Representational and computational issues in uncertainty in robotics: survey and challenges

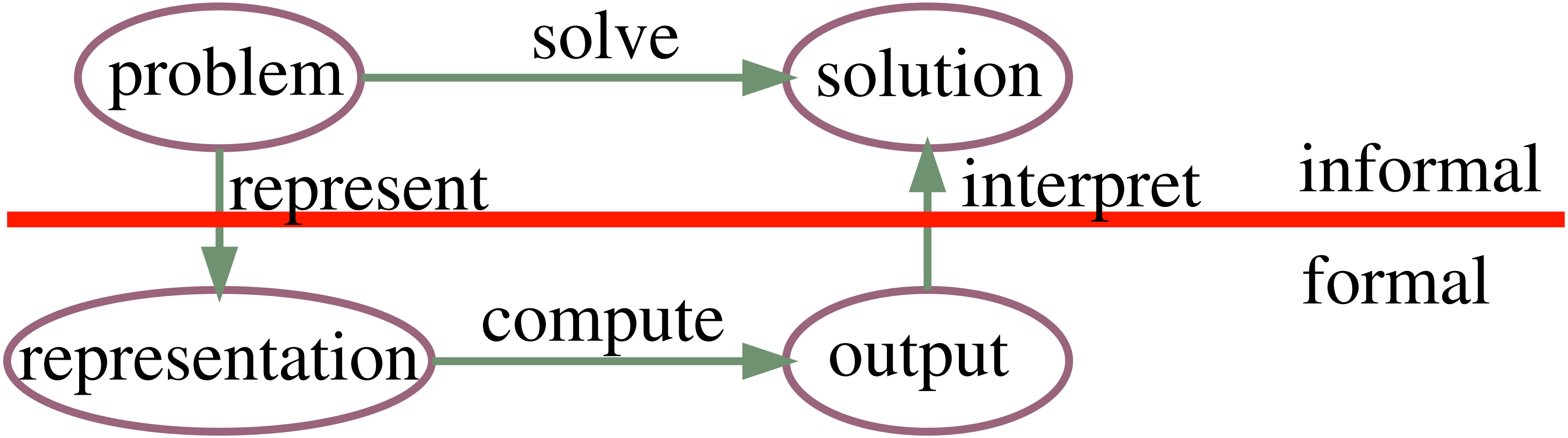
David Poole

University of British Columbia

Overview

- Knowledge representation, Belief Networks
- Uncertainty and Time
- Control
- Learning
- Challenges

Knowledge Representation



What do we want in a representation?

We want a representation to be

- rich enough to express the knowledge needed to solve the problem.
- as close to the problem as possible: compact, natural and maintainable.
- amenable to efficient computation;
  able to express features of the problem we can exploit for computational gain.
- learnable from data and past experiences.
- able to trade off accuracy and computation time.

Bayesians

- Interested in action: what should an agent do?
- Role of belief is to make good decisions.
- Theorems (Von Neumann and Morgenstern):
  (under reasonable assumptions) a rational agent will act as though it has (point) probabilities and utilities and acts to maximize expected utilities.
- Probability as a measure of belief:
  study of how knowledge affects belief
  lets us combine background knowledge and data

Representations of uncertainty

We want a representation for

- probabilities
- utilities
- actions

that facilitates finding the action(s) that maximise expected utility.

Belief networks (Bayesian networks)

- Totally order the variables of interest: $X_1, \ldots, X_n$
- Theorem of probability theory (chain rule):

$$\begin{aligned} P(X_1, \ldots, X_n) &= P(X_1)P(X_2|X_1)\cdots P(X_n|X_1, \ldots, X_{n-1}) \\ &= \prod_{i=1}^{n} P(X_i|X_1, \ldots, X_{i-1}) \end{aligned}$$

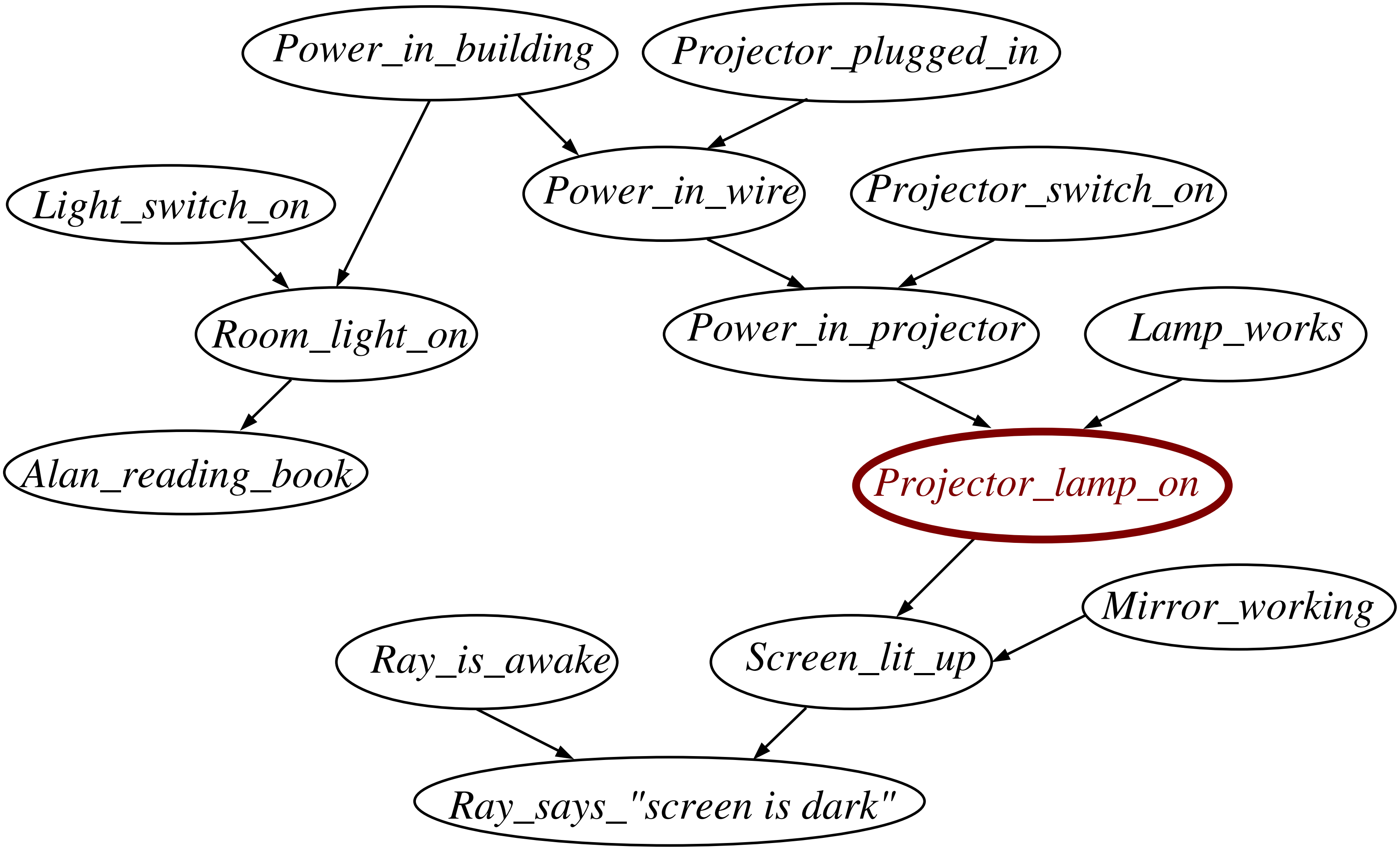
- The parents of X_i $\pi_i \subseteq X_1, \ldots, X_{i-1}$ such that

$$P(X_i|\pi_i) = P(X_i|X_1, \ldots, X_{i-1})$$

- So $P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\pi_i)$

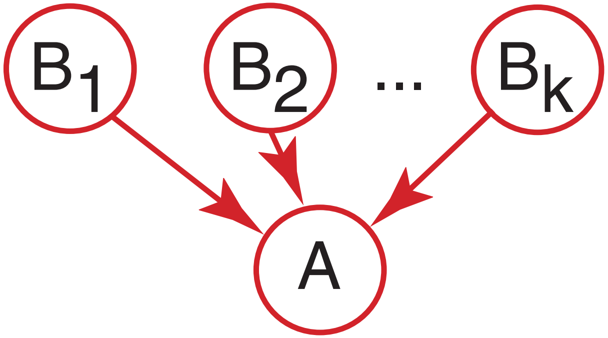
➥ Belief network nodes are variables, arcs from parents

Belief Network for Overhead Projector



Belief Network

- Graphical representation of dependence.
- DAG with nodes representing random variables.
- If $B_1, B_2, \cdots, B_k$ are the parents of A:



we have an associated conditional probability:

$$P(A|B_1, B_2, \cdots, B_k)$$

Probabilistic Inference

To compute the probability of a variable X given evidence $Z_1 = e_1 \wedge \ldots \wedge Z_k = e_k$:

$$P(X | Z_1 = e_1 \wedge \ldots \wedge Z_k = e_k) = \frac{P(X \wedge Z_1 = e_1 \wedge \ldots \wedge Z_k = e_k)}{P(Z_1 = e_1 \wedge \ldots \wedge Z_k = e_k)}$$

Suppose the other variables are $Y_1, \ldots, Y_m$:

$$\begin{aligned} P(X \wedge Z_1 \wedge \cdots \wedge Z_k) &= \sum_{Y_m} \cdots \sum_{Y_1} P(X_1, \ldots, X_n) \\ &= \sum_{Y_m} \cdots \sum_{Y_1} \prod_{i=1}^{n} P(X_i | \pi_i) \end{aligned}$$

Eliminating a variable

- to compute $AB + AC$ efficiently, distribute out A: $A(B + C)$.

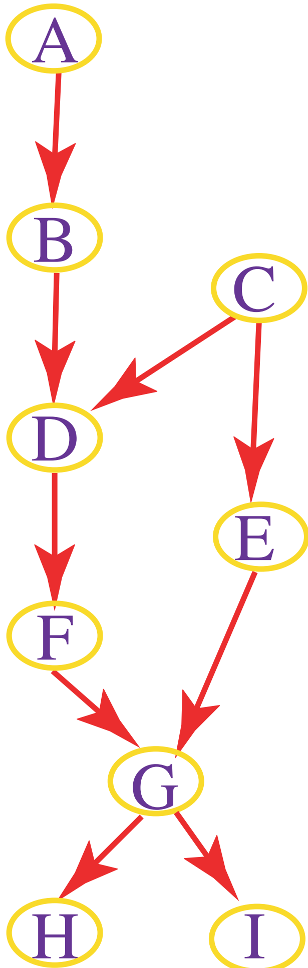
- to compute

$$\sum_{Y_j} \prod_{i=1}^{n} P(X_i|\pi_i)$$

  distribute out those factors that don't involve Y_j.

- Closely related to nonserial dynamic programming [Bertelè & Brioschi, 1972]

Variable Elimination Example



$$\left.\begin{array}{l} P(A) \\ P(B|A) \end{array}\right\} \xrightarrow{\text{elim } A} f_1(B)$$

$$\left.\begin{array}{l} P(C) \\ P(D|BC) \\ P(E|C) \end{array}\right\} \xrightarrow{\text{elim } C} f_2(BDE)$$

$$P(F|D)$$

$$P(G|FE)$$

$$P(H|G) \Big\} \xrightarrow{\text{obs } H} f_3(G)$$

$$P(I|G) \Big\} \xrightarrow{\text{elim } I} f_4(G)$$

Representing Factors

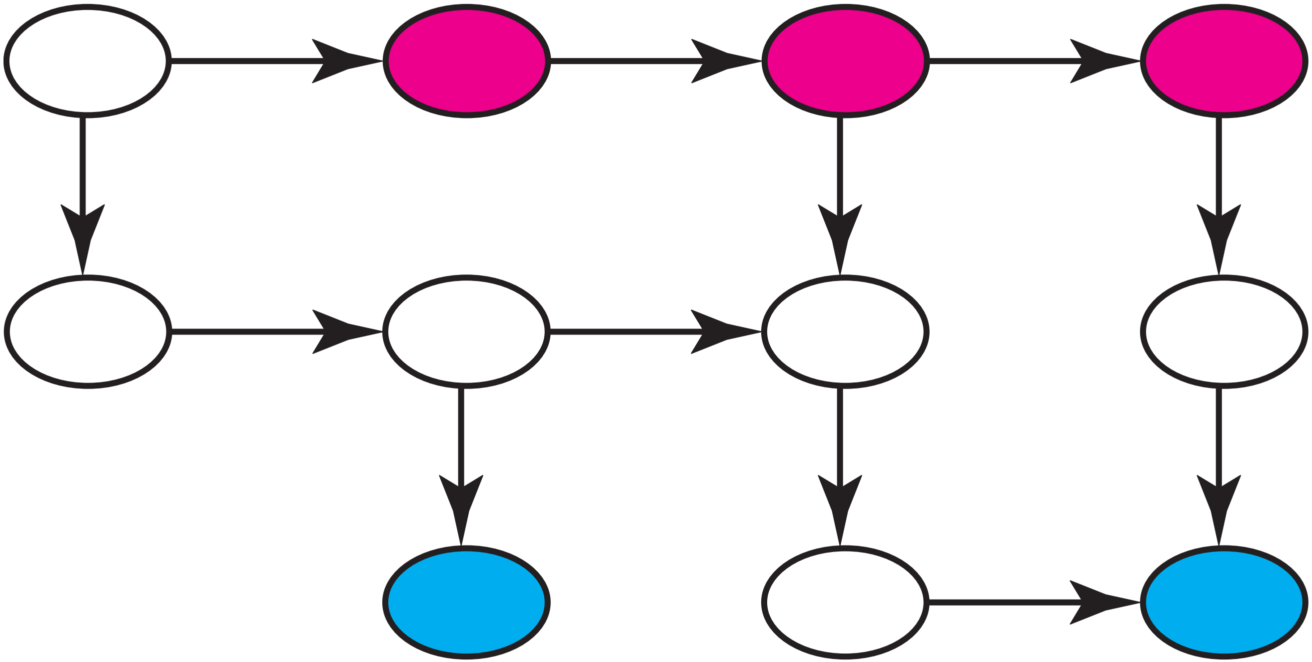
- **Tables** allow for fast indexing
- **Decision trees or rules** allow us to exploit contextual independence
- **Functional Forms** allow us to exploit special forms e.g., causal independence, mixtures of Gaussians
- **Caching** lets us save repeated computation Clique tree propagation = variable elimination + caching

Stochastic Simulation

- $P(x) = 0.234 \leftrightarrow$ in 234 out of 1000 random samples, x will be true.
- $P(x|evidence) = 0.654 \leftrightarrow$ out of every 1000 cases where *evidence* is true, x will also be true in 654 of them.
- Rejection sampling generate 1000 samples where *evidence* is true, estimate the probability of x from these.
- To sample in a belief network: sample parents, sample the variable from the distribution given the parents. Reject a sample that is in conflict with the evidence.

Mixing Exact & Stochastic Simulation

- If we can generate $P(sample|evidence)$ we can weight the sample by that amount.
- Importance sampling



Particle Filtering

- **Idea:** if you have a number of samples "particles" each with (posterior) probability, you can resample these according to their probability.
- particle filtering = importance sampling + resampling

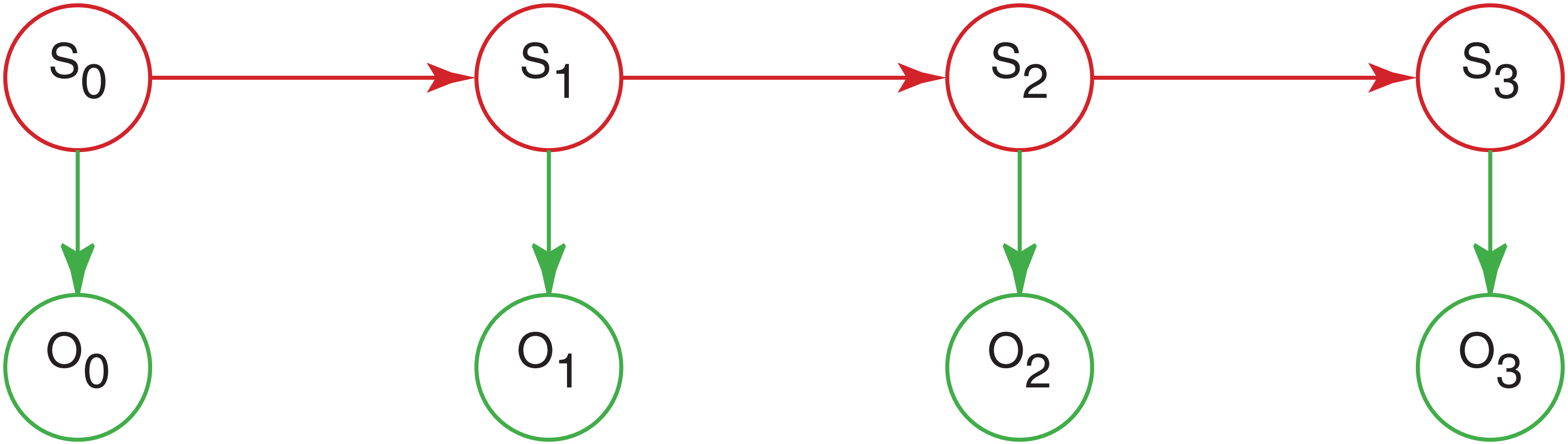
Overview

- Knowledge representation, Belief Networks
- **Uncertainty and Time**
  - Markov Chains
  - Hidden Markov Models
  - HMMS for Localization
- Control
- Learning
- Challenges

Markov Process

- $P(S_{t+1}|S_t)$ specifies the dynamics.
- $P(S_0)$ specifies the initial conditions.

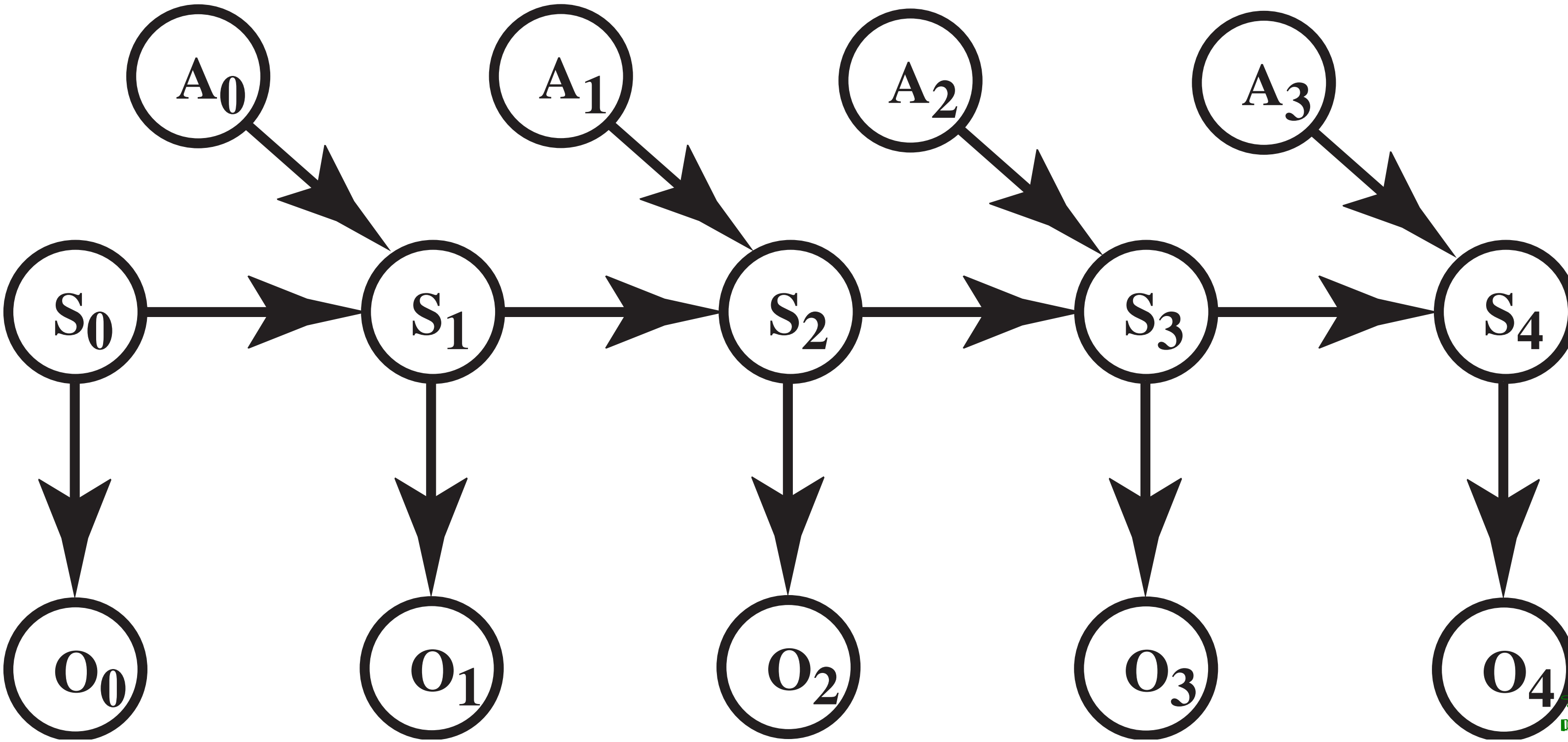
Hidden Markov Model



- $P(S_{t+1}|S_t)$ specifies the dynamics
- $P(S_0)$ specifies the initial conditions
- $P(O_t|S_t)$ specifies the sensor model.
- To find $P(S_i|observations)$ eliminate state variables before S_i and those after S_i. filtering smoothing

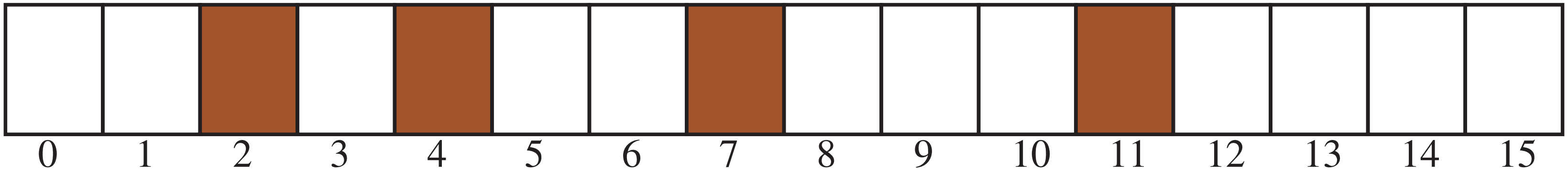
Example: localization

- Suppose a robot wants to determine its location based on its actions and its sensor readings. Called Localization
- This can be represented by the augmented HMM:



Example localization domain

- Circular corridor, with 16 locations:



- Doors at positions: 2, 4, 7, 11.
- Noisy Sensors
- Stochastic Dynamics
- Robot starts at an unknown location and must determine where it is.

Example Sensor Model

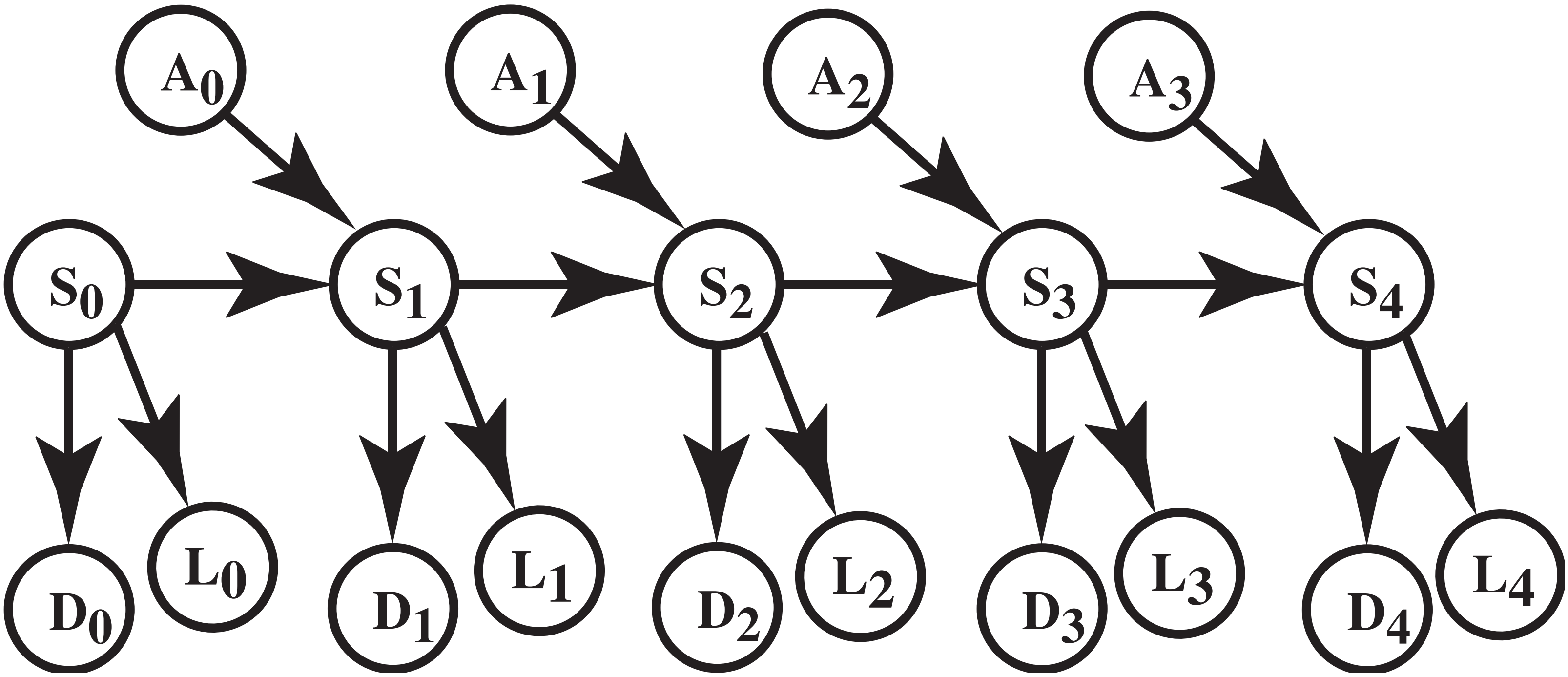
- $P(Observe\ Door \mid At\ Door) = 0.8$
- $P(Observe\ Door \mid Not\ At\ Door) = 0.1$

Example Dynamics Model

- $P(loc_{t+1} = L | action_t = goRight \wedge loc_t = L) = 0.1$
- $P(loc_{t+1} = L + 1 | action_t = goRight \wedge loc_t = L) = 0.8$
- $P(loc_{t+1} = L + 2 | action_t = goRight \wedge loc_t = L) = 0.074$
- $P(loc_{t+1} = L' | action_t = goRight \wedge loc_t = L) = 0.002$
  for any other location L'.
  - All location arithmetic is modulo 16.
  - The action *goLeft* works the same but to the left.

Sensor Fusion

- We can have many (noisy) sensors for a property.
- Example:



D_t is value of door sensor, L_t value of light sensor at time t.

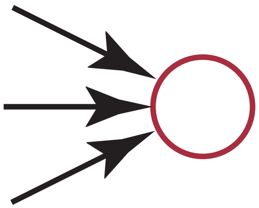
Overview

- Knowledge representation, Belief Networks
- Uncertainty and Time
- Control
  - Utilities and Actions
  - Decision Networks
  - MPDs
  - POMDPs
- Learning
- Challenges

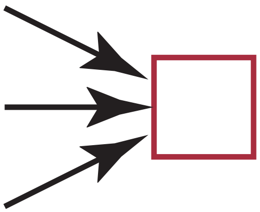
Goals and Utilities

- With goals, there are some equally preferred **goal states,** and all other states are equally bad.
- Not all failures are equal. **For example:** a robot stopping, falling down stairs, or injuring people.
- With uncertainty, we have to consider how good and bad all possible outcomes are.
  - ➥ **utility** specifies a value for each state.
- With utilities, we can model goals by having goal states having utility 1 and other states have utility 0.

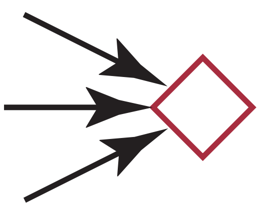
Decisions Networks



- A **random variable** is drawn as an ellipse. Arcs into the node represent probabilistic dependence.

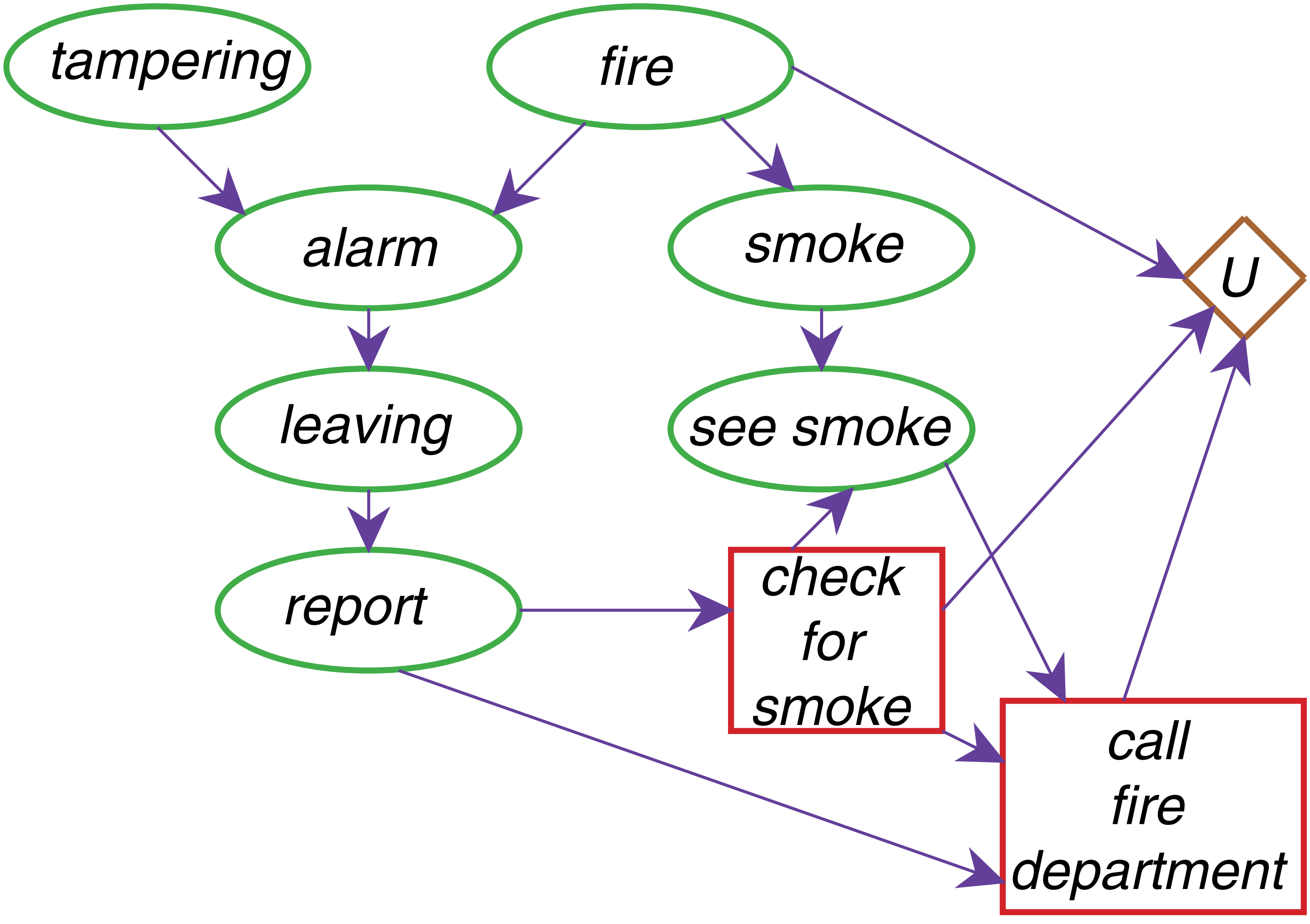


- A **decision variable** is drawn as an rectangle. Arcs into the node represent information available when the decision is make.

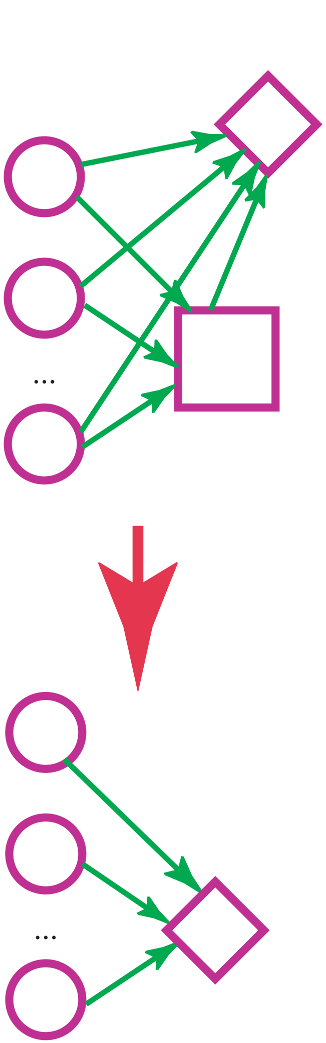


- A **value** node is drawn as a diamond. Arcs into the node represent values that the value depends on.

Example Decision Network



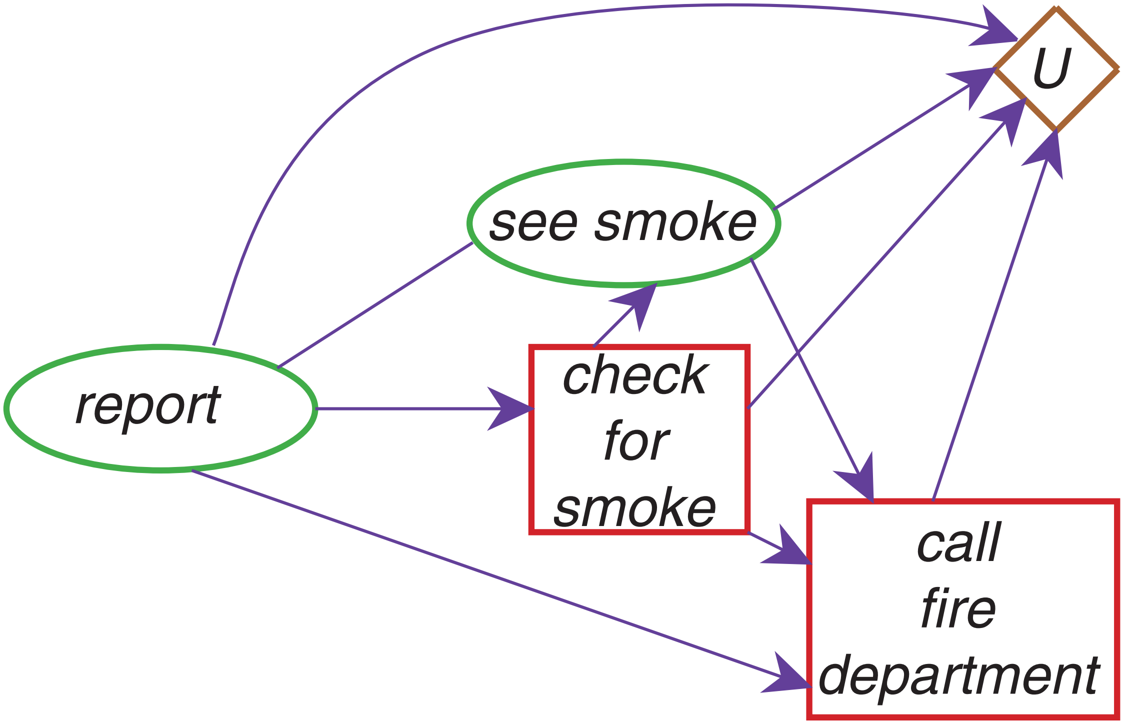
Finding an Optimal Decision



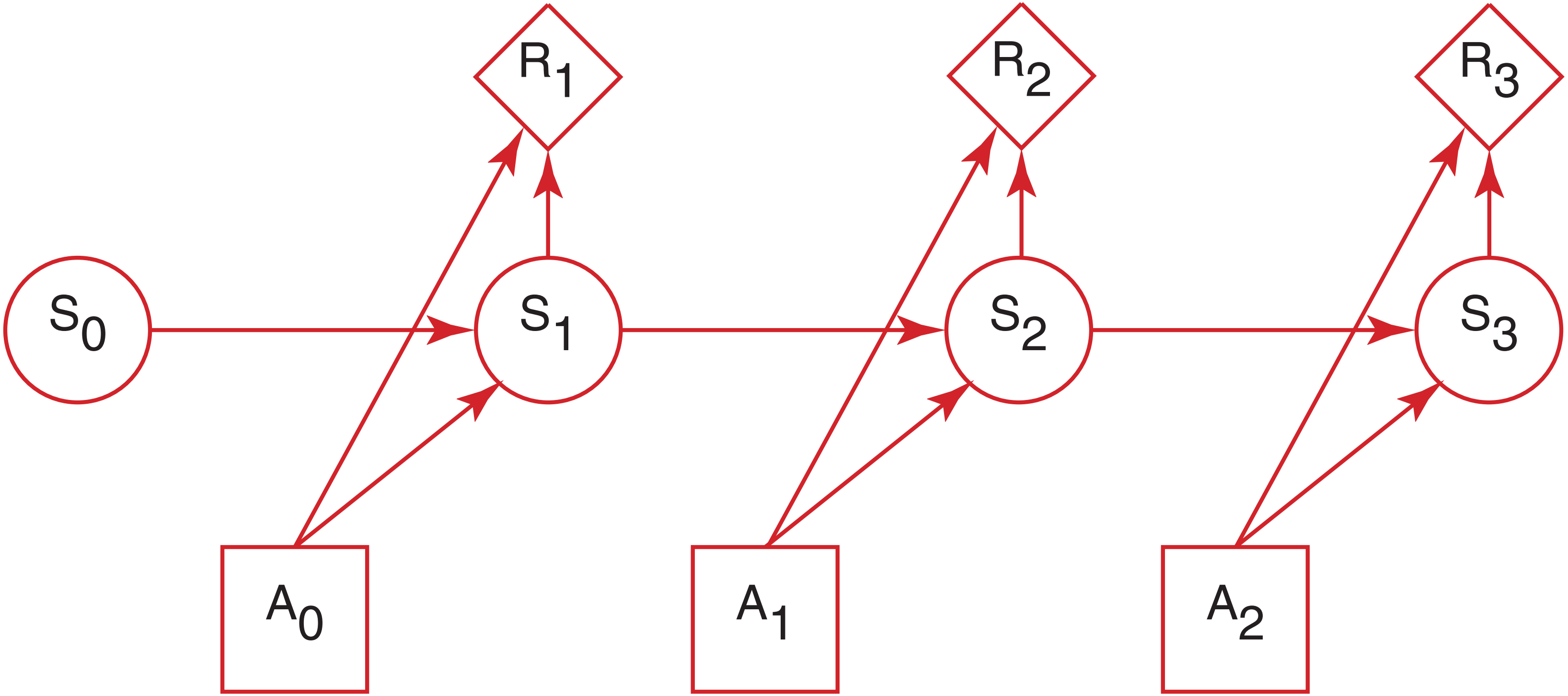
- If value node is only connected to a decision node and (some of) its parents
  ➥ select a decision to maximize value for each assignment to the parent.
- If it isn't of this form, eliminate the non-observed variables.
- If there are k binary parents, there are 2^k optimizations.
- There are 2^{2^k} policies.
- Replace decision node with value node.

Evaluating Decision Networks

Eliminate the non-observed variables for the final decision.



(Finite stage) Markov Decision Process



$P(S_{t+1}|S_t, A_t)$ specified the dynamics

$R(S_t, A_{t-1})$ specifies the reward at time t

Value is $R_1 + R_2 + R_3$.

Policies

- What the agent does based on its perceptions is specified by a policy.
- We assume that the agent can observe it's state (and remember its history).
- If we eliminate the final state, we have a form of the trivial decision problem. value iteration
- Optimal action is a function from observed state into action. A policy is a set of functions $S_i \rightarrow A_i$.

Modelling Assumptions

- deterministic or stochastic dynamics
- goals or utilities
- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents
- perfect rationality or bounded rationality

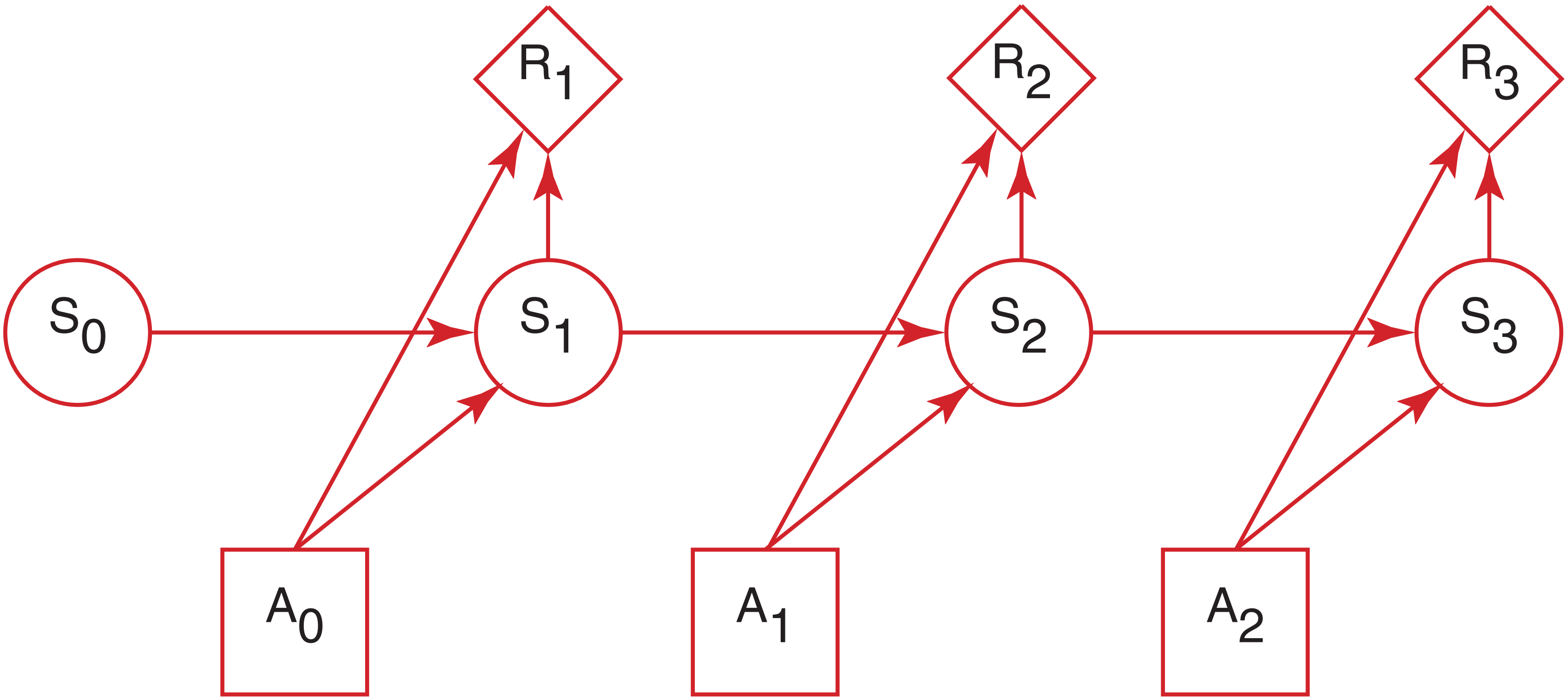
Dimensions of Representations

- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents
- perfect rationality or bounded rationality

Finite stage or infinite stage

- **Finite stage** there is a given number of sequential decisions
- **Infinite stage** indefinite number (perhaps infinite) number of sequential decisions.
- With infinite stages, we can model stopping by having an absorbing state — a state s_i so that $P(s_i|s_i) = 1$, and $P(s_j|s_i) = 0$ for $i \neq j$.
- Infinite stages let us model ongoing processes as well as problems with unknown number of stages.

Markov Decision Process



$P(S_{t+1}|S_t, A_t)$ specified the dynamics

$R(S_t, A_{t-1})$ specifies the reward at time t

Markov Decision Process

- Infinite stage is the limit as horizon gets larger
- Total value of a policy:
  - Sum of rewards (only with absorbing states)
  - Discounted reward $R_1 + \gamma R_2 + \gamma^2 R_3 + \ldots.$
  - Average reward $\lim_{n \to \infty}(R_1 + R_2 + \ldots + R_n)/n.$
- Usually have stationary dynamics: time-independent.
- Two main algorithms
  - Policy iteration: evaluate then improve a given policy.
  - Value iteration: determine the value of the optimal policy working backwards from some point in time.

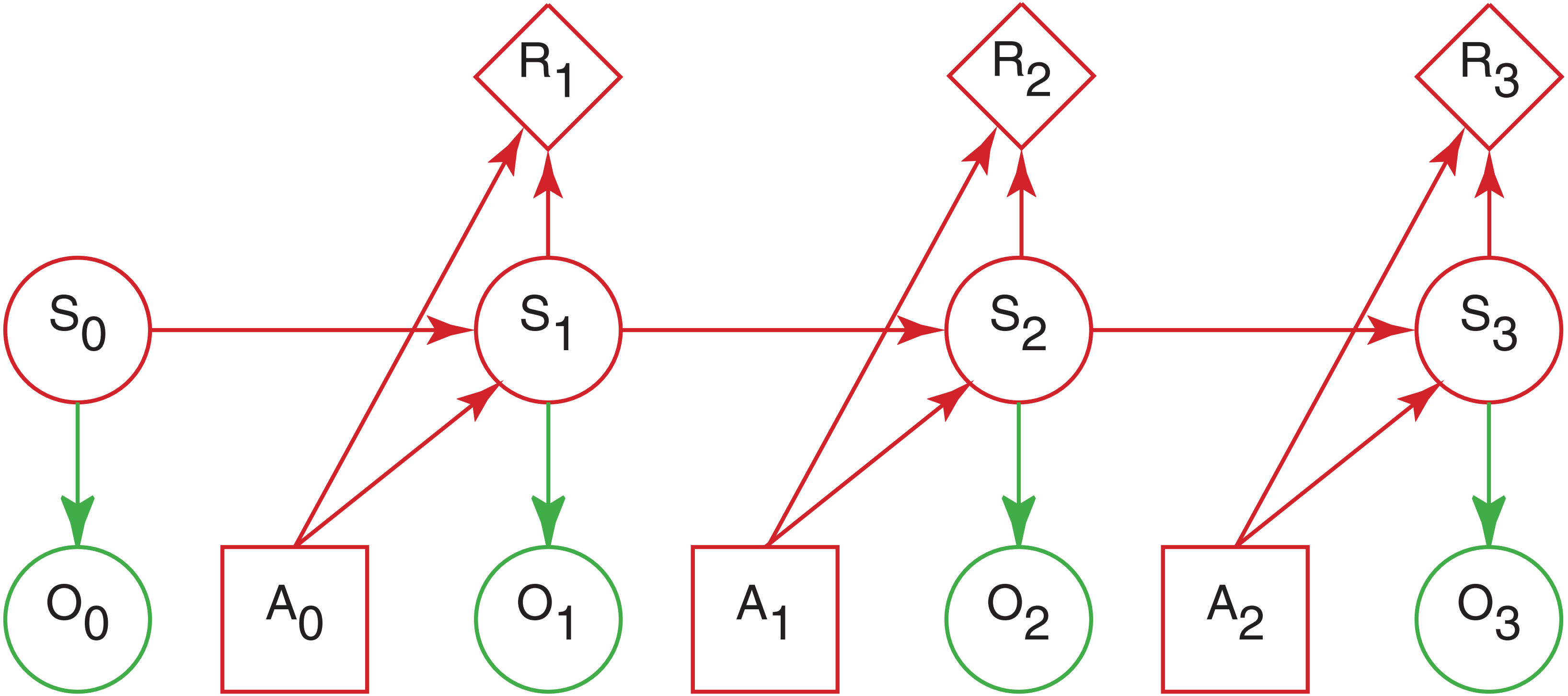
Dimensions of Representations

- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents
- perfect rationality or bounded rationality

Fully observable or partially observable

- Fully observable = can observe actual state before a decision is made.
- Full observability is a convenient assumption that makes computation much simpler.
- Full observability is applicable only for artificial domains, such as games and factory floors.
- Most domains are partially observable, such as robotics, diagnosis, user modelling …

(Finite stage) Partially Observable MDP



$P(S_{t+1}|S_t, A_t)$ specified the dynamics

$P(O_t|S_t)$ specifies the sensor model.

$R(S_t, A_{t-1})$ specifies the reward at time *i*

Policies for Finite Stage POMDPs

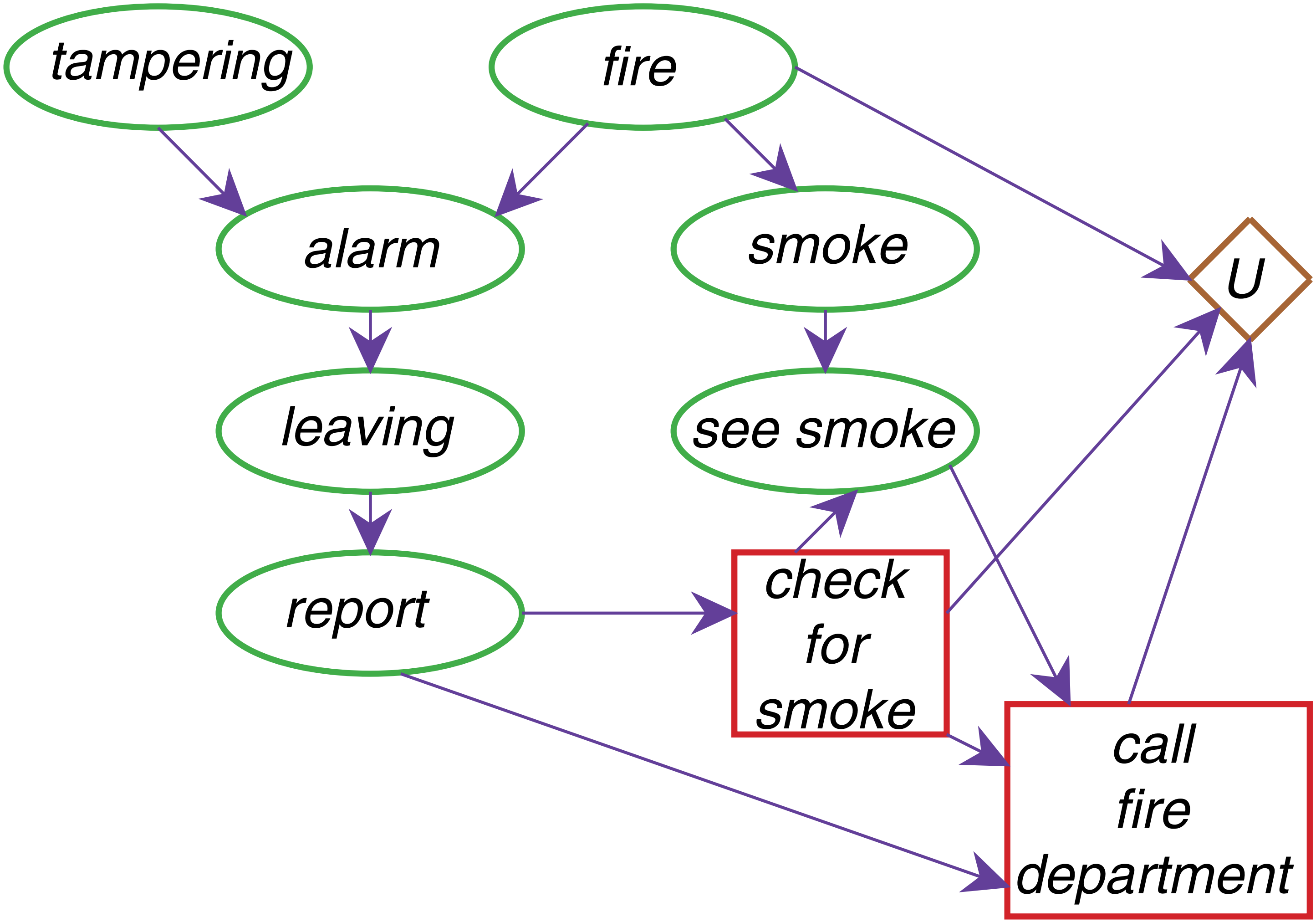
- The information available to the agent at any time is the history of observations and previous actions. Assume the agent is no forgetting.
- What the agent should do is specified by a policy a function from history into actions. For each time t we have:

$$O_0, A_0, O_1, A_1, \ldots, O_{t-1}, A_{t-1}, O_t \rightarrow A_t$$

Dimensions of Representations

- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents
- perfect rationality or bounded rationality

Example Decision Network



Dimensions of Representations

- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents
- perfect rationality or bounded rationality

Policies for Infinite Stage POMDPs

- We can't define a function over the infinite history (unless we cut it off to a finite part somehow).
- A belief state is a probability distribution over states. A belief state is an adequate statistic about the history.

$$policy : B_t \rightarrow A_t$$

  - If there are n states, this is a function on $\Re^n$.
  - If there are only finitely many stages to go, the optimal value function is piecewise linear and convex (the agent can adopt one of a finite number of conditional plans; each of these represents a hyperplane in belief space).

Dimensions of Representations

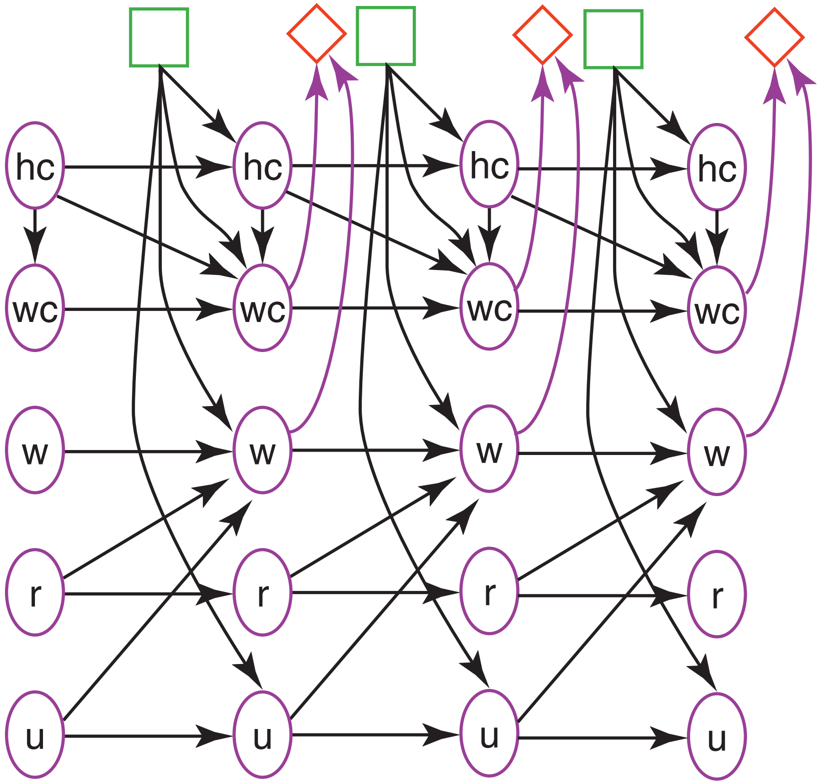
- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents
- perfect rationality or bounded rationality

Explicit state space or properties

- Traditional methods relied on explicit state spaces, and techniques such as sparse matrix computation.
- The number of states is exponential in the number of properties or variables. It may be easier to reason with 30 binary variables than 1,000,000,000 states.
- Bellman labelled this the *Curse of Dimensionality*.

Dynamic Decision Networks

Idea: represent the state in terms of random variables / propositions.



Finding Optimal Policies

- Eliminate the non-observed variables that are not d-separated from the value node by the parents of the last decision.
- Nodes become joined (values function depends on many variables).
- Same problem occurs with belief state monitoring.

Dimensions of Representations

- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents
- perfect rationality or bounded rationality

Zeroth-order or first-order

- The traditional methods are zero-order, there is no logical quantification. All of the individuals must be part of the explicit model.
- There is a lot of work on automatic construction of probabilistic models — providing macros to construct ground representations.

First-order representations

- We want to be able to quantify over individuals, and have relations amongst individuals.
- First-order languages allow recursion.
- We want to be able to exploit first-order representation computationally—as unification does for theorem proving. One step of first-order algorithm corresponds to many ground steps.
- Lets us reason about populations. Someone is running about, what is the probability that someone else is too?

Independent Choice Logic

- We want a first-order language where all uncertainty is handled by Bayesian decision theory (probabilities, agent choices, utilities) rather than by disjunction.
- We start with a language with no uncertainty
  - acyclic logic programs
- We have a choice space of independent choices + a logic program that gives the consequences of the choices.
- Direct mapping from a belief/decision network to ICL.

Independent Choice Logic Semantics

The user specifies a choice space + acyclic logic program

- An alternative is a set of first-order atoms exactly one of which can be true.
- A choice space is a set of pairwise disjoint alternatives.
- A possible world is the selection of one element from each alternative.
- What is true in the possible world is defined by which elements are selected and the logic program.
- We have a probability distribution over alternatives.

Dynamic Belief Networks in ICL

$$r(T+1) \leftarrow r(T) \wedge rain\_continues(T).$$

$$r(T+1) \leftarrow \overline{r(T)} \wedge rain\_starts(T).$$

$$hc(T+1) \leftarrow hc(T) \wedge do(A,T) \wedge A \neq pass\_coffee$$
$$\wedge\, keep\_coffee(T).$$

$$hc(T+1) \leftarrow hc(T) \wedge do(pass\_coffee, T)$$
$$\wedge\, keep\_coffee(T) \wedge passing\_fails(T).$$

$$hc(T+1) \leftarrow do(get\_coffee, T) \wedge get\_succeeds(T).$$

$$\forall T \{rain\_continues(T), rain\_stops(T)\} \in \mathbf{C}$$

$$\forall T \{keep\_coffee(T), spill\_coffee(T)\} \in \mathbf{C}$$

$$\forall T \{passing\_fails(T), passing\_succeeds(T)\} \in \mathbf{C}$$

Dimensions of Representations

- **finite stage** or infinite stage
- **fully observable** or partially observable
- explicit state space or **properties**
- **zeroth-order** or first-order
- **dynamics and rewards given** or learned
- single agent or **multiple agents**
- **perfect rationality** or bounded rationality

Single agent or multiple agents

- Many domains are characterised by multiple agents rather than a single agent.
- Game theory studies what agents should do in a multi-agent setting.
- Agents can be cooperative, competitive or somewhere in between.
- Agents that are strategic can't be modelled as nature.

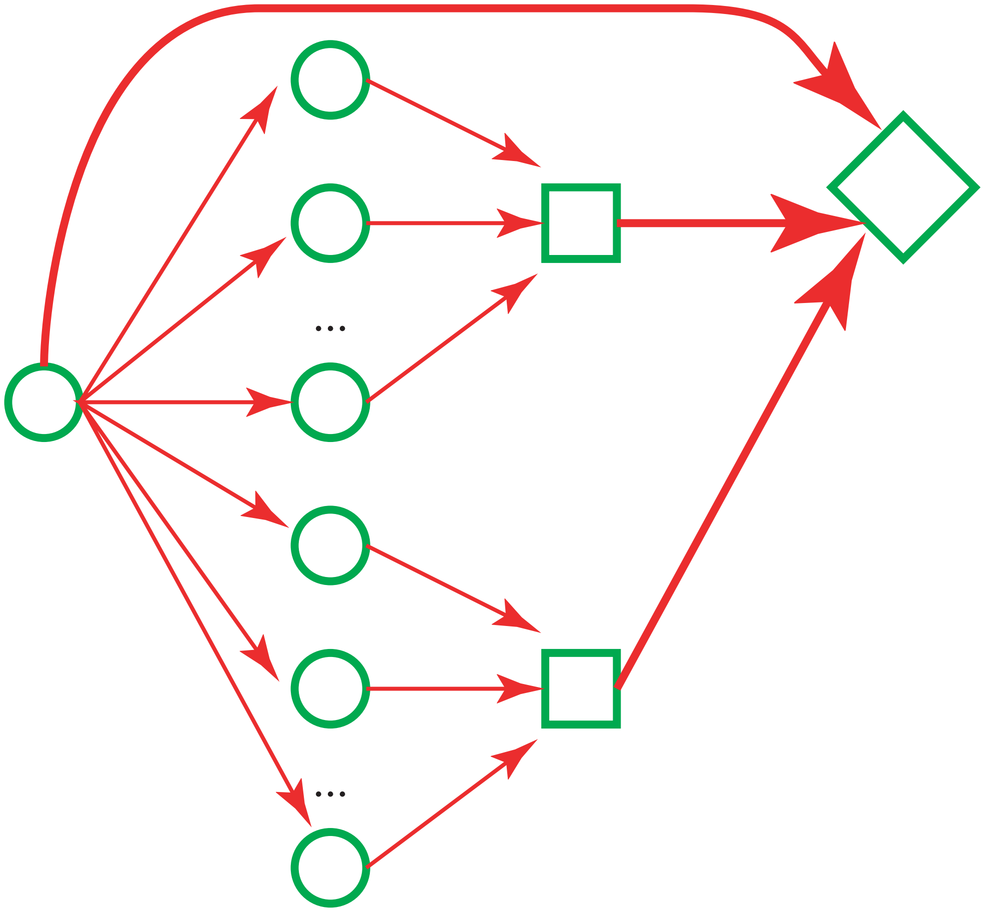
Fully Observable + Multiple Agents

- Perfect Information Games.
- Can do dynamic programming or search:
  Each agent maximises for itself.
- Two person, competitive (zero sum) $\Longrightarrow$ minimax.

Dimensions of Representations

- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents
- perfect rationality or bounded rationality

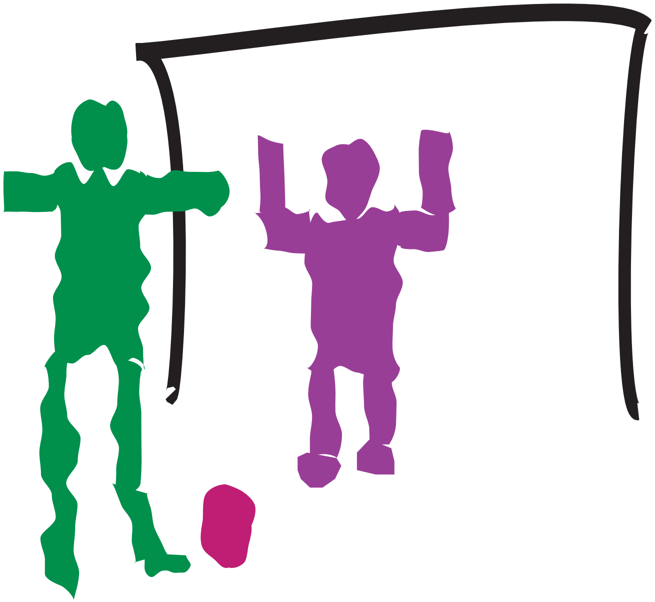
Multiple Agents, shared value



Complexity of Multi-agent decision theory

- It can be exponentially harder to find optimal multi-agent policy even with a shared values.
- Why? Because dynamic programming doesn't work:
  - If a decision node has n binary parents, DP lets us solve 2^n decision problems.
  - This is much better than d^{2^n} policies (where d is the number of decision alternatives).
- Multiple agents with shared values is equivalent to having a single forgetful agent.

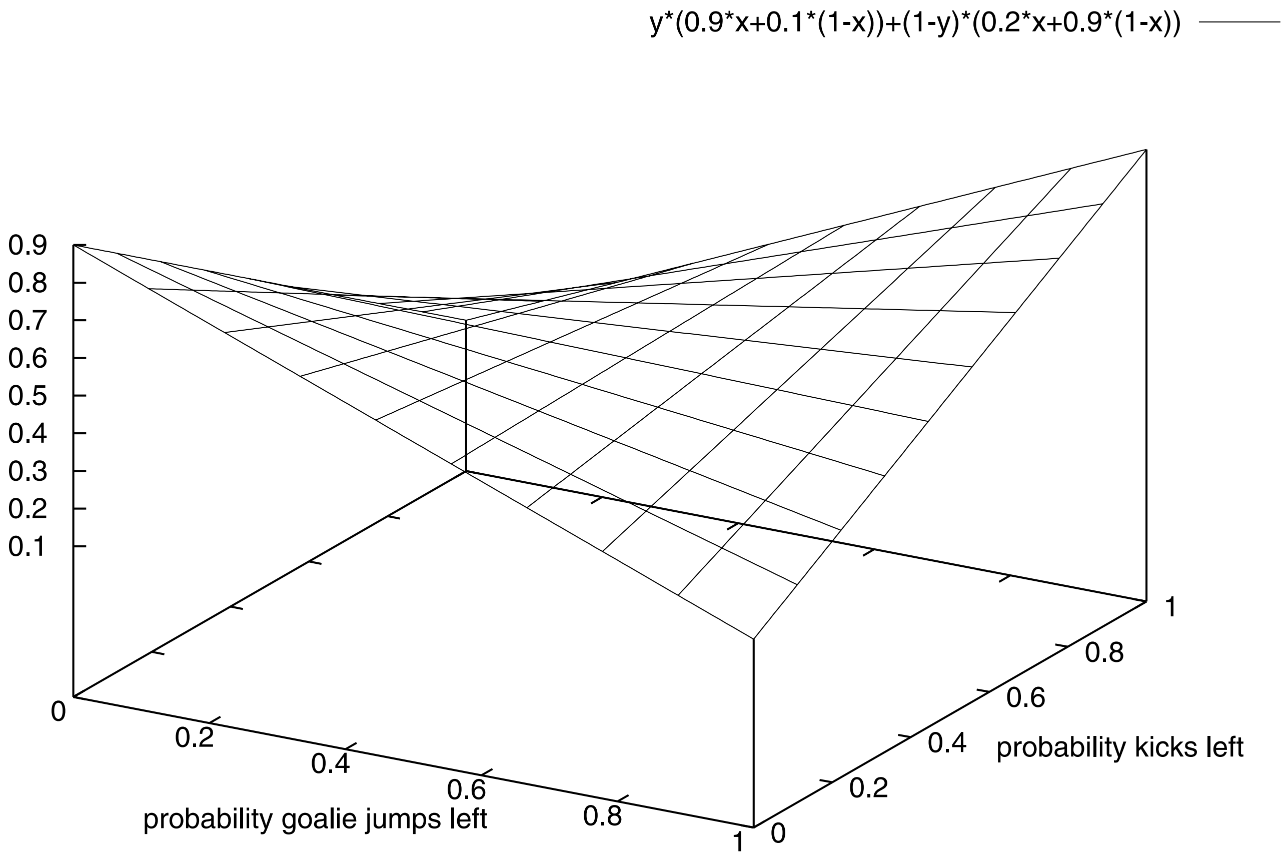
Partial Observability and Competition



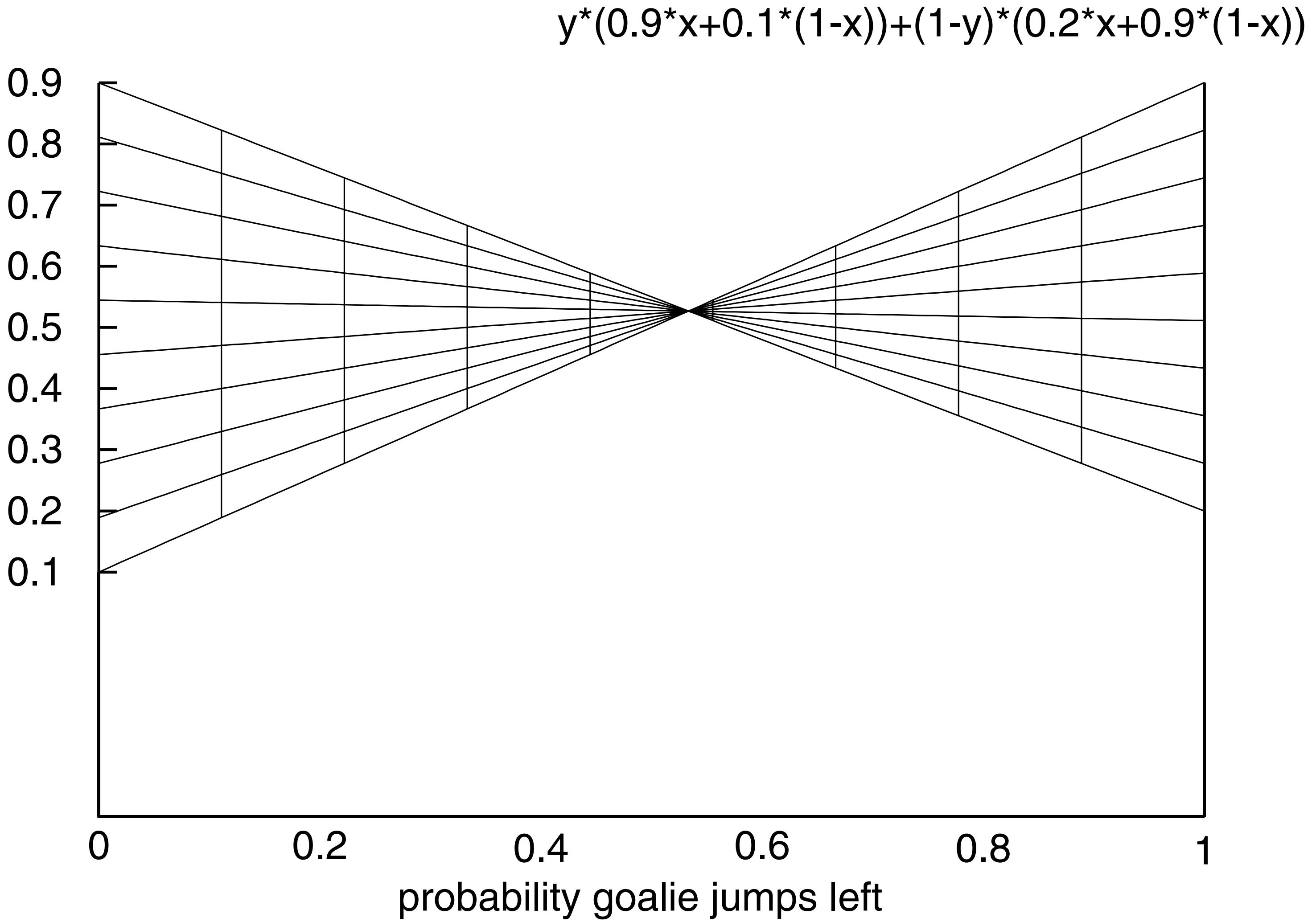
		goalie	
		left	right
kicker	left	0.9	0.1
	right	0.2	0.9

Probability of a goal.

Stochastic Policies



Stochastic Policies—another view



Dimensions of Representations

- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents
- perfect rationality or bounded rationality

Perfect or Bounded Rationality

- We cannot assume agents have unlimited computation time and space.
- It may be better to find a reasonable decision fast than take a long time to find what (was) the best decision.
- Value of computation. Value of space. How much is thinking worth to the agent?
- Offline versus online computation.

Overview

- Knowledge representation, Belief Networks
- Uncertainty and Time
- Control
- Learning
  - Parameter Learning
  - Hidden variables: EM
  - SLAM
  - Reinforcemment Learning
- Challenges

Parameter Learning

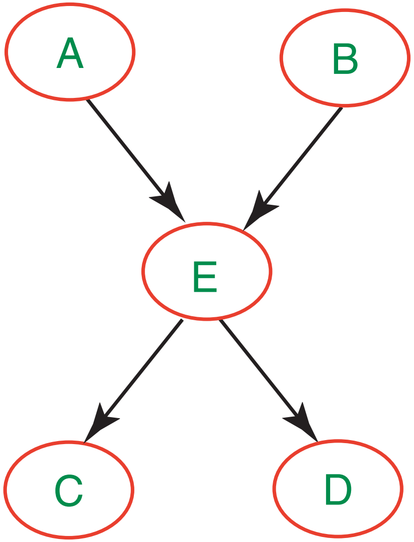
- data $\leftrightarrow$ probabilities
- Still problematic to determine appropriate function of parents.

Learning a Belief Network

- If you
  - know the structure
  - have observed all of the variables
  - have no missing data
- you can learn each conditional probability separately.

Learning belief network example

Model



Data

A	B	C	D	E
t	f	t	t	f
f	t	t	t	t
t	t	f	t	f
		$\ldots$		

→ Probabilities

$P(A)$

$P(B)$

$P(E|A,B)$

$P(C|E)$

$P(D|E)$

Learning conditional probabilities

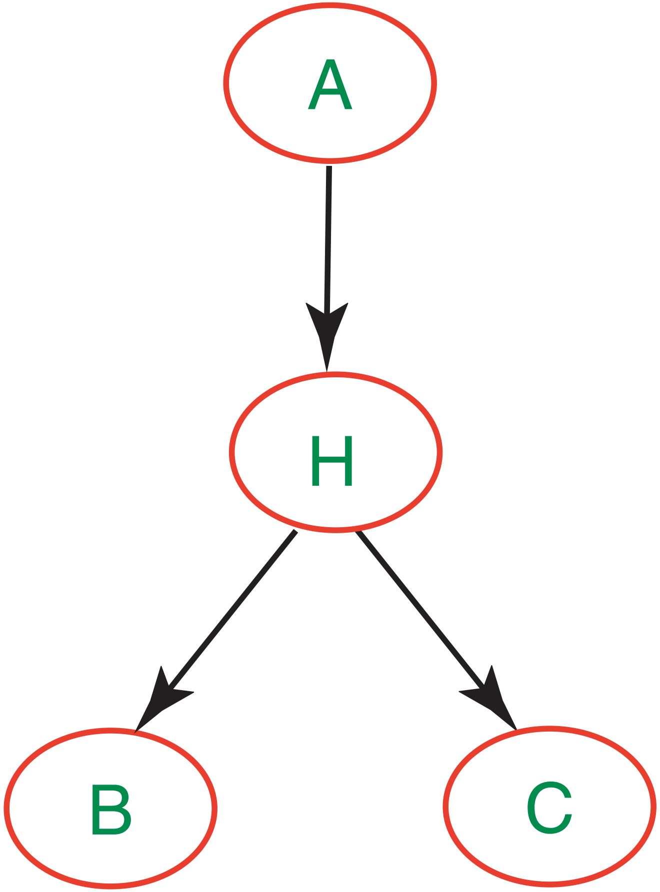
- Each conditional probability distribution can be learned separately:
- For example:

$$P(E = t | A = t \wedge B = f) = \frac{(\#\text{examples: } E = t \wedge A = t \wedge B = f) + m_1}{(\#\text{examples: } A = t \wedge B = f) + m}$$

where m_1 and m reflect our prior knowledge.

- There is a problem when there are many parents to a node as then there is little data for each probability estimate.

Unobserved Variables



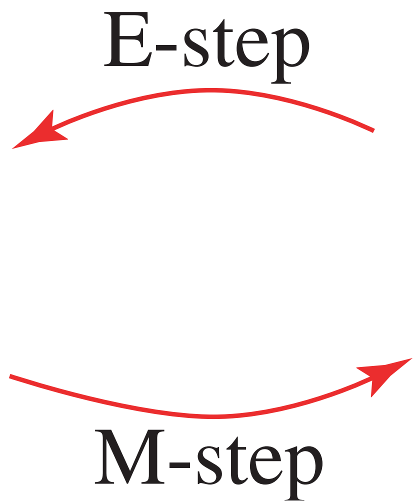
- What if we had only observed values for A, B, C?

A	B	C
t	f	t
f	t	t
t	t	f
	$\ldots$	

EM Algorithm

Augmented Data

A	B	C	H
t	f	t	t
f	t	t	f
t	t	f	t
	$\dots$		



Probabilities

$P(A)$

$P(H|A)$

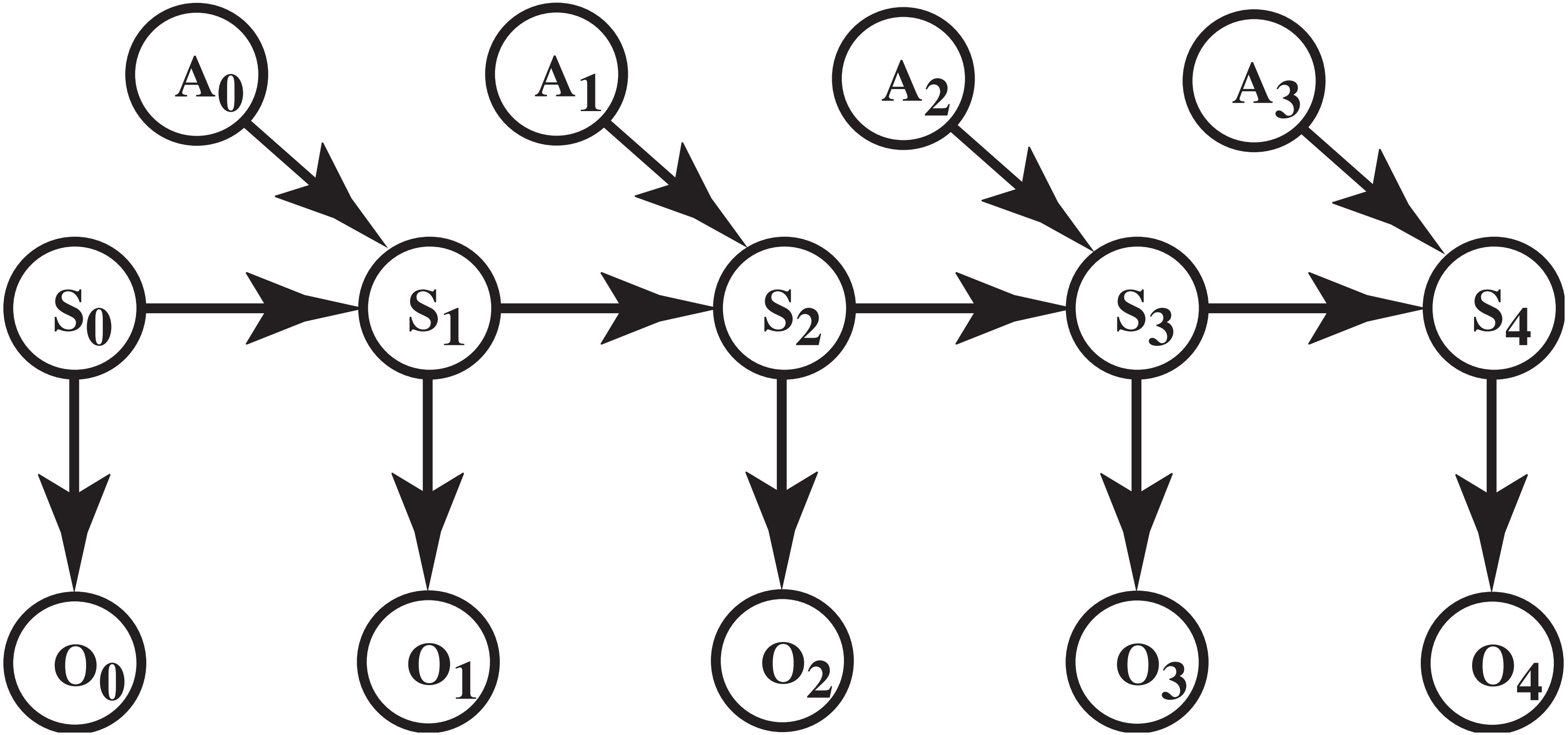
$P(B|H)$

$P(C|H)$

EM Algorithm

- Repeat the following two steps:
  - **E-step** give the expected number of data points for the unobserved variables based on the given probability distribution.
  - **M-step** infer the (maximum likelihood) probabilities from the data. This is the same as the full observable case.
- Start either with made-up data or made-up probabilities.
- EM will converge to a local maxima.

Simultaneous localization and mapping



- Don’t know dynamics or sensor model.
- Want a coherent map.

Reinforcement Learning

- Often we don't know a priori the probabilities and rewards, but only observe the system while controlling it ➥ reinforcement learning.
- Typically modelled as a Markov Decision Process
- Learn either:
  - dynamics + rewards **model-based** - use value or policy iteration
  - $Q(s, a)$ — value of doing a in state s then acting optimally
- Exploration—exploitation tradeoff.

Temporal Differences

To get the average of the first n data values:

$$\begin{aligned} A_n &= \frac{a_1 + \ldots + a_{n-1} + a_n}{n} \\ &= \frac{(a_1 + \ldots + a_{n-1})(n-1)}{(n-1)n} + \frac{a_n}{n} \\ &= \frac{n-1}{n} A_{n-1} + \frac{1}{n} a_n \end{aligned}$$

Let $\alpha = \frac{1}{n}$, then

$$\begin{aligned} A_n &= (1-\alpha) A_{n-1} + \alpha a_n \\ &= A_{n-1} + \alpha (a_n - A_{n-1}) \end{aligned}$$

Modelling Assumptions

- deterministic or stochastic dynamics
- goals or utilities
- finite stage or infinite stage
- fully observable or partially observable
- explicit state space or properties
- zeroth-order or first-order
- dynamics and rewards given or learned
- single agent or multiple agents

Comparison of Some Representations

	CP	DTP	IDs	RL	HMM	GT
stochastic dynamics		✔	✔	✔	✔	✔
values		✔	✔	✔		✔
infinite stage	✔	✔		✔	✔	
partially observable			✔		✔	✔
properties	✔	✔	✔	✔		✔
first-order	✔					
dynamics not given				✔	✔	
multiple agents						✔

Challenges

- Develop solutions to parts that fit together.
- Put them together.
- Some random subproblems:
  - modelling multiple objects
  - hierarchical decomposition
  - spatial reasoning and uncertainty
  - integrating with real sensors (e.g., vision)
  - specification of what we want our robots to do (values)

Where to now?

- Keep the representation as simple as possible to solve your problem, but no simpler.
- Approximate. Bounded rationality: costs and benefits of approximation.
- Approximate the solution, not the problem (Sutton).
- Reasoning at multiple levels of abstraction.
- We want everything, but only as much as it is worth to us.
- Preference elicitation.
- Uncertainty is everywhere. Be certain you are using it appropriately.