CAP6938-02
Plan, Activity, and Intent Recognition

Lecture 10:
Sequential Decision-Making Under Uncertainty (part 1)
MDPs and POMDPs

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu

Reminder

- Turn-in questionnaire
- Homework (due Thurs):
  - 1 page describing the improvements that you plan to make to your project in the next half of the semester

Model: POMDP

- Applications?
- Strengths?
- Weaknesses?
- How does a POMDP differ from an HMM?

Model: POMDP

- Applications?
  - Human-robot interaction
  - Dialog management
  - Assistive technology
  - Agent interaction
- Strengths?
  - Integrates action selection directly with state estimation
- Weaknesses?
  - Intractable for real-world domains
- How does a POMDP differ from an HMM?
  - MDP and POMDP are for calculating optimal decisions from sequences of observations;
  - HMMs are for recognizing hidden state.from sequences of observations.
  - MDP and POMDP: actions and rewards

Markov Decision Processes

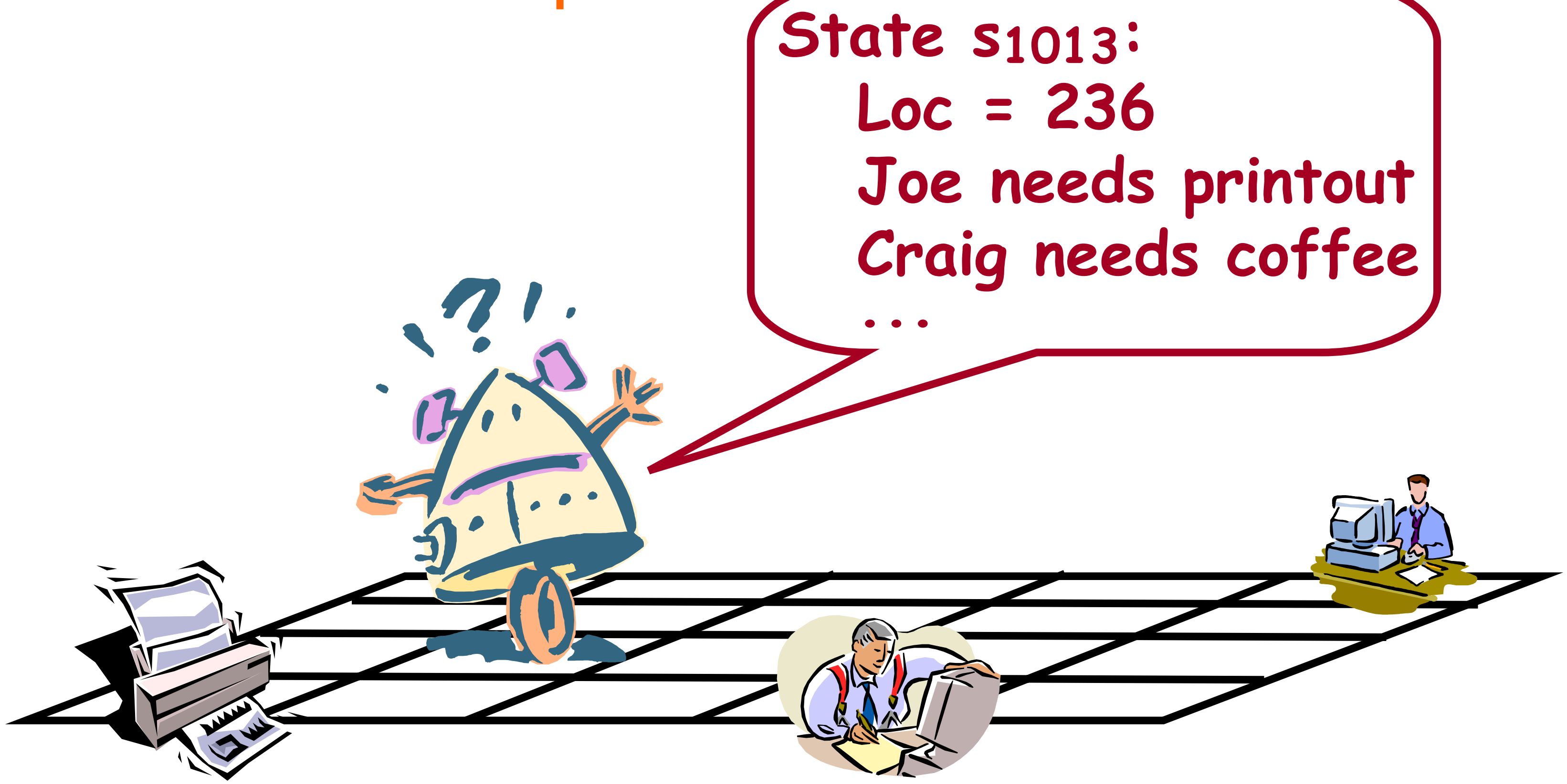
- Classical planning models:
  - logical representation of transition systems
  - goal-based objectives
  - plans as sequences
- Markov decision processes generalize this view
  - controllable, stochastic transition system
  - general objective functions (rewards) that allow tradeoffs with transition probabilities to be made
  - more general solution concepts (policies)

Markov Decision Processes

- An MDP has four components, S, A, R, Pr:
  - (finite) state set S (|S| = n)
  - (finite) action set A (|A| = m)
  - transition function Pr(s,a,t)
    - each Pr(s,a,-) is a distribution over S
    - represented by set of n x n stochastic matrices
  - bounded, real-valued reward function R(s)
    - represented by an n-vector
    - can be generalized to include action costs: R(s,a)
    - can be stochastic (but replacable by expectation)
- Model easily generalizable to countable or continuous state and action spaces

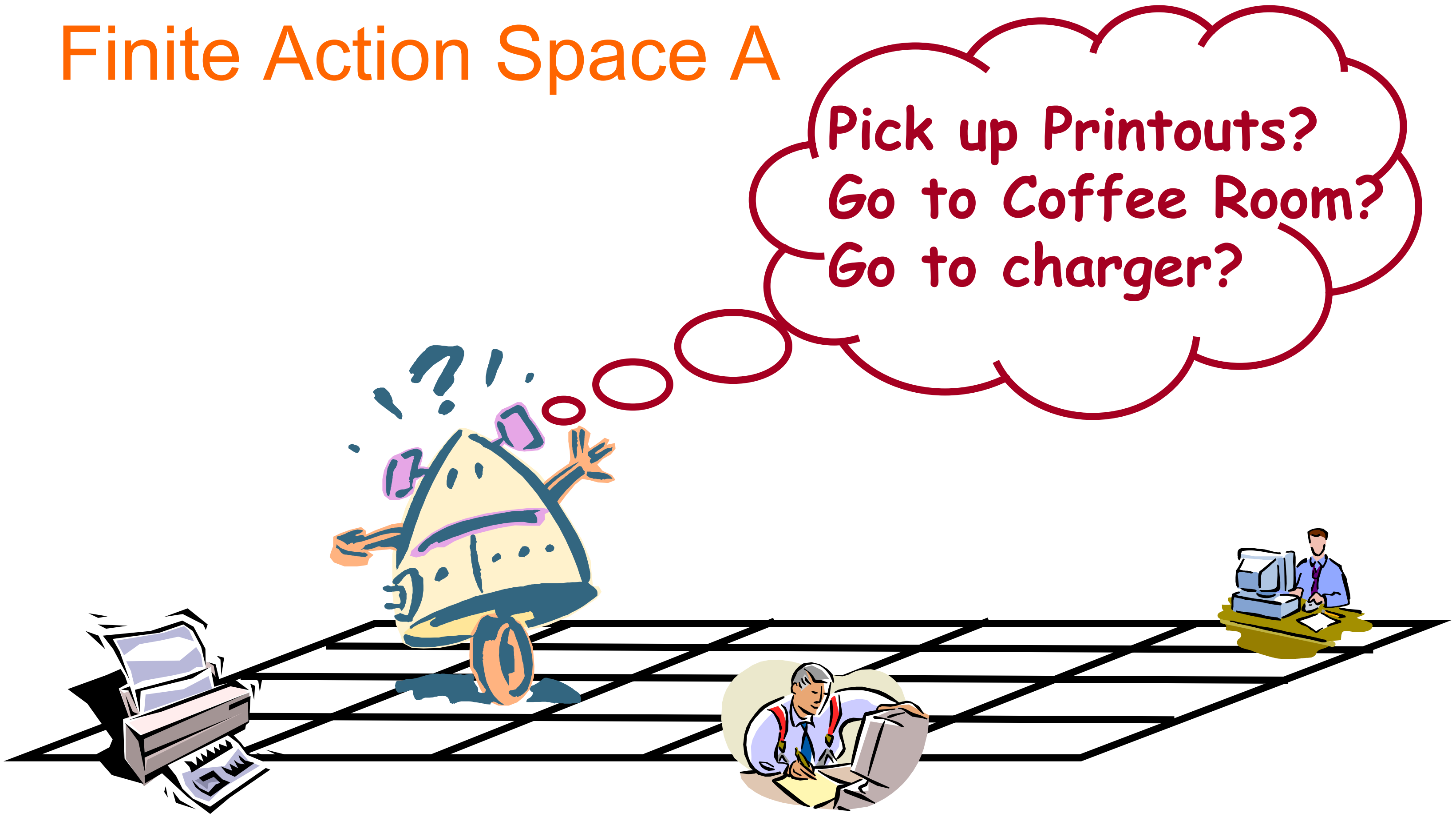
System Dynamics

Finite State Space S



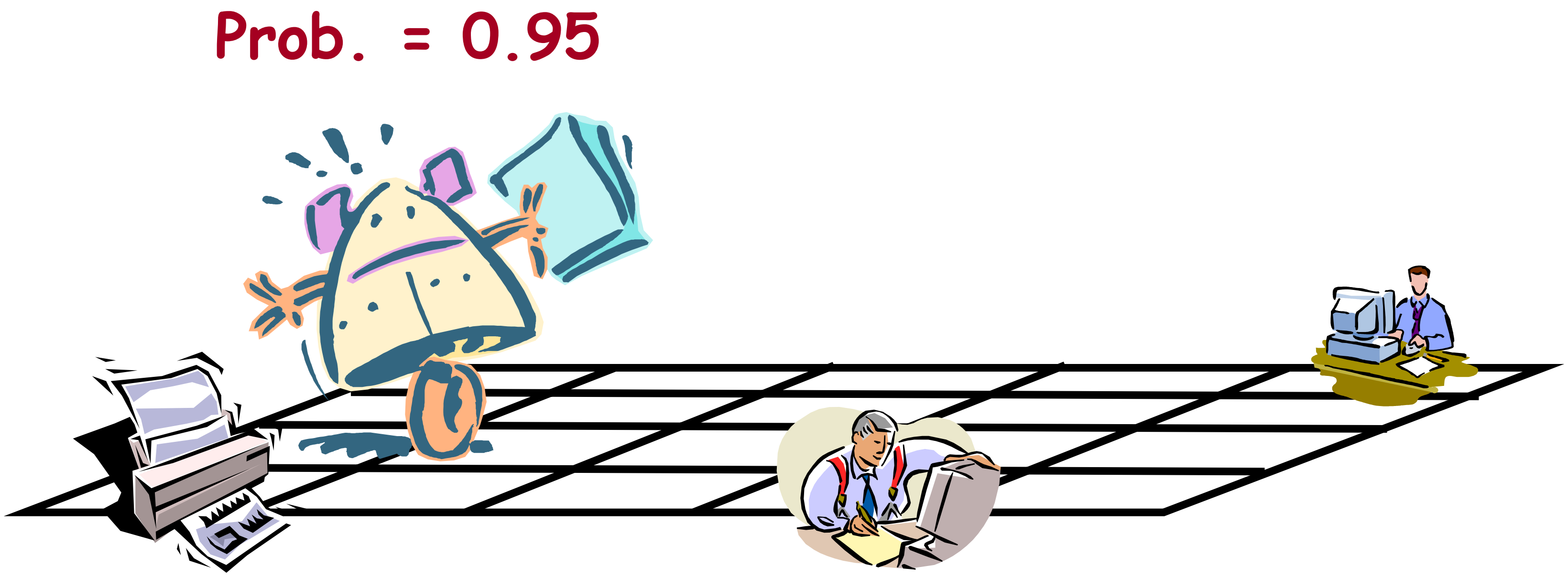
System Dynamics

Finite Action Space A



System Dynamics

Transition Probabilities: $Pr(s_i, a, s_j)$

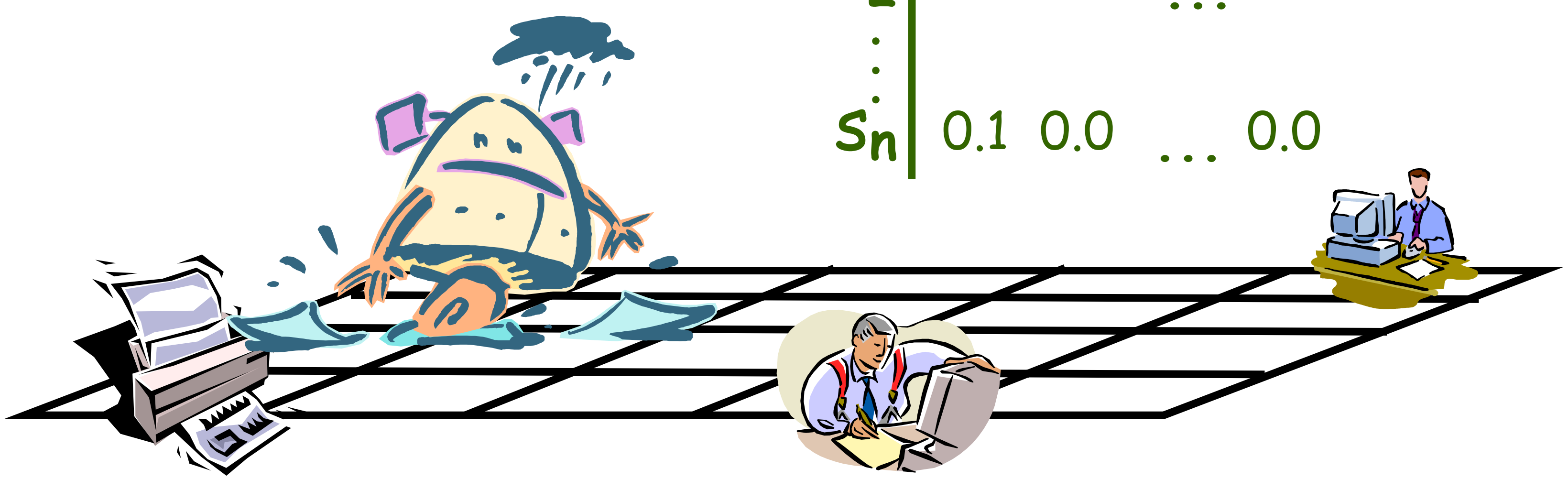


System Dynamics

Transition Probabilities: $Pr(s_i, a, s_k)$

	s_1	s_2	...	s_n
s_1	0.9	0.05	...	0.0
s_2	0.0	0.20	...	0.1
⋮				
s_n	0.1	0.0	...	0.0

Prob. = 0.05



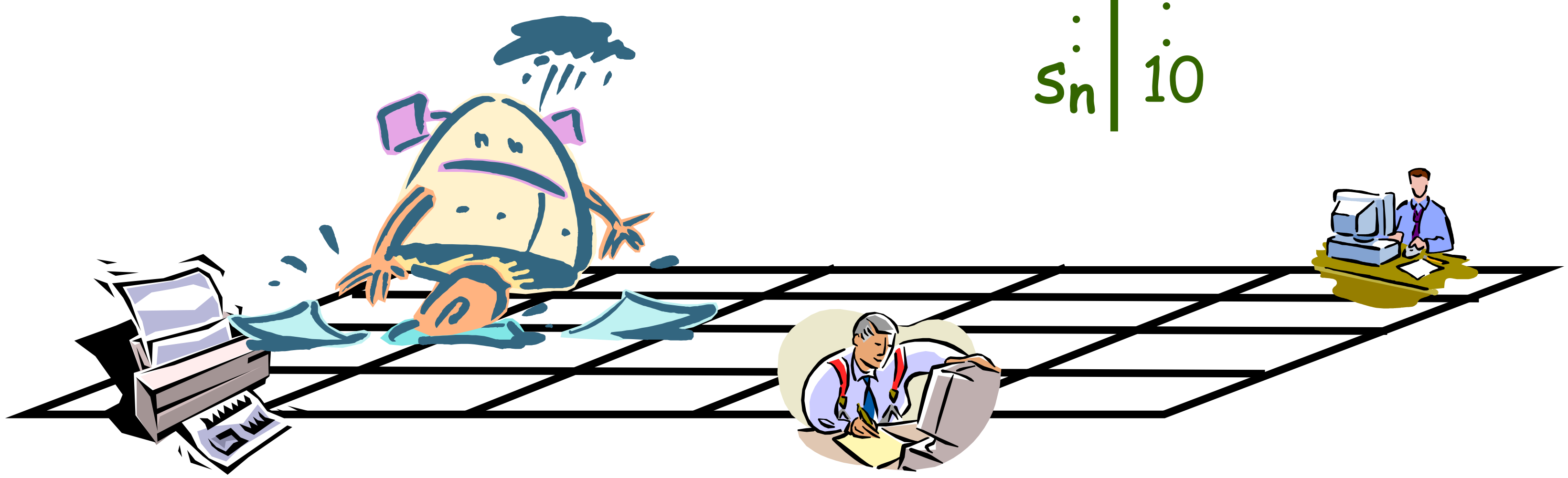
Reward Process

Reward Function: $R(s_i)$

- action costs possible

	R
s_1	12
s_2	0.5
⋮	⋮
s_n	10

Reward = -10



Assumptions

- Markovian dynamics (history independence)
  - $\Pr(S^{t+1}|A^t,S^t,A^{t-1},S^{t-1},\ldots, S^0) = \Pr(S^{t+1}|A^t,S^t)$
- Markovian reward process
  - $\Pr(R^t|A^t,S^t,A^{t-1},S^{t-1},\ldots, S^0) = \Pr(R^t|A^t,S^t)$
- Stationary dynamics and reward
  - $\Pr(S^{t+1}|A^t,S^t) = \Pr(S^{t'+1}|A^{t'},S^{t'})$ for all t, t'
- **Full observability**
  - though we can't predict what state we will reach when we execute an action, once it is realized, we know what it is

Policies

- Nonstationary policy
  - $\pi: S \times T \rightarrow A$
  - $\pi(s,t)$ is action to do at state s with t-stages-to-go
- Stationary policy
  - $\pi: S \rightarrow A$
  - $\pi(s)$ is action to do at state s (regardless of time)
  - analogous to reactive or universal plan
- These assume or have these properties:
  - full observability
  - history-independence
  - deterministic action choices
- *MDP and POMDPs are methods for calculating the optimal lookup tables (policies).*

Value of a Policy

- How good is a policy π? How do we measure "accumulated" reward?
- **Value function** $V: S \rightarrow \mathbb{R}$ associates value with each state (sometimes S x T)
- $V_\pi(s)$ denotes value of policy at state s
  - how good is it to be at state s? depends on immediate reward, but also what you achieve subsequently
  - expected accumulated reward over horizon of interest
  - note $V_\pi(s) \neq R(s)$; it measures *utility*

Value of a Policy (con't)

- Common formulations of value:
  - Finite horizon n: total expected reward given π
  - Infinite horizon discounted: discounting keeps total bounded

Value Iteration (Bellman 1957)

- Markov property allows exploitation of DP principle for optimal policy construction
  - no need to enumerate $|A|^{Tn}$ possible policies
- Value Iteration

Bellman backup

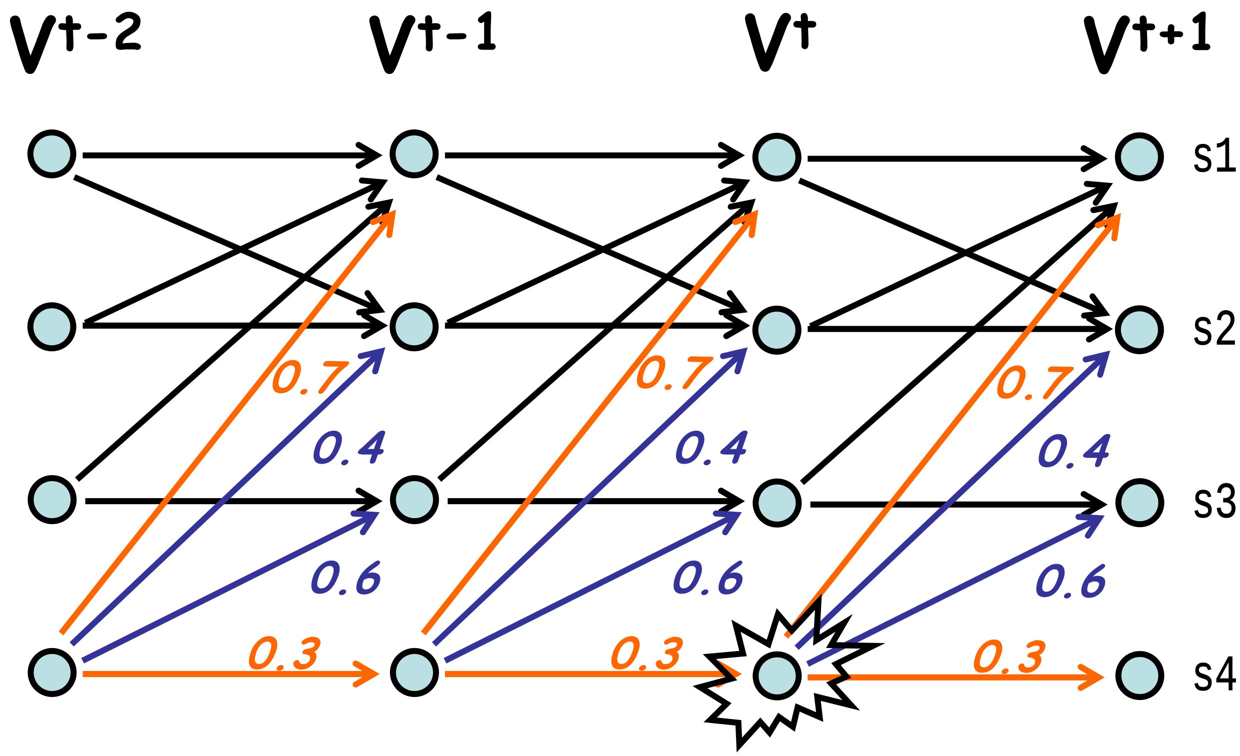
$$V^0(s) = R(s), \quad \forall s$$

$$V^k(s) = R(s) + \max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

$$\pi^*(s,k) = \arg\max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

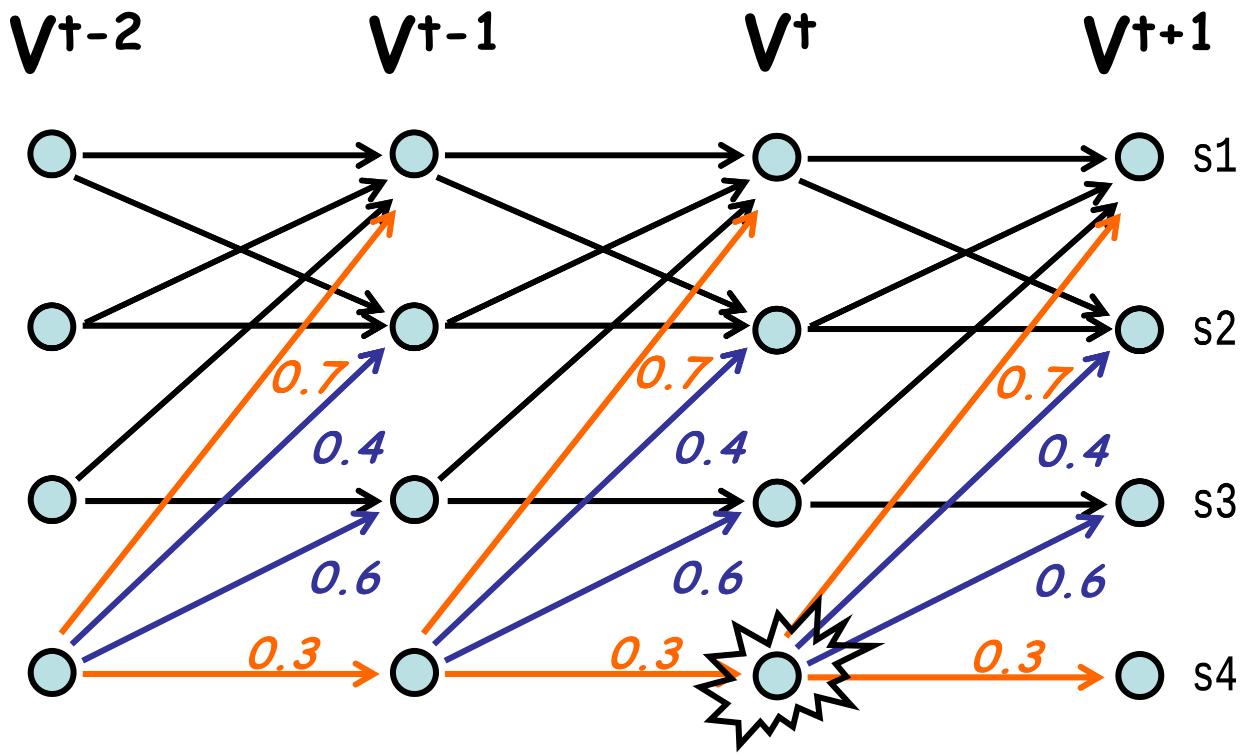
V^k is optimal k-stage-to-go value function

Value Iteration



$$V^t(s4) = R(s4) + \max \{ 0.7\, V^{t+1}(s1) + 0.3\, V^{t+1}(s4)$$
$$0.4\, V^{t+1}(s2) + 0.6\, V^{t+1}(s3) \}$$

Value Iteration



$\Pi^t(s4)$ = max { }

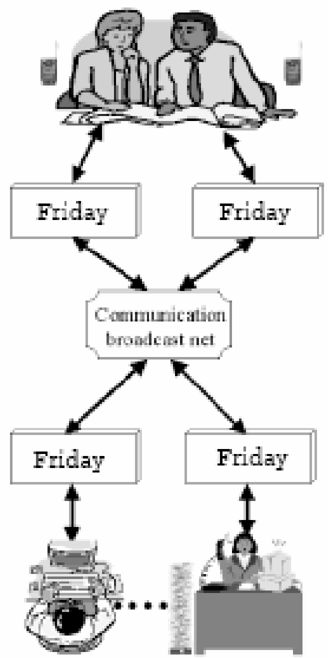
Value Iteration

$V_1(s) := 0$ for all s
$t := 1$
loop
    $t := t + 1$
    loop for all $s \in \mathcal{S}$ and for all $a \in \mathcal{A}$
        $Q_t^a(s) := R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V_{t-1}(s')$
        $V_t(s) := \max_a Q_t^a(s)$
    end loop
until $|V_t(s) - V_{t-1}(s)| < \epsilon$ for all $s \in \mathcal{S}$

Complexity

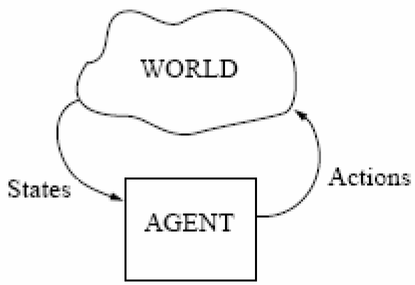
- T iterations
- At each iteration |A| computations of n x n matrix times n-vector: $O(|A|n^3)$
- Total $O(T|A|n^3)$
- Can exploit sparsity of matrix: $O(T|A|n^2)$

MDP Application: Electric Elves

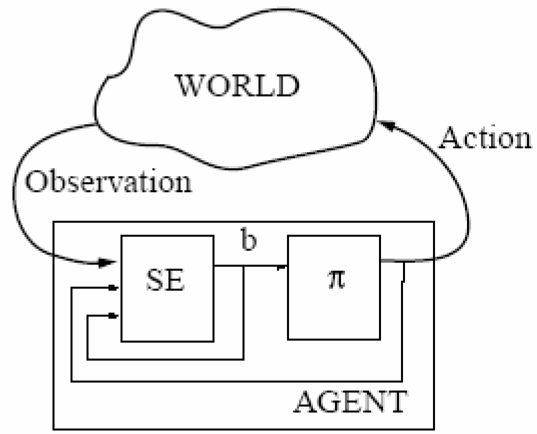


- Calculating optimal transfer of control policy in an adjustable autonomy application
- Dynamically adjusts users' meetings
- State of world is known; future actions of users are unknown

Recognizing User Intent



MDP



POMDP

POMDPs

- *Partially observable Markov Decision Process (POMDP)*:
  - a stochastic system $\Sigma = (S, A, P)$ as before
  - A finite set O of *observations*
    - $P_a(o|s)$ = probability of observation o in state s after executing action a
  - Require that for each a and s, $\sum_{o \text{ in } O} P_a(o|s) = 1$

- O models partial observability
  - The controller can't observe s directly; it can only observe o
  - The same observation o can occur in more than one state

- Why do the observations depend on the action a? Why do we have $P_a(o|s)$ rather than $P(o|s)$?
  - This is a way to model *sensing actions*, which do not change the state but return information make some observation available (e.g., from a sensor)

Example of a Sensing Action

- Suppose there are a state s_1 action a_1, and observation o_1 with the following properties:
  - For every state s, $Pa_1(s|s) = 1$
    - a_1 does not change the state
  - $Pa_1(o_1|s_1) = 1$, and
    $Pa_1(o_1|s) = 0$ for every state $s \neq s_1$
    - After performing a_1, o_1 occurs if and only if we're in state s_1
- Then to tell if you're in state s_1, just perform action a_1 and see whether you observe o_1

- Two states s and s' are *indistinguishable* if for every o and a, $P_a(o|s) = P_a(o|s')$

Belief States

- At each point we will have a probability distribution $b(s)$ over the states in S
  - $b(s)$ is called a *belief state* (our belief about what state we're in)
- Basic properties:
  - $0 \le b(s) \le 1$ for every s in S
  - $\sum_{s \text{ in } S} b(s) = 1$
- Definitions:
  - b_a = the belief state after doing action a in belief state b
    - Thus $b_a(s) = P(\text{in } s \text{ after doing } a \text{ in } b) = \sum_{s' \text{ in } S} P_a(s|s')\, b(s')$
  - $b_a(o) = P(\text{observe } o \text{ after doing } a \text{ in } b)$ **Marginalize over states**
    $= \sum_{s \text{ in } S} P_a(o|s)\, b(s)$
  - $b_a^o(s) = P(\text{in } s \text{ after doing } a \text{ in } b \text{ and observing } o)$

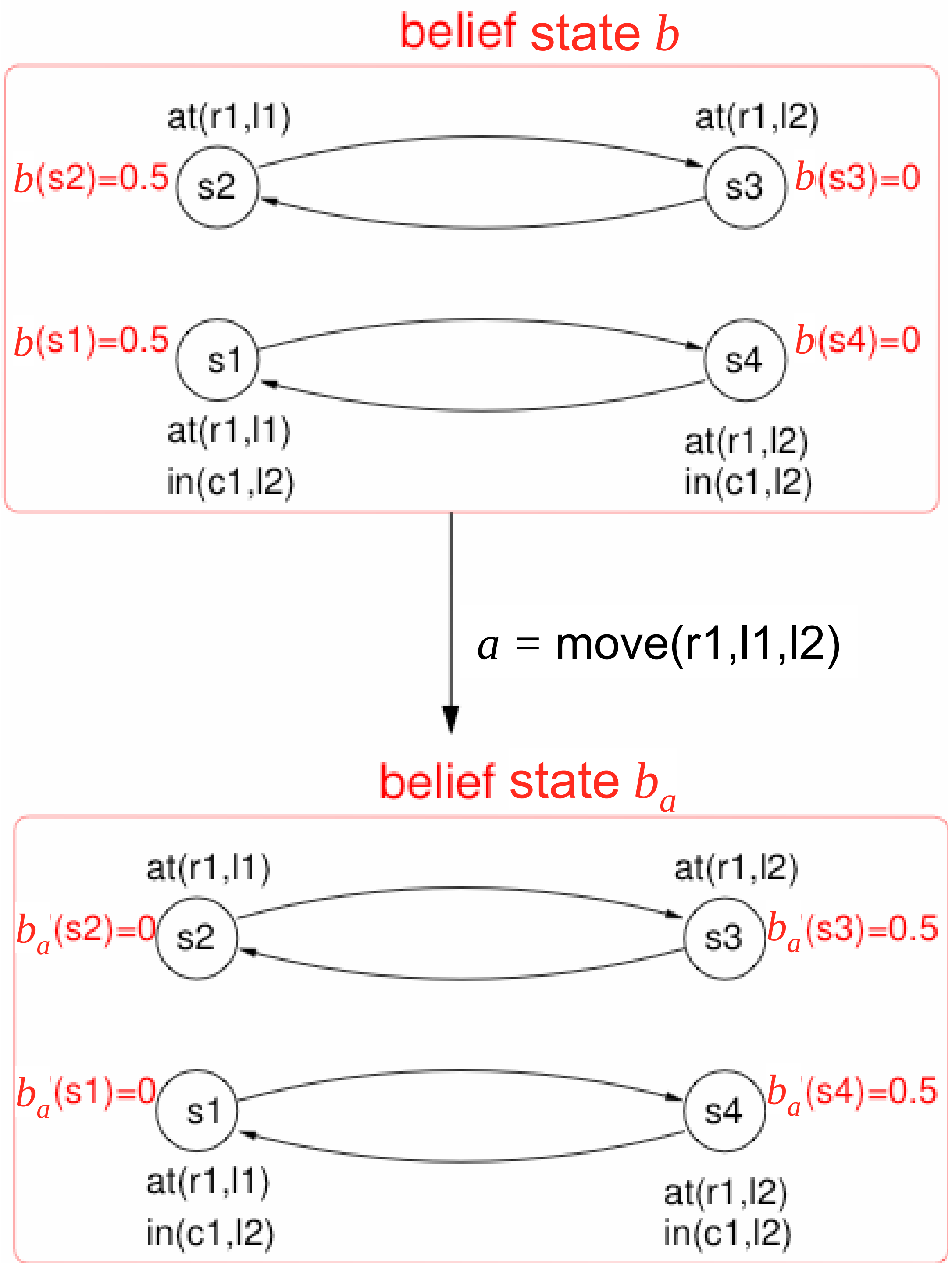
Belief states are n-dimensional vectors representing the probability of being in every state..

Belief States (Continued)

- Recall that in general, $P(x|y,z)\ P(y|z) = P(x,y|z)$
- Thus
  $P_a(o|s)\ b_a(s)$
  = P(observe o after doing a in s) P(in s after doing a in b)
  = P(in s and observe o after doing a in b)
- Similarly,
  $b_a^o(s)\ b_a(o)$
  = P(in s after doing a in b and observing o)
  * P(observe o after doing a in b)
  = P(in s and observe o after doing a in b)
- Thus $b_a^o(s) = P_a(o|s)\ b_a(s)\ /\ b_a(o)$ **Formula for updating belief state**
- Can use this to distinguish states that would otherwise be indistinguishable

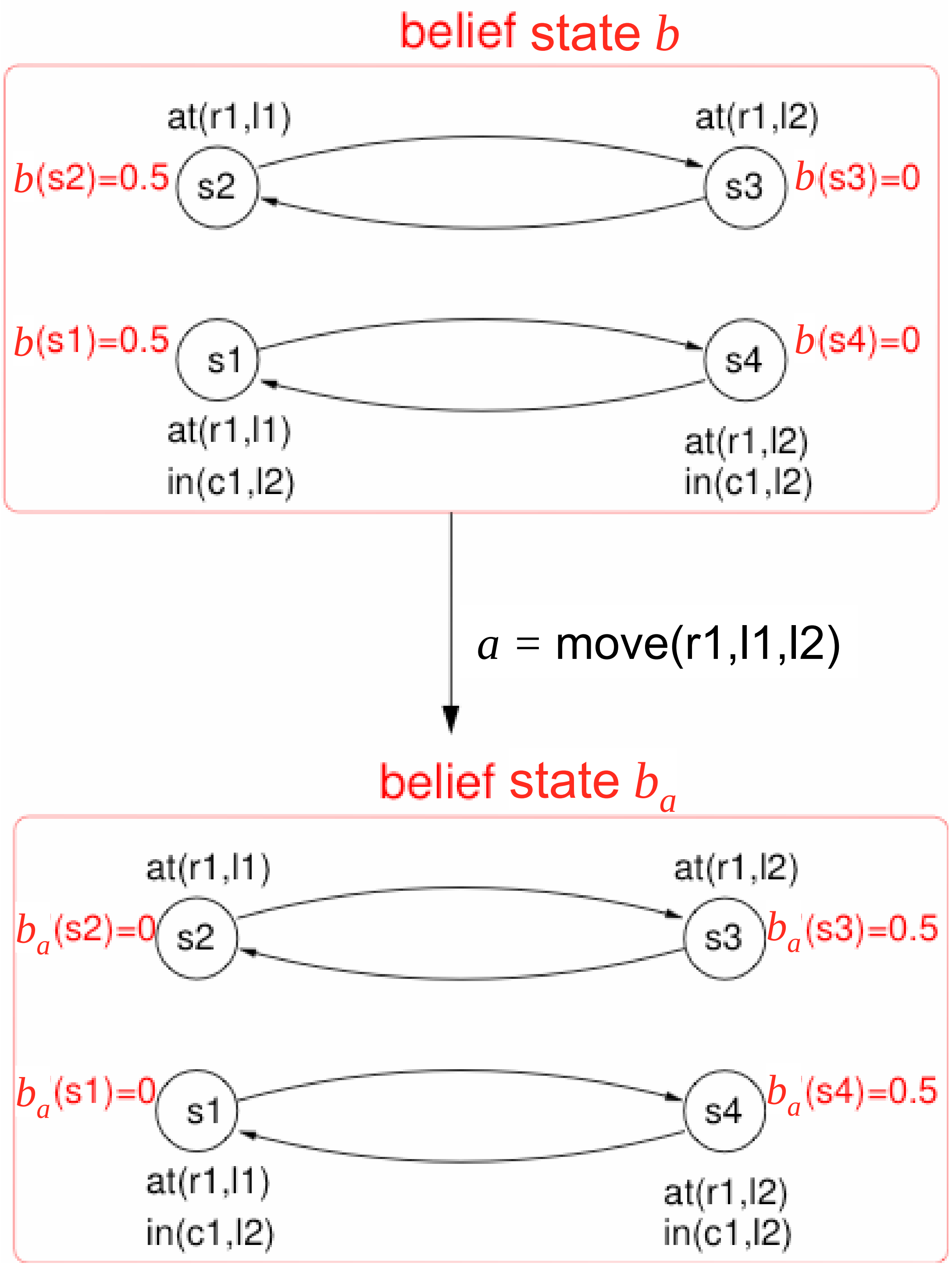
Example

- Robot r1 can move between l1 and l2
  - move(r1,l1,l2)
  - move(r1,l2,l1)
- There may be a container c in location l2
  - in(c1,l2)
- O = {full, empty}
  - full: c1 is present
  - empty: c1 is absent
  - abbreviate full as f, and empty as e



Example (Continued)

- Neither "move" action returns useful observations
- For every state *s* and for *a* = either "move" action,
  - $P_a(\text{f}|s) = P_a(\text{e}|s) = P_a(\text{f}|s) = P_a(\text{e}|s) = 0.5$
- Thus if there are no other actions, then
  - s1 and s2 are indistinguishable
  - s3 and s4 are indistinguishable



Example (Continued)

- Suppose there's a sensing action see that works perfectly in location l2

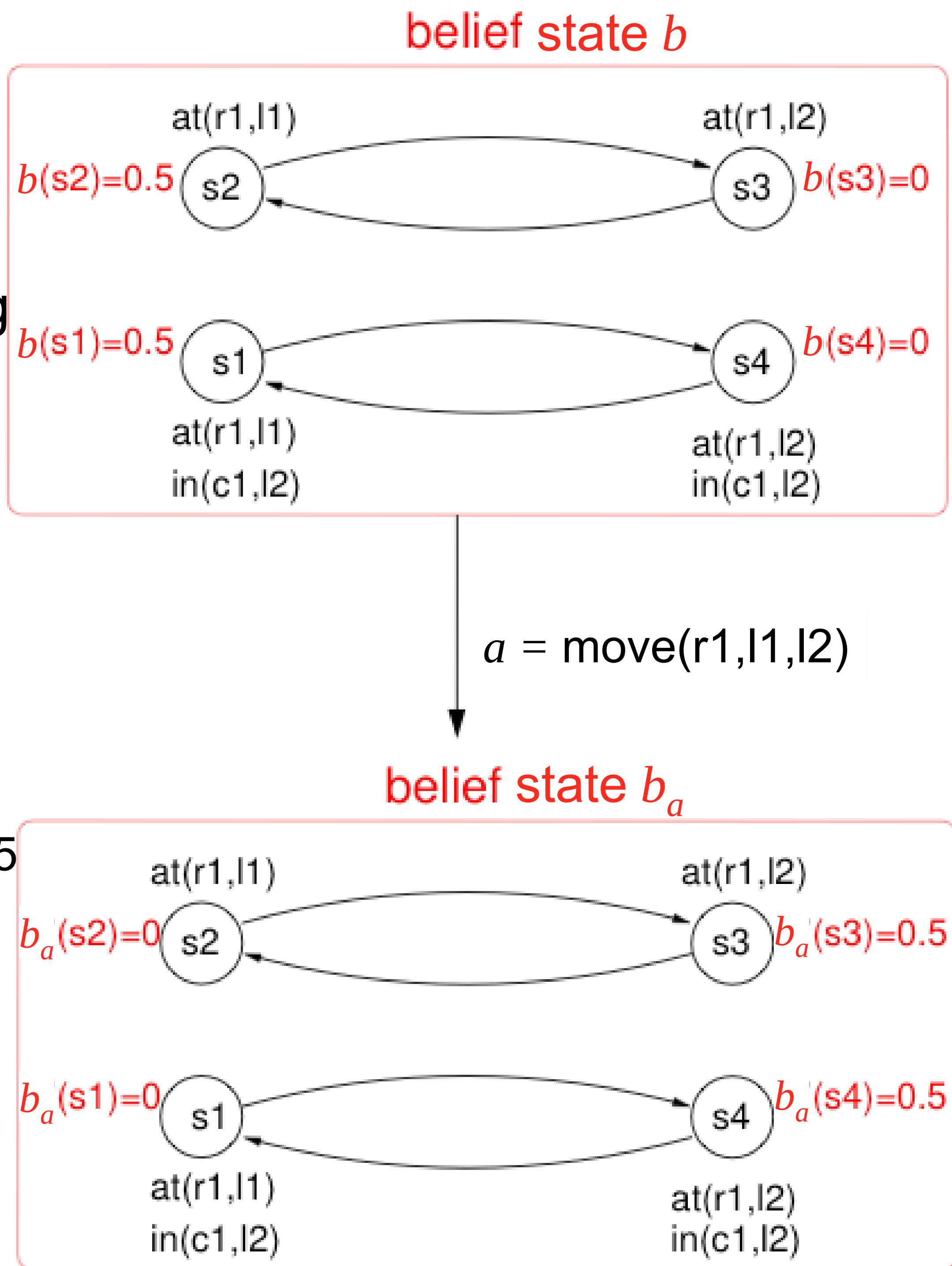
  $P_{see}(f|s4) = P_{see}(e|s3) = 1$

  $P_{see}(f|s3) = P_{see}(e|s4) = 0$

- see does not work elsewhere

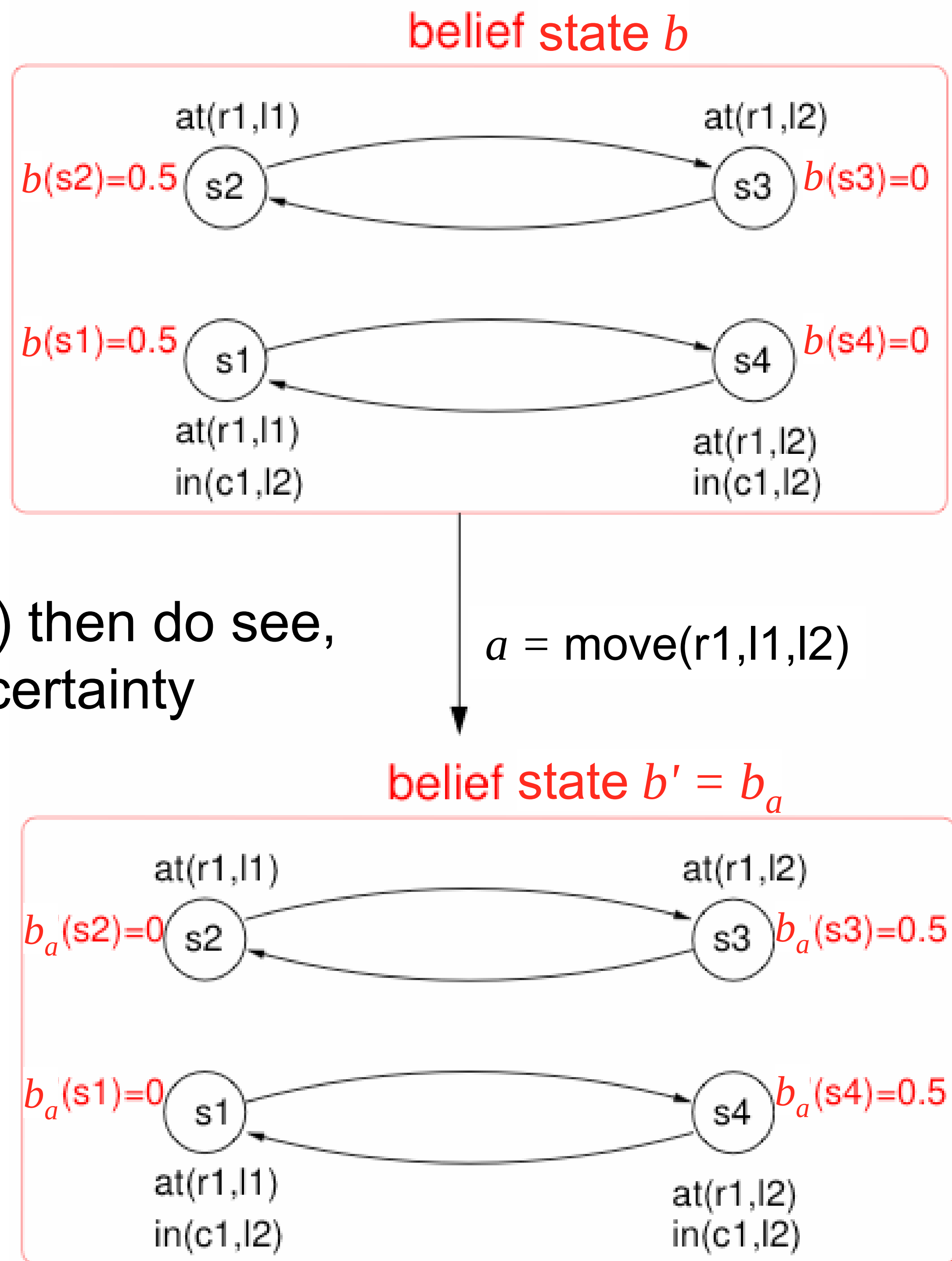
  $P_{see}(f|s1) = P_{see}(e|s1)$
  $= P_{see}(f|s2) = P_{see}(e|s2) = 0.5$

- Then
  - s1 and s2 are still indistinguishable
  - s3 and s4 are now distinguishable



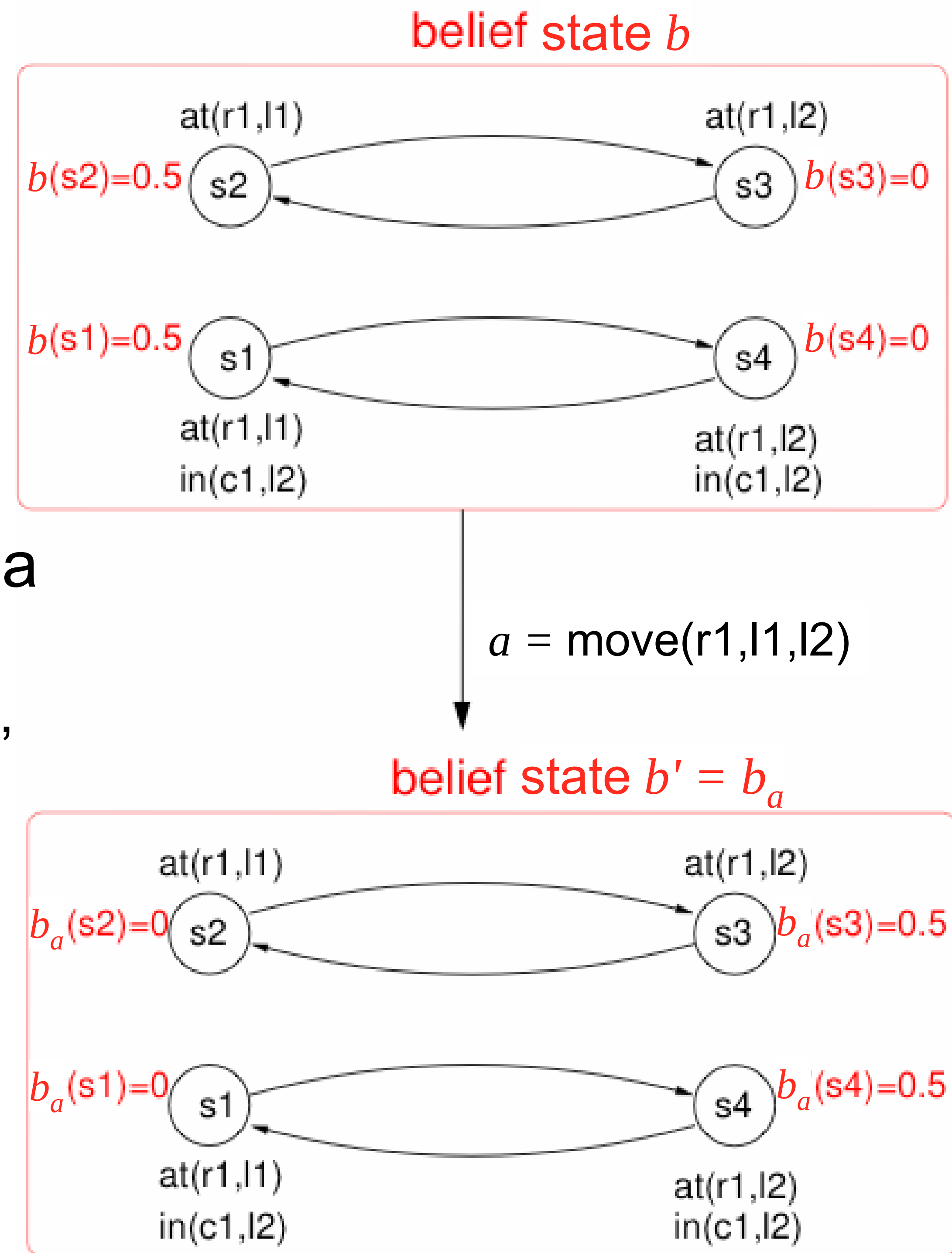
Example (Continued)

- By itself, see doesn't tell us the state with certainty
  - $b_{see}^{e}(s3)$
    $= P_{see}(e|s3) * b_{see}(s3) / b_{see}(e)$
    $= 1 * 0.25 / 0.5 = 0.5$
- If we first do a=move(l1,l2) then do see, this will tell the state with certainty
  - Let $b' = b_a$
  - $b'^{e}_{see}(s3)$
    $= P_{see}(e|s3) * b'_{see}(s3) / b'_{see}(e)$
    $= 1 * 0.5 / 0.5 = 1$



Modified Example

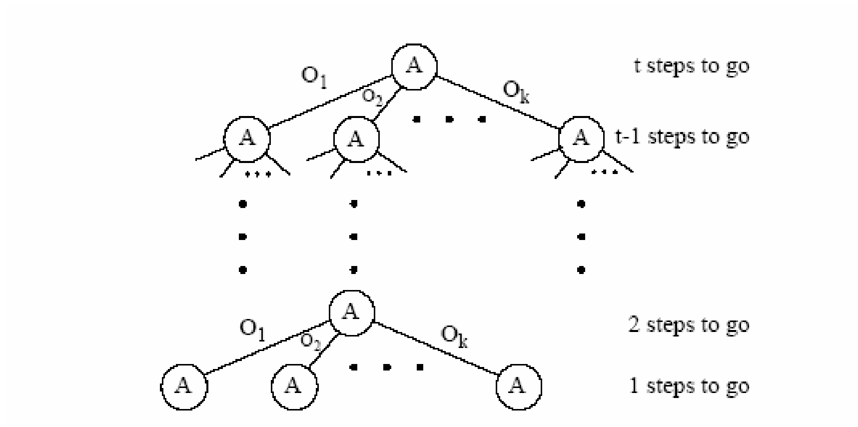
- Suppose we know the initial belief state is *b*
- Policy to tell if there's a container in l2:
  - π = {(*b*, move(r1,l1,l2)), (*b*′, see)}



Solving POMDPs

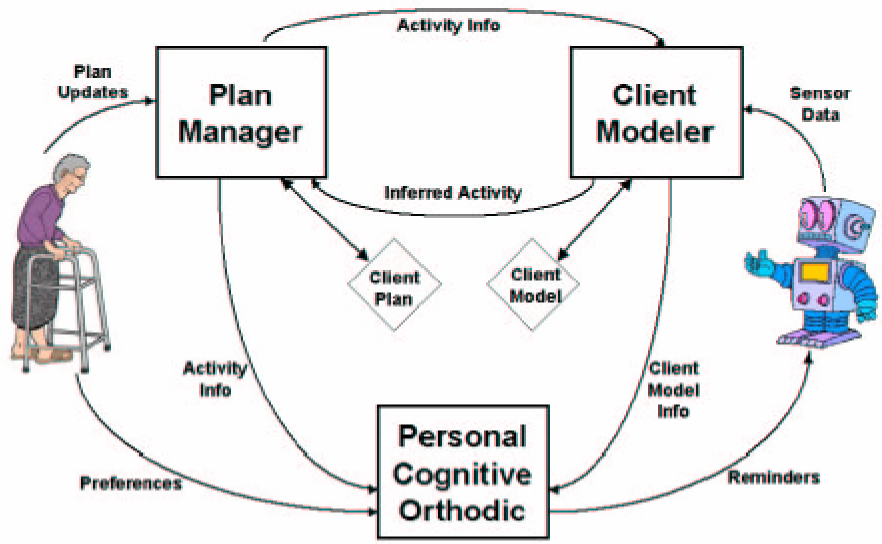
- Information-state MDPs
  - Belief states of POMDP are states in new MDP
  - Continuous state space
  - Discretise
- Policy-tree algorithms

Policy Trees



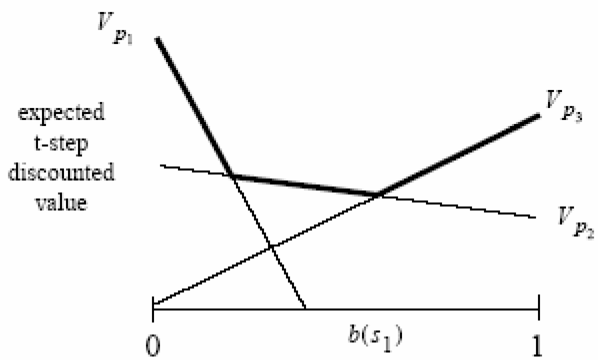
- Policy tree: an agent's non-stationary t-step policy
- Tree(a,T) – create a new policy tree with action a at root and observation z=T(z)
- Vp – vector for value function for policy tree p with one component per state
- Act(p) – action at root of tree p
- Subtree(p,z) – subtree of p after obs z
- Stval(a,z,p) – vector for probability-weighted value of tree p after a,z

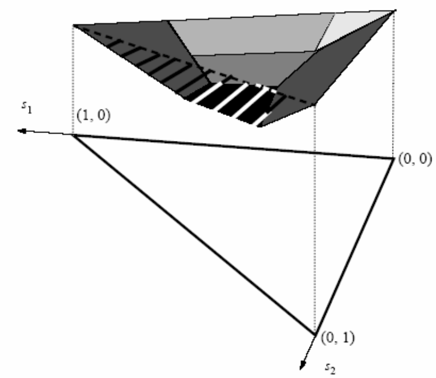
Application: Nursebot



- Robot assists elderly patients
- Model uncertainty about the user's dialog and position
- Exploit hierarchical structure to handle large state space

Value Functions





3-state

2-state

References

- Most slides were taken from Eyal Amir’s course, CS 598, Decision Making under Uncertainty (lectures 12 and 13)
- L. Kaebling, M. Littman, and A. Cassandra, Planning and Acting in Partially Observable Stochastic Domains, Artificial Intelligence, Volume 101, pp. 99-134, 1998