3 Extensions de la théorie des hiérarchies de contraintes

Les cinq parties qui composent ce chapitre visent à donner un aperçu sur les extensions de la théorie des hiérarchies de contraintes décrite au chapitre précédent. Dans un premier temps, on verra les annotations que l'on peut porter sur des variables dans une hiérarchie de contraintes. Nous montrons l'intérêt de ces annotations à travers des exemples pratiques et réels. Dans un deuxième temps, nous verrons la notion des hiérarchies partiellement ordonnées et ce qu'il résulte de cet ordre. En troisième partie de ce chapitre, nous présenterons l'intégration des fonctions objectifs aux hiérarchies de contraintes. En quatrième partie, on discutera de la variation du critère selon les niveaux d'une hiérarchie ainsi que de l'intérêt de cette variation. Et en dernière partie, nous parlerons de l'intégration des hiérarchies de contraintes dans la programmation logique par contraintes.

3.1 Les annotations *lecture-seulement* et *écriture-seulement*

3.1.1 L'annotation *lecture-seulement*

Les systèmes de contraintes basés sur le modèle de perturbation utilisent souvent le principe de l'annotation *lecture-seulement*. Ceci a pour but d'aider à limiter le choix des variables à recalculer pour resatisfaire les contraintes après une perturbation introduite dans le système. Les hiérarchies de contraintes peuvent être vues comme une solution alternative pour spécifier ce choix sans avoir à soulever la généralité des contraintes multi-directionnelles. Cependant, la notion de *lecture-seulement* est utile dans les hiérarchies de contraintes multi-directionnelles. Cette notion est utile à la résolution de contraintes qui font référence à un support externe pour l'acquisition d'une donnée ou qui font référence à une autre source d'information hors site. Un exemple simple est celui de la souris. La contrainte possédant un point qui suit la position de la souris doit être en *lecture-seulement* sur cette position.

Un autre exemple est celui des contraintes qui sont des relations entre des états anciens et des états nouveaux d'un robot. Dans ce cas, on aimerait que les états anciens soient en *lecture-seulement* afin que le futur ne change pas le passé.

Intuitivement, lorsqu'on choisit les meilleures solutions d'une hiérarchie de contraintes, les contraintes ne doivent pas influencer le choix des valeurs de leurs variables marquées par l'annotation *lecture-seulement*. Au contraire, les contraintes peuvent influencer le choix des valeurs de leurs variables qui ne sont pas annotées. Cependant, on veut que les contraintes soient satisfaites si possible tout en respectant leurs niveaux de préférence. En particulier, lorsque les contraintes requises sont soumises à l'annotation *lecture-seulement*, on doit avoir leur satisfaction. Une esquisse informelle de la définition est donnée dans [BFW92, BFW94] et consiste à remplacer les occurrences des variables (marquées par la notation *lecture-seulement*) par des constantes. Ceci a pour but d'empêcher la contrainte d'influencer le choix de ces variables marquées. Ainsi nous commençons la définition de l'ensemble de solutions de la hiérarchie H en formulant un ensemble Q de hiérarchies de contraintes. Chaque élément dans Q est une hiérarchie de contraintes avec un domaine arbitraire d'éléments substitué pour les variables marquées par la notation *lecture-seulement* .

Cette démarche est réalisée en suivant les étapes suivantes: une valeur pour la variable est choisie et une hiérarchie est formulée en considérant cette valeur. Après avoir choisi toutes les valeurs possibles, les solutions résultantes sont triées en enlevant celles qui sont incorrectes (il est à signaler que cette approche a pour objectif de spécifier la sémantique de la notation *lecture-seulement*, et que cela ne construit pas un algorithme raisonnable pour la résolution d'une hiérarchie de contraintes. Les algorithmes de résolution sont discutés dans les chapitres suivants de ce mémoire).

3.1.2 Exemple illustratif

La figure 3 montre une hiérarchie à deux niveaux de préférences (cet exemple est emprunté à [BFW94]). Le symbole *"?"* symbolise la notation *lecture-seulement*. Les éléments de Q sont formulés en remplaçant les occurrences de la variable y par les éléments du domaine D.

FIGURE 3 : Hiérarchie contenant l'annotation lecture-seulement

Hiérarchie initiale	$q \in Q$ *formulé en remplaçant* $y?$ *par un* $d \in D$			
requise $x = y?$	$y? \rightarrow -3.2$ $x = -3.2$	$y? \rightarrow 4$ $x = 4$	$y? \rightarrow 7.8$ $x = 7.8$	....
forte $x = 3$	$x = 3$	$x = 3$	$x = 3$	....
faible $y = 4$	$y = 4$	$y = 4$	$y = 4$	....

Dans la figure 4, on résout les hiérarchies de contraintes dans Q en éliminant les valuations qui associent aux occurrences des variables (non marquées) une valeur différente de celle associée à l'occurrence marquée par la notation *lecture-seulement*. On observe que la valuation $\{y \rightarrow 4,\ x \rightarrow 4\}$ est la seule consistante et constitue une solution à la hiérarchie initiale.

FIGURE 4 : Hiérarchie contenant la notation lecture-seulement

	q ∈ *Q formulé en remplaçant y? par un d* ∈ *D*	
valuation ϕ	*y?* → *−3.2*	*y?* → *4*
hiérarchie q	*requise x = −3.2* *forte x = 3* *faible y = 4*	*requise x = 4* *forte x = 3* *faible y = 4*
valuation θ	*{y* → *4, x* → *−3.2 }*	*{y* → *4, x* → *4}*
consistance	*y?*ϕ ≠ y θ	*y?*ϕ = y θ
jugement	*retirée*	*gardée*

3.1.3 Définitions formelles de la notion *lecture-seulement*

Les définitions formelles illustrant la sémantique de cette notion de *lecture-seulement* sont les suivantes :

Définition 3.1:

Soit θ une valuation, soit l'ensemble des variables $v_1, \ldots, v_n$ les variables de θ.

Soit la valuation θ' telle que θ' = {θ omet $w_1, \ldots, w_n$ }.
Le domaine de la valuation θ' est $\{v_1, \ldots, v_n\} - \{w_1, \ldots, w_n\}$ et
pour tout v appartenant à ce domaine on a : θ'v=θv.
D'une façon similaire, si S est un ensemble de valuations,
S omet $w_1, \ldots, w_n \equiv \{\theta$ omet $w_1, \ldots, w_n / \theta \in S\}$.

Définition 3.2 :

Soit H une hiérarchie de contraintes, et soit D le domaine des contraintes de cette hiérarchie. Soit $v_1, \ldots, v_n$ l'ensemble des variables de H. H a une ou plusieurs occurrences marquées par *lecture-seulement*. Soient $w_1,\ldots,w_m$ des nouvelles variables non référencées dans H et soit J la hiérarchie résultante suite à la substitution de w_i aux occurrences de la variable v_i marquée (les occurrences de la variable v_j non marquées ne sont pas concernées). On définit Q par l'ensemble de toutes les hiérarchies $J\,\rho$ où chaque ρ est formé en substituant à w_i un élément arbitraire du domaine D.

$$Q = \{J\,\rho / d_1 \in D, \ldots, d_1 \in D\} \text{ et } \rho = \{ w_1 \rightarrow d_1, \ldots, w_m \rightarrow d_m\}.$$

Soit *solutions*$(J\,\rho)$ l'ensemble des solutions de $J\,\rho$ (ici, on utilise la définition de l'ensemble de solutions S présenté au chapitre précédent puisque J ne possède pas maintenant de variables marquées par la notation *lecture-seulement*). L'ensemble de solutions consistantes de $J\,\rho$ est défini par :

$$Consistante(J\,\rho) = \{\theta / \theta \in solutions(J\,\rho) \wedge (w_1\,\rho = v_1\,\theta) \wedge \ldots \wedge (w_m\,\rho = v_n\,\theta)\}.$$

Une solution est consistante lorsque ρ associe un élément d_i du domaine à w_i et que θ associe le même élément d_i à v_i. L'ensemble désiré de solutions à la hiérarchie H est l'ensemble des solutions consistantes omettant les valeurs associées aux nouvelles variables w_i introduites.

$$Solutions(H) = (\cup_{(J\,\rho) \in Q} Consistante(J\,\rho)) \text{ omet } w_1,\ldots,w_n .$$

Proposition 3.1 : Pour une hiérarchie H ne contenant que des contraintes requises, soit H' la même hiérarchie que H mais sans la notation *lecture-seulement*, alors $Solutions(H) = Solutions(H')$.

Preuve 3.1 : Dans un premier temps on prouve que : $Solutions(H) \supseteq Solution(H')$. Soit $\{v_1, \ldots, v_n\}$, $\{w_1, \ldots, w_n\}$ et J défini comme auparavant. Soit θ une solution pour H'. On définit $\rho = \{ w_i \rightarrow v_i\,\theta, \ldots, w_m \rightarrow v_m\,\theta \}$ (en d'autres termes: si θ associe d_i à v_i, alors ρ associe d_i à la variable w_i correspondante) et donc il est clair que $\theta \in Solutions(J\ \rho)$ et θ est constante. Par conséquent, $\theta \in Solutions(H)$. Dans un second temps on prouve que : $Solutions(H) \subseteq Solution(H')$. Soit θ une solution pour H. Par définition θ est une solution consistante de $(J\ \rho)$ pour un ρ donné. Puisque H contient seulement des contraintes requises et θ est consistante avec ρ, θ satisfait aussi toutes les contraintes de H' et donc $\theta \in Solutions(H')$.

3.1.4 Exemples pratiques illustrant l'utilisation de l'annotation *lecture-seulement.*

Dans les langages de programmation logique, la notation lecture-seulement a été initialement introduite dans Concurrent Prolog [Sha86, Sar85, Mah87] pour un propos différent. Elle a été introduite pour le contrôle de communication et synchronisation dans des réseaux de processus.

Un exemple trivial mais beaucoup utilisé est une contrainte de la forme *A?+B?+C?=somme*. En peut voir qu'ici l'annotation *lecture-seulement* empêche l'utilisateur d'attribuer une valeur à la variable *somme* et d'avoir à propager ce changement à *A*, *B* ou *C*, mais permet à l'utilisateur d'attribuer une valeur à *A*, *B* ou *C*.

Comme il est mentionné au paragraphe 3.1.1, une utilisation importante de cette annotation est concrétisée dans des contraintes qui font référence à un support extérieur ou une source d'information située à l'extérieur du système. Par exemple si l'on a la contrainte : *le point P doit suivre le déplacement de la souris*. La contrainte doit être marquée par l'annotation *lecture-seulement* sur la position de la souris. *P = position.souris?*.

Considérons un autre exemple, supposons que l'on dispose d'un affichage avec une simple barre de défilement sur l'écran. Lorsque l'ascenseur se déplace entre le point haut et le point bas de la barre de défilement, on aimerait que les points haut et bas de la barre restent à leurs positions. Cependant, on aimerait être capable de repositionner la barre de défilement et donc le fait de fixer les points haut et bas de la barre ne constitue pas une solution correcte (on peut presque réaliser le résultat désiré en étiquetant par *fort* (mais non *requise*) des contraintes de fixation ou maintien des points haut et bas de la souris. Le problème est que d'autres contraintes sur les variables contenant les valeurs de position du curseur peuvent êtres étiquetées par *très-fort*, et par conséquent le point haut ou bas de la barre de défilement peut être changé). Pour résoudre ce problème efficacement, on définit une contrainte dans la figure 5. Cette contrainte est relative aux positions de l'ascenseur, des points haut et bas de la barre de défilement marqués par l'annotation *lecture-seulement* et d'une variable de pourcentage.

Il est bien évident que l'annotation sur les deux points haut et bas est spécifique à cette contrainte et donc la barre de défilement peut être repositionnée par une autre contrainte de déplacement.

FIGURE 5 : Contrainte de la barre de défilement relative à une position.

$$pourcentage = \frac{ascenseur - bas?}{haut? - bas?}$$

3.1.5 Circularité

Si pour plusieurs hiérarchies l'ensemble de solutions est intuitivement clair, cette clarté s'évanouit souvent lorsque la hiérarchie contient des contraintes qui forment un cycle. On présente deux exemples pour montrer ce phénomène [Wil92]. Il s'agit de cas pathologiques qui ne devraient pas être rencontrés dans les applications réelles, mais la théorie doit spécifier comment ils doivent être traitées.

Les deux hiérarchies de la figure 6 contiennent chacune un cycle entre leurs variables marquées.

En H_1, aucune contrainte dans le cycle n'est plus restrictive qu'une autre et donc l'information peut circuler et rendre une solution. Pour cette hiérarchie l'ensemble de solutions est l'ensemble infini $\{\{x \rightarrow d+1, y \rightarrow d\} / d \in R\}$.

En H_2, la contrainte $x? = y+1$ est plus restrictive que la contrainte $x \geq y?$. Ainsi l'information d'inégalité transite. Pour cette hiérarchie, l'ensemble de solutions est $\{\{x \rightarrow 21, y \rightarrow 20\}\}$.

FIGURE 6 : Hierarchies contenant des contraintes qui forment des cycles

$H_1 = \{$*requise* $x? = y+1$, *requise* $x = y?+1\}$ et

$H_2 = \{$*requise* $x? = y+1$, *requise* $y = 20$, *requise* $x \geq y?\}$.

3.1.6 L'annotation *écriture-seulement*

Outre l'annotation *lecture-seulement*, il convient aussi de définir la notation *écriture-seulement*. Lorsqu'une variable est marquée par cette notation, ceci veut dire que l'on veut que l'information circule de la contrainte vers la variable et non le contraire. On pourrait définir les effets de cette notation d'une manière identique à celle de la notation *lecture-seulement*. Cependant, il est plus facile de définir la notation *écriture-seulement* en terme de celle de *lecture-seulement*.

Définition 3.4 :

Soit H une hiérarchie de contraintes, et soit D le domaine des contraintes de cette hiérarchie. Soit $v_1, \ldots, v_n$ l'ensemble des variables de H. H a une ou plusieurs occurrences marquées par la notation *écriture-seulement*. Soient $w_1, \ldots, w_m$ des nouvelles variables non référencées dans H et soit J la hiérarchie résultante suite à la substitution de w_i par les occurrences de la variable v_i marquée (les occurrences de la variable v_i non marquées ne sont pas concernées). Soit J' la hiérarchie formée par ajout des contraintes *fortes* $v_i = w_i?$ pour $1 \leq i \leq m$ dans J. L'ensemble des solutions désiré de H est celui de J' en omettant les w_i.

Solutions(J) = *Solutions(J')* *omet* $w_1, \ldots, w_m$. La définition de *Solutions(J')* est celle donnée précédement.

Par exemple, soit H= *{ requise x! = y, forte x = 4, faible y = 3 }*. La contrainte $x = 4$ est étiquetée par *forte*, ce qui fait d'elle une contrainte plus importante à satisfaire que celle étiquetée par *faible*. Les informations ne sont autorisées à circuler que de la variable y vers la variable x dans la contrainte $x! = y$ puisque x est marquée par la notation *écriture-seulement*. En utilisant la définition, la hiérarchie J' est formée en remplaçant $x!$ par une nouvelle variable introduite w et en ajoutant la contrainte requise $x = w?$. Soit J' = { *requise* $w = y$, *requise* $x = w?$, *forte* $x = 4$, *faible* $y = 3$ }. L'ensemble de solutions à J' est $\{\{w \rightarrow 3, x \rightarrow 3, y \rightarrow 3\}\}$. L'ensemble de solutions à la hiérarchie H est celui de J' en omettant la nouvelle variable introduite, soit $Solutions(H) = \{\{x \rightarrow 3, y \rightarrow 3\}\}$.

3.2 Hiérarchies partiellement ordonnées

Dans certaines applications, le fait d'imposer un ordre total sur les étiquettes des contraintes (ordre total sur les niveaux de la hiérarchie) peut être vu comme une sur-spécification du problème. Pour remédier à ce problème, l'ensemble des solutions d'une hiérarchie à ordre partiel sur ces niveaux a été défini. Pour une hiérarchie partiellement ordonnée, l'ordre des étiquettes des contraintes non requises est partiel.

Informellement, l'ensemble de solutions d'une hiérarchie partiellement ordonnée est défini par la formation de l'ensemble de toutes les hiérarchies à ordre total et qui soient consistantes avec la hiérarchie initiale. Ces hiérarchies à ordre total sont obtenues par l'ajout des ordres permissibles entre l'ordre partiel sur les niveaux de la hiérarchie initiale : *inférieur à* , *supérieur à* ou *égal à*. L'ensemble des solutions désiré est l'union des ensembles de solutions de ces hiérarchies.

Définition 3.5 :

Si H est une hiérarchie à ordre partiel sur ces niveaux, la hiérarchie H' à ordre total sur ces niveaux est construite si :

Les deux hiérarchies H et H' contiennent les mêmes contraintes

Il existe une correspondance m des étiquettes de H à celles de H' telle que

si $s_i < s_j$ dans H alors $m(s_i) < m(s_j)$ dans H'

$\forall i\ s_i\ c_i \in H\ s_i$ et seulement si $m(s_i)\ c_i \in H'$.

Définition 3.6 :

Soit H une hiérarchie à ordre partiel sur ces niveaux alors :

$Solutions(H) = (\cup_{H' \in \mathrm{H}} Solutions(H'))$ où est H l'ensemble de toutes les hiérarchies à ordre total sur leurs niveaux et constantes avec H.

Comme exemple trivial, considérons la hiérarchie suivante : H = *{ petit x=3, faible x=4 }*. Les étiquettes *petit* et *faible* sont toutes les deux non requises et il n'y a pas d'ordre spécifié entre elles. Les ordres totaux qui sont consistants avec cet ordre partiel sont : *petit* est *supérieur à faible*, *petit* est *inférieur à faible* et finalement *petit* est *égal à faible*. En utilisant le comparateur local *localement-prédicat-meilleur* les ensembles de solutions de ces hiérarchies sont respectivement : $\{\{x \rightarrow 3\}\}$, $\{\{x \rightarrow 4\}\}$ et $\{\{x \rightarrow 3\}, \{x \rightarrow 4\}\}$ et donc l'ensemble de solutions à la hiérarchie H est : $\{\{x \rightarrow 3\}, \{x \rightarrow 4\}\}$.

La définition 3.6 entraîne l'ajout de tous les ordres entre les étiquettes de la hiérarchie. Pour les comparateurs locaux, l'ordre *égal à* est non nécessaire puisque chaque solution à la hiérarchie d'ordre total *égal à* est aussi une solution à l'une des hiérarchies à ordre total dans *{inférieur à , supérieur à}*. Ceci n'est pas le cas pour les comparateurs globaux. Considérons par exemple le comparateur *Moindre-Carré*, dans l'exemple précédent les solutions des hiérarchies à ordre total sont respectivement $\{\{x \rightarrow 3\}\}$, $\{\{x \rightarrow 4\}\}$ et $\{\{x \rightarrow 3.5\}\}$ et donc l'ensemble de solutions à la hiérarchie H est : $\{\{x \rightarrow 3\}, \{x \rightarrow 4\}, \{x \rightarrow 3.5\}\}$.

Borning et son équipe dans [BMM+89] considèrent une variante de la définition des solutions aux hiérarchies à ordre partiel. Non seulement deux étiquettes à ordre partiel sont combinées en une seule (i.e *égal à*) mais aussi tous les poids possibles entre les contraintes doivent être utilisés. Dans l'exemple précédent et en utilisant le comparateur global *Moindre-Carré*, l'ensemble infini de hiérarchies à ordre total est : *{{fort x=3, très-faible x=4}, {très-faible x=3, fort x=4}, {moyen[w_1] x=3, moyen[w_2] x=4} pour tout nombre positif w1 et w2}*. L'ensemble de solutions dans ce cas est $\{\{x \rightarrow a\} / a \in [3..4]\}$.

3.3 Les fonctions objectifs

L'objectif dans certains problèmes standards basés sur la programmation linéaire est de minimiser (ou maximiser) la valeur de la fonction linéaire $z(x_1, ..., x_k) = a_1 x_1 + ... + a_k x_k$ comportant k variables de domaine de définition R^{+*} [Mur83]. Le programme contient m contraintes linéaires d'égalité ou d'inégalité sur $x_1, ..., x_k$, et contient aussi des contraintes dite non négatives $x_1 \geq 0, ..., x_k \geq 0$. La fonction à minimiser (ou à maximiser) est appelée fonction objectif.

Lorsqu'il s'agit de minimiser[1] la fonction objectif étant donnés les coefficients positifs de z, on peut facilement représenter le problème de programmation linéaire par une hiérarchie de contraintes. Les k contraintes non négatives et les m contraintes linéaires d'égalité ou d'inégalité seront représentées dans une hiérarchie comme des contraintes requises. La fonction objectif sera représenté par $z(x_1, ..., x_k) = 0$ et sera considérée comme une contrainte de préférence dans la hiérarchie (puisqu'on connaît *a priori* la borne inférieure (qui est 0) de la valeur de la fonction objectif). Cependant, si la borne inférieure n'est pas connue, cette transformation ne peut pas être fiable. On peut mettre une borne ou un objectif g à la fonction objectif et décider que l'on peut être satisfait si l'on atteint ou si l'on dépasse cet objectif g. Dans ce cas, la fonction objectif est représentée par la contrainte de préférence suivante : $z(x_1, ..., x_k) \leq g$.

Une autre approche serait de représenter la fonction objectif comme la contrainte de préférence $z'(x_1, ..., x_k) = 0$ avec :

$$z'(x_1, ..., x_k) = \begin{cases} -1/z(x_1, ..., x_k) \text{ si } z(x_1, ..., x_k) < -1 \\ z(x_1, ..., x_k) + 2 \text{ si } z(x_1, ..., x_k) >= -1 \end{cases}$$

L'inconvénient de cette approche est de convertir un problème linéaire en un problème non linéaire plus difficile à résoudre. Pour remédier à cet inconvénient, la théorie des hiérarchies de contraintes est étendue. Cette extension consiste à inclure explicitement les fonctions objectifs.

1 Les mêmes arguments décrits pour la minimisation de la fonction objectif peuvent être appliqués lorsqu'il s'agit de maximiser la fonction objectif.

Les fonctions doivent être étiquetées par des étiquettes différentes de l'étiquette *requise*. Les fonctions objectifs $z(x_1, ..., x_k)$ à maximiser sont remplacées par des fonctions objectifs $0\text{-}z(x_1, ..., x_k)$ à minimiser. Soit Z_i l'ensemble des fonctions objectives au niveau i de la hiérarchie. La définition du comparateur *localement-meilleur* est donc étendu comme suit (l'expression $z\theta$ dénote la valeur de $z(x_1\theta, ...,x_k\theta)$ c.à.d la valeur de z lors de l'application de θ à $(x_1, ..., x_k)$).

Définition 3.7 :

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

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

$(\forall r\ v_r \leq u_r$ avec $v_r \in g(E(H_k\,\theta))$ et $u_r \in g(E(H_k\,\eta))) \wedge$

$(\forall z \in Z_k\ z\theta \leq z\eta\ ) \wedge$

$(\ (\exists j\ v_j < u_j$ avec $v_j \in g(E(H_k\,\theta))$ et $u_j \in g(E(H_k\,\eta))\ ) \vee (\exists z \in Z_k\ z\theta < z\eta\ ))$

avec $g(V) = V$ et $<>_g$ est définit par : $V <>_g U \equiv \forall j\ v_j = u_j$.

En d'autres termes, pour que θ soit *Localement-Meilleur* que η, les effets de θ doivent être identiques à ceux de η pour toutes les contraintes et les fonctions objectives jusqu'au niveau k-1 de la hiérarchie, et au niveau k, θ doit faire mieux que η pour au moins une contrainte ou une fonction objective et mieux ou identique pour tout le reste (i.e. les contraintes et les fonctions objectives du niveau k). Cette extension est dans le même esprit que la définition initiale du comparateur *Localement-Meilleur*, c.à.d. les fonctions objectives et les contraintes sont considérées individuellement.

Le comparateur *Globalement-Meilleur* combine les erreurs d'un niveau donné de la hiérarchie. La borne inférieure des erreurs des contraintes est égale à 0 tandis qu'en général la fonction objectif n'a pas de valeur minimale définie et donc la combinaison de ces deux valeurs en une seule valeur composée ne semble pas très correcte. Par conséquent, lors de l'utilisation des comparateurs globaux, on limite la hiérarchie de contraintes contenant aussi des fonctions objectifs en une hiérarchie telle qu'à chacun de ses niveaux on ait soit des contraintes soit une fonction objectif (dans le cas où il y a plusieurs fonctions objectifs à un niveau, ces fonctions doivent être remplacées par une seule fonction qui combine les différentes valeurs de ces fonctions objectifs). L'extension du schéma du comparateur *Globalement-Meilleur* est défini par :

Définition 3.8 :

Globalement-Meilleur$(\theta, \eta, H, g) \equiv \exists k \in 1...n /$

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

$(\ g(E(H_k\,\theta) <_g g(E(H_k\,\eta)) \vee (z_k\theta < z_k\eta\ ))$.

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.

Par convention ici, si i est un niveau de la hiérarchie contenant des contraintes, $g(E(H_i\,\theta))$ est défini comme avant et $z_i\theta$ est égal à 0. Tandis que, si i est un niveau contenant une fonction objectif alors $g(E(H_i\,\theta))$ est défini de manière à ce qu'elle soit égale à 0, et $z_i\theta$ est la valeur de la fonction objectif à ce niveau.

3.4 Variation du critère selon les niveaux d'une hiérarchie

Une extension de la théorie des hiérarchies de contraintes a été proposeé par Hiroshi Hosobe dans [HKS+94]. Cette extension consiste à formaliser le cas où les modes de combinaison d'erreur peuvent varier selon les niveaux de la hiérarchie. Par exemple, dans une hiérarchie contenant trois niveaux d'importance, on peut utiliser pour les deux premiers niveaux un critère local (exemple le comparateur *Localement-Prédicat-Meilleur*[1]) pour lequel une solution sera meilleure qu'une autre si elle satisfait un sur-ensemble des contraintes satisfaites par la seconde. Pour le troisième niveau, un comparateur plus fin en général global (par exemple, le comparateur *Moindre-Carrés* ou encore le comparateur *Pire-Cas)* qui impose un ordre total sur les valuations, peut être utilisé pour départager les solutions. Dans ce formalisme, on associe à chaque niveau de la hiérarchie un critère. La solution produite par ce critère est dite *solution type.*

Dans ce mémoire, on se réfère à une *solution type* associée avec le comparateur *Localement-Prédicat-Meilleur* (resp. *Localement-Métrique-Meilleur, Somme-Pondérée-Métrique, Pire-Cas-Metrique, Moindre-Carrés-Métrique*, etc ...) par τ_{LPM} (resp. τ_{LMM}, τ_{SPM}, τ_{PCM}, τ_{MCM}, etc ...) et des contraintes de τ_{LPM} (resp. τ_{LMM}, τ_{SPM}, τ_{PCM}, τ_{MCM}, etc ...) comme des contraintes de *Localement-Prédicat-Meilleur* (resp. *Localement-Métrique-Meilleur, Somme-Pondérée-Métrique, Pire-Cas-Metrique, Moindre-Carrés-Métrique*, etc ...). Dans le chapitre 8 de ce mémoire, nous reviendrons plus en détail sur cette extension.

3.5 Intégration des hiérarchies de contraintes en *PLC*

Récemment, le paradigme des hiérarchies de contraintes a été intégré avec le schéma de programmation logique par contraintes (*PLC*) afin de produire la programmation logique par hiérarchie de contraintes (*PLHC*) [Wil92, DVS+88, BMM+89].La *PLC* et la *PLHC* sont paramétrées par *D* qui est le domaine des contraintes. De plus la *PLHC* est paramétré par le comparateur *C*. Cette intégration de hiérarchies de contraintes à la *PLC* permet à la fondation théorique de la programmation logique d'être complété par l'expressivité des contraintes de préférence. Nous reviendrons en détail sur cet aspect dans la deuxième partie de ce mémoire.

1. Le comparateur *Localement-Prédicat-Meilleur* est une variante de *Localement-Meilleur* utilisant la fonction d'erreur prédicat (qui retourne 0 si la contrainte est satisfaite et 1 sinon)

Synthèse du chapitre

Les différentes parties de ce chapitre présentent les extensions de la théorie des hiérarchies de contraintes. La plupart de ces extensions se manifestent par l'extension des définitions des comparateurs locaux ou globaux.

Les intérêts de ces extensions sont nombreux, par exemple :

la limitation du choix des variables à recalculer pour resatisfaire les contraintes après une perturbation introduite dans le système en intégrant les notations *lecture-seulement* et *écriture-seulement* sur ces variables,

la faculté de pouvoir intégrer des fonctions objectifs dans des hiérarchies de contraintes en redéfinissant les comparateurs,

un remède aux sur-spécifications de certains problèmes (c.à.d ordre total sur les niveaux des hiérarchies) par la définition de l'ordre partiel sur les niveaux des hiérarchies,

la possibilité de faire varier le critère de comparaison selon les niveaux d'une hiérarchie en formalisant le cas où les modes de combinaison d'erreur peuvent varier selon les niveaux de la hiérarchie,

et enfin l'intégration des hiérarchies de contraintes dans le paradigme de la Programmation Logique par Contrainte pour obtenir la Programmation Logique par Hiérarchie de Contraintes.

4 Algorithmes de propagation locale

En préambule de ce chapitre, nous dirons que la recherche d'un algorithme efficace qui pourrait satisfaire tout type de contraintes en utilisant n'importe quel domaine et n'importe quel comparateur serait un essai futile. Ce chapitre donne un aperçu des différents algorithmes existants pour le traitement des hiérarchies de contraintes. Ce chapitre décrit la réponse à la question essentielle posée avant de concevoir tel ou tel algorithme, et qui est : "que fait le système lorsque l'ensemble des contraintes est un ensemble sur-contraint (pas de solution qui satisfasse cet ensemble) ou lorsque l'ensemble des contraintes est un ensemble sous-contraint (il y a plusieurs solutions qui satisfont cet ensemble)". Si le résolveur gère les contraintes d'une application d'interface utilisateur, alors il n'est pas acceptable de signaler une erreur de manipulation. La théorie des hiérarchies de contraintes décrite dans les chapitres précédents indique une voie pour spécifier déclarativement comment le résolveur doit réagir face à cette situation.

Une hiérarchie de contraintes est un ensemble de contraintes où chaque contrainte possède une étiquette qui exprime l'importance de cette contrainte dans le système (ou la préférence de cette contrainte par rapport aux autres contraintes). Etant donnée une hiérarchie de contraintes sur-contrainte, le résolveur peut laisser certaines contraintes parmi les moins préférées non satisfaites pour satisfaire celles qui sont les plus préférées. Si la hiérarchie est sous-contrainte, le résolveur peut choisir n'importe quelle solution. L'utilisateur peut contrôler quelle solution choisir en ajoutant des contraintes étiquetées par des étiquettes faibles. Ces contraintes pourront porter sur les variables pour exprimer le maintien de leurs valeurs (i.e. l'utilisateur exprime via ces contraintes son désir de ne pas changer les valeurs de certaines variables). Ce chapitre discutera aussi un peu plus en détail des algorithmes basés sur la propagation locale. Nous nous attarderons sur quelques aspects intéressants d'ingénierie de certains algorithmes, sur les critères de comparaison utilisés et enfin sur la complexité de ces algorithmes.

Ce chapitre est composé de quatre parties. La première décrit le principe de la propagation locale. La deuxième partie décrit quelques algorithmes existants basés sur la propagation locale pour la résolution de hiérarchie de contraintes fonctionnelles. La troisième partie donne un aperçu sur des algorithmes existants pour la résolution de hiérarchies ayant des contraintes d'égalité et d'inégalité. Et enfin la quatrième partie donne un aperçu sur d'autres algorithmes existants.