# 01DRO1
# Decision Making under Uncertainty

2017/2018

# Basics of Sequential DM

Lecture 2

27.02.2017

# Readings

- Martin Puterman, *Markov decision processes*, John Wiley & Sons, 1994
- D.P. Bertsekas, *Dynamic Programming*, Prentice Hall, 1987
- Richard E. Bellman, *Dynamic Programming,* Dover Publications, 2003

Announcement:

**28.2** 01DROS is cancelled, replacement will be announced later.

# Where are we?

# **Last time..** Agent and Environment

## Agent

(*knowledge*, *observations*) $\to$ *actions*

*observations*

sensors

actuators

*built-in knowledge, DM objectives*

*actions*

## Environment

Agent: human; artificial device; both
Environment:  part of the world

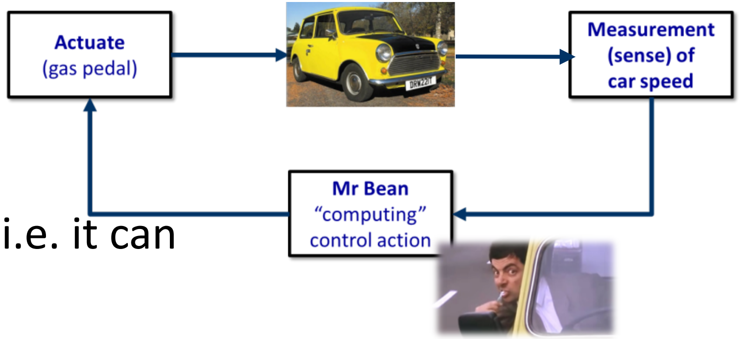
Physical vs. digital  or virtual worlds make no difference

# **Last time..** Environment and Modelling

- Fully observable vs. partially observable
- Deterministic vs. stochastic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous - number of actions/observations is fixed
- Single agent vs. multi-agent
- *Complete vs incomplete model – (learning)*

Hardest case: Partially observable, stochastic, sequential, dynamic, continuous, multi-agent environment $\equiv$ Real world!

# **Last time..** Closed-loop




Feedback can “make good system from bad components”, i.e. it can

- make system insensitive to disturbances
- stabilise unstable system (bicycle)
- produce desired behaviour (e.g. non-linear components produce linear behaviour)

| **Feedback** | **Feedforward** |
|---|---|
| Actions are computed based on the difference “actual output – desired output” | Make a plan of actions and execute it |
| Closed-loop control | Open-loop control |
| System-driven | “Pure” planning |
| Acts whenever there is a deviation of actual from desired => sensor noise is fed into system | Acts according to the planned sequence |
| Potential for instability | No risk of instability |

# **Last time..** Intelligent Rational Agent

- Agent autonomously observes, decides and acts (closed-loop interaction with environment)
- Rational agent acts to achieve best outcome when
  - changing the state of the environment (targeted influence)
  - discovering the state of the environment and make better decisions then (learn, describe)

- Agent should account for other agents in the environment

# **Last time..** DM problem formalisation

needs description (from the agent’s point of view) of:

- environment
- possible actions (actuators)
- possible observations (sensors)
- DM goal (performance measure -utility, loss, reward)

Note that **rationality** considers exploration, learning, adaptation

# Today…

# Uncertainty in DM

Rational DM depends on the relative importance of different DM objectives and the *likelihood* that they will be achieved

Any rule how to select a decision cannot rely on *absolute* knowledge and should *handle uncertain knowledge* and *quantify* it.

*Example*: consider rules for taking umbrella. Sources of uncertainty:

- a rule is not known completely
- the current state (rain, no rain) is unknown
- influence of unknown factors (rule, state) on our reasoning  => 
- complexity (difficult to describe all consequents by rules and/or hard to use them)

To capture uncertainty we express our *degrees of belief* via probability.

# Uncertainty and rational DM

- Agent has a DM goal and executes any action ensuring DM goal
- Different degrees of belief: let action $a$ have a 70% chance of success. How about $a'$ with probability of 90% ? How about $a''$ with a higher cost of executing but same probability of success? (see examples Lecture 1)
- Agent must have *preferences* over possible outcomes of actions.
- *Utility theory* can be used to reason about those preferences

Idea: for an agent every state has a *degree of usefulness;* agent prefers states with a higher utility.

Under some conditions on preferences we can always design the utility function that respects our preferences

# Decision theory = probability theory + utility theory

- Foundation of decision theory:

An agent is *rational* iff it chooses the action yielding the highest expected utility, averaged over all possible outcomes.

- The Maximum Expected Utility (MEU) principle [von Neumann & Morgenstern, 1944] says that a rational agent should choose an action a that maximizes its expected utility $EU(s,a)$ in the current state s:

  $EU(s,a) = \Sigma_i\, P(result_i(a)\,|\,a, s)\; U(result_i(a), a, K),$

  $a^{best} = \mathrm{arg}max_a\, EU(s,a),$ where

  $result_i(a)$ is $i$-th possible outcome of action $a$

  $K$ is other knowledge available for choice $a$

# Wrap: Design for a decision-theoretic agent

General design for a decision-theoretic (utility-based) agent:

- formulate and formalise DM problem
- define $A$, $S$, beliefs and utility
- update *belief state* based on previous *action* and *envir.state*
- find probabilities of next envir.state for considered actions and belief states
- select action with highest expected utility given probabilities of next state and *utility*

# What I have not mentioned

- Knowing the current environment state $s$ requires perception, learning and inference.
- Computing $P(result_i(a)|a, s)$ requires a complete environment model.
- Computing $U(result_i(a), a)$ may require knowledge what utilities could be gained from the state $s$

Way out:

To avoid computationally intractable solution, we need to consider agent's "*resource-bounded* rationality" instead of "perfect rationality".

# Resolving Uncertainty

- Search – uncertainty about which action to take

  Connect together a sequence of actions

- Uncertainty Reasoning – uncertainty about the ‘correct’ interpretation of a piece of information

  Connect together a set of interrelated uncertain information

- Learning – uncertainty about the implication of an observed event

  Connect together observed events that are similar

# Brief summary of basic notions

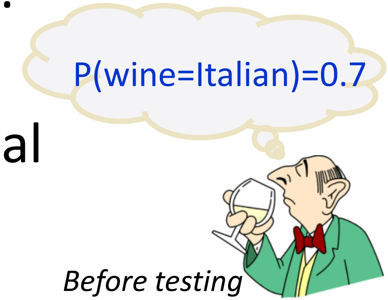
- X - set of possible outcomes of random variable x; or domain of x, dom(x), e.g dom()={1,2,3,4,5,6}.
- P(b) – expresses degree of belief that x=b i.e *your* belief in a, P( =5)=1/6
- $P(\emptyset)=0$,
- If $A \subset B$, then $P(A) \leq P(B)$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ , $P(A \cup B) \leq P(A) + P(B)$
- Conditional probability of A given B: $P(A\,|\,B) = P(A \cap B)/\,P(B)$, assuming $P(B)>0$
- If x is state of  , then P(x) specifies relative a likelihood of any side *you* consider possible. If it is 0, you consider it to be impossible.

# Brief summary of basic properties

- $P(x,y) = P(x|y)\,P(y)$
- $P(x) = \sum_{y \in dom(Y)} P(x|y)P(y)$
- $P(x,y,z)=P(x|y,z)P(y|z)P(z)$ – chain rule
- $P(x|y)=P(y|x)P(x)/P(y) \propto P(y|x)P(x)$ – Bayes rule

# A brief review on probabilistic inference

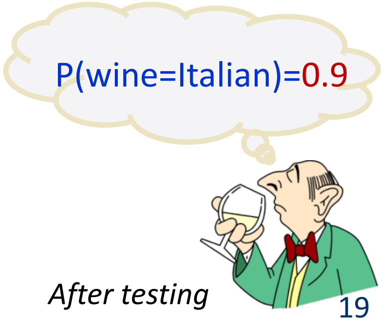
- Agent's **beliefs** about environment is critical for DM => beliefs must quantify degree of agent's uncertainty
- Quantification used: evidential (Bayesian) probability, i.e. the degree to which your belief is supported by the available evidence.
- Data (measurement, etc.) *influence* degrees of belief.
- Before evidence is obtained we speak of prior/unconditional probability, after evidence of posterior probability.




*Before testing*

Do not confuse with:

- degrees of truth are the subject of other methods (fuzzy logic) we do not deal with here
- environment changes that might make any statement true or false.




*After testing*

# Probabilistic inference: conditioning

Given:

- prior $P(x)$ over $x \in X$, representing your degrees of belief
- new measurement $y=y_0$ for a variable $y \in Y$

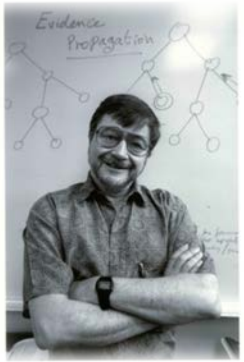
To revise your degrees of belief:

find posterior $P(x|y=y_0)$ by conditioning $P(x)$ on the measured $y_0$

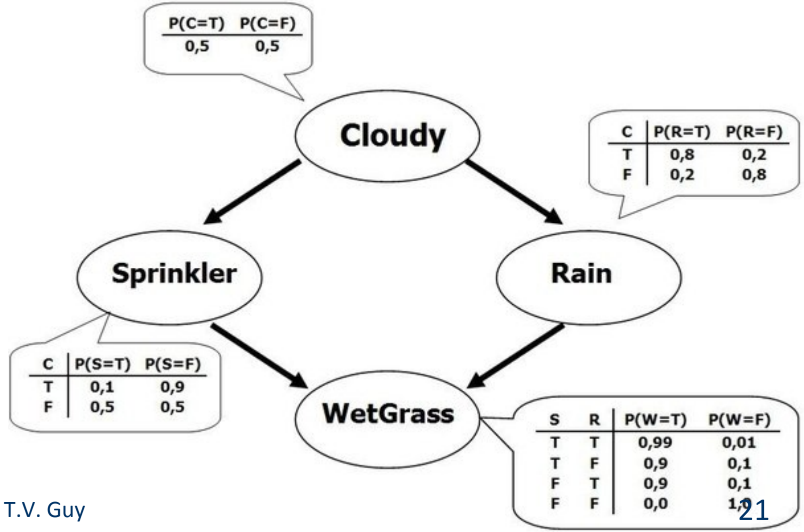
# **Bayesian networks** (belief network)

BN is a graphical representation of the direct dependencies over a set of random variables $\{x_1, x_2, \ldots, x_n\}$ .

- Nodes represent random variables $x_i$ connected by links. An arrow from node $x_i$ to node $x_j$ implies $x_i$ be a parent of $x_j$.
- Each node $x_i$ has a conditional probability table $P(x_i \mid Parents(x_i))$ quantifying the *strength* of influence its parents.




Judea Pearl



# Summary of BN

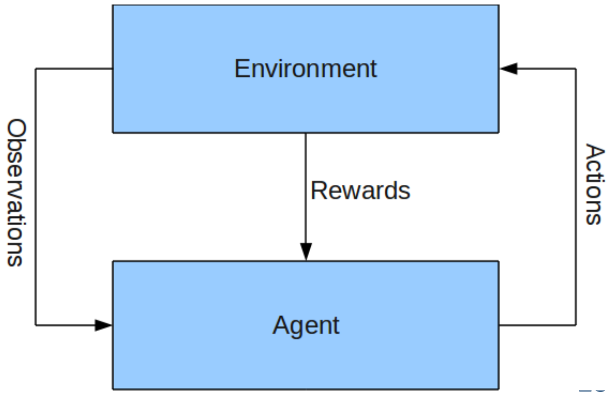
- Info on conditional independence is a robust way to structure information about an uncertain domain.
- **Bayesian networks** are a natural way to represent conditional independence

  Links between nodes represent *qualitative* aspects of the domain, the conditional probability tables represent the *quantitative* aspects.
- BN is a complete representation for the joint probability distribution for the domain, but is often exponentially smaller.
- *Inference in BN* means computing the probability distribution of a set of query variables, given a set of evidence variables. Complexity of inference given by BN structure => constructing a tractable network with $n>100$ need effort.

# Sequential Decision Problems

- Real life has few episodic DM tasks
- Our world needs sequences of decisions that
  - give *opportunity* to take further actions
  - *change* the state of the world (targeted influence)
  - *provide information* that can inform future decisions
  - provide *immediate benefit*

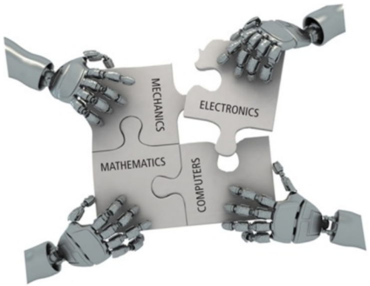
or a combination of all of them.



# Sequential Decision Making

Examples of applications

- Robotics (e.g., control)
- Investments (e.g., portfolio management)
- Computational linguistics (e.g., dialogue management in computer system intended to converse with a human)
- Operations research (e.g., resource allocation, call admission control in VoIP networks)
- …

# Sequential DM

**Static Inference**
*Bayesian Networks*

**Static Decision Making**
*Decision Networks*

**Sequential Inference**
*Hidden Markov Models*
*Dynamic Bayesian Networks*

**Sequential Decision Making**
*Markov Decision Processes*
*Dynamic Decision Networks*

# Very briefly about preference ordering

You are using mobile apps to call a taxi service.

Your preferences:  or  ; 

or 

; 

or 

*new* $\succ$ *old*; *5min* $\succ$ *20min*; *100Kc* $\succ$ *500Kc*; maybe other preferences..

If not satisfied, alternatives can be used => additional cost (more money, less comfort, longer waiting) !

Preference ordering must fulfil: $(x \succcurlyeq y \wedge y \succcurlyeq z) \Rightarrow x \succcurlyeq z$; $(x \succcurlyeq y \wedge y \succcurlyeq x) \Rightarrow x = y$;

$(x \succ y \wedge y \succ x) \Rightarrow \emptyset$

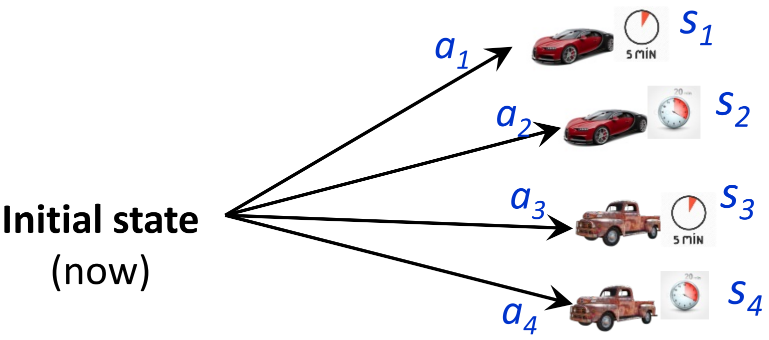
# DM: ‘one-time-never-return’

One-shot decision (irreversible action in partially known environment):

- finite set of actions $A$, and a set of possible states $S$
- given preference ordering $\succcurlyeq$ over S

Deterministic actions: let $f: A \longrightarrow S$ then $\{f(a) \in A\}$ is a set of possible states.

- How to choose $a \in A$ leading to the most preferred state?
- Problem when $a$ points to some sequence=> how to decide which $a$ ?




Clearly $s_1 \succcurlyeq s_2 \succcurlyeq s_3 \succcurlyeq s_4$ implies $a_1$

But if *there is*:

- uncertainty in the next state (environ.)
- stochastic actions (agent)
- uncertainty about initial state

Then no deterministic solution exists!

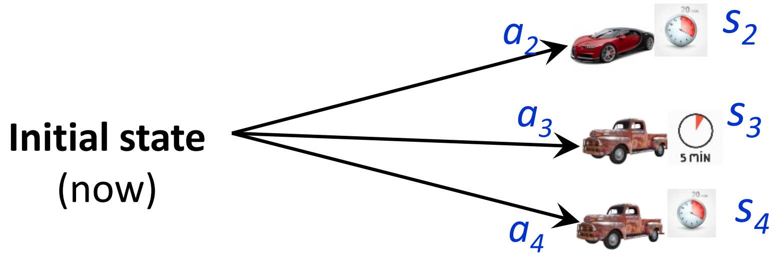
# DM: 'one-time-never-return'

One-shot decision (irreversible action in partially known environment):

- finite set of actions $A$, and a set of possible states $S$
- given preference ordering $\succcurlyeq$ over S

Deterministic actions: let $f: A \longrightarrow S$ then $\{f(a) \in A\}$ is a set of possible states.

- How to choose $a \in A$ leading to the most preferred state?
- Problem when $a$ points to some sequence=> how to decide which $a$ ?




Clearly $s_1 \succcurlyeq s_2 \succcurlyeq s_3 \succcurlyeq s_4$ implies $a_1$

But if *there is*:

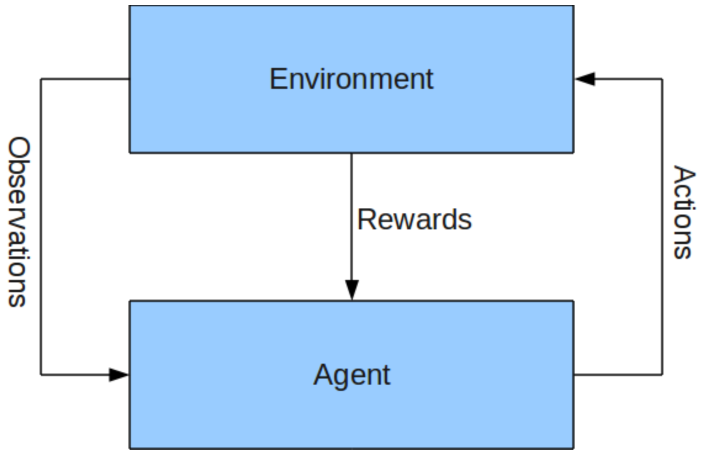
- uncertainty in the next state (environ.)
- stochastic actions (agent)
- uncertainty about initial state

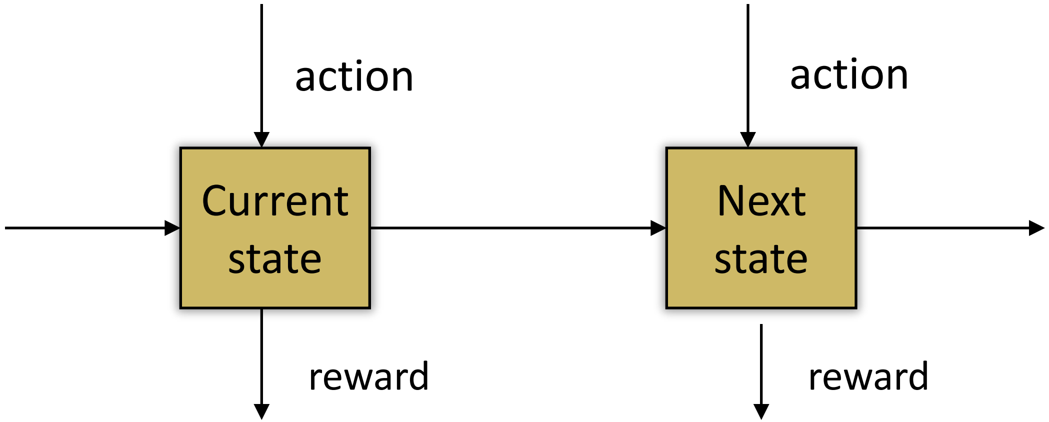
Then no deterministic solution exists!

- more about one-shot decision theory you can find here:

  P.Guo *One short decision theory* IEEE Trans. on System Man and Cybernetics 41(5) 917-926, 2011.
- more about taxi service apps will be at 01DROS on 7.3.

# Sequential decision model: environment evolution



# Example: stochastic environment

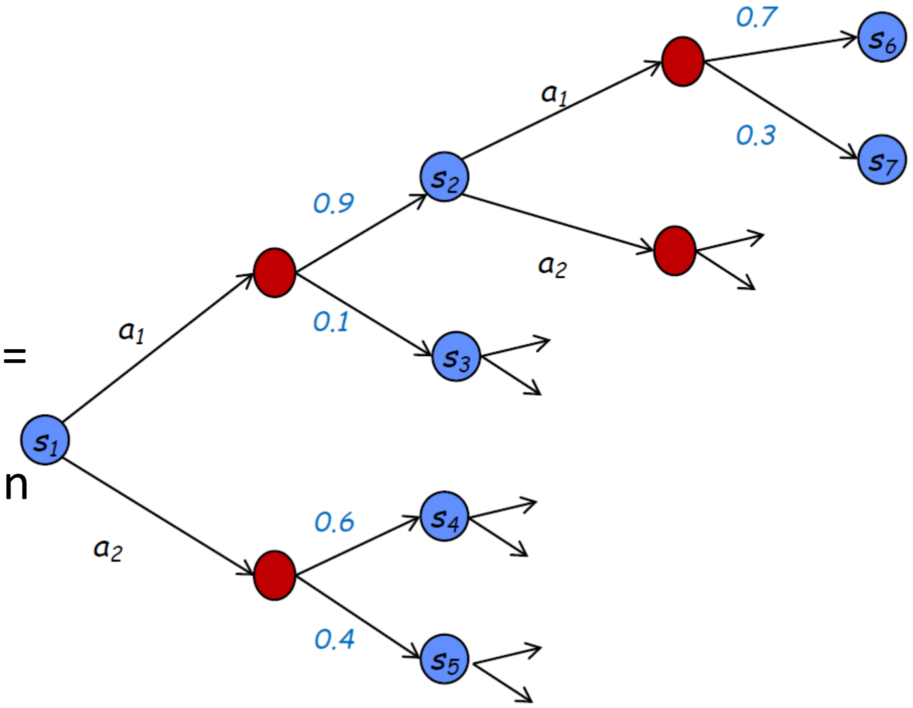
To find best action sequence:

- Assign utility to each trajectory

  e.g., $u(s_1 \rightarrow s_2 \rightarrow s_6)$
- For each action sequence compute probability of any trajectory:

  e.g., $Pr(s_1 \rightarrow s_2 \rightarrow s_6 \mid [a_1,a_1]) = 0.9*0.7 = 0.63$
- Compute expected utility of each action sequence:

  Exp.utility of $[a_1,a_1]$, $[a_1,a_2]$, $[a_2, a_1]$, $[a_2,a_2]$
- Choose the best sequence of actions



# Problems:

- In stochastic environment use of a sequence of actions may result in a state other than the most preferred one (DM goal)
- *Complexity:* $m$ actions, $t$ stages give $m^t$ action sequences to evaluate; and if there are $n$ outcomes per action, the number of possible trajectories is $m^t\, n^t$
- *Practical:* easier to consider utility of particular states (and cost of actions) not utility of entire trajectories (taken as a sum of states utili)
- *Conceptual:* sequences of actions do not reflect 'natural' behaviour:

*After action a1, I go to s2 or s3. It may be better to do a1 again if I end up to s2, but the best to do a2 if I end up at s3.*

# Policy

The mentioned problems be partially solved with use of *policy.* For observable outcomes *policy* = (*do a1; if s2, do a1; if s3, do a2;…*)

- *Policies* specify what the agent should do for *any* reachable state.
- By using *policy* we can make *more* state trajectories possible.

  It (weakly) *increases* expected utility of the best behavior, since policies include sequences as a special case.
- *Complexity problem:* there exist much more policies than sequences => hard computation problem.

  Partial solution: use dynamic programming (next lecture)

# Decision Trees

A decision tree:

- is an simple way to structure sequencing of decisions based on observed events
- explicitly represents all the possible decision sequences from a given state.

Each path corresponds to decisions made by the agent, actions taken, possible observations, state changes, and a final outcome node.

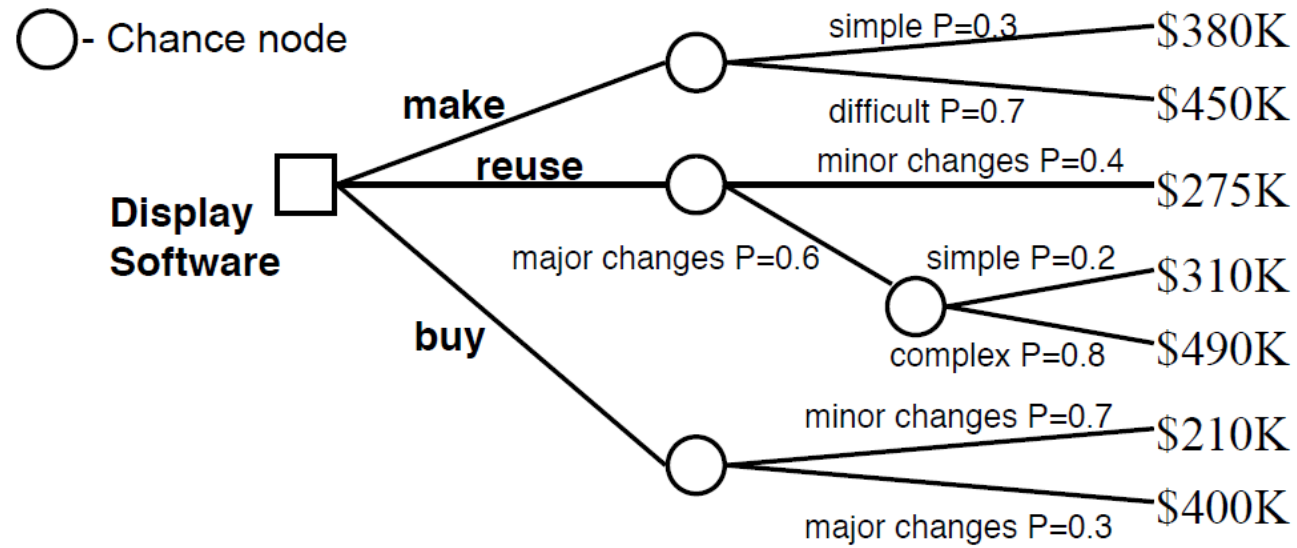
# Nodes in a Decision Tree

- *decision nodes* representing decisions available to agent point to
  i) next decision nodes,
  ii) chance nodes if stochastic.
- *chance nodes* indicate possible outcomes and their probabilities; must be observable
- *terminal nodes:* final outcome of trajectory (labeled with utilities)
- sequencing of decisions based on observed outcomes

Solution: backward induction (simple form of dynamic programming)
helps to compute optimal course of action, or *policy*

! choices at each stage can depend on observed outcomes at any previous stages

# Example 1: Software Development




- **EU(make) = 0.3 * $380K + 0.7 * $450K =** $429K
- **EU(reuse) = 0.4 * $275K + 0.6 * [0.2 * $310K + 0.8 * $490K] =** $382.4K
- **EU(buy) = 0.7 * $210K + 0.3 * $400K =** $267K ; best choice

From V.R. Lesser lecture

# Decision Tree

- In DT information-gathering actions are important and easily modeled => DT is good tool for understanding *value of information* (for instance, pay for tests, trials in order to determine more precise likelihood of the outcomes of certain decisions)
- in *evaluating* trees a direct use of Bayes rule is required

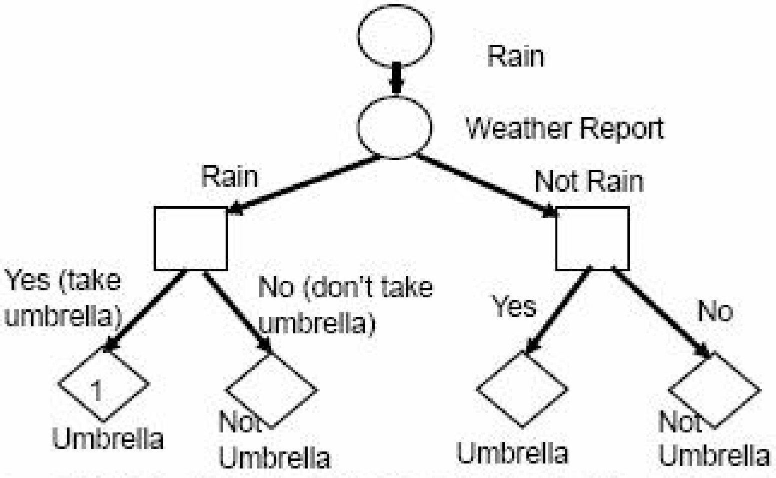
*Further reading: C.Kirkwood: Decision Tree Primer, available on web*

# Nodes in a Decision Network

- Decision networks or influence diagrams are an extension of belief networks.
- *Chance nodes* have conditional probability tables that depend on the states of the parent nodes (chance or decision).
- *Decision nodes* represent decision-making points at which decisions are available to the agent
- Utility nodes provide the overall utility given the states of the parent nodes (nodes that affect utility directly).
- Often nodes describing outcome states are omitted and expected utility associated with actions is expressed (rather than states) – *action-utility tables*

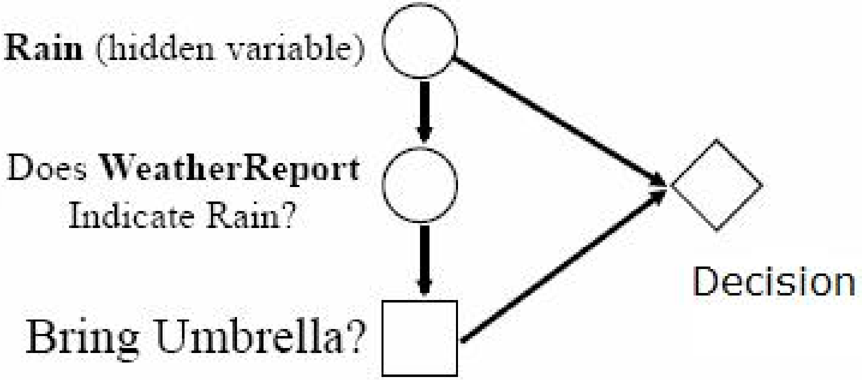
# Decision Tree vs Decision Network

'Taking an Umbrella' as a Decision Tree




Case 1:U(Umbrella|W=Rain)*P(W=Rain|WR=Rain) +U(Umbrella|W=not Rain)*P(W= not Rain|WR=Rain)

'Taking an Umbrella' as an Influence Diagram




Parameters: P(Rain), P(WeatherReport|Rain), P(WeatherReport|¬Rain), Utility(Rain,Umbrella)

Images source www.lumina.com

# Knowledge in Decision Network

- Causal knowledge about *how* events influence each other
- Knowledge about *what* decision sequences are feasible in a given set of circumstances (defines possible temporal ordering of decisions) => admissible policies
- Knowledge about *how desirable* the consequences are => utility

# Evaluating Decision Networks

1. Set the evidence variables (observation) for the current state.

2. For each possible value of the decision node(s):

(a) Set the decision node to that value.

(b) Calculate the posterior probabilities for the parent nodes of the utility node.

(c) Calculate the expected utility for the action/decision.

3. Execute the action/decision with the highest expected utility.

# Decision Trees vs. Decision Networks

Decision trees are *not good* for representing domain knowledge

- Requires large amount of memory
- Multiple decisions nodes → expands tree
- Duplication of knowledge along different paths is unavoidable

Decision network:

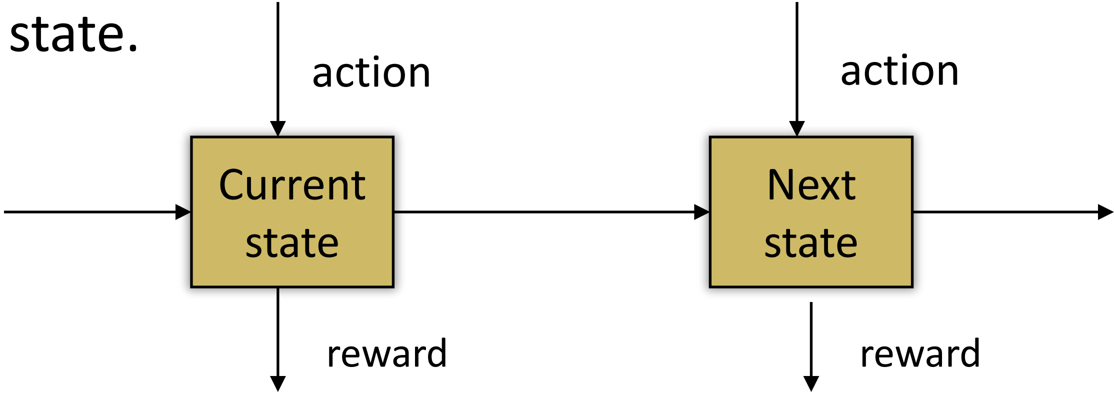
- generates decision tree from more economical forms of knowledge
- has an assumption that the agent remembers all past observations and decisions

A more useful but less structured formalism- Markov Decision Process

# What is a Markov decision process?

A *mathematical* representation of a sequential decision making problem in which:

- A environment evolves through time.
- An agent influences it by taking actions at pre-specified points of time.
- Actions imply immediate costs (rewards) **and** affect the subsequent environment state.



# MDP-Markov Decision Process

А. А. Марков (1886).

- MDP is a math framework for modelling DM where environment states are random and influenced by the agent.
- Markov property is a memoryless property of a stochastic process

$$Pr(s_{t+1} \mid \mathrm{s}_t = s_t, a_t = a_t, \mathrm{s}_{t-1} = s_{t-1}, a_{t-1} = s_{t-1}, \ldots, \mathrm{s}_0 = s_0, a_0 = s_0)$$

$$= Pr(s_{t+1} \mid \mathrm{s}_t = s_t, a_t = a_t)$$

# Historic notes

- A model of sequential decision-making developed in operations research in the 1950's

  Abraham Wald - Sequential Analysis (1940's )

  Richard Bellman - Dynamic Programing (1950's)
- 1950- 1980's development of theory and algorithms, applications
- since 1990's adopted by the AI community as a framework for:
  - Decision-theoretic planning (e.g., [Dean et al., 1995])
  - Reinforcement learning (e.g., [Barto et al., 1995])
- Allows reasoning about actions with uncertain outcomes

  more than 60 years of successful applications

# Markov Decision Processes are also known as:

- Dynamic Programs
- Stochastic Dynamic Programs
- Sequential Decision Processes
- Stochastic Control Problems

# MDP-Markov Decision Process

- Markov decision processes are an extension of Markov chains; the difference is the addition of actions (*allowing choice*) and rewards ( a kind of *feedback* giving *motivation*).
- If only one action exists for each state and all rewards are the same a Markov decision process reduces to a Markov chain.

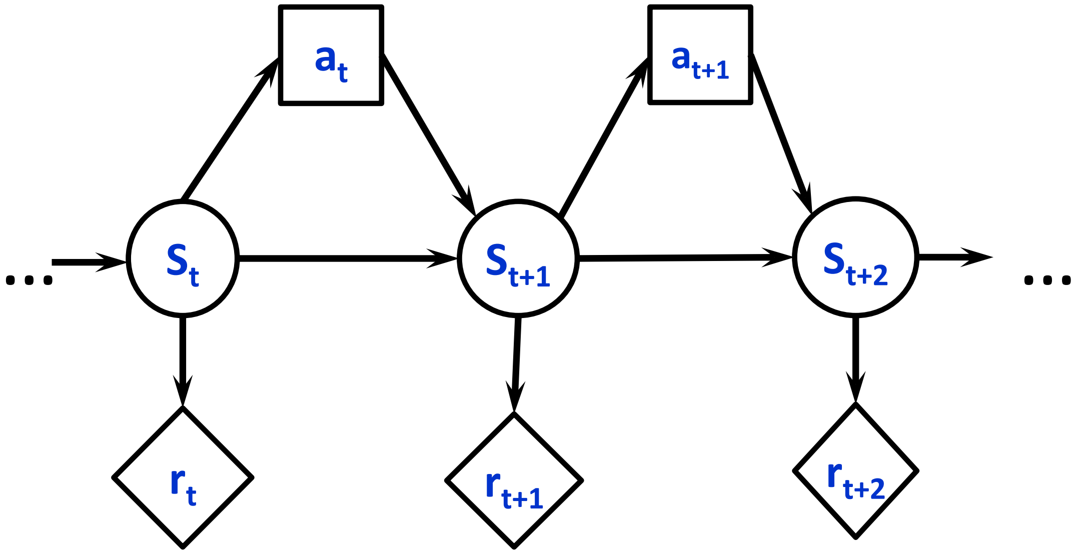
# MDP: formalisation

MDP is defined by $(T, S, A, R, Pr)$:

- $S$ - finite set of all possible *states*, $|S| = n$
- $A_s$ - finite set of allowable *actions* (decisions) in state $s$.
  $A = \cup_{s \in S} A_s$ - the set of all possible actions, $|A| = m$
- $Pr(s_{t+1}|s_t, a_t)$ - *state transition function*
  - represented by set of n x n probability matrices for each $a_t$
  - each $Pr(s_{t+1}|s_t, a_t)$ is a distribution over $S$
- bounded, real-valued *reward function* $R(s)$
  - represented by an $n$-vector
  - can be generalised to include action costs $R(s,a)$
  - can be negative to reflect the cost incurred
  - generally can be stochastic (replaceable by expectation)

# Decision Epochs

- Times at which decisions are made
  (analogous to period start times in Markov Process)
- The set *T* of decisions epochs can be either a discrete set or a continuum.
- The set *T* can be finite (*finite horizon problem*) or infinite (*infinite horizon*).

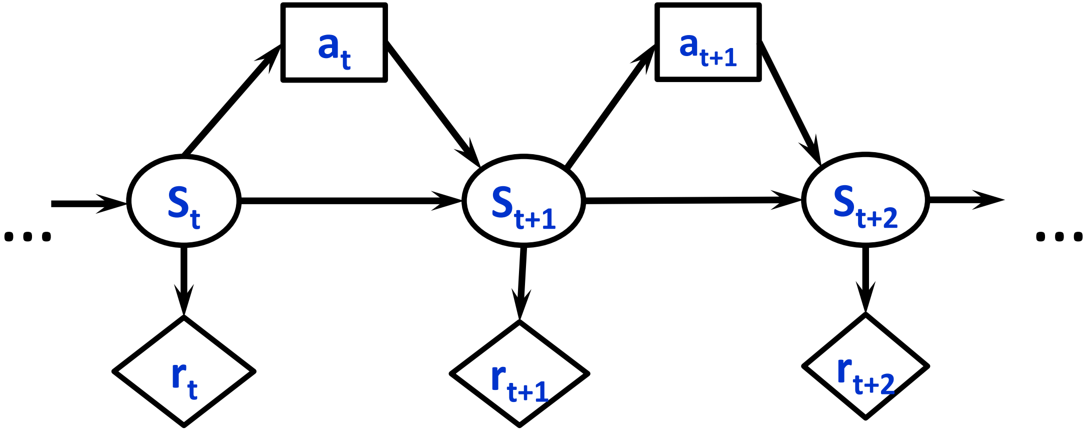


# States and Transition Functions

*States* $s$ are analogous to states in Markov Processes and include all info from the past relevant to the future.

*Transition function* is a distribution that governs how the state changes as actions are taken over time.

As a result of choosing action $a \in A_s$ in state $s$ at decision epoch $t$, the system state at $t+1$ is determined by the probability distribution $p_t(\,.\,|s, a)$.



# Remarks

- If the transition matrix $p_t( . | s, a)$ is independent of the action $a$, MDP is a Markov process => recall course *01MAPR*
- If transition matrix $p_t( . | s, a)$ is independent of $s$ we have an i.i.d. process => recall course 01PRA12