Lösende Planungsgraphen

Sei $G_n = \langle N, E \rangle$ ein Planungsgraph der Ordnung n für $\langle S, O, Z \rangle$.

- Der Planungsgraph $G_0 = \langle S, \emptyset \rangle$ zu einem gegebenen Planungsproblem $\langle S, O, Z \rangle$ ist eine **Lösung**, gdw. $Z \subseteq S$
- Für $n > 0$ ist G_n eine Lösung von $\langle S, O, Z \rangle$, gdw.
  - $Z \subseteq F_n$
  - keine $f, g \in Z$ schließen sich wechselseitig aus in F_n
  - es gibt $L \subseteq O_{n-1}$, eine minimale Menge nicht-ausschließlicher Operatoren, sodass Z in der Vereinigung der Nachbedingungen von L liegt, und für die Vereinigung V der Vorbedingungen von L gilt:
    G_{n-1} ist eine Lösung von $\langle S, O, V \rangle$

↪ „Regression der Ziele über Planungsgraph-Schichten“

Planungsgraphexpansion und ihr Ende

Expansion eines Planungsgraphen $G_n=\langle N,E\rangle$ der Ordnung n: Einfügen der Operatorschicht O_n, der Faktenschicht F_{n+1}, sowie der Mutex-Kanten

Ein **Expansionsfixpunkt** ist ein Planungsgraph, in dem zwei Faktenebenen und alle ihre Mutex-Bedingungen identisch sind.

Satz

Ergibt die Expansion eines Planungsgraphen für ein Planungsproblem $\langle S,O,Z\rangle$ einen Fixpunkt in Tiefe n und ist $Z \not\subseteq F_n$ oder zwei Teilziele aus Z schließen sich aus in F_n, so ist das Planungsproblem unlösbar.

Beweisidee: $F \subset F_i$ für ein i und Mutex-Freiheit ist notwendige Voraussetzung für Lösbarkeit

GRAPHPLAN

Algorithmus GRAPHPLAN(Σ)

Eingabe: $\Sigma = (S, O, F)$: propositionales Planungsproblem

Ausgabe: partiell geordneter Plan oder **fail**

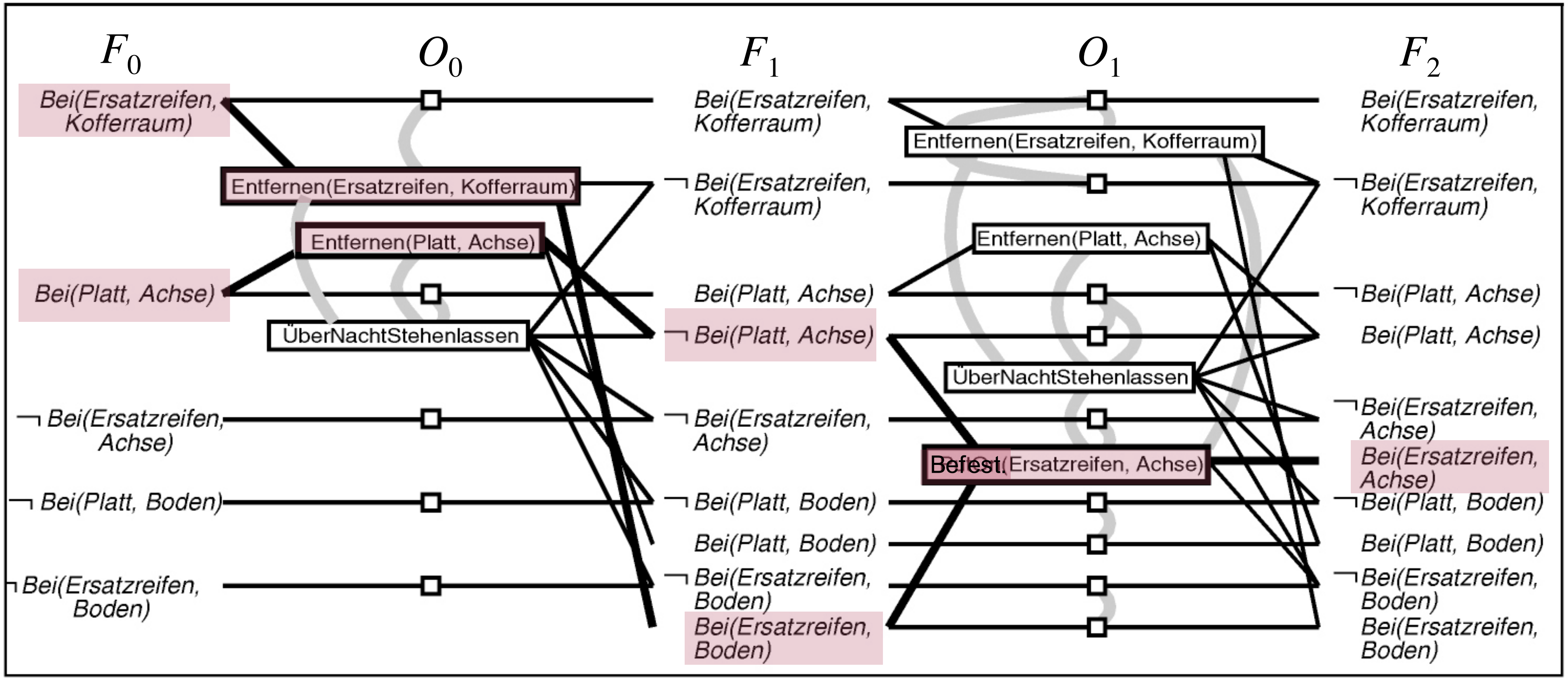
---

0. $\Gamma := (F_0 = S, \emptyset)$
1. **repeat forever**
   - 1.0. **if** Γ ist Lösung von Σ;
   - 1.1. **then** $\Pi :=$ extrahiere einen Lösungsplan aus Γ;
   - 1.2. **return**(Π)
   - 1.3. **else if** Γ ist Expansionsfixpunkt
   - 1.4. **then** **return**(**fail**)
   - 1.5. **else** expandiere Γ um eine (Operator+Fakten-) Schicht

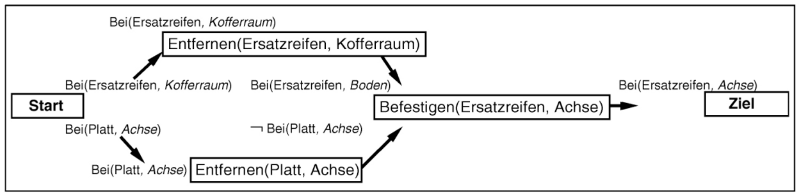
   end repeat

Planextraktion: beginnend mit der Zielmenge in F_n bis F_0 wähle (*backtracking*!) nicht-Mutex-Operatoren in O_{i-1}, die die Ziele in F_i als Nachbedingungen erzeugen. Ziele in F_{i-1} sind die Vorbedingungen der gewählten Op.en in O_{i-1}.

Planextraktion beim Reifenwechseln



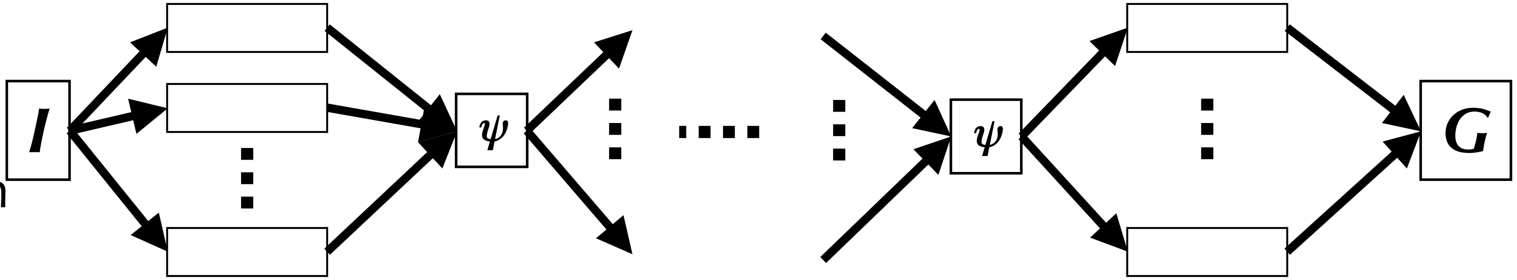
Nur kleiner Teil der Mutexe eingezeichnet!



Lösungsplan wie zuvor

Eigenschaften von GRAPHPLAN

ψ: Pseudo-Operatoren zur Synchronisation



- Pläne sind Folgen von „Zeitschritten" (*time steps*) untereinander ungeordneter Operatoren.
- Literale im Planungsgraph wachsen monoton über die Faktenebenen (**Grund**: einmal erschienen, erhalte Lit. durch Persistenzoperator)
- Operatoren im Planungsgraph wachsen monoton über die Operatorebenen (**Grund**: Monotonie der Literale)
- Mutexe zwischen gleichen Objekten schrumpfen monoton über die Planungsgraphebenen (**Grund**: Ausschluss von *Fakten* kann entfallen mit neuen Operatoren; damit auch Konkurrenz von Operatoren.)
- GRAPHPLAN terminiert (**Grund**: Monotonien + Expansionsfixpunkt)

Planen als propositionales Erfüllbarkeitsproblem

- Überführe STRIPS-Problembeschreibungen in aussagelogisches Format:
  - Startsituation ist Konjunktion von Grundfakten
  - Zielsituation ist Konjunktion von Grundfakten
  - Formuliere Vor- und Nachbedingungen in Invarianten über Situationen um („*state constraints*“) (Beispiel s. Russell/Norvig Kap. 11.5; effiziente Repräsentationen können bereichsspezifisch sein!)
- Unterschiede zu Situationskalkül-Repräsentation:
  - Endliches Herbrand-Universum: Verwende propositionale Schemata statt FOL-Formeln
  - *Zeitschritte* wie bei Graphplan
  - Maximalgrenze von Zeitschritten (Terminerung!)
- Als Planer nimm prop. *Model Checker* (DPLL, WALKSAT)

Propositionale Axiomenschemata I

Startzustand $\bigwedge_{f \in s_0} f_0 \wedge \bigwedge_{f \notin s_0} \neg f_0$ Index 0 in f_0: Zeitschritt!

Bspl.: Roboter R, der zwischen Positionen L,M fahren kann
$\mathrm{At}(R,L,0) \wedge \neg \mathrm{At}(R,M,0)$

Zielzustand (Zeitschritt) $\bigwedge_{f \in g^+} f_n \wedge \bigwedge_{f \notin g^-} f_n$

Bspl. (n=1): $\mathrm{At}(R,M,1)$ evtl. zusätzlich: $\wedge \neg \mathrm{At}(R,L,1)$

Notw. Aktionseigenschaften $a_i \Rightarrow \left(\bigwedge_{p \in \mathrm{precond}(a)} p_i \wedge \bigwedge_{p \in \mathrm{effect}(a)} e_{i+1} \right)$

Bspl.: $\mathbf{Move}(R,L,M,0) \Rightarrow \mathrm{At}(R,L,0) \wedge \mathrm{At}(R,M,1) \wedge \neg \mathrm{At}(R,L,1)$
$\mathbf{Move}(R,M,L,0) \Rightarrow \mathrm{At}(R,M,0) \wedge \mathrm{At}(R,L,1) \wedge \neg \mathrm{At}(R,M,1)$

Propositionale Axiomenschemata II

Wandel-Axiome

$$\neg f_i \wedge f_{i+1} \Rightarrow \left(\bigvee_{a \in A \mid f_i \in \text{effect}^+(a)} a_i \right) \wedge$$

$$f_i \wedge \neg f_{i+1} \Rightarrow \left(\bigvee_{a \in A \mid f_i \in \text{effect}^-(a)} a_i \right)$$

Bspl.: $\neg \mathrm{At}(R,L,0) \wedge \mathrm{At}(R,L,1) \Rightarrow \mathbf{Move}(R,M,L,0)$

$\neg \mathrm{At}(R,M,0) \wedge \mathrm{At}(R,M,1) \Rightarrow \mathbf{Move}(R,L,M,0)$

$\mathrm{At}(R,L,0) \wedge \neg \mathrm{At}(R,L,1) \Rightarrow \mathbf{Move}(R,L,M,0)$

$\mathrm{At}(R,M,0) \wedge \neg \mathrm{At}(R,M,1) \Rightarrow \mathbf{Move}(R,M,L,0)$

Operatorausschluss-Axiome (soweit erforderlich) $a_i \Rightarrow \neg b_i$

Bspl.: $\mathbf{Move}(R,M,L,0) \Rightarrow \neg \mathbf{Move}(R,L,M,0)$

$\mathbf{Move}(R,L,M,0) \Rightarrow \neg \mathbf{Move}(R,M,L,0)$

SATPLAN

```
function SATPLAN(problem, T_max) returns solution or failure
   inputs: problem, a planning problem
           T_max, an upper limit for plan length

   for T = 0 to T_max do
      cnf, mapping ← TRANSLATE-TO-SAT(problem, T)
      assignment ← SAT-SOLVER(cnf)
      if assignment is not null then
         return EXTRACT-SOLUTION(assignment, mapping)
   return failure
```

- Attraktiv durch „saubere Semantik“ und Verifizierbarkeit
- Größe der Probleme begrenzt durch exponenzielle Klausenzahl
- Pragmatische Kombination (BLACKBOX-Planer) mit Planungsgraphen

Krücken und Methoden

- „Neo-/Klassisches" Planen macht Voraussetzungen für Anwendbarkeit (s. Folien 326, 328)
- Nicht alle Domänen erfüllen diese Voraussetzungen „objektiv"
- „Leichte Verletzungen" von Algorithmen-Voraussetzungen kann man pragmatisch „überbrücken" (im Planen wie auch sonst)
- Russell/Norvig (Kap. 12.3-6) beschreiben solche „Krücken" (Nichtdeterminismus, Informationsmangel, Ausführungsfehler)
- Erfüllt die Domäne grundsätzlich nicht die Voraussetzungen für klassisches Planen, sollte man grundsätzlich andere Verfahren wählen
- Methoden zum Planen (Entscheiden) unter Unsicherheit behandeln Russell/Norvig in Kap.16-17.

Rationale Entscheidungen unter Unsicherheit

- Planungsvariante nun:
  - **Aktionen**: möglicherweise nicht-deterministisch
  - **Bereichsinformation** bei Planung und/oder Ausführung möglicherweise unvollständig
- Verwende elementare Konzepte der **Nutzentheorie**
- Modelliere zunächst Entscheidungen über einzelne Aktionen, dann Pläne (*policies*, „Politiken“)

Prinzip des maximalen erwarteten Nutzens (MEU)

Für Aktionen A und Evidenz E maximiere:

$$\boldsymbol{EU(\alpha|E)} := \max_A \sum_i \left[P(\mathrm{Result}_i(A) \mid \mathrm{Do}(A), E) \times U(\mathrm{Result}_i(A)) \right]$$

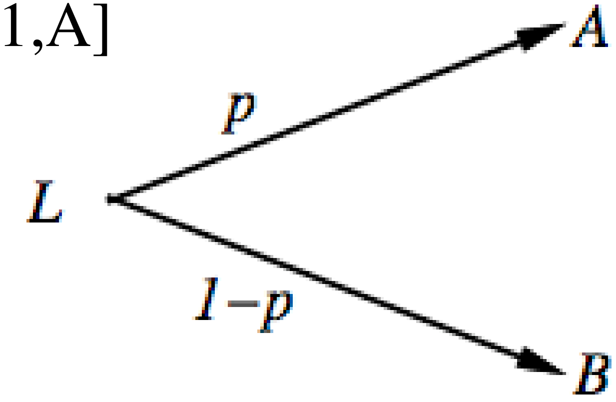
Lotterien

Formalisierung von Aktionen mit nicht-deterministischen Effekten:

$L = [p_1,C_1; p_2,C_2; \ldots; p_n,C_n]$

für alternative Zustände oder Lotterien C_i
und ihre W'keiten p_i, $\sum_i p_i = 1$

Zustände entsprechen Lotterien der Form [1,A]



Notation für Lotterien:

$A > B$: A ist **präferiert** gegenüber B

$A \sim B$: **Indifferenz** bzgl. A und B

$A \geq B$: Präferenz oder Indifferenz

Die Axiome der Nutzentheorie

Wohlgeordnetheit $(A \succ B) \vee (B \succ A) \vee (A \sim B)$

Transitivität $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$

Kontinuierlichkeit $A \succ B \succ C \Rightarrow \exists p \; [p, A;\; 1-p, C] \sim B$

Substituierbarkeit $A \sim B \Rightarrow [p, A;\; 1-p, C] \sim [p, B; 1-p, C]$

Monotonie $A \succ B \Rightarrow (p \geq q \Leftrightarrow [p, A;\; 1-p, B] \succsim [q, A;\; 1-q, B])$

Zerlegbarkeit $\left[p, A; (1-p), [q, B; (1-q), C]\right] \sim \left[p, A; (1-p)q, B; (1-p)(1-q), C\right]$

Die Nutzenfunktion

… ist eine abgeleitete Funktion, gegeben Lotterien und Präferenzen

Konzeptuell verhalten sich Agenten nach Präferenzen, nicht nach Nutzenfunktionen!

Satz (Ramsey, 1931; von Neumann&Morgenstern, 1944):

Gegeben Präferenzen entsprechend den Axiomen. Dann existiert eine reellwertige Funktion U, sodass

1. $U(A) \geq U(B) \Leftrightarrow A \geq B$
2. $U([p_1,C_1; \ldots; p_n,C_n]) = \sum_i p_i U(C_i)$

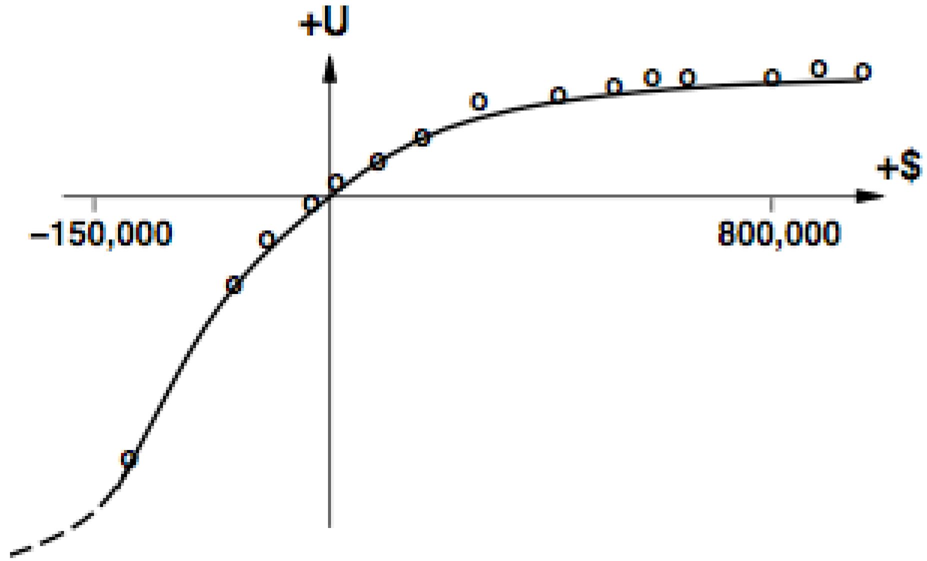
Normalisierung der Nutzenfunktion: $\forall A.\ 0 \leq U(A) \leq 1$

Exkurs: Ist Kontostand eine Nutzenfunktion?

Für die meisten Menschen **nicht**!

z.B.: [1,„Gewinne 1 Mio €“] > [0.5,„Gewinne 0 €“; 0.5,„Gewinne 3 Mio €“]

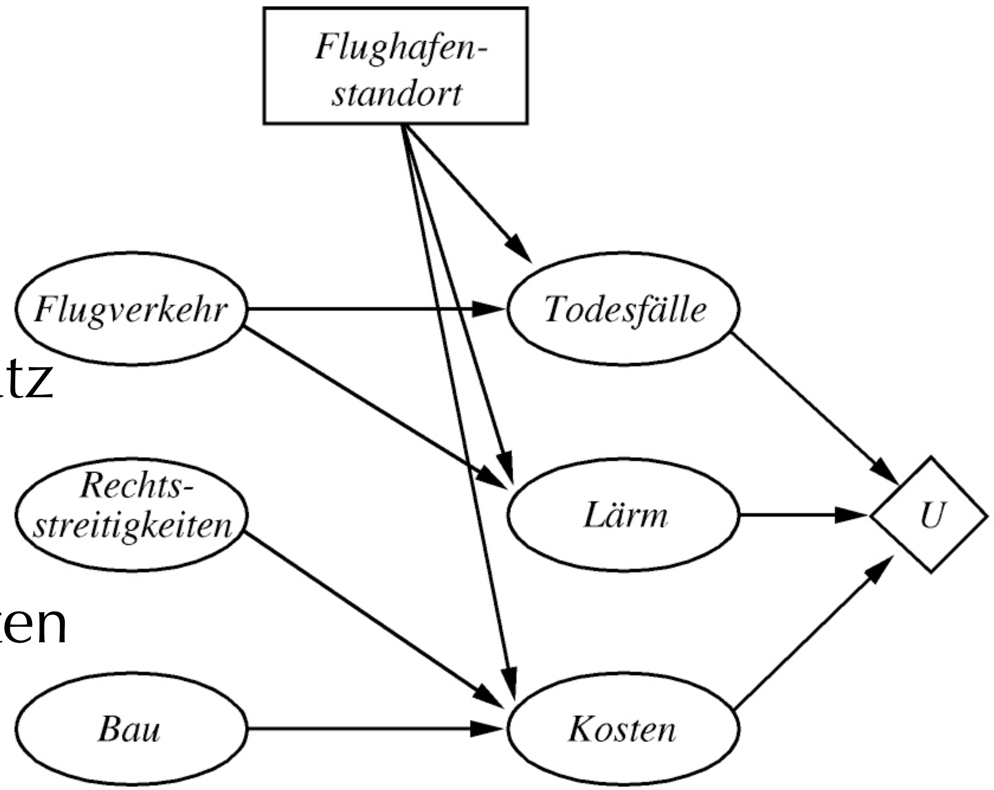
Empirisch ermitteltes U:



normative vs.
deskriptive
Entscheidungstheorie

Entscheidungsnetze

Für Entscheidungen bei mehreren Variablen (multivariat):
Modellierung durch Bayes-Netze mit Aktions- und Nutzenknoten



Beispiel

Standortwahl für neuen Flugplatz

- Zufallsvariablen wie in Bayes-Netzen
- Rechteck: Entscheidungsknoten
- Raute: Nutzenfunktion – **multivariat**!

Multivariate Nutzenfunktionen I

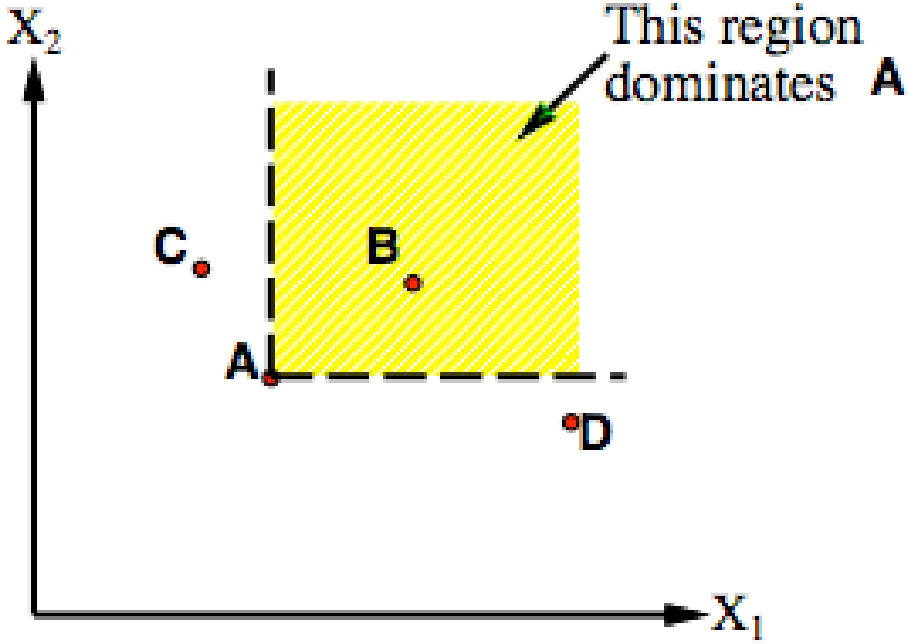
- Wie definiere Nutzenfunktionen $U(X_1,\ldots,X_n)$ *mehrerer* Var.n?
  Beispiel: Was ist $U(\mathit{Lärm}, \mathit{Kosten}, \mathit{Todesfälle})$?
  Wie vergleiche U(„20000 beeinträchtigt“, 4,6*Mrd*€, „0,06 Tote/mpm“)
  mit U(„70000 beeinträchtigt“, 4,2*Mrd*€, „0,06 Tote/mpm“)?
- Eigentliches Problem: Was ist die zu Grunde liegende Präferenzfunktion?
- Idee 1: Identifiziere Formen von Unabhängigkeit der Variablen bzgl Präferenzen (analog bedingter Unabh‘keit in Bayes-Netzen)
- Behandle (approximiere) Variablen als (wechselseitig) **präferenziell unabhängig**:

$$U(X_1,\ldots,X_n) = V(X_1,\ldots,X_n) = \sum_i V_i(X_i)$$

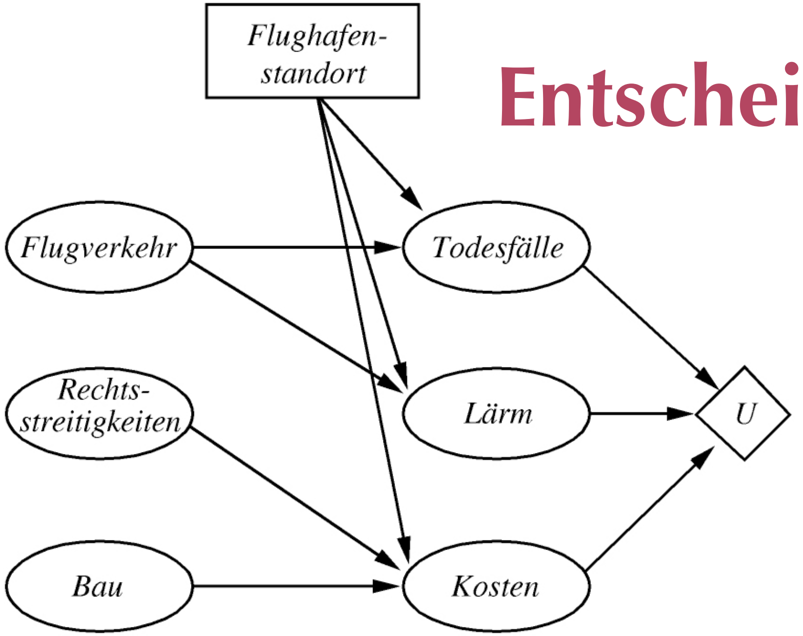
additive Wert-Funktion

Multivariate Nutzenfunktionen II

- Idee 2: Identifiziere Formen von **Dominanz** von Variablen über Abhängige
  (z.B.: „je stadtferner der Flughafen, desto geringer die Grundstückskosten“)
  z.B. **strikte** Dominanz: $\forall i.\ X_i(B) \geq X_i(A)$, folglich $U(B) \geq U(A)$



Deterministic attributes



Entscheiden mit Netzen

1. Setze die Evidenzvariablen im aktuellen Zustand
2. Für alle möglichen Entscheidungen im Entscheidungsknoten:
   a) Setze den Entscheidungsknoten entsprechend
   b) berechne die a posteriori W'keiten der Elternknoten des Nutzen-Knotens (z.B. mit Sampling-Algorithmen aus 4.5)
   c) Berechne Nutzwert für die Entscheidung
3. Gib Aktion mit höchstem Nutzen aus