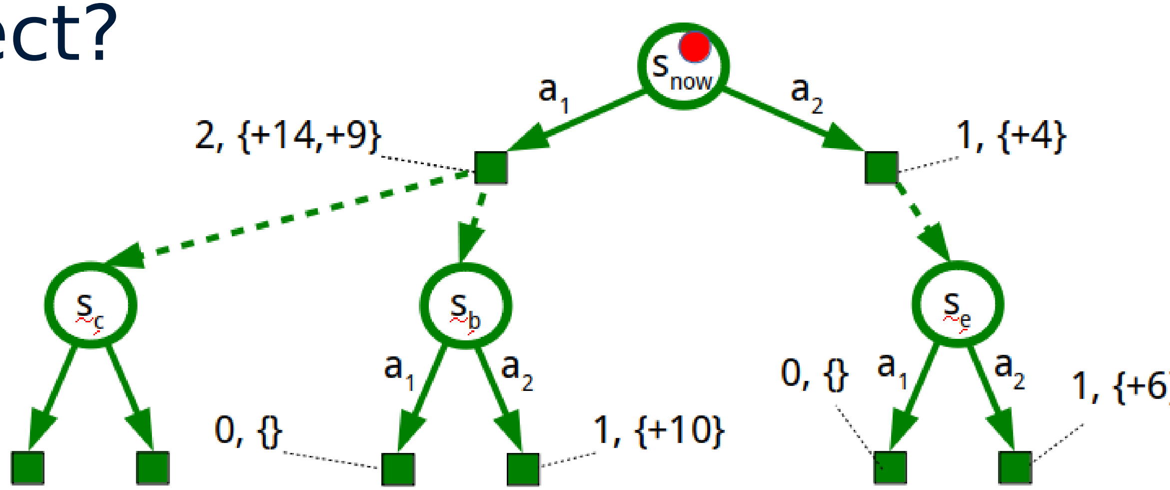
- What actions to select?




- Balance:
  - **exploitation:** focus on good branches
  - **exploration:** see if there could be better branches

# Action Selection in the tree?

- What actions to select?




- Balance:
  - **exploitation:** focus on good branches
  - **exploration:** s
    branches

▶Typical approach: **exploration bonus**

▶E.g., the "UCT" algorithm [Kocsis&Szepesvári'06]

$$U(h,a) = Q(h,a) + c\sqrt{\log(N_h+1)/N_a}$$

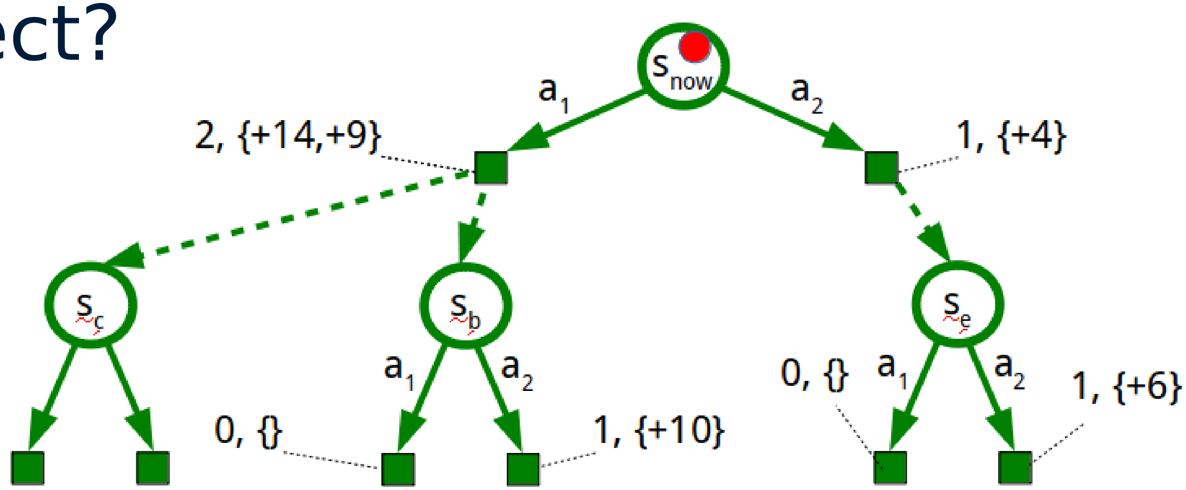
upper confidence bound of node *h*

mean return

exploration bonus

# Action Selection in the tree?

- What actions to select?




- Balance:
  - **exploitation:** focus on good branches
  - **exploration:** s
    branches

▸ Typical approach: **exploration bonus**

▸ E.g., the “UCT” algorithm [Kocsis&Szepesvári'06]

$$U(h,a) = Q(h,a) + c\sqrt{\log(N_h+1)/N_a}$$

upper confidence bound of node *h*

mean return

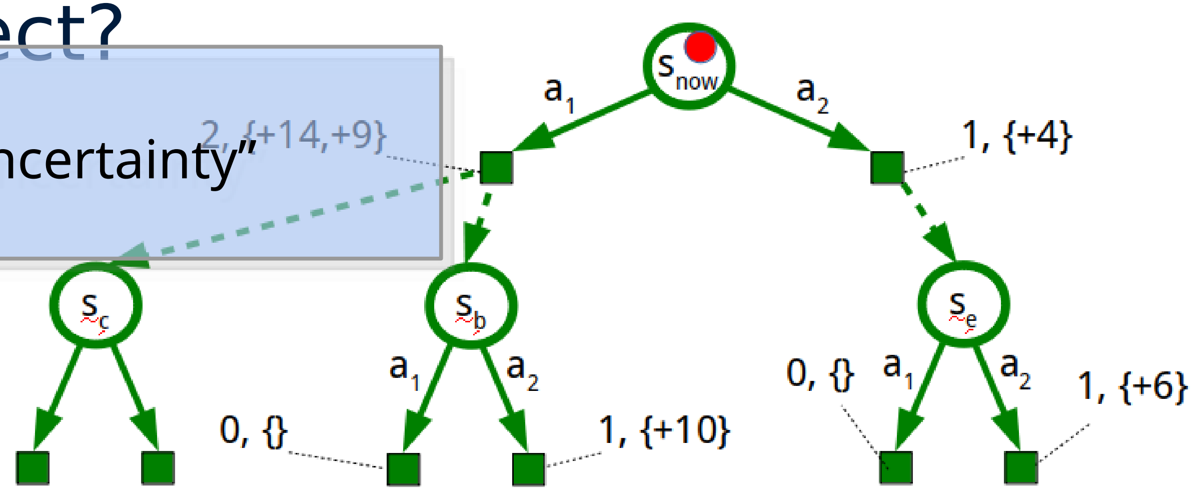
exploration bonus

if *a* tried more often → less bonus

# Action Selection in the tree?

- What actions to select?

"Optimism in the face of uncertainty"




- Balance:
  - **exploitation:** focus on good branches
  - **exploration:** s… branches

▸Typical approach: **exploration bonus**

▸E.g., the "UCT" algorithm [Kocsis&Szepesvári'06]

$$U(h,a) = Q(h,a) + c\sqrt{\log(N_h+1)/N_a}$$

upper confidence bound of node *h*

mean return

exploration bonus

if *a* tried more often → less bonus