Decentralized POMDPs:

A Framework for Multiagent Planning under Uncertainty

Frans A. Oliehoek

Maastricht University

Outline

- Multiagent Systems & Uncertainty
- The Dec-POMDP model
- Policies and their values

    <break>

- Planning for Dec-POMDPs
    - backward: DP
    - forward: heuristic search

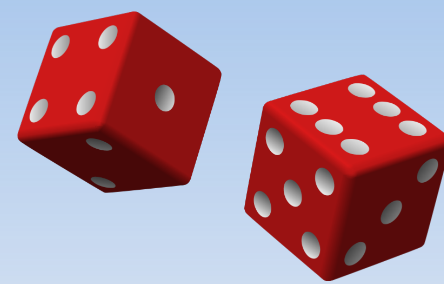
Multiagent Systems (MASs)

Why MASs?

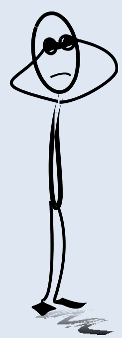
- 1 intelligent agents → soon there will be many...
- Physically distributed systems:
  centralized solutions expensive and brittle.
- Can potentially provide [Vlassis, 2007,Sycara, 1998]
  - Speedup and efficiency
  - Robustness and reliability (‘graceful degradation’)
  - Scalability and flexibility (adding additional agents)

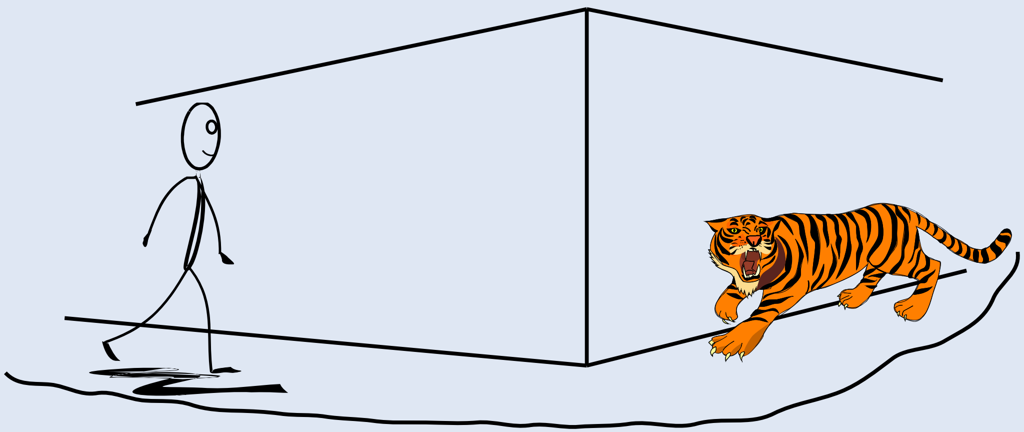
Uncertainty

- Outcome Uncertainty



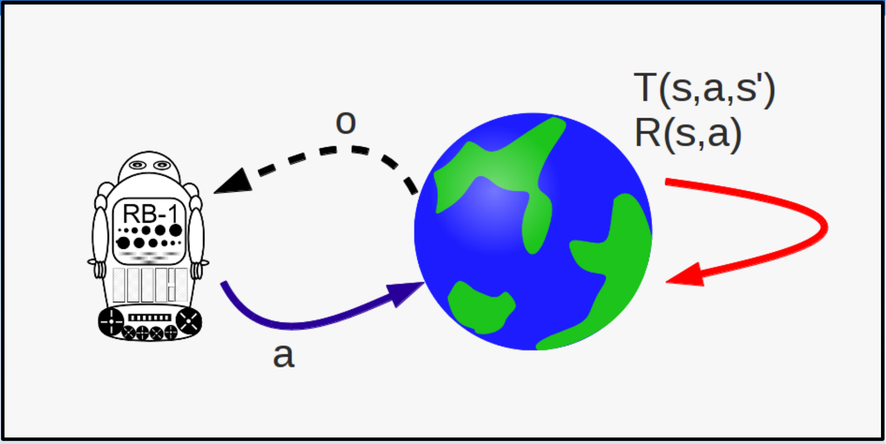
- Partial Observability





- Multiagent Systems: uncertainty about others

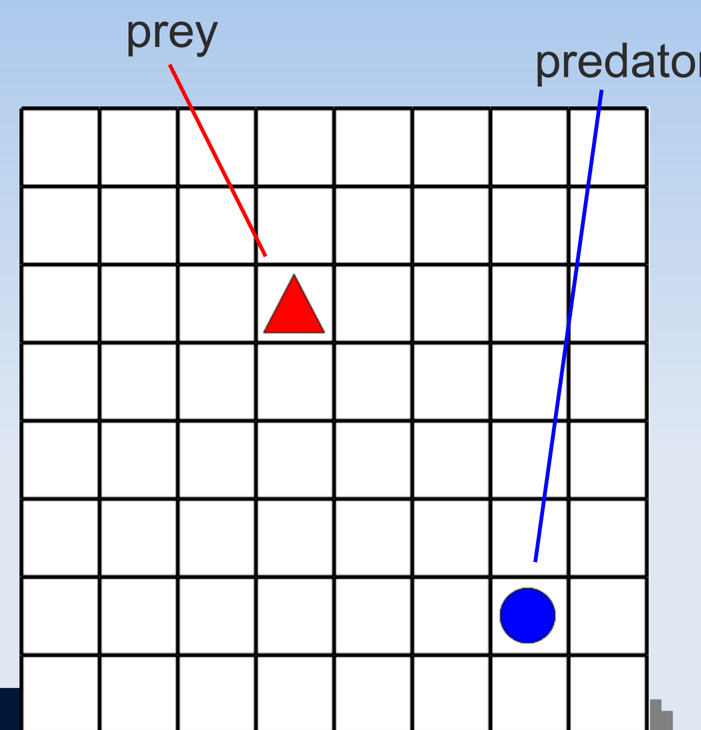
Single-Agent Decision Making



- Background: MDPs & POMDPs
- An MDP $\langle S, A, P_T, R, h \rangle$
  - S – set of states
  - A – set of actions
  - P_T – transition function $P(s'|s,a)$
  - R – reward function $R(s,a)$
  - h – horizon (finite)
- A POMDP $\langle S, A, P_T, O, P_O, R, h \rangle$
  - O – set of observations
  - P_O – observation function $P(o|a,s')$

Example: Predator-Prey Domain

- Predator-Prey domain
  - 1 agent: predator
  - prey: part of environment
  - on a torus

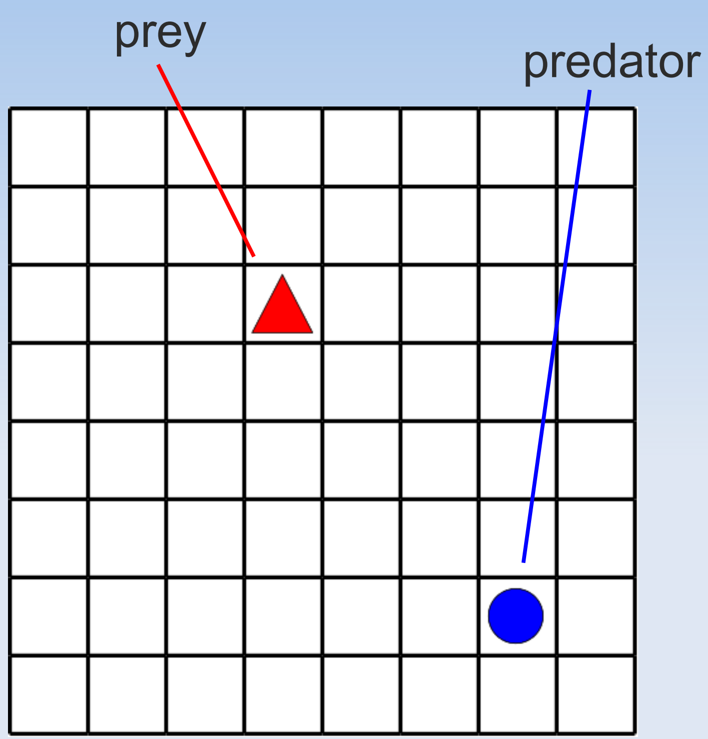


- Formalization:
  - states
  - actions
  - transitions
  - rewards

?

Example: Predator-Prey Domain

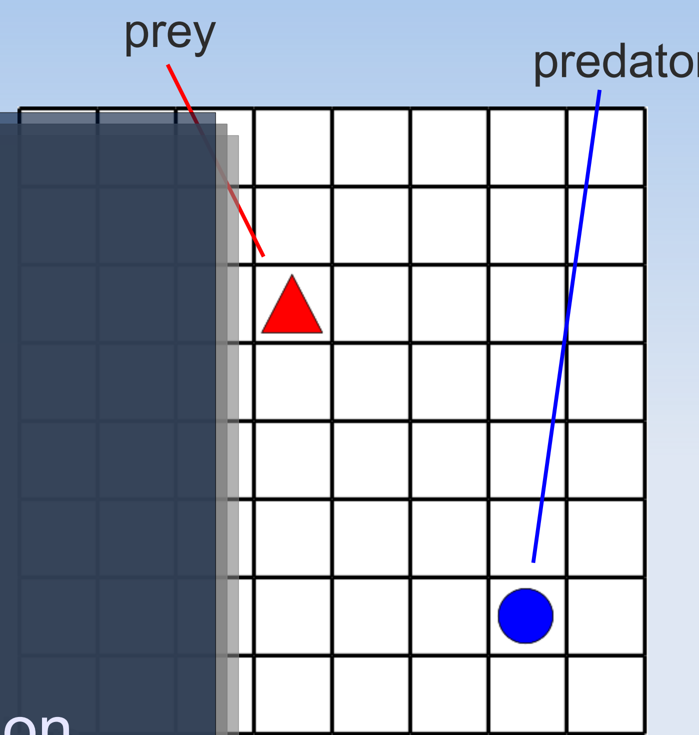
- Predator-Prey domain
  - 1 agent: predator
  - prey: part of environment
  - on a torus



- Formalization:
  - states (-3,4)
  - actions N,W,S,E
  - transitions failing to move, prey moves
  - rewards reward for capturing

Example: Predator-Prey Domain

- Predator-Prey domain



Markov decision process (MDP)

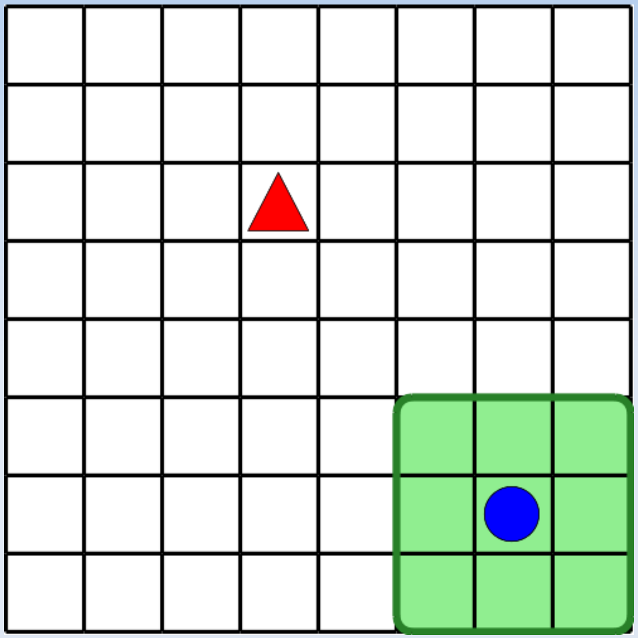
- Markovian state *s*... (which is observed!)
- policy π maps states → actions
- Value function Q(s,a)
- Compute via value iteration / policy iteration

$$Q(s,a)=R(s,a)+\gamma\sum_{s'}P(s'|s,a)\max_{a'}Q(s',a')$$

- rewards reward for capturing

Partial Observability

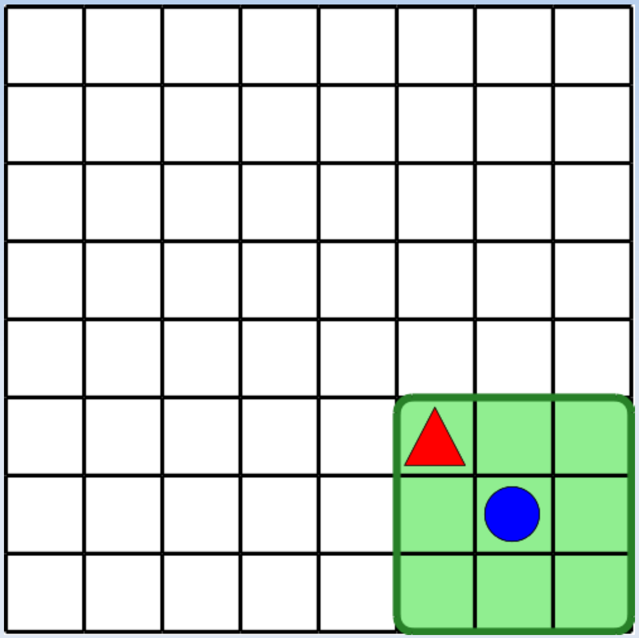
- Now: partial observability
  - E.g., limited range of sight
- MDP + observations
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)



$o = 'nothing'$

Partial Observability

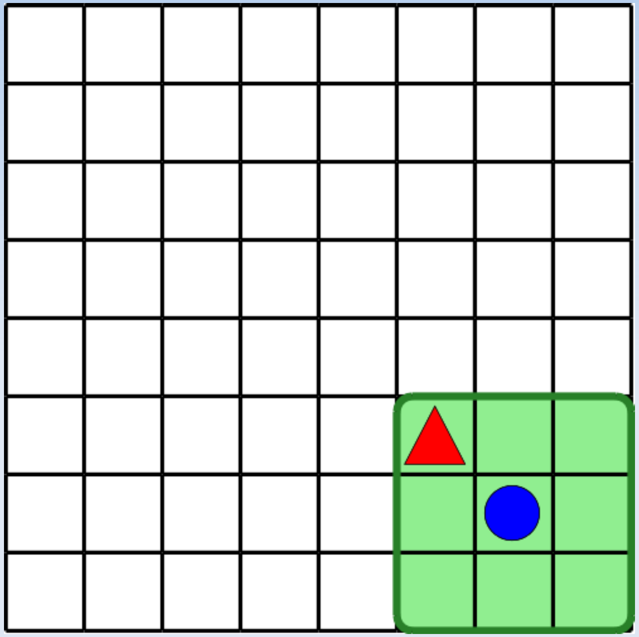
- Now: partial observability
  - E.g., limited range of sight
- MDP + observations
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)



$o = (-1, 1)$

Partial Observability

- Now: partial observability
  - E.g., limited range of sight
- MDP + observations
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)



$o = (-1, 1)$

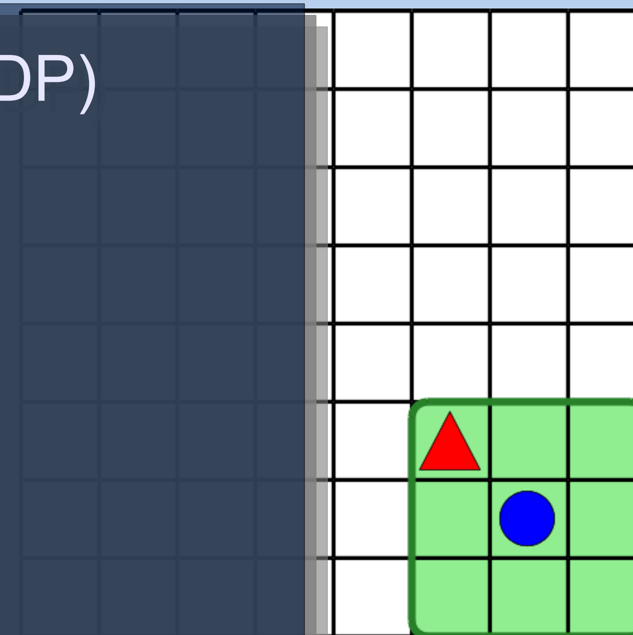
Can not observe the state
→ Need to maintain a belief over states *b(s)*
→ Policy maps beliefs to actions $\pi(b) = a$

Partial Observability

- Now: partial observability
  - E.g.,

Partially Observable MDP (POMDP)

- MDP + observations
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)



$o = (-1, 1)$

Can not observe the state
→ Need to maintain a belief over states *b(s)*
→ Policy maps beliefs to actions $\pi(b) = a$

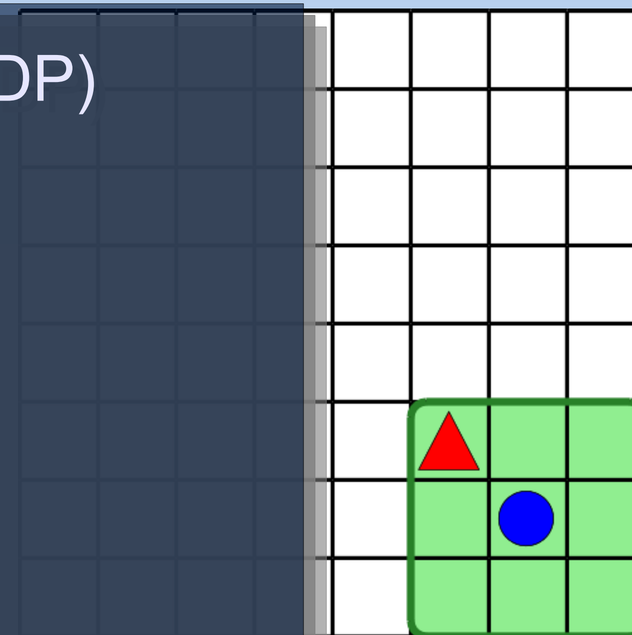
Partial Observability

- Now: partial observability

Partially Observable MDP (POMDP)

- reduction → continuous state MDP (in which the belief **is** the state)

- explicit observations
- observation probabilities
  - noisy observations (detection probability)



$o = (-1, 1)$

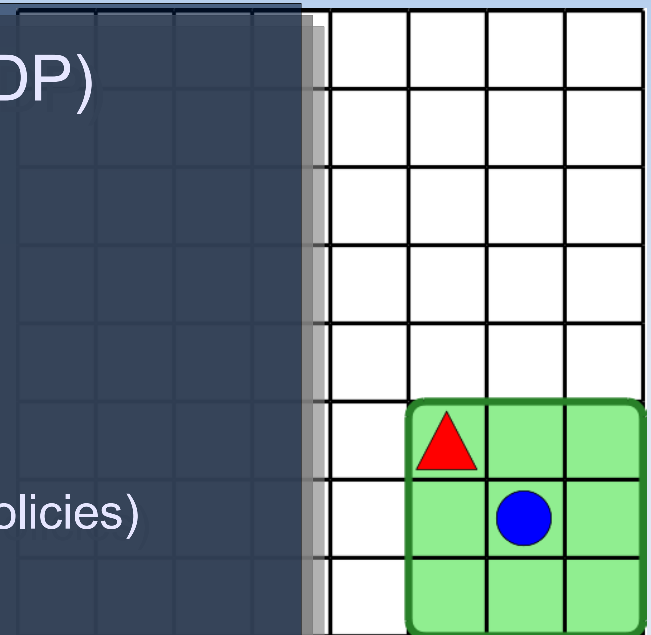
Can not observe the state
→ Need to maintain a belief over states *b(s)*
→ Policy maps beliefs to actions $\pi(b) = a$

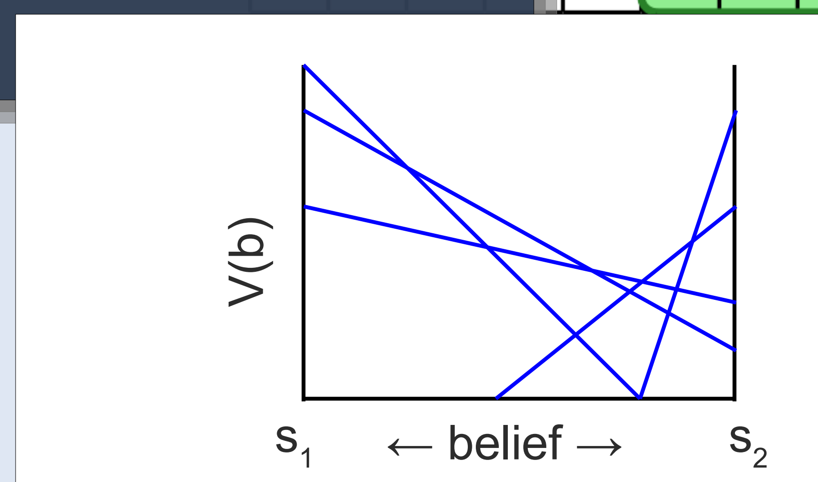
Partial Observability

- Now: partial observability

Partially Observable MDP (POMDP)

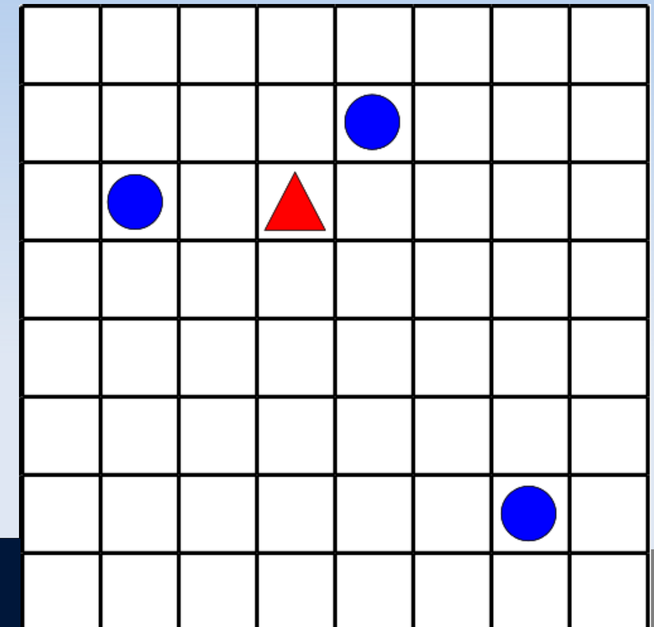
- reduction → continuous state MDP (in which the belief **is** the state)
- Value iteration:
  - make use of α-vectors (↔ complete policies)
  - perform pruning





Multiple Agents

- Now: multiple agents
  - fully observable

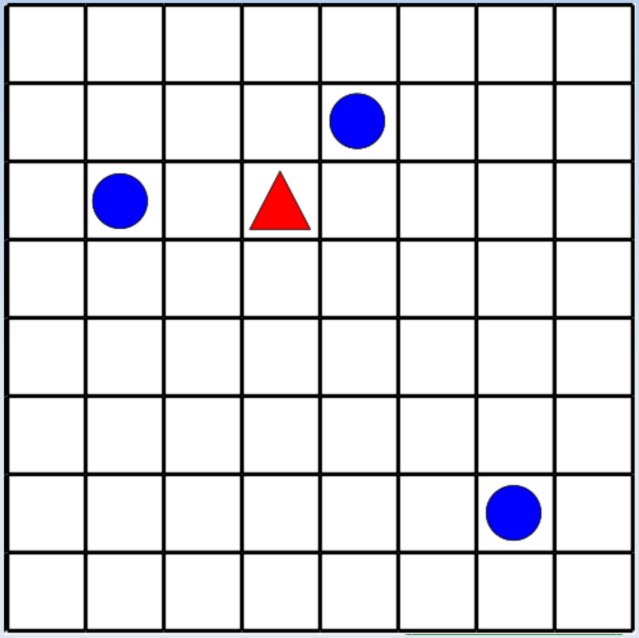


- Formalization:
  - states
  - actions
  - **joint** actions
  - transitions
  - rewards

?

Multiple Agents

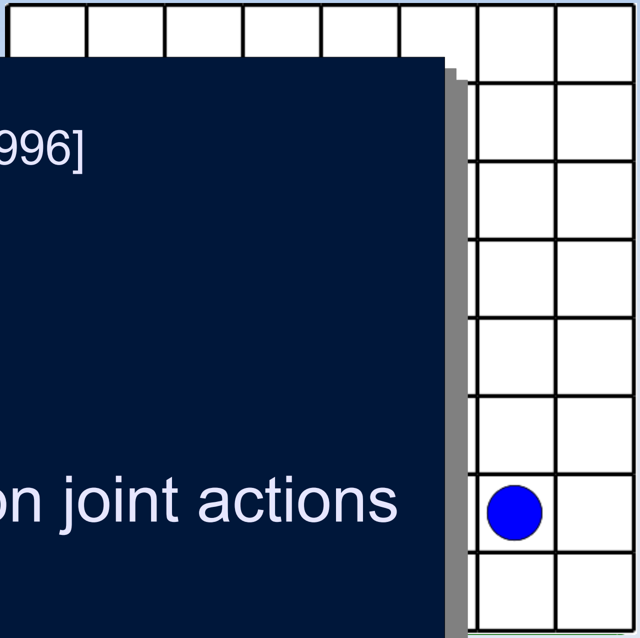
- Now: multiple agents
  - fully observable



- Formalization:
  - states ((3,-4), (1,1), (-2,0))
  - actions {N,W,S,E}
  - **joint** actions {(N,N,N), (N,N,W),...,(E,E,E)}
  - transitions probability of failing to move, prey moves
  - rewards reward for capturing jointly

Multiple Agents

- Now: multiple agents



- Fo

Multiagent MDP [Boutilier 1996]

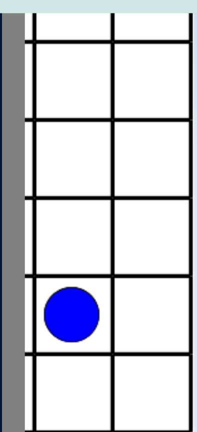
- Differences with MDP
  - *n* agents
  - joint actions $a = \langle a_1, a_2, \ldots, a_n \rangle$
  - transitions and rewards depend on joint actions
- Solution:
  - Treat as normal MDP with 1 'puppeteer agent'
  - Optimal policy $\pi(s) = a$
  - Every agent executes its part

es

- rewards reward for capturing jointly

Multiple Agents

- Now: multiple agents
  - fully observable
- Fo
  - es
  - rewards reward for capturing jointly



Multiagen

- Differences with MDP
  - *n* agents
  - joint actions $a = \langle a_{1,} a_{2,} \ldots , a_n \rangle$
  - transitions and rewards depend on joint actions
- Solution:
  - Treat as normal MDP with 1 'puppeteer agent'
  - Optimal policy $\pi(s) = a$
  - Every agent executes its part

Catch: …?

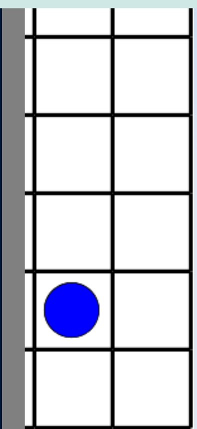
Multiple Agents

- Now: multiple agents
  - fully observable
- Fo

Catch: number of joint actions is **exponential**!
(but other than that, conceptually simple.)

Multiagen

- Differences with MDP
  - n agents
  - joint actions $a = \langle a_1, a_2, \ldots, a_n \rangle$
  - transitions and rewards depend on joint actions
- Solution:
  - Treat as normal MDP with 1 'puppeteer agent'
  - Optimal policy $\pi(s) = a$
  - Every agent executes its part

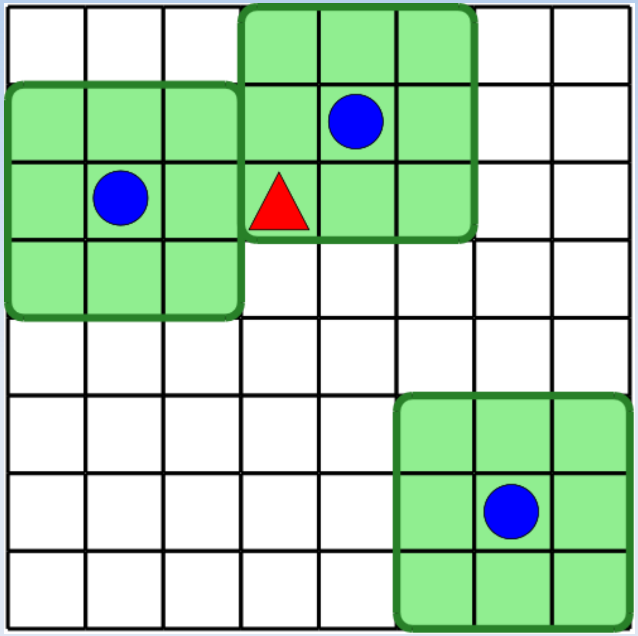


es

- rewards reward for capturing jointly

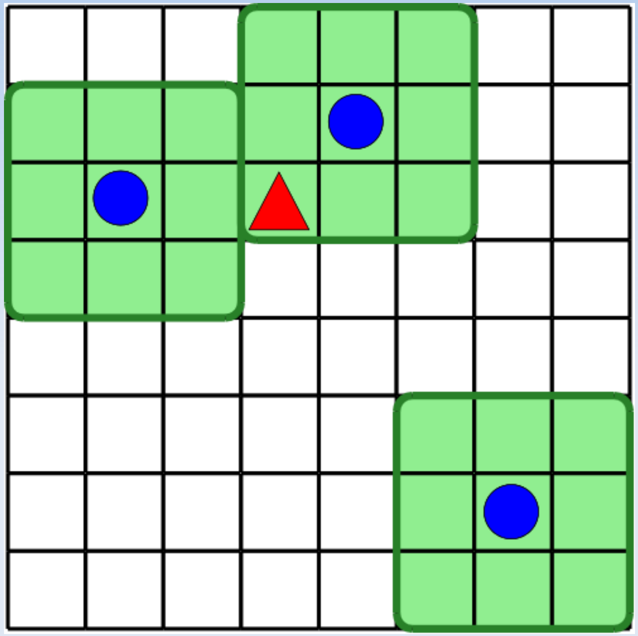
Multiple Agents & Partial Observability

- Now both...
  - partial observability
  - multiple agents



Multiple Agents & Partial Observability

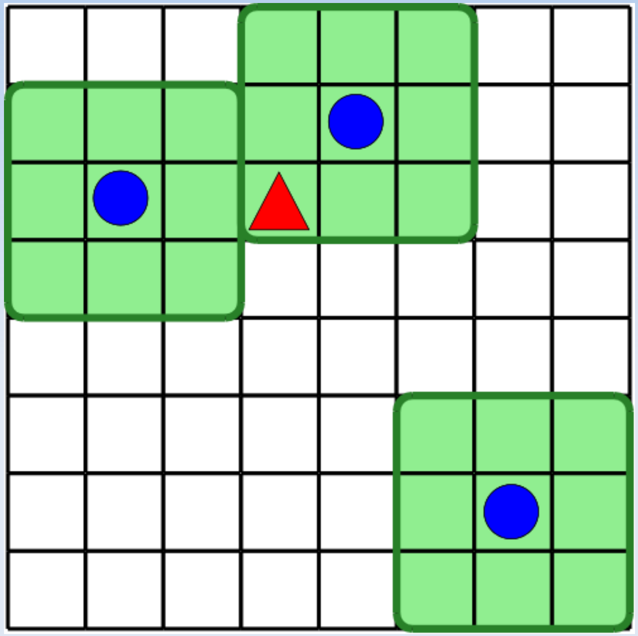
- Now both...
  - partial observability
  - multiple agents
- Decentralized POMDPs (Dec-POMDPs) [Bernstein et al. 2002]
- both
  - joint actions and
  - joint observations



Multiple Agents & Partial Observability

- Again we can make a reduction...

  any idea?

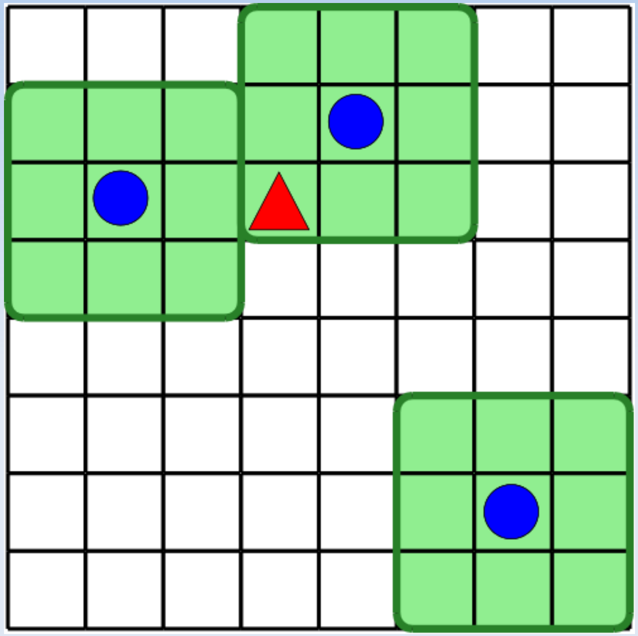


Multiple Agents & Partial Observability

- Again we can make a reduction...

  Dec-POMDPs → MPOMDP

  (multiagent POMDP)

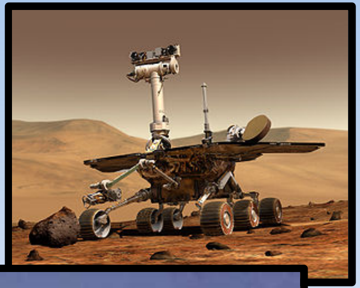


- 'puppeteer agent'
  - receives joint observations
  - takes joint actions
- requires broadcasting observations!
  - instantaneous, cost-free, noise-free communication → optimal [Pynadath and Tambe 2002]
  - Without such communication: no easy reduction.

The Dec-POMDP Model

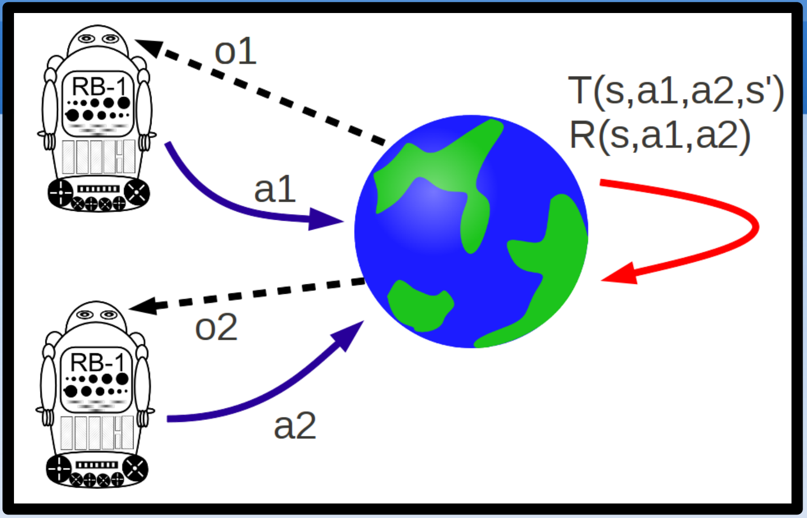
Acting Based On Local Observations

- MPOMDP: Act on global information
- Can be impractical:
  - communication not possible
  - significant cost (e.g battery power)
  - not instantaneous or noise free
  - scales poorly with number of agents!



- Alternative: act based only on local observations
  - Other side of the spectrum: no communication at all
  - (Also other intermediate approaches: delayed communication, stochastic delays)

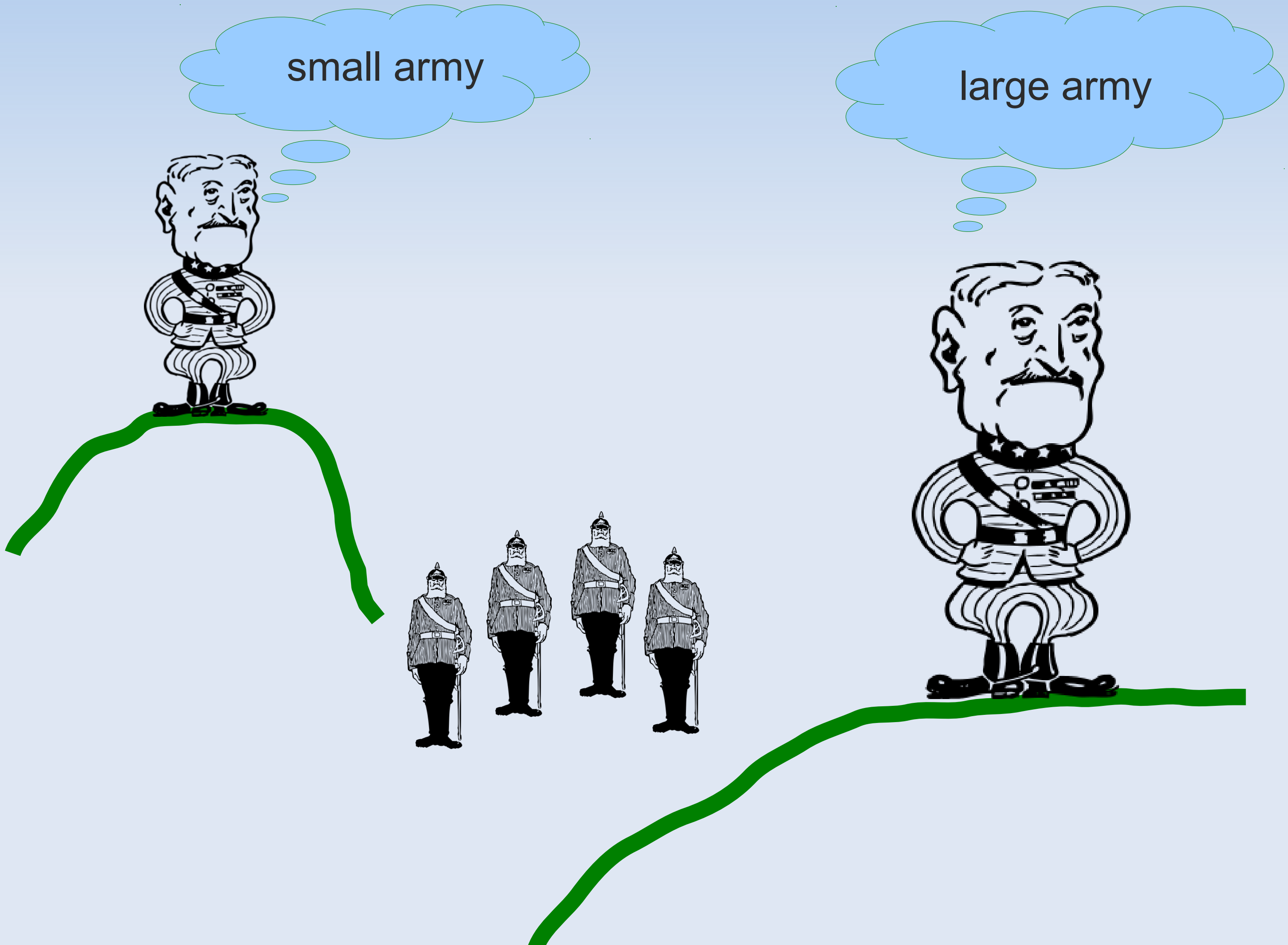
Formal Model



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

Running Example

- 2 generals problem



Running Example

- 2 generals problem

S – { s_L, s_S }

A_i – { (O)bserve, (A)ttack }

O_i – { (L)arge, (S)mall }

Transitions

- Both Observe → no state change
- At least 1 Attack → reset (50% probability s_L, s_S)

Observations

- Probability of correct observation: 0.85
- E.g., P(<L, L> | s_L) = 0.85 * 0.85 = 0.7225
- (reset is not observed!)

Running Example

- 2 generals problem

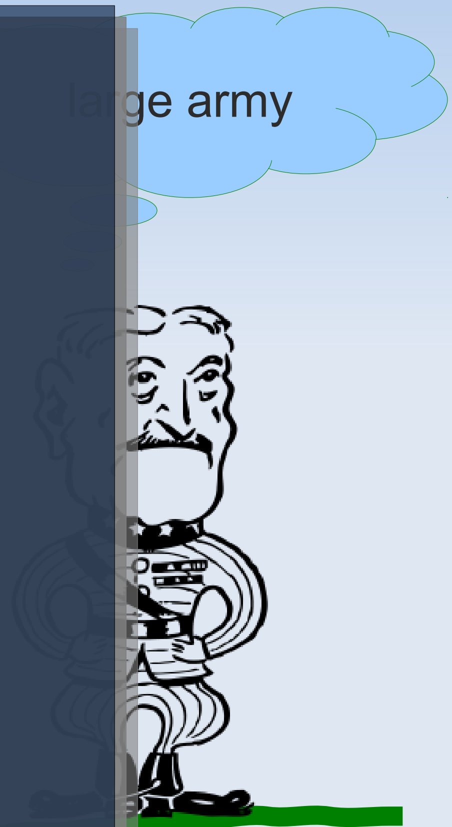
S – { s_L, s_S }

A_i – { (O)bserve, (A)ttack }

O_i – { (L)arge, (S)mall }

Rewards

- 1 general attacks: he loses the battle
  - $R(*,<A,O>) = -10$
- Both generals Observe: small cost
  - $R(*,<O,O>) = -1$
- Both Attack: depends on state
  - $R(s_L,<A,A>) = -20$
  - $R(s_S,<A,A>) = +5$



Running Example

- 2 generals problem

S – { s_L, s_S }

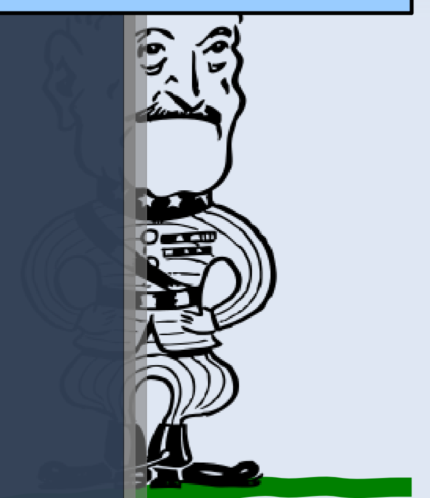
A_i – { (O)bserve, (A)ttack }

O_i – { (L)arge, (S)mall }

Rewards

- 1 general attacks: he loses the battle
  - R(*,<A,O>) = -10
- Both generals Observe: small cost
  - R(*,<O,O>) = -1
- Both Attack: depends on state
  - R(s_L,<A,A>) = -20
  - R(s_R,<A,A>) = +5

suppose h=3,
what do you think is optimal in this problem?



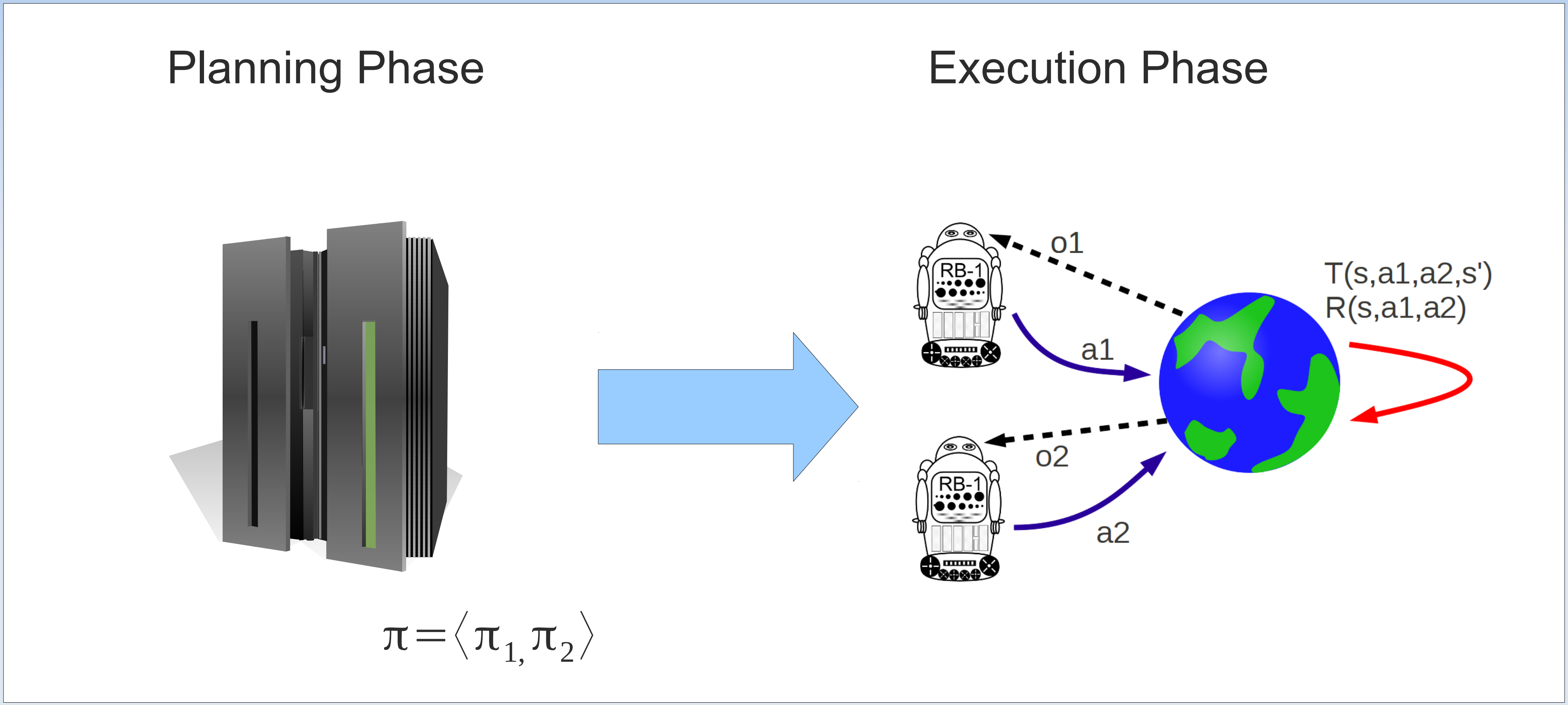
Related Frameworks

- Partially observable stochastic games [Hansen et al. 2004]
  - Non-identical payoff
- Interactive POMDPs [Gmytrasiewicz & Doshi 2005, JAIR]
  - Subjective view of MAS
- Imperfect information extensive form games
  - Represented by game tree
  - E.g., poker [Sandholm 2010, AI Magazine]

Rest of lecture:
planning for Dec-POMDPs...

Off-line / On-line phases

- off-line planning, on-line execution is decentralized



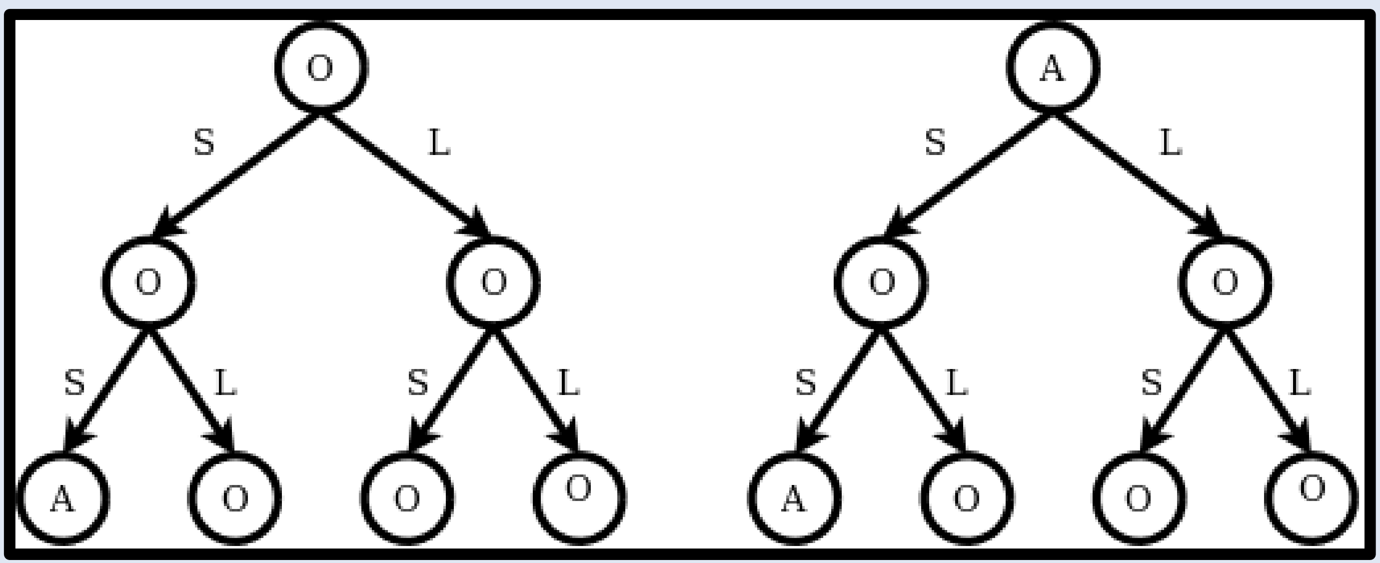
- (Smart generals make a plan in advance!)

Policies and their Values

Policy Domain

- What do policies look like?
  - In general histories → actions
  - in MDP/POMDP: more compact representations...
- Now, this is difficult: no such representation known!

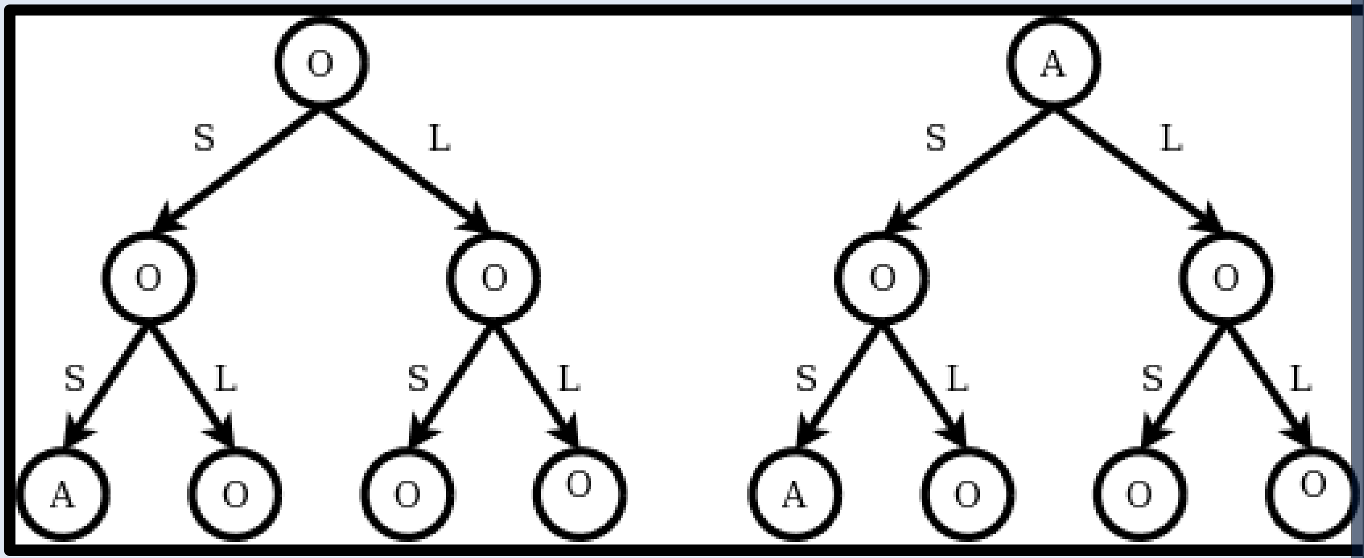
  → So we will be stuck with histories



Policy Domain

- What do policies look like?
  - In general histories → actions
  - in MDP/POMDP: more compact representations...
- Now, this is difficult: no such representation known!

  → So we will be stuck with histories



Most general, AOHs:

$$\left(a_i^0, o_i^1, a_i^1, \ldots, a_i^{t-1}, o_i^t\right)$$

But: can restrict to deterministic policies → only need OHs:

$$\vec{o}_i = \left(o_i^1, \ldots, o_i^t\right)$$

No Compact Representation?

- **Joint Belief**, *b(s)* (as in MPOMDP) [Pynadath and Tambe 2002]
  - compute b(s) using joint actions and observations
  - Problem: ?

No Compact Representation?

- **Joint Belief**, *b(s)* (as in MPOMDP) [Pynadath and Tambe 2002]
  - compute b(s) using joint actions and observations
  - Problem: agents do not know those during execution

Goal of Planning

- Find the **optimal** joint policy $\pi^* = \langle \pi_{1,} \pi_2 \rangle$
  - where individual policies map OHs to actions $\pi_i : \vec{O}_i \rightarrow A_i$

- What is the optimal one?
  - Define **value** as the expected sum of rewards:

$$V(\pi) = \boldsymbol{E}\left[\sum_{t=0}^{h-1} R(s,a) \mid \pi, b^0\right]$$

  - optimal joint policy is one with maximal value (can be more that achieve this)

Goal of Planning

Optimal policy for 2 generals, h=3

value=-2.86743

() --> observe
(o_small) --> observe
(o_large) --> observe
(o_small,o_small) --> attack
(o_small,o_large) --> attack
(o_large,o_small) --> attack
(o_large,o_large) --> observe

() --> observe
(o_small) --> observe
(o_large) --> observe
(o_small,o_small) --> attack
(o_small,o_large) --> attack
(o_large,o_small) --> attack
(o_large,o_large) --> observe

Goal of Planning

Optimal policy for 2 generals, h=3

value=-2.86743

() --> observe
(o_small) --> observe
(o_large) --> observe
(o_small,o_small) --> attack
(o_small,o_large) --> attack
(o_large,o_small) --> attack
(o_large,o_large) --> observe

() --> observe
(o_small) --> observe
(o_large) --> observe
(o_small,o_small) --> attack
(o_small,o_large) --> attack
(o_large,o_small) --> attack
(o_large,o_large) --> observe

conceptually:

what should policy optimize to allow for good coordination (thus high value)

?

Coordination vs. Exploitation of Local Information

- Inherent trade-off

  coordination vs. **exploitation of local information**

  - Ignore own observations → 'open loop plan'
    - E.g., “ATTACK on 2nd time step”

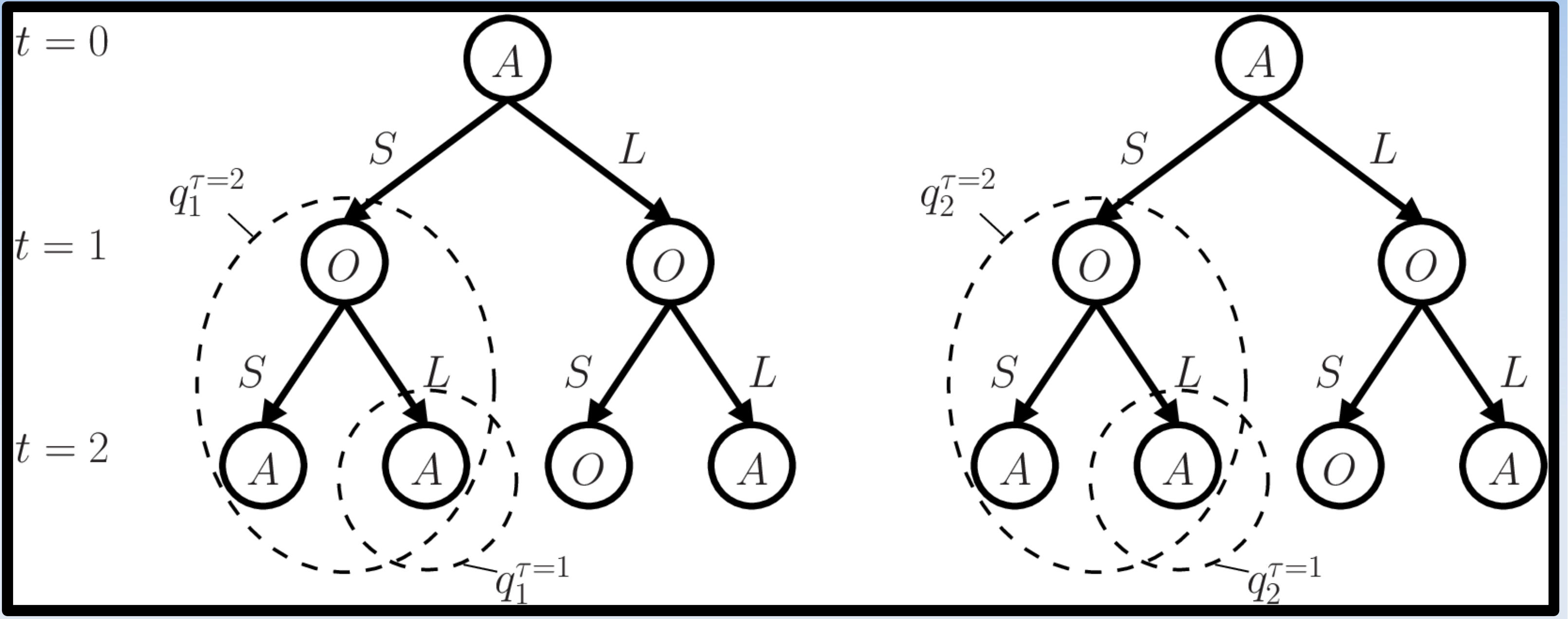
      \+ maximally predictable
      \- low quality
  - Ignore coordination → 'MPOMDP plan'
    - E.g., 'individual belief' $b_i(s)$ and execute the MPOMDP policy

      \+ uses local information
      \- likely to result in mis-coordination

- **Optimal policy π^* should balance between these!**

Value of a Joint Policy

- Sub-tree policies:



- Given a particular joint policy $\pi = q^{\tau=h}$

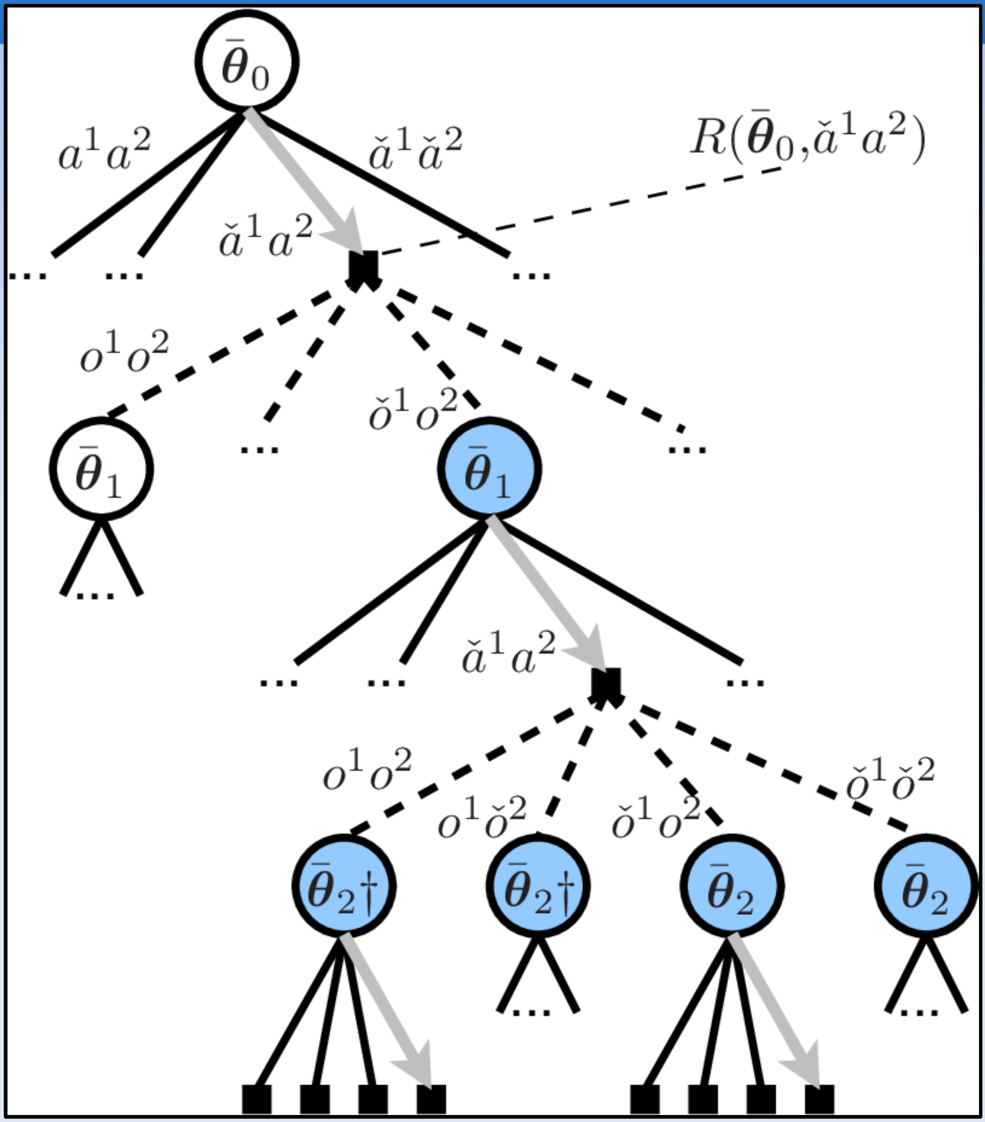
  $\rightarrow$ Just a (complex) Markov Chain

- Value:

$$V(\vec{\theta}, q^{\tau=k}) = R(\vec{\theta}, a) + \sum_o P(o|\vec{\theta}, a) V(\vec{\theta}', q^{\tau=k-1})$$

Optimal Value Functions – 1

- Optimal value functions are difficult!
- Consider selecting the best joint sub-tree policy q^τ



- We *can* compute value...
  ...but *cannot* select the maximizing q^τ independently!

Optimal Value Functions – 2

- *Cannot* select the maximizing q^τ independently...

  → Need to reason over assignment for all AOHs of a stage *t* simultaneously!

- Value stage t $\sum_{\theta} P(\theta|b^{0,}\varphi) V(\theta, q^{\tau=h-t})$

$$= \sum_{\langle\theta_{1,}\theta_2\rangle} P(\langle\theta_{1,}\theta_2\rangle|b^{0,}\varphi) V(\langle\theta_{1,}\theta_2\rangle, \langle q_{1,} q_2\rangle)$$

- Find mappings $\Gamma_{1,}\Gamma_2$ (from AOHs → sub-tree policies)

  that maximize

$$\sum_{\langle\theta_{1,}\theta_2\rangle} \underbrace{P(\langle\theta_{1,}\theta_2\rangle|b^{0,}\varphi)}_{\text{dependence on history}} \underbrace{V(\langle\theta_{1,}\theta_2\rangle, \langle\Gamma_1(\theta_1), \Gamma_2(\theta_2)\rangle)}_{\text{dependence on future}}$$

Planning Methods

Brute Force Search

- We can compute the value of a joint policy $V(\pi)$
- So the **stupidest algorithm** is:
  - compute $V(\pi)$, for all π
  - select a π with maximum value
- Number of joint policies is huge!
  (doubly exponential in horizon h)
- Clearly intractable...

h	num. joint policies
1	4
2	64
3	16384
4	1.0737e+09
5	4.6117e+18
6	8.5071e+37
7	2.8948e+76
8	3.3520e+153

Brute Force Search

- We can compute the value of a joint policy $V(\pi)$

No easy way out...

The problem is
NEXP-complete [Bernstein et al. 2002]

most likely (assuming EXP != NEXP)
doubly exponential time required.

(doubly exponential in horizon h)

- Clearly intractable...

h	num. joint policies
1	4
2	64
3	16384
4	1.0737e+09
5	4.6117e+18
6	8.5071e+37
7	2.8948e+76
8	3.3520e+153

Brute Force Search

- We can compute the value of a joint policy $V(\pi)$
- So the simple algorithm is:
  - compute $V(\pi)$ for all π
- Number of joint policies is huge! (doubly exponential in horizon h)
- Clearly intractable...

h	num. joint policies
1	4
2	64
3	16384
4	1.0737e+09
5	4.6117e+18
6	8.5071e+37
7	2.8948e+76
8	3.3520e+153

No easy way out...

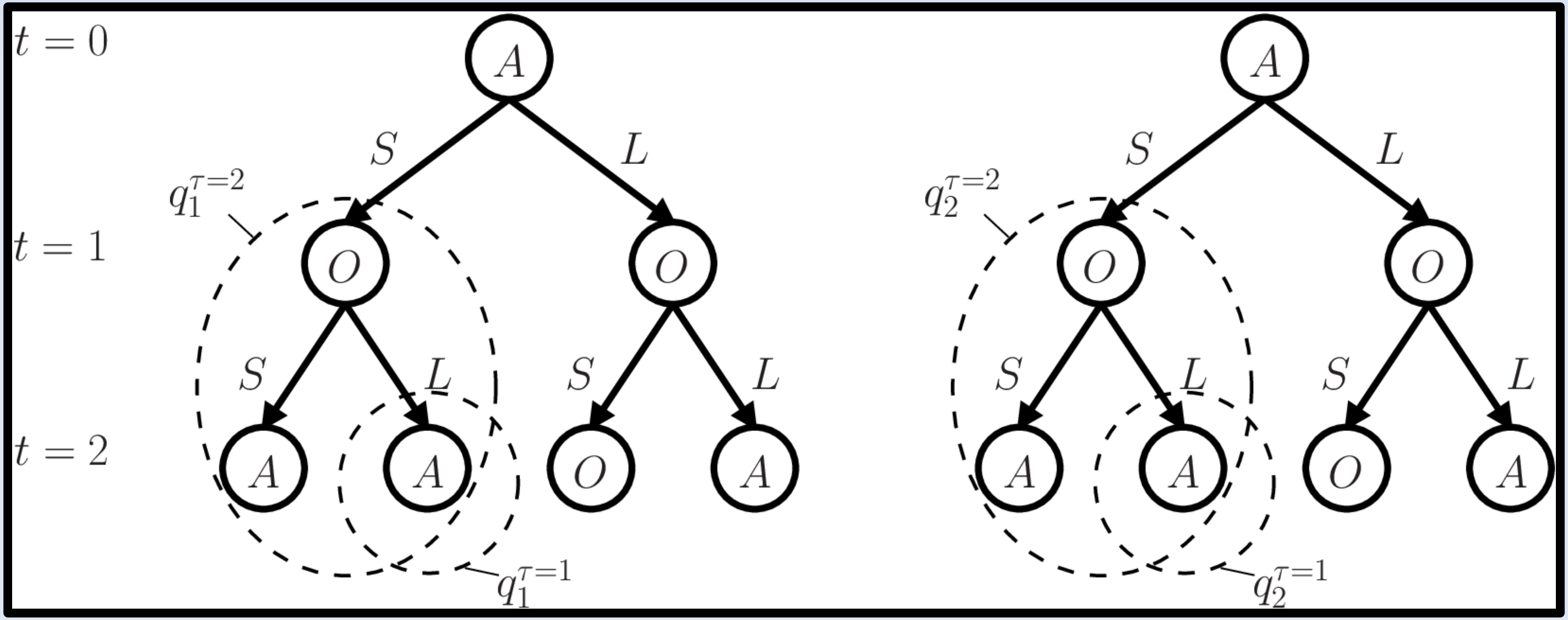
The problem is **NEXP-complete** [Bernstein et al. 2002]

most likely (assuming EXP != NEXP) doubly exponential time required.

- Still, there are better algorithms that work better for at least some problems...
- Useful to gain understanding about problem.

Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - construct all 2-stages-to-go policies $Q^{\tau=2}$, etc.



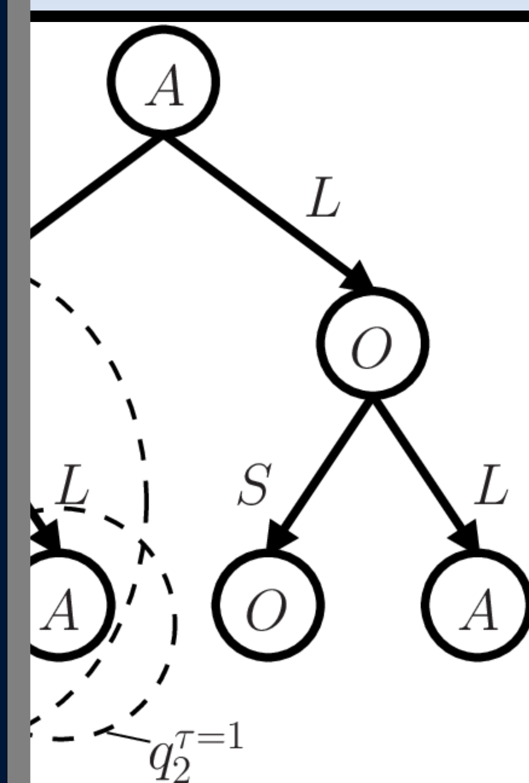
Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - ... etc.

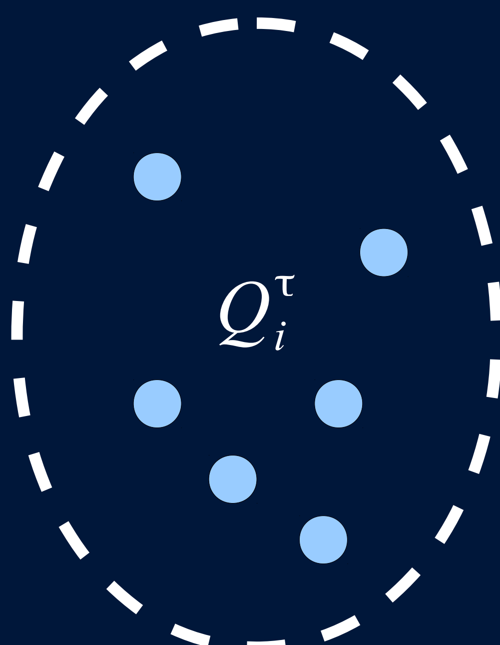
Exhaustive backup operation

Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - etc.

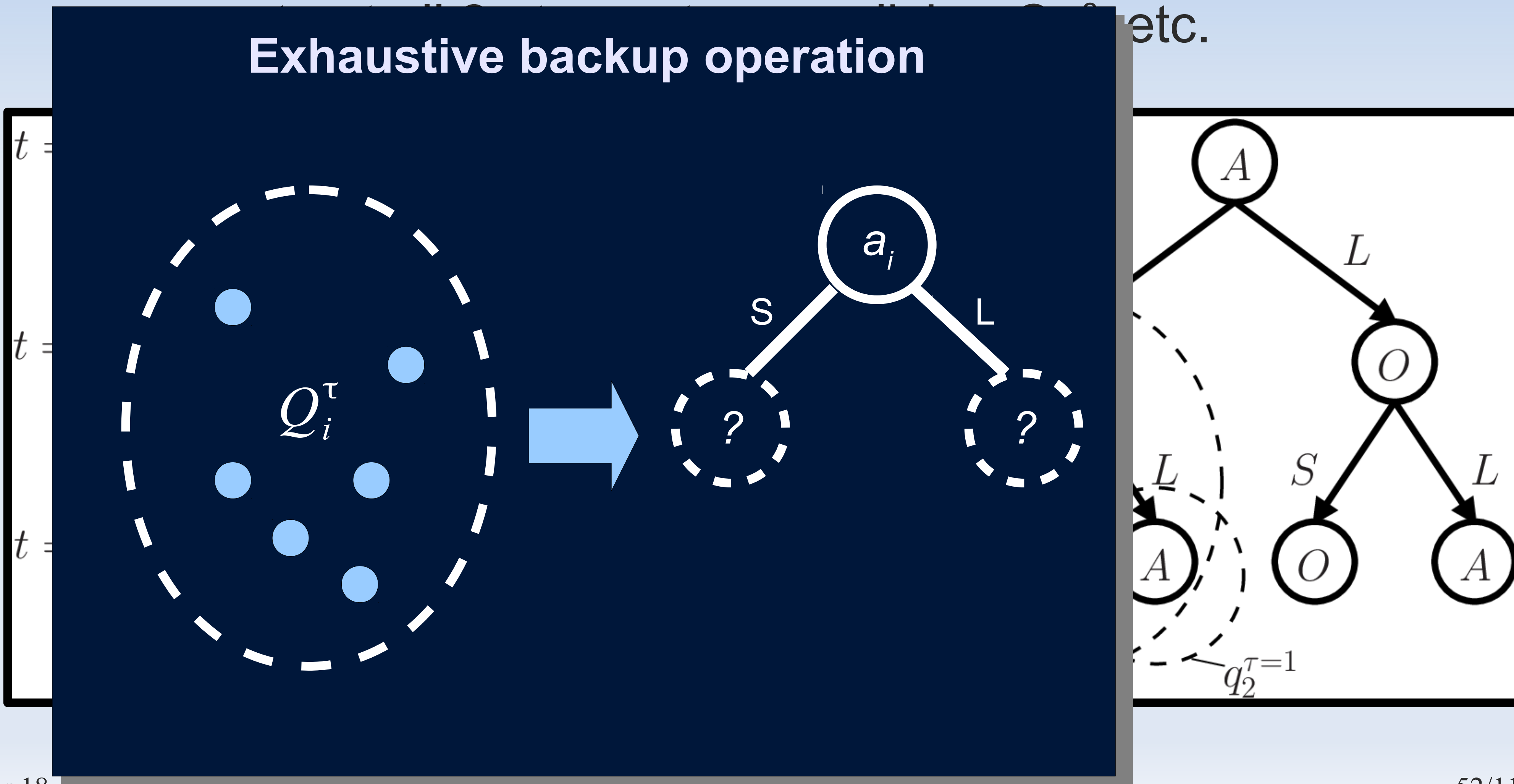


Exhaustive backup operation



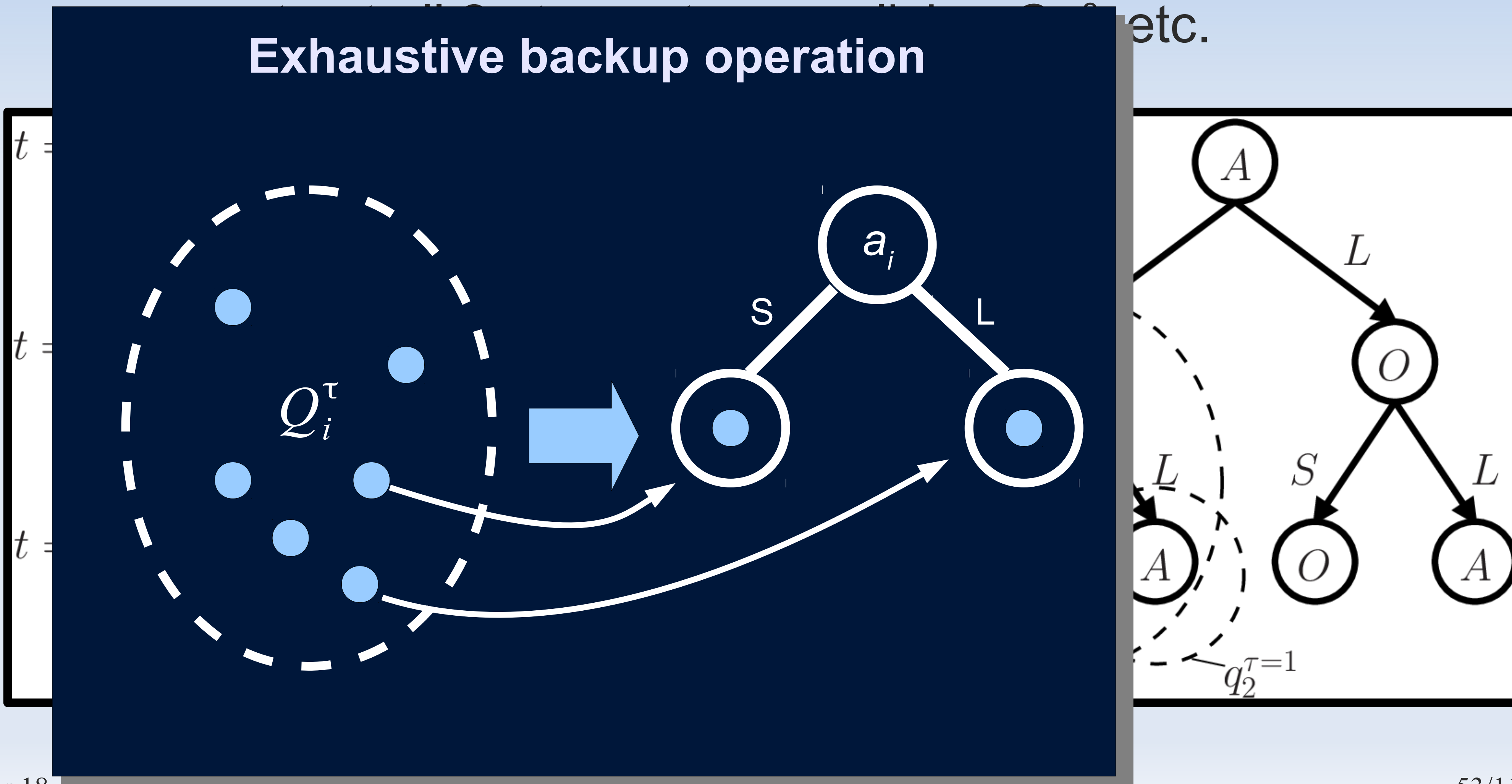
Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - … etc.



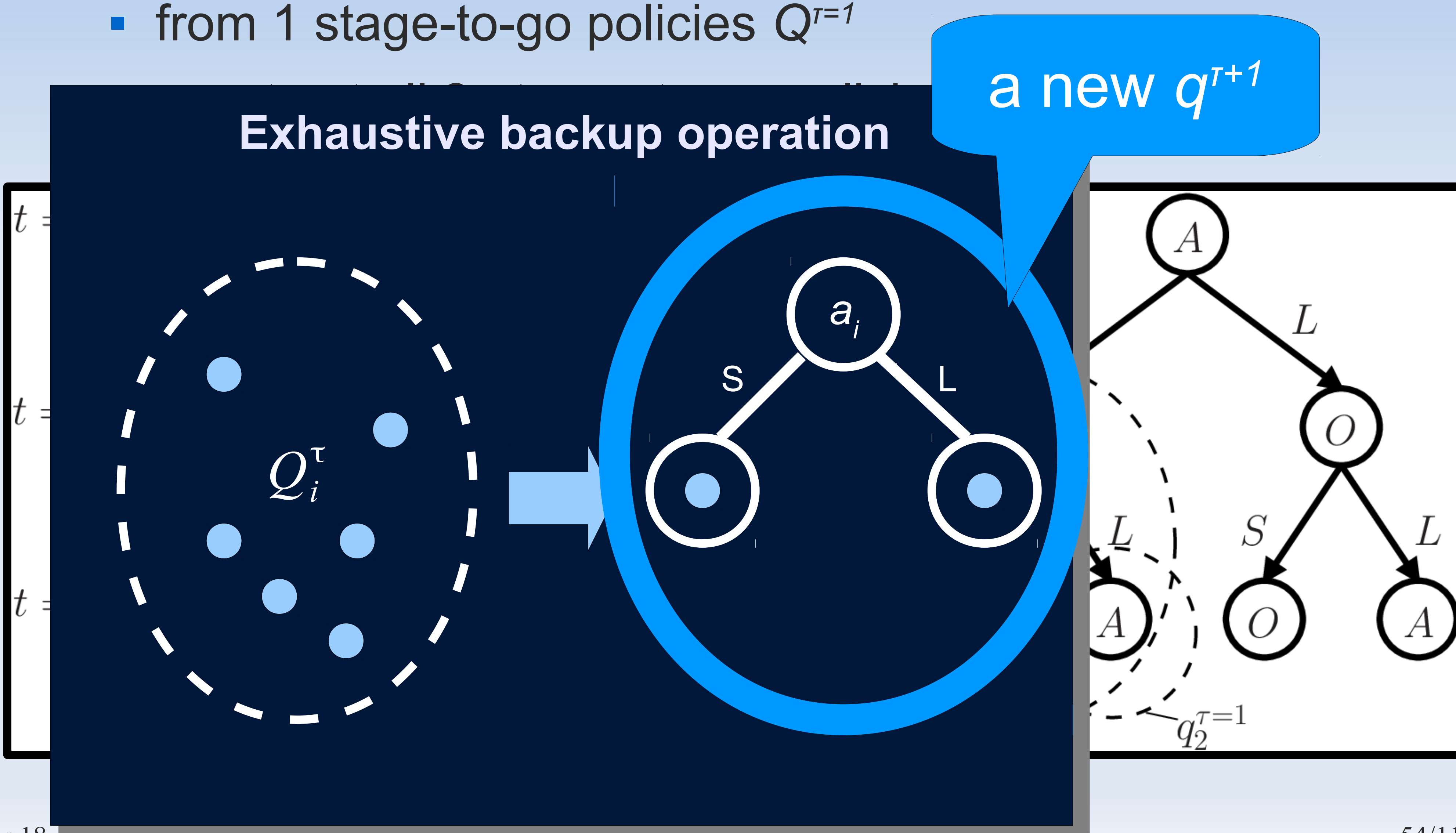
Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - etc.



Dynamic Programming – 1

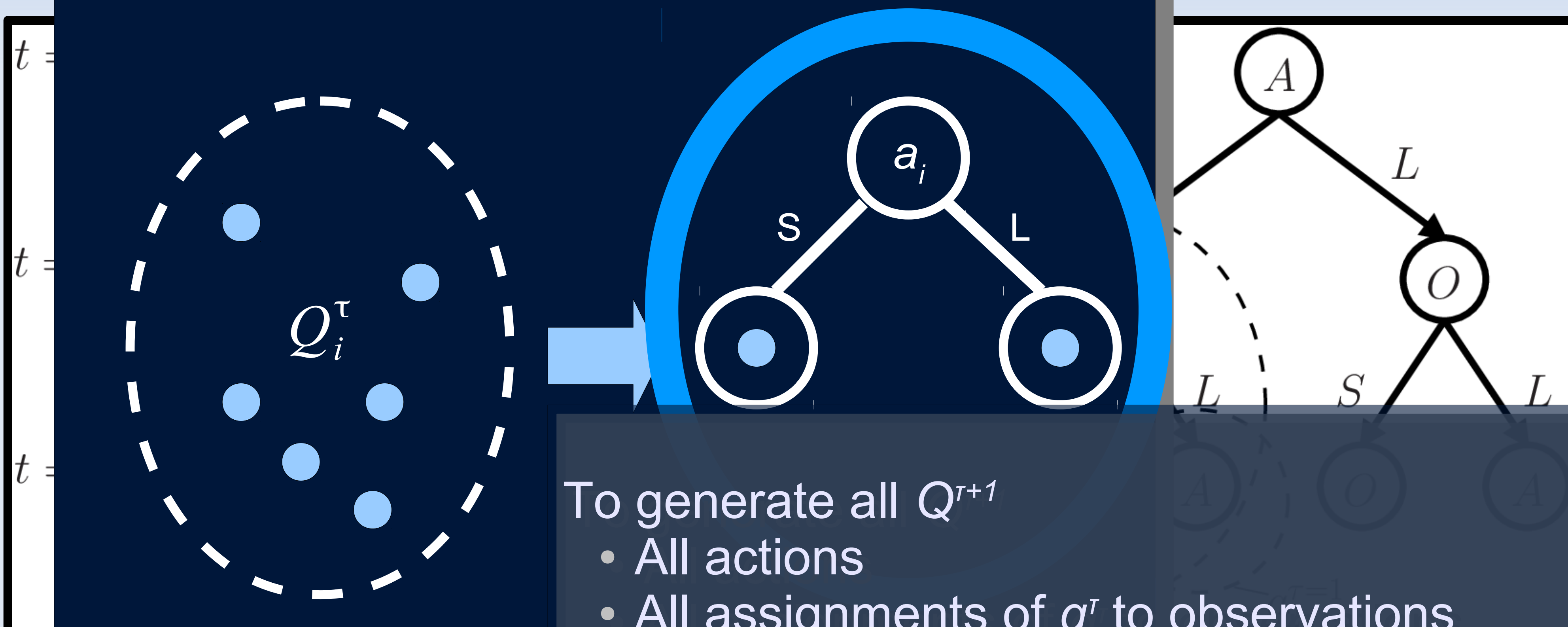
- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$



Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - etc.

Exhaustive backup operation



To generate all $Q^{\tau+1}$
- All actions
- All assignments of q^τ to observations

Dynamic Programming – 2

- (obviously) this scales very poorly...

$Q_1^{\tau=1}$

A O

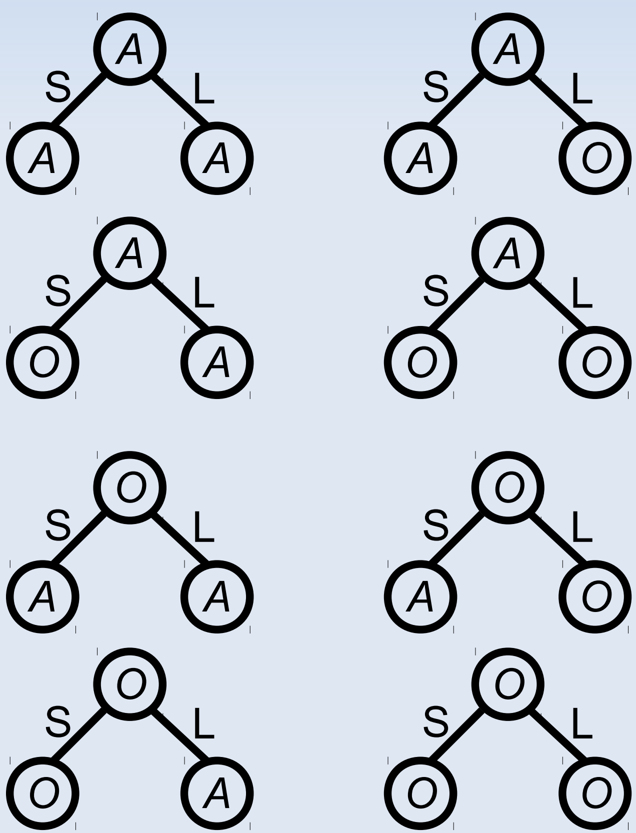
$Q_2^{\tau=1}$

A O

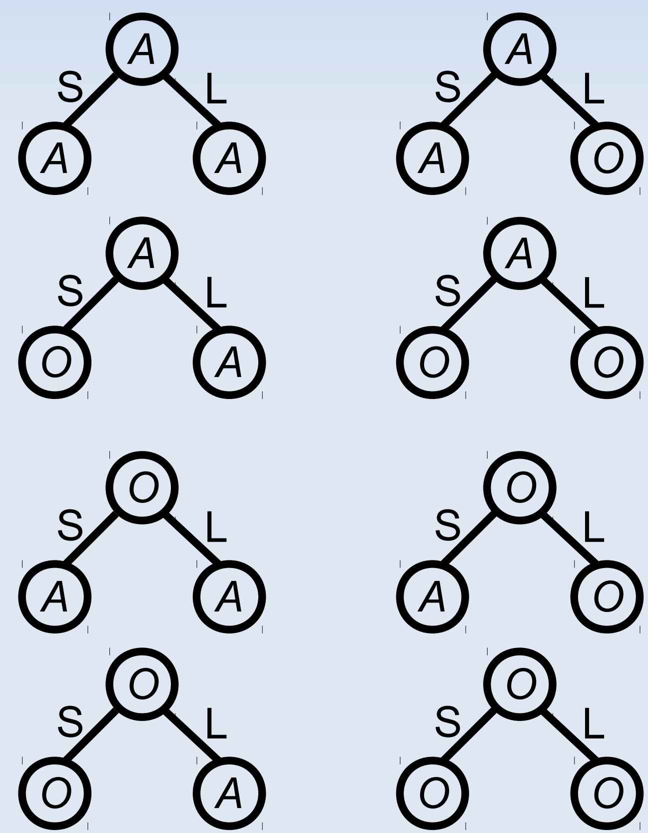
Dynamic Programming – 2

- (obviously) this scales very poorly...

$Q_1^{\tau=2}$



$Q_2^{\tau=2}$



Dynamic Programming – 2

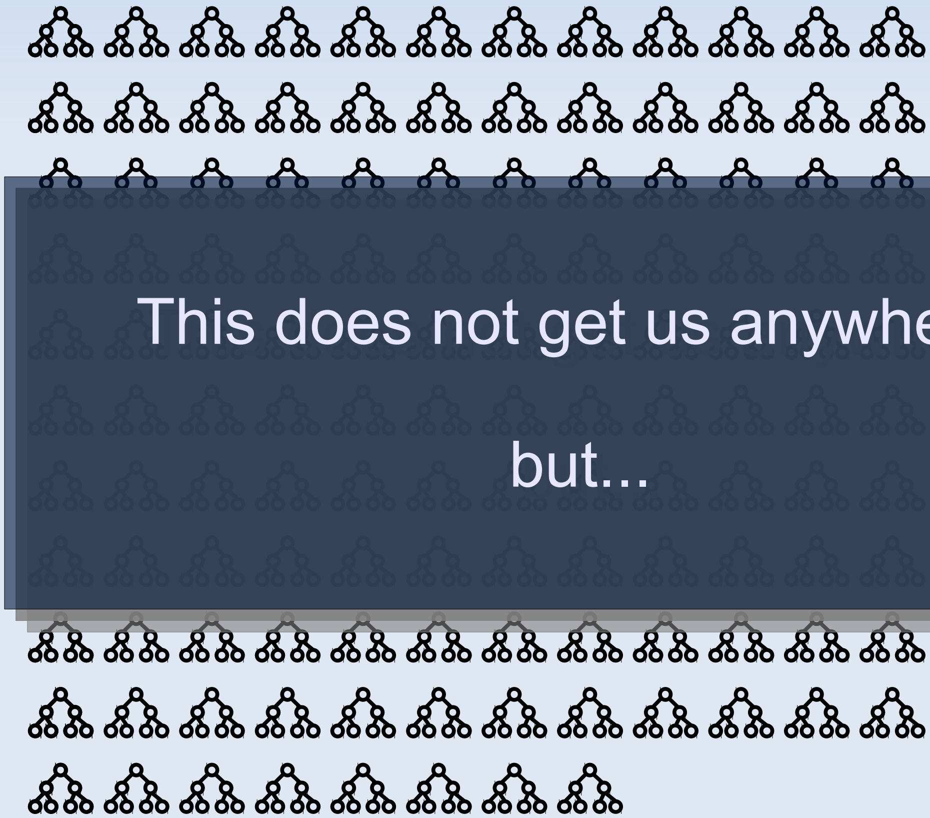
- (obviously) this scales very poorly...

$Q_1^{\tau=3}$ $Q_2^{\tau=3}$

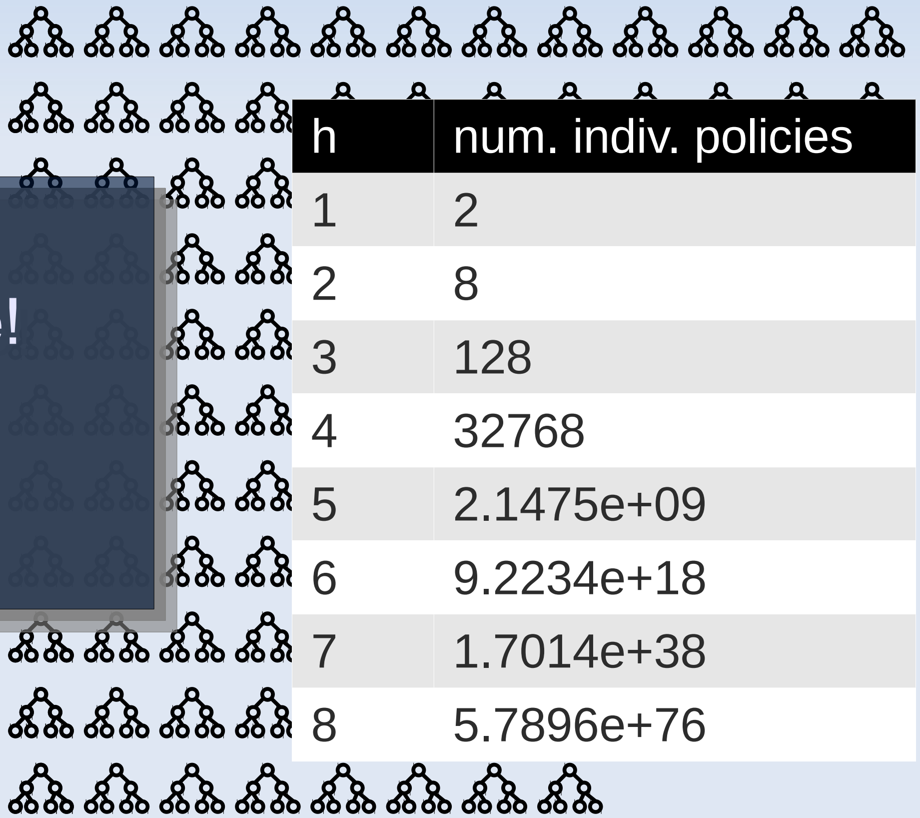
Dynamic Programming – 2

- (obviously) this scales very poorly...

$Q_1^{\tau=3}$



$Q_2^{\tau=3}$



h	num. indiv. policies
1	2
2	8
3	128
4	32768
5	2.1475e+09
6	9.2234e+18
7	1.7014e+38
8	5.7896e+76

This does not get us anywhere!

but...

Dynamic Programming – 3

- Perhaps not all those Q_i^τ are useful!
  - Perform **pruning** of 'dominated policies'!

- Algorithm [Hansen et al. 2004]

$$Q_i^{\tau=1} = A_i$$

```
Initialize Q1(1), Q2(1)
for tau=2 to h
  Q1(tau) = ExhaustiveBackup(Q1(tau-1))
  Q2(tau) = ExhaustiveBackup(Q2(tau-1))
  Prune(Q1,Q2,tau)
end
```

Dynamic Programming – 3

- Perhaps not all those Q_i^τ are useful!
  - Perform **pruning** of 'dominated policies'!

$$Q_i^{\tau=1} = A_i$$

- Algorithm [Hansen et al. 2004]

```
Initialize Q1(1), Q2(1)
for tau=2 to h
  Q1(tau) = ExhaustiveBackup(Q1(tau-1))
  Q2(tau) = ExhaustiveBackup(Q2(tau-1))
  Prune(Q1,Q2,tau)
end
```

Note: cannot prune independently!

- usefulness of a q_1 depends on Q_2
- and vice versa
  → **Iterated elimination** of policies

Dynamic Programming – 3

- Perhaps not all those Q_i^τ are useful!
  - Perform **pruning** of 'dominated policies'!

- Algorithm [Hansen et al. 2004]

$$Q_i^{\tau=1} = A_i$$

```
Initialize Q1(1), Q2(1)
for tau=2 to h
  Q1(tau) = ExhaustiveBackup(Q1(tau-1))
  Q2(tau) = ExhaustiveBackup(Q2(tau-1))
  Prune(Q1,Q2,tau)
end
```

Note: cannot prune independently!

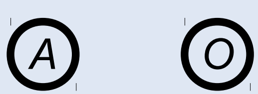
- usefulness of a q_1 depends on Q_2
- and vice versa
  → **Iterated elimination** of policies

pruning itself:
via LP [Hansen et al. 2000]

Dynamic Programming – 4

- Initialization

$Q_1^{\tau=1}$

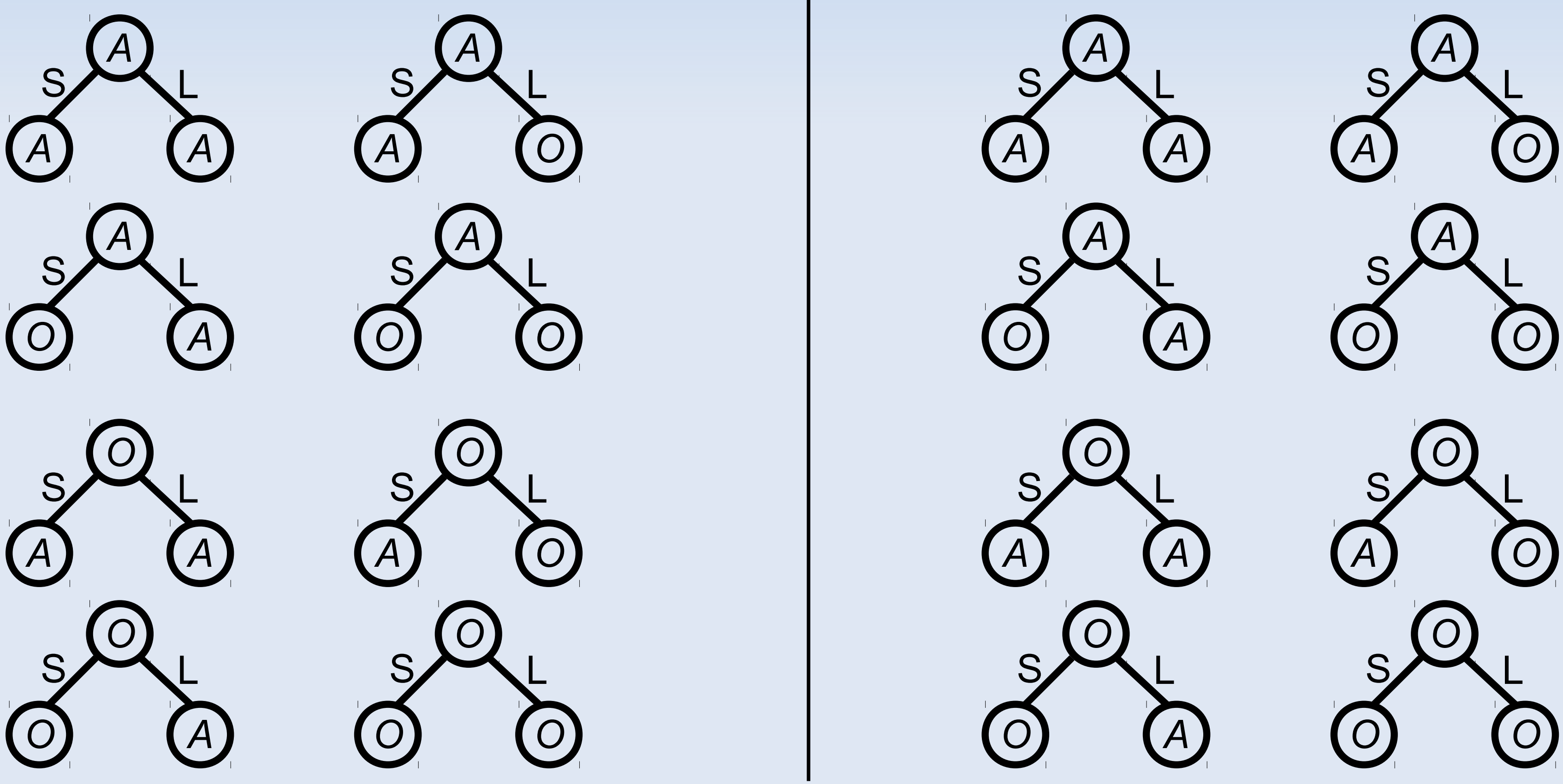


$Q_2^{\tau=1}$

Dynamic Programming – 4

- Exhaustive Backups gives

$Q_1^{\tau=2}$ | $Q_2^{\tau=2}$

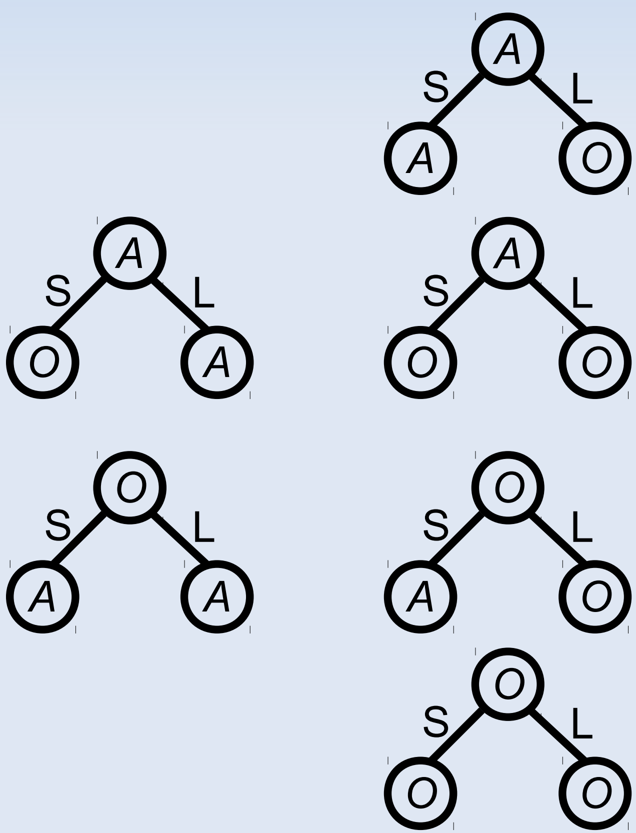


Dynamic Programming – 4

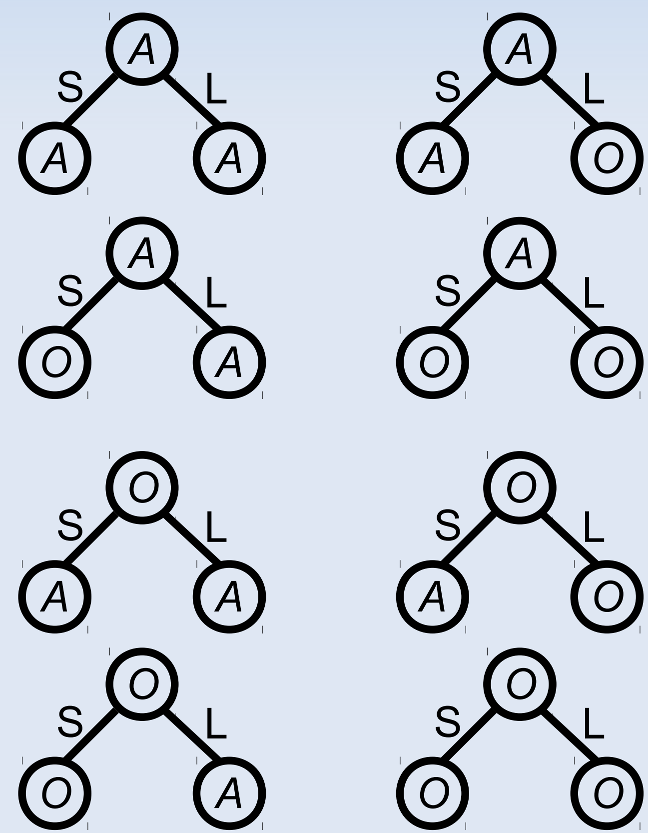
- Pruning agent 1...

Hypothetical Pruning
(not the result of actual pruning)

$Q_1^{\tau=2}$

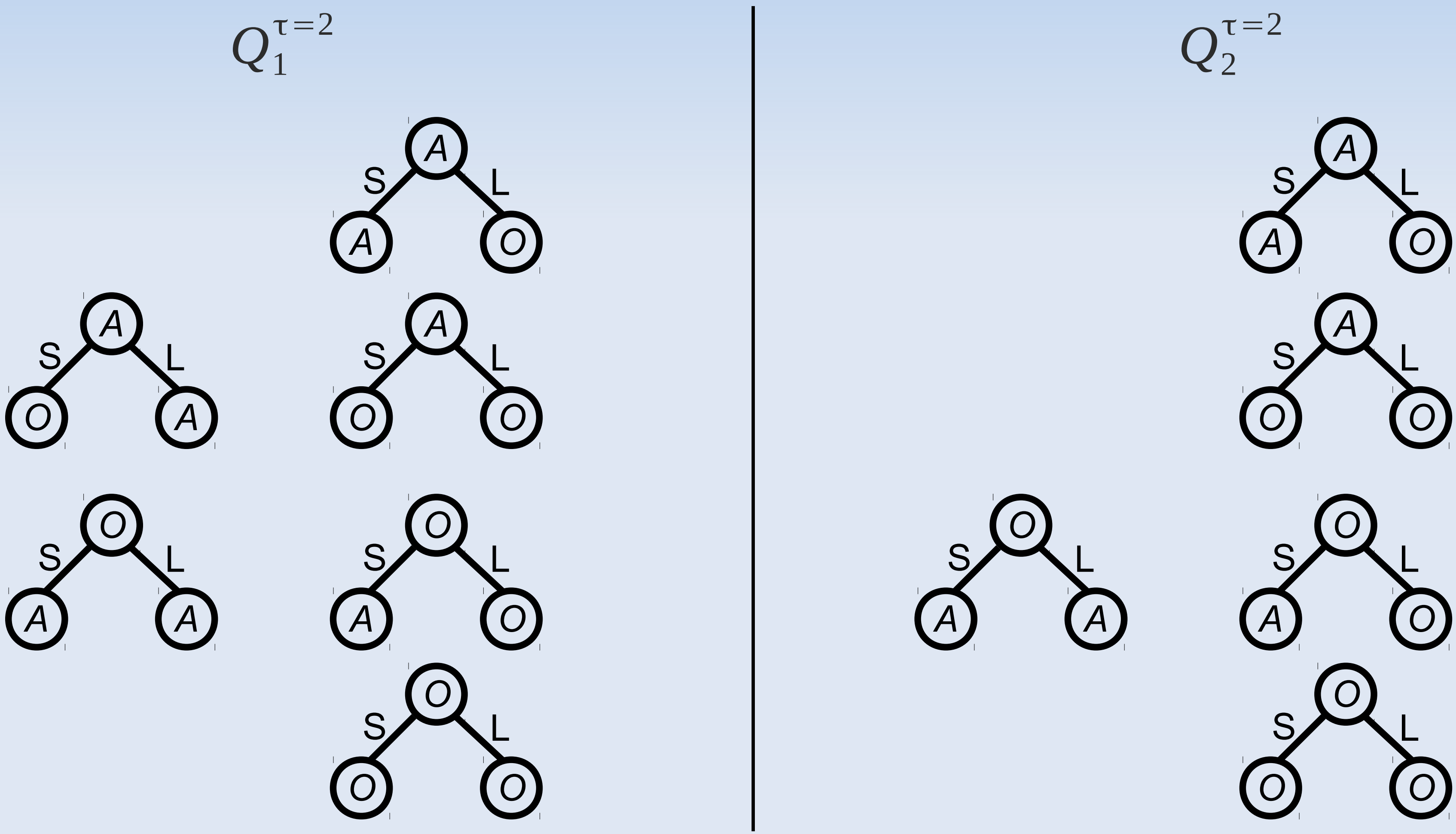


$Q_2^{\tau=2}$



Dynamic Programming – 4

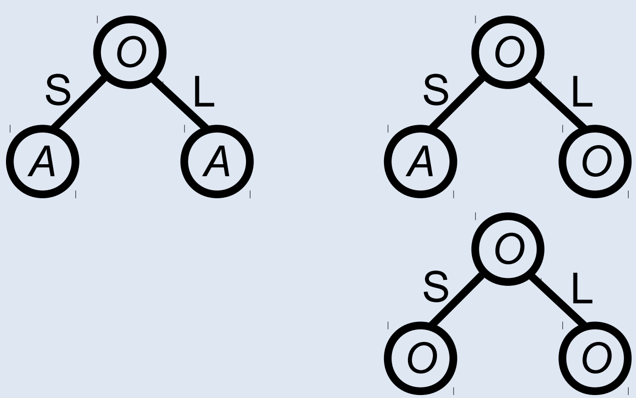
- Pruning agent 2...



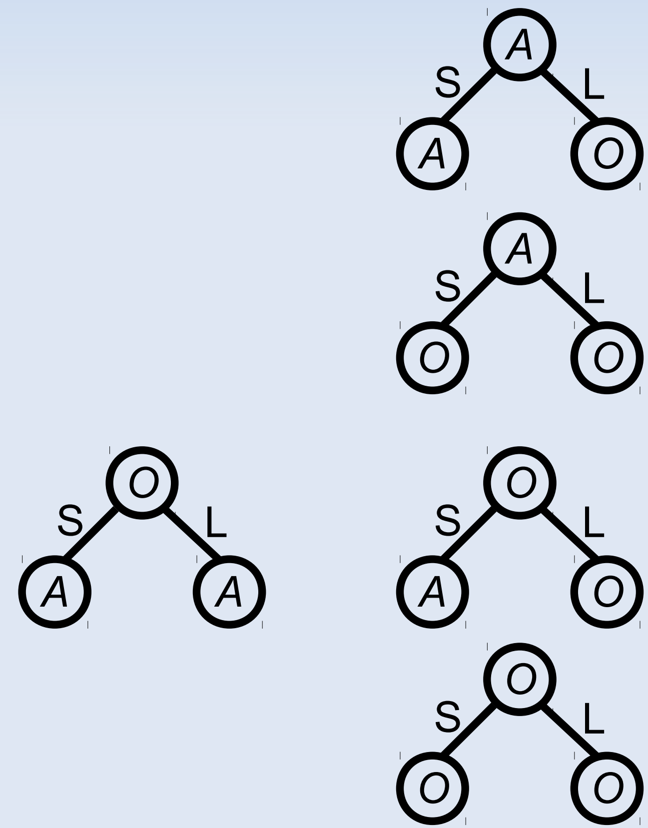
Dynamic Programming – 4

- Pruning agent 1...

$Q_1^{\tau=2}$



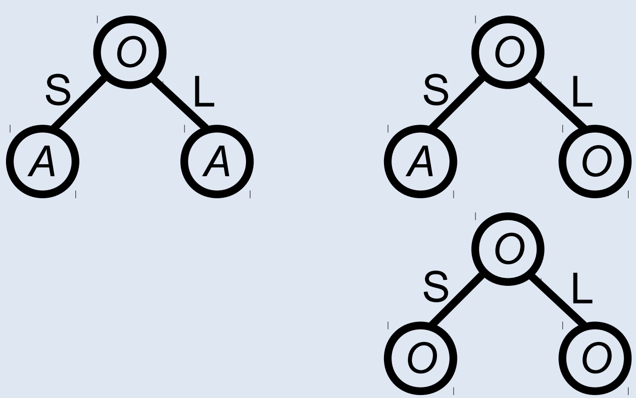
$Q_2^{\tau=2}$



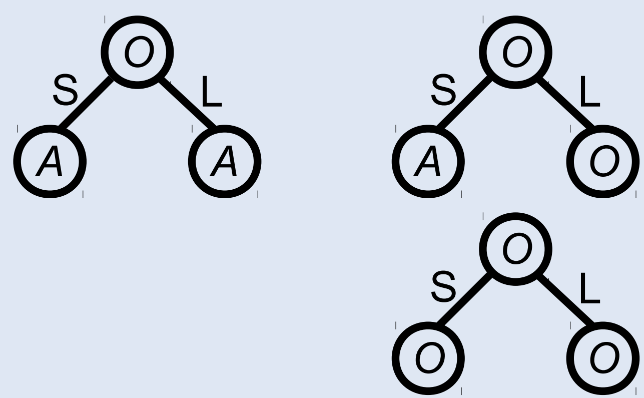
Dynamic Programming – 4

- Etc...

$Q_1^{\tau=2}$



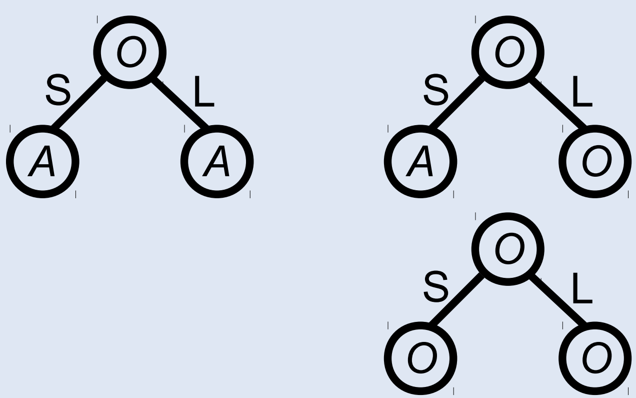
$Q_2^{\tau=2}$



Dynamic Programming – 4

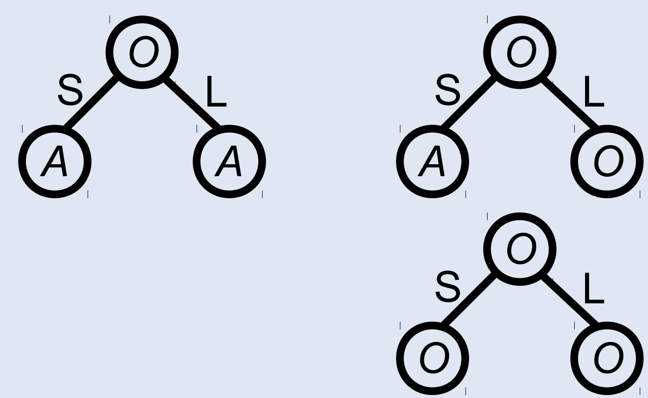
- Etc...

$Q_1^{\tau=2}$



$Q_2^{\tau=2}$

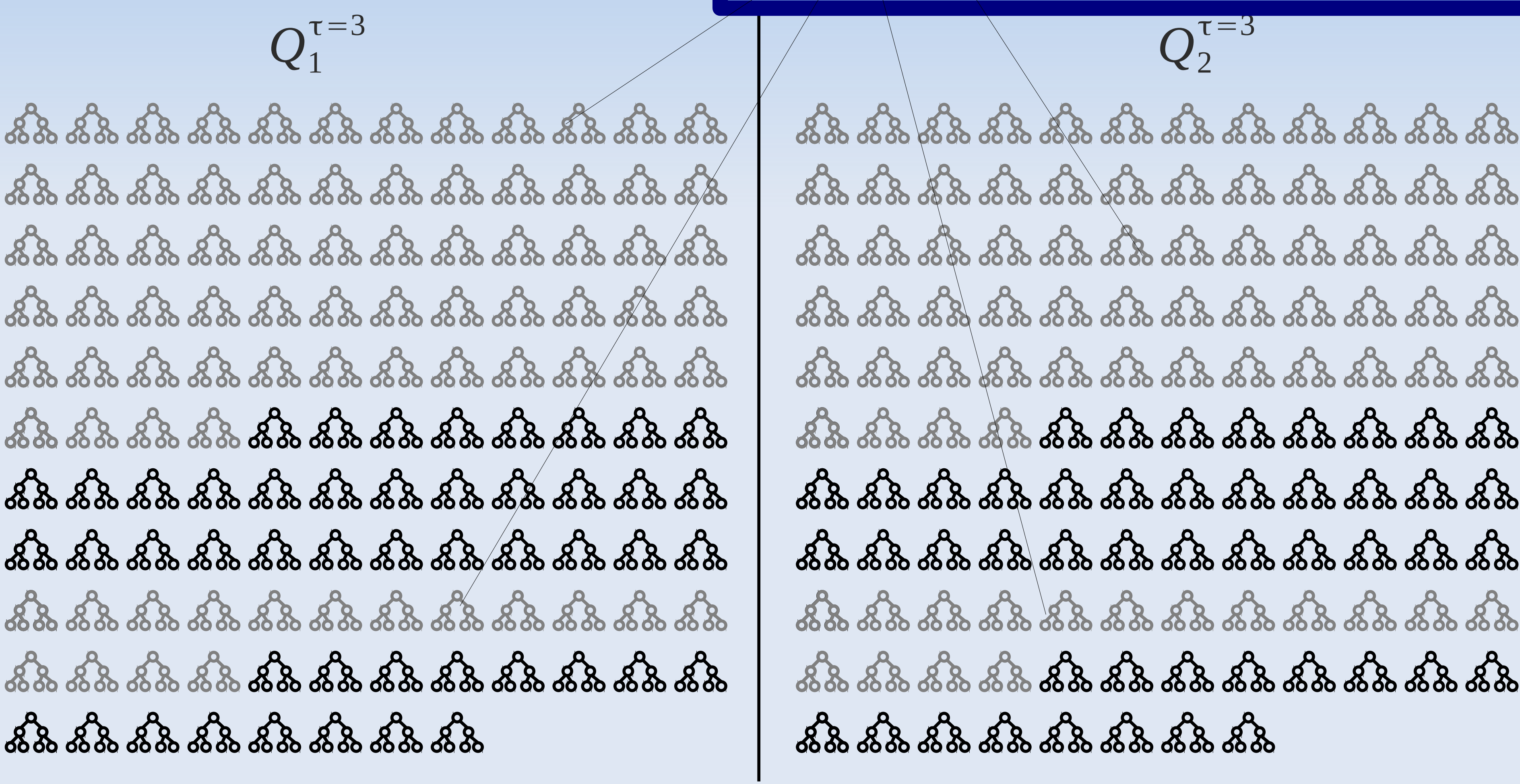
In this case: symmetric
→ but need not be in general!



Dynamic Programming – 4

- Exhaustive backups:

We **avoid** generation of many policies!



Dynamic Programming – 4

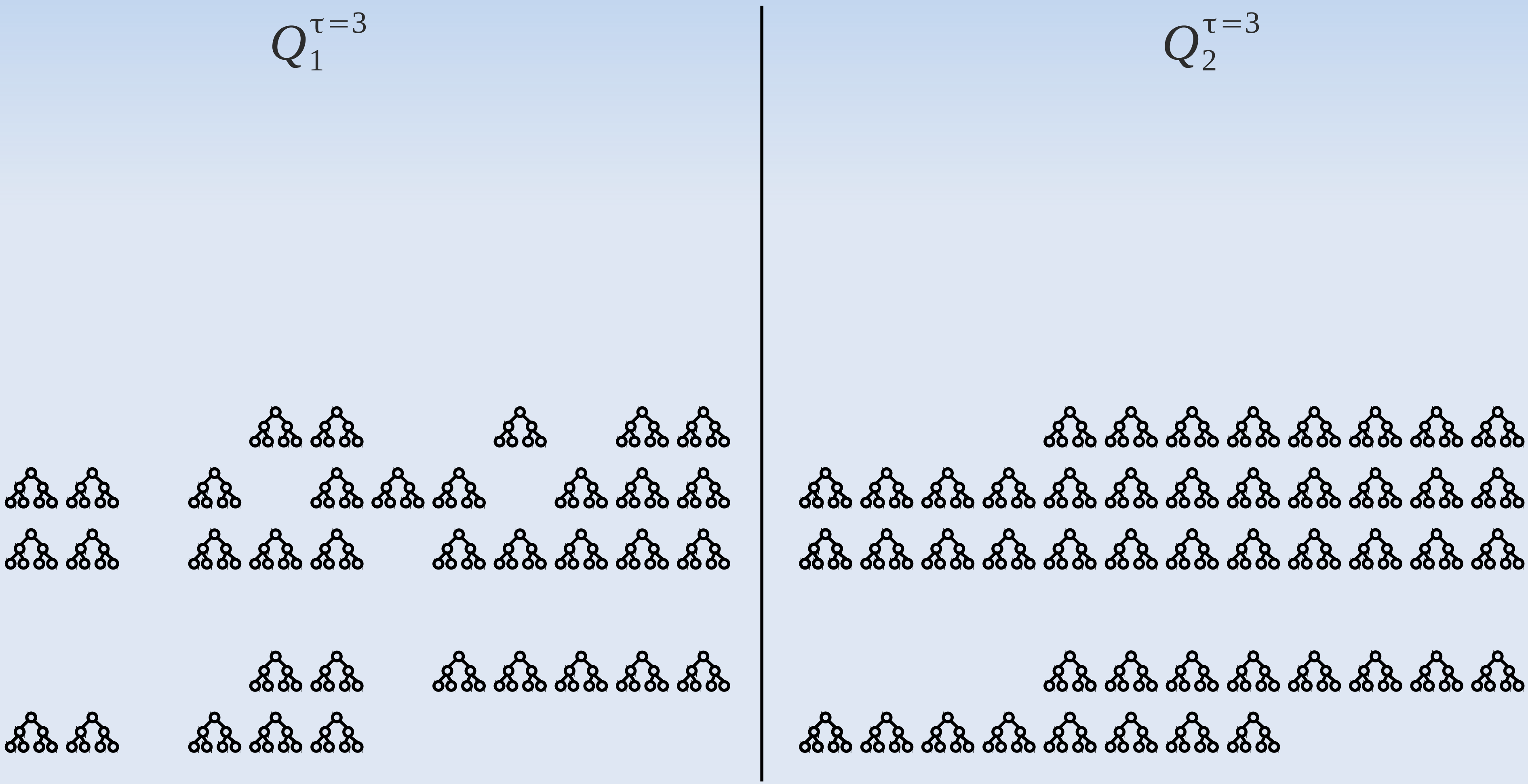
- Exhaustive backups:

$Q_1^{\tau=3}$

$Q_2^{\tau=3}$

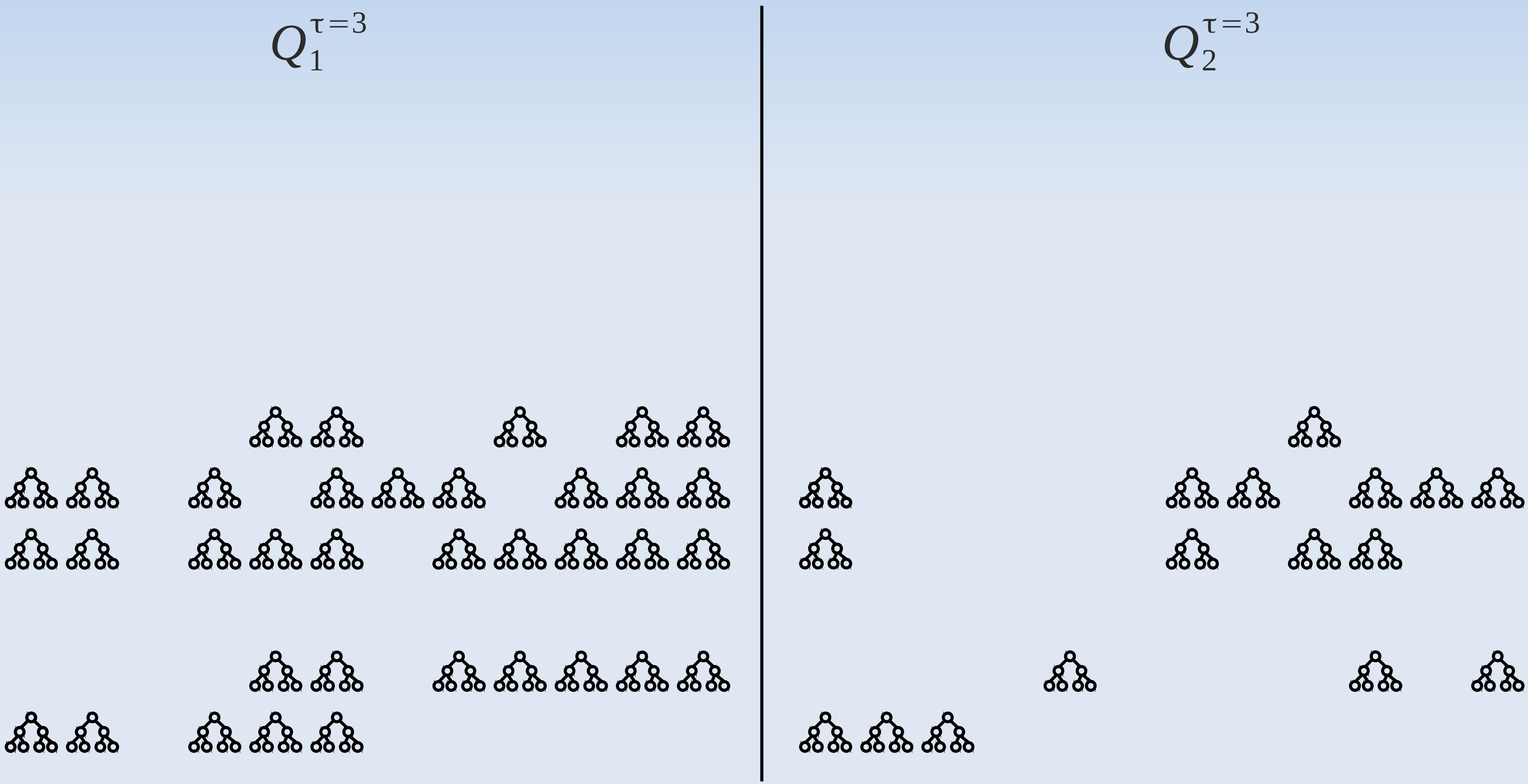
Dynamic Programming – 4

- Pruning agent 1...



Dynamic Programming – 4

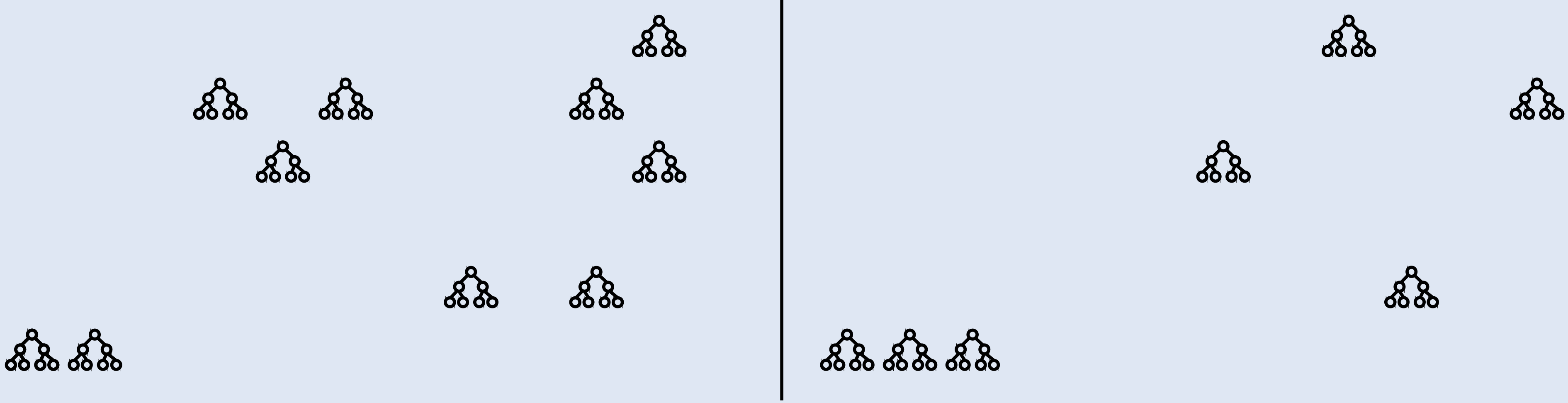
- Pruning agent 2...



Dynamic Programming – 4

- Etc...

$Q_1^{\tau=3}$ $Q_2^{\tau=3}$

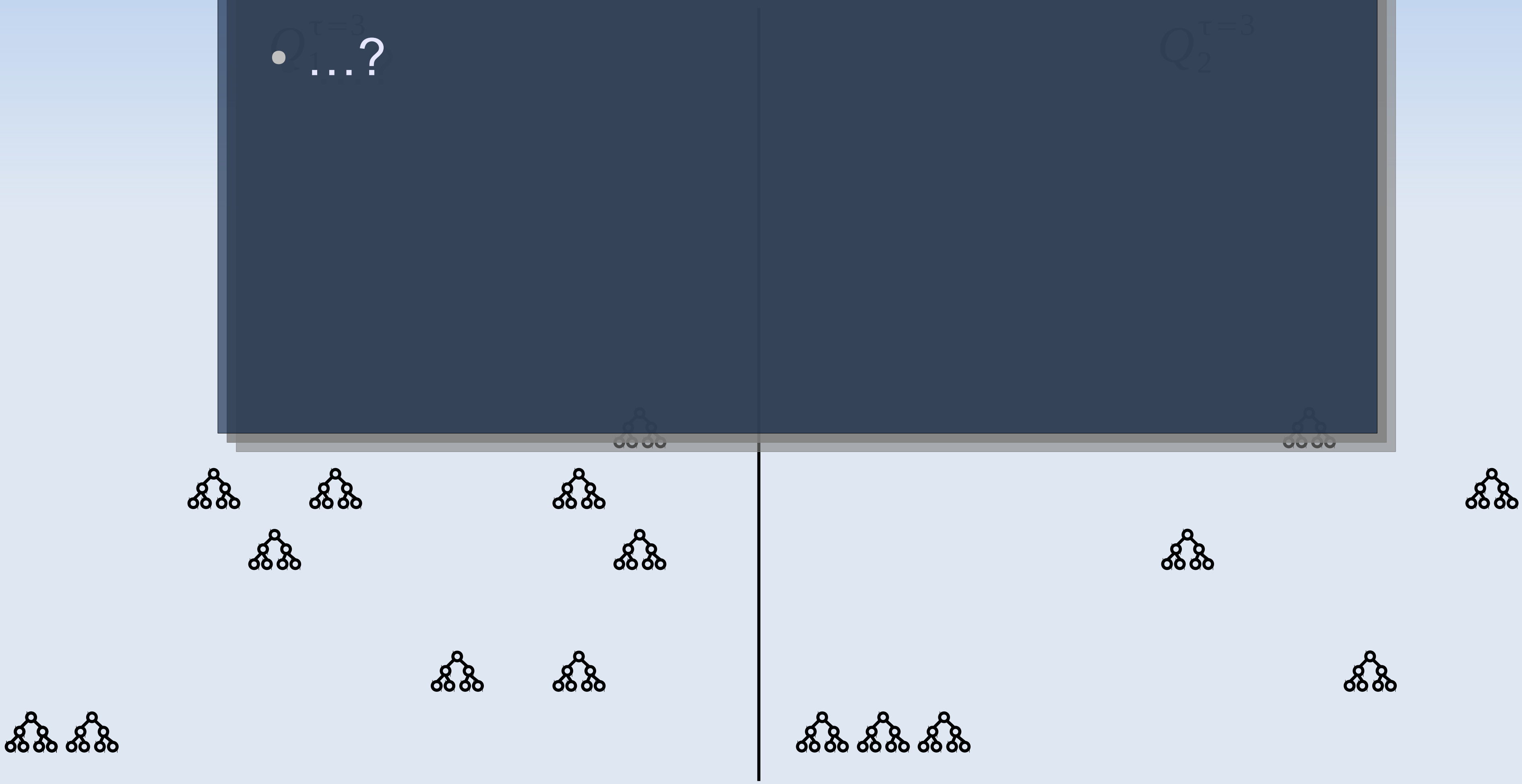


Dynamic Programming – 4

- Etc...

At the very end:

- ...?



Dynamic Programming – 4

- Etc...

At the very end:

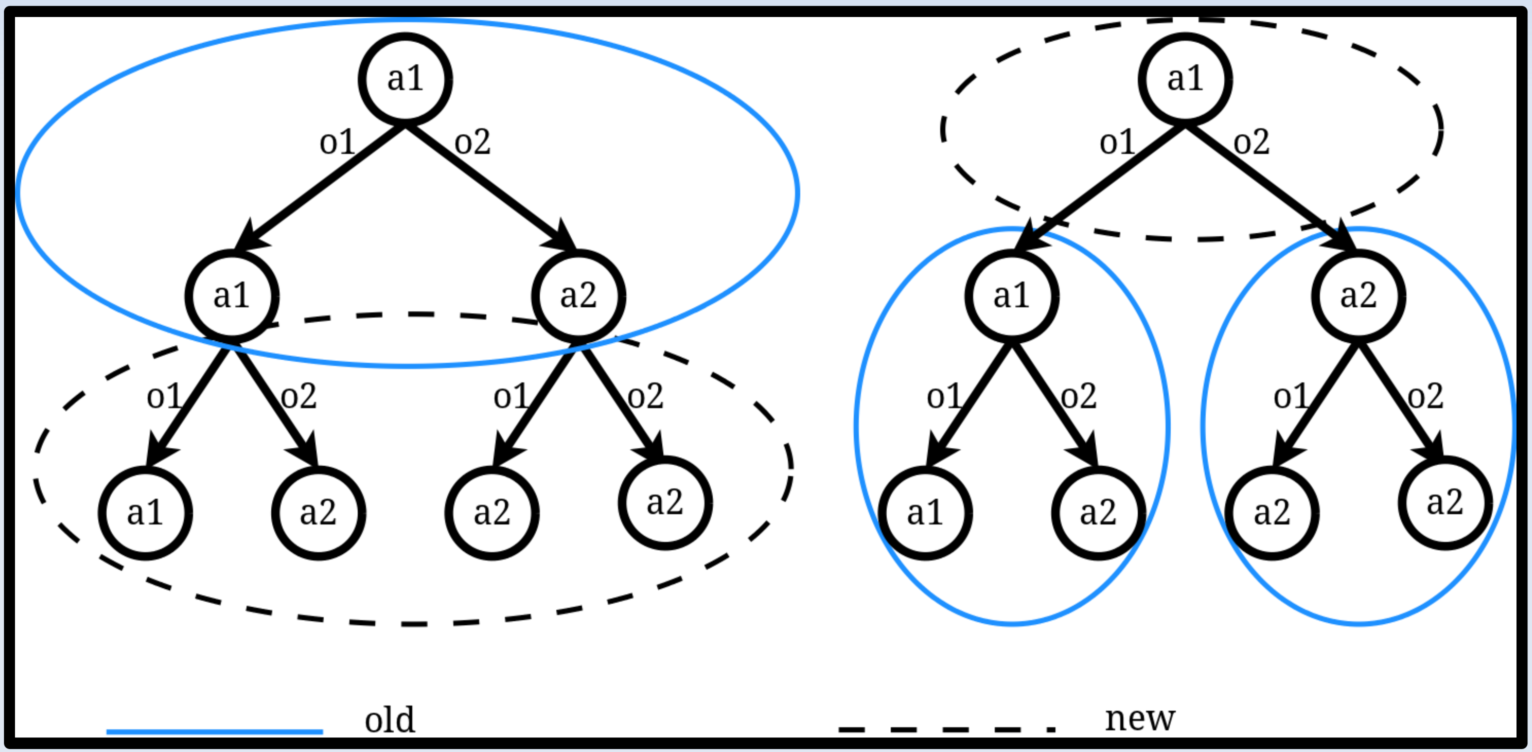
- evaluate all the remaining combinations of policies
- select the best one

$$V(q^{\tau=h}) = \sum_s b^0(s) V(s, q^{\tau=h})$$

Bottom-up vs. Top-down

- DP constructs bottom-up
- Alternatively try and construct top down

  → leads to (heuristic) search [Szer et al. 2005, Oliehoek et al. 2008]

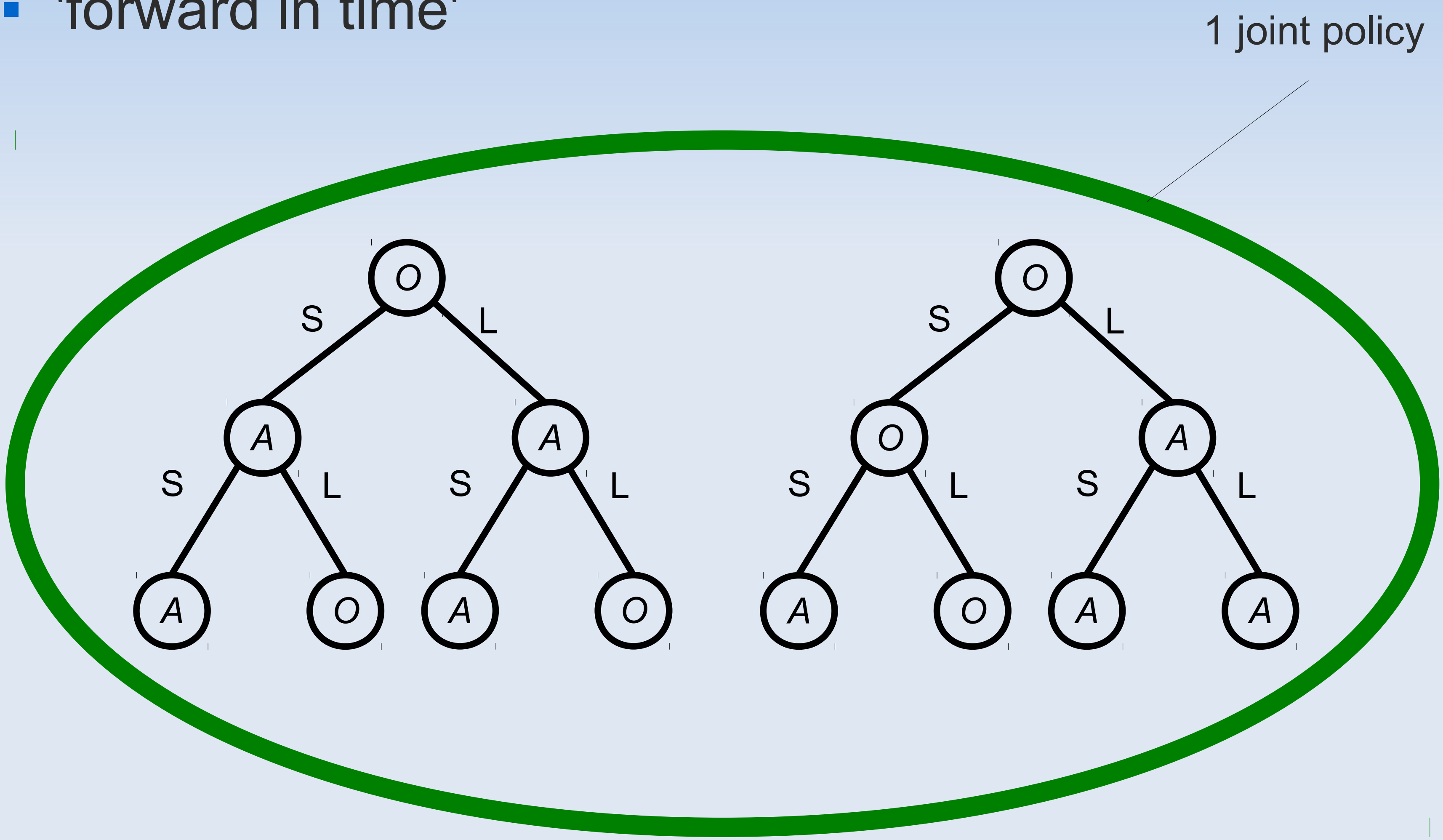


Heuristic Search – Intro

- Core idea is the same as DP:
  - incrementally construct all (joint) policies
  - try to avoid work
- Differences
  - different starting point and increments
  - use **heuristics** (rather than pruning) to avoid work

Heuristic Search – 1

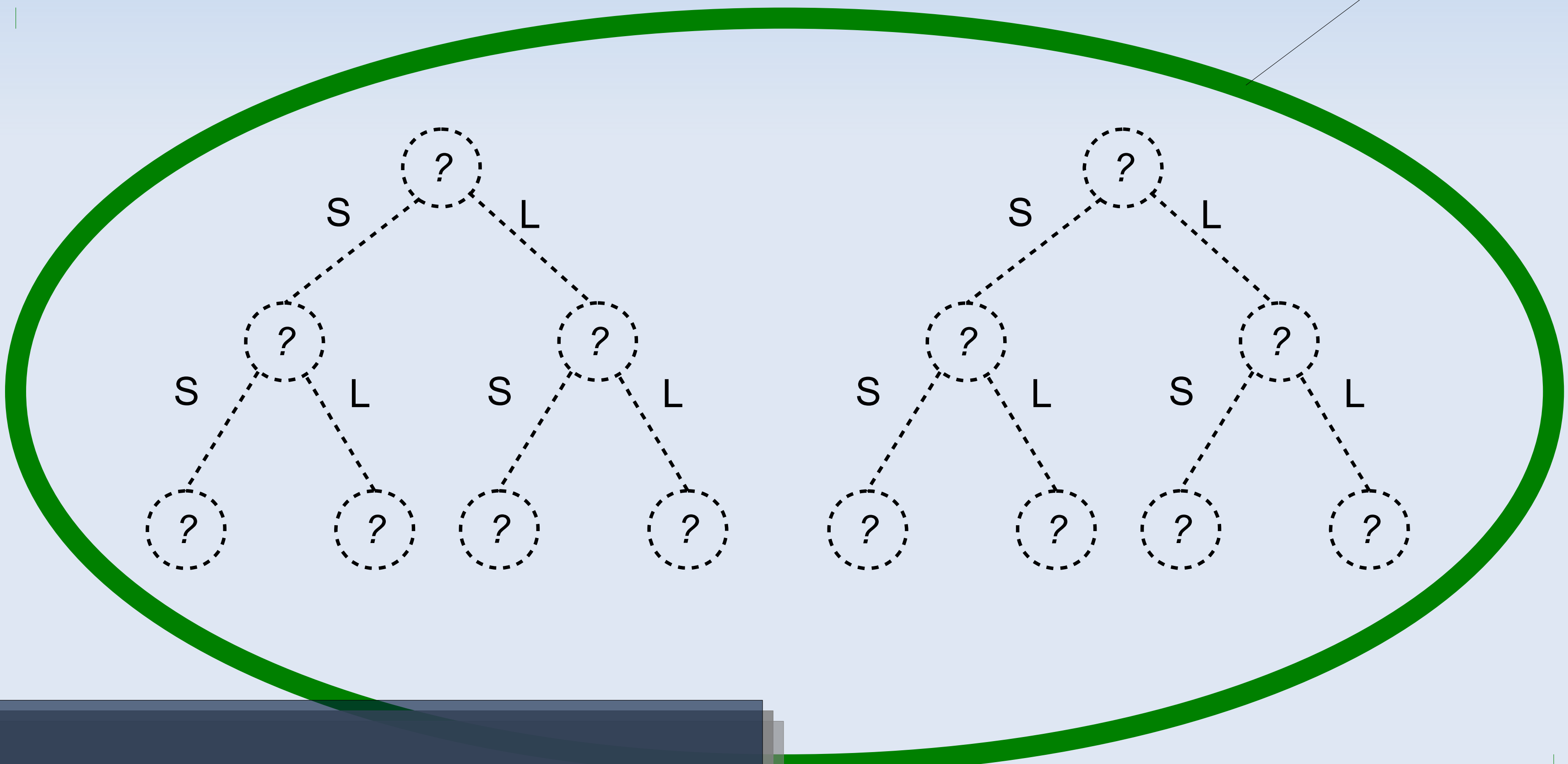
- Incrementally construct all (joint) policies
  - 'forward in time'



Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'

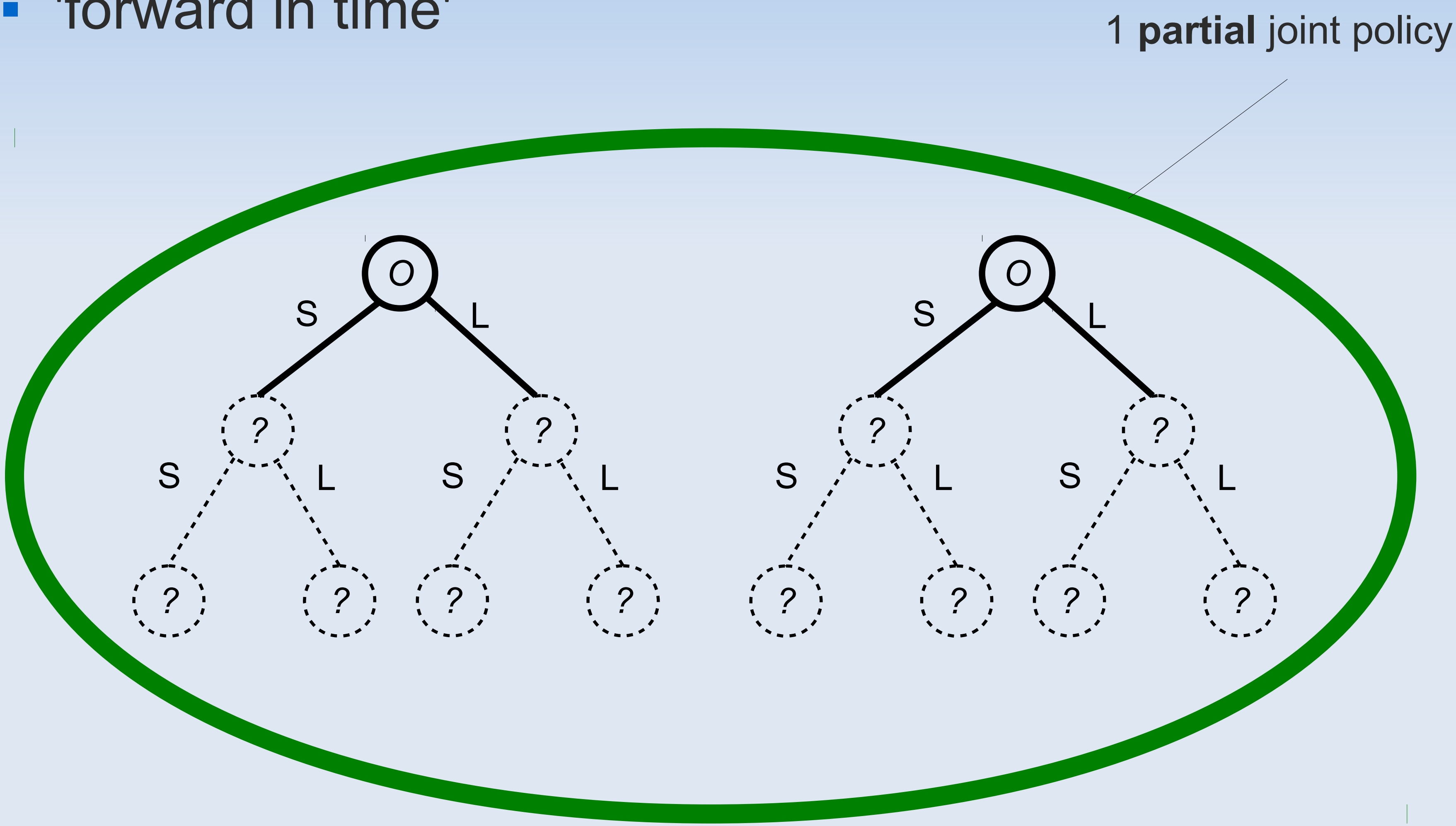
1 **partial** joint policy



Start with unspecified policy

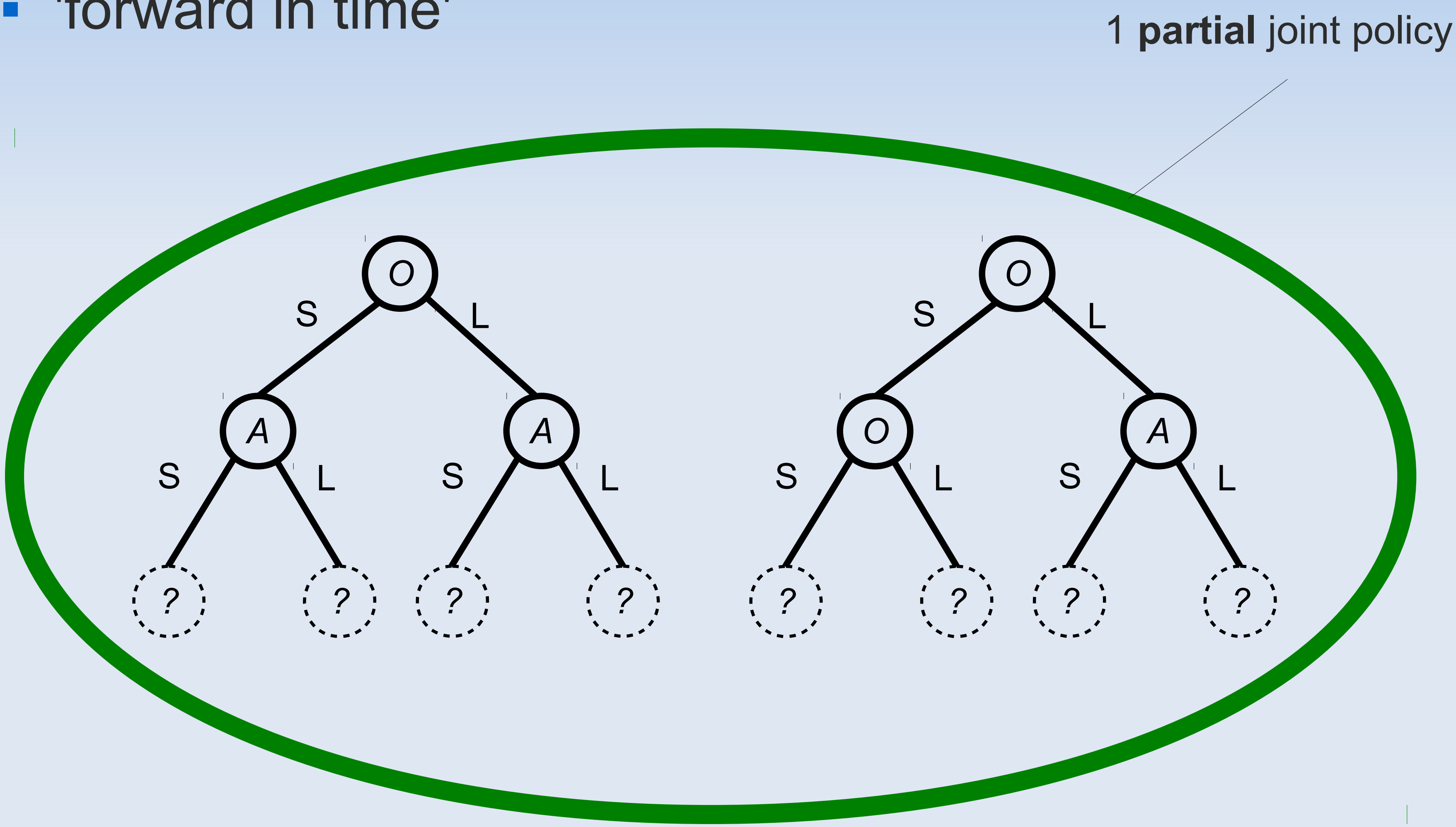
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'



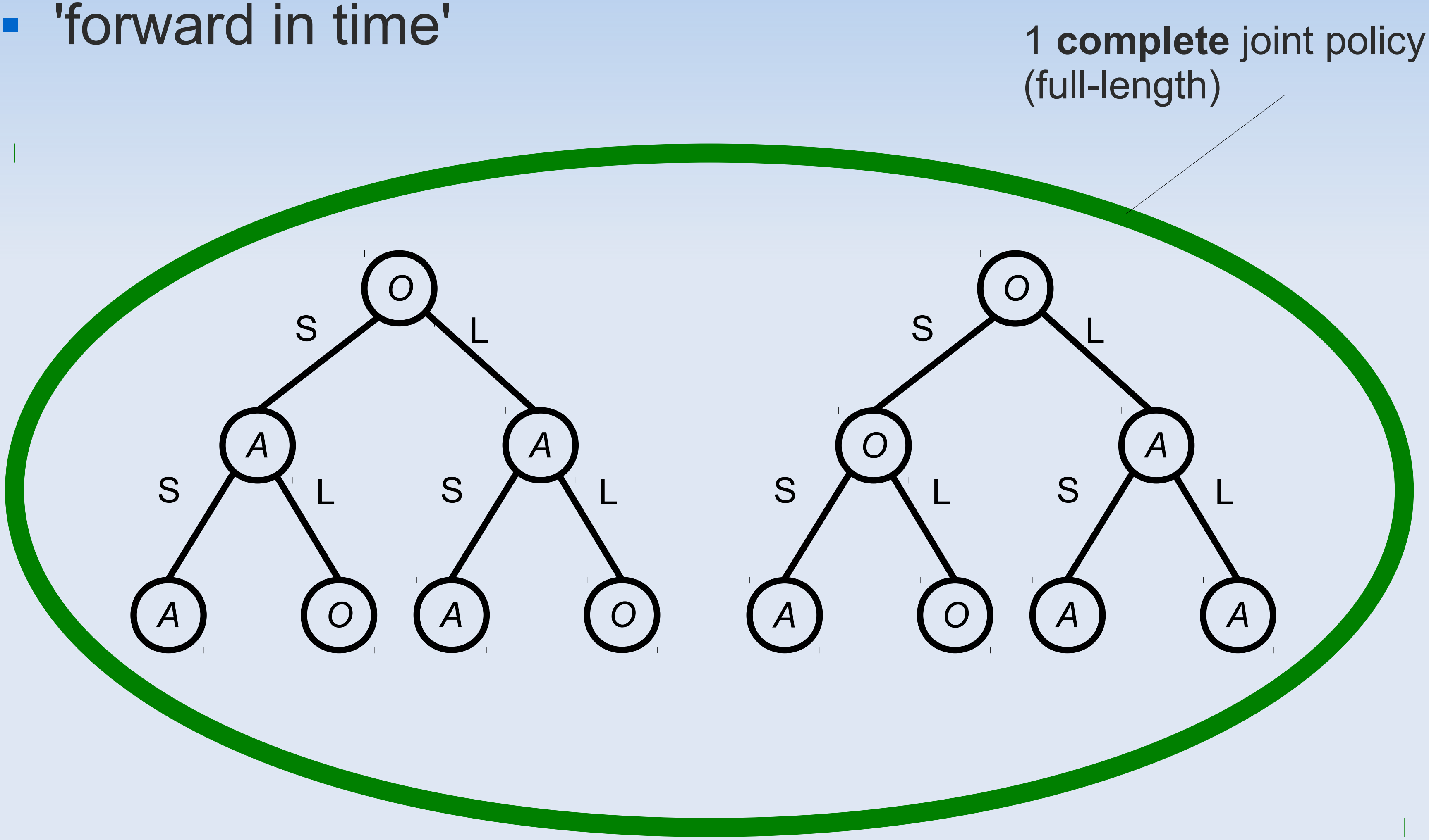
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'



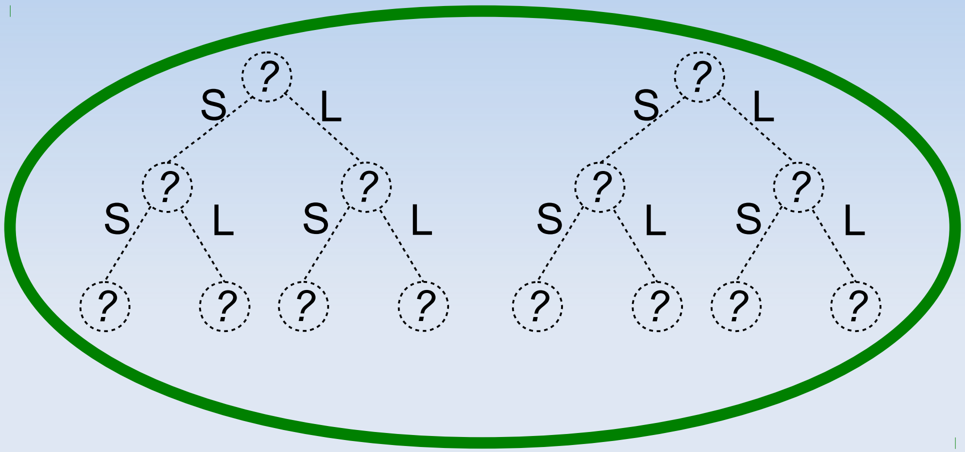
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'



Heuristic Search – 2

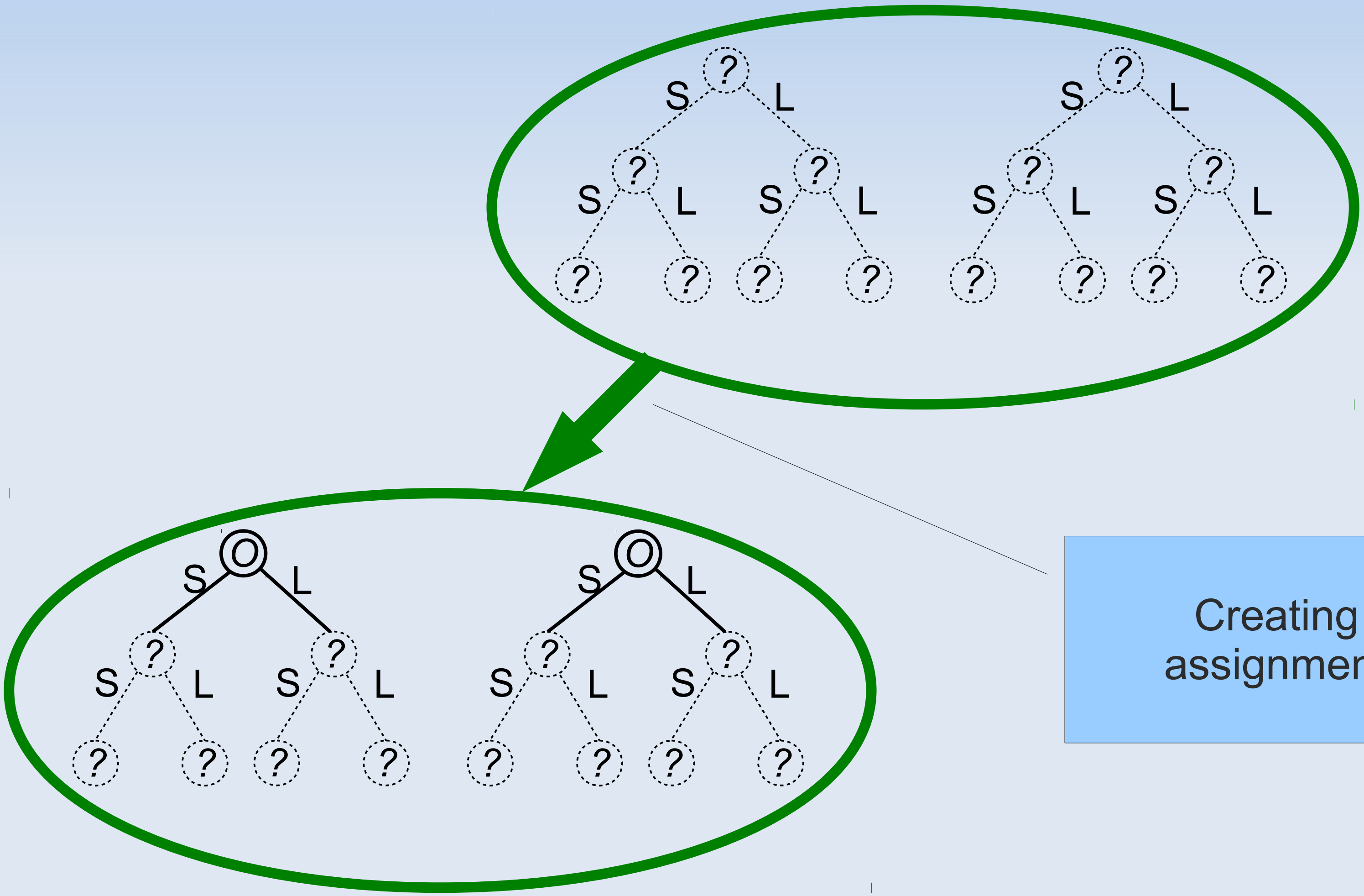
- Creating **ALL** joint policies → tree structure!



Root node:
unspecified joint policy

Heuristic Search – 2

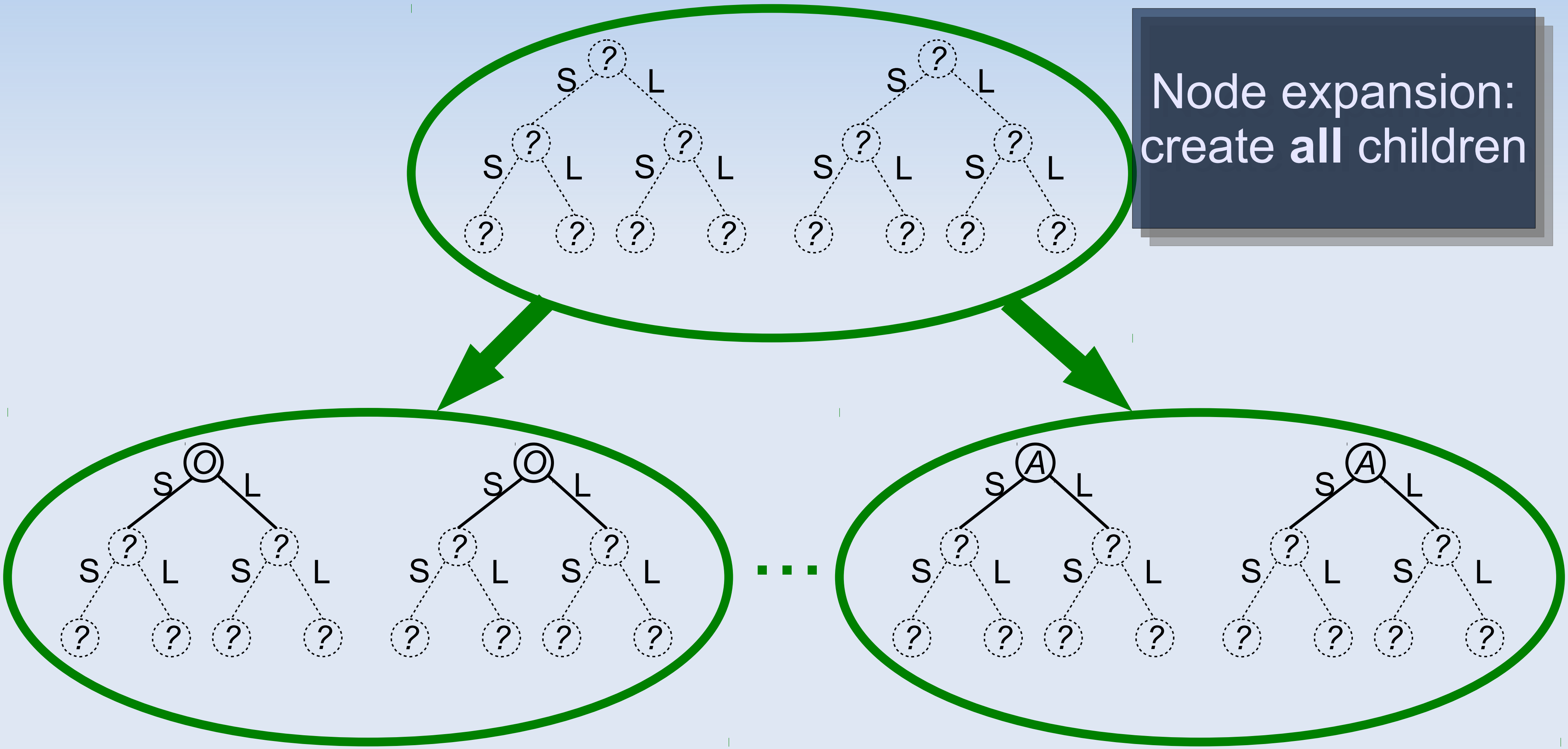
- Creating **ALL** joint policies → tree structure!



Creating a child node:
assignment actions at *t=0*

Heuristic Search – 2

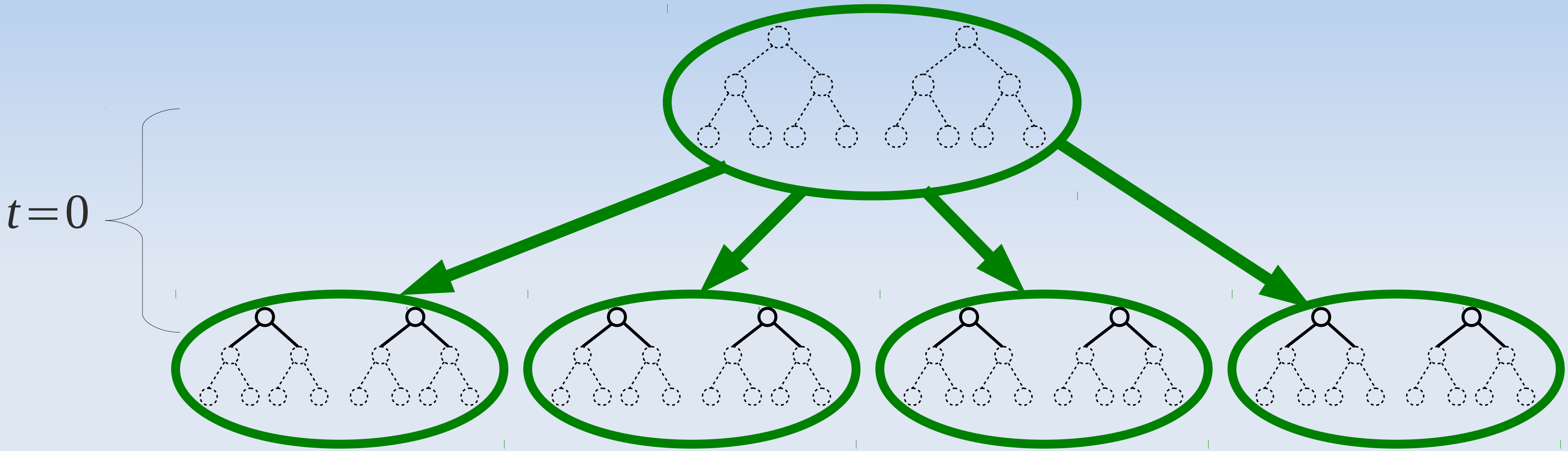
- Creating **ALL** joint policies → tree structure!



Node expansion: create **all** children

Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!

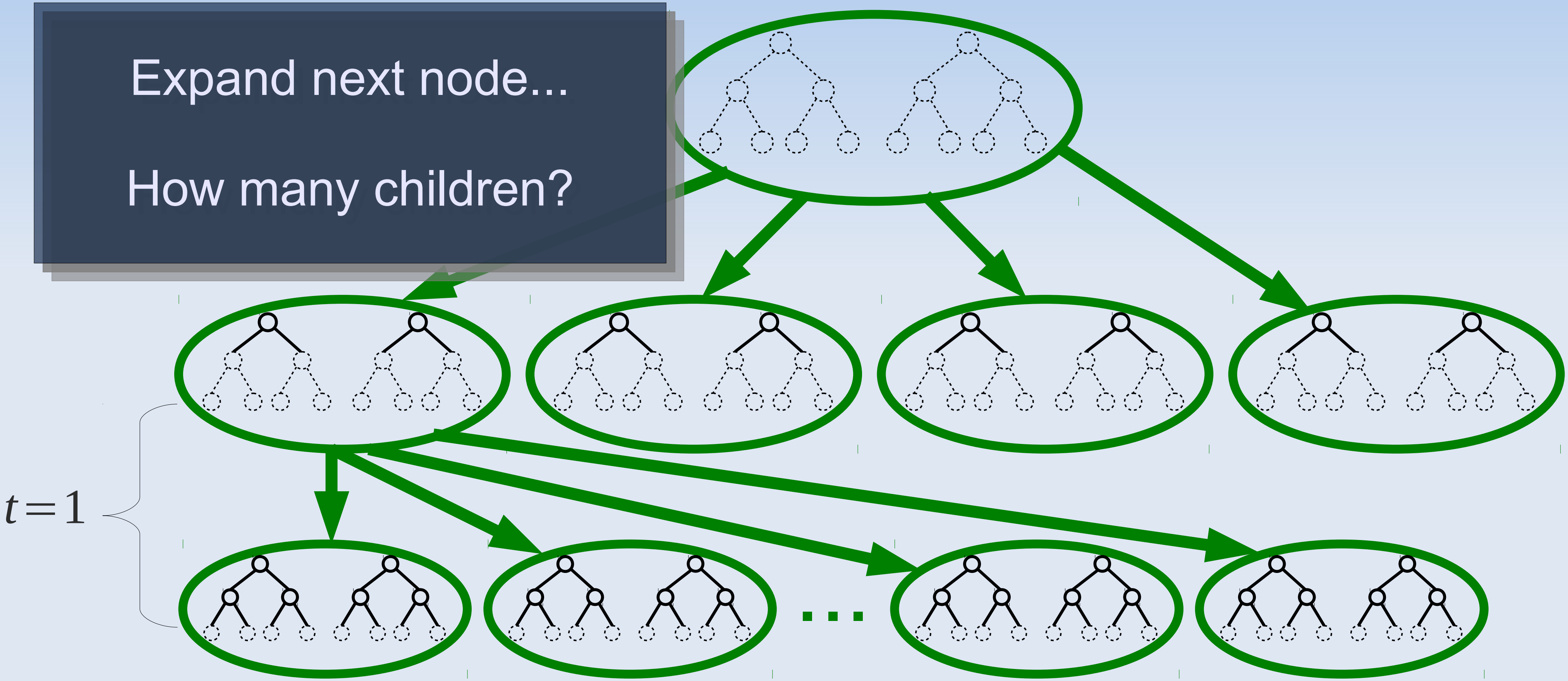


Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!

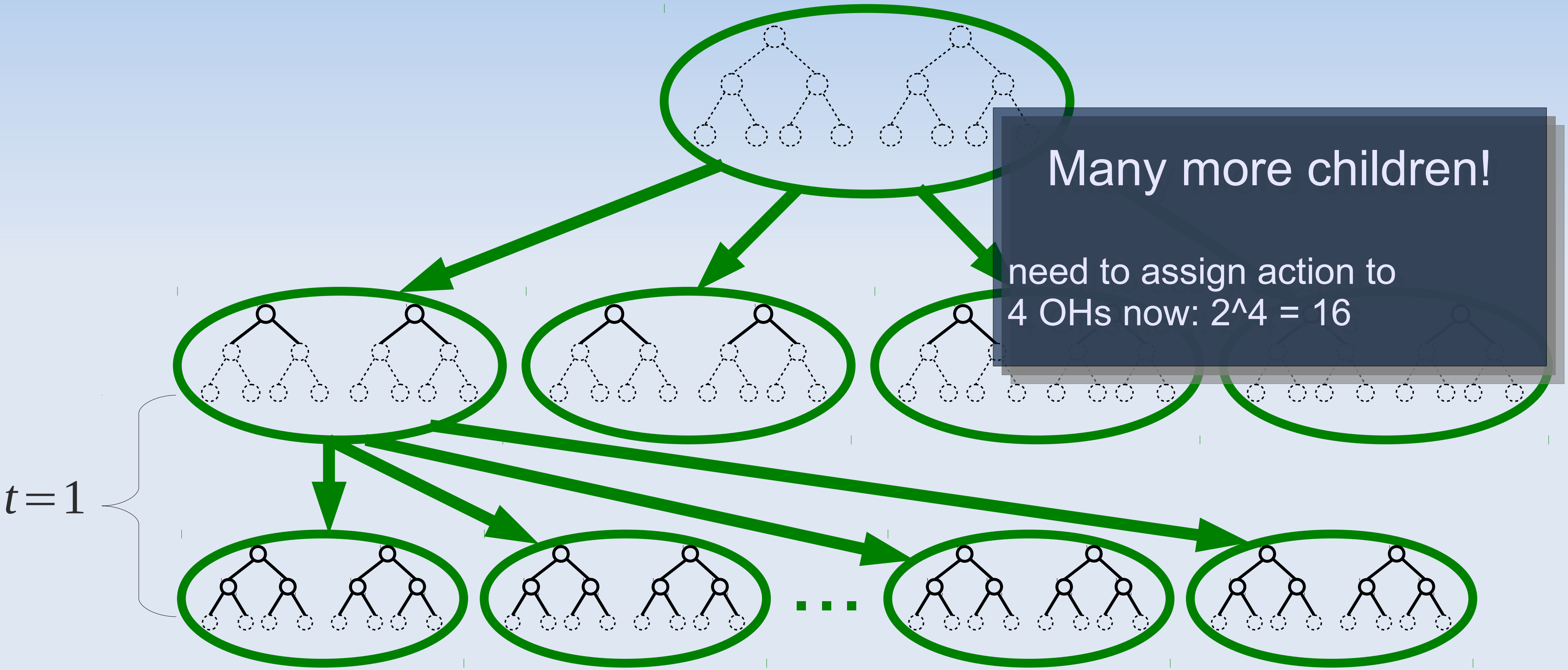
Expand next node...

How many children?



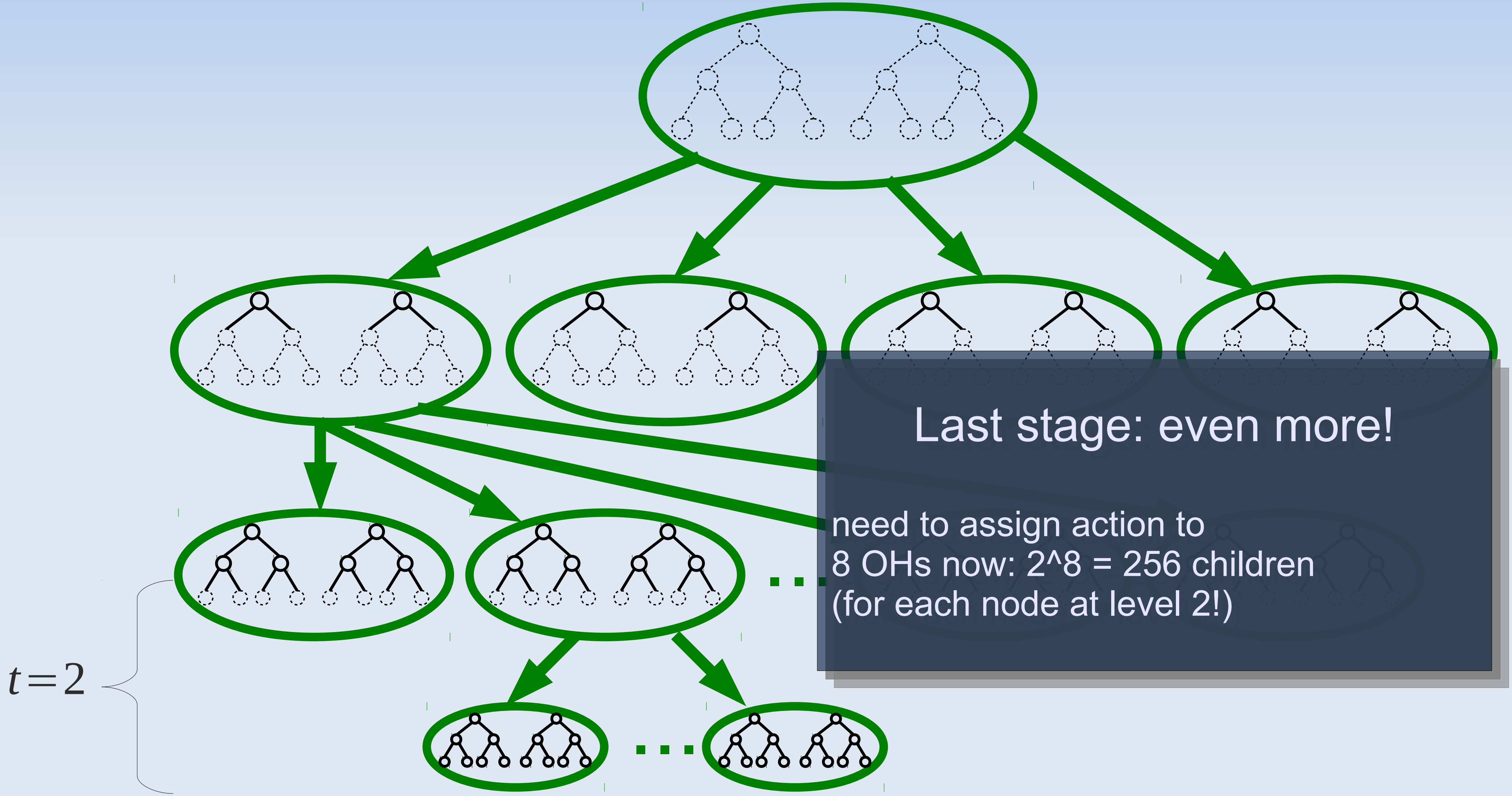
Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!



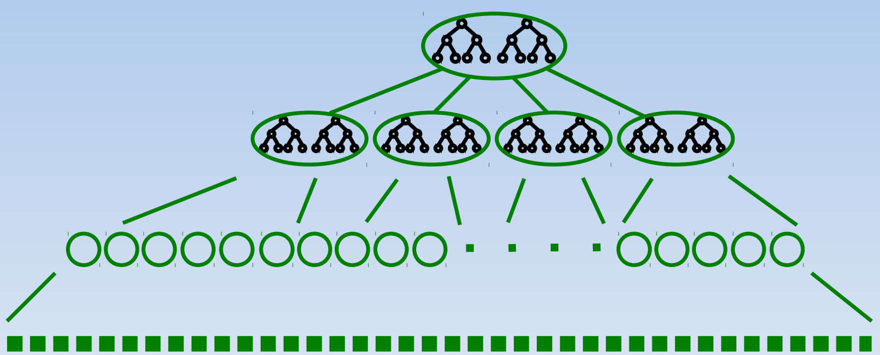
Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!



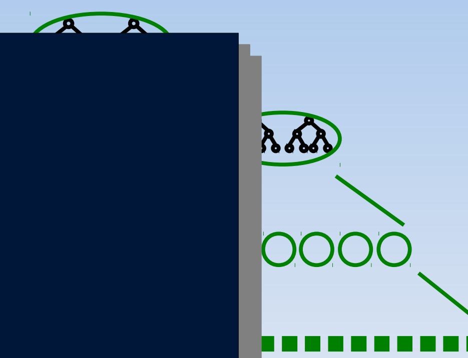
Heuristic Search – 3

- too big to create completely...
- Idea: use **heuristics**
  - avoid going down non-promising branches!
- Apply A* $\longrightarrow$ **Multiagent A*** [Szer et al. 2005]

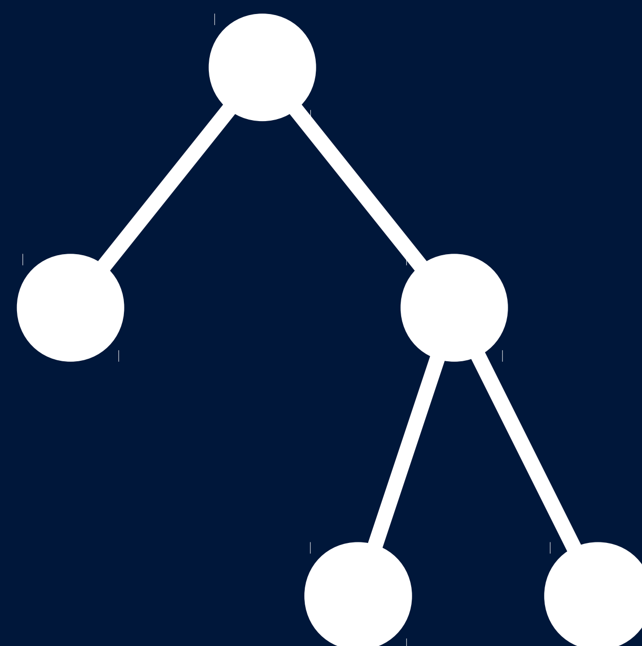


Heuristic Search – 3

- too big to create completely
- Idea:
  - avo
    nor
- Apply



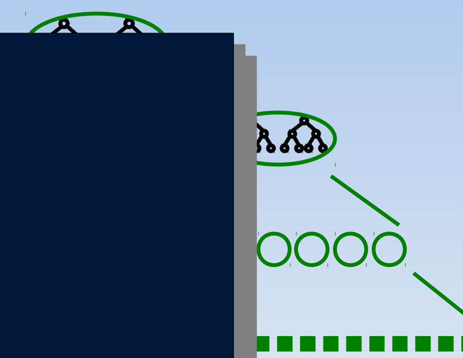
Main intuition A*



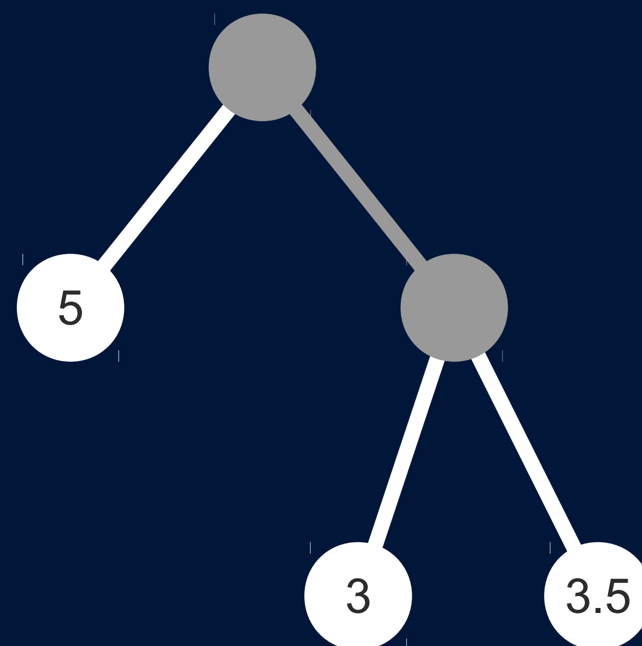
- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]

Heuristic Search – 3

- too big to create completely
- Idea:
  - avo
    non
- Apply



Main intuition A*

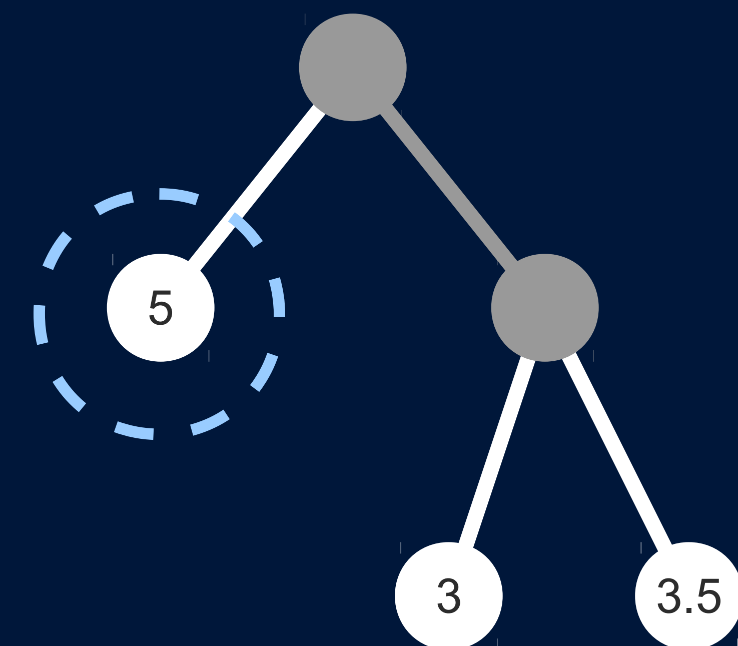


- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]

Heuristic Search – 3

- too big to create completely
- Idea:
  - avo
    nor
- Apply

Main intuition A*

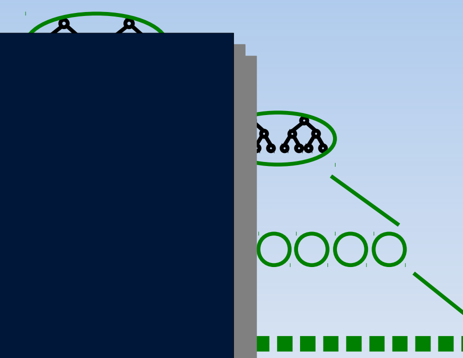


Select highest
valued node
& expand...

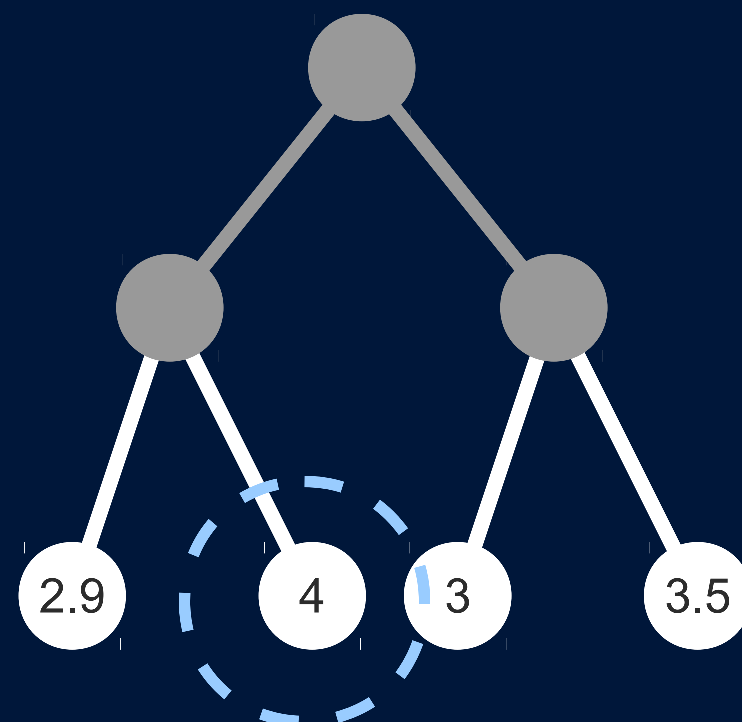
- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]

Heuristic Search – 3

- too big to create completely
- Idea:
  - avo
    non
- Apply



Main intuition A*



- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]

Heuristic Search – 3

- too big to create
- Idea:
  - avo
    non
- Apply

Main intuition

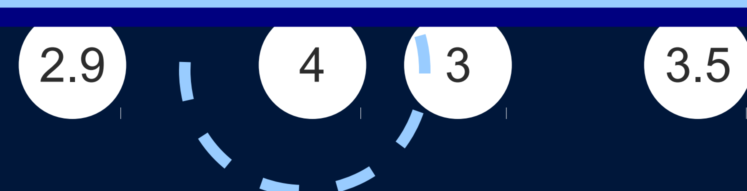
F-Value of a node n

- F(n) is a optimistic estimate
- I.e., F(n) >= V(n') for any descendant n' of n

- F(n) = G(n) + H(n)

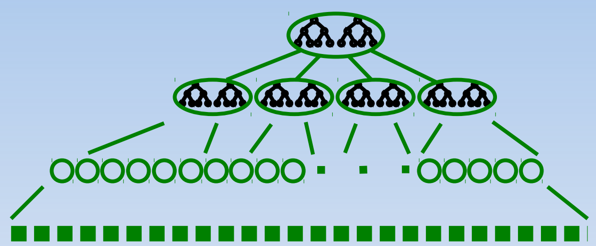
reward up to n
(for first *t* stages)

Optimistic estimate of reward below n
(reward for stages t,t+1,...,h-1)



- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]

Heuristic Search – 4



- Use heuristics F(n) = G(n) + H(n)
- G(n) – actual reward of reaching n
  - a node at depth t specifies φ^t (i.e., actions for first t stages)

    → can compute $V(\varphi^t)$ over stages 0...t-1
- H(n) – should overestimate!
  - pretend that it is an MDP, or POMDP: $\hat{Q}_{MDP}, \hat{Q}_{POMDP}$
  - compute

$$H(n) = H(\varphi^t) = \sum_s P(s|\varphi^t, b^0)\hat{Q}(s)$$

Further Developments

- DP
  - Improvements to exhaustive backup [Amato et al. 2009]
  - Compression of values (LPC) [Boularias & Chaib-draa 2008]
  - (Point-based) Memory bounded DP [Seuken & Zilberstein 2007a]
  - Improvements to PB backup [Seuken & Zilberstein 2007b, Carlin and Zilberstein, 2008; Dibangoye et al, 2009; Amato et al, 2009; Wu et al, 2010, etc.]
- Heuristic Search
  - No backtracking: just most promising path [Emery-Montemerlo et al. 2004, Oliehoek et al. 2008]
  - Clustering of histories: reduce number of child nodes [Oliehoek et al. 2009]
  - Incremental expansion: avoid expanding all child nodes [Spaan et al. 2011]
- MILP [Aras and Dutech 2010]

State of The Art

To get an impression...

- Optimal solutions
  - Improvements of MAA* lead to significant increases
  - but problem dependent

h	MILP	LPC	GMAA-ICE*
4	72	534.9	0.04
6		-	46.43*

dec-tiger – runtime (s)

h	MILP	LPC	GMAA-ICE*
5	25	–	<0.01
500	–	–	0.94*

broadcast channel runtime (s)
* excluding heuristic

- Approximate (no quality guarantees)
  - MBDP: linear in horizon [Seuken & zilberstein 2007a]
  - Rollout sampling extension: up to 20 agents [Wu et al. 2010b]
  - Transfer planning: use smaller problems to solve large (structured) problems (up to 1000) agents [Oliehoek 2010]

Further Topics

- Infinite-horizon planning
- Communication:
  - implicit/explicit
  - delays
  - costs
- Structured Models
  - e.g., factored Dec-POMDPs
- Reinforcement learning

References

- References can be found in

  Frans A. Oliehoek. **Decentralized POMDPs**. In Wiering, Marco and van Otterlo, Martijn, editors, *Reinforcement Learning: State of the Art*, Adaptation, Learning, and Optimization, pp. 471–503, Springer Berlin Heidelberg, Berlin, Germany, 2012.

Some Further Topics

- Further topics
  - Communication
  - Infinite Horizon
  - Reinforcement Learning

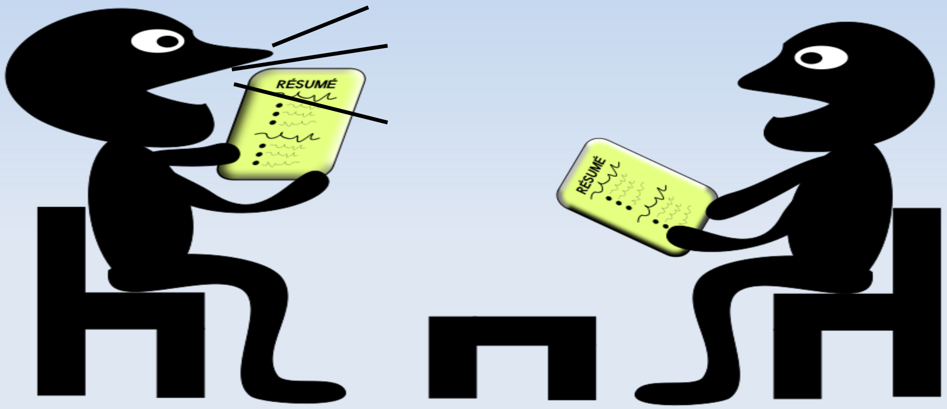
Communication

- instantaneous, cost-free, and noise-free:
  - Dec-MDP → multiagent MDP (MMDP)
  - Dec-POMDP → multiagent POMDP (MPOMDP)

- but in practice:
  - probability of failure
  - delays
  - costs
- Also: implicit communication!
  (via observations and actions)

Implicit Communication

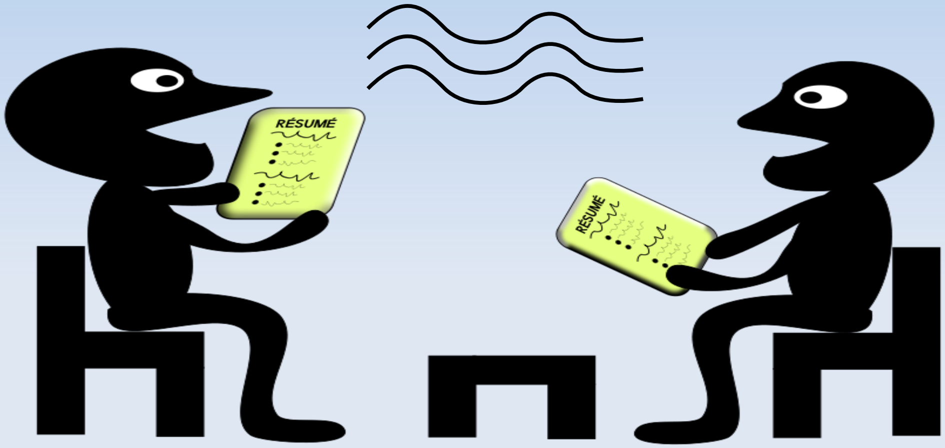
- Encode communications by actions and observations



- Embed the **optimal meaning** of messages by finding the optimal plan [Goldman and Zilberstein 2003, Spaan et al. 2006]

Implicit Communication

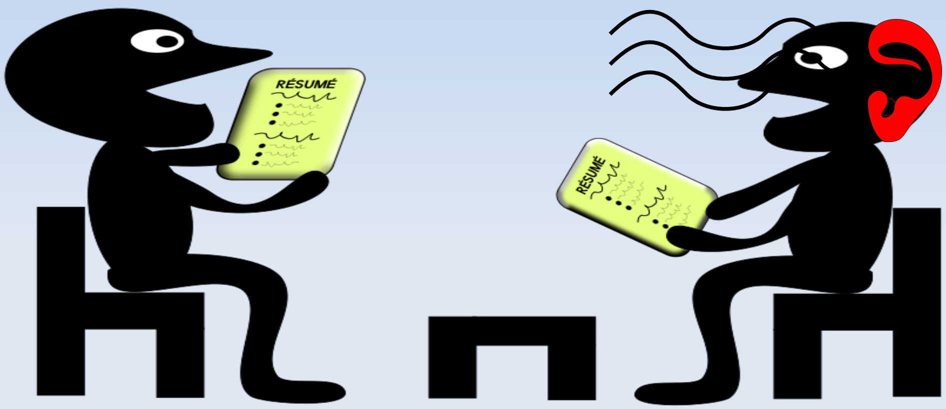
- Encode communications by actions and observations



- Embed the **optimal meaning** of messages by finding the optimal plan [Goldman and Zilberstein 2003, Spaan et al. 2006]

Implicit Communication

- Encode communications by actions and observations



- Embed the **optimal meaning** of messages by finding the optimal plan [Goldman and Zilberstein 2003, Spaan et al. 2006]
- E.g. communication bit
  - doubles the #actions and observations!
  - Clearly, useful... but intractable for general settings
    (perhaps for analysis of very small communication systems)

Explicit Communication

- perform a particular information update (e.g., sync) as in the MPOMDP:
  - each agent broadcasts its information, and
  - each agent uses that to perform joint belief update
- Other approaches:
  - Communication cost [Becker et al. 2005]
  - Delayed communication [Hsu et al. 1982, Spaan et al. 2008, Oliehoek & Spaan 2012]
  - Communicate every k stages [Goldman & Zilberstein 2008]

Infinite-horizon Dec-POMDPs

- Infinite-horizon case: undecidable.
- Can compute ε-approximate solution

- Use finite-state controllers to represent policies.
  - 'back up' operations on controllers, [Bernstein et al. 2009]
  - BPI [Bernstein et al, 2005].
  - NLP [Amato et al, 2010].

Reinforcement Learning

- All this assumed the model is given,
  if not the case: not a great deal of work
  - Plenty of MARL [Busoniu et al, 2008] but not for the general Dec-POMDP setting...
- Exceptions:
  - decentralized gradient ascent [Peshkin et al, 2000]
  - single-agent methods (e.g., Q-learning) [Claus and Boutilier 1998, Crites and Barto 1998]
  - Centralized sample-based planning [Wu et al 2010b]
- problems:
  - when/how the agents observe the rewards? (episodes?)
  - how to learn about coupled dynamics from only individual observations? (cannot even compute a belief *with* the model!)
  - learning in a POMDP is hard!

Extra Slides...

No Compact Representation?

There are a number of types of beliefs considered

- **Joint Belief**, $b(s)$ (as in MPOMDP) [Pynadath and Tambe 2002]
  - compute b(s) using joint actions and observations
  - Problem: agents do not know those during execution
- **Multiagent belief**, $b_i(s,q_{-i})$ [Hansen et al. 2004]
  - Belief over future policies of other agents, q_{-i}
  - Need to be able to predict the other agents!
    - for belief update $P(s'|s,a_i,a_{-i})$, $P(o|a_i,a_{-i},s')$, and prediction of $R(s,a_i,a_{-i})$
  - form of those other policies?
    - most general: $\pi_j : \vec{o}_j \rightarrow a_j$
    - if they use beliefs? → infinite recursion of beliefs!

Coordination vs. Exploitation of Local Information

- Inherent trade-off

  coordination vs. **exploitation of local information**

  - Ignore own observations → 'open loop plan'
    - E.g., "ATTACK on 2nd time step"

      \+ maximally predictable

      \- low quality
  - Ignore coordination

    $b_i(s) = \sum_{q_{-i}} b(s, q_{-i})$

    - E.g., 'individual belief' $b_i(s)$ and execute the MPOMDP policy

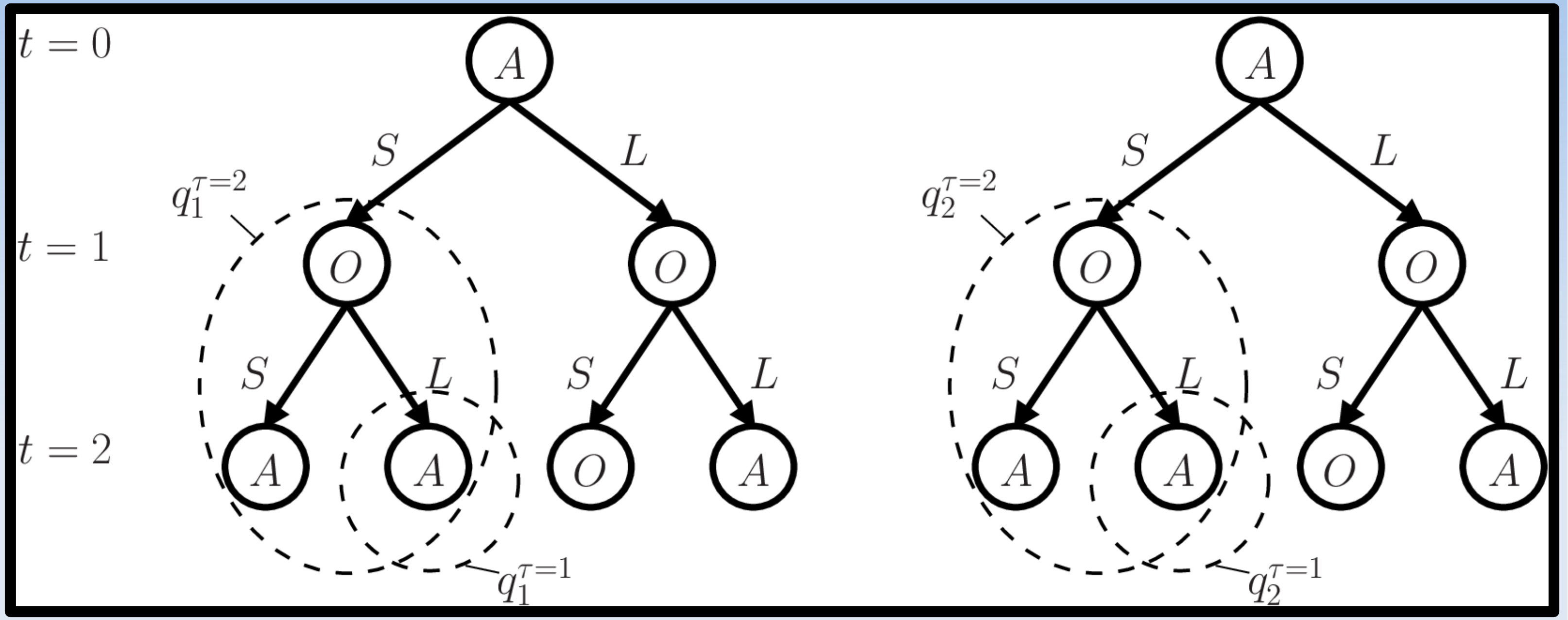
      \+ uses local information

      \- likely to result in mis-coordination

- **Optimal policy π^* should balance between these!**

Value of a Joint Policy

- Sub-tree policies:



- Given a particular joint policy $\pi = q^{\tau=h}$

  $\rightarrow$ Just a (complex) Markov Chain

  - Augmented state $\langle s, q^{\tau=k} \rangle$

$$V(s, q^{\tau=k}) = R(s,a) + \sum_{s'} \sum_{o} P(s', o|s, a) V(s', q^{\tau=k-1})$$

Optimal Value Functions – 1

- Optimal value functions are difficult!
- consider selecting the best joint sub-tree policy q^τ
- We *can* compute value

$$V(\theta, q^{\tau=k}) = \sum_s P(s|\theta, b^0) V(s, q^{\tau=k})$$

- but *cannot* select the maximizing q^τ independently!