Intelligent Agents:
Web-mining Agents

Probabilistic Graphical Models

Decision Making

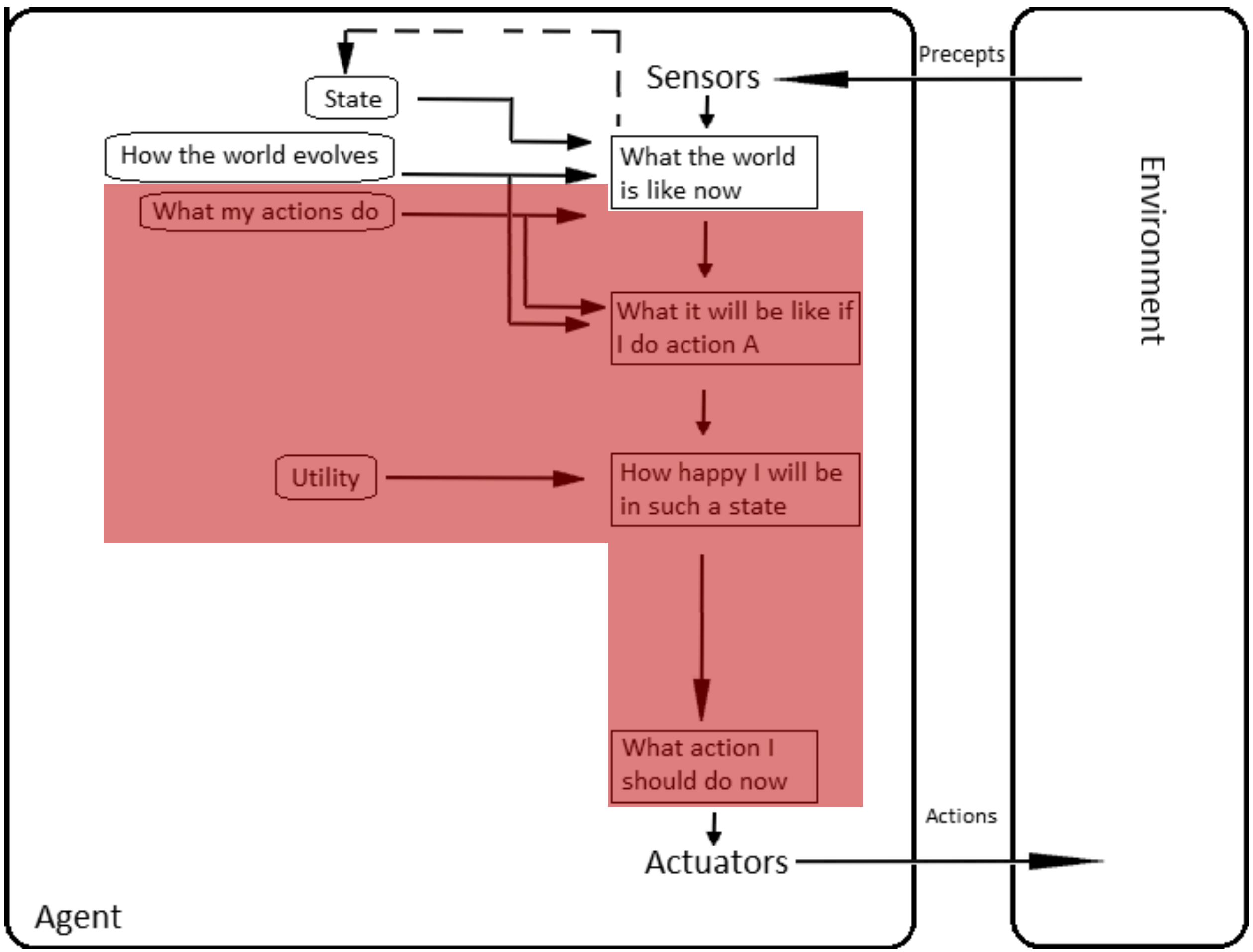
Tanya Braun

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Probabilistic Graphical Models (PGMs)

1. Recap: **Propositional** modelling
   - Factor model, Bayesian network, Markov network
   - Semantics, inference tasks + algorithms + complexity
2. **Probabilistic relational models** (PRMs)
   - Parameterised models, Markov logic networks
   - Semantics, inference tasks
3. **Lifted inference**
   - LVE, LJT, FOKC
   - Theoretical analysis
4. **Lifted learning**
   - Recap: propositional learning
   - From ground to lifted models
   - Direct lifted learning
5. **Approximate Inference: Sampling**
   - Importance sampling
   - MCMC methods
6. **Sequential models & inference**
   - Dynamic PRMs
   - Semantics, inference tasks + algorithms + complexity, learning
7. **Decision making**
   - (Dynamic) Decision PRMs
   - Semantics, inference tasks + algorithms, acting
8. **Continuous Space**
   - Gaussian distributions
   - Probabilistic soft logic

Setting: Agent with Utilities



Outline: 7. Decision Making

A. *Static decision making*
- Utility theory
- Parameterised decision models (PDecM)
  - Modelling, semantics, inference tasks
  - Inference algorithm: LVE as an example
- Value of information

B. Sequential decision making
- Parameterised dynamic decision models (PDDecM)
- Temporal MEU problem, inference
- Acting

Expected Utility

- Randvar R with n range values $r_1, \dots, r_n$ and distribution $(p_1, \dots, p_n)$
  - E.g.: R encodes the state reached after doing an action $A = a$ under uncertainty
- Function U of R
  - E.g., U is the utility of a state
- The expected utility of $A = a$ is

$$EU[A = a] = \sum_{i=1}^{n} P(R = r_i | A = a) \cdot U(R = r_i)$$

MEU Principle

- A rational agent should choose the action that maximises agent's expected utility
- This is the basis of the field of decision theory
- The MEU principle provides a normative criterion for rational choice of action

AI is solved!!!

Not quite…

- Must have complete model of:
  - Actions
  - Utilities
  - States
- Even if you have a complete model, it might be computationally intractable
- In fact, a truly rational agent takes into account the utility of reasoning as well – bounded rationality
- Nevertheless, great progress has been made in this area, and we are able to solve much more complex decision-theoretic problems than ever before

Setting

- Agent can perform actions in an environment
  - Environment
    - Episodic, i.e., not sequential
      - Next episode does not depend on the previous episode
      - So called static models (vs. dynamic/temporal, next lecture)
    - Non-deterministic
      - Outcomes of actions not unique
      - Associated with probabilities (→ probabilistic model)
    - Partially observable
      - Latent, i.e., not observable, random variables
  - Agent has preferences over states/action outcomes
    - Encoded in utility or utility function → Utility theory
- "Decision theory = Utility theory + Probability theory"
  - Model the world with a probabilistic model
  - Model preferences with a utility (function)
  - Find action that leads to the maximum expected utility, also called decision making

Utility Theory

Preferences, Utilities, Dominance, Preference structure

Preferences

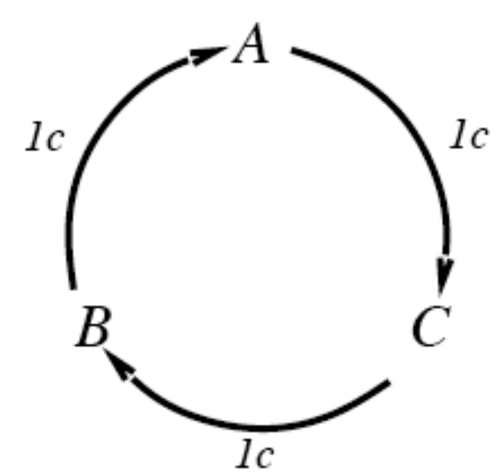
- An agent chooses among prizes (A, B, etc.) and lotteries, i.e., situations with uncertain prizes
  - Outcome of a nondeterministic action is a lottery
- Lottery $L = [p, A; (1-p), B]$
  - A and B can be lotteries again
  - Prizes are special lotteries: $[1, R; 0, \text{not } R]$
  - More than two outcomes:
    - $L = [p_1, S_1; p_2, S_2; \cdots; p_n, S_n], \sum_{i=1}^{n} p_i = 1$
- Notation
  - $A \succ B$ A preferred to B
  - $A \sim B$ indifference between A and B
  - $A \succsim B$ B not preferred to A

Rational preferences

- Idea: preferences of a rational agent must obey constraints
- Rational preferences ⇒ behaviour describable as maximisation of expected utility

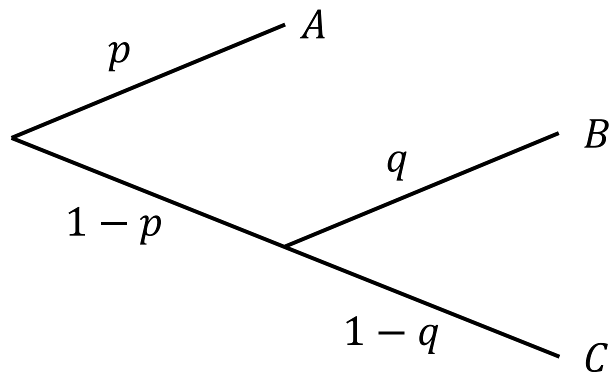
Rational preferences contd.

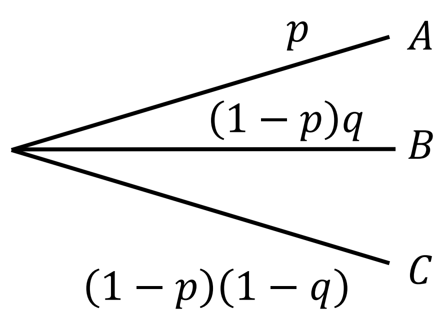
- Violating constraints leads to self-evident irrationality
- Example
  - An agent with intransitive preferences can be induced to give away all its money
  - If $B \succ C$, then an agent who has C would pay (say) 1 cent to get B
  - If $A \succ B$, then an agent who has B would pay (say) 1 cent to get A
  - If $C \succ A$, then an agent who has A would pay (say) 1 cent to get C



Axioms of Utility Theory

1. Orderability
   - $(A \succ B) \vee (A \prec B) \vee (A \sim B)$
   - $\{\prec, \succ, \sim\}$ jointly exhaustive, pairwise disjoint
2. Transitivity
   - $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$
3. Continuity
   - $A \succ B \succ C \Rightarrow$
     $\exists p \; [p, A;\; 1-p, C] \sim B$
4. Substitutability
   - $A \sim B \Rightarrow$
     $[p, A; 1-p, C] \sim [p, B; 1-p, C]$
   - Also holds if replacing $\sim$ with $\succ$
5. Monotonicity
   - $A \succ B \Rightarrow$
     $(p \geq q \Leftrightarrow$
     $[p, A; 1-p, B]$
     $\gtrsim \; [q, A; 1-q, B])$
6. Decomposability
   - $[p, A;\; 1-p, [q, B;\; 1-q, C]] \sim$
     $[p, A;\; (1-p)q, B; (1-p)(1-q), C]$





Decomposability: There is no fun in gambling.

And Then There Was Utility

- Theorem (Ramsey, 1931; von Neumann and Morgenstern, 1944):
  - Given preferences satisfying the constraints, there exists a real-valued function U such that

$$U(A) \geq U(B) \Leftrightarrow A \gtrsim B$$
$$U([p_1, S_1; \ldots; p_n, S_n]) = \sum_i p_i U(S_i)$$

- MEU principle
  - Choose the action that maximises expected utility

- Note: an agent can be entirely rational (consistent with MEU) without ever representing or manipulating utilities and probabilities
  - E.g., a lookup table for perfect tictactoe

Utilities

- Utilities map states to real numbers.
  Which numbers?

- Standard approach to assessment of human utilities:
  - Compare a given state A to a standard lottery L_p that has
    - "best possible outcome" $\top$ with probability p
    - "worst possible catastrophe" $\bot$ with probability $(1 - p)$
  - Adjust lottery probability p until $A \sim L_p$

$$\text{pay-\$30-and-continue-as-before} \sim L \begin{cases} \xrightarrow{0.999999} \text{continue as before} \\ \xrightarrow{0.000001} \text{instant death} \end{cases}$$

Utility scales

- Normalised utilities: $u_\top = 1.0, u_\perp = 0.0$
  - Utility of lottery $L \sim$ (pay-\$30-and-continue-as-before): $U(L) = u_\top \cdot 0.999999 + u_\perp \cdot 0.000001 = 0.999999$
- Micromorts: one-millionth chance of death
  - Useful for Russian roulette, paying to reduce product risks, etc.
- QALYs: quality-adjusted life years
  - Useful for medical decisions involving substantial risk
- Behaviour is invariant w.r.t. positive linear transformation

$$U'(r) = k_1 U(r) + k_2$$

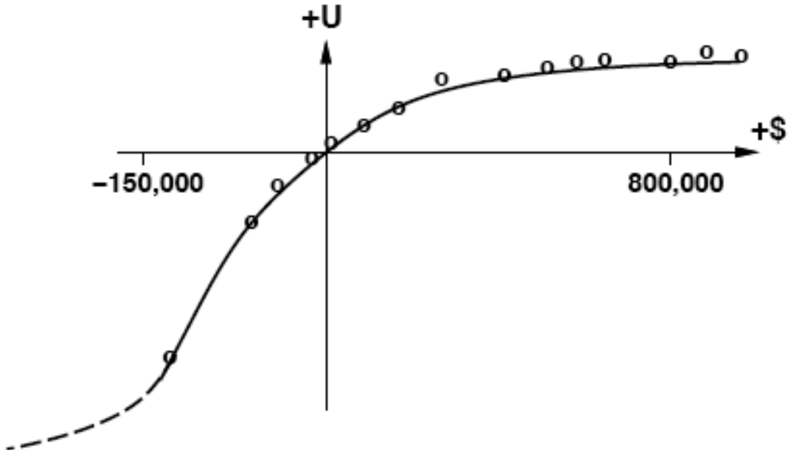
  - No unique utility function; $U'(r)$ and $U(r)$ yield same behaviour

Ordinal Utility Functions

- With deterministic prizes only (no lottery choices), only ordinal utility can be determined, i.e., total order on prizes
  - Ordinal utility function also called value function
  - Provides a ranking of alternatives (states), but not a meaningful metric scale (numbers do not matter)

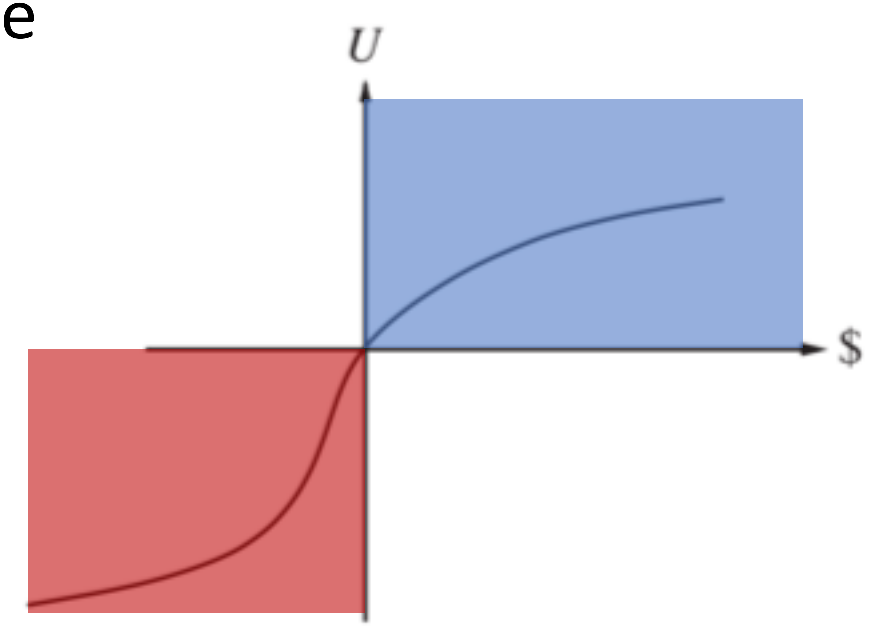
Money

- Money does not behave as a utility function
- Given a lottery L with expected monetary value $EMV(L)$, usually $U(L) < U\left(S_{EMV(L)}\right)$, i.e., people are risk-averse
  - S_n: state of possessing total wealth \$n
  - Utility curve
    - For what probability p am I indifferent between a prize x and a lottery $[p, \$M; (1-p), \$0]$ for large M?
    - Right: Typical empirical data, extrapolated with risk-prone behaviour for negative wealth



Money Versus Utility

- Money $\neq$ Utility
  - More money is better, but not always in a linear relationship to the amount of money
- Expected Monetary Value
  - Risk-averse
    - $U(L) < U(S_{EMV(L)})$
  - Risk-seeking
    - $U(L) > U(S_{EMV(L)})$
  - Risk-neutral
    - $U(L) = U(S_{EMV(L)})$
    - Linear curve
    - For small changes in wealth relative to current wealth



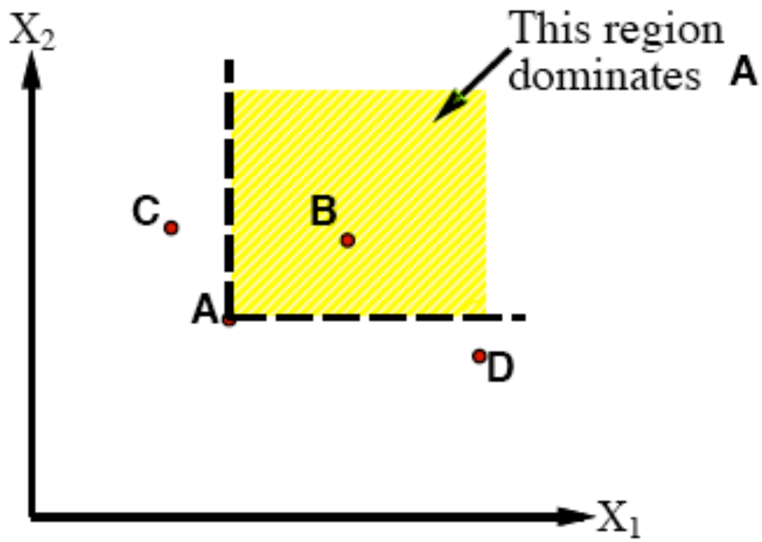
Multi-attribute Utility Theory

- A given state may have multiple utilities
  - ...because of multiple evaluation criteria
  - ...because of multiple agents (interested parties) with different utility functions

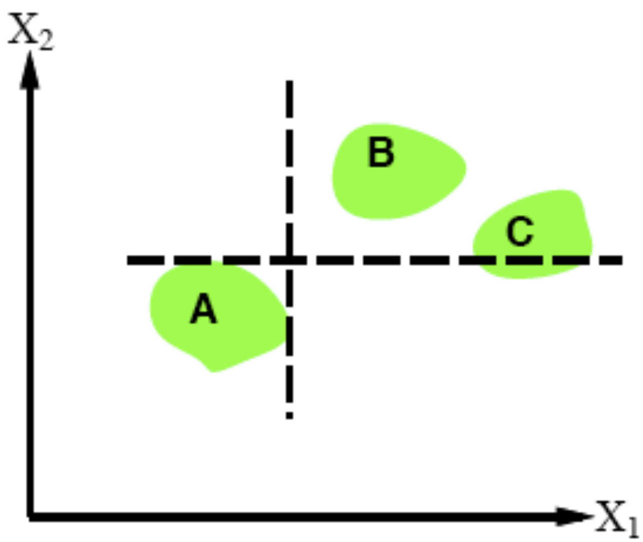
- We will look at
  - Cases in which decisions can be made *without* combining the attribute values into a single utility value
    - Strict dominance
  - Cases in which the utilities of attribute combinations can be specified very concisely

Strict dominance

- Typically define attributes such that U is monotonic in each →
- Strict dominance
  - Choice B strictly dominates choice A iff
    $\forall\ i : X_i(B) \geq X_i(A)$ (and hence $U(B) \geq U(A)$)



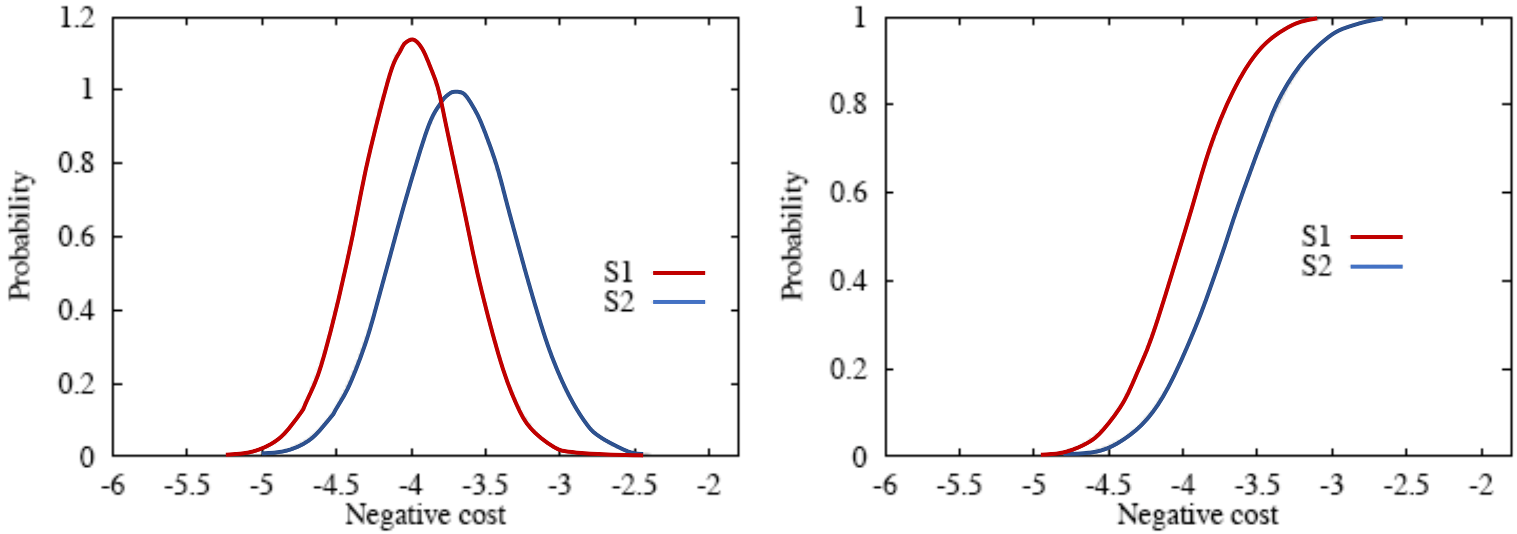
Deterministic attributes



Uncertain attributes

Stochastic dominance

- Cumulative distribution p_1 first-order stochastically dominates distribution p_2 iff
$$\forall x : p_2(x) \leq p_1(x)$$
  - With a strict inequality for some interval
  - Then, $E_{p_1} > E_{p_2}$ (E referring to expected value)
    - The reverse is not necessarily true
  - Does not imply that every possible return of the superior distribution is larger than every possible return of the inferior distribution
- Example:
  - As we have *negative costs*, S2 dominates S1 with $\forall x : p_{S_2}(x) \leq p_{S_1}(x)$



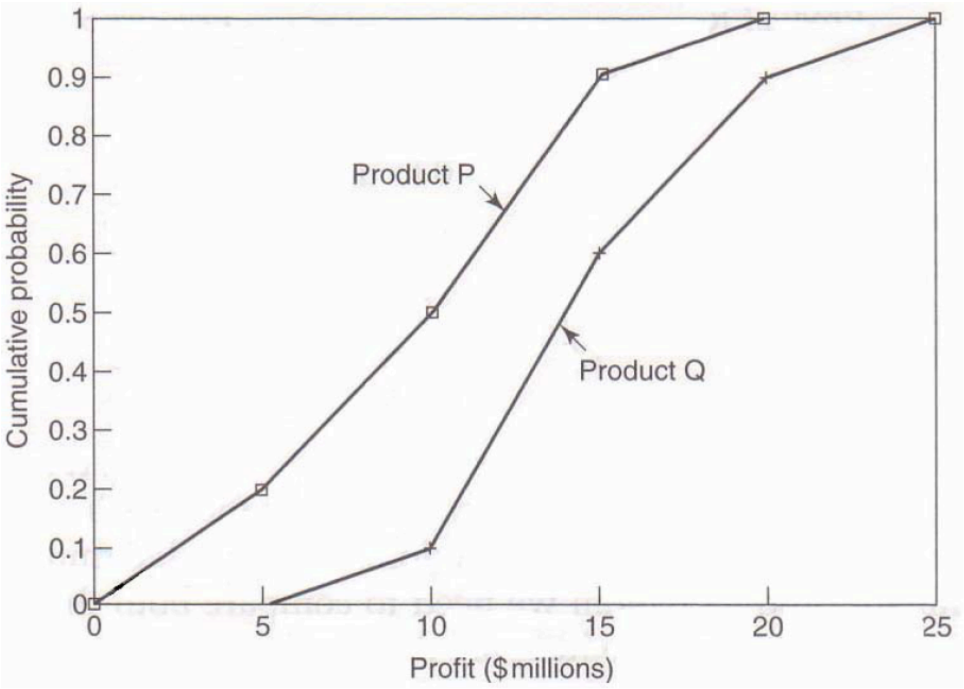
Example

- Product P

Profit ($m)	Probability
0 to under 5	0.2
5 to under 10	0.3
10 to under 15	0.4
15 to under 20	0.1

- Product Q

Profit ($m)	Probability
0 to under 5	0.0
5 to under 10	0.1
10 to under 15	0.5
15 to under 20	0.3
20 to under 25	0.1



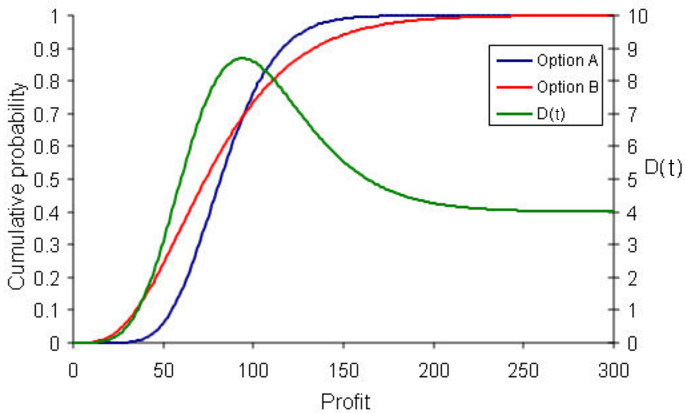
P first-order stochastically dominates Q.

Stochastic dominance

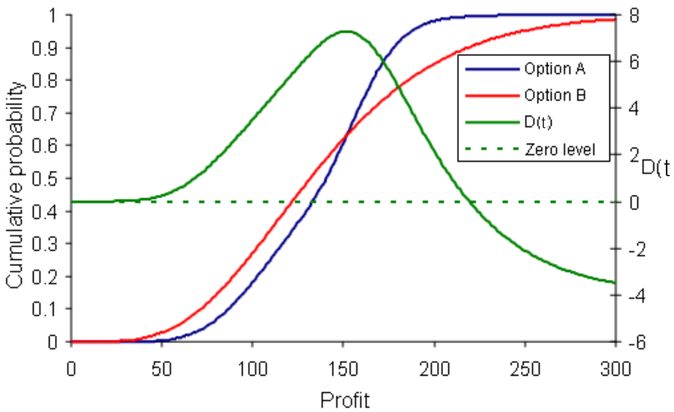
- Cumulative distribution p_1 second-order stochastically dominates distribution p_2 iff

$$\forall\, t : \int_{-\infty}^{t} p_2(x)\, dx \leq \int_{-\infty}^{t} p_1(x)\, dx$$

  - Or: $D(t) = \int_{-\infty}^{t} p_1(x) - p_2(x)\, dx \geq 0$
  - With a strict inequality for some interval
  - Then, $E_{p_1} \geq E_{p_2}$ (E referring to expected value)
- Example:
  - Second-order stochastic dominance



  - No dominance



https://people.duke.edu/~dgraham/ECO_463/Handouts/StochasticDominance.pdf
Figures: https://www.vosesoftware.com/riskwiki/Stochasticdominancetests.php (t=z)

Preference Structure

- To specify the complete utility function $U(r_1, \dots, r_n)$, we need d^n values in the worst case
  - n attributes
  - each attribute with d distinct possible values
  - Worst case meaning: Agent's preferences have no regularity at all
- Supposition in multi-attribute utility theory
  - Preferences of typical agents have much more structure
- Approach
  - Identify regularities in the preference behaviour
  - Use so-called representation theorems to show that an agent with a certain kind of preference structure has a utility function

$$U(r_1, \dots, r_n) = F[f_1(r_1), \dots, f_n(r_n)]$$

    - where F is hopefully a simple function such as addition

Preference structure: Deterministic

- R_1 and R_2 preferentially independent (PI) of R_3 iff
  - Preference between $\langle r_1, r_2, r_3 \rangle$ and $\langle r_1', r_2', r_3 \rangle$ does not depend on r_3
  - E.g., $\langle Noise, Cost, Safety \rangle$
    - $\langle 20{,}000\ suffer, \$4.6\ billion, 0.06\ deaths/month \rangle$
    - $\langle 70{,}000\ suffer, \$4.2\ billion, 0.06\ deaths/month \rangle$
- Theorem (Leontief, 1947)
  - If every pair of attributes is PI of its complement, then every subset of attributes is PI of its complement
    - Called mutual PI (MPI)
- Theorem (Debreu, 1960):
  - MPI ⇒ ∃ *additive* value function

$$V(r_1, \ldots, r_n) = \sum_i V_i(r_i)$$

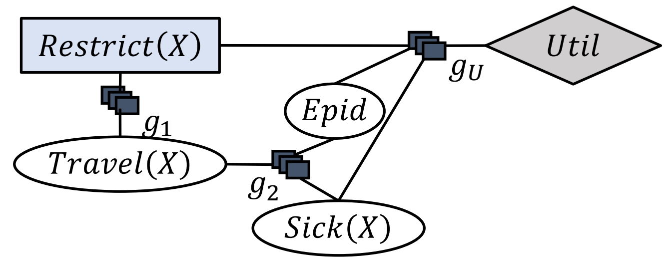
  - Hence assess n single-attribute functions
  - Often a good approximation

Preference structure: Stochastic

- Need to consider preferences over lotteries
- R is utility-independent (UI) of S iff
  - Preferences over lotteries in R do not depend on s
- Mutual UI (Keeney, 1974): each subset is UI of its complement $\Rightarrow \exists$ *multiplicative* utility function
  - For $n = 3$:
    $$\begin{aligned} U = {} & k_1U_1 + k_2U_2 + k_3U_3 \\ & + k_1k_2U_1U_2 + k_2k_3U_2U_3 + k_3k_1U_3U_1 \\ & + k_1k_2k_3U_1U_2U_3 \end{aligned}$$
  - I.e., requires only n single-attribute utility functions and n constants

Interim Summary

- Preferences
  - Preferences of a rational agent must obey constraints
- Utilities
  - Rational preferences = describable as maximisation of expected utility
  - Utility axioms
  - MEU principle
- Dominance
  - Strict dominance
  - First-order + second-order stochastic dominance
- Preference structure
  - (Mutual) preferential independence
  - (Mutual) utility independence



Parameterised Decision Models (PDecMs)

Modelling, Semantics, Inference Tasks

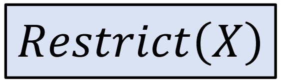
Inference with LVE

Decision Networks/Models

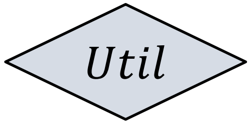
- Extend a PGM to handle actions and utilities
  - Decision variables
  - Utility variables
- Also called influence diagrams
- Use an inference method of one's choosing to find actions that lead to the highest expected utility
- Also allows to perform so-called *Value of Information* calculations
  - Is it worth it to spend resources on getting more information (in the form of evidence)?

Decision PRVs & Utility PRVs

- Action and utility PRVs follow the same syntax as normal PRVs
- Decision PRV A:
  range $\mathcal{R}(A) = \{a_i\}_{i=1}^{n}$ set of possible actions
  - Actions a_i mutually exclusive (consistent with range def.)
  - Depicted by a rectangle in a graphical representation
  - E.g., possible travel restrictions for people X: $Restrict(X)$
    - Range values: $ban, free$



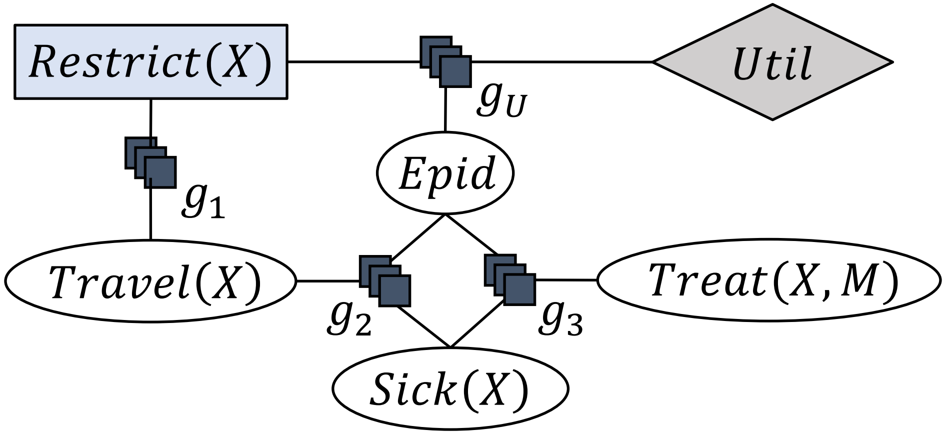
- Utility PRV U:
  range $\mathcal{R}(U) = \mathbb{R}$
  - Output variable, i.e., gets assigned a value by utility function
    - Therefore, always a leaf
  - Depicted by a diamond in a graphical representation



Decision PRVs in Parfactors

- As arguments of a parfactor
- Parfactor $\phi(\mathcal{A})_{|C}$
  - May now also contain decision PRVs in its arguments $\mathcal{A}$
  - E.g., $\phi_1\big(Restrict(X), Travel(X)\big)$

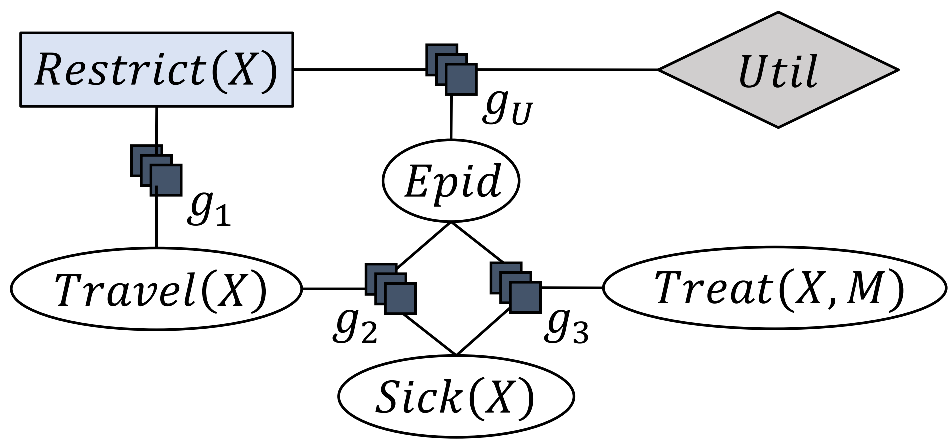
$Restrict(X)$	$Travel(X)$	ϕ_1
$free$	$false$	1
$free$	$true$	1
ban	$false$	1
ban	$true$	0



Utility PRVs in Parfactors

- Utility PRVs get assigned value by potential function
- Utility parfactor $\phi_U(\mathcal{A})_{|C}$
  - Arguments $\mathcal{A} = (A_1, \dots, A_m)$ a sequence of (decision) PRVs
  - U a utility PRV that receives the output value
  - $\phi: \times_{i=1}^{m} \mathcal{R}(A_i) \mapsto \mathcal{R}(U)$
  - C a constraint for logvars $lv(\mathcal{A}) \cup lv(U)$
  - Set of PRVs: $rv\left(\phi_U(\mathcal{A})_{|C}\right) = \{A_1, \dots, A_m\}$ (without U)
  - E.g., $\phi_{Util}(Restrict(X), Epid)$

$Restrict(X)$	$Epid$	$Util$
$free$	$false$	10
$free$	$true$	-10
ban	$false$	-20
ban	$true$	5

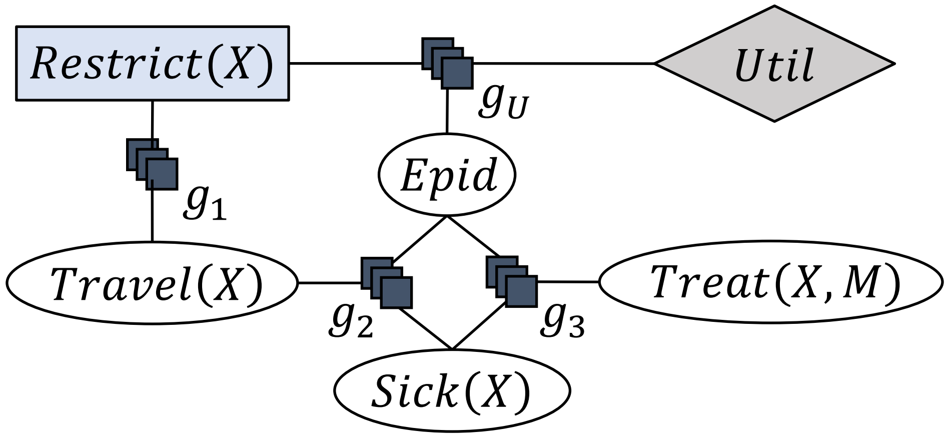


PDecMs

- PDecM = PM that allows decision PRVs in the arguments in its parfactors as well as utility parfactors
  - For simplicity, let us consider models with *one utility parfactor* mapping to one utility *randvar*
    → Strict dominance by one utility
  - Formally, PDecM

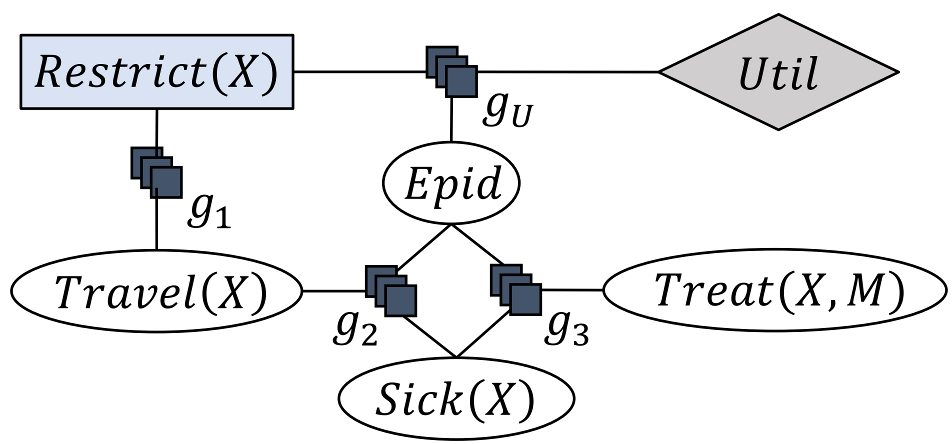
$$G = \{g_i\}_{i=1}^{n} \cup \{g_U\}$$

  - E.g.,
    $G = \{g_1, g_2, g_3, g_U\}$
    - T constraints



PDecMs: Action Assignments

- Let $\boldsymbol{A} = \{A_1, \dots, A_k\}_{|C}$ be the set of decision PRVs occurring in G with a constraint C for the logvars in $\boldsymbol{A}$
- Then, $\boldsymbol{a}$ is a compound event for $\boldsymbol{A}$ that assigns each decision PRV A_i a range value a_i
  - Refer to $\boldsymbol{a}$ as an action assignment
- E.g., without evidence in G ($\boldsymbol{E} = \emptyset$, $\top$ constraints)
  - Action $Restrict(X)$ with range $\{ban, free\}$
    - $\boldsymbol{a}_1 = \{ban\}$
    - $\boldsymbol{a}_2 = \{free\}$
  - Given another action A with range $\{a', a'', a'''\}$
    - $\boldsymbol{a}_1 = \{ban, a'\}$
    - $\boldsymbol{a}_2 = \{ban, a''\}$
    - $\boldsymbol{a}_3 = \{ban, a'''\}$
    - $\boldsymbol{a}_4 = \{free, a'\}$
    - $\boldsymbol{a}_5 = \{free, a''\}$
    - $\boldsymbol{a}_6 = \{free, a'''\}$



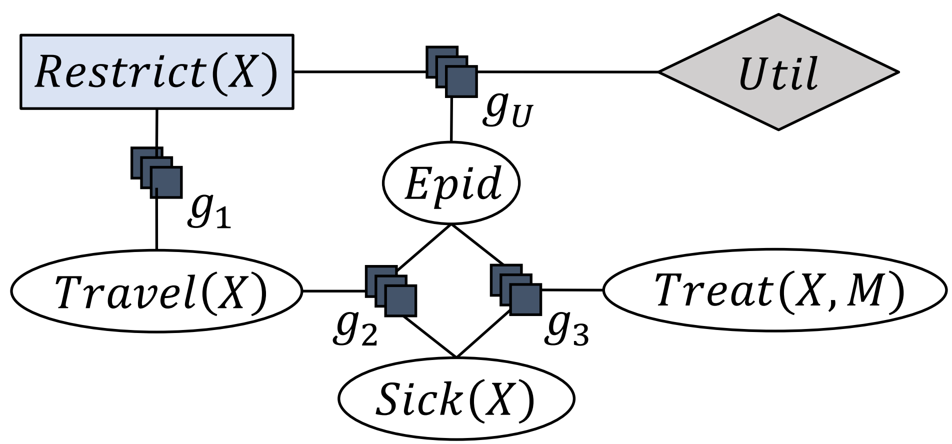
PDecMs: Setting Decisions

- Given a PDecM G and an action assignment $\boldsymbol{a}$
- Let $G_{\boldsymbol{a}}$ refer to G with $\boldsymbol{a}$ set, i.e.,
  $$G_{\boldsymbol{a}} = \text{absorb}(G, \boldsymbol{a})$$
  - In each g with decision PRV A_i,
    - Drop the lines where $A_i \neq a_i$ and the column of A_i
  - E.g., set $\boldsymbol{a}_1 = \{ban\}$ in $G = \{g_1, g_2, g_3, g_U\}$
    - $\boldsymbol{E} = \emptyset$
    - Absorb $\boldsymbol{a}_1$ in g_1 and g_U
    - $G_{\boldsymbol{a}_1} = \{g_1', g_2, g_3, g_U'\}$
      - $g_1' = \phi_1'(Travel(X))$
      - $g_U' = \phi_{Util}'(Epid)$

$Restrict(X)$	$Travel(X)$	ϕ_1
~~$free$~~	~~$false$~~	~~1~~
~~$free$~~	~~$true$~~	~~1~~
ban	$false$	1
ban	$true$	0

$Travel(X)$	ϕ_1'
$false$	1
$true$	0

$Epid$	$Util$
$false$	-20
$true$	5



PDecMs: Semantics

- Semantics of PDecM $G = \{g_i\}_{i=1}^{n} \cup \{g_U\}$
  - Given an action assignment $\boldsymbol{a}$ for the *grounded* set of decision PRVs $\boldsymbol{A} = \{A_1, \dots, A_k\}_{|C}$ occurring in G
  - Then, the semantics is given by grounding and building a full joint distribution for the non-utility parfactors

$$P_G[\boldsymbol{a}] = \frac{1}{Z} \prod_{f \in gr(G_{\boldsymbol{a}} \setminus \{g_U\})} f$$

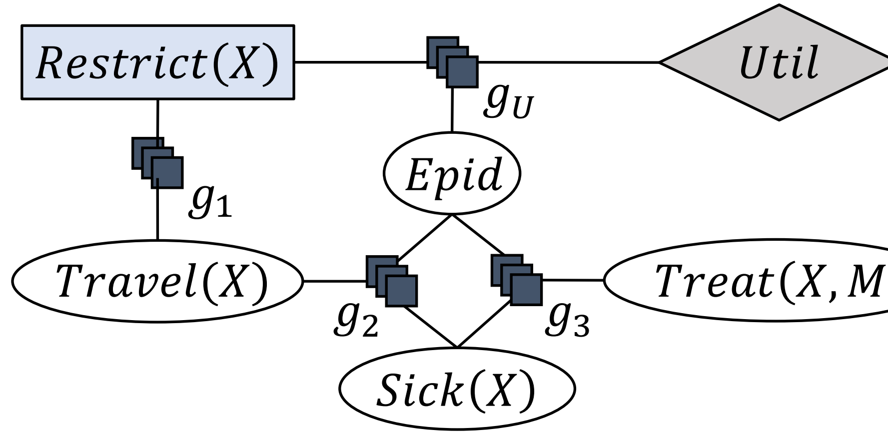
$$Z = \sum_{r_1 \in \mathcal{R}(R_1)} \dots \sum_{r_N \in \mathcal{R}(R_N)} \prod_{f \in gr(G_{\boldsymbol{a}} \setminus \{g_U\})} f$$

Semantics multiplicative with an inner product and outer sum: Multiply parfactors, then sum out PRVs.
→ Sum-product algorithms

- Utility parfactors irrelevant for probabilistic behaviour

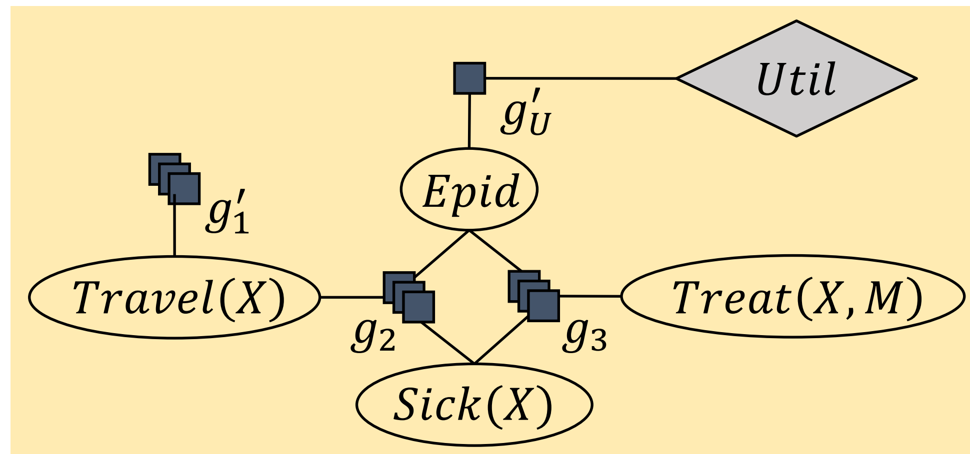
- PdecM

  $G = \{g_1, g_2, g_3, g_U\}$
  - T constraints



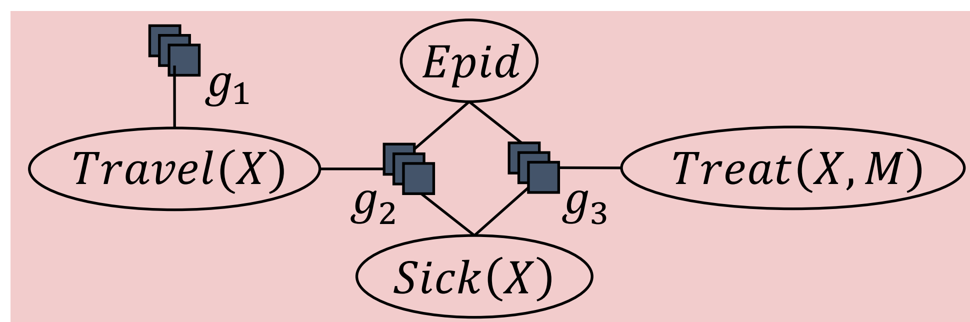
- G with $\boldsymbol{a}_1 = \{ban\}$ set

  $G_{\boldsymbol{a}_1} = \{g_1', g_2, g_3, g_U'\}$
  - $g_1' = \phi_1'(Travel(X))$
  - $g_U' = \phi_{Util}'(Epid)$



- Model relevant for query answering:

  $G = \{g_1', g_2, g_3\}$



PDecMs: Expected Utility Queries

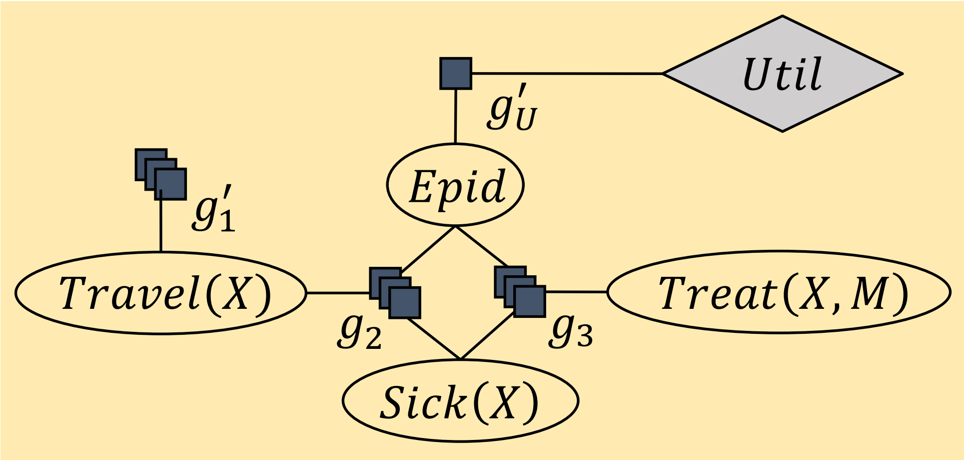
- Given a PDecM $G = \{g_i\}_{i=1}^{n} \cup \{g_U\}$
  - One can ask queries for (conditional) marginal distributions or events as before given an action assignment $\boldsymbol{a}$ based on the semantics, $P_G[\boldsymbol{a}]$
  - New query type: query for an expected utility (EU)
    - What is the expected utility of decisions $\boldsymbol{a}$ in G?

$$eu(\boldsymbol{E}, \boldsymbol{a}) = \sum_{\boldsymbol{r} \in \mathcal{R}(rv(g_U))} P(\boldsymbol{r} | \boldsymbol{E}, \boldsymbol{a}) \cdot \phi_U(\boldsymbol{r})$$

    - If a g_U contains decision PRVs or is affected by $\boldsymbol{E}$, then, of course, g_U needs to be modified accordingly
    - $P(\boldsymbol{r} | \boldsymbol{E}, \boldsymbol{a})$ means that the PRVs not occurring in this expression need to be eliminated accordingly

PDecMs: EU Query – Example

- Expected utility of $\boldsymbol{a}_1 = \{ban\}$ in $G = \{g_1, g_2, g_3, g_U\}$
$$eu(\boldsymbol{a}_1) = \sum_{e \in \mathcal{R}(Epid)} P(Epid = e | \boldsymbol{a}_1) \cdot \phi_U(Epid = e)$$
  - With $\boldsymbol{E} = \emptyset$
- Compute $P(Epid = e | \boldsymbol{a}_1)$ in G
  - By computing $P(Epid = e)$ in $G_{\boldsymbol{a}_1} = \{g_1', g_2, g_3, g_U'\}$
    - Depicted on the right



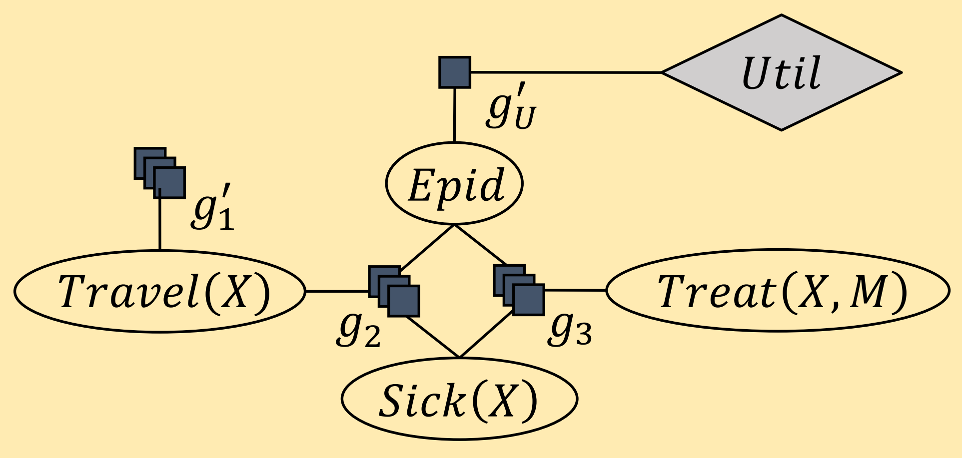
PDecMs: EU Query – Example

- Compute $P(Epid = e)$ in $G_{\boldsymbol{a}_1} = \{g_1', g_2, g_3, g_U'\}$
  - Using LVE, eliminate all other terms in $G_{\boldsymbol{a}_1}$:
    - Eliminate $Treat(X, M)$
    - Eliminate $Travel(X)$
    - Eliminate $Sick(X)$
    - Normalise result to get $P(Epid = e)$ in $G_{\boldsymbol{a}_1}$: $\phi(Epid)$
      - Corresponds to $P(Epid = e | \boldsymbol{a}_1)$ in G

$Travel(X)$	ϕ_1'
$false$	1
$true$	0

$Epid$	$Util$
$false$	-20
$true$	5

The parfactors g_1' and g_U' would look differently, had we set $\boldsymbol{a}_2 = \{free\}$.

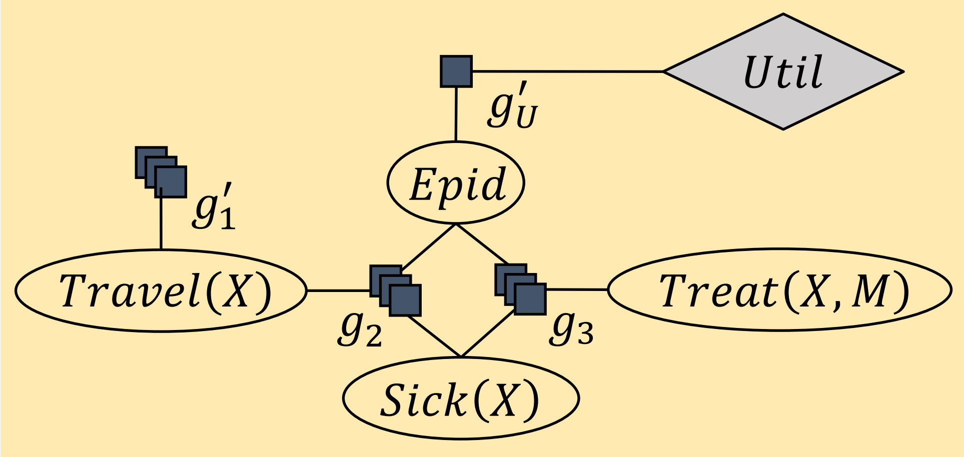


PDecMs: EU Query – Example

- Calculations with $|\mathcal{D}(M)| = 2, |\mathcal{D}(X)| = 3$:
  - Sum out $Treat(X, M)$, exponentiate result for M

E	$S(X)$	$Tt(X,M)$	ϕ_3
$false$	$false$	$false$	9
$false$	$false$	$true$	1
$false$	$true$	$false$	5
$false$	$true$	$true$	6
$true$	$false$	$false$	3
$true$	$false$	$true$	4
$true$	$true$	$false$	4
$true$	$true$	$true$	5

E	$S(X)$	ϕ_3'
$false$	$false$	$(9+1)^2 = 100$
$false$	$true$	$(5+6)^2 = 121$
$true$	$false$	$(3+4)^2 = 49$
$true$	$true$	$(4+5)^2 = 81$



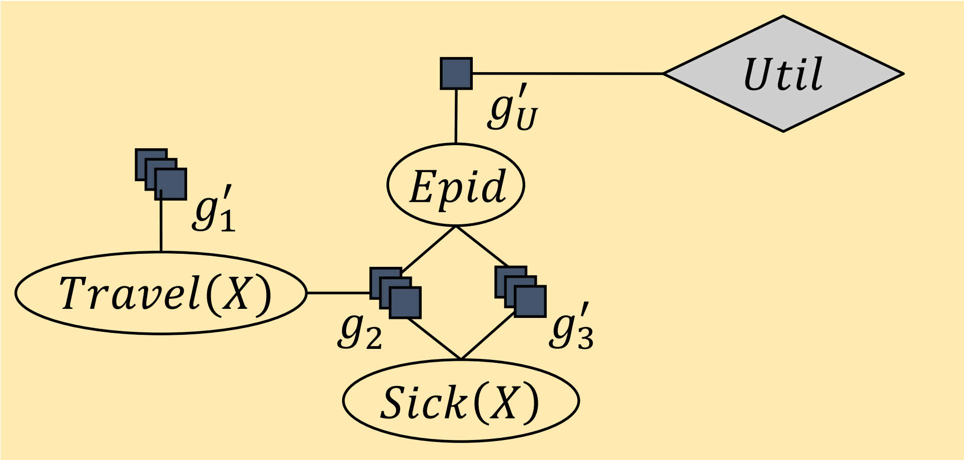
PDecMs: EU Query – Example

- Calculations with $|\mathcal{D}(M)| = 2, |\mathcal{D}(X)| = 3$:
  - Multiply g_1', g_2, sum out $Travel(X)$

E	$S(X)$	$Tl(X)$	$\phi_2 \cdot \phi_1'$
$false$	$false$	$false$	$10 \cdot 1 = 10$
$false$	$false$	$true$	$9 \cdot 0 = 0$
$false$	$true$	$false$	$4 \cdot 1 = 4$
$false$	$true$	$true$	$2 \cdot 0 = 0$
$true$	$false$	$false$	$8 \cdot 1 = 8$
$true$	$false$	$true$	$3 \cdot 0 = 0$
$true$	$true$	$false$	$5 \cdot 1 = 5$
$true$	$true$	$true$	$1 \cdot 0 = 0$

E	$S(X)$	ϕ_{12}'
$false$	$false$	$10 + 0 = 10$
$false$	$true$	$4 + 0 = 4$
$true$	$false$	$8 + 0 = 8$
$true$	$true$	$5 + 0 = 5$

$Travel(X)$	ϕ_1'
$false$	1
$true$	0

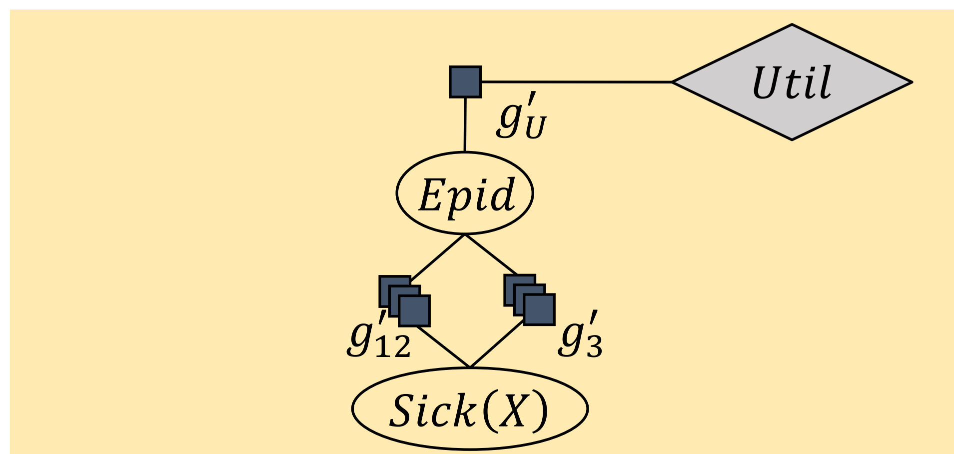


PDecMs: EU Query – Example

- Calculations with $|\mathcal{D}(M)| = 2, |\mathcal{D}(X)| = 3$:
  - Multiply g'_{12}, g'_3, sum out $Sick(X)$, exponentiate for X, normalise

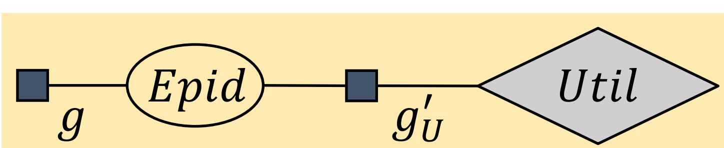
E	$S(X)$	$\phi'_{12} \cdot \phi'_3$
$false$	$false$	$10 \cdot 100 = 1000$
$false$	$true$	$4 \cdot 121 = 484$
$true$	$false$	$8 \cdot 49 = 392$
$true$	$true$	$5 \cdot 81 = 405$

E	ϕ'
$false$	$(1000 + 484)^3 = 3{,}268{,}147{,}904$
$true$	$(392 + 405)^3 = 506{,}261{,}573$



Result after normalising: $g = \phi(Epid)$

$Epid$	ϕ
$false$	0.87
$true$	0.13



- Result $\phi(Epid)$ for $P(Epid = e|\boldsymbol{a}_1)$ in G
- Expected utility of $\boldsymbol{a}_1 = \{ban\}$ in $G = \{g_1, g_2, g_3, g_U\}$

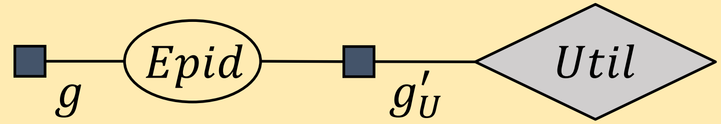
$$eu(\boldsymbol{a}_1)$$

$$= \sum_{e \in \mathcal{R}(Epid)} P(Epid = e|\boldsymbol{a}_1) \cdot \phi'_U(Epid = e)$$

$$= \sum_{e \in \mathcal{R}(Epid)} \phi(Epid = e) \cdot \phi'_U(Epid = e)$$

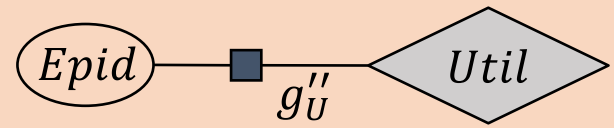
$$= \sum_{e \in \mathcal{R}(Epid)} \phi''_U(Epid = e)$$

$$= \phi'''_U(.)$$

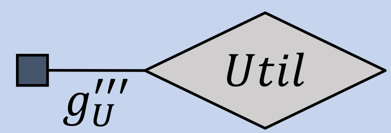


$Epid$	ϕ
$false$	0.87
$true$	0.13

$Epid$	$Util$
$false$	-20
$true$	5



$Epid$	$Util$
$false$	$0.87 \cdot (-20) = -17.4$
$true$	$0.13 \cdot 5 = 0.65$



$.$	$Util$
$.$	$-17.4 + 0.65 = -16.75$

Answering EU-Queries (with LVE)

- Given a PDecM $G = \{g_i\}_{i=1}^{n} \cup \{g_U\}$, evidence $\boldsymbol{E}$, and an action assignment $\boldsymbol{a}$ (*)
  - Absorb $\boldsymbol{E}$ in G and set $\boldsymbol{a}$ in G
  - Calculate the posterior, $P(\boldsymbol{R}|\boldsymbol{E},\boldsymbol{a})$, of the *Markov blanket of the utility node*
    - I.e., $\boldsymbol{R} = rv(g_U) \setminus rv(\boldsymbol{a}) \setminus rv(\boldsymbol{E})$ (remaining PRVs in g_U after previous step)
    - Using LVE: With $\boldsymbol{R}$ as the query terms, eliminate all non-query terms in G, i.e., call $\mathrm{LVE}(G \setminus \{g_U\}, \boldsymbol{R}, \emptyset)$
      - Evidence already absorbed, decisions made → $\boldsymbol{E} = \emptyset$ in the call
  - Calculate the expected utility by summing over the range values of $\boldsymbol{R}$: $eu(\boldsymbol{E},\boldsymbol{a}) = \sum_{\boldsymbol{r}\in\mathcal{R}(\boldsymbol{R})} P(\boldsymbol{r}|\boldsymbol{E},\boldsymbol{a}) \cdot \phi_U(\boldsymbol{r})$
    - Using LVE: Eliminate remaining PRVs in G,
      - Result: parfactor mapping empty argument to a single value (U)

(*) We need to talk about evidence and action assignments later.

PDecMs: MEU Problem

- Given a PDecM G and evidence $\boldsymbol{E}$
- Maximum Expected Utility (MEU) problem
  - Find the action assignment that yields the highest expected utility in G
  - Formally,

$$\operatorname{meu}(G|\boldsymbol{E}) = \left(\boldsymbol{a}^*, eu(\boldsymbol{E}, \boldsymbol{a}^*)\right)$$

$$\boldsymbol{a}^* = \underset{\boldsymbol{a} \in \mathcal{R}(\boldsymbol{A})}{\operatorname{argmax}}\, eu(\boldsymbol{E}, \boldsymbol{a})$$

Additive semantics with inner sum and outer max: Sum up utilities, then pick maximum
→ Max-sum algorithms

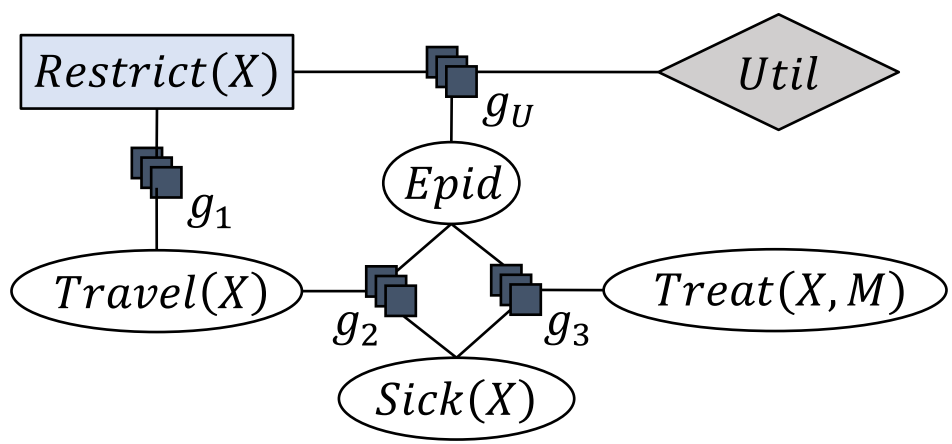
  - For an exact solution, $\operatorname{meu}(G|\boldsymbol{E})$ requires an algorithm to go through all $\boldsymbol{a} \in \mathcal{R}(\boldsymbol{A})$
    - Size of $\mathcal{R}(\boldsymbol{A})$ exponential in $|\boldsymbol{A}|$

Alternative specification

$$\operatorname{meu}(G|E) = \left(\underset{\boldsymbol{a} \in \mathcal{R}(\boldsymbol{A})}{\operatorname{argmax}}\, eu(\boldsymbol{E}, \boldsymbol{a}), \max_{\boldsymbol{a} \in \mathcal{R}(\boldsymbol{A})} eu(\boldsymbol{E}, \boldsymbol{a})\right)$$

PDecMs: MEU – Example

- Problem instance with $G = \{g_1, g_2, g_3, g_U\}, \boldsymbol{E} = \emptyset$:
  $\mathrm{meu}(G) = \left(\boldsymbol{a}^*, eu(\boldsymbol{a}^*)\right) \quad \boldsymbol{a}^* = \operatorname{argmax}_{\boldsymbol{a} \in \{\boldsymbol{a}_1, \boldsymbol{a}_2\}} eu(\boldsymbol{a})$
  - $\boldsymbol{a}_1 = \{ban\}, \boldsymbol{a}_2 = \{free\}$
  - Expected utility of $\boldsymbol{a}_1 = \{ban\}$: $eu(\boldsymbol{a}_1) = -16.75$
  - Expected utility of $\boldsymbol{a}_2 = \{free\}$: $eu(\boldsymbol{a}_2) = 8.8$
- Solution
  - $\boldsymbol{a}^* = \operatorname{argmax}_{\boldsymbol{a} \in \{\boldsymbol{a}_1, \boldsymbol{a}_2\}} eu(\boldsymbol{a}) = \boldsymbol{a}_2$
  - $\mathrm{meu}(G) = (\boldsymbol{a}_2, 8.8)$
  - Decision that leads to maximum EU: No travel restrictions



Lifted MEU

$$\text{meu}(G|\boldsymbol{E}) = \left(\boldsymbol{a}^*, eu(\boldsymbol{E}, \boldsymbol{a}^*)\right)$$
$$\boldsymbol{a}^* = \underset{\boldsymbol{a} \in \mathcal{R}(\boldsymbol{A})}{\text{argmax}}\, eu(\boldsymbol{E}, \boldsymbol{a})$$

- In terms of semantics, $\boldsymbol{a} \in \mathcal{R}(\boldsymbol{A})$ means
  - Grounding $\boldsymbol{A}$ and going through all possible combinations of assignments to $gr(\boldsymbol{A})$
- But: grounding is a bad idea
  - Combinatorial explosion: number of action assignments to test exponential in size of $gr(\boldsymbol{A})$
  - Grounds any parfactor in G containing a logvar of $\boldsymbol{A}$
- Also: Grounding to full extent often unnecessary
  - Within groups of indistinguishable constants, the same decision will lead to its maximum influence in the MEU solution
    - Only need to test each assignment for complete group
- Therefore: Test out all possible combinations of assignments w.r.t. the groups occurring in G
  - No longer exponential in the size of $gr(\boldsymbol{A})$!

Lifted MEU: Groups

$$\text{meu}(G|\boldsymbol{E}) = \left(\boldsymbol{a}^*, eu(\boldsymbol{E}, \boldsymbol{a}^*)\right)$$
$$\boldsymbol{a}^* = \underset{\boldsymbol{a} \in \mathcal{R}(\boldsymbol{A})}{\text{argmax}}\, eu(\boldsymbol{E}, \boldsymbol{a})$$

- In parameterised models without evidence (or evidence for complete domains), $\boldsymbol{a} \in \mathcal{R}(\boldsymbol{A})$ means
  - Going through all possible combinations of assignments to $\boldsymbol{A}$
    - One group per logvar
- In parameterised models with evidence affecting parfactors containing decision PRVs, $\boldsymbol{a} \in \mathcal{R}(\boldsymbol{A})$ means
  - Going through all possible combinations of assignments for each group resulting after evidence handling
    - Specifically, after shattering
- Effect: size exponential in number of groups

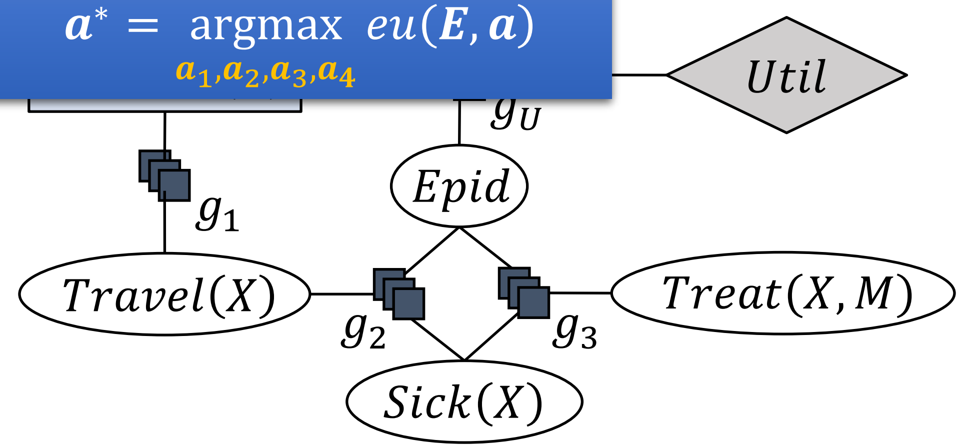
(*) Now is later.

Lifted MEU: Groups – Example

- PDecM $G = \{g_1, g_2, g_3, g_U\}$
  - ⊤ constraints
- Ev. $\boldsymbol{E} = \{Sick(X') = true\}$
  - $\mathcal{D}(X') = \{x_1, \ldots, x_{10}\}$
- Action $Restrict(X)$ with range $\{ban, free\}$
  - $\mathcal{D}(X) = \{x_1, \ldots, x_n\}$
- Overlap in domain/constraint
  - Shattering of G
    - Duplicates all parfactors for
      - $\mathcal{D}(X') = \{x_1, \ldots, x_{10}\}$
      - $\mathcal{D}(X'') = \{x_{10}, \ldots, x_n\}$
    - Could also restrict constraints

- Action assignments
  - $\boldsymbol{a}_1 = \{Restrict(X'') = ban,\ Restrict(X') = ban\}$
  - $\boldsymbol{a}_2 = \{Restrict(X'') = ban,\ Restrict(X') = free\}$
  - $\boldsymbol{a}_3 = \{Restrict(X'') = free,\ Restrict(X') = ban\}$
  - $\boldsymbol{a}_4 = \{Restrict(X'') = free,\ Restrict(X') = free\}$

$$\mathrm{meu}(G|\boldsymbol{E}) = \left(\boldsymbol{a}^*, eu(\boldsymbol{E}, \boldsymbol{a}^*)\right)$$
$$\boldsymbol{a}^* = \underset{\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3, \boldsymbol{a}_4}{\operatorname{argmax}}\ eu(\boldsymbol{E}, \boldsymbol{a})$$



Answering EU-Queries for MEU

- Given a PDecM $G = \{g_i\}_{i=1}^{n} \cup \{g_U\}$, evidence $\boldsymbol{E}$, and an action assignment $\boldsymbol{a}$ for groups in G after shattering
  - Absorb $\boldsymbol{E}$ in G and set $\boldsymbol{a}$ in G
  - Calculate the posterior, $P(\boldsymbol{R}|\boldsymbol{E}, \boldsymbol{a})$, of the *Markov blanket of the utility node*
    - I.e., $\boldsymbol{R} = rv(g_U) \setminus rv(\boldsymbol{a}) \setminus rv(\boldsymbol{E})$ (remaining PRVs in g_u's after previous step)
    - Using LVE: With $\boldsymbol{R}$ as the query terms, eliminate all non-query terms in G, i.e., call $\mathrm{LVE}(G \setminus \{g_U\}, \boldsymbol{R}, \emptyset)$
      - Evidence already absorbed, decisions made → $\boldsymbol{E} = \emptyset$ in the call
  - Calculate the expected utility by summing over the range values of $\boldsymbol{R}$: $eu(\boldsymbol{E}, \boldsymbol{a}) = \sum_{\boldsymbol{r} \in \mathcal{R}(\boldsymbol{R})} P(\boldsymbol{r}|\boldsymbol{E}, \boldsymbol{a}) \cdot \phi_U(\boldsymbol{r})$
    - Using LVE: Eliminate remaining PRVs in G,
      - Result: parfactor mapping empty argument to a single value (U)

(*) Now is later.

LVE for MEU Problems

function $\mathbf{MEU{-}LVE}(G = \{g_i\}_{i=1}^{n} \cup \{g_U\}, \boldsymbol{E})$
  Absorb $\boldsymbol{E}$ in G
  $\boldsymbol{a}^* \leftarrow (\,)$
  $eu_{max} \leftarrow -\infty$
  for each action assignment $\boldsymbol{a}$ in G **do**
    Set $\boldsymbol{a}$ in G
    $g \leftarrow \mathrm{LVE}(G \setminus \{g_U\}, rv(g_U), \emptyset)$ ▹ g normalised
    $eu(\boldsymbol{E}, \boldsymbol{a}) \leftarrow \mathrm{LVE}(\{g_U, g\}, \emptyset, \emptyset)$
    if $eu(\boldsymbol{E}, \boldsymbol{a}) > eu_{max}$ **then**
      $\boldsymbol{a}^* \leftarrow \boldsymbol{a}$
      $eu_{max} \leftarrow eu(\boldsymbol{E}, \boldsymbol{a})$
  return $\boldsymbol{a}^*$

LVE-MEU

- Modify to save all assignments that lie within ε-margin

(Sets of) Utilities and Logvars

Let us consider four cases:

1. Set of utility parfactors mapping to the same utility randvar
   - $G = \{g_i\}_{i=1}^{n} \cup \{g_u\}_{u=1}^{m}$
   - Each g_u mapping to the same utility randvar U
2. Logvars in the Markov blanket of the utility randvar
   - Possible given a single utility parfactor mapping to a utility randvar
     - In example so far logvars only in decision PRV, which is always set
   - Grounding the parfactor leads to a set of utility factors mapping to the same utility randvar U
3. Logvars in the utility PRV
   - A set of utility randvars in the ground case, but with indistinguishable behaviour from the model side
4. A set of utility PRVs

1. Set of Utility Parfactors

- Set of utility parfactors, all map to a utility randvar U
  - $G = \{g_i\}_{i=1}^{n} \cup \{g_u\}_{u=1}^{m}$, each g_u mapping to U
    - Refer to set of utility parfactors of G by $G_U = \{g_u\}_{u=1}^{m}$
  - Since all g_u map to U, g_u's are partial descriptions of an overall utility function g_U
- Can combine G_U into one utility parfactor g_U

$$G = \{g_i\}_{i=1}^{n} \cup \{g_u\}_{u=1}^{m} = \{g_i\}_{i=1}^{n} \cup \{g_U\}$$

  - where

$$g_U = \sum_{\boldsymbol{r} \in \mathcal{R}\left(rv\left(\{g_u\}_{u=1}^{m}\right)\right)} \sum_{u=1}^{m} \phi_u\left(\pi_{rv(g_u)}(\boldsymbol{r})\right)$$

    - Additive semantics: Sum over the utilities that $\boldsymbol{r}$ maps to in the different g_u (join of arguments, addition of utilities)

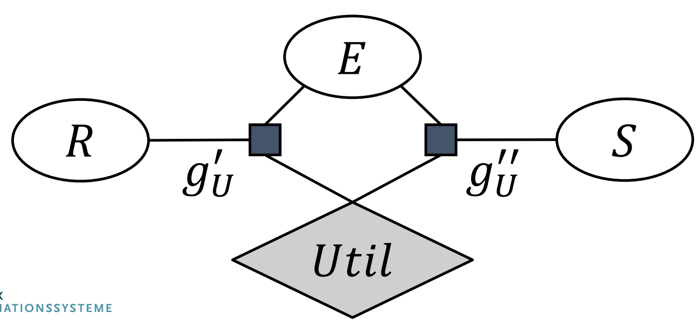
Compare combining parfactors with potentials (multiplicative semantics): Join of arguments, multiplication of potentials

1. Set of Utility Parfactors: Example

- Two utility factors mapping to utility randvar $Util$
  - $\phi'_{Util}(E, R)$
  - $\phi''_{Util}(E, S)$

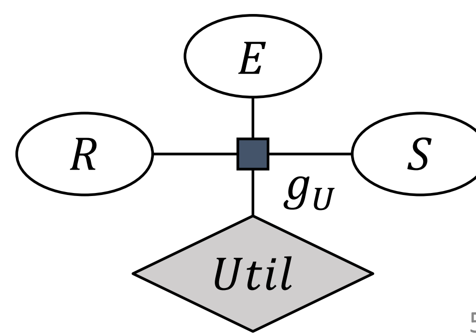
E	R	ϕ'
$false$	$false$	5
$false$	$true$	0
$true$	$false$	-5
$true$	$true$	-10

E	S	ϕ''
$false$	$false$	8
$false$	$true$	3
$true$	$false$	0
$true$	$true$	1



- Resulting g_{Util}

E	R	S	ϕ
$false$	$false$	$false$	$5 + 8$
$false$	$false$	$true$	$5 + 3$
$false$	$true$	$false$	$0 + 8$
$false$	$true$	$true$	$0 + 3$
$true$	$false$	$false$	$-5 + 0$
$true$	$false$	$true$	$-5 + 1$
$true$	$true$	$false$	$-10 - 0$
$true$	$true$	$true$	$-10 + 1$



1. Set of Utility Parfactors

- Combination into one utility parfactor g_U in terms of semantics yields for EU queries:

$$eu(\boldsymbol{E}, \boldsymbol{a}) = \sum_{\boldsymbol{r} \in \mathcal{R}\left(rv(\{g_u\}_{u=1}^m)\right)} P(\boldsymbol{r} | \boldsymbol{E}, \boldsymbol{a}) \cdot \sum_{u=1}^{m} \phi_u \left(\pi_{rv(g_u)}(\boldsymbol{r}) \right)$$

  - Caution: Expected utility requires marginal distribution over the Markov blanket of U
    - Even though utility function specified over a set of parfactors, which may have only few arguments, query for $P(\boldsymbol{r}|\boldsymbol{E}, \boldsymbol{a})$ will combine all PRVs in $\{g_u\}_{u=1}^m$ into one parfactor g
  - Changes in LVE: $\boldsymbol{R} = rv(\{g_u\}_{u=1}^m) \setminus rv(\boldsymbol{a}) \setminus rv(\boldsymbol{E})$
  - Changes in MEU-LVE:

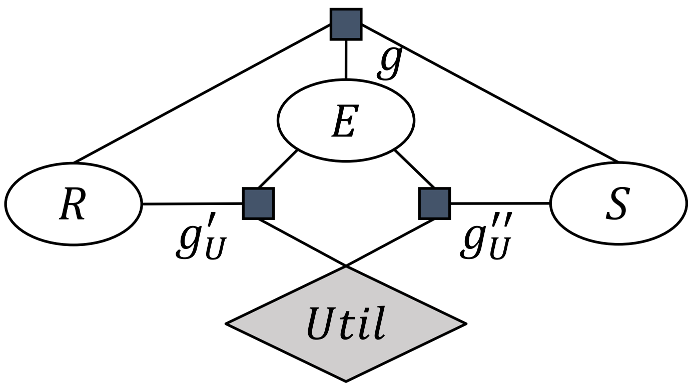
$g \leftarrow \text{LVE}(G \setminus \{g_u\}_{u=1}^m, rv(\{g_u\}_{u=1}^m), \emptyset)$ ▹ g normalised
$eu(\boldsymbol{E}, \boldsymbol{a}) \leftarrow \text{LVE}(\{g_u\}_{u=1}^m \cup \{g\}, \emptyset, \emptyset)$

1. Set of Utility Parfactors: EU

- Two utility factors g'_U, g''_U

E	R	ϕ'
$false$	$false$	5
$false$	$true$	0
$true$	$false$	-5
$true$	$true$	-10

E	S	ϕ''
$false$	$false$	8
$false$	$true$	3
$true$	$false$	0
$true$	$true$	1



- LVE result g of $P(E, R, S)$

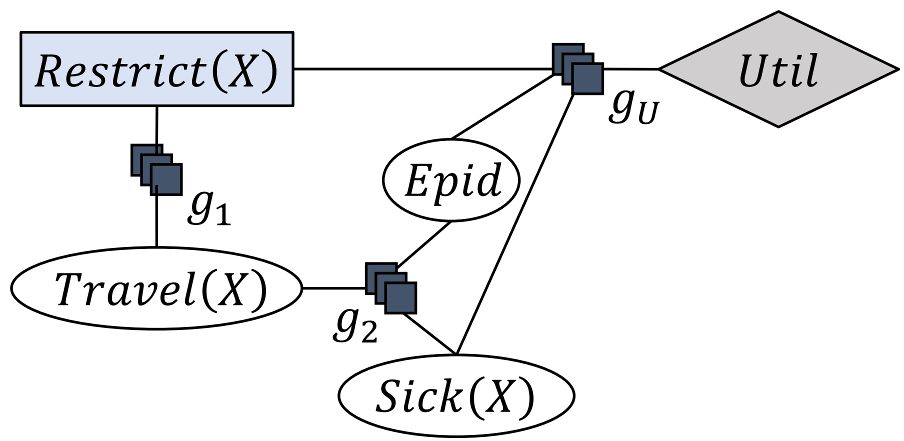
E	R	S	ϕ
$false$	$false$	$false$	0.01
$false$	$false$	$true$	0.02
$false$	$true$	$false$	0.03
$false$	$true$	$true$	0.04
$true$	$false$	$false$	0.10
$true$	$false$	$true$	0.20
$true$	$true$	$false$	0.25
$true$	$true$	$true$	0.35

To get EU, multiply, e.g., g''_U into g and sum out S, then multiply g'_U into the result and sum out R and E.

Not computing g_U may reduce the size of some multiplications but does not reduce overall space requirements.

2. Logvars in the Markov Blanket

- Logvars in Markov blanket of utility randvar
  - I.e., in arguments of utility parfactors that are not decision PRVs
  - E.g., $g_U = \phi_{Util}(Restrict(X), Epid, Sick(X))$
  - Lifted version of the previous case
    - Grounding leads to a set of utility parfactors mapping to the same utility randvar
- In terms of calculations, $P(\boldsymbol{r}|\boldsymbol{E}, \boldsymbol{a})$ now contains parameterised query terms in EU query
  - Effect: logvars counted (or grounded) in result



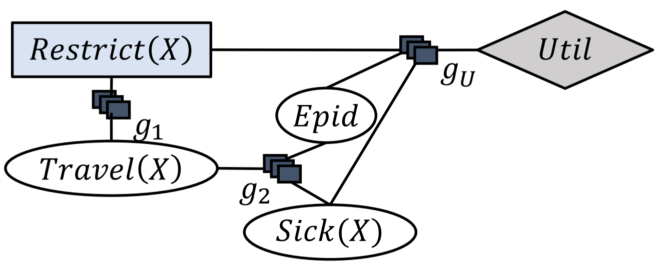
To get a distribution over all constants represented by logvars, counting (or, if counting not possible, grounding) unavoidable.

2. Logvars in the Markov Blanket

- Computing $P(Epid, Sick(X)|\boldsymbol{a})$ in example model
  - Assume elimination result as depicted on the right above
  - To normalise, take into account that $|\mathcal{D}(X)| = 3$, i.e.,
    - Count X
    - Normalise by including $Mul(h)$ in Z

E	$S(X)$	ϕ'_{12}
$false$	$false$	10
$false$	$true$	4
$true$	$false$	8
$true$	$true$	5

E	$\#_X[S(X)]$	$\phi^\#$	ϕ
$false$	[0,3]	$4^0 \cdot 10^3 = 1000$	0.202
$false$	[1,2]	$4^1 \cdot 10^2 = 400$	0.081
$false$	[2,1]	$4^2 \cdot 10^1 = 160$	0.032
$false$	[3,0]	$4^3 \cdot 10^0 = 64$	0.013
$true$	[0,3]	$5^0 \cdot 8^3 = 512$	0.104
$true$	[1,2]	$5^1 \cdot 8^2 = 320$	0.065
$true$	[2,1]	$5^2 \cdot 8^1 = 200$	0.040
$true$	[3,0]	$5^3 \cdot 8^0 = 125$	0.025

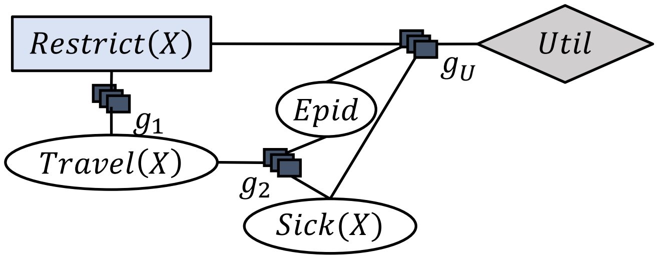


2. Logvars in the Markov Blanket

- Possible mismatch in representation of logvars between result of $P(\boldsymbol{r}|\boldsymbol{E}, \boldsymbol{a})$ and utility parfactors
  - Utility parfactor g_U under $\boldsymbol{a}$: $\phi_{Util}(Epid, Sick(X))$
  - Result of $P(Epid, Sick(X)|\boldsymbol{a})$ given by $\phi(Epid, \#_X[Sick(X)])$
  - X counted in elimination result but uncounted in utility parfactor

E	$S(X)$	ϕ_{Util}
$false$	$false$	5
$false$	$true$	0
$true$	$false$	-5
$true$	$true$	-10

E	$\#_X[S(X)]$	ϕ
$false$	[0,3]	0.202
$false$	[1,2]	0.081
$false$	[2,1]	0.032
$false$	[3,0]	0.013
$true$	[0,3]	0.104
$true$	[1,2]	0.065
$true$	[2,1]	0.040
$true$	[3,0]	0.025



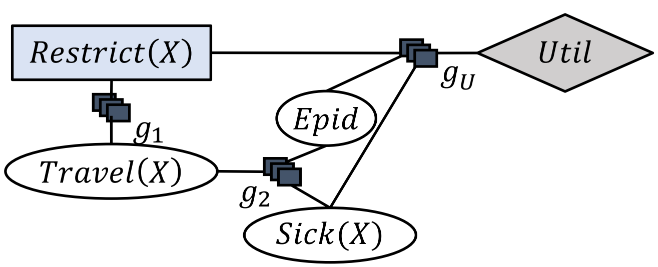
2. Logvars in the Markov Blanket

- Ensure logvars occur in same form
  - Count or ground logvars in g_u's to match description in result of $P(\boldsymbol{r}|\boldsymbol{E},\boldsymbol{a})$
    - If logvar not countable in a g_u, grounding of logvar in result necessary
- In example,
  - X counted in elimination result but uncounted in utility parfactor

  → Count logvar in g_U
    - Fulfils counting preconditions

E	$S(X)$	ϕ_{Util}
$false$	$false$	5
$false$	$true$	0
$true$	$false$	-5
$true$	$true$	-10

E	$\#_X[S(X)]$	ϕ
$false$	[0,3]	0.202
$false$	[1,2]	0.081
$false$	[2,1]	0.032
$false$	[3,0]	0.013
$true$	[0,3]	0.104
$true$	[1,2]	0.065
$true$	[2,1]	0.040
$true$	[3,0]	0.025



But: Counting with utilities and additive semantics!

2. Logvars in the Markov Blanket

- Count conversion with utilities
  - Additive semantics: Add up utilities
    - Instead of multiplicate semantics: Multiply, leading to exponentiation

E	$S(X)$	ϕ_{Util}
$false$	$false$	5
$false$	$true$	0
$true$	$false$	-5
$true$	$true$	-10

- Given histogram $[n_1, .., n_k]$ and utilities $u_1, .., u_k$
  - New utility: $n_1 \cdot u_1 + \ldots + n_k \cdot u_k$
    - Instead of: $p_1^{n_1} \cdot \cdots \cdot p_k^{n_k}$
  - Formally,

$$\phi'_U(\ldots, a_{i-1}, h, a_{i+1}, \ldots) = \sum_{a_i \in \mathcal{R}(A_i)} h(a_i) \cdot \phi_U(\ldots, a_{i-1}, a_i, a_{i+1}, \ldots)$$

- Compare with grounding and combining resulting set of utility factors

Counting with Utilities

E	$S(X)$	ϕ'_{12}
$false$	$false$	10
$false$	$true$	4
$true$	$false$	8
$true$	$true$	5

$p_1^{n_1} \cdot \ldots \cdot p_k^{n_k}$
vs.
$n_1 \cdot u_1 + \ldots + n_k \cdot u_k$

E	$S(X)$	ϕ_{Util}
$false$	$false$	5
$false$	$true$	0
$true$	$false$	-5
$true$	$true$	-10

E	$\#_X[S(X)]$	$\phi^\#$
$false$	$[0,3]$	$4^0 \cdot 10^3 = 1000$
$false$	$[1,2]$	$4^1 \cdot 10^2 = 400$
$false$	$[2,1]$	$4^2 \cdot 10^1 = 160$
$false$	$[3,0]$	$4^3 \cdot 10^0 = 64$
$true$	$[0,3]$	$5^0 \cdot 8^3 = 512$
$true$	$[1,2]$	$5^1 \cdot 8^2 = 320$
$true$	$[2,1]$	$5^2 \cdot 8^1 = 200$
$true$	$[3,0]$	$5^3 \cdot 8^0 = 125$

E	$\#_X[S(X)]$	ϕ'_{Util}
$false$	$[0,3]$	$0 \cdot 0 + 3 \cdot 5 = 15$
$false$	$[1,2]$	$1 \cdot 0 + 2 \cdot 5 = 10$
$false$	$[2,1]$	$2 \cdot 0 + 1 \cdot 5 = 5$
$false$	$[3,0]$	$3 \cdot 0 + 0 \cdot 5 = 0$
$true$	$[0,3]$	$0 \cdot -10 + 3 \cdot -5 = -15$
$true$	$[1,2]$	$1 \cdot -10 + 2 \cdot -5 = -20$
$true$	$[2,1]$	$2 \cdot -10 + 1 \cdot -5 = -25$
$true$	$[3,0]$	$3 \cdot -10 + 0 \cdot -5 = -30$

2. Logvars in the Markov Blanket

- With identical forms for all logvars, proceed with EU
  - In terms of MEU-LVE, proceed as before but when calling LVE for computing $eu(\boldsymbol{E}, \boldsymbol{a})$ utility-count conversion needed

$g \leftarrow \mathrm{LVE}(G \setminus \{g_u\}_{u=1}^{m}, rv(\{g_u\}_{u=1}^{m}), \emptyset)$ ▹ g normalised
$eu(\boldsymbol{E}, \boldsymbol{a}) \leftarrow \mathrm{LVE}(\{g_u\}_{u=1}^{m} \cup \{g\}, \emptyset, \emptyset)$

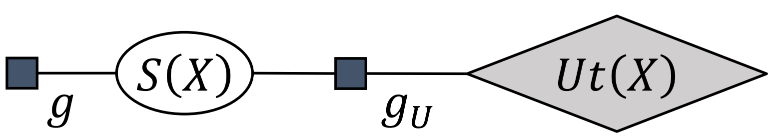
E	$\#_X[S(X)]$	ϕ'_{Util}	ϕ	$\phi'_{Util} \cdot \phi$	$Mul(h)$	$eu(\boldsymbol{a}) =$
$false$	$[0,3]$	15	0.202	3.03	1	$3.03 \cdot 1$
$false$	$[1,2]$	10	0.081	0.81	3	$+ 0.81 \cdot 3$
$false$	$[2,1]$	5	0.032	0.16	3	$+ 0.16 \cdot 3$
$false$	$[3,0]$	0	0.013	0	1	$+ 0.00 \cdot 1$
$true$	$[0,3]$	-15	0.104	-1.56	1	$- 1.56 \cdot 1$
$true$	$[1,2]$	-20	0.065	-1.3	3	$- 1.30 \cdot 3$
$true$	$[2,1]$	-25	0.040	-1	3	$- 1.00 \cdot 3$
$true$	$[3,0]$	-30	0.025	-0.75	1	$- 0.75 \cdot 1$
						$= -3.27$

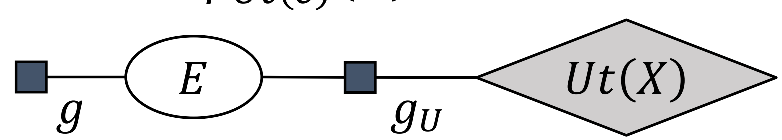
3. Logvars in Utility PRVs

- Logvars in a utility PRV: $U(X_1, \ldots, X_n)$
  - No longer only a propositional randvar

    $\rightarrow$ In terms of semantics: *set of propositional utility randvars* $gr\big(U(X_1, \ldots, X_n)\big)$
    - Compare multi-attribute utility theory
- Generalisation: set of utility PRVs

- E.g., utility PRV $Ut(X)$
  - Ground: $Ut(a), Ut(b), Ut(c)$
  - Utility parfactor: $g_U = \phi_{Ut(X)}\big(S(X)\big)$
    - $\phi_{Ut(a)}\big(S(a)\big)$
    - $\phi_{Ut(b)}\big(S(b)\big)$
    - $\phi_{Ut(c)}\big(S(c)\big)$
  - Utility parfactor: $g_U = \phi_{Ut(X)}(E)$
    - $\phi_{Ut(a)}(E)$
    - $\phi_{Ut(b)}(E)$
    - $\phi_{Ut(c)}(E)$





4. Set of Utility PRVs

- Given a PDecM $G = \{g_i\}_{i=1}^{n} \cup \{g_u\}_{u=1}^{m}$
  - Utility parfactors $g_u = \phi_{U_u}(\mathcal{A})_{|C}$ map argument values to utility PRV U_u
    - Different g_u may map to the same utility PRV
    - G contains a set of utility PRVs, referred to as $rv_U(G)$
      - Do not confuse with $rv(G_U)$, the set of PRVs in $G_U = \{g_u\}_{u=1}^{m}$
  - Requires a specification of how to combine $rv_U(G)$ into one utility value U: combination function $\varphi_U\big(rv_U(G)\big)$
    - If considered MPI/mutually UI: specify an additive/ multiplicative function as seen earlier
      - Simplest case: additive function (as seen during Case 1)

$$\varphi_U\big(rv_U(G)\big) = \sum_{\boldsymbol{r}\in\mathcal{R}\left(rv(G_u)\right)} \sum_{g_u \in G_U} \phi_{U_u}\left(\pi_{rv(g_u)}(\boldsymbol{r})\right)$$

      - Approximation: Assume independence
    - Specify/learn a more complex function

4. Set of Utility PRVs: φ_U

- Given a PDecM $G = \{g_i\}_{i=1}^{n} \cup \{g_u\}_{u=1}^{m}$ and a combination function φ_U
  - Query for an expected utility (EU)
    - What is the expected utility of decisions $\boldsymbol{a}$ in G?

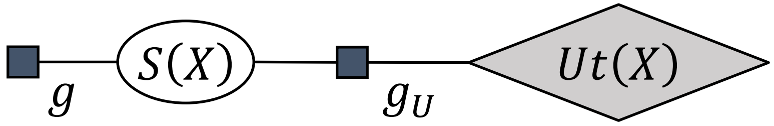
$$eu(\boldsymbol{E}, \boldsymbol{a}) = \sum_{\boldsymbol{r} \in \mathcal{R}(rv(G_U))} P(\boldsymbol{r} | \boldsymbol{E}, \boldsymbol{a}) \cdot \varphi_U\left(\pi_{rv_U(G)}(\boldsymbol{r})\right)$$

$$= \sum_{\boldsymbol{r} \in \mathcal{R}(rv(G_U))} P(\boldsymbol{r} | \boldsymbol{E}, \boldsymbol{a}) \cdot \sum_{u=1}^{m} \phi_u\left(\pi_{rv(g_u)}(\boldsymbol{r})\right)$$

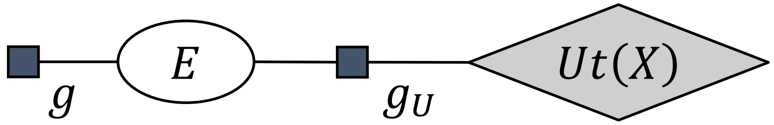
  - Multiplication with $P(\boldsymbol{r} | \boldsymbol{E}, \boldsymbol{a})$ brings about the same problems as discussed previously (combines all $\mathcal{R}(rv(G_U))$ into one parfactor, $\mathcal{R}(rv(G_U))$ might be parameterised)

3. Logvars in Utility PRVs Treated as 4.

- $g_U = \phi_{Ut(X)}(S(X))$
- Result of $P(S(X)|\boldsymbol{a})$
  - $\phi(\#_X[S(X)])$
- Count X in g_U
- Multiply $\phi(\#_X[S(X)])$ and $\phi_{Ut(X)}(\#_X[S(X)])$
- Add up all entries



- $g_U = \phi_{Util(X)}(E)$
- Result of $P(E|\boldsymbol{a})$
  - $\phi(E)$
- Multiply $\phi(E)$ and $\phi_{Ut(X)}(S(X))$
- Add up all entries
- Multiply with $|\mathcal{D}(X)|$
  - Given ⊤ constraints



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

4. Set of Utility PRVs: Approximate $\boldsymbol{U}$

- Approximation
  - Move $P(\boldsymbol{r}|\boldsymbol{E},\boldsymbol{a})$ into the inner sum:

$$eu(\boldsymbol{E},\boldsymbol{a}) \approx \sum_{\boldsymbol{r}\in\mathcal{R}(rv(G_U))} \sum_{U_u \in rv_U(G)} P\left(\pi_{rv(g_{U_u})}(\boldsymbol{r})|\boldsymbol{E},\boldsymbol{a}\right) \cdot \phi_{U_u}\left(\pi_{rv(U_u)}(\boldsymbol{r})\right) \quad (1)$$

$$\approx \sum_{\boldsymbol{r}\in\mathcal{R}(rv(G_U))} \sum_{u=1}^{m} P\left(\pi_{rv(g_u)}(\boldsymbol{r})|\boldsymbol{E},\boldsymbol{a}\right) \cdot \phi_u\left(\pi_{rv(g_u)}(\boldsymbol{r})\right) \quad (2)$$

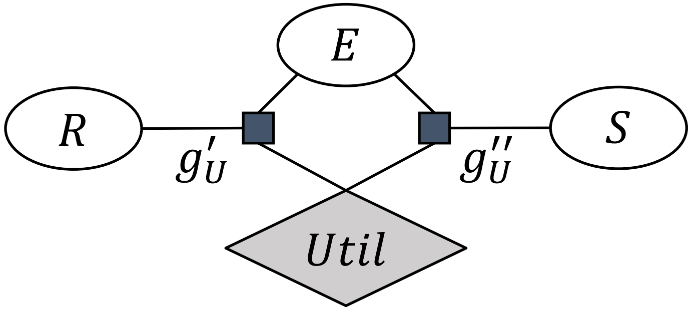
- Instead of asking one query with all $rv(G_U)$ as query terms
  1. Either ask $|rv_U(G)|$ queries with $rv(g_{U_u})$ as query terms where $g_{U_u} = \phi_{U_u}\left(\pi_{rv(U_u)}(\boldsymbol{r})\right)$ refers to the combination of all utility parfactors that map to utility PRV U_u
  2. Or ask m queries with $rv(g_u)$ as query terms

What's the difference?

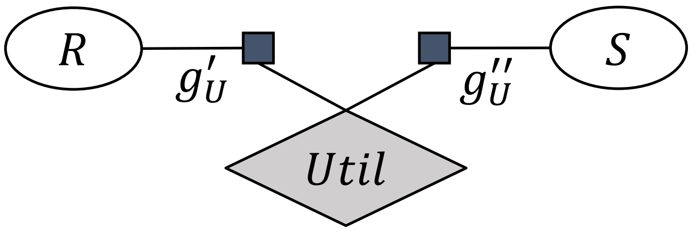
Can be exact if independence given between sets of query terms
(Compare Boyen-Koller algorithm for sequential inference)

4. Set of Utility PRVs: Approximate U

- Two parfactors mapping to same utility PRV, dependent
  - Markov blanket of $Util$: E, R, S
    - $P(E, R, S)$ exact
    - $P(E, R), P(E, S)$ approximate



- Two parfactors mapping to same utility PRV, independent
  - Markov blanket of $Util$: R, S
    - $P(R, S)$ exact
    - $P(R), P(S)$ also exact as R, S independent



- Same holds if logvars occur in PRVs

MEU-LJT

- Answer MEU query, probability and assignment queries
  - Assuming an *additive* combination function
  - Build an FO jtree for PDecM
  - Can add a helper parfactor g over the union of Markov blankets of the utility PRVs for computing $P(\boldsymbol{r}|\boldsymbol{E},\boldsymbol{a})$
    - Without g, use subgraph that covers $\boldsymbol{r}$ for each $P(\boldsymbol{r}|\boldsymbol{E},\boldsymbol{a})$
      - Parclusters may be smaller → better for other queries
    - With g, $\boldsymbol{r}$ contained in one parcluster → easier for $P(\boldsymbol{r}|\boldsymbol{E},\boldsymbol{a})$
      - But parclusters may be larger overall → worse for other queries
    - If using one of the approximations: either g may be over the Markov blanket of each utility PRV or no g necessary
      - To compute overall EU query, need to ask set of EU queries either for each of the utility PRVs or for each utility PRV in each utility parfactor and *add* results up

Claims about FO jtree construction

1. Ignore decision PRVs as they are always set before any calculations occur
2. Ignore utility parfactors as they do not carry information relevant for messages

Some References

- MEU in PDecMs
  - Warning: not as detailed as in these slides

Version using an early version of LVE, mashing early parfactor graphs and MLNs:
Udi Apsel and Ronan I. Brafman. Extended Lifted Inference with Joint Formulas. In: *UAI-11 Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.
PDecMs: Marcel Gehrke, Tanya Braun, Ralf Möller, Alexander Waschkau, Christoph Strumann, and Jost Steinhäuser. Towards Lifted Maximum Expected Utility. In: *Proceedings of the First Joint Workshop on Artificial Intelligence in Health in Conjunction with the 27th IJCAI, the 23rd ECAI, the 17th AAMAS, and the 35th ICML*, 2018.
Marcel Gehrke, Tanya Braun, Ralf Möller, Alexander Waschkau, Christoph Strumann, and Jost Steinhäuser. Lifted Maximum Expected Utility. In: *Artificial Intelligence in Health*, 2019.

- Markov logic decision networks (MLDNs)
  - MLN + parameterised decisions + utility weights
    - Probability + utility weights per first-order formula
  - Use weighted model counting to solve MEU problem

MLDNs: Aniruddh Nath and Pedro Domingos. A Language for Relational Decision Theory. In: *Proceedings of the International Workshop on Statistical Relational Learning*, 2009.
MLDNs + WMC: Udi Apsel and Ronan I. Brafman. Lifted MEU by Weighted Model Counting. In: *AAAI-12 Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2012.

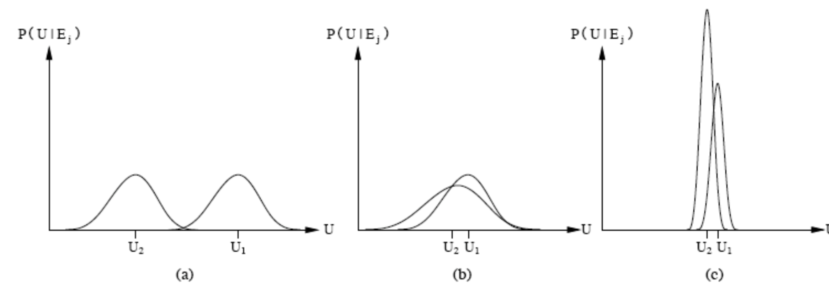
- Decision-theoretic Probabilistic Prolog (DTProbLog)
  - Utilities of DTProbLog programs combined into EU over theory defined by programs

DTProbLog: Guy Van den Broeck, Ingo Thon, Martijn van Otterlo, and Luc De Raedt. DTProbLog: A Decision-Theoretic Probabilistic Prolog. In: *AAAI-10 Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2010.

Interim Summary

- Decision networks
  - Probabilistic graphical model extended with decision and utility variables
  - Parameterised version: PDecM
    - Decision PRVs, utility PRVs, utility parfactors
    - Collective decisions for groups of indistinguishable constants
  - EU queries, MEU problem
    - Find set of actions (decisions) that lead to maximum expect utility
    - MEU-LVE using calls to LVE and LVE operators to answer EU queries
  - Combination function necessary if using a set of utility PRVs

    → Multi-attribute utility theory



Value of Information

Value of perfect information

Information-gathering agent

Decision Making in Decision Nets

- Assumes that all available information provided to agent before it makes its decision
  - Hardly ever the case
  - Know what questions to ask!
- Information value theory
  - Choose what information to acquire
  - Assume that prior to selecting an action represented by a decision PRV, the agent can acquire the value of any of the potentially observable PRVs/randvars
    - Simplified version of sequential decision making (next section)
      - Observation actions affect only agent's belief state, not the external physical state
      - Sequential decision making: actions have effect on surroundings

Value of information

- Idea: Compute value of acquiring each possible piece of evidence
  - Can be done directly from decision network
- Example: Buying oil drilling rights
  - Two blocks A and B, exactly one has oil, worth k
  - Prior probabilities 0.5 each, mutually exclusive
  - Current price of each block is $k/2$
  - “Consultant” offers accurate survey of A
  - Fair price for survey?
  - Solution: Compute expected value of information
    - = expected value of best action given the information minus expected value of best action without information
    - Survey may say “oil in A” or “no oil in A”, probability 0.5 each (given!)
      $$= [0.5 \cdot \text{value of “buy A” given “oil in A”} \\ + 0.5 \cdot \text{value of “buy B” given “no oil in A”}] - 0$$
      $$= (0.5 \cdot k/2) + (0.5 \cdot k/2) - 0 = k/2$$

General formula

- Current evidence E, current best action α, possible action outcomes S_i, potential new evidence E_j

$$EU(\alpha|E) = \max_{\mathrm{a}} \sum_i U(S_i)P(S_i \mid E, a)$$

- Suppose we knew $E_j = e_{jk}$, then we would choose a_{jk} such that

$$EU\left(\alpha_{e_{jk}} \mid E, E_j = e_{jk}\right) = \max_{\mathrm{a}} \sum_i U(S_i)P\left(S_i \mid E, a, E_j = e_{jk}\right)$$

- E_j is a random variable whose value is currently unknown
  ⇒ must compute expected gain over all possible values:

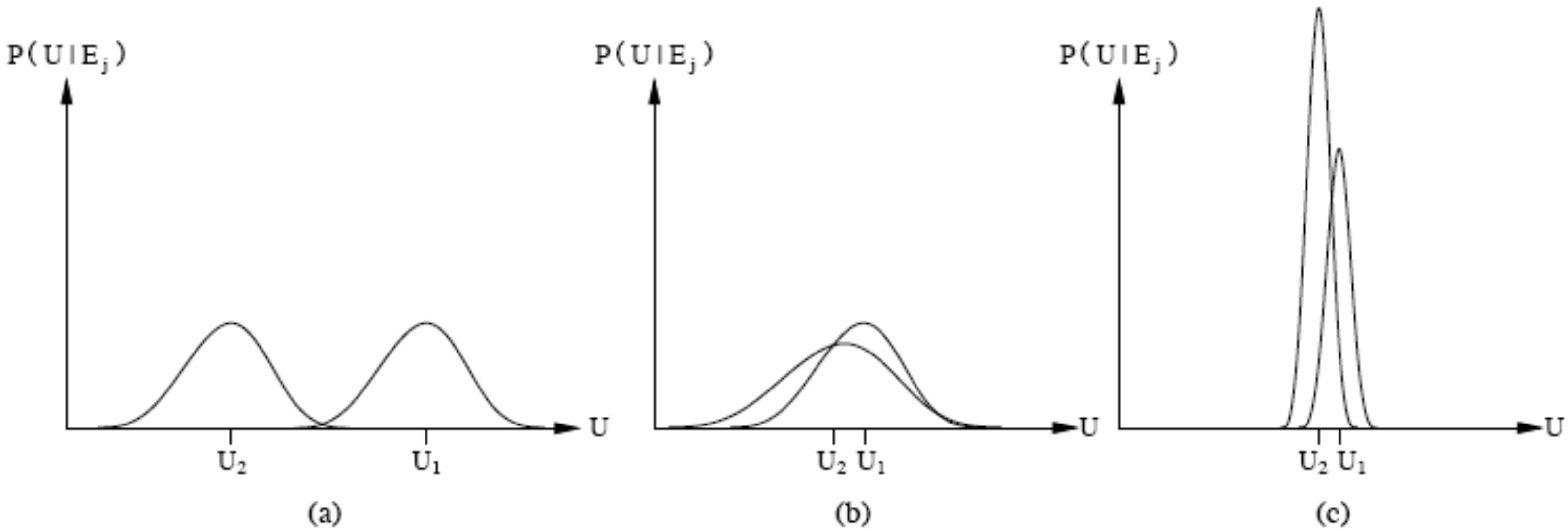
  $$VPI_E(E_j)$$
  $$= \left(\sum_k P(E_j \mid E)\, EU\left(\alpha_{e_{jk}} \mid E, E_j = e_{jk}\right)\right) - EU(\alpha, E)$$

  - VPI = value of perfect information

Properties of VPI

- Non-negative – in expectation
$$\forall j, E : VPI_E(E_j) \geq 0$$
- Non-additive – consider, e.g., obtaining E_j twice
$$VPI_E(E_j, E_k) \neq VPI_E(E_j) + VPI_E(E_k)$$
- Order-independent
$$VPI_E(E_j, E_k) = VPI_E(E_j) + VPI_{E,E_j}(E_k)$$
$$= VPI_E(E_k) + VPI_{E,E_k}(E_j)$$
- Note: When more than one piece of evidence can be gathered, maximising VPI for each to select one is not always optimal
  - → Evidence-gathering becomes a sequential decision problem

Qualitative behaviors



a) Choice is obvious, information worth little
b) Choice is non-obvious, information worth a lot
c) Choice is non-obvious, information worth little

- *Information has value to the extent that it is likely to cause a change of plan and to the extent that the new plan will be significantly better than the old plan*

Information Gathering Agent

```
function INFORMATION-GATHERING-AGENT(percept)
returns: an action
persistent: D, a decision network

   integrate percept into D
   j ← the value that maximises VPI(E_j)/Cost(E_j)
   if VPI(E_j) > Cost(E_j) then
      return Request(E_j)
   else
      return the best action from D
```

- Ask questions $Request(E_j)$ in a reasonable order
- Avoid irrelevant questions
- Take into account importance of piece of information j in relation to $Cost(E_j)$

Interim Summary

- Value of information
  - How much does it cost to obtain a new piece of information?
  - Will that piece of information change the current best plan?
  - How much more utility can one expect from the new best plan?
  - Is it worth it?

- Information-gathering agent
  - Request piece of evidence if the expected utility of action/plan outweighs the cost

Outline: 7. Decision Making

A. *Static decision making*
- Utility theory
- Parameterised decision models (PDecM)
  - Modelling, semantics, inference tasks
  - Inference algorithm: LVE as an example
- Value of Information

B. ***Sequential decision making***
- Parameterised dynamic decision models (PDDecM)
- Temporal MEU problem, inference
- Acting

Decision Making over Time

So far:

- Calculate the expected utility of a decision and its effect on the (current) state

With time

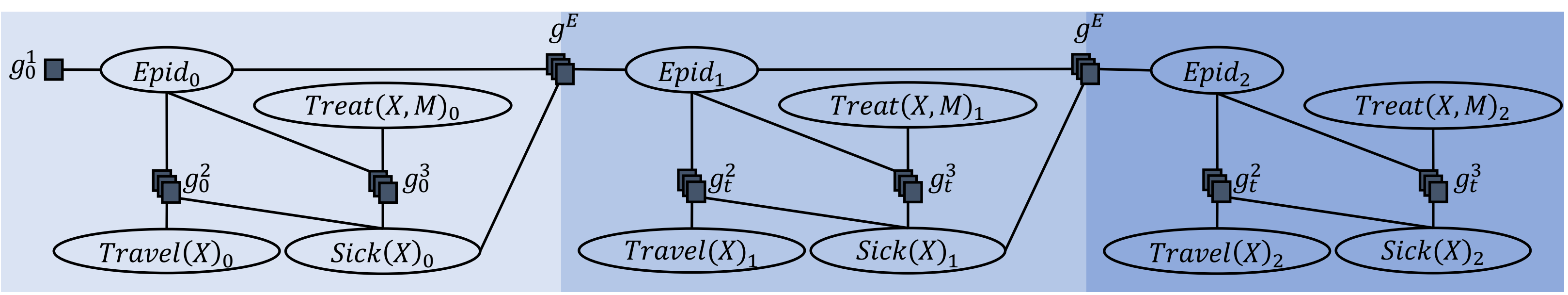
- Decisions need to be made at each time step
- Each time step yields a utility
- Decisions/actions have an effect not only on the current state/utility but also on the future and therefore, future decisions

→ Need to consider a temporal sequence of decisions and project its effect into the future

→ Requires calculating the expected utility over a sequence

Recap: PDM

- Assumptions: Markov-1, stationary process
- PDM $G = (G_0, G_\rightarrow)$ where
  - G_0 is a PM describing the intra-slice behaviour for $t = 0$:
    $$G_0 = \{g_0^i\}_{i=1}^{n_0}$$
  - $G_\rightarrow$ is a PM describing the intra- and inter-slice behaviour for $t > 0$
    $$G_\rightarrow = G_{t-1} \cup G_t \cup G_{t-1,t}$$
    - $G_t = \{g_t^k\}_{k=1}^{n_t}, G_{t-1} = G_{t|t \text{ replaced by } t-1}$
    - $G_{t-1,t} = \{g^j\}_{j=1}^{n}, g^j = \phi\left(A_\pi^1, \dots, A_\pi^{l_j}\right)_{|C}, \pi \in \{t-1, t\}$
- Semantics: Unrolling G for a given maximum step T, grounding, and building a full joint distribution

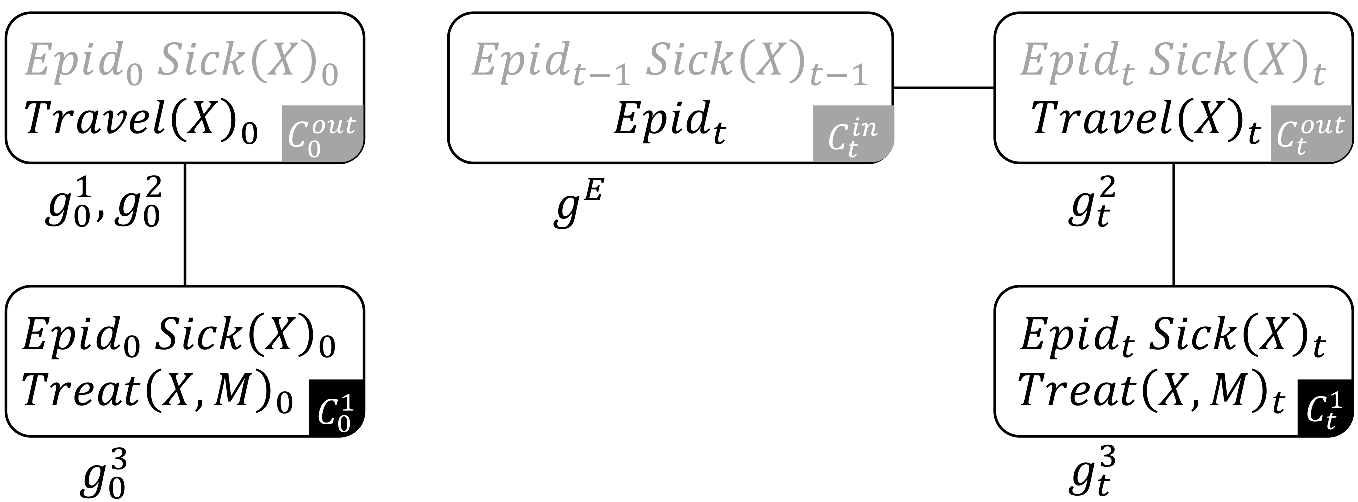


Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

Recap: Inference with PDMs

- Inference tasks:
  Query $P(R_\pi | \boldsymbol{E}_{0:\tau})$,
  τ the current step
  - Filtering: $\pi = \tau$
  - Prediction: $\pi > \tau$
  - Hindsight: $\pi < \tau$

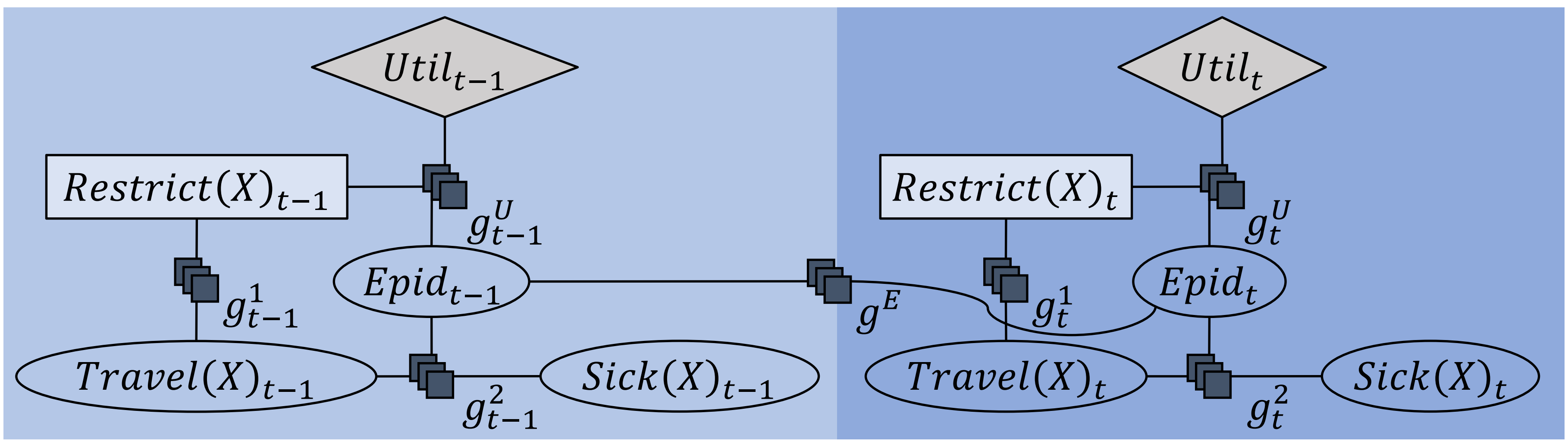
- Algorithm: LDJT
  - Two FO jtrees $(J_0, J_\rightarrow)$ with interfaces for separation between past and present
  - LJT as a subroutine for query answering
  - Forward/backward messages to move in time



Decision Making over Time

- Basis: a dynamic model such as a PDM $(G_0, G_\rightarrow)$
  - Describe behaviour over time using interslice parfactors
  - Within a slice, describe intra-slice (static) behaviour

→ Extend intra-slice parts with decision + utility PRVs
  - Extending PMs for decision making
  - PDMs allow for predicting effect of sequence of decisions



Time slices

A First Version of a PDDecM

- $G = (G_0, G_\rightarrow)$ is a PDM where the two static models $G_0, G_\rightarrow$ are PDecMs, i.e.,
  - G_0 is a PDecM describing the intra-time slice behaviour for $t = 0$
    $$G_0 = \{g_0^i\}_{i=1}^{n_0} \cup \{g_0^u\}_{u=1}^{m_0}$$
  - $G_\rightarrow$ is a PDecM describing the intra- and inter-slice behaviour for $t > 0$
    $$G_\rightarrow = G_{t-1} \cup G_t \cup G_{t-1,t}$$
    - $G_t = \{g_t^i\}_{i=1}^{n_t} \cup \{g_t^u\}_{u=1}^{m_t}$
    - $G_{t-1} = G_{t|t \; replaced \; by \; t-1}$
    - $G_{t-1,t} = \{g^j\}_{j=1}^{n}$

Assumptions about Actions

- Assumptions in terms of time
  - Discrete time steps
  - Markov-1
  - Stationary process
- Lead to assumptions/ constraints on decisions
  - Actions can be carried out from one time step to the next (no duration)
  - Effect/outcome of actions immediately captured in the next time step
  - Actions do not affect the stationary process

Actions over Time

- Given a set of decision PRVs
  - $\boldsymbol{A}_t = \{A_t^1, \dots, A_t^k\}$
- Temporal sequence of decisions = temporal action assignment
  - Compound event for a sequence of steps $t_1 : t_2$
  - I.e.,

$$\boldsymbol{a}_{t_1:t_2} = (\boldsymbol{a}_{t_1}, \boldsymbol{a}_{t_1+1}, \dots, \boldsymbol{a}_{t_2-1}, \boldsymbol{a}_{t_2})$$

    - $\boldsymbol{a}_{t_i} = \{A_{t_i}^1 = a^1, \dots, A_{t_i}^k = a^k\}$
- Decision making over time usually involves calculating the best sequence starting at current step τ, making decisions for κ steps
  - I.e., finding the best $\boldsymbol{a}_{\tau:\tau+\kappa}$

So far, we used utilities to determine “the best”. How can we use them with time?

Utilities over Time

- One has to iterate over temporal action assignments and pick the one that yields the maximum expected utility
  - Each individual slice has a utility (or reward)
    - → Multi-attribute utility theory
    - All individual utilities need to be combined into one expected utility of the complete sequence
  - Assumption: Preference of one sequence over the other does not depend on time → preferences are stationary
    - Formally, if two state sequences $(\boldsymbol{r}_0, \boldsymbol{r}_1, \boldsymbol{r}_2, \dots)$ and $(\boldsymbol{s}_0, \boldsymbol{s}_1, \boldsymbol{s}_2, \dots)$ have the same starting state ($\boldsymbol{r}_0 = \boldsymbol{s}_0$), then the two sequences $(\boldsymbol{r}_1, \boldsymbol{r}_2, \dots)$ and $(\boldsymbol{s}_1, \boldsymbol{s}_2, \dots)$ should be preference-ordered in the same way as $(\boldsymbol{r}_0, \boldsymbol{r}_1, \boldsymbol{r}_2, \dots)$ and $(\boldsymbol{s}_0, \boldsymbol{s}_1, \boldsymbol{s}_2, \dots)$

  → Preference independence between the different slices

  → Use additive combination function

Temporal Combination Functions

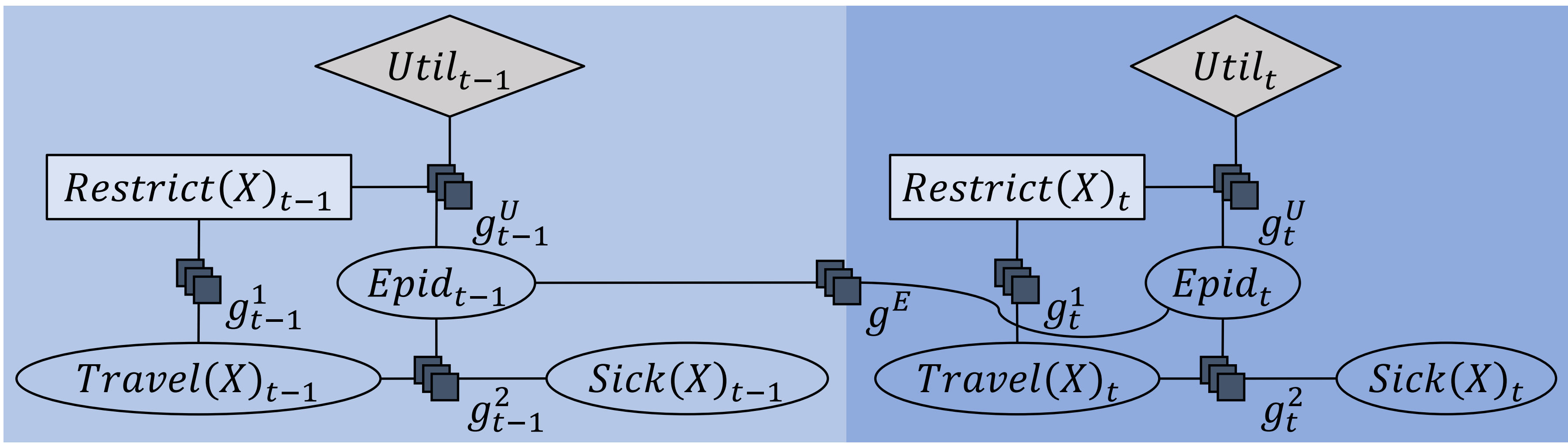
- Actually, two approaches to combine utilities
- Additive
  - Sum over individual utilities
    $$U(\boldsymbol{r}_0, \boldsymbol{r}_1, \boldsymbol{r}_2, \dots) = U(\boldsymbol{r}_0) + U(\boldsymbol{r}_1) + U(\boldsymbol{r}_2) + \cdots$$
- Discounted
  - Using a discount factor $\gamma \in [0,1]$
    - Reward now may be more important than one in t steps
    - If $\gamma = 1$: additive
  - Utility of a sequence = sum over discounted individual utilities
    $$U(\boldsymbol{r}_0, \boldsymbol{r}_1, \boldsymbol{r}_2, \dots) = \gamma^0 \cdot U(\boldsymbol{r}_0) + \gamma^1 \cdot U(\boldsymbol{r}_1) + \gamma^2 \cdot U(\boldsymbol{r}_2) \dots$$
- Model for dynamic decision making needs to include a function to combine utilities over time

EU Query over Time: Exact

- EU query at step τ in a PDM with PDecMs

$$eu(\boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa}) = \sum_{\boldsymbol{r} \in \mathcal{R}\left(rv(G^U_{\tau:\tau+\kappa})\right)} P(\boldsymbol{r} | \boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa}) \cdot \sum_{\tau'=0}^{\kappa} \gamma^{\tau'} \varphi^U_{\tau+\tau'}\left(rv_U(G_{\tau+\tau'})\right)$$

  - $\varphi^U_{\tau'}$ combination function for utility PRVs at step τ'
  - Additive/discounted combination over time



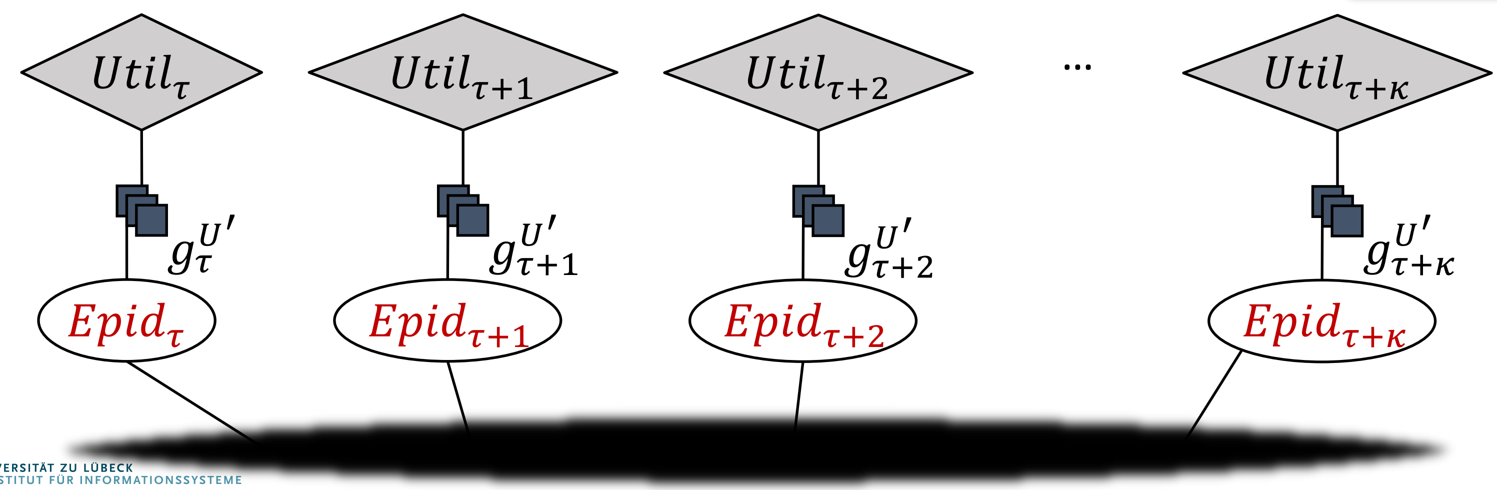
EU Query over Time: Exact

- EU query at step τ in a PDM with PDecMs

$$eu(\boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa}) = \sum_{\boldsymbol{r} \in \mathcal{R}\left(rv\left(G^U_{\tau:\tau+\kappa}\right)\right)} P(\boldsymbol{r} | \boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa}) \cdot \sum_{\tau'=0}^{\kappa} \gamma^{\tau'} \varphi^U_{\tau+\tau'}\left(rv_U(G_{\tau+\tau'})\right)$$

  - Problem: query involves query terms from all slices $\tau : \tau + \kappa$, which also means unrolling the model
    - Unlikely that query terms are independent

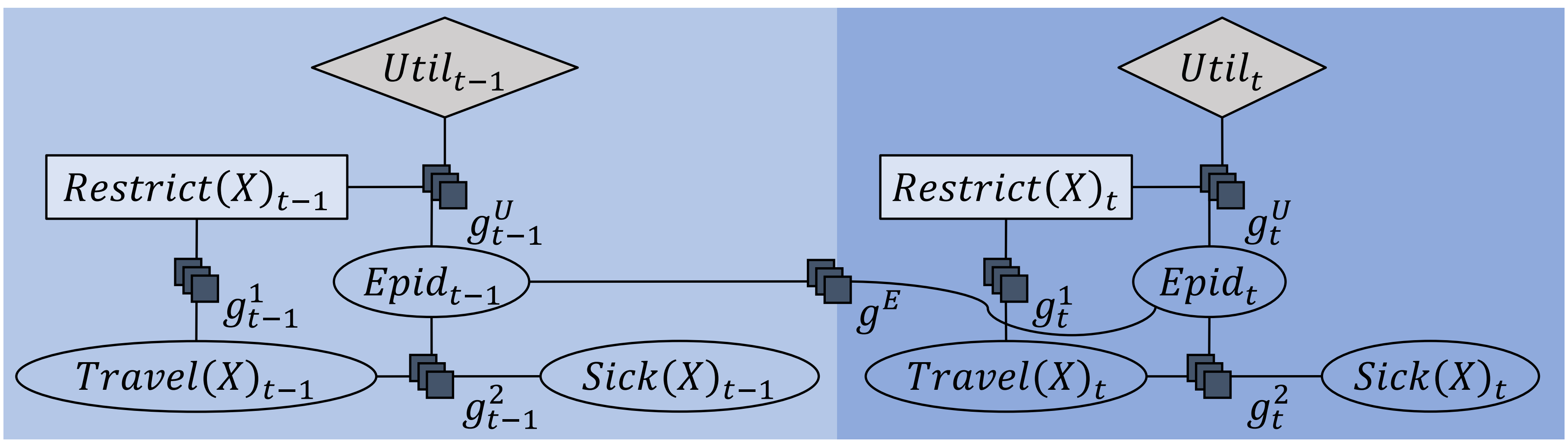
Realistically **not** computable!



EU Query over Time: Approximate

- Move the probability query into the sum

$$eu(\boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa})$$
$$= \sum_{\boldsymbol{r} \in \mathcal{R}\left(rv\left(G^U_{\tau:\tau+\kappa}\right)\right)} \sum_{\tau'=0}^{\kappa} \gamma^{\tau'} P\left(\pi_{rv\left(G^U_{\tau+\tau'}\right)}(\boldsymbol{r}) | \boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa}\right) \varphi^U_{\tau+\tau'}\left(rv_U(G_{\tau+\tau'})\right)$$
$$= \sum_{\tau'=0}^{\kappa} \gamma^{\tau'} \sum_{\boldsymbol{r} \in \mathcal{R}\left(rv\left(G^U_{\tau+\tau'}\right)\right)} P(\boldsymbol{r} | \boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa}) \varphi^U_{\tau+\tau'}\left(rv_U(G_{\tau+\tau'})\right)$$

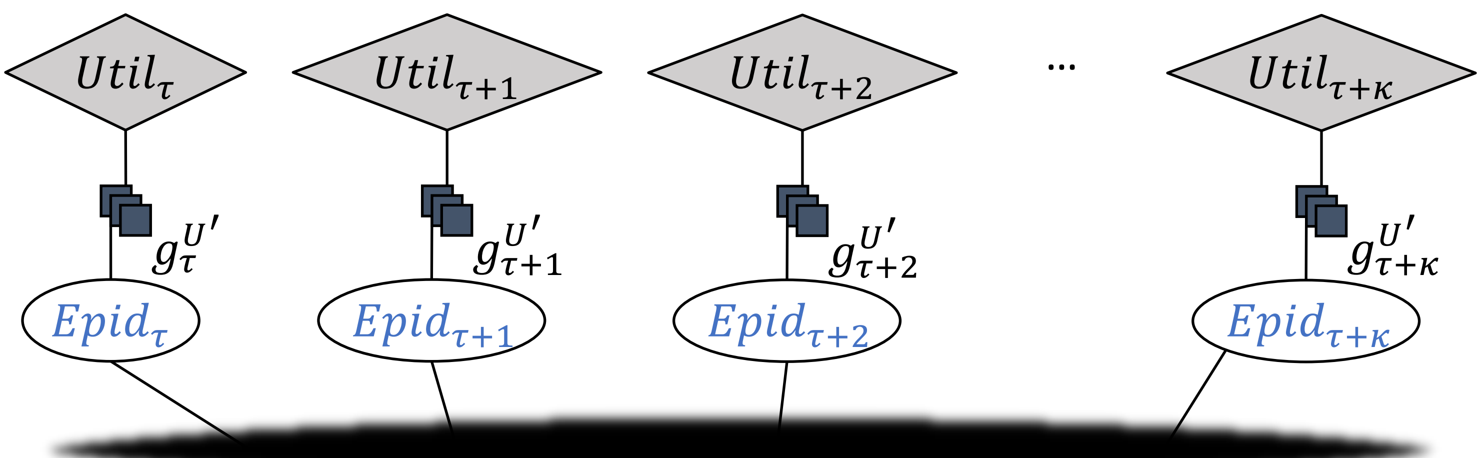


EU Query over Time: Approximate

- Temporal EU query sums over individual EU queries

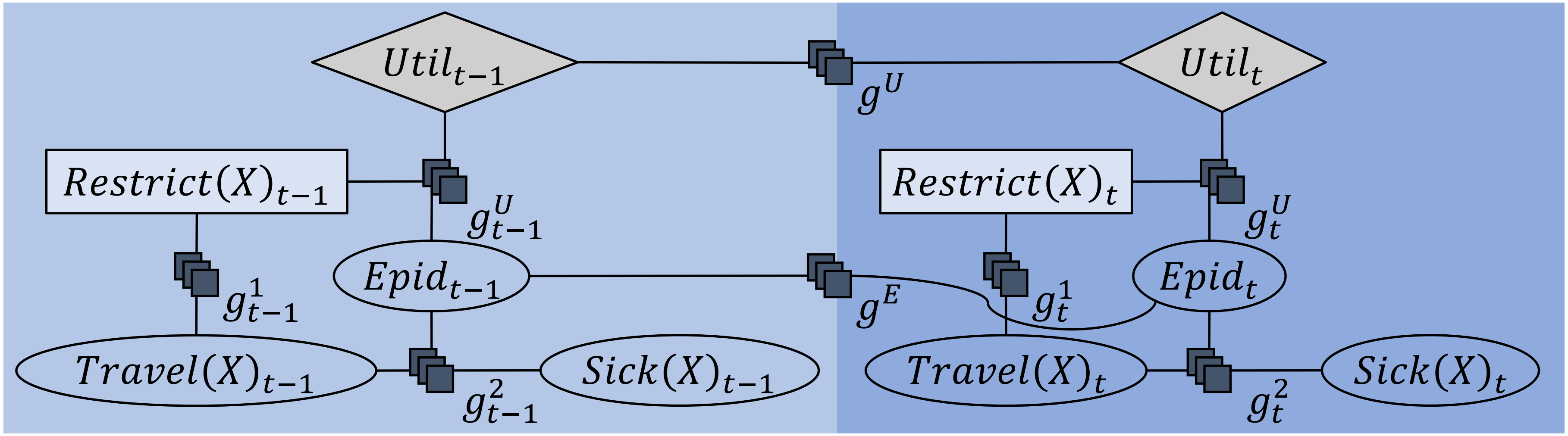
$$eu(\boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa}) = \sum_{\tau'=0}^{\kappa} \gamma^{\tau'} \sum_{\boldsymbol{r} \in \mathcal{R}\left(rv\left(G^{U}_{\tau+\tau'}\right)\right)} P(\boldsymbol{r} | \boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa}) \varphi^{U}_{\tau+\tau'}\left(rv_U(G_{\tau+\tau'})\right)$$

$$= \sum_{\tau'=0}^{\kappa} \gamma^{\tau'} eu(\boldsymbol{E}_{1:\tau}, \boldsymbol{a}_{\tau:\tau+\kappa})$$



Utility Transfer Function

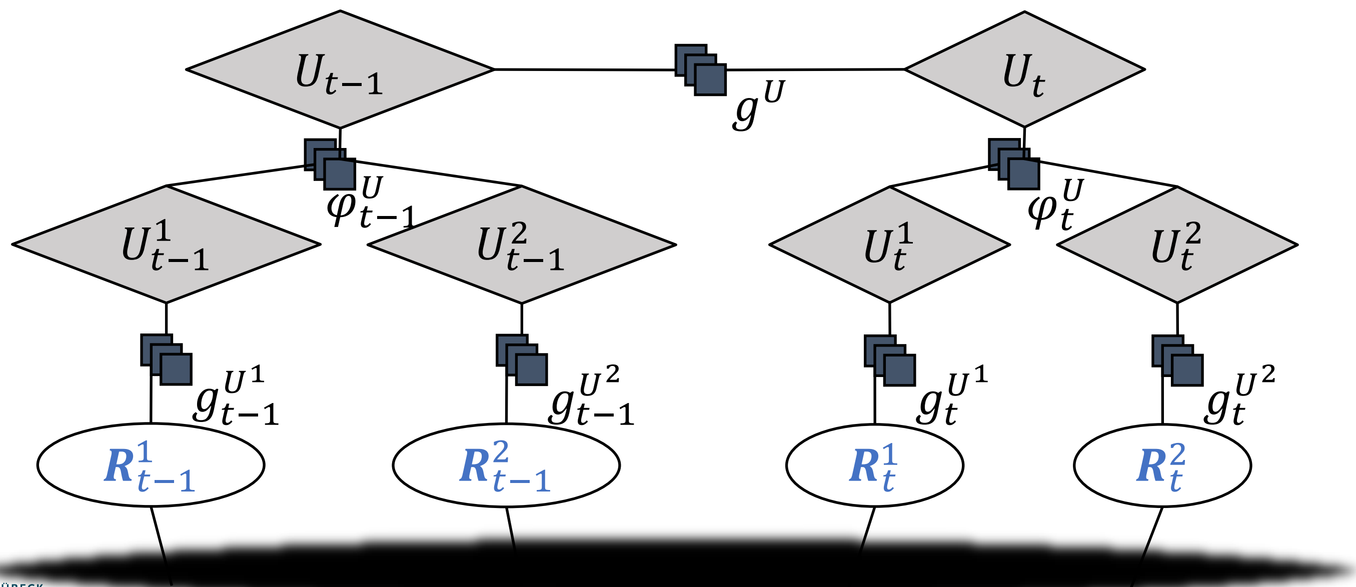
- Transfer (expected) utility at slice $t-1$ to slice t
- Simple case: single utility PRV per slice U_t
  - Parfactor as utility transfer function takes U_{t-1} and U_t as arguments and assigns the (discounted) sum to U_t again
  - $g^U = \phi^{U_t}(U_{t-1}, U_t)_{|C} = U_{t-1} + \gamma^c U_t$
    - Inter-slice parfactor as PRVs from both $t-1$ and t occur
    - c depends on how far in the temporal action assignment we are
    - May only be calculated once U_t is set by answering the EU query



Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Temporal Maximum Expected Utility. In: *CAI-19 Proceedings of the 32nd Canadian Conference on Artificial Intelligence*, 2019.

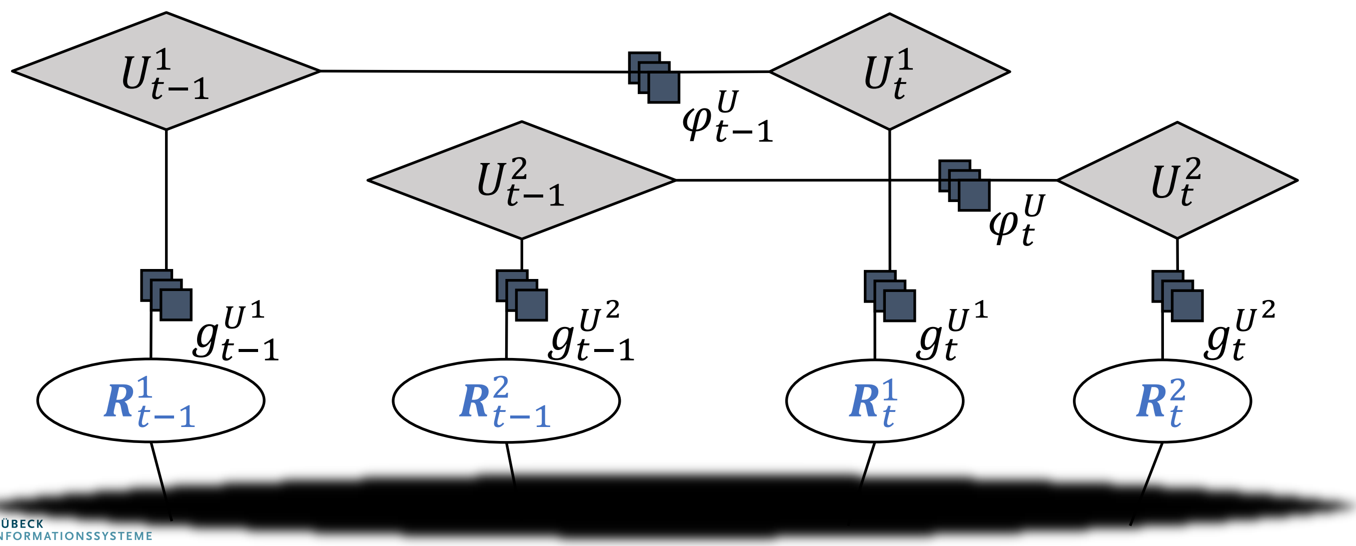
Utility Transfer Function

- Transfer (expected) utility at slice $t-1$ to slice t
- General case: set of utility PRVs per slice $\{U_t^u\}_{u=1}^{m_t}$
  - Two approaches
  1. Use a combination function φ_t^U to combine $\{U_t^u\}_{u=1}^{m_t}$ into one utility U_t and use a single utility transfer function g^U as seen on previous slide



Utility Transfer Function

- Transfer (expected) utility at slice $t-1$ to slice t
- General case: set of utility PRVs per slice $\{U_t^u\}_{u=1}^{m_t}$
  - Two approaches
  2. Use a set of utility transfer functions g^{U_u} for (sets of) utility PRVs $\{U_t^u\}_{u=1}^{m_t}$ and only combine them at the end using a combination function φ^U



A Complete PDDecM

- A PDDecM $G = (G_0, G_\rightarrow)$ is a PDM where the two static models $G_0, G_\rightarrow$ are PDecMs, i.e.,
  - G_0 is a PDecM describing the intra-time slice behaviour for $t = 0$
    $$G_0 = \{g_0^i\}_{i=1}^{n_0} \cup \{g_0^u\}_{u=1}^{m_0} \cup \{\varphi_0^U\}$$
  - $G_\rightarrow$ is a PDecM describing the intra- and inter-slice behaviour for $t > 0$
    $$G_\rightarrow = G_{t-1} \cup G_t \cup G_{t-1,t}$$
    - $G_t = \{g_t^i\}_{i=1}^{n_t} \cup \{g_t^u\}_{u=1}^{m_t} \cup \{\varphi_t^U\}$
      - φ_t^U combination function, may be omitted (in Approach 2)
    - $G_{t-1} = G_{t|t\ replaced\ by\ t-1}$
    - $G_{t-1,t} = \{g^j\}_{j=1}^{n} \cup \{g^u\}_{u=1}^{m}$
      - g^u utility transfer function, may be singleton (in Approach 1)

Approach 2 requires a combination function φ^U for computing an EU query at the end

Temporal MEU Problem

- Given a PDDecM G, evidence $\boldsymbol{E}_{1:t}$, and a number κ
- Temporal MEU problem
  - Find the temporal action assignment that yields the highest expected utility in G
  - Formally,

$$\mathrm{meu}(G|\boldsymbol{E}_{1:t},\kappa) = \left(\boldsymbol{a}^*_{t:t+\kappa}, eu(\boldsymbol{E}_{1:t},\boldsymbol{a}^*_{t:t+\kappa})\right)$$

$$\boldsymbol{a}^*_{t:t+\kappa} = \underset{\boldsymbol{a}_{t:t+\kappa}\in\mathcal{R}(\boldsymbol{A}_{t:t+\kappa})}{\mathrm{argmax}} eu(\boldsymbol{E}_{1:t},\boldsymbol{a}_{t:t+\kappa})$$

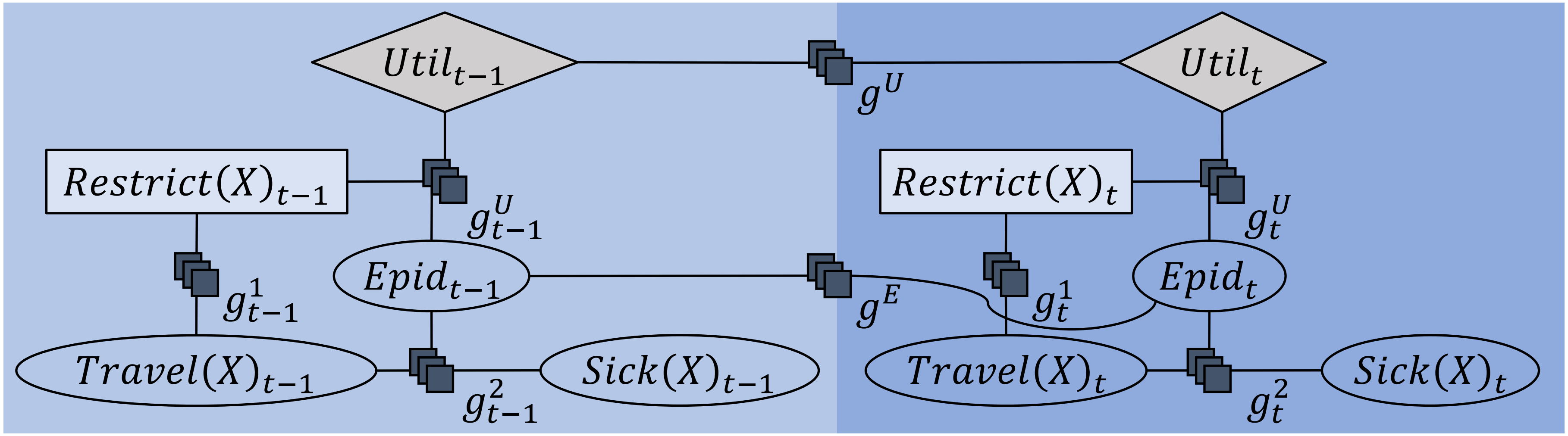
    - Size of $\mathcal{R}(\boldsymbol{A}_{t:t+\kappa})$ exponential in number of groups (as with static decision making) and κ

Example

- Given no evidence, current time step $\tau = 2$, and $\kappa = 2$

$$\left(\boldsymbol{a}_{2:4}^{i}\right)^{*} = \underset{i \in \{1,\dots,8\}}{\operatorname{argmax}}\, eu\left(\boldsymbol{a}_{2:4}^{i}\right)$$

	$\boldsymbol{a}_2^i$	$\boldsymbol{a}_3^i$	$\boldsymbol{a}_4^i$
$\boldsymbol{a}_{2:4}^1$	ban	ban	ban
$\boldsymbol{a}_{2:4}^2$	ban	ban	$free$
$\boldsymbol{a}_{2:4}^3$	ban	$free$	ban
$\boldsymbol{a}_{2:4}^4$	ban	$free$	$free$
$\boldsymbol{a}_{2:4}^5$	$free$	ban	ban
$\boldsymbol{a}_{2:4}^6$	$free$	ban	$free$
$\boldsymbol{a}_{2:4}^7$	$free$	$free$	ban
$\boldsymbol{a}_{2:4}^8$	$free$	$free$	$free$



Solving a Temporal MEU Problem

- Answering temporal MEU queries using LDJT
- But: utility transfers/EU queries per step have to be dealt with → Bookkeeping
- Two approaches:
  I. In parallel to FO jtree, keep *counters* for each utility transfer function and add the current, possibly discounted, EU value to the existing counter
  II. After message passing, calculate all necessary EU queries at corresponding parclusters, perform a *utility message pass* that sends the EU results to the outcluster to be sent over to the next step with the forward message and be distributed to the corresponding parclusters during message passing

MEU-LDJT

```
procedure LDJT((G_0, G_→), Q_{0:T_q}, E_{0:T_e})
    Build 1.5-slice model G_→^{1.5}
    Construct FO jtree (J_0, J_→) for G_0 and G_→^{1.5}
    for τ in 0 ... T_q do
        Instantiate J_τ
        Add α_{τ-1} to incluster of J_τ                ▸ if τ > 0
        Enter evidence E_τ into J_τ
        Pass messages in J_τ
        Answer queries Q_τ
        Calculate α_τ
```

- Build two FO jtrees for PDDecM
- Proceed step-wise (τ)
  - Answer probability/ assignment queries as before
  - Given an MEU query with κ,
    - Calculate the set of temporal action assignments $\boldsymbol{a}$ based on groups in G_τ and κ
    - For each $\boldsymbol{a}$, proceed for κ steps
      - Set actions at each step
      - Calculate EU queries + bookkeeping (Approach I or II)
        - For ease of computation, include helper parfactors during construction according to the EU queries
    - Return $\boldsymbol{a}^*$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Temporal Maximum Expected Utility. In: *CAI-19 Proceedings of the 32nd Canadian Conference on Artificial Intelligence*, 2019.

Acting

- After computing a temporal action assignment, assignment is acted out, including:
  - Actuators get commands to carry out the action behind a current assignment,
  - Internal state is updated: Decision PRVs are set, (messages are passed,) and then time moves on
- Agent can then continue to act according to the temporal assignment or recalculate as new evidence comes in
  - Recalculate each step
  - Recalculate once

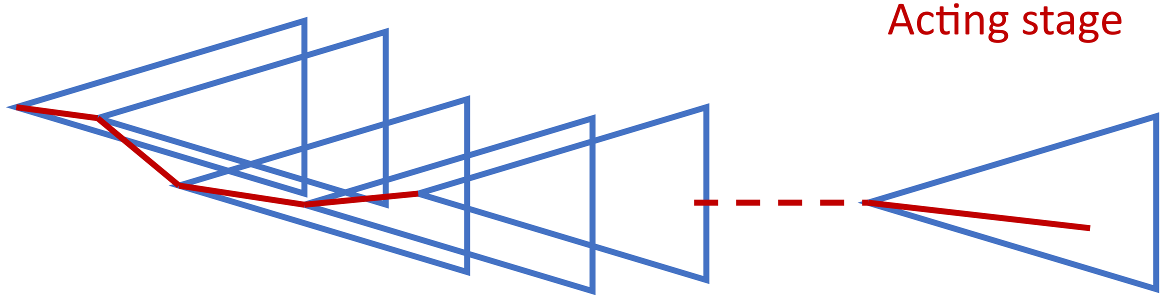
Next three slides adapted from material provided by Dana Nau, http://www.laas.fr/planning

Using Decision Making in Acting

- Receding horizon:
  - Call meu, obtain a temporal action assignment π, perform 1st action, call meu again …
  - Like game-tree search (chess, checkers, etc.)
- Useful when unpredictable things are likely to happen
  - Re-plans immediately
- Potential problem:
  - May pause repeatedly while waiting for meu to return

```
Run-MEU(G,κ)
    while E ← new evidence do
        absorb E in G
        π ← meu(G,κ)
        a ← pop-first(π)
        perform a
        update G with a
```

Decision making stage
Acting stage



Using Decision Making in Acting

- Call meu, execute the temporal action assignment as far as possible, do not call meu again unless necessary
- Simulate tests whether the assignment will execute correctly
  - Lower-level refinement, physics-based simulation, prediction accuracy < some threshold
- Potential problems
  - May might miss opportunities to replace π with a better assignment
  - Without Simulate, may not detect problems until it is too late

```
Run-Lazy-MEU(G,κ)
    while E ← new evidence do
        absorb E in G
        π ← meu(G,κ)
        while π ≠ () and
                Simulate(G,π)
                    ≠ failure do
            a ← pop-first(π)
            perform a
            update G with a
```

Using Decision Making in Acting

- May detect opportunities earlier than Run-Lazy-MEU
  - But may miss some that Run-MEU would find
- Without Simulate, may fail to detect problems until it is too late
  - Not as bad at this as Run-Lazy-MEU

```
Run-Concurrent-MEU(G,κ)
    π ← ⟨⟩
    // thread 1 + 2 run
          concurrently
thread 1:
    while E ← new evidence do
        absorb E in G
        π ← meu(G,κ)
thread 2:
    while E ← new evidence do
        absorb E in G
        if π ≠ () and
                Simulate(G,π)
                    ≠ failure then
            a ← pop-first(π)
            perform a
            update G with a
```

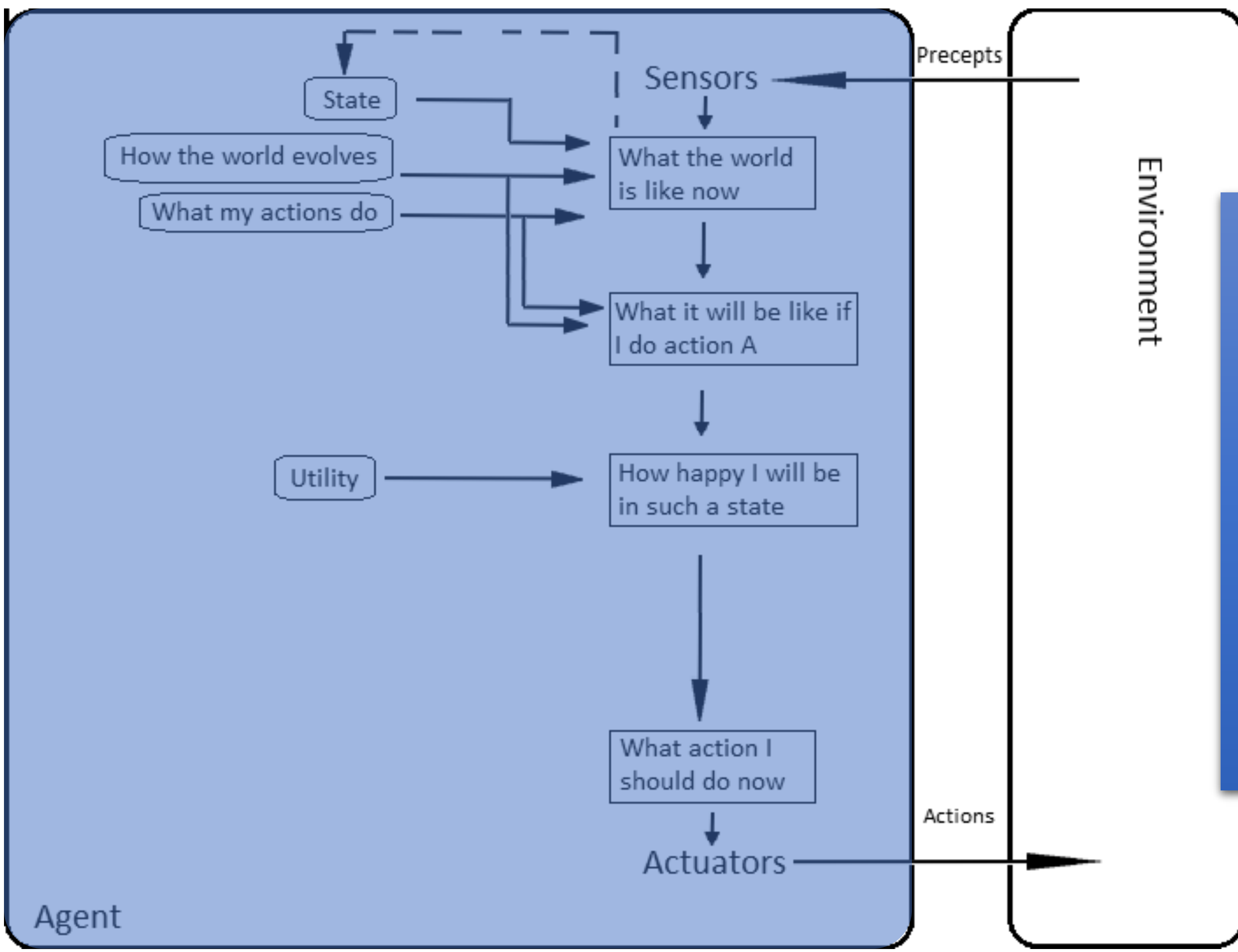
Offline vs. Online Decision Making

- Online decision making: Depends on how far one looks into the future
  - Can extend the look-ahead further and further, propagating future effects back to present to see if the current best decision still holds
  - Extending the look-ahead to infinity, one will eventually reach a fix point: offline decision making
- Offline decision making
  - Solving a partially observable Markov decision process (POMDP) – Ch. 17 in Russell/Norvig's AIMA3
    - Basically find the steady state in terms of actions that leads to the maximum expected utility
      - Look-up table, fast during acting, huge overhead beforehand
      - Reacting to extreme situations/evidence no possible
    - No longer factorised model but a transition function over complete state space
  - Part of the lecture: *Advances in Data Science and AI (Summer Term) – Automated Planning and Acting*

Interim Summary

- PDDecMs as a combination of
  - PDMs for temporal aspect
  - PDecMs for decision making
- Temporal expected utilities
  - Additive or discounted utilities
  - Exact solution virtually impossible to compute
  - Approximation by summing over EU query result for each step
- Inference based on LDJT
  - Interface again for sequential independence
  - Bookkeeping of EU results

Setting: Agent with Utilities



PDDecMs
- Uncertainty by probabilities
- Relational aspect by parameterisation
- Temporal aspect by time indexing
- Decisions and effects by actions & utilities in a temporal model

Outline: 7. Decision Making

A. *Static decision making*
- Utility theory
- Parameterised decision models (PDecM)
  - Modelling, semantics, inference tasks
  - Inference algorithm: LVE as an example
- Value of Information

B. *Sequential decision making*
- Parameterised dynamic decision models (PDDecM)
- Temporal MEU problem, inference
- Acting

$\Longrightarrow$ Next: Continuous Space