

MAKING COMPLEX DECISIONS

Outline

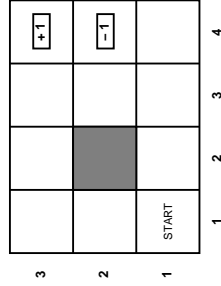
- Sequential Decision Problems
- Value Iteration
- Policy Iteration
- Decision-Theoretic Agent Design
- Dynamic Belief/Decision Networks

Based on the textbook by S. Russell & P. Norvig:

Artificial Intelligence, A Modern Approach, Chapter 17

SEQUENTIAL DECISION PROBLEMS

Example. Consider an agent situated in the following environment:



- The agent may perform actions *North*, *South*, *East*, and *West* in order to move between squares (or states) $(1, 1), \dots, (4, 3)$.
- Moving towards a wall results in no change in position.
- The operation of the agent stops and it receives a *reward/punishment* if it reaches a square marked with $+1/-1$.

Transition Model

- In a *deterministic* setting the outcomes of actions are known, and the agent may **plan** a sequence of actions which moves it to $(4, 3)$.
- This becomes impossible if actions are *nondeterministic/unreliable*.
- A **transition model** assigns a probability M_{ij}^a to the event that the agent reaches state j if it performs action a in state i .

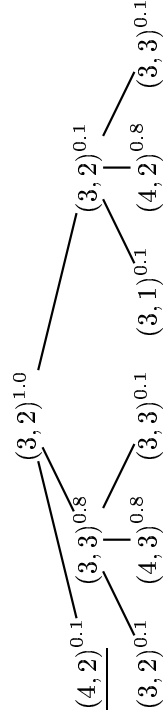
Example. Each one of the four actions *North*, *South*, *East*, and *West* moves the agent

1. to the intended direction d with a probability of 0.8, and
2. at right angles to the direction d with probabilities 0.1 and 0.1.

Example. If an action sequence $S = [North, East]$ is performed in state $(3, 2)$ the agent reaches states with following probabilities:

$$\begin{array}{rcl}
 P_{(3,1)} & = & 0.1 \times 0.1 = 0.01 \\
 P_{(3,2)} & = & 0.8 \times 0.1 = 0.08 \\
 P_{(3,3)} & = & 0.8 \times 0.1 + 0.1 \times 0.1 = 0.09 \\
 P_{(4,2)} & = & 0.1 + 0.1 \times 0.8 = 0.18 \\
 P_{(4,3)} & = & 0.8 \times 0.8 = 0.64 \\
 & & \hline
 & & 1.00
 \end{array}$$

These are easily inspected from a (partial) *reachability graph*:



Utilities and Plans

- Utility function U is based on a sequence of states (an **environment history**) rather than a single state.

Example. In our example, the utility is defined as the value of the terminal state (+1 or -1) minus $\frac{1}{25}$ of the length of the sequence.

- Considering sequences of actions as long actions implies committing to an entire sequence of actions before executing it.
- In practice, the agent should be able to choose a new action in each state *given any additional information provided by sensors*.
- In stochastic environments, plans have to be conditional and it may be impossible to set a limit for lengths of conditional plans.

© 2002 HUT / Laboratory for Theoretical Computer Science

Policies

- We concentrate on **accessible** environments where the agent's percepts are always sufficient for determining the state it is in.
- A **policy** is a complete mapping from states to actions.
- Given a policy, it is possible to calculate the expected utility of the possible environment histories generated by that policy.
- It is non-trivial to compute an **optimal policy** that results in the highest expected utility (recall the MEU principle).
- If the agents knows an optimal policy, then it can choose an action in a deterministic fashion in every state.

© 2002 HUT / Laboratory for Theoretical Computer Science

Example. An optimal policy for the square world appears on the left.

	→	→	→	□+1
3	→	→	→	□+1
2	↑	↑	↑	□-1
1	↑	←	←	←
	1	2	3	4

	0.812	0.868	0.918	□+1
3	0.812	0.868	0.918	□+1
2	0.762	0.660	0.660	□-1
1	0.705	0.655	0.611	0.388
	1	2	3	4

The expected utilities for individual states are given on the right.

- The policy is very conservative (tries to avoid punishment).
- If the cost of moves is increased, then the optimal policy becomes different for the state (3, 1): $West$ is replaced by $North$.
- If the cost of moves is decreased to $\frac{1}{100}$, then $West$ is chosen instead of $North$ in state (3, 2).

© 2002 HUT / Laboratory for Theoretical Computer Science

Markov Decision Problems

- The problem of calculating an optimal policy in an accessible, stochastic environment with a known transition model is called a **Markov decision problem** (MDP).
- It is said that the **Markov property** holds if the transition probabilities depend only on the state (not on previous history).
- In the sequel, we will study two basic techniques for solving MDPs, namely **value iteration** and **policy iteration**.
- In an inaccessible environment, the corresponding problem is called a **partially observable** MDP (or POMDP).
- Solving POMDPs is much more difficult than solving MDPs.

© 2002 HUT / Laboratory for Theoretical Computer Science

VALUE ITERATION

- In **value iteration**, the idea is to compute the utility $U(s)$ for each state s and to use these utilities for selecting optimal actions.
- It is difficult to determine $U(s)$ because of uncertain actions.
- Let $H(s, p)$ denote the history tree which results when – starting from a state s – actions are taken according to a policy p .
- Given a transition model M , the expected utility of a state s is

$$\begin{aligned} U(s) &= \text{EU}(H(s, \text{policy}^*) \mid M) \\ &= \sum P(H(s, \text{policy}^*) \mid M) U_h(H(s, \text{policy}^*)) \end{aligned}$$

where policy^* is an optimal policy defined by M and the utility function U_h on state histories.

© 2002 HUT / Laboratory for Theoretical Computer Science

How to Derive an Optimal Policy?

- It is required that the utility function U_h on histories is **separable**:
- $$U_h([s_0, \dots, s_n]) = f(s_0, U_h([s_1, \dots, s_n])) \text{ for some } f.$$
- The simplest form of a separable utility function is **additive**:
- $$U_h([s_0, s_1, \dots, s_n]) = R(s_0) + U_h([s_1, \dots, s_n])$$
- where R is a **reward function** on individual states s .
- Given an additive utility function U_h , an optimal policy policy^* in state i can be defined by the standard MEU principle:

$$\text{policy}^*(i) = \arg \max_a \sum_j M_{ij}^a U(j).$$

- Similarly, the utility of a state can be expressed as follows:

$$U(i) = R(i) + \max_a \sum_j M_{ij}^a U(j).$$

© 2002 HUT / Laboratory for Theoretical Computer Science

Dynamic Programming

- Dynamic programming involves an n -step decision problem where the terminal states reached after n steps have known utilities.
-
- The expected utilities of other states can be computed backwards (layer by layer): $n - 1^{\text{th}}$ layer, $n - 2^{\text{th}}$ layer, etc.
 - In this fashion, the time complexity of computing utilities is of $O(n \cdot A \cdot |S|)$ where $|S|$ is the number of reachable states.
 - Unfortunately, the dynamic programming approach is no longer applicable if environment histories are of unbounded length.

© 2002 HUT / Laboratory for Theoretical Computer Science

Value Iteration Algorithm

- There is an iterative procedure that approximates the utilities of states to any degree of accuracy.
- The next estimate $U_{t+1}(i)$ is based on the old utility estimates of the neighboring states: $U_{t+1}(i) = R(i) + \max_a \sum_j M_{ij}^a U_t(j)$.
- There is no bound on the length of actions sequences.

```

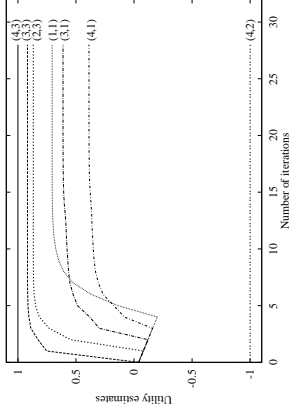
function VALUE-ITERATION( $M, R$ ) returns a utility function
inputs:  $M$ , a transition model
        $R$ , a reward function on states
local variables:  $U$ , utility function, initially identical to  $R$ 
                 $U'$ , utility function, initially identical to  $R$ 
repeat
   $U' \leftarrow U$ 
  for each state  $i$  do
     $U'(i) \leftarrow R(i) + \max_a \sum_j M_{ij}^a U(j)$ 
  end
until CLOSE-ENOUGH( $U, U'$ )
return  $U'$ 

```

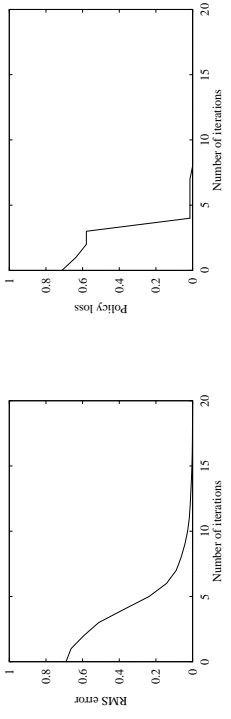
© 2002 HUT / Laboratory for Theoretical Computer Science

Convergence

- As t grows, the utility values will converge to stable values given certain conditions on the environment.
- Given a stabilized utility function, the corresponding *optimal policy* [shown by Bellman and Dreyfus, 1962] is easy to compute.



- Unfortunately, it is difficult to estimate how long the value iteration algorithm should be run to get an optimal policy.
- The progress of value iteration can be measured using **root mean square error (RMS error)** if the correct values are known.
- Alternatively, policies can be evaluated using **policy loss**, i.e., the difference of expected utility with respect to the optimal policy.



An optimal policy is reached long before utilities converge.

POLICY ITERATION

- The optimal policy is often not very sensitive to the utility values.
- The basic idea in **policy iteration** is to choose a policy p , calculate utilities using p as policy, and update p (repeatedly).
- The **value determination** (utilities) is simpler given a policy p :

$$U_{t+1}(i) = R(i) + \sum_j M_{ij}^{p(i)} U_t(j).$$

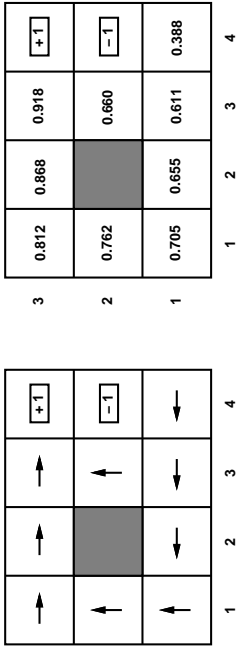
- A modified value iteration algorithm can be used.
- Unfortunately, value iteration may converge very slowly.
- Another approach is to solve utilities directly using equations

$$U(i) = R(i) + \sum_j M_{ij}^{p(i)} U(j)$$

that characterize stabilized utility values ($\forall i: U_{t+1}(i) = U_t(i)$).

Example. The utilities of states (3, 2) and (3, 3) are solved as follows:

$$\begin{cases} u_{(3,2)} = -0.04 + 0.8u_{(3,3)} + 0.1u_{(3,2)} - 0.1 \\ u_{(3,3)} = -0.04 + 0.8 + 0.1u_{(3,3)} + 0.1u_{(3,2)} \\ -0.8u_{(3,3)} = -0.9u_{(3,2)} - 0.14 \\ 8.1u_{(3,3)} = 0.9u_{(3,2)} + 6.84 \end{cases} \Rightarrow \begin{cases} u_{(3,3)} = \frac{6.7}{7.3} \sim 0.918 \text{ and } u_{(3,2)} = \frac{0.8u_{(3,3)} - 0.14}{0.9} \sim 0.660. \end{cases}$$



- Once the utilities of all states are known, it is straightforward to update the current policy using the MEU principle.

function POLICY-ITERATION(M, R) **returns** a policy

inputs: M , a transition model

R , a reward function on states

local variables: U , a utility function, initially identical to R

P , a policy, initially optimal with respect to U

repeat

$U \leftarrow$ VALUE-DETERMINATION(P, U, M, R)

unchanged? \leftarrow true

for each state i **do**

if $\max_a \sum_j M_{ij}^a U[j] > \sum_j M_{ij}^{P(i)} U[j]$ **then**

$P[i] \leftarrow$ arg $\max_a \sum_j M_{ij}^a U[j]$

unchanged? \leftarrow false

end

until *unchanged?*

return P

© 2002 HUT / Laboratory for Theoretical Computer Science

How Immortal Agents Decide What to Do?

- The total reward obtained by a policy can easily be unbounded if the lifetime of an agent is not limited.
- In **discounting**, rewards received in the future are considered less valuable than rewards received in the current time step.
- Given a **discount factor** $0 \leq \gamma < 1$, the sum $U(H) = \sum_{i=1}^{\infty} \gamma^i R_i$ of rewards R_1, R_2, \dots (bounded by R) in a history H converges.
- Discounting conforms to a preference-independence assumption called **stationarity**: if $R_1 = S_1$ holds for two reward sequences R_1, R_2, \dots and S_1, S_2, \dots , then these sequences should be preference ordered in the same way as R_2, R_3, \dots and S_2, S_3, \dots .
- An optimal policy yields a constant **system gain** in the long run.

© 2002 HUT / Laboratory for Theoretical Computer Science

DECISION-THEORETIC AGENT DESIGN

- Recall the schematic design of decision theoretic agents performing **decision cycles** repeatedly:

function DECISION-THEORETIC-AGENT(*percept*) **returns** *action*

calculate updated probabilities for current state based on

available evidence including current percept and previous action

calculate outcome probabilities for actions

given action descriptions and probabilities of current states

select *action* with highest expected utility

given probabilities of outcomes and utility information

return *action*

- The components of the cycle are refined gradually in the sequel.
- We begin with the problem of determining the current state.

© 2002 HUT / Laboratory for Theoretical Computer Science

Determining the Current State of the World

- In general, it is assumed that a set of state variables \mathbf{X}_t (indexed by time t) refers to the current state of the world.
- Given the percept history $\mathbf{E}_1, \dots, \mathbf{E}_t$ and the previous actions A_1, \dots, A_{t-1} , we are interested in the probability distribution $Bel(\mathbf{X}_t) = \mathbf{P}(\mathbf{X}_t | \mathbf{E}_1, \dots, \mathbf{E}_t, A_1, \dots, A_{t-1})$.
- The direct evaluation of $Bel(\mathbf{X}_t)$ is out of the question, as it requires conditioning on many variables.
- Conditional independence statements can be introduced in order to simplify the expression for $Bel(\mathbf{X}_t)$.

© 2002 HUT / Laboratory for Theoretical Computer Science

Simplifying Assumptions

- ▶ Assuming the Markov property, we obtain

$$\mathbf{P}(\mathbf{X}_t | \mathbf{X}_1, \dots, \mathbf{X}_{t-1}, A_1, \dots, A_{t-1}) = \mathbf{P}(\mathbf{X}_t | \mathbf{X}_{t-1}, A_{t-1}).$$
- ▶ The Markov property can be established by introducing state variables that record relevant information from percepts.

Example. If the robot is battery-powered, then the state variable *BatteryLevel_t* is needed to restore the Markov property.
- ▶ Percepts are causally determined by the state of the world:

$$\mathbf{P}(\mathbf{E}_t | \mathbf{X}_1, \dots, \mathbf{X}_t, A_1, \dots, A_{t-1}, \mathbf{E}_1, \dots, \mathbf{E}_{t-1}) = \mathbf{P}(\mathbf{E}_t | \mathbf{X}_t).$$
- ▶ The action taken depends only on the percepts received to date:

$$\mathbf{P}(A_{t-1} | A_1, \dots, A_{t-2}, \mathbf{E}_1, \dots, \mathbf{E}_{t-1}) = \mathbf{P}(A_{t-1} | \mathbf{E}_1, \dots, \mathbf{E}_{t-1}).$$

© 2002 HUT / Laboratory for Theoretical Computer Science

Calculating the State Estimate $Bel(\mathbf{X}_t)$

- ▶ The calculation takes place in two phases:
 1. **Prediction phase:** the prior probability distribution $\widehat{Bel}(\mathbf{X}_t)$ based on the previous state and how actions affect states:

$$\sum_{\mathbf{x}_{t-1}} \mathbf{P}(\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{x}_{t-1}, A_{t-1}) Bel(\mathbf{X}_{t-1} = \mathbf{x}_{t-1}).$$
 2. **Estimation phase:** the effect of the most recent percept \mathbf{E}_t is incorporated to the distribution $\widehat{Bel}(\mathbf{X}_t)$ by Bayesian updating:

$$Bel(\mathbf{X}_t) = \alpha \mathbf{P}(\mathbf{E}_t | \mathbf{X}_t) \widehat{Bel}(\mathbf{X}_t)$$
 where α is a normalization constant.
- ▶ The equations for Bel and \widehat{Bel} form a generalization of **Kalman filtering** – a technique of classical control theory.

© 2002 HUT / Laboratory for Theoretical Computer Science

The Complete Decision-Theoretic Design

- ▶ The remaining steps of the decision cycle are straightforward.
- ▶ The **sensor model** $\mathbf{P}(\mathbf{E}_t | \mathbf{X}_t)$ describes how the environment generates the sensor data.
- ▶ The **action model** $\mathbf{P}(\mathbf{X}_t | \mathbf{X}_{t-1}, A_{t-1})$ gives the effects of actions.

function DECISION-THEORETIC-AGENT(\mathcal{E}_t) returns an action

inputs: \mathcal{E}_t , the percept at time t

static: BN , a belief network with nodes \mathbf{X}

$Bel(\mathbf{X})$, a vector of probabilities, updated over time

$\widehat{Bel}(\mathbf{X}_t) \leftarrow \sum_{\mathbf{x}_{t-1}} \mathbf{P}(\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{x}_{t-1}, A_{t-1}) Bel(\mathbf{X}_{t-1} = \mathbf{x}_{t-1})$

$Bel(\mathbf{X}_t) \leftarrow \alpha \mathbf{P}(\mathbf{E}_t | \mathbf{X}_t) \widehat{Bel}(\mathbf{X}_t)$

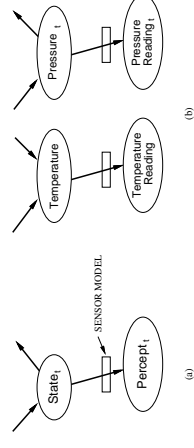
action $\leftarrow \arg \max_{A_t} \sum_{\mathbf{x}} [Bel(\mathbf{X} = \mathbf{x}) \sum_{\mathbf{x}_{t+1}} P(\mathbf{X}_{t+1} = \mathbf{x}_{t+1} | \mathbf{X}_t = \mathbf{x}, A_t) U(\mathbf{x}_{t+1})]$

return action

Sensing in Uncertain Worlds

- ▶ A sensor model is **stationary** if it holds for all t that

$$\mathbf{P}(\mathbf{E}_t | \mathbf{X}_t) = \mathbf{P}(\mathbf{E} | \mathbf{X}).$$
- ▶ A fixed model $\mathbf{P}(\mathbf{E} | \mathbf{X})$ can be used at each time step.
- ▶ All variables affecting the sensors have to be included in \mathbf{X} .
- ▶ A sensor model is easily implemented as a conditional probability table in a belief network (the figure on the left hand side):



© 2002 HUT / Laboratory for Theoretical Computer Science

Calculating the State Estimate $Bel(\mathbf{X}_t)$

- ▶ The calculation takes place in two phases:
 1. **Prediction phase:** the prior probability distribution $\widehat{Bel}(\mathbf{X}_t)$ based on the previous state and how actions affect states:

$$\sum_{\mathbf{x}_{t-1}} \mathbf{P}(\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{x}_{t-1}, A_{t-1}) Bel(\mathbf{X}_{t-1} = \mathbf{x}_{t-1}).$$
 2. **Estimation phase:** the effect of the most recent percept \mathbf{E}_t is incorporated to the distribution $\widehat{Bel}(\mathbf{X}_t)$ by Bayesian updating:

$$Bel(\mathbf{X}_t) = \alpha \mathbf{P}(\mathbf{E}_t | \mathbf{X}_t) \widehat{Bel}(\mathbf{X}_t)$$
 where α is a normalization constant.
- ▶ The equations for Bel and \widehat{Bel} form a generalization of **Kalman filtering** – a technique of classical control theory.

© 2002 HUT / Laboratory for Theoretical Computer Science

© 2002 HUT / Laboratory for Theoretical Computer Science

- The values of sensors are *causally* related to the state of the world.
- A perfect sensor corresponds to a purely deterministic CPT.
- Possible *noise* and *errors* in the sensor are taken into account in the probabilities of incorrect readings.

Example. In the burglar-alarm example, both *JohnCalls* and *MaryCalls* can be viewed as sensors for the *Alarm* state variable.

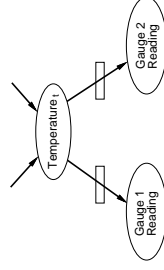
- Typically, each sensor only measures some small aspects of the total state (as illustrated in the figure on the right hand side).
- Decomposing the overall sensor model into several components may reduce the size of the CPTs required.

Example. Measuring *Pressure* and *Temperature* with sensors that measure *Pressure/Temperature* and *Pressure* \times *Temperature* leads to complicated sensor models that depend on both state variables.

© 2002 HUT / Laboratory for Theoretical Computer Science

Sensor Fusion

- There are often several sensors measuring the same state variable.



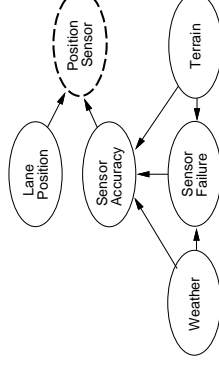
- The sensor values are conditionally independent of each other, given the actual value of the state variable.
- **Sensor fusion** or **data fusion** is about interpreting and putting together perceptual information from multiple sensors.

Example. If the readings from gauges are $13.6^\circ K (\pm 0.5^\circ K)$ and $14.4^\circ K (\pm 0.5^\circ K)$, the temperature is between $13.9^\circ K$ and $14.1^\circ K$.

© 2002 HUT / Laboratory for Theoretical Computer Science

Sensor Failures

- It may be difficult to detect a sensor failure.
- To handle sensor failures in the first place, the possibility of failure has to be taken into account in the sensor model.
- Sensor fusion may discount the readings of a failed sensor.
- One possibility is to add a detailed sensor failure model:



© 2002 HUT / Laboratory for Theoretical Computer Science

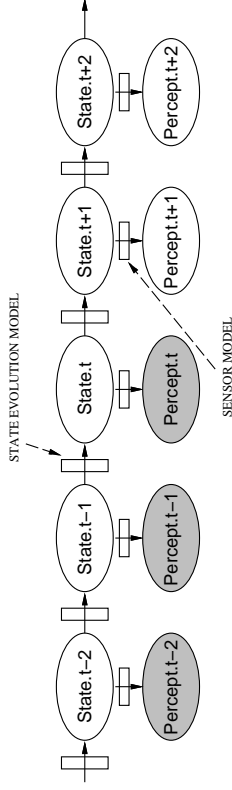
DYNAMIC BELIEF NETWORKS

- A **dynamic belief network** (DBN) represents how the state of the environment evolves over time.
- In analogy to sensor models, a **stationarity** assumption is made: the distribution $P(\mathbf{X}_t | \mathbf{X}_{t-1}, A_{t-1})$ is the same for all t .
- Moreover, the agent is assumed to be passively monitoring and predicting a changing environment (i.e., it performs no actions).
- A **state evolution model** where a sequence of \mathbf{X}_t values is based on a fixed distribution $P(\mathbf{X}_t | \mathbf{X}_{t-1})$ is called a **Markov chain**.
- DBNs will be generalized for the decision of actions later on.

© 2002 HUT / Laboratory for Theoretical Computer Science

Structure of Dynamic Belief Networks

- For each time step t , there is one node for each state variable X_t and sensor variable Y_t – including appropriate interconnections.



- The task is to calculate the probability distribution for $State_t$, given the evidence for $\dots, Percept_{t-1}, Percept_t$.
- Probabilistic projection** means estimating how the state of the environment (i.e., $State_{t+n}$ with $n > 0$) evolves in the future.

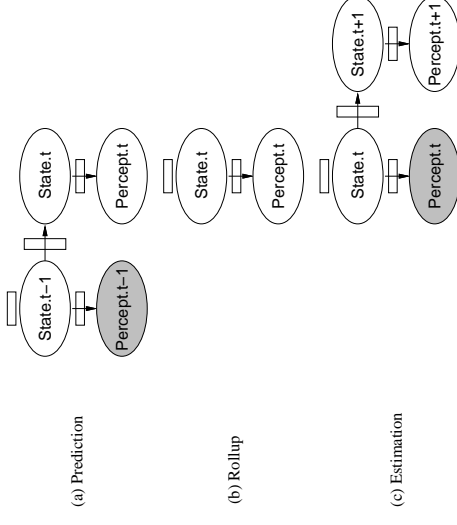
© 2002 HUT / Laboratory for Theoretical Computer Science

Prediction and Estimation with Belief Networks

- The prediction and estimation phases of the refined decision cycle can be implemented as operations on belief networks.
- It is sufficient to consider two time steps t and $t - 1$ (also called the **slices** of the network):
 - Prediction:** using $Bel(\mathbf{X}_{t-1})$ calculate the prior distribution $\widehat{Bel}(\mathbf{X}_t) = \sum_{\mathbf{x}_{t-1}} \mathbf{P}(\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{x}_{t-1}) Bel(\mathbf{X}_{t-1} = \mathbf{x}_{t-1})$.
 - Rollup:** remove the slice for $t - 1$ from the network and add the prior probability tables (based on $\widehat{Bel}(\mathbf{X}_t)$) for \mathbf{X}_t .
 - Estimation:** add the new percept \mathbf{E}_t , calculate $Bel(\mathbf{X}_t)$ by updating the network, and add the slice for $t + 1$.
- After these three steps, the network is ready for the next cycle.

© 2002 HUT / Laboratory for Theoretical Computer Science

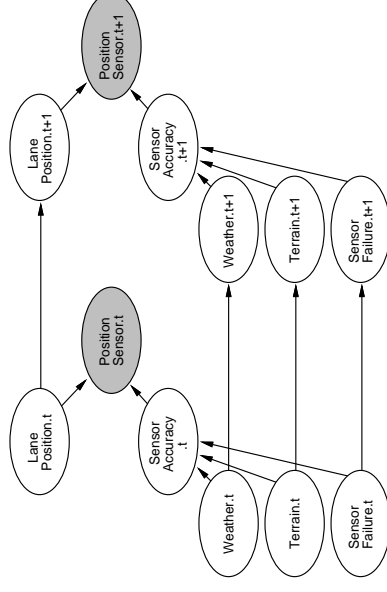
- Let us illustrate the three steps for prediction and estimation:



- Moreover, probabilistic projection is possible by adding the respective slices (*but without percept nodes*) for future time steps.

© 2002 HUT / Laboratory for Theoretical Computer Science

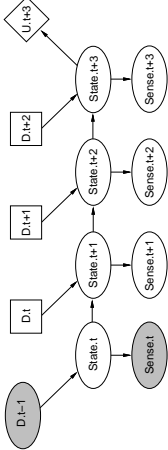
- Example.** Let us extend the sensor failure model presented earlier by adding state evolution models for the state variables *Weather*, *Terrain*, *SensorFailure* and *LanePosition*:



- The model for the variable *SensorFailure* determines that the sensor usually stays broken once it gets broken.

© 2002 HUT / Laboratory for Theoretical Computer Science

DYNAMIC DECISION NETWORKS

- Any dynamic belief network can be converted into a **dynamic decision network** by adding utility nodes and decision nodes:
- 
- The goal is to calculate the value of D_t by the MEU principle.
 - The utility of a decision sequence \vec{d} is a weighted sum of utilities associated with each possible percept sequence given \vec{d} .
 - The probabilities of percept sequences given \vec{d} are used as weights.
 - Dynamic decision networks solve POMDPs only approximately.

© 2002 HUT / Laboratory for Theoretical Computer Science

Discussion

- DDNs provide solutions to many problems arising in AI systems:
- Uncertainty is handled correctly, and sometimes efficiently.
 - Continuous streams of sensor input can be dealt with.
 - Unexpected events are supported, since fixed plans are not used.
 - Noisy and failing sensors can be modeled.
 - The relevance of information can be estimated before acquisition.
 - Relatively large state spaces can be handled if states can be represented by state variables with sparse connections.
 - There are techniques for approximative reasoning.

© 2002 HUT / Laboratory for Theoretical Computer Science

SUMMARY

- A **optimal policy** associates an optimal decision with every state that the agent might reach.
- **Value iteration** and **policy iteration** are two methods for calculating optimal policies.
- Unbounded action sequences can be dealt with **discounting**.
- **Dynamic belief networks** can handle sensing and updating over time, and provide a direct implementation of the update cycle.
- **Dynamic decision networks** can solve sequential decision problems arising for agents in complex, uncertain domains.

© 2002 HUT / Laboratory for Theoretical Computer Science

QUESTIONS

1. Recall the belief network that you designed for representing the ball tracking mechanism of a soccer playing agent.
 - Is it possible to identify a state evolution model and a sensor model from your network?
 - If not, reconstruct the network by keeping these in mind.
2. Continue the analysis of soccer playing agents.
 - Can you identify other problems in this domain that can be considered as real sequential decision problems?
 - Try to formalize such a problem as a dynamic decision network.

© 2002 HUT / Laboratory for Theoretical Computer Science