KINI9ID1 – Intelligence Artificielle et Décision

M1 informatique

Septembre 2022

Préférences pour l'aide à la décision

Introduction

Problème général :

- Une liste d'objets / items / alternatives
- Un-e « Decision Maker » :

$\Rightarrow$ Ordonner / *rank* tout ou partie de ces objets pour sélectionner / choisir / ...

Introduction

Problème général :

- Une liste d'objets / items / alternatives
- Un-e « Decision Maker » :
  - Présidente de jury
  - Client d'un site de e-commerce
  - Utilisateur d'un système de configuration de voiture / ordinateur, ...
  - ...

$\Rightarrow$ Ordonner / *rank* tout ou partie de ces objets
pour sélectionner / choisir / ...

Introduction

Problème général :

- Une liste d'objets / items / alternatives
- Un-e « Decision Maker » :
  - Présidente de jury
  - Client d'un site de e-commerce
  - Utilisateur d'un système de configuration de voiture / ordinateur, ...
  - ...
- Convives qui doivent choisir un menu commun
- Électeurs

$\Rightarrow$ Ordonner / *rank* tout ou partie de ces objets
pour sélectionner / choisir / ...

Introduction : Alternatives, ordre, critères

$\mathcal{O}$: ensemble d'alternatives / objets / items.....

$\succeq$: pré-ordre sur $\mathcal{O}$

= relation binaire, réflexive et transitive

Introduction : Alternatives, ordre, critères

$\mathcal{O}$: ensemble d'alternatives / objets / items.....

$\succeq$: pré-ordre sur $\mathcal{O}$

= relation binaire, réflexive et transitive

$\Rightarrow$ on écrit :

- $o \succeq o'$ si $(o, o') \in \succeq$;
- $o \succ o'$ si $(o, o') \in \succeq$ et $(o', o) \notin \succeq$: o strictement meilleure que o' ;
- $o \sim o'$ si $(o, o') \in \succeq$ et $(o', o) \in \succeq$: o et o' également préférées ;
- $o \bowtie o'$ si $(o, o') \notin \succeq$ et $(o', o) \notin \succeq$: o et o' incomparables.

Introduction : Alternatives, ordre, critères

$\mathcal{O}$: ensemble d'alternatives / objets / items.....

$\succeq$: pré-ordre sur $\mathcal{O}$

= relation binaire, réflexive et transitive

Idéalement $\succ$ est un ordre strict total, mais pas toujours connu / possible.

Introduction : Alternatives, ordre, critères

Définir $\succeq$ directement sur $\mathcal{O}$:

- problème cognitif pour le DM, même si $\mathcal{O}$ pas trop grand (exemple : menus)
  il faut ordonner toutes les alternatives de $\mathcal{O}$;
- problème informatique : $\mathcal{O}$ peut être très grand.

Introduction : Alternatives, ordre, critères

Définir $\succeq$ directement sur $\mathcal{O}$:

- problème cognitif pour le DM, même si $\mathcal{O}$ pas trop grand (exemple : menus)
  il faut ordonner toutes les alternatives de $\mathcal{O}$;
- problème informatique : $\mathcal{O}$ peut être très grand.

Exemple 1 : Choix de menus.

- Plusieurs convives, il faut choisir un menu commun

$\Rightarrow$ Chaque convive donne son ordre de préférence

$+$ Une règle de vote

Plat	**Dessert**	**Vin**
Risotto	Profiterolles	Rouge
Truffade	Flan	Rosé
		Rouge

$\Rightarrow$ Ordonner 12 menus ??

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / variables

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / variables
- $\underline{X} = D_X$ = domaine de la variable X
  = ensemble des valeurs possibles pour X

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / variables
- $\underline{X} = D_X$ = domaine de la variable X
  = ensemble des valeurs possibles pour X
- $o[X]$: valeur du critère X pour l'alternative o alternative o :

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / variables
- $\underline{X} = D_X$ = domaine de la variable X
  = ensemble des valeurs possibles pour X
- $o[X]$: valeur du critère X pour l'alternative o alternative o :

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / variables
- $\underline{X} = D_X$ = domaine de la variable X
  = ensemble des valeurs possibles pour X
- $o[X]$: valeur du critère X pour l'alternative o alternative o :

Notations

- $U, V, \ldots \subseteq \mathcal{X}$ ensembles / vecteurs de variables

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / variables
- $\underline{X} = D_X$ = domaine de la variable X
  = ensemble des valeurs possibles pour X
- $o[X]$: valeur du critère X pour l'alternative o alternative o :

Notations

- $U, V, \ldots \subseteq \mathcal{X}$ ensembles / vecteurs de variables
- $\underline{U}$: ens. des tuples de valeurs possibles pour les variables de U

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / variables
- $\underline{X} = D_X$ = domaine de la variable X
  = ensemble des valeurs possibles pour X
- $o[X]$: valeur du critère X pour l'alternative o alternative o :

Notations

- $U, V, \ldots \subseteq \mathcal{X}$ ensembles / vecteurs de variables
- $\underline{U}$: ens. des tuples de valeurs possibles pour les variables de U

On suppose que $\mathcal{O} \subseteq \underline{\mathcal{X}}$.

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / variables
- $\underline{X} = D_X$ = domaine de la variable X
  = ensemble des valeurs possibles pour X
- $o[X]$: valeur du critère X pour l'alternative o alternative o :

Notations

- $U, V, \ldots \subseteq \mathcal{X}$ ensembles / vecteurs de variables
- $\underline{U}$: ens. des tuples de valeurs possibles pour les variables de U

  On suppose que $\mathcal{O} \subseteq \underline{\mathcal{X}}$.

- $o[U]$ = tuple de valeurs des variables de U pour l'alternative o

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / variables
- $\underline{X} = D_X$ = domaine de la variable X
  = ensemble des valeurs possibles pour X
- $o[X]$: valeur du critère X pour l'alternative o alternative o :

Notations

- $U, V, \ldots \subseteq \mathcal{X}$ ensembles / vecteurs de variables
- $\underline{U}$: ens. des tuples de valeurs possibles pour les variables de U

  On suppose que $\mathcal{O} \subseteq \underline{\mathcal{X}}$.

- $o[U]$ = tuple de valeurs des variables de U pour l'alternative o
- $u[V] = u[U \cap V]$ = valeurs données par u pour les variables de $V \cap U$

Introduction : Alternatives, ordre, critères

- $\mathcal{X}$: ensemble de critères / features / caractéristiques / $\underline{\text{variables}}$
- $\underline{X} = D_X$ = domaine de la variable X
  = ensemble des valeurs possibles pour X
- $o[X]$: valeur du critère X pour l'alternative o alternative o :

Notations

- $U, V, \ldots \subseteq \mathcal{X}$ ensembles / vecteurs de variables
- $\underline{U}$: ens. des tuples de valeurs possibles pour les variables de U

  On suppose que $\mathcal{O} \subseteq \underline{\mathcal{X}}$.

- $o[U]$ = tuple de valeurs des variables de U pour l'alternative o
- $u[V] = u[U \cap V]$ = valeurs données par u pour les variables de $V \cap U$

Les variables permettent des représentations compactes de $\succeq$ sur $\underline{\mathcal{X}}$

Introduction :

Dans la suite on aborde :

- Modèles / langages de représentation de préférences sur $\underline{\mathcal{X}}$

Introduction :

Dans la suite on aborde :

- Modèles / langages de représentation de préférences sur $\underline{\mathcal{X}}$
  - Utilités additives / réseaux de fonctions de coûts

Introduction :

Dans la suite on aborde :

- Modèles / langages de représentation de préférences sur $\underline{\mathcal{X}}$
  - Utilités additives / réseaux de fonctions de coûts
  - Ordres lexicographiques généralisés

Introduction :

Dans la suite on aborde :

- Modèles / langages de représentation de préférences sur $\underline{\mathcal{X}}$
  - Utilités additives / réseaux de fonctions de coûts
  - Ordres lexicographiques généralisés
  - Préférences « ceteris paribus »

Introduction :

Dans la suite on aborde :

- Modèles / langages de représentation de préférences sur $\underline{\mathcal{X}}$
  - Utilités additives / réseaux de fonctions de coûts
  - Ordres lexicographiques généralisés
  - Préférences « ceteris paribus »
- Algorithmes pour comparer / optimiser

Introduction :

Dans la suite on aborde :

- Modèles / langages de représentation de préférences sur $\underline{\mathcal{X}}$
  - Utilités additives / réseaux de fonctions de coûts
  - Ordres lexicographiques généralisés
  - Préférences « ceteris paribus »
- Algorithmes pour comparer / optimiser
- Apprentissage automatique

Modèles « numériques »

- Une fonction $v : \mathcal{O} \longrightarrow \mathbb{R}$
- $o \succeq o'$ ssi $v(o) \geq v(o')$

Modèles « numériques » : Utilités additives

$$v(o) = \sum_{X \in \mathcal{X}} \mathrm{ut}_X(o[X])$$

où $\mathrm{ut}_X : \underline{X} \longrightarrow \mathbb{R}$, $\forall X \in \mathcal{X}$

Modèles « numériques » : Utilités additives

$$v(o) = \sum_{X \in \mathcal{X}} \mathrm{ut}_X(o[X])$$

où $\mathrm{ut}_X : \underline{X} \longrightarrow \mathbb{R}, \forall X \in \mathcal{X}$

Exercice 1.

Donner une utilité additive qui représente l'ordre suivant :

$$(\text{lamb}, \text{red wine}) \succ (\text{vegetable}, \text{red wine}) \succ (\text{beef}, \text{red wine})$$
$$\succ (\text{lamb}, \text{white wine}) \succ (\text{vegetable}, \text{white wine}) \succ (\text{beef}, \text{white wine})$$

Modèles « numériques » : Utilités additives

$$v(o) = \sum_{X \in \mathcal{X}} \mathrm{ut}_X(o[X])$$

où $\mathrm{ut}_X : \underline{X} \longrightarrow \mathbb{R}, \forall X \in \mathcal{X}$

Exercice 1.

Donner une utilité additive qui représente l'ordre suivant :

$$(\text{lamb, red wine}) \succ (\text{vegetable, red wine}) \succ (\text{beef, red wine})$$
$$\succ (\text{lamb, white wine}) \succ (\text{vegetable, white wine}) \succ (\text{beef, white wine})$$

Remarque 1.

On voit que

- lamb préféré à vegetable, préféré à beef
- red wine préféré à white wine
- il y a un *compromis* : le choix du vin semble plus important que le choix du plat

Modèles « numériques » : Utilités additives

$$v(o) = \sum_{X \in \mathcal{X}} \mathrm{ut}_X(o[X])$$

où $\mathrm{ut}_X : \underline{X} \longrightarrow \mathbb{R}, \forall X \in \mathcal{X}$

Exercice 2 : Gonzales, Perny, 2004.

Démontrer qu'aucune utilité additive ne permet de représenter :

$$(\text{lamb}, \text{red wine}) \succ$$
$$(\text{vegetable}, \text{red wine}) \sim (\text{lamb}, \text{white wine}) \sim (\text{vegetable}, \text{white wine})$$
$$\succ (\text{beef}, \text{red wine}) \succ (\text{beef}, \text{white wine})$$

Modèles « numériques » : Utilités additives généralisées

$$v(o) = \sum_{U \in \mathcal{U}} \mathrm{ut}_U(o[U])$$

où $\mathrm{ut}_U : \underline{U} \longrightarrow \mathbb{R}, \forall U \in \mathcal{U}$
et $\mathcal{U} \subseteq 2^{\mathcal{X}}$ est un ensemble de "tuples" de variables

Remarque 2.

- Permet de représenter *tout* préordre partiel sur $\mathcal{O}$ (Pourquoi ?)
- Plus les U sont grands, plus la représentation est coûteuse.
- On verra plus loin comment on peut apprendre de telles représentations.

Modèles « numériques » : Contraintes valuées / wCSPs

$$v(o) = \sum_{U \in \mathcal{U}} c_U(o[U])$$

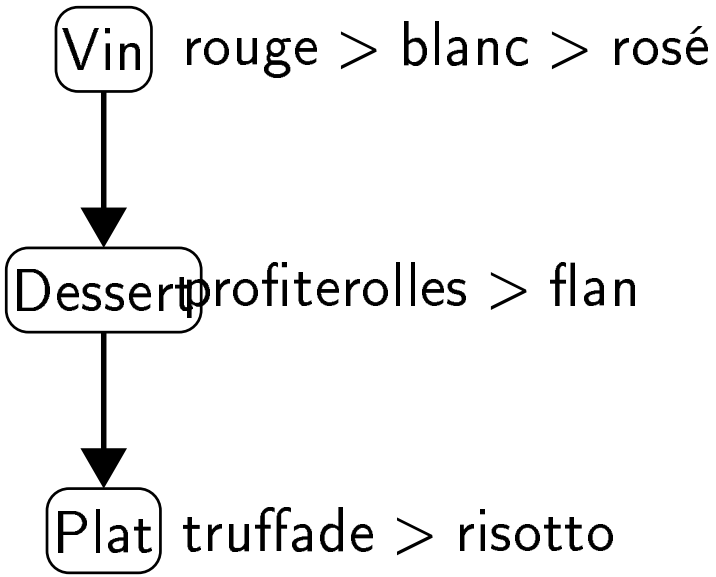
où $c_U : \underline{U} \longrightarrow \mathbb{R}^+ \cup +\infty$, $\forall U \in \mathcal{U}$

On parle de fonctions de coûts.

On cherche à minimiser v : $o \succeq o'$ ssi $v(o) \leq v(o')$.

Si les seules valeurs sont 0 et $+\infty$, alors on retrouve les CSPs.

Modèles lexicographiques

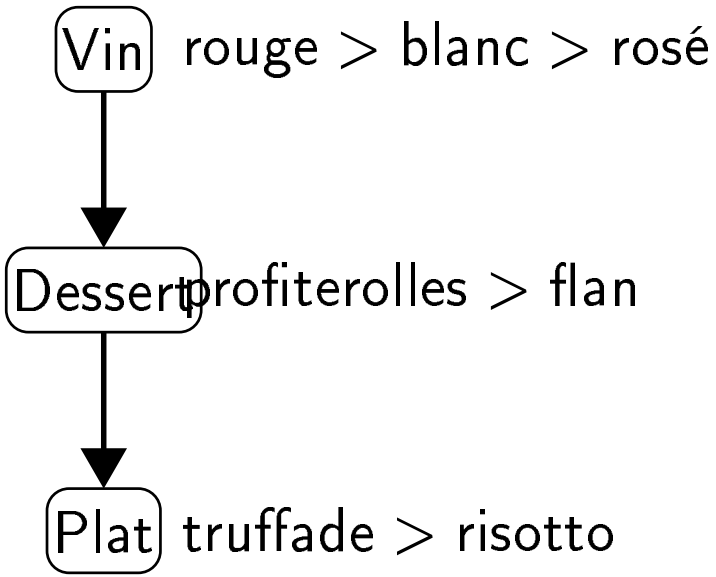


- Un ordre d'*importance* $\rhd$sur les variables
  sur l'exemple : Vin $\rhd$ Dessert $\rhd$ Plat
- Pour chaque variable X :
  un ordre *local* total strict sur $\underline{X}$
- Pour comparer o et o' :
  on les compare successivement sur les variables dans l'ordre $\rhd$
  jusqu'à la première qui les différencie

Remarque 3.

Expressivité très limité : moins expressif que les utilités additives...

Modèles lexicographiques



- Un ordre d'*importance* $\rhd$sur les variables
  sur l'exemple : Vin $\rhd$ Dessert $\rhd$ Plat
- Pour chaque variable X :
  un ordre *local* total strict sur $\underline{X}$
- Pour comparer o et o' :
  on les compare successivement sur les variables dans l'ordre $\rhd$
  jusqu'à la première qui les différencie

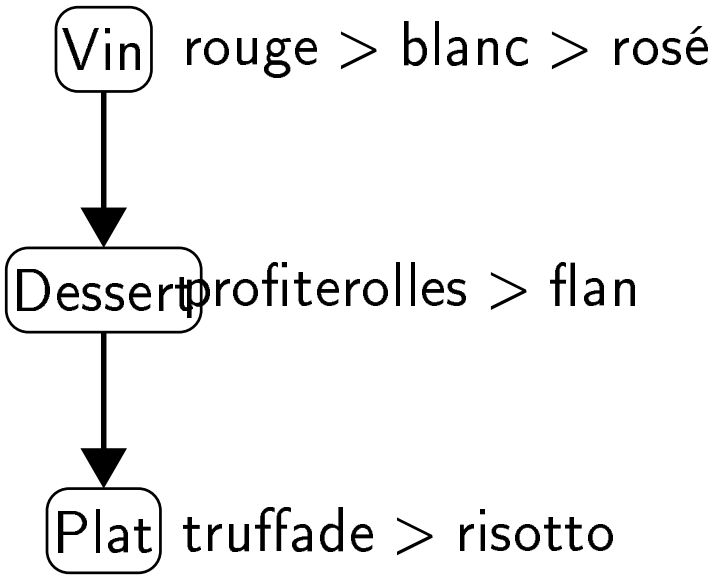
Exemple 2.

Donner un ordre lexicographique qui représente l'ordre de l'exercice 1.

Remarque 3.

Expressivité très limité : moins expressif que les utilités additives...

Modèles lexicographiques



- Un ordre d'*importance* $\rhd$sur les variables
  sur l'exemple : Vin $\rhd$ Dessert $\rhd$ Plat
- Pour chaque variable X :
  un ordre *local* total strict sur $\underline{X}$
- Pour comparer o et o' :
  on les compare successivement sur les variables dans l'ordre $\rhd$
  jusqu'à la première qui les différencie

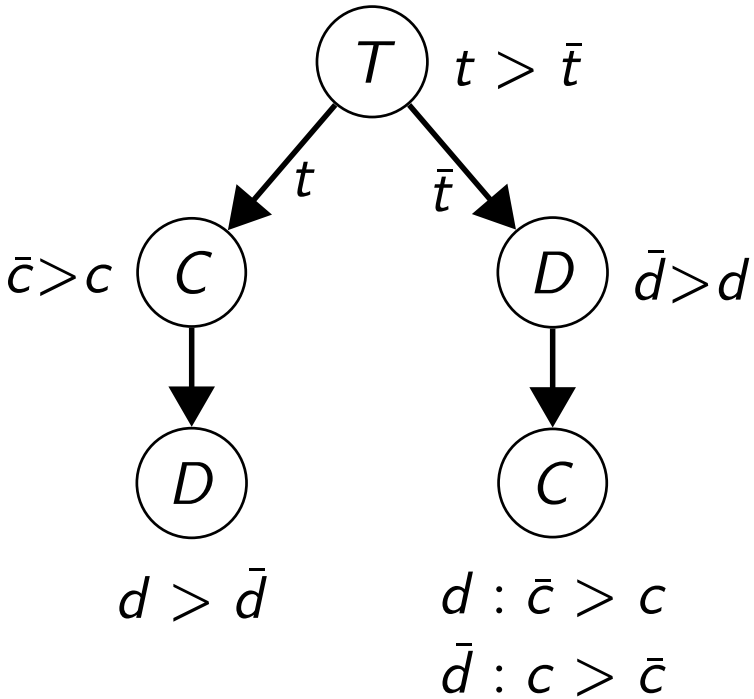
Exercice 3.

Donner une méthode permettant de calculer une utilité additive équivalente à un ordre lexicographique donné.

Remarque 3.

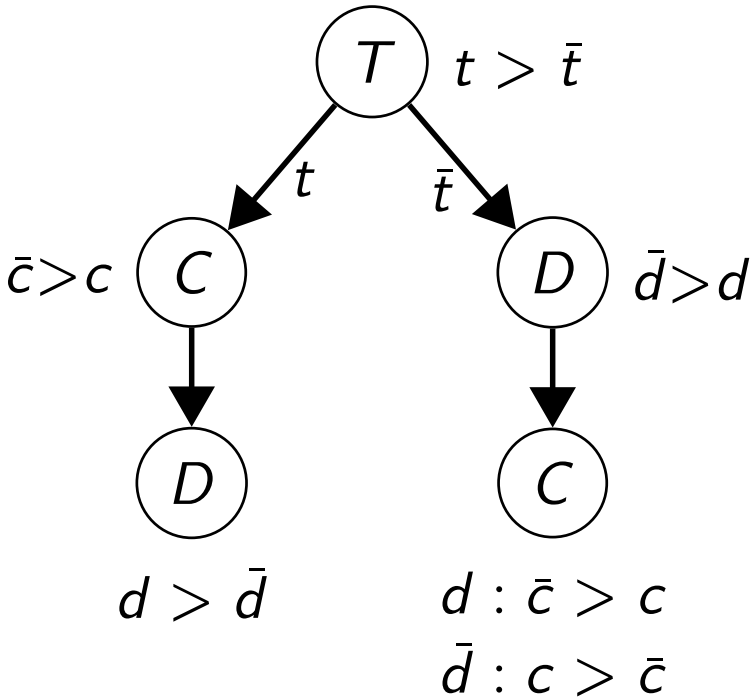
Expressivité très limité : moins expressif que les utilités additives...

Modéles lexic. : Arbres de préf. lexic. conditionnelles



- Un arbre enraciné
- À chaque nœud N :
  - 1 variable X
  - 1 *table de préférences locales conditionnelles* CPT(N)
    = ens. de formules de la forme
    $$u : x_1 > x_2$$
    « *si* u, *alors* x_1 *préféré à* x_2 »
  - 0 enfant (feuille), ou
    1 enfant, ou
    $|\underline{X}|$ enfants (1 pour chaque $x \in \underline{X}$)

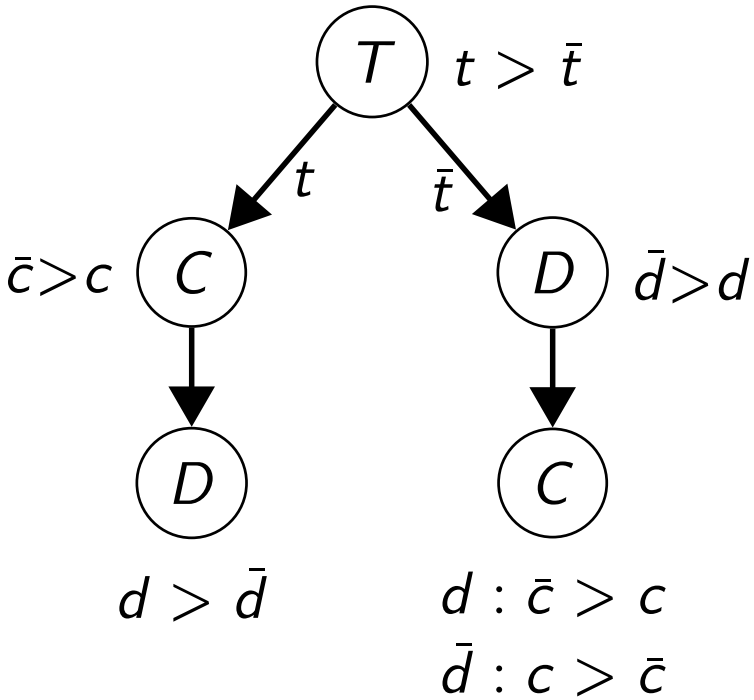
Modéles lexic. : Arbres de préf. lexic. conditionnelles



$cdt \succ \bar{c}d\bar{t}$

- Un arbre enraciné
- À chaque nœud N :
  - 1 variable X
  - 1 *table de préférences locales conditionnelles* CPT(N)
    = ens. de formules de la forme
    $$u : x_1 > x_2$$
    « *si* u, *alors* x_1 *préféré à* x_2 »
  - 0 enfant (feuille), ou
    1 enfant, ou
    $|\underline{X}|$ enfants (1 pour chaque $x \in \underline{X}$)

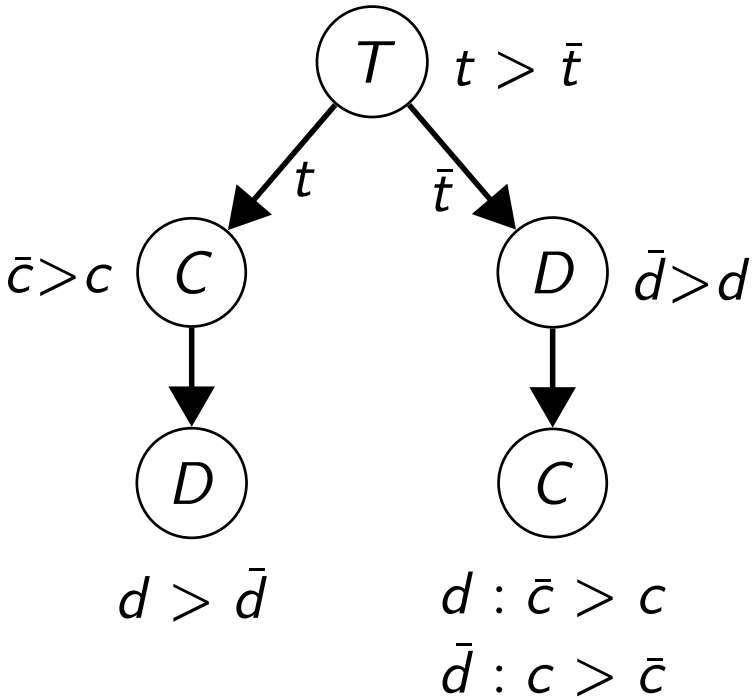
Modéles lexic. : Arbres de préf. lexic. conditionnelles



Alternative préféree : $\bar{c}dt$

- Un arbre enraciné
- À chaque nœud N :
  - 1 variable X
  - 1 *table de préférences locales conditionnelles* CPT(N)
    = ens. de formules de la forme
    $$u : x_1 > x_2$$
    « *si* u, *alors* x_1 *préféré à* x_2 »
  - 0 enfant (feuille), ou
    1 enfant, ou
    $|\underline{X}|$ enfants (1 pour chaque $x \in \underline{X}$)

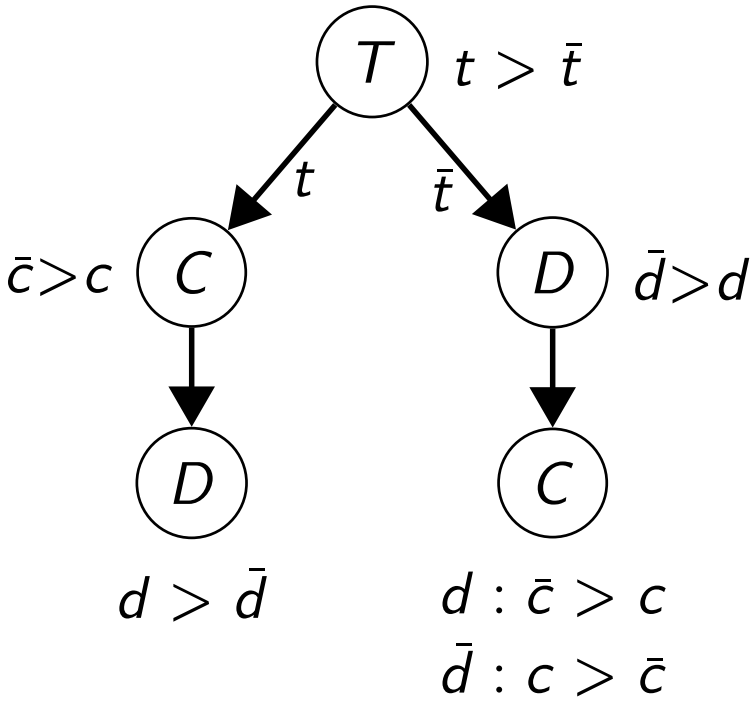
Modéles lexic. : Arbres de préf. lexic. conditionnelles



- Un arbre enraciné
- À chaque nœud N :
  - 1 variable X
  - 1 *table de préférences locales conditionnelles* CPT(N)
    = ens. de formules de la forme
    $$u : x_1 > x_2$$
    « *si* u, *alors* x_1 *préféré à* x_2 »
  - 0 enfant (feuille), ou
    1 enfant, ou
    $|\underline{X}|$ enfants (1 pour chaque $x \in \underline{X}$)

$$\bar{c}dt \succ \bar{c}\bar{d}t \succ cdt \succ c\bar{d}t \succ c\bar{d}\bar{t} \succ \bar{c}\bar{d}\bar{t} \succ \bar{c}d\bar{t} \succ cd\bar{t}$$

Modéles lexic. : Arbres de préf. lexic. conditionnelles

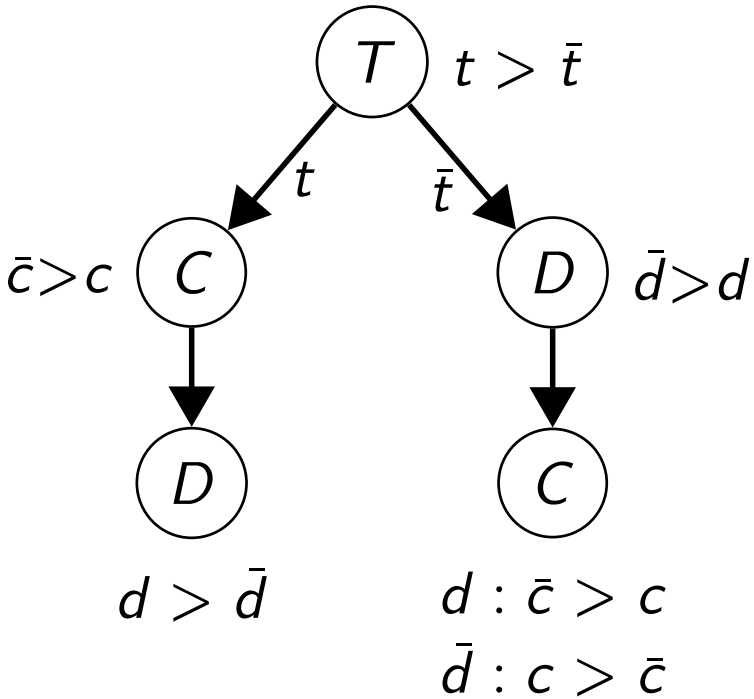


- Un arbre enraciné
- À chaque nœud N :
  - 1 variable X
  - 1 *table de préférences locales conditionnelles* CPT(N)
  - 0 enfant (feuille), ou
    1 enfant, ou
    $|\underline{X}|$ enfants (1 pour chaque $x \in \underline{X}$)

Notations / Précisions

- Anc(N) = ens. des variables des nœuds au-dessus de N
- NonInst(N) = ens. des var. des nœuds au-dessus de N avec 1 enfant
- si $u : x_1 > x_2 \in \text{CPT}(N)$: $u \in \underline{U}$ pour un certain $U \subseteq \text{NonInst}(N)$
- CPT(N) spécifie un order total strict $>$ sur $\underline{X}$, $\forall u \in \underline{\text{NonInst}(N)}$
- Chaque variable apparaît exactement une fois sur chaque branche

Modéles lexic. : Arbres de préf. lexic. conditionnelles

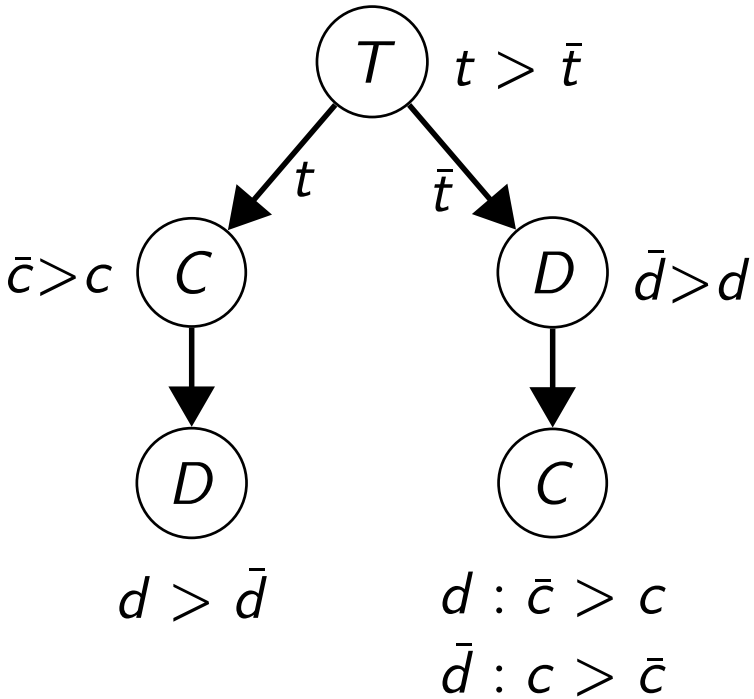


- Un arbre enraciné
- À chaque nœud N :
  - 1 variable X
  - 1 *table de préférences locales conditionnelles* CPT(N)
  - 0 enfant (feuille), ou
    1 enfant, ou
    $|\underline{X}|$ enfants (1 pour chaque $x \in \underline{X}$)

Propriété 1.

Un arbre de préf. lexic. cond. définit un ordre linéaire (total strict) $\succeq$ sur $\underline{\mathcal{X}}$: pour comparer o et o', on les compare successivement sur les variables aux nœuds rencontrés en partant de la racine, jusqu'à la première X qui les différencie. Alors $o \succ o'$ ssi $o[X] > o'[X]$.

Modéles lexic. : Arbres de préf. lexic. conditionnelles



- Un arbre enraciné
- À chaque nœud N :
  - 1 variable X
  - 1 *table de préférences locales conditionnelles* CPT(N)
  - 0 enfant (feuille), ou
    1 enfant, ou
    $|\underline{X}|$ enfants (1 pour chaque $x \in \underline{X}$)

Remarque 4.

Les arbres de préférences lexicographiques permettent de représenter

- des préférences locales *conditionnelles*
- un ordre d'importance *conditionnel*

Exercice 4.

Démontrer qu'aucun arbre de préférences lexicographiques ne permet de représenter l'ordre suivant :

$$abc \succ a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ ab\bar{c} \succ \bar{a}b\bar{c} \succ \bar{a}\bar{b}c \succ \bar{a}bc.$$

où $\mathcal{X} = \{A, B, C\}$, $\underline{A} = \{a, \bar{a}\}$, $\underline{B} = \{b, \bar{b}\}$, $\underline{C} = \{c, \bar{c}\}$

Exercice 4.

Démontrer qu'aucun arbre de préférences lexicographiques ne permet de représenter l'ordre suivant :

$$abc \succ a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ ab\bar{c} \succ \bar{a}b\bar{c} \succ \bar{a}\bar{b}c \succ \bar{a}bc.$$

où $\mathcal{X} = \{A, B, C\}$, $\underline{A} = \{a, \bar{a}\}$, $\underline{B} = \{b, \bar{b}\}$, $\underline{C} = \{c, \bar{c}\}$

$\Rightarrow$ k-arbre de préférences lexicographiques :

- Chaque nœud peut contenir jusqu'à k variables
- Toute variable doit apparaître une et une seule fois sur chaque branche

Modéles lexic. : Arbres de préf. lexic. conditionnelles

Exercice 4.

Démontrer qu'aucun arbre de préférences lexicographiques ne permet de représenter l'ordre suivant :

$$abc \succ a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ ab\bar{c} \succ \bar{a}b\bar{c} \succ \bar{a}\bar{b}c \succ \bar{a}bc.$$

où $\mathcal{X} = \{A, B, C\}$, $\underline{A} = \{a, \bar{a}\}$, $\underline{B} = \{b, \bar{b}\}$, $\underline{C} = \{c, \bar{c}\}$

$\Rightarrow$ k-arbre de préférences lexicographiques :

- Chaque nœud peut contenir jusqu'à k variables
- Toute variable doit apparaître une et une seule fois sur chaque branche

Exercice 5.

Donnez un k-arbre de préférences lexicographiques qui permet de représenter :

$$abc \succ a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ ab\bar{c} \succ \bar{a}b\bar{c} \succ \bar{a}\bar{b}c \succ \bar{a}bc.$$

Exercice 4.

Démontrer qu'aucun arbre de préférences lexicographiques ne permet de représenter l'ordre suivant :

$$abc \succ a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ ab\bar{c} \succ \bar{a}b\bar{c} \succ \bar{a}\bar{b}c \succ \bar{a}bc.$$

où $\mathcal{X} = \{A, B, C\}$, $\underline{A} = \{a, \bar{a}\}$, $\underline{B} = \{b, \bar{b}\}$, $\underline{C} = \{c, \bar{c}\}$

$\Rightarrow$ k-arbre de préférences lexicographiques :

- Chaque nœud peut contenir jusqu'à k variables
- Toute variable doit apparaître une et une seule fois sur chaque branche

Remarque 5.

- Tout ordre total strict sur $\underline{\mathcal{X}}$ peut être représenté par un k-arbre de préf. lexicographique
  au pire avec $k = |\mathcal{X}|$

Modèles logiques

- Une *logique* des préférences
- Grande souplesse de représentation / grande expressivité permet de représenter des relations *partielles*
- Certaines requêtes sont *difficiles*

Exemple 3.

Menu = starter + main course + desert + wine

Exemple 3.

Menu = starter + main course + desert + wine

- profiterolles $\geq$ flan *unconditional, ceteris paribus semantics*

  Between 2 menus identical except for desert,

  I prefer the one with profiterolles to the one with flan

  for instance : soup, risotto, profiterolles, white $\succeq$ soup, risotto, flan, white

Exemple 3.

Menu = starter + main course + desert + wine

- profiterolles $\geq$ flan *unconditional, ceteris paribus semantics*
  for instance : soup, risotto, profiterolles, white $\succeq$ soup, risotto, flan, white
- risotto : red $\geq$ white *unconditional, ceteris paribus semantics*
  Between 2 menus identical except for wine,
  if they both have risotto,
  I prefer the one with red to the one with white

Exemple 3.

Menu = starter + main course + desert + wine

- profiterolles $\geq$ flan *unconditional, ceteris paribus semantics*
  for instance : soup, risotto, profiterolles, white $\succeq$ soup, risotto, flan, white
- risotto : red $\geq$ white *unconditional, ceteris paribus semantics*
- soup|Dessert : risotto $\geq$ truffade
  Between 2 menus identical except for main course,
  and possibly for desert
  if they both have soup,
  I prefer the one with risotto to the one with truffade
  Desert is *less important* than main course when soup is chosen

Modèles logiques : CP - statements

Simple preference statements

$$w \geq w'$$

for $w, w' \in \underline{W}$, $W \subseteq \mathcal{X}$.

Everything else being equal,
I prefer w to w'

Modèles logiques : CP - statements

Simple preference statements

$$w \geq w'$$

for $w, w' \in \underline{W}$, $W \subseteq \mathcal{X}$.

Everything else being equal,
I prefer w to w'

***Ceteris Paribus* semantics : $o \succeq o'$ if**

- $o[W] = w$ and $o'[W] = w'$, and
- $o[Y] = o'[Y]$, $\forall Y \notin W$.

for every $o, o' \in \underline{\mathcal{X}}$,

Modèles logiques : CP - statements

Simple preference statements

$$w \geq w'$$

for $w, w' \in \underline{W}$, $W \subseteq \mathcal{X}$.

Everything else being equal,
I prefer w to w'

***Ceteris Paribus* semantics : $o \succeq o'$ if**

- $o[W] = w$ and $o'[W] = w'$, and
- $o[Y] = o'[Y]$, $\forall Y \notin W$.

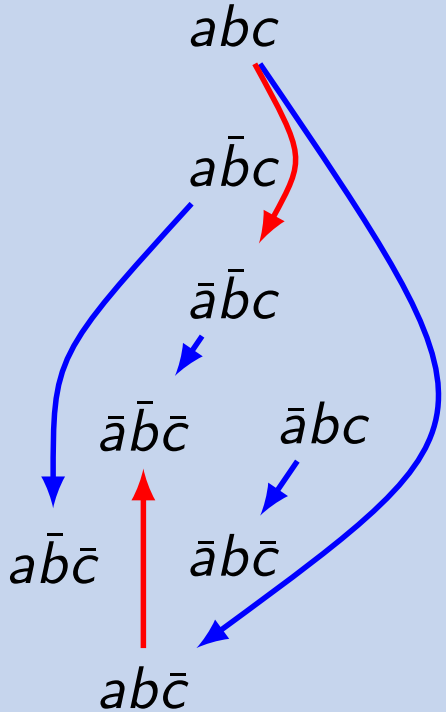
for every $o, o' \in \underline{\mathcal{X}}$,

Caution :

- $\succeq$ for preorder over $\underline{\mathcal{X}}$
- $\geq$ for statements

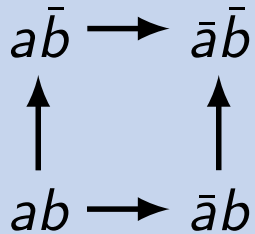
Exemple 4.

$c \geq \bar{c}$, $ab \geq \bar{a}\bar{b}$



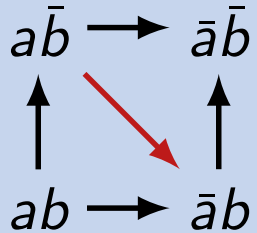
Exemple 5.

$$a \geq \bar{a}, b \geq \bar{b}$$



Exemple 5.

$$a \geq \bar{a}, b \geq \bar{b}, a\bar{b} \geq \bar{a}b$$



Modèles logiques : CP - statements

Conditional Preference statements α propositional formula

$$\alpha : w \geq w'$$

for $w, w' \in \underline{W}$, $W \subseteq \mathcal{X}$, $\text{Var}(\alpha) = U$, $U \cap W = \emptyset$.

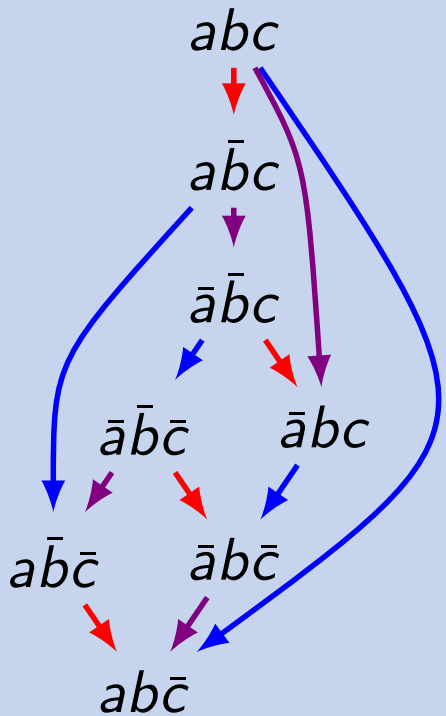
Everything else being equal, for alternatives that satisfy α,
I prefer w to w'.

***Ceteris Paribus* semantics : $o \succeq o'$ if**

- $o[W] = w$ and $o'[W] = w'$, and
- $o[U] = o'[U] \models \alpha$, and
- $o[Y] = o'[Y]$, $\forall Y \notin U \cup W$.

Exemple 6.

$c > \bar{c}$, $c : a > \bar{a}, \bar{c} : \bar{a} > a$, $ac : b > \bar{b}, \bar{a}c : \bar{b} > b, a\bar{c} : \bar{b} > b, \bar{a}\bar{c} : \bar{b} > b$



Modèles logiques : CP - statements

"Full" CP-statements = Cond. Pref. statements with "free" variables

$$\alpha | V : w \geq w'$$

for $w, w' \in \underline{W}$, $W \subseteq \mathcal{X}$, $\mathrm{Var}(\alpha) = U$, with U V, W disjoint.

Everything else being equal, for alternatives that satisfy α,
and whatever the values for V,
I prefer w to w'.

***Ceteris Paribus* semantics with relative importance : $o \succeq o'$ if**

- $o[W] = w$ and $o'[W] = w'$, and
- $o[U] = o'[U] \models \alpha$, and
- $o[Y] = o'[Y]$, $\forall Y \notin U \cup V \cup W$.

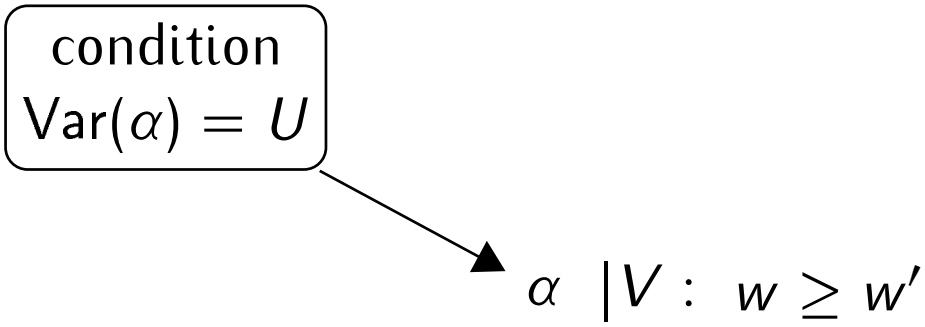
Attributes in V are less important than those in W
(because it is the values of those in W that decide which alternative is preferred)

Modèles logiques : CP - statements

$$\alpha \ |V : \ w \geq w'$$

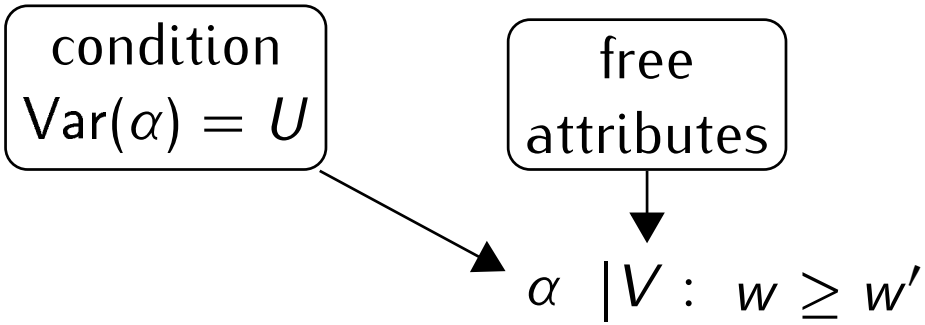
Everything else being equal, for alternatives that satisfy α,
and whatever the values for V,
I prefer w to w'.

Modèles logiques : CP - statements



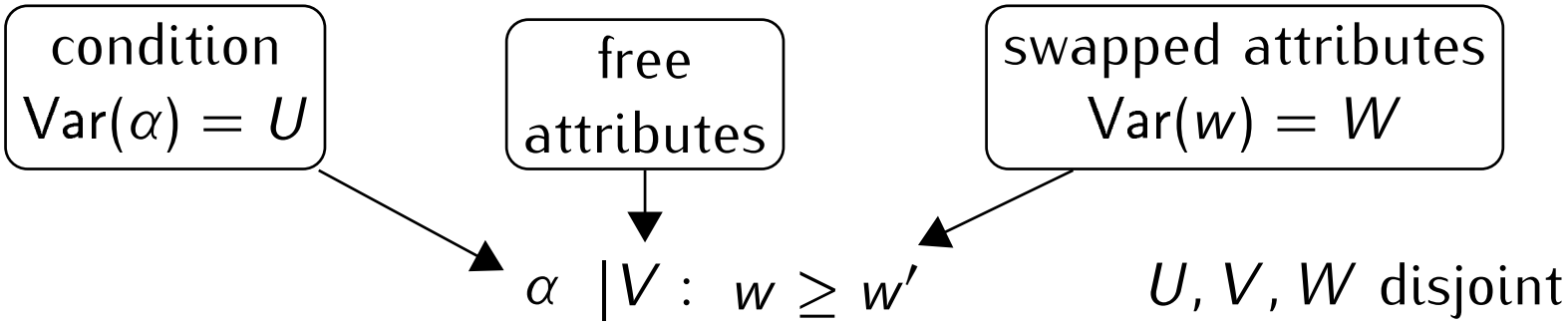
Everything else being equal, for alternatives that satisfy α,
and whatever the values for V,
I prefer w to w'.

Modèles logiques : CP - statements



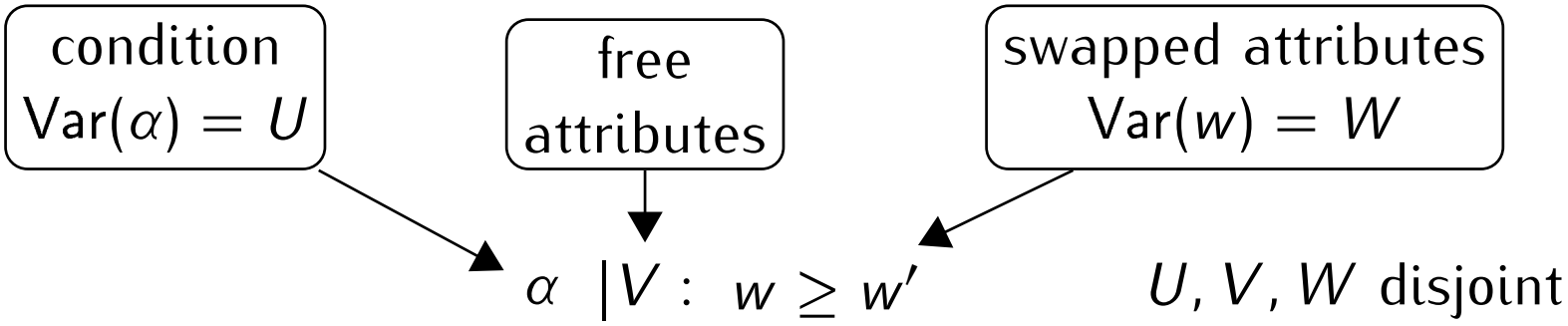
Everything else being equal, for alternatives that satisfy α,
and whatever the values for V,
I prefer w to w'.

Modèles logiques : CP - statements



Everything else being equal, for alternatives that satisfy α,
and whatever the values for V,
I prefer w to w'.

Modèles logiques : CP - statements

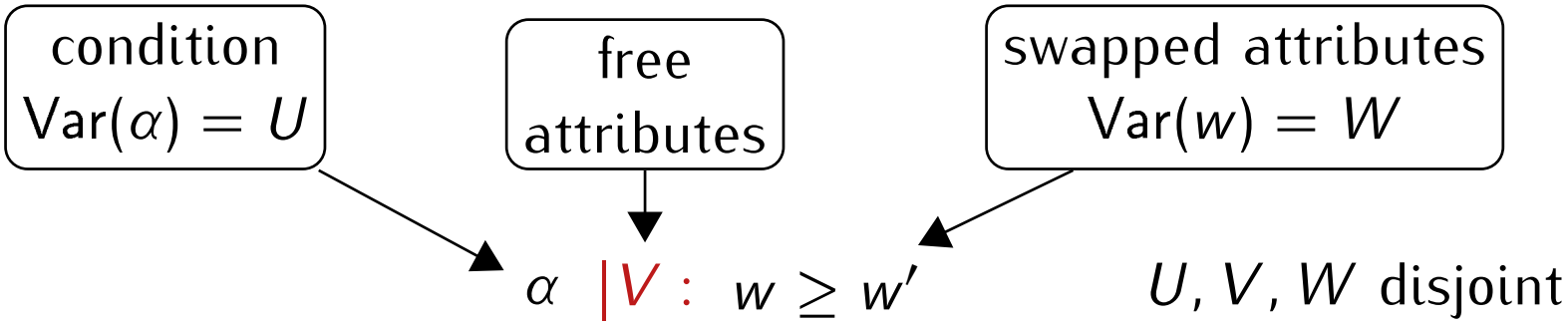


Everything else being equal, for alternatives that satisfy α,
and whatever the values for V,
I prefer w to w'.

***Ceteris Paribus* semantics**

$$[\![\alpha|V : w \geq w']\!] = \left\{ (o, o') \;\middle|\; \begin{array}{l} o[W] = w \text{ and } o'[W] = w', \text{ and} \\ o[U] = o'[U] \models \alpha, \text{ and} \\ o[Y] = o'[Y] \; \forall Y \notin U \cup V \cup W \end{array} \right\}$$

Modèles logiques : CP - statements



Everything else being equal, for alternatives that satisfy α,
and whatever the values for V,
I prefer w to w'.

***Ceteris Paribus* semantics with relative importance**

$$[\![\alpha | V : w \geq w']\!] = \left\{ (o, o') \;\middle|\; \begin{array}{l} o[W] = w \text{ and } o'[W] = w', \text{ and} \\ o[U] = o'[U] \models \alpha, \text{ and} \\ o[Y] = o'[Y] \; \forall Y \notin U \cup V \cup W \end{array} \right\}$$

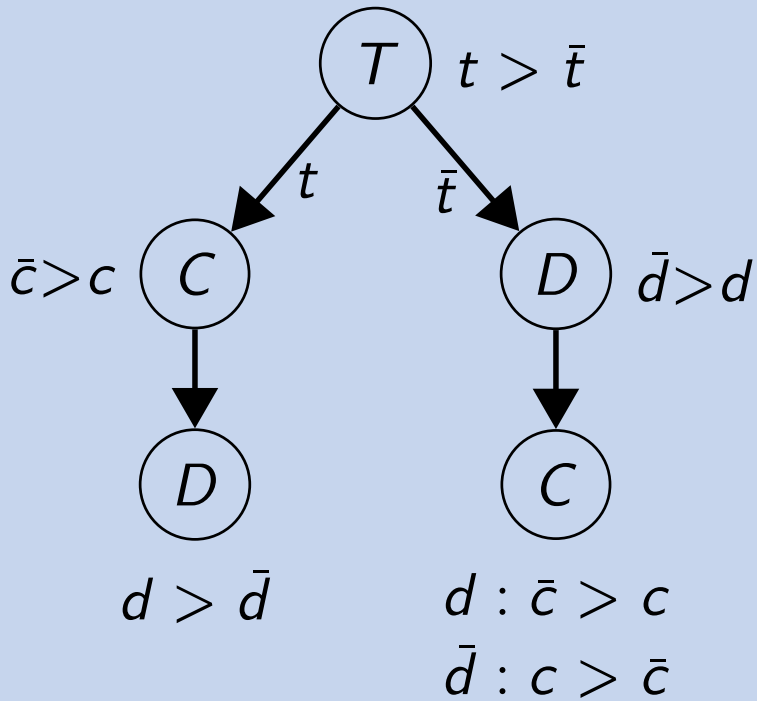
Attributes in V are less important than those in W :
the values of those in W decide which alternative is preferred.

Exercice 6 : (Example A in [Wil11], slightly extended).

Consider planning a holiday, with three choices / attributes : wait til next month ($W = w$) or leave now ($W = \bar{w}$), going to city 1, 2 or 3 ($C = c_1$, $C = c_2$ or $C = c_3$), travelling by plane ($P = p$) or by car ($P = \bar{p}$). Give the set of CP-statemants that represents the following : "I would rather go now, irrespective of the other attributes. All else being equal, I prefer to go to city 3, city 1 being my second best choice. Also, if I go now, I prefer to fly. I prefer flying to city 1 rather than driving to city 3. Finally, if I go later, I prefer to drive, irrespective of the city."

Exemple 7.

Donner un ensemble de CP-statements équivalents à cet arbre de préférences lexicographiques :



(Rappels de) Classes de Complexité

Définition 1.

P = classe des problèmes de décision qu'on sait résoudre en temps polynomial.

(Rappels de) Classes de Complexité

Définition 1.

P = classe des problèmes de décision qu'on sait résoudre en temps polynomial.

Définition 2.

NP est la classe des problèmes de décision pour lesquels on peut *vérifier* une solution en temps polynomial.

(Rappels de) Classes de Complexité

Définition 1.

P = classe des problèmes de décision qu'on sait résoudre en temps polynomial.

Définition 2.

NP est la classe des problèmes de décision pour lesquels on peut *vérifier* une solution en temps polynomial.

Définition 3.

PSPACE est la classe des problèmes de décision qu'on peut résoudre en *espace* polynomial.

(Rappels de) Classes de Complexité

Théorème 1 : Bylander, 1994.

STRIPS planning est PSPACE-complet.

(Rappels de) Classes de Complexité

Théorème 2.

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathit{PSPACE} = \mathit{NPSPACE} \subseteq \mathit{EXPTIME}$$

On sait aussi que $\mathsf{P} \neq \mathit{EXPTIME}$.


Nic Wilson.
Computational techniques for a simple theory of conditional preferences.
Artificial Intelligence, 175 :1053–1091, 2011.