Decision Theory: Optimal Policies for Sequential Decisions

CPSC 322 – Decision Theory 3

Textbook §9.3

April 4, 2011

Lecture Overview

- Recap: Sequential Decision Problems and Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

Recap: Single vs. Sequential Actions

- Single Action (aka One-Off Decisions)
  - One or more primitive decisions that can be treated as a single macro decision to be made before acting

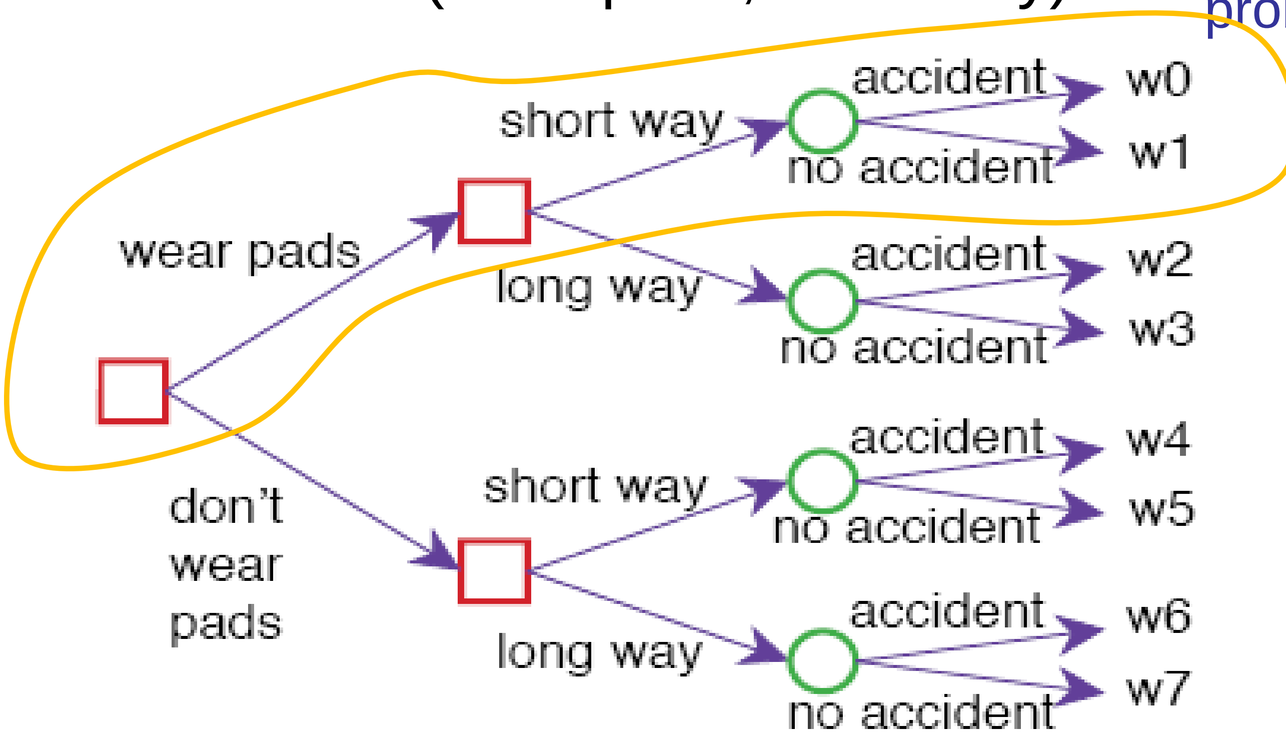
- Sequence of Actions (Sequential Decisions)
  - Repeat:
    - observe
    - act
  - Agent has to take actions not knowing what the future brings

Recap: Optimal single-stage decisions

Definition (optimal single-stage decision)
An optimal single-stage decision is the decision $D=d_{max}$ whose expected value is maximal:

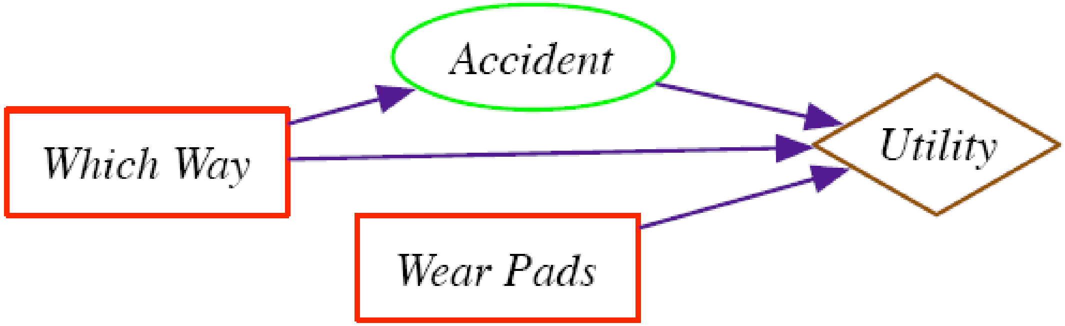
$$d_{max} \in \underset{d_i \in dom(D)}{\operatorname{argmax}} \; E[U|D=d_i]$$

Best decision: (wear pads, short way)



Conditional probability	Utility	E[U|D]
0.2	35	83
0.8	95	
0.01	30	74.55
0.99	75	
0.2	3	80.6
0.8	100	
0.01	0	79.2
0.99	80	

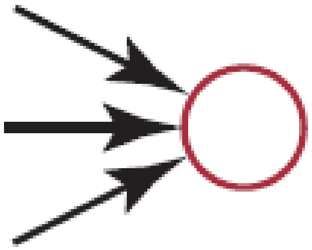
Recap: Single-Stage decision networks



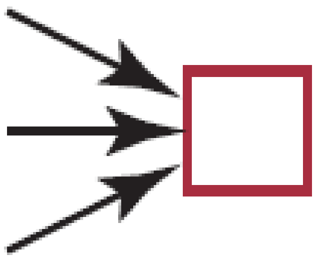
- Compact and explicit representation
  - Compact: each random/decision variable only occurs once
  - Explicit: dependences are made explicit
    - e.g., which variables affect the probability of an accident?

- Extension of Bayesian networks with
  - Decision variables
  - A single utility node

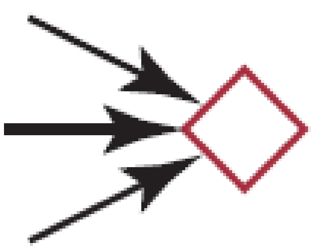
Recap: Types of nodes in decision networks



- A random variable is drawn as an ellipse.
  - Parents pa(X): encode dependence
    Conditional probability p(X | pa(X))
    Random variable X is conditionally independent of its non-descendants given its parents
  - Domain: the values it can take at random

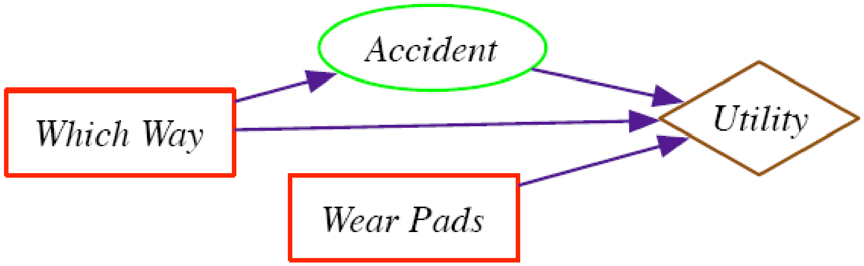


- A decision variable is drawn as an rectangle.
  - Parents pa(D)
    information available when decision D is made
    - Single-stage: pa(D) only includes decision variables
  - Domain: the values the agents can choose (actions)



- A utility node is drawn as a diamond.
  - Parents pa(U): variables utility directly depends on
    - utility U(pa(U)) for each instantiation of its parents
  - Domain: does not have a domain!

Recap: VE for computing the optimal decision



- Denote
  - the random variables as $X_1, \ldots, X_n$
  - the decision variables as D

$$E[U|D=d] = \sum_{w} P(w|D=d)U(w)$$

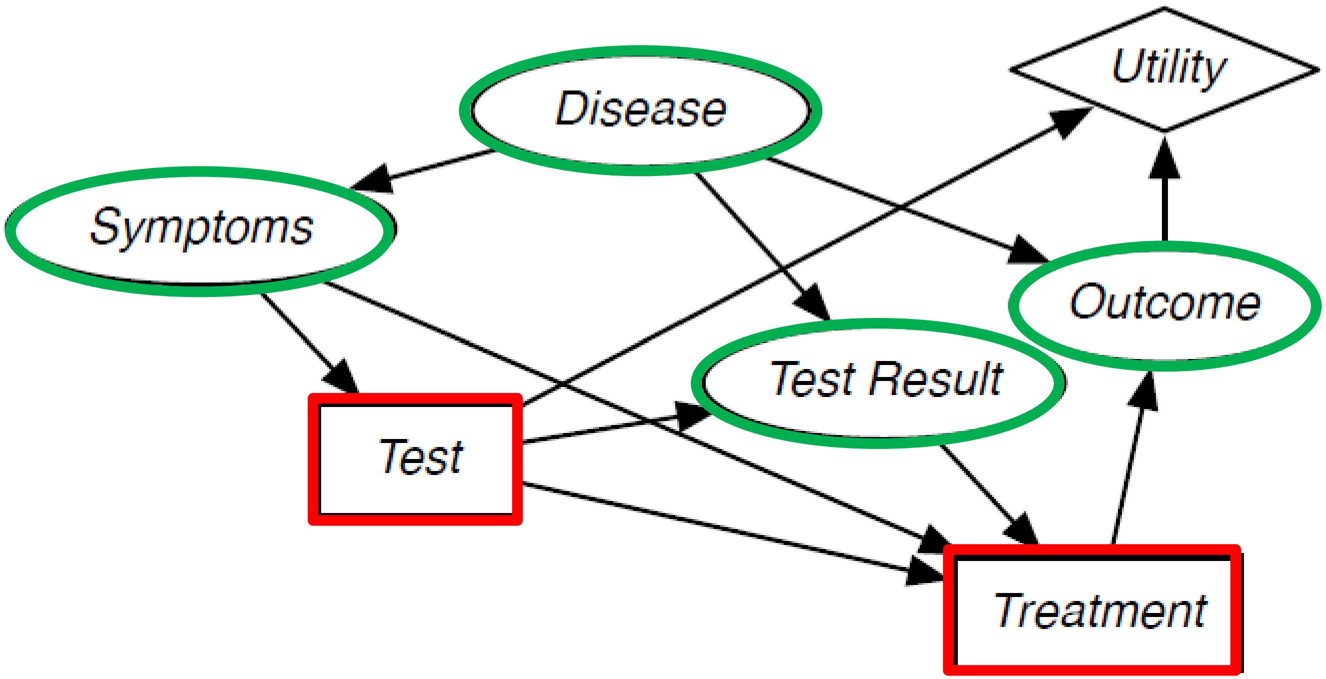
$$= \sum_{X_1,\ldots,Xn} P(X_1,\ldots,Xn|D=d)U(pa(U))$$

$$= \sum_{X_1,\ldots,X_n} \prod_{i=1}^{n} P(X_i|pa(X_i))\, U(pa(U))$$

- To find the optimal decision we can use VE:
  1. Create a factor for each conditional probability and for the utility
  2. Sum out all random variables, one at a time
     - This creates a factor on D that gives the expected utility for each d_i
  3. Choose the d_i with the maximum value in the factor

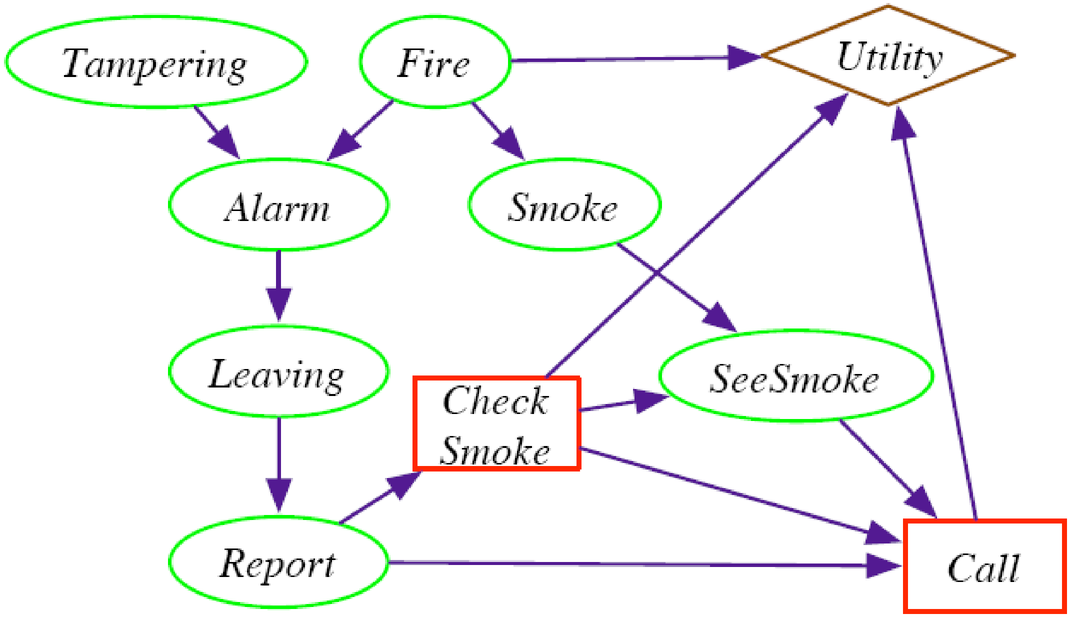
Recap: Sequential Decision Networks

- General Decision networks:
  - Just like single-stage decision networks, with one exception: the parents of decision nodes can include random variables



Recap: Sequential Decision Networks

- General Decision networks:
  - Just like single-stage decision networks, with one exception: the parents of decision nodes can include random variables

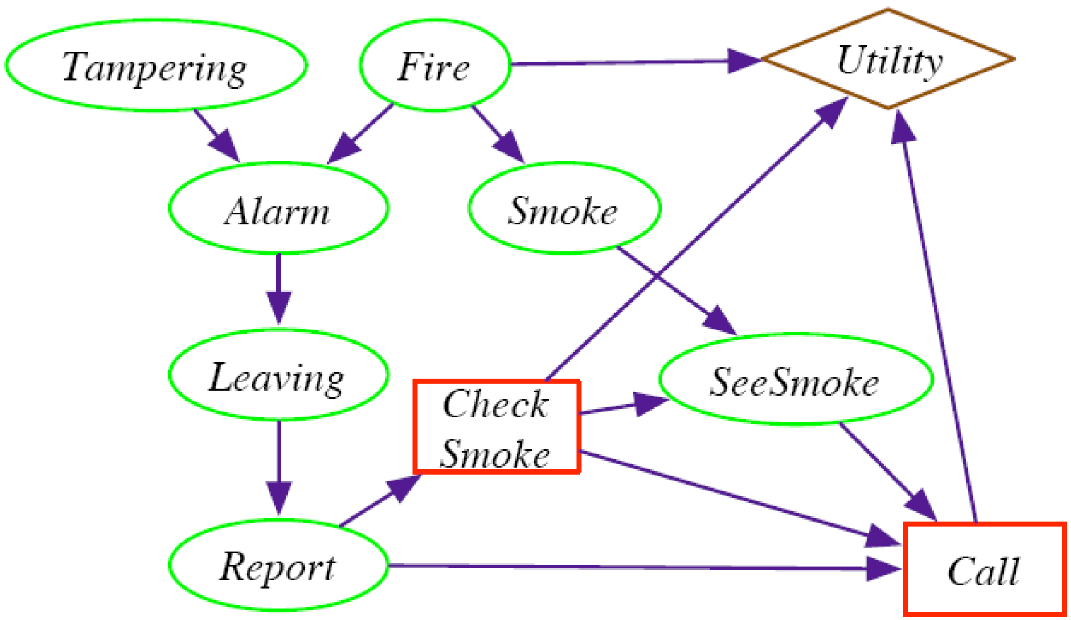


Recap: Policies for Sequential Decision Problems

Definition (Policy)
A policy π is a sequence of δ_1 ,….., δ_n decision functions

$$\delta_i : \mathrm{dom}(pa(D_i)) \rightarrow \mathrm{dom}(D_i)$$

I.e., when the agent has observed $o \in \mathrm{dom}(pD_i)$, it will do $\delta_i(o)$



- One example for a policy:
  - Check smoke (i.e. set CheckSmoke=true) if and only if Report=true
  - Call if and only if Report=true, CheckSmoke=true, SeeSmoke=true

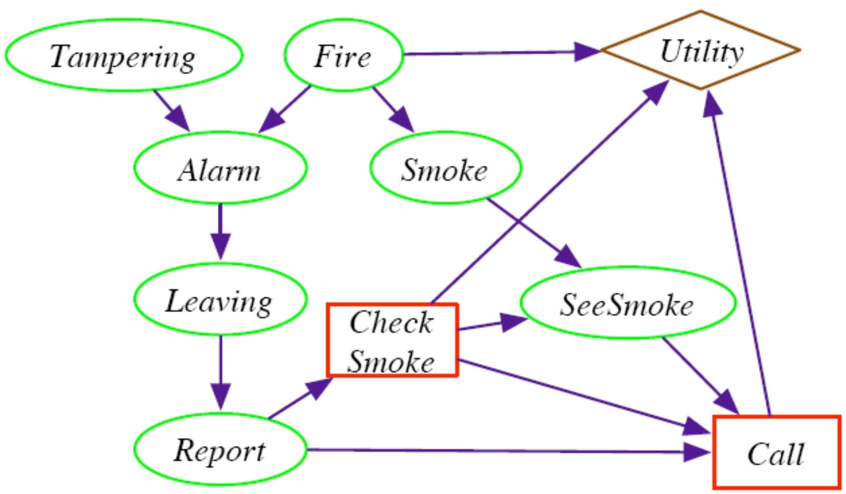
Recap: Policies for Sequential Decision Problems

Definition (Policy)

A policy π is a sequence of δ_1 ,….., δ_n decision functions

$$\delta_i : \text{dom}(pa(D_i)) \rightarrow \text{dom}(D_i)$$

I.e., when the agent has observed $o \in \text{dom}(pD_i)$, it will do $\delta_i(o)$



There are 2^2=4 possible decision functions δ_{cs} for Check Smoke:

- Each decision function needs to specify a value for each instantiation of parents

	R=t	*R=f*
$\delta_{cs}1(R)$	T	T
$\delta_{cs}2(R)$	T	F
$\delta_{cs}3(R)$	F	T
$\delta_{cs}4(R)$	F	F

Recap: Policies for Sequential Decision Problems

Definition (Policy)
A policy π is a sequence of $\delta_1, \ldots\ldots, \delta_n$ decision functions

$$\delta_i : \mathrm{dom}(pa(D_i)) \rightarrow \mathrm{dom}(D_i)$$

I.e., when the agent has observed $o \in \mathrm{dom}(pD_i)$, it will do $\delta_i(o)$

There are 2^8=256 possible decision functions δ_{cs} for Call:

	R=t, CS=t, SS=t	*R=t, CS=t, SS=f*	*R=t, CS=f, SS=t*	*R=t, CS=f, SS=f*	*R=f, CS=t, SS=t*	*R=f, CS=t, SS=f*	*R=f, CS=f, SS=t*	*R=f, CS=f, SS=f*
$\delta_{call}1(R)$	T	T	T	T	T	T	T	T
$\delta_{call}2(R)$	T	T	T	T	T	T	T	F
$\delta_{call}3(R)$	T	T	T	T	T	T	F	T
$\delta_{call}4(R)$	T	T	T	T	T	T	F	F
$\delta_{call}5(R)$	T	T	T	T	T	F	T	T
…	…	…	…	…	…	…	…	…
$\delta_{call}256(R)$	F	F	F	F	F	F	F	F

Copy-paste typos in printout

Recap: How many policies are there?

- If a decision D has k binary parents, how many assignments of values to the parents are there?
  - 2^k
- If there are b possible value for a decision variable, how many different decision functions are there for it if it has k binary parents?

2^{kp}	$b*2^k$	b^{2^k}	2^{k^b}

Recap: How many policies are there?

- If a decision D has k binary parents, how many assignments of values to the parents are there?
  - 2^k
- If there are b possible value for a decision variable, how many different decision functions are there for it if it has k binary parents?
  - b^{2^k}, because there are 2^k possible instantiations for the parents and for every instantiation of those parents, the decision function could pick any of b values
- If there are *d* decision variables, each with *k* binary parents and *b* possible actions, how many policies are there?

db^k | b^{dk} | $d(b^{2^k})$ | $(b^{2^k})^d$

Recap: How many policies are there?

- If a decision D has k binary parents, how many assignments of values to the parents are there?
  - 2^k

- If there are b possible value for a decision variable, how many different decision functions are there for it if it has k binary parents?
  - b^{2^k}, because there are 2^k possible instantiations for the parents and for every instantiation of those parents, the decision function could pick any of b values

- If there are *d* decision variables, each with *k* binary parents and *b* possible actions, how many policies are there?
  - $(b^{2^k})^d$, because there are b^{2^k} possible decision functions for each decision, and a policy is a combination of d such decision functions

Lecture Overview

- Recap: Sequential Decision Problems and Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

Possible worlds satisfying a policy

Definition (Satisfaction of a policy)
A possible world w satisfies a policy π, written $w \models \pi$, if the value of each decision variable in w is the value selected by its decision function in policy π (when applied to w)

- Consider our previous example policy:
  - Check smoke (i.e. set CheckSmoke=true) if and only if Report=true
  - Call if and only if Report=true, CheckSmoke=true, SeeSmoke=true
- Does the following possible world satisfy this policy?

  $\neg$tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, call

  Yes No

Possible worlds satisfying a policy

Definition (Satisfaction of a policy)
A possible world w satisfies a policy π, written $w \models \pi$, if the value of each decision variable in w is the value selected by its decision function in policy π (when applied to w)

- Consider our previous example policy:
  - Check smoke (i.e. set CheckSmoke=true) if and only if Report=true
  - Call if and only if Report=true, CheckSmoke=true, SeeSmoke=true

- Do the following possible worlds satisfy this policy?

  ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, call

  - Yes! Conditions are satisfied for each of the policy's decision functions

  ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, ¬call

  Yes No

Possible worlds satisfying a policy

Definition (Satisfaction of a policy)
A possible world w satisfies a policy π, written $w \models \pi$, if the value of each decision variable in w is the value selected by its decision function in policy π (when applied to w)

- Consider our previous example policy:
  - Check smoke (i.e. set CheckSmoke=true) if and only if Report=true
  - Call if and only if Report=true, CheckSmoke=true, SeeSmoke=true
- Do the following possible worlds satisfy this policy?

$\neg$tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, call

- Yes! Conditions are satisfied for each of the policy's decision functions

$\neg$tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, $\neg$call

- No! The policy says to call if Report and CheckSmoke and SeeSmoke all true

$\neg$tampering,fire,alarm,leaving,$\neg$report,$\neg$smoke,$\neg$checkSmoke,$\neg$seeSmoke,$\neg$call

Yes No

- Yes! Policy says to neither check smoke nor call when there is no report

Expected utility of a policy

Definition (expected utility of a policy)
The expected utility $E[\pi]$ of a policy π is:

$$E[\pi] = \sum_{w \models \pi} P(w)\, U(w)$$

This term is zero if D_j's value does not agree with what the policy dictates given D_j's parents.

$$E[\pi] = \sum_{w \models \pi} P(w)\, U(w)$$

$$= \sum_{X_1,\ldots,X_n,D_1,\ldots,D_m} \prod_{i=1}^{n} P(X_i | pa(X_i)) \prod_{j=1}^{m} (\delta_j(pa(D_j)) = D_j)\ U(pa(U))$$

Optimality of a policy

Definition (expected utility of a policy)

The expected utility $E[\pi]$ of a policy π is:

$$E[\pi] = \sum_{w \models \pi} P(w)\, U(w)$$

Definition (optimal policy)

An optimal policy π_{max} is a policy whose expected utility is maximal among all possible policies $\prod$:

$$\pi_{max} \in \underset{\pi \in \prod}{\operatorname{argmax}}\, E[\pi]$$

Lecture Overview

- Recap: Sequential Decision Problems and Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

One last operation on factors: maxing out a variable

- Maxing out a variable is similar to marginalization
  - But instead of taking the sum of some values, we take the max

$$\boxed{\left(\max_{X_1}\right)\left(X_2,\ldots,X_j\right)=\max_{x\in dom(X_1)} f(X_1=x, X_2,\ldots,X_j)}$$

$$\boxed{\max_B f_3(A,B,C) = f_4(A,C)}$$

B	A	C	f_3(A,B,C)
t	t	t	0.03
t	t	f	0.07
f	t	t	0.54
f	t	f	0.36
t	f	t	0.06
t	f	f	0.14
f	f	t	0.48
f	f	f	0.32

A	C	f_4(A,C)
t	t	0.54
t	f	0.36
f	t	?
f	f	

0.32 0.06 0.48 0.14

One last operation on factors: maxing out a variable

- Maxing out a variable is similar to marginalization
  - But instead of taking the sum of some values, we take the max

$$\boxed{\left(\max_{X_1}\right)(X_2,\ldots,X_j) = \max_{x \in dom(X_1)} f(X_1 = x, X_2, \ldots, X_j)}$$

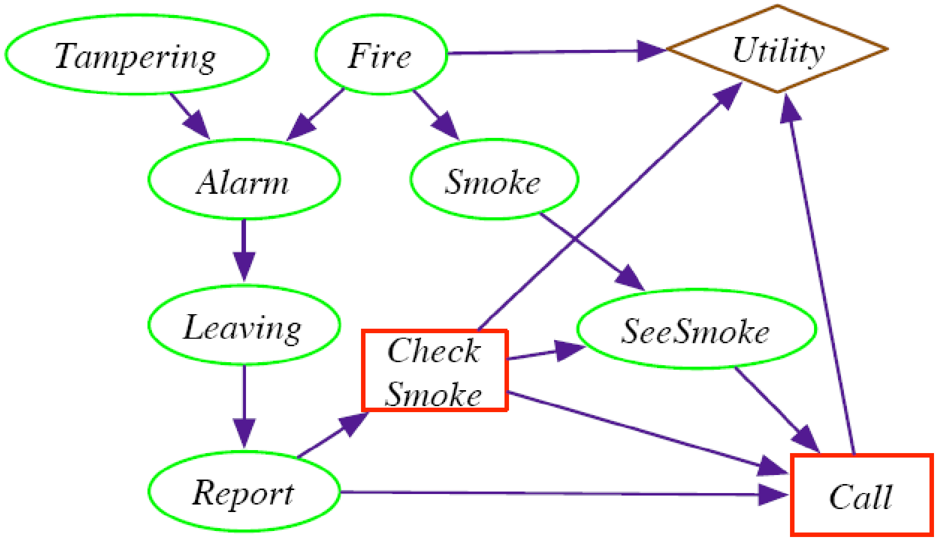
$$\boxed{\max_B f_3(A,B,C) = f_4(A,C)}$$

B	A	C	$f_3(A,B,C)$
t	t	t	0.03
t	t	f	0.07
f	t	t	0.54
f	t	f	0.36
t	f	t	0.06
t	f	f	0.14
f	f	t	0.48
f	f	f	0.32

A	C	$f_4(A,C)$
t	t	0.54
t	f	0.36
f	t	0.48
f	f	0.32

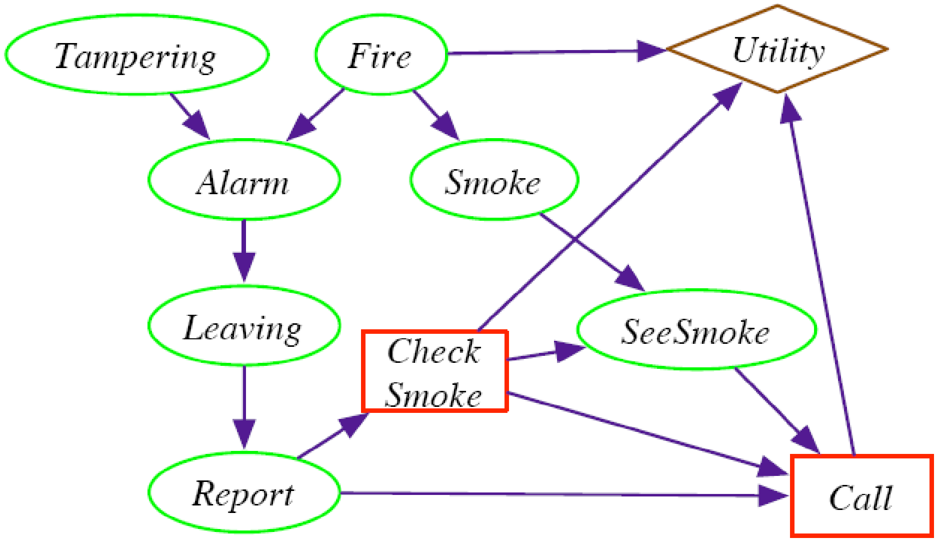
The no-forgetting property

- A decision network has the no-forgetting property if
  - Decision variables are totally ordered: $D_1, \ldots, D_m$
  - If a decision D_i comes before D_j ,then
    - D_i is a parent of D_j
    - any parent of D_i is a parent of D_j



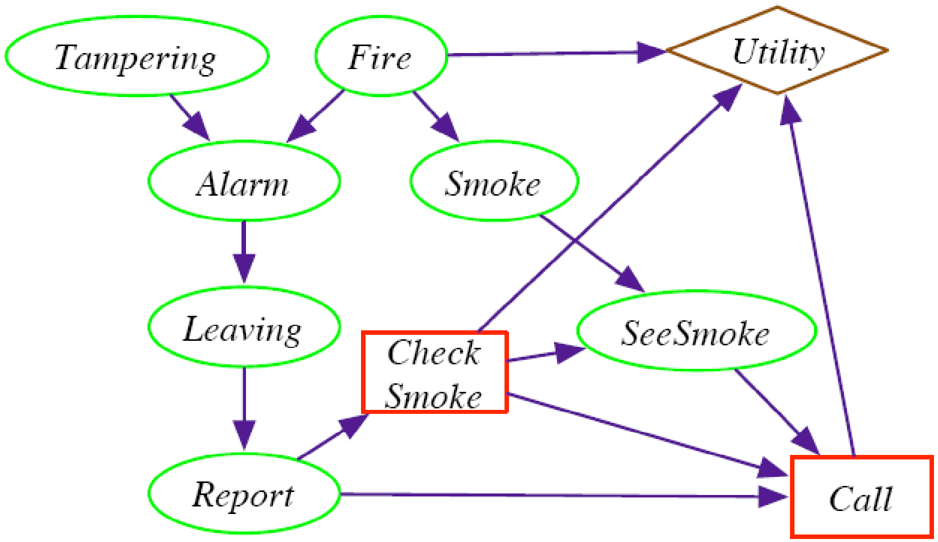
Idea for finding optimal policies with VE

- Idea for finding optimal policies with variable elimination (VE): Dynamic programming: precompute optimal future decisions
  - Consider the last decision D to be made
    - Find optimal decision D=d for each instantiation of D's parents
      - For each instantiation of D's parents, this is just a single-stage decision problem
    - Create a factor of these maximum values: max out D
      - I.e., for each instantiation of the parents, what is the best utility I can achieve by making this last decision optimally?
    - Recurse to find optimal policy for reduced network (now one less decision)



Finding optimal policies with VE

1. Create a factor for each CPT and a factor for the utility
2. While there are still decision variables
   - 2a: Sum out random variables that are not parents of a decision node.
     - E.g Tampering, Fire, Alarm, Smoke, Leaving
   - 2b: Max out last decision variable D in the total ordering
     - Keep track of decision function
3. Sum out any remaining variable:
   this is the expected utility of the optimal policy.



Computational complexity of VE for finding optimal policies

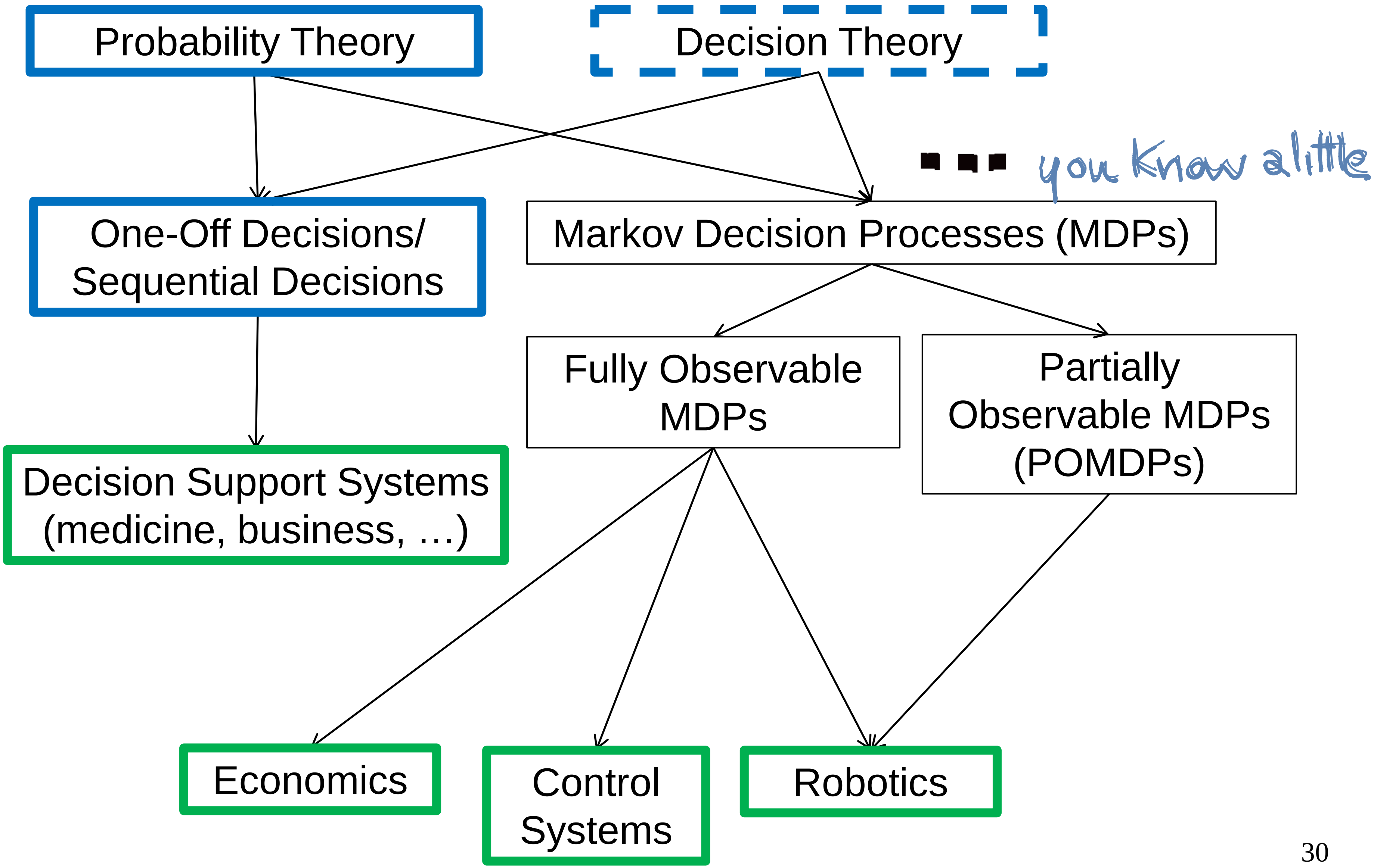
- We saw:
  For *d* decision variables (each with *k* binary parents and *b* possible actions), there are $(b^{2^k})^d$ policies
  - All combinations of (b^{2^k}) decision functions per decision

- Variable elimination saves the final exponent:
  - Dynamic programming: consider each decision functions only once
  - Resulting complexity: $O(d * b^{2^k})$
  - Much faster than enumerating policies (or search in policy space), but still doubly exponential
  - CS422: approximation algorithms for finding optimal policies

Lecture Overview

- Recap: Sequential Decision Problems and Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

Big Picture: Planning under Uncertainty



Decision Theory: Decision Support Systems

E.g., Computational Sustainability

- New interdisciplinary field, AI is a key component
  - Models and methods for decision making concerning the management and allocation of resources
  - to solve most challenging problems related to sustainability
- Often constraint optimization problems. E.g.
  - Energy: when are where to produce green energy most economically?
  - Which parcels of land to purchase to protect endangered species?
  - Urban planning: how to use budget for best development in 30 years?

Source: http://www.computational-sustainability.org/

Planning Under Uncertainty

- Learning and Using POMDP models of Patient-Caregiver Interactions During Activities of Daily Living

- Goal: Help older adults living with cognitive disabilities (such as Alzheimer's) when they:
  - forget the proper sequence of tasks that need to be completed
  - lose track of the steps that they have already completed

Source: *Jesse Hoey UofT 2007*

Planning Under Uncertainty

Helicopter control: MDP, reinforcement learning

(states: all possible positions, orientations, velocities and angular velocities)

Source: *Andrew Ng*, 2004

Planning Under Uncertainty

Autonomous driving: DARPA Grand Challenge

Source: *Sebastian Thrun*

Learning Goals For Today's Class

- Sequential decision networks
  - Represent sequential decision problems as decision networks
  - Explain the non forgetting property
- Policies
  - Verify whether a possible world satisfies a policy
  - Define the expected utility of a policy
  - Compute the number of policies for a decision problem
  - Compute the optimal policy by Variable Elimination

Announcements

- Final exam is next Monday, April 11. DMP 310, 3:30-6pm
  - The list of short questions is online … please use it!
  - Also use the practice exercises (online on course website)
- Office hours this week
  - Simona: Tuesday, 1pm-3pm (change from 10-12am)
  - Mike: Wednesday 1-2pm, Friday 10-12am
  - Vasanth: Thursday, 3-5pm
  - Frank:
    - X530: Tue 5-6pm, Thu 11-12am
    - DMP 110: 1 hour after each lecture
- Optional Rainbow Robot tournament: this Friday
  - Hopefully in normal classroom (DMP 110)
  - Vasanth will run the tournament,
    I'll do office hours in the same room (this is 3 days before the final)