2 Contraintes et hiérarchies de contraintes

Dans ce chapitre, nous présentons un état de l'art sous la forme de définitions formelles des contraintes, des hiérarchies de contraintes, et des solutions à une hiérarchie de contraintes [BFW94,Wil92, HKS+94]. Nous incluons également des exemples d'utilisation des hiérarchies de contraintes tirés de [BFW92, WB89, FWB92] et des propriétés sur différents comparateurs qui servent à ordonner les solutions d'une hiérarchie de contraintes [BFW92, BFW94].

2.1 Définitions

Une contrainte est une relation entre variables sur un domaine D (c.a.d. entiers, réels, booléens, domaine fini). Une contrainte étiquettée, notée ec, est une contrainte c avec une étiquette e qui exprime l'importance de la contrainte. Nous associons des noms symboliques aux différentes étiquettes des contraintes. Dans la suite de ce chapitre, nous associons à chacun de ces noms symboliques un entier compris entre 0 et n, où n est le nombre de niveaux de la hiérarchie. Le premier niveau 0 est associé au nom symbolique *requise*. Ce niveau de la hiérarchie est toujours réservé aux contraintes dures (c.a.d. absolues, requises).

Une hiérarchie de contraintes est un ensemble de contraintes étiquetées. Etant donnée une hiérarchie de contraintes H, H_0 dénote les contraintes requises dans H. Dans le même esprit, les ensembles H_1, ... H_n sont associés respectivement aux niveaux 1, ...n sont définis.

Une solution à la hiérarchie de contraintes H est une valuation pour les variables dans H (c.a.d. une fonction qui associe à chaque variable dans H un élément de D). On désire définir l'ensemble S qui contient toutes les solutions de H. Plus clairement, chaque valuation dans S doit satisfaire toutes les contraintes dures. De plus, elle doit satisfaire aussi bien que possible l'ensemble des contraintes de préférences en respectant leurs étiquettes relatives. Pour formaliser ce souhait, on définit en premier l'ensemble de valuations noté S_0 tel que chaque valuation dans S_0 satisfasse l'ensemble des contraintes dans H_0. Après et en utilisant S_0, on définit l'ensemble S en éliminant les valuations qui sont moins "bonnes" que d'autres dans S_0. Cette élimination est établie par l'utilisation d'un prédicat "*meilleur*" qui est un comparateur .

$S_0 = \{ \theta: \forall c \in H_0, c\,\theta = vrai \}$ et $S = \{ \theta: \theta \in S_0 \wedge \forall \eta \in S_0 \neg meilleur(\eta, \theta, H) \}$

où (c θ) dénote le résultat booléen de l'application de la contrainte c sur la valuation θ

Le prédicat *meilleur* peut être défini de plusieurs façons différentes. Chaque définition associée à ce prédicat doit satisfaire deux propriétés, à savoir l'irréflexivité et la transitivité.

$\forall\theta\ \forall H \neg meilleur(\theta, \theta, H)$.

$\forall\theta, \eta, \omega\ \forall H\ meilleur(\theta, \eta, H) \wedge meilleur(\eta, \omega, H) \rightarrow meilleur(\theta, \omega, H)$.

Le prédicat *meilleur* doit respecte la sémantique de la hiérarchie : s'il existe une valuation θ dans S_0 qui satisfait complètement toutes les contraintes jusqu'au niveau k, alors toute valuation dans S doit satisfaire toutes les contraintes jusqu'au niveau k.

Si $\exists\theta \in S_0 \wedge \exists k>0$ /

$\forall i \in 1...k\ \forall c \in H_i\ c\,\theta = vrai$

alors $\forall\eta \in S\ \forall i \in 1...k\ \forall c \in H_i\ c\,\eta = vrai$.

Généralement, le prédicat *meilleur* ne produit pas un ordre total sur S_0. Il peut exister dans S deux valuations θ et η telles que θ n'est pas préférée à η et η n'est pas préférée à θ.

2.1.1 Les fonctions d'erreurs

Borning et son équipe dans [BFW92, BFW94] définissent plusieurs alternatives pour le comparateur *meilleur*. Dans chacune de ces alternatives, la fonction d'erreur $e(c\ \theta)$ est définie. Cette fonction retourne un nombre réel positif indiquant à quel degré la contrainte c n'est pas satisfaite par la valuation θ. Cette fonction retourne *0* si et seulement si la valuation θ satisfait la contrainte c. Pour n'importe quel domaine, la fonction d'erreur triviale qui retourne *0* si la contrainte est satisfaite et 1 sinon, peut être utilisée. Le comparateur utilisant cette fonction est appelé *comparateur-prédicat*. Pour un domaine qui est un espace métrique, il peut être souhaitable d'utiliser sa métrique pour déterminer l'erreur plutôt que d'utiliser la fonction d'erreur triviale définie précédemment. Par exemple, l'erreur pour la contrainte $x=y$ peut être la distance qui sépare x de y. Le comparateur utilisant ce type d'erreur est appelé *comparateur-métrique*.

La fonction d'erreur par niveau $E_i(H_i\ \theta)$ applique la fonction e sur chacune des contraintes dans $H_i = [c_1, ..., c_k]$. Le vecteur d'erreurs obtenu par cette application est : $E_i(H_i\ \theta) = [e(c_1\ \theta), ..., e(c_k\ \theta)]$. La séquence d'erreurs pour une hiérarchie de contraintes H contenant n niveaux est le vecteur : $[E_1(H_1\ \theta), ..., E_n(H_n\ \theta)]$.

2.1.2 Les Fonctions d'agrégation d'erreurs

Certains comparateurs auxquels nous nous sommes intéressés combinent les erreurs d'un niveau donné de la hiérarchie avant de comparer les valuations. Pour réaliser cela, ces comparateurs utilisent une fonction d'agrégation g. Cette fonction est appliquée au vecteur de valeurs réelles et retourne des valeurs qui sont comparées en utilisant deux relations: $<>_g$ et $<_g$. Quelques instances possibles de cette fonction d'agrégation g sont la somme des valeurs réelles dans le vecteur ou la valeur maximale dans le vecteur.

On exige que la relation $<_g$ soit irréflexive et transitive. On exige également que la relation $<>_g$ soit réflexive et symétrique. (On utilise la notation $<>_g$ au lieu d'utiliser = puisque pour quelques comparateurs la relation n'est pas transitive). La relation $<>_g$ indique deux valuations ne pouvant pas être ordonnées en utilisant la relation $<_g$ puisque, pour certains comparateurs, elles sont égales et que pour d'autres comparateurs elles sont incomparables.

La fonction d'agrégation G est une généralisation (entres niveaux) de la fonction g. Cette fonction est appliquée à une séquence d'erreurs et retourne une séquence de valeurs qui peuvent être comparées en utilisant les deux relations $<>_g$ et $<_g$.

Soit $R = [E(H_1\ \theta), \ldots, E(H_n\ \theta)]$ alors $G(R) = [g(E(H_1\ \theta)), \ldots, g(E(H_n\ \theta))]$.

L'ordre lexicographique $<_{lex}$ peut être défini d'une façon standard sur les séquences d'erreurs combinées $u_1,\ldots, u_n$ et $w_1,\ldots,w_n$.

$u_1,\ldots, u_n <_{lex} w_1,\ldots,w_n$ si $\exists k \in 1\ldots n$ tel que : $\forall i \in 1\ldots k\text{-}1 \quad u_i <>_g v_i \quad \wedge \quad u_k <_g v_k$

2.1.3 Types de comparateurs

Les séquences d'erreurs des contraintes dans les niveaux $H_1,\ldots, H_n$ sont comparées on utilisant l'ordre lexicographique. Si une valuation θ est considérée meilleure qu'une autre valuation η alors il existe un niveau k de la hiérarchie tel que pour $1 \leq i < k$, $g(E(H_i\ \theta)) <>_g g(E(H_i\ \eta))$, et au niveau k, $g(E(H_k\ \theta) <_g g(E(H_k\ \eta))$.

Borning dans [BFW92, BFW94] classifie deux types de comparateurs : les comparateurs locaux et globaux. Wilson dans [Wil92] distingue un troixième type de comparateurs appelé régionaux. Pour un comparateur local, chaque contrainte de la hiérarchie est considérée individuellement. Une valuation sera meilleure qu'une autre s'il existe un niveau dans lequel elle satisfait un sur-ensemble des contraintes de la seconde, et si pour tous les niveaux plus importants elles satisfont toutes les deux les mêmes contraintes. Pour un comparateur global, les erreurs de toutes les contraintes d'un niveau donné sont agrégées en utilisant la fonction g. Finalement, pour un comparateur régional, chaque contrainte d'un niveau donné est considérée individuellement (comme dans un comparateur local). Cependant, contrairement à un comparateur local, deux valuations qui sont incomparables aux niveaux importants de la hiérarchie, peuvent être comparées à un niveau moins important et par conséquent on obtient l'élimination d'une de ces deux valuations. En général, un comparateur global discrimine plus qu'un comparateur régional et ce dernier discrimine plus qu'un comparateur local. La suite de ce paragraphe présente les définitions formelles de ces comparateurs ainsi que quelques instances de ces définitions utilisées pour la résolution des hiérarchies de contraintes.

Comparateur local

On dira qu'une valuation θ est *Localement-Meilleur* qu'une autre valuation η, si pour chacune des contraintes jusqu'au niveau $k\text{-}1$, l'erreur obtenue sur θ est égale à celle obtenue sur η, et au niveau k, elle est strictement inférieure pour au moins une contrainte et inférieure ou égale pour tout le reste.

Formellement, cette définition est :

Localement-Meilleur$(\theta, \eta, H) \equiv \exists k \in 1\ldots n$ /

$\forall i \in 1\ldots k\text{-}1\ g(E(H_i\ \theta)) <>_g g(E(H_i\ \eta)) \wedge g(E(H_k\ \theta) <_g g(E(H_k\ \eta))$

avec $g(V) = V$ et $<>_g$ et $<_g$ sont définis par :

$$V <_g U \equiv \forall i\ v_i \leq u_i \wedge \exists j\ v_j < u_j .$$
$$V <>_g U \equiv \forall i\ v_i = u_i .$$

Si au niveau *k+1* de la hiérarchie, l'erreur obtenue sur θ est strictement supérieure à celle obtenue sur η pour une ou plusieurs contraintes, alors d'aprés la définition de ce comparateur, cela ne modifie en rien la véracité du comparateur *Localement-Meilleur.* La sémantique de la hiérarchie est toujours respectée.

Comparateur régional

On dira qu'une valuation θ est *Régionalement-Meilleur* qu'une autre valuation η, si pour tout niveau de la hiérarchie jusqu'au niveau *k-1* les deux valuations sont incomparables, et au niveau *k* (*k>1*), l'erreur obtenue après application de θ est strictement inférieure à celle obtenue après application de η pour au moins une contrainte, et inférieure ou égale pour tout le reste.

Formellement cette définition est :

Régionalement-Meilleur(θ, η, H) ≡ $\exists k \in 2...n$ /

$\forall i \in 1...k\text{-}1\ g(E(H_i\ \theta)) <>_g g(E(H_i\ \eta)) \wedge g(E(H_k\ \theta) <_g g(E(H_k\ \eta))$

avec $g(V) = V$ et $<>_g$ et $<_g$ sont définis comme suit :

$V <_g U \equiv \forall i\ v_i \leq u_i \wedge \exists j\ v_j < u_j .$

$V <>_g U \equiv \neg\ ((V <_g U) \vee (U <_g V))$.

Les définitions des comparateurs globaux que nous allons présenter considèrent que les contraintes de chaque niveau d'une hiérarchie peuvent être pondérées avec des poids réels. Pour les comparateurs locaux ou régionaux, cette pondération n'est pas utile puisque le résultat est équivalent avec ou sans pondération.

Comparateur Global

On dira qu'une valuation θ est *Globalement-Meilleur* qu'une autre valuation η, si pour chaque niveau de la hiérarchie jusqu'au niveau *k-1*, la valeur obtenue par agrégation des erreurs sur θ est égale à celle obtenue sur η, et au niveau *k*, elle est strictement inférieure.

Formellement, cette définition est :

Globalement-Meilleur(θ, η, *H*, *g*) ≡ $\exists k \in 1...n$ /

$\forall i \in 1...k\text{-}1\ g(E(H_i\ \theta)) <>_g g(E(H_i\ \eta)) \wedge g(E(H_k\ \theta) <_g g(E(H_k\ \eta))$.

Les relations $<>_g$ et $<_g$ sont équivalentes respectivement à = et < pour les réels, et *g(V)* est la fonction qui agrège la séquence d'erreurs dans le vecteur *V.*

Parmi les comparateurs globaux utilisant une fonction d'agrégation on peut citer par exemple : *Somme-Pondérée*, *Pire-Cas* ou encore *moindre-carrés*. Ces différents comparateurs prennent en compte des hiérarchies où les contraintes dans chaque niveau sont pondérées avec des poids réels. Les définitions de ces comparateurs sont:

Somme-Pondérée(θ, η, *H*) ≡ *Globalement-Meilleur*(θ, η, *H*, *g*)

avec $g(V) = \left(\sum_{i=1}^{|V|} w_i v_i \right)$.

Pire-Cas(θ, η, *H*) ≡ *Globalement-Meilleur*(θ, η, *H*, *g*)

avec $g(V) = (Max_{i:1..|V|} \{w_i v_i\})$.

Moindre-Carrés(θ, η, *H*) ≡ *Globalement-Meilleur*(θ, η, *H*, *g*)

avec $g(V) = \left(\sum_{i=1}^{|V|} w_i v_i^2 \right)$.

Indépendamment du choix de *Localement-Meilleur*, *Régionalement-Meilleur* ou d'une instance de *Globalement-Meilleur*, on peut choisir une fonction d'erreur appropriée pour les contraintes. Par exemple, le comparateur *Localement-Prédicat-Meilleur* est un dérivé de *Localement-Meilleur* (utilisant la fonction triviale qui retourne 0 si la contrainte est satisfaite et 1 sinon).

Un autre exemple est celui de *Localement-Métrique-Meilleur*, ce comparateur est un dérivé de *Localement-Meilleur* utilisant le domaine métrique pour compter l'erreur d'une contrainte. Nous avons aussi *Somme-Pondérée-Prédicat* et *Somme-Pondérée-Métrique* qui sont définis respectivement en utilisant l'erreur prédicat et l'erreur métrique.

Le comparateur *Nombre-Contraintes-Non-Satisfaites* est un cas particulier du comparateur *Somme-Pondérée-Prédicat* utilisant le poids 1 pour chacune des contraintes de chaque niveau de la hiérarchie. Ce comparateur compte le nombre de contraintes non satisfaites par niveau pour deux valuations quelconques, et établit leur comparaison.

On peut également remarquer l'existence du comparateur *Moindres-Carrés-Prédicat*, mais ce comparateur n'est pas particulièrement utile puisqu'il est équivalent au comparateur *Somme-Pondérée-Predicat* (puisque $1^2=1$). Le comparateur *Moindres-Carrés* est plutôt utilisé en se référant à l'erreur métrique, ce qui implique l'appellation suivante : *Moindres-Carrés-Métrique.*

2.1.4 Exemples illustratifs

Dans ce paragraphe, on présente deux exemples empruntés à [BFW94]. Le premier illustre la domination des contraintes situées au niveau fort d'une hiérarchie par rapport aux contraintes situées à un niveau plus faible. Le deuxième illustre les variétés des solutions obtenues par les différents comparateurs définis précédemment.

FIGURE 1 : Hiérarchie a trois niveaux de contraintes

niveau		*contraintes*
H_0	*requise*	*Celsius* × *1.8* = *Fahrenheit* − *32*
H_1	*forte*	*Fahrenheit* = 212
H_2	*faible*	*Celsius* = *0*

Le premier exemple considère la hiérarchie décrite dans la figure 1. Cette hiérarchie inclut la contrainte canonique Celsius-Fahrenheit. L'ensemble S_0 contient toutes les valuations qui satisfont la contrainte requise dans H_0. Pour cette hiérarchie, l'ensemble S_0 est infini, et contient tous les couples *(c,f)* de températures valide. S_0 = *{....,(-60, -76), (-40, -40), (0, 32), (10, 50), (100, 212), ...}* et S contient la seule valuation θ = *(100, 212)*. S ne contient aucune des autres valuations car celles-ci sont jugées noins bonnes que θ. Prenons par exemple la valuation η = *(10, 50)*, les valuations θ et η satisfont toutes les deux la contrainte requise dans H_0 et donc les erreurs de ces deux valuations sur ce niveau de la hiérarchie sont les mêmes. La valuation θ satisfait la contrainte du niveau H_1 de la hiérarchie tandis que la valuation η ne la satisfait pas, et donc il existe un niveau $k > 0$ ($k=1$) pour lequel l'erreur est strictement inférieure pour θ que celle obtenue pour η. Par conséquent, on a *Localement-Meilleur*(θ, η, H).

Considérons maintenant la valuation ω = *(0, 32)*. ω satisfait la contrainte dans H_1 et non celle dans H_2. Intuitivement, du fait que θ satisfait mieux les contraintes des niveaux supérieurs de la hiérarchie que ω, on a bien *Localement-Meilleur*(θ, ω, H). Cet exemple produit exactement le même ensemble S pour les comparateurs *Localement-Meilleur-Prédicat*, *Localement-Meilleur-Métrique* ou les instances du comparateur *Globalement-Meilleur*. Bien entendu, ceci ne peut pas être le cas général.

Le deuxième exemple illustre les variétés des solutions obtenues par les différents comparateurs présentés auparavant. Considérons un simple calculateur graphique comme il est décrit dans [BFW92]. Supposons que qu'on a la relation $A+B=C$. Initialement les valeurs des variables de cette relation sont respectivement : 2, 3 et 5. L'utilisateur désire modifier la valeur de la variable C et la rendre égale à 7. La hiérarchie dans la figure 2 exprime la sémantique de cette action. La contrainte $C = A+B$ est la contrainte requise et qui représente la relation de la somme entre les variables. La contrainte $C = 7$ est considérée au niveau le plus fort[1] de la hiérarchie (après le niveau des contraintes requises) puisqu'elle représente le désir de l'utilisateur. Les deux contraintes $A = 2$ et $B = 3$ sont considérées au dernier niveau de la hiérarchie et expriment le désir que le reste du système soit changé le moins possible en prenant en compte la nouvelle valeur de la variable C. Sans ces deux dernières, la valuation θ = *{ A = 1000000, B = –999993, C = 7}* peut parfaitement être une solution valide. On illustre maintenant les solutions des différents comparateurs.

FIGURE 2 : Hiérarchie d'un simple calculateur graphique.

niveau		*contraintes*
H_0	*requise*	$C = A + B$
H_1	*forte*	$C = 7$
H_2	*faible*	$A = 2$
	faible	$B = 3$

Le comparateur *Localement-Prédicat-Meilleur* produit deux valuations dans l'ensemble S :

1 Le fait de mettre cette contrainte au niveau des contraintes de plus grande préférence et non au niveau des contraintes requises, permet au système de ne pas modifier la variable C si cette dernière est déterminée par une contrainte requise.

$\theta = \{ A = 2, B = 5, C = 7 \}$ et $\eta = \{ A = 4, B = 3, C = 7 \}$.

Les séquences d'erreurs obtenues par θ et η sont respectivement :

$g(E(H_0\ \theta)) = g([0]) = [0]$, $g(E(H_1\ \theta)) = g([0]) = [0]$, $g(E(H_2\ \theta)) = g([0, 1]) = [0, 1]$ et $g(E(H_0\ \eta)) = g([0]) = [0]$, $g(E(H_1\ \eta)) = g([0]) = [0]$, $g(E(H_2\ \eta)) = g([1, 0]) = [1, 0]$.

Les deux valuations possèdent les mêmes séquences d'erreurs pour les deux niveaux 0 et 1 de la hiérarchie. Au dernier niveau de cette hiérarchie, la première valuation satisfait la contrainte $A = 2$ mais pas la contrainte $B = 3$ et la deuxième valuation satisfait la contrainte $B = 3$ mais pas la contrainte $A = 2$, ce qui fait que ces deux valuations ne satisfont pas les mêmes contraintes, et par conséquent elles sont mises toutes les deux dans l'ensemble S puisqu'elles ne sont pas comparables.

Le comparateur *Localement-Métrique-Meilleur* produit une infinité de solutions dans l'ensemble S :

$\{ A = x, B = 7-x, C = 7$ *pour* $x \in [2 \ldots 4] \}$. Aucune des valuations dans S n'est meilleure qu'une autre dans S. Par exemple, considérant les deux valuations suivantes dans S : la valuation $\theta = \{ A = 2.9, B = 4.1, C = 7 \}$ et $\eta = \{ A = 2, B = 5, C = 7 \}$. Les séquences d'erreurs obtenues par θ et η sont respectivement :

$g(E(H_0\ \theta)) = g([0.0]) = [0.0]$, $g(E(H_1\ \theta)) = g([0.0]) = [0.0]$, $g(E(H_2\ \theta)) = g([0.9, 1.1]) = [0.9, 1.1]$ et

$g(E(H_0\ \eta)) = g([0.0]) = [0.0]$, $g(E(H_1\ \eta)) = g([0.0]) = [0.0]$, $g(E(H_2\ \eta)) = g([0.0, 2.0]) = [0.0, 2.0]$.

Les deux valuations possèdent les mêmes séquences d'erreurs pour les deux niveaux 0 et 1 de la hiérarchie. Au dernier niveau de cette hiérarchie, la première valuation satisfait la contrainte $A = 2$ moins bien que la deuxième valuation (puisque $0.0 < 0.9$) et la deuxième valuation satisfait la contrainte $B = 3$ moins bien que la première valuation (puisque $1.1 < 2.0$), et par conséquent les deux valuations sont dans S. Cependant, les valuations du type $\{ A = 1000000, B = -999993, C = 7 \}$ sont considérées moins bonnes et ne sont pas dans S.

Le comparateur *Somme-Pondérée-Prédicat* produit le même ensemble de valuations que celui obtenu par le comparateur *Localement-Prédicat-Meilleur* si l'on considère que les deux contraintes du dernier niveau de la hiérarchie sont pondérées avec des poids équivalents. Dans le cas contraire, selon l'importance du poids, une des deux valuations sera choisie et mise dans S.

Le comparateur *Somme-Pondérée-Métrique* produit le même ensemble infini de valuations que celui obrenu par le comparateur *Localement-Métrique-Meilleur* si l'on considère que les deux contraintes du dernier niveau de la hiérarchie sont pondérées avec un poids équivalent. Dans le cas contraire, selon l'importance du poids, une des deux valuations suivantes : $\{A = 2, B = 5, C = 7\}$ $\{A = 4, B = 3, C = 7\}$, sera choisie et mise dans S.

Le comparateur *Moindres-Carrés-Métrique* produit la seule valuation : $\{A = 3, B = 4, C = 7\}$ lorsque le poids pondéré sur chacune des contraintes du dernier niveau de la hiérarchie est le même. Le comparateur *Pire-Cas-Métrique* produit la même valuation, en considérant la condition précédente. Les comparateurs régionaux produisent les mêmes solutions que les comparateurs locaux pour cette hiérarchie.

2.2 Comportement des comparateurs.

Une objection à ce qu'on a présenté serait : le fait d'avoir une hiérarchie contenant plus de deux niveaux (le premier pour les contraintes requises et le second pour les contraintes de préférences) n'est pas nécessaire puisque les contraintes de préférences peuvent être pondérées avec des poids pour déterminer l'ensemble *S*. Il y a deux raisons pour rejeter cette objection. La première est raison est conceptuelle et la deuxième est pragmatique. La raison conceptuelle constitut la raison fondamentale de l'utilisation des hiérarchies de contraintes. Ces raisons sont :

- ne pas laisser les contraintes contraintes faibles influencer le résultat. Pour illustrer cette raison conceptuelle, considérons une application graphique interactive où nous avons à déplacer une ligne par la souris. La ligne est contrainte par trois contraintes : deux contraintes sont considérées au niveau fort de la hiérarchie et la troisième contrainte est considérée à un niveau en dessous. Les deux premières contraintes sont : " *la ligne doit rester horizontale*", " *un point extrémité de la ligne doit suivre le déplacement de la souris*". La troisième contrainte est : "*la ligne est attachée à des points fixes dans le diagramme*". Le désir de l'utilisateur dans ce cas est que la ligne demeure exactement horizontale tout en suivant précisément la souris (laissant la contrainte faible non satisfaite), au lieu de laisser la contrainte faible influencer le résultat (ce qui résulterait en ce que la ligne serait proche de l'horizontale ou en ce qu'elle serait proche de la position de la souris).
- les solutions d'une hiérarchie contenant plusieurs niveaux de préférence (où un niveau supérieur domine complètement un niveau inférieur) peuvent être obtenues beaucoup plus facilement que si la hiérarchie possède un seul niveau de préférence contenant les contraintes pondérées (bien sûr ceci dépend des algorithmes utilisés).

La plupart des concepts des hiérarchies de contraintes sont dérivés des concepts de certaines branches de la recherche opérationnelle comme la programmation linéaire [Mur83], la programmation par buts [Ign83] ou encore la programmation par buts généralisée [Ign85]. Le domaine des contraintes dans la recherche opérationnelle est souvent l'ensemble des réels *R*, et parfois le domaine est celui des entiers naturels *N*, selon le type de problème.

Dans [BFW92, Wil92], l'approche de la hiérarchie de contraintes a été précédée par l'approche des problèmes multi-objectifs qui consiste à considérer la fonction objectif en priorité. Le concept des solutions obtenues par le comparateur *Localement-Meilleur* est dérivé de celui du vecteur minimal d'un problème de programmation linéaire multi-objectifs. D'une façon identique, le concept des solutions obtenues par le comparateur *Somme-Pondérée* ou par le comparateur *Pire-Cas* sont tous les deux dérivés de concepts analogues des problèmes de programmation linéaire multi-objectifs et programmation par buts généralisée.

Pour une fonction d'erreur donnée, on peut remarquer l'existence de plusieurs relations entre les comparateurs locaux et les comparateurs globaux :

Proposition 2.1 : $\forall\theta, \eta\ \forall H$ *Localement-Métrique-Meilleur*$(\theta, \eta, H) \rightarrow$ *Somme-Pondérée-Métrique*(θ, η, H).

Preuve : Supposons que *Localement-Métrique-Meilleur*(θ, η, H) est vrai, et donc qu'il existe un niveau $k>0$ dans H tel que l'erreur résultante après l'application à chacune des contraintes sur θ jusqu'au niveau $k-1$ est égale à celle obtenue après application sur η. Ceci implique bien que la somme des erreurs pondérées par les poids après application sur θ de toutes les contraintes jusqu'au niveau $k-1$ est égale à la somme des erreurs pondérées par les poids après application de η. De plus au niveau k, et par utilisation de *Localement-Métrique-Meilleur*(θ, η, H), l'erreur après application

sur θ est strictement inférieure pour au moins une contrainte et inférieure ou égale pour toutes les contraintes restantes de ce niveau après application sur η. Ceci implique bien aussi que la somme des erreurs pondérées par les poids après application sur θ des contraintes du niveau k est strictement inférieure à la somme des erreurs pondérées par les poids après application sur η. Par conséquent, on a bien *Somme-Pondérée-Métrique*(θ, η, *H*).

Corollaire 2.1: Pour une hiérarchie de contraintes donnée, soit S_{LMM} l'ensemble des solutions obtenu par l'utilisation du comparateur *Localement-Métrique-Meilleur*, et S_{SPM} celui obtenu par l'utilisation du comparateur *Somme-Pondérée*, on a : $S_{SPM} \subseteq S_{LMM}$.

Proposition 2.2 : ∀θ, η ∀*H Localement-Métrique-Meilleur*(θ, η, *H*) → *Moindre-Carré*(θ, η, *H*).

Preuve : Identique à celle de la proposition 1.

Corollaire 2.2: Pour une hiérarchie de contraintes donnée, soit S_{LMM} l'ensemble des solutions obtenu par l'utilisation du comparateur *Localement-Métrique-Meilleur*, et S_{MC} celui obtenu par l'utilisation du comparateur *Moindre-Carré*, on a : $S_{MC} \subseteq S_{LMM}$.

Les propositions 2.1 et 2.2 concernent particulièrement deux instances du schéma du comparateur *Globalement-Meilleur*. Pour une fonction d'agrégation g arbitraire, le comparateur *Localement-Meilleur* n'implique pas toutes les instances du comparateur *Globalement-Meilleur*. En particulier, *Localement-Meilleur* n'implique pas *Pire-Cas* ou encore d'autres comparateurs que nous définirons dans les chapitres suivant.

2.3 Erreurs produites par des inégalités

Certains problèmes sont soulevés lorsqu'on a à résoudre des contraintes d'inégalité stricte en utilisant l'erreur métrique. Par exemple: que doit être la fonction d'erreur pour la contrainte $x>y$, lorsque le domaine des variables x et y est l'ensemble des réels ? Si x est supérieur à y alors l'erreur doit être nulle. Si x n'est pas supérieur à y, on aimerait que l'erreur soit égale au plus petit réel proche de y-x. L'évidente fonction d'erreur $e(x>y) = 0$ si $x>y$ sinon y-x. n'est pas correcte puisqu'elle retourne la valeur 0 lorsque la valeur de x est égale à celle de y. Cependant, si l'erreur est égale a un nombre d positif lorsque $x=y$, alors on aura une erreur plus petite lorsque $y=x+d/2$, et donc ceci contredit notre souhait que l'erreur diminue quand la valeur de x s'approche de celle de y. Pour résoudre ce problème, un nombre infiniment petit noté ε est introduit [Rob66] : ε est plus grand que 0 et plus petit que tout nombre positif. En utilisant ε, la fonction erreur e est définie comme suite :

$$e(x>y) = \begin{cases} y - x \text{ si } x<y \\ \varepsilon \text{ si } x=y \\ 0 \text{ si } x>y \end{cases}$$

$$e(x \neq y) = \begin{cases} 0 \text{ si } y \neq x \\ \varepsilon \text{ si } x=y \end{cases}$$

$$e(x<y) = \begin{cases} x - y \text{ si } x>y \\ \varepsilon \text{ si } x=y \\ 0 \text{ si } x<y \end{cases}$$

Il est à noter que ε est ajouté seulement aux valeurs possibles de la fonction erreur et non au domaine D. Si l'on essaye de changer le domaine lui même, on tombera dans le problème cité auparavant[1].

2.4 Existence de solutions

Intuitivement, on peut supposer que l'ensemble S des solutions à une hiérarchie de contraintes est non vide lorsque l'ensemble S_0 de solutions pour les contraintes requises de cette hiérarchie est non vide. Cependant, ceci n'est pas toujours le cas. Considérons l'erreur métrique et la hiérarchie suivante: *requise* $x>0$ et *fort* $x=0$ avec x ayant pour domaine R. L'ensemble S_0 est celui de R^{+*} mais l'ensemble S est vide puisque pour chaque valuation dans l'ensemble S_0 (par exemple d), il existe une autre valuation (par exemple $d/2$) qui satisfait mieux la contrainte de préférence $x=0$.

Proposition 2.3: Si S_0 est non vide et fini alors S est non vide.

Preuve : Supposons que S soit vide, et prenons une valuation θ_1 de S_0. Puisque $\theta_1 \notin S$, il doit exister une autre valuation $\theta_2 \in S_0$ telle qu'on ait *meilleur*(θ_2, θ_1, H). D'une façon similaire, puisque $\theta_2 \notin S$, il doit exister une autre valuation $\theta_3 \in S_0$ telle qu'on ait *meilleur*(θ_3, θ_2, H), et ainsi de suite pour la chaîne infinie θ_4, θ_5... θ_n. Puisque *meilleur* est transitive on a par induction que $\forall i,j > 0$ [$i > j \rightarrow$ *meilleur*(θ_i,θ_j, H)]. La propriété d'irréflexivité du comparateur *meilleur* impose que $\forall i > 0$ on a $\neg$ *meilleur*(θ_i,θ_i, H) et donc toutes les θ_i sont distinctes. On aboutit donc à leur infinité, ce qui est en contradiction avec l'hypothèse.

La plupart des applications pratiques utilisent des hiérarchies de contraintes finies. Par exemple pour les programmes en programmation impérative par contraintes ou encore en programmation logique avec hiérarchie de contraintes, si le programme se termine, l'ensemble des contraintes résultant doit être fini. La proposition suivante nous montre que dans plusieurs cas d'importance pratique, si les contraintes requises peuvent être satisfaites, alors une solution à la hiérarchie existe.

Proposition 2.4: Si S_0 est non vide, et H est fini et le comparateur utilisé utilise l'erreur prédicat, alors S est non vide.

Preuve : Supposons que S soit vide, et utilisons les mêmes arguments décrits dans la preuve précédente, on présume donc qu'il doit exister un nombre infini de valuations distinctes $\theta_i \in S_0$. Cependant, si le comparateur est un prédicat, une valuation ne peut être meilleure qu'une autre si toutes les deux satisfont exactement le même sous ensemble de contraintes dans H. Par conséquent, chacune des valuations $\theta_i \in S_0$ doit satisfaire un sous ensemble de contraintes différent dans H, et donc ceci suppose que l'on doit avoir H infini ce qui est en contradiction avec l'hypothèse.

2.5 Les aspects non-monotones des comparateurs

Il est temps de considérer les effets de l'ajout d'une contrainte à une hiérarchie existante pour raffiner l'ensemble de valuations solutions de la hiérarchie [Wil92, Bor81, JM87]. Ceci peut avoir lieu, par exemple dans les environnements graphiques interactifs où l'on possède une série de buts successifs relatifs à la position d'un objet sur l'écran. Chaque but représente une commande de l'uti-

1. Que doit être l'erreur pour la contrainte 0>ε/2 ?. Selon la définition, l'erreur est de ε/2, mais cette erreur est inférieure à celle de la contrainte 0>0, quoique la contrainte 0>0 soit mieux satisfaite.

lisateur. Si une des commandes n'a pas encore été exécutée, et si l'on veut déterminer la position courante de l'objet pour pouvoir l'afficher, on aura besoin de manipuler les contraintes d'une façon incrémentale.

Malheureusement, les comparateurs qui respectent la hiérarchie ne possèdent pas la propriété d'ordre décrite ci-après. Intuitivement, ceci veut dire que l'ajout d'une nouvelle contrainte à la hiérarchie peut conduire à une solution qui n'est pas une solution construite à partir de la solution précédente, mais plutôt une solution différente.

Définition 2.2

Soient H et J deux hiérarchies de contraintes. Soit C un comparateur. C est dit ordonné si et seulement si : $S_{(H\cup J)}(C) \subseteq S_{(H)}(C)$. Un comparateur qui n'est pas ordonné est appelé désordonné [WB89].

Il est à noter qu'en programmation logique avec contraintes on a l'aspect ordonné, puisque l'ajout d'une contrainte requise à un ensemble initial de contraintes requises ne peut que réduire ou laisser constant l'ensemble de valuations qui satisfont cet ensemble initial de contraintes. Cette propriété est similaire à la propriété de "*la stabilité de rejet*" décrite dans [Wyk80].

Proposition 2.5 : Soit D un domaine contenant plus qu'un élément. Tout comparateur C qui respecte la hiérarchie est désordonné.

Preuve : Soit H=*{faible x=a}* et soit J=*{forte x=b}*, avec a et b deux éléments distincts du domaine. Soit C un comparateur qui respecte la hiérarchie. $S_{(H)}(C)$ contient la valuation qui associe x à la valeur a, et $S_{(H\cup J)}(C)$ contient uniquement la valuation qui associe x à la valeur b. On n'a pas la relation $S_{(H\cup J)}(C) \subseteq S_{(H)}(C)$ puisque a et b sont distincts. Ceci implique bien que C est non ordonné.

Donc, si l'on possède un résolveur incrémental, l'ajout d'une nouvelle contrainte doit en général nous obliger à rétracter la solution précédente. L'aspect désordonné des comparateurs montre que l'implémentation d'un résolveur incrémental général présente quelques difficultés. Cependant, un certain nombre de solutions sont disponibles pour des cas pratiques. Considérons par exemple une application graphique interactive. Comme il est mentionné plus haut, une implémentation raisonnable en programmation logique peut être de créer une série de buts en relation avec les différents états successifs d'un objet en mouvement à l'écran. En principe, on peut déterminer une solution d'une façon incrémentale pour un de ces états, mais après l'ajout d'une nouvelle contrainte sur cet état, la précédente solution peut devenir invalide.

En pratique et pour les interfaces graphiques, la programmation logique avec hiérarchie de contraintes est utilisée dans le sens où toutes les contraintes sur un état donné doivent être connues à l'avance [WB89]. De plus, les contraintes doivent porter seulement sur les états futurs et non sur les états courants. Une solution à ce problème serait d'inclure une variante d'opérateurs dans le langage. Un opérateur doit agir comme une coupure et doit aussi résoudre la hiérarchie courante et enfin imposer que la solution trouvée soit une contrainte requise sur les variables de la hiérarchie. Après ce processus, l'ajout d'une nouvelle contrainte au système ne peut que rendre la solution plus fine et non l'invalider.

Synthèse du chapitre

Nous avons présenté dans ce chapitre l'aspect formel d'une hiérarchie de contraintes. La résolution d'une hiérarchie passe d'abord par la résolution des contraintes dures dans cette hiérarchie. L'ensemble des valuations obtenu à l'issue de cette résolution est ensuite utilisé pour bâtir l'ensemble des valuations qui sont solutions de la hiérarchie. La construction de ce dernier ensemble nécessite l'utilisation d'un critère de comparaison. Ce critère de comparaison peut être local, régional ou global.

Nous avons présenté des instances de ces trois types de comparateurs et montré leurs effets sur des exemples de hiérarchies de contraintes. Nous avons présenté également le comportement de ces types de comparateurs, certaines de leurs propriétés et les relations existantes entres leurs différentes instances. Enfin, nous avons montré le comportement de ces types de comparateurs dans un environnement incrémental où les contraintes sont ajoutées (ou retirées) à la hiérarchie.