Carte de compilation des diagrammes de décision ordonnés à valeurs réelles

Hélène Fargier[1] Pierre Marquis[2]
Alexandre Niveau[3] Nicolas Schmidt[1,2]

[1] IRIT-CNRS, Univ. Paul Sabatier, Toulouse, France

[2] CRIL-CNRS, Univ. Artois, Lens, France

[3] GREYC-CNRS, Univ. Caen, France

JIAF — 13 juin 2014

Plan

Plan

Exemple introductif

- Problème de configuration interactive de produit : voiture
- Paramètres :
  - moteur — solaire ou à pédales
  - couleur – bleue ou rouge
  - taille – monospace ou biplace
  - autoradio – avec ou sans

Exemple introductif

- Problème de configuration interactive de produit : voiture
- Paramètres :
  - moteur — solaire ou à pédales
  - couleur – bleue ou rouge
  - taille – monospace ou biplace
  - autoradio – avec ou sans
- Règles :
  - les voitures à pédales doivent être rouges
  - les panneaux solaires ne tiennent pas sur la biplace
  - les monospaces ont tous un autoradio de série
- Le programme doit pouvoir dire si chaque choix respecte les règles

Problème

- Produit configurable $\rightarrow$ problème de satisfaction de contraintes (CSP)

$$\begin{cases} \textit{moteur} = \text{pedales} & \rightarrow & \textit{couleur} = \text{rouge} \\ \textit{moteur} = \text{solaire} & \rightarrow & \textit{taille} > \text{biplace} \\ \textit{taille} = \text{biplace} & \vee & \textit{radio} = \text{avec} \end{cases}$$

  - chaque variable correspond à un choix
  - chaque solution correspond à une configuration admissible

Problème

- Produit configurable $\rightarrow$ problème de satisfaction de contraintes (CSP)

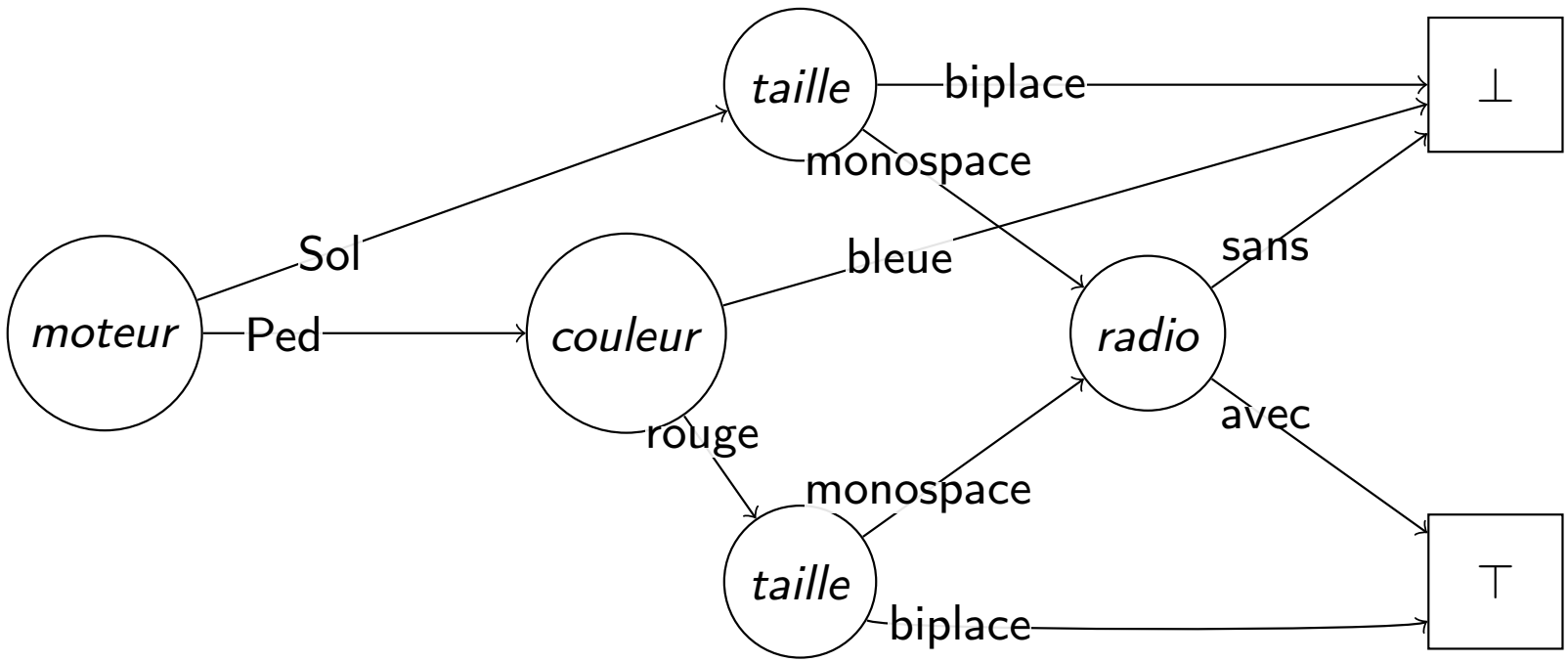
$$\begin{cases} \mathit{moteur} = \text{pedales} & \rightarrow & \mathit{couleur} = \text{rouge} \\ \mathit{moteur} = \text{solaire} & \rightarrow & \mathit{taille} > \text{biplace} \\ \mathit{taille} = \text{biplace} & \vee & \mathit{radio} = \text{avec} \end{cases}$$

  - chaque variable correspond à un choix
  - chaque solution correspond à une configuration admissible
- Processus de configuration :
  - chaque choix fixe la valeur d'une certaine variable
  - le CSP est-il encore cohérent ?
- Problème NP-complet... mais l'utilisateur ne veut pas attendre trop longtemps après chaque choix

Une solution : la compilation

- Le CSP est une partie fixe du problème

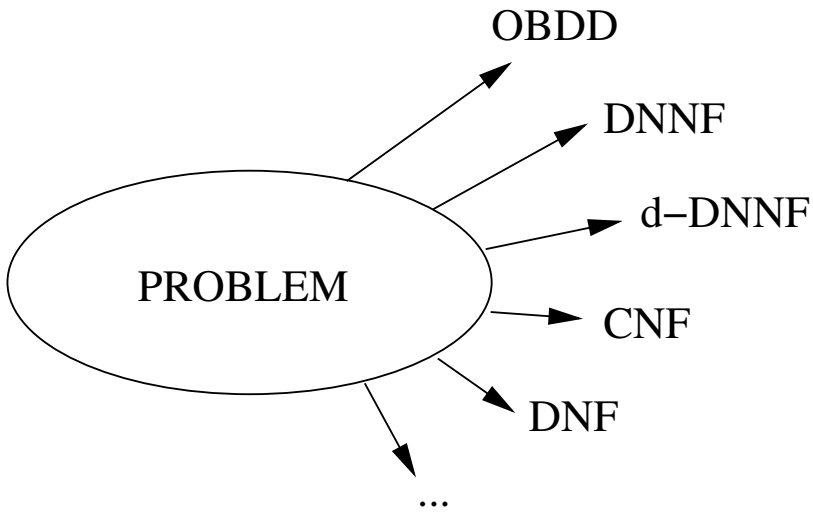
→ On compile ce CSP en une structure de données bien adaptée, par exemple un OBDD :



- Assigner des valeurs aux variables (conditionnement) et vérifier la cohérence sont des opérations polynomiales sur les OBDDs

→ l'attente de l'utilisateur est réduite

Choix d'un langage de compilation



- Quel est le meilleur langage pour mon application ?

→ On peut utiliser la carte de compilation [Darwiche and Marquis, 2002]

- Compare les langages selon deux critères :
  1. efficacité des opérations
  2. concision

Carte de compilation : opérations

- Toutes les manipulations en ligne se ramènent à des requêtes et transformations élémentaires

L	CO (consistency)	VA (validity)	CE (clause entailmt.)	IM (implicant check)	EQ (equivalence)	SE (entailment)	CT (model count)	ME (model enum.)
NNF	○	○	○	○	○	○	○	○
DNNF	√	○	√	○	○	○	○	√
BDD	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	○	√	√
DNF	√	○	√	○	○	○	○	√
CNF	○	√	○	√	○	○	○	○

L	CD (conditioning)	FO (forgetting)	SFO (single forg.)	∧C (conjunction)	∧BC (bounded conj.)	∨C (disjunction)	∨BC (bounded disj.)	¬C (negation)
NNF	√	○	√	√	√	√	√	√
DNNF	√	√	√	○	○	√	√	○
BDD	√	○	√	√	√	√	√	√
FBDD	√	•	○	•	○	•	○	√
OBDD	√	•	√	•	○	•	○	√
DNF	√	√	√	•	√	√	√	•
CNF	√	○	√	√	√	•	√	•

√ polynomial
○ non polynomial sauf si P = NP
• non polynomial

Carte de compilation : opérations

- Toutes les manipulations en ligne se ramènent à des requêtes et transformations élémentaires

L	CO (consistency)	VA (validity)	CE (clause entailmt.)	IM (implicant check)	EQ (equivalence)	SE (entailment)	CT (model count)	ME (model enum.)
NNF	○	○	○	○	○	○	○	○
DNNF	√	○	√	○	○	○	○	√
BDD	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	○	√	√
DNF	√	○	√	○	○	○	○	√
CNF	○	√	○	√	○	○	○	○

L	CD (conditioning)	FO (forgetting)	SFO (single forg.)	∧C (conjunction)	∧BC (bounded conj.)	∨C (disjunction)	∨BC (bounded disj.)	¬C (negation)
NNF	√	○	√	√	√	√	√	√
DNNF	√	√	√	○	○	√	√	○
BDD	√	○	√	√	√	√	√	√
FBDD	√	●	○	●	○	●	○	√
OBDD	√	●	√	●	○	●	○	√
DNF	√	√	√	●	√	√	√	●
CNF	√	○	√	√	√	●	√	●

√ polynomial
○ non polynomial sauf si P = NP
● non polynomial

Carte de compilation : opérations

- Toutes les manipulations en ligne se ramènent à des requêtes et transformations élémentaires

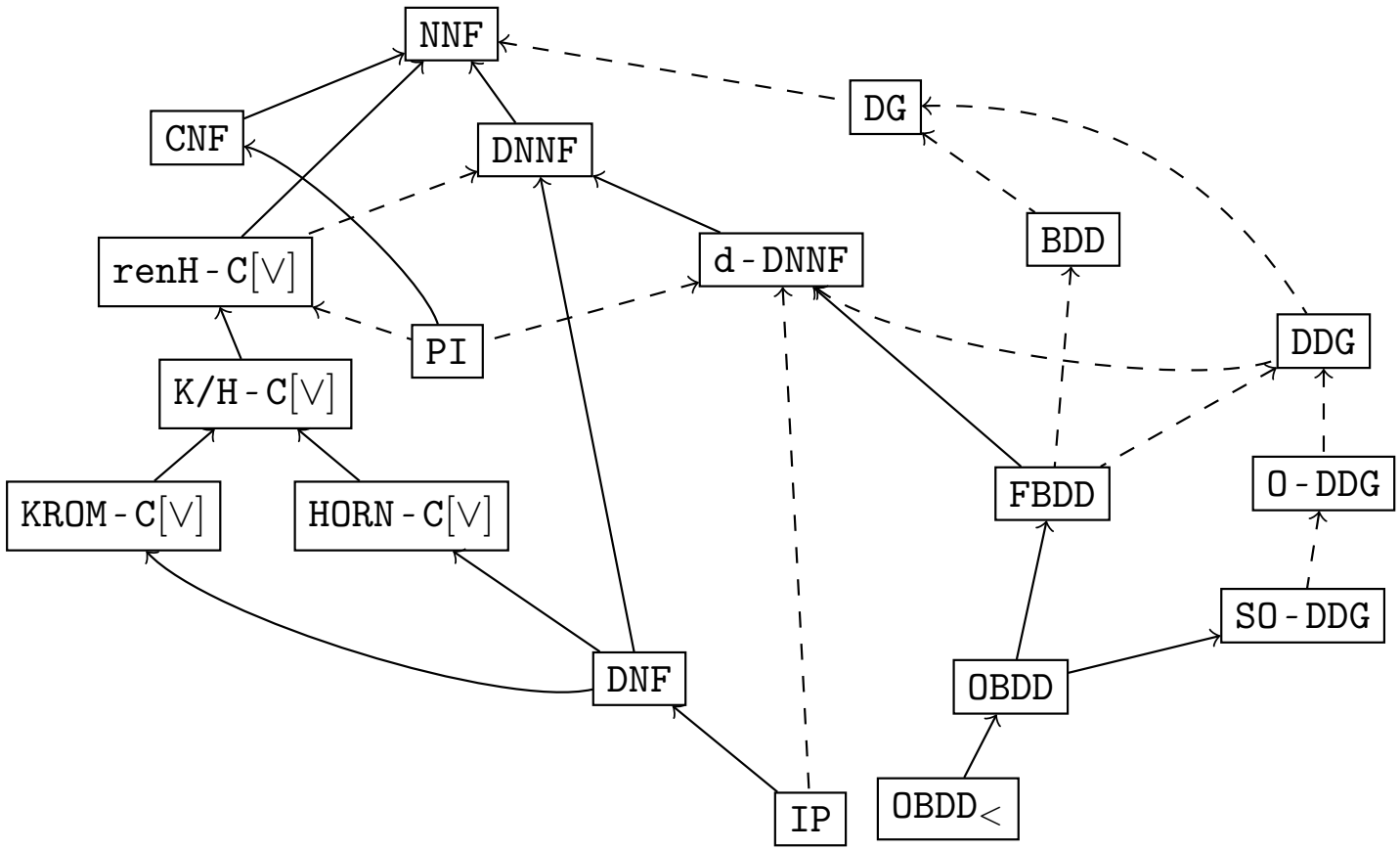
L	CO (consistency)	VA (validity)	CE (clause entailmt.)	IM (implicant check)	EQ (equivalence)	SE (entailment)	CT (model count)	ME (model enum.)
NNF	○	○	○	○	○	○	○	○
DNNF	√	○	√	○	○	○	○	√
BDD	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	○	√	√
DNF	√	○	√	○	○	○	○	√
CNF	○	√	○	√	○	○	○	○

L	CD (conditioning)	FO (forgetting)	SFO (single forg.)	∧C (conjunction)	∧BC (bounded conj.)	∨C (disjunction)	∨BC (bounded disj.)	¬C (negation)
NNF	√	○	√	√	√	√	√	√
DNNF	√	√	√	○	○	√	√	○
BDD	√	○	√	√	√	√	√	√
FBDD	√	•	○	•	○	•	○	√
OBDD	√	•	√	•	○	•	○	√
DNF	√	√	√	•	√	√	√	•
CNF	√	○	√	√	√	•	√	•

√ polynomial
○ non polynomial sauf si P = NP
• non polynomial

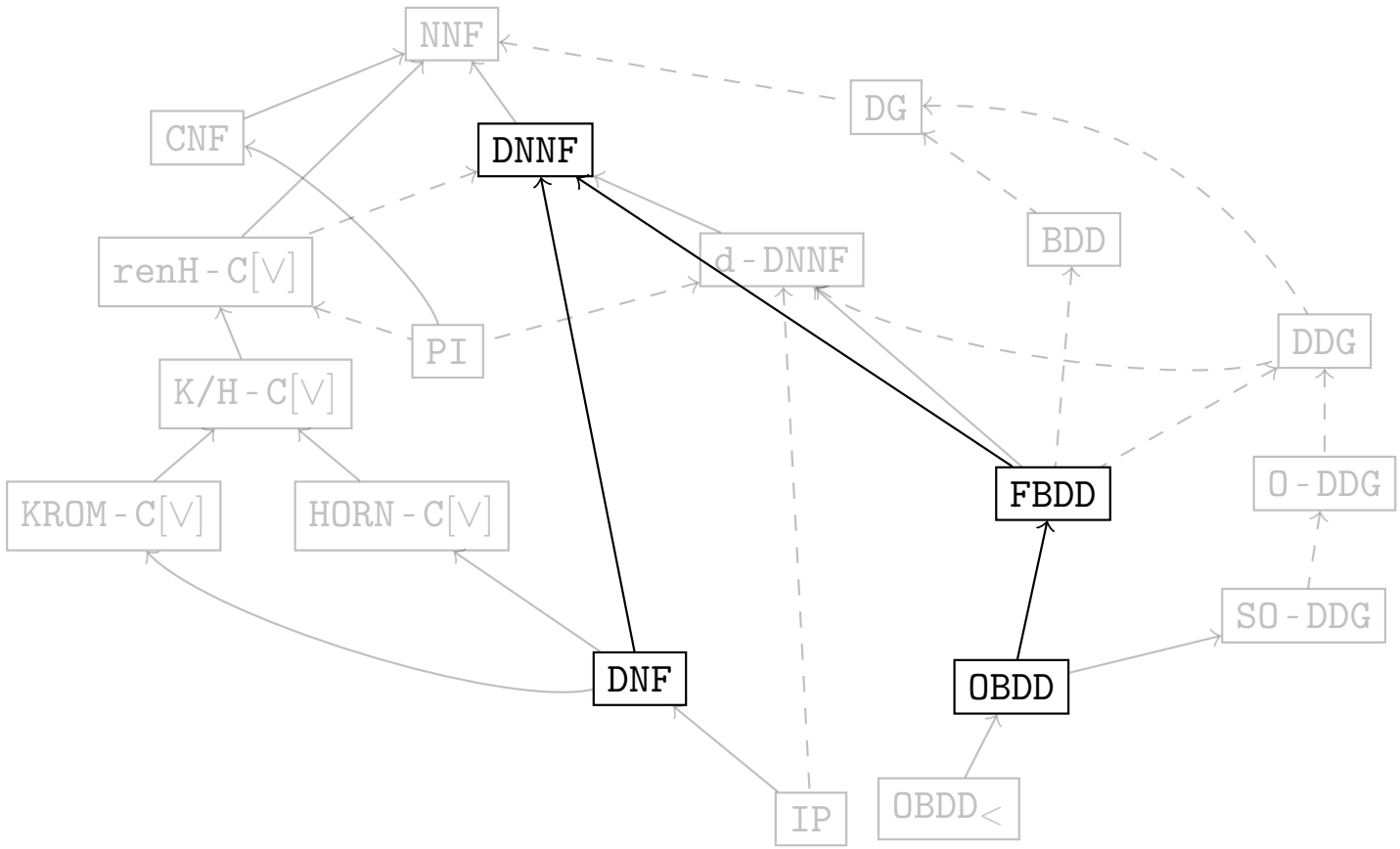
Carte de compilation : concision

- Relation de concision ($\leq_s$) : ordonne les langages
- $L_1 \leq_s L_2$ signifie « L_1 est au moins aussi concis que L_2 »



Carte de compilation : concision

- Relation de concision ($\leq_s$) : ordonne les langages
- $L_1 \leq_s L_2$ signifie « L_1 est au moins aussi concis que L_2 »



Plan

Valeurs non booléennes

- Nombreuses applications en IA nécessitent de manipuler des fonctions à valeurs non booléennes
  - fonctions de coût, d'utilité
  - distributions de probabilité
  - bases de connaissances pondérées...
- CSP valués, GAI-nets, réseaux bayésiens : optimisation difficile

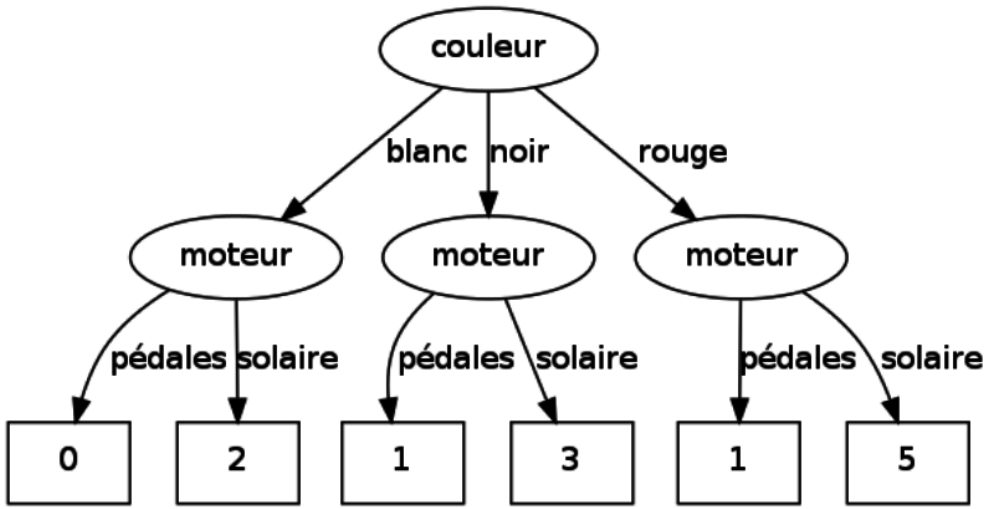
$\rightarrow$ Même problème que précédemment pour les applications interactives

- Ex. pour la configuration : prix maximal de la voiture en cours de configuration

$\rightarrow$ Compilation vers un langage approprié

ADDs

- ADDs, *algebraic decision diagrams* [Bahar et al., 1993]
- Comme les OBDDs, mais les feuilles ont des valeurs dans un ensemble $\mathcal{V}$



$(\mathcal{V} = \mathbb{R}^+)$

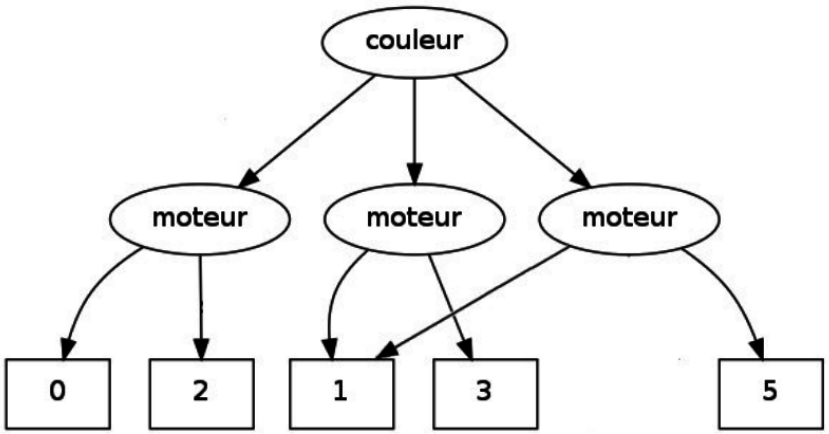
- Optimisation triviale

SLDDs

- Problème des ADDs : une feuille par valeur
- Idée : faire remonter les valeurs sur les arcs, pour les factoriser
- Valeur d'un chemin = agrégation des valeurs rencontrées

$\rightarrow$ SLDD, semiring-labeled decision diagram [Wilson, 2005]

Exemple en configuration :
$\mathcal{V} = \mathbb{R}^+$, agrégation par la somme
$\rightarrow$ langage $\mathtt{SLDD}_+$

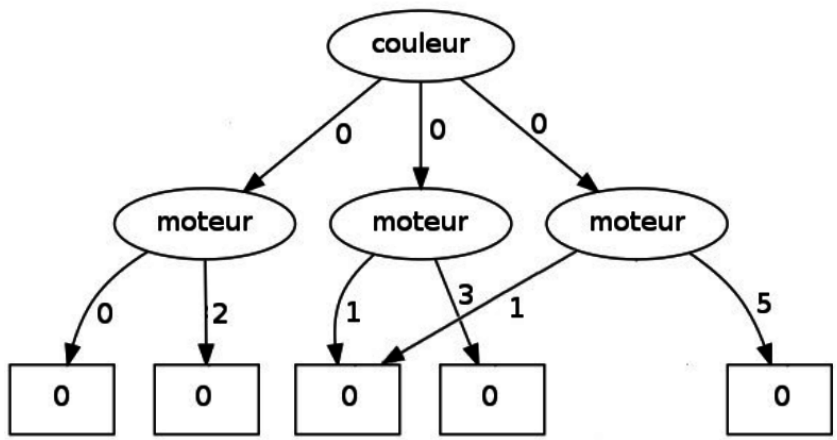


SLDDs

- Problème des ADDs : une feuille par valeur
- Idée : faire remonter les valeurs sur les arcs, pour les factoriser
- Valeur d'un chemin = agrégation des valeurs rencontrées

$\rightarrow$ SLDD, semiring-labeled decision diagram [Wilson, 2005]

Exemple en configuration :
$\mathcal{V} = \mathbb{R}^+$, agrégation par la somme
$\rightarrow$ langage $\mathtt{SLDD}_+$

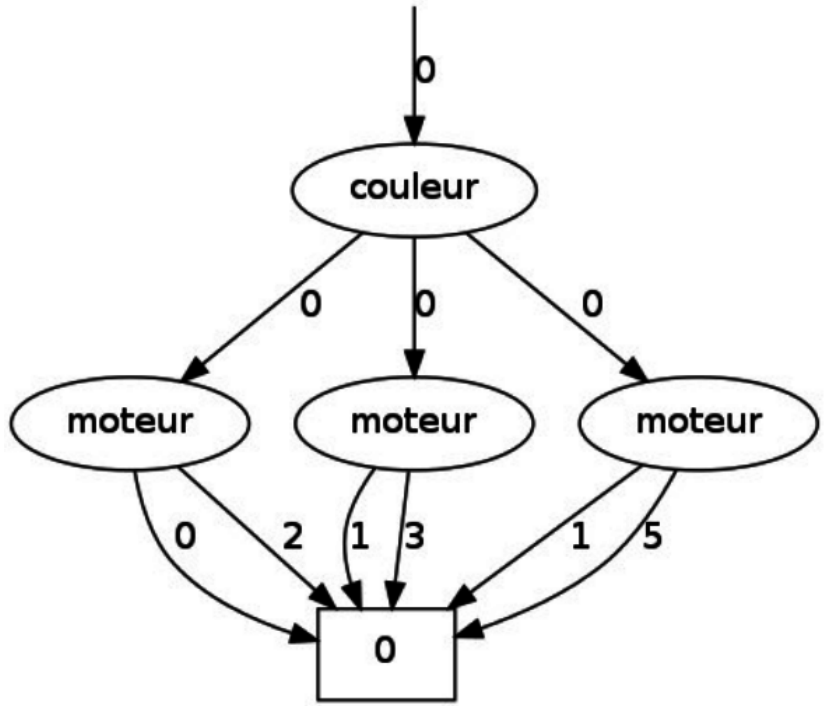


SLDDs

- Problème des ADDs : une feuille par valeur
- Idée : faire remonter les valeurs sur les arcs, pour les factoriser
- Valeur d'un chemin = agrégation des valeurs rencontrées

→ SLDD, semiring-labeled decision diagram [Wilson, 2005]

Exemple en configuration :
$\mathcal{V} = \mathbb{R}^+$, agrégation par la somme
→ langage $\mathtt{SLDD}_+$

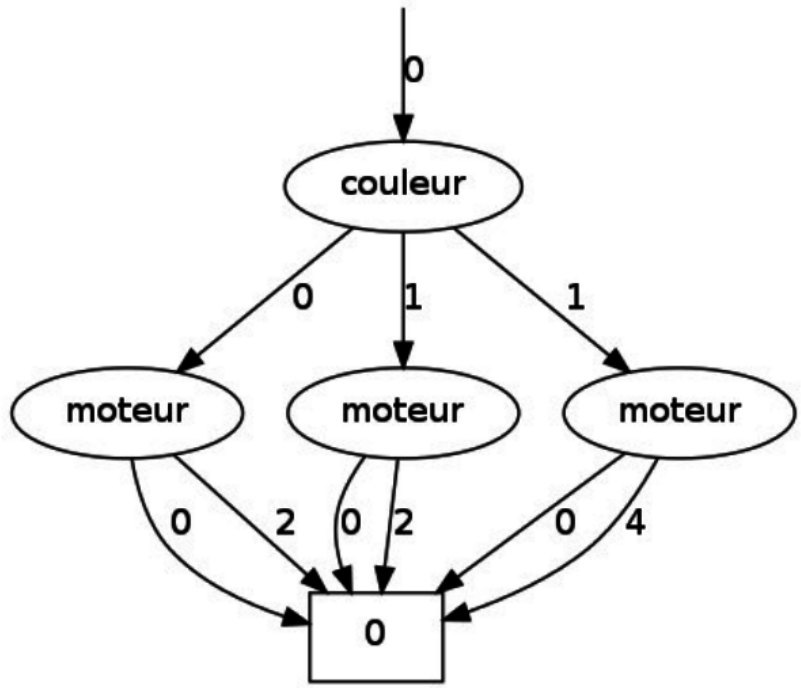


SLDDs

- Problème des ADDs : une feuille par valeur
- Idée : faire remonter les valeurs sur les arcs, pour les factoriser
- Valeur d'un chemin = agrégation des valeurs rencontrées

→ SLDD, semiring-labeled decision diagram [Wilson, 2005]

Exemple en configuration :
$\mathcal{V} = \mathbb{R}^+$, agrégation par la somme
→ langage $\mathtt{SLDD}_+$

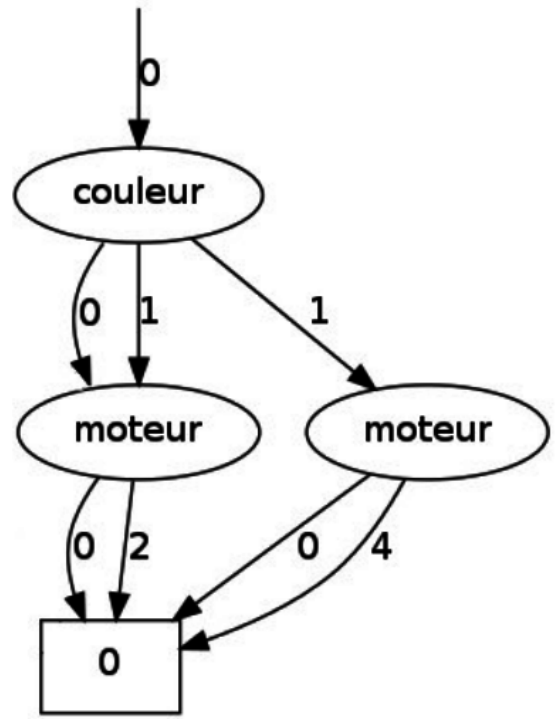


SLDDs

- Problème des ADDs : une feuille par valeur
- Idée : faire remonter les valeurs sur les arcs, pour les factoriser
- Valeur d'un chemin = agrégation des valeurs rencontrées

→ SLDD, semiring-labeled decision diagram [Wilson, 2005]

Exemple en configuration :
$\mathcal{V} = \mathbb{R}^+$, agrégation par la somme
→ langage $\mathtt{SLDD}_+$

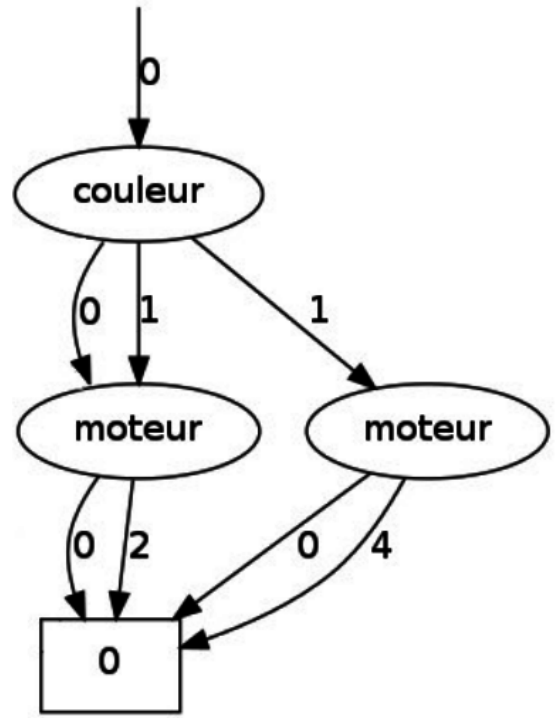


SLDDs

- Problème des ADDs : une feuille par valeur
- Idée : faire remonter les valeurs sur les arcs, pour les factoriser
- Valeur d'un chemin = agrégation des valeurs rencontrées

→ SLDD, semiring-labeled decision diagram [Wilson, 2005]

Exemple en configuration :
$\mathcal{V} = \mathbb{R}^+$, agrégation par la somme
→ langage $\mathtt{SLDD}_+$



- Autre possibilité pour $\mathcal{V} = \mathbb{R}^+$: agrégation par le produit → langage $\mathtt{SLDD}_\times$ → pour les distributions de probabilités

AADDs

- Encore une autre variante de SLDD pour $\mathcal{V} = \mathbb{R}^+$: on agrège à la fois par la somme et le produit

$\rightarrow$ AADD, affine ADDs : deux coefficients sur chaque arc a, un additif et un multiplicatif $\langle q, f \rangle$

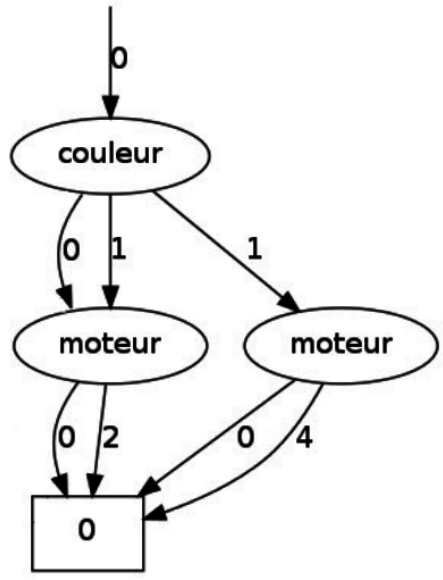
- Chemin commençant par a : valeur $q + f \times V_{\text{rec}}$, avec V_{rec} la valeur de la suite du chemin

AADDs

- Encore une autre variante de SLDD pour $\mathcal{V} = \mathbb{R}^+$: on agrège à la fois par la somme et le produit

$\rightarrow$ AADD, affine ADDs : deux coefficients sur chaque arc a, un additif et un multiplicatif $\langle q, f \rangle$

- Chemin commençant par a : valeur $q + f \times V_{\text{rec}}$, avec V_{rec} la valeur de la suite du chemin

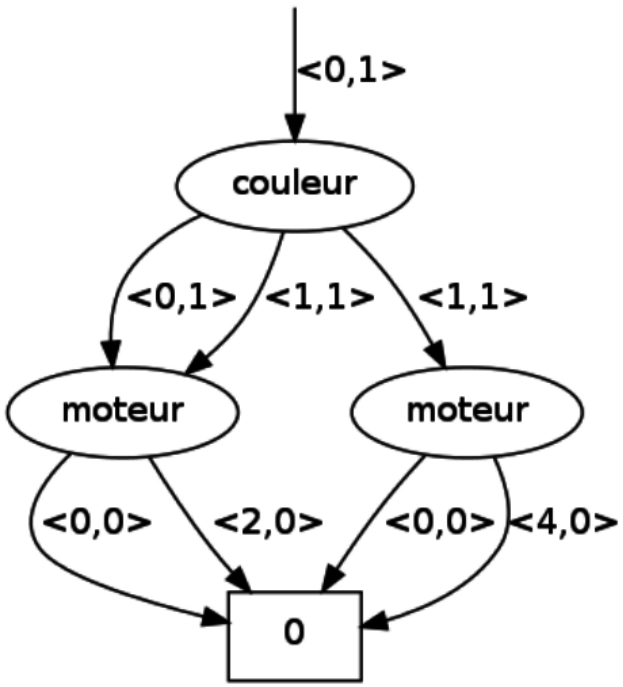


AADDs

- Encore une autre variante de SLDD pour $\mathcal{V} = \mathbb{R}^+$: on agrège à la fois par la somme et le produit

$\rightarrow$ AADD, affine ADDs : deux coefficients sur chaque arc a, un additif et un multiplicatif $\langle q, f \rangle$

- Chemin commençant par a : valeur $q + f \times V_{\text{rec}}$, avec V_{rec} la valeur de la suite du chemin

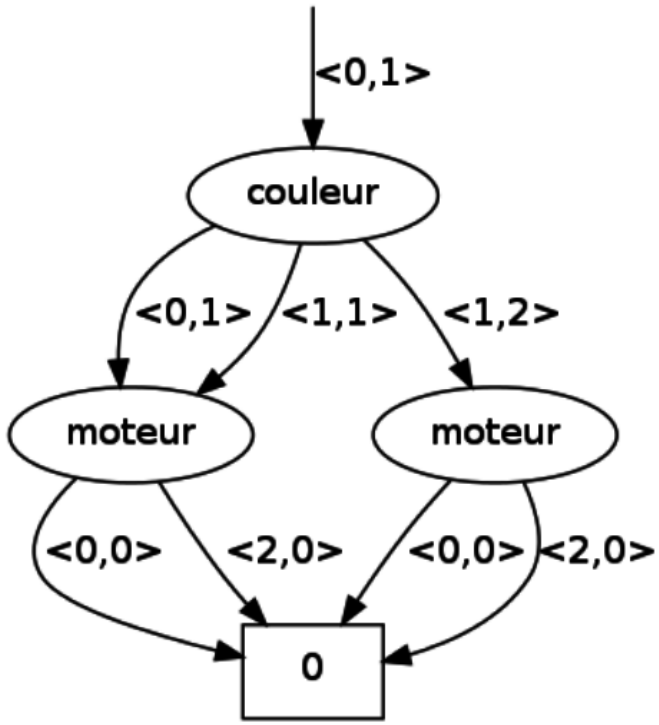


AADDs

- Encore une autre variante de SLDD pour $\mathcal{V} = \mathbb{R}^+$: on agrège à la fois par la somme et le produit

$\rightarrow$ AADD, affine ADDs : deux coefficients sur chaque arc a, un additif et un multiplicatif $\langle q, f \rangle$

- Chemin commençant par a : valeur $q + f \times V_{\text{rec}}$, avec V_{rec} la valeur de la suite du chemin

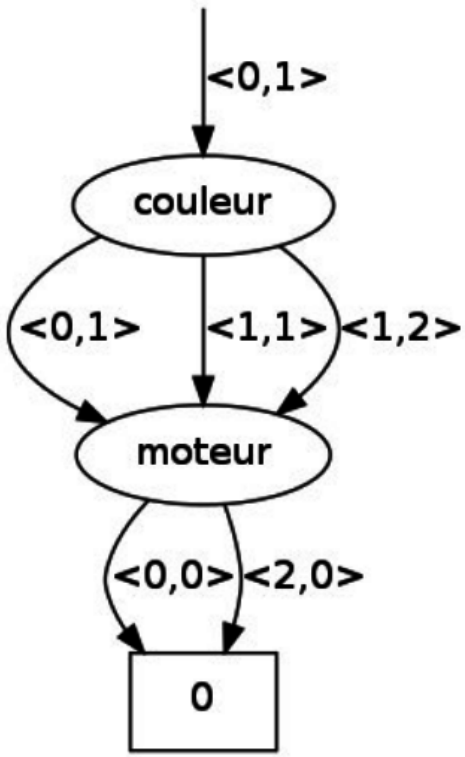


AADDs

- Encore une autre variante de SLDD pour $\mathcal{V} = \mathbb{R}^+$: on agrège à la fois par la somme et le produit

$\rightarrow$ AADD, affine ADDs : deux coefficients sur chaque arc a, un additif et un multiplicatif $\langle q, f \rangle$

- Chemin commençant par a : valeur $q + f \times V_{\text{rec}}$, avec V_{rec} la valeur de la suite du chemin

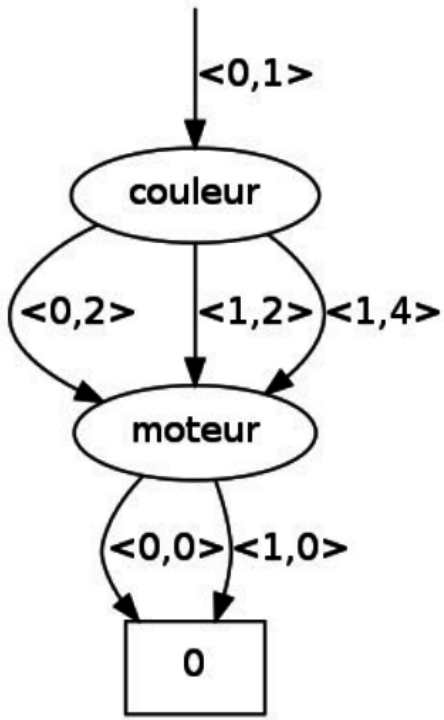


AADDs

- Encore une autre variante de SLDD pour $\mathcal{V} = \mathbb{R}^+$: on agrège à la fois par la somme et le produit

$\rightarrow$ AADD, affine ADDs : deux coefficients sur chaque arc a, un additif et un multiplicatif $\langle q, f \rangle$

- Chemin commençant par a : valeur $q + f \times V_{\text{rec}}$, avec V_{rec} la valeur de la suite du chemin

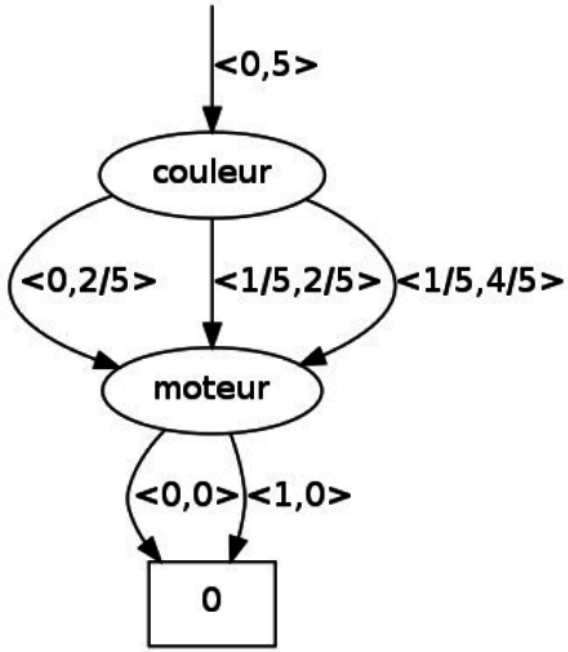


AADDs

- Encore une autre variante de SLDD pour $\mathcal{V} = \mathbb{R}^+$: on agrège à la fois par la somme et le produit

$\rightarrow$ AADD, affine ADDs : deux coefficients sur chaque arc a, un additif et un multiplicatif $\langle q, f \rangle$

- Chemin commençant par a : valeur $q + f \times V_{\text{rec}}$, avec V_{rec} la valeur de la suite du chemin

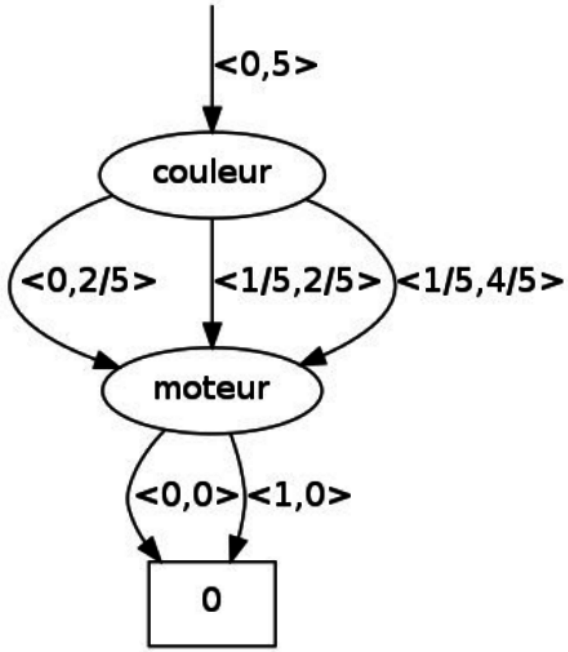


AADDs

- Encore une autre variante de SLDD pour $\mathcal{V} = \mathbb{R}^+$: on agrège à la fois par la somme et le produit

$\rightarrow$ AADD, affine ADDs : deux coefficients sur chaque arc a, un additif et un multiplicatif $\langle q, f \rangle$

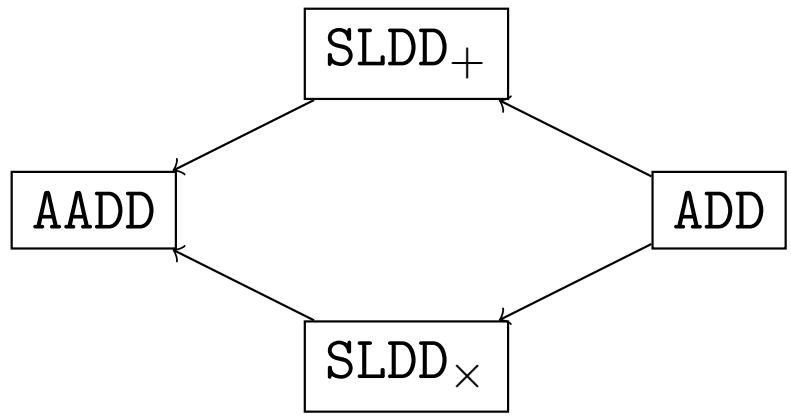
- Chemin commençant par a : valeur $q + f \times V_{\text{rec}}$, avec V_{rec} la valeur de la suite du chemin
- Conditions de normalisation $\rightarrow$ valeur de tout chemin d'un nœud à la feuille $\in [0, 1]$; extrema lisibles sur l'*offset* de la racine



Plan

Carte des VDDs dans $\mathbb{R}^+$: concision

- On se restreint aux langages ADD sur $\mathbb{R}^+$, SLDD_+, $\text{SLDD}_\times$ et AADD.
- Concision entre les 4 langages : étudiée précédemment [Fargier et al., 2013]



- Dans cet article, on traite la partie requêtes et transformations

Choix des opérations

- Il faut commencer par décider quelles sont les requêtes et transformations pertinentes pour comparer les VDDs
- Cadre général, non restreint à $\mathcal{V} = \mathbb{R}^+$
  $\rightarrow$ opérations utilisables dans d'autres contextes
- Les seules qui nécessitent une hypothèse sur $\mathcal{V}$: requêtes d'optimisation $\mathbf{OPT}_{\mathsf{min}}$ et $\mathbf{OPT}_{\mathsf{max}}$, qui visent à renvoyer en temps polynomial la valuation optimale d'une formule
  $\rightarrow$ nécessitent que $\mathcal{V}$ soit totalement ordonné
- Formellement : un langage $\mathtt{L}$ satisfait $\mathbf{OPT}_{\mathsf{min}}$ s'il existe un algorithme polynomial associant à toute formule α de $\mathtt{L}$, la valeur $\min_{\vec{x}} f_\alpha^{\mathtt{L}}(\vec{x})$.

Coupes

La plupart des autres requêtes s'appuient sur des coupes

Soit f une fonction à valeurs dans $\mathcal{V}$, $\preceq$ un préordre sur $\mathcal{V}$ et $\gamma \in \mathcal{V}$, on définit les ensembles suivants :

- $CUT^{\preceq\gamma}(f) = \{ \vec{x} \mid f(\vec{x}) \preceq \gamma \}$
  $\rightarrow$ les voitures de moins de 10 000 euros
- $CUT^{\sim\gamma}(f) = \{ \vec{x} \mid f(\vec{x}) \sim \gamma \}$
  $\rightarrow$ les voitures qui coûtent exactement 10 000 euros
- $CUT^{\min}(f) = \{ \vec{x}^* \mid \forall \vec{x}, \neg(f(\vec{x}) \prec f(\vec{x}^*)) \}$
  $\rightarrow$ les voitures les moins chères

Requêtes sur les coupes

Coupe $\approx$ ensemble de « modèles »

$\rightarrow$ mêmes requêtes qu'en booléen, mais paramétrées

- $\mathbf{CT}_{\mathsf{min}}$: compter les éléments minimaux pour $\preceq$ (renvoyer le cardinal de $CUT^{\mathsf{min}}(f_\alpha^{\mathtt{L}})$)
- Cohérence partielle $\mathbf{CO}_{\sim\gamma}$: indiquer si $\exists \vec{x}, f_\alpha^{\mathtt{L}}(\vec{x}) \sim \gamma$ (indiquer si $CUT^{\sim\gamma}(f_\alpha^{\mathtt{L}}) \neq \varnothing$)

  $\rightarrow$ y a-t-il une voiture valant exactement 10 000 euros ?
- $\mathbf{MX}_{\preceq\gamma}$, $\mathbf{ME}_{\preceq\gamma}$: exhiber un $\vec{x}$, lister tous les $\vec{x}$ vérifiant $f_\alpha^{\mathtt{L}}(\vec{x}) \preceq \gamma$

  $\rightarrow$ quelles sont les voitures de moins de 10 000 euros ?

. . .et les autres combinaisons

Carte pour les requêtes

Requête	ADD	SLDD_+	$\text{SLDD}_\times$	AADD	VCSP_+
EQ	√	√	√	√	?
SE	√	√	√	?	○
$\textbf{OPT}_{\text{min}}$	√	√	√	√	○
$\textbf{MX}_{\text{min}}$ / $\textbf{ME}_{\text{min}}$	√	√	√	√	○
$\textbf{CT}_{\text{min}}$	√	√	√	√	○
$\textbf{CO}_{\sim\gamma}$ / $\textbf{MX}_{\sim\gamma}$ / $\textbf{ME}_{\sim\gamma}$	√	○	○	○	○
$\textbf{CO}_{\preceq\gamma}$ / $\textbf{MX}_{\preceq\gamma}$ / $\textbf{ME}_{\preceq\gamma}$	√	√	√	√	○
$\textbf{CT}_{\sim\gamma}$ / $\textbf{CT}_{\preceq\gamma}$	√	○	○	○	○

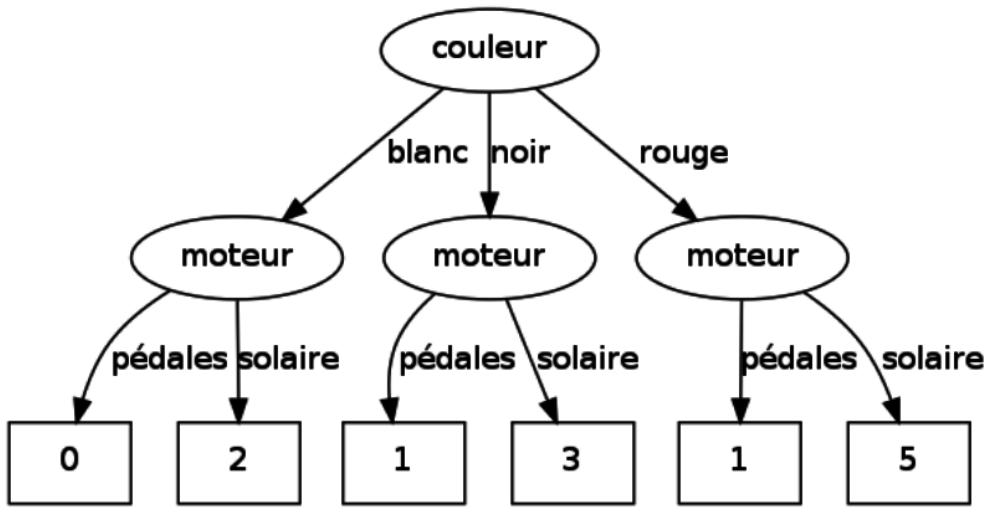
- ADD satisfait toutes les requêtes
- SLDD_+, $\text{SLDD}_\times$ et AADD ont le même profil sur les requêtes
- Les requêtes sur les coupes optimales sont faciles
- Compter est difficile sur les γ-coupes
- Toutes les requêtes sur les γ-coupes exactes sont difficiles

Transformations de coupe

- Plus direct : on veut obtenir une représentation d'une coupe

$\rightarrow$ fonction booléenne dont les modèles sont les éléments de la coupe

- Transformations $\mathbf{CUT}_{\min}$, $\mathbf{CUT}_{\preceq\gamma}$, $\mathbf{CUT}_{\sim\gamma}$
- Sur `ADD`, cela revient à construire un OBDD :

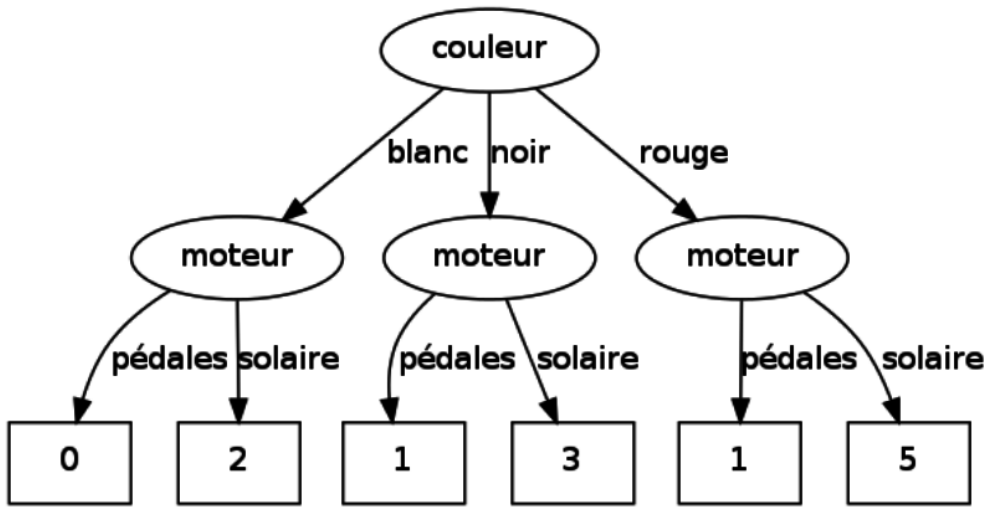


Transformations de coupe

- Plus direct : on veut obtenir une représentation d'une coupe

$\rightarrow$ fonction booléenne dont les modèles sont les éléments de la coupe

- Transformations $\mathbf{CUT}_{\min}$, $\mathbf{CUT}_{\preceq\gamma}$, $\mathbf{CUT}_{\sim\gamma}$
- Sur ADD, cela revient à construire un OBDD :



$\rightarrow$ polynomial : cela explique pourquoi ADD satisfait toutes les requêtes

Carte des transformations de coupe

Transformation	ADD	$\texttt{SLDD}_+$	$\texttt{SLDD}_\times$	AADD
$\mathbf{CUT}_{\text{min}}$	$\surd$	$\surd$	$\surd$	$\surd$
$\mathbf{CUT}_{\sim\gamma}$	$\surd$	$\bullet$	$\bullet$	$\bullet$
$\mathbf{CUT}_{\preceq\gamma}$	$\surd$	$\bullet$	$\bullet$	$\bullet$

- Couper à l'optimum est facile même sur SLDD et AADD : après normalisation, les chemins minimaux passent par les arcs de coefficient 0
- Couper suivant un seuil n'est pas polynomial (cela peut obliger à tout déplier)

Transformations

- Un langage $\mathtt{L}$ satisfait une transformation s'il existe un algorithme polynomial qui l'effectue en restant dans $\mathtt{L}$
- Conditionnement **CD** défini comme en booléen
- Les autres transformations sont paramétrées par un opérateur binaire $\odot$ associatif et commutatif sur $\mathcal{V}$
- $\odot$**C** : combiner n formules par $\odot$ (i.e., construire une formule de $\mathtt{L}$ représentant la fonction $\bigodot_{i=1}^{n} f_{\alpha_i}^{\mathtt{L}}$)
- $\odot$**BC** : combiner un nombre borné de formules

Transformations : élimination de variables

- $\odot$**Elim**, élimination de variables Y suivant $\odot$: construire une formule de $\mathtt{L}$ représentant $\bigodot_{\vec{y}} f_\alpha^{\mathtt{L}}|_{\vec{y}}$
  $\rightarrow$ exemple : *forgetting* $=$ max-élimination
- $\odot$**Marg**, marginalisation sur une variable suivant $\odot$: éliminer toutes les variables sauf une
  $\rightarrow$ $+$-marginalisation sur une variable dans les réseaux bayésiens
- **S**$\odot$**Elim** : éliminer une seule variable
- **SB**$\odot$**Elim** : éliminer une seule variable à domaine borné

Carte des transformations

Transformation	ADD	SLDD$_+$	SLDD$_\times$	AADD
CD	$\surd$	$\surd$	$\surd$	$\surd$
min**BC**	$\surd$	$\bullet$	$\bullet$	$\bullet$
$+$**BC**	$\surd$	$\surd$	$\bullet$	$\bullet$
$\times$**BC**	$\surd$	$\bullet$	$\surd$	$\bullet$
min**C** / $+$**C** / $\times$**C**	$\bullet$	$\bullet$	$\bullet$	$\bullet$
min**Elim**/ $+$**Elim** / $\times$**Elim**	$\bullet$	$\bullet$	$\bullet$	$\bullet$
S**min**Elim / S$+$Elim / S$\times$Elim	$\bullet$	$\bullet$	$\bullet$	$\bullet$
SBmax**Elim** / SBmin**Elim**	$\surd$	$\bullet$	$\bullet$	$\bullet$
SB$+$**Elim**	$\surd$	$\surd$	$\bullet$	$\bullet$
SB$\times$**Elim**	$\surd$	$\bullet$	$\surd$	$\bullet$
min**Marg**	$\surd$	$\surd$	$\surd$	$\surd$
$+$**Marg**	$\surd$	$\surd$	$\surd$	$\surd$
$\times$**Marg**	$\surd$	?	$\surd$	?

Carte des transformations : combinaisons bornées

Transformation	ADD	$\texttt{SLDD}_+$	$\texttt{SLDD}_\times$	AADD
min**BC**	$\surd$	•	•	•
$+$**BC**	$\surd$	$\surd$	•	•
$\times$**BC**	$\surd$	•	$\surd$	•

- ADD satisfait toutes les combinaisons bornées : algorithme *apply* comme pour les OBDDs
- $\texttt{SLDD}_+$ satisfait la combinaison par $+$ mais pas par $\times$, et inversement pour $\texttt{SLDD}_\times$
- AADD ne satisfait aucune combinaison bornée, même le min !

Carte des transformations : combinaisons non bornées

Transformation	ADD	SLDD_+	$\text{SLDD}_\times$	AADD
min**C** / $+$**C** / $\times$**C**	•	•	•	•
min**Elim**/ $+$**Elim** / $\times$**Elim**	•	•	•	•
Smin**Elim** / **S**$+$**Elim** / **S**$\times$**Elim**	•	•	•	•
SBmax**Elim** / **SB**min**Elim**	$\surd$	•	•	•
SB$+$**Elim**	$\surd$	$\surd$	•	•
SB$\times$**Elim**	$\surd$	•	$\surd$	•

Aucun langage ne satisfait les combinaisons non bornées

$\rightarrow$ aucun ne satisfait l'élimination même d'une seule variable, si son domaine n'est pas borné

Carte des transformations : marginalisation

Transformation	ADD	$\texttt{SLDD}_+$	$\texttt{SLDD}_\times$	AADD
min**Marg**	$\surd$	$\surd$	$\surd$	$\surd$
$+$**Marg**	$\surd$	$\surd$	$\surd$	$\surd$
$\times$**Marg**	$\surd$	?	$\surd$	?

La marginalisation est facile (mais il reste des questions ouvertes)

Conclusion et perspectives

- Identification de requêtes et transformations pertinentes pour la manipulation de fonctions à valeurs non booléennes
- Analyse de la complexité des langages de la famille des VDDs à valeurs dans $\mathbb{R}^+$ pour ces opérations

$\rightarrow$ Carte de compilation de ces langages presque complète

- $\mathbb{R}^+$ est un cas particulier (totalement ordonné) : on ne tire pas parti de la généralité du cadre

$\rightarrow$ Que donnerait `SLDD` (ou d'autres langages) sur un $\mathcal{V}$ moins linéaire ?

- Optimisation multicritère : besoin d'autres requêtes et transformations ?


Bahar, R. I., Frohm, E. A., Gaona, C. M., Hachtel, G. D., Macii, E., Pardo, A., and Somenzi, F. (1993).
Algebraic decision diagrams and their applications.
In *Proceedings of ICCAD'93*, pages 188–191.


Darwiche, A. and Marquis, P. (2002).
A knowledge compilation map.
Journal of Artificial Intelligence Research (JAIR), 17 :229–264.


Fargier, H., Marquis, P., and Schmidt, N. (2013).
Semiring labelled decision diagrams, revisited : Canonicity and spatial efficiency issues.
In *Proceedings of IJCAI'13*, pages 884–890.


Wilson, N. (2005).
Decision diagrams for the computation of semiring valuations.
In *Proceedings of IJCAI'05*, pages 331–336.