La résolution des réseaux de contraintes algébriques et qualitatives : une approche d'aide à la conception en ingéniérie

Vu Nguyen Duong

ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

THÈSE

pour le diplôme de

DOCTEUR de l'ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

par

Vu Nguyen DUONG

Directeur de Thèse : Jacques RILLING

La Résolution des Réseaux de Contraintes Algébriques et Qualitatives : Une Approche d'Aide à la Conception en Ingénierie.

Soutenue le 16 Novembre 1990 devant le Jury composé de :

PRÉSIDENT	:	M. JOHN CAMPBELL,	PROFESSEUR
RAPPORTEURS	:	M. ERICK GAUSSENS,	DR. ÈS SC.
		M. BERTRAND DELCAMBRE,	DIR. RECHERCHE, IPC
EXAMINATEURS	:	M. FRANÇOIS ARLABOSSE,	DIR. RECHERCHE
		M. ALAIN CONSTANS,	PROFESSEUR
		M. ALBERT DUPAGNE,	PROFESSEUR
		M. JACQUES RILLING,	PROFESSEUR

"And so once more
I lived through the brief, beautiful, difficult
and exciting time in which
a narrative passes its crisis, the time in which
all the thoughts and moods
that have a bearing on the mythical figure
stand before me with the greatest
sharpness, clarity, and urgency."

Hermann Hesse (A Night's Work. 1928)

À Mỹ-Châu et Viêt-An.
À ma Grande Famille.

Je tiens à témoigner ma sincère reconnaissance à Monsieur Jacques Rilling, Professeur à l'Ecole Nationale des Ponts et Chaussées, Directeur Scientifique au CSTB, qui m'a initié à la recherche, et a dirigé avec beaucoup de bienveillance cette thèse, et ceci malgré son emploi du temps très chargé. Son enseignement a été déterminant dans mes reflexions scientifiques. J'exprime ici toute ma gratitude.

Je voudrais exprimer toute ma reconnaissance à Monsieur le Professeur John Campbell de University College of London qui m'a honoré pour accepter d'être Président du Jury; à Messieurs les Professeurs Albert Dupagne de l'Université de Liège et Alain Constans de l'Université Pierre et Marie Curie, qui m'ont honoré de leurs présences le jury.

Une partie du présent travail a été réalisée au Département Recherche et Développement de Framentec, que son chef, François Arlabosse recoive ici mes sincères remerciements.

Je remercie particulièrement Erick Gaussens, pour sa gentillesse, pour les conseils qu'il a su me prodiguer, le temps qu'il a bien voulu me consacrer tout au long de mes travaux et pour avoir accepté d'être rapporteur de cette thèse.

Je tiens à remercier Monsieur Bertrand Delcambre, Ingénieur en Chef des Ponts et Chaussées, Chef du Servive Informatique et Bâtiment au CSTB Sophia Antipolis qui a bien voulu discuter ce travail depuis son début et me faire l'honneur d'être rapporteur de cette thèse.

Mes remerciements et l'expression de ma sympathie vont au Professeur Benjamin Kuipers de l'Université de Texas à Austin qui ma fourni l'outil QSIM et à Michel Curtat, chef de division Modélisation Feu au CSTB qui m'a aidé dans le réglement des questions sur la modélisation physique.

Je remercie également toutes les personnes qui ont contribué directement ou indirectement à la rédaction de ce mémoire. Je pense en particulier à Nathalie Porté pour avoir relu plusieurs versions de ce mémoire ; à Rémi Saccoman qui m'a répondu à toutes questions sur LaTeX TGif, PostScript ; à Daniel Gureghian, Philippe LePage, Alain Berger, Jean-Bernard Thevenon, Simon King, Suzanne Bléry, Jérôme Thoméré et à tous les autres membres de l'équipe R&D de Framentec qui m'ont encouragé. Je leur exprime toute ma reconnaissance et toute ma sympathie. J'oublie certainement plusieurs, qu'ils me pardonnent.

Les encouragements les plus appréciés sont dus à mon cher ami Guillaume Bailby qui m'a soutenu dans les moments les plus difficiles, qu'il accepte ici mes remerciements.

Enfin, je voudrais dire à Mỹ-Châu, ma femme, que j'ai beaucoup apprécié son soutien et cette thèse sera notre cadeau de bienvenue au monde à Viêt-An.

RÉSUMÉ

La thèse est que la représentation par contraintes, en terme de langage et la déduction par la gestion automatique des *réseaux de contraintes* est une voie adéquate pour les système d'aide à la conception en ingénierie. Deux techniques de l'Intelligence Artificielle à savoir la *Propagation de Contraintes* et le *Raisonnement Qualitatif* sont abordées.

La *Propagation de Contraintes* est un mécanisme d'inférences déductives sur les réseaux de contraintes. Ce mécanisme présente plusieurs difficultés, particulièrement dans les calculs pas-à-pas de la résolution des relations entre valeurs des variables. Pour améliorer ce mécanisme, il est proposé dans le présente travail des algorithmes de *filtrage* sur la consistance locale dans des réseaux de contraintes algébriques n-aires. Ainsi, le concept de réseaux dynamiques est exploité comme un module d'aide à la conception.

Le *Raisonnement Qualitatif* permet aux concepteurs d'analyser les modèles quand la connaissance est incomplète. L'approche de la *Simulation Qualitative* est proposée. L'adéquation de celle-ci est illustrée par d'une part, la représentation de connaissances incomplètes en termes de contraintes qualitatives, et d'autre part, sa capacité de saisir les comportements qualitatifs possibles des modèles.

Un ensemble de logiciels est développé et leur efficacité démontrée sur des cas simples.

mots-clés : intelligence artificielle, ingénierie, conception assistée, réseau de contraintes, problème de satisfaction de contraintes, propagation de contraintes, algorithme de filtrage, physique qualitative, raisonnement qualitatif, simulation qualitative,

NOTE DE SYNTHÈSE

La thèse que nous présentons est que la représentation par contraintes, en terme de langage, et la déduction par la gestion automatique de réseaux de contraintes, sur le plan de raisonnement, est une voie potentiellement adéquate pour les systèmes d'aide à la conception.

Ce travail présente un ensemble de problèmes relatifs à la résolution automatique des réseaux de contraintes dynamiques. Les concepts fondamentaux dans notre approche de résolution sont :

1. DÉCOMPOSITION UNIFORME : tout problème physique peut être décomposé en termes de réseaux de contraintes. Chaque noeud de ce type de réseau représente une variable ou un paramètre physique et chaque contrainte représente une relation entre un groupe de paramètres. La dynamique d'un modèle physique ne repose pas seulement sur la dynamique des groupes de noeuds mais aussi sur la dynamique des groupes de relations.

2. REPRÉSENTATION PAR CONTRAINTES : la dynamique des relations joue un rôle essentiel dans la conception. Le problème de conception correspond en fait au problème de résolution des réseaux de contraintes dynamiques, c'est à dire le problème d'analyse de la consistance entre les paramètres, ses domaines de valeurs et les contraintes exerçées sur ces valeurs en prenant en compte la possibilité de manipuler toutes les variables, relations ou valeurs dynamiquement.

3. ANALYSE PAR CAUSALITÉ : la gestion de contraintes ne se restreint pas seulement au problème d'analyse de la consistance d'un réseau. L'explication des solutions consistantes, l'élimination des contraintes redondantes et la résolution de l'incohérence, *etc.* sont en effet, les fonctions d'aide à la conception. Elles reposent sur la notion de la causalité : il n'y a jamais de fumée sans feu.

4. DÉDUCTION PAR L'ÉLIMINATION : le mécanisme de déduction consiste à rechercher le domaine de "falsification" du problème posé, i.e. les variables, valeurs et contraintes qui ne satisfont pas les règles de consistance, puis à les éliminer de l'espace de recherche. Ainsi, quand l'espace de solutions est petit, le risque de l'explosion combinatoire dans la recherche de solution est minimisé. On peut apparenter cette démarche à celle de Karl Popper.

Une grande partie de notre travail consiste à développer un formalisme sur les propriétés de la consistance locale dans un réseau de contraintes n-aires. A savoir, la consistance de n-arc, la consistance de chemin pour les contraintes n-aires. A partir de ce formalisme, nous proposons les algorithmes de filtrage sur la consistance des domaines de valeurs associés aux variables et des contraintes connectées à celles-ci. Enfin, un mécanisme de résolution et de gestion de réseaux de contraintes n-aires dynamiques est dévéloppé dans l'optique d'aide à la conception.

Pour cela, nous analysons d'abord comment le concept des réseaux de contraintes dynamiques peut être exploité. Nous avons trouvé qu'à chaque modification d'un modèle, soit progressivement par la démarche du type *incrémental*, soit dynamiquement par la démarche du type *génération-vérification*, un réseau change en conséquence. Le problème est donc de définir un mécanisme de gestion de ces réseaux, particulièrement le mécanisme de recherche de solutions cohérentes. Ce problème est parfois appelé le *"Problème de Satisfaction de Contraintes"*.

Nous examinons ensuite, l'état de l'art de cette technique. En effet, la solution du problème de satisfaction de contraintes est obtenue par un algorithme de *"propagation de contraintes"*. Cet algorithme est un mécanisme d'inférences déductives sur les contraintes, attribue des valeurs aux variables en satisfaisant toutes les contraintes sur les variables du réseau. Une solution existe si les contraintes sont satisfaites et les variables ont des valeurs. Cette étude nous permet d'identifier la position des problèmes qui nous sont posés par notre approche et les éléments théoriques correspondants. Puis, à partir de ces éléments et des travaux existants traitant les contraintes binaires, particulièrement de ceux de Freuder, Mackworth, Dechter, Pearl et Steele Jr, nous dévéloppons le formalisme ainsi que les algorithmes que nous avons cité ci-dessus. De ces algorithmes, nous dévéloppons un ensemble de logiciels de résolution et de gestion des réseaux de contraintes algébriques, que nous l'appelons NETGEN, pour tester nos idées et notre approche.

Une des difficultés que nous avons rencontré est celle de l'analyse des problèmes dynamiques i.e. la résolution automatique des systèmes d'équations différentielles avec l'approche de contraintes. Celle-ci nous a amèné à l'étude de la technique de l'analyse qualitative. Plusieurs tentatives ont été effectuées au long de ce travail pour rechercher un mécanisme de gestion des contraintes qualitatives homogène avec celui traitant les contraintes algébriques. La première a consisté à introduire la notion de contraintes fonctionnelles (i.e. contraintes sur les tendances de croissance ou de décroissance des valeurs de paramètres) pour gérer la dynamique des systèmes. Elle n'a pas donné de résultats positifs. La seconde consiste à introduire la méthode de Vaschy-Buckingham dans l'analyse dimensionnelle pour générer les relations d'ordres entre paramètres et à gérer ces relations par une opération sur les contraintes. Cette dernière tentative nous semble faisable mais nécessite une recherche plus approfondie associant l'étude de la logique évènementielle. Finalement, après une analyse sur l'état de l'art de la technique du *Raisonnement Qualitatif*, encore appelé la *"Physique Qualitative"*, nous trouvons que l'approche de la *Simulation Qualitative*, initiée par Kuipers en 1985, présente plusieurs points communs avec notre approche, particulièrement par les extensions récentes concernant résolution des contraintes qualitatives d'ordres supérieurs et l'introduction des valeurs numériques dans la simulation.

L'algorithme de *Simulation Qualitative* QSIM présenté est en d'autres termes, une forme de résolution et de gestion des réseaux de *contraintes qualitatives*. L'adéquation de cette approche est illustrée par d'une part, la représentation de connaissances incomplètes des modèles physiques par des équation différentielle qualitatives, et d'autre part, sa capacité à prédire ou à saisir les descriptions qualitatives des variations dynamiques des paramètres.

A travers cette étude exploratoire, nous concluons que l'approche de résolution des réseaux de contraintes algébriques et qualitatives est une voie adéquate pour les systèmes d'aide à la conception en ingénierie.

Chapitre 1
Introduction

"Les machines avaient le type d'imagination idéale, la capacité de construire un avenir nécessaire à partir d'un fait présent. L'Homme, lui, avait une imagination d'un type différent : l'imagination illogique et brillante qui ne voit que vaguement l'avenir, sans connaître ni le pourquoi, ni le comment : une imagination qui surpasse la machine et sa précision."

John W. Campbell (The Ultimate Machine)

1.1 Champs de Recherches

La conception, aussi bien pour un artiste ou un ingénieur, est un art. L'informatisation de cet art est pratiquement impensable, mais des systèmes de conception assistée par ordinateur peuvent apporter leur aide. Cependant, ces systèmes ne fûrent conçus qu'en deux dimensions: verticalement et horizontalement. Verticale puisqu'ils se comportent comme un ensemble d'outils, analogue à des pinceaux pour un peintre. Horizontale, comme une pallette de couleurs, prêtes à être appliquées sur une toile vierge. Ces dimensions peuvent-elles vraiment aider les ingénieur-concepteurs à conçevoir ? ou ne peuvent-elles donner à un concepteur que les moyens de matérialiser ses objets de conceptions ?

Il a été rapidement identifié le besoin d'une autre dimension: la profondeur, où l'expression des concepts peut impliquer un sens, une image mentale et où les liens entre concepts sont bien compris par l'ordinateur. Il apparaît alors la nécessité d'incorporer des techniques issues de l'inteligence artificielle dans les systèmes d'aide à la conception pour un certain

domaine d'applications[1].

Cette thèse n'a pas l'ambition de creuser en profondeur ces systèmes. Le travail se restreint à une étude exploratoire d'un moyen automatisé, permettant aux ingénieurs-concepteurs d'exprimer et de raisonner sur une image particulière : *les modèles algébriques et qualitatifs* de la connaissance physique d'un domaine spécifique, l'ingénierie de bâtiment.

Nous proposons la modélisation par les contraintes algébriques et qualitatives et la résolution de ces modèles comme une approche d'aide à la conception en ingénierie. Cette approche est fondé sur le principe de relations dynamiques entre paramètres.

1.2 Approche de Relations Dynamiques

La conception en ingénierie de bâtiment est une activité complexe. Cette complexité s'exprime à travers la coordination entre plusieurs acteurs spécialistes de leurs sous-domaines, à savoir la conception architecturale, la conception des structures porteuses, la conception des installations thermiques, ventilations, acoustiques, protections contre l'incendie etc.

Comme un orchestre qui joue une symphonie et qui n'a qu'une suite de mélodies au lieu des partitions dédiées à chaque instrument. Comment les musiciens peuvent-ils s'exprimer ? D'où vient cette harmonie dans la communication? De leur intuition respective ? De leur "bon sens" ou bien est-ce le fruit d'une expertise ?

En effet, à travers ces lignes de mélodie, les notes représentent non seulement des *contraintes* mais aussi des *concepts*, des expressions mentales. En bref, il y a une image, un *modèle abstrait* décrivant les *contraintes* entre

[1]par exemple HI-RISE [Maher 1984], DESTINY [Sriram 1984], etc.

concepts ; il représente une *compréhension* entre ces musiciens.

Le prérequis le plus important de la capacité de "comprendre" est une représentation interne, ou un modèle approprié, capable d'enregistrer des informations. Dans l'ingénierie, un modèle physique est une expression de la compréhension d'un phénomène physique ; En Informatique, un modèle de calcul (computational model) est l'expression de la compréhension d'un processus de calcul.

Le processus de modélisation des systèmes physiques est une tâche souvent très compliquée et itérative. Un modèle, représentant l'abstraction du système, ne peut décrire que certains aspects de sa nature dans un environnement réel. Néanmoins, on peut considérer un modèle physique comme une collection *d'individus* et de *relations* en interactions pour transformer un ensemble de quantités d'entrée Q_{in} en un ensemble de quantités de sortie Q_{out}. Le processus de transformation est effectué en accord avec un ensemble de fonctions ou règles de transformation R_i. Par l'analogie à la définition des systèmes formels, on peut définir un modèle physique comme suit:

Définition 1 *Le modèle $\Re$ d'un phénomène physique φ est une représentation abstraite d'un couple $(\{O\}, \{\mathcal{F}\})$:*

$$\Re(\varphi) = (\{O\}, \{\mathcal{F}\}) \tag{1.2.1}$$

- $\{O\}$*: ensemble des objets représentant les entités du phénomène, i.e. paramètres physiques, sous-modèles, etc...Chaque élément de* $\{O\}$ *peut avoir des valeurs variables ou constantes.*
- $\{\mathcal{F}\}$*: ensemble des relations entre les elements de O, i.e. lois physiques, couplages, etc. Chaque élement de* $\{\mathcal{F}\}$ *est utilisé*

pour déduire les valeurs de certains autres éléments de $\{O\}$ *à partir des valeurs connues.*

- *Soit* $\{Q_{in}\}$*, l'ensemble des valeurs connues des éléments de* $\{O\}$*,* $\{Q_{out}\}$ *obtenu par l'application de* $\{\mathcal{F}\}$ *sur* $(\{O\}, \{Q_{in}\})$*. L'expression de la variation d'un phénomène dans le temps peut se représenter comme ci-dessous:*

$$\mathcal{F}(\{O\}, \{Q_{in}(t)\}) = \{Q_{out}(t)\} \tag{1.2.2}$$

En d'autres termes, le problème de représentation des modèles physiques revient donc à la formalisation des élements de l'ensemble des relations $R_i \in \{\mathcal{F}\}$ entre entités $x_i \in \{O\}$ du phénomène. La causalité est la notion de relation $R_{i,\ldots,k} \in \{\mathcal{F}\}$ entre "cause et effets" établis par un ou plusieurs changements de valeurs des paramètres $x_i, \ldots, x_k \in \{O\}$.

Cette notion couvre l'ensemble des relations logiques et fonctionnelles entre les valeurs attribuées à l'ensemble d'entités décrivant le modèle. Deux types de relations causales sont considérés : logique et fonctionnelle.

La causalité logique implique l'utilisation de notions de constantes, variables, prédicats, connexions axiomatiques pour représenter le monde en terme de formules dans une certaine axiome formelle du premier ordre, second ordre, multi-valeurs, modale, floue, etc. Un modèle physique, dans ce sens, est une collection de formules logiques dont le but est de fournir une description partielle d'un état. Par exemple, le transfert thermique entre deux milieux (solides ou gazeux) est due à la différence de température de ces deux milieux.

La causalité fonctionelle implique l'utilisation de notions d'états ou d'évènements pour représenter le dynamique du monde physique en terme

de variations dans le temps des paramètres physiques representés avec la causalité logique. Par exemple, l'équilibre thermique entre deux milieux est décrit par une séquence d'états commençant par un étatéquilibre initial, une perturbation (différence de températures), et une succession d'états de transfert qui mène le système à un équilibre.

Mais comment un ordinateur peut *comprendre* cette notion de causalité ? et comment peut-il raisonner sur la causalité "intelligemment" ?

Marvin Minsky dans son ouvrage sur le traitement d'information sémantique a cité que :

"In discussing "intelligence", people are usually too concerned with the peaks of human creativity innovations. First-rate creativity is not the most common nor the most essential quality of intelligence. ... . We have to find how to represent and use *concepts before we can work directly upon their invention."*

Il est vrai qu'avant toute discussion sur la compréhension, le langage est le moyen *sine qua non* pour la transmission d'idées et est le support du raisonnement.

Le langage est une structure syntaxique et sémantique servant à transmettre des idées. Un modèle est un langage qui englobe la puissance de déduction. Néanmoins, certains modèles sont plus *intéressants* que d'autres par-ce qu'ils peuvent évoquer des images mentales et des analogies que d'autres n'évoquent pas. En outre, ces images et analogies peuvent guider les reflexions sur le problème à résoudre. La recherche d'un modèle de représentation de connaissance *interessant* revient à un sujet classique en Intelligence Artificielle : le problème de la structure interne de l'information. Quelle structure permet d'exprimer la sémantique des concepts et, en

même temps, faciliter la déduction à travers ces liens ?

Le raisonnement est, par contre, fondamentalement une pratique individuelle. Sa formalisation permet d'en faire une technique. Cette technique est transmissible et peut être un objet de recherche spécifique. Il est bien connu dans le monde de l'Intelligence Artificielle que la puissance d'expression d'un langage de représentation conditionne l'efficacité des raisonnements qu'il peut décrire. Néanmoins, le raisonnement demeure complexe puisqu'il reflète, entre autres, un style de résolution. Ce style est une démarche qui n'est pas repétitive mais qui correspond à un mode de penser, d'analyser et de synthétiser. Il est comparable à un modèle mental : il évoque des images et des analogies. Les images sont les principes du raisonnement, les analogies sont les déductions sur les causes et effets des instances de ces images.

Il ne faut cependant pas penser qu'il suffit d'élaborer un langage aussi performant soit-il pour rendre simple le raisonnement ; le raisonnement demeure complexe. En particulier, il doit refléter le style de résolution de chaque individu.

Un progrès sensible dans l'automatisation du processus de conception serait d'élaborer un langage de représentation et un outil de déduction qui ne conditionnent pas, de manière directe, le style de résolution de chaque individu.

La thèse que nous présentons est que la représentation par contraintes, en terme de langage, et la déduction par la gestion automatique de réseaux de contraintes, sur le plan de raisonnement, est une voie potentiellement adéquate.

Nous allons essayer de démontrer la faisabilité de cette démarche sur

une classe limitée de problèmes. Un élement important de cette démonstration est le développement de l'ensemble de logiciels réalisé.

1.3 Contenu de ce Mémoire

La méthode de contraintes algébriques ne s'attache qu'aux problèmes statiques. Les difficultés liées aux problèmes dynamiques nous ont amèné à considérer la technique d'analyse qualitative.

La suite de ce mémoire est organisée en deux parties. La première, du chapitre 2 au 5, présente la technique de résolution et de gestion des réseaux de contraintes ; La deuxième partie, du chapitre 6 au 8, décrit la technique de raisonnement qualitatif.

Le chapitre 2 introduit l'approche de conception par la manipulation des modèles de contraintes.

Le chapitre 3 introduit l'état de l'art de la représentation par contraintes et les fondements théoriques de l'approche proposée. Dans la section 3.1, nous définissons la notion de relations entre deux entités ainsi que la notion de contraintes. En fait, la déductivité du langage utilisé s'effectue via une technique dite *instanciation* ou *propagation de contraintes.* Un panorama global des techniques de la propagation de contraintes, résumant l'état de l'art de cette technique, est décrit dans les sections 3.2 et 3.3. La section 3.4 introduit au *problème de satisfaction de contraintes* et ses solutions algorithmiques de base. Dans la section 3.5, nous décrivons les problèmes qui nous sont posés et les éléments conceptuels sur lesquels nous dévéloppons les algorithmes de base de notre mécanisme. Les détails sont présentées dans les sections 3.6,3.7,3.8.

Le chapitre 4 décrit une tentative d'amélioration des mécanismes abor-

dés dans le chapitre 3, en étendant les algorithmes de filtrages binaires pour gérer les contraintes n-aires. Nous y décrivons les concepts et les algorithmes développés pour la résolution et pour la gestion des réseaux de contraintes algébriques.

Le chapitre 5 montre quelques applications avec les problèmes simples en ingénierie afin d'illustrer la flexibilité de notre approche par rapport aux mécanimes traditionnels en informatique.

Le chapitre 6 concerne les techniques de raisonnement qualitatif. Il commence par une introduction. Les sections 6.2 et 6.3 résument les deux méthodes les plus courantes dans la recherche en analyse qualitative, à savoir la méthode *"Physique Qualitative à base de Confluences"* de Johan de Kleer et John Seely Brown [deKleer 1984b] et la méthode *"Simulation Qualitative"* de Benjamin Kuipers [Kuipers 1985a].

Le chapitre 7 discute les problèmes rencontrés avec une description plus approfondie sur les méthodes de base. La dernière section exprime les points de vue de l'auteur vis-à-vis de l'application de cette technique dans le domaine de l'ingénierie.

Le chapitre 8 introduit la modélisation et la simulation des modèles physiques avec un exemple en ingénierie de l'incendie.

Le chapitre 9 présente une reflexion sur les recherches futures et quelques conclusions tirées de cette expérience.

1.4 Cadre du Travail

Commme ce travail n'est pas en mesure de resoudre tous les problèmes de conception, les résultats attendus seront insérés dans un contexte plus large. Nous proposons une architecture globale d'un système d'aide à la

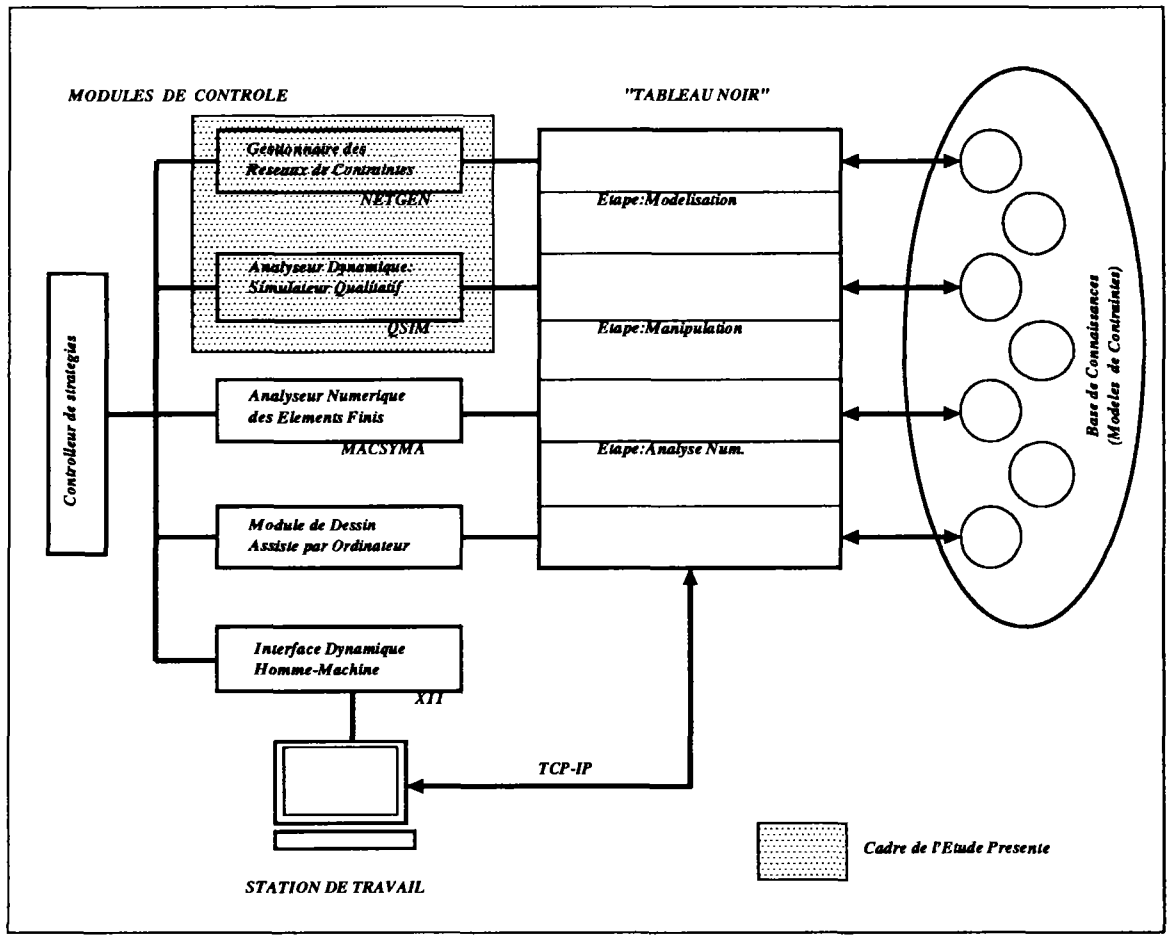


Figure 1: Une Architecture Conceptuelle.

conception en ingénierie. Cette architecture se situe actuellement au niveau conceptuelle, et ne fait pas l'objectif de ce travail.

L'architecture globale est composée par un ensemble de modules de traitement, opérant sur une structure hétérogène distribuée. Chaque module est comparable à un processeur. Une fois activé, il peut aider à accomplir une action spécifique dans la tâche de conception. Ces modules incluent une base de données, un analyseur des éléments finis, une boîte à outils mathématiques tel que MACSYMA, un analyseur de coûts, une module de DAO, etc...

Les modules de base de cette architecture sont les suivants:

- Un contrôleur de stratégies de conception permettant à surveiller, co-ordonner l'ensemble de l'activité de conception et responsable de toutes assistances vis-à-vis de l'utilisateur. Ce composant devrait se comporter comme un chef d'orchestre. Le mécanisme de base de ce composant est identique au mécanisme de contrôle d'un système de tableau noir (blackboard système) [HayesRoth 1985, HayesRoth 1983, Arlabosse 1989a].

- Un générateur d'expressions algébriques, permettant aux utilisateur d'exprimer les relations algébriques entre paramètres quantifiés des artéfacts de conception. Les détails de ce module sont exposés dans la partie I de cette thèse. Le principe de base de ce module est le traitement de contraintes.

- Un simulateur des modèles qualitatifs, permettant aux concepteurs d'étudier les variations des valeurs qualitatives des paramètres de conception en temps discrétisé. Les détails de ce module sont exposés dans la partie II de cette thèse.

- Une base de connaissances dans domaine physique, codée avec le générateur d'expression algébriques et les procedures ayant la structure des sources de connaissances du tableau noir (Blackboard Knowledge Sources) [HayesRoth 1985].

- Une interface de dialogue Homme-machine avec les fonctionnalités Icônes, Souris du type X-Windows ou Desk-Tops.

- Un ensemble des modules d'analyse numérique, ayant MACSYMA comme noyau de base. Ces modules peuvent être un

analyseur des éléments finis, une base de données des standards, règlementations, etc.

- Enfin, un module de Dessin Assisté par Ordinateur ayant des canaux de communications avec les langages de base comme Lisp, Fortran ou C.

Le tout devrait être integré dans un environnement Common-Lisp du type Allegro Common-Lisp de Franz Inc. ayant la possibilité d'exploiter les fonctions externes, écrites en C ou Fortran. Ces modules opèrent sur un système d'exploitation du type UNIX fonctionnant avec les stations de travail SUN, VAX ou MIPS connectées en réseau local par IP-TCP.

Chapitre 2
La Conception et les Modèles de Contraintes

"Beaucoup de bonnes idées sont
en réalité une combinaison de deux idées
qui forment une passerelle
entre deux domaines de la pensée ou
entre différents points de vue."

Marvin Minsky (p.233, La Société de l'Esprit, 1988)

2.1 Le Modèle de Conception par Contraintes

La plupart des connaissances manipulées en conception en ingénierie sont normalement exprimées par des relations entre quantités de valeurs. Par exemple, l'analyse des variations met en jeu les relations spatio-temporelles ; la conception des éléments géométriques met en jeu celles de dimensions, d'angles ; l'analyse des phénomènes physiques met en jeu les relations entre les masses, températures, énergies, *etc.* Pour représenter ces connaissances sur un ordinateur en vue d'automatiser la *"maintenance"* de ces relations, il faut donc pouvoir les manipuler et savoir réaliser des calculs ou des comparaisons.

L'un des problèmes essentiels est alors de maitriser une difficulté classique en Intelligence Artificielle et ici aigue : A partir d'une relation quantitative, plusieurs autres relations peuvent être déduites. Par exemple, si l'ingénieur-concepteur commence par la relation, disons $0 < a < b$, il peut en déduire d'autres comme $a < \sqrt{a.b}$; $e^a < e^b$; $\int_0^a e^{-x^2} dx < \int_0^b e^{-x^2} dx$. Il est donc difficile de coder toutes les formes de relations qui sont possibles

à partir des relations simples. Par ailleurs, il est bien connu que des chemins extrêmement complexes de déduction sont parfois nécessaires, à partir de certaines relations pour mener à des conclusions désirées, même si les relations et les conclusions ont des formes simples.

Nous considérons chaque relation comme une contrainte. Le processus de conception en ingénierie peut alors se décrire en termes de gestion, de résolution et de maintenance des contraintes.

Une contrainte peut se définir comme une connaissance décrivant une relation R_{ij} entre au moins deux paramètres i et j. La causalité représentée par chaque relation R_{ij} est analysée ou ajoutée à un ensemble de connaissances (ou contraintes) afin de réduire l'espace de conception qui est relativement ouvert à l'origine.

Le concept de causalité ou relations dynamiques ne décrit pas seulement la représentation de connaissance du domaine mais aussi la manipulation de ces connaissances. En suivant notre approche, la causalité peut permettre de :

- réduire l'éventail des choix de valeurs pour les paramètres ;
- exprimer une vue partielle sur un objet (de conception) indépendamment de l'ensemble des objets à concevoir ;
- verifier des propositions de solutions par une analyse de la cohérence entre elles ;
- déduire d'autres valeurs inconnues à partir de celles connues pour un ensemble de paramètres ;
- ajouter, supprimer ou affiner des connaissances dans un modèle en fonction des exigences fonctionnelles ou relationnelles ;

- explorer les influences d'une valeur d'un paramètre sur d'autres.

Initialement, les contraintes sont apparues comme une approche pour aborder certains problèmes posés par certaines classes d'applications en intelligence artificielle. Plusieurs chercheurs ont développé des mécanismes génériques à base de contraintes pour effectuer des calculs, et les ont appliqués à la conception et au diagnostic des circuits d'électroniques [dK 1980] [McAllester 1980b] ; dans la robotique [Lensky 1986] ; dans la génie logicielle [Borning 1986] ; dans la reconnaisance des formes [Hu 1989, Zucker 1988, Stevens 1983] ; dans l'analyse des structures chimiques [Brinkley 1986] ; dans la plannification des processus industriels [Fox 1982, Fox 1983], etc.

Dans le contrôle de l'élaboration d'une solutions, on peut envisager deux types d'utilisation de contraintes :

- *a priori*, pour guider la génération des solutions parmi l'ensemble des choix possibles, de manière à n'envisager que des valeurs satisfaisant l'ensemble de contraintes,

- *a posteriori*, pour tester si une proposition de solutions satisfait les conditions exprimées via le modèle,

La première démarche, du type *incrémental*, consiste à insérer progressivement les contraintes[2] dans un modèle encore très ouvert[3]. A chaque insertion de contrainte, un mécanisme de propagation est invoqué pour répercuter les effets de cette contrainte sur les valeurs des paramètres qui sont directement et indirectement liés. L'espace de solutions est ainsi réduit de petit à petit jusqu'à l'obtention de solutions finales.

[2] "constraint posting"
[3] "under-constrainted"

La deuxième démarche, de type *génération-vérification*, consiste à générer un ensemble de solutions a priori et ensuite tester la performances de ces solutions en simulant les variations des valeurs.

Nous proposons de construire un générateur de contraintes répondant aux deux types d'utilisation cités ci-dessus.

2.2 Modèles de Contraintes dans la Conception

La description initiale de notre problème de conception se traduit souvent par un modèle constitué d'un ensemble de solutions algébriques. Cette description est ensuite raffinée en prenant en compte successivement les descriptions partielles fournies par les différentes contraintes.

Les contraintes de conception ont comme caractéristiques de pouvoir être contradictoires. On peut, après avoir ajouté une contrainte, se trouver dans une situation où le nouvel ensemble de contraintes est insoluble : il n'existe plus d'objet qui vérifie les spécifications données par l'ensemble de contraintes imposées. Le problème est alors sur-contraint. Pour pouvoir poursuivre la recherche de solutions et envisager d'autres contraintes, il est nécessaire dans une telle situation, de revenir à un état soluble par la *relaxation de contraintes*. Ainsi, notre processus d'analyse et de manipulation de connaissances est défini comme une séquence d'opérations d'insertion et de relaxation de contraintes. figure 2.

Le principe d'utilisation de notre approche repose sur l'interaction entre l'ingénieur concepteur et le générateur pour la gestion de ses préférences. Cette méthode fait un usage différent des connaissances exprimées par les lois structurelles (i.e. équations) et celles décrivant les exigences

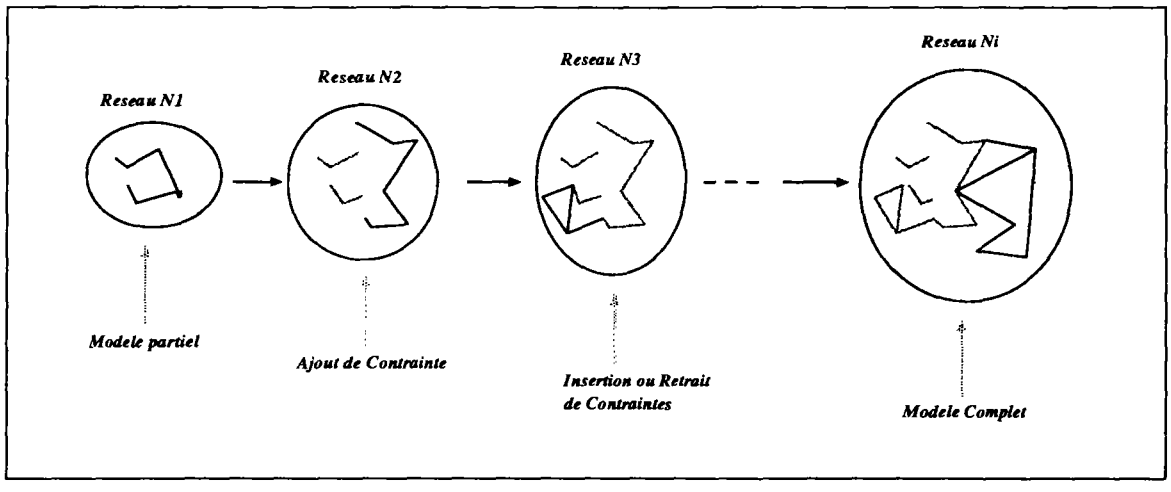


Figure 2: Analyse et Manipulation de Connaissances avec le Réseau de Contraintes.

fonctionelles (i.e. valeurs souhaitées). Elle reflète les deux étapes critiques dans le processus de conception:

- ANALYSE : prise en compte des *connaissances sur la structure de l'objet* de conception,
- MANIPULATION : prise en compte des *exigences fonctionnelles* de l'objet de conception,

Dans l'étape d'analyse, seules sont considérées les contraintes structurelles. Toutes ces contraintes devant être impérativement satisfaites, elle peuvent être manipulées directement par le générateur. Cette étape conduit à cerner la description des objets valides.

Dans l'étape de manipulation, les contraintes provenant des exigences fonctionnelles sont introduites pour rechercher les solutions pertinentes. Le mécanisme de gestion de contraintes doit alors répercuter sur la description de l'objet en cours d'élaboration les décisions prises par l'utilisateur. Ce travail est assuré par un mécanisme de propagation de contraintes. Cette

séquence d'insertion et retrait de contraintes se termine par décision d'utilisateur, lorsque les exigences fonctionnelles sont satisfaites.

Un système basé sur cette approche doit, pour assister l'utilisateur dans sa tâche, lui fournir des informations concernant l'état courant de la recherche. Parmi ces informations il est essentiel de signaler, après toute opération effectuée par l'utilisateur, si le nouvel ensemble de contraintes peut être satisfait. Le système doit également indiquer à l'utilisateur l'effet des opérations que celui-ci a décidé, c'est à dire une projection de l'objet en cours d'élaboration. La projection la plus précise serait l'ensemble des valeurs attribuées aux paramètres décrivant cet objet de conception.

Nous fixons à présent plus précisement la classe de modèles représentant les problèmes à considérer. Ces modèles nous amènerons à proposer une formalisation en termes de problèmes de résolution de contraintes.

2.3 Classe de Modèles

Les modèles que nous allons étudier se restreignent à la classe des modèles algébriques. Nous supposons que ces modèles ont une structure unique et connue *a priori* : les équations et inéquations algébriques. Les valeurs à attribuer aux inconnus de ces modèles sont soit des valeurs numériques (i.e. entiers, réels, intervalles) soit des valeurs symboliques (i.e. ordres de grandeurs, tendances, *etc*).

Le problème de satisfaction des contraintes associé à ces modèles consiste à déterminer les valeurs possibles de chacun des paramètres du modèle. A chacun de ces paramètres est associé un domaine fini de valeurs possibles. Un objet spécifique de la conception correspond donc à une instanciation d'un modèle ; il est décrit par un ensemble de paires (paramètres, valeurs)

qui satisfait l'ensemble de contraintes spécifiées ou générées pendant le processus de conception.

A noter que les modèles algébriques sont ceux le plus couramment utilisés dans l'ingénierie de bâtiments, à savoir les modèles de transfert thermiques en régime stationnaire, les modèles de calcul de structures statiques, *etc.*

2.4 Le Rôle du Gestionnaire de Contraintes

Rappelons que notre approche est fondée sur le principe de réduction de l'espace de solution par l'insertion incrémentale de contraintes dans un modèle décrivant un objet ou un phénomène physique. Cette méthode incrémentale peut, d'une manière dualle, être décrite comme une séquence de modèles $(\mathcal{N}_o, \mathcal{N}_1, \ldots \mathcal{N}_n)$, chacun des $\mathcal{N}_i$ devant satisfaire l'ensemble de contraintes qu'il contient ; $\mathcal{N}_o$ devant satisfaire les contraintes initiales de l'objet à concevoir. Cette séquence d'insertion et retrait nécessite un générateur sur lequel les interactions entre l'ingénieur-concepteur et le système peuvent être effectuées.

Le gestionnaire doit, à chaque étape, fournir à l'utilisateur des informations qui lui permetteront d'effectuer ses choix d'opérations : la solution au problème si elle existe ; la valeur attribuée à un paramètre ; comment elle a été attribuée, par quelles valeurs choisies dans l'ensemble de celles connues ; si une solution est inconsistante, l'explication de cette incohérence, *etc.*

Le développement de ce gestionnaire repose sur la technique de *Satisfaction de Contraintes* ou la *Propagation de Contraintes.* Nous allons décrire ses fondements théoriques dans les chapitres suivants.

Chapitre 3
l'Etat de l'Art et le Fondement de l'Approche

"Il existe d'innombrables types de réseaux contenant des boucles. Mais tous les réseaux dépourvus de boucles sont fondamentalement identiques : ils ont tous la forme de simples chaînes."

3.1 Les Réseaux de Contraintes

Définition 2 Réseaux de Contraintes

Un réseau de contraintes est une structure déclarative exprimant des relations entre paramètres. Il consiste en un ensemble de noeuds, un ensemble de valeurs et un ensemble de contraintes appliquées sur les valeurs de ces noeuds. Chaque noeud représente un paramètre individuel qui peut avoir une certaine valeur, connue ou inconnue. Une contrainte représente une relation entre les valeurs des noeuds entre lesquels elle établit une connexion.

Définition 3 Relations[*Dechter 1987b*]

Soit un ensemble X de n variables $\{X_1, \ldots, X_n\}$ et un ensemble D de n domaines de valeurs $\{D_1, \ldots, D_n\}$:

- *Une* **relation n-aire** ρ *sur ces variables est un sous-ensemble du produit cartésien de leurs domaines:*

$$\rho \subseteq D_1 \times \cdots \times D_n$$

- *Une* **contrainte binaire** R_{ij} *entre deux variables* X_i *et* X_j *est un sous-ensemble du produit Cartésien de* D_i *et* D_j*:*

$$(R_{ij} \subseteq D_i \times D_j)$$

Si $i = j$, R_{ii} *représente une* **relation unaire** *sur* X_i.

- *Un* **réseau de contraintes binaires** $\mathcal{N}$ *est constitué d'un ensemble* X *de variables* $\{X_1, \ldots, X_n\}$, *un ensemble* D *de* n *domaines de valeurs* $\{D_1, \ldots, D_n\}$, *et d'un ensemble de contraintes binaires et unaires* C *imposées sur les domaines de ces variables. Il représente une relation* n*-aires définie par l'ensemble de tous les* n*-uplets satisfaisant toutes contraintes dans* C. *La relation* ρ *représentée par un réseau de contraintes est :*

$$\rho = \{(x_1, x_2, \ldots, x_n) \mid x_i \in D_i \text{ et } (\forall i, j) : (x_i, x_j) \in R_{ij}$$

- *Une contrainte est* **symétrique** *si:*

$$\forall x_i \in D_i; \forall x_j \in D_j : [(x_i, x_j) \in R_{ij}] \Rightarrow [(x_j, x_i) \in R_{ji}]$$

- *Plusieurs opérations sur les contraintes binaires sont possible.* Union *de deux contraintes s'opérant sur les deux variables est une nouvelle contrainte qui admettra tous les paires de valeurs admises par l'une ou l'autre.* L'intersection *de deux contraintes n'admet que les paires de valeurs admises par les deux contraintes.*

  La composition *de deux contraintes,* R_{12} *et* R_{23}, *"induit" une contrainte* R_{13} *définie comme suit:*

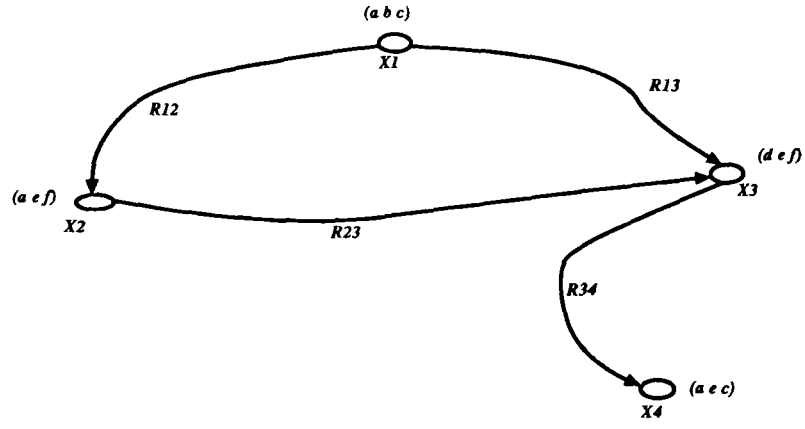


Figure 3: Exemple d'un Réseau de Contraintes.

Une paire (x_1, x_3) *est admise par* R_{13} *s'il existe au moins une valeur* $x_2 \in D_2$ *telle que* $(x_1, x_2) \in R_{12}$ *et* $(x_2, x_3) \in R_{23}$. *En analogie avec la notation matricielle, la contrainte induite* R_{13} *s'écrit :* $R_{13} = R_{12}.R_{23}$.

Chaque réseau de contraintes binaires peut se représenter par un **graphe de contraintes** où les variables sont représentées par des noeuds ; les contraintes par des arcs et chaque contrainte spécifie un ensemble de couples de valeurs admises. La figure 3 ilustre en exemple.

Définition 4 Instanciation de Contraintes

Soit un n-uplet de variables $< X_1, X_2, \ldots, X_n >$, *une* **instanciation** V *est une application de ces variables à un n-uplet de valeurs* $< v_1, v_2, \ldots, v_n >$. *On note:*

$$V(X_i) = v_i$$

Exemple: $V(< X_1, X_2 >, < 3, 4 >) \Leftrightarrow (V(X_1) = 3, V(X_2) = 4)$.

Définition 5 Contraintes Algébriques

Une **contrainte** n**-aire** C *sur* n *variables* $X_1, \cdots, X_n$ *est une relation* n-*aire* R *sur l'instanciation de ces variables. Les contraintes peuvent être structurées en type de contraintes relatif aux types de relations.*

$$C(X_1, \cdots, X_n) = R(V(X_1, \cdots, V(X_n))$$

Une **contrainte binaire** C_{ij} *sur deux variables* X_i, X_j *est une relation* R_{ij} *sur l'instanciation de ces variables.*

$$C(X_i, X_j) = R(V(X_i), V(X_j))$$

Les contraintes algébriques sont définies par les relations algébriques, si R_{ij} *est symétrique,* C_{ij} *est symétrique, par exemple:*

$$\begin{aligned} C^+(X_1, X_2) &= R(V(X_1), V(X_2)) = (v_1 + v_2) = (v_2 + v_1) \\ C^*(X_1, X_2) &= R(V(X_1), V(X_2)) = (v_1 * v_2) = (v_2 * v_1) \end{aligned}$$

3.2 La Propagation de Contraintes

La déduction "en avant" sur les réseaux de contraintes correspond à la notation de **propagation de contraintes**. Dans cette technique, les informations sont déduites d'un groupe local de contraintes et de noeuds, elles sont ensuite enregistrées dans le réseau. Ces enregistrements serviront ensuite pour déduire d'autres changements. Donc les conséquences de chaque processus seront propagées à travers le réseau. Plusieurs chercheurs ont travaillé sur ce mécanisme :

3.2.1 Les calculs des flux de données

Les calculs des flux de données par les processeurs cablés directionnels (parallèles), proposés par Dennis [Dennis 1973] et Arvind [Arvind 1978], constituent un modèle dans lequel le calcul des valeurs est organisé sous forme d'un réseau de processeurs, traitant en parallèle les valeurs qui circulent d'une manière asynchrone en fonction des connexions entre processeurs. Dans ce formalisme, les directions de connexions sont pré-déterminées en fonction du flux d'informations. Le flux d'informations a pour objectif d'exprimer l'ordre des calculs (cascade) que gère normalement les langages de programmation, sans soucis des conditions de séquences qui sont imposées par les structures de contrôle.

Le réseau de flux de données exprime les séquences nécessaires pour le calcul, un processeur n'est activé qu'en présence des informations disponibles. Toutes fois, ce système a tous les inconvénients d'un système de processeurs interconnectés physiquement par cablage. Par exemple, les dépendances des données entre une étape de calcul et une autre ne peuvent pas être enregistrées.

3.2.2 L'Algorithme de Waltz

L'algorithme de Waltz [Waltz 1972], conçu pour le problème de filtrage des étiquettes dans la vision artificielle, est considéré comme une forme spéciale de la propagation de contraintes, quelques fois appelé la *propagation de contraintes booléennes*. Le *filtre de Waltz* est une technique pour simplifier le problème d'explosion combinatoire dans les arbres de concepts. L'algorithme de Waltz propage les informations à travers des segments déterminés entre des jonctions de lignes (en informatique graphique). A chaque jonc-

tion de lignes, un calcul est effectué en fonction des informations circulant sur ces lignes. Waltz a trouvé que cette technique, en propageant les valeurs d'une manière locale ne pouvait pas resoudre les ambiguités globales. Dans la pratique, il ne peut converger que vers une solution unique ou un ensemble de solutions alternatives qui ne pourraient être discriminées que par l'analyse globale du problème. Ce modèle a l'avantage de pouvoir exprimer simultanément tous les états validés d'un système physique. Par contre, il a l'inconvenient de ne pas pouvoir maintenir les dépendances ou cohérences.[4]

3.2.3 La Propagation d'étiquettes symboliques

La propagation d'étiquettes symboliques est un champ de recherche provenant des travaux de Quillian sur les réseaux sémantiques [Quillian 1968]. Ce principe consiste à propager les étiquettes au lieu des quantités dans un réseau. En effet, les réseaux sémantiques de Quillian constituent un ensemble de noeuds représentant les concepts primitifs avec les pointeurs appartenant à ces concepts. La signification d'un noeud consiste en la somme totale de ses relations avec les autres noeuds. Donc, on peut comparer deux concepts en propageant deux étiquettes symboliques de deux concepts et ensuite analyser les points où elles se croisent. Grossman [Grossman 1976] utilise les expressions de contraintes pour représenter les relations complexes des bases de données, incluant les relations du type union, intersection, partition d'ensembles. Le problème est qu'un nombre croissant d'informations est représenté dans la structure des étiquettes au lieu des réseaux, et la structure d'une étiquette devient alors difficile à manipuler tant du point de vue de l'analyse que du point de vue de la propagation.

[4]Notons que cet algorithme a été implémenté dans l'algorithme QSIM de Kuipers que nous aborderons dans la partie 2 de ce mémoire.

3.2.4 L'algorithme de Propagation de Steele

L'algorithme de propagation de Steele [SteeleJr 1980] construit les calculs sur un réseau de relations par les déductions 'pas-à-pas' locales. L'histoire de chaque calcul est retenue sous la forme des dépendances entre valeurs. Cette information peut être utilisée pour expliquer le processus de calcul, en sa totalité ou bien par étapes. En outre, les mêmes informations seront employées pour guider la gestion des cohérences, et/ou pour gérer les éventuelles contradictions introduites par le changement des paramètres dans le réseau en utilisant la technique de *"dependency directed backtracking"*. En addition, un mécanisme de gestion des hypothèses est inclu pour gerer les valeurs par défaut. Enfin, un mécanisme de résolution doté d'un enregistreur des prémisses *nogoods* permet de limiter le problème d'explosion combinatoire.

3.2.5 La Méthode de Freuder

La Méthode de Freuder [Freuder 1978, Freuder 1982, Freuder 1985a] propage en les synthétisant les contraintes d'ordre supérieur : Freuder décrit une méthode de propagation de contraintes par la synthèse des nouvelles contraintes. En résumé, supposons que le nombre d'éléments dans l'ensemble de noeuds d'un réseau est n, chaque contrainte sur k noeuds (avec $k \leq n$) est représentée comme un ensemble explicite de k-uplets représentant les combinaisons valides de valeurs de ces noeuds. Alors tous les sous-ensembles de combinaisons sont considérés, dans l'ordre de leurs occurences. Dans cette ordre, les contraintes d'ordre k seront synthétisées à partir de celles d'ordre $(k-1)$. Plus précisement, la contrainte d'ordre k sur un ensemble de noeuds J est synthétisée par la combinaison des k

contraintes d'ordre $(k-1)$ sur les sous-ensembles de J. Ceci constitue une sorte de processus de calcul employant la technique de programmation dynamique. Dès que l'algorithme s'arrête, la contrainte individuelle d'ordre n est un ensemble de solutions pour l'ensemble du réseau. On peut considérer la représentation de contraintes de Freuder comme une collection des 'ensembles corrects'[5], repésentant la combinaison des valeurs admissibles. Nous trouvons que cette technique est complémentaire à celle proposée par Steele [SteeleJr 1980]. Dans cette dernière, Steele suppose que toutes les combinaisons sont possibles, et il utilise les ensembles 'incorrects' (nogood sets) pout rejetter les combinaisons invalides.

Dans la section 3.6, nous décrivons le détail de l'algorithme K-consistance que nous servira de base dans la suite.

3.2.6 La Technique de propagation locale de Sussman

Sussman [Sussman 1980, dK 1980] a conçu cette technique pour le domaine de l'électricité. C'est une application naturelle puisque les réseaux de contraintes correspondent directement aux schémas de circuits électroniques. Cette technique a été démontrée comme une solution adéquate pour le diagnostic des circuits électroniques. En effet, dans ce type d'applications, la technique de propagation locale peut déterminer (automatiquement) les variables susceptiples d' être éliminées dans un ensemble d'équations[6]. Les dépendances sont enregistrées dans cet algorithme, et sont utilisées pour fournir des explications et pour maintenir la cohérence des valeurs déduites. Les termes *"Dependency Directed Backtracking"* [Stallman 1977, Stallman 1979] et les notions de *"in"*, *"out"*, *"nogoods"* font parties de cet

[5]les termes 'good sets' et 'no-good sets' sont largement utilisé dans la technique de maintenance des vérités

[6]Alors la matrice de coefficients est peu-dense

algorithme.

3.2.7 Systèmes de Maintenance de Vérité

Basé sur la technique de propagation locale de Sussman et la technique d'enregistrement des dépendances, une série d'études et recherches a eu lieu au M.I.T.; Le sytème AMORD [Doyleetals 1977] [deKleeretals 1978] fournit un mécanisme répertoriant des base de données pour enregistrer les faits représentés d'une manière symbolique. Son gestionnaire de dépendances, appellé *Système de Maintenance de Vérité* (Truth Maintenance System ou TMS), est utilisé pour enregistrer les relations logiques entre les faits. Doyle [Doyle 1977] [Doyle 1979] propose une méthode pour separer le TMS des mécanismes des bases de données. Goodwin [Goodwin 1982] améliore cet algorithme en proposant un algorithme de mise-à-jour et ajoute des contraintes non-monotones. Mc Allester [McAllester 1980a] améliore ce système en ajoutant un mécanisme de contrôle sur les prémisses et utilise la logique propositionnelle pour donner de la souplesse dans l'utilisation de ce système, son système est appellé LTMS (Logic-based TMS). Le système le plus récent, le fameux ATMS (Assumption-based TMS) de de-Kleer [deKleer 1986a], est concu pour les contextes multiples. Ces travaux constituent aujourd'hui un champs de recherche important en Intelligence Artificielle.

3.3 Les Catégories de Propagation de Contraintes

Davis [Davis 1987] distingue six catégories de propagation de contraintes, dépendant du type d'information à traiter :

1. **Déduction de contraintes,** où de nouvelles contraintes sont déduites et sont ajoutées au reseaux de contraintes. Les systèmes ENV de Kuipers[Kuipers 1985b] et Quantity-Lattice de Simmons en sont des exemples.

2. **Déduction d'étiquettes,** chaque noeud est étiquetté par un ensemble de valeurs possible. Dans le processus de déduction, les contraintes sont utilisées pour diminuer cet ensemble.

3. **Déduction de valeurs,** les noeuds sont attribués par des constantes, les contraintes sont employées pour déduire les valeurs des noeuds jusqu'alors non-attribués à partir de ce ceux déjà attribués.

4. **Déduction d'expressions** est une généralisation de la déduction de valeurs. Dans cette technique, les noeuds sont étiquettés par une valeur exprimée par des termes contenant les valeurs d'autres noeuds. Si un noeud est étiqueté par deux differents étiquettes, ils sont rendus égaux, et l'équation est résolue. La déduction d'expression est utilisée dans CONSTRAINT[Sussman 1980].

5. **Relaxation,** à tous les noeuds on attribut des valeurs exactes pouvant etre incohérentes avec les contraintes imposées sur ces noeuds. Le processus "d'assimilation" consiste à "pousser" ces valeurs dans le réseau de manière à ce que toutes les contraintes soient satisfaites. Cette technique a été employée pour résoudre les problèmes en dehors du domaine de l'I.A. tels que les équations différentielles partielles.

6. **Relaxation d'étiquettes**, Les noeuds sont validés par un processus d'attribution de valeurs probables. La mise à jour consiste à changer les probabilités en combinant la probabilité sur un noeud à partir des probabilités d'autres noeuds.

Notons au passage que l'implémentation de notre générateur décrit dans le chapitre 4 est largement influencée par l'algorithme de Steele [SteeleJr 1980], étendant les algorithmes traitant la consistance locale de Mackworth [Mackworth 1977, Mackworth 1985], Freuder [Freuder 1978, Freuder 1982] [Freuder 1985b, Freuder 1989], Montanary [Montanari 1974], Dechter [Dechter 1987a], Dechter et Pearl [Dechter 1987b, Dechter 1989], et Mohr [Mohr 1986, Mohr 1988]. Le type de valeurs a propager est identique au concept des intervalles de valeurs introduit par Davis [Davis 1987].

3.4 L'Algorithme de Résolution des PSC

Le Problème de Satisfaction des Contraintes (PSC) [Dechter 1985] consiste à attribuer des valeurs aux variables selon un ensemble de contraintes. La spécification des contraintes est un moyen d'exprimer les connaissances déclaratives par les relations locales entre entités du domaine. La résolution du problème PSC consiste à générer des interprétations explicites à partir des connaissances données sous une forme déclarative implicite.

Définition 6 *[Dechter 1987b]*

Supposons que tous les variables $X_1, \cdots, X_n$ *ont des domaines de valeurs finis et discrets* $D_1, \cdots, D_n \in D$. *La* **satisfiabilité des contraintes** *est exprimée par la logique de prédicats de premier ordre:*

$$(\exists X_1), (\exists X_2), \cdots, (\exists X_n) | (X_1 \in D_1), (X_2 \in D_2), \cdots, (X_n \in D_n):$$

$$R_1(X_1) \wedge R_2(X_2) \wedge \ldots \wedge R_n(X_n) \wedge R_{12}(X_1, X_2) \wedge \ldots \wedge R_{n-1,n}(X_{n-1}, X_n) \models N \quad (3.4.3)$$

Algorithme de Résolution du PSC

L'algorithme de résolution du PSC est l'algorithme *'backtrack'*[Mackworth 1977]. Dans la version de base, *'backtrack'* traverse les variables dans l'ordre prédéterminé par l'expression 3.4.3, *instancier* une sous-séquence de variables $(X_1, \ldots, X_i)$ par des valeurs provisoires et puis tendre à insérer une nouvelle instanciation de X_{i+1} dans la sous-séquence ci-dessus de manière à ce que l'ensemble de la nouvelle sous-séquence $(X_1, \ldots, X_i, X_{i+1})$ soit cohérent. Si aucune instanciation n'est cohérente[7] pour $(X_1, \ldots, X_i, X_{i+1})$, l'algorithme retourne en arrière vers la variable la plus récente, change sa validation et continue à partir de cette variable. L'algorithme ci-dessous est résumé dans [Dechter 1987b], il contient deux procédures recursives *"Calcul-Avant"* et *"Retour-Arrière"*. La première propage une validation partielle courante, la deuxième est utilisée dans le cas de bloquage dans la propagation. Ces deux procédures maintiennent les listes de candidats C_i associées à chaque variable X_i:

Calcul-Avant $X_1, \ldots, X_i$

Debut

1. Si $i = n$, Fin.
2. $L_{i+1} \leftarrow$ **Calcul-Candidats**$(X_1, \ldots, X_i, X_{i+1})$;
3. Si $L_{i+1} \neq 0$
4. Alors $X_{i+1} \leftarrow$ car[8] L_{i+1};
5. $L_{i+1} \leftarrow L_{i+1} - X_{i+1}$;

[7]Une *instanciation* d'un sous-ensemble de variables est cohérente si elle satisfait tous les contraintes imposées sur ce sous-ensemble.

6. **Calcul-Avant**$(X_1, \ldots, X_i, X_{i+1})$.

7. Sinon **Retour-Arriere**$(X_1, \ldots, X_i)$.

Fin

Retour-Arriere$(X_1, \ldots, X_i)$

Debut

1. Si $i = 0$, Fin

2. Si $L_i \neq 0$,

3. Alors $X_i \leftarrow \text{car}(L_i)$;

4. $L_i \leftarrow L_i - X_i$;

5. **Calcul-Avant**$(X_1, \ldots, X_i)$.

6. Sinon **Retour-Arriere**$(X_1, \ldots, X_{i-1})$.

Fin

La procédure *"Retour-Arrière"* est initialisée par la procédure *"Calcul-Avant* avec $i = 0$; la procédure *"Calcul-Candidats*$(X_1, \ldots, X_i, X_{i+1})$*"* sélectionne dans $D(X_{i+1})$ toutes les valeurs cohérentes avec les validations précédentes.

Cet algorithme présente plusieurs inconvenients pratiques. En particulier, le temps nécessaire pour trouver une solution tend à être exponentiel par rapport au nombre de variables. D'après Mackworth [Mackworth 1977] et Freuder [Freuder 1985a] les difficultés sont les suivantes :

1. La première difficulté provient de l'inefficacité des relations unaires. Si le domaine de valeurs D_i pour la variable x_i contient une valeur qui ne satisfait pas $R_i(x)$, alors cette valeur est la cause des intanciations et défaillances repétitives. Cette boucle

(d'instanciations et retour-arrière) peut être éliminée en écartant simplement, une fois pour toute, les éléments de D_i qui ne satisfont pas la relation un-aire correspondante.

2. La deuxième difficulté apparaît dans la situation suivante : Supposons que les variables sont instanciées dans un ordre total $X_1, X_2, \ldots, X_n$ et que $X_i = a, R_{ij}(a, X_j)$ (où $j > i$) ne se vérifie avec aucune valeur de X_{j-1}. La procédure *"retour-arrière"* examinera toutes les valeurs de X_j, échouera et continuera à examiner les valeurs de X_{j-1}. Pour chaque valeur de X_{j-1}, elle devra ré-examiner toutes les valeurs de X_j. Ce processus continue à vérifier tous les combinaisons de valeurs pour $X_{i+1}, X_{i+2}, \ldots, X_j$ avant de découvrir finalement que a n'est pas une valeur possible pour X_i. Cette difficulté peut devenir très grave si ce processus est repété pour tous les autres ensembles de valeurs pour $X_1, X_2, \ldots, X_{i-1}$ avec $X_i = a$.

3. La troisième difficulté apparaît dans le cas où $X_i = a, X_j = b$, et $R_i(a), R_j(b)$ et $R_{ij}(a, b)$ sont satisfaites mais il n'existe aucune valeur v pour une troisième variable X_k telque $R_{ik}(a, v), R_k(v)$, et $R_{kj}(v, b)$ sont simultanément satisfaites comme elles devraient en être dans toutes les solutions. Comme dans le cas précédent, ce processus n'est pas seulement coûteux pour l'instanciation en avant, il l'est aussi bien pour le *retoure-arrière*.

3.5 Position du Problème à Traiter

Considérons une réseau de contraintes binaires contenant un ensemble de variables X. Chacune de variables $X_i \in X$ est associée à une attribution éventuellement infinie de valeurs $D_i \in D$ et un ensemble de contraintes C. Dans le cas des attributions infinies, les questions sont les suivantes:

1. Existe-t-il des séquences finies de l'instanciation de contraintes amènant une solution ?
2. Cette solution est-elle unique ?
3. Comment trouver cette solution ?

Le développement d'un gestionnaire de contraintes est fondé sur la base des formalismes du Problème de Satifaction de Contraintes (PSC) précédement décrit. Nous devons résoudre les problèmes suivants:

1. Pour un triplet $(\{X\}, \{D\}, \{C\})$ donné, existe-t-il des solutions pour tous les $V(X_i)$ pour toute variable $X_i \in \{X\}$?
2. Quelles sont les valeurs $D_j \in \{D\}$ qui n'apparaissent dans aucune solutions de PSC sur $\{C\}$ pour $V(X_i)|\forall X_i \in \{X\}$?
3. Comment gérer les contradictions si elles existent dans le processus d'instanciation $V(X_i) = D_j$?
4. Quelles sont les moyens que le générateur peut fournir aux utilisateur dans la tâche de conception ?

Pour approfondir le questions qui nous sont posées, il convient à présent d'examiner les bases théoriques liées à notre proposition d'extensions. Nous décrivons par la suite ces notions de base. Il y en a trois :

- La notion de filtrage associé aux réseaux de contraintes;
- La notion de k-consistance;
- La notion de processeurs actifs;

Il importe de noter qu'à part de la notion de processeur actif, les autres travaux concernent principalement les réseaux de contraintes *binaires*. Le modèle que nous proposons concerne principalement les contraintes *n-aires*. Les extensions seront décrites dans le chapitre suivant.

3.6 Le Filtrage Associé aux Réseau de Contraintes

A chaque noeud dans un réseau de contraintes est attribuée un ensemble de valeurs. Ces valeurs peuvent être de nature numérique ou symbolique. Le terme "étiquette" est utilisé pour décrire la valeur attribuée à un noeud. Dans la gestion de contraintes algébriques, chaque étiquette est généralement un intervalle de valeurs. En fonction des contraintes imposées sur ces noeuds, les intervalles de valeurs sont déduits et/ou raffinés. Par exemple, si le modèle est défini par $(x + y = z)$ et l' ensemble d'étiquettes est $(x \geq 3, y = 1)$. On peut déduire $(z \geq 4)$ et ajouter ce résultat à l'ensemble d'étiquettes satisfaisant les contraintes imposées par le modèle. L'action de déduction des étiquettes est appelée *filtrage*,

Définition 7 Filtrage

Si une solution existe pour un réseau $\mathcal{N}$, le résultat du filtrage associé à $\mathcal{N}$ est un ensemble de paires $(X_i, D_j) : X_i \in \{X\}$ et $D_j \in \{D\}$ qui est la solution.

Soit C, une contrainte s'exerçant sur les noeuds $X_1, \ldots, X_k$. Soit S_i l'ensemble d'étiquettes de X_i. le filtrage $\mathcal{F}(C, X_j)$ *est l'*ensemble de valeurs consistantes *pour X_j vis-à-vis de la contrainte C et de tous les ensembles d'étiquettes S_i :*

$$\mathcal{F}(C, X_j) = \{a_j \in S_j \mid \exists (a_i \in S_i, i \neq j) : C(a_1, \ldots, a_j, \ldots, a_k)\}$$

Une valeur a_j appartient à $\mathcal{F}(C, X_j)$ si a_j appartient à S_j et est une élément de k-uplet $a_1, \ldots, a_k$ satisfaisant la contrainte C et tous les S_i.

Notons que le filtrage est déductif ; si un n-uplet satisfait la contrainte et les étiquettes initiales, alors il satisfait l'étiquette filtrée.

L'application de la fonction de filtrage sur l'algorithme de propagation de contraintes (calcul-avant et retour-arrière) résulte en un algorithme identique à celui de Waltz [Waltz 1972]. Cet algorithme (voir figure 4) consiste à filtrer, pour une contrainte donnée, les noeuds l'un après l'autre, jusqu'à ce que le filtrage ne produise plus aucun changement, alors le réseau de contraintes est dit *consistant.*

Exemple: Supposons un ensemble d'étiquettes ($x \in [1, 10], y \in [3, 8], z \in [2, 7]$) s'opère sur deux contraintes:

$$C_1 \ : \ x + y = z$$

$$C_2 \ : \ y \leq x$$

La propagation par WALTZ sera comme suit:

1. C_1 s'applique:

   - $(x \geq 1) \wedge (y \geq 3)$ C_1:$(x + y = z) \Rightarrow (z \geq 4)$: filtrage $(z \in [4, 7])$,

procédure FILTRAGE $(C, x_1 \dots x_k)$

début noeuds-revisés $\longleftarrow \vee$

pour $i = 1, k$:

$S \longleftarrow \mathcal{F}(C, x_i)$

si $S = \vee$

alors stop

sinon si $S \neq S_i$

noeuds-revisés $\longleftarrow$ noeuds-revisés $+ x_i$

fin

retour noeuds-revisés

fin

procédure WALTZ $((C_1 \dots C_k)\ (x_1 \dots x_j))$

début $\mathcal{C} \longleftarrow (C_i, \dots, C_k)$

pour $i = 1, k$:

$\mathcal{C} \longleftarrow \mathcal{C} - C_i,$

contraintes-revisées $\longleftarrow$ FILTRAGE$(C_i, (x_1 \dots x_j))$

$\forall x_i \in$ contraintes-revisées :

$\forall C_i' \neq C_i \mid x_i$ est connectée :

$\mathcal{C} \longleftarrow \mathcal{C} + C_i'.$

fin

fin

Figure 4: Algorithme de Filtrage

- $(z \leq 7 \wedge y \geq 3) C_1 : (x+y = z) \Rightarrow (x \leq 4)$: filtrage $(x \in [1,4])$,
- x et z ont été validées, invoquer C_2.

2. C_2 s'applique:

- $(x \leq 4), C_2 : (y \leq x) \Rightarrow (y \leq 4)$: filtrage $(y \in [3,4])$
- $(y \geq 3), C_2 : (y \leq x) \Rightarrow (x \geq 3)$: filtrage $(x \in [3,4])$,
- x et y ont été validées, invoquer C_1.

3. C_1 s'applique:

- $((x \geq 3) \wedge (y \geq 3)), C_1 : (x + y = z) \Rightarrow (z \geq 6)$: filtrage $(z \in [6,7])$,
- Seulement z est changée, les contraintes sont satisfaites. Fin.

Résultat:

$$(x \in [1,10], y \in [3,8], z \in [2,7]) \wedge C_1 \wedge C_2 \Rightarrow (x \in [3,4], y \in [3,4], z \in [6,7])$$

.

3.7 La Notion de K-Consistance

Le filtrage peut amèner à des séries d'attribution infinies de valeurs des variables. Par exemple, si l'on considère deux contraintes $C_1 : x = y$ et $c_2 : x = 2y$; les domaines de valeurs initiaux sont $D_x : x \in [0,100]$, et $D_y : y \in [0,100]$. On sait que la solution consistante est $(x \in [0,0])$ et $(y \in [0,0])$ mais l'application du filtrage mène à une suite infinie d'opérations partant de $x \in [0,100]$; $y \in [0,50]$ puis $x \in [0,50]$; $y \in [0,25]$ et ainsi jusqu'à infini avec $x \in [0,\epsilon]$; $y \in [0,\epsilon]$.

Plusieurs propositions visant à traiter les instanciations et attributions infinies dans la résolution des contraintes sont due à Mackworth [Mackworth 1977, Mackworth 1985] ; Freuder [Freuder 1978, Freuder 1982] [Freuder 1985b, Freuder 1985a, Freuder 1989] ; Dechter [Dechter 1985, Dechter 1986] et Pearl [Dechter 1987b, Dechter 1989] ; Mohr [Mohr 1986, Mohr 1988] ; Bibel [Bibel 1988] : Lowe [Lowe 1987] ; Berliner [Berliner 1985] ; Haralick [Haralick 1978, Haralick 1980]. Ils proposent principalement le calcul d'une solution localement cohérente afin de minimiser les appels de *"retour-arrière"*. Une notion qui généralise celle de la consistance locale est appelé la **K-consistance**.

La solution d'un réseau, quand elle existe, consiste à trouver les valeurs des variables représentées par les noeuds qui vérifient toutes les contraintes représentées par les arcs. Dans ces conditions, une mesure de la progression de la résolution est le nombre de noeuds qui, à une étape donnée de la résolution, vérifie l'ensemble des contraintes. Si l'on appelle K, le nombre de noeuds, on dira alors le réseau est K-consistant.

Algorithme de K-Consistance de Freuder

Freuder a introduit la notion qu'il appelle *K-Consistance*. Il définit de manière recursive: la K-consistance est définie à partir de la $K-1$-consistance, et ainsi jusqu'à 1-consistance, ce que Mackworth désigne part *noeud-consistance*. L'algorithme de noeud-consistance est le suivant:

Noeud-Consistance(i),

Debut

1. $D_i \leftarrow D_i \cap \{X \mid R_i(X)\}$;

2. Pour chaque $i = 1$ n faire:

Noeud-Consistance(i).

Fin

Définition 8 : k-consistance *[Freuder 1985b]*

La k-consistance d'un réseau de contraintes $\mathcal{N} = (X, C, D)$ *correspond à l'instanciation de k variables appartenant à l'ensemble de n variables* $X = \{X_1, \ldots, X_n\}$ *, en vue de satisfaire l'ensemble de m contraintes* $C = \{C_1, \ldots, C_m\}$ *par l'attribution des sous-ensembles de valeurs dans* $\mathcal{P}$*, produits cartésiens de* $D_i \in D$.

$\mathcal{N}$ *est k-consistant si et seulement si pour ensemble de* $k-1$ *variables* $X_{k-1} = \{X_1, \ldots, X_{k-1}\}$*, une instanciation* $V(X_{k-1})$ *de l'ensemble* X_{k-1} *satisfaisant l'ensemble de contraintes* $\{C_1, \ldots, C_m\} \subseteq C$ *portant sur ces variables, on peut prolonger* $V(X_{k-1})$ *à une instanciation* $V(X_k)$ *de l'ensemble* $\{X_1, \ldots, X_{k-1}, X_k\}$ $(X_k \notin X_{k-1})$ *de telle manière que* $V(X_k)$ *satifasse des contraintes* $\{C_1, \ldots, C_m\} \subseteq C$ *sur l'ensemble* $X_{k-1} \cap \{X_k\}$.

En d'autres termes, un réseau de contraintes $\mathcal{N}$ est k-consistant si toute instanciation consistante de $k-1$ variables peut être prolongée à une k^{eme} variable. Lorsqu'un réseau est k'-consistant pour tout $k' \leq k$, il est *fortement k-consistant.* Cette définition généralise la *consistance d'arc* [Mackworth 1977] qui correspond à la 2-consistance, et la *consistance de chemin* [Montanari 1974] qui correspond à la 3-consistance.

Proposition 1 Filtrage avec K-consistance

Le théorème centrale qui établirera les conditions pour une séquence de

filtrage[9] *est celui de K-consistance de Freuder. La solution pour* $V(x_i) \mid \forall x_i \in \{X\}$ *dans un réseau de contraintes binaires* $\mathcal{N} = (\{X\}, \{D\}, \{C\})$ *existe si* $\mathcal{N}$ *est fortement K-consistant.*

3.8 La Notion du Processeur Actif

Comme on l'a vu, la structure d'une situation physique peut se décrire par un ensemble de symboles $\{O\}$, représentant les paramètres physiques du système, et un ensemble d'équations de contraintes $\{\mathcal{F}\}$, décrivant les relations entre ces paramètres, (voir définition 1.2.1).

Nous considérons chaque paramètre comme une function $f: [a, b] \Rightarrow \Re | \Re \in]-\infty, +\infty[$. Les contraintes sont exprimées en termes de processeurs au lieu d'être exprimée en termes de fonctions. La gestion des contradictions, dans ce cas, peut être localisée dans chaque processeur. Pendant le processus d'instanciation, si une étiquette attribuée à un noeud contredit l'étiquette déjà attribuée (i.e. $D_{i,t} \not\subseteq D_{i,t-1}$), le processeur auquel ce noeud est connecté sera activé pour proposer soit une revision de l'ensemble de $\{D\}$ qui ne satisfait pas les contraintes, soit un retrait de contraintes dans le réseau.

Un *processeur de contrainte* est, en fait, un opérateur de calcul. Un avantage des processeurs de contraintes est qu'une expression d'une relation simple peut être utilisée d'une manière reversible. Il n'est pas nécessaire de spécifier quelles sont les valeurs à l'entrée ou à sortie d'un processeur. Les valeurs seront calculées automatiquement par un processeur dès que leurs supports d'informations sont disponibles. Par cette approche, le problème de l'explosion combinatoire dans l'algorithme de bactracking sera limité par

[9] Un filtrage est une propagation locale

les opérations (de contrôle) sur les processeurs.

Nous résumons les points clées de notre approche de gestion de contradictions de la manière suivante:

- Chaque processeur représente un opérateur de calcul simple. Il n'est connecté qu'à un nombre limité de noeuds ;
- Les processeurs peuvent être classés en fonction de leur type. Un processeur particulier est une instanciation de sa classe, et peut hériter de toutes les propriétés de sa classe.
- Le mecanisme de chaque processeur est analogue à celui des processeurs d'informations en électronique (notion des valeurs E/S).
- La gestion de contradictions sera effectuée localement aux processeurs et aux noeuds qu'ils connectent.

Ces éléments nous ont permis d'approfondir les problèmes posés et de proposer un développement. Dans le chapitre suivant, nous décrivons notre proposition d'extension et les algorithmes correspondants que nous avons développé.

Chapitre 4
Le Développement d'un Gestionnaire de Contraintes

"Cette théorie ne vaut rien.
Elle n'est même pas fausse !"
Wolfgang Pauli

4.1 Extension avec les Contraintes n-aires

Rappelons que notre approche de résolution est fondée sur la technique de filtrage associée aux réseaux de contraintes. Cependant, le plupart des travaux dans ce domain concerne principalement les contraintes binaires [Freuder 1985b], [Nachtcheim 1989], [Mulder 1988], [Berliner 1985], [Kasif 1986], [Morgenstern 1984]. En effet, les contraintes n-aires peuvent être exprimées par une association de contraintes binaires. Néanmoins, une telle association s'avère être peu pratique dans la conception puisque l'expression des contraintes binaires s'éloigne des expressions habituellement utilisées dans la conception.

La technique de représentation des contraintes n-aires (par les relations binaires) la plus connue est celle des graphes de contraintes. Dechter et Pearl [Dechter 1989] distinguent deux modes de représentation par graphes:

- *le Graphe Dual* (Dual Graph) est un graphe représentatif des hypergraphes de contraintes dont chaque noeud est étiquetté par un ensemble de variables dont les valeurs satisfont les contraintes binaires dans cet ensemble. Une contrainte (arc) est

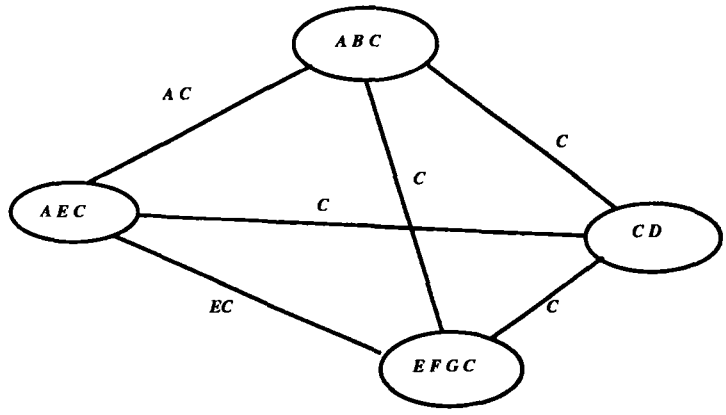


Figure 5: Un exemple du Graphe Dual.

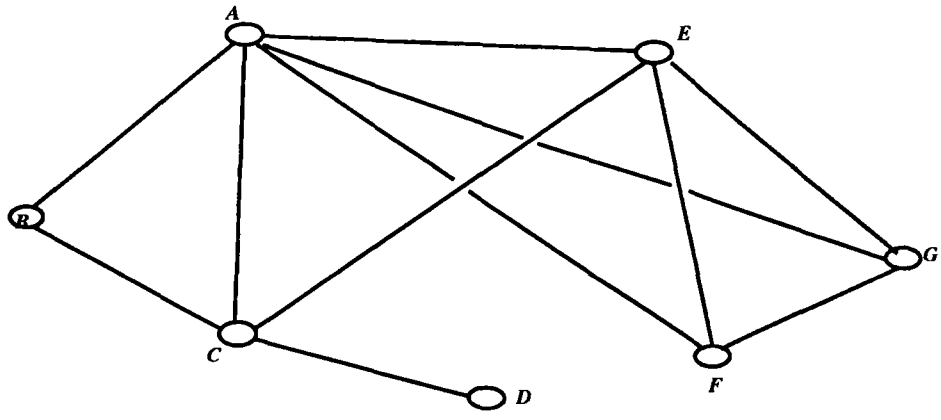


Figure 6: Un exemple du Graphe Elementaire.

introduite entre deux noeuds si l'intersection entre les deux étiquettes est un ensemble non-vide. Par exemple, la représentation par le graphe dual des relations entre abc, aec, cd et $egfc$ est donnée par la figure 5.

- *le Graphe Elémentaire* (Primal Graph) est une graphe dont chaque noeud représente une variable et chaque arc représente une relation binaire entre deux noeuds. Pour le même exemple cité ci-dessus, la représentation par graphe élémentaire est illustré par la figure 6.

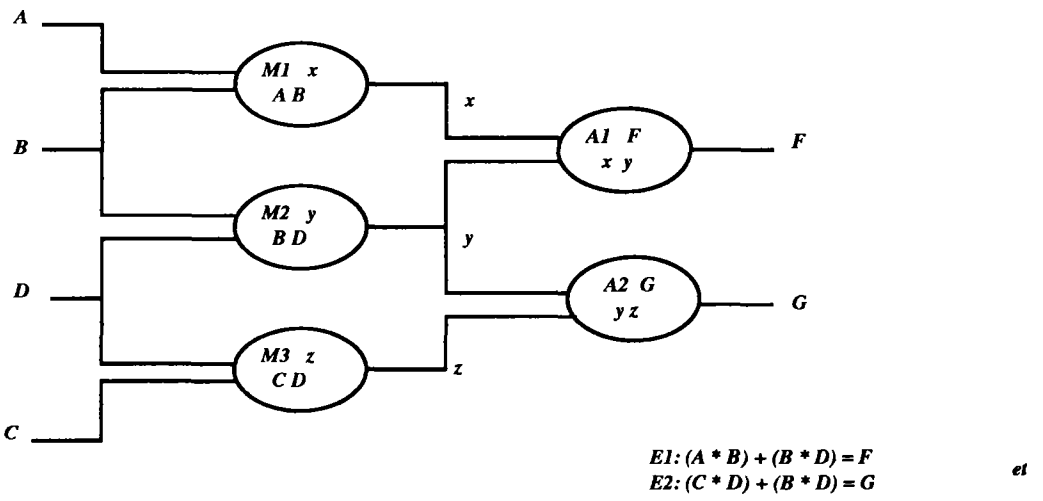


Figure 7: Un exemple de Représentation de Sussmann.

Une autre mode de repésentation des contraintes n-aires est due à Waltz [Waltz 1972] pour les applications dans le domaine de la vision et à Sussman [dK 1980] qui, pour ses applications dans le domaine de diagnostic des circuits en électronique, a introduit les contraintes ternaires. Un exemple de représentation est illustré par le figure 7. Cependant, la technique de résolution de contraintes utilisée dans [Sussman 1980] est limitée aux valeurs entières, uniques et non pas aux domaines de valeurs que nous souhaitons obtenir à cause de la nature notre problème.

Une variante de la méthode de représentation de Sussman par les graphes donne un réseau de contraintes n-aires. Chaque contrainte du réseau peut être représentée par un noeud particulier. Ce noeud a comme domaine de valeurs un sous-ensemble du produit cartésien des domaines de valeurs des variables auxquelles elle est connectée. Les variables et leurs domaines de valeurs associés sont représentées par les noeuds et les étiquettes comme dans les réseaux binaires. Les noeuds représentant les contraintes n-aires sont appelés noeud-contraintes. Pour le même exemple utilisé dans

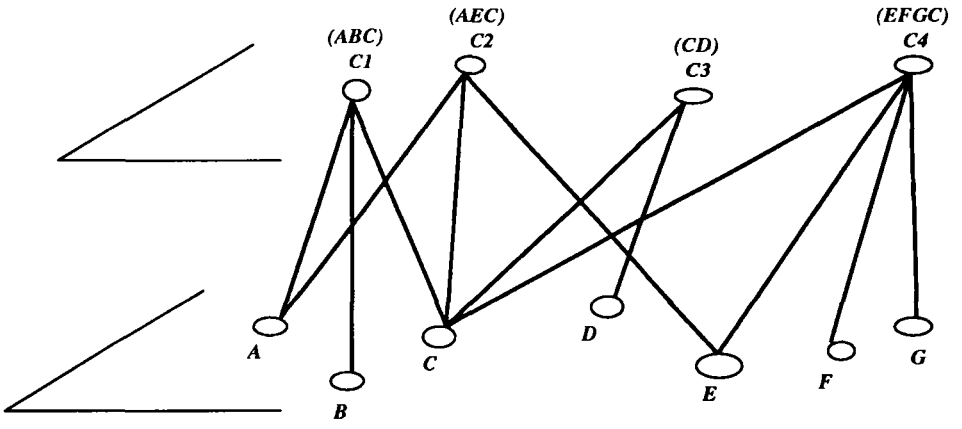


Figure 8: Exemple de Représentation par Graphe de Dépendances.

la représentation par graphe dual et graphe élémetaire ci-dessus, la représentation graphique que nous utilisons est illustrée par le figure 8. Cette représentation est identique à celle des dépendances utilisée dans les systèmes de maintenance de vérité [Goodwin 1982]. Nous lui donnerons le nom de *Graphe de Dépendances.*

Notre proposition consiste à étendre les algorithmes de résolution aux problèmes de satisfaction de contraintes binaires pour les contraintes n-aires. Nous démontrons l'application de notre proposition sur une classe restreinte de contraintes : les contraintes algébriques sur trois variables identiques à celles utilisées par Sussman et de Kleer [dK 1980] aussi que par Steele [SteeleJr 1980].

Nous présentons notre développement sur trois points :

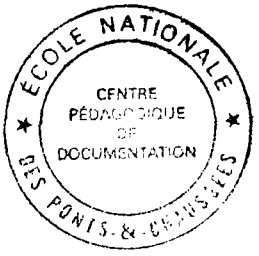
1. L'EXTENSION DU FILTRAGE DE LA CONSISTANCE D'ARC (2-consistance) pour vérifier la consistance d'étiquettes (ou de domaines) dans les résaux n-aires,

2. L'EXTENSION DU FILTRAGE DE LA CONSISTANCE DE CHEMIN (3-consistance) pour vérifier la consistance de relations (ou de

contraintes) dans les réseaux de contraintes n-aires,

3. LA GESTION DES CONTRAINTES n-aires dynamiques (insertions, retraits, explications, etc..) dans l'optique de l'aide à la conception.

Dans les sections suivantes, nous décrivrons le développement point par point. Avant chaque description de notre proposition, nous rappellerons les techniques correspondantes développées pour des réseaux de contraintes binaires.



4.2 La Consistance d'Arc

4.2.1 Notions de Base

Définition 9 *[Mackworth 1977]*

Soit un réseau de contraintes binaires $\mathcal{N} = (X, C, D)$; X est l'ensemble de noeuds représentant les variables x_k ; C, l'ensemble de contraintes binaires ; D est l'ensemble de domaines de valeurs d_k associés aux variables x_k.

Soit une contrainte $C_{ij} \in C$ s'exerçant sur deux variables x_i et x_j appartenant à X ;

Un domaine de valeur d_i associé à une variable x_i est appelé **arc consistant** *si et seulement si quelque soit la valeur a dans d_i, l'instanciation $(V(x_i) = a)$ entraine au moins une valeur $b \in d_j$ telle que $(V(x_j) = b)$ et la relation entre a et b satisfait la contrainte C_{ij} :*

$$AC(d_i) = \ vraie \iff \forall a \in d_i, \forall C_{ij}(x_i, x_j) \in C :$$

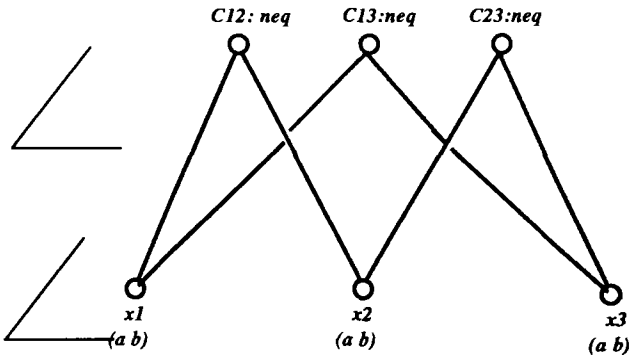


Figure 9: Exemple d'un Réseau Arc-Consistant avec aucune Solution.

$$(V(x_i) = a \Rightarrow \exists V(x_j) = b \in d_j \mid R(a,b) \models C_{ij}(x_i, x_j))$$

Un réseau $\mathcal{N}$ est arc-consistant si tout domaine $d_i \in D$ est arc-consistant. Si $\mathcal{N}$ n'est pas arc-consistant, il n'a évidemment pas de solutions.

Nous remarquons que cette définition correspond à la définiton de k-consistance de Freuder avec $k = 2$. Ainsi, lorsqu'une opération de filtrage est effectuée, tous les $d_i \in D$ possède d'une valeur compatible avec un d_j par une contrainte binaire C_{ij} mettant en jeu x_i et x_j. Cependant, la consistance d'arc reste encore *locale* entre deux variables et n'assure pas nécessairement l'existance de solution *globale* d'un réseau. Par exemple, pour le cas où $X = \{x_1, x_2, x_3\}$; $C = \{C_{12}, C_{23}, C_{13}\}$ avec C_{12}, C_{13}, C_{23} sont identiques et représentent la relation "différence de" ($\neq$). Si $d_{x_1} = d_{x_2} = d_{x_3} = (a, b)$, $\mathcal{N}$ est arc consistant mais ne possède pas de solution. Ce problème est illustré par la figure 9.

Lorsqu'un réseau de contraintes binaires $\mathcal{N}$ n'est pas arc-consistant, on recherche un réseau $\mathcal{N}^\star = (X, C, D^\star)$ dont tous les $d_i^\star$ sont arc-consistants. $\mathcal{N}^\star$ est obtenu à partir de $\mathcal{N}$ en réduisant les domaines de valeurs, de ma-

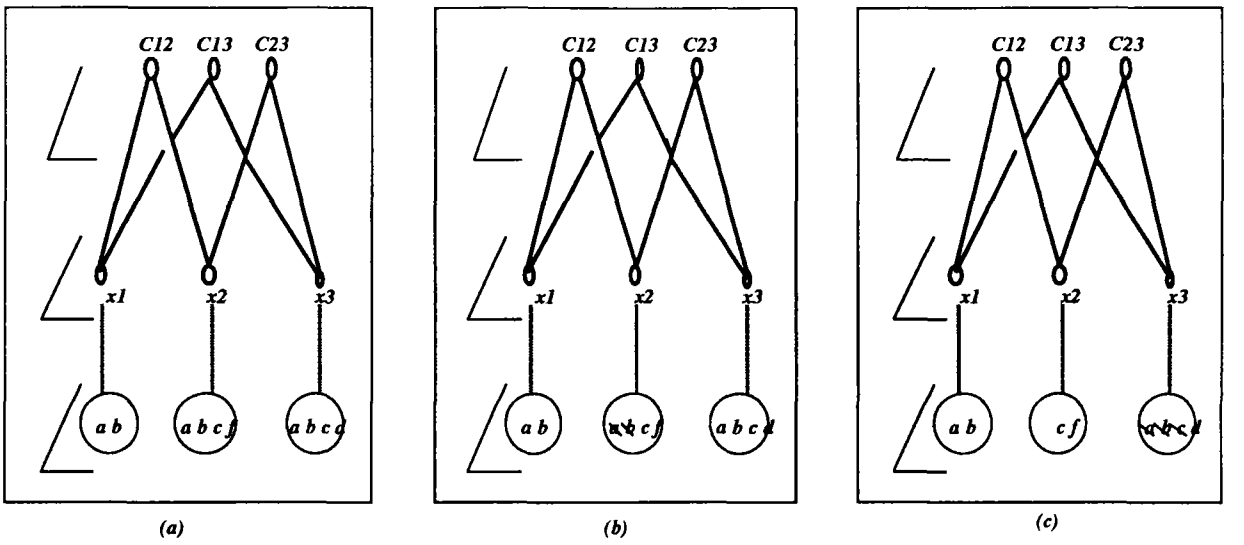


Figure 10: Exemple de Filtrage d'Etiquettes Arc Consistantes: a) état initial b) filtrage sur C_{12}, C_{13} c)filtrage sur C_{23}.

nière à ce que $\mathcal{N}^\star$ vérifient la propriété de consistance d'arc. On appele cette opération le *filtrage d'étiquettes arc-consistantes.* Si, après ces filtrages, aucun domaine de $\mathcal{N}^\star$ n'est vide, il est arc-consistant, sinon, $\mathcal{N}$ n'a pas de solution. Cette procédure définit un filtrage au sens où elle permet de supprimer des valeurs, sans modifier l'ensemble de solutions. Le figure 10 illustre un exemple de filtrage associé aux propriétés d'arc consistance.

Plusieurs algorithmes de consistance d'arc ont été proposés pour améliorer l'algorithme de filtrage proposé par Waltz [Waltz 1972]. Mackworth [Mackworth 1977] a proposé les algorithmes AC-1, AC-2 et AC-3 introduisant les contraintes pour éliminer une partie des valeurs ; Mohr et Henderson [Mohr 1986] proposent une amélioration du temps de calcul par l'algorithme qu'ils appellent AC-4.

En résumé, la consistance d'arc est une propriété pour vérifier la consistance locale d'un réseau de contraintes avec un temps de calcul meilleur qu'avec l'algorithme *"backtrack"* de base. La vérification de cette consis-

tance est effectuée à l'aide d'une opération de filtrage. Il s'avère que, dans le cas général, la consistance d'arc locale ne permet pas l'élimination de toutes les inconsistances. La notion de *"consistance de chemin"* [Montanari 1974] a été proposée comme un filtrage complémentaire aux filtrage associés à la consistance d'arc. Nous examinerons les problèmes liées à la consistance de chemin dans la section 4.3. Pour instant, nous allons examiner le problème de la consistance d'arc appliquée aux contraintes n-aires.

4.2.2 Extension pour les Réseaux n-aires

Nous avons vu que la propriété d'arc consistance est une condition locale entre les domaines de valeurs et les contraintes binaires. Sa définition peut être étendue aisement aux contraintes n-aires par la proposition suivante:

Proposition 2 n-arc-consistance

Soit un réseau de contraintes n-aires $\mathcal{N} = (X, C, D)$. *Soit* $C_{1,\ldots,n} \in C$ *une contrainte n-aires s'exerçant sur les variables* $x_1, \ldots, x_n \in X$.

Un domaine $d_{x_i} \in D$ *est* **n-arc consistant** *si et seulement si quelque soit une valeur* v_i *dans* d_{x_i}, *une instanciation* $V(x_i) = v_i$ *entraine pour chaque variable* $x_j \in \{x_1, \ldots, x_n\}$ *au moins une valeur* $v_j \in d_{x_j}$ *telle que* $V(x_j) = v_j$ *et la relation entre* $(v_1, \ldots, v_n)$ *satisfait la contrainte* $C_{1,\ldots,n}$:

$$NAC(d_{x_i}) = \ vraie \iff \forall v_i \in d_{x_i}; \forall x_j \in \{x_1, \ldots, x_n\}; \forall C_{x_1,\ldots,x_i,\ldots,x_n} :$$

$$V(x_i) = v_i \Rightarrow \exists \{V(x_j) = v_j\}, (v_j \in d_j \text{ et } x_i \neq x_j) \mid R(v_1, \ldots, v_i, \ldots, v_n) \models C_{x_1,\ldots,x_i,\ldots,x_n}$$

Un réseau $\mathcal{N} = (X, C, D)$ *est n-arc-consistant si tous ses domaines* $d_i \in D$ *sont n-arc consistants.*

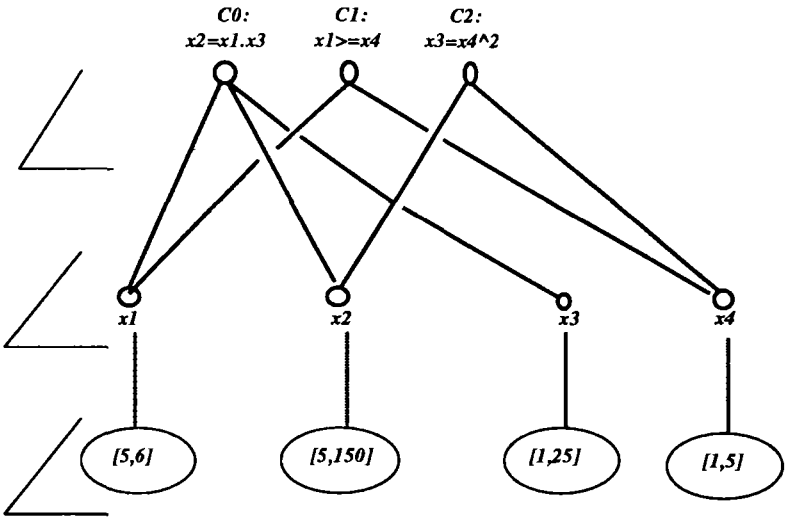


Figure 11: Exemple d'un Réseau N-Arc Consistant.

La figure 11 illustre le cas d'un réseau de contraintes n-arc consistant et la figure 12 donne un cas contraire.

Nous pouvons remarquer que la consistance n-arc n'est pas équivalente à la 2-consistance définie par Freuder. Néanmoins, il est possible de définir à partir de la consistance d'arc une opération de filtrage sur les réseaux n-aires.

4.2.3 Proposition d'un Algorithme de Filtrage

Dans le chapitre précédent, nous avons décrit les techniques de propagation de contraintes et de maintenance de vérité. Une des notions de base dans la maintenance de vérité est celle de l'ensemble *nogood* [Stallman 1979, deKleer 1986a]. Nous avons remarqué que les systèmes de maintenance de vérité opèrent sur des clauses de la logique propositionelle, e.g. LTMS[10] de [McAllester 1980a], ou des clauses de Horn dans le cas du ATMS à base de

[10]Pour distiguer les différents systèmes de maintenance de vérité, Forbus a donné le nom LTMS (Logic-based Truth Maintenance System) à celui de McAllester, JTMS (Justification-based TMS) pour celui de Doyle. ATMS (Assumption-based TMS) étant le label de celui de de Kleer.

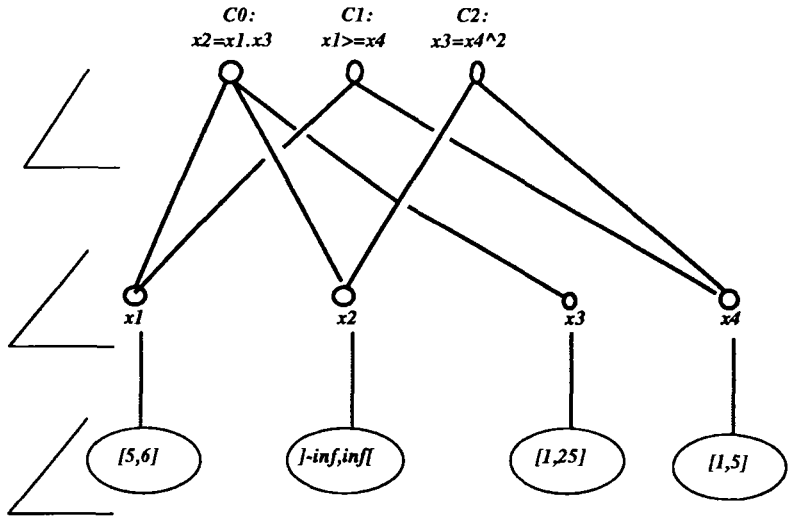


Figure 12: Exemple d'un Réseau N-Arc Inconsistant.

suppositions [deKleer 1986a]. Guy Steele Jr., dans sa thèse sur un langage de contraintes [SteeleJr 1980], utilise également les principes de l'ensemble *nogood* pour gérer les contradictions.

Nous proposons une extension de l' ensemble "nogood" pour le calcul de la consistance de n-arc. Cependant, il n'existe aucune définition formelle de cet ensemble. Nous essayons de formuler cette notion en relation avec les réseaux de contraintes et ensuite de proposer un algorithme pour calculer la n-arc consistance.

Définition 10 contrainte "nogood"

Soit en ensemble de m variables $x_1, \ldots, x_m$ *et un ensemble de m domaines* $d_1, \ldots, d_m$*:*

- *Une* **contrainte n-aire** $C_{1,\ldots,n}$ *est dite* **"nogood"** *sur un sous-ensemble du produit cartesien de* $d_1, \ldots, d_i, \ldots, d_n$ *si ce produit contient au moins un domaine qui n'est pas n-arc consistant.*

- *Une* **relation** $R_{i,\ldots,k}$ *entre les valeurs* $v_i, \ldots, v_k$ *est dite* **"nogood"** *s'il existe aumoins une valeur appartenant à un domain n-arc inconsistant.*

- *Un* **couple de variable-valeur** (x_i, v_i) *est dit* **"nogood"** *si et seulement si pour tout* v_i *appartenant à* d_i*, domaine de valeur associé à* x_i*, l'ensemble de contraintes* C_k*, mettant en jeu la variable* x_i*, aucune combinaison avec* v_i *peut permettre de vérifier la contrainte* C_k*.*

- *Un* **ensemble nogood** *contient les couples de variable-valeur nogoods*[11]*. En d'autres termes, cela signifie tout point de cet ensemble est en contradiction avec une ou plusieurs contraintes.*

Par exemple, considérons un réseau $\mathcal{N} = (X, C, D)$:

$$\mathcal{N} = \begin{cases} X = \{x_1, x_2, x_3, x_4, x_5\} \\ C = \{(C_1 : x_1 \times x_2 = x_3) \\ \qquad (C_2 : x_1 \times x_4 = x_5)\} \\ D = \{(d_{x1} = (a, b, c)) \\ \qquad (d_{x2} = (b, c, d)) \\ \qquad (d_{x3} = (a^2, b^2, c^2, d^2)) \\ \qquad (d_{x4} = (b, f)) \\ \qquad (d_{x5} = (g))\} \end{cases}$$

On constate que $(C_2 : x_1 \times x_4 = x_5)$ est une contrainte "nogood" ; les relations[12] $R^{\times}(a, b, g); R^{\times}(b, b, g)$ *etc*, sont "nogood" pour C_2 ; les couples de variable-valeur $(x_1, a), (x_2, d), (x_3, a^2)$ *etc*, sont "nogood" pour C_1, etc.

[11] at all!

[12] par exemple, la relation $R^{\times}(xyz)$ signifie $x \times y = z$

Notre stratégie dans le filtrage de la consistance de n-arc consiste à (1) construire l'ensemble nogood et à identifier les contraintes nogoods ; (2) pour les contraintes nogoods, ajuster les domaines de valeurs qui ne vérifient pas la consistance de n-arc et supprimer les valeurs dans les couples nogoods. A chaque suppression recommencer à partir de (1) jusqu'à ce que l'ensemble nogood redevienne vide. (3) Finalement, mémoriser les calculs effectués pour chaque déduction de valeurs.[13]

Pour ce filtrage, nous avons besoin de calculer les éléments suivants:

1. INDICATEUR DE COUPLES NOGOODS : Pour tout couple (x_i, v_i) avec $x_i \in X$, $v_i \in d_{xi}$ et pour toute contrainte C_{ak} contenant x_i, $\Delta(x_i, v_i, C_{ak})$ désigne le nombre de tuples de $P_{C_{ak}}$, combinaisons de valeurs des variables vérifiant la contrainte C_{ak}, qui admettent la valeur v_i pour la variable x_i :

$$\Delta(x_i, v_i, C_{ak}) = \begin{cases} 1 & \text{ssi } \exists V(x_i) = v_i | R(v_a, \ldots, v_i, \ldots, v_k) \subseteq P_{C_k} \\ 0 & \text{sinon} \end{cases}$$

Dans cet exemple, $\Delta(x_1, b, C_1) = 1$ puisque $R^{\times}(b, b, b^2) \subseteq P_{C_1}$. Par contre, $\Delta(x_1, a, C_1) = 0$ puisque $R^{\times}(a, b, b^2)$, etc... ne vérifient pas C_1.

Ce calcul permet de construire l'ensemble nogood et de supprimer les valeurs qui ne sont pas admises par la contrainte définissant les relations entre valeurs des variables. Il permettra en outre, de calculer si une contrainte est nogood ou pas.

[13]On peut apparenter cette démarche à la méthode de Karl Popper. En effet, l'ensemble nogood est une façon de définir le domaine de *falsification* du problème posé. On comprend bien qu'augmenter l'étendu de nogood restreint le domaine dans lequel il faut ensuite chercher la solution pour remettre contre l'explosion combinatoire.

2. COMPTEUR DE VALEURS ADMISES : Pour toute contrainte C_{ak} liant $x_a, \ldots, x_k$; $\Gamma(C_{ak}, x_i, d_i)$ indique le nombre de valeurs $v_i \in d_i$, associé à x_i, admises par C_{ak}. S'il existe un couple (x_i, d_i) tel que $\Gamma(C_{ak}, x_i, d_i) = 0$, on dira que C_{ak} est une contrainte nogood.

$$\Gamma(C_{ak}, x_i, d_i) = \begin{cases} 0 & \text{ssi } \forall v_i \in d_i : \Delta(x_i, v_i, C_{ak}) = 0 \\ n & \text{ssi } \forall v_i \in d_i : \sum \Delta(x_i, v_i, C_{ak}) = n \end{cases}$$

Ce calcul permet d'isoler les contraintes nogoods pour les traiter ensuite.[14]

3. INDICATEUR DE "DÉDUCTIVITÉ" :

Si l'on appelle C_k une contrainte nogood, liant un ensemble X de n variables $x_1, \ldots, x_n$. Pour toute contrainte nogood (C_k), $\eta(C_k, X)$ correspond au nombre de variables dans X possédant au moins une valeur admise par le tuple de relations vérifiant C_k :

$$\eta(C_{\vec{k}}, X) = \text{ Card } \{\forall x_i \in X : \Gamma(C_k, x_i, d_i) \neq 0\}$$

Ce calcul s'opère seulement sur les contraintes nogoods. Une contrainte n-aire est déductive[15] si et seulement si au moins $(n-1)$ variables liées à cette contrainte possèdent une valeur admissible. Or, si une contrainte est nogood, il existe au moins un domaine de valeurs qui n'est pas n-arc consistant. Si le nombre de domaines de valeurs n-arc inconsistants est égale à 1, on est

[14] pour éviter des comptages rédondants, on pratique un marquage des variables comptées

[15] On peut l'appeller déduc*ative* puisqu'elle est active pour être déduite

dans le cas précédent. Au contraire, s'il est strictement supérieur à 1, on dira que cette contrainte n-aire n'est pas déduc*active*. Ce calcul permet de vérifier l'existence de solutions.

Résumons nous dans une première phase, nous cherchons à séparer les domaines consistants et les domaines "nogoods". On définit donc un ensemble nogood, puis dans la phase qui vient d'être présentée, on affine l'analyse de l'ensemble nogood pour esayer de rechercher si certaines des contraintes qui conduisent à définir l'ensemble nogood sont réellement "nogoods"ou si elles sont "nogoods" mais déductives. ce qui permet de penser qu'il existe une solution sous réserve de vérification de la cohérence. Si cette deuxième phase d'analyse conduit à concluire qu'il existe au moins une contrainte qui est nogood et indéductive, on peut concluire à l'impossibilité de resoudre le réseau.

Proposition 3 Algorithme de filtrage d'étiquettes

Le filtrage d'étiquettes n-arc consistantes d'un réseau de contraintes n-aires consiste donc à rechercher le sous-réseau dont les domaines de valeurs D vérifient la consistance de n-arc . L'algorithme que nous proposons procède en trois étapes:

1. *la première étape construit le graphe associé et calcule les structures "Δ", "Γ" et "η" ;*

   Le calcul de "Δ" permet de construire l'ensemble "nogood" contenant les paires $(x_i, d \in d_i)$ qui ne peuvent apparaitre dans le filtrage de la consistance n-arc. Ces paires $(x_i, d \in d_i)$ sont telles que l'un des compteurs $\Delta(x_i, d, C_k)$ est nul.

   Le calcul de "Γ" permet d'identifier les contraintes "nogoods" et

le nombre de tuples de valeurs admissibles par chaque contrainte nogood.

Le calcul de "η" permet d'évaluer l'existence de solution. Si aucune solution n'existe, on propose à l'utilisateur de reviser ou d'ajuster les domaines de valeurs susceptible d'être modifiés (donc le calcul de $\Gamma(C_k, x_i, d_i) = 0$).

2. *la seconde étape consiste à supprimer les paires (x, d) contenues dans l'ensemble "nogood" ; à modifier les domaines de valeurs et à propager les conséquences de ces suppressions aux noeuds voisins (filtrage de domaines de valeurs propement dit). Les calculs de "Δ", "Γ" et de "η" sont mis à jour après chaque suppression et peuvent permettre de déduire d'autres noeuds dans l'ensemble nogood. Ainsi, le processus s'opère récursivement sur les contraintes.*

3. *la dernière étape consiste à construire le graphe représentatif de la n-arc consistance du réseau et à mémoriser les causes et effets des calculs effectués.*

4.3 La Consistance de Chemin

4.3.1 Notions de Base

La consistance de chemin a été initialement définie, pour les réseaux binaires, comme une condition de la consistance locale, portant sur l'existence de valeurs compatibles entre deux noeuds reliés par des contraintes distinctes.

Dans le cas montré dans 4.2.1, elle signifie la consistance entre les valeurs associées à x_1 et x_3 via C_{12} et C_{23} et est illustrée par le figure 13.

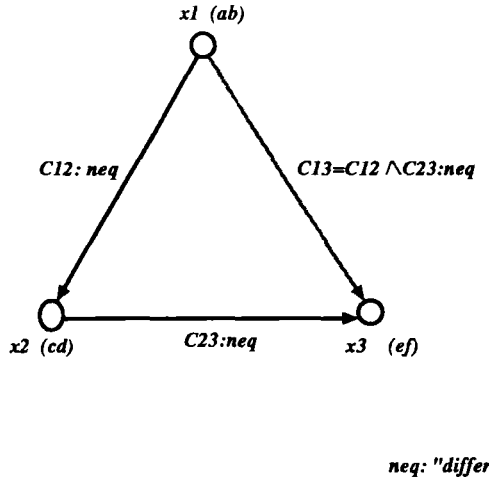


Figure 13: Exemple de la Consistance de Chemin binaire.

La définition de cette condition de consistance peut être résumée ainsi :

Définition 11 *[Montanari 1974]*

Considérons deux contraintes binaires $C_{ij}(x_i, x_j)$ *et* $C_{jk}(x_j, x_k)$. *Le couple de variables* (x_i, x_k) *est dit* **chemin-consistant** *si et seulement si quelque soit la valeur* $v_i \in d_i$ *associée à* x_i, *quelque soit la valeur* $v_k \in d_k$ *asssociée à* x_k, *il existe au moins une valeur* $v_j \in d_j$ *associée à* x_j *telle que la relation induite* $R(V(x_i) = v_i, V(x_k) = v_k)$, *associant* $R(V(x_i) = v_i, V(x_j) = v_k)$ *et* $R(V(x_j) = v_j, V(x_k) = v_k)$, *vérifie les contraintes* C_{ij} *et* C_{jk}.

Soit un réseau de contraintes binaires $\mathcal{N} = (X, C, D)$; *Soit* $R_{ij}(v_i, v_j)$ *la relation qui représente le cas où* (v_i, v_j) *est admis par la contrainte* C_{ij}. *Un couple de variable* (x_0, x_n) *est dit chemin-consistant à travers une séquence de noeuds* $(x_0, x_1, \ldots, x_n)$ *si et seulement si quelque soit la valeur* $v_0 \in d_{x_0}$ *et* $v_n \in d_{x_n}$ *telles que* $R_{x_0 x_n}(v_0, v_n)$, *il existe une séquence de va-*

leurs $v_1 \in d_{x_1}, \ldots, v_{n-1} \in d_{x_{n-1}}$ *telle que* $R_{x_0x_1}(v_0, v_1)$ *et* $R_{x_1x_2}(v_1, v_2)$ *et* ... *et* $R_{x_{n-1}x_n}(v_{n-1}, v_n)$.

$\mathcal{N}$ *est "chemin-consistant" si et seulement si tout couple de variables est chemin-consistant.*

Cette propriété correspond à la k-consistance de Freuder quand $k = 3$. Ainsi, lorsqu'un réseau est chemin consistant, toute instanciation compatible de couples de variables peut être propagée à une troisième variable.

On peut, comme pour la consistance d'arc, définir une opération de filtrage associée aux propriétés de la consistance de chemin. Rappelons que la consistance d'arc permet de filtrer les domaines $d_i \in D$ associés aux variables $x_i \in X$ tandis que la consistance de chemin permet de filtrer sur les contraintes $C_k \in C$ reliant les variables x_i. Ce filtrage est appelé *filtrage de contraintes consistantes*. Son effet est de réduire les relations définies par les contraintes, en éliminant les couples de valeurs des variables liées par les contraintes qui ne vérifient pas la consistance de chemin. Elle peut également conduire à l'insertion de nouvelles contraintes dans un réseau intial ; lorsqu'une relation binaire $R_{ij}(x_i, x_j)$ est inserée entre un couple de variables (x_i, x_j), on considère deux cas de figure:

1. s'il existe une contrainte directe C_k entre x_i, x_j, cette nouvelle relation $R_{ij}(x_i, x_j)$ remplacera la contrainte C_k initialement définie.

2. s'il n'exsite pas de contrainte directe C_k liant x_i, x_j, on dira que x_i, x_j sont liées de manière implicite par une relation induite. Insérer une contrainte $R_{ij}(x_i, x_j)$ a l'effet de restreindre la relation à un sous-ensemble de $d_i \times d_j$.

Identiquement à l'opération de filtrage associé aux propriétés d'arc consistance, à partir d'un réseau de contraintes binaires $\mathcal{N} = (X, C, D)$, on cherche un réseau $\mathcal{N}^\star = (X, C, D^\star)$, chemin consistant et équivalent à $\mathcal{N}$.

On trouve dans la littérature l'algorithme "path-consistency-3" de Mohr et Henderson [Mohr 1986] traitant le filtrage de chemin consistant. On note que les propriétés de la consistance de chemin est complémentaire à celle d'arc consistant. Elle permet d'éliminer davantage de valeurs inconsistantes mais pour un temps de calcul plus élévé. Une différence essentielle entre ces deux filtrages provient du fait que la consistance d'arc ne modifie pas le graphe de contraintes alors que la consistance de chemin peut permettre d'insérer de nouvelles contraintes dans un réseau, ce qui est essentiel dans l'approche d'aide à la conception.

4.3.2 Extension aux Contraintes n-aires

Nous avons décrit notre approche pour traiter la consistance d'arc dans les réseaux de contraintes n-aires. Nous nous intéressons à présent à l'extension de la consistance de chemin appliquée aux réseaux n-aires. Nous proposons pour cela un algorithme de filtrage reposant sur les ensembles "nogoods" comme le problème de n-arc consistance. Cependant, pour les réseaux binaires, le calcul de la consistance de chemin croît linéairement avec le nombre de variables et le nombre de contraintes mais il est exponentiel dans le cas de réseau de contraintes n-aires puisque dans le dernier cas, il faut calculer la cohérence des valeurs non seulement entre deux variables mais entre une variable et toute autre variable liée aux contraintes indirectes. Par exemple, pour une contrainte donnée, liant un nombre n de variables, le filtrage de chemin consistance consiste à vérifier la cohé-

rence de toutes les variables contenues dans une contrainte avec m autres variables contenues dans une autre contrainte qui a avec la première une variable commune. Il faut donc vérifier un nombre n^{m-1} de couples (x_i, x_j). Ce calcul est coûteux puisqu'il est exponentiel avec le nombre de variables et contraintes.

Nous tentons, à présent, de définir les propriétés de la consistance de chemin pour les réseaux de contraintes n-aire.

Pour introduire la notion de la consistance de chemin dans les réseaux de contraintes n-aires, rappelons la définition générale des relations induites sur les valeurs des couples de variables (cf. définition 3). La consistance de chemin exprime une condition sur cette relation : les valeurs a et b pour un couple de variables (x_i, x_j) peuvent être en relation, si pour toute contrainte où figure la variable x_i, il existe un tuple de contraintes admettant la valeur a pour x_i et tel que toutes les valeurs de ce tuple sont en relation avec la valeur b de x_j satisfaisant la relation R_{ij}.

La figure 14 montre la différence entre la consistance de chemin dans le cas de contraintes binaires et dans le cas de contraintes n-aires.

On remarque que dans un réseau de contraintes n-aires, il existe toujours au moins un chemin entre deux variables, x_i et x_j quelque soit i, j. S'il il est impossible de trouver le chemin entre une variable x_i et x_j quelconques, le réseau représente donc deux problèmes indépendants dont chaque variable appartient à un problème.

Définition 12 Consistance de Chemin dans un Réseau de Contraintes n-aires

Soit un réseau de contraintes n-aires $\mathcal{N} = (X, C, D)$, toute contrainte n-aire C_i opérant sur n variables $x_{i1}, \ldots, x_{in}$. Considérons deux variables x_{ik},

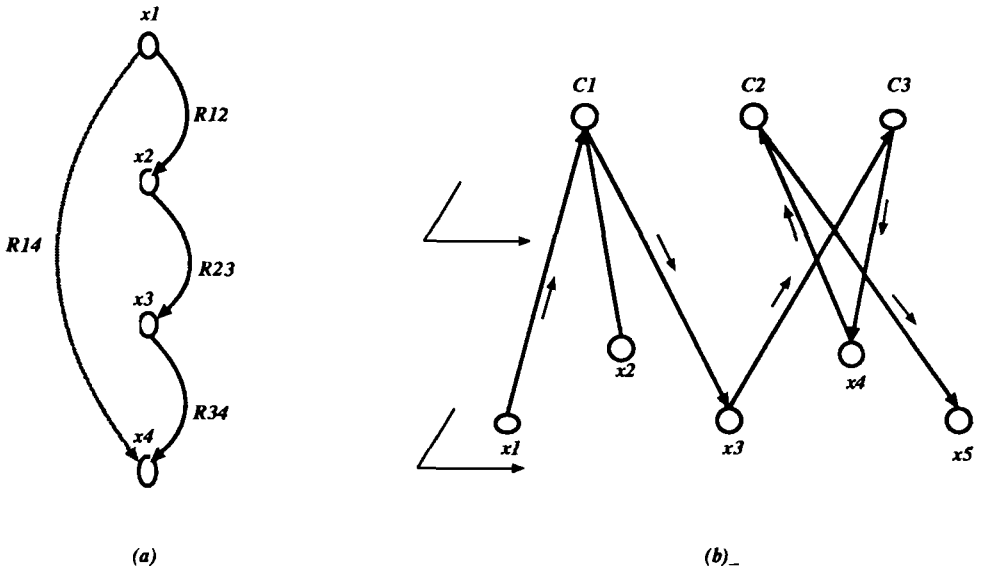


Figure 14: Chemin Consistant (a) dans le cas binaires (b) dans le cas n-aires.

contenue dans C_i, et x_{jl}, contenue dans C_j.

Un chemin, défini par le couple de variable (x_{ik}, x_{jl}) est dit consistant si et seulement si quelque soit l'instanciation $V(x_{ik})$ de x_{ik} et quelque soit l'instanciation $V(x_{jl})$ de x_{jl}, il existe une séquence de validation des variables contenues dans la contrainte indirecte C_{ij}, reliant C_i et C_j, telle que la relation $R_{ij}(V(x_{ij1}), \ldots, V(x_{ijn}))$ vérifie la contrainte indirecte C_ij.

$$\forall V(x_{ik}) \in d_{ik}, \forall V(x_{jl}) \in d_{jl}|$$

$$PC(x_{ik}, x_{jl}) \Longleftrightarrow \exists (V(x_{ij_1}) \in d_{ij_1}, \ldots, V(x_{ij_m}) \in d_{ij_m}) :$$

$$R_{ij}(V(x_{ij_1}), \ldots, V(x_{ij_m})) \models C_{ij}$$

Si un séquence de m contraintes indirectes $(C_{ij_1}, \ldots, C_{ij_m})$ est nécessaire pour établir un chemin entre x_{ik} et x_{jl} dont $x_{ik} \in C_{ij_1}$ et $x_{jl} \in C_{ij_m}$. Le couple de variables (x_{ii}, x_{jj}) est chemin consistant si et seulement si quelque soit l'instanciation de x_{ik} dans d_{ik} et de x_{jl} dans d_{jl}, il existe une

séquence d'instanciation des variables appartenant à $C_{ij_1}, \ldots, C_{ij_m}$ *telle que les relations entre ces instanciations vérifient ces contraintes indirectes.*

Un réseau de contraintes n-aires $\mathcal{N}$ *est dit chemin-consistant si et seulement si tout couple de variables est chemin consistant.*

Il importe de noter que la définition que nous proposons n'est pas équivalente à la 3-consistance définie par Freuder. Cependant, la consistance de chemin de (x_i^a, x_i^a) signifie que toute valeur appartenant à d_i associée à x_i est admise par toute ou aucune des contraintes contenant x_i.

Comme dans le cas des contraintes binaires, la consistance de chemin ne mène pas à de relations directes entre la consistance de n-arc et celle de chemin. Un réseau de contraintes peut être n-arc consistant mais pas nécessairement chemin consistant ou inversement. Un réseau est dit *globalement consistant* s'il est à la fois n-arc consistant et chemin consistant. Dans le paragraphe suivante, nous proposons un algorithme de filtrage associé à la propriété de la consistance globale faisant intervenir le calcul de la consistance de n-arc et de chemin.

4.3.3 Proposition d'un Algorithme de Filtrage

La nécessité de mettre en pratique la notion de programmation adéquate nous a amèné à introduire les formules supplémentaires suivants pour vérifier la consistance de chemin d'un réseau de contraintes:

1. Pour tout couple de variables (x_i, x_j), $\Pi(x_i, x_j)$ est un prédicat indiquant l'existence d'au moins une contrainte modélisant la relation entre x_i et x_j:

$$\Pi(x_i, x_j) = \text{ vrai ssi } \exists C_{ij} \in C : C_{ij} \subseteq d_i \times d_j$$

2. Pour tout couple de variables (x_i, x_j) et pour tout couple de valeurs $(a, b) \subseteq d_i \times d_j$, le couple de prémisses $\Upsilon(x_i^a, x_j^b)$ est un prédicat indiquant l'existence dans $\mathcal{N}$ d'une contrainte reliant x_i et x_j et admettant les valeurs a pour x_i et b pour x_j:

$$\Upsilon(x_i^a, x_j^b) = \text{ vrai ssi } \Pi(x_i, x_j) = C_{ij} \text{ et } R_{ij}(V(x_i) = a, V(x_j) = b) \models C_{ij}$$

Proposition 4 algorithme de filtrage de contraintes consistantes

Le filtrage de contraintes consistantes d'un réseau de contraintes n-aires $\mathcal{N} = (X, C, D)$ consiste à rechercher le réseau $\mathcal{N}_1^\star = (X, D, C_1^\star)$ qui vérifie la consistance de chemin. L'algorithme que nous proposons procède en trois étapes:

1. CONSTRUIRE LES STRUCTURES $\Pi(x_i, x_j)$ ET $\Upsilon(x_i^a, x_j^b)$: *Vérifier pour tout $x_i \in X$ et pour tout $x_j \in X$ les indicateurs $\Pi(x_i, x_j)$ et $\Upsilon(x_i^a, x_j^b)$, puis marquer les contraintes et les instanciations vérifiées ;*

2. INSTANCIATION DES COUPLES VARIABLES-VALEURS: *Pour toute variable $x_i \in X$ et toute variable $x_j \in X$, considerons deux cas de figures:*

   - *si $\Pi(x_i, x_j)$ est vérifié, alors on instancie une variable inetrene x_{ij} et on l'ajoute dans X, puis on vérifie l'existence de $\Upsilon(x_i^a, x_j^b)$ pour toute valeur $a \in d_i$ et toute valeur $b \in d_j$, à chaque itération, si $\Upsilon(x_i^a, x_j^b) = T$, on ajoute (a, b) dans D_{ij}, domaine de valeurs associé à x_{ij};*

   - *si $\Pi(x_i, x_j)$ n'est pas vérifié, on instancie tout de même x_{ij} comme une variable interne et en ajoute dans X, son do-*

maine de valeurs D_{ij} qui est le produit cartesien $d_i \times d_j$. Ce domaine D_{ij} sera traité par le filtrage de n-arc.

On ajoute enfin, le domaine D_{ij} nouvellement créé dans D.

3. CONSTRUIRE L'ENSEMBLE DE CONSTRAINTES $C_1^\star$: *Pour chaque variable $x_{ij} \in X$ établir une relation induite*[16] *composant la contrainte contenant x_i et celle contenant x_j. Le domaine de valeurs associé à chaque relation est calculé à partir de ces nouvelles contraintes et enregistrer dans une structure provisoire associée aux nouvelles contraintes*[17]*. Revenir à l'étape (1) pour examiner les nouvelles contraintes et les nouveaux couples de variables.*

Dans l'étape (3) de la proposition 4, nous nous limitons au cas des contraintes ternaires. Cependant, pour traiter les contraintes $n - aires$, les combinaisons de relations peuvent être obtenues par les types de contraintes à manipuler. A titre d'exemple, pour représenter une équation du type $\sum_{i=1}^{n} X_i = m$, il faudra définir une fonction récursive sur chaque X_i pour construire les contraintes entre d, X_i et les autres variables X_i ; $C_1^\star$ sera très grand par rapport à C, dont le calcul devient très coûteux.

4.4 Un Algorithme de Filtrage Global

Le filtrage de contraintes consistantes permet de simplifier les instanciations inutiles en interdisant des couples de valeurs qui ne satisfont pas les contraintes. Cette opération ne permet pas le filtrage sur les domaines. Nous

[16] cette opération d'instanciation de relations induites est effectuée par des règles de calcul définies par le type de contraintes, e.g. $a + b = c$ permet d'induire $c - b = a$ et $c - a = b$ par exemple.

[17] ces domaines de valeurs sont parfois appelés domaines de classe d'équivalence [SteeleJr 1980]

proposons une opération de filtrage global qui permettra de réduire à la fois les contraintes et les domaines d'un réseau. Cet algorithme consiste en trois étapes:

- étape 1: filtrage de contraintes consistantes pour vérifier la consistance de chemin (cf. proposition 4),
- étape 2: filtrage de domaines (ou d'étiquettes) consistants pour vérifier la consistance de n-arc (cf. proposition 3),
- étape 3: transformer le résultat en un réseau globalement consistant (cf. proposition 5).

Le résultat de l'étape 1 est un réseau qui vérifie la consistance de chemin. Il est ensuite vérifié dans l'étape 2 par le filtrage d'étiquettes consistantes qui produit un graphe représentatif des contraintes $C_1^\star$ et des domaines $D^\star$ consistants. Pour ramener à un réseau $\mathcal{N} = (X, C, D^\star)$ globalement consistant, solution de $\mathcal{N} = (X, C, D)$, il faut donc de transformer $\mathcal{N}^\star = (X, C_1^\star, D^\star)$. La dernière étape de l'algorithme de filtrage global est proposée comme suit:

Proposition 5 transformation de $\mathcal{N}^\star = (X, C_1^\star, D^\star)$ en $\mathcal{N} = (X, C, D^\star)$

Cet algorithme consiste en deux étapes:

1. réduction des relations induites: *la réduction des relations induites est dynamique et est effectuée par l'attribution de domaines vides aux contraintes qui ne sont pas activées pour le filtrage de la consistance de n-arc.*

2. "purge" des ensembles "nogoods": *les ensembles de valeurs associés aux relations induites sont supprimés du réseau*[18].

L'algorithme de filtrage de la consistance globale a été implémenté en langage Allegro Common-Lisp de Franz Inc. et a été testé sur les machines MacIntosh II, Sun/unix et Symbolics 36xx.

4.5 Gestion des Contraintes: Fonctionalités d'aide à la conception

L'algorithme que nous proposons ci-dessus vise à traiter les deux problèmes fondamentaux posés : la réduction des instanciations inutiles par le filtrage et la gestion de la consistance dans les réseaux de contraintes n-aires. Il reste le problème de gestion des insertions et retraits de contraintes et les fonctions d'aide aux utilisateurs.

Rappelons que le processus de conception par modèles de contraintes décrit dans le chapitre 2 consiste à augmenter progressivement l'ensemble des contraintes pour diminuer l'espace des solutions. Le choix alternatif des solutions requiert la possibilité de retrait de contraintes. En plus, notre approche d'aide à la conception est basée sur le dialogue entre l'utilisateur et le mécanisme de gestion de contraintes. Pour ce faire, il est nécessaire de développer les fonctions d'aide à la conception comme par exemple, l'explication des dépendances entre une variable et d'autres variables, ou comment une valeur a été trouvée dans l'espace de solutions. Ce mécanisme est basé sur la mémorisation des calculs effectués au cours des calculs de la consistance.

[18]cette opération est en fait une fonction de "garbage collection" qui consiste à supprimer les pointers qui ne sont pas utiles dans la mémoire virtuelle du système

Comment formaliser ces mécanismes ?

Dans cette section, nous décrivons les principes de base que nous avons utilisés dans l'implémentation du gestionnaire de contraintes.

4.5.1 Gestion de l'Insertion de Contraintes

Si l'on considère un réseau de contraintes initial $\mathcal{N}$, représentant une description partielle de l'artéfact à concevoir, chaque insertion ou retrait de contrainte transformera $\mathcal{N}$ en $\mathcal{N}_1$. Ainsi, une séquence de n insertions et retraits peut être définie par une séquence de n réseaux $\mathcal{N}, \mathcal{N}_1, \ldots, \mathcal{N}_n$ où chaque réseau est le résultat d'une modification du réseau précedent. On peut appliquer le mécanisme défini dans la section précédente comme un mécanisme de résolution indépendamment sur chaque réseau. Cependant, il est fastidieux de procéder ainsi puisque la résolution de $\mathcal{N}_i$ peut s'appuyer sur les informations fournies par $\mathcal{N}_{i-1}$. Avec ces informations, les calculs répétitifs sur les mêmes informations peuvent être évités.

Le problème de gestion de l'insertion de contraintes est défini comme suit :

Considérons un réseau $\mathcal{N}_i = (X, C, D)_i$ globalement consistant et possèdant des solutions, chaque insertion de contraintes dans $\mathcal{N}_i$ consiste à transformer $\mathcal{N}_i$ en $\mathcal{N}_{i+1} = (X, C, D)_{i+1}$. La gestion de l'insertion de contraintes consiste à repondre aux questions suivantes:

1. $\mathcal{N}_{i+1}$ possèdera-t-il des solutions consistantes ? et
2. Sachant qu'à chaque insertion, la taille de $\mathcal{N}_{i+1}$ est supérieure à celle de $\mathcal{N}_i$, comment exploiter les informations fournies par $\mathcal{N}_i$?

En ce qui concerne la question (1), le problème est identique à celui

de résolution des réseaux de contraintes traitée précédement. Il s'agit alors de filtrer $\mathcal{N}_{i+1}$ et d'en vérifier la consistance globale.

En ce qui concerne l'exploitation des informations disponibles dans $\mathcal{N}_i$, on peut envisager deux possibilités:

- pour le filtrage des domaines de $\mathcal{N}_{i+1}$, les domaines et tuples fournis par $\mathcal{N}_i$ ne seront pas remis en cause, donc pour toute combinaison de variables, on ne considère que les nouveaux couples.

- pour le filtrage de contraintes $\mathcal{N}_{i+1}$, seules les domaines de valeurs des variables définies dans les nouveaux couples de variables seront soumis à l'opération de suppression de valeurs.

En résumé, à chaque insertion d'une nouvelle contrainte C_i dans le réseau, seuls les couples de valeurs représentant les relations induites par C_i dans C seront soumis à l'opération de filtrage globale. Si au cours de l'opération, un domaine de valeur d'une variable x_i existant est réduit, les couples de variables dans lesquels x_i figure seront à leur tour filtrés. Ainsi, l'opération se propage jusqu'à ce que tous les domaines soient consistants.

4.5.2 Gestion du Retrait de Contraintes

Il se peut que pendant le processus de conception, l'ingénieur-concepteur ait besoin d'explorer plusieurs cas de figures avec plusieurs ensembles de valeurs ou de contraintes. Dans ce sens, le retrait de contraintes est une fonctionalité que le gestionnaire de réseau de contraintes doit assumer.

A une étape donnée i dans le processus de conception, un réseau $\mathcal{N}_i$ est globalement consistant donc pour toute variable $x_i \in X \mid \exists a \in d_i$ satisfaisant toute contrainte C_k contenant x_i. Le retrait d'une contrainte C_j dans

$\mathcal{N}$ est une transformation de $\mathcal{N}_i$ en $\mathcal{N}_{i+1}$ où l'ensemble de contraintes dans $\mathcal{N}_{i+1}$ est exactement le même que pour $\mathcal{N}_i$ à une contrainte supplémentaire C_j près.

Si l'on ne condidère que les contraintes ternaires du type C_{xyz} dans les modèles de contraintes algébriques, le problème de retrait de contraintes peut être posé de la manière suivante:

1. Au retrait d'une contrainte C_{xyz}, les couples de variables $((xy)z)$, $((zy),x)$, $((zx),y)$ détermineront les autres couples de variables dans lesquelles (xy), (zx), (zy), x,y,z interviennent pour procéder à une opération de filtrage avec le nouvel ensemble de contraintes. Est-ce que le nouveau réseau $\mathcal{N}_{i+1}$ possèdera des solutions ?

2. Si le retrait d'une contrainte provoque des contradictions au niveau des domaines de valeurs des variables jusqu'alors consistants, comment gérer ces contradictions ?

3. Si $\mathcal{N}_{i+1}$ ne possède aucune solution, le filtrage aurait du supprimé toutes ou certaines valeurs associées aux variables, comment connait-on l'effet de chaque contrainte et donc comment déterminer quelles sont les suppressions qu'elles ont entrainées ?

En ce qui concerne le problème d'existence de solution, lorsqu'une contrainte est retirée de l'ensemble de contraintes, deux cas de figures peuvent apparaitre:

- (a). Si les chemins entre x,y,z sont consistants avec les autres variables, seules les valeurs de x,y,z sont effaçées pour calculer de nouvelles valeurs chemin-consistantes. Le réseau possède

de solutions qui peuvent être différentes de celles connues jusqu'alors.

- (b). Si un des chemins entre x, y, z et les autres variables n'est pas consistant, le réseau ne possède pas de solution, et les domaines de valeurs associés aux variables "coupables" de l'inconsistance seront supprimés de $\mathcal{N}_{i+1}$. De nouveaux domaines de valeurs sont alors nécessaires pour obtenir des solutions.

Le gestionnaire de contraintes, dans ce dernier cas, devra interrompre le filtrage pour demander aux utilisateurs de donner de nouveaux domaines de valeurs.

En ce qui concerne le problème de l'existence d'une contradiction entre deux valeurs de deux variables[19], il est nécessaire de retirer soit l'un ou l'autre domaine de valeurs initialement défini. Nous avons mis en place un mécanisme permettant de vérifier le "poids" d'une valeur par rapport à une autre. Ce "poids" est évalué par un *degré de croyance* associé à l'attribution d'une valeur à une variable. Pour l'instant, trois niveaux de croyance sont inclus dans le mécanisme :

1. **constante** est le niveau le plus "fort", la valeur attribuée avec ce degré de croyance ne peut pas être changée au cours de la déduction ;

2. **défaut** est le niveau intermédiaire, plus "souple" qu'une valeur *constante*, elle peut être remplacée par une valeur ayant le degré de croyance "constante" s'il existe une contradiction entre ces deux valeurs ;

[19]connue par le filtrage de domaines consistants (n-arc consistance)

3. **hypothèse** est le niveau le plus "souple". S'il existe une contradiction entre une valeur de ce degré de croyance et une valeur ayant un autre degré de croyance ("constante" ou "défaut"), la première sera remplacée par la dernière.

Dès qu'une contradiction est détectée, deux cas sont possibles :

- (a). Si les deux valeurs appartiennent à deux degrés de croyance différents, celle ayant le degré plus "fort" remplacera automatiquement celle ayant le degré plus "souple" et le filtrage de n-arc continue ;

- (b). Si les deux valeurs ont le même degré de croyance, le gestionnaire s'interrompt pour demander à l'utilisateur de retirer l'une ou l'autre valeur attribuée aux variables dites "coupables".

Dans le cas où le réseau ne possède aucune solution à cause de la suppression automatique de certaines valeurs au cours des opérations de filtrages, pour retrouver les raisons pour lesquelles ces suppressions ont été effectuées, il est nécessaire de mémoriser les causes et effets de chaque suppression. La technique adoptée consiste à introduire la notion de "source" et de "produit" dans l'opération de filtrage de domaines consistants. Une valeur est dite "source" si elle permet de calculer une autre valeur, la dernière est appelée "produit". Nous pouvons considérer une contrainte comme un processeur, ayant des *terminaux*[20]. Une variable élémentaire est connectée à un terminal via un *conduit* (arc), une variable peut être connectée à plusieurs processeurs par plusieurs arcs-conduits. Lorsqu'une variable alimente une valeur à un processeur, ou qu'une valeur est déduite pour une variable,

[20]Une contrainte ternaire a trois terminaux ; une contrainte n-aire a n terminaux

les informations correspondantes sont enregistrées dans un *"registre"* associé au processeur en question. Sur ce registre, on peut retracer les causes et effets de chaque déduction et supression. Cependant, si l'on mémorise toutes les informations pour tous les réseaux dans la séquence d'insertions et retraits de contraintes, la taille mémoire risque de poser un problème. Nous choisissons de mémoriser uniquement les causes et effets pour le dernier réseau et d'effacer ceux appartenant à l'avant dernier. Il importe de noter que nous ne sommes pas complètement satisfait de ce mécanisme de mémorisation des cause-effets. Cependant, des améliorations peuvent être apportées par une gestion plus avancée de la mémoire virtuelle du système dépendant de l'architecture de la machine utilisée.

4.5.3 Explication des Résultats

Dans le premier chapitre, nous avons décrit la notion de causalité. En effet, une contrainte exprime une relation entre les valeurs de variables contenues dans cette contrainte et il est courant que la valeur d'une variable est la "cause" de l'attribution d'une autre valeur à une autre variable, "effet" de la relation exprimée. Nous nous appuyons sur cette causalité logique pour établir une autre fonction d'aide à la conception : l'explication des résultats de calculs.

A tout moment pendant le processus de génération ou de suppression de contraintes, l'utilisateur peut s'informer sur l'état du réseau, notamment sur la valeur d'une variable quelconque. La fonction d'explication consiste à synthétiser les contraintes et les variables par lesquelles une information est obtenue. En d'autres termes, cette fonction consiste à établir un chemin entre une variable et celle questionnée. Si ce chemin est consistant, elle

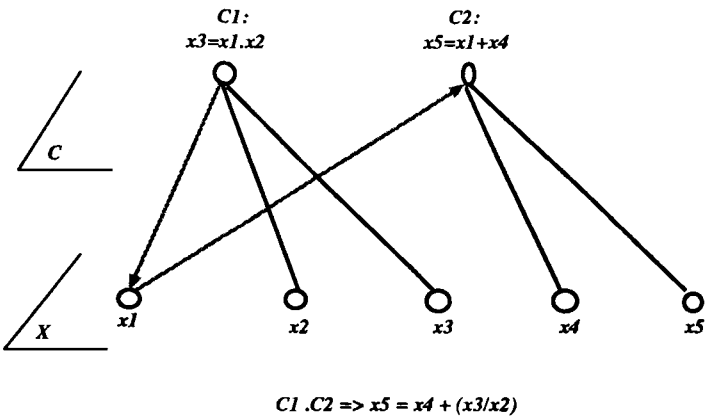


Figure 15: Exemple d'un réseau de contraintes, Explication de Résultats.

affiche les causes et effets de chaque opération de calcul à travers les contraintes induites ou par une séquence d'instanciations sur les contraintes intermédiaires.

Le mécanisme de base est celui utilisé pour mémoriser les causes et effets décrit dans la sous-section 4.2.2. Cependant, nous avons uniquement testé les contraintes exprimant les relations du type équations algébriques simples. Par exemple (cf.figure 15) :

```
>(Etat? x_5)

;|La valeur b^2 dans noeud x_5 est calculee de maniere suivante:
;|    x_5 <- (+ (/ x_3 x_2) x_4)
;|    x_4 <- (defaut b)
;|    x_3 <- (hypothese b)
;|    x_2 <- (constant b^2)
```

Ainsi, pour les calculs plus compliqués, le chemin le plus court est établi et pour chaque variable déduite, il explique les causes de chaque calcul.

Le gestionnaire de contraintes, contenant l'ensemble des mécanismes de filtrage, de gestion de contradiction, de mémorisation des causes-effets, d'explication des résultats, est implémenté et testé sur les machines Symbolics avec Symbolics-Lisp, sur les machines Unix et MacIntosh II avec Allegro-

Common-Lisp. Nous lui donnons le nom NETGEN.

Dans le chapitre suivant, nous montrons quelques exemples simples de l'utilisation de NETGEN.

Chapitre 5
Exemple de Manipulation des Contraintes - Conclusion

"Il y a des cas où la manipulation formelle
réussit dans l'immédiate ;
mais il n'est jamais sage de s'en contenter
sans espérer encore mieux."

Benoit Mandelbrot
(p.228, Les Objets Fractals, 1989)

Dans le chapitre précédent, nous avons présenté le développement d'un gestionnaire de contraintes NETGEN. Nous décrivons à présent l'utilisation de ce gestionnaire et quelques exemples simples de manipulation avec insertion et retrait de contraintes et de variables.

5.1 Utilisation de NETGEN

Considérons une contrainte C_i représentant la relation $x + y = z$. Cette contrainte est représentée dans NETGEN par la fonction suivante:

```
(constraint 'Ci '(+ x y z))
```

De manière interne, NETGEN effectue les opérations suivantes:

```
(creer Ci C+)
(associer (variable x) (terminal a Ci))
(associer (variable y) (terminal b Ci))
(associer (variable z) (terminal c Ci))
```

Nous obtenons un réseau contenant trois variables, une contrainte et des domaines de valeurs vides. Pour attribuer les valeurs aux variables, rappelons qu'il existe trois dégrés de croyance - "constante", "défaut" et

"hypothese". L'attribution des valeurs aux variables est effectuée pas-à-pas pour chaque variable. Supposons $x \in [1,5]; y \in [2,6]; z \in [3,5]$, la valeur de x est donnée par défaut, de y par hypothèse et la valeur de z est une constante :

```
(associer x (defaut      '(1 . 5)))
(associer y (hypothese   '(2 . 6)))
(associer z (constante   '(3 . 5)))
```

Le réseau est chemin-consistant puisqu'il ne contient qu'une seule contrainte. Le filtrage de domaine s'effectue, trouvant que $y \in [2,6]$ n'est pas n-arc consistant avec $x \in [1,5]$ et $z \in [3,5]$. La limite supérieure de $y = 6$ est en contradiction avec celle de $z = 5$. Etant donnée que $z \in [3,5]$ est associé avec le dégré de croyance dit constante, le couple $y \in [2,6]$ est nogood :

```
;| Activer <Y:ADDITION-1> parce que son terminal C prend la valeur [3.5].
;| <Y:ADDITION-1> calcule [2,2] pour son terminal B de A et B.
;| Contradiction apparait dans la fusion de:
;|       ** <ELEM:S1:(Y de UTILISATEUR) , valeur = [2.6]>
;|       ** <ELEM:S2:(Z de UTILISATEUR) , valeur = [3.5]>
;| La valeur [2.6] dans Y semble etre coupable,
;| L'ensemble nogood devrait contenir les couples suivants:
;|       ** <((hypothese-1:y) valeur = [2.6]),
;|       ** <((defaut-1:x)    valeur = [1.5]).
```

Ce couple est inclu dans l'ensemble nogood et est ensuite supprimé du réseau. Le filtrage continue et calcule la valeur $y \in [2,0]$, mais sa limite supérieure est inférieure de sa limite inférieure, ce qui n'est pas logique, une valeur par défaut est donnée pour $y \in [2,2]$. A son tour la limite supérieure de $x \in 5$ est nogood, pour que le réseau soit globalement consistant, x doit être inclue dans $[1,3]$ pour $y \in [2,2]$:

```
;| Retrait de la pr\'emisse <Y=[2.6]>,
;| Activer <X:ADDITION-1> parce que son terminal B prend la valeur [2.2].
;| <X-ADDITION-1> calcule [1.3] pour son terminal A de B et C.
;| Contradiction apparait dans la fusion de:
;|       ** <ELEM:S3:(X de DEFAUT-1   ) , valeur = [1.5]>
;|       ** <ELEM:S4:(Z de UTILISATEUR) , valeur = [3.5]>
```

```
;| La valeur [1.5] dans X semble etre coupable,
;| L'ensemble nogood devrait contenir les couples suivants:
;|       ** <((defaut-1:X) valeur = [1.5])
;| Retrait de la pr\'emisse <X=[1.5]>,
;| Activer <X:ADDITION-1> parce que son terminal A prend la valeur [1.3].
```

L'état du réseau est obtenu par:

```
>(etat? y)
; La valeur [2 . 2] dans y est calculee de maniere suivante:
;     y <- z - x
;          y <- (constante [3 . 5])
;          x <- (defaut    [1 . 3])
```

On constate que le réseau est sous-contraint puis qu'il est impossible de déterminer un intervalle fixe pour x et y, Supposons qu'une nouvelle contrainte soit inserée:

```
(contrainte 'Cj '(* x y k))
(associer k (defaut '(2.4)))
```

Le réseau est chemin consistant puisque tout couple de variables est chemin consistant, le réseau contient deux contraintes et quatre variables. Le filtrage de domaines trouve donc que les couples $(x^u = 3)$ et $y^u = 2$ sont nogoods, les supprime et trouve que $x \in [1.1]$ et $y \in [2.4]$ consistant pour C_i et C_j. Le réseau est globalement consistant et possède une solution. Cette solution est affichée lorsque l'on interroge l'état de y :

```
>(etat? y)
; La valeur [2 . 4] dans y est calculee de maniere suivante:
;     y <- z - x
;          y <- (constante [3 . 5])
;          x <- (defaut    [1 . 1])
;
```

Si l'on supprime cette contrainte, le réseau possède toujours une solution :

```
>(etat? y)
; La valeur [2 . 4] dans y est calculee de maniere suivante:
;     y <- k : x
;         k <- (defaut    [2 . 4])
;         x <- (defaut    [1 . 1])
;
```

Dans la pratique, on peut toujours établir les modèles de contraintes *a priori*. Ces modèles peuvent représenter les exigences structurelles de l'objet à conçevoir. Puis au cours de la conception, on ajoute *a posteriori* les contraintes ou les domaines de valeurs représentant les exigences fonctionnelles.

5.2 Exemple de Calcul des Déperditions Thermiques

On admet comme règle de calcul [CSTB 1986] pour les déperditions thermiques par transmission de chaleur à travers une paroi ;

$$d = \sum_{i=1}^{n}(K_i.A_i) + \sum_{j=1}^{m}(k_j.l_j) \tag{5.2.4}$$

- K_i est le *coefficient de transmission surfacique* du constituant i de la paroi,
- A_i est la *surface intérieure* du constituant i,
- k_j est le *coefficient de transmission linéique* de la liaison j entre deux composants i et $i+1$,
- l_j est la longueur de la liaison j.

Pour une paroi composée d'un seul élément i, les déperditions par transmission de 5.2.4 se calculent par $d = K_iA_i + k_jl_j$. Le modèle de contraintes correspondant est établi par le processus suivant:

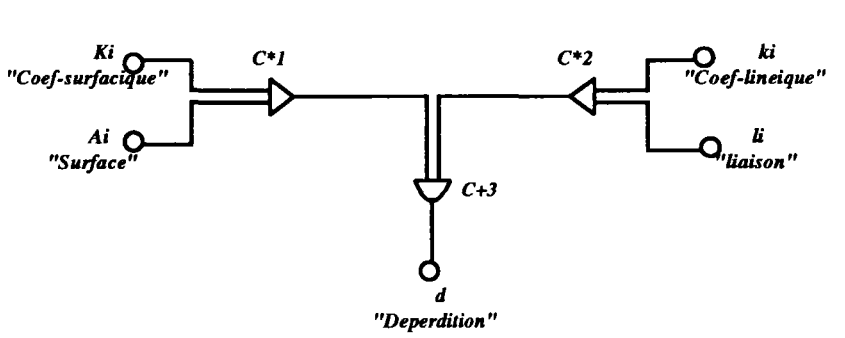


Figure 16: Modèle de Calcul des Déperditions Simple.

1. identification de l'ensemble de paramètres physiques:

   $\mathcal{F} = \{d, K_i, A_i, k_j, l_j\}$,

2. spécification des contraintes entre paramètres:

   $\mathcal{R} = \{\ C^{\times}(K_i, A_i, x),\ C^{\times}(k_j, l_j, y),\ C^{+}(x, y, d))\}$,

3. interprétation en réseau de contraintes (figure 16),

Ce modèle s'écrit en langage de NETGEN comme suit:

```
(defun modele-deperdition ()
      ;;; declaration des variables
          (variable-set  '(deperdition coef-surfacique
                           coef-lineique surface liaison))
          (processor-set '((c1 c*)(c2 c*)(c3 c+)))
      ;;; specification des contraintes
          (associer coef-surfacique (terminal a c1))
          (associer surface         (terminal b c1))
          (associer coef-lineique   (terminal a c2))
          (associer liaison         (terminal b c2))
          (associer (terminal c c1) (terminal a c3))
          (associer (terminal c c2) (terminal b c3))
          (associer deperdition     (terminal c c3))
       )
```

L'exécution de ce modèle crée un réseau de contraintes avec l'ensemble d'étiquettes vides.

Supposons que $K_i = 1.5$, $k_j = 2.3$ sont des valeurs hypothétiques et celles de $A_i = 15.0, l_j = 19.3$ sont données par défaut. On ajoute ces suppositions dans le réseau par la fonction d'association *"associer"* :

```
(associer coef-surfacique (hypothese 1.5))
(associer coef-lineique   (hypothese 2.3))
(associer surface         (defaut 15.0))
(associer liaison         (defaut 19.3))
```

Le générateur propage ces valeurs de manière suivante:

```
;|Activer <C1:C*37> parce que son  terminal A prend la valeur 1.5.
;|Activer <C2:C*38> parce que son  terminal A prend la valeur 2.3.
;|Activer <C1:C*37> parce que son  terminal B prend la valeur 15.0.
;|Activer <C1:C*37> parce que son  terminal C prend la valeur 22.5.
;|Activer <C3:C+25> parce que son  terminal A prend la valeur 22.5.
;|Activer <C2:C*38> parce que son  terminal B prend la valeur 19.3.
;|Activer <C2:C*38> parce que son  terminal C prend la valeur 44.39.
;|Activer <C3:C+25> parce que son  terminal B prend la valeur 44.39.
;|Activer <C3:C+25> parce que son  terminal C prend la valeur 66.89.
EFFECTUE
```

A tout moment on peut interroger l'état de calcul d'une variable. Par exemple, on s'intéresse à la variable d:

```
(etat? deperdition)

;La valeur 66.89 dans deperdition a ete calcule de maniere suivante:
;   DEPERDITION     <- (+ (* COEF-SURFACIQUE SURFACE)
;                         (* COEF-LINEIQUE   LIAISON))
;   COEF-LINEIQUE   <- (SUPPOSITION 2.3)
;   COEF-SURFACIQUE <- (SUPPOSITION 1.5)
;   SURFACE         <- 15.0
;   LIAISON         <- 19.3
OKAY?
```

Pour examiner le processus de filtrage, on utilise la commande *detail?*. Pour la variable déperdition, le filtrage de n-arc s'opère de manière suivante:

```
(detail? deperdition)

;La valeur 66.89 est dans DEPERDITION parce-que:
;      elle est connectee a' (TERMINAL C C3),
;   Et elle est derivee ultimmement de:
;        noeud: SURFACE ,        valeur = 15.0.
;        noeud: COEF-SURFACIQUE, valeur = 1.5.
;        noeud: COEF-LINEIQUE,   valeur = 2.3.
;        noeud: LIAISON,         valeur = 19.3.
;   DEPERDITION     <- (+ (* COEF-SURFACIQUE SURFACE)
;                         (* COEF-LINEIQUE   LIAISON))
```

On constate que la déperdition est de 66.89 $w/\,^{o}C$. Supposons que la valeur souhaitée soit 55.0 $w/\,^{o}C$ et que celle-ci est imposée par une certaine

exigence fonctionnelle. Dans la pratique, il est toujours possible de choisir les coefficients surfaciques et linéiques plus faibles que ceux choisis dans la première hyppothèse. Prenons par exemple $K = 1.3, k = 1.85$.

L'introduction de l'hypothèse $K = 1.3$ dans le réseau provoquera une instanciation $V(K) = 1.3$ qui donne à K un domaine de valeur $d_K = \{1.3, 1.5\}$. Le filtrage d-arc consistant continue et constate que ces deux valeurs sont associées à un même dégré de croyance *hypothèse*. Il existe donc une contradiction entre deux couples $(V(K) = 1.3)$ et $(V(K) = 1.5)$:

```
? (associer coef-surfacique (hypothese 1.3))

;|Contradiction apparait dans la fusion de:
;|          ** <N-5: (COEF-SURFACIQUE de UTILISATEUR), valeur = 1.5>
;|      et  ** <N-22:(PORT            de SUPPOSITION5), valeur = 1.3>.
;|L'ensemble ci-dessous {} est estime d'etre nogood:
;|    * ((TERMINAL PORT SUPPOSITION5) VALUE= 1.3)
;|    * ((TERMINAL PORT SUPPOSITION3) VALUE= 1.5)
;|Il apparait que 1.5 dans N-18 (calculee de la regle SUPPOSITION-RULE) est coupable
```

Etant donné que $K = 1.3$ vient d'être inseré dans le réseau, par défaut, l'ancienne valeur est estimée coupable de cette contradiction. Le couple de valeur $(V(K) = 1.5)$ est supprimé du réseau pour que la consistance de n-arc soit vérifiée. Cette suppression a pour conséquence d'entrainer la suppression des couples de variable-valeurs consistants avec $(V(K) = 1.5)$. Dans ce cas, c'est le produit $(K \times A) = 22.5$ par le filtrage de n-arc et $d = 63.89$ par le filtrage de chemin consistant avec $K = 1.5$:

```
;|Supprimer 1.5 de (TERMINAL PORT SUPPOSITION3).
;|Supprimer 22.5 de  (TERMINAL C C1).
;|   la raison est:  (TERMINAL A C1) ASSOCIER A (TERMINAL PORT SUPPOSITION3).
;|Supprimer 66.89 de (TERMINAL C C3).
;|   la raison est:  (TERMINAL A C3) ASSOCIER A (TERMINAL C C1).
```

Le nouveau couple de variable-valeur $(K, 1.3)$ remplace donc l'ancien couple $(K, 1.5)$. Le filtrage continue et calcule la consistance globale du réseau. Ce qui donne :

```
;|Activer <C1:C*37> parce que son  terminal A prend la valeur 1.3.
;|Activer <C1:C*37> parce que son  terminal C prend la valeur 19.5.
;|Activer <C3:C+25> parce que son  terminal A prend la valeur 19.5.
;|Activer <C3:C+25> parce que son  terminal C prend la valeur 63.89.
EFFECTUE
```

Si l'on s'interroge sur l'état du réseau, la fonction *"etat?"* de la variable déperdition d donne :

```
? (etat? deperdition)

;La valeur 63.89 dans  N-4 a ete calcule de maniere suivante:
;   DEPERDITION     <- (+ (* COEF-SURFACIQUE SURFACE)
  (* COEF-LINEIQUE LIAISON))
;   COEF-LINEIQUE   <- (SUPPOSITION 2.3)
;   COEF-SURFACIQUE <- (SUPPOSITION 1.3)
;   SURFACE         <- 15.0
;   LIAISON         <- 19.3
OKAY?
```

De même, lorsque l'on introduit l'hypothèse $l = 1.85$, une contradiction sera traitée par les filtrages. NETGEN donne ci-dessous la trace des calculs effectués par le générateur :

```
? (associer coef-lineique (hypothese 1.85))

;|Contradiction apparait dans la fusion de:
;|          ** <N-6: (COEF-LINEIQUE de UTILISATEUR) , valeur = 2.3>
;|      et  ** <N-23:(PORT de <SUPPOSITION6>) ,       valeur = 1.85>.
;|L'ensemble ci-dessous {} est estime d'etre nogood:
;|    * ((TERMINAL PORT SUPPOSITION6) VALUE= 1.85)
;|    * ((TERMINAL PORT SUPPOSITION4) VALUE= 2.3)
;|Il apparait que 2.3 dans N-19 (calculee de la regle SUPPOSITION-RULE) est coupable
;|Retrait de la premisse <N-19:(PORT de <SUPPOSITION4>) , valeur = 2.3>.
;|Supprimer 2.3 de (TERMINAL PORT SUPPOSITION4).
;|Supprimer 44.39 de (TERMINAL C C2).
;|   la raison est:  (TERMINAL A C2) ASSOCIER A (TERMINAL PORT SUPPOSITION4).
;|Supprimer 63.89 de (TERMINAL C C3).
;|   la raison est:  (TERMINAL B C3) ASSOCIER A (TERMINAL C C2).
;|Activer <C2:C*38> parce que son  terminal A prend la valeur 1.85.
;|Activer <C2:C*38> parce que son  terminal C prend la valeur 35.705.
;|Activer <C3:C+25> parce que son  terminal B prend la valeur 35.705.
;|Activer <C3:C+25> parce que son  terminal C prend la valeur 55.205.
EFFECTUE
```

L'interrogation sur l'état de calcul de la déperdition donne :

```
;La valeur 55.205 dans  N-4 a ete calcule de maniere suivante:
;   DEPERDITION <- (+ (* COEF-SURFACIQUE SURFACE) (* COEF-LINEIQUE LIAISON))
;   COEF-LINEIQUE <- (SUPPOSITION 1.85)
;   COEF-SURFACIQUE <- (SUPPOSITION 1.3)
;   SURFACE <- 15.0
;   LIAISON <- 19.3
OKAY?
```

La valeur de d se rapproche donc la valeur intialement souhaitée. Si l'exigence fonctionnelle pour la déperdition dans ce cas est définie par un ensemble de valeurs, disons $d_K = \{55, 57, 59, 61\}$, une séquence de filtrages est nécessaire pour trouver les solutions correpondantes à chaque instanciation. Le mécanisme de la reconnaissance de contradiction est identique à l'exemple montré précémment. Ci-dessous est une solution du réseau :

```
?(associer deperdition (parmi '(55 57 59 61)))

;|Contradiction apparait dans la fusion de:
;|          ** <N-4:(DEPERDITION de UTILISATEUR) , valeur = 53.0>
;|      et  ** <N-26:(PORT de <PARMI-VALS4>) ,     valeur = 55.0>.
;|;;; Les premisses suivantes semblent etre fautives:
;|;;;        * (<N-25:(PORT de <PARMI-VALS3>) , valeur = 53.0> DEPERDITION).
;|;;;        * (<N-26:(PORT de <PARMI-VALS4>) , valeur = 55.0>).
;|
;|Retrait de la premisse <N-25:(PORT de <PARMI-VALS3>) , valeur = 53.0>.
;|Supprimer 53.0 de (TERMINAL PORT PARMI-VALS3).
;|Supprimer 33.5 de (TERMINAL B C3).
;|   la raison est:  (TERMINAL C C3) ASSOCIER A (TERMINAL PORT PARMI-VALS3).
;|Supprimer 1.7357514 de (TERMINAL A C2).
;|   la raison est:  (TERMINAL C C2) ASSOCIER A (TERMINAL B C3).
;|Activer <C3:C+25> parce que son  terminal C prend la valeur 55.0.
;|Activer <C2:C*38> parce que son  terminal C prend la valeur 35.5.
;|Activer <C2:C*38> parce que son  terminal A prend la valeur 1.8393784.
;|Activer <C3:C+25> parce que son  terminal B prend la valeur 35.5.
EFFECTUE
```

On constate que le système a ajusté la valeur du coefficient linéique, l'a attribué la valeur $k = 1.84$ à la place de celle donnée précémment ($k = 1.85$) :

```
? (etat? coef-lineique)

;La valeur 1.8393784 dans coef-lineique a ete calcule de maniere suivante:
;   COEF-LINEIQUE   <- (/ (- (PARMI (55.0 57.0 59.0 61.0))
;                            (* COEF-SURFACIQUE SURFACE))
;                         LIAISON)
;   COEF-SURFACIQUE <- (SUPPOSITION 1.3)
;   SURFACE         <- 15.0
;   LIAISON         <- 19.3
OKAY?
```

Ce processus peut être prolongé jusqu'à ce que l'utilisateur soit satisfait. La solution finale du réseau est un ensemble de couples variables-valeurs consistants. Cet ensemble contient donc la description de l'objet

à concevoir. On peut dissocier un réseau en plusiers sous-réseaux, et manipuler ces sous-réseax indépendamment l'un de l'autre. Le couplage final sera effectué par une connexion physique, soit via une contrainte soit directement d'un sous-réseau à un autre, entrainera le filtrage local sur ces connexions, si elles sont consistantes, le réseau globale est consistant.

En effet, les réseaux de contraintes permettent à l'utilisateur d'imposer sa méthode de résolution en lui donnant la liberté dans la formulation et décomposition du problème et dans la manipulation des valeurs, variables et contraintes.

Il mérite de souligner qu'il est extrêmement différent de ce que ce serait la manipulation d'un logiciel de calcul thermique classique telque ceux qui sont commercialisés actuellement. La valeur de déperditions est toujours le résultat d'un calcul, elle ne peut être définie *a priori* parce-qu'elle n'est pas conçu pour, et la logique de calcul ne fait pas pour calculer à partir de cette valeur "à l'envers". En quelque sorte, on replace donc une procédure de "tester-erreur" pour remplacer des variables par un processus direct de déduction. C'est d'une certaine manière le traitement que l'on pourrait appeller de *"problème inverse"*. On peut d'ailleurs, mais ce sera pas dans le cadre de cette thèse, traiter un certain type de problèmes inverses bien connus et qui soit difficile à traiter.

5.3 Conclusion

Nous avons démontré dans ce chapitre la flexibilité et l'adéquation de la méthode de conception par modèle de contraintes. Un élément important dans cette démonstration est le développement de NETGEN, un gestionnaire de contraintes.

Nous avons restreint notre exploration seulement à l'espace des modèles algébriques simples pour démontrer la faisabilité de cette technique. Cependant, la résolution des équations polynomiales et les systèmes linaires de la forme $A.\vec{x} = B$ est en cours de développement pour completer la gamme des équations algébriques. La gestion des systèmes d'équations non-linéaires via les méthodes de Morgan [Morgan 1987] et base de Gröbner [Buchberger 1985] est également en cours.

Il ne reste plus qu'un problème qui nous semble difficile à traiter avec ce gestionnaire : celui de l'étude des variations des paramètres puisque cette analyse nécessite la mémorisation de l'ensemble des calculs pour décrire les états de changements des systèmes physiques.

Nous avons tenté d'introduire la notion de contraintes fonctionnelles pour simplifier les systèmes physiques mais cette tentative ne nous donnait pas de résultat positif. Cette idée consiste à décrire les valeurs qualitativement en termes de régions de croissance ou de décroissance monotone. Les cas les plus importants et les plus communs sont des relations monotones $M+$ qui s'appliquent lorsque la valeur d'une paramètre croît de manière monotone. La contrainte $M-$ s'applique losque sa valeur décroît de manière monotone. Une fonction sur une valeur croît ou décroît de manière monotone si sa dérivée est strictement positive ou négative respectivement.

Ainsi, les contraintes qualitatives permettent de définir des relations abstraites entre les équations de contraintes qualitatives et les équations différentielles ordinaires d'une manière plus précise. Donc, Si un mécanisme peut se décrire par une équation differentielle sous une certaine hypothèse, il existe un ensemble d'équations de contraintes qualitatives correspondant, mais plus faible, pour le même mécanisme. 'Plus faible' parce que tout

comportement satisfaisant l'équation differentielle doit satisfaire les contraintes qualitatives, mais pas nécessairement l'inverse. En d'autres termes, on décompose une équation differentielle ordinaire en un ensemble équivalent d'équations simultanées par l'introduction de termes à chaque sous-expression de l'équation differentielle origine. Enfin, chaque équation peut être associée à une contrainte qualitative.

Cependant, le principe des réseaux de contraintes dynamiques avec gestion de contradiction, explication, d'insertion et de retrait de contraintes ne permet pas de mémoriser l'ensemble des opérations de tous les réseaux. Cette difficulté nous a mené sur le piste de *l'analyse qualitative*. Nous allons décrire cette technique dans le chapitre suivante.

Partie II

Analyse Qualitative des Modèles de Contraintes

"The sciences do not try to explain,
they hardly even try to interpret, they mainly make models.
By a model is meant a mathematical construct which,
with the addition of certain verbal interpretations, describes observed phenomena.
The justification of such a mathematical construct is solely and precisely that it is expected to work."

John Von Neumann
(The Character of the Equations, 1949)

Chapitre 6
Le Raisonnement Qualitatif sur les Modèles Physiques

"La production de variations sur un thème est au coeur de la créativité. Il ne s'agit toutefois pas de quelque processus magique et mystérieux qui se produit lorsque deux concepts indivisibles entre en collision ; bien au contraire, elle est une conséquence de la divisibilité des concepts en éléments sous-conceptuels déjà porteurs de sens."

Douglas Hofstadter (in German Journey)

6.1 Motivations

Le raisonnement qualitatif a pour objectif de déduire des descriptions comportementales d'un monde physique à partir de ses descriptions qualitatives et équationnelles. L'avantage le plus significatif du raisonnement qualitatif est la possibilité d'expliquer le(s) changement(s) dans le comportement du monde physique et la possibilité d'en tirer des conclusions partielles quand la connaissance est incomplète ou incertaine.

Le problème de la "physique" *qualitative* ou plus généralement du "raisonnement" *qualitatif* peut se définir comme un problème de *"mapping"* d'un ensemble de valeurs qualitatives ($\mathcal{X}$)[21] sur un ensemble de paramètres ($\mathcal{P}$) connaissant les contraintes jouant sur ces paramètres ($\mathcal{C}$) pour déduire un ensemble de valeurs qualitatives associées à tous les paramètres $\{\mathcal{X},\mathcal{P}\}$ (cf. définition 1.1).

Au début, La motivation pour la recherche sur le raisonnement qua-

[21]signs de croissances, décroissances, constantes

litatif était liée au developpement des techniques pour la représentation et l'exploitation des connaissances *profondes*[22] ou *physiques*[23]. Ces connaissances venaient compléter celles représentées par des bases de règles qui connaissent des limites sévères dans certaines applications. Depuis, ces approches ont développé leurs propres chemins en Intelligence Artificielle, et en particulier, des méthodes puissantes pour le raisonnement et ont été rapidement appréciées dans la communauté de la recherche en Intelligence Artificielle [deKleer 1984a],[Bobrow 1985].

Plusieurs chercheurs dans les industries chimique et micro-électronique ont introduit des méthodes spécifiques du raisonnement qualitatif dans les systèmes à base de connaissances pour accroître leurs performances. Mohammed et Simmons [Mohammed 1986] ont développé une bibliothèque de processus souvent employés dans la fabrication de semi-conducteurs pour simuler la production des composants électroniques. La simulation leur permet de determiner les paramètres qui peuvent influencer les paramètres mesurés et enfin d'indentifier ou de créer les ensembles de défauts sup connés. Utilisant les mesures paramétriques des composants électroniques, Pan et Tenenbaum [Pan 1986] ont décrit une hiérarchie de relations causales permettant d'identifier des démarches singulières dans une fabrication, susceptibles d'avoir généré des défauts.

Plusieurs autres chercheurs dans les laboratoires de recherche ont développé des méthodes plus générales pour le raisonnement qualitatif sur les systèmes physiques [deKleer 1975], [deKleer 1979], [Hayes 1985], [Forbus 1984], [Kuipers 1985a], [Yip 1987], [Williams 1987], [Struss 1987a], [Struss 1987b],

[22]le terme "deep model" est utilisé pour se distinguer du terme "shallow knowledge" (heuristique) des systèmes à base de règles.

[23]physical model

[Cross 1983], [Falkenhainer 1986]. Les méthodes les plus connues sont celle de De Kleer et Brown [deKleer 1984b] connue sous le nom de *"La Physique Qualitative à base de Confluences"*, celle de Forbus [Forbus 1984] appelée *"La Théorie de Processus Qualitatifs"* et celle de Kuipers [Kuipers 1987b], [Kuipers 1987a], [Kuipers 1985a], [Kuipers 1986], appelée *"La Simulation Qualitative"*. Bien que chacune de ces méthodes ou théories aient une sémantique propre, elles ont la même motivation : A partir d'une description du mécanisme physique, un modèle physique peut être établi. Sur ce modèle, un algorithme s'applique pour déterminer les comportements du système physique sans avoir besoin de connaissances précises sur les valeurs des paramètres et des relations fonctionnelles entre ces paramètres. Cependant, chacun s'appuie sur un aspect différent de l'approche traditionnelle dans l'ingénierie pour analyser les processus physiques.

Dans ce chapitre, nous allons examiner l'état de l'art dans la recherche en cette matière pour identifier un sous-ensemble cohérent avec les méthodes de travail dans la conception en ingénierie. Pour ce faire, il convient de "voir" les deux méthodes de base les plus répandues dans le raisonnement qualitatif : l'approche de la *"Physique Qualitative à base de Confluences"* de de Kleer et Brown (section 6.2) et l'approche de la *"Simulation Qualitative"* de Kuipers (section 6.3).

6.2 Physique Qualitative a base des Confluences

6.2.1 Concepts Généraux

Dans l'approche proposée par de Kleer et Brown [deKleer 1984b], un système physique est décrit en terme de composants et d'interconnexions

entre composants. Chaque comportement de composant est modélisé par au moins une équation distincte. Ainsi, le comportement d'un système est caracterisé par un ensemble d'équations differentielles qualitatives, appelées *confluences*. L'analyse de ce système d'équations, représenté par un ensemble de contraintes entre les équations et la structure physique, résulte en la description qualitative du comportement et la séquence causale expliquant ce comportement du système.

Il importe de noter que dans cette approche, une situation physique est un système, décrit par ses constituants. Trois types de constituants composent un système: Matières, Composants et Conduits. Dans un système, les matières tels que les gaz, la fumeé, l'eau, les électrons, etc, sont transportés d'un composant à un autre par les conduits reliant ces derniers. Chaque composant peut changer la forme et les caractéristiques des matières qu'il "détient". Chaque Conduit est un constituant simple, passif, transportant une matière d'un composant à un autre, et ne peut pas changer en quoique ce soit les aspects de la matière qu'il transporte.

En d'autres termes, la structure physique d'un système peut se représenter par un réseau dans lequel les noeuds représentent les composants et les arcs représentent les conduits.

Chaque objet COMPOSANT est décrit par les attributs nom, type, conduits, terminal où NOM est l'identité du composant, TYPE est le type générique du composant, CONDUITS décrit les noms des conduits auxquels les *terminaux* du composant sont connectés, TERMINAL indique le terminal particulier auquel le conduit est attaché.

Chaque objet CONDUIT peut se représenter par une collection de CONNEXIONS. Chaque connexion se fait entre deux terminaux.

6.2.1.1 Variables Qualitatives

Les variables qualitatives ne peuvent prendre qu'un nombre limité de valeurs. Elles sont soient des nombres réels, soient des symboles représentant un domaine de valeurs dans $\Re$. (La notation "$[x]$" indique la valeur qualitative 'x_q' respectant l'espace de quantités "q".)

Dans cette approche, la propriété la plus importante d'une quantité est sa croissance, sa décroissance ou sa stabilité ce qui correspond au signe de sa derivée $\oplus, \odot, \ominus$ (i.e. dérivée positive, negative, ou nulle).

On note:

$$X_0 = \begin{cases} \oplus & \text{si et seulement si } X > 0 \\ \odot & \text{si et seulement si } X = 0 \\ \ominus & \text{si et seulement si } X < 0 \end{cases}$$

6.2.1.2 Arithmétique Qualitative: Addition et Multiplication

$[X]$	$[Y]$	$[X]+[Y]$	$[X].[Y]$
$\oplus$	$\oplus$	$\oplus$	$\oplus$
$\oplus$	$\odot$	$\oplus$	$\odot$
$\oplus$	$\ominus$	$(\ominus, \odot, \oplus)$	$\ominus$
$\odot$	$\oplus$	$\oplus$	$\odot$
$\odot$	$\odot$	$\odot$	$\odot$
$\odot$	$\ominus$	$\ominus$	$\odot$
$\ominus$	$\oplus$	$(\ominus, \odot, \oplus)$	$\ominus$
$\ominus$	$\odot$	$\ominus$	$\odot$
$\ominus$	$\ominus$	$\ominus$	$\oplus$

Dans cette arithmétique, on note que: $[X.Y] = [X].[Y]$; $[X+Y] \neq [X]+[Y]$. Si l'on veut distinguer X par rapport à une valeur A, il suffit de définir une nouvelle variable $Y = X - A$. Alors $[Y] = \oplus$ signifie $(X > A)$. En effet, le formalisme ' X *croit*' est $[dX/dt] = \oplus$. On note δX pour $[dX/dt]$; $\delta^n X$ pour $[d^n X/dt^n]$. Cependant, $\delta^{n+1} X$ ne peut être obtenue par la différentielle de $\delta^n X$, on doit, dans ce cas, revenir à la définition quantitative $\delta^{n+1} X = [d^{n+1} X/dt^{n+1}]$.

δX et $[X]$ sont indépendants instannément. Si le temps croit, les valeurs de $[x]$ sont guidées par δX (*i.e.* integration). Les lois[24] doivent être invariantes du point de vue du temps, donc toutes les lois doivent s'appliquer à un instant du temps.

Notons que les deux expressions X *croit*, $-X$ *décroit* sont équivalentes: $[X] = -[-X].[d^2X/dt^2] = [X].\delta X = \oplus$ si X s'éloigne du 0, et $[X] \otimes \delta X = \ominus$ si X se rapproche de 0.

6.2.1.3 Valeurs Qualitatives

Soit une variable X, une valeur qualitative V de X est un élement de l'ensemble des intervalles disjoints appartenant au domaine de valeurs quantitatives $\mathcal{D}_X$.

$(V_1 =]-\infty, 0[= \ominus; V_2 = [0,0] = \odot; V_3 =]0, +\infty[= \oplus)$ est un exemple.

Notion de Continuité: Chaque valeur de variables change de manière continue en fonction du temps. Si $[V(t_1)]_q = A_k$ et $[V(t_2)]_q = A_m$, alors pour chaque A_n définie entre A_k et A_m, il existe au moins un t_i tel que $t_1 < t_i < t_2$ et $[V(t_i)]_q = A_n$.

Notion de Valeurs Moyennes: Soit $[X(t_1)] = [X(t_2)]$ avec $t_1 < t_2$, Il existe un t_i telque $dX(t_i) = [X(t_2)] - [X(t_1)]$. Si $[X(t_1)] = [X(t_2)]$ avec $t_1 < t_2$, alors $\delta X(t_i) = \odot$ pour $t_i \leq t_1 \leq t_2$.

Soit $t' = t + dt$, Si $[X(t)] = [X(t')] = C$, où C est une valeur qualitative, alors $\delta X(t) = \odot$.

Ces notions établissent la base pour le raisonnement temporel dans la

[24]les lois décrivant le comportement en fonction du temps nécessitent alors les intégrales explicites dans leurs formulations.

physique qualitative de De Kleer et Brown. Considerons quelques exemples utilisant l'ensemble de valeurs $\{\oplus, \odot, \ominus\}$. Si $[X(t)] = \oplus$ est suivie par $[X(t')] = \ominus$, il existe une contradiction violant le principe de continuité. Dans plusieurs contextes, le théorème de la valeur moyenne peut s'ecrire $[X(t')] = [X(t)] + \delta X(t)$:

si $[X(t)] = \odot$ et $\delta X(t) = \oplus$, alors $[X(T')] = \oplus$.

si $[X(t)] = \oplus$ et $\delta X(t) = \oplus$ ou $\odot$, alors $[X(T')] = \oplus$.

si $[X(t)] = \oplus$ et $\delta X(t) = \ominus$, alors $[X(T')] = \oplus$ ou $\odot$.

6.2.2 Modèles Qualitatifs

La *Confluence* est la première primitive de la modélisation qualitative. Le modèle d'un composant caractérise les comportements potentiels d'un composant. Le comportement physique d'un composant est exprimé par un ensemble d'équations differentielles qualitatives (confluences). Chaque confluence s'exprime par un ensemble de termes dont la somme est une constante. Un *terme* est soit une variable (i.e. un attribut), soit sa négation, ou soit le produit d'un constante et d'une variable. Par exemple, nous considerons la forme de la confluence $\delta P + \delta A - \delta Q = \odot$. Cette confluence consiste en trois termes: $\delta P, \delta A$ et δQ dont la sommation est $\odot$, une constante (zéro).

Un ensemble de valeurs satisfait une confluence ;

1. Si l'égalité qualitative suit strictement le tableau d'arithmétique qualitative (addition et multiplication) précédemment décrit, ou

2. Si la partie gauche de la confluence ne peut être évaluée. Par exemple $\delta P = \oplus, \delta A = \ominus$; la confluence $\delta P + \delta A - \delta Q = \odot$ est

satisfaite parce que $\delta P + \delta A$ ne peut pas être évalué.

Un ensemble de valeurs contradit une confluence si:

1. toutes variables ont des valeurs,
2. la partie gauche s'évalue,
3. la confluence n'est pas satisfaite.

Exemple: $\delta P = \oplus, \delta A = \oplus$ et $\delta Q = \ominus$ est contradictoire quand $\delta P + \delta A - \delta Q = \odot$.

On note que d'après cette définition, une confluence ne peut être satisfaite ni contredite si toutes ses variables n'ont pas reçu de valeurs.

La deuxième primitive de la modélisation qualitative de de Kleer et Brown est *l'état qualitatif*. Les états qualitatifs découpent le comportement d'un composant en différentes regions, chaque region est décrite par un ensemble de confluences. La notion d'états n'est pas nécessairement exprimée dans l'analyse quantitative lorsque une équation mathématique peut modéliser de manière adéquate le comportement d'un composant. Pourtant, il est parfois utile d'introduire l'état dans l'analyse quantitative pour délimiter des regions où certains effets sont négligeables. D'autre part, dans le régime qualitatif, la notion d'états est non négligeable, puisqu'il est souvent impossible de formuler un ensemble de confluences simples qui peut caractériser le comportement d'un composant dans le mécanisme général du système .

Pour chaque état qualitatif, le modèle d'un composant est décrit par les confluences qui génèrent le comportement dans cet état. Ceci correspond aux conditions de quantités dans la théorie du processus qualitatif de Forbus

[Forbus 1984]. La forme générale d'un modèle est décrite par la syntaxe suivante:

```
< ETAT:  [specifications] , <confluences> >
```

Par exemple:

Modèle (Valve)

Ouvert	:	$[A = A_{max}]$,	$[P] = \odot$,	$\delta P = \odot$,
Normal	:	$[0 < A < A_{max}]$,	$[P] = [Q]$,	$\delta P + \delta A - \delta Q = \odot$,
Fermé	:	$[A = 0]$,	$[Q] = \odot$,	$\delta Q = \odot$,

6.2.3 Un Exemple de Modelisation par Approche de Confluences

Prenons un exemple simple pour illustrer la modélisation par confluence: le mouvement du gaz à travers une ouverture.

Sachant que le débit volumique à travers une orifice étant donné par la loi de Cochin:

$$\dot{Q} = cA\sqrt{\frac{2P}{\rho}}, \text{ avec } P > 0$$

- c : le coefficient de décharges de l'orifice,
- A : la surface d'ouverture de l'orifice,
- P : la pression à travers l'orifice,
- ρ : densité massique du fluide, dans ce cas c'est du gaz; $\rho = \frac{P}{rT}$.

La modélisation est la suivante:

1. Pour obtenir les confluences relatives, l'équation differentielle quantitative est nécessaire

$$\frac{d\dot{m}}{dt} = c.\sqrt{\frac{2P}{\rho}}.\frac{dA}{dt} + \frac{cA}{p}.\sqrt{\frac{\rho}{2P}}.\frac{dP}{dt}$$

2. On la transforme en équations qualitatives:

$$\begin{aligned} \delta\dot{m} &= [C].[\sqrt{\frac{2P}{\rho}}].\delta A + [c].[A].[\sqrt{\frac{\rho}{2P}}].\delta P \\ \delta\dot{m} &= [P].\delta A + [A].[P]\delta P \end{aligned}$$

3. Confluences obtenues:

   Si l'orifice est ouvert ($0 < A < A_{max}$ et la pression est toujours positive ($P > 0$), on obtient la confluence $\delta\dot{m} = \delta A + \delta P$,

4. Interprétation: le débit massique ($\dot{m}$) est plus important si la surface de l'orifice est plus grand que la surface de réference A ou quand la pression à travers l'orifice A augmente par rapport à la pression de réference P.

Le processus de transformation introduit les états par (i) l'équation quantitative d'origine peut se baser sur les états (eg. $P > 0$); (ii) les spécifications d'états peuvent s'introduire pour éviter les ambiguités. Par exemple, la transformation $[e_1 + e_2] \Longrightarrow [e_1] + [e_2]$ constitue une perte d'information. Cette information peut se reconstituer : par exemple $[e_1 + e_2]$ peut prendre la valeur $\oplus$ si $x > A$ auquel cas, l'introduction d'un état est nécessaire.[25]

[25]En général, si le modèle physique est de la forme $\sum[f_i]\delta X_i = [f_0]$, on s'interesse à analyser les régions de $[f_i]$ en introduisant les états correspondant à chacune d'elles.
Une autre méthode est de linéariser via l'extension de Taylor, produisant une équation de la forme $\sum g_i.dX/dt = g_0$, ou g_0 représente les contributions de dérivatifs d'ordre supérieurs. Cette équation est alors transformée en une collection d'états correspondant aux régions de g_i, chaque état est décrit par ses confluences. Donc, si le modèle physique d'origine s'est bien comporté, une procédure simple déterminera les états qualitatifs et les confluences du composant type.

6.2.4 La Causalité

L'explication du comportement d'un système peut prendre des formes différentes. Par exemple, le concept *preuve-par-explications* explique le comportement en terme de déductions pas à pas. La séquence causale est une explication particulière, cohérente avec les intuitions humaines. Dans la théorie de Johan de Kleer et Brown [deKleer 1984b], le diagramme d'états englobe la notion classique de causalité et ne dépend pas du choix arbitraire des variables. Chaque changement d'état est attribuable au changement d'une variable spécifique donc les effets ont des causes uniques. L'action est locale lorsque une variable cause un changement d'état dans les composants adjacents. Les états sont ordonnés temporellement donc une cause survient naturellement avant un effet.

L'intuition de base dans la notion de séquences causales de de Kleer [deKleer 1986b] est que le comportement d'un système est produit par l'interaction entre les processeurs individuels. Chaque processeur est programmé avec les modèles de confluences, et il possède une capacité de traitement et de mémoire. L'interconnexion entre composants peut se représenter comme un réseau. Un mécanisme de propagation sur ce réseau peut expliquer la cause d'un effet et la cause d'autres causes pour produire le comportement du système. Ce mécanisme est similaire à celui de propagation des contraintes introduit dans la partie I.

6.3 La Simulation Qualitative

Dans l'approche de Kuipers [Kuipers 1985a] [Kuipers 1986], un modèle physique est souvent simplifié en un système d'équations différentielles. L'étude du comportement du phénomène physique modélisé s'effectue via

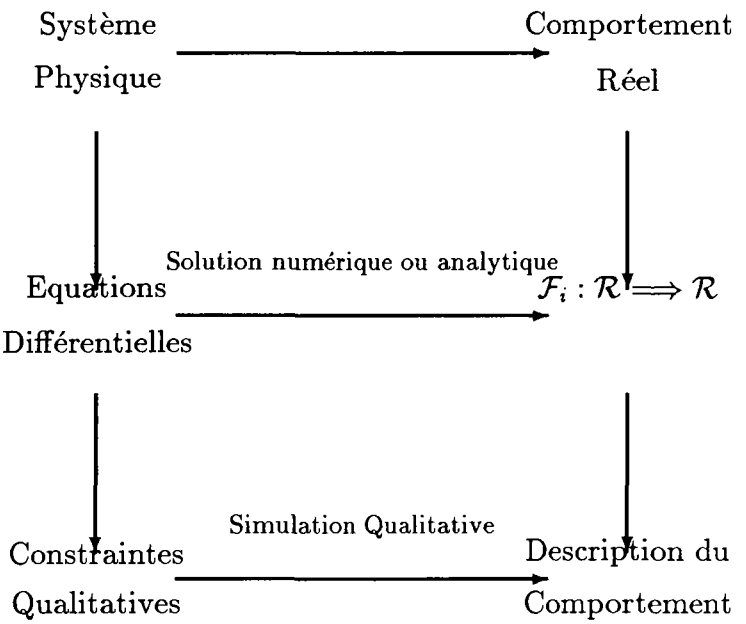


Figure 17: Modelisation des Systèmes Physiques (par Kuipers 86)

la solution, soit analytique soit numérique, du système d'équations représentatif du modèle. L'approche utilisée dans la simulation qualitative est similaire, les modèles qualitatifs sont utilisés pour représenter la réalité physique des systèmes physiques et sont modélisés par les équations différentielles qualitatives. La perte d'information est apparemment inévitable dans la "qualitativisation" des équations descriptives. Donc un modèle qualitatif n'est qu'une abstraction d'un modèle physique traditionelle. Kuipers montre deux niveaux d'abstraction dans la modelisation des systèmes physiques: le niveau quantitatif et le niveau qualitatif. La figure 17 est une représentation graphique des liens entre le monde physique et ces deux niveaux d'abstraction.

Le mécanisme de résolution est identique pour ces deux niveaux:

comme la résolution des systèmes déquations différentielle ordinaire a pour but de fournir des solutions analytiques ou numériques, la résolution des systèmes d'équations différentielles qualitatives a pour but de donner l'abstraction qualitative du comportement du modèle analysé. Dans les modèles qualitatifs, la connaissance incomplète du phénomène peut s'exprimer directement par les contraintes qualitatives, ou par une transformation des équations ordinaires en contraintes qualitatives. L'algorithme de représentation et de simulation de QSIM permet de raisonner sur la description mathématique et de démontrer des théorèmes intéressants sur la validité des prédictions.

6.3.1 Algorithme QSIM

Une description précise de l'algorithme QSIM est donnée dans [Kuipers 1986]. Seuls certains aspects pertinents de cet algorithme seront résumés dans cette section.

Dans l'approche de Kuipers, la simulation qualitative d'un système physique avec QSIM utilise un ensemble de contraintes modélisant la structure d'un phénomène physique et son état initial, QSIM produit un graphe décrivant les états futurs possibles de ce phénomène. En bref, QSIM prédit les variations des paramètres d'un phénomène en appliquant l'ensemble de contraintes qualitatives sur la description de l'état intial du phénomène. Puis QSIM déduit le changement en utilisant la règle de *continuité* identique à celle décrite dans l'approche de de Kleer et Brown. Ce graphe d'états[26] décrit la sucession d'état à partir de l'état initial. Les *comportements possibles* du phénomène sont les chemins dans le graphe, commencant par l'état initial. Chaque comportement représente une solution qualitative cohérente

[26]aussi appelée un *"envisonnement"* dans l'approche de confluences de de Kleer et Brown

au réseau de contraintes qualitatives modélisé.

Le modèle de contraintes consiste en un ensemble de symboles représentant les *paramètres physiques*[27] du phénomène et un ensemble de *contraintes* décrivant les interconnexions entre ces paramètres. Les contraintes sont identiques à celles définies dans les réseaux de contraintes précédement décrits.

Chaque paramètre physique est une fonction du temps, continuément différentiable sur un nombre fini de points critiques. La valeur d'un paramètre à tout instant est spécifiée de manière qualitative. Cette spécification est exprimée en termes d'une application d'un paramètre à son ensemble ordonné de valeurs souvent appelé *"domaine de valeurs significatives"* (landmark values). Les valeurs significatives d'un paramètre incluent zéro, $+\infty, -\infty$ et toutes ses valeurs critiques ou significatives qui sont connues *a priori*. Les valeurs significatives d'un paramètre constituent son *espace de quantités*. Tous les domaines de valeurs significatives sont représentés d'une manière symbolique, à l'exception de zéro. La relation ordinale interne dans un espace de quantités est une propriété essentielle des domaines de valeurs significatives. Une valeur qualitative, ou *qval*, est soit égale à un domaine de valeurs significatives, soit égale à un intervalle ouvert entre les domaines de valeurs adjacents. A l'exécution de la simulation qualitative, l'algorithme QSIM peut découvrir de nouveaux points critiques et ainsi augmente par des nouveaux domaines de valeurs significatives l'espace de quantités. L'*état qualitatif* d'un paramètre est défini par ses relations ordinales avec les domaines de valeurs significatives (ses valeurs qualitatives) et son évolution (direction de variation), nommée *qdir*, qui est

[27] variables physiques qui sont des fonctions de valeurs réeles continuément différentiables

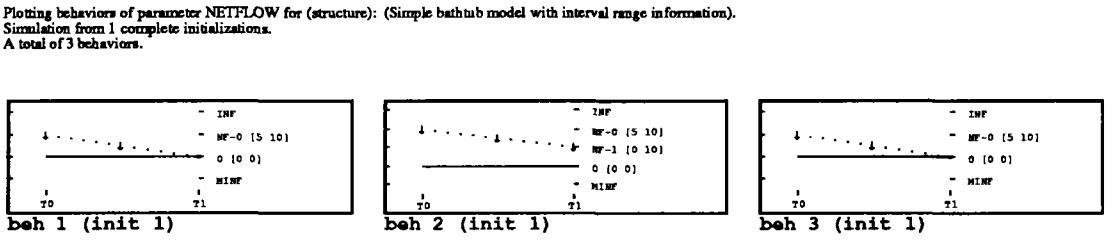


Figure 18: Résultat Graphique de la Variation Qualitative d'un Paramètre.

soit croissante (*inc*), décroissante (*dec*) ou stable (*std*). Par exemple, si la valeur d'un paramètre est positive et croît, l'état qualitatif du paramètre est $((0, \infty)inc)$.

Le temps est représenté comme un ensemble ordonné de *points-de-temps distincts* et symboliques. Tous les points-de-temps sont générés comme résultat d'une simulation. Les comportements simulés sont affichés par les graphiques analogues à celui de la figure 18.

Dans ce graphique, l'axe vertical représente les domaines de valeurs significatives du paramètre et l'axe horizontal représente l'ensemble ordonné de points-de-temps distincts. Les descriptions qualitatives représentant les variations de sa valeur sont affichées au milieu des intervalles et sont représentées par les symboles $(\uparrow), (\downarrow)$ ou (o), indiquant les directions d'évolution (croissantes, décroissantes, ou stables respectivement).

L'état initial d'un phénomène est défini par l'ensemble des valeurs

qualitatives attribuées aux paramètres caractérisant ce phénomène. Un état initial est crée en propageant un ensemble minimum de valeurs à travers le modèle de contraintes, attribuant les valeurs jusqu'alors inconnues aux paramètres par des déductions via les contraintes spécifiées dans le modèle. Cet ensemble minimum de valeurs-variables représente les conditions initiales de l'équation différentielle qualitative. Dans QSIM, Il y a en général autant de conditions générales qu'il y en a dans les équations différentielles ordinaires. Ainsi, pour l'équation qualitative de premier ordre, une valeur initiale d'un paramètre indépendant est spécifiée pour déterminer les comportements qualitatifs possibles. Pour les systèmes d'ordre supérieur, les valeurs initiales sont nécessaires pour la dérivée des paramètres indépendants. Cependant, à cause de la nature qualitative des contraintes, plus d'un état initial peut être cohérent avec les valeurs initiales données.

En déterminant le comportement dynamique d'un système, QSIM utilise une stratégie de proposition et filtrage des ensembles de transitions qualitatives pour simuler tous les états susceptibles à apparaître au prochain point-de-temps (ou de l'intervalle ouvert). Les états incompatibles sont détectés par l'application de la règle de continuité et des contraintes du système. Les filtres globaux indentifieront les états et les qualifieront d'incompatibles, au repos, divergents, ou identiques à un état précédent (dénotant le comportement cyclique) ; les états restants sont encore une fois, soumis à la procédure de proposition-filtrage. Dans certains cas, un état peut générer plus d'un état successif, cohérent donc la simulation produit un arbre d'états. Chaque chemin dans l'arbre constitue un comportement éventuel du phénomène. Cependant, cette inférence est correcte mais incomplète : QSIM peut assurer la prédiction de tous les états logiques éventuels, mais

elle ne peut pas exclure les états qui sont mathématiquement impossibles [Kuipers 1985a, Kuipers 1986]. Ceci est lié à la granularité des informations. Plus d'information permettrait d'éliminer certains de ces états.

6.3.2 La Validité Mathématique des Etats Qualitatifs

Dans QSIM, une équation différentielle qualitative donnée est une abstraction d'une ou plusieurs équations différientielles ordinaires. En particulier, dans le cas où une EDQ peut imposer $y = M^+(x)$, les équations différentielles ordinaires correpondantes pourraient annoncer que $y = f(x)$ pour plusieurs fonctions spécifiques f telles que $F(h) = k.\sqrt{h}$ par exemple. Comme plusieurs EDOs peuvent donner lieu par abstraction à une même EDQ, les prédictions générées par l'EDQ peuvent inclure des comportements impossibles pour la situation particulière modélisée. Ces solutions peuvent correspondre à d'autres EDOs abstraites par les mêmes EDQs.

Etant donné une équation différentielle qualitative et une description de son état au point-de-temps t_0, l'algorithme de QSIM prédit les comportements possibles de l'EDQ sous forme d'une arbre d'états. Un comportement particulier est une séquence d'états constituant un chemin à travers l'arbre, commencant par le racine (état initial) et terminant à une feuille (état final):

$$\textit{Comportement } C = [\textit{état}(t_0),\ \textit{état}(t_0, t_1),\ \textit{état}(t_1),\ \ldots,\ \textit{état}(t_n)].$$

QSIM simule un ensemble d'états possibles et les interprète comme une disjonction de comportements:

$$\text{QSIM: (EDQ,état}(t_0)) \longrightarrow (C_1 \vee C_2 \vee \cdots \vee C_k).$$

On note que cette validité est correcte (sound) mais incomplète: [Kuipers 1986]

- *Correcte:* Si l'EDQ est une abstraction d'une certaine EDO, alors chaque solution à l'EDO correspond à une C_i dans la prédiction de QSIM. Or, si l'EDQ est valide, la prédiction de la séquence de comportements est valide.

- *Incomplète:* Si QSIM prédit les disjonctions ($C_1 \vee C_2 \vee \cdots \vee C_k$.) à partir d'une EDQ donnée, il est toujours possible qu'une disjonction C_i quelconque ne correspond à aucune solution de l'EDO dont l'EDQ est son abstraction. Or, QSIM n'est pas capable de démontrer ce problème. Elle est incomplète.

Dans le chapitre suivant, nous discusterons les détails de ces deux techniques.

Chapitre 7
Discussion sur les Méthodes de Raisonnement Qualitatif

"C'est la nature de l'esprit
qui rend les individus proches, et
les différences de forme, de structure, ou
de manière des atomes matériels
dont les relations complexes constituent l'esprit
sont parfaitement triviales."

Issac Asimov

7.1 Comparaison des Différents Concepts

Bien que les méthodes générales proposées pour le raisonnement qualitatif soient fondées sur différentes approches, à savoir celles à base des *confluences* [deKleer 1984b], à base de *contraintes* [Kuipers 1986] ou à base de *processus* [Forbus 1984], elles présentent certaines similarités. Bredeweg et Wielinga [Bredeweg 1988] ont classifié ces similarités en deux types: ontologiste et activiste. Nous aborderons uniquement les notions fondamentales.

7.1.1 La Notion de Modèles

Un modèle physique représentant un phénomène ou un système physique consiste généralement en cinq ensembles: les paramètres physiques intervenant dans le phénomène; les valeurs de ces paramètres; les relations entre ces paramètres, la structure du phénomène (i.e. couplage entre sous-systèmes); et les éléments du phénomène (i.e. sous-systèmes).

Le tableau 19 recapitule les concepts fondamentaux de la modélisation

Eléments Conceptuels					
Approche	Paramètres	Valeurs	Relations	Structures	Elements
Confluences de de-Kleer et Brown	variés (i.e. pression, débit)	$(\oplus, \odot, \ominus)$	Equations, Inéquations (confluences)	Etats qualitatifs	Composants, Conduits,
Processus Qualitatif de Forbus	Variés (i.e. pression, températures, volume)	Variés (ie. $\oplus, \odot, \ominus$, points-critiques)	Influences, Proportionnelles, Inégalités, Correspondances	Points de vue, processus,	Variés
Simulation Qualitative de Kuipers	Variés (i.e. pression, températures, volume, énergy)	Variés (ie. $\oplus, \odot, \ominus$, espaces de quantités)	Mathématiques, Fonctionnelles, Dérivatives	Néant	Néant

Figure 19: Tableau de Synthèse des Concepts dans Différentes Approches de Modélisation

Approches	Formulation	Bibliothèque	Modèle
de-Kleer et Brown	Oui	Etats Qualitatifs	Topologie du Système
Forbus	Oui	Processus et Vues	Topologie du système
Kuipers	Oui	Néant	Modèle Mathématique

Figure 20: Synthèse des prérequisites dans différentes approches selon Bredeweg (89)

selon les trois approches proposées.

Par ailleurs, la modélisation qualitative peut consister en trois sous-activités: la formulation d'un cadre (framework); l'etablissement d'une bibliothèque; et la construction du modèle physique représentatif du phénomène. Bredeweg [Bredeweg 1990] a structuré d'une manière identique[28] et a comparé les différentes approches. Le tableau 20 recapitulent son analyse.

[28]"formulation of a framework, modeling the contents of a library and modeling the input system"

7.1.2 La Notion d'Etats

Une hypothèse centrale pour le raisonnement qualitatif est la discrétisation du temps. Ce travail ne consiste pas à diviser le temps en intervalles égaux. Les intervalles du temps à considérer sont "les périodes de temps dans laquelle le comportement du système ne varie pas". Ces périodes de temps, appelées comme *états*, sont caracterisées par les variations des paramètres descriptifs d'un système. Cette notion de temps est néanmoins subtile puisque le monde réel peut changer, alors que son comportement peut rester constant du point de vue qualitatif. L'exemple utilisé par Bredeweg et Wielinga est le cas de l'évaporation de liquide: dans la phase de l'évaporation, le liquide est transformé en matière gazeuse. Donc le monde change. Mais dans le raisonnement qualitatif, ce processus est considéré comme constant parce-que aucun des éléments du système "change son apparance qualitative". Ces élements ne changent que lorsque le liquide est completement transformé en gaz.

7.1.3 La Déduction

En ce qui concerne la déductivité de chacune de ces trois approches, du fait qu'elles sont basées essentiellement sur la notion d'état constant, la déduction dans chacune d'entre elles s'appuie sur la prédiction *d'intra-états* et *d'inter-états*. L'activité de prédiction intra-états s'intéresse au raisonnement à l'intérieur d'un état tandis que la prédiction d'inter-états s'intéresse aux transitions d'un état à un autre d'un même système.

7.1.3.1 la déduction intra-états:

Le but de la déduction intra-états est le développement d'une description "complète" d'un état particulier du système. Cette déduction consiste à attribuer les valeurs (qualitatives ou numériques) aux paramètres descriptifs du système. Pour ce fait, deux démarches sont nécessaires:

1. Identifier le modèle mathématique relatif,

2. Sur la base des valeurs connues de certains paramètres, Calculer les valeurs pour les autres paramètres en fonction du modèle mathématique identifié.

L'identification du modèle est différente selon les approches. Dans celle de Kuipers (à base de contraintes), le modèle mathématique est spécifié par l'utilisateur, respectant les règles de formulation par contraintes sur les espaces de quantités. Sur ce modèle, la déduction ne s'applique que pour déduire les valeurs qualitatives. Par contre, dans les approches de Forbus (à base de processus) et de de Kleer et Brown (à base de confluences), l'utilisateur ne spécifie que la topologie du système et la déduction doit a priori s'appliquer pour déterminer le modèle mathématique correspondant. Sur ce modèle, les valeurs qualitatives seront déduites *a posteriori.* Pour ces deux principes, le modèle mathématique est établi en spécifiant les structure du système correspondant à la description de la situation initiale. Chaque structure introduit (entre autre) un groupe de relations entre paramètres. Ensemble, ces relations constituent le modèle descriptif du système ou phénomène. Notons que l'approche des confluences utilise la stratégie *depth-first* pour générer le modèle, tandis que l'approche de processus utilise la stratégie *breadth-first.*

Une fois le modèle mathématique établi, la fonction d'attribution des valeurs aux paramètres est invoquée. Les méthodes utilisées sont la *satisfaction de contraintes* et la procédure de *génerer-et-tester*. Bien que chacune de ces approches ait son propre ensemble de relations et sa propre sématique attachée à ces relations, le processus de déduction est le même. Il consiste à évaluer les relations spécifiées pour déduire les valeurs des paramètres jusqu'alors non-attribuées en fonction des valeurs connues et des relations entre paramètres.

Deux *"problèmes"* peuvent s'introduire pendant l'exécution de ce processus d'attribution de valeurs: *l'ambiguité* et *la contradiction*. L'ambiguité, représente une incertitude sur la valeur d'un paramètre alors que la contradiction correspond à l'attribution de deux valeurs différentes à un même paramètre par deux équations différentes. L'ambiguité est simplement résolue par un rejet de la valeur en considération et la contradiction est résolue par le rejet de l'état dans lequel une contradiction apparaît. Ces mécanismes simples ont fait l'objet de plusieurs discussions. Nous reviendrons sur ce sujet dans la section suivante.

7.1.3.2 La déduction inter-états:

La déduction inter-états consiste à identifier les états sucessifs du système. Un état se termine dès que le comportement du système change et les valeurs nouvellement déduites ne sont plus compatibles avec la description de l'état connu. Pour déterminer quand et comment un état particulier se termine et passe à un autre état, chaque approche utilise un ensemble de règles. Ces règles sont de en trois types:

1. les règles de *terminaison* spécifient les conditions sous lesquelles un état pariculier se termine. Par exemple, au moment où une propriété change, la valeur qualitative courante du paramètre descriptif peut sortir de l'espace de quantités correspondant et doit prendre une autre valeur. L'état courant se termine dèsque cette évènement se produit,

2. les règles de *précedence* spécifient l'ordre dans laquelle les changements se produisent. Dan un état particulier, si plus d'un évènement de terminaison se produit, les règles de précédence s'appliquent pour déterminer l'ordre d'occurence. Par exemple, dans un modèle où deux liquides sont présents. Si la température d'ambiance croît, ces deux liquides vont atteindre leurs températures critiques. L'ambiguité apparaît puisqu'il y a deux transitions d'états qui sont possibles à cause de deux changements de valeurs. Cependant, les règles de précédence s'appliquent si l'on sait quelle température critique est supérieure à l'autre. L'ambiguité peut se résoudre,

3. les règles de *transition* spécifient les conditions sous lesquelles un état successif est valide, respectant la notion de continuité (présentée dans les sections précédentes).

Notons que dans QSIM, les règles de transitions sont des combinations implicites des règles de terminaisons et de transitions.

7.2 Les Limites Connues du Raisonnement Qualitatif

7.2.1 Le problème d'Ordonnancement Causal dans les équations qualitatives

Bien qu'elle ne manipule pas des valeurs qualitatives de paramètres physiques, l'approche de l'ordonnancement causal [Iwasaki 1986a] montre quelques exemples frappants, illustrant les difficultés éprouvées dans l'analyse des sytèmes physiques liée à leurs descriptions mathématiques. Brièvement résumé dans la section concernant la méthode de confluences, cette approche consiste à résoudre un système d'équations silmutanées en déterminant les étapes nécessaires pour calculer les valeurs inconnues. Ce processus s'effectue selon une procédure itérative, examinant les *sous-systèmes complets minimaux* du système d'équations en question[29]. Alors, il est supposé que les inconnues de ces systèmes peuvent être déduites à partir des valeurs connues. Iwasaki et Simon [Iwasaki 1986b] soutiennent que l'ordre et les dépendances de calcul des variables dérivées de ces itérations déterminent les relations causales au sein d'un état du système physique.

Par exemple, étant donné le système d'équations suivant:

$$\begin{cases} a &= a_o \\ b &= b_o \\ a &= c \times b + d \\ d &= b - a \times c \\ e &= c + d \end{cases}$$

[29]Un sous-système complet minimal (minimal complete subsystem) est un sous-ensemble d'équations contenant autant d'inconnues que d'équations, et minimal pour cette propriété, c'est-à-dire n'incluant pas strictement un tel sous-ensemble plus petit. En particulier, chaque équation d'affectation d'une valeur à une variable exogène est un sous-ensemble complet minimal à une équation

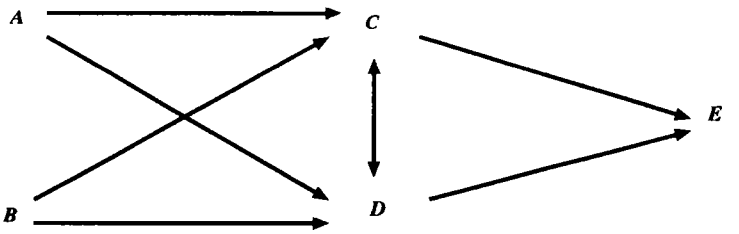


Figure 21: L'Ordonnancement Causal. Exemple 1.

l'algorithme détecte les deux premières équations correspondant à des *variables exogènes*[30], chacune étant un système contenant une équation avec une inconnue. L'étape suivante identifie les deux équations suivantes comme un sous-système complet minimal contenant deux inconnues et dépendant de a et b. Finalement, e est obtenu à partir de c et d, et l'ordonnancement causal s'exprime par le figure 21.

Il est intéressant de se demander si cette notion de causalité cerne bien la notion physique de causalité. Un exemple illustré par Struss [Struss 1987a] démontrerait la difficulté du principe de causalité proposé: Le cas de d'un circuit avec deux résistances R_1 et R_2 alimentées par une puissance V_0.

Deux solutions mathématiques existent en appliquant la loi d'Ohm:

$$\begin{cases} V_1 & = R_1 \times i \\ V_2 & = R_2 \times i \\ R_{serie} & = R_1 + R_2 \\ V_0 & = V_1 + V_2 \end{cases}$$

[30]Les variables exogènes sont les variables du système déterminées par le milieu extérieur, les autres variables s'en déduisant. Cette appellation "procède d'une idée qui vient assez naturellement à l'esprit lorsque l'on réfléchit à la notion de causalité: les causes sont ce sur quoi l'extérieur du système agit; les effets sont ce qui en découle". Les variables exogènes apparaîssent dès lors naturellement comme les "variables causes", les autres (endogènes) étant les "variables effets"[Caplain 1990]

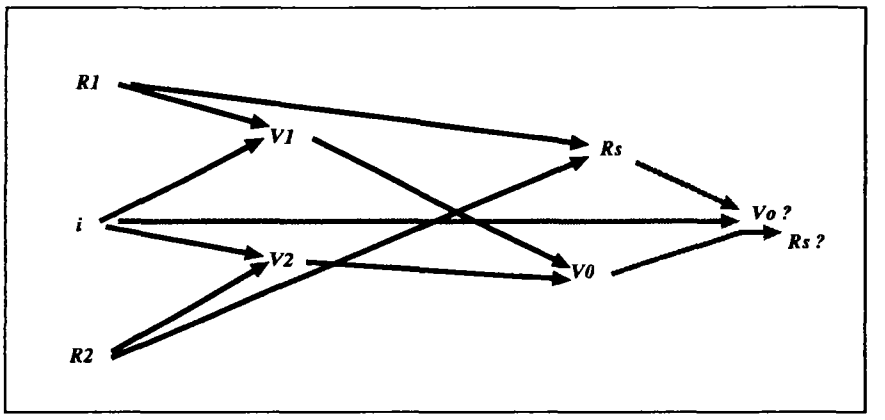


Figure 22: L'Ordonnancement Causal. Exemple de difficulté d'interpretation.

$$\begin{cases} V_1 & = R_1 \times i \\ V_2 & = R_2 \times i \\ R_{serie} & = R_1 + R_2 \\ V_0 & = R_{serie} \times i \end{cases} \tag{7.2.5}$$

On peut constater que dans cet exemple, deux systèmes d'équations ne donnent pas nécessairement le même réseau causal. Les relations "causales" ainsi mise en évidence ne sont donc pas intrinsèques (voir figure 22). L'analyse du deuxième système d'équations est même erronné puisqu'il est illogique de dire que le courrant i en coopération avec les résistances $R_1 R_2$ *causent* les chutes respectives de voltage. On dit alors que le système 7.2.5 donne ce résultat *aphysique* parce que l'équation $V_0 = R_{serie} \times i$ viole le principe de *no-function-in-structure* proposé par de Kleer et Brown dans [deKleer 1984b]. Or, pour surmonter cette difficulté ou bien pour éviter les éventuels résutats erronnés, il est nécessaire d'imposer une certaine condition dans la modélisation du système i.e. le principe de "no-function-in-

structure", par exemple.

Il importe cependant de noter que, dans l'approche de la simulation qualitative à base de contraintes, la notion de causalité physique n'apparait pas essentielle. Pourtant la causalité est le fondement de la propagation de contraintes employée dans cet algorithme. Par ailleurs, la sensibilité de l'ordonnancement causal à la formulation ou reformulation de l'ensemble d'équations devrait attirer l'attention sur un problème aussi essentiel: comment l'algorithme peut vérifier ou analyser la causalité a priori?

7.2.2 Le problème de perte d'information dans l'approche de Confluences

Selon de Kleer et Brown [deKleer 1984b], une information qualitative sur une grandeur physique est une description de la variation de cette grandeur dans un intervalle de nombres réels, délimité par des valeurs critiques (i.e. des seuils de changement de phase). Les "signes" sont proposés comme des symboles représentatifs des variations dans cet intervalle (ou domaine de valeurs). Considerons $\Re$, l'ensemble de nombres réels, une fonction linéaire $y = a(x)$ varie dans $\Re$ et ses informations qualitatives appartiennent à un emsemble d'intervalles $\{[-\infty, 0[, [0, 0],]0, +\infty[\}$. Pour un tel système d'équations, il est nécessaire de définir une algèbre de signes[31].

Considerons les valeurs qualitatives $(\oplus, \odot, \ominus)$ associées à l'algèbre de signes proposée par de Kleer et Brown [deKleer 1984b]. Pour I un intervalle appartenant à un espace quantitatif d'une variable X, on note que:

variable	symbole qualitatif	tendance de dérivation	signe associé
X	$[X]_I$	$\delta X \simeq \nearrow$	$\oplus$
X	$[X]_I$	$\delta X \simeq \rightarrow$	$\odot$
X	$[X]_I$	$\delta X \simeq \searrow$	$\ominus$

[31]pour une analyse plus complète, cette algèbre de signes doit se baser sur une algèbre d'intervalles d'où l'idée d'incorporer les informations quantitatives dans les méthodes qualitative comme une extension de QSIM

Il est clair que pour

$$[X]_I = \oplus \text{ et } [Y]_I = \oplus, [X]_I + [Y]_I = \oplus,$$

mais si

$$[X]_I = \oplus \text{ et } [Y]_I = \ominus, [X]_I + [Y]_I = (\oplus \vee \odot \vee \ominus).$$

Il est donc difficile de déterminer la valeur de $[X]_I + [Y]_I$. Cependant, il importe de noter que si $[X]_I + [Y]_I$ est spécifié comme une confluence, $[X]_I + [Y]_I$ doit être égale à $\odot$, or pour $[X]_I = \oplus$, $[X]_I + [Y]_I$ est toujours égale à $\odot$ quelle que soit la valeur de $[Y]_I$. La perte de l'information est évidente.

Il se pose le problème de comment tirer le maximum d'informations quatitatives du système de confluences que l'on a à traiter. Ainsi le problème de transformation quantitative-qualitative est un indice pour une solution probable. Ce sujet a été traité par plusieurs chercheurs pour trouver un fondement plus "solide" sur la base du *calcul formel.* Peter Struss propose une arithémitique d'intervalle [Struss 1987a] et l'utilisation de la théorie des systèmes dynamiques [Struss 1988] pour analyser la structure topologique des espaces de solution des équations différentielles afin de réduire l'ensemble de valeurs probables ; Jean-Luc Dormoy [Dormoy 1987] utilise la règle de Gauss[32] pour le contrôle des solutions qualitatives et démontre l'équivalence, via une transformation polynomiale en temps, entre le problème de l'existence de solutions d'un système linéaire qualitatif et le problème N-SAT[33]. Jean-Pierre Aubin [Aubin 1989] propose une définition

[32]l'émination Gaussienne pour les systèmes linéaires dans un corps

[33]Rappelons que le problème SAT est le problème de la satisfiabilité d'une conjonction de disjonctions de variables booléennes. Le problème de N-SAT, avec N entier, est la restriction du problème SAT dans laquelle chaque clause contient N variables ou moins. Le problème de N-SAT est NP-complet, dès que $N > 2$

formelle du *cadre qualitatif* sur lequel l'existence de solutions qualitatives "strictes" d'équations linéaires et nonlinéaires sous forme de confluences est démontrée. Gilbert Caplain [Caplain 1990] suggère une recherche sur la combinaison de l'algèbre d'intervalles avec un mécanisme de calcul du genre MACSYMA.

Objectivement, la difficulté dans la "mathématique" qualitative n'est pas complètement résolue et les recherches dans ce domaine tendent vraisemblablement vers une direction qui consiste à utiliser des formulations mathématique lourdes. Ceci semble contredire un des principes de base de la physique qualitative: allèger les calculs formels trop complexes. Qu'en conclure? Que faire?

7.2.3 Le problème de solutions erronnés dans QSIM

Rappelons que l'approche de simulation qualitative [Kuipers 1986] utilise un ensemble ordonné de *domaines de valeurs significatives* (landmark values) ($D = I_1, \cdots, I_n \subset \Re$). Cet ensemble de valeurs sera propagé via les contraintes spécifiées dans le modèle pour en déduire les nouveaux domaines de valeurs. L'analyse qualitative sur les paramètres physiques du système s'ouvre sur leurs *états qualitatifs* (comme décrit dans la section 7.1). En résumé, pour une fonction f continuément différentiable dans le temps et représentant une paramètre physique F, son état qualitatif est défini par un couple *(qval, qdir)*. *qval* est la valeur qualitative de ce paramètre, appartenant à son espace de quantité D, supposons par exemple I_i; *qdir* appartient à l'ensemble de tendances $\{inc, std, dec\}$ équivalent à l'ensemble $\{\oplus, \odot, \ominus\}$ précédement décrit. On le formule comme suit:

$$E_q(f,t) \approx (\text{qval}, \text{qdir}),$$

$$\text{qval} = \begin{cases} I_j & \text{si } f(t) = I_j \subset D \\ (I_j, I_{j+1}) & \text{si } f(t) \in (I_j, I_{j+1}) \end{cases}$$

$$\text{qdir} = \begin{cases} \text{inc} & \text{si } f'(t) > 0 \\ \text{std} & \text{si } f'(t) = 0 \\ \text{dec} & \text{si } f'(t) < 0 \end{cases}$$

En pratique, l'algorithme QSIM utilise 16 règles de transition qui déterminent les états qualitatifs sucessifs d'un état courant. Ces règles matérialisent le *principe de continuité*. Par exemple, considérons un état E_i décrit par $((I_j, I_{j+1}), dec)$. Les états potentiellement sucessifs de E_i est une tuple d'états décrite par { $((I_j, I_{j+1}), dec)$[34] , (I_j, dec), (I_j, std) } mais pas {(I_{j+1}, std) ou (I_j, inc)}.

Il semblerait que l'information prise en compte, pour produire cet ensemble d'états, provient principalement des propriétés (du principe de continuité) d'une seule fonction et ignore les dépendances fonctionnelles entre les paramètres physiques décivant le système. En effet, les relations entre paramètres sont tranformées à un réseau de contraintes. Rappelons que les types de contraintes sont:

type de contrainte	*interprètation*
ADD(f, g, h)	$f(t) + g(t) = h(t)$
MULT(f, g, h)	$f(t) \times g(t) = h(t)$
MINUS(f, g)	$f(t) = -g(t)$
DERIV(f, g)	$f(t) = g'(t)$
$M^+(f, g)$	$f(t) = H(g(t)) \wedge H'(x) > 0$

[34]constant

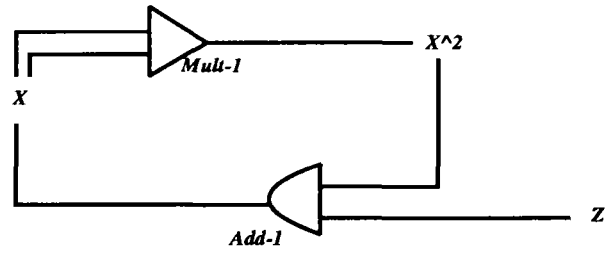


Figure 23: Réseau de contraintes issu de 7.2.6.

Maintenant, considérons un modèle simple:

$$z = x - x^2 \tag{7.2.6}$$

Soient $D_x = \{0, \frac{1}{2}, 1\}$ et $D_z = \{-1, 0, \frac{1}{4}, 1\}$ les domaines de valeurs significatives de x et z respectivement. L'équation 7.2.6 produirerait un réseau de contraintes comme dans le figure 23, incluant la création d'une certaine fonction auxiliaire $y = x \times x$ avec $D_y = \{0, \frac{1}{4}, 1\}$.

Pour $x = 1$, nous obtenons $z = 0$ quantitativement. QSIM produirera l'état global pour un t_0 initial:

$$\begin{aligned} E_q(x, t_0) &= (1, \text{std}) \\ E_q(y, t_0) &= (1, \text{std}) \\ E_q(z, t_0) &= (0, \text{std}) \end{aligned}$$

Ce qui est correct. Mais dès l'application des règles de transition, nous obtenons:

$$\begin{aligned} E_q(x, t_1) &= ((\frac{1}{2}, 1), \text{dec}) \\ E_q(y, t_1) &= ((frac14, 1), \text{dec}) \\ E_q(z, t_1) &= ((-1, 0), \text{dec}) \end{aligned}$$

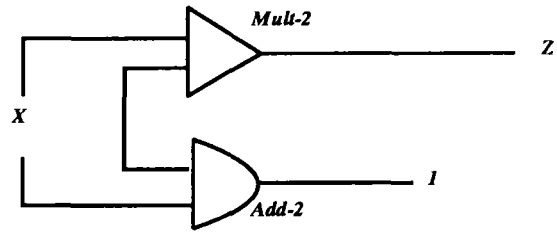


Figure 24: Réseau de contraintes issue de (7.2.7).

comme un état successif parmis les états potentiels. Il apparait dans cet exemple que l'équation 7.2.6 n'est pas prise en considération lorsque les règles de transition sont appliquées : si $x \in (\frac{1}{2}, 1)$, alors $y \in (\frac{1}{4}, 1)$ est cohérent avec la contrainte MULT-1 mais la contrainte ADD-1 ne permet pas de conclure que la somme $x = y + z$ est $\in (\frac{1}{2}, 1)$ avec $z \in (-1, 0)$. Or QSIM donne un résultat erronné puisqu'après 7.2.6, si $0 < x < 1, \Rightarrow 0 < z < \frac{1}{4}$.

Maintenant, si nous reformulons la fonction 7.2.6 par

$$z = x(1 - x) \tag{7.2.7}$$

et dérivons le réseau de contraintes 24. L'état global du 7.2.7 est:

$$\begin{aligned} E_q(x, t_0) &= (1, \text{std}) \\ E_q(w, t_0) &= (0, \text{std}) \\ E_q(z, t_0) &= (0, \text{std}) \end{aligned}$$

Un candidat pour l'état sucesseur est déduit de la contrainte MULT-2:

$$\begin{aligned} E_q(x, t_1) &= ((\frac{1}{2}, 1), \text{dec}) \\ E_q(w, t_1) &= ((0, \frac{1}{2}), \text{inc}) \\ E_q(z, t_1) &= ((-1, 0), \text{dec}) \end{aligned}$$

On peut constater que cette solution est bien meilleure. En d'autres termes, ceci est un problème identique au problème que nous avons relevé dans la sous-section sur l'ordonnancement causal: différentes expressions équivalentes peuvent donner des solutions différentes. Comment le resoudre ? Il importe de noter que dans la formulation des modèles EDQ, les paramètres indépendants peuvent être spécifiés. Une attention particulière sur cette spécification peut permettre d'éviter des solutions potentiellement erronnées.

7.3 Choix d'une Méthode de Raisonnement Qualitatif

Nous avons vu que de Kleer et Brown [deKleer 1984b] s'appuie sur la modélisation des composant physiques individuels et sur la dérivation de comportements en utilisant les connexions entre composants pour contraindre le comportement du système global. Williams [Williams 1984], [Williams 1986] [Williams 1987] [dK 1987] a contribué à cette approche en proposant des modèles génériques applicables dans le domaine de l'ingénierie de l'électronique.

Par ailleurs, Forbus [Forbus 1981a] [Forbus 1981b] [Forbus 1983] [Forbus 1984] [Forbus 1987] s'est concentré sur la modélisation de processus physiques. Ces derniers s'activent une fois les conditions appropriées existent et que les objects physiques (individus) sont présents. Les modèles de contraintes sont alors dérivés de l'ensemble de processus activés pour déduire le comportement global.

Dans les deux approches de De Kleer et Brown et de Forbus, surtout dans le concept de bibliothèques des abstractions physiques, quelques si-

militudes peuvent être avancées : l'abstraction de composants et leurs confluences de de Kleer et Brown sont identiques au concept de processus et leurs influences dans l'approche de Forbus. Ces deux méthodes nécessitent seulement la capacité de construire un modèle physique à partir de ces abstractions. La connaissance du domaine physique est d'ores et déjà inclue dans ces abstractions.

Par contre, la modélisation dans l'approche de la simulation qualitative proposée par Kuipers [Kuipers 1985a] consiste en la spécification d'un modèle de contraintes d'un phénomène ou d'un processus physique par une représentation qualitative des relations mathématiques i.e. l'addition, la multiplication, et le differentiel. Dans ce sens, un modèle de contraintes est un *réseau de contraintes* identique à celui que nous avons utilisé dans la déduction algébrique : les paramètres physiques sont connectés via des contraintes algébriques. La structure résultante représente une version qualitative des équations différentielles ordinaires et est appelée une *structure "d'équations différentielles qualitatives"* modelisant le phénomène physique. L'algorithme de QSIM [Kuipers 1986] s'applique pour reviser les formulations et établir une correspondance formelle entre les équations différentielles qualitatives et ordinaires. Dans cette approche, les connaissances du domaine physique sont représentées par les relations mathématiques comme dans la modélisation analytique conventionnelle. Par conséquent, l'utilisateur est libre de fournir la connaissance du domaine physique par lui-même, sans faire appel à des spécialistes de l'ingénierie de la connaissance comme dans l'approche des systèmes à base de règles.

Il importe de noter que, sur les approches citées, récemment, plusieurs chercheurs ont développé des outils spécifiques dans différents domaines

d'application du raisonnement qualitatif. Oyeleye et Kramer [Oyeleye 1988] ont combiné l'utilisation des graphes orientés avec les confluences de de Kleer et Brown pour analyser qualitativement les effets des états stables (steady-states) dans les usines de production chimique. Sacks [Sacks 1985] [Sacks 1988] a développé un programme pour l'analyse des transitions de phases dans les systèmes non-linéaires décrits par les équations differentielle ordinaires utilisant les abstractions linéaires individuelles (piece-wise linear). Mavrovouniopotis et Stephanopoulos [Mavrovouniotis 1988] ont présenté une analyse formelle du raisonnement avec les ordres de grandeurs et ont appliqué leur approche pour le raisonnement qualitatif sur les phénomènes dans l'ingénierie chimique et biochimique. Williams [Williams 1988] a développé une algèbre qualitative symbolique pour le raisonnement sur les équations algébriques pour surmonter les difficultés que représente l'arithmétique de confluences dans l'approche de de Kleer et Brown. Une autre difficulté provenant de la technique de propagation locale dans la déduction sur les connexions entre confluences a amené Iwasaki et Simon [Iwasaki 1986a] à proposer une technique de l'ordonnement causale pour déduire la causalité à partir un modèle mathématique. Porté [Porte 1988] sur cette approche a proposé un algorithme d'ordonnancement causal, et récemment, Iwasaki et Bhandari [Iwasaki 1988] présentent une approche pour abstraire les systèmes physiques avec un grand nombre de variables en un ensemble de sous-systèmes où chacun peut être traité séparement. Dans cette approche, le comportement du système peut se décrire en termes des comportements des sous-systèmes. La dernière proposition en date est celle de Weld [Weld 1986][Weld 1988], dans sa thèse de Doctorat au MIT entité *"Analyse Comparative"*, Weld propose deux techniques pour détermi-

ner comment le comportement d'un système change lorsque l'on change sa structure. Le point particulier dans cette approche est l'analyse comparative sans avoir besoin de simuler aucune structure du modèle (i.e. l'analyse de perturbations quatitatives).

Vu les difficultés apparues dans l'analyse des variations des paramètres physiques avec les modèles de réseaux de contraintes algébriques, nous cherchons à savoir laquelle de ces méthodes est la plus adéquate dans notre démarche d'analyse des variations qualitatives dans les modèles physiques en ingénierie. Nous avons préferé l'approche de Kuipers puisqu'elle représente un avantage *pratique* : QSIM utilise une représentation par contraintes pour modeliser les systèmes. En effet, l'approche de confluences est inadéquate puisque les composants constituant les phénomènes physiques sont trop complexes pour respecter la notion de *no-function-in-structure* et les états sont parfois trop complexe pour être spécifiés d'une manière adéquate. L'approche de la théorie de processus qualitatif pourrait être plus adéquate avec son langage descriptif, une variété de processus peut être définie pour la conception en ingénierie comme par exemple les processus de transfert thermique par conduction et par convection sont prédéfinis. Cependant, les phénomènes plus complexes comme la diffusion de l'humidité, les écoulements naturels par différents champs de pressions sont apparemment moins bien définies. En outre, la théorie de processus qualitative nécessite une charge substantielle de logiciels, un Système de Maintenance de Vérité à base d'Hypothèses (ATMS)[35] [Forbus 1988] et est souvent limitée par les besoins en terme de mémoire de machine [Falkenhainer 1988]. Dans [Bonissone 1985] et [Bylander 1988], on peut trouver une description plus

[35]Assumption-based Truth Maintenance System

complète sur les points forts et faibles de ces approches.

Alternativement, les autres techniques du raisonnement qualitatif[36] peuvent être utilisées dans l'analyse des phénomènes physiques avec les connaissances incomplètes, mais elles ont également certaines inconvénients. Les techniques proposées par Oyeleye et Kramer [Oyeleye 1988] et Williams [Williams 1988] sont limitées dans le spectre de l'analyse algébrique et des états fixes. La technique de Weld [Weld 1988] est seulement adéquate pour déterminer l'état ultime d'un système sujet à un changement. Les travaux de Sacks [Sacks 1988] nécessitent une connaissance complète des valeurs des paramètres et leurs relations fonctionnelles. Le travail de Iwasaki et Simon [Iwasaki 1986a] ne peut réellement établir une étude de variation par simulation et a été fortement critiqué par l'applicabilité de leur ordonnancement causal. Le travail de Porté [Porte 1988] est seulement applicable aux systèmes linéaires. De même, la technique proposée par Iwasaki et Bhandari [Iwasaki 1988] est également basée sur une représentation de systèmes linéaires qui ne permettent pas de représenter certains modèles physiques.

Dans cette partie de notre travail, l'approche de la Simultation Qualitative (QSIM) [Kuipers 1986] est utilisée pour l'analyse des variations qualitatives des paramètres physiques. Les arguments pour ce choix sont:

- la similarité entre QSIM et NETGEN dans la modélisation par contraintes;
- la puissance de QSIM puisqu'il permet de modéliser les systèmes

[36]Au moment que ces lignes sont redigées (7/90), Bredeweg et Berliner de l'Université d'Amsterdam signalent l'achèvement de leur développpement d'un outil qu'ils appellent GARP, destiné à la modélisation qualitative des systèmes physiques. Nous regrettons de n'avoir pas eu le temps pour évaluer ce système.

dynamiques à toutes complexités (ou ordres de complexités);

- la clareté, la flexibilité de QSIM dans l'utilisation;
- l'extensibilité de QSIM (une version Q2 de QSIM est en cours).

En effet, l'adéquation de l'approche de la simulation qualitative est illustrée par d'une part la représentation de connaissances incomplètes des phénomènes physiques en termes d'équations différentielles qualitatives et d'autre part, sa capacité de saisir toutes les descriptions dynamiques [37] possibles du phénomène. En outre, les récentes extensions de l'agorithme QSIM, dans le but d'incorporer les connaissances partielles concernant les informations quantitatives dans les modèles qualitatifs, sont très attrayantes puisque la connaissance quantitative peut "assister" le mécanisme d'inférences dans QSIM et ainsi réduire le nombre de prédictions générées et rendre les résultats plus représentatifs.

L'outil QSIM utilisé dans cette thèse[38] représente l'état de l'algorithme de simulation qualitative publié en Janvier 1990. Notons que depuis la première version de QSIM en 1986, plusieurs extensions ont été portées à l'algorihme de base. Le travail de Kuipers et Kassirer [Kuipers 1987c] portant sur les modèles complexes était limité à (la dynamique) des relations de première ordre. Les extensions aux ordres supérieurs sont présentées en 1988 par Lee et Kuipers [Kuipers 1988], puis par Kuipers et Dvorak [Kuipers 1989] Ces extensions incluent les techniques tel que filtrage du second ordre, filtrage de transitions de phases.

[37] Le travail de Paddy Toal de Queen Mary College (UK) doit aussi être pris en compte. Notamment sur son analyse de l'introduction des hypothèses dans le modèle analytiques de simulation. Cependant, sa thèse n'est pas encore publiée.

[38] Nous remercions vivement le Professeur Benjamin Kuipers pour nous avoir fourni les codes sources de QSIM, écrit en Common-Lisp et exécutable sur les machines Symbolics 36xx utilisant le système d'exploitation Genera 7.0. Une version de QSIM sur MacIntosh II avec Allegro Common-Lisp a été implémentée dans le cadre de cette thèse.

Nous trouvons que QSIM est un outil adéquat pour le raisonnement sur les variations de paramètres physiques dans les modèles en ingénierie. Cependant comment QSIM peut être appliqué dans la conception ?

Nous allons montrer un exemple de simulation avec QSIM dans le chapitre suivant.

Chapitre 8
Simulation Qualitative des Modèles

"Connaître la cause d'un phénomène,
c'est savoir, tout au moins en théorie,
comment changer ou commander certains aspects
de certaines entités sans affecter tout le reste."
Marvin Minsky (The Society of Mind)

Plusieurs exemples de modélisation et de simulation avec QSIM sont détaillés dans [Molle 1989]. Nous montrons dans ce chapitre la modélisation et la simulation d'un modèle de désenfumage d'un local incendié.

Dans une démarche de modélisation, on procède de manière progressive, et en générale, la démarche du physicien modélisateur est alors de vérifier si le comportement du modèle (e.g. comportements asymtomiques, signes des tendances de croissances, sensibilités, etc..) par une certaine variation est correcte. Ce comportement qualitatif est la représentation mentale de l'observation du problème.

Identiquement à la démarche de conception par la manipulation des contraintes algébriques[39], la démarche par la simulation qualitative[40] permet d'approcher par étapes la modélisation :

1. Introduire un modèle partiel[41] pour l'étude des comportements.

   Cette étude est avantageuse dès que la connaissance du système

[39]du type INCRÉMENTAL, cf. section 2.1
[40]du type GÉNÉRATION-VÉRIFICATION
[41]modèle sous-contraint

est incomplète, i.e. il n'est pas nécessaire de spécifier tous les détails des relations. La simulation d'un tel modèle a pour but d'aider le concepteur à identifier les variations qualitatives des valeurs de certains paramètres. Notons que QSIM produit tous les comportements qui satisfont les contraintes et les règles de transitions de QSIM, même les comportements "physiquement impossibles".

2. Les comportements "impossibles" sont éliminés par l'insertion de nouvelles contraintes. L'insertion de contraintes a un double effet dans un modèle qualitatif :

   (a) changer le modèle dans le but de rapprocher l'objectif de la conception.

   (b) raffiner les contraintes et/ou les valeurs incomplètes.

3. analyser les comportements et reformuler le modèle récursivement de l'étape (1) jusqu'à ce que l'objectif de conception du modèle soit atteint.

8.1 Un Exemple : Le Modèle de Désenfumage d'un Local Incendié

Le désenfumage dans un bâtiment est un composant des systèmes de protection d'incendie. Il existe deux principes communement utilisés: le premier est d'introduire de l'air neuf dans la zône à protéger et d'évacuer les produits de combustion de façon à rendre praticable cette zône; le second est de réaliser une hiérarchie des pressions autour du local sinistré de façon à contrôler la propagation des produits de combustion.

Considérons un local incendié qui communique avec un autre local via une porte, illustré par la figure 25. Considérons le premier principe de désenfumage : l'air neuf est introduit par convection naturelle entre deux locaux à travers la porte et les fumées sont évacuées par extraction mécanique.

Le problème de conception de la protection par désenfumage d'un local incendié se pose donc dans les termes :

- dimensionner un débit de ventilation mécanique pour modifier les écoulements naturels dans la (les) porte(s) de communication avec les locaux voisins et obtenir que la fumée ne sorte pas du local,
- dimensionner ce même débit pour augmenter le plus possibe la hauteur libre entre le sol du local et la nappe de fumée,

On peut formaliser ce problème par le modèle simplifié illustré par la figure 25.

Avec :

- T_a : température de l'air,
- ρ_a : masse volumique de l'air à température T_a,
- T_g : température des gaz chauds,
- ρ_g : masse volumique des gaz à température T_g,
- m_a : masse d'air en zône "libre" dans le local sous la fumée,
- m_g : masse de fumée en zône haute,

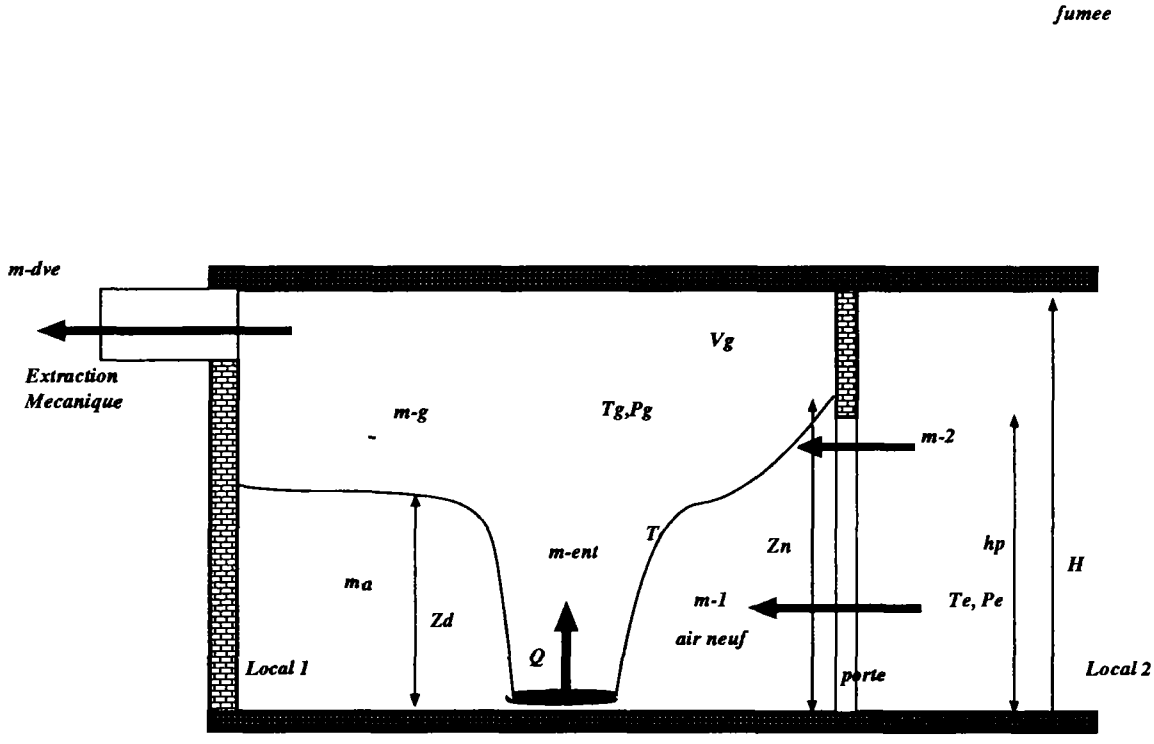


Figure 25: Un Modèle de Désenfumage par Extraction Mécanique

- $\dot{m}_1$: le débit d'air à température T_a entrant dans la zône "libre" en provenance de la pièce à protéger,
- $\dot{m}_2$: le débit d'air à température T_a entrant dans les gaz chauds,
- $\dot{m}_{ent}$: le débit du panache de fumée induit par le feu dans la zône de gaz chaud,
- $\dot{m}_{dve}$: le débit d'extraction mécanique,
- $\dot{Q}$: le débit calorifique du feu,
- H : hauteur du local,
- h_p : hauteur de la porte
- l : largeur de la porte,

- S : surface du local incendié,
- Z_D : hauteur libre sous la nappe de fumée,
- Z_N : hauteur de la "zône neutre" entre le local sinistré et le local à protéger.[42]
- ΔP : différence de pression dans la porte entre les deux locaux,
- C,C' : constantes de débit à travers la porte.

Avec quelques hypothèses simplicatrices liées en particulier au fait que la température des gaz chauds atteint facilement 800 à 1000 K, et donc leur masse volumique est de l'ordre du tier de celle de l'air ambiant, on représente le modèle par :

- l'équation de conservation de la masse en zône basse :

$$\frac{dm_a}{dt} = \dot{m}_1 - \dot{m}_{ent} \tag{8.1.8}$$

- l'équation de conservation de la masse en zône gazeuse g :

$$\frac{dm_g}{dt} = \dot{m}_{ent} + \dot{m}_2 - \dot{m}_{dve} \tag{8.1.9}$$

Avec :

$$\begin{aligned}
\Delta P &= \rho_a.g.(Z_N - Z_D) \text{ pour } Z_D \leq Z_N \\
\dot{m}_1 &= C.l.Z_D.\sqrt{2.\rho_a.\Delta P} \\
\dot{m}_2 &= C'.l.\rho_a.g^{\frac{1}{2}}.(Z_N - Z_D)^{\frac{3}{2}} \\
\dot{m}_{ent} &= f(\dot{Q}).Z_D^{\frac{5}{2}}
\end{aligned}$$

[42] A ce niveau Z_N, les pressions hydrostatiques absolues sont égales dans les deux locaux.

$$
\begin{aligned}
m_a &= S.\rho_a.Z_D \\
m_g &= S.\rho_g.(H - Z_D) \\
T_g &= \frac{T_a.(C.\dot{m}_{ent} + \dot{m}_2) + \dot{Q}}{\dot{m}_{dve}}
\end{aligned}
$$

Et la contrainte structurelle $h_p \leq Z_N$; les contraintes évidentes $Z_D \geq 0$ et $Z_D \leq H$.

Les hypothèses de simplifications retenues rendent inutile l'écriture du bilan d'énergie, donc le calcul de la variation de température des gaz chauds.

8.2 Modèle Qualitatif

Rappelons que dans l'approche de QSIM, un modèle qualitatif est décrit à partir des équation différentielles ordinaires. La structure d'un modèle qualitatif est représenté par le macro *"define-QDE"*, définissant un ensemble de clauses[43]. Les clauses élémentaires sont les suivants :

- la clause *"espace de quantités"* décrit l'ensemble de paramètres contenus dans le modèle qualitatif et leurs domaines de valeurs significatives correspondants. Notons que la spécification de ces domaines n'est pas toujours indispensable ; QSIM est capable de trouver les domaines correspondants à partir d'un sous-ensemble de base (minf 0 inf) associé à chaque paramètre. Cependant, les valeurs symboliques représentant les valeurs significatives d'un paramètre peuvent être exprimées comme les valeurs de seuil connues *a priori* ;

[43]Les détails des macros et des fonctions d'utilisateur de QSIM sont décrites dans [Farquhar 1990]

- la clause *"contraintes"* décrit les relations qualitatives entre les paramètres. Notons que les contraintes exprimées dans les équations différentielles ordinaires peuvent être converties directement en contraintes qualitatives. Cependant, plusieurs relations peuvent être simplifiées en relations fonctionelles, i.e. M^+ ou M^- [44] si la connaissance sur ces relations est incomplète ou bien si l'utilisateur estime qu'il n'est pas nécessaire de spécifier tous les détails.

- la clause *"transition"* est spécifiée si pour un certain paramètre, les contraintes décrites dans la clause *contraintes* ne peuvent être appliquées que sur un sous-ensemble de valeurs de seuil, dépassant ces seuils, d'autres contraintes sont nécessaires. Dans ce cas, une autre structure QDE est requise.

- la clause *"indépendants"* décrit un ensemble de paramètres qui sont considérés comme les paramètres "exogènes" (cf. section 7.2).

- la clause *"histoire"* inclut l'ensemble de paramètres "endogènes"(cf. section 7.2).

- les clauses *"layouts"* et *"print-names"* sont spécifiées pour les besoins de l'affichage des résultats de la simulation.

La conversion des équations 8.1.8 et 8.1.9 en contraintes qualitatives donne la structure de l'équation différentielle qualitative illustrée par la figure 26. Le réseau de dépendances est illustré par la figure 27.

[44]Relations de croissance monotonie et décroissance monotonie

```
(define-QDE desen-local
   (text "Modele de desenfumage d'un local avec extracteur mecanique.")
   (quantity-spaces
    (m-a    (0 ma+ ma* inf))
    (m-g    (0 mg+ mg* inf))
    (m-1 (0 m1+ m1* inf))
    (m-2 (0 m2+ m2* inf))
    (m-dve (0 mdve+ mdve* inf))
    (m-ent (0 ment+ ment* inf))
    (Q      (0 Q+ Q* inf))
    (fQ     (0 fQ+ fQ* inf))
    (Z-n    (0 Zn+ Zn* inf))
    (Z-d    (0 Zd+ Zd* inf))
    (H      (0 H+ H* inf))
    (rho-a (0 pa+ pa* inf))
    (rho-g (0 pg+ pg* inf))
    (Ta     (0 ta+ ta* inf))
    (Tg     (0 tg+ tg* inf))
    (d-ma       (minf 0 inf))
    (d-mg       (minf 0 inf))
      ;;; variables intermediaires
    (d-Z1       (0 dZ1+ dZ1* inf))
    (d-Z2       (0 dZ2+ dZ2* inf))
    (d-Z1>      (0 inf))
    (m-ent+m-2 (0 inf))
    (ta[m-ent+m-2] (0 inf))
    (tg.m-dve       (0 inf))
    )
   (constraints
    ((add d-ma m-1 m-ent))
    ((add m-ent m-2 m-ent+m-2))
    ((add d-mg m-dve m-ent+m-2))
    ;;; contraintes liees au m-a:
    ((mult rho-a Z-d m-a) (0 0 0) (pa+ Zd+ ma+)(pa* Zd* ma*) (inf inf inf))
    ((M- Ta rho-a) (ta* pa+) (ta+ pa*))
    ;;; contraintes liees au m-1:
    ((M+ Z-d m-1) (0 0) (Zd+ m1+) (Zd* m1*) (inf inf))
    ((add d-Z1 Z-d Z-n) (dZ1* Zd+ Zn+) (dZ1+ Zd* Zn+) (dZ1* Zd+ Zn*) (dZ1+ Zd* Zn*))
    ((add d-Z2 Z-d H) (dZ2* Zd+ H+) (dZ2* Zd+ H*) (dZ2+ Zd* H+) (dZ2+ Zd* H*))
    ;;; contraintes liees au m-ent:
    ((mult fQ Z-d m-ent))
    ((M+ Q fQ) (0 0) (Q+ fQ+)(Q* fQ*)(inf inf))
    ;;; contraintes liees au m-2:
    ((mult d-Z1 d-Z1 d-Z1>))
    ((M+ d-Z1> m-2) (0 0) (inf inf))
    ;;; contraintes liees au m-g:
    ((mult rho-g d-Z2 m-g) (0 0 0) (pg+ dZ2+ mg+) (pg* dZ2* mg*) (inf inf inf))
    ((M- Tg rho-g) (Tg* pg+)(Tg+ pg*))
    ;;; equation de bilan de chaleur
    ((mult ta m-ent+m-2 ta[m-ent+m-2]))
    ((mult tg m-dve tg.m-dve))
    ((add ta[m-ent+m-2] Q tg.m-dve))
    ;;; variations
    ((d/dt m-a d-ma))
    ((d/dt m-g d-mg))
    ))
```

Figure 26: Modèle qualitatif en QSIM du modèle de désenfumage local

On remarque que dans ce modèle qualitatif, les contraintes fonctionnelles $M+$ ont été utilisées pour décrire les relations entre certaines variables qui ont des valeurs constantes (e.g. les débits massiques) et les volumes de l'air ou du gaz entrainés ou évacués. En d'autre termes, les masses de l'air et des gaz dans le local est connu comme une fonction qui croit de manière monotone avec les valeurs des côtes Z_D et Z_N.

En résumé, le modèle numérique représenté par les équations 8.1.8, 8.1.9 est transformé en modèle qualitatif comportant les équations qualitatives suivantes :

- conservation de masse en zône basse :

$$\frac{d}{dt}m_a = \dot{m}_1 - \dot{m}_{ent} \tag{8.2.10}$$

- conservation de masse en zône haute :

$$\frac{d}{dt}m_g = \dot{m}_2 + \dot{m}_{ent} - \dot{m}_{dve} \tag{8.2.11}$$

- transformation quantitative-qualitative sur $\dot{m}_1$:

$$\dot{m}_1 = ClZ_D.\sqrt{2\rho_a\Delta P} = k.(Z_D^{\frac{3}{2}}) \simeq M^+(Z_D) \tag{8.2.12}$$

- transformation quantitative-qualitative sur $\dot{m}_2$:

$$\dot{m}_2 = C'l\rho_a g^{1/2}(Z_N - Z_D)^{\frac{3}{2}} = \alpha.(Z_N - Z_D)^{\frac{3}{2}}) \simeq M^+(Z_N - Z_D)^2 \tag{8.2.13}$$

- transformation quantitative-qualitative sur $\dot{m}_{ent}$:

$$\dot{m}_{ent} = f(\dot{Q}).Z_D^{5/2} \simeq M^+(\dot{Q}).Z_D \tag{8.2.14}$$

- transformation quantitative-qualitative sur $\dot{m}_a$:

$$\dot{m}_a = S.\rho_a.Z_D \simeq \rho_a.Z_D \tag{8.2.15}$$

- transformation quantitative-qualitative sur $\dot{m}_g$:

$$\dot{m}_g = S.\rho_g.(H - Z_D) \simeq \rho_g.(H - Z_D) \tag{8.2.16}$$

- relation entre T_a et T_g :

$$T_a.(\dot{m}_{ent} + \dot{m}_2) + \dot{Q} \simeq T_g.\dot{m}_{dve} \tag{8.2.17}$$

Le réseau de contraintes représentant ce modèle est illustré par la figure 27.

8.3 Simulation du Modèle Qualitatif

Dans l'optique d'aide à la conception, un modèle physique est simulé pour prédire les comportements du système en fonction des variations des paramètres. Le concepteur observera les résultats de cette simulation pour vérifier si les variations de valeurs des paramètres ne viole pas certaines exigences. Au fur et à mesure, il peut modifier le modèle pour que les paramètres ne varient que dans l'espace de quantités qu'il impose.

Pour tester la capacité prédictive du modèle de contraintes, les tests pas-à-pas sont établis sur les variables de l'entrée, et les comportements résultants sont comparés avec ceux d'un modèle quantitatif. La simulation (prédiction) du modèle qualitatif avec QSIM nécessite une description de l'état du système avec lequel la simulateur commence les tests. De manière

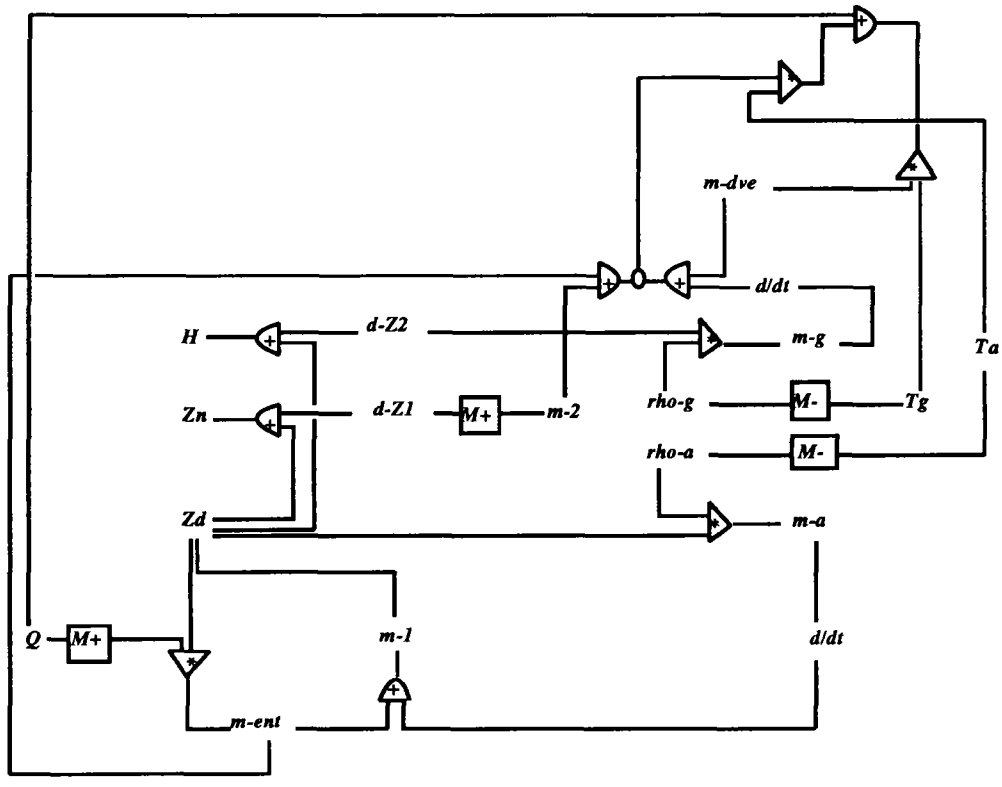


Figure 27: Réseau de Dépendances du modèle qualitatif de désenfumage local.

```
(defun init ()
   (setq *state-limit* 10)
   (let* ((normal (make-initial-state desen-local
                                      '((m-1   ((m1+ m1*) nil))
                                        (m-2   ((m2+ m2*) nil))
                                        (m-ent ((ment+ ment*) nil))
                                        (m-dve (mdve+  std))
                                        (m-a   ((ma+ ma*) nil))
                                        (m-g   ((mg+ mg*) inc))
                                        (Q     ((Q+ Q*) std))
                                        (Z-n   ((Zn+ Zn*) nil))
                                        (Z-d   ((Zd+ Zd*) nil))
                                        (H     ((H+ H*) std))
                                        (Ta    ((ta+ ta*) std))
                                        (Tg    ((tg+ tg*) inc))}
     "Etat Normal feu developpe, debit d'extraction constant."))
  )
    (qsim normal)
    (qsim-display normal)))
```

Figure 28: Description de l'Etat Initial du Modèle Feu Développé

intuitive, l'état initial du modèle est décrit par un ensemble de couples (variable, valeur)[45].

Pour le modèle de désenfumage donné par léquation différentielle qualitative dans la figure 26, l'initialisation d'un pas de test est décrite dans la figure 28.

L'algorithme QSIM est appelé pour simuler les comportements du modèle qualitatif. Il exécute les pas suivants :

1. selectionner un état qualitatif, dans ce cas, l'état *"normal"*,
2. pour chaque paramètre décrit dans la clause *"espaces de quantités"* du modèle 26, déterminer l'ensemble de transitions possibles de l'état qualitatif courant,
3. pour chaque contrainte définie dans la clause *"contraintes"*, générer l'ensemble de tuples de transitions de ses paramètres et

[45]rappelons qu'une valeur qualitative dans le modèle de QSIM est représentée par un couple de valeur significative et une direction de changement, i.e. (qmag , qdir).

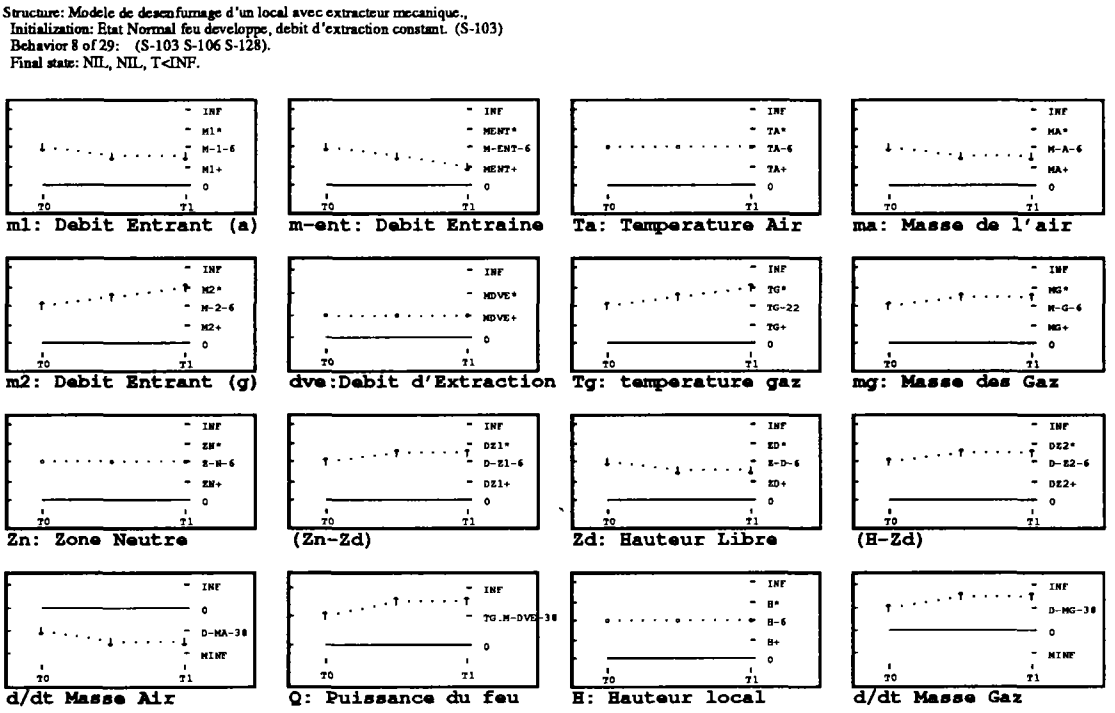


Figure 29: Simulation d'un modèle d'Evacuation de Fumée. Comportement 1: débit d'extraction constant, Q varie

filtrer la consistance de cette contrainte,

4. exécuter le filtrage de la consistance de domaines de valeurs significatives associés aux contraintes définies dans le modèle ; appliquer le critère de la consistance pour que la transition attribuée à un paramètre partagé entre deux contraintes adjacentes soit consistante pour ces contraintes.

5. générer toutes les interprétations globales qui sont possibles à partir des tuples restants. Si aucune interpretation n'est possible, marquer le comportement correspondant comme un comportement inconsistant. Sinon, créer des nouveaux états qualitatifs résultant de chaque interprétation, les considérer comme les successeurs de l'état courant,

6. appliquer les règles du filtrage global[46] aux nouveaux états récursivement de (2) à (6).

En résumé, QSIM analyse le comportement d'un modèle qualitatif dans le temps en : (1) commençant avec un état initial, (2) déterminer les changements qualitatifs dans chaque paramètre, (3) identifier les quantités correspondantes aux transitions et contraintes, (4) puis construire un ensemble d'états issue de ces transitions, et finallement, (5) pour chaque nouvel état, recommencer l'analyse recursivement de (2) à (5). Le résultat est affiché sous forme d'un arbre de comportements dont la racine est l'état spécifié. Chaque comportement est décrit par le chemin de la racine à une feuille de l'arbre. Les figures 29, 30 et 31 illustrent quelques comportements

[46]Les détails de ces règles sont décrites dans [Kuipers 1985a], elles consistent en règles appelées "value continuity", "contradiction avoidance", "instant change", "derivative continuity", "derivative instant change rules", *etc.*

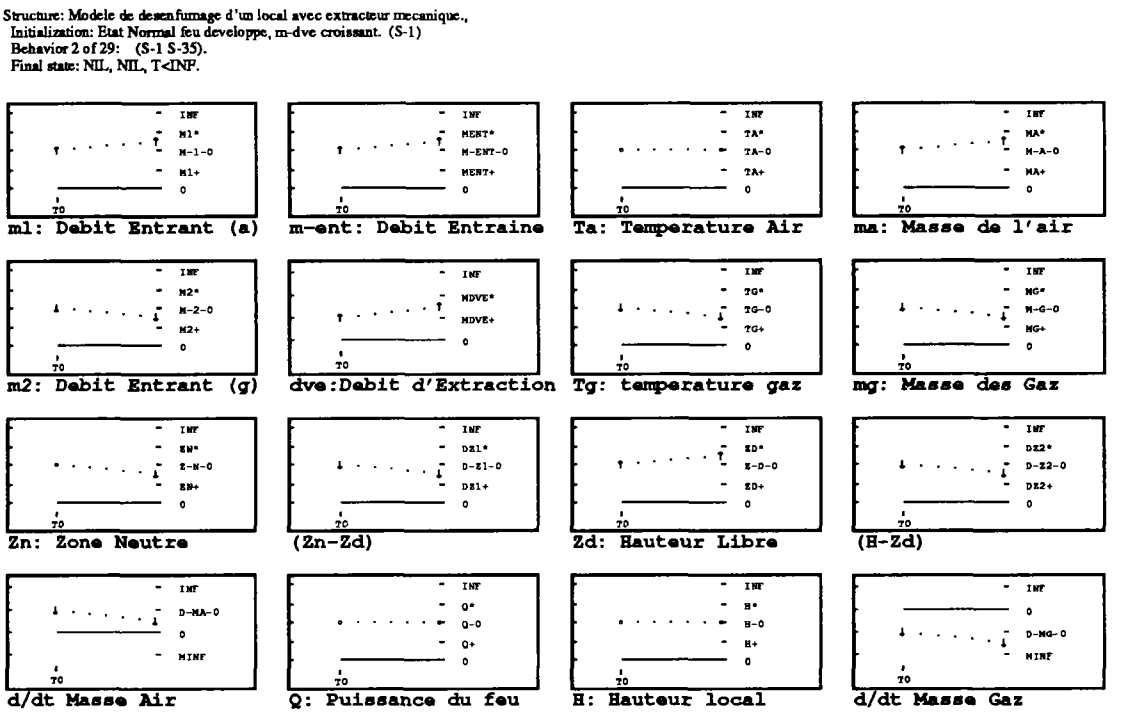


Figure 30: Simulation d'un modèle d'Evacuation de Fumée. Comportement 2: Q constant, débit d'extraction décroit

issus du modèle de désenfumage décrit dans la figure 26.

8.4 Application à la Conception

Rappelons l'objectif du problème est le dimensionnement du débit d'extraction mécanique de telle manière que si un feu s'est développé dans l'un, le gaz produit par ce feu ne se propage pas dans l'autre. C'est à dire, pour un feu de puissance Q et pour la zône neutre plus haute que la hauteur h de la porte qui sépare les deux locaux, la zône "libre" en dessous de la nappe de fumée soit suffisamment haute.

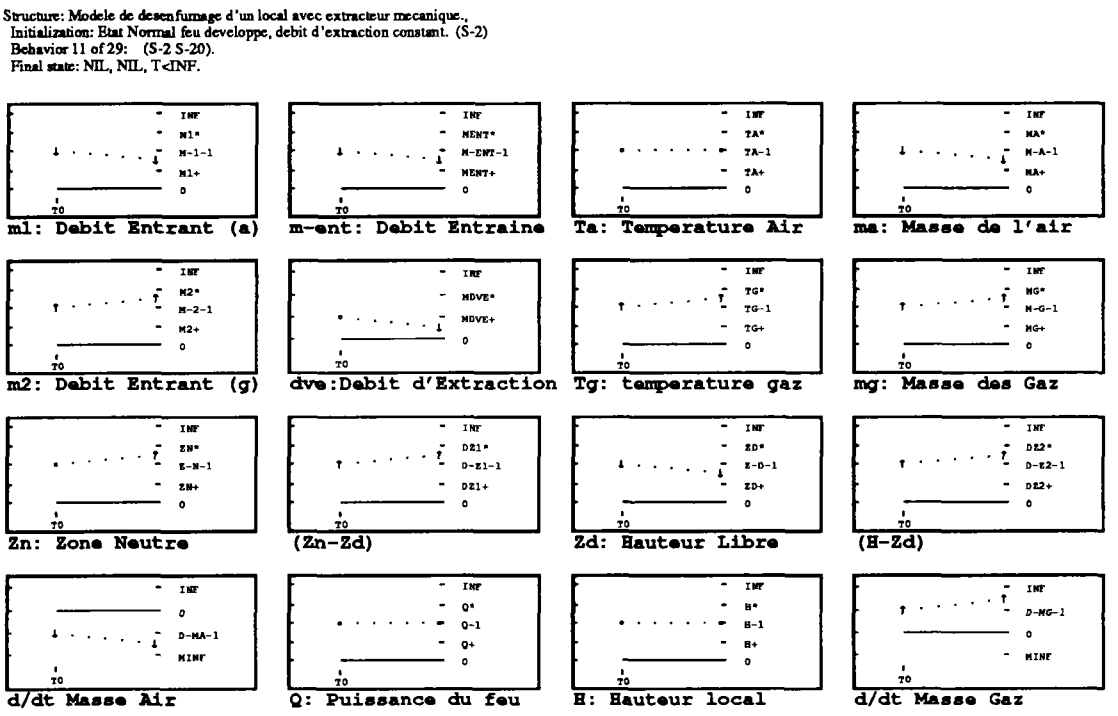


Figure 31: Simulation d'un modèle d'Evacuation de Fumée. Comportement 3: Q constant, débit d'extraction accroit

Pour illustrer le comportement du modèle dans le cas où le débit d'extraction est insuffisant, on peut faire varier le feu de manière croissance. Par le comportement illustré dans la figure 29, on constate que Z_D décroit continuément pour une feu de puissance $\dot{Q}$ croissante. Si la valeur qualitative `Z-d*` est la limite admise, cette limite est facilement atteinte par une feu plus violent. Il est donc préférable d'augmenter le débit d'extraction. Pour une feu de puissance `Q+` donnée, quelques comportements possibles du modèle sont illustrés par les figures 30 et 31.

Par une manipulation sur les tendances de variation des paramètres,

les comportements qualitatifs des paramètres peuvent être obtenus comme résultats. L'affichage graphique de ces comportements permet aux concepteurs de prédire les variations et enfin de délimiter les domaines de valeurs associes aux paramètres.

8.5 Resumé du Chapitre

Dans ce chapitre, nous avons montré un exemple de l'utilisation de QSIM dans la conception ainsi que les démarches de modélisation qualitative, interprétation des résultats. En résumé, le processus de travail pour un concepteur consiste à :

1. transformer un modèle quantitatif partiel en un modèle qualitatif,

2. déterminer un état initial pour lancer l'algorithme de simulation de QSIM,

3. interpréter les résultats de QSIM pour raffiner les contraintes et-ou les états initiaux.

4. relancer la simulation avec le modèle modifié ou raffiné avec un nouvel état et/ou avec les espaces de quantités spécifiés.

5. une fois le comportement souhaité obtenu, les valeurs qualitatives correspondantes sont interprètées et inserées dans le modèle algébrique avec NETGEN. Ceci jusqu'à ce que l'objectif de conception soit atteint.

Notons que plusieurs capacités de QSIM ne sont pas encore abordées, par exemple l'analyse des portraits de phases entre deux paramètres, l'in-

sertion des informations quantitatives dans le modèle qualitatif, *etc.* Les détails de ces capacités et les limites de QSIM sont décrites dans la thèse de David Dalle Molle [Molle 1989].

Cependant, seule la capacité de prédiction de QSIM dans l'exemple d'un modèle de désenfumage que nous montrons ci-dessus est bénéfique : QSIM peut fournir des informations qualitatives nécessaires pour guider le concepteur dans son processus de conception même si la connaissance est incomplète.

Il importe de noter qu'une de nos idées initiales a consisté à créer un pont entre le réseau algébrique et le réseau de contraintes qualitatives afin de pouvoir analyser les contraintes et les variations en même temps. Cette tentative a échoué à cause de la limite liée à la simulation qualitative que nous avons abordé dans 7.2., nottamment le problème de perte d'information dès que les contraintes fonctionnelles sont utilisées pour remplir les connaissances incomplètes. Néanmoins, nous pensons qu'avec une architecture de contrôle adéquate [Arlabosse 1988, Arlabosse 1989b], la coopération NETGEN et QSIM peut être effectuée sans difficultés majeures. Aussi dans cette optique, d'autres modules d'analyse, e.g. MACSYMA, analyse d'élements finis, *etc.*, pourront également coopérer avec les deux modules de gestion de contraintes algébriques (NETGEN) et qualitatives (QSIM) pour aider l'ingénieur dans sa tâche de conception en "vrai profondeur".

Bien entendu, des études approfondies sont nécessaires pour réalisé ce que nous avons appelé la "vraie profondeur" d'un système de conception assisté par ordinateur. Ce travail est une étude exploratoire d'une méthode de l'intelligence artificielle appliquée dans la conception en ingénierie : l'approche de contraintes algébriques et qualitatives. Nous abordons

le problème de recherches futures dans le chapitre suivant.

Chapitre 9
Conclusion Générale

"Je me démenais pour unir la pensée à venir
A la pensée précédente.
Mais leur ordre m'échappait en s'enmêlant
Tout comme des pelotes roulant sur le plancher."

Emily Dickinson

La thèse que nous présentons est que la représentation par contraintes, en terme de langage, et la déduction par la gestion automatique de réseaux de contraintes, sur le plan de raisonnement, est une voie potentiellement adéquate pour les systèmes d'aide à la conception.

Cet travail a présenté un ensemble de problèmes relatifs à la génération automatique des contraintes dynamiques. Les concepts fondamentaux dans notre approche de résolution sont :

1. *décomposition uniforme :* tout problème physique peut être décomposé en termes de réseaux de contraintes. Chaque noeud de ce type de réseau représente une variable ou un paramètre physique et chaque contrainte représente une relation entre un groupe de paramètres. La dynamique d'un modèle physique ne repose pas seulement sur la dynamique des groupes de noeuds mais aussi sur la dynamique des groupes de relations, c'est è dire sur les modifications que le concepteur peut être amèné à apporter à sa représentation du problème et des exigences qu'il impose à

la solution.

2. *représentation par contraintes :* la dynamique des relations joue un rôle essentiel dans la conception. Le problème de conception correspond en fait au problème de résolution des réseaux de contraintes dynamiques, c'est à dire le problème d'analyse de la consistance entre les paramètres, ses domaines de valeurs et les contraintes exerçées sur ces valeurs en prenant en compte la possibilité de manipuler toutes les variables, relations ou valeurs dynamiquement.

3. *analyse par causalité :* la gestion de contraintes ne se restreint pas seulement au problème d'analyse de la consistance d'un réseau. L'explication des solutions consistantes, l'élimination des contraintes redondantes et la résolution de l'incohérence, *etc.* sont en effet, les fonctions d'aide à la conception. Elles reposent sur la notion de la causalité : il n'y a jamais de fumée sans feu.

4. *déduction par l'élimination :* le mécanisme de déduction consiste à rechercher le domaine de "falsification" du problème posé, i.e. les variables, valeurs et contraintes qui ne satisfont pas les règles de consistance, puis à les éliminer de l'espace de recherche. Ainsi, quand l'espace de solutions est petit, le risque de l'explosion combinatoire dans la recherche de solution est minimisé.

Une grande partie de notre travail a consisté à développer un formalisme de gestion de réseaux de contraintes n-aires sur la base des travaux existants pour les contraintes binaires. Les chapitres 3 et 4 sont consacrés à

l'établissement de ce formalisme. Ils nous ont servi pour réaliser un ensemble de logiciels. Nous espérons avoir démontré la faisabilité et la flexibilité de l'approche choisie dans le chapitre 5 ainsi que dans l'annexe A.

Une des difficultés que nous avons rencontré est celle de l'analyse des problèmes dynamiques i.e. la résolution automatique des systèmes d'équations différentielles avec l'approche de contraintes. Celle-ci nous a amèné sur la piste de la *physique qualitative*. La deuxième partie de ce travail concerne l'étude de cette technique.

Plusieurs tentatives ont été effectuées au long de ce travail pour rechercher un mécanisme de gestion des contraintes qualitatives homogène avec celui traitant les contraintes algébriques. La première consiste à introduire la notion de contraintes fonctionnelles pour gérer la dynamique des systèmes (cf. section 5.4), la seconde consiste à introduire la méthode de Vaschy-Buckingham[47] et à gérer les relations entre les contraintes par une opération sur les contraintes. Ces tentatives n'ont pas donné de résultats positifs. Les chapitres 6 et 7 sont consacrés à rechercher une méthode d'analyse qualitative la plus proche possible de notre approche. Nous avons trouvé que l'algorithme de simulation qualitative QSIM est adéquate pour le problème de conception. Le chapitre 8 a consisté à concrétiser par un exemple démontrant quelques possibilités de QSIM.

Nous avons par cette thèse, contribué à la recherche des algorithmes de filtrage de la consistance dans des réseaux de contraintes, particulièrement dans des réseaux de contraintes n-aires. Cependant, ce travail exploratoire

[47]Cette méthode de l'analyse adimensionnel permet de ré-écrire les groupes de nombre Π faisant sortir les relations d'ordres entre paramètres en mettant en jeu les relations entre les unités de grandeurs de chaque paramètre via une matrice $(m \times n)$, où m représente les unités de grandeurs, n représente le nombre de paramètres intervenant dans le système physique à modéliser.

s'appliquent uniquement à une classe limitée de problème des contraintes algébriques. Néanmoins, le formalisme que nous avons mis en place peut être généralisé pour gerer une classe d'algèbre de contraintes. Nous pensons qu'une recherche sur une "ALGÈBRE DE CONTRAINTES" est une voie prometteuse pour une application plus large de la technique de réseaux de contraintes dans les domaines où les *"problèmes inverses"* sont difficiles à résoudre.

Par contre, en ce qui concerne la technique d'analyse qualitative, nous avons trouvé que :

- la construction des modèles qualitatifs n'est pas une procédure bien définie. Une technique peut être adéquate pour certains modèles mais peut être inadéquate voire inapplicable pour certains d'autres,

- dans la transformation des relations mathématiques en contraintes qualitatives, plusieurs transformations sont possibles faute des simplifications. Dans certains cas, le modèle qualitatif s'éloigne du modèle numérique. Il est difficile pour un non-spécialiste de choisir un modèle qui peut représenter tous les "réalismes" dans un système d'équations différentielles ordinaires.

- QSIM reste encore prometteur avec ses capacités de gestion de contraintes d'ordres supérieurs et de l'introduction des informations quantitatives (Q2) dans les modèles qualitatifs.

Nous pensons que l'approche de la "MATHÉMATIQUE QUALITATIVE" pour la simulation comme celle présentée par Sacks [Sacks 1990] ou bien

par [Dordan 1990] mérite d'être examinée dans les recherches futures.

Enfin, nous espérons qu'à travers les développements de ce mémoire, le but de ce travail est atteint, à savoir montrer que l'approche de résolution des réseaux de contraintes algébriques et qualitatives dans les système d'aide à la conception en ingénierie est une voie adéquate.

Bibliographie

[Arlabosse 1988] F. Arlabosse, V. Duong, E. Gaussens, and P. Le Page. CELL-TISSUE: a control architecture for knowledge-based systems. In *Proceedings of the Eight International Workshop on Expert Systems and their Applications, AVIGNON 88*, EC2, 1988.

[Arlabosse 1989a] F. Arlabosse, V. Duong, E. Gaussens, and P. Le Page. Blackboard and alternative architectural design for two real-scale kbs industrial application. In *Blackboard Systems and Applications*, Ed. Larry Baum et als., Academic Press, Cambridge USA., 1989.

[Arlabosse 1989b] F. Arlabosse, V. Duong, E. Gaussens, and P. Le Page. Cell-tissue: une architecture de controle pour les systemes a base de connaissance. *TSI-Techniques et Science de l'Informatique*, 7(2):, 1989.

[Arvind 1978] Gostelow Arvind, P. Kim, and Wil. Plouffe. *An Asynchronous Programming Language and Computing Machine.* Technical Report xx, Dept. of Information and Computer Science, University of California, December 1978.

[Aubin 1989] Jean-Pierre Aubin. Theorie de la viabilite. 1989. CEREMADE, Universite Paris 9, Dauphine.

[Berliner 1985] Hans Berliner and G. Goetsch. The study of search methods: the effort of constraint satisfaction in adventurousness. In *Proc. of IJCAI-85*, pages 1079–1082, IJCAI, Inc., Los Angeles, CA, August 1985.

[Bibel 1988] W. Bibel. Constraint satisfaction from a deductive viewpoint. *Artificial Intelligence*, 35(3):401–413, July 1988.

[Bobrow 1985] D. G. (ed.) Bobrow. *Qualitative Reasoning about Physical Systems.* Massachusetts Institute Technology Press, Cambridge, MA, 1985.

[Bonissone 1985] P. Bonissone and K.P. Valavanis. A comparative study of different approaches to qualitative physics theories. In *Proceedings of the Second Conf. Artificial Intelligence Applications*, IEEE Computer Society, 1985.

[Borning 1986] Alan Borning and Robert Duisberg. Constraint-based tools for building user interfaces. *ACM Transactions on Graphics*, 5(4):345–374, October 1986.

[Bredeweg 1988] Bert Bredeweg and Bob J. Wielinga. Integrating qualitative reasoning approaches. In *Proceedings of the European Conference on Artificial Intelligence ECAI-88*, ECAI, 1988. Munich, 1988.

[Bredeweg 1990] Bert Bredeweg, Martin Reinders, and Bob J. Wielinga. *GARP: A Unifying Approach to Qualitative Reasoning.* Technical Report VF-memo 117, Unviversity of Amsterdam, Netherlands, 1990.

[Brinkley 1986] J. Brinkley, C. Cornelius, R. Altman, B. Hayes-Roth, O. Lichtarge, B. Duncan, B. Buchanan, and O. Jardetzky. *Application of Constraint Satisfaction Techniques to the Determination of Protein Tertiary Structure.* Technical Report KSL Report No. KSL-86-28, Stanford University, Knowledge Systems Laboratory, March 1986.

[Buchberger 1985] Bruno Buchberger. Grobner bases: an algorithmic method in polynomial ideal theory. In *Multidimensional Systems Theory*, Reidel, Dordrecht, The Netherlands., 1985.

[Bylander 1988] T. Bylander. A critique of qualitative simulation from a consolidation viewpoint. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-18(2):252–263, March-April 1988.

[Caplain 1990] Gilbert Caplain. *Quelques Reflexions sur la Physique Qualitative.* Technical Report interne, CERMA- Ecole Nationale des Ponts et Chaussees et Dept. Recherche et Developpement Framentec, 1990.

[Cross 1983] S. E. Cross. *Qualitative Reasoning in an Expert System.* PhD thesis, Univ. of Illinois, Champaign-Urbana, IL, 1983.

[CSTB 1986] CSTB. *REEF Thermique et Aeraulique.* Centre Scientifique et Technique du Batiment, Paris, France, 1986.

[Davis 1987] Ernest Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331, July 1987.

[Dechter 1985] R. Dechter and J. Pearl. The anatomy of easy problems: a constraint-satisfaction formulation. In *Proc. of IJCAI-85*, pages 1066–1072, IJCAI, Inc., Los Angeles, CA, August 1985.

[Dechter 1986] R. Dechter. Learning while searching in constraint-satisfaction problems. In *Proc. AAAI-86*, pages 178–183, AAAI, Philadelphia, PA, August 1986.

[Dechter 1987a] A. Dechter and Rina Decter. Removing redundencies in constraint networks. In *Proceedings AAAI-87*, 1987.

[Dechter 1987b] Rina Dechter and Judea Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 1987.

[Dechter 1989] Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 1989.

[deKleer 1975] Johan de Kleer. *Qualitative and Quantitative Knowledge in Classical Mechanics.* PhD thesis, MIT, Cambridge, MA, 1975.

[deKleer 1979] J. de Kleer. Qualitative and quantitative reasoning in classical mechanics. In *Artificial Intelligence: An MIT Perspective*, pages 9–30, The MIT Press, Cambridge, 1979.

[deKleer 1984a] J. de Kleer and Daniel G. Bobrow. Qualitative reasoning with higher-order derivatives. In *Proc. AAAI-84 Nat'l Conf.*, AAAI, University of Texas at Austin, TX, August 1984.

[deKleer 1984b] Johan de Kleer and John Silly Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24(Special):7–83, 1984.

[deKleer 1986a] Johan de Kleer. An assumption based truth maintenance system. *Artificial Intelligence*, 28(2):127–162, March 1986.

[deKleer 1986b] Johan de Kleer and John Silly Brown. Theories of causal ordering. *Artificial Intelligence*, 26(1):33–61, July 1986.

[deKleeretals 1978] Johan de Kleer et als. *AMORD: A Deductive Procedure System.* Technical Report AIM-435, M.I.T. Artificial Inteligence Laboratory, January 1978.

[Dennis 1973] J. B. Dennis. *First Version of a Data Flow Procedure Language.* Technical Report TM-93, M.I.T. Laboratory of Computer Science, November 1973.

[dK 1980] Johan de Kleer and Gerald J. Sussman. Propagation of constraints applied to circuit synthesis. *Circuit Theory and Applications*, 8:127–144, 1980.

[dK 1987] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, April 1987.

[Dordan 1990] Olivier Dordan. *Analyse Qualitative.* PhD thesis, Universite Paris 9 Dauphine, 1990.

[Dormoy 1987] Jean Luc Dormoy. *Resolution Qualitative : Completude, Interpretation Physique et Controle. Mise en Oeuvre dans un langage a base de regles: BOOJUM.* PhD thesis, Universite Paris 6, 1987.

[Doyle 1977] Jon Doyle. *Truth Maintenance Systems for Problem Solving.* Master's thesis, M.I.T., June 1977. 96 pages.

[Doyle 1979] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979.

[Doyleetals 1977] Jon Doyle et als. Amord: explicit control on reasoning. In *Proceedings AI and Programming Languages Conference*, pages 116–125, ACM SIGPLAN, ACM, August 77 1977. Rochester, Newyork, USA.

[Falkenhainer 1986] Brian C. Falkenhainer and Ryszard S. Michalski. Integrating quantitative and qualitative discovery: the abacus system. *Machine Learning*, 1(4):367–401, 1986.

[Falkenhainer 1988] B.C. Falkenhainer and K.D. Forbus. Setting up large scale qualitative models. In *AAAI-88*, AAAI, 1988.

[Farquhar 1990] A. Farquhar, B.J. Kuipers, and D. Throop. *QSIM user's Manual and Maintainer's Guide.* Technical Report AI.TR. 90-123 and 90-124, University of Texas at Austin, 1990.

[Forbus 1981a] K.D. Forbus. Qualitative reasoning about physical processes. In *IJCAI-81*, IJCAI, Vancouver, British Columbia, Canada, 1981.

[Forbus 1981b] K.D. Forbus. *A Study of Qualitative and Geometric Knowledge in Reasoning about Motion.* PhD thesis, M.I.T. A.I. Lab, 1981. AI-TR-615.

[Forbus 1983] K.D. Forbus. Measurement interpretation in qualitative process theory. In *Proc. IJCAI-83*, pages 315–320, IJCAI, Karlsruhe, West Germany, August 1983. Volume 1.

[Forbus 1984] K.D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24(1-3):85–168, December 1984.

[Forbus 1987] Kenneth D. Forbus. *The Qualitative Process Engine: A Study in Assumption based Truth Maintenance.* Technical Report Draft, Dept. Computer Science, University of Illinois, February 1987. Submitted to the Q.P.W. 87.

[Forbus 1988] Kenneth D. Forbus and Johan de Kleer. Focusing the atms. In *Proceedings on the Eightth National Conference on Artificial Intelligence*, AAAI, Morgan Kaufmann, August 1988. St Paul USA.

[Fox 1982] M. S. Fox, B. Allen, and G. Strohm. Job-shop scheduling: an investigation in constraint-directed reasoning. In *Proc. AAAI-82*, pages 155–158, Pittsburgh, 1982.

[Fox 1983] M.S. Fox, B. Allen, S. Smith, and G. Strohm. Isis: a constraint-directed search approach to job-shop scheduling. In *Proc. IEEE Computer Society Trends and Applications*, IEEE, National Bureau of Standards, Washington, D.C., 1983.

[Freuder 1978] Eugene C. Freuder. Synthetizing constraint expression. *Communication of the ACM*, 1978.

[Freuder 1982] Eugene C. Freuder. A sufficient condition of backtrack-free search. *Communication of the ACM*, 1982.

[Freuder 1985a] E. Freuder and M. Quinn. Taking advantage of stable sets of variables in constraint satisfaction problems. In *Proc. of IJCAI-85*, pages 1076–1078, IJCAI, Inc., Los Angeles, CA, August 1985.

[Freuder 1985b] Eugene C. Freuder. A sufficient condition for backtrack-bounded search. *Communication of the ACM*, 1985.

[Freuder 1989] Eugene C. Freuder. Partial constraint satisfaction. In *Proceedings IJCAI-89*, 1989.

[Goodwin 1982] James Goodwin. *An Improved Algorithm for Non-Monotonic Dependency Net Update.* Technical Report MAT-R-82-23, Linkoping Institute of Technology, 1982.

[Grossman 1976] Richard W. Grossman. *Some Data Base Applications of Constraints Expressions.* Technical Report TR-158, M.I.T. Laboratory of Computer Science, February 1976.

[Haralick 1978] R.M. Haralick, Larry S. Davis, and Azriel Rosenfeld. Reduction operations for constraint satisfaction. *Information Sciences*, 14, 1978.

[Haralick 1980] R.M. Haralick and G.L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *AI*, 14(3):263–313, October 1980.

[Hayes 1985] Patrick J. Hayes. The second naive physics manisfesto. In R. Moore, editor, *Formal Theories of the Commonsense World*, pages 1–36, Ablex Pub., 1985.

[HayesRoth 1983] Barbara Hayes-Roth. *The Blackboard Architecture: A General Framework for Problem Solving?* Technical Report HPP-83-30, Stanford University, Dept. Computer Science, may 1983.

[HayesRoth 1985] Barbara Hayes-Roth. A blackboard achitecture for control. *Artificial Intelligence*, 26(3):251–321, 1985.

[Hu 1989] Gongzhu Hu and George Stockman. 3-d surface solution using structured light and constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(4):390–402, April 1989.

[Iwasaki 1986a] Yumi Iwasaki and Hebert A. Simon. Causality in device behavior. *Artificial Intelligence*, 29(1):3–32, July 1986.

[Iwasaki 1986b] Yumi Iwasaki and Hebert A. Simon. Theories of causal ordering: reply to de kleer and brown. *Artificial Intelligence*, 29(1):63–72, July 1986.

[Iwasaki 1988] Yumi Iwasaki and I. Bhandari. Formal basis for abstraction of dynamic systems. In *Proceedings of the AAAI 88*, AAAI, August 1988. St Paul, USA.

[Kasif 1986] S. Kasif. On the parallel complexity of some constraint satisfaction problems. In *Proc. AAAI-86*, pages 349–353, AAAI, Philadelphia, PA, August 1986.

[Kuipers 1985a] Benjamin J. Kuipers. The limits of qualitative simulation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, IJCAI-85, August 1985. Los Altos, CA.

[Kuipers 1985b] B.J. Kuipers. Getting the envisionnement right. In *Proc. of IJCAI-85*, pages 128–136, IJCAI, Inc., Los Angeles, CA, August 1985.

[Kuipers 1986] Benjamin J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29(3):289–338, 1986.

[Kuipers 1987a] Benjamin J. Kuipers. Abstraction by time-scale in qualitative simulation. In *Proceedings of the Seventh National Conference on Artificial Intelligence AAAI-87*, pages 621–625, AAAI, Morgan Kaufmann, Los Altos, CA., 1987.

[Kuipers 1987b] Benjamin J. Kuipers. Qualitative simulation as causal explanation. In *IEEE Transactions on Systems, Man and Cybernetics*, IEEE, Kaufmann, 1987.

[Kuipers 1987c] B.J. Kuipers and Kassirer J.P. Knowledge acquisition by analysis of verbatim protocols. In A. Kidd, editor, *Knowledge Acquisition for Expert Systems*, Plenum NY., 1987.

[Kuipers 1988] B.J. Kuipers and W.W. Lee. Non-intersection of trajectories in qualitative phase space: a global constraint for qualitative simulation. In *Proceedings of AAAI-88*, AAAI, St Paul, USA, August 1988.

[Kuipers 1989] B.J. Kuipers and D. Dvorak. Model-based monitoring of dynamic systems. In *Proc. IJCAI-89*, IJCAI, Detroit USA, August 1989.

[Lensky 1986] A. V. Lensky, A. M. Lizunov, A. B.and Formal'sky, and A. Yu. Schneider. Manipulator motion along a constraint. *Robotica*, 4(4):247–254, October-December 1986.

[Lowe 1987] David G. Lowe. The viewpoint consistency constraint. *International Journal of Computer Vision*, 1(1):57–72, Winter 1987.

[Mackworth 1977] Alan K. Mackworth. Consitency in network of relations. *Artificial Intelligence*, 1977.

[Mackworth 1985] Alan K. Mackworth and Eugene C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 1985.

[Maher 1984] M.L. Maher and S.J. Fenves. Hi-rise: an expert system for the preliminary structural design of high-rise buildings. In *Knowledge Engineering in Computer Aided Design*, J. Gero. North Holland, 1984.

[Mavrovouniotis 1988] M.L. Mavrovouniotis and G. Stephanopoulos. Formal order-of-magnitude reasoning in process engineering. *Computers and Chemical Engineering*, 12(9-10):867–880, 1988.

[McAllester 1980a] David Allen McAllester. *An Outlook on Truth Maintenance.* Technical Report memo-551, M.I.T. - The Artificial Intelligence Laboratory, August 1980.

[McAllester 1980b] David Allen McAllester. *The Use of Equality in Deduction and Knowledge Representation.* Technical Report AI-550, M.I.T. - The Artificial Intelligence Laboratory, January 1980.

[Mohammed 1986] J. Mohammed and R. Simmons. Qualitative simulation of semiconductor fabrication. In *Proc. AAAI-86*, pages 794–799, AAAI, Philadelphia, PA, August 1986.

[Mohr 1986] Roger Mohr and Thomas C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 1986.

[Mohr 1988] Roger Mohr and Masini G. Good old discrete relaxation. In *Proceedings of ECAI 88*, 1988.

[Molle 1989] David Dalle Molle. *Qualitative Simulation of Dynamic Chemical Processes.* PhD thesis, University of Texas at Austin, Austin, Tx, USA, 1989.

[Montanari 1974] U. Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Science*, 1974.

[Morgan 1987] Alexander Morgan. *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems.* Prentice Hall, Inc., 1987.

[Morgenstern 1984] Matthew Morgenstern. Constraint equations: a concise compilable representation for quantified relational constraints in semantic networks. In *AAAI-84 Nat'l Conf.*, AAAI, University of Texas at Austin, TX, August 1984.

[Mulder 1988] Alan K.; Mulder, Jan A.; Mackworth and William S. Havens. Knowledge structuring and constraint satisfaction: the mapsee approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-10(6):866–879, November 1988.

[Nachtcheim 1989] Philip R. Nachtcheim. Solving constraint satisfaction problems. *AI Expert*, 4(6):30–35, June 1989.

[Oyeleye 1988] O.O. Oyeleye and M.A. Kramer. Qualitative simulation of chemical process systems: steady state analysis. *AIChE Journal*, 34(9):1441–1459, 1988.

[Pan 1986] J.Y. Pan and Tenenbaum. PIES: an engineer's do-it-yourself knowledge system for interpretation of parametric test data. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, AAAI, Morgan Kauffmann, 1986. Pa, USA.

[Porte 1988] Nathalie Porte, Stephane Boucheron, Jean Sallantin, and Francois Arlabosse. An algorithmic view at causal ordering. In *Proceedings of the Second Qualitative Physics Workshop*, Framentec-CRIM, 1988. Paris, 1988.

[Quillian 1968] M. Ross Quillian. Semantic memory. In Marvin Minsky, editor, *Sematic Information Processing*, M.I.T. Press, 1968.

[Sacks 1985] Elisha P. Sacks. Qualitative mathematical reasoning. In *Proc. of IJCAI-85*, pages 137–139, IJCAI, Inc., Los Angeles, CA, August 1985.

[Sacks 1988] Elisha P. Sacks. Qualitative analysis by piecewise linear approximation. *Artificial Intelligence in Engineering*, 3(3):151–155, 1988.

[Sacks 1990] Elisha P. Sacks. Automatic qualitative analysis of dynamic systems using piecewise linear approximation. *Artificial Intelligence*, 41(2):313–364, 1990.

[Sriram 1984] Duvvruru Sriram. Knowledge-based expert systems in structural design. In *NASA conference on Advances in Structural Mechanisms.*, NASA, Washington D.C. USA, 1984.

[Stallman 1977] Richard M. Stallman and Gerald J. Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9:135–196, 1977.

[Stallman 1979] Richard M. Stallman and Gerald J. Sussman. Problem solving about electrical circuit. In *Artificial Intelllgence: An MIT Perspective*, MIT Press, 1979.

[SteeleJr 1980] Guy Lewis Steele Jr. *The Definition and Implementation of a Computer Programming Language based on Constraints.* PhD thesis, Massachusetts Institute of Technology - AI. Lab., August 1980. 371 pages.

[Stevens 1983] Kent A. Stevens. The line of curvature constraint and the interpretation of 3-d shape from parallel surface contours. In *Proc. IJCAI-83*, pages 1057–1061, IJCAI, Karlsruhe, West Germany, August 1983. Vol. 2.

[Struss 1987a] Peter Struss. *Mathematical Aspects of Qualitative Reasoning.* Technical Report draft, Siemens Corp, Munich RFA, May 1987.

[Struss 1987b] Peter Struss. *Multiple Representation of Structure and Function.* Technical Report none, Siemens Corp, Munich RFA, 1987. To appear in: John Gero(ed.), Expert Systems in Computer Aided Design, North Holland 1987.

[Struss 1988] Peter Struss. Global filters for qualitative behaviors. In *Proceedings on the Eighth National Conference on Artificial Intelligence AAAI-88*, Siemens Corp, Munich RFA, Morgan Kaufmann, August 1988. St. Paul, Minnesota, USA.

[Sussman 1980] Gerald J. Sussman and Guy L. Steele Jr. CONSTRAINTS - a language for expressing almost-hierarchical descriptions. *Artificial Intelligence*, 14(1):1–39, 1980.

[Waltz 1972] David L. Waltz. *Generating Semantic Descriptions from Drawings of Scenes With Shadows.* Technical Report ATR-271, M.I.T. Artificial Inteligence Laboratory, November 1972.

[Weld 1986] David S. Weld. The use of aggregation in causal simulation. *Artificial Intelligence*, 30(1), 1986.

[Weld 1988] David S. Weld. Choices for comparative analysis : dq analysis or exaggeration. *Artificial Intelligence in Engineering*, 3(3):174–180, 1988.

[Williams 1984] Brian C. Williams. The use of continuity in a qualitative physics. In *Proc. AAAI-84*, AAAI, University of Texas at Austin, TX, August 1984.

[Williams 1986] B.C. Williams. Doing time: putting qualitative reasoning on firmer ground. In *Proc. AAAI-86*, pages 105–112, AAAI, Philadelphia, PA, August 1986.

[Williams 1987] Brian C. Williams. *Beyond Qualitative Reasoning.* Technical Report Extended Abstract, M.I.T. Artificial Intelligence Lab., February 1987. Paper Submitted to the Qualitative Physics Workshop 87.

[Williams 1988] B.C. Williams. Minima;a symbolic approach to qualitative algebraic reasoning. In *Proc. AAAI-88*, AAAI, St Paul, USA, August 1988.

[Yip 1987] Kenneth Man-Kam Yip. *Extracting Qualitative Dynamics from Numerical Experiments.* Technical Report Draft, M.I.T. Artificial Intelligence Lab., February 1987. published in proceedings of AAAi 87.

[Zucker 1988] Steven W. Zucker and Sheldon Davis. Points and endpoints: a size/spacing constraint for dot grouping. *Perception*, 17(2):229–247, March 1988.

Sommaire

Liste des illustrations