N° d'ordre : 2556

THÈSE

présentée devant

L'UNIVERSITÉ PAUL SABATIER DE TOULOUSE (SCIENCES)

en vue de l'obtention du titre de

Docteur de l'Université Paul Sabatier

Spécialité : **INFORMATIQUE**

par

Philippe DEJEAN

Un Formalisme pour les Entités du Traitement et de l'Analyse des Images

Soutenue le 18 décembre 1996, devant la commission d'examen :

Serge CASTAN	:	Professeur à l'Université Paul Sabatier (Toulouse)	Directeur
Marc BERTHOD	:	Directeur de Recherche à l'INRIA (Sophia-Antipolis)	Rapporteur
Marinette REVENU	:	Professeur à l'ISMRA (Caen)	Rapporteur
Monique THONNAT	:	Directeur de Recherche à l'INRIA (Sophia-Antipolis)	Rapporteur
Patrice DALLE	:	Maître de conférence à l'Université Paul Sabatier (Toulouse)	Examinateur
Henri FARRENY	:	Professeur à l'INP-ENSEEIHT (Toulouse)	Examinateur
Gérard PADIOU	:	Professeur à l'ENSEEIHT (Toulouse)	Examinateur

Résumé

Dans le cadre de l'analyse et de l'interprétation d'images, nous proposons une nouvelle approche pour la modélisation des entités de traitement d'images. Cette approche diffère de celles implémentées dans des systèmes classiques. En effet, notre modèle de données est unique quelle que soit la dimension ou la nature de l'entité manipulée. Il se décompose en cinq éléments caractéristiques dont les expressions sont développées dans un langage rigoureux : LEO. Le modèle d'opérateurs rend compte des transformations d'informations effectuées par ceux-ci, et décrit précisément les entités produites intervenant comme indices visuels durant le processus d'interprétation. Ainsi une séquence d'opérateurs est considérée comme un constructeur de concepts fournissant une description de ceux-ci dans le langage proposé. Le formalisme que nous avons développé, constitué des modèles de données et d'opérateurs et du langage, permet ainsi de décrire de façon rigoureuse le traitement d'images en tant que manipulation de données, mais aussi en tant que manipulation de concepts d'un domaine spécifique. De cette façon, nos modèles peuvent être utilisés par un système de résolution d'objectifs d'analyse d'images. Par le formalisme utilisé, ils fournissent des éléments intéressant pour une bonne compréhension de la chaîne produite ou pour une aide à la configuration et à la mise au point de ces chaînes.

Mots-clés :
Traitement d'images, Modélisation, Indices visuels, Transformation d'information, Langage d'expression, Planification

Abstract

Within the framework of image analysis and image understanding, we propose a new approach for the modeling of image processing entities. This approach differs from those implemented in classical systems. Effectively, our data model is the same whatever dimension or nature of the handled entities. It is formed by a set of five caracteristic elements for which expressions are developped in a rigorous language : LEO. Our operator model accounts for the transformation of informations made by the operator, and describes accurately the produced entities intervening as visual cues during the interpretation process. These visual cues will become concepts according to the application area. Therefore, an operator sequence is considered as a concept constructor supplying a description of these concepts. The formalism we have developped formed by LEO with the data and operator models allows to describe rigorously the image processing as data handling and a system transforming concepts of specifics areas. In this way, our models can be used by a system providing help in configuration to describe image processing objectives and to plan their resolution.

Keywords : Image processing, Modeling, Visual cues, Information transformation, Expression language, Planning.

Je remercie Monsieur Serge CASTAN, Professeur à l'Université Paul Sabatier, qui a bien voulu m'accueillir dans son équipe.

Je tiens à remercier particulièrement

Mesdames Marinette REVENU,
Professeur à l'Institut des Sciences de la Matière et du Rayonnement de Caen, Responable Scientifique du Thème Image du Groupe de Recherche en Informatique, Image, Instrumentation de Caen

et Monique THONNAT,
Directeur de Recherche à l'Institut National de Recherche en Informatique et en Automatique, Responsable Scientifique du projet ORION à l'Unité de Recherche de Sophia-Antipolis,

ainsi que Monsieur Marc BERTHOD
Président du Comité des Projets de l'Unité de Recherche de Sophia-Antipolis, Responsable Scientifique du projet Pastis

pour l'intérêt qu'ils ont bien voulu porter à ce travail en acceptant la tâche de rapporteurs.

Je tiens aussi à remercier Messieurs Henri FARRENY et Gérard PADIOU, Professeurs à l'École Nationale Supérieure d'Électrotechnique, d'Électronique, d'Informatique et d'Hydraulique de Toulouse, d'avoir porter un autre regard sur ce travail que celui de spécialistes de l'image.

Je tiens à exprimer ma gratitude à Monsieur Patrice DALLE, Maître de Conférences à l'Université Paul Sabatier, pour toutes les discussions que nous avons pu avoir autour de petits schémas ! Il m'a donné le goût du traitement et de l'analyse d'images et la curiosité pour la recherche.

Mesdames, Messieurs, je vous remercie de l'honneur que vous me faites de siéger à ce jury.

Je remercie les membres de l'équipe Traitement et Compréhension d'Images -je ne les cite pas de peur d'en oublier- pour leur soutien moral tout au long de ces années et les riches discussions informelles. C'est promis, je compterai plus souvent : 1, 2, 3, ...

Je remercie J.-P. CECCATO et J.-P. BARITAUD pour le travail de reprographie de ce document ainsi que pour leur sympathie.

Je ne dois pas oublier les Éditions Gallimard pour le droit de reproduction qu'ils m'ont accordé.

Enfin, je remercie toute ma famille et mes amis pour leur soutien de tous les instants.
À mon épouse Marie-Josèphe et mes enfants chéris Eulalie, Solène et Théophile, merci pour votre patience, votre amour... mais les mots sont bien trop pauvres !

À Eulalie, Solène, Théophile
et Marie-Josèphe

À mes parents

Ku ñuy diri,
doo taamu fu ñu lay jaarale

Celui qu'on traîne
ne choisit pas l'endroit où on le fait passer

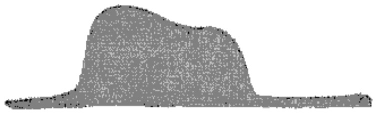
-Proverbe Wolof-

Préambule

...

On disait dans le livre : « Les serpents boas avalent leur proie tout entière, sans la mâcher. Ensuite ils ne peuvent plus bouger et ils dorment pendant les six mois de leur digestion. »

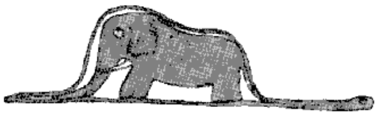
J'ai alors beaucoup réfléchi sur les aventures de la jungle et, à mon tour, j'ai réussi, avec un crayon de couleur à tracer mon premier dessin. Mon dessin numéro 1. Il était comme ça :



J'ai montré mon chef-d'œuvre aux grandes personnes et je leur ai demandé si mon dessin leur faisait peur.

Elles m'ont répondu : « Pourquoi un chapeau ferait-il peur ? »

Mon dessin ne représentait pas un chapeau. Il représentait un serpent boa qui digérait un éléphant. J'ai alors dessiné l'intérieur du serpent boa, afin que les grandes personnes puissent comprendre. Elles ont toujours besoins d'explications. Mon dessin numéro 2 était comme ça :

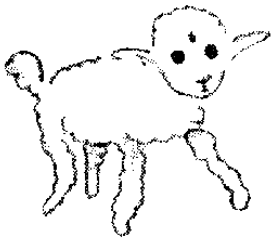


Les grandes personnes m'ont conseillé de laisser de côté les dessins de boas ouverts ou fermés, et de m'intéresser plutôt à la géographie, à l'histoire, au calcul et à la grammaire. C'est ainsi que j'ai abandonné à l'âge de six ans, une magnifique carrière de peintre.

...

...

Alors, j'ai dessiné.



Il regarda attentivement, puis :
–Non ! Celui-là est déjà très malade. Fais-en un autre.

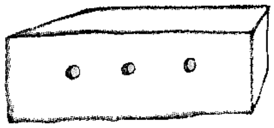
Je dessinai :

Mon ami sourit gentiment, avec indulgence :
–Tu vois bien... ce n'est pas un mouton, c'est un bélier. Il a des cornes...

Je refis mon dessin :

Mais il fut refusé, comme les précédents :
– Celui-là est trop vieux. Je veux un mouton qui vive longtemps.

Alors, faute de patience, comme j'avais hâte de commencer le démontage de mon moteur, je griffonnai ce dessin-ci.



Et je lançai :
– Ça c'est la caisse. Le mouton que tu veux est dedans.
Mais je fus bien surpris de voir s'illuminer le visage de mon jeune juge :
– C'est tout à fait comme ça que je le voulais ! crois-tu qu'il faille beaucoup d'herbe à ce mouton ?
–Pourquoi ?
–Parce que chez moi c'est tout petit...
–Ça suffira sûrement. Je t'ai donné un tout petit mouton.
Il pencha la tête vers le dessin :
–Pas si petit que ça... Tiens ! Il s'est endormi...
Et c'est ainsi que je fis la connaissance du petit prince.

...

Le Petit Prince

Antoine de Saint-Exupéry

Les dessins des enfants ont cela de merveilleux qu'ils nous bousculent dans nos références tant graphiques que spatiales, temporelles ou chromatiques.

Il faut noter que ce mélange de références temporelles et spatiales se trouve être présent dans les œuvres des peintres qui, inspirés de l'art nègre, lancent la première phase, dite analytique, du mouvement cubiste. Nous pouvons ainsi citer Georges Braque -*Maisons à l'Estaque (1908.Kunstmuseum.Bern)*- Pablo Picasso -*Portrait d'Ambroise Vollard (1910.Musée Pouchkine.-Moscou)*- ou encore Fernand Léger -*La noce (1911-12.Musée d'Art Moderne.Paris)*.

Il m'est ainsi souvent arrivé de me sentir désarmé devant mes enfants me commentant leurs dessins. Les explications qu'elles fournissent témoignent alors de la trivialité, dans leur esprit, de la représentation : lien entre dessin et concept dessiné. Le ton employé montre alors leur étonnement face à mon manque évident de discernement !

Et Solène de dessiner « *La lune dans le ciel, haut, haut, on ne peut pas l'attraper* » ou Eulalie de dessiner un personnage, dans plusieurs situations successives, sur un même dessin, pour raconter une histoire...

Cette capacité des enfants à représenter spontanément des objets ou des situations en excluant du dessin tout référentiel est parfaitement illustrée dans ces deux extraits du Petit Prince.
Dans le premier extrait, nous imaginons le petit Antoine de Saint-Exupéry avec ses rêves de Forêt Vierge plein la tête, son étonnement à l'écoute de l'histoire du boa avalant un fauve. Il transpose alors ses idées, ses réflexions et ses connaissances des « *aventures de la jungle* » sur le dessin du *boa fermé*.

«... Je leur ai demandé si mon dessin leur faisait peur. ...»

Hors de tout ce contexte, comment « *un chapeau ferait-il peur ?* »
Avec son second dessin, il modifie son propre système de représentation du concept : « *boa qui digérait un éléphant* ». Ce dessin du *boa ouvert* ne semble pas le satisfaire : il est seulement fait « *pour que les grandes personnes puissent comprendre. Elles ont toujours besoin d'explication* ».

Cette histoire montre tout d'abord le fait qu'un même dessin -dessin numéro 1- peut être interprété de plusieurs façons selon le référent conceptuel utilisé. Le dessin en lui-même n'est pas remis en cause, l'erreur d'interprétation provient uniquement de l'absence d'évocation du contexte dans lequel il a été exécuté.
D'autre part, il montre comment une modification du système de représentation -*ouverture* du boa- peut influer sur la perception des concepts.

Le second extrait va plus loin. Il illustre le fait qu'un concept -le *mouton*- puisse être décrit par une forme tout à fait à l'opposé de ce que l'on peut concevoir pour sa représentation - la caisse. Antoine de Saint-Exupéry représente une caisse et indique que « *Le mouton que tu veux est dedans.*». Il fait ainsi une distinction entre l'animal et la boîte le contenant, alors que le Petit Prince fait de la caisse et du mouton deux représentations tout à fait cohérentes d'un même concept *mouton* : la caisse EST le mouton qu'il désirait. Ceci est évoqué de façon simple dans son acceptation immédiate du dessin, sans poser de questions, sans faire de remarque ; plus tard il jugera même la taille de son nouveau compagnon et lui donnera vie :

« -Pas si petit que ça... Tiens! Il s'est endormi ...»

Comme nous venons de le voir, il est facile pour un enfant d'expliquer sa vision des choses par un simple dessin. Il va pour cela à l'essentiel, au support du concept à représenter -Solène dessinait ainsi une petite tâche de chacune des couleurs qui étaient à sa disposition pour évoquer les lever de soleil que nous admirions en partant à l'école. Le plus difficile en est l'interprétation à cause très souvent de nombreuses omissions liées au contexte de son exécution. Parallèlement, il est souvent facile d'emprunter à un domaine particulier des concepts spécifiques, pour en représenter d'autres dans un domaine totalement différent.

Le travail d'acquisition et de représentation des connaissances d'experts humains, pour une implantation dans un système d'interprétation et d'analyse d'images, doit tenir compte de cette problématique : la totalité des informations extrinsèques des images à interpréter ne peut être décrite ; il faudra donc tenir rigoureusement compte du contexte dans lequel est effectuée cette acquisition pour représenter un objectif d'analyse, d'autant qu'une bonne partie des processus d'interprétation, utilise des informations non purement visuelles tels que le centre de gravité ou le degré de compacité.
Une image numérique est obtenue à partir d'une scène réelle par un processus faisant intervenir l'éclairement, les objets et un capteur (son système optique et son dispositif de numérisation). Ces éléments ainsi que leur disposition naturelle ou maîtrisée par une mise en scène prédéfinie, influencent la description générique des concepts de l'application (parler d'objets sombre dépendra de l'éclairement et des propriétés radiométriques de l'objet considéré), mais aussi des caractéristiques géométriques de l'objet, généralement 3D, dans sa projection image 2D effectuée par le capteur (surfaces cachées, par exemple). Elle contient donc des informations géométriques et radiométriques en plus des informations spécifiques au domaine de l'expertise. Les raisons de cette impossibilité à réunir toutes les informations sont essentiellement de deux ordres : soit il est matériellement impossible de les représenter selon un ou des modèles établis et cohérents entre eux, ou de les traiter dans des temps acceptables, soit il est tout simplement impossible d'en établir une liste exhaustive.

Introduction

Les possibilités du traitement d'images ont conduit les experts de nombreux domaines -médical, télé-surveillance, contrôle de qualité, classification, etc- à solliciter l'aide des "traiteurs" d'images pour résoudre leurs problèmes particuliers. Ainsi, des systèmes dédiés ont été élaborés, permettant de fournir des solutions à des objectifs spécifiques. Les recherches menées dans le cadre du traitement d'images et autour des concepts de l'analyse et de l'interprétation ont alors vu se développer une nouvelle famille de systèmes, plus complexes, permettant de générer des systèmes dédiés d'interprétation. La principale difficulté dans la mise au point de ces générateurs est d'établir des liens précis entre les concepts des domaines d'application et ceux du traitement d'images.

Cependant, leur objectif final est de pouvoir générer des programmes résolvant un problème d'un domaine précis. De ce fait, la conception de ces systèmes demande une modélisation des entités -données, opérateurs- manipulées par le traitement d'images suffisamment fine pour pouvoir construire des outils fiables de résolution de problème.

Dans la plupart des travaux relatifs au traitement et à l'analyse des images, les entités sont modélisées en fonction des spécificités des algorithmes ou des systèmes mis en œuvre. Ces représentations permettent de développer des outils performants de traitement d'images. En effet, elles ont été développées à partir du cahier des charges du module de résolution utilisé : tableau noir, système expert, graphe conceptuel, etc.

Si par contre nous nous plaçons dans le cadre de la génération de plan de traitement d'images, nous avons besoin d'un niveau d'abstraction supplémentaire, permettant de représenter et de manipuler les opérateurs eux-mêmes. En effet, l'automatisation ou l'aide à la configuration de séquences d'opérateurs de traitement d'images, implique que nous puissions déterminer l'opération à effectuer à chaque étape du processus -sélection des données, choix de la fonction à appliquer, choix des opérateurs implémentant cette fonction, détermination des paramètres. De même, les processus de contrôle de l'exécution de la chaîne produite doivent être à même d'analyser la donnée produite et de déterminer les actions à exécuter, que ce soit pour ajuster les résultats, sélectionner des séquences particulières ou proposer une correction en termes de modification d'objectif.

Nous nous sommes donc attaché à déterminer un formalisme rigoureux et précis permettant de développer des modèles de données et d'opérateurs de traitement d'images. Nous verrons que ceux-ci s'expriment en termes d'indices visuels et de transformations d'information.

La première partie de ce document est consacrée à la mise en évidence des besoins d'un formalisme précis. Nous développerons les aspects de la compréhension du traitement d'images pour une utilisation interactive, pour automatiser les traitements et pour exprimer la dualité de ceux-ci dans le cadre d'une interprétation d'image. Nous donnerons alors les différents éléments de modélisation au travers des problèmes que soulève l'analyse d'un graphe d'opérateurs. Enfin, nous montrerons comment différents systèmes de résolution d'objectifs intègrent de tels modèles en faisant une place particulière au système ExTI qui sert de plateforme d'intégration de nos modèles. Nous irons enfin chercher dans l'étude des grammaires, les éléments qui nous permettront de caractériser notre propre langage de description de données et d'expression d'objectifs.

Dans une seconde partie, nous proposons les modèles de donnée, ce Langage d'Expression d'Objectifs (LEO) et le modèle d'opérateurs. Ces trois éléments forment un noyau complet et rigoureux de descriptions des entités de traitement d'images manipulées lors d'une résolution d'objectif. Nous montrerons alors en quoi le formalisme proposé peut être utilisé pour la résolution d'objectifs, et pour mieux décrire le traitement d'images. Nous poserons alors les contraintes spécifiques du traitement d'images pour développer un processus de planification symbolique et nous montrerons comment ce noyau permet d'intégrer des outils de contrôle d'exécution et de supervision.

La troisième partie sera alors consacrée à la mise en œuvre et à l'exploitation de ces modèles dans le système de planification automatique de chaînes de traitement d'images, ExTI. Nous montrerons alors l'implantation de ces modèles, et détaillerons les intérêts et les principes de la simplification des expressions de LEO. Après avoir développé le rôle essentiel du module d'appariement d'expressions et ses spécificités, nous présenterons le module de réutilisation de plan que nous avons développé, permettant ainsi d'exploiter la notion de vocabulaire exécutable. Enfin, nous terminerons cette partie en rapportant des résultats intéressants relatifs à divers points développés.

Les annexes contiendront des compléments d'information sur des éléments ponctuels ou des bilans généraux comme les notations utilisées.

Première partie

De l'utilité d'un formalisme

Chapitre 1

Comprendre le Traitement d'images

Les utilisations du traitement d'images se multiplient de façon considérable dans le domaine industriel (robotique, contrôle de qualité, ...), spatial, médical et s'étendent à de nouveaux domaines, avec la vision de dispositifs autonomes, la surveillance, l'accès aux banques de données images, etc ... Ce phénomène a provoqué le développement de recherches de plus en plus complexes, tant au niveau des modèles et méthodes mathématiques que de l'algorithmique. Enfin, de nombreux produits sont maintenant commercialisés, allant de l'application spécialisée à l'environnement de développement. Avec le développement constant de nouveaux opérateurs, le traitement d'images est devenu un domaine scientifique difficile à appréhender dans toutes ses dimensions et complexe à utiliser. Il est donc nécessaire d'étudier les entités de traitement d'images en ayant le souci de fournir des aides adaptées aux différents types d'utilisateurs. Dans ce chapitre, après avoir évoqué les différentes utilisations du traitement d'images et les types d'aides qui ont été proposées, nous essayons de définir un cahier des charges des fonctionnalités à mettre en œuvre, basées sur la compréhension et la manipulation des entités du traitement d'images. Nous présentons alors les différentes méthodes de modélisation de ces entités et nous montrons qu'une modélisation centrée sur les manipulations de l'information véhiculée dans les structures de données procure un formalisme précis. Ce formalisme peut être exploité par différents systèmes d'apprentissage ou d'exploitation du traitement d'images, dont nous donnerons les principes plus loin.

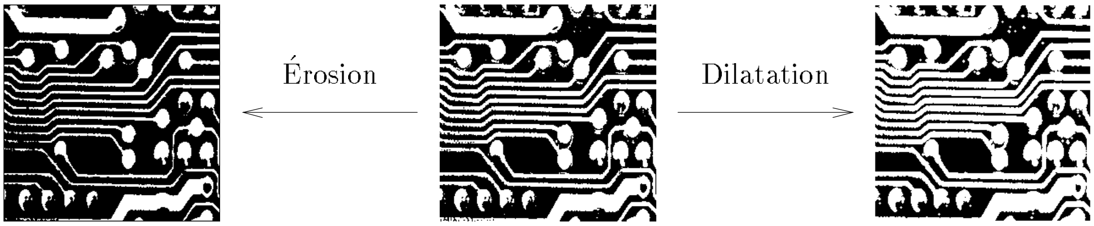
1.1 Pour une utilisation interactive

1.1.1 Les caractéristiques

Une image numérique est obtenue à partir d'une scène réelle par un processus faisant intervenir l'éclairement de la scène, les objets (leur nature et leur géométrie), un capteur (son système optique et son dispositif de numérisation). Elle contient donc des informations géométriques et radio-métriques. Le traitement a pour but d'extraire des informations concernant les objets de la scène correspondant à l'image. Il consiste à construire des entités représentant les concepts recherchés, en inversant plus ou moins complètement le processus de formation de l'image. Les opérateurs utilisés implémentent imparfaitement ces fonctionnalités du processus de reconstruction, à cause des difficultés introduites par la nature discrète des donnés, le caractère généralement mal conditionné du système, la nature non inversible de la projection perspective, le bruit présent dans les données et le caractère incertain et imprécis des informations *a priori* sur la scène. Aussi les bibliothèques de traitement d'images comportent un grand nombre d'opérateurs concurrents pour chaque famille de traitement, et la résolution d'une application particulière peut souvent être abordée de différentes façons difficilement comparables. En outre, ces solutions sont encore le plus souvent élaborées par un processus d'essais-erreurs, l'ingénierie du traitement d'images n'en étant qu'à ses débuts.

1.1.2 L'apprentissage

Suivant ses objectifs, un utilisateur peut apprendre le traitement d'images afin de pouvoir utiliser un logiciel plus ou moins spécialisé ou dédié à son application. Il s'agit donc pour lui de comprendre les fonctionnalités et les opérateurs qu'il devra manipuler. La comparaison des opérateurs répondant à une même fonctionnalité se fait, parfois à l'aide d'outils spécifiques, en examinant les résultats qu'ils produisent sur les données ou leur expression algorithmique, parfois sur la base de fonctions mathématiques. Mais est-il toujours facile pour un néophyte de comparer deux opérateurs aussi simples que la dilatation et l'érosion (fig.1.1) ou l'ouverture et la fermeture, pour rester dans la morphologie mathématique, ou deux séries de filtres tels ceux de Sobel et Prewitt. . . ?



$X \ominus B = \{x | B_x \subset X\}$ X $X \oplus B = \{x | B_x \cap X \neq \emptyset\}$

FIG. 1.1 - *Comparaison Dilatation / Érosion*

De même, la compréhension du travail réalisé par un opérateur ou un enchaînement d'opérateurs se fait généralement en examinant l'effet produit sur des données. Une séquence d'opérateurs de traitement d'images pouvant être considérée comme un macro-opérateur, l'utilisateur doit pouvoir décomposer un tel opérateur, c'est à dire comprendre les étapes de transformations et de combinaisons qu'il réalise. Inversement, connaissant l'effet de chaque opérateur élémentaire, il doit pouvoir synthétiser leurs traitements individuels. La difficulté est alors de pouvoir exprimer les transformations effectuées sur la donnée initiale par cette séquence d'opérateurs élémentaires, sans pour cela faire référence aux transformations intermédiaires. Ceci afin d'obtenir une compréhension "compilée" de l'opérateur qu'ils composent.

Le caractère visuel est donc important pour la validation des résultats et l'aide à la compréhension. Cette caractéristique sera exploitée pour montrer que le formalisme que nous proposons se prête bien au développement d'outils d'aide à la compréhension et l'apprentissage en se basant sur des descriptions de données et d'opérateurs en terme d'indices visuels. Le système qui l'exploite rend compte précisément de l'ensemble des opérations qu'il a effectuées : description des structures

de données d'entrée et de sortie, transfert de l'information qu'elles supportent, effet produit dans les images.
Il doit rendre compte aussi de la différence de traitement et de résultat réalisé par ces opérateurs à partir d'une donnée commune. Cette information doit d'une part permettre de mieux comprendre la spécificité de chaque opérateur et d'autre part, faciliter la sélection du plus pertinent dans un contexte applicatif donné.

1.1.3 La construction d'applications

Cet utilisateur peut ensuite souhaiter construire une application (séquence d'opérateurs), en intégrant des opérateurs d'une bibliothèque pour résoudre un problème d'analyse ou d'interprétation ou simplement pour modifier les caractéristiques de l'image. Aider à la conception d'une chaîne de traitement consiste donc à fournir les éléments nécessaires à la construction itérative d'un graphe d'opérateurs. On doit donc fournir un moyen permettant à l'utilisateur d'établir plus facilement une correspondance entre le concept qu'il veut extraire et dont il voit une instance dans la donnée image qu'il manipule et le traitement qui va transformer cette image de telle sorte qu'un élément de la donnée de sortie représente ce concept. Nous verrons que ceci peut être réalisé en exprimant les concepts en termes d'indices visuels et en décrivant symboliquement les opérateurs comme constructeurs de ces indices. Il s'agira alors de faire correspondre leur description -le système apportant une aide à la formulation du problème- à la description du concept recherché. De ce fait, actuellement, pour exploiter les ressources d'une telle bibliothèque d'opérateurs, l'utilisateur doit nécessairement appréhender le traitement d'images, même si des outils d'aide -contextuelle ou en ligne- sont à sa disposition.

En effet, l'approche naturelle pour l'utilisateur -approche descendante (fig.1.2)- est de déterminer et expliciter le domaine de l'application (modèle des objets, conditions d'interprétation et objectif d'interprétation) puis élaborer des descriptions d'objets génériques pour parvenir aux entités de traitement d'images. Ceci défini, il détermine les opérateurs génériques. Rien ne garantit alors qu'il pourra sélectionner les opérateurs réels et donc élaborer la chaîne de traitement d'images répondant à son objectif.

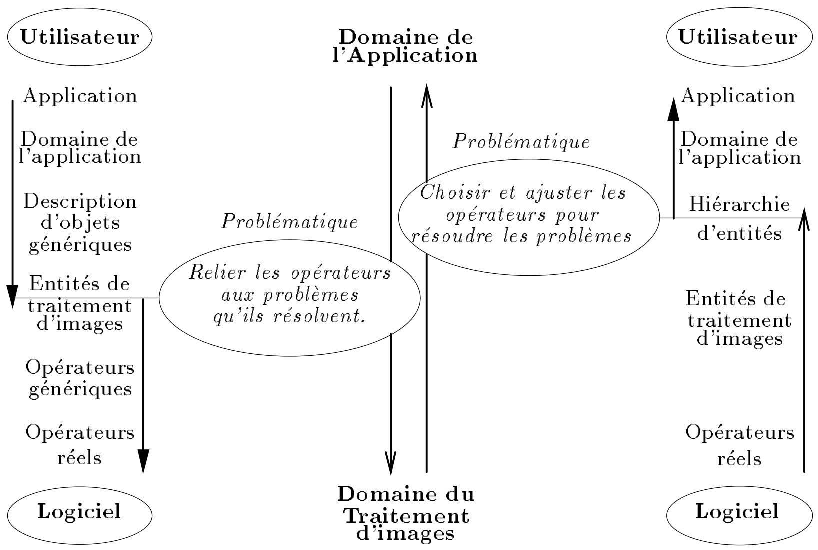


FIG. 1.2 - *Approches Descendante / Ascendante*

L'approche ascendante (fig.1.2), qui construit des entités de plus en plus complexes lève cette incertitude. Cependant, l'utilisateur reste confronté à trois types de difficultés :

1. Raisonner en termes d'opérateurs de traitement d'images.
2. Configurer la chaîne, c'est à dire à chaque étape être capable de choisir un objectif local *(sous objectif)*, la fonction à exécuter, l'opérateur réalisant cette fonction, les valeurs des paramètres de l'opérateur et enfin être capable d'évaluer les résultats.
3. Justifier les résultats, c'est à dire garantir que la chaîne fournit le bon résultat dans d'autres cas que celui qui a servit à la générer.

Comme précédemment, ceci se fait par essais-erreurs en observant les résultats et en effectuant des mesures sur les données produites ; ces mêmes données ne fournissant aucune indication sur les causes d'échec -type de fonctionnalité, choix d'opérateur, ajustement de paramètres à remettre en cause.

L'approche de génération d'application spécifiques implémentée dans ExTI, pour lequel nous avons développé le formalisme présenté ici, consiste à considérer l'élaboration de la chaîne de traitement d'images dans un processus de coopération Homme-Machine basé sur un formalisme bas-niveau de description d'image pouvant être couvert par un langage plus représentatif du domaine (fig.1.3). Dans cette approche, ascendante, constructive, l'utilisateur n'intervient que dans

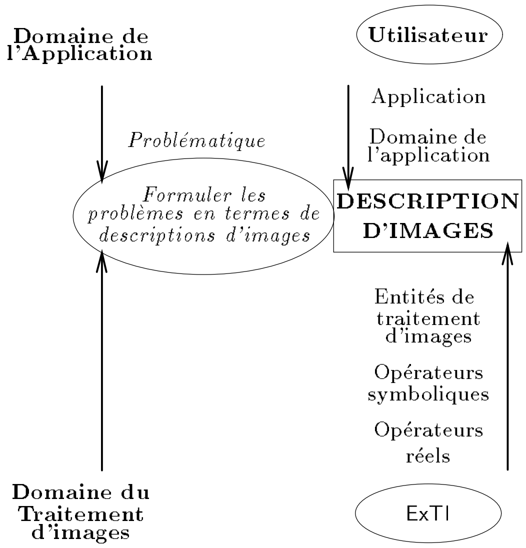


FIG. 1.3 - *Approche ExTI*

la formulation de l'objectif en termes de description d'images. Cette formulation fait l'objet d'une recherche sur des interfaces spécifiques de manipulation des descripteurs [Cap95]. Le système ExTI planifie la chaîne d'opérateurs symboliques par reconstruction de la description de l'objectif. Ces opérateurs symboliques correspondent aux descriptions des transformations qu'assurent les opérateurs réels sur les données. Comme nous le voyons, l'intervention de l'utilisateur est réduite, et fait appel essentiellement à ses connaissances et compétences sur le domaine de l'application.

1.1.4 Les environnements

Outre les systèmes spécialisés, dont nous ne parlerons pas et qui contiennent leurs propres systèmes d'aide, et les langages spécialisés ou étendus pour la programmation du traitement d'images (Olive [Fli90]), nous trouvons plusieurs familles de produit :

– les systèmes interactifs de traitement d'images, tels que Visilog [Noe92],Optimas [IMA], comprenant une bibliothèque d'opérateurs de traitement et d'analyse d'images, des utilitaires et une couche interactive pour la présentation des opérateurs, leur sélection, leur enchaînement et pour la présentation des résultats. Cet environnement peut comporter un langage visuel (Khoros [Uni92]) qui facilite la manipulation des opérateurs mais pas leur compréhension ni leur sélection.
– les environnements de développement, construits sur un langage facilitant le prototypage (Smalltalk) et dans lequel sont re-déclarés les opérateurs d'une bibliothèque de traitement d'images.

Certains de ces environnements intègrent des aides en ligne décrivant les fonctionnalités de chaque opérateur ou illustrant leur effet sur des petites images.
D'autre part, la recherche sur les aides à la configuration de chaînes de traitement ([Cle93], [Clo93], [Cap94]), à la réutilisation de programmes ou au contrôle d'exécution (Workshop KBUP [KBU95]), sur lesquels nous reviendrons au chapitre 2, est en plein développement et vise à faciliter l'exploitation du traitement d'images dans des applications.

1.2 Pour une automatisation des traitements

Cette détermination d'un formalisme précis peut aussi fournir des éléments pertinents dans le cadre d'une mise au point d'environnements et d'outils pour le développements d'applications de traitement d'images. Nous définirons une application comme un graphe d'opérateurs. Il résout un problème particulier dans un domaine spécifique tel que l'inspection de circuits imprimés, l'imagerie médicale ou l'analyse d'images aériennes.
Ainsi nous focaliserons notre recherche sur la modélisation des entités de traitement d'images car nous voulons être capable d'effectuer ces traitements de façon formelle, pour mieux choisir les opérateurs, planifier des séquences ou pour prédire leurs résultats et décrire les informations présentes dans les données produites. Dans un premier temps nous nous limitons aux opérateurs de traitement d'images vus en tant qu'entités transformant des informations, c'est à dire sous un angle purement symbolique. Nous verrons ensuite, à propos de la supervision de l'exécution, qu'il faudra compléter ces modèles pour intégrer des opérations de mesures et de contrôle.

D'après l'analyse précédente nous voyons qu'un système d'aide doit manipuler une représentation particulière des entités de traitement d'images. Cette modélisation doit permettre à la fois de comprendre et de planifier une chaîne de traitement. Elle doit donc rendre compte du traitement effectué, de façon interne sur les données et de façon externe en terme d'effet produit, et permettre de relier un opérateur et un objectif de traitement d'images. Différents modèles de représentation ont été proposés, généralement dans le cadre de la conception de modules ou de systèmes d'interprétation d'images (planifieurs, systèmes experts en traitement d'images) ou de leur exploitation (contrôle d'exécution).
Pour développer des systèmes généraux de conception de systèmes dédiés -nous pouvons citer OCAPI [Moi95c] et son dérivé temps réel PLANETE, BORG [Clo94], LLVE [Mat89], ou ExTI [Ing92]- les modèles utilisés vont devoir recouvrir et intégrer des connaissances du traitement d'images pour faciliter l'intégration de connaissances spécifiques du domaine de l'application. La difficulté consiste à établir le lien entre les processus mis en œuvre par l'expertise du photo-interprète pour la détection des entités de la donnée et les processus de traitement d'images permettant d'automatiser

cette interprétation ; les deux domaines (domaine d'application et traitement d'images) étant généralement très différents quant aux termes et concepts utilisés. Ces connaissances du traitement d'images peuvent être réparties entre quatre classes :

1. Les connaissances sur les données.
   Elles regroupent, outre des informations particulières dues à des modélisation particulières inhérentes à des traitements spécifiques -algorithmes rapides, etc-, toutes les notions classiques de types de données que l'on rencontre dans les systèmes interactifs, mais peuvent correspondre aussi aux caractéristiques de codage -0 255/0 1 pour les représentations d'images binaires, RGB/RLE pour les images couleur, etc- aux structures vectorielles ou raster des données images ou à des informations plus simples comme leur taille. Il suffit d'examiner les différents formats des fichiers image et les structures des entêtes de ces formats pour avoir une idée assez globale des différentes informations liées aux données.
2. Les connaissances sur les techniques de traitement et d'analyse.
   Elles sont liées aux méthodes développées en ingénierie du traitement d'image et relatives aux phases de prétraitements -filtrage du bruit, détections de contours, segmentation, etc- mises en œuvre successivement dans un processus global d'analyse d'images.
3. Les connaissances sur les opérateurs.
   Elles correspondent à des informations sur les conditions d'application des opérateurs en termes de données d'entrée et de sortie, -tant dans un cadre général d'utilisation que dans un cadre particulier- à un endroit particulier de la chaîne de traitement -opérateur de reconstruction, par exemple. Elles correspondent aussi à des caractéristiques spécifiques de certains opérateurs telles que l'idempotence -nous pouvons citer l'ouverture ou la fermeture morphologique-, la commutativité, etc. Enfin, nous trouvons des informations sur le rôle de l'opérateur dans la séquence, et ses caractéristiques par rapport à la fonctionnalité qu'il est sensé réaliser, en association ou non avec d'autres opérateurs.
4. Les connaissances sur les enchaînements d'opérateurs.
   Elles sont relatives à la définition éventuelle d'un macro-opérateur ou d'un enchaînement d'opérateurs par une séquence plus élémentaire déterminée selon un critère particulier relevant, par exemple, d'une heuristique de stratégie de planification. Mais elles peuvent correspondre aussi à des enchaînements spécifiques se traduisant par un effet symétrique de certains opérateurs tels que les opérateurs logiques -et, ou, non, etc.

1.3 Pour exprimer une dualité de traitement

La différence évoquée précédemment et relative aux connaissances et expertises des domaines du traitement d'images d'une part et des domaines dans lesquels sont développées les applications d'autre part nous amène à considérer les opérateurs de manipulation d'images sous les deux points de vue : traitement des concepts, traitement des données. Du point de vue d'un utilisateur, expert dans un domaine, les opérateurs de traitement d'images travaillent de manière plus ou moins abstraite ou plus ou moins complexe sur des entités particulières de l'image qu'il peut identifier par des termes spécifiques inhérents à son expertise. Abstraite, car le processus, voire l'algorithme, mis en œuvre par l'opérateur importe peu, seuls les résultats produits sont importants ; et généralement les différences entre les effets d'opérateurs voisins -Sobel, Prewitt- ne sont pas prises en compte. Complexe, car l'analyse d'un résultat, *a fortiori* lorsqu'il s'agit d'un résultat intermédiaire, oblige cet expert à interpréter la donnée produite dans des termes de son domaine.

De façon plus générique, nous pouvons considérer les opérateurs de traitement d'images comme des outils de manipulation des constituants de l'image, ces derniers n'étant plus des pixels, contours, régions ou composantes connexes mais des concepts du domaine de l'application. Si nous prenons l'exemple de la figure 1.4, nous pouvons considérer l'opérateur de seuillage comme un outil créant le concept "objets clairs" (traitement des concepts) à partir d'une partition de l'histogramme des valeurs des pixels (traitement de la donnée).

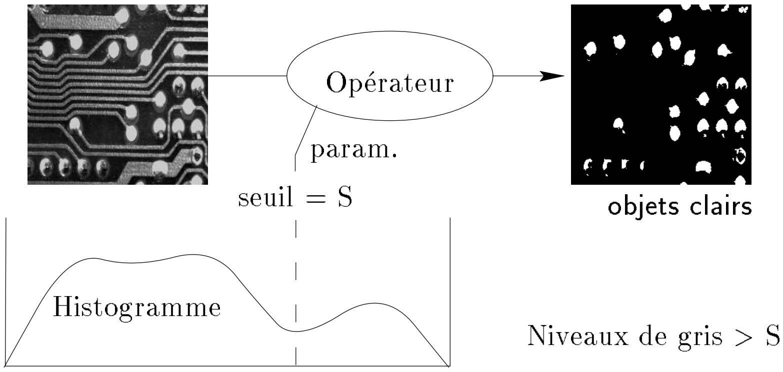


FIG. 1.4 - *Constructeur de concept - 1*

Mais nous pouvons aller plus loin en considérant le traitement de ce même opérateur avec la même valeur de paramètre sur la même image d'entrée, mais dans deux contextes différents. Il peut construire le concept "objets clairs" que nous venons de voir, mais peut aussi créer le concept "N points les plus clairs" (fig.1.5). Dans un premier temps le paramètre S correspond à un degré d'intensité définissant la frontière binaire entre clair et sombre. Dans un second temps S est déterminé en fonction de la mesure d'aire N sur l'histogramme.

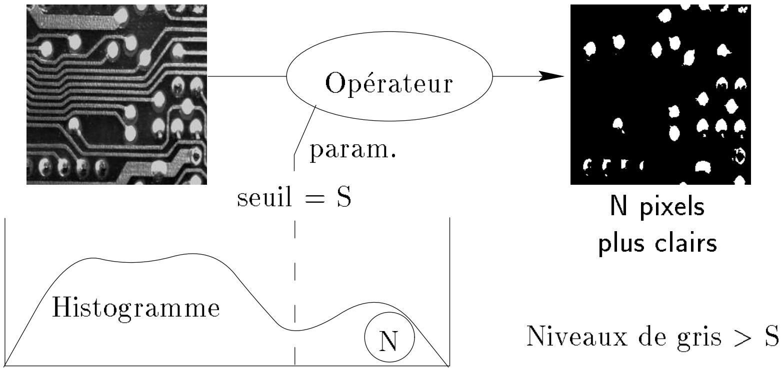


FIG. 1.5 - *Constructeur de concept - 2*

Ceci pour expliciter le fait qu'au niveau du traitement des concepts, un même opérateur (opérateur + paramètres) peut exprimer des concepts différentes tout en effectuant le même traitement sur les données, bien évidemment.

Un autre aspect intéressant à relever est l'utilisation d'un même opérateur à plusieurs endroits de la chaîne de traitements. Là encore, cet opérateur va construire ou manipuler des concepts différents.

Ainsi, dans la chaîne reportée dans la figure 1.6, le premier opérateur de seuillage -*Sll* (1)- effectue une partition des pixels de la donnée selon un seuil correspondant à une valeur d'intensité.

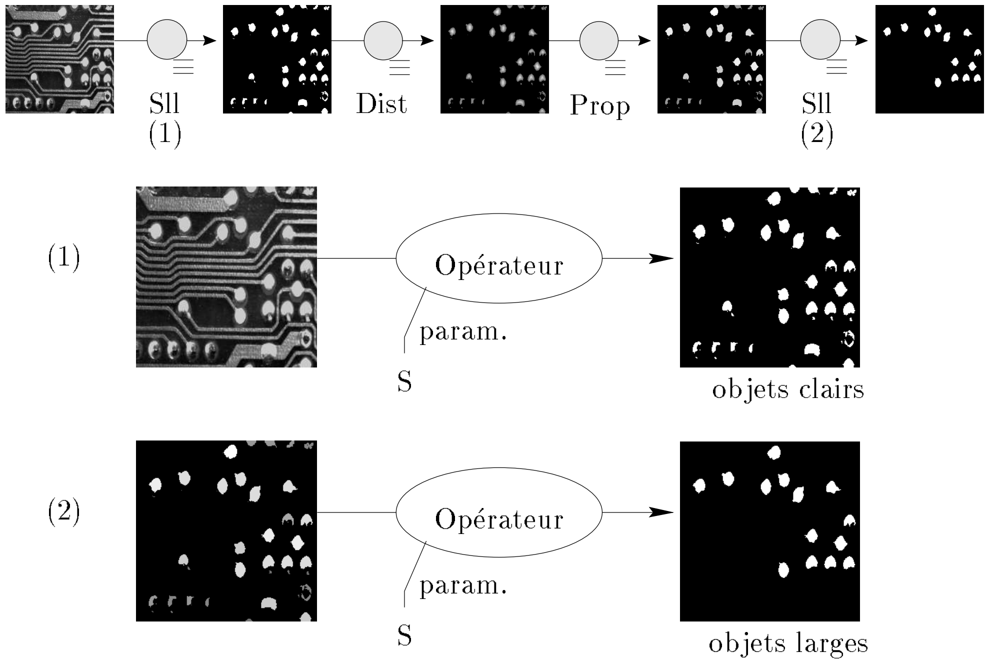


FIG. 1.6 - *Constructeur de concept - 3*

Après une mesure des valeurs de distance -*Dist*- effectuée sur les composantes connexes ainsi formées et la propagation -*Prop*- de ces mesures dans les composantes elles-même, le second opérateur de seuillage -*Sll* (2)- effectue une sélection de ces composantes selon une valeur de paramètre liée au critère de distance.

Ainsi donc, le formalisme que nous cherchons à obtenir doit recouvrir ces notions importantes de transformation de concepts, pour nous permettre de développer un système de planification d'objectif sans que celui-ci ne soit dédié à un domaine particulier, c'est à dire que ses composants principaux -modèles de données et d'opérateurs- ne soient représentés en fonction du contexte de l'application. La transformation des concepts se faisant en parallèle de la transformation des données, tout au long de la chaîne des traitements, les modèles que nous proposons se doivent d'être rigoureux pour, d'une part, décrire le contenu des données à un moment précis de cette chaîne, et d'autre part, décrire en terme de transformation d'information comment ces données ont été obtenues ; c'est ce que nous appellerons l'"histoire" de la donnée.

1.4 Éléments de modélisation

Pour implémenter toutes ces notions que nous venons de voir, divers modèles de la donnée de traitement d'images ont été proposés, la plupart du temps au regard des outils de modélisation du traitement d'images lui-même et des intérêts de ces modèles.

1.4.1 Principaux centres d'intérêt

Beaucoup de travaux ont été effectués pour essayer de modéliser le comportement et le raisonnement d'un utilisateur du traitement d'images ou pour faciliter la mise au point de ses programmes. L'objectif général et commun à ces recherches est d'obtenir des caractéristiques comportementales de résolution d'objectif. À partir de là, il peut être possible d'expliciter des règles de manipulation des outils mis à la disposition de cet utilisateur ou des règles de résolution de problèmes d'analyse. Toutes les approches tendent à rendre compte dans leurs propres termes (algorithmiques, fonctionnels, par types, flux de données, informations symboliques, ...) des manipulations d'informations effectuées par le traitement d'images. Ainsi, de manière plus ou moins effective et explicite selon les cas, quatre centres d'intérêts se dégagent :

Expliciter ou modéliser le raisonnement
Expliciter ou modéliser le raisonnement, donnent la possibilité à un utilisateur de mieux se servir des outils proposés et de mieux appréhender le traitement d'images.
Dans ce sens, des systèmes comme OCAPI [Cle93] ou CONNY [Lie86, Lie92] permettent d'intégrer, lorsqu'elle existe, une expertise de résolution d'objectif dans un domaine précis sous la forme de règles d'application d'opérateurs. Parmi les systèmes conversationnels, certains fournissent à l'utilisateur des aides à la configuration de leur environnement, ou encore intègrent des modules de gestion dynamique et adaptatives de menus [Bel85b]. Le système ExTI [Dej96d] quant à lui dispose de modèles d'opérateurs (voir chap.3, 2^e^ partie) permettant d'expliciter les transformations d'informations effectuées à l'aide d'un langage particulier (voir chap.2, 2^e^ partie) permettant de mieux approcher le traitement d'images [Dej96c, Dej96b].

Faciliter la mise au point
La mise au point interactive de chaînes d'opérateurs est facilitée en explicitant les erreurs et en fournissant des outils de visualisation et de correction par la possibilité de manipuler textuellement la chaîne de commandes au travers d'un script.
Ces possibilités sont généralement mises en œuvre par l'introduction de langages visuels [Cha86, Miy95] dans des systèmes proposant une interface graphique (KHOROS [Uni92], ALLEGORY [Vig92], VIPEX [Haa90]). Ces systèmes facilitent la mise au point de programmes par une présentation claire de la chaîne de traitement et donnent la possibilité à l'utilisateur de visualiser les résultats intermédiaires. Par contre, ils ne permettent pas de caractériser les erreurs en fournissant une réelle définition ou description de la différence entre l'objectif formulé et attendu et l'erreur rencontrée.

Planifier des chaînes
La planification des chaînes d'opérateurs automatise le processus de conception de ces chaînes. Cette automatisation de la construction de chaînes d'opérateurs fait appel à différents modules et techniques propres au domaine de la planification de tâches (planification hiérarchique, planification dynamique, planification par macro-opérateurs, etc...) et se retrouve dans des systèmes comme OCAPI [Tho93], BORG [Clo93, Clo94], ExTI [Ing92], etc...

Contrôler l'exécution
L'exécution d'un programme générique, adaptatif, est contrôlée (pilotée) en sélectionnant les bonnes branches du graphe des opérateurs et en ajustant les paramètres.
Nous trouvons de telle méthodes de travail dans des systèmes comme SIGMA [Mat90] dans lequel le module LLVE assure un savoir-faire dans la résolution de problèmes. Les systèmes CYPRESS (Grasper-CL [Kar94]) [Wil95] et VSDE [The92] intègrent quant à eux des modules de dialogue Homme-Machine permettant une correction d'erreurs et éventuellement une sélection interactive de arcs du graphe ([Moi95b]).

1.4.2 Un modèle des opérateurs induisant une représentation des données

En examinant les procédés de modélisation des connaissances en traitement d'images, on constate que l'objectif est le plus souvent de représenter les opérateurs de traitement ce qui entraîne alors un choix de représentation des données qu'ils manipulent.
Pour décrire un opérateur, on peut chercher à expliquer :

Pourquoi a-t-il été conçu ?
C'est l'approche "fonctionnelle". Le traitement d'images est décrit en termes de fonctions et les opérateurs sont les outils informatiques qui implémentent ces fonctions. Une description symbolique des données peut être introduite, sous la forme de description d'entrées/sorties, pour effectuer des contrôles syntaxiques, mais elle n'intervient pas dans la sélection des opérateurs. Cette approche est utilisée, par exemple, dans tous les environnement interactifs de traitement d'images.

Comment a-t-il été conçu ?
C'est une décomposition algorithmique qui peut être décrite en termes de flux de données et être utilisée pour organiser de manière homogène la description d'une bibliothèque de traitement d'images. Elle permet une meilleure compréhension des algorithmes, l'insertion de nouveaux opérateurs dans la bibliothèque ou le développement d'une nouvelle fonction par analogie [Ser89]. Cette analyse algorithmique peut aussi viser un but de standardisation de représentation des techniques du traitement d'images et introduit la notion d'algèbre d'images (AFATL[1] Standard Image Algebra [Rit90, Dav93]).

Ce qu'il produit...
... en termes de types de données
Cette approche peut être utilisée pour standardiser les représentations des données et construire des environnements d'interprétation d'images, comme dans le programme [IUE94], pour définir des langages de traitement d'images, construits sur des types abstraits de données comme dans SAPIN [Bel85a], pour faciliter le développement de programmes spécifiques ou pour planifier des opérateurs, en structurant l'espace des données sous la forme d'un graphe d'entités. Les opérateurs sont alors des processus de transfert entre ces types, comme dans LLVE.
... en termes d'effets
On construit un vocabulaire décrivant les caractéristiques géométriques extraites ou mises en valeur par les opérateurs [Sch89]. Si un objectif peut être décrit à l'aide de ces effets prédéfinis, des règles de raisonnement permettront alors de sélectionner les opérateurs qui les produisent.

Quelles sont ses propriétés
Celles-ci seront exprimées en termes mathématiques [Sch92]. Ceci permet de contrôler l'utilisation des opérateurs, en exploitant par exemple des propriétés d'idempotence ou d'associativité, pour contrôler le nombre d'itérations ou les combinaisons d'opérateurs.

Comment l'utilise-t-on ?
Cette information sera exploitée dans un système de contrôle d'exécution tel qu'OCAPI. Pour cela, on établit des liens, lors de la conception d'un système spécialisé, entre les objectifs à atteindre et les opérateurs résolvant cet objectif. La donnée de traitement d'images est perçue comme un ensemble de caractéristiques mesurables. Des règles expertes évalueront ces caractéristiques et le contexte d'application, pour sélectionner les opérateurs ou ajuster leurs paramètres [Tho93]. De même, le système CONNY intègre des mécanismes d'utilisation des données considérées comme éléments d'apprentissage ou de test, pour valider ou adapter les chaînes préétablies qu'il contient.

Autres descriptions.
Signalons enfin que des descriptions originales de données de traitement d'images peuvent être utilisées pour décrire les objets de l'image -par exemple Jungert [Jun93] utilise une algèbre d'image pour représenter symboliquement les relations spatiales entre objets- ou pour implanter des algorithmes particuliers et performants -modélisation de l'image binaire de Boomgard [van92], représentations pyramidales, en lacets, en files d'attentes, représentation polynômiale de GRASP [Ebe91] pour modéliser les couples (données,filtres), ...

1. Air Force Armament Laboratory

1.5 Analyse d'un graphe d'opérateurs

Si nous analysons une chaîne de traitement élaborée pour résoudre un problème de traitement d'images, nous pouvons considérer quatre points qui nous paraissent important : la planification, la validation, l'exécution et l'explication de la chaîne.

Or, dans un processus de développement d'un graphe d'opérateurs résolvant un problème d'interprétation, les seuls éléments qui permettent d'analyser les résultats, qu'ils soient finals ou intermédiaires, sont les évaluations des données. Leur rôle est donc important et les connaissances et processus de traitements mis en œuvre pour les générer doivent être suffisamment rigoureux pour permettre, d'une part, une interprétation correcte et d'autre part, une analyse précise pour déterminer la validité de la chaîne.

1.5.1 La planification

Ce que nous entendons ici par planification c'est le processus, automatique ou non, de construction de la chaîne de traitements (fig.1.7) résolvant un objectif. Il met en œuvre des connaissances spécifiques quant à la propagation des informations relatives aux opérateurs.

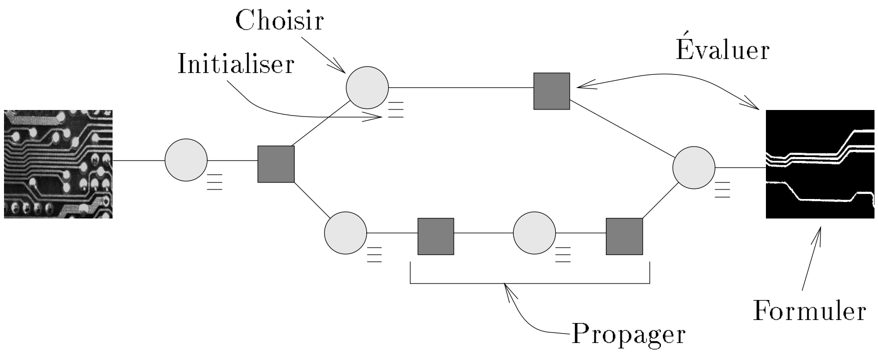


FIG. 1.7 - *Planification d'une chaîne*

Dans un premier temps, ces connaissances sont de haut niveau et correspondent à des enchaînements de fonctionnalités relevant de techniques du traitement d'images pour résoudre un objectif. Dans un second temps, les connaissances mises en œuvre font référence au choix des opérateurs ou séquences d'opérateurs réalisant ces fonctions. Alors que la phase de conception fonctionnelle de la chaîne de traitements peut être élaborée hors de toute instance de la donnée, la phase de choix des opérateurs va dépendre de la nature et du type de cette donnée. Ceci d'autant plus qu'elle va s'accompagner d'un choix initial des paramètres, choix qui, comme nous le verrons dans le paragraphe suivant est un élément sensible de la conception de la chaîne.

Mais tout ce processus est guidé par l'objectif à atteindre. Sa formulation peut revêtir plusieurs formes et être exprimée de plusieurs façons. Ainsi, nous pouvons avoir une forme rigoureuse de description de l'objectif répondant à des critères particuliers comme une base de faits d'un système expert, un modèle CAO des objets, etc. Nous pouvons avoir aussi une formulation plus subjective du but à atteindre comme un ensemble d'images tests, ou une représentation mentale de la donnée résultat dans des cas de traitements non automatiques.

Qu'elle soit rigoureuse ou subjective, la formulation de l'objectif peut être exprimée de plusieurs façons rendant plus ou moins complexe le processus d'évaluation des résultats, intermédiaires ou finals. En effet, les expressions peuvent être élaborées dans les termes du traitement d'images ou de l'application ; ce qui est généralement le cas lorsque on traite des représentations mentales du problème à résoudre.

Le processus d'évaluation doit alors mettre en œuvre de façon plus ou moins complexe une phase de transcription de l'expression de l'objectif et de celle de l'état courant. Cette évaluation est alors relativement aisée à implémenter dans des systèmes de génération automatique de la chaîne de traitement qui, généralement, utilisent le même formalisme. Les outils de résolution de problèmes, issus des recherches en Intelligence Artificielle -systèmes experts, etc.- nous permettent alors d'obtenir des procédés fiables dans la mesure où les entités qu'ils manipulent sont bien représentatives des éléments de la donnée qu'elles décrivent. Par contre, cette évaluation est beaucoup plus complexe à effectuer dans le cas où les formes de description appartiennent à des domaines distincts ou font appel à des représentations subjectives de l'objectif. Et ceci se rencontre souvent lorsqu'un expert d'un domaine spécifique manipule des opérateurs de traitement d'images pour construire manuellement sa chaîne de traitement. Ceci nous amène directement au problème de la validation de la chaîne de traitement.

1.5.2 La validation

Elle a pour but de justifier la validité de la chaîne par rapport au cahier des charges de l'application et de garantir que la chaîne fournit ou ne fournit pas le bon résultat dans d'autres cas que celui qui a permis de la concevoir. Cette notion recouvre en fait deux concepts dépendant de l'échec ou du succès de l'exécution de la chaîne, c'est à dire de l'évaluation de la donnée courante par rapport à l'objectif (fig.1.8).

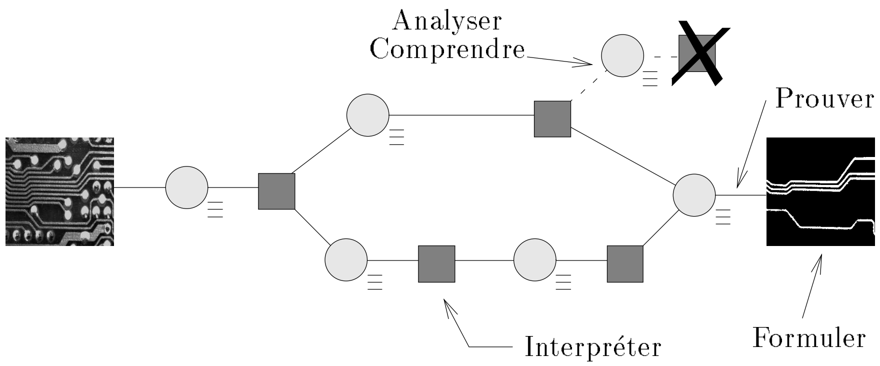


FIG. 1.8 - *Validation d'une chaîne*

Dans le premier cas, l'échec doit pouvoir être analysé et donc compris. De cette compréhension va dépendre la justesse des décisions prises pour rectifier les choix émis lors du processus précédent : choix de la fonction à remplir, choix du ou des opérateurs réalisant cette fonction, choix des valeurs de paramètre des opérateurs sélectionnés. Nous pouvons ainsi voir qu'une erreur de paramétrage en début de chaîne peut mettre en jeu des processus complexes de validation ou d'invalidation des résultats. Si ceci semblait évident, il faut pourtant bien considérer à nouveau cette phase de paramétrage d'opérateurs comme une des plus importantes du processus de conception mais aussi du processus de validation car c'est souvent de l'interprétation correcte de l'effet d'une valeur de paramètre sur les données que va dépendre la compréhension de toute la séquence exécutée. Par exemple, un seuillage trop élevé ou un filtrage trop fort en début de chaîne de traitement peuvent éliminer les éléments que le reste de la chaîne cherche à mesurer ou mettre en évidence.

Dans le second cas, le succès de la résolution de l'objectif par la chaîne créée doit amener le concepteur de cette dernière à tenter d'en donner la preuve, c'est à dire à fournir les caractéristiques de l'ensemble des objectifs pouvant être résolus. Les opérateurs utilisés et les paramètres qui leur sont attribués définissent en effet un ensemble de propriétés que doivent respecter les données initiales, intermédiaires et finales, à l'image de la description de la boîte définissant le mouton du PETIT PRINCE. Si nous considérons que l'objectif pour lequel est construit la chaîne est développé dans une représentation rigoureuse et exploité par un système de génération automatique, la formulation de l'objectif peut être considérée comme la preuve de cette chaîne si elle contient cet ensemble de propriétés caractéristiques.

Dans tous les cas, le problème d'évaluation peut être considéré comme un problème d'interprétation des résultats, soit en termes de traitement des informations contenues dans la donnée, soit dans des termes spécifiques du domaine de l'application.

1.5.3 L'exécution

Ce processus d'interprétation des résultats lors d'échecs intervient régulièrement dans la phase d'exécution de la chaîne -et le contrôle de cette exécution-, *a fortiori* si la conception de celle-ci se fait par un procédé d'essais-erreurs. Il faut pouvoir alors établir le lien entre ce que l'on souhaitait produire et ce que l'on a produit (fig.1.9).

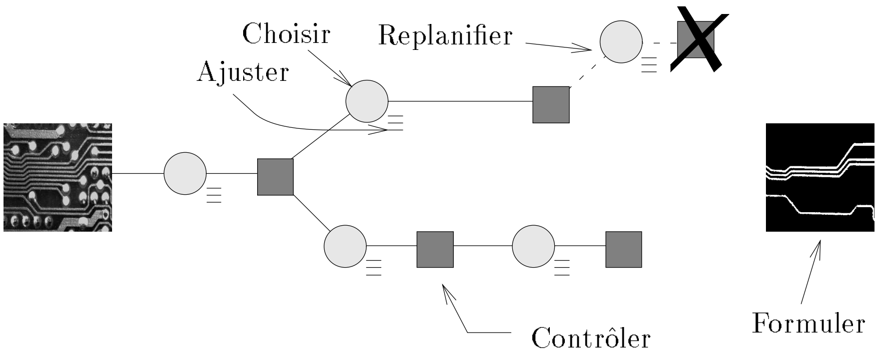


FIG. 1.9 - *Exécution d'une cxchaîne*

Les problèmes liés à l'exécution vont être de déterminer les points critiques de la séquence du plan déjà exécuté. Ces points peuvent faire référence simplement à des paramètres dont l'ajustement permettrait de résoudre le conflit, mais aussi à des opérateurs mal choisis pour instancier une fonctionnalité particulière. Dans ces deux cas, les modifications de ces éléments remettent en cause la preuve de la chaîne toute entière. Il faut donc pouvoir être capable de répercuter ces changements le long de cette chaîne afin de tenir compte des nouvelles conditions de traitement, lors de la phase de validation. Ceci implique, de disposer de modèles particuliers de description de donnée et d'opérateur pouvant témoigner de ces changements et de les propager jusqu'à l'étape courante, voire l'objectif.

Mais ce qui peut être plus gênant encore est le fait que ces changements risquent de remettre en cause la construction même de tout ou partie de la chaîne de traitement. Le seul recours possible est alors de replanifier la séquence. Le problème consiste alors à évaluer le défaut dans le but de modifier l'objectif. Cette évaluation doit être faite en considérant la séquence, généralement à partir du point critique considéré. Cette replanification peut être également nécessaire lorsque l'évaluation de la donnée courante fournit un résultat considéré comme trop éloigné de l'objectif à atteindre. Dans ce cas il est quelques fois indispensable de fournir d'autres éléments relatifs au résultat à obtenir.

1.5.4 L'explication

Dans un objectif plus large d'apprentissage du traitement d'images, la chaîne développée doit contenir les éléments nécessaires à la compréhension des choix des opérateurs et de leurs paramètres. Plus finement, la représentation selon les modèles utilisés de chacun de ces opérateurs de la chaîne doit rendre compte de son rôle sur les données et les concepts de l'application, en faisant référence à son contexte d'utilisation, c'est à dire à l'endroit où il est utilisé (fig.1.10).

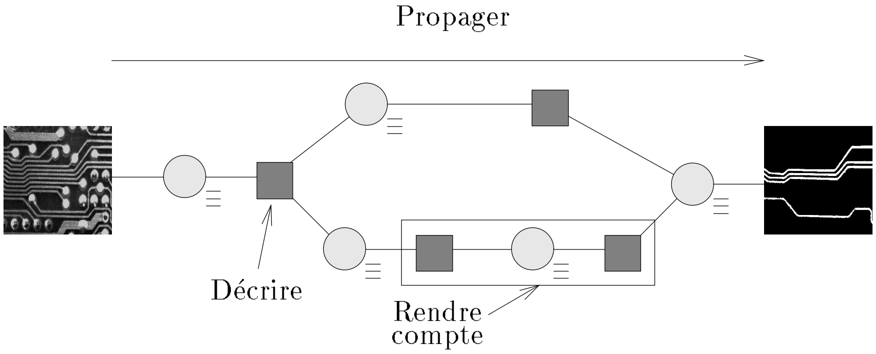


FIG. 1.10 - *Explication d'une chaîne*

Ce rôle doit alors être exprimé en fonction d'une description des données et des traitements effectués par les opérateurs. La compréhension de la chaîne est alors rendue possible par la capacité à propager ces informations tout au long des traitements. Là encore vont s'opposer les différentes formes de représentations que nous avons cités précédemment. En effet, selon que l'on évolue dans le domaine de l'application ou du traitement d'image, dans une représentation rigoureuse des traitements ou des représentations mentales des entités, le niveau d'explication ou de compréhension de la chaîne sera plus ou moins élevé.

1.6 Conclusion

Le traitement d'images est une technique informatique de traitement et d'extraction de l'information contenue dans un signal. Il est toujours utilisé dans une application spécifique à un domaine, manipulant cette information. Il faut donc pouvoir étudier le traitement d'images sous le triple aspect : son action en terme de traitement du signal image, son action sur les structures de représentation des données, son action sur l'information (la sémantique) associée à ces structures de représentation, autrement dit, la manipulation de concepts qu'il permet de réaliser.

L'analyse de ces quatre points -planification, validation, exécution, explication- nous permet de dégager les éléments à mettre en œuvre dans des systèmes automatique de conceptions de programmes. Ces moyens sont les suivants : une expression de l'objectif en termes d'information à extraire, un modèle d'opérateur rendant compte des transformations d'informations effectuées sur les données et un modèle de donnée décrivant les informations intrinsèques en termes d'indices visuels. Ils vont ainsi nous permettre d'élaborer un formalisme permettant d'évaluer les résultats produits afin de choisir les opérateurs et leurs paramètres -et d'ajuster ces derniers- et de prouver la validité du graphe. Il devra permettre également d'analyser et de comprendre les causes d'échecs de planification ou d'appariement prédiction / production, que ces échecs soient dus à un défaut d'opérateur, de paramètre ou encore de formulation d'objectif.

Ce formalisme permet de construire un environnement complet de compréhension et de simulation du traitement d'images. Il offre de nombreuses possibilités d'exploitation pour l'enseignement du traitement d'images aussi bien pour la compréhension de ses fonctionnalités que pour son utilisation ou son enrichissement par de nouvelles fonctionnalités.

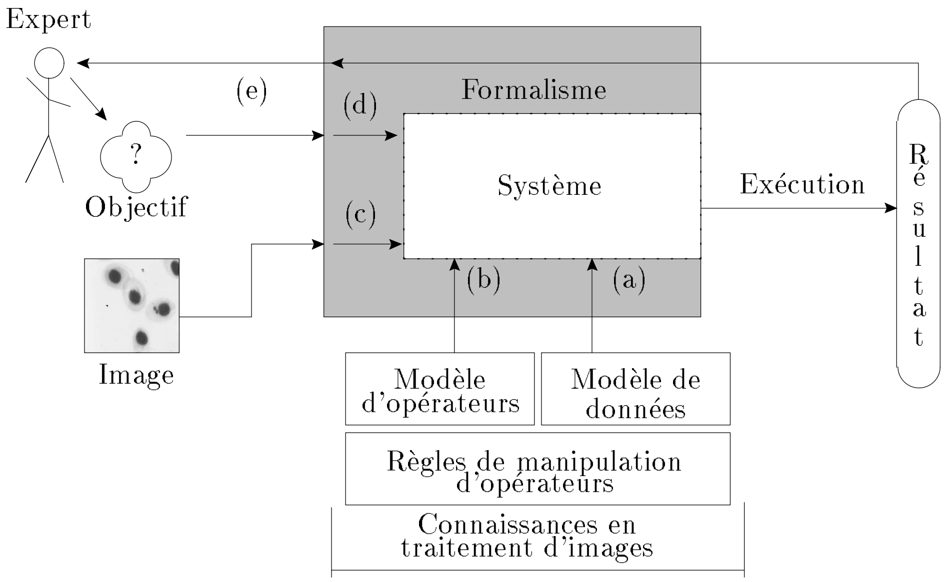


FIG. 1.11 - *Intégration d'un formalisme particulier*

Parallèlement un des objectifs de l'exploitation de ce formalisme est son utilisation pour la planification automatique des chaînes d'opérateurs de traitement d'images répondant à un objectif d'analyse formulé par un expert du domaine d'application (fig.1.11). Le formalisme intègre alors les connaissances du traitement d'images dans différents modules d'un tel système : un modèle de donnée (a) et un modèle d'opérateur (b) auxquels nous ajoutons des règles de manipulation (simplification, pertinence d'utilisation, etc). D'autre part, il met en place des langages internes de description de données (c) et de formulation d'objectifs (d), ce dernier devant être utilisé par un processus de gestion d'interface sollicitant l'expert pour une bonne description de haut-niveau (e).

Chapitre 2

Systèmes de résolution de problèmes et expression d'objectifs

Dans ce chapitre nous allons présenter divers systèmes de génération de plan en essayant d'examiner comment sont représentées les entités manipulées. Dans une seconde partie, nous présenterons plus en détails le système ExTI qui a servi de base à la construction de notre formalisme, puis nous regarderons quels sont les outils que nous proposent les recherches sur les grammaires afin de construire un langage d'expression d'objectif.

2.1 Systèmes de résolution de problèmes

2.1.1 LLVE

Dans [Mat89, Mat90] -la principale référence ([Mat86]) est en japonais- Matsuyama rappelle le principe du *système de segmentation d'images dirigé par le but*[1] : LLVE (Low Level Vision Expert). Ce système d'extraction de primitives fonctionne sur la base de règles de production. Celles-ci représentent la connaissance nécessaire à l'extraction de caractéristiques. Cette connaissance peut elle-même être décrite sous forme de réseaux dont les éléments correspondent à des entités caractéristiques pouvant être extraites (lignes, contours, régions homogènes, ..., cercles, parallélogrammes, etc...) et dont les liens correspondent à des propriétés de spécialisation ou généralisation ou des traitements pré-établis entre ces entités.
L'objectif d'extraction est formulé à partir d'exemples décrits dans une image de référence. L'opérateur doit pour cela spécifier la structure principale du but (*Goal Frame*) puis ses propriétés (*Goal Properties*). L'exemple suivant est donné dans [Mat89] et décrit le but correspondant à l'extraction

1. Goal-Directed Image Segmentation System

d'une forme rectangulaire de taille comprise entre 100 et 400 et située dans la zone image spécifiée (rectangle englobant ((165,89),(204,115))).

Goal Frame :	RECTANGLE
Goal Properties :	
AREA-SIZE	(INTEGER):
	(100 400)
COMPLEXITY	(SYMBOLIC):
	HIGH
WINDOW	(COORDINATE):
	((165 89)(204 115))

Le processus de recherche de la solution va donc établir un lien entre les éléments représentés dans le but exprimé et le réseau complet de la connaissance de LLVE. Le sous-réseau optimal (fig.2.1) étant trouvé, il est configuré selon les propriétés de l'objectif, puis exécuté. Nous voyons que dans un tel système, les éléments du langage à partir desquels sont exprimés les objectifs correspondent à un ensemble de faits permettant à un système expert de déclencher des règles de résolution d'objectif. Ainsi, de par l'implantation de ces réseaux sémantiques, le système effectue une distinction entre

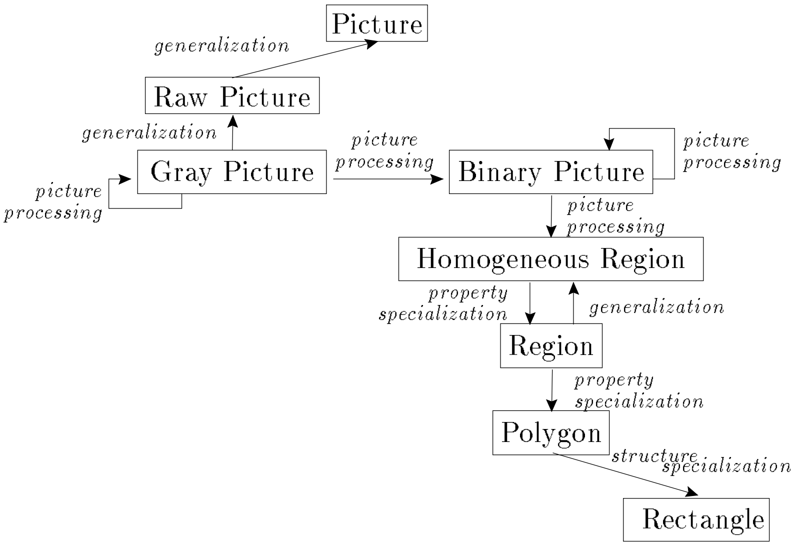


FIG. 2.1 - *Sous-réseau solution de LLVE*

la réalité image pour laquelle est développé un objectif sous forme de base de faits, et le processus de résolution qui exploite cette base en déterminant la solution en fonction du réseau, établi de façon préalable pour la reconnaissance de la scène. Si l'on considère enfin la structure particulière de l'expression de l'objectif et une trop grande spécialisation des règles utilisées pour l'extraction des entités, tout ceci explique les difficultés du système à extraire les objets complexes d'une image.

2.1.2 VSDE

Le système VSDE (Vision Systems Development Environment) [The92, Bod95] s'appuyant sur l'expérience de l'environnement ALLEGORY [Vig92] (résultats compatibles, base d'opérateurs et format d'image IPL) est un outil de développement de systèmes spécialisés, configurables selon l'application d'inspection traitée. Il est basé sur le système expert XIA développé chez THOMSON-CSF. La description du contexte de l'application est fournie sous forme de faits reconnus par XIA. Un objectif sera donc un ensemble de faits déclenchant des *règles de production* représentant, avec les notions de *hiérarchie de concepts* -connaissances fonctionnelles-, et les *performances des tâches*

-connaissance sur les résultats produits-, la connaissance du traitement d'image introduite dans ce système. Pour élaborer cet objectif, le système établit un dialogue avec l'expert du domaine. Ce dialogue est géré par un ensemble de règles qui vont permettre de construire ou d'instancier une hiérarchie de concepts. Celle-ci représente dans ce cas la connaissance *a priori* sur le domaine de l'application ainsi que la connaissance du traitement d'images intégrée dans le système. La hiérarchie de concepts peut être rapprochée des réseaux sémantiques de LLVE, les nœuds renfermant des renseignements sur les concepts, les arcs établissant des relations sémantiques entre ces concepts. Par le biais de ce dialogue, le système obtient une description de l'environnement, un modèle des objets de la scène, un ensemble d'images tests et la description de la tâche d'inspection.
L'avantage de ce développement interactif de l'objectif, guidé par une implémentation de connaissances particulières, est de fournir au système VSDE une description précise des défauts à détecter sur les objets. Par contre, la construction de l'objectif n'étant plus à la charge directe de l'utilisateur, il sera alors difficile à ce dernier de généraliser ou préciser cet objectif de façon simple ; ce qui lui permettrait de réinvestir les résultats produits par le système pour d'autres applications, ou d'appréhender le processus d'interprétation des interactions du dialogue afin de comprendre les erreurs de détection.

2.1.3 OCAPI

Dans le générateur de systèmes experts OCAPI (Outil de Contrôle Automatique de Procédures Images [Moi95c, Moi95a]), comme dans LLVE, les buts sont prédéfinis dans la base d'opérateurs puisque la notion de but est ici liée à la notion de résultat produit par les opérateurs. Le niveau le plus haut de l'expression des objectifs accessibles aux utilisateurs (humain ou programme) est constitué par les requêtes ; chaque requête étant formée d'un but et de paramètres reflétant les conditions d'application et de résolution : pré et post-contraintes sur les données (types, etc...).
Ces requêtes sont divisées en deux ensembles : les *requêtes initiales* et les *requêtes internes*. Les requêtes initiales, qui sont des éléments de la base de faits d'OCAPI, correspondent aux objectifs de haut niveau pouvant être résolus par le système et constituent ainsi la base du raisonnement. Les requêtes internes correspondent à des sous objectifs liés aux décompositions d'opérateurs complexes ; les opérateurs élémentaires résolvant quant à eux les buts pour lesquels ils ont été écrits.

De ce fait, la résolution de la requête faite au système est effectuée en recherchant les liens existant entre requêtes-buts, buts-opérateurs et opérateurs-requêtes ou opérateurs-résolution (fig.2.2) pour obtenir une arborescence d'opérateurs plus ou moins complexes. C'est sur cette arborescence que vont s'appliquer des règles d'initialisation, de configuration, d'ajustement ou de sélection, qui constituent la partie la plus importante de la connaissance du traitement d'images implantée dans ce système.
Le principal intérêt de ce système réside dans son module de supervision mettant en œuvre un contrôle de l'exécution du plan de façon interactive et autorisant ainsi l'ajustement de paramètres de même que la remise en cause de parties de ce plan ; cette supervision étant rendue possible par la modélisation des opérateurs en termes d'objectifs à atteindre.

2.1.4 BORG

Le système BORG (Blackboard orienté Objet pour la Résolution de problèmes par Génération de plans) est basé sur une hiérarchisation du traitement d'images selon cinq niveaux d'abstraction [Clo95, Fic96] : les *Requêtes* exprimant les objectifs, les *Tâches* permettant de décomposer le traitement de l'objectif en sous-problèmes, les *Fonctionnalités* exprimant de façon abstraite les méthodes de traitement d'images, les *Procédures* qui sont les outils ou savoir-faire (ensemble d'outils)

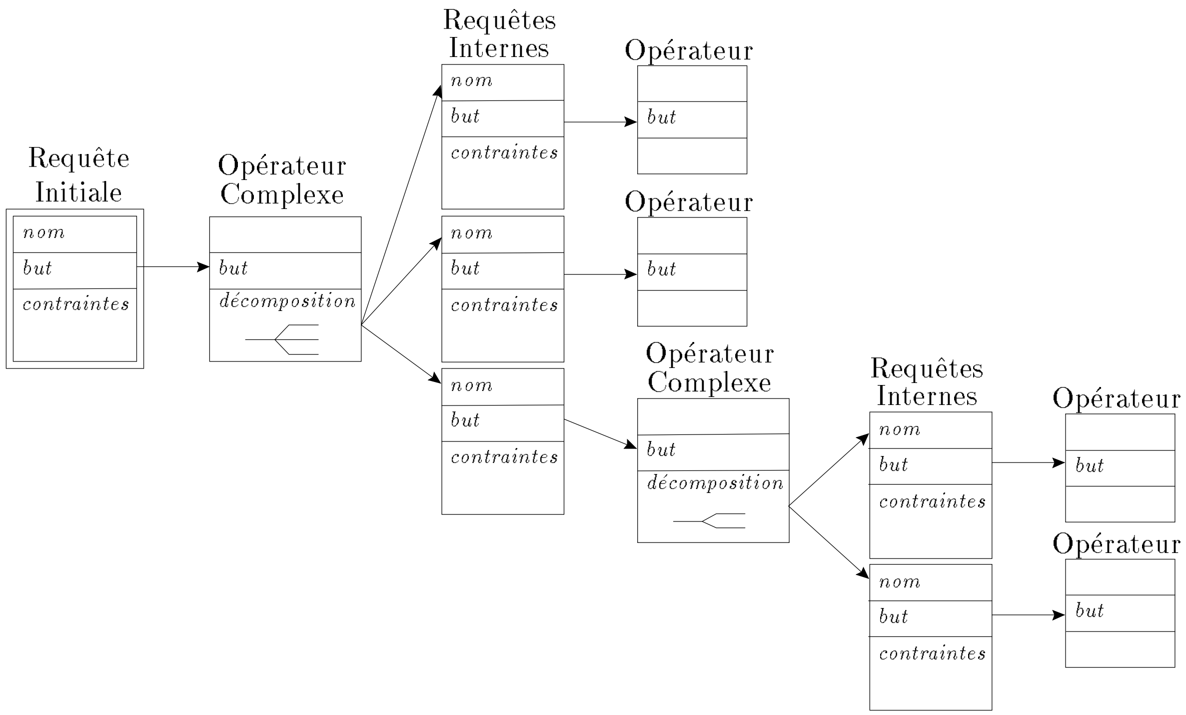


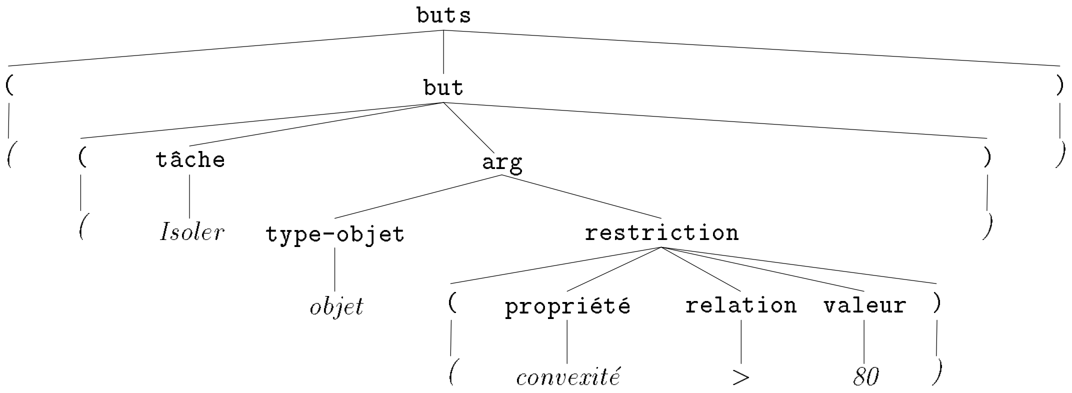
FIG. 2.2 - *Lien requêtes-buts,buts-opérateurs, opérateurs-requêtes / opérateurs-résolution*

symboliques décrites par les fonctionnalités, les *Opérateurs* qui sont finalement les réalisations des procédures dans une bibliothèque particulière.
L'objectif (fig.2.3) y est donc décrit sous la forme d'une requête constituée d'un ensemble de tâches à accomplir (buts) et de contraintes à respecter. Le vocabulaire utilisé est d'un niveau symbolique empruntant les termes des concepts et propriétés rencontrés en traitement d'images: contours, segmentation, texture, etc....

1. La requête: elle consiste en une conjonction de buts exprimés à l'aide de la grammaire suivante (extrait de [Clo94]):
```
buts         ::  ( <but>* )
but          ::  ( <tâche><arg>* )
arg          ::  <type-objet><restriction>* | <propriété><restriction>*
restriction  ::  ( <propriété><relation><valeur>)
```
2. Les contraintes seront exprimées sous la forme d'une succession d'éléments prédéfinis permettant d'ajuster le résultat ou fournissant des informations quant aux choix possibles d'opérateurs lors de la construction du plan solution.

Le processus de résolution développe un plan de façon dynamique, contrairement à LLVE dans lequel il y a sélection d'arbre de résolution dans le réseau. La construction de ce plan, qui est un graphe d'actions possédant les cinq niveaux d'abstraction précédents, est effectuée par l'application de règles de production de *Sources de Connaissances* expertes en processus de création, modification d'actions ou de lien entre actions. D'autres Sources de Connaissances permettront ensuite d'instancier, d'exécuter, d'évaluer et de corriger ce plan en fonction des données effectives et des interactions (validation) avec l'utilisateur.

Le langage d'expression de l'objectif de ce système a pour intérêt de permettre à l'utilisateur de développer une requête selon une syntaxe précise, employant des termes faisant référence à des



Buts = ((Isoler objet (convexité > 80)))
Contraintes = ((contours précis) (sous-segmentation))

FIG. 2.3 - *Exemple d'expression de but et arbre syntaxique correspondant*

mesures d'entités des données (propriétés) ou des termes symboliques de haut niveau (tâches) qui seront résolus de façon abstraite par les fonctionnalités.

2.1.5 Le système de Lemaire

Le système développé au Cert-Onera par J.Lemaire [Lem95, Lem96], quoique limité à un domaine spécifique de détection d'objets dans un réseau de tuyauterie, a pour avantage de proposer clairement un langage de haut niveau d'expression d'objectifs. Nous reparlerons de ce système lorsque nous aborderons le problème de l'appariement entre la description de l'objectif -appelée ici **D**escription de la **S**cène **A**ttendue (DSA)- et la description de l'état courant -appelée ici **D**escription de la **S**cène **P**erçue (DSP)- (voir troisième partie chap.3).

Le langage utilisé pour exprimer la DSA est constitué d'un ensemble $\mathcal{L}_T$ de types d'objets pouvant être rencontrés dans la scène (*tuyau, vanne, cuve, etc...*), d'un ensemble $\mathcal{L}_{R_1}$ d'attributs se rapportant aux objets (*couleur, orientation, etc...*) et d'un ensemble $\mathcal{L}_{R_{n\geq 2}}$ de relations n-aires dénotant des contraintes topologiques entre les objets.
Une description d'objectif aura alors la forme linéaire (fig.2.4) d'un cheminement permettant d'atteindre l'objet recherché ou une forme arborescente (fig.2.5) de profondeur 1 pouvant fournir des informations complémentaires. Les exemples suivantes sont tirés de [Lem95].

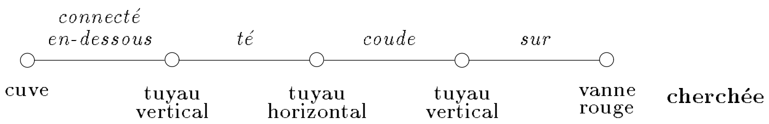


FIG. 2.4 - *DSA linéaire*

De son côté la DSP est constituée d'un ensemble formé par les éléments de la DSA -types et attributs- perçus dans la scène par le capteur auxquels sont affectés des degrés de confiance. Après traitement par des modules de détection *ad hoc*, un système d'appariement déterminera alors la **D**escription de la **S**cène **R**econnue. Cette dernière phase fait de ce système un des plus proches d'ExTI [Ing92] dans le processus de recherche de solution.

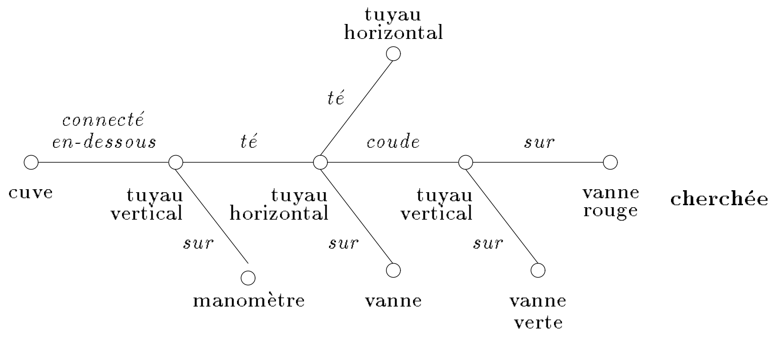


FIG. 2.5 - *DSA Arborescente*

2.1.6 D'autres systèmes

Enfin, nous devons citer d'autres systèmes tels CONNY [Lie86, Lie92] ou IMPRESS [Has86]. Ici, le langage d'expression d'objectif revêt des formes internes plus spécifiques. Le langage externe, tourné vers l'utilisateur consiste à la fois en une partie déclarative des caractéristiques des entités de l'image à retrouver et en une base d'images de tests, comme dans CONNY, ou en une donnée (image ou dessin) renfermant les caractéristiques à extraire comme dans IMPRESS.

2.2 Le système ExTI

Le système ExTI développé par J.-M. Inglebert [Ing92] propose une implémentation particulière de la notion de transformation d'information. Exploitant des modèles de données et d'opérateurs exprimés en termes d'indices visuels et de transformation de ces indices, un module de planification gère la construction de la chaîne de traitement. Dans ce chapitre, nous allons préciser les particularités de ce système et relever ses limites qui nous ont amené à proposer de nouveaux modèles (deuxième partie de ce rapport) et à reconsidérer certains modules et concepts du processus de planification (troisième partie).

2.2.1 Structure générale du système

Le système ExTI est constitué de deux modules indépendants Ex et TI.

La partie Experte gère, par un module de planification, la construction automatique des chaînes de traitement d'images répondant à un objectif. Elle met en œuvre une base de description symbolique d'opérateurs développée à partir d'un modèle de représentation particulier, axé sur la transformation d'informations, ainsi qu'une base de types de données de traitement d'images.

La partie Traitement d'Images établit, à l'aide d'une base d'opérateurs réels, la transcription de la chaîne symbolique issue de la planification par Ex et permet l'exécution de cette chaîne.

2.2.2 Les modèles utilisés

2.2.2.1 Un langage de description de concepts

Les transformations d'informations peuvent alors être classées en deux groupes :

1. Les quantifications d'une entité de traitement d'image selon une caractéristique liée à l'opérateur *(par exemple : nombre de classes, surfaces des composantes connexes)*.
2. Les définitions d'une entité, qui comporte : les sélections d'entités de la donnée d'entrée selon un critère donné *(par exemple : composantes connexes dont la surface est supérieure à un seuil déterminé)* et la construction d'entité par combinaison d'entités spécifiques.

Les opérations de quantification d'une entité permettent d'exprimer des buts ou des sous-buts d'interprétation traduisant une mesure sur ces entités et ainsi d'élaborer les descriptions des objectifs d'interprétation spécifiques ou de décrire les paramètres des critères de sélection d'entités. Ces sélections d'entités, fournissent quant à elles des concepts de traitement d'images pouvant être combinés ou liés entre eux.

Une sélection d'entités va s'effectuer de deux façons différentes :

1. La première consiste à spécialiser un concept. Cette spécialisation se fait en ajoutant une propriété, ou sélecteur, à un concept déjà formé ou en mettant en relation deux ou plusieurs concepts.
2. La seconde consiste à généraliser un concept en étendant sa description par disjonction avec d'autres descriptions de concepts ou en étendant le domaine d'une de ces propriétés.

Afin de manipuler aisément ces notions, de les implanter et de les exploiter lors de la planification, ExTI utilise une représentation des modèles basée sur le calcul des prédicats [Nil80]. Les propriétés $P = [p\ r\ v]$ sont adaptées des sélecteurs décrits par Michalski [Mic83], [Mic84] dans le calcul des prédicats annotés. Elles sont composées d'un descripteur p *(exemple : surface, niveau de gris, distance, ...)* d'une relation d'ordre r -les relations implantées sont $<$ et $\geq$- et d'une valeur v. Conformément à cette définition de la propriété, la sélection par mise en relation de plusieurs concepts sera formée par des prédicats composés de type *"R(wff1 wff2 ...)"*.

2.2.2.2 Les types de données

Les types de donnée correspondent à des classes génériques regroupant différentes entités manipulées par les opérateurs de traitement d'images. Ainsi, nous trouvons dans ces types des classes comme *image binaire*, *image binaire multi-niveaux*, *image de labels*, mais aussi *vecteur*, *entier*, etc. Chaque donnée est alors une instance de l'une de ces classes et est représentée par l'ensemble de caractéristiques suivant :

Classe	Cl
`nom`	*nom* de l'instance
`s-m`	*structure manipulable* : nom des sous-structures de la donnée liée à la classe
`repr`	*représente* : descriptions des concepts associés aux sous structure
`o-a`	*opérateurs applicables* : liste des opérateurs applicables à la donnée
`d-a`	*donnée accessible* : liste des noms des classes relatives aux données produites par chaque opérateur applicable.
`nœud`	*nœud* associé à cette instance dans le graphe d'opérateur.

2.2.2.3 Les descriptions des données

La donnée est réellement décrite par les champs `s-m` et `repr` de ces classes. Les autres champs déterminent le contexte de manipulation -création, utilisation- de la donnée pour la résolution de l'objectif.

La syntaxe utilisée pour ces descriptions est donnée dans [Ing92] et permet d'écrire des expressions d'objectif :

```
<disjonction>  ::=  ( <vecteur>* )
<vecteur>      ::=  ( <propriété>* )
                |   ( <propriété>* <disjonction>)
<propriété>    ::=  ( <descripteur1> <comparateur> <valeur>)
                |   ( <descripteur2> <disjonction> <disjonction>)
                |   ( <descripteur3> <disjonction>)
```

```
<descripteur1>   ::=  nivg | ordre | surface | périmètre | max_cc | min_cc
                  |   #(#:var:<nom>. symbol)
<descripteur2>   ::=  dans | hors | intersecte | nintersecte | et_c
                  |   #(#:var:<nom>. symbol)
<descripteur3>   ::=  valeurs | max | min | nombre | euler
                  |   #(#:var:<nom>. symbol)
<comparateur>    ::=  >= | < | #(#:var:<nom>. symbol)
<valeur>         ::=  ℝ | ℕ |<courbe>
<courbe>         ::=  (cercle <x> <y> <r>)
                  |   (fenetre <x1> <y1> <x2> <y2>)
                  |   #(#:var:<nom>. cons)
```

Des éléments de cette syntaxe se retrouveront bien évidemment dans le langage que nous avons construit et que nous détaillons dans le chapitre 2 (2^{e} partie).

2.2.2.4 Le modèle d'opérateur

Un opérateur de traitement d'images est décrit symboliquement comme une fonction permettant d'instancier le champ `repr` de la donnée produite.

Une description d'opérateur est dédiée aux types des données manipulées. Ainsi, un opérateur de seuillage, par exemple, possède deux ensembles de représentation : l'un dédié au type *image multi-niveaux* et l'autre dédié au type *image binaire multi-niveaux*. Chacun de ces ensembles est alors constitué de descriptions dépendant de la nature des paramètres, instanciables par une valeur numérique entière déterminée au cours du processus d'appariement chargé de contrôler la cohérence des états du plan par rapport à l'objectif, ou issus d'un résultat d'une instance de la classe *entier*. Les quatre représentations font référence au même opérateur réel de traitement d'images.

```
( seuiller1  im    #(#:var:seuil . fix)
( seuiller2  im    fix)
( seuiller3  imb+  #(#:var:seuil . fix)
( seuiller4  imb+  fix)
```

2.2.3 Un module de planification

Possédant des modèles de données et d'opérateurs construits sur la même notion d'indices visuels, le système général de génération de plans a été développé et implémente une stratégie de recherche de solution en profondeur d'abord. Il permet de résoudre un problème de traitement d'images dans un domaine d'application défini en utilisant ses concepts et la connaissance de ses experts.

Dans le but de fournir une aide à la compréhension et l'analyse de la chaîne de traitement d'images, il permet d'accéder à l'information liée au graphe d'opérateurs planifié (fig.2.6). Ces informations pourront alors être exploitées pour développer un langage descriptif du domaine d'application et pour maîtriser l'*histoire* du plan.

Ce module de planification, met en œuvre un module d'appariement permettant de constater la *distance*[2] qui sépare la donnée produite à l'objectif. Le moteur de planification est chargé de construire le graphe d'opérateurs *(arcs)* à appliquer aux données *(nœuds)* pour parvenir à l'objectif visé. De cette façon, un tel graphe, ou plan, peut être vu comme la trace de la propagation des transformations contenues dans les diverses données, qu'elles figurent à un niveau intermédiaire, ou terminal -solution ou même échec. Cette planification permet entre autre, et cela pour chaque nœud

2. Le terme de *distance* paraît peut-être impropre si l'on considère sa définition mathématique, basée sur une notion de métrique. Nous verrons au chapitre 3 (3^{e} partie) pourquoi, et sous quelles définitions, nous employons ce terme.

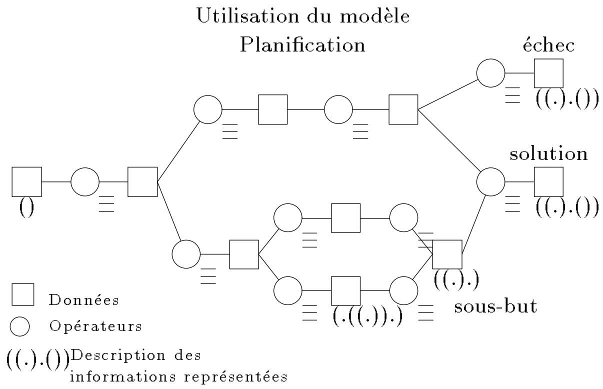


FIG. 2.6 - *Graphe d'opérateurs*

du graphe, de connaître la description en termes d'entités de traitement d'images de l'information qu'il renferme.

Le planifieur utilisé possède une représentation du monde constituée de :

- un état initial déterminé par une donnée D_I et sa description $\delta(D_I)$
- un état final qui est la description $\delta(D_F)$ issue de la représentation mentale de l'objectif à atteindre
- un ensemble d'opérateurs génériques applicables selon le type de la donnée manipulée et instanciés par le module d'appariement

Nous présentons ici l'algorithme de la partie de ce module chargée de la transformation des informations d'une donnée.

```
Transformer(D_X) :
Tantque l'objectif n'est pas atteint δ(D_X) ≠ δ(D_F)
        Établir la liste des opérateurs applicables à D_X
        Tantque un opérateur op est applicable
                générer la description δ(op(D_X))
                Instances ←Filtrage(δ(op(D_X)), δ(D_F))
                si Instances est non vide alors
                   D_Y = Créer(op(D_X), Instances)
                   si (∀D_α ∈ Plan, δ(D_Y) ≠ δ(D_α)) alors
                      Transformer(D_Y)
```

L'établissement de la liste des opérateurs applicables à une donnée est effectué en fonction de la description de l'état final $\delta(D_F)$, et du type de la donnée.

2.2.4 Limites du système

La faisabilité d'un tel processus de génération de plan a été montrée au travers du système ExTI. Le plan ainsi développé répond à un objectif exprimé en termes d'indices visuels et est guidé par les transformations d'informations et non par des connaissances sur la conception classique de chaînes d'opérateurs de traitement d'images. Néanmoins, les concepts de ce système doivent être approfondis.

D'une part, le processus de planification doit lui-même être remis en question afin de mieux exploiter la notion de nomination[3] de concept de l'application qui consiste à associer un terme spécifique à une description d'entité. En effet, si nous considérons ces concepts en tant qu'objectifs ou sous-objectifs d'une application, un plan peut leur être associé de la même façon que leur est associée la description permettant de générer ce plan. Nous pouvons donc supposer qu'il est possible de réutiliser un tel plan, répondant au concept, lorsque le nom de celui-ci apparaît dans une description d'objectif. L'étude de ce procédé de réutilisation de plan fait l'objet de la section 1.2.2 (3^e partie).

D'autre part, des problèmes d'ordre combinatoire de certains graphes d'opérateurs générés pour la résolution d'objectifs complexes, doivent amener une réflexion sur l'intérêt de décrire les opérateurs en fonction des types des données ainsi que sur le contenu de la base d'opérateurs symboliques. En effet, la diversité des types de données entraîne des multiplications d'opérateurs symboliques : plusieurs opérateurs de seuillage, par exemple, selon qu'ils manipulent une image multi-niveaux, "binaire multi-niveaux", d'étiquettes, de distances, etc, ou qu'ils manipulent des seuils de types réels, entiers ou issus de mesures sur d'autres données. Ceci sera résolu par la mise au point d'un modèle unique de représentation des données (voir chap.1, 2e partie). De plus, les performances du module de planification, en termes de temps et de complexité, sont réduites par la redondance des utilisations d'opérateurs symboliques, liées à des propriétés telles que la commutativité, ou par la présence dans une base d'opérateurs symboliques de macro-opérateurs pour lesquels il existe une chaîne planifiée constituée de plusieurs opérateurs de cette même base.

De ces deux points découle l'intérêt de posséder un système d'appariement, plus puissant, ne retournant plus les valeurs nécessaires pour instancier les opérateurs mais une seule valeur indiquant, le plus tôt possible dans ce processus, l'intérêt de la donnée produite pour la résolution de l'objectif. Les valeurs des instanciations jusqu'alors retournées devront ainsi être déterminées au moment de l'acquisition de l'expression de cet objectif. Pour faciliter la mise au point de ces deux processus -filtrage et acquisition de l'objectif-, les expressions fournies au système ou développées par lui, doivent pouvoir être reconnues par des grammaires correctement construites. L'écriture de telles grammaires nous permettra alors d'exploiter des techniques spécifiques (voir chapitre suivant), afin de simplifier le développement des expressions, mais aussi afin de construire ce processus d'acquisition des objectifs qui aura en charge de vérifier leur syntaxe, d'en réduire leur expression et de récupérer les valeurs associées aux descripteurs utilisés. L'étude de ce nouveau processus d'appariement fait l'objet du chapitre 3 (3e partie), le processus d'acquisition étant quant à lui évoqué dans les chapitres suivants et détaillé dans le rapport [Dej96a].

Les temps de calculs devront aussi être un souci lorsque des modifications de modules ou des intégrations de nouveaux opérateurs devront être entreprises.

2.3 Grammaires et langages

Dans cette section, nous allons dans un premier temps passer en revue les grands courants d'étude des grammaires afin de déterminer, dans un second temps, les éléments qu'il sera intéressant d'exploiter pour notre Langage d'Expression d'Objectif, LEO. Des présentations plus approfondies de ces grammaires peuvent être lues dans les ouvrages de Sabah [Sab88], Pollard et Sag [Pol94], Horrocks [Hor87] et Ligozat [Lig94] pour ce qui est du traitement du langage naturel, ou Meyer [Mey92] et évidemment Aho, Sethi et Ullman [Aho91] pour une approche plus informatique du traitement des langages et des grammaires.

3. Le terme de nomination a été préféré à celui de dénomination pour traduire l'action de désigner une description de donnée comme étant la représentation d'un concept de l'application

2.3.1 Traitement du langage naturel

2.3.1.1 Éléments linguistiques et connaissances

Dans son livre ([Lig94]), G. Ligozat fait un inventaire détaillé des connaissances linguistiques mises en œuvre lors du processus de compréhension de phrases en français :

- connaissances phonétiques -étude des sons d'une langue- et phonologiques -étude de la nature des phonèmes (sons) et leur utilisation- liées à la décomposition de la phrase entendue en sons puis recomposition de celle-ci en unités distinctes, identifiables (phonèmes).
- connaissances morphologiques liées à l'association des phonèmes pour former les mots du langage, puis à la reconnaissance de formes que peuvent prendre ces mots dans le langage.
- connaissances syntaxiques liées à l'agencement et à l'utilisation des mots dans la construction d'une phrase.
- connaissances sémantiques sur l'emploi d'un mot dans la phrase mais aussi sur la construction de cette même phrase, celle-ci fournissant une information supplémentaire sur les intentions, sentiments, etc...de son auteur.
- et enfin, connaissances pragmatiques liées à la connaissance du contexte dans lequel est émise cette phrase.

Nous verrons que nous pouvons faire un lien entre certaines de ces connaissances mises en œuvre dans le langage naturel et celles qui concernent les éléments de description de la donnée et de l'objectif de traitement d'images. S'ils ne se trouvent pas tous directement exploités dans LEO, ces liens peuvent fournir des éléments de caractérisation des structures de ce langage pour l'intégrer plus facilement dans des interfaces Homme-Machine adaptées à notre problématique sur l'expression d'objectif.

2.3.1.2 Linguistique structurale

La linguistique structurale (jusqu'à la théorie de Chomsky -1957) est basée sur la décomposition de la phrase en syntagmes, mots, morphèmes puis phonèmes. Cette décomposition est faite indépendamment de tout sens intrinsèque à la phrase et à sa construction. Ceci s'opposait alors à la recherche d'une grammaire universelle tendant à déterminer des principes communs à tout langage, et que Chomsky appelait *structures profondes*.

2.3.1.3 Chomsky et les grammaires génératives

Chomsky, dans *"Syntactic structures" (1957)*, présente une nouvelle approche du traitement des langages naturels par l'utilisation de grammaires formelles (voir page 49). L'objectif dans les grammaires génératives, comme l'explique Sabah [Sab88], est alors d'appliquer les méthodes classiques à l'étude de la syntaxe menée dans les grammaires formelles. L'opposition entre grammaires traditionnelles et grammaire universelle n'a alors plus lieu d'être. Dans [Hor87], Horrocks définit les grammaires génératives comme étant des grammaires totalement explicites -c'est à dire complètement définies- et répondant par *oui* ou par *non* à une question portant sur l'appartenance d'une chaîne de mots à l'ensemble des phrases du langage étudié.

2.3.1.4 Grammaires transformationnelles

L'apparition et l'exploitation de la notion de structures profondes dans les grammaires formelles a permis à Chomsky de proposer ce que l'on a appelé *la théorie standard*[4] dont l'idée principale

4. Standard Theory

est l'introduction dans les grammaires formelles :

- des composants sémantiques permettant de construire des structures du même nom pour l'interprétation de la phrase,
- des règles de transformations des structures profondes,
- des composants phonologiques permettant, après transformation de structures profondes en une structure de surface d'obtenir une phrase phonologiquement correcte (lire [Hor87]) ; une structure de surface correspondant à une représentation non-terminale d'un développement ultime d'une structure profonde.

L'objectif de ces grammaires transformationnelles est de fournir à partir d'une même structure profonde issue de l'analyse d'une phrase, des structures de surfaces correspondant à des formes différentes de la phrase initiale. Pour illustrer cette propriété, nous pouvons reprendre l'exemple de Sabah [Sab88] :

Phrase initiale	:	*Le garçon lit un livre*
Forme interrogative	:	*Qui lit un livre ?*
Forme passive	:	*Un livre est lu par le garçon*

Une règle de transformation est alors constituée de trois éléments principaux qui sont :

1. une description structurelle correspondant à la séquence de non-terminaux de la structure profonde à transformer,
2. un changement structurel correspondant à la séquence de substitution dans cette même structure profonde,
3. et éventuellement des conditions d'application permettant de traiter des cas particuliers de transformation.

Dans l'exemple suivant -toujours tiré de [Sab88]- nous montrons la transformation d'une forme active d'une phrase en sa forme passive.

Phrase initiale	:	*Le garçon*		*lit*	*un livre*
Structure de surface	:	*Le garçon*	présent	lire	*un livre*
Description structurelle	:	GrNominal	Aux	GrVerbal	GrNominal
		1	2	3	4
Changement structurel	:	4	2 > être + part.passé	3	*par* # 1
Forme passive	:	*Un livre*	*est*	*lu*	*par le garçon*

L'intérêt principal de ce type de grammaires réside dans la possibilité de manipuler des expressions sur plusieurs niveaux de représentations. En l'occurence, pour notre langage, nous pouvons modéliser la notion de remplacement des noms de sous-objectifs par leurs expressions étendues, en développant des règles particulières afin de modifier les structures des éléments du langage d'expressions d'objectifs (voir 2.5.1). Ainsi l'écriture, le développement et la compréhension des expressions de LEO pourront être représentés et facilités par ces remplacements.

2.3.1.5 Grammaires syntagmatiques

Certaines des règles de transformation sont soit difficiles à expliciter, soit trop lourdes à mettre en œuvre parce qu'elles font appel à trop de règles de production de la structure profonde de la phrase. Un exemple est la place de l'adverbe dans une phrase simple constituée d'un groupe nominal GN, d'un groupe verbal GV et d'un adverbe Adv :

$$\begin{array}{lll} \mathrm{P} & \longrightarrow & \mathrm{Adv\ GN\ GV} \\ \mathrm{P} & \longrightarrow & \mathrm{GN\ Adv\ GV} \\ \mathrm{P} & \longrightarrow & \mathrm{GN\ GV\ Adv} \end{array}$$

L'idée maîtresse des grammaires syntagmatiques[5] puis des grammaires syntagmatiques guidées par les têtes[6] ([Pol94]) -typage des structures restreignant les possibilités des GPSG- consiste à rendre compte par l'utilisation de traits [Lig94] de certaines modifications dans la phrase en restant dans le cadre des structures de surface, c'est à dire sans utiliser les règles de transformation des grammaires précédentes.
Les règles de production de ces grammaires sont constituées de deux entités appelés *dominance immédiate* et *précédence linéaire* fournissant, pour la première, la liste des items terminaux ou non-terminaux de la partie droite, sans ordre établi, et pour la seconde, les caractéristiques d'ordonnancement de ces items. Ainsi, pour l'exemple précédent, la production équivalente est :

Dominance immédiate	:	P $\longrightarrow$ Adv,GN,GV
Précédence linéaire	:	GN $\leq$ GV

Ces grammaires peuvent être considérées comme des extensions des grammaires transformationnelles. Les modifications dans les structures des expressions de LEO étant suffisamment simples pour être gérées par des règles transformationnelles, les principes des grammaires syntagmatiques ne seront pas utilisés tant que les manipulations effectuées consistent en des développements d'expressions. Cependant, il est possible que l'intégration de la structure de la grammaire de LEO dans le processus de reconnaissance d'expressions -processus d'appariement (voir chap.3, 3^{e} partie)- passe par l'utilisation des propriétés de ces grammaires.

2.3.1.6 Autres grammaires

Il faut enfin citer d'autres courants d'étude des grammaires appliquées essentiellement au langage naturel.
Les *grammaires d'unification* exploitent la notion de structure de traits pour la représentation de caractéristiques sémantiques dans des grammaires manipulant des structures de surface.
Les *grammaires de cas* étiquettent les éléments terminaux des phrases afin de fournir des caractéristiques sémantiques -cas AGENT, CONTRE-AGENT, OBJET, RÉSULTAT, etc...- permettant d'informer sur les rôles des éléments de la phrase.
Les *grammaires systémiques*, dans un même ordre d'idée, vont fournir des informations quant aux structures de la phrase. Ces informations, selon les systèmes développés pourront être de type sémantique (Groupe Nominal sujet, objet) amenant à des informations plus syntaxiques (Forme passive, active).
Enfin, les *grammaires de clauses définies* ([Per80]) tirent partie de notions de la programmation logique et de la syntaxe utilisée dans PROLOG : les parties déclaratives et procédurales d'une clause de ce langage correspondant aux membres gauche et droit, respectivement, des règles de production des grammaires formelles classiques.

Ces grammaires manipulent des entités particulières que nous n'avons pas à considérer pour notre application tant la syntaxe de LEO est bien formée et gère des concepts sémantiques de bas niveau : propriétés de sélection, relations, mesures, etc.

2.3.2 Langages et grammaires formels

2.3.2.1 Présentation

Le principe essentiel des grammaires formelles réside dans leur conception modulaire et hiérarchique permettant le traitement de langages complexes. Une grammaire formelle G est alors définie

5. GPSG : Generalized Phrase Structure Grammar
6. HPSG : Head-Driven Phrase Structure Grammar

par Chomsky comme constituée de quatre entités :

Un vocabulaire non terminal. Il correspond à l'ensemble des symboles utilisés pour décrire le langage étudié. Ce vocabulaire est souvent noté N et appelé *ensemble des non-terminaux* de G.

Un vocabulaire terminal. Il correspond à l'ensemble des symboles que nous trouvons dans les expressions du langage étudié. Ce vocabulaire est souvent noté Σ et appelé *ensemble des terminaux* de G.

Des règles de réécriture. Les règles de cet ensemble vont permettre de générer ou reconnaître les phrases bien formées[7] du langage ; une règle R établissant une relation de substitution entre un élément P_G -partie gauche- non vide ($P_G \neq \lambda$) et un autre élément P_D -partie droite- du monoïde libre formé sur l'union des vocabulaires terminal et non-terminal.

$$R = P_G \longrightarrow P_D,\ avec\ (P_G, P_D) \in ((N \cup \Sigma)^*)^2\ et\ P_G \neq \lambda$$

Cet ensemble de règles est souvent noté P et appelé ensemble des *règles de production* du langage.

Un symbole origine. Ce symbole est un élément particulier du vocabulaire non terminal par lequel débute tout processus de génération ou de reconnaissance d'expression du langage. Cet élément, aussi appelé *axiome* est noté S.

La grammaire formelle G se notera alors par le quadruplet :

$$G = (N, \Sigma, P, S)$$

Leur simplicité de représentation et la rigueur qu'elles proposent font que notre choix de grammaire pour LEO va se porter sur des grammaires formelles. Nous pourrons ainsi exploiter de nombreux outils et algorithmes qui ont été développés tels que les tables d'analyse avec récupération sur erreur (voir Annexe B).

2.3.2.2 Classification selon Chomsky

Chomsky détermine quatre types -notés de 0 à 3- de grammaires formelles selon qu'elles vérifient une ou plusieurs des propriétés suivantes. L'imbrication de ces types de grammaires peut être résumée dans le schéma de la figure ci-après (fig.2.7).

Type 0 : Grammaire formelle.

Type 1 : Grammaire contextuelle. Une grammaire formelle est dite contextuelle si quelle que soit une règle de production de P, sa partie gauche a une longueur, exprimée en nombre de symboles, inférieure à la longueur de sa partie droite.

Type 2 : Grammaire non-contextuelle ou algébrique. Une grammaire formelle est dite non contextuelle[8] si ses règles de production possèdent un élément en partie gauche appartenant au vocabulaire non terminal N, et non plus au monoïde $(N \cup \Sigma)^*$

Type 3 : Grammaire régulière ou rationnelle. Une grammaire non-contextuelle est dite régulière à droite ou à gauche si ses règles de production sont de la forme :

régulière à gauche $A \longrightarrow \alpha B,\ avec\ (A, B) \in N^2\ et\ \alpha \in \Sigma$
régulière à droite $A \longrightarrow B\alpha,\ avec\ (A, B) \in N^2\ et\ \alpha \in \Sigma$

7. Well-Formed Sentences
8. Context-Free

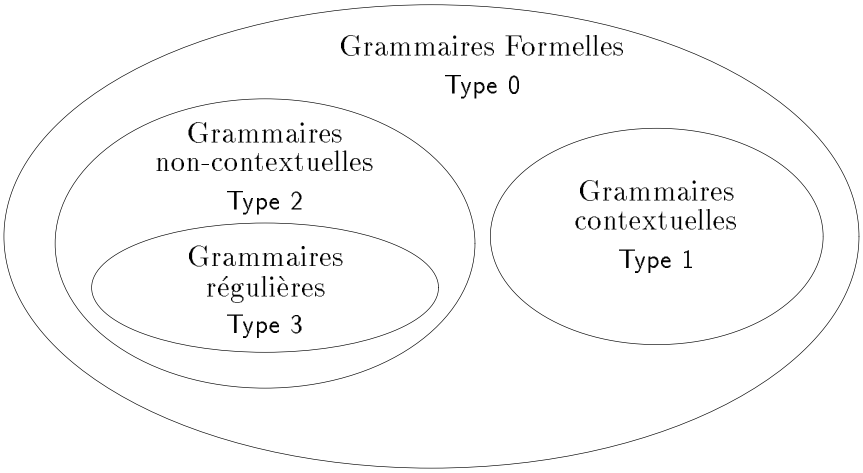


FIG. 2.7 - *Types des grammaires formelles*

Parmi les divers types de grammaires formelles, celles de type 3, plus strictes quant à la forme des règles de production, offrent de nombreuses facilités de mise en œuvre de processus de traduction, de reconnaissance, de transformation et de manipulation pour intégrer dans un système tel que ExTI des modules rigoureux d'analyse syntaxique, des modules d'appariement, etc. Nous allons donc essayer d'amener les expressions manipulées à être reconnues par de telles grammaires.

2.3.2.3 Réseaux de transitions

Dans un processus d'interprétation, d'analyse ou de génération d'une expression d'un langage formel, l'ensemble des règles de production utilisées forme ce que l'on appelle un *arbre syntaxique* ou *réseau de transitions* dans lequel chaque feuille est décorée d'un symbole terminal, chaque nœud intermédiaire est décoré d'un symbole non-terminal et chaque arc correspond à la production utilisée (fig.2.8).

P $\longrightarrow$ SN SV
SN $\longrightarrow$ GN
SV $\longrightarrow$ GV GN
GN $\longrightarrow$ Art Nom
GV $\longrightarrow$ Verbe
Art $\longrightarrow$ *le, un, ...*
Nom $\longrightarrow$ *garçon, livre, ...*
Verbe $\longrightarrow$ *lit, ...*

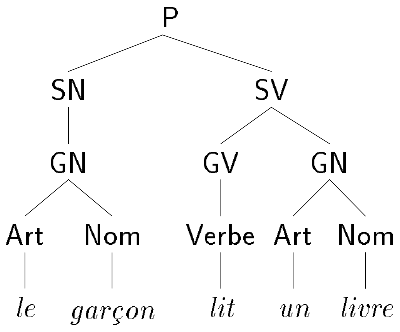


FIG. 2.8 - *Exemple d'arbre syntaxique*

Réseaux de transition augmentés.
L'absence de toute considération sémantique dans les grammaires formelles et la structure arborescente des résultats d'analyses d'expressions du langage ont conduit à la décoration des arcs de ces réseaux à l'aide de variables (registres, piles, etc) permettant le déclenchement d'actions ou de tests lors de l'évaluation des transitions. Cette nouvelle décoration conduit alors à transformer un arbre syntaxique en *réseau de transition augmenté*.

Ces notions de transitions augmentées présentent un intérêt particulier en rapport avec la simplification des expressions fournies comme objectif à atteindre. En effet, par le biais du processus de contrôle de la syntaxe de l'expression établissant l'arbre syntaxique de cette expression, nous pourrons déclencher les règles transformationnelles permettant de gérer ces simplifications.

2.3.2.4 Grammaires à attributs

On appelle *grammaire à attributs*, ou encore *définition dirigée par la syntaxe*, une grammaire non contextuelle (formelle de type 2-3) dans laquelle :

- tout symbole terminal ou non-terminal possède une liste de champs ou *attributs* correspondant à des caractéristiques sémantiques liées à ce symbole.
- à toute règle de production correspond des règles sémantiques fournissant des informations sur les symboles utilisés dans la règle.

Lors de la construction ou l'analyse d'un réseau de transition, les valeurs des champs sont alors données, en chacun des nœuds, par les règles sémantiques liées aux règles de production utilisées. Comme précédemment, la notion de sémantique est basée sur la syntaxe du langage. Mais il s'agit, dans le cas présent, d'une extension aux symboles d'ajouts de caractéristiques sémantiques. Il faut noter que ce type de grammaire est très utilisé dans la conception de compilateurs [Mey92],[Aho91].

Comme précédemment, le processus d'analyse syntaxique va exploiter cette notion d'attributs. Ceux-ci vont être liés aux descripteurs utilisés dans LEO. Ces attributs vont permettre de récupérer les valeurs associées à ces descripteurs afin d'instancier correctement les opérateurs symboliques génériques, c'est à dire d'initialiser les paramètres des opérateurs de la chaîne de traitement (voir §2.5.2 de la deuxième partie).

2.3.3 Caractéristiques de LEO

La structure de LEO va donc reposer sur des grammaires formelles de type 3 nous permettant de construire des expressions de façon rigoureuse et de développer des systèmes de reconnaissance -arbre syntaxiques- ou d'analyse -tables d'analyses (Annexe B).

L'exploitation de la notion d'attributs ou affixes, permettra, par le biais d'un analyseur syntaxique, de configurer le système intégrant ce langage. Dans le cas d'ExTI, il s'agira entre autres de déterminer les domaines de validité des sélecteurs manipulés (Annexe C) et de récupérer les valeurs numériques permettant certaines initialisations d'opérateurs.

Enfin, la notions de transition augmentée associé aux règles transformationnelles des grammaires du même nom, permettra, toujours par le biais de l'analyseur syntaxique, de passer facilement d'expressions de haut-niveau mettant en œuvre des termes du domaine d'application à des expressions de plus bas-niveau d'écriture. Des règles de simplifications de ces expressions permettront alors d'améliorer le processus de reconnaissance d'expressions (voir chap.2 et 3 de la troisème partie).

Deuxième partie

Modèles proposés

Dans cette partie, nous allons détailler une nouvelle approche de la représentation de la donnée (chap.1). Celle-ci a été implantée dans une nouvelle version du système ExTI [Ing92] qui constitue le noyau d'un processus de coopération Homme-Machine construisant des applications de traitement d'images. Il lui procure un formalisme bas-niveau de description d'images pouvant être couvert par un langage plus représentatif du domaine d'application. Nous déterminerons et détaillerons ensuite le type de grammaire utilisable pour développer un Langage d'Expression d'Objectif (LEO) en terme d'indices visuels (chap.2). Puis nous étudierons le modèle d'opérateur adapté à ces représentations (chap.3) pour terminer par extraire les caractéristiques d'un module de planification permettant d'intégrer tous les concepts proposés (chap.4).

Dans les trois modèles présentés ici nous avons utilisé abondamment les lettres grecques. Ce choix permet de bien séparer les éléments théoriques -génériques-, représentés par ces lettres, des éléments instanciés. L'ensemble de ces notations est disponible en annexe A.

Chapitre 1

Un modèle pour les données

Le modèle présenté ici s'insère dans le processus de descriptions symboliques de données de traitement d'images exprimées en termes d'indices visuels. Ce processus permet ainsi de décrire des objectifs qui devront pouvoir être reconstruits par planification d'opérateurs par un système de résolution de problème.
Comme tout vecteur de communication, une image peut être considérée comme étant composée de deux éléments:

Le support : c'est la grille rectangulaire ou hexagonale en traitement d'images qui va permettre de manipuler et de véhiculer le signal.

L'information : il peut s'agir d'une fonction mathématique, d'un simple codage, ou de tout autre système de représentation.

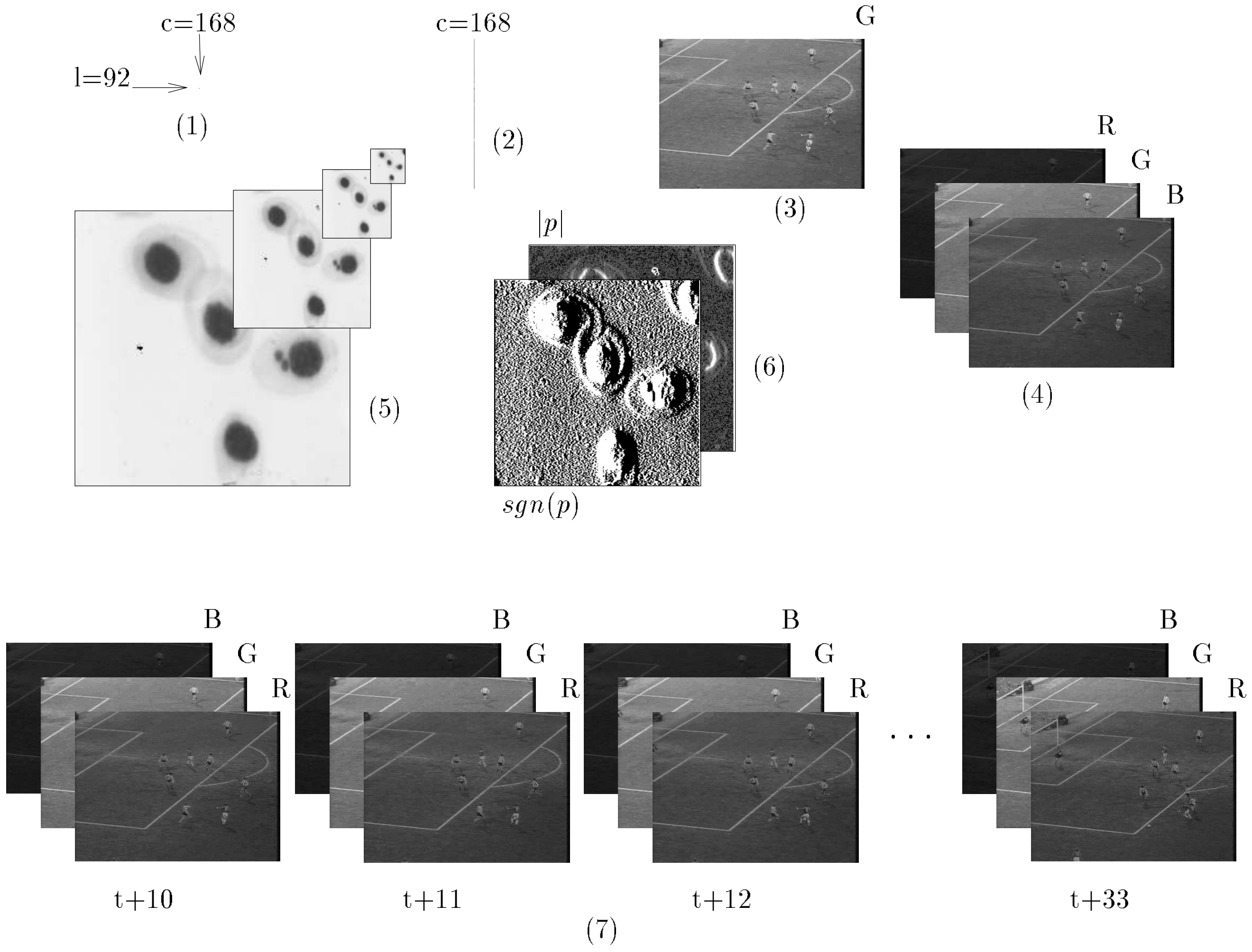
Lorsqu'on examine le travail effectué par un expert du domaine au cours d'un processus d'interprétation d'images, on s'aperçoit qu'il occulte, consciemment ou non, certaines parties de cette image. Les parties délaissées et celles qui déterminent une bonne interprétation peuvent être considérées comme des composantes particulières du support, masques ne laissant apparaître que l'information à traiter. Un exemple simple consiste à écarter, lors d'une détection d'objets déterminés, des indices visuels touchant les bords d'une image.
À partir de cette idée de cache laissant apparaître l'information pertinente et de la décomposition d'image en un couple (support, information), nous avons développé ce nouveau modèle de représentation de l'objet image et des opérateurs qui le manipulent. Si nous le comparons au modèle précédemment implanté dans le système ExTI, il permet de mieux rendre compte de la nature des transformations et de leurs incidences sur les objets de par la décomposition plus fine des données manipulées. De plus, il offre des possibilités nouvelles pour la conception du planifieur de chaînes

d'opérateurs et pour la mise en œuvre de plusieurs stratégies du fait de son unicité, quel que soit le type et la dimension des entités manipulées.

1.1 Éléments mathématiques

Pour décrire ce modèle, nous utiliserons les termes de trame, de cache, d'index, de signal et de codage qui deviendront ainsi les éléments constituant la donnée que nous souhaitons manipuler. Avant de détailler le modèle lui-même, il nous faut définir trois notions que nous retrouverons à tous les niveaux de la description : la dimension, le point et le référentiel.

Cette donnée que nous allons génériquement noter D dans la suite du mémoire, n'est pas simplement une image. De ce fait, le modèle que nous proposons permet de représenter tout type de données manipulées par des opérateurs de traitement, allant du simple pixel ou valeurs numériques à la séquence d'images vidéo couleurs en passant par les codages de Freeman, les chaînes 2-D de Chang (*2-D strings* [Cha87a, Jun93][Lee91]) ou les deux plans $(|p|, sgn(p))$ d'une image de différence.



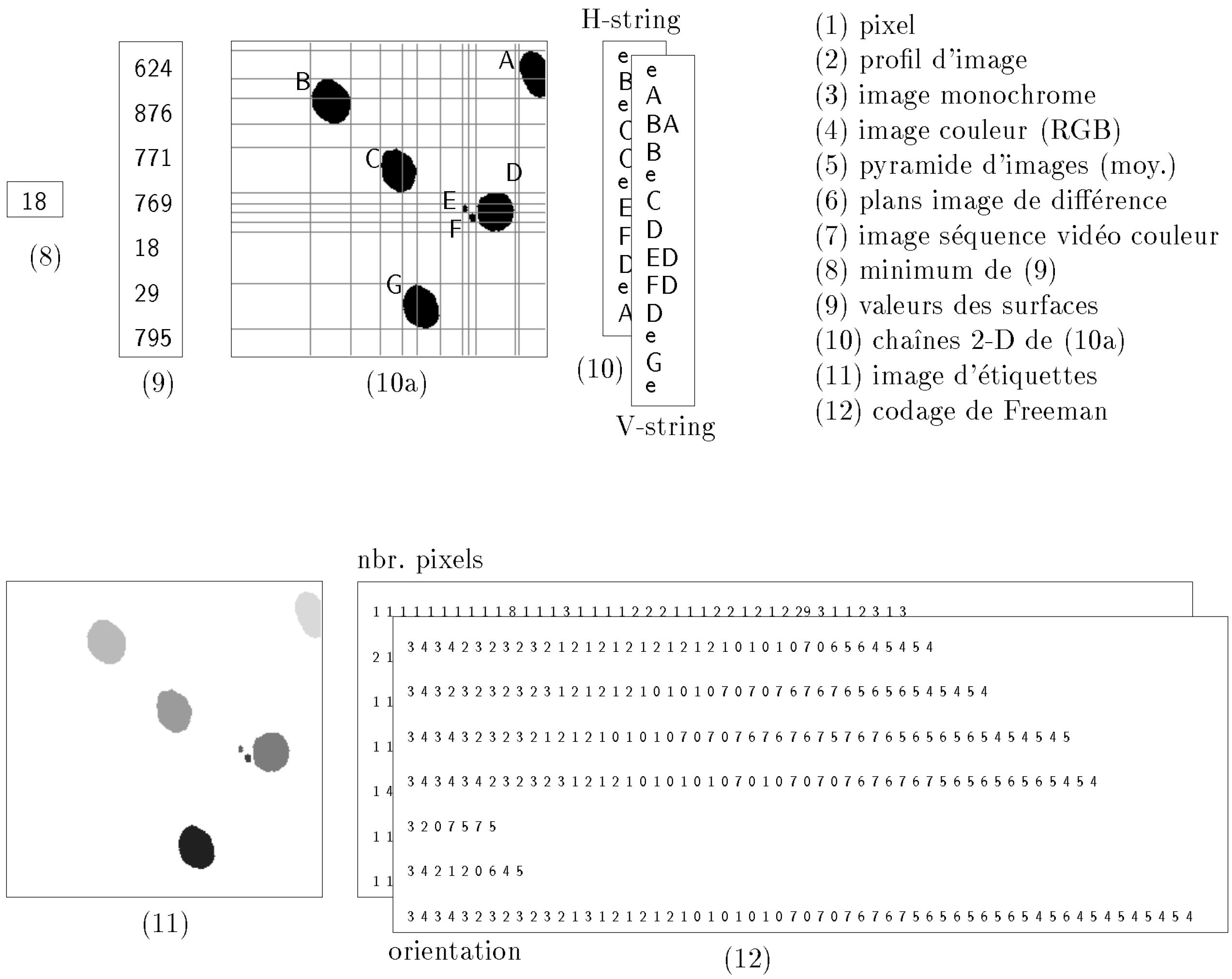


FIG. 1.1 - *Diverses données de traitement d'images*

1.1.1 La dimension

La dimension est le nombre d'éléments de la base de l'espace vectoriel supportant la donnée. Soit D^n une donnée de traitement d'images. Nous dirons que D^n est de dimension n, avec $n \in \mathbb{N}$, si tout élément π^n de D^n est déterminé de façon unique par un n-uplet appelé coordonnées de π^n dans D^n, ces coordonnées étant exprimées dans un repère R_{D^n} lié à D^n.
La donnée de traitement d'images peut correspondre alors, comme nous venons de le voir dans la figure 1.1, à l'ensemble des plans d'une séquence d'images RVB (n=4), aux plans images RVB, aux deux plans d'une image complexe, ... (n=3), à une image de niveaux de gris, une coupe d'image couleur, ... (n=2), à une coupe d'image ou au tableau de valeurs, ... (n=1) ou enfin à une valeur de mesure ou à une intensité de pixels, ... (n=0). Ce dernier cas, particulier, permet d'uniformiser le modèle. En effet nous pouvons établir une analogie entre une considération (support, information) du signal image et la variable numérique -mesure sur une donnée, coordonnée, ...- au sens informatique du terme, qui consiste en un doublet (identificateur, valeur).

1.1.2 Le point

Le point désignera cette entité π^n la plus élémentaire de la donnée de traitement d'images. Soient π^n un point de la donnée D^n à traiter, (plan image, image, ...fig.1.2) et n la dimension de D^n, nous dirons que π^n est de dimension n s'il existe un unique n-uplet $(x_1, \cdots, x_n)$ de $\mathbb{N}^n$ déterminant les coordonnées de π^n dans R_{D^n}.

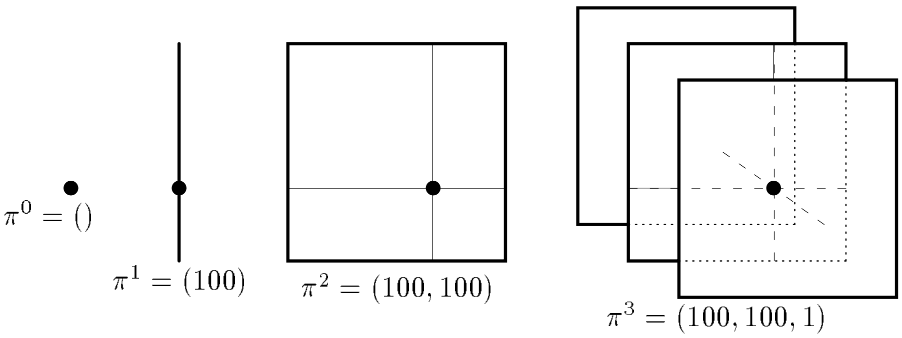


FIG. 1.2 - *Le point dans les données*

1.1.3 Le référentiel

Comme chaque donnée d'une même dimension n possède son propre repère R_{D^n}, nous allons par la suite appeler référentiel l'entité géométrique Υ^n de dimension n sur laquelle nous définirons le repère $R^n = (0, \vec{\imath_0}, \cdots, \vec{\imath_n})$. Pour toute dimension n fixée, ce référentiel sera considéré comme unique et invariant pour tout traitement, R^n étant lui-même invariant sur Υ^n.

$$\Upsilon^n = (n, R^n) \tag{1.1}$$

Pour pouvoir exprimer simplement les transformations d'informations entre ces différentes dimensions de données (exemple : sélection d'une image d'une séquence), nous posons la contrainte d'unicité des vecteurs $\vec{\imath_x}$ avec $x \in [1, n]$ sur les repères R^n pour tout $n \in \mathbb{N}$ (fig 1.3). Cette contrainte permet ainsi d'exprimer un repère R^n à l'aide des composantes du repère $R^{m=(n-1)}$ de la façon suivante :

$$R^n \equiv (0, \{\vec{\imath}_{x\in[1,m]}\}, \vec{\imath}_n) \tag{1.2}$$

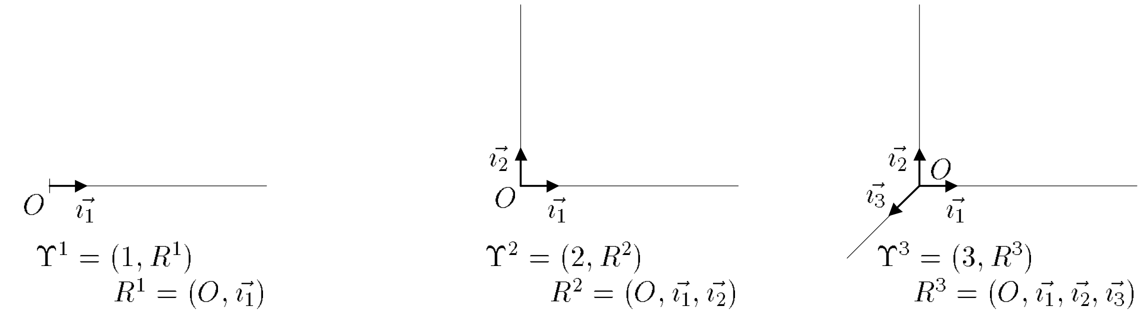


FIG. 1.3 - *Contrainte sur les référentiels*

Cette notion de référentiel est d'autant plus importante qu'elle va conditionner les expressions des descriptions de données lors de traitements liés à leurs supports. En effet de telles opérations de traitement d'images considèrent les données selon une même référence de point ; la notion de repère de la donnée que nous avions introduite n'intervenant qu'à de rares exceptions -détermination et extraction de sous-images, par exemple- dans les conditions opératoires. Nous pourrons alors effectuer des transformations passant d'une dimension à une autre sans changement de repère associé au référentiel (fig 1.4), mais aussi exprimer des changements de support tels qu'une translation d'image (fig 1.5).

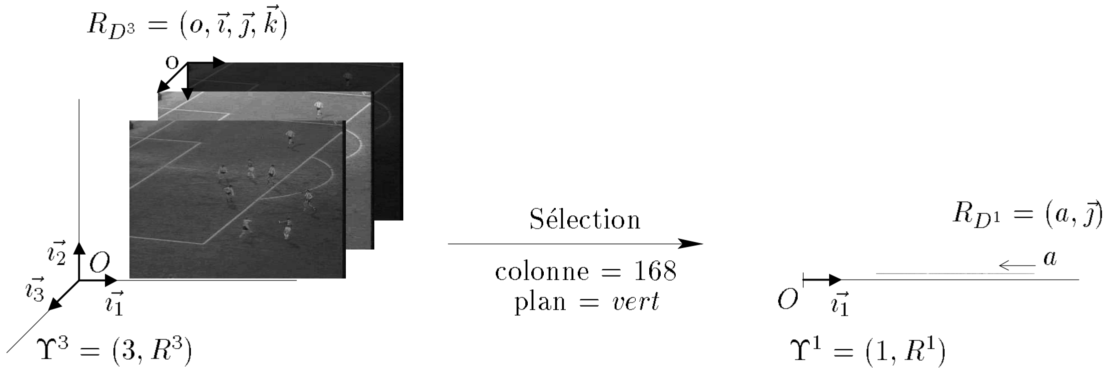


FIG. 1.4 - *Conséquences d'un changement de dimension*

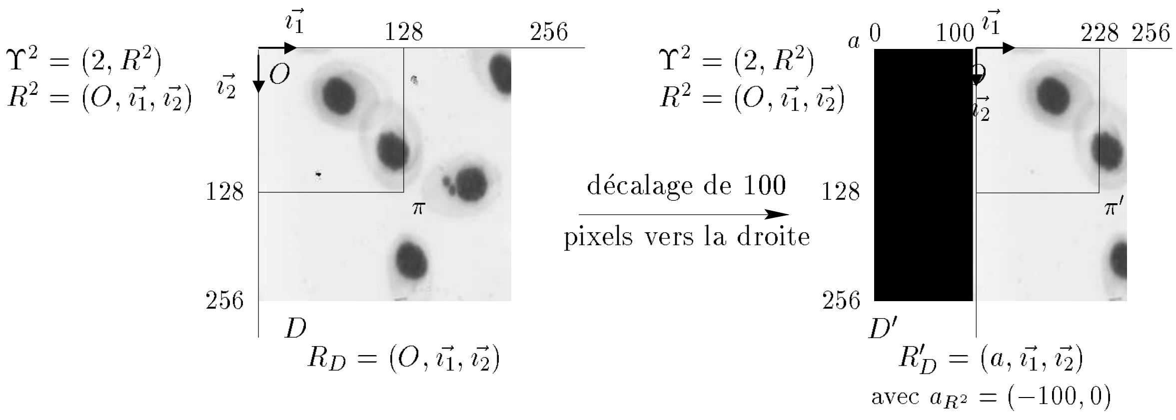


FIG. 1.5 - *Conséquences d'une translation*

Remarque : Pour plus de simplicité dans l'écriture, et lorsqu'il n'y aura pas d'ambiguïté, nous omettrons de mentionner la dimension n des entités rencontrées.

1.2 Structure du modèle de données

Toute donnée sera considérée sous la forme d'une structure complexe constituée de deux ensembles décrivant, pour l'un les parties de la donnée D correspondant aux zones d'intérêt et, pour l'autre, correspondant au complémentaire de ces mêmes zones. Par la suite, nous appellerons D le premier de ces ensembles, $\overline{D}$ désignant son complémentaire. Chacun de ces deux ensembles est constitué des cinq éléments suivants : la Trame, le Cache, l'Index, le Signal et le Codage.

1.2.1 La Trame τ

Définition : Une donnée D de traitement d'images est supportée par une trame τ, de même dimension, définie par l'ensemble des points π de D. Toute donnée étant finie, il existe pour chaque dimension i de D un intervalle fini $[I_i, S_i[$ de $\mathbb{N}$ tel que la composante x_i des coordonnées de π dans D prend ses valeurs dans cet intervalle. Cette trame τ est donc une restriction finie du référentiel Υ de même dimension n (fig.1.6).

$$\tau^n = \{\pi^n = (x_1, \cdots, x_i, \cdots, x_n) \ / \ \forall n \in \mathbb{N}, \forall i \in [1, n], \exists (I_i, S_i) \in \mathbb{N}^2, x_i \in [I_i, S_i[\ \} \tag{1.3}$$

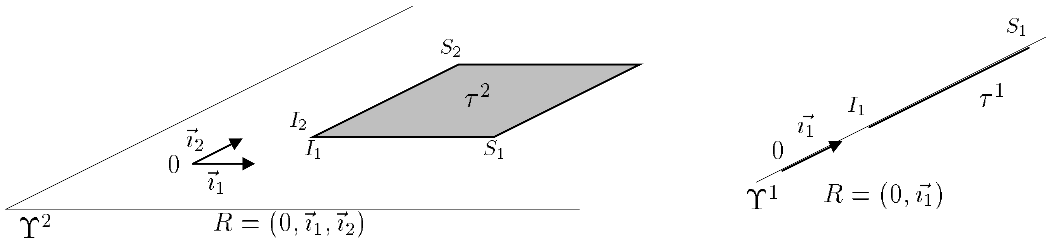


FIG. 1.6 - *Trames τ^2 et τ^1*

Valeurs des points et des intervalles $[I_i, S_i[$: Les valeurs caractéristiques I_i et S_i de la trame s'expriment dans le repère R du référentiel. De leur côté les points d'une donnée possèdent des coordonnées exprimées dans le repère même de la donnée. Cette double référence garantira la conservation de l'information liée au support de la donnée lors de traitements successifs.
Dans l'exemple de la figure 1.5, les trames des données D et D' ont pour expression :
pour D : $\tau = \{[0\ 256[, [0\ 256[\}$
pour D' : $\tau = \{[-100\ 156[, [0\ 256[\}$
tandis que les points π et π' ont pour coordonnées :
$\pi = (128, 128)$
$\pi' = (228, 128)$

Taille d'une trame : Nous appellerons *taille de la trame* τ de dimension n le n-uplet $\mathcal{T}(\tau^n) = (t_1, \cdots, t_i, \cdots, t_n)$ dans lequel chaque t_i est égal à la différence des bornes du couple (I_i, S_i) déterminant l'intervalle de validité $[I_i, S_i[$ de la composante x_i de tout point π de τ ($\mathcal{T}(\tau^n) \in \mathbb{N}^n$).

$$\mathcal{T}(\tau^n) = \{(S_i - I_i)_{i \in [1,n]} \ / \ \forall \pi^n = (x_1, \cdots, x_i, \cdots, x_n) \in \tau^n, x_i \in [I_i, S_i[\ \} \tag{1.4}$$

Toute trame τ peut être alors déterminée par sa dimension et les couples déterminant sa taille (fig.1.7). La notation proposée indique donc à la fois la taille de la trame, sa dimension et la position dans Υ de son repère R_D. Il faut de plus noter que le point origine O des repères du référentiel peut ne pas appartenir à la trame elle-même.

$$\tau^n = \Big(\ (I_1, S_1), \cdots, (I_i, S_i), \cdots, (I_n, S_n) \ \Big) \tag{1.5}$$

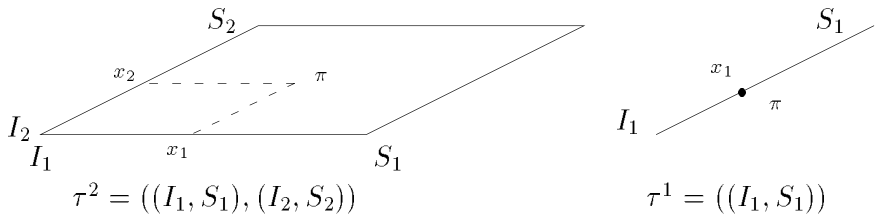


FIG. 1.7 - *Trame de dimension 2, Trame de dimension 1*

Ensemble des trames : Nous appellerons T_n l'ensemble des trames τ de dimension n, c'est à dire l'ensemble des trames dont les points ont un système de coordonnées à n composantes caractéristiques :

$$T_n = \Big\{ \ \tau^n \Big/ \ \forall \pi^n \in \tau^n, \pi^n = (x_1, \cdots, x_i, \cdots, x_n) \ \Big\} \tag{1.6}$$

1.2.2 Le Cache κ

1.2.2.1 La fonction cache

Comme nous l'avons évoqué dans l'introduction de cette section, le processus d'interprétation d'images ne s'intéresse qu'à certaines parties de l'image. Ces zones d'intérêt correspondent à des éléments précis de l'application (concepts), identifiables par l'expert qui peut en donner une description -que nous noterons δ- formulée en termes d'indices visuels [Ing92].
La description d'un concept [Ing96] est alors considérée comme une WFF [1] composée de termes regroupant les propriétés ou les relations sur d'autres expressions. Les propriétés correspondent à une adaptation de la notion de sélecteurs décrits par Michalski [Mic83, Mic84] qui définit le calcul des prédicats annotés. Elles sont composées d'un descripteur *(surface, niveau de gris, distance au bord, ...)* et d'un domaine déterminant leurs valeurs associées et constitué d'union d'intervalles semi-ouverts [x,y[:

$P = [p\ d]$: propriété (appelé sélecteur par Michalski)
p : descripteur de la propriété
d : domaine de la propriété

Le domaine de toute propriété P est inclus dans ce que nous nommerons le domaine de validité Δ_p associé au descripteur p de P (voir Annexe C). Pour donner un exemple, l'expression

```
((surface 100 200) ((vois8) (((nivg 0 128)))))
```

définit un ensemble de composantes connexes constituées des pixels de l'image d'intensité comprise dans l'intervalle [0,128[formant des amas 8-voisins [2] dont les surfaces sont comprises dans l'intervalle [100,200[.

La fonction cache f_δ, à valeurs booléennes, va mettre en jeu de telles descriptions et va permettre de créer des caches applicables au référentiel de la donnée D traitée. Une telle fonction s'applique

1. Well-Formed Formula
2. distance *chessboard*

donc aux points π de Υ sur laquelle est définie D et détermine si la description de chacun d'eux, notée $\Lambda_\Upsilon(\pi)$, est une spécialisation -au sens de Michalski- de la description δ des zones d'intérêt de D.

$$f_\delta \left| \begin{array}{l} \Upsilon \longrightarrow \{\mathsf{True}, \mathsf{False}\} \\ \pi \longmapsto (\Lambda_\Upsilon(\pi) \subseteq \delta) \end{array} \right. \tag{1.7}$$

Si nous désignons par ∇_Υ l'ensemble des descriptions δ valides -sémantiquement correctes- dans Υ, nous pouvons définir l'ensemble F_Υ des fonctions caches définies sur Υ, c'est à dire l'ensemble des fonctions f_δ telles que δ soit un élément de ∇_Υ.

$$F_\Upsilon = \left\{ f_\delta \middle/ \delta \in \nabla_\Upsilon \right\} \tag{1.8}$$

1.2.2.2 Le Cache

Définition : Le cache à description δ, que nous noterons κ_δ, défini sur la donnée D, est déterminé par l'ensemble des points π de τ vérifiant la fonction cache f_δ. Il a donc la même dimension que la trame et fournit une information de type géométrique ou morphologique sur les éléments pertinents de la donnée (fig.1.8). Il correspond donc à un sous-ensemble des points de la trame ayant tous les mêmes caractéristiques que l'on retrouve dans la description δ.

$$\kappa_\delta = \left\{ \pi \in \tau \middle/ f_\delta(\pi) = \mathsf{True} \right\} \tag{1.9}$$

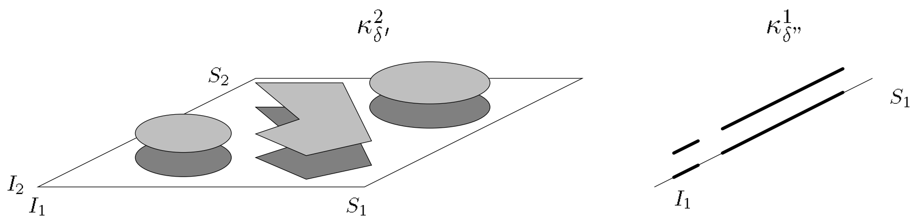


FIG. 1.8 - *Cache 2D à description δ', Cache 1D à description δ"*

Cache complémentaire : Chaque descripteur de propriété p des prédicats de description possédant un domaine de validité Δ_p, il est possible de déterminer la propriété inverse $\overline{P}$ de P définie comme :

$$\overline{P} = \overline{[p\ d]} = [p\ \overline{d}]\ avec\ \overline{d} \cup d = \Delta_p \tag{1.10}$$

Par utilisation des règles du calcul des prédicats, il nous est donc possible de construire les complémentaires $\overline{\delta}$ des descriptions δ manipulées.

Nous appellerons $\overline{\kappa_\delta}$ le complémentaire du cache κ_δ défini sur la trame τ de la façon suivante (fig.1.9) :

$$\begin{aligned} \overline{\kappa_\delta} &= \left\{ \pi \in \tau \middle/ f_\delta(\pi) = \mathsf{False} \right\} \\ &= \left\{ \pi \in \tau \middle/ f_{\overline{\delta}}(\pi) = \mathsf{True} \right\} \\ &= \kappa_{\overline{\delta}} \end{aligned} \tag{1.11}$$

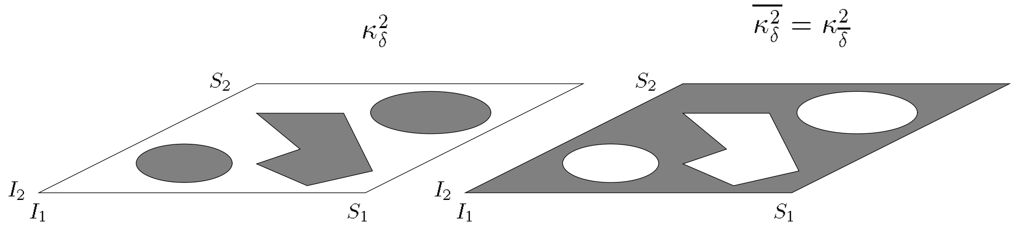


FIG. 1.9 - *Caches κ_δ^2 et $\overline{\kappa_\delta^2}$*

Cache, fonction cache et référentiel : Il faut bien considérer le fait qu'un cache κ est la restriction à sa trame associée τ de l'application de sa fonction cache f_δ à tous les points π de l'entité géométrique sur laquelle est déterminée τ. La description δ peut donc très bien correspondre à des points n'appartenant pas à la trame et donc au cache (fig.1.10).
C'est ainsi qu'un utilisateur pourra décrire des entités d'une image en omettant (volontairement

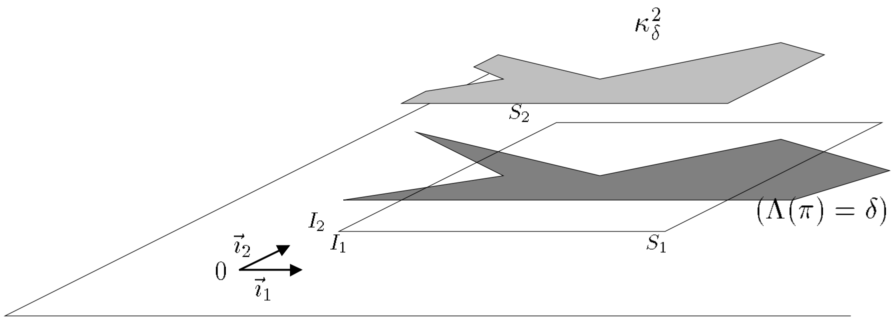


FIG. 1.10 - *Cache et fonction cache*

ou non) que celle-ci est une réduction spatiale de la scène observée. Ainsi dans les images de la figure 1.11, une description des noyaux par une expression du type *objets sombres* sera suffisante pour décrire les noyaux des cellules de l'image de gauche alors qu'elle sera trop générale pour les discriminer dans celle de droite, car elle inclura alors les artéfacts (petites tâches sombres externes aux cellules).

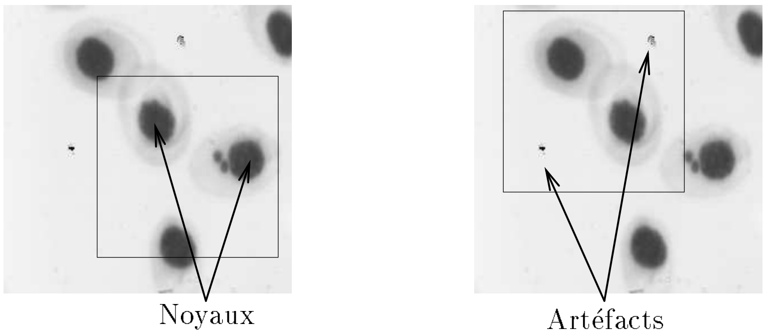


FIG. 1.11 - *Scène observée et images extraites*

Ceci peut alors poser des problèmes lors de l'utilisation d'une même description sur un ensemble d'images extraites d'une même scène.
La description d'un cache, pour toutes les propriétés relatives aux points de la donnée sur laquelle il est appliqué, s'exprime dans le repère de celle-ci. Ainsi, si nous reprenons l'exemple de la figure 1.5 (fig. 1.12), l'expression des pixels de D conservés par translation est : $\delta = (coord_x\ 100\ 256)$.

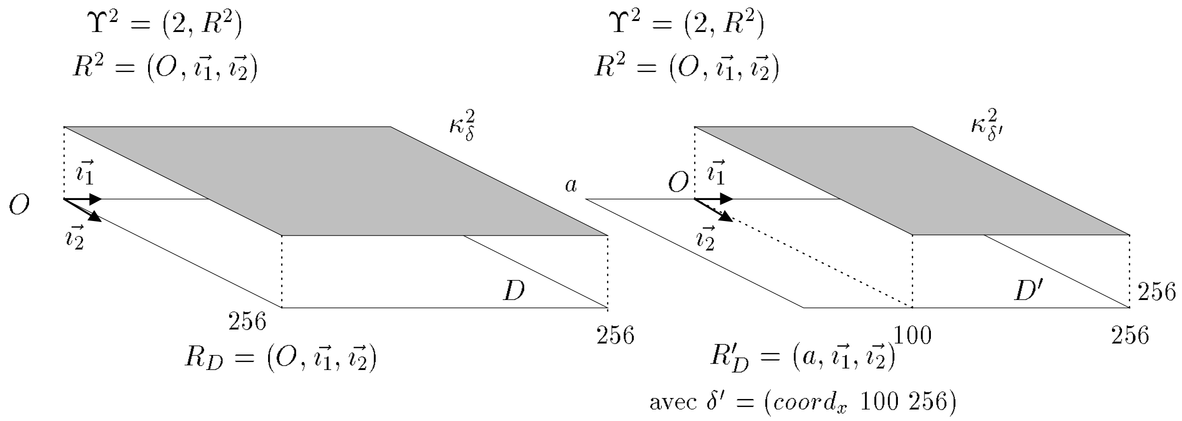


FIG. 1.12 - *Expression du cache par rapport au référentiel*

Ensemble des caches : Nous désignerons enfin par K_n l'ensemble des caches définis sur la trame τ, c'est à dire l'ensemble des κ_δ générés par les fonctions caches de F_{Υ^n}.

$$K_n = \left\{ \kappa_\delta \Big/ \delta \in \nabla_{\Upsilon^n} \right\} \tag{1.12}$$

1.2.3 L'Index ι

Définitions : La composante index ι d'une donnée D permet de regrouper les zones d'intérêt représentant un même concept par étiquetage des composantes de son cache κ associé. Elle peut être représentée par une table d'index dont chaque élément fait référence à une composante constituant le cache de la donnée dans un balayage déterminé (ligne, colonne, plan, séquence). Les valeurs de cette table d'index déterminent le mode de considération de ces composantes connexes. En effet, ces dernières, à même description δ, peuvent être considérées dans leur globalité, indépendamment les unes des autres ou répondant à un même critère $\mathcal{C}$. Pour traduire ce dernier point, nous associons à ι ce critère et pouvons ainsi définir trois ensembles d'index :

$I_{\mathcal{G}l}$: l'ensemble des index de globalité, c'est à dire des ι dont la caractéristique $\mathcal{C}$ est $\mathcal{C} = \mathcal{G}l$. Toutes les composantes sont considérées comme formant une seule entité sur laquelle seront effectuées d'autres transformations ou traitements (fig.1.13).

$$I_{\mathcal{G}l} = \left\{ \iota = (\mathcal{C}\ [\ldots]) \Big/ \mathcal{C} = \mathcal{G}l \right\} \tag{1.13}$$

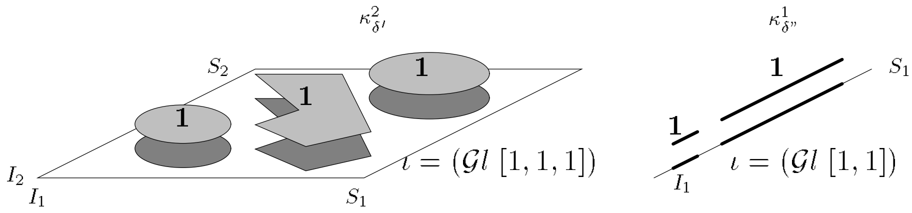


FIG. 1.13 - *Index de globalité*

$I_{\mathcal{I}n}$: l'ensemble des index d'indépendance, c'est à dire des ι dont la caractéristique $\mathcal{C}$ est $\mathcal{C} = \mathcal{I}n$. Toutes les composantes sont considérées comme indépendantes les unes des autres ; les futurs traitements ou transformations seront effectuées sur chacune d'entre elles (fig.1.14).

$$I_{\mathcal{I}n} = \left\{ \iota = (\mathcal{C}\ [\ldots]) \Big/ \mathcal{C} = \mathcal{I}n \right\} \tag{1.14}$$

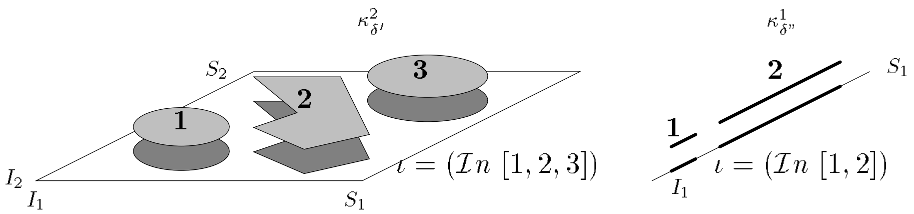


FIG. 1.14 - *Index d'indépendance*

I_χ : l'ensemble des index dépendant d'un critère spécifique $\chi = \{< critère >\}$. Comme dans $I_{\mathcal{I}n}$ les futurs traitements ou transformations seront effectuées sur chacun des groupes de composantes répondant au critère de la transformation (fig.1.15).

$$I_\chi = \left\{ \ \iota = (\mathcal{C}\ [\ldots]) \Big/ \ \mathcal{C} \in \chi \ \ \right\} \tag{1.15}$$

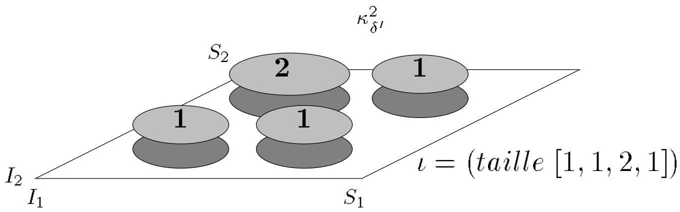


FIG. 1.15 - *Index spécifique $\mathcal{C}$="taille"*

Ainsi, l'index ι d'une donnée D possédant un cache κ est modélisée sous la forme du doublet $(\mathcal{C}\ [< table >])$.

Index de la donnée complémentaire : En complément à cette table d'index ι, nous construisons la table $\overline{\iota}$ contenant uniquement l'index nul, que nous noterons 0, permettant de désigner de façon globale le complémentaire $\overline{\kappa_\delta}$ de κ_δ.

$$\overline{\iota} = (\mathcal{G}l\ [0]) \tag{1.16}$$

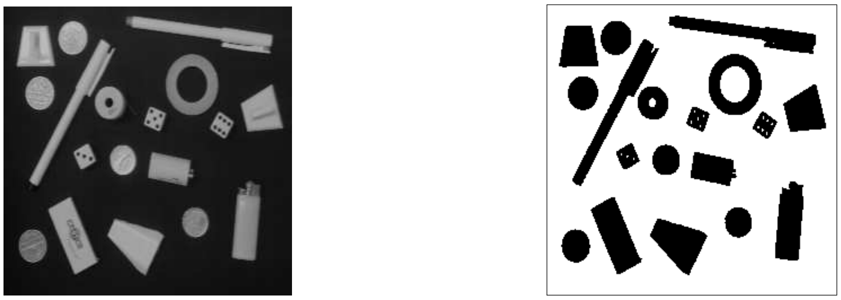
Ensemble des index : Enfin, nous désignerons par I l'ensemble de tous les index. I est défini de la façon suivante :

$$I = I_{\mathcal{G}l} \cup I_{\mathcal{I}n} \cup I_\chi \tag{1.17}$$

1.2.4 Le Signal σ

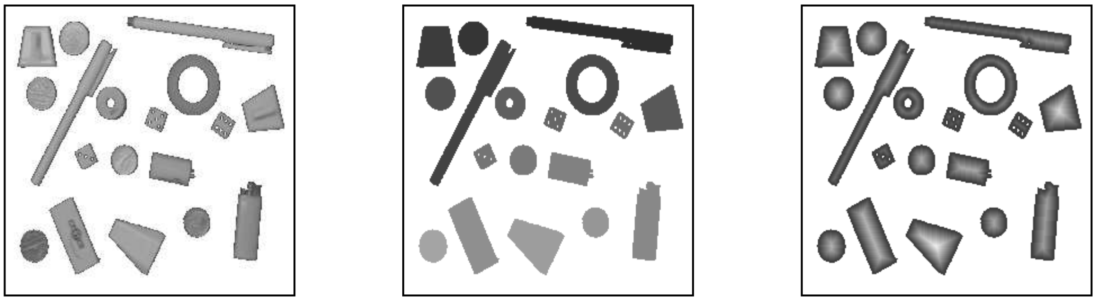
Définition : Les composantes de la donnée D vues au travers d'un cache κ indexé par une table ι, possèdent une caractéristique de représentation correspondant aux valeurs numériques associées aux intensités de leurs points.
Prenons une donnée image $\mathcal{D}$ sur laquelle nous définissons un cache κ dont la description δ associée correspond aux objets de l'image (fig.1.16). Les points de cette donnée, vus au travers de son cache, renferment des informations d'ordre sémantique que l'on doit représenter (fig.1.17). Nous appellerons ainsi signal -noté σ- la nature de l'intensité des points du cache reportés sur la donnée. Ainsi défini, ce signal devient un des éléments de construction des prédicats manipulés lors de la description des caches et possède donc un domaine de validité Δ_σ comme défini plus haut.

Signal de la donnée complémentaire : Comme précédemment, afin de compléter cette analyse, il faut considérer le non-cache $\overline{\kappa}$ de la donnée pour lui attribuer un signal $\overline{\sigma}$.



Donnée $\mathcal{D}$ Cache κ_δ^2 sur $\mathcal{D}$

FIG. 1.16 - *Donnée et cache associé*



Niveaux de gris Étiquettes Distances aux bords

FIG. 1.17 - *Différents signaux*

Ensemble des signaux : Nous désignerons par Sgn^n l'ensemble des signaux des données pouvant être définies sur un référentiel Υ^n.

1.2.5 Le Codage ω

Définition : Le signal caractérisant la sémantique de l'information renfermée dans l'intensité du point de la donnée, il nous reste maintenant à considérer la valeur numérique de celle-ci. Considérant une donnée D sur laquelle est décrit un cache κ, nous appellerons $\mathcal{C}_\sigma$ la fonction de codage appliquée à tous les points π de κ sur Δ_σ, qui déterminera les valeurs de représentation de ce signal (fig.1.18).

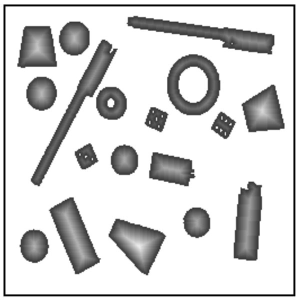
$$\mathcal{C}_\sigma \left| \begin{array}{l} \kappa \longrightarrow \Delta_\sigma \\ \pi \longmapsto v \end{array} \right. \tag{1.18}$$

Pour une donnée D déterminée possédant le signal σ sur un cache κ, nous définirons le codage de σ sur D, que nous noterons ω, comme étant le sous-ensemble de Δ_σ correspondant aux valeurs numériques que *peuvent* prendre les intensités des points π de ce cache. Ainsi nous aurons les inclusions suivantes :

$$\{v = \mathcal{C}_\sigma(\pi), \forall \pi \in \kappa\} \subseteq \omega \subseteq \Delta_\sigma \tag{1.19}$$

Codage de la donnée complémentaire : Par analogie, nous définirons $\overline{\omega}$ comme étant le codage des points appartenant à $\overline{\kappa}$ et dont le signal est $\overline{\sigma}$. Ainsi, quel que soit le couple $(\sigma,\overline{\sigma})$ défini sur D nous devrons avoir :

$$\omega \cup \overline{\omega} = \Delta_\sigma \tag{1.20}$$



Codage de $\sigma = dist$
$\omega = [1\ 128[$

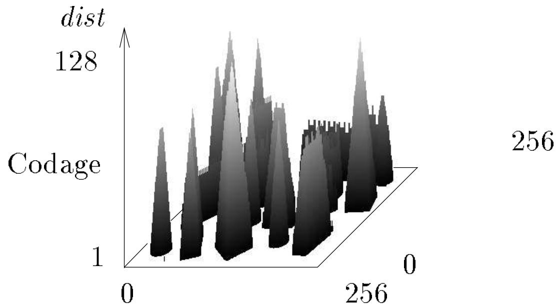


FIG. 1.18 - *Codage du signal distance*

1.3 Donnée de traitement d'images

1.3.1 Représentation symbolique

Une donnée symbolique de traitement d'images est définie par une structure (fig.1.19) comportant toutes les informations développées précédemment : la trame τ, la description du cache d'interprétation δ, l'index ι, le signal σ et l'ensemble des valeurs de codage de ce signal ω. De plus, à tous les niveaux de l'étude des données de traitement d'images, nous pouvons considérer qu'une donnée D est liée à une donnée complémentaire que nous noterons $\overline{D}$ de même trame τ et composée des éléments $\overline{\delta}$, $\overline{\iota}$, $\overline{\sigma}$ et $\overline{\omega}$.

$$D = \left\{ \begin{array}{l} \tau = ([0,256[\ [0,256[) \\ \delta = t \\ \iota = (\mathcal{G}l\ [1]) \\ \sigma = nivg \\ \omega = [0,256[\end{array} \right\} \quad \overline{D} = \left\{ \begin{array}{l} \overline{\delta} = () \\ \overline{\iota} = \cdot \\ \overline{\sigma} = \cdot \\ \overline{\omega} = \emptyset \end{array} \right\}$$

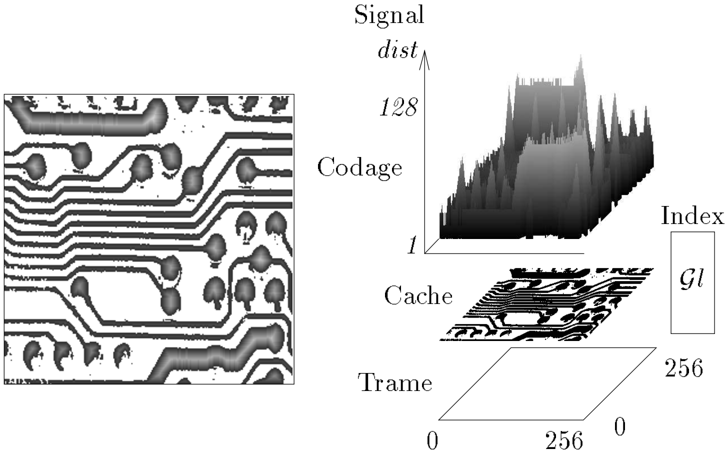
FIG. 1.19 - *Structure d'une donnée image*

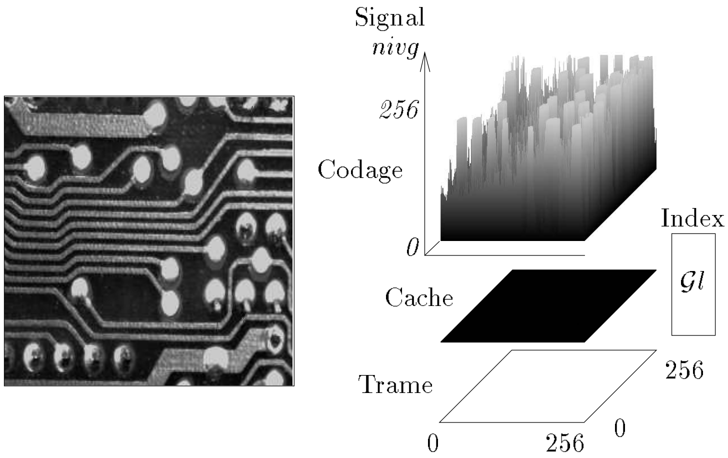
Si l'on considère maintenant qu'une session de traitement d'image débute par une transformation d'information, il est nécessaire de définir la structure de la donnée λ que nous appellerons donnée vide et que nous pouvons représenter par :

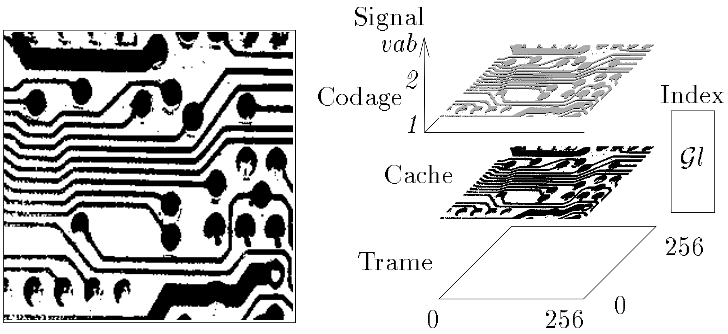
$$\lambda = \left\{ \begin{array}{l} \tau = \cdot \\ \delta = \cdot \\ \iota = \cdot \\ \sigma = \cdot \\ \omega = \cdot \end{array} \right\} \quad \text{avec} \quad \overline{\lambda} = \left\{ \begin{array}{l} \overline{\delta} = \cdot \\ \overline{\iota} = \cdot \\ \overline{\sigma} = \cdot \\ \overline{\omega} = \cdot \end{array} \right\} \tag{1.21}$$

Remarque : Afin de désigner une composante χ d'une donnée D et pour éviter des confusions dans les écritures, nous utiliserons la notation $D.\chi$. Pour étendre cette notation, nous noterons $\overline{D.\chi}$ (ou plus simplement $D.\overline{\chi}$) la composante $\overline{\chi}$ du complément de la donnée D.

1.3.2 Exemples de représentation







Chapitre 2

LEO : Langage d'Expression des Objectifs

Dans le cadre d'un processus d'analyse et d'interprétation d'image, le système formé des modules ExTI [Ing92] et SATI [Cap95] permet à un utilisateur, expert dans un domaine particulier de s'affranchir d'une connaissance du traitement d'images (fig.2.1).

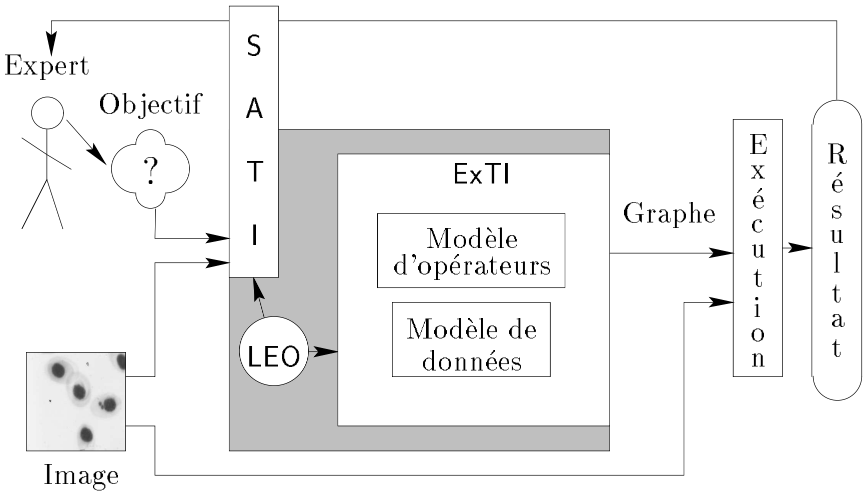


FIG. 2.1 - *Système d'analyse et d'interprétation d'image*

ExTI est un module de planification automatique de chaînes d'opérateurs intégrant la connaissance du traitement d'images sous la forme de modèles d'opérateurs et de données particuliers et de règles de simplification d'expression. Ces expressions sont les descriptions des données qui dans le cas du traitement d'images sont uniquement manipulées par les opérateurs. La donnée finale du plan généré correspond alors à la représentation mentale de l'objectif décrit par l'utilisateur à l'aide du module d'expression d'objectifs SATI. Ce dernier peut à son tour proposer de consulter le plan élaboré par ExTI et permettre d'intervenir sur les expressions des données.

La communication entre ces deux modules est assurée par le Langage d'Expression d'Objectif : LEO. Ce langage, étudié dans cette section, permet de représenter les données de traitement d'images en termes d'indices visuels. Ces données sont décrites selon un modèle précédemment défini [Dej96e] par le quintuplet (Trame, Cache, Index, Signal, Codage). En outre, ce langage fait référence à des notions telles que concepts, mesures, etc, qui sont elles-mêmes exprimées en termes d'indices visuels. Pour construire un tel langage, nous avons utilisé une extension des notations et du principe du calcul des prédicats annotés de R.S. Michalski [Mic83, Mic84].

L'objectif de ce chapitre n'est pas de recréer le langage utilisé -on se référera pour cela à [Ing90, Ing92, Ing96]- mais de l'étudier en détail et de l'enrichir afin d'en déterminer les caractéristiques qui devront permettre d'écrire sa grammaire et d'implémenter un système de traitement de l'objectif, pour simplifier celui-ci, pour préparer la phase de planification, etc [Dej96f].

2.1 Un double enjeu pour LEO

2.1.1 Notion de Vocabulaire Exécutable

Comme il est rapporté dans [Ing92], nous pouvons associer une description textuelle de transformation d'informations à un opérateur de traitement d'images, et de ce fait par propagation, associer une description à une séquence d'opérateurs (fig.2.2). Cette possibilité permet de rendre compte de la dualité du traitement (données/concepts) présentée au paragraphe 1.3 dans la première partie. Mais à toutes les descriptions correctement développées correspond un terme spécifique résumant

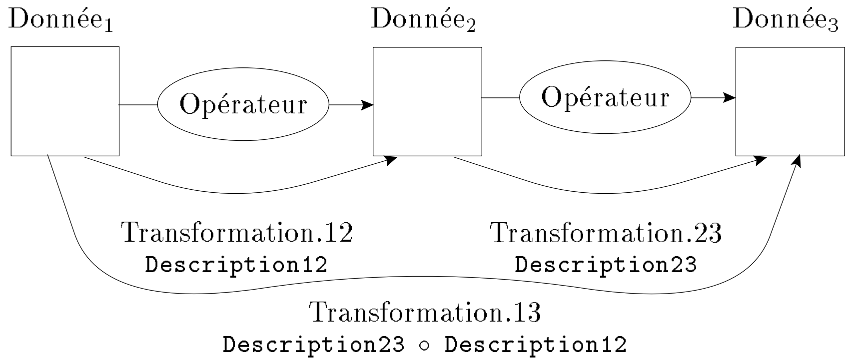


Fig. 2.2 - *Propagation*

ou identifiant l'entité décrite. Ceci permet d'avoir, au plus-haut niveau de la hiérarchie de ces expressions un *vocabulaire exécutable.*

De telles descriptions textuelles des transformations d'information, permettant de développer un langage bas-niveau de description de concept, utilisent une extension du calcul des prédicats annotés [Mic83] et toutes les règles de généralisation, de spécialisation et de simplification d'expressions.
Nous rappelons brièvement ici les principales notations que nous allons utiliser :

˜	négation
&	conjonction
$\vee$	disjonction
$\models$	reformulation
$\ltimes$	spécialisation
$\triangleright$	généralisation
::>	implication liant la description au nom de concept
$DOM(p)$	domaine de l'attribut p

Si nous partons d'un ensemble conséquent de descriptions correctement développées, et si nous faisons une rapide étude de cet ensemble d'un point de vue purement et strictement orienté traitement de concepts, nous nous apercevons que certaines structures de descriptions, manipulées dans les différents domaines traités, sont réutilisées pour exprimer des constructions ou manipulations d'entités de l'image et correspondent le plus souvent à des concepts génériques.
Ainsi nous pouvons exprimer et manipuler des concepts et descriptions de caches comme :

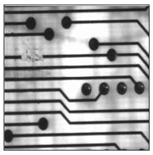
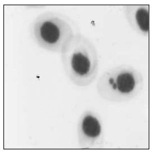
$(fenetre\ 0\ 0\ 0\ 255)$	::>	$\texttt{BordGauche}.\kappa$
$(fenetre\ 255\ 0\ 255\ 255)$	::>	$\texttt{BordDroit}.\kappa$
$Max(Dist(\texttt{Objets}.\kappa))$	::>	$\texttt{EpaisseurObjets}.\kappa$
$\texttt{Objets}.\kappa\ \&\ [MaxDist = Max(Dist(\texttt{Objets}.\kappa))]$	::>	$\texttt{ObjetsDePlusGrandeEpaisseur}.\kappa$
$Inter(Inter(\texttt{Objets}.\kappa, \texttt{BordGauche}.\kappa), \texttt{BordDroit}.\kappa)$	::>	$\texttt{ObjetsTraversantHorizontalement}.\kappa$

et maintenir un lien entre les descriptions d'entités issues du traitement d'images -propagation par les opérateurs-, et les noms des entités manipulées.

Ce principe est la base du système ExTI pour la planification de chaînes d'opérateurs résolvant un problème d'analyse.

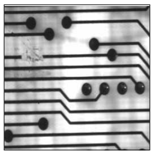
2.1.2 Quatre types de descriptions

En approfondissant cette idée de concepts génériques, nous pouvons distinguer quatre types de description d'entités utilisées dans l'analyse des images[1] :

circuit  objets  dés  cellules 

1. Les descriptions que nous appellerons *descriptions spécifiques au domaine* qui font référence à des mesures ou des caractéristiques particulières des entités présentes dans les données :
   Exemples : surface, position de certains objets, . . . (fig.2.3)
2. Les descriptions que nous appellerons *descriptions liées aux données* qui font référence à des caractéristiques propres aux données, indépendamment des entités qui s'y trouvent :
   Exemples : taille de l'image, codage des pixels, . . .
   –Image 'circuit' :
   msk1 *: (((coordonnee < (fenetre 130 131 250 160))))*
   –Image 'des' :
   msk1 *: (((coordonnee < (cercle 128 128 95))))*
   Dans le nouveau modèle de donnée :

1. Nous prendrons, pour illustrer, des exemples tirés de la thèse de J.-M. Inglebert (Annexe 1)

circuit 

objets (((nivg < 110)))

cc1 (((surface >= 200)(surface < 400)(objets)))

pix4 (((nivg >= 50)(cc2)))

objects 

objets (((nivg >= 110)))

cc4 (((perimetre >= 350)(objets)))

cc9 (((surface >= 50)(surface < 32000)(n_objets)))

FIG. 2.3 - *Descriptions spécifiques au domaine*

((0 256) (0 256))	::>	`Image.`τ
(*fenetre* 0 0 0 255)	::>	`BordGauche.`κ
(*fenetre* 255 0 255 255)	::>	`BordDroit.`κ
(*nfenetre* 1 1 254 254)	::>	`Bords.`κ

3. Les descriptions que nous appellerons *descriptions génériques* qui font référence à des manipulations ou combinaisons particulières d'entités déjà décrites et que nous nommons `Objets`:
   Exemples : objets sans trous, objets pleins, objets traversant, ...
   –Image 'cellules'
   dilatés_pleins : *(couronnes_non_centres (dans (interieurs_non_centres) (dilates)) centres)*
   –Image 'des'
   dis1 : *(pix2 (dans (n_pix2) (pix_2)))*
   –Image 'objets'
   cc1 : *(((euler >= 1) (objets)))*
   cc14 : *(((euler >= 1) (pix2)))*
   cc11 : *(((max >= (sig1)) (((ordre)(objets)))))*
   avec ***sig1 :*** *(valeur_max (((max)(((ordre)(objets))))))*
4. Les descriptions que nous appellerons *descriptions complémentaires* qui expriment, comme leur nom l'indique, les compléments de certaines descriptions.
   Exemple :
   –Image 'cellules'
   dilatés *(couronnes_non_centres centres)*
   couronnes_non_centres *(((ordre < 3) (not_centres)))*
   interieurs_non_centres *(((ordre >= 3) (not_centres)))*
   –Image 'circuit'
   objets *(((nivg < 110)))*
   n_objets *(((nivg >= 110)))*
   –Image 'des'
   pix2 *(((coordonnee < (cercle 128 128 95)) (nivg >= 110)))*
   n_pix2 *(((coordonnee < (cercle 128 128 95)) (nivg < 110)) ((coordonnee >= (cercle 128 128 95))))*

2.1.3 Manipulation de ce vocabulaire

Hormis les descriptions spécifiques aux domaines dont la construction relève d'une interaction avec l'utilisateur afin de définir les éléments à extraire (SATI [Cap92]), les trois autres types de descriptions peuvent être traitées de façon plus automatique -dès lors que l'on a isolé les éléments caractéristiques qui font le rattachement de ces descriptions à leur type- en développant un langage de plus haut-niveau basé sur la notion de noms des concepts et de la manipulation de ces noms.

Base de descriptions liées aux données :
Les descriptions liées aux données constituent un ensemble d'expressions paramétrées par les caractéristiques des données. Comme nous le verrons plus loin avec la notion de domaine de validité (voir §2.2.5.2 et annexe C), certains de ces paramètres liés au contexte et aux conditions opératoires influent aussi sur les caractéristiques des expressions complémentaires et entrent en jeu dans les règles de simplification des descriptions produites.

bord_droit	((fenetre (1- #xImage) 0 (1- #xImage) (1- #yImage)))
bord_gauche	((fenetre 0 0 0 (1- #yImage)))
bord	((nfenetre 1 1 (- #xImage 2) (- #yImage 2)))

Descriptions complémentaires :
Parce qu'elles sont le plus souvent difficiles à exprimer, les descriptions complémentaires de descriptions étiquetées n'auront plus à être développées dans les expressions de concepts. À leur place, nous utiliserons un nom de concept précédé du symbole ~. De ce fait, les expressions rencontrées ci-dessus s'exprimeront comme suit :

couronnes_non_centres	(((ordre < 3) (~centres)))
interieurs_non_centres	(((ordre >= 3) (~centres)))

sans plus avoir à exprimer le concept 'non_centres'.

Le principal avantage de cette notation est la simplification de la génération des descriptions par un système tel que SATI ainsi que la possibilité de décrire de façon plus complète des descriptions génériques mettant en jeu des expressions de complémentaires.

Base de descriptions génériques :
Les descriptions génériques peuvent être représentées sous la forme de fonctions manipulant des descriptions. Le rôle de ces fonctions est alors de générer les expressions de certains champs de la donnée-objectif.

Exemples :

–La fonction *à_n_trous* appliquée au concept générique *objets* et paramétrée par *#n* sera définie par : *à_n_trous(objets,#n)=(((euler (- #n 1) 0) (objets)))*

–La fonction *sans_trous* appliquée au concept générique *objets* sera définie par :
sans_trous(objets)=à_n_trous(objets,0)

–La fonction *touche_bords_droit* appliquée au concept générique *objets* sera définie par :
touche_bords_droit(objets)=((intersecte (objets) ((fenetre (1- #xImage) 0 (1- #xImage) (1- #yImage)))))

–La fonction *traversant_horizontalement* appliquée au concept générique *objets* sera définie par :
traversant_horizontalement(objets)=touche_bords_droit(touche_bords_gauche(objets))

et retournera l'expression :

((intersecte ((intersecte (objets) ((fenetre 0 0 0 (1- #yImage)))))
((fenetre (1- #xImage) 0 (1- #xImage) (1- #yImage)))))

–La fonction *plein* appliquée au concept générique *objets* sera définie par :
plein(objets)=(objets (dans ("~"objets) (objets)))

permettant ainsi de définir le concept 'dilatés_pleins' précédent de façon plus simple :

dilatés_pleins : (dilates (dans (~dilates) (dilates)))

2.1.4 Description et contexte de l'application

Ces descriptions peuvent être alors utilisées pour exprimer des concepts de plus en plus spécialisés permettant de développer un langage spécifique pour un domaine d'application. Chaque terme de ce langage étant lié à une description d'objectif et chaque description étant liée à une chaîne de traitement après planification, nous obtenons donc le *vocabulaire exécutable* spécialisé dans une application et pour lequel chaque concept est associé à une séquence d'opérateurs de traitement d'images [Dej96d]. Ainsi, l'utilisateur, via une interface telle que SATI[Cap94], pourra manipuler de la même façon -ou de façon unifiée- des descriptions d'entités ou des instances d'objets validées par le traitement d'images au cours du processus d'exécution d'une chaîne.

Dans l'exemple suivant, nous proposons quelques descriptions simples de concepts de circuits imprimés (fig.2.4) :

$$\begin{array}{rcl} [nivg = 78..255] & ::> & \texttt{Objets} \\ \texttt{ObjetsPleins}\ \&\ [MaxDist = 5..127] & ::> & \texttt{P+FCP} \\ NonInter(\texttt{P+FCP}, \texttt{BordsImage}) & ::> & \texttt{PD} \\ \texttt{ObjetsPleins}\ \&\ [MaxDist = 2..4] & ::> & \texttt{FDP} \\ Inter(Inter(\texttt{FDP}, \texttt{BordGauche}), \texttt{BordDroit}) & ::> & \texttt{TH(FDP)} \end{array}$$

avec

$$\texttt{TH(FDP)} \models Inter(\ Inter(\ ([nivg = 78..255] \ \vee \ Inter(\ [nivg = 0..77], [nivg = 78..255])\ )\ \& \ [MaxDist = 2..4], (fenetre\ 0\ 0\ 0\ 255)), (fenetre\ 255\ 0\ 255\ 255))$$

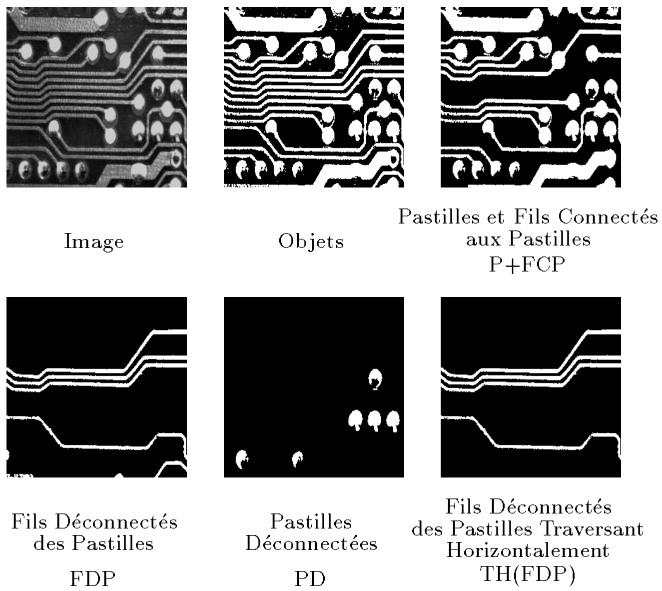


FIG. 2.4 - *Concepts d'un domaine d'application*

Ainsi, à partir de ces descriptions génériques, des vocabulaires spécifiques distincts peuvent être développés. Interviennent là les notions de contexte telles qu'elles sont présentes dans l'histoire du PETIT PRINCE. En effet, si nous prenons le concept générique *objet traversant_horizontalement*, il permet de décrire une rivière ou certains fils (fig.2.5).

De ce fait, une même description peut représenter plusieurs choses dans des domaines totalement différents et il est de fait possible d'utiliser des expressions d'un domaine pour décrire des entités d'un autre domaine.
Il est plus intéressant de constater que, pour décrire des objectifs de traitement d'images, la description d'une entité particulière de l'image est liée à l'ensemble des entités de cette même image. En effet, comme l'illustrent les figures suivantes, des descriptions -*carré* dans la figure 2.6 et *dé* dans la figure 2.7- ne peuvent être valides que dans des contextes particuliers pour lesquels l'élément discriminant de la description est suffisamment spécifique -au sens de Michalski- pour ces données de l'application. Nous retrouvons là le problème évoqué lors de la définition des fonctions cache dans le modèle de la donnée.

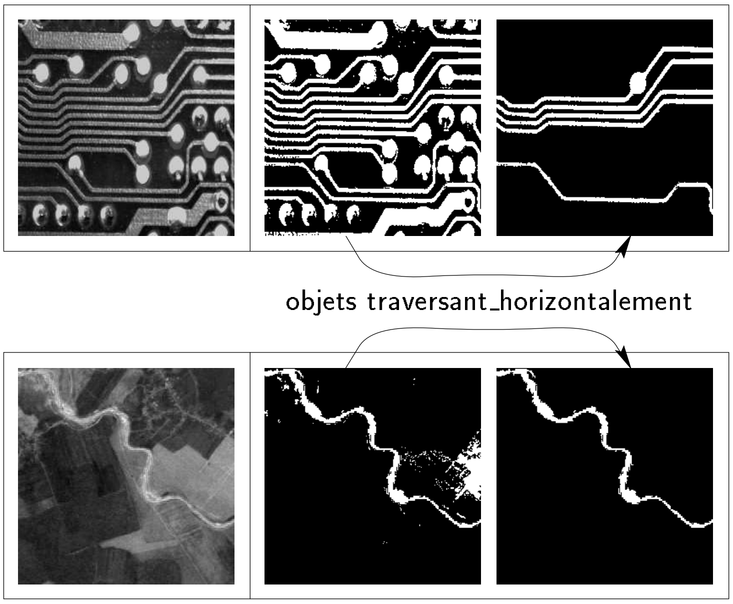


FIG. 2.5 - *Transformation d'une description générique*

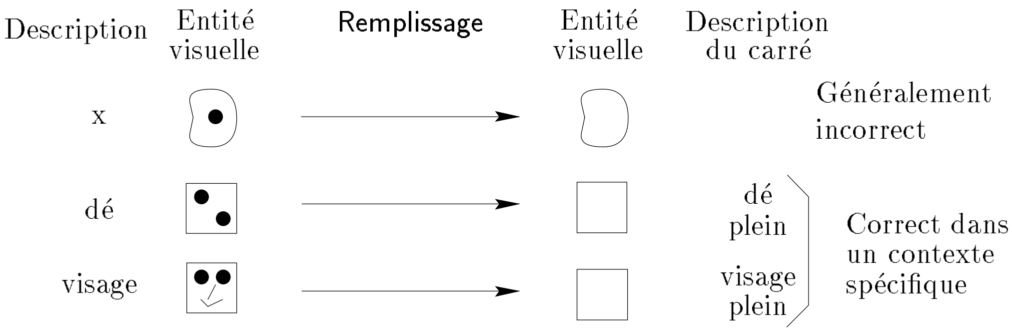


FIG. 2.6 - *Discrimination par remplissage*

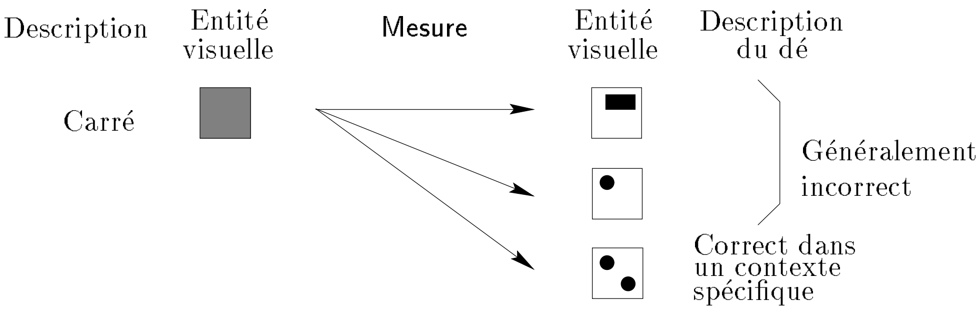


FIG. 2.7 - *Discrimination par [euler=-5..0]*

2.2 Présentation du langage de base

2.2.1 Expression des connaissances

Le langage que nous présentons se doit d'intégrer plusieurs types de connaissances à la fois sur les données de traitement d'images dans leur généralité, sur le contexte de leur utilisation, et sur le traitement d'images proprement dit. Nous pouvons faire alors un lien avec les connaissances évoquées en introduction (page 47) sur les connaissances des éléments linguistiques dans le langage naturel (*phonétiques* et *phonologiques*, *morphologiques*, *syntaxiques*, *sémantiques* et *pragmatiques*).

Une classe de connaissances que nous pourrions rapprocher des connaissances phonologiques -bien qu'ici il ne s'agisse pas de sons-, considère l'ensemble des descripteurs dans leur relation avec les indices visuels qu'ils représentent. Ces connaissances sont représentées par le rattachement des descripteurs aux opérateurs de traitement d'images qui mettent en œuvre ces indices visuels, mais aussi par l'attribution à chacun de ces descripteurs d'un ensemble de valeurs appelé *domaine de validité*.

Une classe de connaissance que nous pourrions rapprocher des connaissances morphologiques, considère l'ensemble des descripteurs utilisés et les relations entre eux -éléments modificatifs ajoutant des informations sur les descripteurs (maximum, opposé, ...)- et l'utilisation de ces descripteurs pour former des concepts -structuration / déstructuration, distribution- [Ing96].

Une classe de connaissances sémantiques, plus importante en ce qui concerne l'expression d'objectifs au regard du domaine d'application, considère les connaissances sur la façon dont a été fournie la description de la donnée. En effet, comme nous l'avons vu dans la section précédente, les descriptions sont étiquetées et le nom qu'elles possèdent dans un contexte d'application donné témoigne de la présence d'entités particulières dans la donnée. Ainsi, ces noms sont directement utilisées dans d'autres descriptions pour en simplifier l'écriture dans une expression de haut-niveau.
Cette expression de haut-niveau, manipulant directement le vocabulaire de l'application et son utilisation, est exploitée par un nouveau moteur de planification que nous avons installé dans le système ExTI. L'étude de ce moteur, travaillant par ré-utilisation de plan, fait l'objet de la section 1.2.2 (3^{e} partie).

Une classe de connaissances syntaxiques mettant en œuvre des règles de réécriture et de simplification (voir les règles transformationnelles page 99 et chap.2, 3^{e} partie). Celles-ci permettent de passer, pour les premières, de l'expression de haut-niveau manipulant des étiquettes d'objectifs à une expression quasi complètement développée -à l'exception des références aux complémentaires étiquetés-, et pour les secondes, de cette expression à son expression équivalente dans la forme canonique définie plus loin.
D'autres règles de manipulation syntaxiques sont représentées, implémentant des connaissances liées au traitement d'images lui-même. Ces règles gèrent par exemple les redondances de traitements (exemple : idempotence de l'ouverture) ou les opérations logiques au cours du traitement.

Enfin une classe de connaissances pragmatiques est formée par les connaissances sur les conditions opératoires. Elles seront représentées par un ensemble de variables définissant le contexte de l'utilisation des données manipulées. Elles caractériseront ainsi la taille, le type et la nature, etc, des données initiales.

2.2.2 Quels choix pour la représentation du langage

Comme nous l'avons dit plusieurs fois, les expressions décrivant les objets sont représentées sous des formes étendues du calcul des prédicats annotés. Une donnée de traitement d'images sera alors décrite par neuf expressions : cinq décrivant chacune un champ, et quatre décrivant leur complémentaire. L'intérêt de ce choix est de relever le caractère descriptif des expressions tant sur le plan des transformations effectuées par les opérateurs sur les données de traitement d'images que sur le plan des manipulations des concepts de l'application. Ainsi, cette notation se rapproche plus des graphes conceptuels [Sow84],[Bok93] dont elle possède les éléments relationnels entre les concepts manipulés, que des bases de faits classiquement utilisées en résolution de problèmes, dont elle possède les caractéristiques informatives.

Ainsi, l'objectif `rel5` tiré de [Ing92] et adapté aux nouveaux modèles de données et d'expression s'exprime dans notre base d'objectifs de la façon suivante :

```
-- objets
   (((nivg 100 256)))
   Les objets sont constitués des pixels de l'image à niveaux de gris supérieurs ou égaux
   à 100.
-- cc10
   (((mxx 0 1)(srf 6 11)((vois4)(~objets))))
   Les cc10 sont des ~objets dont la surface est comprise entre 6 et 10 inclus et dont le
   moment d'ordre 2 en x est inférieur à 1.
-- rel1
   (objets (dans ((vois4)( objets))(objets)))
   Les rel1 sont des objets ou des ~objets dans les objets.
-- rel4
   ((inter ((vois4)(rel1))(cc10)))
   Les rel4 sont des rel1 qui touchent des cc10 selon les 4-voisins.
-- rel5
   ((inter ((vois4)(objets))(rel4)))
   Les rel5 sont des rel4 qui touchent des objets selon les 4-voisins.
```

et peut être représentée sous la forme du graphe conceptuel de la figure 2.8 faisant apparaître les relations entre concepts.

Sous sa forme complètement développée et simplifiée, la même description `rel5` a pour forme :

```
((inter ((vois4)(((nivg 100 256)))) ((inter ((vois4)(((nivg 100 256)) (dans ((vois4)(((nivg 0 100))))
(((nivg 100 256)))))) (((mxx 0 1) (srf 6 11) ((vois4)(((nivg 0 100)))))))))
```

Remarque : Ces formes d'expressions étant assez littérales, nous avons effectué des essais de génération d'expressions à partir de formes verbales de descriptions d'objectifs à l'aide de la version de base du système expert Andromac [Voy87]. Mais si la faisabilité peut être montrée sur des exemples simples, de nombreux problèmes liés aux simplifications d'expressions -on peut se référer à [Den95] au sujet des problèmes posés par les représentations mentales des images décrites- nous ont fait arrêter temporairement nos recherches dans cette direction.

Parce que les descriptions, enrichies depuis les versions de [Ing92], renferment toutes les connaissances évoquées plus haut, et en particulier les connaissances pragmatiques, deux niveaux d'expression seront utilisées et donc plusieurs sortes de grammaires formelles seront mises en œuvres :

- Au plus bas niveau, nous trouverons une grammaire formelle du type 2 de Chomsky pour laquelle nous essaierons de minimiser le nombre de terminaux permettant le déterminisme

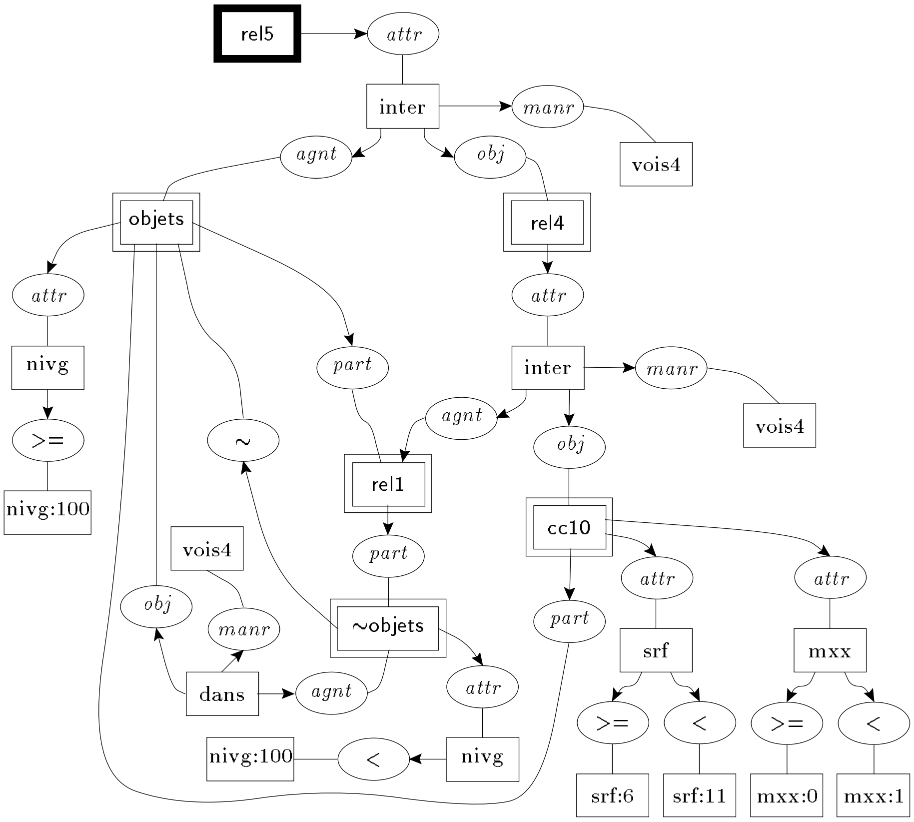


FIG. 2.8 - *Graphe conceptuel de l'objectif* `rel5`

entre deux règles ou schémas de règles de production -terme k des grammaires dites LL(k).

- À un niveau supérieur, nous déterminerons les règles d'une grammaire transformationnelle dont les caractéristiques -catégories syntaxiques et ensembles de terminaux- sont identiques à celles de bas-niveau mais qui permettent d'utiliser les champs des données décrites et étiquetées par un terme du vocabulaire de l'application pour exprimer d'autres champs d'une nouvelle description de donnée.
  Ces règles sont réparties en trois groupes -transformationnelles et méta-règles [Gaz79] et de simplification- et gèrent les substitutions puis la cohérence syntaxique des descriptions ainsi que les simplifications permettant de fournir des expressions d'objectif équivalentes sous une forme canonique afin d'en assurer l'interprétation correcte et la résolution.
- Enfin, un ensemble d'attributs -affixes- peuvent être apposés aux terminaux des grammaires des catégories syntaxiques dans le but de récupérer des informations particulières à partir de la description complètement développée. Des règles d'affixes gèrent ces attributs sémantiques.

2.2.3 LEO=LDC+LDD

Le langage LEO va pouvoir être décrit par une grammaire formelle, non contextuelle dont l'étude détaillée d'une précédente version -automate, déterminisation, minimisation et construction des analyseurs- est rapportée dans [Dej96a].

Comme nous l'avons dit précédemment, ce langage utilise des notions déjà étudiées de description et de manipulation de concepts. Les entités comme *sélecteurs*, *propriétés*, *relations*, etc..., et les fonctions qui les manipulent (*spécialisation*, *généralisation*, etc...) étant très importantes pour notre application -rappelons la dualité d'action de l'opérateur sur les concepts d'une application et sur les données numériques-, le langage LEO va se décomposer en deux ensembles de catégories syntaxiques caractérisant :

1. le Langage de Description de Concepts (LDC) qui va consister en un ensemble de huit catégories syntaxiques, qui seront dites de base, totalement imbriquées mais dont l'étude et la caractérisation peuvent être effectuées les unes indépendamment des autres (voir §2.3.1).
2. le Langage de Description des Données (LDD) qui va être formé des langages de description de chacun des cinq champs de la représentation de la donnée, en menant une analyse analogue à celle des catégories syntaxiques de base qu'elles utilisent.

Chacune de ces grammaires est au moins de type 2, c'est à dire non-contextuelle.

2.2.3.1 Notations, rappels et compléments

Nous ne rappelons dans cette section que les notations et les définitions essentielles auxquelles nous ajoutons quelques compléments permettant de gérer les catégories syntaxiques imbriquées comme si elle étaient indépendantes.

Notations

1. Expressions :
   $a^* = \lambda + a + aa + aaa + \ldots$ (lemme de l'étoile de S.C. Kleene)
   $a^+ = a + aa + aaa + \ldots$
2. Grammaires : $G = (N, T, \mathcal{P}, S \in N)$
   avec N ensemble de non-terminaux, T ensemble de terminaux, $\mathcal{P}$ ensemble de productions et S non-terminal axiome.

Rappels

1. Une grammaire est LL(k) [2] si k symboles d'entrée de prévision doivent être utilisés à chaque étape nécessitant la prise d'une décision d'action d'analyse descendante.
2. Une grammaire non contextuelle est auto-imbriquée s'il existe un non-terminal A et des séquences α, β de terminaux et de non-terminaux non vides tels que A se dérive en $\alpha A \beta$:
   $A \stackrel{*}{\Longrightarrow} \alpha A \beta$
3. Ensembles $PREMIER_k()$:
   Soit α la partie droite d'une production définissant le non-terminal A, on définit $PREMIER_k(\alpha)$ comme étant l'ensemble des k premiers terminaux des séquences de terminaux générées à partir de $A \rightarrow \alpha$.
4. Ensembles $SUIVANT_k()$:
   Soit A un non-terminal, on définit $SUIVANT_k(A)$ comme étant l'ensemble des k premiers terminaux des séquences de terminaux générées à partir de dérivations de la forme $S \stackrel{*}{\Longrightarrow} \alpha A \beta$.
   De fait, $PREMIER_k(\beta) \in SUIVANT_k(A)$

Compléments

1. Lorsque des noms de catégories syntaxiques $(Y, Z, \ldots)$ sont utilisés comme symboles terminaux pour l'expression d'une autre catégorie syntaxique distincte X, nous dirons que X est restreinte par $\{Y, Z, \ldots\}$ et nous noterons $X_{|_{\{Y,Z,\ldots\}}}$ la catégorie syntaxique restreinte.
2. Nous appellerons catégorie syntaxique X-restreinte, une catégorie syntaxique pour laquelle chaque nom de catégorie syntaxique, distinct de X, utilisé dans son expression est considéré comme un symbole terminal pour cette expression.

2. LL() : Left to right scanning , Leftmost derivation

3. Par abus de langage, nous dirons qu'une catégorie syntaxique X est déterministe si la grammaire de la restriction de X par un ensemble $\mathcal{R}$ est déterministe et si on peut vérifier la propriété suivante :
   S'il existe deux productions de $\mathcal{R}$ ayant pour partie gauche L_0 se dérivant en αL_x et βL_y avec α et β des terminaux non vides, alors α et β sont distincts.

2.2.3.2 Contrainte de l'implémentation

Afin de faciliter l'implémentation en LISP des modules de reconnaissance de la grammaire de LEO, toutes les règles de productions -ou plus exactement les *schémas de règles*- doivent être proposées sous une forme parenthésée, c'est à dire dans laquelle nous trouvons deux parenthèses ouvrante et fermante d'un même niveau de parenthésage. Qu'elles soient de Greibach[3] sous la forme (action-état) ou de Chomsky[4] pour les plus connues, les *formes normales* ne se prêtent pas à cette forme d'écriture. De ce fait, les parties droites des schémas de règles de productions présentées ici ont étés écrites à partir d'une forme, issues des classiques diagrammes de transitions et définies par

des productions vides :	$A \to \lambda$	avec $A \in N$
ou des productions du type :	$A \to aB$	avec $A \in N$, $B \in N \cup \{\lambda\}$ et $a \in T$
	$A \to (B)C$	avec $(A, B) \in N^2$ et $C \in N \cup \{\lambda\}$
	$A \to BC$	avec $(A, B, C) \in N^3$

en appliquant l'algorithme suivant à l'ensemble des règles.

	Algorithme de simplification d'un ensemble de règles de productions d'une grammaire non ambiguë, non récursive :
–	**si** il existe une seule production définissant L_x ayant λ pour partie droite : $L_x \to \lambda$, **alors** il faut remplacer toute production $L_y \to \alpha L_x$ avec $x \neq y$ par $L_y \to \alpha$.
–	**si** il existe une seule production définissant L_x ayant λ pour partie droite : $L_x \to \lambda$, **et si** L_x ne figure dans aucune des parties droites des autres productions, **alors** il faut enlever la production $L_x \to \lambda$ de l'ensemble.
–	**si** il existe deux productions de la forme $L_x \to (L_y$ et $L_z \to)\beta$ -(et) étant de même niveau de parenthésage- **et si** L_y se dérive en αL_z : $L_x \stackrel{*}{\Longrightarrow} (\alpha L_z$, **alors** il faut remplacer les productions $L_x \to (L_y$ et $L_z \to)\beta$ par $L_x \to (L_y)\beta$ et $L_z \to \lambda$ respectivement.
–	**si** il existe une seule production définissant L_x ayant L_y pour partie droite, **alors** il faut remplacer toute production de la forme $L_z \to \alpha L_x \beta$ par $L_z \to \alpha L_y \beta$ **et** enlever ensuite la production $L_x \to L_y$ de l'ensemble.
–	**Réexaminer** les règles précédentes jusqu'à ce qu'aucune d'entre elles ne soit applicable.

Une description de la grammaire complète de LEO comprenant les schémas de règles de production et les tables d'analyse prédictive avec récupération sur erreur est donnée en annexe B.

2.2.4 Les ensembles de terminaux

Dans sa thèse, J.M. Inglebert [Ing92] propose une décomposition de l'ensemble des symboles terminaux utilisés pour la description d'objectifs en six *classes de caractéristiques*. Chacune d'entre

3. Forme normale de Greibach :
   -Grammaire $G = (N, T, \mathcal{P}, S)$ sans vide (l'unique production vide autorisée est $S \to \lambda$)
   -Règles de production de la forme : $A \to a\alpha$ avec $A \in N$, $a \in T$ et $\alpha \in N^*$
4. Forme normale de Chomsky :
   -Grammaire $G = (N, T, \mathcal{P}, S)$ sans vide
   -Règles de production de la forme : $A \to BC$ ou $A \to a$ avec $(A, B, C) \in N^3$, $a \in T$.

elles (*pixel*, *voisinage*, *composante connexe*, *relation*, *image*, *arithmétique* et *signal*) regroupe des items syntaxiques possédant des caractéristiques communes en termes de traitement d'images. Cette partition de l'ensemble de descripteurs est à présent complétée par une autre décomposition -qui n'est pas une partition- faisant apparaître dans T_{LEO} des sous-ensembles de descripteurs relevant d'une ou plusieurs catégories syntaxiques. Les descripteurs utilisés sont donc regroupés en ensembles de terminaux pour les catégories étudiées :

$$T_{LEO} = \{(,), \upsilon, \varsigma, \breve{\varsigma}, \theta, \rho, \phi, \zeta, \hat{\zeta}, \eta, \mu, \nu, \mho\}$$

2.2.4.1 L'ensemble υ des valeurs numériques

Comme son nom l'indique, cet ensemble contient les valeurs numériques.
De fait, $\upsilon = \mathrm{IR}$.

2.2.4.2 L'ensemble ς des sélecteurs

L'ensemble ς contient l'ensemble des symboles du langage pouvant être utilisés comme éléments de caractéristiques de sélection au sens de Michalski. Ces symboles appartiennent de fait à plusieurs *classes de caractéristiques* étant donné le rôle de l'opération de sélection d'entité -spécialisation de concepts-, identique quelle que soit la donnée -sa dimension et son type.
Les descripteurs de signal rassemblés dans cet ensemble ς correspondent donc à tous les symboles caractérisant les notions sémantiques liées aux points des données. Cet ensemble correspond exactement à la *classe de caractéristique signal* qui est présentée comme un terme générique désignant un élément de la donnée traitée.
Exemples : `perimetre, nivg, dist, euler, ..., freeman, valeur-binaire`

2.2.4.3 L'ensemble $\breve{\varsigma}$ des descripteurs de sous-sélecteurs

Lorsque certaines opérations sont effectuées en modifiant le signal de la, ou des, données, le signal résultant peut être considéré comme composé de plusieurs éléments. Des exemples de sous-sélecteurs sont les descripteurs `valeur-absolue` et `signe` du sélecteur `différence` de deux données de même signal. Ce terme vient du fait que ces descripteurs ajouteront une information sur le signal de la donnée d'entrée afin d'en extraire une des parties relative à l'opération effectuée.
Cette notion est voisine de la précédente et l'utilisation de ces descripteurs dans la construction d'expressions quasi identique. La différence essentielle réside dans le fait que les éléments de cet ensemble peuvent être regroupés pour former un descripteur de signal de plus haut niveau sémantique ; par exemple la valeur absolue et le signe pour décrire la différence.
Exemples :
`valeur-absolue` et `signe` pour la différence de données,
`direction` et `nombre-pixels` pour le codage de Freeman,
`rouge`, `vert` et `bleu` pour la couleur, ...

2.2.4.4 L'ensemble θ des descripteurs d'opération arithmétique

Cet ensemble θ contient les symboles des opérations arithmétiques pouvant être effectuées sur les valeurs numériques (υ) et les données de dimension 0, comme sur les données de dimension n : différences d'images, ...
Exemples : `+, -, *, /, ...`

2.2.4.5 L'ensemble ρ des descripteurs de relation

L'ensemble ρ contient des descripteurs déterminant une sélection d'entités de l'image, décrites en référence à d'autres entités. Il correspond exactement à la *classe de caractéristique relation*.
Exemples : `inter / ninter , dans / hors, ...`

2.2.4.6 L'ensemble ϕ des descripteurs de figure

Cet ensemble particulier, ϕ, contient les symboles caractérisant des figures de synthèse permettant de décrire des composants de masques à appliquer aux données.
Exemples : `cercle, ncercle, fenetre, nfenetre, ...`

2.2.4.7 Les ensembles ζ et $\hat{\zeta}$ des descripteurs de déstructuration et des pointeurs

Les symboles de déstructuration permettent de fournir l'indication sur la façon dont ont été déstructurées (structurées) les entités de la donnée. Pour rappeler les éléments vus dans la description du modèle de la donnée, nous pouvons dire qu'ils contribuent à la détermination du champ index de ce modèle. Cette notion est nouvelle par rapport à la description proposée dans [Ing92] qui ne permettait pas, par exemple, de distinguer les composantes par la prise en compte d'une notion de voisinage.
De fait, les éléments déstructurants de l'ensemble ζ, tels ceux désignant un étiquetage, peuvent être regroupés selon le type de déstructuration qu'ils effectuent. Nous pouvons ainsi considérer un sur-ensemble de ces index identifié par $\hat{\zeta}$ dans lequel se trouveront ces descripteurs de type d'index.
Exemples :
`vois4 / vois8...` pour ζ
`etiq...` pour $\hat{\zeta}$
À ces ensembles s'ajoutent deux descripteurs de déstructuration spécifiques qui sont :
`gl` : désignant l'index de globalité : `gl`$\in \zeta$ et `gl`$\in \hat{\zeta}$
λ : (mot vide) désignant son complémentaire.

2.2.4.8 L'ensemble η des descripteurs modificatifs de descripteurs

Comme leur nom l'indique, les symboles de cet ensemble caractériseront des descripteurs permettant de modifier la sémantique des symboles qu'ils réfèrent sans en changer la nature ; le maximum des niveaux de gris est toujours un niveau de gris. Ce sont généralement des descripteurs agissant comme élément structurant (déstructurant) par changement de dimension de la donnée.
Exemples : `maximum, opposé, ...`

2.2.4.9 L'ensemble μ des descripteurs de jonction de concepts

La notion de descripteurs de jonction de concepts a été introduite par l'impossibilité qu'il y avait parfois en utilisant le langage proposé par J.M. Inglebert d'exprimer des conjonctions de concepts. Nous reparlerons de l'exploitation et de l'intérêt de ces descripteurs dans le chapitre traitant de la simplification des expressions (voir chap.2, 3e partie).
Exemples : `et, ou, xou, ...`

2.2.4.10 L'ensemble ν des descripteurs de jonction de sélecteurs

Par analogie avec l'ensemble μ des jonctions de concepts, nous proposons un ensemble ν de descripteurs de jonction de sélecteurs. Ils permettent de décrire le signal d'une donnée issue d'une

opération effectuant des traitements sur plusieurs autres données et dont l'effet modifie ce champ signal.
Exemples : `xou, +, -, ...`

Une remarque particulière est faite en 2.6 sur le descripteur `xou`

2.2.5 Remarques sur les descripteurs

2.2.5.1 Descripteurs complémentaires

Du fait de la dualité de la donnée dans le modèle qui est proposé, les descripteurs manipulant les composantes connexes vont devoir être regroupés par deux afin de permettre de décrire les entités cache κ et complémentaire du cache $\overline{\kappa}$ de la dite donnée. Ces descripteurs sont ceux appartenant aux ensembles ρ des descripteurs de relation (fig.2.9), ϕ des descripteurs de figure (fig.2.10), ζ des descripteurs de déstructuration (fig.2.11) et μ des descripteurs de jonction de concepts (fig.2.12).

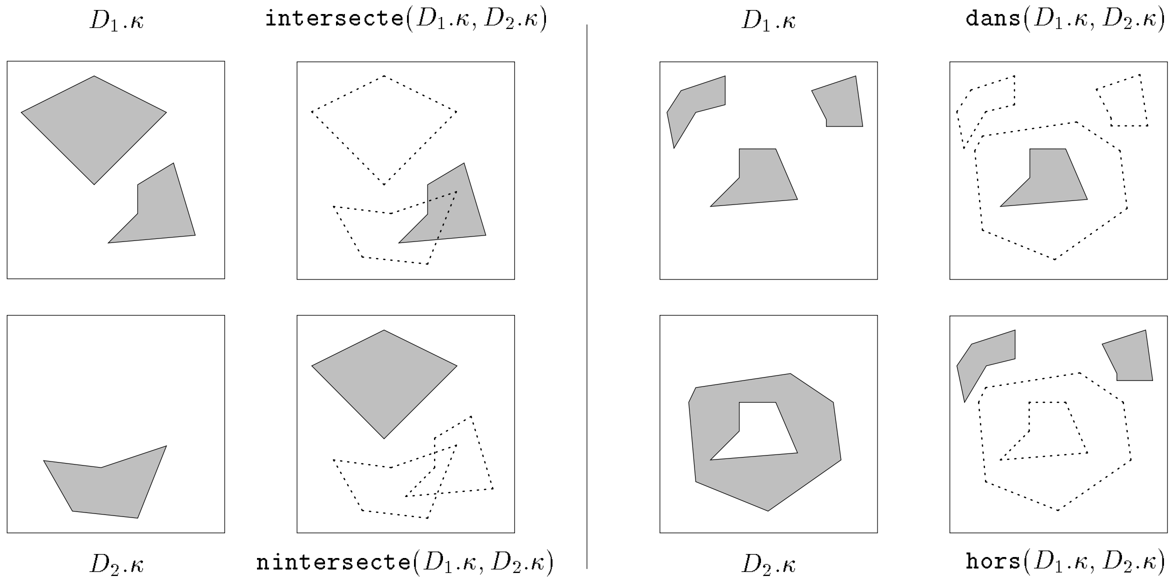


FIG. 2.9 - *Descripteurs de relations* `(inter,ninter)` *et* `(dans,hors)`

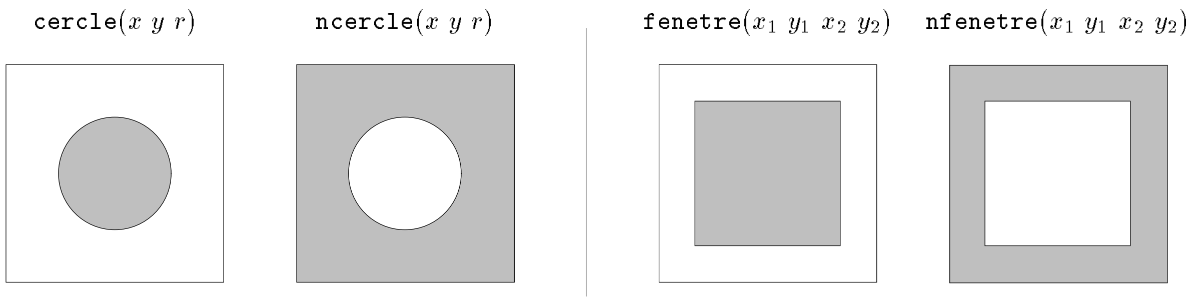


FIG. 2.10 - *Descripteurs de figures* `(cercle,ncercle)` *et* `(fenetre,nfenetre)`

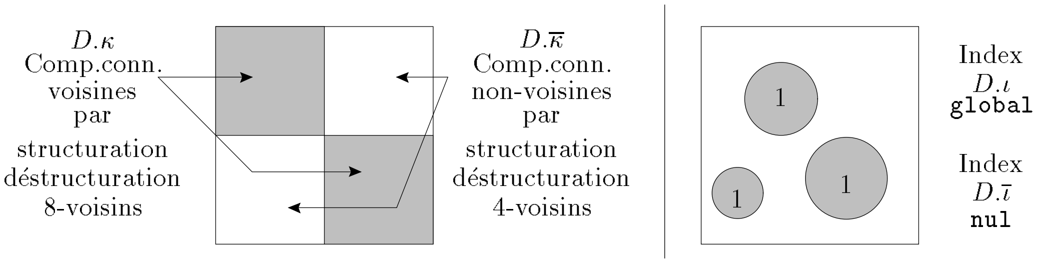


FIG. 2.11 - *Descripteurs de déstructuration (4-voisins,8-voisins) et (global,nul)*

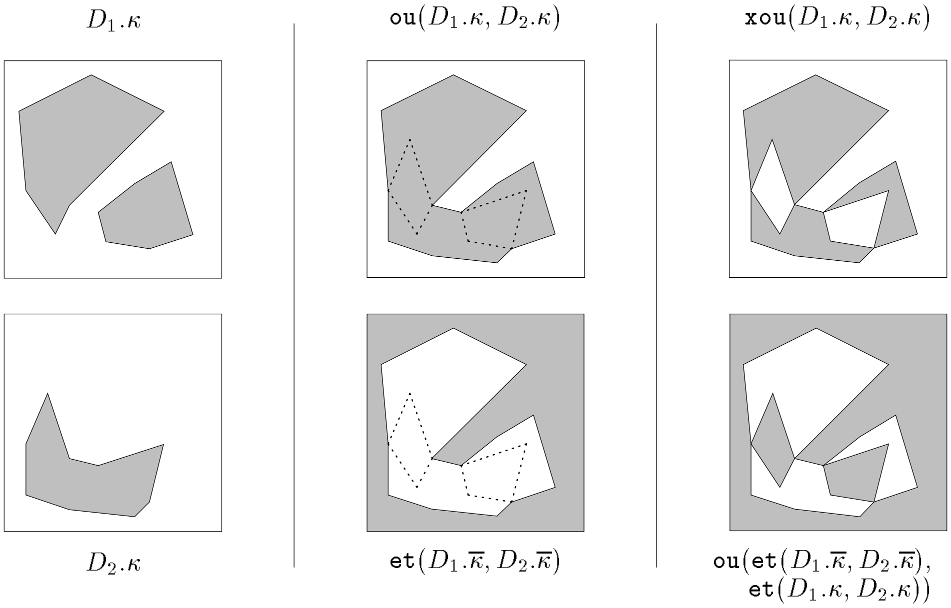


FIG. 2.12 - *Descripteurs de jonction de concepts (et,ou) et (xou,(et,ou))*

2.2.5.2 Domaine de validité et prise en compte du contexte

Toujours en référence à cette dualité de la composante *cache* de la donnée, et par delà la dualité de toutes les autres composantes, les descripteurs d ($d \in \varsigma$) de sélection possèdent tous une entité appelée *domaine de validité* et que nous notons Δ_d. Cette notion est à rapprocher des domaines d'attributs du calcul des prédicats [Die86, Mic84]. Bien qu'importante dans le cadre de la simplification des expressions de description (voir chap.2, 3[e] partie) elle n'était pas implantée effectivement dans le système. Un tel domaine, pour un descripteur donné est constitué de l'ensemble des valeurs pouvant lui être associées. Généralement formé d'un intervalle, il est fonction des caractéristiques (*taille, ...*) des données manipulées. Ainsi donc, l'ensemble de ces domaines de validité va constituer une partie de la connaissance du contexte lié aux données, et en particulier celui de leur acquisition. Nous trouverons en annexe C les expressions déterminant les domaines de validité pour les descripteurs actuellement implantés dans ExTI.

2.3 LDC : le Langage de Description de Concepts

La notion de concepts est ici quelque peu ambiguë. En effet, dans le système ExTI, elle est liée à la possibilité de relier un ensemble d'indices visuels décrits à une caractéristique possédant un nom dans domaine de l'application : un concept de l'application. Tout élément de description -mesure, propriété, vecteur de propriétés, disjonction de tels vecteurs, etc...- peut donc être appelé concept dans la mesure où il possède une valeur sémantique.
Nous allons garder ce terme de *concepts* pour qualifier le langage étudié ici, mais nous allons conserver, tout au long de ce document, la définition relative à la forme normale disjonctive (*DNF*) que prendra la description d'un cache.

2.3.1 Méthode utilisée pour l'étude du LDC

Avant de rentrer dans les détails de ces catégories syntaxiques et du fait même de leur imbrication que nous allons étudier, il nous a semblé important de les énumérer en donnant simplement leur expression.

2.3.1.1 Présentation des catégories

Sel : $<$*sélecteur*$>$ ::= sélecteur
| $<$*mesure*$>$
| (sous sélecteur)
| (sous $<$*mesure*$>$)
| (modifieur sélecteur)
| (modifieur $<$*mesure*$>$)
| (modifieur sous sélecteur)
| (modifieur sous $<$*mesure*$>$)
Mes : $<$*mesure*$>$ ::= valeur_numérique
| ($<$*sélecteur*$>$ $<$*déstructuration_d'une_base*$>$)
| $<$*fonction_sur_des_mesures*$>$
Fct : $<$*fonction_sur_des_mesures*$>$ ::= (opérateur $<$*mesure*$>$ $<$*mesure*$>$)
Dom : $<$*domaine*$>$ ::= [$<$*mesure*$>$ $<$*mesure*$>$]*
Ppt : $<$*propriété*$>$ ::= ($<$*sélecteur*$>$ $<$*domaine*$>$)

Cpt : <*concept*> ::= (<*propriété*>$^{+}$)
| (<*propriété*>$^{+}$ <*déstructuration_d'une_base*>)
| (relation <*déstructuration_d'une_base*> <*base*>$^{+}$)
| (figure <*mesure*>$^{+}$)
| (jonction <*base*> <*base*>)
| ~nom_de_concept
Bse : <*base*> ::= (<*concept*>*)
Dst : <*déstructuration_d'une_base*> ::= (déstructurant <*base*>)

2.3.1.2 La méthode

La méthode utilisée pour cette étude se décompose en trois phases :

Tout d'abord nous étudierons les langages associés aux descriptions de concepts, mesures, etc, qui sont les catégories syntaxiques de base du Langage de Descriptions des Données. Nous analysons chacune de ces catégories de façon indépendante en posant des hypothèses fortes sur celles-ci (fig.2.13) :

Hypothèse 1 *La catégorie syntaxique Mes, définissant les mesures est un terminal pour l'expression de la catégorie syntaxique Sel définissant les sélecteurs complexes.*

Hypothèse 2 *Les catégories syntaxiques Sel, définissant les sélecteurs complexes, Dst, définissant les éléments déstructurants (structurants), et Fct, définissant les fonctions sur des mesures, sont des terminaux pour l'expression de la catégorie syntaxique Mes définissant les mesures.*

Hypothèse 3 *La catégorie syntaxique Mes, définissant les mesures, est un terminal pour l'expression de la catégorie syntaxique Fct définissant les fonctions sur des mesures.*

Hypothèse 4 *La catégorie syntaxique Mes, définissant les mesures, est un terminal pour l'expression de la catégorie syntaxique Dom définissant les domaines.*

Hypothèse 5 *Les catégories syntaxiques Sel, définissant les sélecteurs complexes et Dom, définissant les domaines, sont des terminaux pour l'expression de la catégorie syntaxique Ppt définissant les propriétés.*

Hypothèse 6 *Les catégories syntaxiques Ppt définissant les propriétés, Bse, définissant les disjonctions de concepts, Dst, définissant les déstructurations de bases, et Mes, définissant les mesures, sont des terminaux pour l'expression de la catégorie syntaxique Cpt définissant les concepts.*

Hypothèse 7 *La catégorie syntaxique Cpt, définissant les concepts, est un terminal pour l'expression de la catégorie syntaxique Bse définissant les disjonctions de concepts.*

Hypothèse 8 *La catégorie syntaxique Bse, définissant les disjonctions de concepts, est un terminal pour l'expression de la catégorie syntaxique Dst définissant les déstructurations de bases.*

Dans un second temps nous enlevons les éventuelles ambiguïtés liées à ces hypothèses afin de rendre déterministes les catégories étudiées. Nous pourrons alors étudier les caractéristiques finales de ces catégories syntaxiques de base (fig.2.13) [5] et donner leur analyseur (voir Annexe B).

Trois des catégories syntaxiques proposées ont leur grammaire LL(2) -*Sel* et *Mes*- et LL(3) -*Cpt*. Le choix de ne pas les rendre LL(1) est délibéré. En effet, pour ce faire, les catégories syntaxiques manipulées dans la description de ces trois catégories auraient dû être dérivées pour permettre la factorisation des termes, limitant alors la lisibilité au niveau sémantique de ces mêmes catégories.

5. Les caractéristiques mentionnées sur ce diagramme font références aux restrictions des classes telles qu'elles ont été déterminées dans [Dej96a].

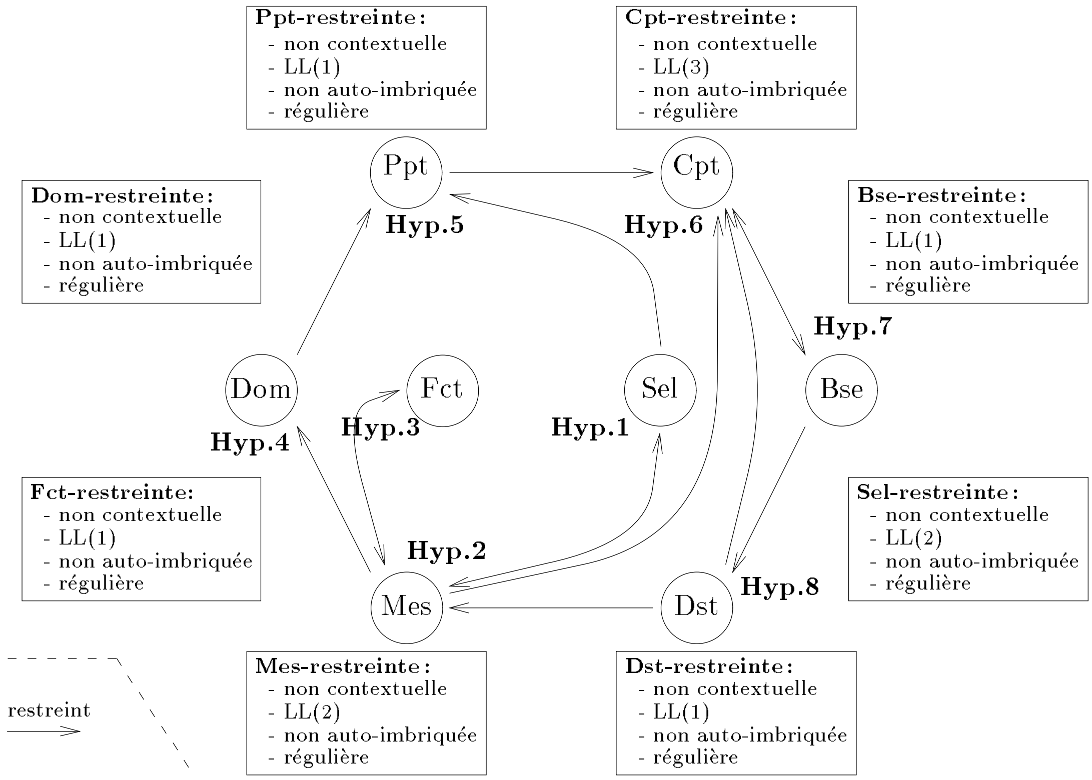


FIG. 2.13 - *Interdépendance des catégories syntaxiques de base*

2.3.2 Les catégories syntaxiques

Nous présentons ici les catégories syntaxiques définies dans le LDC. Pour chacune d'entre elles, nous allons donner la grammaire formelle qui lui est associée : son expression, souvent régulière, les ensembles de terminaux manipulés et ses schémas de règles de production en spécifiant les catégories syntaxiques manipulées. Un diagramme d'états est aussi proposé montrant le déterminisme de chaque catégorie syntaxique. Après avoir donné les caractéristiques de cette grammaire, nous détaillerons le rôle de la catégorie en terme de description en rappelant les éléments de son expression concernés, puis nous donnerons des exemples d'entités décrites par elle.

2.3.2.1 La catégorie <*sélecteur*>

◦ Son rôle

La catégorie syntaxique définissant les sélecteurs complexes a pour expression en Forme de Backus-Naur (BNF) :

<*sélecteur*> ::= sélecteur
| <*mesure*>
| (sous sélecteur)
| (sous <*mesure*>)
| (modifieur sélecteur)
| (modifieur <*mesure*>)
| (modifieur sous sélecteur)
| (modifieur sous <*mesure*>)

Le sélecteur décrit toute entité permettant d'identifier la nature de l'objet considéré, soit pour effectuer une mesure, soit pour en extraire une partie avec une propriété spécialisant un concept, soit simplement en nommant le signal. Il a donc la forme d'un terminal sélecteur simple **sélecteur** ou d'une mesure $<\mathit{mesure}>$, les deux pouvant être sous-sélectionnés (sous sélecteur), (sous$<\mathit{mesure}>$), tous pouvant être enfin altérés par des éléments modificatifs (modifieur sélecteur), (modifieur $<\mathit{mesure}>$), (modifieur sous sélecteur), (modifieur sous $<\mathit{mesure}>$).

◦ Exemples

`coul`	sélecteur de couleur
`prm`	sélecteur de périmètre
`(max prm)`	sélecteur précédent modifié par le maximum.
`(roug coul)`	sous sélecteur rouge des couleurs
`(max roug coul)`	sélecteur précédent modifié par le maximum.
`(abs (diff 128 (max (prm C))))`	sous sélecteur valeur absolue d'un calcul arithmétique.
`(max abs (diff 128 (max (prm C))))`	sélecteur précédent modifié par le maximum.

◦ De $<$*sélecteur*$>$ à Sel

L'expression des éléments produits est alors la suivante :

$$\begin{aligned} Sel \;=\; & \varsigma \mid Mes \mid \\ & (\ \eta^{+}\{\varsigma \mid Mes \mid \breve{\varsigma}\{\varsigma \mid Mes\}\}\ ) \mid \\ & (\ \breve{\varsigma}\ \{\varsigma \mid Mes\}\}\ ) \end{aligned}$$

Sa grammaire $G_{Sel} = \{\ N_{Sel}\ ,\ T_{Sel}\ ,\ \mathcal{P}_{Sel}\ ,\ Sel_0\}$ utilise les ensembles de terminaux suivants : $\{\varsigma, \breve{\varsigma}, \eta, (,)\ \} \subset T_{Sel}$

- ς : descripteurs de sélecteurs.
- $\breve{\varsigma}$: descripteurs de sous-sélecteurs.
- η : descripteurs modificatifs de descripteurs.

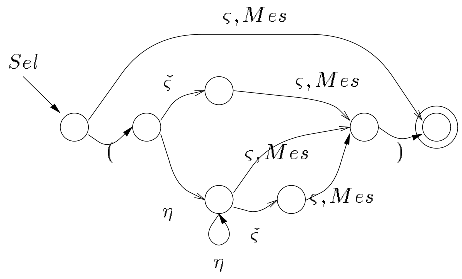
et manipule la catégorie syntaxique (Hyp.1) :

Mes : de définition d'une *mesure*.

$\{Mes\} \subset T_{Sel}$

dans les schémas de règles de production :

$$\mathcal{P}_{Sel} = \left| \begin{array}{lcl} Sel_0 & \longrightarrow & Sel_1 \\ Sel_0 & \longrightarrow & (Sel_2) \\ Sel_1 & \longrightarrow & \varsigma \\ Sel_1 & \longrightarrow & Mes_0 \\ Sel_2 & \longrightarrow & \breve{\varsigma}\ Sel_1 \\ Sel_2 & \longrightarrow & \eta\ Sel_3 \\ Sel_3 & \longrightarrow & \eta\ Sel_3 \\ Sel_3 & \longrightarrow & Sel_1 \\ Sel_3 & \longrightarrow & \breve{\varsigma}\ Sel_1 \end{array} \right.$$



◦ Ses caractéristiques

La grammaire G_{Sel} de cette catégorie $<$*sélecteur*$>$ est LL(2), déterministe, non auto-imbriquée et régulière.

2.3.2.2 La catégorie $<$*mesure*$>$

◦ Son rôle

La catégorie syntaxique définissant les mesures a pour expression en Forme de Backus-Naur (BNF) :

$$\begin{array}{lcl} <\mathit{mesure}> & ::= & \textsf{valeur\_numérique} \\ & \mid & (\ <\mathit{sélecteur}>\ <\mathit{déstructuration\_d'une\_base}>\ ) \\ & \mid & <\mathit{fonction\_sur\_des\_mesures}> \end{array}$$

La mesure décrit toute entité numérique fournie directement par la valeur valeur_numérique, ou résultant d'une réelle mesure sur des données ($<$*sélecteur*$>$ $<$*déstructuration_d'une_base*$>$), ou de fonctions sur d'autres mesures $<$*fonction_sur_des_mesures*$>$.

○ Exemples

`128`	valeur numérique `128` fournie
`(nombre-comp-conn C)`	mesure du nombre de composantes connexes décrites par `C`
`((max prm) C)`	résultat du calcul de la valeur maximale des périmètres des composantes déstructurées représentées par `C`

○ De <*mesure*> à Mes

L'expression après transformation des éléments produits est alors la suivante :

$$Mes = v \ | \ (\ Sel \ Dst \ ) \ | \ Fct$$

Sa grammaire $G_{Mes} = \{ \ N_{Mes} \ , \ T_{Mes} \ , \ \mathcal{P}_{Mes} \ , \ Mes_0 \}$ utilise l'ensemble de terminaux suivant : $\{v, (,) \ \} \subset T_{Mes}$

| v : valeurs numériques.

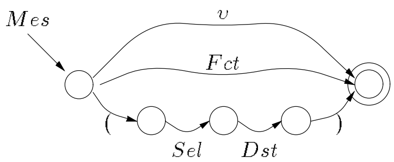
et manipule les catégories syntaxiques (Hyp.2) :

Sel : de définition d'un *sélecteur complexe.*
Dst : de définition d'une *déstructuration de bases.*
Fct : de définition d'une *fonction sur des mesures.*

$\{Sel, Dst, Fct\} \subset T_{Mes}$

dans les schémas de règles de production :

$$\mathcal{P}_{Mes} = \left| \begin{array}{lll} Mes_0 & \longrightarrow & v \\ Mes_0 & \longrightarrow & Fct_0 \\ Mes_0 & \longrightarrow & (Mes_1) \\ Mes_1 & \longrightarrow & Sel_0 \ Mes_2 \\ Mes_2 & \longrightarrow & Dst_0 \end{array} \right.$$



○ Ses caractéristiques

La grammaire G_{Mes} de cette catégorie <*mesure*> est LL(2), déterministe, non auto-imbriquée et régulière.

2.3.2.3 La catégorie <*fonction_sur_des_mesures*>

○ Son rôle

La catégorie syntaxique définissant les fonctions sur des mesures a pour expression en Forme de Backus-Naur (BNF) :

<*fonction_sur_des_mesures*> ::= (opérateur <*mesure*> <*mesure*>)

La fonction sur des mesures décrit toute entité résultant de calculs arithmétiques sur deux autres mesures opérateur <*mesure*> <*mesure*>.

○ Exemples

`(+ 128 (max (prm C)))` calcul arithmétique sur deux des mesures précédents.

○ De <*fonction_sur_des_mesures*> à Fct

L'expression des éléments produits est alors la suivante :

$$Fct = (\ \theta \ Mes \ Mes \ )$$

Sa grammaire $G_{Fct} = \{ \ N_{Fct} \ , \ T_{Fct} \ , \ \mathcal{P}_{Fct} \ , \ Fct_0 \}$ utilise les ensembles de terminaux suivants : $\{\theta, (,)\} \subset T_{Fct}$

| θ : descripteurs d'opération arithmétique.

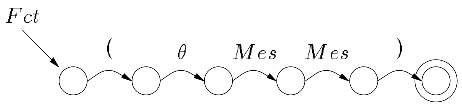
et manipule le catégorie syntaxique (Hyp.3) :

Mes : de définition d'une *mesure.*

$\{Mes\} \subset T_{Fct}$

dans les schémas de règles de production :

$$\mathcal{P}_{Fct} = \left| \begin{array}{lcl} Fct_0 & \longrightarrow & (\ Fct_1\ ) \\ Fct_1 & \longrightarrow & \theta\ Fct_2 \\ Fct_2 & \longrightarrow & Mes_0\ Fct_3 \\ Fct_3 & \longrightarrow & Mes_0 \end{array} \right.$$



∘ Ses caractéristiques
La grammaire G_{Fct} de cette catégorie <*fonction_sur_des_mesures*> est LL(1), déterministe, non auto-imbriquée et régulière.

2.3.2.4 La catégorie <*domaine*>

∘ Son rôle
La catégorie syntaxique définissant les domaines a pour expression en Forme de Backus-Naur (BNF) :

<*Domaine*> ::= [<*mesure*> <*mesure*>]*

Le domaine décrit un ensemble au cardinal pair de mesures ordonnées -si possible- de façon croissante. Chaque paire de mesure définit un intervalle semi-ouvert à droite ($[x, y[$), la succession de ces paires décrivant l'union de ces intervalles.

∘ Exemples

`128 256`	définit l'intervalle $[128, 256[$
`50 100 1500 2000`	définit l'union d'intervalle $[50, 100[\cup[1500, 2000[$
`10 ((max prm) C)`	définit un intervalle impossible à ordonner, et dont une des bornes correspond à la valeur maximale des périmètres des composantes `C`

∘ De <*domaine*> à Dom
L'expression des éléments produits est alors la suivante :

$$Dom = \{Mes\ Mes\}^*$$

Sa grammaire $G_{Dom} = \{\ N_{Dom}\ ,\ T_{Dom}\ ,\ \mathcal{P}_{Dom}\ ,\ Dom_0\}$ utilise les ensembles de terminaux suivants : $\{\ \lambda\ \} \subset T_{Dom}$

| λ : mot vide.

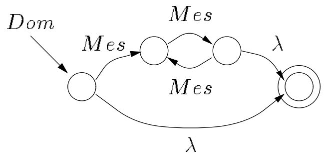
et manipule la catégorie syntaxique (Hyp.4) :

Mes : catégorie syntaxique de définition d'une mesure.

$\{Mes\} \subset T_{Dom}$

dans les schémas de règles de production :

$$\mathcal{P}_{Dom} = \left| \begin{array}{lcl} Dom_0 & \longrightarrow & Mes_0\ Dom_1 \\ Dom_0 & \longrightarrow & \lambda \\ Dom_1 & \longrightarrow & Mes_0\ Dom_2 \\ Dom_2 & \longrightarrow & Mes_0\ Dom_1 \\ Dom_2 & \longrightarrow & \lambda \end{array} \right.$$



∘ Ses caractéristiques
La grammaire G_{Dom} de cette catégorie <*domaine*> est LL(1), déterministe, non auto-imbriquée et régulière.

2.3.2.5 La catégorie <*propriété*>

∘ Son rôle
La catégorie syntaxique définissant les mesures a pour expression en Forme de Backus-Naur (BNF) :

<*propriété*> ::= (<*sélecteur*> <*domaine*>)

La propriété décrit une caractéristique de sélection déterminée par un sélecteur -au sens de Michalski- associé à un domaine décrit par la catégorie correspondante <*domaine*> donnant ainsi :

(*<sélecteur>* *<domaine>*).

○ **Exemples**

`(nivg 128 256)`	niveaux de gris compris entre 128 inclus et 256 exclus
`(srf 50 100 1500 2000)`	surface des composantes dans l'intervalle $[50, 100[\cup [1500, 2000[$
`((max dist) 1 4)`	maximum des distances des pixels au bord de sa composante compris entre 1 inclus et 4 exclus

○ **De <*propriété*> à** Ppt

L'expression après transformation des éléments produits est alors la suivante :

$$Ppt = (Sel\ Dom)$$

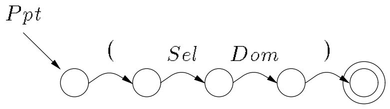
Sa grammaire $G_{Ppt} = \{\ N_{Ppt}\ ,\ T_{Ppt}\ ,\ \mathcal{P}_{Ppt}\ ,\ Ppt_0\}$ n'utilise pas d'ensembles de terminaux mais manipule les catégories syntaxiques (Hyp.5) :

Sel : de définition d'un *sélecteur.*
Dom : de définition d'un *domaine.*

$$\{Sel, Dom\} \subset T_{Ppt}$$

dans les schémas de règles de production :

$$\mathcal{P}_{Ppt} = \left| \begin{array}{lcl} Ppt_0 & \longrightarrow & (\ Ppt_1\ ) \\ Ppt_1 & \longrightarrow & Sel_0\ Ppt_2 \\ Ppt_2 & \longrightarrow & Dom_0 \end{array} \right.$$



○ **Ses caractéristiques**

La grammaire G_{Ppt} de cette catégorie <*propriété*> est LL(1), déterministe, non auto-imbriquée et régulière.

2.3.2.6 La catégorie <*concept*>

○ **Son rôle**

La catégorie syntaxique définissant les concepts a pour expression en Forme de Backus-Naur (BNF) :

<concept>	::=	(*<propriété>*$^+$)
	|	(*<propriété>*$^+$ *<déstructuration_d'une_base>*)
	|	(relation *<déstructuration_d'une_base>* *<base>*$^+$)
	|	(figure *<mesure>*$^+$)
	|	(jonction *<base>* *<base>*)
	|	∼nom_de_concept

Le concept représente la description d'entités élémentaires que l'on peut extraire de l'image. Il est formé d'une accumulation (conjonction) de propriétés (*<propriété>*$^+$) pouvant caractériser éventuellement une base possédant un élément de déstructuration (*<propriété>*$^+$ *<déstructuration_d'une _base>*) (voir catégorie syntaxique *<base>* et *<déstructuration_d'une_base>*). Il peut être formé aussi par une relation entre bases (relation *<déstructuration_d'une_base>* *<base>*), par une description de figure de synthèse (figure *<mesure>*$^+$), ou par une jonction (conjonction ou disjonctions) particulière entre bases (jonction *<base>* *<base>*). Enfin, il peut être constitué d'une référence au complémentaire d'un concept existant et nommé ∼nom_de_concept.

○ **Exemples**

`((nivg 128 256))`	objets clairs (niveaux de gris supérieurs ou égaux à 128)
`((surf 1 150)`	petits objets clairs
`((vois4)(((nivg 128 256))))`	(surface des objets clairs inférieure à 150)

`(fenetre 100 150 200 250)`	objet rectangulaire défini par les points : supérieur-gauche $(x_1, y_1) = (100, 150)$ et inférieur-droit $(x_2, y_2) = (200, 250)$
`(inter ((vois4)(((nivg 128 256))))` `((fenetre 100 150 200 250)))`	objets clairs touchant (`inter`section)... ...un objet rectangulaire prédéfini.
`(et (((nivg 128 256)))` `((fenetre 100 150 200 250)))`	points des objets clairs ... ...qui sont aussi dans les rectangles
∼OBJETS_CLAIRS	objets non clairs (niveaux de gris inférieurs à 128)

○ **De <*concept*> à** *Cpt*

L'expression des éléments produits est alors la suivante :

$$Cpt = (Ppt^+\ \{\lambda|Dst\}) \mid (\rho\ Dst\ Bse^+) \mid (\phi\ Mes^+) \mid (\mu\ Bse\ Bse) \mid \mho$$

Sa grammaire $G_{Cpt} = \{\ N_{Cpt}\ ,\ T_{Cpt}\ ,\ \mathcal{P}_{Cpt}\ ,\ Cpt_0\}$ utilise les ensembles de terminaux suivants : $\{\ \rho, \phi, \mu, (,), \mho\ \} \subset T_{Cpt}$

- ρ : descripteur de relation.
- ϕ : descripteur de figure.
- μ : descripteur de jonction.
- $\mho$: complément de la description du cache de l'objectif étiqueté Ω ($\mho = \Omega._{\overline{\kappa}}$).

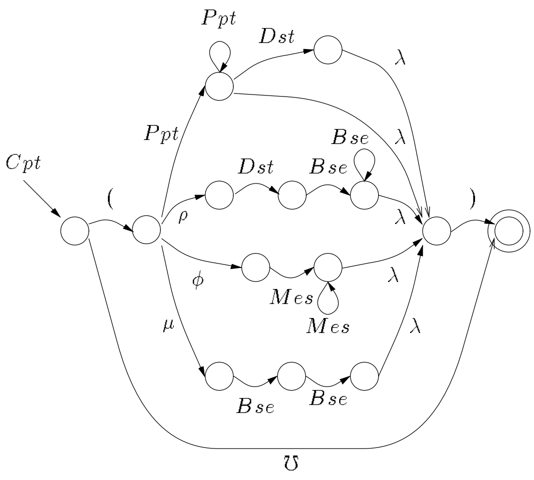
et manipule les catégories syntaxiques (Hyp.6) :

- *Ppt* : de définition d'une *propriété*.
- *Mes* : de définition d'une *mesure*.
- *Bse* : de définition d'une *base*.
- *Dst* : de définition d'une *déstructuration de base*.

$$\{Ppt, Mes, Bse, Dst\} \subset T_{Cpt}$$

dans les schémas de règles de production :

$$\mathcal{P}_{Cpt} = \left|\begin{array}{lcl} Cpt_0 & \longrightarrow & (Cpt_1) \\ Cpt_1 & \longrightarrow & \mho \\ Cpt_1 & \longrightarrow & Ppt_0\ Cpt_2 \\ Cpt_1 & \longrightarrow & \rho\ Cpt_3 \\ Cpt_1 & \longrightarrow & \phi\ Cpt_5 \\ Cpt_1 & \longrightarrow & \mu\ Cpt_7 \\ Cpt_2 & \longrightarrow & Ppt_0\ Cpt_2 \\ Cpt_2 & \longrightarrow & Dst_0 \\ Cpt_2 & \longrightarrow & \lambda \\ Cpt_3 & \longrightarrow & Dst_0\ Cpt_4 \\ Cpt_4 & \longrightarrow & Bse_0\ Cpt_4 \\ Cpt_4 & \longrightarrow & \lambda \\ Cpt_5 & \longrightarrow & Mes_0\ Cpt_6 \\ Cpt_6 & \longrightarrow & Mes_0\ Cpt_6 \\ Cpt_6 & \longrightarrow & \lambda \\ Cpt_7 & \longrightarrow & Bse_0\ Cpt_8 \\ Cpt_8 & \longrightarrow & Bse_0 \end{array}\right.$$



○ **Ses caractéristiques**

La grammaire G_{Cpt} de cette catégorie <*concept*> est LL(3), déterministe, non auto-imbriquée et régulière.

2.3.2.7 La catégorie <*base*>

○ **Son rôle**

La catégorie syntaxique définissant les bases a pour expression en Forme de Backus-Naur (BNF) :

$$<base> \quad ::= \quad (\ <concept>^*\ )$$

La base est une description d'entités définie par une disjonction de concepts.

◦ Exemples

`(((nivg 128 256)))`	Ensemble des objets clairs
`(((nivg 0 128))`	Ensemble des objets sombres
`  (inter (((nivg 128 256)))`	ou objets clairs touchant (`inter`section)
`        ((fenetre 100 150 200 250))))`	un objet rectangulaire

◦ De <*base*> à Bse

L'expression des éléments produits est alors la suivante :

$$Bse = (Cpt^*)$$

Sa grammaire $G_{Bse} = \{\ N_{Bse}\ ,\ T_{Bse}\ ,\ \mathcal{P}_{Bse}\ ,\ Bse_0\}$ utilise les ensembles de terminaux suivants : $\{\ (,)\ \} \subset T_{Bse}$

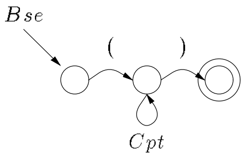
et manipule la catégorie syntaxique (Hyp.6) :

Cpt : de définition d'un *concept*.

$\{Cpt\} \subset T_{Bse}$

dans les schémas de règles de production :

$$\mathcal{P}_{Bse} = \left| \begin{array}{lcl} Bse_0 & \longrightarrow & (Bse_1) \\ Bse_1 & \longrightarrow & Cpt_0\ Bse_1 \\ Bse_1 & \longrightarrow & \lambda \end{array} \right.$$



◦ Ses caractéristiques

La grammaire G_{Bse} de cette catégorie <*base*> est LL(1), déterministe, non auto-imbriquée et régulière.

2.3.2.8 La catégorie <*déstructuration_d'une_base*>

◦ Son rôle

La catégorie syntaxique définissant la déstructuration d'une base a pour expression en Forme de Backus-Naur (BNF) :

<*déstructuration_d'une_base*> ::= (déstructurant <*base*>)

La déstructuration d'une base permet de considérer des entités constituant des composantes plus complexes et définies par la <*base*>. L'élément **déstructurant** désignera alors la façon d'examiner les entités de ces composantes.

◦ Exemples

`((vois4) (((nivg 128 256))))`	Ensemble des points des objets clairs de l'image examinés selon les 4-voisins.

Sa grammaire

L'expression des éléments produits est alors la suivante :

$$Dst = ((\zeta)Bse)$$

Sa grammaire $G_{Dst} = \{\ N_{Dst}\ ,\ T_{Dst}\ ,\ \mathcal{P}_{Dst}\ ,\ Dst_0\}$ utilise les ensembles de terminaux suivants : $\{\ \zeta, (,)\ \} \subset T_{Bse}$

$\big|\ \zeta$: descripteur de *déstructuration de base*.

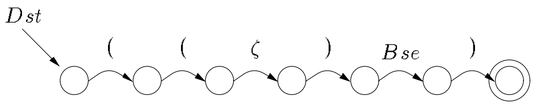
et manipule la catégorie syntaxique (Hyp.8):

Bse : catégorie syntaxique de définition d'une *base*.

$\{Bse\} \subset T_{Dst}$

dans les schémas de règles de production :

$$\mathcal{P}_{Dst} = \left| \begin{array}{lcl} Dst_0 & \longrightarrow & (Dst_1) \\ Dst_1 & \longrightarrow & (Dst_2)\ Bse_0 \\ Dst_2 & \longrightarrow & \zeta \end{array} \right.$$



◦ Ses caractéristiques
La grammaire G_{Dst} de cette catégorie <*déstructuration_d'une_base*> est LL(1), déterministe, non auto-imbriquée et régulière.

2.4 LDD : le Langage de Description des Données

Dans cette section, nous allons étudier le formalisme de description des données de traitement d'image ; nous retrouvons donc ses cinq champs : trame, cache, index, signal et codage dont nous ne rappellerons pas les définitions mais pour lesquels nous reprendrons quelques exemples.

2.4.1 La catégorie syntaxique <*trame*>

◦ Sa forme BNF
La catégorie syntaxique définissant les trames a pour expression en Forme de Backus-Naur :

<*trame*> ::= ([(<*mesure*> <*mesure*>)]*)

◦ Exemples

`()`	Trame nulle de la donnée de dimension 0
`((0 512) (0 512))`	Trame d'une image de taille 512x512
`((0 512) (0 512) (0 2))`	Trame d'une image couleur (RGB) de taille 512x512

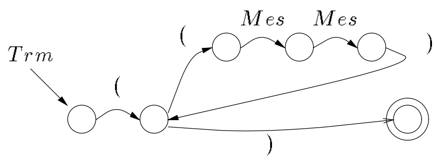
◦ De <*trame*> à Trm
L'expression des éléments produits est alors la suivante :

$$Trm = (\{(Mes\ Mes)\}^*)$$

Sa grammaire $G_{Trm} = \{\ N_{Trm}\ ,\ T_{Trm}\ ,\ \mathcal{P}_{Trm}\ ,\ Trm_0\}$ utilise les ensembles de terminaux suivants :
$\{\ (,)\ \} \subset T_{Trm}$
et manipule la catégorie syntaxique :
Mes : de définition d'une *mesure*.
$\{Mes\} \subset T_{Trm}$

dans les schémas de règles de production :

$$\mathcal{P}_{Trm} = \left| \begin{array}{lcl} Trm_0 & \longrightarrow & (Trm_1) \\ Trm_1 & \longrightarrow & (Trm_2)\ Trm_1 \\ Trm_1 & \longrightarrow & \lambda \\ Trm_2 & \longrightarrow & Mes_0\ Mes_0 \end{array} \right.$$



◦ Ses caractéristiques
La grammaire G_{Trm} de cette catégorie <*trame*> est LL(1), déterministe, non auto-imbriquée et régulière.

2.4.2 La catégorie syntaxique <*cache*>

◦ Sa forme BNF
La catégorie syntaxique définissant les caches a pour expression en Forme de Backus-Naur :

<*cache*> ::= `t`
| <*base*>

◦ Exemples

`t`	Cache global rassemblant tout pixel
`( )`	Cache nul (son complémentaire)
`(((nivg 128 256)))`	Cache déterminé par les pixels *clairs*

◦ De <*cache*> à Cch

L'expression des éléments produits est alors la suivante :

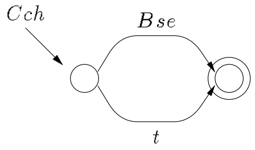
$$Cch = t \mid Bse$$

Sa grammaire $G_{Cch} = \{\ N_{Cch}\ ,\ T_{Cch}\ ,\ \mathcal{P}_{Cch}\ ,\ Cch_0\}$ utilise les terminaux suivants : $\{t\} \subset T_{Cch}$ et manipule la catégorie syntaxique :

Bse : catégorie syntaxique de définition d'une *base*.
$\{Bse\} \subset T_{Cch}$

dans les schémas de règles de production :

$$\mathcal{P}_{Cch} = \left| \begin{array}{lcl} Cch_0 & \longrightarrow & t \\ Cch_0 & \longrightarrow & (\ ) \\ Cch_0 & \longrightarrow & Bse_0 \end{array} \right.$$



◦ Ses caractéristiques

La grammaire G_{Cch} de cette catégorie <*cache*> est LL(1), déterministe, non auto-imbriquée et régulière.

2.4.3 La catégorie syntaxique <*index*>

◦ Sa forme BNF

La catégorie syntaxique définissant les caches a pour expression en Forme de Backus-Naur :

<*index*> ::= ( )
| (*gl*)
| (pointeur déstructurant)

◦ Exemples

`(gl)`	Index global
`()`	Index nul (complémentaire du précédent)
`(etiq vois4)`	Index par étiquetage 4-voisins
`(gl vois8)`	Index global avec information de déstructuration (complémentaire du précédent)

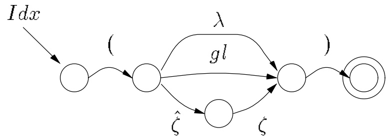
◦ De <*index*> à Idx

L'expression des éléments produits est alors la suivante :

$$Idx = () \mid (\hat{\zeta}\ \zeta) \mid (gl)$$

Sa grammaire $G_{Idx} = \{\ N_{Idx}\ ,\ T_{Idx}\ ,\ \mathcal{P}_{Idx}\ ,\ Idx_0\}$ utilise les terminaux suivants :
$\{\hat{\zeta}, \zeta, (,), gl\} \subset T_{Idx}$
et ne manipule pas de catégorie syntaxique dans les schémas de règles de production :

$$\mathcal{P}_{Idx} = \left| \begin{array}{lcl} Idx_0 & \longrightarrow & (Idx_1) \\ Idx_1 & \longrightarrow & \hat{\zeta}\ Idx_2 \\ Idx_1 & \longrightarrow & gl \\ Idx_1 & \longrightarrow & \lambda \\ Idx_2 & \longrightarrow & \zeta \end{array} \right.$$



◦ Ses caractéristiques

La grammaire G_{Idx} de cette catégorie <*index*> est LL(2), déterministe, non auto-imbriquée et régulière.

2.4.4 La catégorie syntaxique <*signal*>

◦ Sa forme BNF

La catégorie syntaxique définissant les signaux a pour expression en Forme de Backus-Naur :

<*signal*>	::=	<*sélecteur*>
	|	(jonction <*signal*> <*signal*>)

Le signal d'une donnée est décrit par un sélecteur <*sélecteur*>, ou par une jonction sur des signaux (jonction <*signal*> <*signal*>).

◦ Exemples

`vab`	signal d'une image binaire
`((prm) C)`	signal du vecteur contenant les périmètres des composantes `C`
`((moy nivg))`	signal de la valeur moyenne des niveaux de gris
`(+ 128 (nivg))`	signal d'une image à niveaux d'intensité rehaussés.
`(xou (nivg C) ((opp nivg) ~C))`	jonction de deux mesures sur des composantes `C` et son complémentaire `~C`.

◦ De <*signal*> à Sgn

L'expression des éléments produits est alors la suivante :

$$Sgn = Sel \ | \ (\nu \ Sgn \ Sgn)$$

Sa grammaire $G_{Sgn} = \{ \ N_{Sgn} \ , \ T_{Sgn} \ , \ \mathcal{P}_{Sgn} \ , \ Sgn_0\}$ utilise l'ensemble de terminaux suivants : $\{\nu, (,)\} \subset T_{Sgn}$

| ν : descripteurs de jonction de sélecteurs.

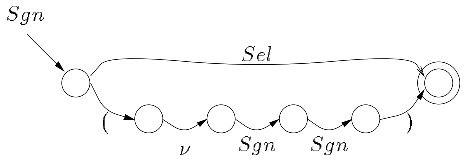
et manipule les catégories syntaxiques :

Sel : de définition d'un *sélecteur*.

$\{Sel\} \subset T_{Sgn}$

dans les schémas de règles de production :

$$\mathcal{P}_{Sgn} = \left| \begin{array}{lcl} Sgn_0 & \longrightarrow & Sel_0 \\ Sgn_0 & \longrightarrow & (Sgn_1) \\ Sgn_1 & \longrightarrow & \nu Sgn_2 \\ Sgn_2 & \longrightarrow & Sgn_0 \ Sgn_3 \\ Sgn_3 & \longrightarrow & Sgn_0 \end{array} \right.$$



◦ Ses caractéristiques

La grammaire G_{Sgn} de cette catégorie <*signal*> est LL(2), déterministe, auto-imbriquée et régulière.

2.4.5 La catégorie syntaxique <*codage*>

◦ Sa forme BNF

La catégorie syntaxique définissant les codages a pour expression en Forme de Backus-Naur :

<*codage*> ::= (<*domaine*>)

◦ Exemples

`(0 256)` pour une image de niveau de gris

◦ De <*codage*> à Cdg

L'expression des éléments produits est alors la suivante :

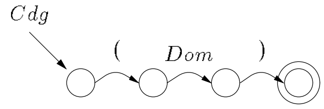
$$Cdg = (\ Dom \ )$$

Sa grammaire $G_{Cdg} = \{ \ N_{Cdg} \ , \ T_{Cdg} \ , \ \mathcal{P}_{Cdg} \ , \ Cdg_0\}$ utilise les terminaux suivants : $\{ \ (,) \ \} \subset T_{Cdg}$

et manipule la catégorie syntaxique :

Dom : catégorie syntaxique de définition d'un *domaine*.
$\{Dom\} \subset T_{Cdg}$
dans les schémas de règles de production :

$\mathcal{P}_{Cdg} = \mid Cdg_0 \longrightarrow (Dom_0)$



○ **Ses caractéristiques**
La grammaire G_{Cdg} de cette catégorie <*codage*> est LL(1), déterministe, non auto-imbriquée et régulière.

2.5 Règles transformationnelles et d'affixes de LEO

2.5.1 Règles transformationnelles

Deux types de règles transformationnelles sont utilisées. Les premières -*règles de substitution*- vont permettre de passer d'une expression de haut-niveau à une expression ne contenant pas de références à des expressions nommées. Les secondes -*méta-règles et règles de manipulation de concepts*- sont liées à certaines configurations des expressions rencontrées dans l'étude des catégories syntaxiques et vont permettre de simplifier ces dernières afin de générer des expressions reconnues par la grammaire développée précédemment. Pour exprimer ces transformations, nous allons utiliser la notation de Winograd -issue de Akmajian et Heny- rapportée dans [Sab88] :

0	suppression
+	adjonction comme frère
>	adjonction comme fils à droite
<	adjonction comme fils à gauche
#	adjonction complexe

2.5.1.1 Règles de substitution

La simplification d'expression par manipulation des opérateurs implique un objectif correctement écrit (forme canonique). L'utilisation d'un vocabulaire exécutable pour développer des expressions d'objectifs de haut-niveau, implique, pour la résolution de ces objectifs, le développement de règles transformationnelles permettant d'effectuer des substitutions des noms et champs de données par leur description. Ainsi, nous allons aborder ces règles en examinant les substitutions des entités étiquetées label.x par les expressions des champs x d'une donnée $\mathcal{D}$; c'est à dire en partant des axiomes des grammaires des catégories syntaxiques relatives à ces champs.

Règle de substitution du champ *trame* : $Trm_0 \longrightarrow (\ Trm_1\ )$
La réutilisation d'une trame ne peut être faite qu'à l'intérieur d'un champ *trame*. Nous aurons alors deux règles de substitution :

1. Remplacement de $\{(Mes\ Mes)\}^n$ par Trm_0 dans $Trm = (\{(Mes\ Mes)\}^*)$:
   Suppression du premier niveau de parenthésage de l'expression de substitution.
   $(\ \alpha\ \text{label}.\tau\ \beta\ ) \longrightarrow (\ \alpha\ Trm_0\ \beta\ ) \longrightarrow (\ \alpha\ (\ Trm_1\ )\ \beta\ )$

Description Structurelle	(	α	(	Trm_1	)	β	)
	1	2	3	4	5	6	7
Changement Structurel	1	2	0	4	0	6	7
	(	α		Trm_1		β	)

2. Aucune transformation dans les autres cas :
   label.$\tau \longrightarrow Trm_0$

Règle de substitution du champ *cache* : $Cch_0 \longrightarrow t \mid Bse_0$
La réutilisation d'un cache peut être faite chaque fois que la catégorie syntaxique $Bse = (Cpt^+)$ est utilisée, qu'elle se produise dans une relation $(\rho\ Dst\ Bse^+)$, dans une jonction de bases $(\mu\ Bse\ Bse)$, ou dans une déstructuration de base $((\zeta)\ Bse)$. Nous aurons alors quatre règles de substitution :

1. Remplacement de Cpt par t dans $Bse = (Cpt^+)$:
   (α label.κ β) $\longrightarrow$ (α Cch_0 β) $\longrightarrow$ (α t β)

Description Structurelle	(	α	t	β	)
	1	2	3	4	5
Changement Structurel	0	0	3	0	0
			t		

2. Remplacement de Cpt par Bse_0 dans $Bse = (Cpt^+)$:
   Suppression du premier niveau de parenthésage de l'expression de substitution :
   (α label.κ β) $\longrightarrow$ (α Cch_0 β) $\longrightarrow$ (α Bse_0 β) $\longrightarrow$ (α (Bse_1) β)

Description Structurelle	(	α	(	Bse_1	)	β	)
	1	2	3	4	5	6	7
Changement Structurel	1	2	0	4	0	6	7
	(	α		Bse_1		β	)

3. Aucune transformation dans les autres cas :
   label.κ $\longrightarrow Cch_0$

Règle de substitution du champ *index* : $Idx_0 \longrightarrow () \mid (gl) \mid (\hat{\zeta}\ \zeta)$
La réutilisation d'un index peut être faite chaque fois que la catégorie syntaxique $Dst = ((\zeta)\ Bse)$ est utilisée, qu'elle se produise dans une mesure $(Sel\ Dst)$, dans un concept $(Ppt^+\ Dst)$, ou dans une relation $(\rho\ Dst\ Bse^+)$. Nous aurons alors quatre règles de substitution de ζ :

1. Remplacement de ζ par () dans $Dst = ((\zeta)\ Bse)$:
   Cette substitution est incorrecte au regard de la grammaire de LEO. Cependant, si nous considérons des règles de simplification permettant d'annuler l'effet de cette substitution, celle-ci peut être traduite par la transformation nulle :
   ((label.ι) Bse) $\longrightarrow$ ((Idx_0) Bse) $\longrightarrow$ ((( )) Bse)

Description Structurelle	(	(	(	)	)	Bse	)
	1	2	3	4	5	6	7
Changement Structurel	1	0	0	0	0	0	7
	(						)

2. Remplacement de ζ par (gl) dans $Dst = ((\zeta)\ Bse)$:
   ((label.ι) Bse) $\longrightarrow$ ((Idx_0) Bse) $\longrightarrow$ (((gl)) Bse)

Description Structurelle	(	(	(	gl	)	)	Bse	)
	1	2	3	4	5	6	7	8
Changement Structurel	1	2	0	4	0	6	7	8
	(	(		gl		)	Bse	)

3. Remplacement de ζ par $(\hat{\zeta}\ \zeta)$ dans $Dst = ((\zeta)\ Bse)$:
   Suppression du premier niveau de parenthésage de l'expression de substitution :
   ((label.ι) Bse) $\longrightarrow$ ((Idx_0) Bse) $\longrightarrow$ (($(\hat{\zeta}\zeta)$) Bse)

Description Structurelle	(	(	(	$\hat{\zeta}$	ζ	)	)	Bse	)
	1	2	3	4	5	6	7	8	9
Changement Structurel	1	2	0	0	5	0	7	8	9
	(	(			ζ		)	Bse	)

4. Aucune transformation dans les autres cas :
   label.ι $\longrightarrow Idx_0$

Règle de substitution du champ *signal*: $Sgn_0 \longrightarrow Sel \mid (\nu\ Sgn\ Sgn)$
La réutilisation d'un signal peut être faite chaque fois que les catégories syntaxiques <*sélecteur*> ou <*mesure*> sont utilisées, que ce soit dans une mesure ($Sel\ Dst$), dans une propriété ($Sel\ Dom$), pour les sélecteurs, ou dans une fonction sur des mesures ($\theta\ Mes\ Mes$), dans un domaine $\{Mes\ Mes\}^*$, dans un concept décrit par une figure ($\phi\ Mes^+$), dans une description de trame ($\{(Mes\ Mes)\}^*$), ou dans un autre signal ($\nu\ Sgn\ Sgn$), pour les mesures. De plus, les substitutions qui interviennent dans la catégorie syntaxique <*sélecteur*> peuvent s'effectuer sur le `sélecteur`; ces substitutions sont donc à examiner en premier. Nous aurons alors onze règles de substitution :

1. Remplacement de ς par $Sgn = Sel = (\ldots\ \varsigma)$ dans $Sel = (\eta^+\ \varsigma)$:
   Suppression du premier niveau de parenthésage de l'expression de substitution :
   $(\ (\ \eta^+\ \mathsf{label}.\sigma\ ) \longrightarrow (\ (\ \eta^+\ Sel_0\ )\ \longrightarrow (\ (\ \eta^+\ (\alpha\ \varsigma)\ )$

Description Structurelle	(	η^+	(	α	ς	)	)
	1	2	3	4	5	6	7
Changement Structurel	1	2	0	4	5	0	7
	(	η^+		α	ς		)

2. Remplacement de ς dans $Sel = (\eta^*\ \breve{\varsigma}\ \varsigma)$:
   Les substitutions de ς dans $Sel = (\eta^*\ \breve{\varsigma}\ \varsigma)$ seront incorrectes si l'élément de substitution n'est ni un sélecteur ς ni une fonction sur des mesures. Dans ces deux cas, nous appliquerons des règles transformationnelles nulles.
3. Remplacement de ς par $Sgn = Mes = \upsilon$:
   Ces substitutions sont incorrectes au regard de la grammaire de LEO. Des règles de transformations nulles seront appliquées.
4. Remplacement de ς par $Sgn = Mes = (Sel\ Dst)$ dans $Sel = (\eta^+\ \varsigma) \mid (\eta^*\ \breve{\varsigma}\varsigma) \equiv (\alpha\ \varsigma)$:
   Ces substitutions suivent une contrainte de non associativité liée à la catégorie syntaxique <sélecteurs>. La transformation est alors exprimée par :
   $(\ (\ \alpha\ \mathsf{label}.\sigma\ ) \longrightarrow (\ (\ \alpha\ Sel_0\ )\ \longrightarrow (\ (\ \alpha\ (\varsigma\ Dst)\ )$

Description Structurelle	(	α	(	ς	Dst	)	)
	1	2	3	4	5	6	7
Changement Structurel	1>(	2	0	4	)<5	0	7
	((	α		ς	) Dst		)

5. Remplacement de ς par $Sgn = (\nu\ Sgn\ Sgn)$:
   Ces substitutions de nature complexe seront acceptées telles quelles, mais seront gérées par des règles syntaxiques s'appuyant sur les notions de distributivité liée aux terminaux de l'ensemble ν. Elles seront ainsi gérées par une méta-règle.
6. Remplacement de Sel dans $Mes = (Sel\ Dst)$:
   Les substitutions de Sel par autre chose que Sel sont incorrectes au regard de la grammaire de LEO. Comme nous l'avons vu précédemment, cette impossibilité peut être traduite par des règles transformationnelles générant l'élément nul ( ).
7. Substitution particulière de Sel par $Sgn = Mes = (Sel\ Dst)$ dans $Ppt = (Sel\ Dom)$:
   Lorsque la propriété Ppt est utilisée dans un concept $Cpt = (Ppt^+\ Dst)$ et que les deux éléments de déstructuration sont identiques, alors il y a factorisation de cet élément. Ce phénomène décrit dans [Ing92] pour les spécialisations et disjonction de concepts ne peut être représenté directement par une règle transformationnelle. Le problème sera donc géré par les méta-règles et des règles de manipulation d'expression.
8. Substitution particulière de Sel par $Sgn = (\nu\ Sgn\ Sgn)$ dans $Ppt = (Sel\ Dom)$:
   Lorsque la propriété Ppt est utilisée dans un concept $Cpt = (\ldots(Sel\ Dom)\ldots)$, la substitution de Sel par une jonction va entraîner la création d'un concept plus complexe formé par la jonction analogue sur les bases déstructurées. Ceci ne peut s'exprimer par une règle

transformationnelle du modèle des précédentes, mais sera gérée par une des méta-règles que nous verrons plus loin.

9. Autres substitutions de Sel dans $Ppt = (Sel\ Dom)$:
Les substitutions de Sel par autre chose que Sel et la dérivation précédente $Mes = (Sel\ Dst)$ sont incorrectes et se traduiront par une transformation nulle.
10. Remplacement de Mes par $Sgn = Sel$ dans $Sel = Mes$:
Les substitutions des mesures par un signal seront incorrectes, et donc traduites par des transformations nulles, dans le cas où l'entité de remplacement est une sélecteur $Sgn = Sel$.
11. Remplacement de Mes par $Sgn = (\nu\ Sgn\ Sgn)$ dans $Sel = Mes$:
Un autre cas est traduit par la transformation nulle. Il considère le signal équivalent à la jonction sur les mesures $Sgn = (\nu\ Sgn\ Sgn)$. Cette substitution sera annulée lorsque le terminal de jonction ne s'apparie pas avec un élément de l'ensemble des terminaux arithmétiques θ.
12. Aucune transformation dans les autres cas :
$\mathsf{label}.\sigma \ \longrightarrow Sgn_0$

Règle de substitution du champ *codage* : $Cdg_0 \ \longrightarrow\ (\ Dom\ )$
La réutilisation d'un codage peut être faite chaque fois que la catégorie syntaxique $Dom = \{MesMes\}^*$ est utilisée, c'est à dire dans une propriété $Ppt = (Sel\ Dom)$. Nous aurons alors deux règles de substitution :

1. Suppression du premier niveau de parenthésage de l'expression de substitution dans $Ppt = (Sel\ Dom)$:
$(\ Sel\ \alpha\ \text{label}.\omega\ \beta\ ) \longrightarrow (\ Sel\ \alpha\ Cdg_0\ \beta\ ) \longrightarrow (\ Sel\ \alpha\ (Dom_0)\ \beta\ )$

Description Structurelle	(	Sel	α	(	Dom_0	)	β	)
	1	2	3	4	5	6	7	8
Changement Structurel	1	2	3	0	5	0	7	8
	(	Sel	α		Dom_0		β	)

2. Aucune transformation dans les autres cas :
$\mathsf{label}.\omega \ \longrightarrow Cdg_0$

2.5.1.2 Méta-règles de transformation

Nous allons aborder ici les règles syntaxiques permettant de gérer les substitutions non effectuées par les règles précédentes. À ces règles vont s'ajouter les règles de simplification d'expressions étudiées au chapitre 2 (3^{e} partie).

1. Méta-règle pour la substitution de ς par $Sgn = (\nu\ Sgn\ Sgn)$:
Comme précédemment, nous allons considérer cette substitution dans un terme générique $Sel = (\alpha\ \varsigma)$ où $\alpha = \{\eta^+\check{\varsigma} \mid \eta^*\}$. La méta-règle gérant cette substitution est liée aux règles de production $Sgn \longrightarrow (\alpha\ \varsigma)$ pour s'exprimer de la façon suivante :
$$\begin{aligned} Sgn &\overset{*}{\Longrightarrow} (\alpha\ (\nu\ \underbrace{Sgn}_{1}\ \underbrace{Sgn}_{2})) \\ &\longrightarrow (\nu\ (\alpha\ \underbrace{Sgn}_{1})\ (\alpha\ \underbrace{Sgn}_{2})) \end{aligned}$$
2. Méta-règle pour la substitution de Sel par $Sgn = (\nu\ Sgn\ Sgn)$ dans $Ppt = (Sel\ Dom)$:
Cette méta-règle liée à la production $Cpt \longrightarrow (Ppt^+\ Dst)$ sera voisine de la précédente et s'exprimera ainsi :
$$\begin{aligned} Cpt &\overset{*}{\Longrightarrow} (\alpha\ ((\nu\ \underbrace{Sgn}_{1}\ \underbrace{Sgn}_{2})\ Dom)\ \beta\ Dst) \\ &\longrightarrow (\nu\ (\alpha\ (\underbrace{Sgn}_{1}\ Dom)\beta\ Dst)(\alpha\ (\underbrace{Sgn}_{2}\ Dom)\ \beta\ Dst)) \end{aligned}$$

3. Méta-règle pour la substitution de Sel par $Sgn = Mes = (Sel\ Dst)$ dans $Ppt = (Sel\ Dom)$: La méta-règle gérant cette substitution est liée à la règle de production $Cpt \longrightarrow (Ppt^+\ Dst)$ et s'exprime de la façon suivante :

$$\begin{array}{rcl} Cpt & \stackrel{*}{\Longrightarrow} & (\ldots((Sel\ \underbrace{Dst}_{1})\ Dom)\ldots\ \underbrace{Dst}_{2}) \\ & \longrightarrow & (\ldots(Sel\ Dom)\ldots\ \underbrace{Dst}_{1=2})\ \text{si les } Dst \text{ sont identiques} \\ & & \text{sinon, transformation incorrecte.} \end{array}$$

2.5.2 Règles d'affixes

Les attributs positionnés sur certains éléments terminaux permettent de récupérer les informations d'ordre sémantique -pragmatiques- présentes dans les descriptions. Ces informations correspondent aux valeurs numériques manipulées dans les catégories syntaxiques <*mesure*> et aux éléments déstructurants utilisés dans la catégorie <*déstructuration_d'une_ base*> et l'index de la donnée décrite.
Deux attributs -*valeur* et *déstructurant*- seront alors utilisés pour recueillir ces informations. Celles-ci seront représentées sous forme de liste, renfermant les valeurs effectivement présentes, et attribuées aux éléments terminaux auxquels elles font référence.

2.5.2.1 Attribut des valeurs numériques

Les valeurs numériques sont rapportées par la catégorie syntaxique <*mesure*>. Nous ne considérons donc uniquement que les cas où nous avons la production $Mes_0 \longrightarrow v$ qui peut être utilisée dans les expressions suivantes :

- Sel
  Cette information se rencontre dans le champ *signal* lorsque la donnée décrite est justement une valeur numérique. Cette information ne peut donc être attribuée à aucun descripteur particulier.
- $Fct = (\theta\ MesMes)$
  Lorsqu'une fonction arithmétique est appliquée sur des mesures, la liste des valeurs manipulées est attribuée au descripteur θ :
  $$\theta._{valeur} \longleftarrow \theta._{valeur} \cup \{v\ /\ Mes_0 \rightarrow v\}$$
- $Dom = \{Mes\ Mes\}^*$
  Une domaine ne représente rien en tant quel tel, s'il n'est pas rattaché à un contexte particulier. Les mesures utilisées sont donc étudiées par dérivation des catégories manipulant les domaines.
- $Ppt = (Sel\ Dom) \stackrel{*}{\Longrightarrow} (Sel\ \{Mes\ Mes\}^*)\ \underbrace{\stackrel{*}{\Longrightarrow}}_{transf.}\ ((\ldots\varsigma)\ \{Mes\ Mes\}^*)$
  Après dérivation et transformation par les règles précédentes, la propriété manipule un ensemble de mesures faisant référence à un descripteur de sélection. La liste de ces mesures et les valeurs du domaine de validité propre à ce descripteur lui sont donc attribuées :
  $$\varsigma._{valeur} \longleftarrow \varsigma._{valeur} \cup \{v\ /\ Mes_0 \rightarrow v\} \cup \Delta_\varsigma$$
- $Cpt = (\phi\ Mes+)$
  Dans le cas d'un concept décrivant une figure, les informations sur les mesures utilisées ne sont pas prises en compte puisque, ici, toute la figure correspond à une information d'ordre pragmatique. La description complète est donc récupérée et conservée pour la résolution d'objectifs.

- $Trm = (\{MesMes\}^*)$
  Les mesures présentes dans la trame font référence aux positions minimales et maximales que peuvent prendre les coordonnées des points de la donnée. Ces coordonnées sont représentées par les sélecteurs *cxpi*, *cypi*, *czpi*, *ctpi*, présents dans cet ordre. Les valeurs rencontrées leur seront donc attribuées de la même façon que pour les sélecteurs de la propriété, c'est à dire en incluant le domaine de validité.
- $Cdg = (Dom)$
  Les mesures de codage se réfèrent quant à elles au sélecteur déterminant le signal. Dans ce cas aussi, la procédure de l'attribution aux sélecteurs désignés tiendra compte du domaine de validité.

2.5.2.2 Attribut des éléments déstructurants

Les éléments déstructurants sont rapportés par les catégories syntaxiques <*index*> et <*déstructuration_d'une_base*>. Nous ne considèrerons donc que les dérivations $Dst_0 \stackrel{*}{\Longrightarrow} ((\zeta)Bse_0)$ et $Idx_0 \stackrel{*}{\Longrightarrow} (\hat{\zeta}\zeta)$ dans les expressions suivantes. Dans tous les cas, la valeur de l'élément déstructurant ζ récupérée est attribuée au ou aux pointeurs $\hat{\zeta}$ dont il dépend ; ceci afin d'accéder à tous les éléments déstructurants relatifs au pointeur considéré. Pour certaines productions cet élément ζ sera aussi attribué à d'autres descripteurs :

- $Idx = (\hat{\zeta}\zeta)$ ou $Cpt = (Ppt^+ \ Dst)$
  Dans ces deux cas, l'élément déstructurant rapporté est attribué simplement au pointeur dont il dépend. Ceci peut être représenté par :
  $$\hat{\zeta}._{déstructurant} \longleftarrow \hat{\zeta}._{déstructurant} \cup \{\zeta\} \text{ avec } \zeta \in \hat{\zeta}$$
- $Mes = (Sel \ Dst)$
  Cette fois, en plus de la règle d'attribution précédente, nous attribuerons l'élément déstructurant au descripteur de sélecteur ς rapporté par la dérivation de Sel :
  $$\varsigma._{déstructurant} \longleftarrow \varsigma._{déstructurant} \cup \{\zeta\}$$
- $Cpt = (\rho \ Dst \ Bse)$
  Enfin, dans le cas d'une relation, le descripteur ρ prendra pour attribut, l'élément ζ rapporté par Dst ; la règle d'attribution du premier point s'appliquant toujours :
  $$\rho._{déstructurant} \longleftarrow \rho._{déstructurant} \cup \{\zeta\}$$

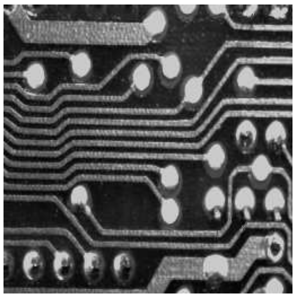
2.6 Remarque sur le descripteur `xou`

2.6.1 Données multi-signaux

Nous allons faire une étude toute particulière pour le descripteur *ou exclusif* `xou`. Dans le langage de description de J.M. Inglebert, des traitements, comme la dilatation d'une image multi-niveaux engendrant des données possédant une double caractéristique sémantique quant à la représentation des valeurs des pixels (*zones homogènes*, *zones non-homogènes*), fournissent des descriptions du résultat dans lesquelles se trouvent plusieurs "concepts" témoignant des différentes mesures effectuées.

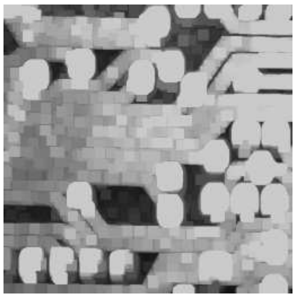
Considérons l'exemple suivant (fig.2.14) montrant une donnée D_2 issue de la dilatation multi-niveau (5 itérations) d'une image D_1 de niveau de gris.
La description fait clairement apparaître la mesure de détermination du `max` sur l'élément structurant (`ordre < 6`) de l'opérateur de dilatation. Cette caractéristique se devait donc d'être reportée dans le nouveau langage, en mettant en valeur la dissociation entre les *zones homogènes* affectées par la mesure et les *zones non-homogènes* dont le signal est inchangé. Cette possibilité est rendue

D_1



Description :
`(((nivg)))`

D_2



Description :

```
( ((max) (((ordre < 6) (((nivg)))))
  ((nivg) (((ordre >= 6) (((nivg)))))) )
```

FIG. 2.14 - *Dilatation multi-niveau (5 itérations) - Ancien langage de description*

dans LEO par la double utilisation du descripteur `xou` qui va effectuer une partition du cache en deux composantes de distance spatiale nulle :

```
( (xou  (((ordre 1 6) ((vois4) (((nivg))))))
        (((ordre 6 256) ((vois8) (((nivg))))))) )
```

... puis une disjonction exclusive sur les signaux de chacune des composantes du cache :

```
( xou  (max nivg)
       nivg )
```

La notion de distance spatiale entre concepts a été introduite par J.M. Inglebert qui en donne la définition suivante :
« *La Distance Spatiale d'une Disjonction de concepts est la distance minimale séparant deux instances quelconques appartenant à deux concepts différents. Cette distance est exprimée en pixels et est mesurée suivant les axes horizontal et vertical* ».
Cette notion est rappelée dans le chapitre 2 (3e partie) sur la simplification d'expression.

2.6.2 Exploitation pour l'introduction de nouveaux opérateurs

La partition de l'image faite par le cache et son complémentaire se trouve maintenant renforcée par l'expression d'entité exclusivement disjointe. Nous pouvons ainsi créer une partition du cache ainsi qu'une partition de son complémentaire, tout en leur associant, comme nous l'avons vu dans l'étude de LEO, des signaux distincts. Cette possibilité va nous permettre d'exprimer de nombreux opérateurs difficiles, voire impossible à décrire jusqu'à présent, ou de justifier l'introduction de mesures telles que *prewitt* ou *sobel*.

Dans l'exemple suivant (fig.2.15,2.16,2.17), nous avons exprimé le résultat issu de la différence de deux images, issues d'un décalage à droite et d'un décalage à gauche de 50 pixels de la même image source. Nous avons utilisé pour cela deux de ces opérateurs : l'opérateur de translation linéaire selon les axes vertical et horizontal et l'opérateur de différence de deux images multi-niveaux. La première figure (fig.2.15) montre les images source nommée S, décalée à droite DD et décalée à gauche DG. Rappelons que les expressions de la trame et du cache en terme de coordonnées sont exprimées dans le repère du référentiel pour la trame et le repère de la donnée pour le cache. Les expressions respectives de ces données sont alors :

$$S = \left\{ \begin{array}{l} \tau = ((0\ 256)\ (0\ 256)) \\ \delta = t \\ \iota = (\mathcal{G}l) \\ \sigma = nivg \\ \omega = (0\ 256) \end{array} \right\} \quad DD = \left\{ \begin{array}{l} \tau = ((-50\ 206)\ (0\ 256)) \\ \delta = (((cxpi\ 50\ 256))) \\ \iota = (\mathcal{G}l) \\ \sigma = nivg \\ \omega = (0\ 256) \end{array} \right\} \quad DG = \left\{ \begin{array}{l} \tau = ((50\ 306)\ (0\ 256)) \\ \delta = (((cxpi\ 0\ 206))) \\ \iota = (\mathcal{G}l) \\ \sigma = nivg \\ \omega = (0\ 256) \end{array} \right\}$$

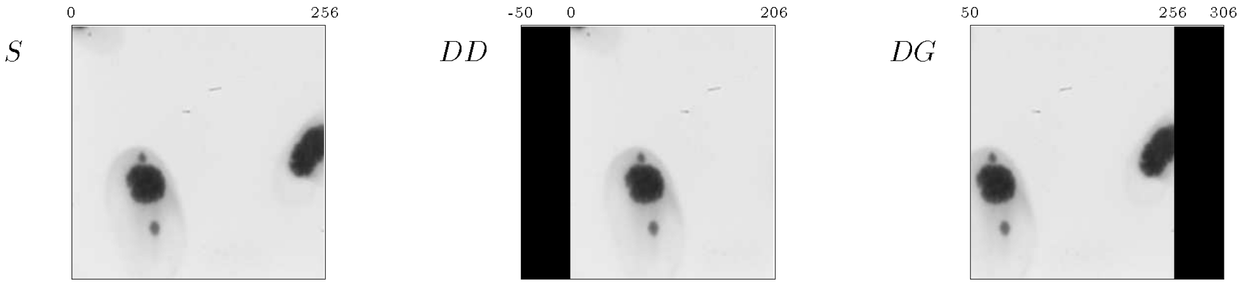
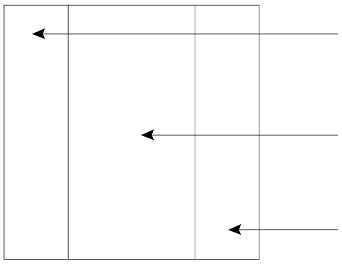


FIG. 2.15 - *Image source et décalages droite et gauche*

La seconde figure (fig.2.16) montre les trois parties qui vont se retrouver dans l'image résultat R de cette différence, ainsi que les signaux qui leur sont associés. Il faut noter que, comme nous effectuons la différence de DD par DG, la trame et le cache de R seront exprimés en fonction de DD. Ainsi l'expression du cache de DG doit être recadré sur DD.



Partie de $R_\kappa :\sim DD_\kappa = ((cxpi\ 0\ 50))$
signal : $(opp\ (DG_\sigma\ ((gl)\ DG_\kappa)))$
$= ((opp\ nivg)((gl)(((cxpi\ 0\ 206))$

Partie de $R_\kappa : (et\ (DD_\kappa)\ (DG_\kappa)) = ((cxpi\ 50\ 206))$
signal : $(diff\ (DD_\sigma\ ((gl)\ DD_\kappa))(DG_\sigma\ ((gl)\ DG_\kappa)))$
$= (diff\ (nivg\ ((gl)(((cxpi\ 50\ 256)))))\ (nivg\ ((gl)(((cxpi\ 0\ 206))))))$

Partie de $R_\kappa : (\sim DG_\kappa) = ((cxpi\ 206\ 256))$
signal : $(DD_\sigma\ ((gl)\ DD_\kappa))$
$= (nivg((gl)(((cxpi\ 50\ 256)))))$

FIG. 2.16 - *Partition de la différence*

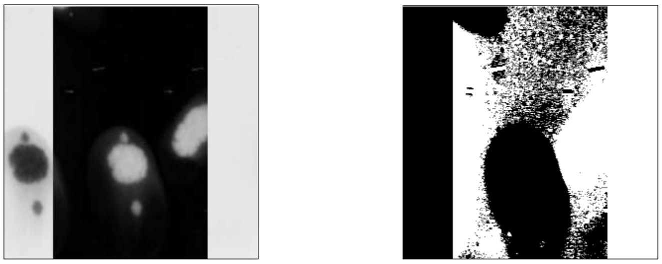
Le résultat $R = DD - DG$ (fig.2.17) est donc une donnée que nous représentons par deux images, une image des valeurs absolues de la différence et une image de signe. La donnée R s'exprime alors de la façon suivante :

$$R = \left\{ \begin{array}{l} \tau = ((-50\ 206)\ (0\ 256)\ (0\ 2)) \\ \delta = ((xou\ (((cxpi\ 50\ 206)))\ ((xou\ (((cxpi\ 0\ 50)))\ (((cxpi\ 206\ 256)))))) \\ \iota = (\mathcal{G}l) \\ \sigma = (xou\ (diff\ (nivg((gl)(((cxpi\ 50\ 256)))))\ (nivg((gl)(((cxpi\ 0\ 206))))))) \\ \quad (xou\ ((opp\ nivg)((gl)(((cxpi\ 0\ 206)))))\ (nivg((gl)(((cxpi\ 50\ 256))))))) \\ \omega = (-256\ 256) \end{array} \right\}$$

2.6.3 Vers une notion de macro-descripteur

Si nous effectuons maintenant une opération de différence sur les résultats de ces décalages de la donnée précédente, nous obtenons la description de la donnée *C-101* suivante qui n'est autre que le résultat de la dérivée en x de l'image :

R

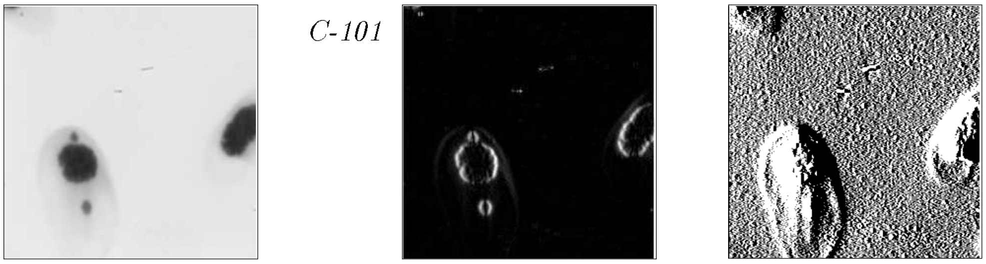


$|\, R \,|$ $\qquad$ $Sgn(R)$

FIG. 2.17 - *Images de la donnée résultat*

$$C\text{-}101 = \left\{ \begin{array}{l} \tau = ((-1\ 255)\ (0\ 256)\ (0\ 2)) \\ \delta = ((xou\ (((cxpi\ 1\ 255)))\ ((xou\ (((cxpi\ 0\ 1)))\ (((cxpi\ 255\ 256))))))) \\ \iota = (\mathcal{G}l) \\ \sigma = (xou\ (diff\ (nivg((gl)(((cxpi\ 1\ 256)))))\ (nivg((gl)(((cxpi\ 0\ 255))))))) \\ \quad (xou\ ((opp\ nivg)((gl)(((cxpi\ 0\ 255)))))\ (nivg((gl)(((cxpi\ 1\ 256))))))) \\ \omega = (-256\ 256) \end{array} \right\}$$

S $\qquad$ *C-101*



$|C\text{-}101|$ $\qquad$ $Sgn(C\text{-}101)$

FIG. 2.18 - *Images de la dérivée C-101*

Ce fait permet tout d'abord de pouvoir décrire les opérations pouvant être traitées par des masques de convolutions qui manquaient au système. Mais surtout, il permet de justifier l'introduction de mesures -dans le sens de la catégorie syntaxique définie dans LEO- comme *sobel* ou *prewitt* qui peuvent ainsi être considérés maintenant comme des macro-descripteurs.

Pour illustrer ceci, nous pouvons montrer comment exprimer la description du résultat de l'application d'un masque de convolution $\mathcal{M}_{101\times101}$ sur l'image source S (fig.2.19,2.20). Ce masque ne correspond à rien en termes de traitement d'images mais possède les mêmes caractéristiques que celui de *Prewitt* en x ; il en est une grossière exagération.

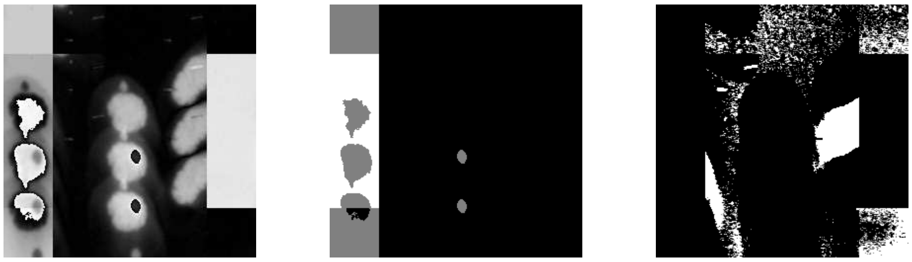
$$Pwx = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

Pwx

$|Pwx|_{256}$ $\qquad$ $|Pwx|\ /256$ $\qquad$ $Sgn(Pwx)$

FIG. 2.19 - *Application de Pwx sur l'image source*

$$\mathcal{M}_{101\times101} = \begin{pmatrix} 1 & 0 & \cdots & 0 & -1 \\ 0 & 0 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 0 & 0 \\ 1 & 0 & \cdots & 0 & -1 \\ 0 & 0 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 0 & 0 \\ 1 & 0 & \cdots & 0 & -1 \end{pmatrix}$$



$\mid P50 \mid_{256}$ $\mid P50 \mid /256$ $Sgn(P50)$

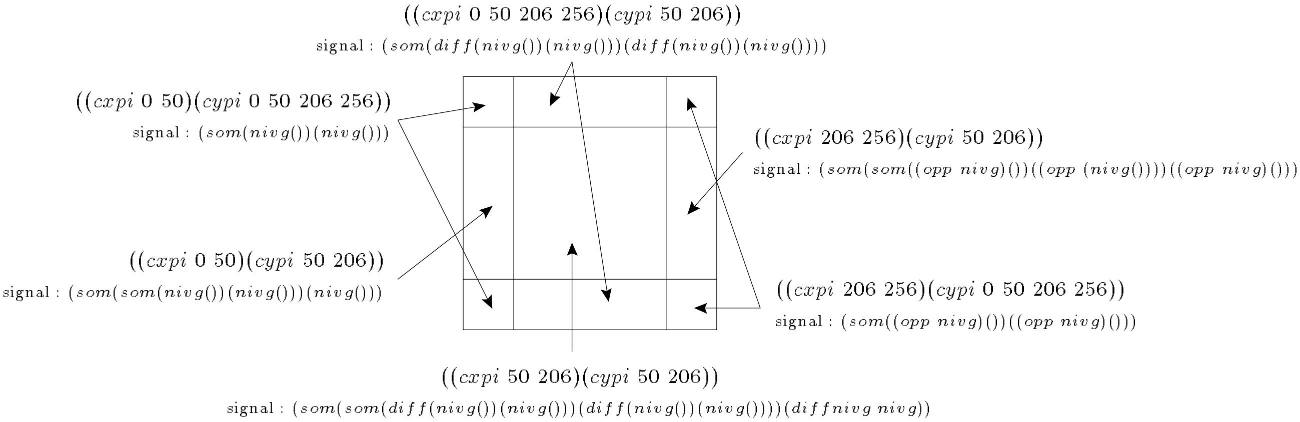


FIG. 2.20 - *Description de l'application de $\mathcal{M}_{101\times101}$ à l'image source*

Chapitre 3

Un modèle pour les opérateurs

Les opérateurs de traitement d'images sont décris en termes de transformations d'informations réalisées et non en termes de conditions d'utilisation et/ou de résultats obtenus. Une transformation est représentée par une fonction de modification d'expressions décrivant un élément de la donnée, utilisant les notions de *structuration, déstructuration, spécialisation, généralisation* et *nomination* d'information. Le modèle d'opérateur symbolique décrivant la transformation d'informations de l'opérateur réel de traitement d'images doit maintenant tenir compte de la séparation entre le support de l'information et l'information supportée. Il est donc constitué d'une double structure complexe concernant la nature de la transformation de la trame -fonctions de l'ensemble τ^t-, des caches -fonctions de l'ensemble κ^t-, des index -fonctions de l'ensemble ι^t-, des signaux -fonctions de l'ensemble σ^t-, des codages -fonctions de l'ensemble ω^t- constituant la donnée et son complémentaire.
Nous pourrons ainsi représenter ces opérateurs de la façon suivante (opérateur de seuillage du système KHOROS [Uni92] - voir Annexe E) : vthresh -i D -l m -u (M-1) -o R) :

$$Sll(D,m,M) = \left(\begin{bmatrix} \tau^t = \mathcal{I}d^{\tau} \\ \kappa^t = \kappa^-_{\{D.\sigma \in [m,M[\}} \\ \iota^t = \Gamma^- \\ \sigma^t = \Sigma_{vab} \\ \omega^t = \omega_{vab} \end{bmatrix} , \begin{bmatrix} \kappa^t = \kappa^+_{\{D.\sigma \in (\Delta_{D.\sigma} \setminus [m,M[)\}} \\ \iota^t = 0 \\ \sigma^t = \Sigma_{vab} \\ \omega^t = \Delta_{vab} \setminus \omega_{vab} \end{bmatrix} \right)$$

3.1 Transformation de la trame

3.1.1 Introduction sur τ^t

Parce qu'une trame possède deux caractéristiques, *Trame=(Dimension, Taille)*, une opération de transformation de trame est constituée d'un couple de fonctions de transformation de $F^{\eta} \times F^{\mathcal{T}}$.

$\underline{F^\eta :}$ Nous désignerons par F^η l'ensemble des fonctions de transformation de dimension de trame. Celles-ci seront de la forme $f^{i,j}$ avec i désignant la dimension de la trame d'application et j celle de la trame image. Nous verrons qu'il existe deux types de transformations de dimension de trame selon qu'elles effectuent des structurations (déstructurations) ou des spécialisations (généralisations). Nous noterons $\mathcal{I}d^\eta$ la famille de fonctions $f^{k,k}$ avec k désignant une dimension de trame quelconque.

$$F^\eta = \left\{ f^{i,j} \left| \begin{array}{l} T^i \longrightarrow T^j \\ \tau^i \longmapsto \tau^j \end{array} \right. \Big/ \ (i,j) \in \mathbb{N}^2 \right\} \tag{3.1}$$

$\underline{F^\mathcal{T} :}$ Nous désignerons par $F^\mathcal{T}$ l'ensemble des fonctions de transformation de taille de trame. Nous pourrons ainsi distinguer pour toute donnée de dimension n :

1. $\mathcal{I}d^\mathcal{T}$: la transformation identique ou identité.
2. Φ_d : l'ensemble des fonctions à tailles déterminées :

$$\Phi_d = \left\{ f^\mathcal{T} \Big/ \ \mathcal{T} = ((I_1,S_1),\cdots,(I_n,S_n)) \right\} \tag{3.2}$$

3. Φ_c : l'ensemble des fonctions à tailles calculées :

$$\Phi_c = \left\{ f^\chi \Big/ \ \chi = ((f_{I_1},f_{S_1}),\cdots,(f_{I_n},f_{S_n})) \right\} \tag{3.3}$$

Ainsi l'ensemble des fonctions de transformation de taille de trames s'exprimera de la façon suivante :

$$F^\mathcal{T} = \left\{ \mathcal{I}d^\mathcal{T} \right\} \cup \Phi_d \cup \Phi_c \tag{3.4}$$

3.1.2 Transformation identique

$\underline{\tau^t = \mathcal{I}d^t :}$ Comme son nom l'indique, cette opération ne modifie aucune des deux caractéristiques de la trame τ sur laquelle elle s'applique (fig.3.1). $\tau^t = \mathcal{I}d^t$ est alors constituée des couples définis par :

$$\mathcal{I}d^t = \left(\mathcal{I}d^\eta, Id^\mathcal{T} \right) \tag{3.5}$$

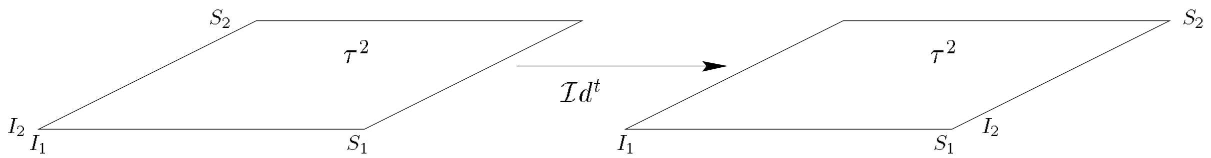


FIG. 3.1 - *Transformation* $\tau^t = \mathcal{I}d^t$

3.1.3 Transformations par réduction de trame

Il existe deux types de transformations pour réduire la trame selon qu'elles affectent ou non de façon globale la nature de la donnée ; cette notion, nous le verrons plus tard (voir §3.4), est fortement liée à la notion de déstructuration du signal de la donnée. Nous trouverons donc des transformations effectuant des spécialisations ou des structurations de la trame.

3.1.3.1 Spécialisation par diminution de la taille de la trame

$\tau^t = \mathcal{T}^- \in \mathcal{S}p^t$: Cette transformation ne tient compte que de l'élément taille ; la dimension de la trame étant alors inchangée (la fonction de F^η considérée est donc $\mathcal{I}d^\eta$). Seule la fonction de variation de la taille de la trame est considérée (fig.3.2,3.3). Cette fonction appartient donc à $\Phi_d \cup \Phi_c$.

$$\mathcal{T}^- = \left\{ \ (\mathcal{I}d^\eta, f) \Big/ \ f \in \Phi_d \cup \Phi_c \ \right\} \tag{3.6}$$

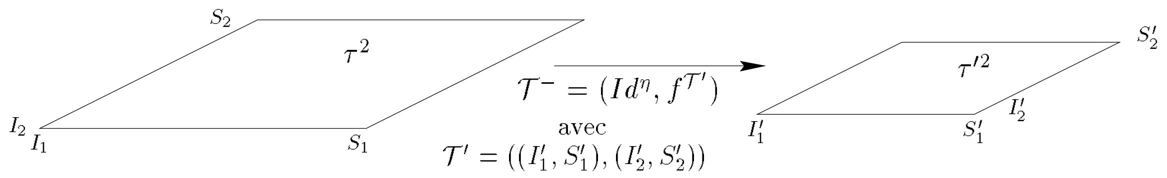


FIG. 3.2 - *Transformation* $\mathcal{S}p^t = \mathcal{T}^-$ *avec* $f^{\mathcal{T}} \in \Phi_d$

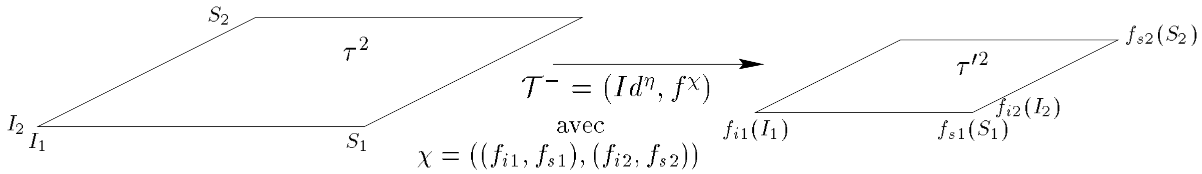


FIG. 3.3 - *Transformation* $\mathcal{S}p^t = \mathcal{T}^-$ *avec* $f^\chi \in \Phi_c$

3.1.3.2 Spécialisation par diminution de la dimension de la trame

$\tau^t = \eta_G^- \in \mathcal{S}p^t$: La dimension de la trame déterminée par spécialisation est atteinte en sélectionnant, sur la trame, une entité géométrique définissable (fig.3.4). Cette transformation se traduira par un couple de fonctions $(f^{i,j}, f^G)$ avec $j < i$ désignant en fait la dimension de l'entité géométrique décrite par l'équation affine G exprimée dans le repère de la trame.

$$\eta_G^- = \left\{ \ (f^{i,j}, f^G) \Big/ \ j < i, G \ affine \ \right\} \tag{3.7}$$

Ainsi, si $\tau = \left(\ i, \mathcal{T} \ \right)$ alors $\eta_G^-(\tau) = \left(\ f^{i,j<i}, f^G \ \right)(\tau) = \tau' = \left(\ j, \mathcal{T}' \ \right)$; $\mathcal{T}'$ étant calculée par normalisation de l'application de G aux points de la trame.

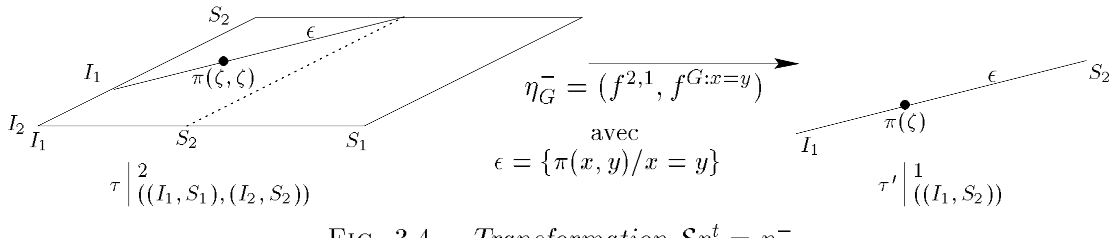


FIG. 3.4 - *Transformation* $\mathcal{S}p^t = \eta_G^-$

3.1.3.3 Structuration de la trame

$\tau^t = \eta_{\mathcal{T}_\chi}^- = \mathcal{S}t^t$: La nouvelle dimension déterminée par structuration des éléments de la trame est liée à une caractéristique de mesure χ (fig.3.5). Comme précédemment, la fonction de F^η est $f^{i,j}$

avec $j < i$. La fonction de transformation de la taille, dépendant de l'application de la mesure sur la donnée, est quant à elle un élément de Φ_d.

$$\mathcal{S}t^t = \eta^-_{\mathcal{T}_\chi} = \left\{ \ (f^{i,j}, f^{\mathcal{T}_\chi}) \Big/ \ j < i, \mathcal{T}_\chi = ((I_1, S_1), \cdots, (I_j, S_j)) \quad \right\} \tag{3.8}$$

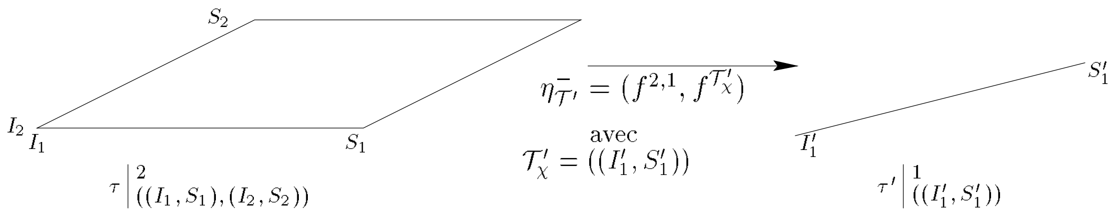


FIG. 3.5 - *Transformation* $\mathcal{S}t^t = \eta^-_{\mathcal{T}_\chi}$

3.1.4 Les transformations par augmentation de la trame

Toutes les fonctions développées précédemment peuvent s'inverser pour déterminer les fonctions de généralisation d'une trame.

3.1.4.1 Généralisation par augmentation de la taille de la trame

$\underline{\tau^t = \mathcal{T}^+ \in \mathcal{G}n^t:}$ Tout comme pour la transformation $\mathcal{T}^-$ la dimension de la trame est inchangée ; seule la fonction de variation de la taille de la trame est efficiente et appartient à $\Phi_d \cup \Phi_c$.

$$\mathcal{T}^+ = \left\{ \ (\mathcal{I}d^\eta, f) \Big/ \ f \in \Phi_d \cup \Phi_c \quad \right\} \tag{3.9}$$

3.1.4.2 Généralisation par augmentation de la dimension de la trame

$\underline{\tau^t = \eta^+_G \in \mathcal{G}n^t:}$ Comme précédemment, la dimension de la trame déterminée par généralisation est atteinte en considérant la trame comme une entité géométrique définissable dans la trame image (nouvelle trame). Cette transformation se traduit par un couple de fonctions $(f^{i,j}, f^G)$ avec $j > i$ désignant en fait la dimension de l'entité géométrique décrite par l'équation affine G^{-1} exprimée dans le repère de la trame transformée.

$$\eta^+_G = \left\{ \ (f^{i,j}, f^G) \Big/ \ j > i, G^{-1} \ affine \quad \right\} \tag{3.10}$$

Ainsi, si $\tau = \Big(\ i, \mathcal{T} \ \Big)$ alors $\eta^+_G(\tau) = \Big(\ f^{i,j>i}, f^G \ \Big)(\tau) = \tau' = \Big(\ j, \mathcal{T}' \ \Big)$; $\mathcal{T}'$ étant calculée par normalisation de l'application de G aux points de la trame.

3.1.4.3 Déstructuration de la trame

$\underline{\tau^t = \eta^-_{\mathcal{T}_\chi} = \mathcal{D}s^t:}$ La nouvelle dimension est déterminée par déstructuration des éléments de la trame selon une caractéristique donnée. Comme précédemment, la fonction de F^η est $f^{i,j}$ avec $j > i$. La fonction de transformation de la taille, dépendant de l'application de la caractéristique sur la donnée est quant à elle un élément de Φ_d.

$$Ds^t = \eta^+_{\mathcal{T}_\chi} = \left\{ \ (f^{i,j}, f^{\mathcal{T}_\chi}) \Big/ \ j > i, \mathcal{T} = ((I_1, S_1), \cdots, (I_j, S_j)) \quad \right\} \tag{3.11}$$

3.2 Transformation du cache

3.2.1 Introduction sur κ^t

Un cache est étroitement lié à la trame sur laquelle il est défini, et en particulier à la dimension de celle-ci. Ainsi le premier type de transformation considère le fait que toute modification de la dimension de la trame ($\tau^t = \eta^{\pm}$) entraîne obligatoirement une transformation du cache par changement de sa dimension. Le second type de transformation du cache est alors lié à la non transformation de la dimension de la trame ($\tau^t \in \mathcal{I}d^t \cup \mathcal{T}^{\pm}$).

3.2.2 Transformations avec variation de la dimension

Les variations de la dimension de la trame considérées ici correspondent à des transformations de spécialisation ou de généralisation par sélection ou expansion d'entités géométriques G ou des transformations de structuration ou déstructuration. Elles peuvent être regroupées dans une partition de τ^t définie par :

$$\eta^{\pm} = \eta^{\pm}_{G} \cup \eta^{\pm}_{\mathcal{T}_X} \tag{3.12}$$

3.2.2.1 Transformation pseudo-identique

$\underline{\kappa^t = \mathcal{I}d^{\kappa}}$: Dans le cas d'une transformation de la trame par sélection (expansion) d'une entité ε d'équation G (rappel : $\tau^t = \eta^{\pm}_{G}$), la transformation du cache peut prendre la forme d'une transformation identique (fig.3.6) basée sur l'identité de la description de la fonction cache associée.
Restreinte à ε, (respectivement basée sur ε) elle conserve ainsi les informations présentes dans la description du cache avant transformation, à laquelle elle ajoute l'information structurante (respectivement déstructurante) relative à ε. Il s'agit donc d'une forme dégénérée de transformation identique dans le sens où le cache de la donnée d'entrée est réduit (étendu) selon une propriété décrivant ε.

$$\mathcal{I}d^{\kappa} \left| \begin{array}{l} K_{\tau} \longrightarrow K_{\tau'=\eta^{\pm}_{G}(\tau)} \\ \\ \kappa \left| \begin{array}{l} \tau \\ f_{\delta} \end{array} \right. \longmapsto \kappa' \left| \begin{array}{l} \tau' \\ f_{\delta'=\mathcal{S}p(\varepsilon,\delta)} \end{array} \right. \end{array} \right. \tag{3.13}$$

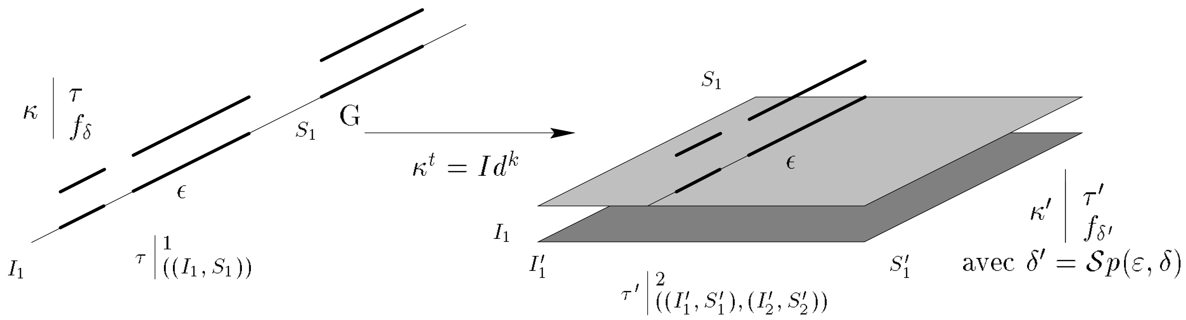


FIG. 3.6 - *Transformations identique de cache liée à* $\eta^{\pm}_{G}$

3.2.2.2 Transformation de globalité

$\underline{\kappa^t = 1^t}$: Lorsqu'il y a structuration ou déstructuration des éléments d'une trame (rappel : $\tau^t = \eta^{\pm}_{\mathcal{T}_X}$), l'unique opération applicable au cache est la transformation de globalité (fig.3.7). Celle-ci crée un cache global sur la trame transformée défini par tous les points de celle-ci. La description δ associée au cache ainsi formé est notée δ_o ou True ce qui induit l'écriture de la fonction cache associée :

$$f_{\delta_o} = f_{\mathsf{True}} \left| \begin{array}{l} \tau \longrightarrow \{\mathsf{True}, \mathsf{False}\} \\ \pi \longmapsto \mathsf{True} \end{array} \right. \tag{3.14}$$

1^t est alors définie par la fonction de l'ensemble K_τ dans $K_{\tau'}$, avec τ' image de τ par $\eta^{\pm}_{\mathcal{T}_\chi}$ qui à tout cache κ de K_τ fait correspondre le cache κ' de $K_{\tau'}$ ayant f_{δ_o} pour fonction cache.

$$1^t \left| \begin{array}{l} K_\tau \longrightarrow K_{\tau'=\eta^{\pm}_{\mathcal{T}_\chi}(\tau)} \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \kappa' \left| \begin{array}{l} \tau' \\ f_{\delta_o} \end{array} \right. = \{\pi' \in \tau'\} \end{array} \right. \tag{3.15}$$

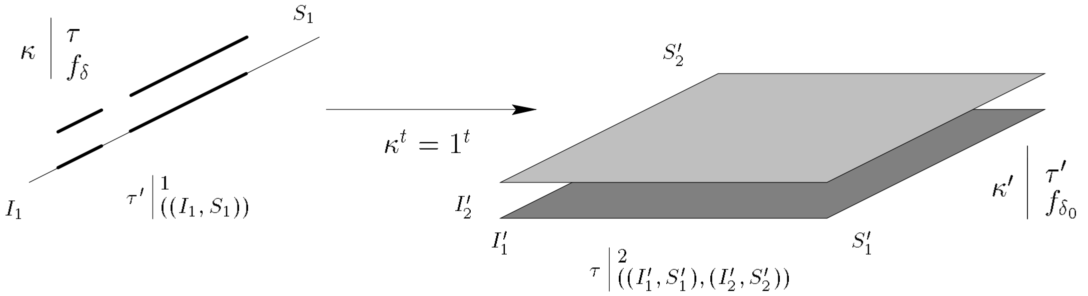


FIG. 3.7 - *Transformation $\kappa^t = 1^t$*

3.2.3 Transformation sans variation de la dimension

Les transformations du cache qu'il nous reste maintenant à examiner sont liées aux transformations de la trame du type $\tau^t = (\mathcal{I}d^\eta, \cdots)$; c'est à dire à des transformations ne faisant varier uniquement que le cache en considérant le fait que celui-ci n'est que la restriction à sa trame associée τ de l'application de sa fonction cache f_δ aux points du support de τ. Comme il est montré dans le schéma ci-dessous (fig.3.8), et en reprenant leur définition, nous pouvons constater que deux caches peuvent être identiques sans pour cela que leurs descriptions δ et δ' le soient. Pour faciliter

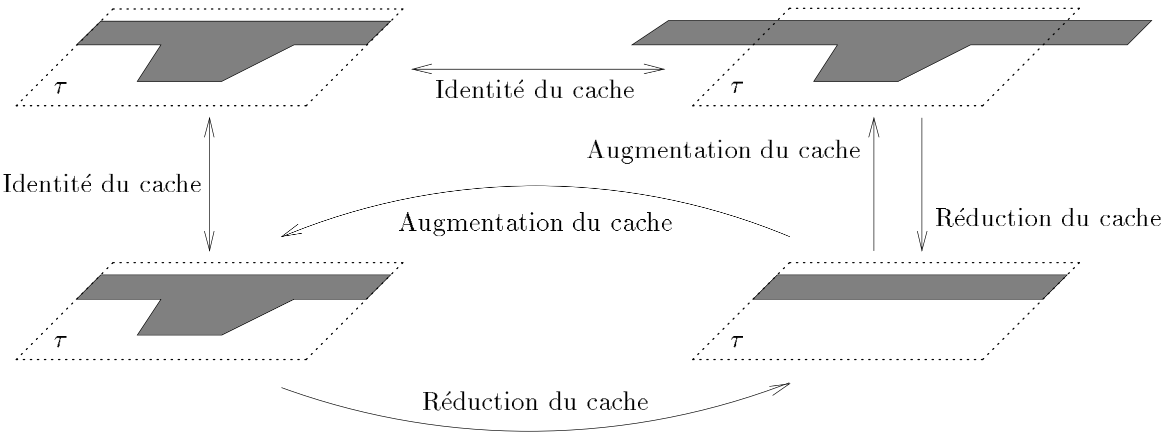


FIG. 3.8 - *Transformations du cache sur une trame*

l'écriture de telles transformations, nous devons en expliciter les différentes formes en considérant

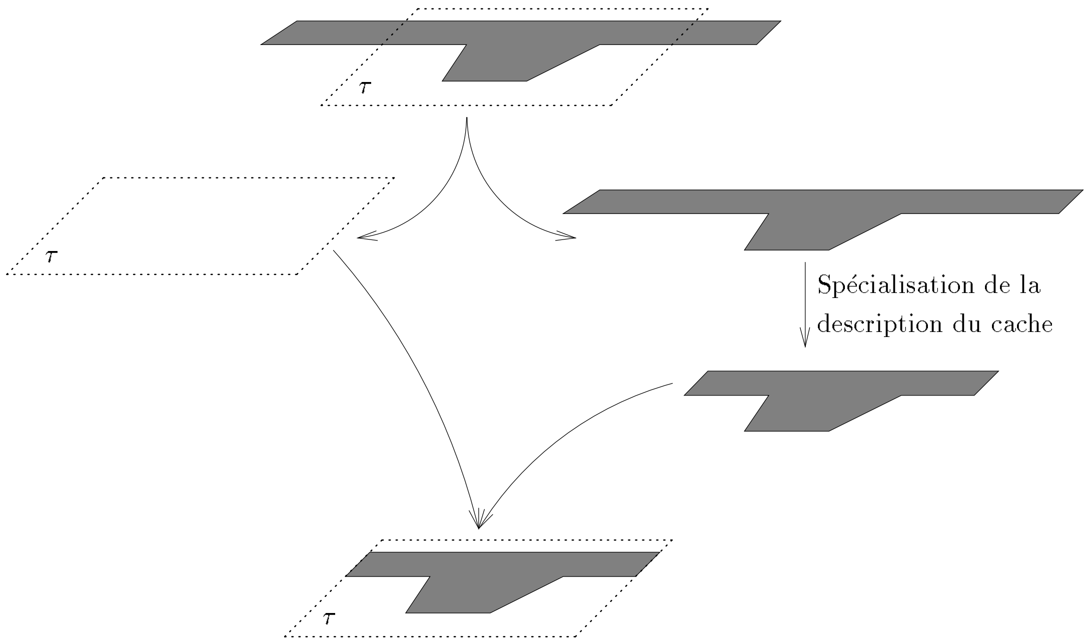


FIG. 3.9 - *Séparation du cache et de la trame*

tout d'abord les caches indépendamment de leur trame (fig.3.9). Ensuite, seulement, nous rapporterons la transformation au niveau de la trame en construisant le cache. De cette façon, nous avons réduit notre étude en étendant artificiellement la notion de trame à la totalité de son support et écarté les problèmes supplémentaires liés aux transformations de réduction ou d'augmentation de la taille de la trame ($\tau^t = \mathcal{T}^{\pm}$) (fig.3.10).

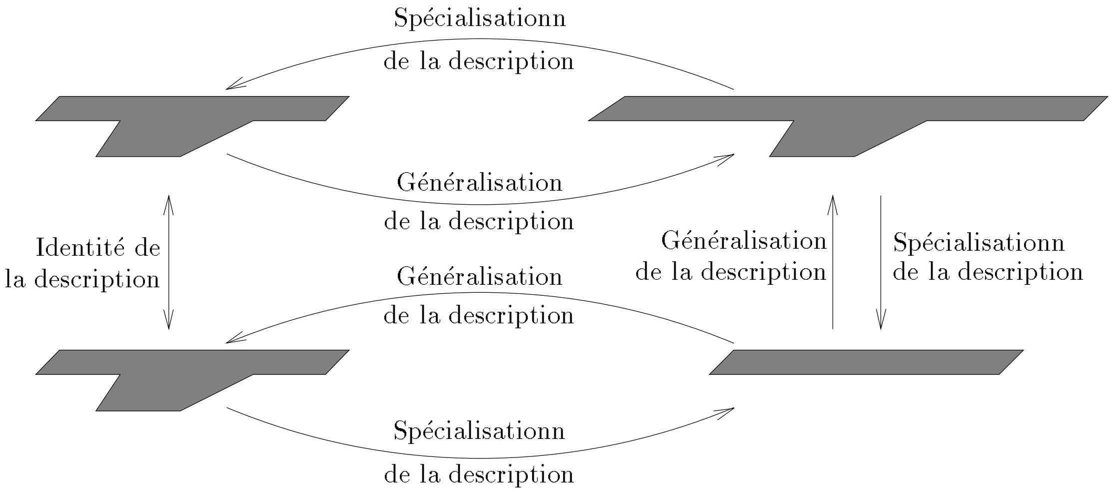


FIG. 3.10 - *Transformations de la description du cache*

3.2.3.1 Transformation identique

$\underline{\kappa^t = \mathcal{I}d^{\delta}}$: Cet ensemble de transformations identiques (fig.3.11) par invariant de la trame, correspond à l'invariance de la fonction cache à description δ. Ceci peut se traduire par $\mathcal{I}d^{\nabla}$ désignant l'application identique sur ∇_τ -ensemble des descriptions valides sur τ- définie par :

$$\mathcal{I}d^{\nabla} \left| \begin{array}{l} \nabla_\tau \longrightarrow \nabla_\tau \\ \delta \longmapsto \delta \end{array} \right. \tag{3.16}$$

Ainsi, la fonction traduisant cette transformation identique s'écrit :

$$\mathcal{I}d^{\delta} \left| \begin{array}{l} K_\tau \longrightarrow K_\tau \\ \kappa \left| \begin{array}{l} \tau \\ f'_\delta \end{array} \right. \longmapsto \kappa \left| \begin{array}{l} \tau \\ f_{\delta' = \mathcal{I}d^{\nabla}(\delta)} \end{array} \right. \end{array} \right. \tag{3.17}$$

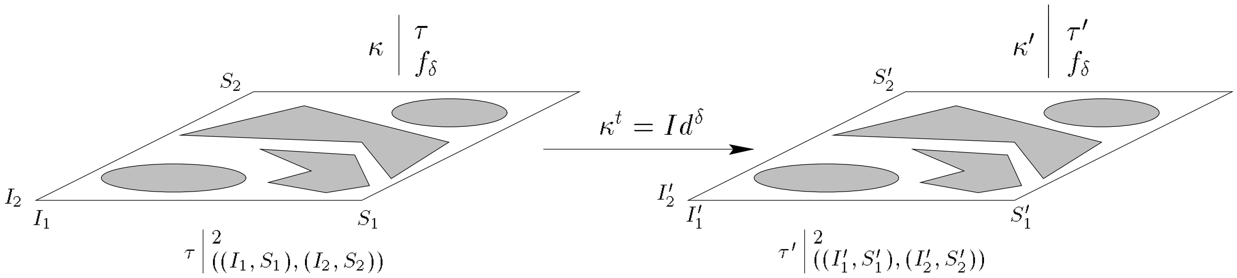


FIG. 3.11 - *Transformation identique $\mathcal{I}d^\delta$*

3.2.3.2 Deuxième forme de transformation identique

$\underline{\kappa^t = \mathcal{I}d^\chi}$: Cette deuxième transformation identique (fig.3.12) correspond à une partition du cache sur lequel elle s'applique. Cette partition est déterminée par une caractéristique χ rassemblant deux ou plusieurs propriétés de partition $\chi_1, \chi_2, \ldots, \chi_n$.
Ceci peut se noter de la façon suivante :

$$\mathcal{I}d^{\chi} \left| \begin{array}{l} K_\tau \longrightarrow K_\tau \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \tau \\ f_{\delta' = \mathcal{S}p(\chi_1,\delta) \cup \mathcal{S}p(\chi_2,\delta) \cup \ldots \cup \mathcal{S}p(\chi_n,\delta)} \end{array} \right. \tag{3.18}$$

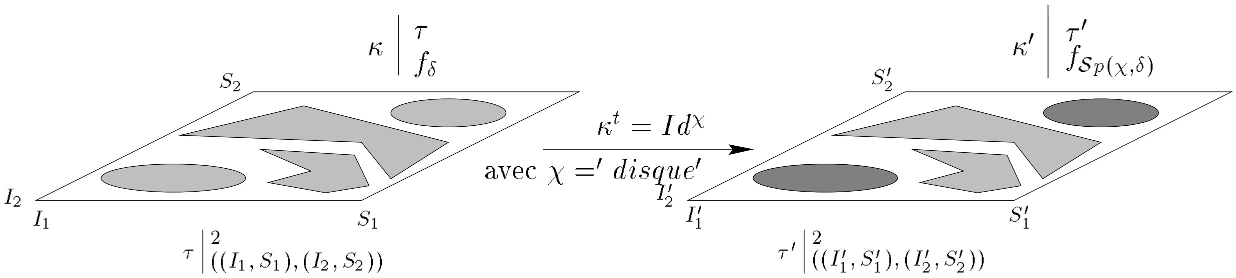


FIG. 3.12 - *Transformation identique $\mathcal{I}d^\chi$*

3.2.3.3 Spécialisation de la description de la fonction cache

$\underline{\kappa^t = \kappa^-_\chi}$: La transformation qui consiste à réduire la taille du cache se traduit par la spécialisation de la description δ de la fonction cache f_δ qui lui est associée. Cette spécialisation s'effectue par l'utilisation d'une caractéristique χ ajoutée à la description. Elle peut correspondre à une propriété

p (fig.3.13) permettant de préciser la description courante, ou bien une relation (fig.3.14) entre deux données vues au travers de leur cache. Une telle transformation du cache peut s'écrire :

$$\kappa_\chi^- \left| \begin{array}{l} K_\tau \longrightarrow K_\tau \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \kappa' \left| \begin{array}{l} \tau \\ f_{\delta'=\mathcal{S}p(\chi,\delta)} \end{array} \right. \end{array} \right. \tag{3.19}$$

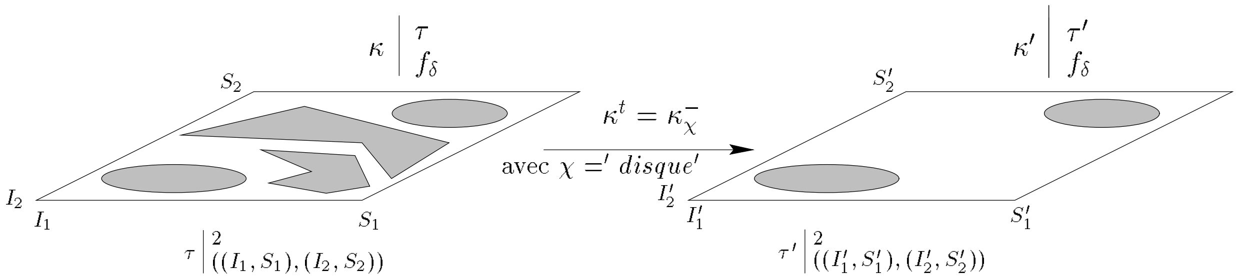


FIG. 3.13 - *Réduction du cache par ajout de la propriété "disque"*

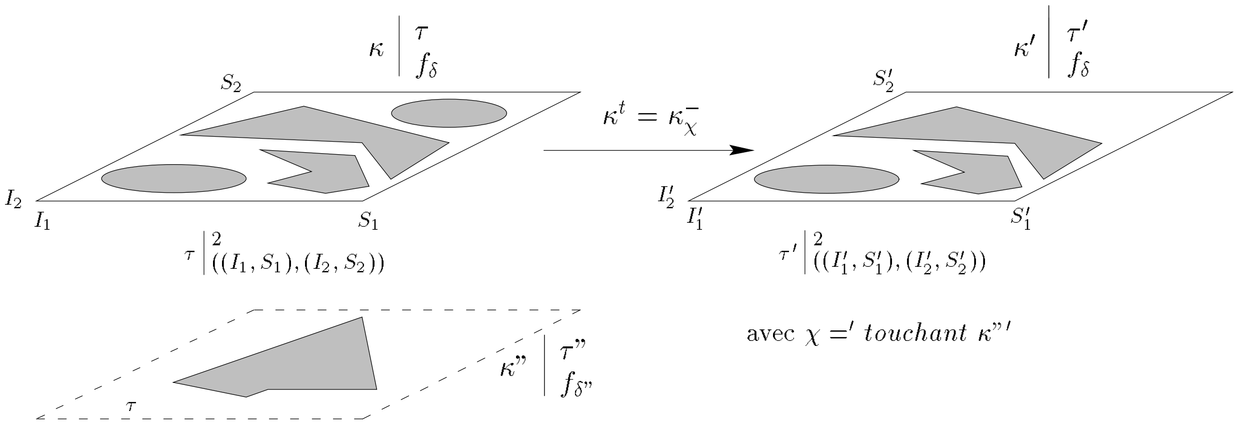


FIG. 3.14 - *Réduction du cache par mise en relation*

3.2.4 Généralisation de la description de la fonction cache

$\underline{\kappa^t = \kappa_\chi^+}$: La transformation qui consiste à augmenter la taille du cache peut être considérée comme l'inverse de la transformation précédente. Elle se traduit par la généralisation de la description δ de la fonction cache f_δ associée au cache. Comme précédemment, elle s'effectue par l'utilisation d'une caractéristique χ qui cette fois est retirée de la description. χ peut correspondre à une propriété p (fig.3.15) ou à la description δ du cache d'une seconde donnée de même dimension (fig.3.16). Cette transformation du cache peut alors s'écrire :

$$\kappa_\chi^+ \left| \begin{array}{l} K_\tau \longrightarrow K_\tau \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \kappa' \left| \begin{array}{l} \tau \\ f_{\delta'=\mathcal{G}n(\chi,\delta)} \end{array} \right. \end{array} \right. \tag{3.20}$$

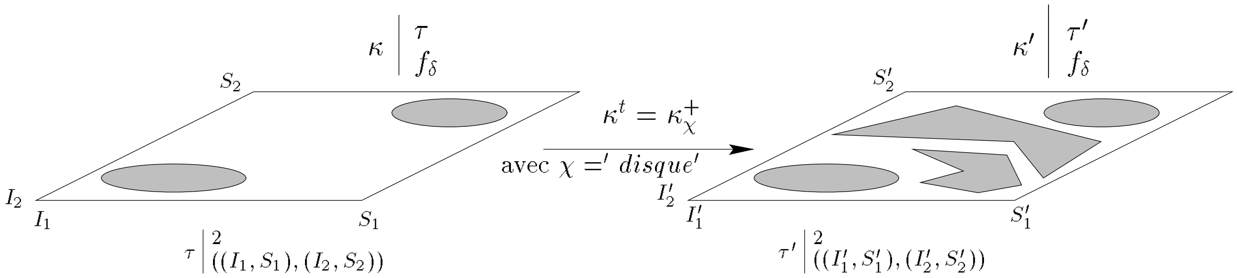


FIG. 3.15 - *Augmentation du cache par retrait de la propriété "disque"*

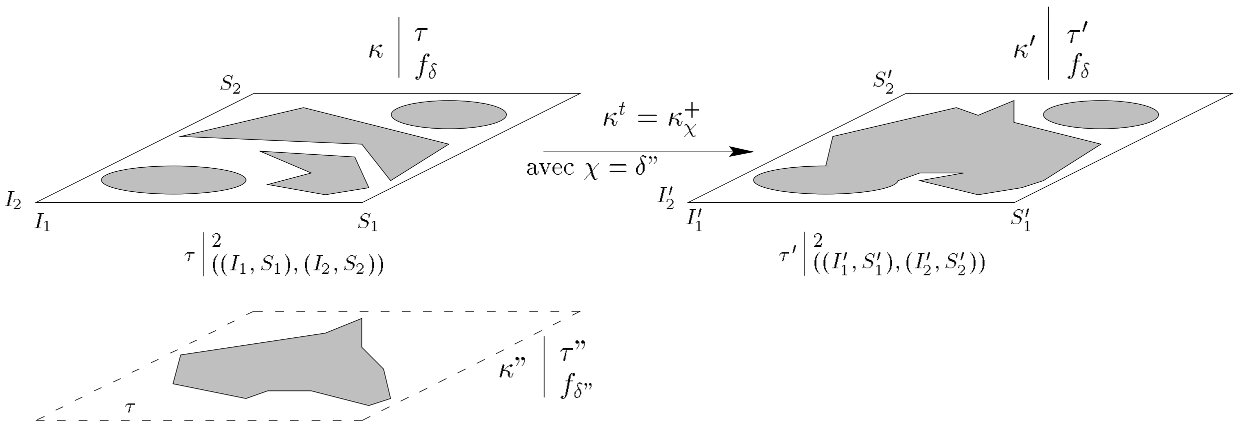


FIG. 3.16 - *Augmentation du cache par disjonction*

3.3 Transformation de l'index

3.3.1 Introduction sur ι^t

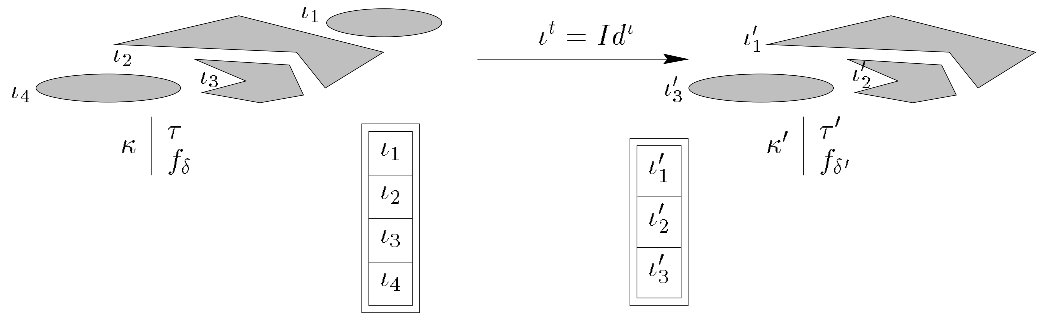
Nous avons défini trois sortes d'index -global, d'indépendance et dépendant d'un critère- fournissant une information sur le mode de considération des éléments du cache. N'étant pas lié au contenu de celui-ci, l'index peut être considéré indépendamment. Nous pourrons alors définir des transformations simples de structuration (déstructuration) et spécialisation (généralisation) d'index auxquelles nous ajouterons une fonction d'identité. Les notions de spécialisation (généralisation) de l'information index sont plus intuitives que celles précédemment utilisées. En effet, cet élément de la donnée reflète une information liée à une sorte d'étiquetage mental des composantes du cache qui n'est pas forcément issu de l'application d'un opérateur d'étiquetage proprement dit.
Avec les modifications des index, nous abordons donc des ensembles de fonctions de transformation plus élémentaires dans leur conception mais plus proche d'une représentation mentale des éléments de la donnée manipulée. Ceci sera nettement perceptible pour les fonctions de spécialisation et de généralisation.

3.3.2 Transformation identique

$\underline{\iota^t = \mathcal{I}d^\iota :}$ La fonction d'identité de l'index d'une donnée n'affecte en rien la sémantique de cet index mais peut modifier sa table (fig.3.17). Elle s'écrit de façon simple :

$$\mathcal{I}d^\iota \left| \begin{array}{l} I \longrightarrow I \\ \iota \longmapsto \iota \end{array} \right. \tag{3.21}$$

Dans toutes ces transformations sur les index, seule la caractéristique sémantique nous intéresse pour la modélisation symbolique ; la table d'index ne pouvant être disponible qu'à l'exécution.



$\iota = (\mathcal{C}\ [\iota_1, \iota_2, \iota_3, \iota_4])$ $\quad \iota = (\mathcal{C}\ [\iota_1', \iota_2', \iota_3'])$

FIG. 3.17 - *Transformation identique* $\iota^t = \mathcal{I}d^\iota$

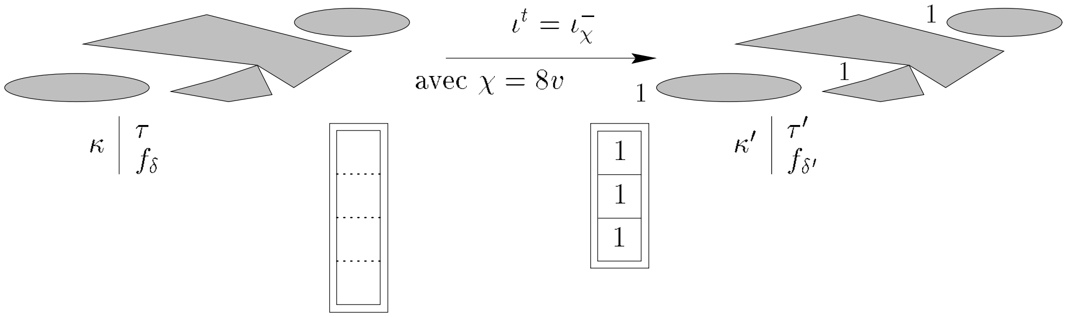
3.3.3 Transformations de spécialisation et de généralisation

Ces deux transformations apparaissent clairement dans les autres séries de traitement d'information des champs de la donnée comme des fonctions permettant de sélectionner (généraliser) une entité. Cette notion est un peu différente dans le cas présent et, contrairement aux autres, ne peut réellement se représenter graphiquement. En effet ces deux transformations agissent sur l'index en ajoutant (enlevant) une information pour préparer les composantes du cache de la donnée à un traitement spécifique comme par exemple un étiquetage particulier. Nous pouvons caractériser ces deux transformations par le fait qu'elles ne modifient pas de façon symbolique le nombre de références dans la table d'index ; elles peuvent néanmoins en modifier les valeurs. Encore une fois cette notion est délicate à expliciter par un schéma qui, comme dans la pratique sur des données réelles, doit mettre en œuvre une notion de voisinage, celle-ci n'apparaissant pas toujours nécessaire pour des opérateurs de traitement d'images.

3.3.3.1 Transformation de spécialisation d'index

$\underline{\iota^t = \iota_\chi^-}$: Cette fonction de transformation va donc consister à ajouter une information χ sur l'index (fig.3.18) pour obtenir un nouveau point de vue du cache, dans lequel les composantes même resteront inchangées. L'expression de cette fonction est la suivante :

$$\iota_\chi^- \left| \begin{array}{l} I \longrightarrow I \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{S}p(\chi, \mathcal{C})\ [\ldots]) \end{array} \right. \tag{3.22}$$



$\iota = (\mathcal{G}l\ [1])$ $\quad \iota = (\mathcal{S}p(8v, \mathcal{G}l)\ [1])$

FIG. 3.18 - *Transformation de spécialisation* $\iota^t = \iota_\chi^-$

3.3.3.2 Transformation de généralisation d'index

$\iota^t = \iota^+_\chi$: Cette fonction de transformation va consister à retirer une information χ sur l'index. Dans l'exemple suivant (fig.3.19) nous montrons l'effet de l'index sur des éléments d'un cache. Dans la donnée d'entrée, ses composantes sont regroupées par coupes de tores (1.x, 2.x) sur lesquels nous avons considéré chaque composante connexe. (1.1, 1.2, 2.1, 2.2). Le résultat donnera un point de vue plus général des composantes du cache : il n'y a pas création d'une nouvelle référence dans la table d'index, ni destruction d'une référence existante.

$$\iota^+_\chi \left| \begin{array}{l} I \longrightarrow I \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{G}n(\chi, \mathcal{C})\ [\ldots]) \end{array} \right. \tag{3.23}$$

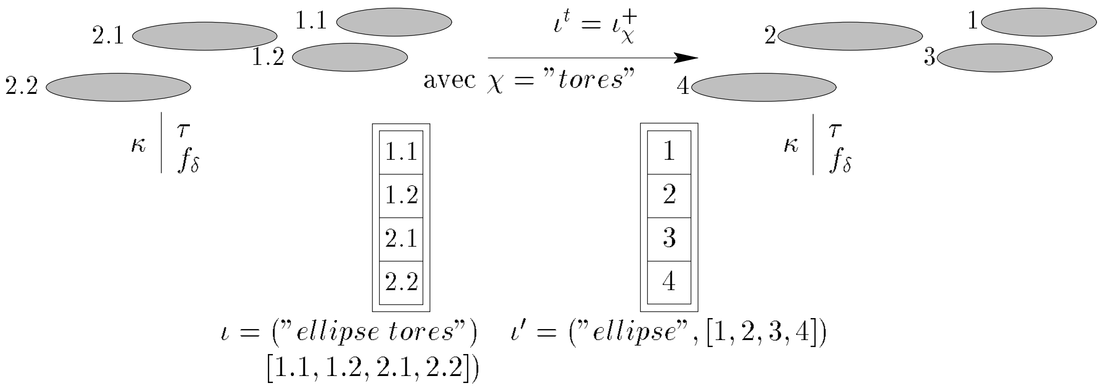


FIG. 3.19 - *Transformation de généralisation* $\iota^t = \iota^+$

3.3.4 Transformations de structuration

3.3.4.1 Structuration de globalité

$\iota^t = \Gamma^-$: La fonction de structuration Γ^- appliquée à un index ι aura pour effet de regrouper sous une même valeur d'index (index unitaire) tous les éléments caractérisés par le cache sur lequel il est appliqué (fig.3.20). Nous pouvons ainsi définir Γ^- par :

$$\Gamma^- \left| \begin{array}{l} I \longrightarrow I_{\mathcal{G}l} \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{G}l\ [1]) \end{array} \right. \tag{3.24}$$

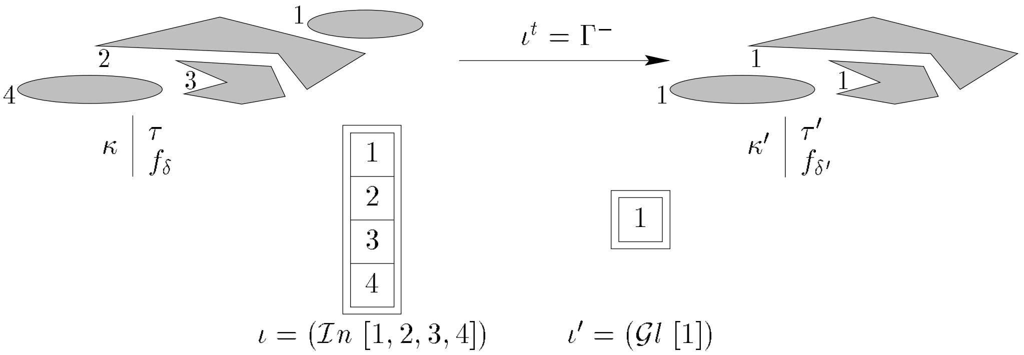
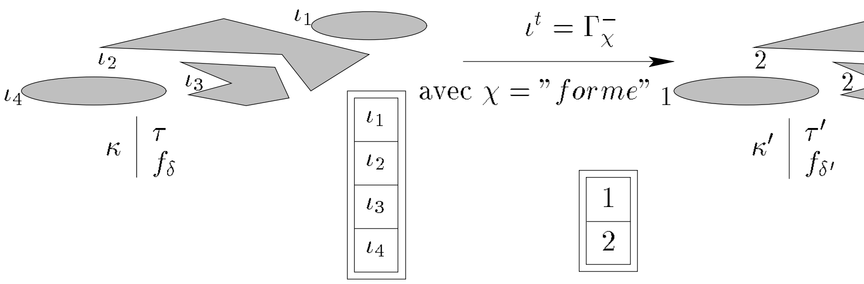


FIG. 3.20 - *Structuration de globalité*

3.3.4.2 Transformation de structuration par une caractéristique

$\iota^t = \Gamma^-_\chi$: La fonction de structuration par une caractéristique χ appliquée à un index ι aura pour effet de regrouper sous une même valeur d'index les composantes du cache possédant la caractéristique χ (fig.3.21). La transformation Γ^-_χ s'énonce alors :

$$\Gamma^-_\chi \left| \begin{array}{l} I \longrightarrow I \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{S}t(\chi, \mathcal{G}l)\ [\ldots]) \end{array} \right. \tag{3.25}$$



$\iota = (\mathcal{C}\ [\iota_1, \iota_2, \iota_3, \iota_4])$ $\iota' = (\mathcal{S}t(\text{"}forme\text{"}, \mathcal{C})\ [1, 2])$

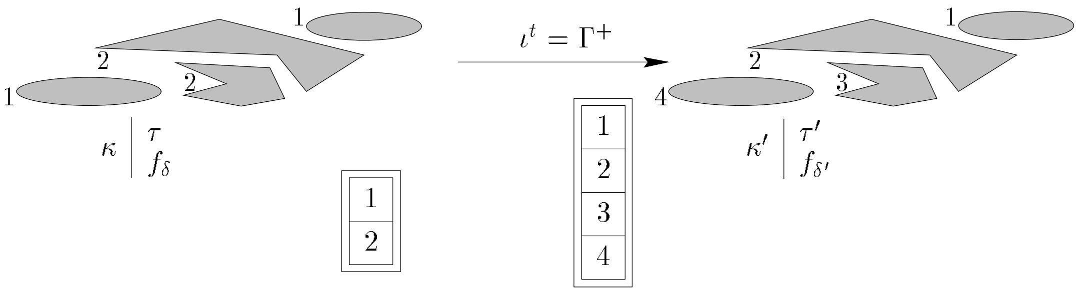
FIG. 3.21 - *Structuration par χ*

3.3.5 Transformations de déstructuration

3.3.5.1 Déstructuration d'indépendance

$\iota^t = \Gamma^+$: La fonction de déstructuration Γ^+ appliquée à un index ι aura l'effet contraire de la fonction précédente. Elle permettra de dissocier par des index distincts tous les éléments caractérisés par le cache sur lequel il est appliqué (fig.3.22). Nous pouvons alors considérer Γ^+ comme la transformation inverse de Γ^- :

$$\Gamma^+ \left| \begin{array}{l} I \longrightarrow I_{\mathcal{I}n} \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{I}n\ [\ldots]) \end{array} \right. \tag{3.26}$$



$\iota = (\text{"}formes\text{"}\ [1, 2])$ $\iota' = (\mathcal{I}n\ [1, 2, 3, 4])$

FIG. 3.22 - *Déstructuration d'indépendance*

3.3.5.2 Déstructuration par une caractéristique

$\iota^t = \Gamma^+_\chi$: La fonction de déstructuration par une caractéristique aura pour effet de décomposer les éléments du cache regroupés sous une même valeur d'index afin de créer un nouveau point de vue faisant apparaître la nouvelle caractéristique χ (fig.3.23). À l'image de la transformation Γ^-_χ, Γ^+_χ s'énonce de la façon suivante :

$$\Gamma^+_\chi \left| \begin{array}{l} I \longrightarrow I \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{D}s(\chi, \mathcal{G}l)\ [\ldots]) \end{array} \right. \tag{3.27}$$

(fig.3.23)

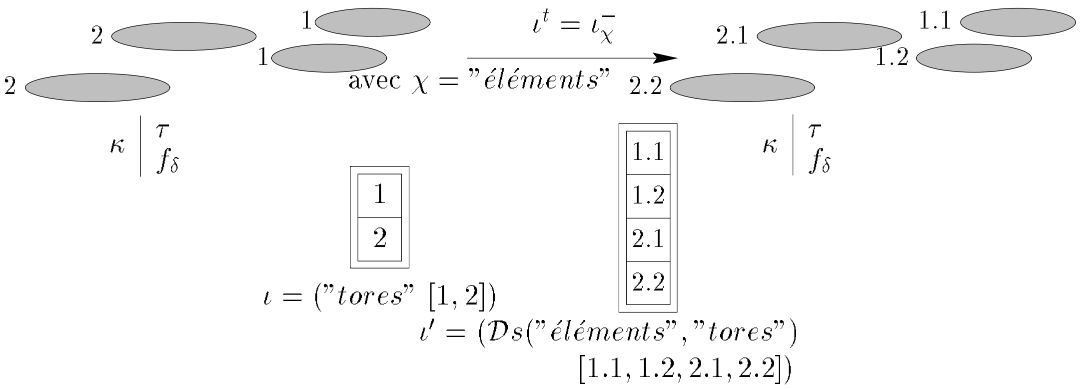


FIG. 3.23 - *Déstructuration par χ*

3.4 Transformation du signal

3.4.1 Introduction sur σ^t

Comme il a été évoqué précédemment (voir 1.2.4 page 67), l'ensemble Sgn des signaux peut être considéré comme l'union des sous-ensembles Sgn^i des signaux potentiels d'une donnée de dimension i. Il faut toutefois noter que tout signal σ de Sgn^i appartient par l'effet de la transformation par sélection de trame aux ensembles de signaux Sgn^k avec $k < i$. De ce fait un ensemble Sgn^n avec $n \in \mathbb{N}$ peut être défini comme l'ensemble des signaux de Sgn tels que si un de ses éléments σ appartient à un ensemble de signaux Sgn^k alors $k < n$. Ainsi :

$$Sgn^n = \bigcup_{i=0}^{n} Sgn^i \tag{3.28}$$

D'autre part, le signal étant une caractéristique relative à la sémantique de l'information renfermée dans les points de la donnée, nous pouvons déduire de ce qui précède que la caractéristique *signal* d'une donnée va évoluer comme la caractéristique *cache*, selon la variation de la dimension de la donnée. De ce fait nous allons considérer tout d'abord les fonctions de transformation du signal non liées à une variation de dimension, puis celles liées à une telle variation, et enfin celles liées à une invariance de cette dimension.

3.4.2 Transformations indépendante d'une variation de dimension

3.4.2.1 Transformation identique

$\underline{\sigma^t = \mathcal{I}d^\sigma}$: Cette fonction de transformation du signal n'est liée à aucune transformation des autres caractéristiques de la donnée. Ainsi, elle peut s'appliquer lors de transformation de la trame par sélection d'entité géométrique (fig.3.24) ou de transformation du cache (fig.3.25). Quelle que soit la dimension de la donnée sur laquelle elle est appliquée, cette transformation peut s'écrire tout simplement :

$$\mathcal{I}d^\sigma \left| \begin{array}{l} Sgn \longrightarrow Sgn \\ \sigma \longmapsto \sigma \end{array} \right. \tag{3.29}$$

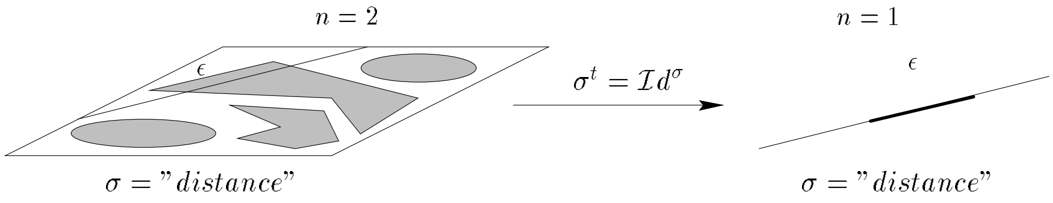


FIG. 3.24 - *Transformation identique (1)*

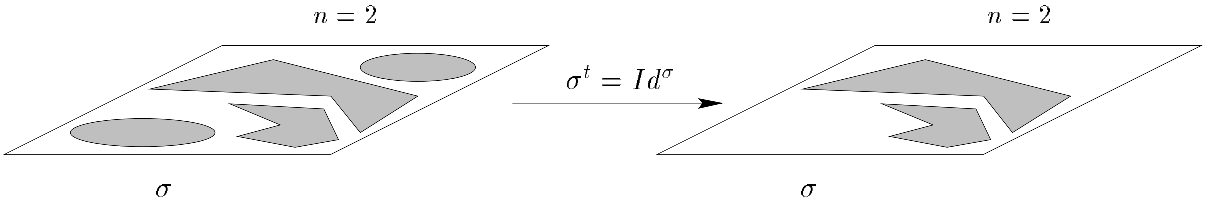


FIG. 3.25 - *Transformation identique (2)*

3.4.2.2 Transformation par fixation du signal

$\sigma^t = \Sigma_\varsigma$: Cette forme de fonction de transformation du signal se fait par l'intermédiaire d'une caractéristique ς (voir 3.1.3.3 page 111) correspondant au signal de la donnée transformée (fig.3.26). Dans le cadre d'une éventuelle variation de la dimension de la donnée, la fixation du signal consiste à fournir un signal indépendant de celui de la donnée initiale ; c'est le cas lorsque des mesures sont effectuées. Dans le cadre d'une non variation de cette dimension, la fixation du signal consiste à suivre le sens de structuration donné au cache ; c'est le cas pour les signaux "valeur-binaire" (sens structurant) et mesure de distance (sens déstructurant). Pour ces raisons, la transformation par fixation de signal peut être considérée comme une fonction de structuration / déstructuration.

Ainsi nous pouvons décrire la transformation par la fonction :

$$\Sigma_\varsigma \left| \begin{array}{l} Sgn^i \longrightarrow Sgn^{i \pm k} \\ \sigma \longmapsto \varsigma \end{array} \right. \tag{3.30}$$

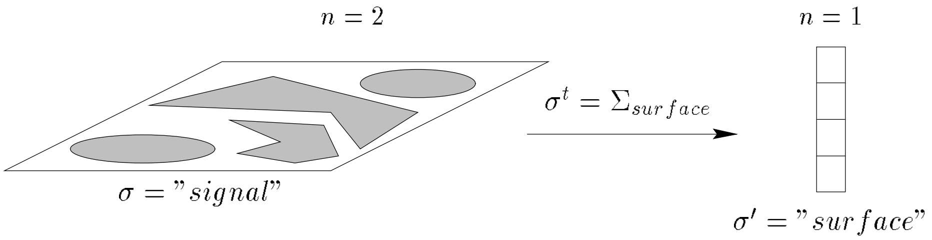


FIG. 3.26 - *Transformation par fixation du signal*

3.4.3 Transformations sans variation de dimension

3.4.3.1 Transformation de déstructuration du signal

$\underline{\sigma^t = \Sigma^+ :}$ La fonction de déstructuration d'un signal va permettre de considérer un signal de Sgn^n comme constitué d'un ensemble fini de signaux de Sgn^j tels que $j < n$ (fig.3.27).

$$\sigma \in \Sigma^n = \left\{ \sigma_i \in \Sigma^i \Big/ i < n, card(\sigma) \leq n \right\} \tag{3.31}$$

Pour cela, nous pouvons appeler sous-signal chacun de ces signaux σ_j avec $j \in [1, i \leq n]$ et définir la fonction de déstructuration de signal de la façon suivante :

$$\forall n \in \mathbb{N},\ \Sigma^+ \left| \begin{array}{l} Sgn^n \longrightarrow \{\sigma_i \in \Sigma^i / i < n, card(\sigma) \leq n\} \\ \sigma \longmapsto \{\sigma_i \in \Sigma^i\} \end{array} \right. \tag{3.32}$$

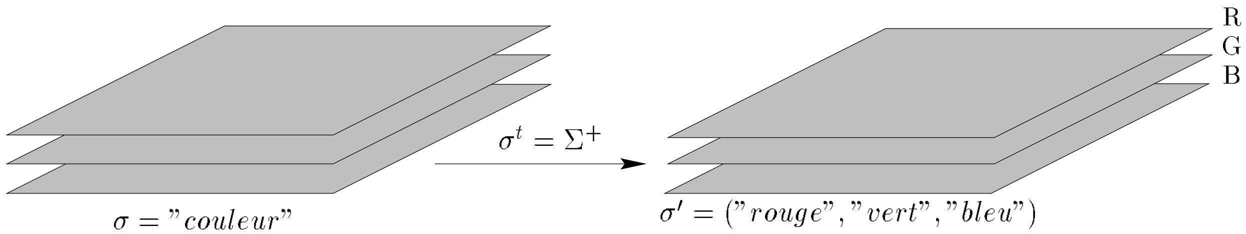


FIG. 3.27 - *Déstructuration du signal couleur*

3.4.3.2 Transformation de structuration du signal

$\underline{\sigma^t = \Sigma^- :}$ À l'opposé de la précédente, la fonction de structuration de signal permet de construire un terme global pour un ensemble spécifique de sous-signaux (fig.3.28). Comme pour les autres fonctions de ce type, la structuration du signal peut être considérée comme l'inverse de la transformation de déstructuration de ce même signal. La description de cette transformation est la suivante :

$$\Sigma^- \left| \begin{array}{l} \{\sigma_i \in \Sigma^i / i < n, card(\sigma) \leq n\} \longrightarrow Sgn^n \\ \{\sigma_i \in \Sigma^i\} \longmapsto \sigma \end{array} \right. \tag{3.33}$$

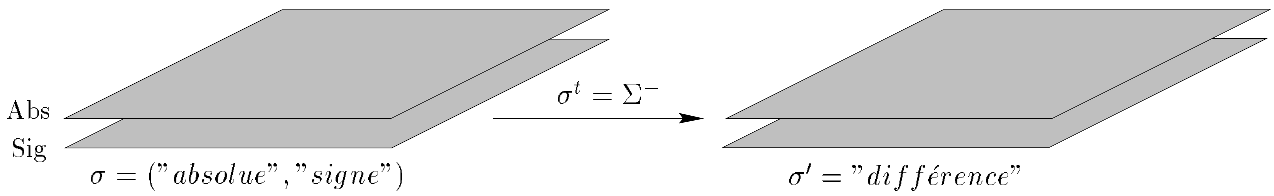


FIG. 3.28 - *Structuration du signal différence*

3.4.3.3 Transformation de généralisation du signal

$\underline{\sigma^t = \sigma_\chi^+ :}$ Les fonctions de généralisation du signal accompagnant une invariance de la dimension de la donnée permettent de regrouper les points de celle-ci au sein de chaque élément du cache (fig.3.29). Ces fonctions de transformation sont alors mises en œuvre par le biais d'une caractéristique χ, élément modificatif du signal de la donnée. Pour toute dimension n de $\mathbb{N}$, cette transformation est définie par :

$$\sigma_\chi^+ \left| \begin{array}{l} Sgn^n \longrightarrow Sgn^n \\ \sigma \longmapsto \mathcal{G}n(\chi, \sigma) \end{array} \right. \tag{3.34}$$

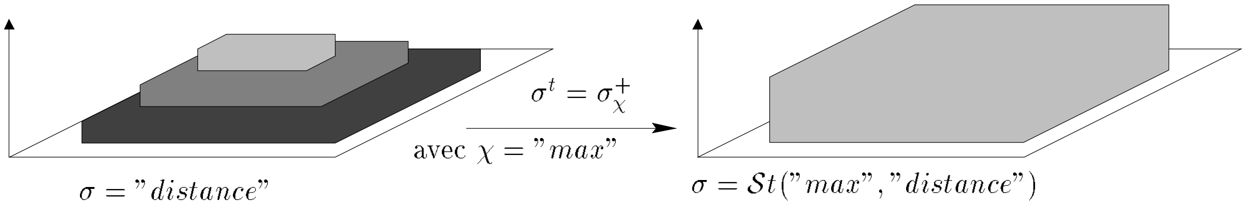


Fig. 3.29 - *Généralisation du signal*

3.4.3.4 Transformation de spécialisation du signal

$\sigma^t = \sigma_\chi^-$: Comme pour les autres fonctions, la spécialisation va effectuer des traitements complémentaires à la fonction de généralisation ; il s'agira donc pour elles de fournir une information supplémentaire quant à la sémantique de la valeur des points de la donnée (fig.3.30). Les fonctions de spécialisation du signal s'exprimeront de la façon suivante :

$$\sigma_\chi^- \left| \begin{array}{l} Sgn^n \longrightarrow Sgn^n \\ \sigma \longmapsto \mathcal{S}p(\chi, \sigma) \end{array} \right. \tag{3.35}$$

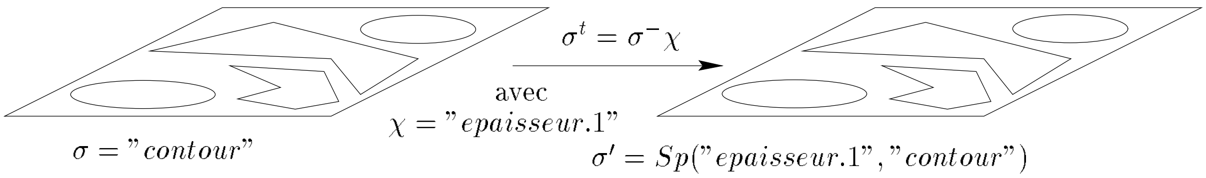


Fig. 3.30 - *Spécialisation du signal*

3.4.4 Transformations avec variation de dimension

3.4.4.1 Transformation de spécialisation

$\sigma^t = \sigma_\chi^-$: Ce sont en fait ces fonctions de spécialisation du signal qui vont entraîner des variations de la dimension de la donnée. Ces spécialisations se feront soit en ajoutant au signal une caractéristique χ, élément modificatif des signaux (fig.3.31), soit en caractérisant ce signal par un sous-signal (fig.3.32). Ces fonctions s'écrivent de la façon suivante :

$$\forall i \in \mathbb{N},\ \sigma_\chi^- \left| \begin{array}{l} Sgn^i \longrightarrow Sgn^{i-k} \\ \sigma \longmapsto \mathcal{S}p(\chi, \sigma) \end{array} \right. \tag{3.36}$$

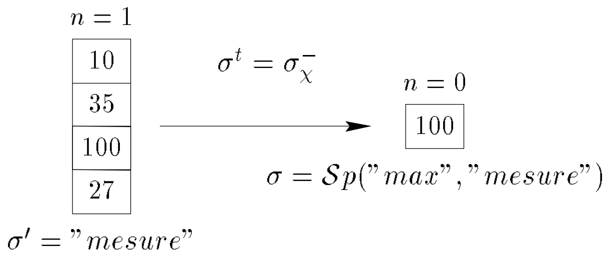


FIG. 3.31 - *Transformation de spécialisation par un élément modificatif*

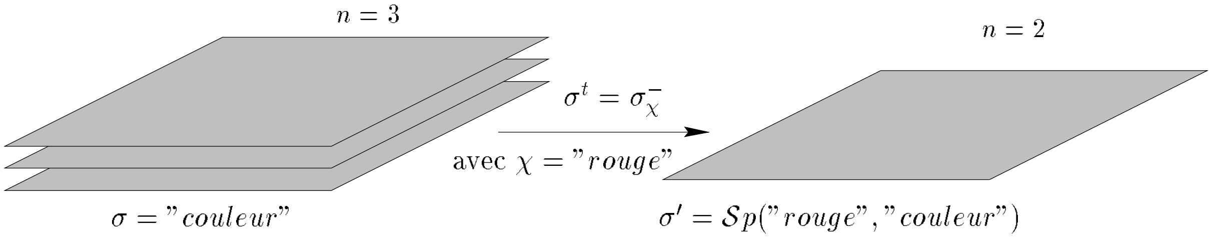


FIG. 3.32 - *Transformation de spécialisation par un sous-signal*

3.4.4.2 Transformation de généralisation

$\sigma^t = \sigma_\chi^+$: Les fonctions de généralisation suivront la même règle de complément des fonctions de spécialisation. Nous les définissons donc comme des fonctions permettant de généraliser un signal de dimension n soit en accédant à un signal de dimension $n+k$, soit en enlevant une caractéristique χ (fig.3.33). Ces fonctions s'écrivent ainsi :

$$\forall i \in \mathsf{IN},\ \sigma_\chi^+ \left| \begin{array}{l} Sgn^i \longrightarrow Sgn^{i+k} \\ \sigma \longmapsto \mathcal{G}n(\chi, \sigma) \end{array} \right. \tag{3.37}$$

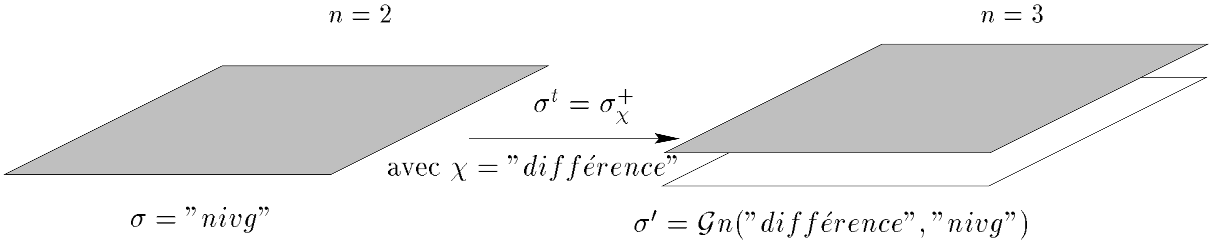


FIG. 3.33 - *Transformation de généralisation par un sous-signal*

3.5 Transformation du codage

3.5.1 Introduction sur ω^t

Nous abordons enfin les transformations du dernier composant de la donnée. Étroitement lié à la caractéristique *signal*, le codage va être manipulé par quatre ensembles de fonctions élémentaires qui sont les fonctions d'identité et de fixation du codage applicables dans tous les cas ainsi que les transformations de spécialisation et de généralisation du codage. Il faut noter que les fonctions de structuration et de déstructuration n'existent pas pour le codage. En effet, ces notions impliquent une hiérarchisation des entités manipulées. Or, mis à part dans des domaines particuliers -tels l'ensemble des complexes- où nous trouvons des éléments spécifiques qui peuvent néanmoins être exprimés autrement -$i = 0 + 1i \equiv (0,1)$-, il n'existe pas de forme de codage dans lequel nous puissions trouver des éléments de codage déstructurables dans le sens que nous lui avons donné jusqu'à présent. Pour simplifier les écritures suivantes, nous noterons par un élément générique $\mathcal{X}$, les ensembles IN, IR, caractères alphabétiques, etc...

3.5.2 Transformation identique

$\underline{\omega^t = \mathcal{I}d^\omega :}$ C'est une transformation laissant inchangé l'ensemble ω des valeurs de codage que peut prendre le signal de la donnée (fig.3.34). Cette transformation peut s'écrire de la façon suivante :

$$\mathcal{I}d^\omega \left| \begin{array}{l} \mathcal{X} \longrightarrow \mathcal{X} \\ \omega \longmapsto \omega \end{array} \right. \tag{3.38}$$

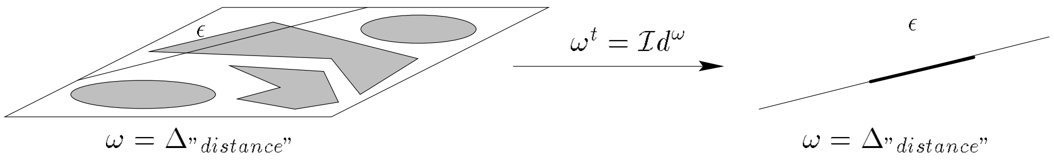


FIG. 3.34 - *Transformation identique*

3.5.3 Transformation par fixation du codage

$\underline{\omega^t = \omega^\sigma :}$ Le signal d'une donnée fournit une information sur la représentation effective des points de la donnée et regroupés sous une même valeur de codage. Chaque signal possède donc son propre ensemble de valeurs que nous avons déjà rencontré et appelé domaine de validité du signal et noté Δ_σ. Cette transformation par fixation du codage va donc accompagner une transformation du signal et attribuer à la donnée transformée le domaine de validité de son nouveau signal privé de l'élément nul (fig.3.34). Elle est décrite par la formule suivante :

$$\omega_\sigma \left| \begin{array}{l} \mathcal{X} \longrightarrow \mathcal{X} \\ \omega \longmapsto \Delta_\sigma \setminus 0 \end{array} \right. \tag{3.39}$$

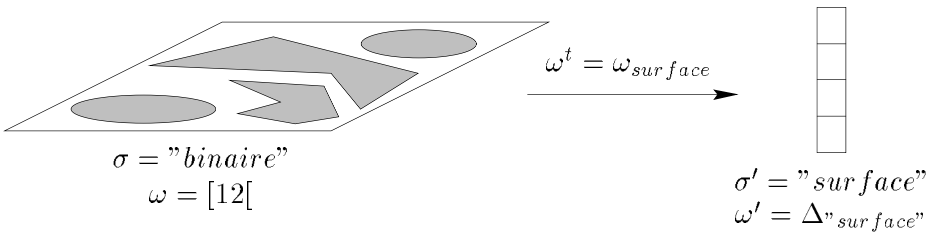


FIG. 3.35 - *Transformation par fixation*

3.5.4 Transformation de spécialisation

$\underline{\omega^t = \omega_\chi^-}$: Ces fonctions de spécialisation du codage vont s'appliquer lorsque des caractéristiques χ issues d'autres transformations sur la donnée restreignent les valeurs du codage de celle-ci (fig.3.36).

Ces fonctions s'écrivent de la façon suivantes :

$$\omega_\chi^- \left| \begin{array}{l} \mathcal{X} \longrightarrow \mathcal{X} \\ \omega \longmapsto \mathcal{S}p(\chi, \omega) \end{array} \right. \tag{3.40}$$

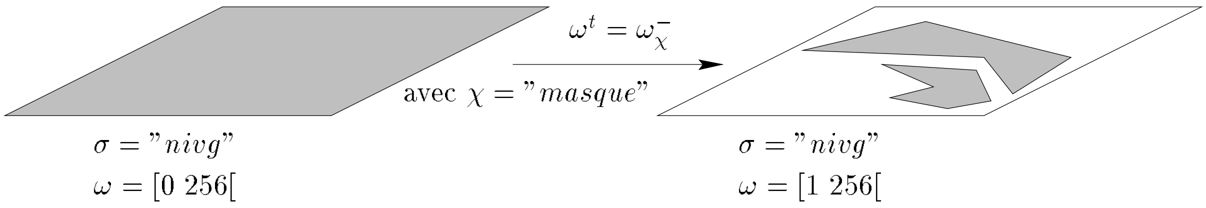


FIG. 3.36 - *Transformation de spécialisation par un sous-signal*

3.5.5 Transformation de généralisation

$\omega^t = \omega_\chi^+$: Les fonctions de généralisation vont s'appliquer de la même façon à ceci près que le codage résultat d'une généralisation doit toujours respecter la contrainte du domaine de validité du signal de la donnée : $D.\omega \in \Delta_{D.\sigma}$ (fig.3.37). Ces fonctions s'écrivent ainsi :

$$\omega_\chi^+ \left| \begin{array}{l} \mathcal{X} \longrightarrow \mathcal{X} \\ \omega \longmapsto \mathcal{G}n(\chi, \omega) \end{array} \right. \tag{3.41}$$

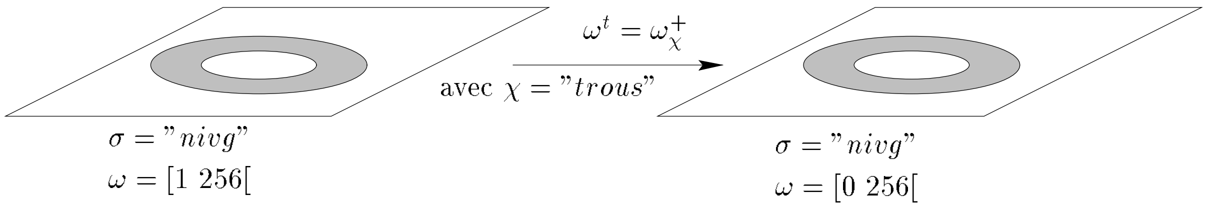


FIG. 3.37 - *Transformation de généralisation*

Chapitre 4

Utilisation pour la résolution d'objectif

Nous avons présenté une nouvelle approche concernant la modélisation des opérateurs et des données de traitement d'images. Nous allons maintenant montrer comment ces modèles permettent de rendre bien compte des transformations d'information effectuées par les opérateurs et décrivent précisément les entités produites, intervenant dans les indices visuels. De cette façon, une séquence d'opérateurs est considérée comme un constructeur de concepts [Dej96d] fournissant une description de ceux-ci dans un domaine particulier.

Nous allons valider ces modèles, en montrant tout d'abord leur cohérence dans l'analyse de séquences d'opérateurs, et leur exploitation pour la compréhension du traitement d'images, puis en examinant leur intégration dans un système de planification.

4.1 Comprendre le traitement d'images

Un opérateur d'analyse et de traitement, extrait une information particulière de l'image ou plus généralement de la donnée de traitement d'images. Une description de cette extraction se doit d'expliciter les étapes de détection, de représentation et de modification qui rendent possible la distinction de l'information à atteindre du reste de la donnée [Dal91]. Ainsi, un opérateur peut être considéré sous deux points de vues :

1. comme manipulateur de données à l'aide des opérations de transformations étudiées dans le chapitre précédent. Ces opérations sont combinées afin de définir des opérateurs plus ou moins complexes.

2. comme manipulateur de concepts, assurant des divisions ou regroupements d'entités qui vont correspondre à des spécialisations ou généralisations des concepts, et des opérations de représentation qui correspondront à des nominations de ces concepts [Ing96].

Pour faire le lien entre ces deux points de vue et montrer comment une séquence d'opérateurs peut être exprimée symboliquement, nous allons utiliser une représentation étendue de celle présentée dans [Ing92].

4.1.1 Une représentation des transformations

La représentation symbolique des transformations que nous proposons est basée sur les manipulations de concepts que nous avons déjà rencontrées ([Mic83, Mic84],[Ing96]) construisant et manipulant les entités de la donnée *(pixels, composantes connexes, ...)* en appliquant les mécanismes suivants :

- La *spécialisation* extrait l'information associée à un concept spécifiant un nouveau prédicat *(caractéristique, domaine)* vérifié par les instances de ce concept.
- La *généralisation* consiste à combiner de façon disjonctive plusieurs concepts ou à étendre les domaines de certains prédicats.
- La *nomination* de concepts est matérialisée par la représentation, dans le domaine d'expertise, de la donnée produite. En traitement d'images, l'espace de représentation est plutôt insuffisant *(image binaire, image 8-bit, ...)*. En conséquence, les informations considérées comme secondaires pour les opérations effectuée sont ainsi mélangées aux informations intéressantes dans la même description d'entité ; elles sont donc représentées de la même façon. Par exemple, dans des images binaires, les valeurs nulles contiennent souvent tout ce qui a été filtré pendant les étapes précédentes du processus de traitement, sans distinction entre elles.
  Le modèle que nous proposons rend compte de cette réalité du traitement d'images en fournissant à l'intérieur d'une description l'*historique* -ou trace- de ces traitements successifs.
- La *structuration* consiste en un regroupement dans un niveau supérieur de considération des entités (pixels, composantes connexes), de celles possédant une caractéristique commune en respectant une certaine relation -distance- spatiale *(par exemple : regrouper les pixels 4-voisins dont la valeur d'intensité est supérieure à un seuil donné)*.
- La *déstructuration* permet à l'inverse de considérer, afin de les traiter séparément, les informations relatives aux éléments constituants de l'entité manipulée *(par exemple : la valeur de distance au bord de la forme des pixels d'une composante connexe)*.

Notre représentation n'a pas pour but de représenter les fonctionnalités ni les types des données manipulées par les opérateurs de traitement d'images mais plutôt les transformations d'information qu'ils produisent sur les données. De ce fait, elle ne tient pas compte du modèle de la donnée ni des opérations de transformations introduites plus haut mais elle intègre les cinq étapes élémentaires (fig.4.1) qui peuvent avoir lieu au cours d'un traitement :

- L'*identification* détermine les sous-structures de la donnée initiale. Ces éléments de la donnée correspondant à même un identificateur feront l'objet d'un même traitement.
- La *mesure* produit une valeur directement calculée sur une sous-structure de la donnée. Cette évaluation est effectuée en utilisant une caractéristique spécifique, liée à l'opérateur.
- La *partition* consiste en la séparation des entités en plusieurs classes conformément à un prédicat donné -sélecteur, domaine- constitué de la mesure précédente pour le sélecteur et des paramètres définissant le domaine, eux aussi très liés à l'opérateur. Ceci constitue donc une opération de spécialisation de l'information attachée à la sous-structure par le prédicat propre à l'opérateur.

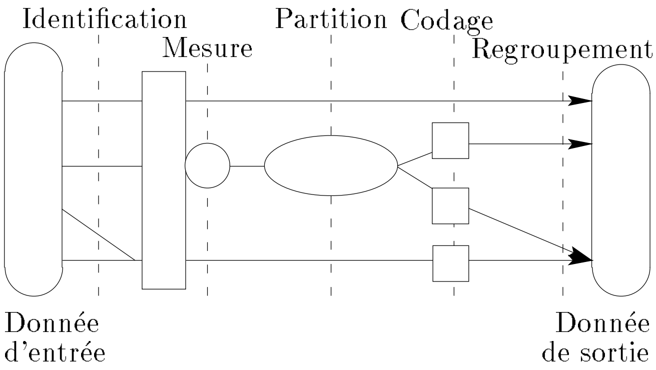


FIG. 4.1 - *Transformations d'information*

- Le *codage* consiste en l'attribution d'une valeur de sortie aux différentes entités réunies dans les classes provenant de la partition.
- Le *regroupement* rassemble, sous la même valeur de la donnée produite, les éléments issus des opérations précédentes qui ont reçu la même valeur de codage.

4.1.2 La propagation d'information : étude autour d'exemples

Comme les opérateurs manipulent les champs des données, nous obtenons par enchaînement de ces manipulations des expressions de plus en plus complexes et vérifiant la notion de propagation d'information. Cette notion peut se traduire par le fait qu'à chaque étape d'une séquence d'opérateurs, la description de la donnée manipulée peut s'écrire à l'aide des termes du langage de description de concepts, auxquels s'ajoutent les seuls éléments de description de la donnée initiale. Pour exprimer toutes ces notions, nous avons choisi de nous appuyer sur des exemples simples d'opérateurs. Nous avons sélectionné les opérateurs d'érosion et de dilatation binaire (élément structurant : disque de rayon égal à un) pour leur particularité morphologiques bien connues [Ser86]. À ces opérateurs, nous ajoutons les opérateurs de calculs de distance au bord des composantes, et les opérateurs logiques binaires : et, ou et complémentaires. Ces opérateurs simples font partie de la distribution KHOROS et se retrouvent dans tout logiciel de traitement d'images. Nous trouverons en annexe E les commandes relatives à ces opérateurs, ainsi que leur représentation dans le modèle proposé et en terme de transformation.

4.1.2.1 Décomposition d'opérateur

Prenons donc l'exemple de l'opérateur de dilatation par un disque β de rayon 1 et centré à l'origine, appliqué à une image binaire X.

Description de la Morphologie Mathématique :

$X \oplus \beta = \{x | \beta_x \cap X \neq \emptyset\} = X \cup \{x \notin X | \beta_x \cap X \neq \emptyset\}$ où β_x est le translaté de β par x.

Si nous décomposons le processus de formation de l'image dilatée (fig.4.2), et si nous l'examinons en termes d'indices visuels, nous pouvons voir que :

1. Les données d'entrée et de sortie de cet opérateur sont des images binaires. L'opérateur de dilatation ne transforme donc pas la nature de la représentation des sous-structures constituant l'image.
2. Les entités représentées par des valeurs à 1 dans l'image de sortie proviennent de deux entités de l'image d'entrée :

   - Des entités représentées par des valeurs de pixels à 1
     (X) *sous-structure : entrée.val=1*

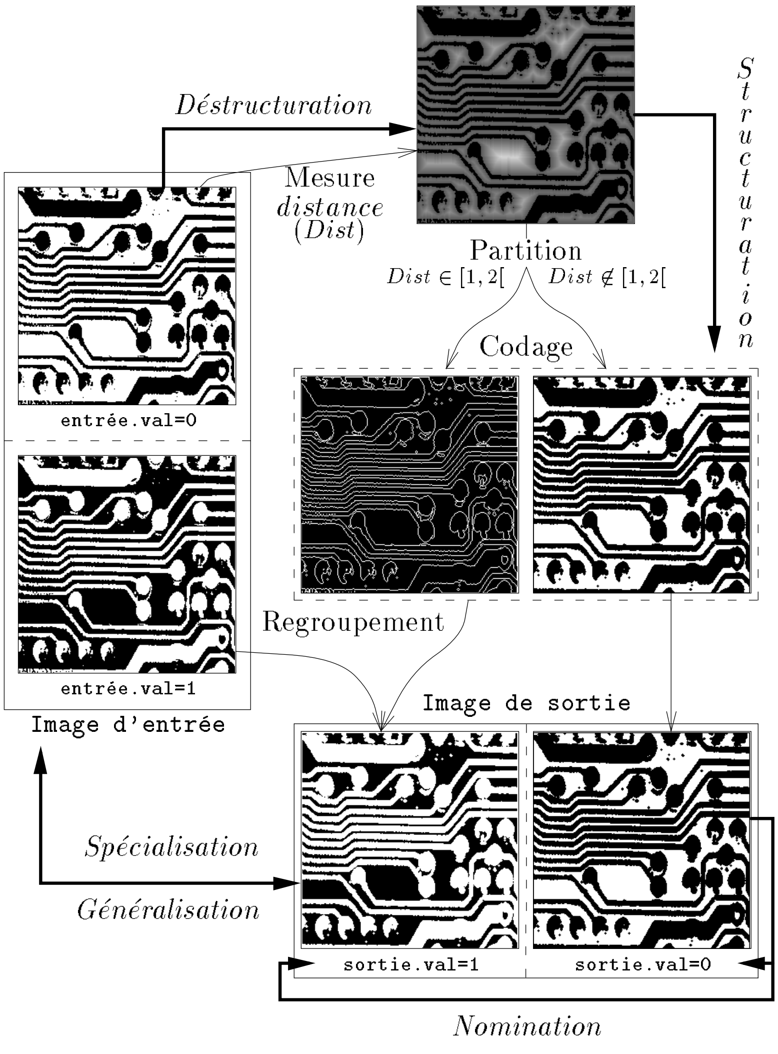


FIG. 4.2 - *Décomposition de l'opération de dilatation*

- D'une partie des entités représentées par des valeurs de pixels à 0 $(\{x \notin X | \beta_x \cap X \neq \emptyset\})$ *sous-classe de la sous-structure : entrée.val=0*

Cette partie de la sous-structure *entrée.val=0* correspond à la séparation de ses pixels selon un critère déterminé par l'évaluation de la mesure de *distance* : ($dist \in [1, 2[$).
Deux sous-classes apparaissent alors :

- Des entités représentées par des valeurs de pixels à 0 possédant une *distance* différente de 1
  $sous - classe\ :\ (\{entre.val = 0\}\&\{dist \notin [1, 2[\})$
- Des entités représentées par des valeurs de pixels à 0 possédant une mesure de *distance* égale à 1
  $sous - classe\ :\ (\{entre.val = 0\}\&\{dist \in [1, 2[\})$

Si nous considérons maintenant que le codage de cette seconde partie peut donner à ses pixels la même valeur que celle des pixels de la sous-structure *entrée.val=1*, nous pouvons établir une représentation des entités de la sous-structure *sortie.val=1* de l'image de sortie se basant sur la description de la connaissance des transformations effectuées par cet opérateur.
Cette représentation sera formulée comme étant le regroupement de la sous-structure *entrée.val=1* et de la sous-classe $(\{entre.val = 0\}\&\{dist\ [1, 2[\})$.

$$\{entre.val = 1\} \vee (\{entre.val = 0\}\&\{dist \in [1, 2[\})$$

3. Les entités représentées par des valeurs à 0 dans l'image de sortie proviennent, elles, de la sous-classe restante $(\{entre.val = 0\}\&\{dist \notin [1, 2[\})$.

Puisque nous avons pu le décomposer en une suite de transformations plus élémentaires, nous allons considérer cet opérateur de dilatation comme un macro-opérateur s'exprimant par la séquence de la figure 4.3.

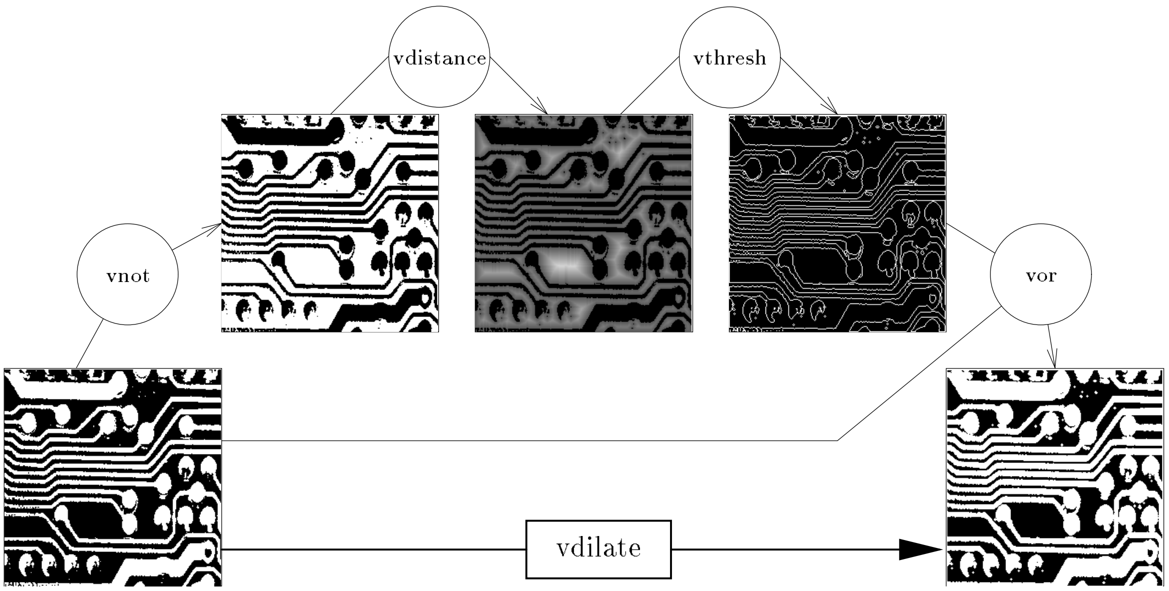


FIG. 4.3 - *Séquence pour la dilatation*

4.1.2.2 Représentations de la décomposition

L'exemple précédent montre comment la dilatation peut être décomposée à l'aide des opérateurs élémentaires de la même bibliothèque de traitement d'images. Si nous employons maintenant les schémas symboliques des opérateurs élémentaires le composant, nous pouvons représenter le macro-opérateur (fig4.4), et montrer comment cette décomposition peut être rendue par les modèles que nous avons proposé.

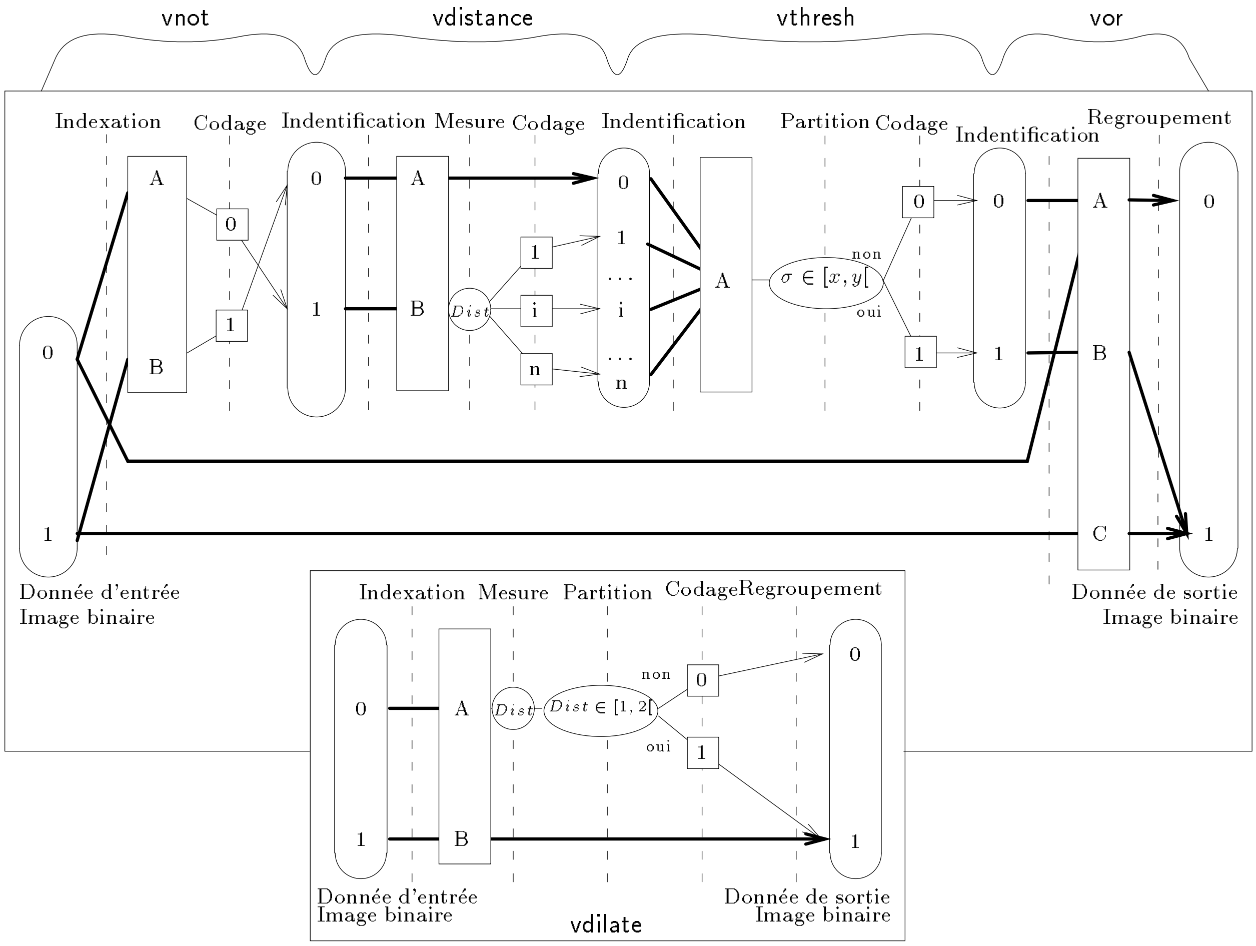


FIG. 4.4 - *Schéma symbolique de la dilatation*

Soit D une donnée image binaire. Si nous utilisons les représentations des modèles précédents de donnée et d'opérateur, la dilatation de D peut s'écrire :

$$D = \left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\delta \\ \iota = D.\iota \\ \sigma = D.\sigma = vab \\ \omega = D.\omega = [1, 2[\end{array} \right\}$$

$$Non(D) = \left[\begin{array}{l} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \mathcal{I}d^{X.\overline{\delta}} \\ \iota^t = \Gamma^- \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{array} \right] \left(\left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\delta \\ \iota = D.\iota \\ \sigma = D.\sigma \\ \omega = D.\omega \end{array} \right\} \right) = \left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\overline{\delta} \\ \iota = \mathcal{G}l \\ \sigma = vab \\ \omega = [1, 2[\end{array} \right\}$$

$$Dst(Non(D)) = \begin{bmatrix} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \mathcal{I}d^\delta \\ \iota^t = \mathcal{I}d^\iota \\ \sigma^t = \Sigma_{dist} \\ \omega^t = \omega_{dist} \end{bmatrix} \left(\left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\overline{\delta} \\ \iota = \mathcal{G}l \\ \sigma = vab \\ \omega = [1,2[\end{array} \right\} \right) = \left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\overline{\delta} \\ \iota = \mathcal{G}l \\ \sigma = dist \\ \omega = [1, min(\mathcal{T}(D.\tau))/2[\end{array} \right\}$$

$$\begin{aligned} Sll(Dst(Non(D)),1,2) &= \begin{bmatrix} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \kappa^-_{\{X.\sigma\in[m,M[\}} \\ \iota^t = \Gamma^- \\ \sigma^t = \Sigma_{vab} \\ \omega^t = \omega_{vab} \end{bmatrix} \left(\left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\overline{\delta} \\ \iota = \mathcal{G}l \\ \sigma = dist \\ \omega = [1, min(\mathcal{T}(D.\tau))/2[\end{array} \right\}, 1, 2 \right) \\ &= \left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\overline{\delta} \ \& \ (dist\ 1\ 2) \\ \iota = \mathcal{G}l \\ \sigma = vab = D.\sigma \\ \omega = [1,2[= D.\omega \end{array} \right\} \end{aligned}$$

$$\begin{aligned} Ou(D, Sll(Dst(Non(D)),1,2)) &= \begin{bmatrix} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \kappa^+_{X'.\delta} \\ \iota^t = \Gamma^- \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{bmatrix} \left(\left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\overline{\delta} \ \& \ (dist\ 1\ 2) \\ \iota = \mathcal{G}l \\ \sigma = D.\sigma \\ \omega = D.\omega \end{array} \right\}, \left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\delta \\ \iota = D.\iota \\ \sigma = D.\sigma \\ \omega = D.\omega \end{array} \right\} \right) \\ &= \left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\delta \ \vee \ (D.\overline{\delta} \ \& \ (dist\ 1\ 2)) \\ \iota = \mathcal{G}l \\ \sigma = D.\sigma \\ \omega = D.\omega \end{array} \right\} \\ &= \begin{bmatrix} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \mathcal{G}n(\mathcal{S}p((dist\ 1\ (i+1)), X.\overline{\delta}), X.\delta) \\ \iota^t = \mathcal{G}l \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{bmatrix} \left(\left\{ \begin{array}{l} \tau = D.\tau \\ \delta = D.\delta \\ \iota = D.\iota \\ \sigma = D.\sigma \\ \omega = D.\omega \end{array} \right\} \right) \\ &= Dil(D,1) \end{aligned}$$

Nous constatons bien que la propagation des informations à travers la mini séquence précédente s'exprime uniquement en fonction de la donnée initiale D.

4.1.3 Exploitation pour une aide à l'utilisation du traitement d'images

La modélisation des opérateurs et des données que nous avons implantée peut permettre de fournir à un utilisateur trois niveaux d'aide [Dej96c] :

4.1.3.1 Assistance lors de la conception manuelle

La première aide consiste en un complément d'information sur les traitements réalisés, lors d'une conception manuelle d'un graphe d'opérateurs en fournissant à l'utilisateur des descriptions textuelles des données qu'il manipule. Il s'agit donc d'assister l'utilisateur d'un logiciel de traitement d'images lorsqu'il sélectionne des opérateurs, et de constituer un script résolvant son problème d'analyse ou d'interprétation d'image. L'utilisation de notre modèle permet de fournir, lors de l'élaboration du script, une source d'informations supplémentaire correspondant à la description de ce qui s'est produit lors de l'application de l'opérateur. En plus du script et des résultats de son exécution, l'utilisateur a ainsi accès à la description formelle des données obtenue par propagation des transformations le long de la chaîne et exprimée en termes d'indices visuels, ce que nous avons montré dans les paragraphes précédents. En cas d'erreur, il dispose ainsi d'indications pour ajuster les paramètres de l'opérateur ou remettre en cause sa sélection.

4.1.3.2 Aide à la conception d'applications

La seconde aide consiste à proposer une nouvelle méthodologie de conception mieux adaptée à l'utilisateur car privilégiant la spécification du problème à résoudre plutôt que la sélection des opérateurs. Cette possibilité de construire, par propagation, la description d'un résultat d'enchaînement peut être exploitée pour proposer une nouvelle méthodologie de conception beaucoup plus adaptée aux compétences de l'expert d'un domaine. Au lieu de lui présenter des opérateurs de traitement d'images, ce qui supposerait qu'il soit compétent en traitement d'images, l'interface -SATI [Cap94, Cap95] pour le système ExTI- lui procure des outils lui permettant de décrire son objectif à l'aide d'indices visuels basés sur la notion de mesure. Le système construit alors la chaîne d'opérateurs qui produit la même description. La construction d'une application se fait de façon incrémentale, l'utilisateur définissant et combinant progressivement les éléments de plus en plus complexes qui composent son domaine d'application. Il dispose d'un retour visuel affichant les éléments extraits de l'image. Les principales opérations disponibles sont la nomination de concepts, la généralisation et la spécialisation. Des outils spécifiques d'interaction (langage visuel) agissant dans plusieurs espaces (image, caractéristiques, description formelle), correspondant à plusieurs vues du même concept, permettent d'effectuer ces opérations dans la vue la plus pertinente pour manipuler le concept. Ils consistent en des interactions très explicites pour l'utilisateur, telles que la désignation d'exemples ou de contre-exemples ou la définition d'un domaine. Ils permettent également de garantir l'exactitude syntaxique et la pertinence sémantique de ces interactions, déchargeant l'utilisateur des contraintes spécifiques à l'implantation du traitement d'images et lui permettant de se concentrer sur l'objectif de son application. Enfin ce formalisme fournit également une justification "contractuelle" du résultat.

4.1.3.3 Aide à l'apprentissage

Enfin, il est possible de fournir une description précise des opérations pour l'apprentissage du traitement d'images [Dej96b]. Le spécialiste en traitement d'images, qui doit développer de nouvelles fonctionnalités, utiliser ou modifier un opérateur, doit avoir une information très fine sur l'effet des opérateurs de traitement d'images sur les divers types de données. Un opérateur d'une bibliothèque est généralement un macro-opérateur. Le formalisme proposé peut mettre en évidence la composition de fonctions élémentaires qu'il réalise. Il permet de concevoir des outils de visualisation des effets d'un opérateur sur chaque champ des données -trame, cache, index, signal, codage. Associée à la description des données, la description symbolique de l'opérateur sur ces cinq champs permet de développer des variantes d'un opérateur : l'utilisateur peut apporter une modification sur l'un de ces champs, simuler son utilisation dans une chaîne et ne le réaliser que s'il donne satisfaction. De même ce modèle permet de bien mettre en évidence les différences et les points communs de deux opérateurs implémentant une même fonctionnalité. Ces comparaisons peuvent être exprimées en termes d'opérations effectuées ou en termes d'effets produits sur les images. Nous pouvons ainsi appréhender l'utilisation de l'érosion du complémentaire du cache dans l'opération de dilatation d'une image binaire :

$$Dil(X,1) = Ou(X, Sll(Dst(Non(X)), 1, 2)$$
$$Dil(X,1) = Ou(X, Ero(Non(X)))$$

$$Ero(X,i) = \left(\begin{array}{l} \left[\begin{array}{l} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \mathcal{S}p((dist\ (i+1)\ min(\mathcal{T}(X.\tau))/2), X.\overline{\delta}) \\ \iota^t = \mathcal{G}l \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{array} \right], \\ \left[\begin{array}{l} \kappa^t = \mathcal{G}n(\mathcal{S}p((dist\ 1\ (i+1)), X.\overline{\delta}), X.\delta) \\ \iota^t = 0 \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{array} \right] \end{array} \right)$$

$$Dil(X,i) = \left(\begin{array}{l} \left[\begin{array}{l} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \mathcal{G}n(\mathcal{S}p((dist\ 1\ (i+1)), X.\overline{\delta}), X.\delta) \\ \iota^t = \mathcal{G}l \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{array} \right], \\ \left[\begin{array}{l} \kappa^t = \mathcal{S}p((dist\ (i+1)\ min(\mathcal{T}(X.\tau))/2), X.\overline{\delta}) \\ \iota^t = 0 \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{array} \right] \end{array} \right)$$

Tous ces informations peuvent alors être utilisées par un système didactique d'aide à l'apprentissage du traitement d'images. Ces différentes pistes doivent être exploitées, mais nous avons fourni un formalisme rigoureux qui permet de construire ces interfaces. La première version du Système d'Aide au Traitement d'Images -SATI- est un exemple de ce qui pourrait être proposé dans ce domaine.

4.2 Planification d'opérateurs

Dans le cadre de l'analyse d'images d'un domaine spécifique, nous désirons exécuter une chaîne de traitement permettant d'extraire d'une image des concepts de ce domaine. Cette chaîne est considérée comme une solution à un problème traduit comme un objectif d'interprétation. Elle peut donc être vue comme un plan dont on va contrôler l'exécution.

4.2.1 Génération de plans

Nous pouvons considérer la génération de ce plan de deux façons :

La première est effectuée à partir d'une bibliothèque plus ou moins bien décrite de gros opérateurs que l'expert essaie d'utiliser. Dans ce cas, l'élaboration de la chaîne et de ce fait les résultats qu'elle va fournir sont conditionnés par la connaissance de cet expert en matière de traitement d'images. Il existera donc une différence entre l'objectif à réaliser, relevant de concepts du domaine d'application, et l'entité effectivement atteinte par l'exécution de la séquence d'opérateurs ; cette différence étant plus ou moins grande selon la connaissance en traitement d'images de l'utilisateur d'une telle bibliothèque d'opérateurs. Il est alors nécessaire de mettre en place un processus d'enchaînement de ces opérateurs permettant de créer des squelettes de façon manuelle ou à l'aide d'un générateur. Ce dernier doit alors intégrer les descriptions souvent trop peu précises des opérateurs de cette bibliothèque ainsi que la description de l'objectif à atteindre ; ces descriptions devant être établies selon un même formalisme. Il nous faut ensuite adapter ces squelettes d'opérateurs aux conditions opératoires c'est à dire intégrer des mécanismes de contrôle d'exécution. Le problème qui se pose est alors de déterminer quelles connaissances pertinentes sont à intégrer dans ces bibliothèques et quelles connaissances de contrôle d'exécution sont à implémenter pour adapter ces opérateurs au domaine de l'application.

La seconde façon que nous implémentons dans le système ExTI [Ing92], consiste à considérer plus particulièrement l'objectif d'interprétation et à axer cette génération sur l'automatisation de la création de plans. La description de cet objectif dans le formalisme spécifique que nous venons de développer et la mise en place d'un système de résolution basé sur un ensemble d'opérateurs élémentaires dont la description symbolique est suffisamment fine pour rendre compte des transformations d'informations effectuées, permettent de concevoir des macro-opérateurs comme l'enchaînement de ces micro-opérateurs. Le problème qui se pose est alors de répartir la connaissance du traitement d'images entre le planifieur et le processus de supervision gérant l'exécution de ce plan.

Nous avons vu que pour pouvoir utiliser un programme de traitement d'images, diverses connaissances liées à l'utilisation des opérateurs qui le composent doivent être modélisées. Ainsi, selon l'utilisation qui en est faite, les modes de représentation des données ou des opérateurs vont varier : caractéristiques des résultats [Sch89], conditions d'application [Cle93], représentation en termes mathématiques [Sch92], par types de données [Mat90] par transformation d'information [Ing92], etc. Tous fournissent cependant des informations permettant, de façon plus ou moins automatique, la construction de chaînes complexes dans le système dans lequel ils sont implantés. Pour former un ensemble cohérent de tels programmes, la base de connaissance qui les intègre doit contenir les éléments nécessaires à la validation des résultats produits par ses opérateurs, ceux-ci pouvant être considérés indépendamment les uns des autres ou enchaînés pour former des macro-opérateurs et enrichir cette base.

4.2.2 Quel type de planifieur automatique?

Nous venons de montrer que les modèles de représentation des entités de traitement d'images que nous avons développé sous un formalisme particulier, permettaient de manipuler ces données en donnant un nouveau point de vue des transformations effectuées par les opérateurs et par delà par une séquence d'opérateurs. Pour essayer de déterminer les caractéristiques d'un système général de génération de plans dans lequel s'intègrent nos modèles, il nous faut établir une sorte de cahier des charges de ce planifieur en analysant le domaine particulier du traitement d'images dans lequel nous évoluons.
Remarque : Le prototype du planifieur utilisé faisant l'objet de la thèse de J.-M. Inglebert [Ing92], nous n'allons pas le présenter ici mais seulement le caractériser. Les éléments importants qui ont été rajoutés à ce système sont rapportés dans la troisième partie de ce mémoire.

4.2.2.1 Un objectif spécifique

L'objectif spécifique du système qui a été développé et qui intègre les entités de traitement d'images exprimées en terme d'indices visuels et de transformation de ces indices, est essentiellement de résoudre un problème de traitement d'images dans un domaine d'application défini en utilisant ses concepts et la connaissance de ses experts. De plus, afin d'exploiter les modèles et les caractéristiques que nous venons de voir, il doit permettre d'accéder à l'information liée aux noeuds du graphe d'opérateurs, tout au long du plan afin de développer un langage du domaine de l'application, de maîtriser l'*historique* du plan pour des réutilisations et de permettre d'établir des contrôles de ce plan au moment de son exécution.

4.2.2.2 Les caractéristiques générales

Le choix du planifieur est guidé par notre domaine de travail dans une approche par opérateur de transformations d'information. Les approches logiques (logique du premier ordre qui revient à une démonstration de formule) ou classiques (monotones, par défaut ou de type STRIPS [Fik71]) ne peuvent être applicables dans ce cas précis. En effet, les actions représentées dans de tels systèmes manipulent les données constituant le monde connu dans l'état initial alors que dans le traitement d'images les données sont *a priori* inconnues car créées au fur et à mesure des transformations. De plus, la description du monde à l'état $\mathcal{E}$ ne témoigne pas de l'*historique* des transformations opérées pour y parvenir.

Pour prendre en compte toutes ces contraintes il a donc été développé un procédé général de planification [Ing92], spécifique à notre objectif et constitué des éléments suivants :

Le monde :
Dans des approches classiques, les actions manipulent les objets constituant le monde connu dans l'état initial. Or en traitement d'images, le monde, constitué des images et des résultats de mesures effectuées, est construit tout au long de la phase d'exécution -réelle ou simulée- de la chaîne et seuls les opérateurs manipulent les données. De ce fait, la gestion de l'évolution de l'environnement par des événements extérieurs n'est pas à prendre en compte (SPEEDY [Bas96]).

La représentation du monde :
Certains systèmes de planification utilisent un modèle d'opérateurs mettant en jeu des pré-conditions ou post-conditions d'utilisation (STRIPS [Fik71] et ses descendants RSTRIPS, ABSTRIPS). Ceci est nécessaire lorsqu'on souhaite diriger la création du plan en introduisant de la connaissance sur les opérateurs pour garantir la cohérence du plan. Comme nous l'avons vu précédemment, chaque "état" du monde dans lequel nous évoluons est décrit dans un langage précis, en termes d'indices visuels. Les opérateurs qui manipulent ces états n'ont alors pas de pré-condition ou post-condition concernant leur utilisation car c'est au moment même de la planification que l'on vérifie si l'opérateur va produire l'effet voulu grâce à la description précise de l'objet créé ; cette description pouvant être comparée à l'objectif.

La recherche d'une solution :
Trois types de recherches sont couramment utilisées. La première est la recherche dans un espace d'états : recherche en avant (application des opérateurs de l'état initial vers l'état objectif), recherche en arrière ou mixte. La seconde est la recherche dans un espace de sous-problèmes dans lequel l'objectif est décomposé en un arbre ET/OU de sous-objectifs. Enfin la troisième est la recherche dans un espace de plans (NOAH [Sac75]) décomposant lui aussi l'objectif en sous-problèmes réalisés par des plans déjà construits -problème de reconnaissance de plans, planification non-linéaire (ABTWEAK [Yan90]). Encore une fois le traitement d'images pose ses propres contraintes car les décompositions d'objectifs en sous-objectifs ne sont généralement pas simples à mettre en œuvre, l'état courant étant composé d'un unique élément : une image ou une mesure sur l'image.

La stratégie :
Nous employons donc une recherche en avant et en profondeur d'abord mettant en œuvre un module de filtrage permettant de constater la différence entre la donnée produite et l'objectif (GPS [Win77]) afin de déterminer le point d'arrêt. Cette différence ne peut être calculée que grâce à la mise en œuvre de règles de simplifications d'expressions garantissant une forme canonique de celles-ci.
De plus, un algorithme de réutilisation de plans a été développé afin d'optimiser la recherche de la solution en utilisant la notion de vocabulaire du domaine -une expression d'objectif est attachée à un terme du domaine d'application (voir [Dej96d]).

Le contrôle d'exécution :
Le contrôle d'exécution dépend essentiellement du degré de représentation des entités manipulées. Une étude sur ce problème est menée au paragraphe 4.3 et développe les caractéristiques du contrôle selon les niveaux de fiabilité du résultat de la planification.

4.2.3 Caractéristiques du planifieur réalisé

Nous avons donc développé un planifieur spécifique à notre objectif. C'est un système de planification linéaire d'opérateurs de traitement d'images, sans retour ("backtracking") dans l'arborescence grâce à un processus de conservation des opérateurs instanciés applicables aux données dont les descriptions complètes sont toujours présentes et accessibles. Ce processus permet d'éviter l'utilisation d'heuristiques (savoir-faire en utilisation du traitement d'images), et les inconvénients d'une explosion combinatoire du fait de l'utilisation de règles de simplification et de mesure de différence entre descriptions. De cette façon, un plan peut être vu comme la trace de la propagation des transformations contenues dans les diverses données, qu'elles figurent à un niveau intermédiaire, ou terminal -solution ou même échec. Cette planification permet entre autres, et cela pour chaque nœud du graphe, de connaître la description en termes d'entités de traitement d'images de l'information qu'il renferme. Outre ces éléments déjà présents dans [Ing92], nous pouvons considérer deux notions que nous développerons plus tard, mais dont il est intéressant de parler ici pour compléter cette étude des caractéristiques du planifieur : les notions de base minimale et de réutilisation de plan.

4.2.3.1 Une base minimale

Certains enchaînements pouvant être reconnus comme des opérateurs existant -que nous appellerons macro-opérateurs-, comme nous l'avons vu dans un paragraphe précédent, la base des opérateurs peut être considérée comme minimale. Elle est alors formée des seuls opérateurs élémentaires, non exprimables par une séquence d'opérateurs de cette même base. L'introduction d'un opérateur dans cette base peut donc être faite dans la mesure où il n'a pu être décrit par une telle séquence ou qu'il entre dans la décomposition d'un nouveau macro-opérateur.

Nous avons donc décidé d'enlever ces opérateurs de la base initiale pour pouvoir les gérer de façon plus intéressante dans la phase de sélection de plan. Celui-ci étant ainsi généré à l'aide des seuls opérateurs élémentaires, la séquence développant un macro-opérateur peut être reconnue au moment de l'exécution et remplacée par cet opérateur. Dans le chapitre 1 de la troisième partie, nous développons l'implémentation des opérateurs et les principes de remplacement des macro-opérateurs.

4.2.3.2 Une réutilisation de plans

Si la base d'opérateurs est minimale, il est alors possible de développer un ensemble de commandes correspondant à des plans exécutables résolvant des objectifs d'un domaine particulier. Ce vocabulaire exécutable spécifique au domaine, est cette fois-ci directement lié à un plan résolvant l'objectif nommé. Il viendra enrichir un ensemble d'opérateurs plus spécifique à ce domaine, contenant les opérateurs de la base précédente, les commandes exécutables ainsi qu'un ensemble de fonctions génériques de description visuelle d'images traitant par exemple de caractéristiques morphologiques, spatiales, etc, des objets. La création des commandes est établie par planification de descriptions de concepts de plus en plus complexes. Leur ré-utilisation s'effectuera par un processus complet de construction de *sous*-plan lors de la planification d'un objectif les utilisant.

Ceci peut être rapproché des structures utilisées dans des systèmes implémentant des modules de planification hiérarchiques et notamment VISIPLAN [Gon95] qui manipule ce type de notion d'objets liés cette fois à des ensembles de règles expertes. Mais les algorithmes classiques ne peuvent être appliqués dans notre approche. En effet, la simplification des expressions de description d'objectif, nécessaire pour éviter une détermination trop complexe de la solution, ainsi que les propriétés du traitement d'images empêchent toute hypothèse sur la protection des sous-buts [Kno94, Kno95] telle qu'elle apparaît dans les algorithmes non linéaires tels SNLP, NONLIN ou TWEAK et tout traitement de ces sous-buts de façon conjonctive ([Jos90],[Cha87b]).

Le vocabulaire exécutable -quadruplet (nom, objectif, plan, programme)- aide à l'écriture d'un objectif en ré-utilisant uniquement le nom attribué au concept décrit. Cette écriture est réalisée à l'aide du formalisme en termes d'indices visuels mais aussi en ré-utilisant les éléments du vocabulaire. La fusion de plans entraînera alors un processus de reformulation de l'objectif principal en appliquant des formules de simplification lors de l'intégration des objectifs associés au vocable utilisé.

Un algorithme spécifique du planifieur assure (voir sect.1.2.2, 3e partie) la fusion maximale des plans ré-utilisés -appelés *sous*-plans- en considérant leurs nœuds comme des nœuds de fusion éventuels du plan général. À ces nœuds de fusion correspondent des données renfermant des informations pertinentes dans le domaine de l'application.

4.2.4 Validation des séquences planifiées

La validation des programmes -séquence d'opérateurs- d'interprétation d'images, c'est à dire l'estimation de leur capacité à recouvrir un concept du domaine d'application, dépendra de la cohérence et de la bonne description des opérateurs dans le modèle de représentation choisi. Le problème de la validation d'un objectif révèle une caractéristique propre au domaine de l'image. L'analyse d'images consiste en effet à extraire des informations d'une image et à les interpréter dans le cadre d'un modèle établi. La question qui se pose alors est de savoir dans quel espace de travail se situe cette modélisation : est-ce dans le domaine des objets de la scène vue ou dans l'image qui est faite de cette scène ? Généralement, le processus d'interprétation exploite les deux, en faisant intervenir des notions spécifiques du domaine d'application et en recherchant dans l'image de la scène les éléments permettant d'exprimer ces concepts. Ces éléments sont alors fonction des conditions opératoires et, hors de ce contexte domaine-image, peuvent être interprétés différemment. Pour énoncer plus simplement cette ambiguïté, il faut considérer une image comme une représentation par un tableau de pixels de la scène vue dépendant de conditions opératoires de création.

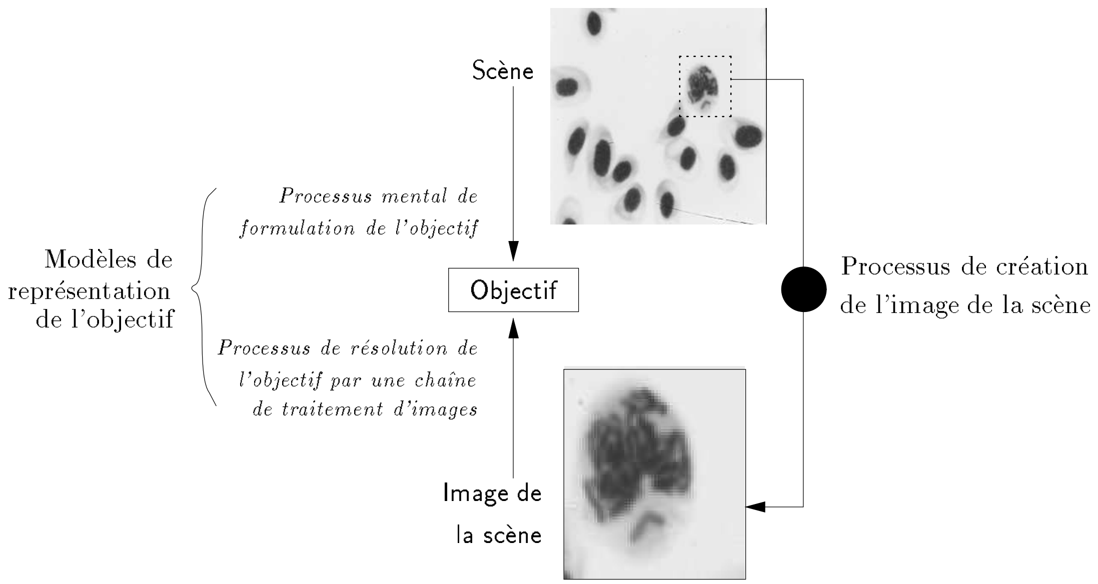


FIG. 4.5 - *Analyse d'image*

Une chaîne de traitement d'image répondant à un objectif d'interprétation, quel qu'il soit, formulé mentalement tentera d'inverser le processus de création de cette image en mettant en évidence certains objets de la scène (fig.4.5). La présence dans le processus d'analyse de deux modèles -modèle des entités réelles, vues, et modèle de ces mêmes entités dans l'image- et leur éventuelle mauvaise mise en correspondance peuvent être la source d'erreur d'extraction d'entité.

Ceci ne peut être généralisé à tous les domaines de traitement de signal, mais il est important de bien déterminer l'espace de représentation des entités manipulées. En effet, ce phénomène devra être pris en compte lors de la détermination de la responsabilité de l'échec de l'exécution de la chaîne de traitement afin de caractériser les méthodes de contrôle d'exécution à mettre en œuvre. Le langage LEO permet de caractériser un tel espace de représentation, en fournissant une description précise de chaque donnée générée le long de la chaîne d'opérateurs.

Si nous prenons l'exemple consistant à déterminer les points marqués après un lancé de dés (fig.4.6), la description suivante de l'objectif permet de préciser correctement l'ensemble des configurations pouvant être traitées par le problème tel qu'il a été posé :

```
points_marques.τ = '()
points_marques.κ = 't
points_marques.ι = '(gl)
points_marques.σ = '(nbcc ((vois8)((et (((nivg 0 200)))((dans ((vois4)(((eul -262144 -5 1 262144)
      ((vois4)(((nivg 200 256))))) ((nivg 0 200)) (ninter ((vois4)(((eul -5 1) ((vois4)
      (((nivg 200 256)))))))((cercle 241 241 210))))) ((inter ((vois4)(((eul -5 1)((vois4)
      (((nivg 200 256))))))) ((cercle 241 241 210))))))))))
points_marques.ω = '(0 131073)
```

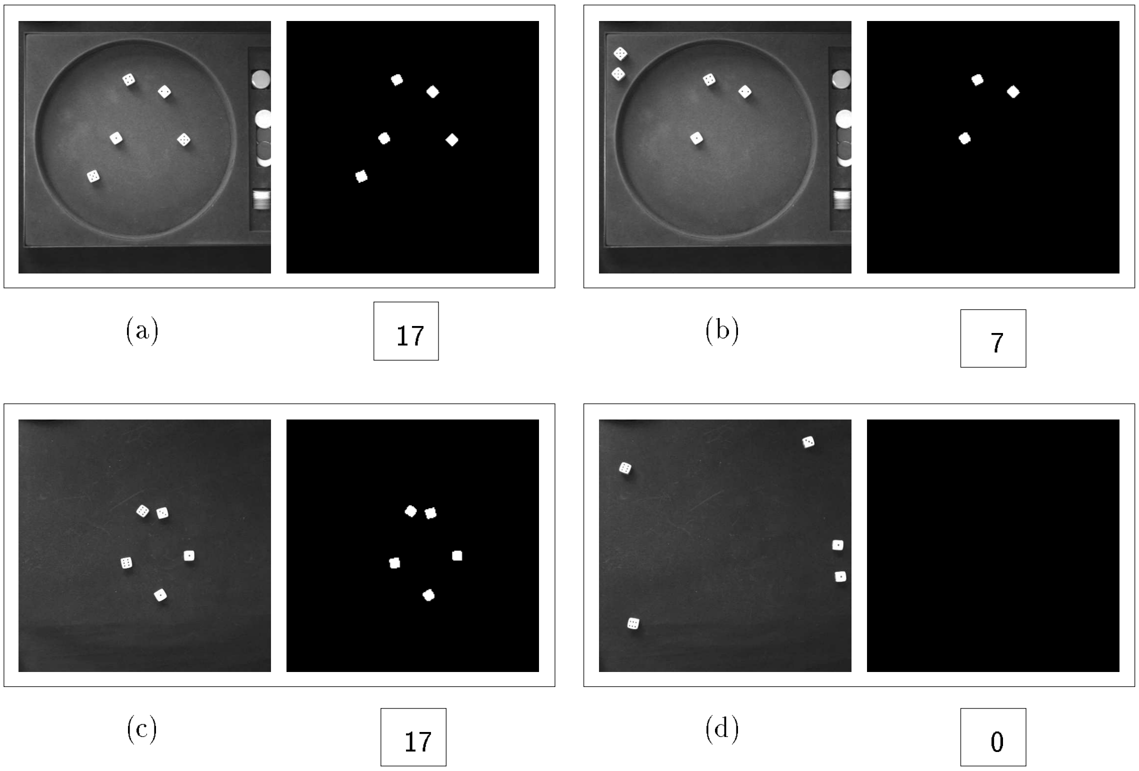


FIG. 4.6 - *Validation d'objectif - dés dans une piste*

Les dés sont ici considérés comme des zones claires `(nivg 200 256)` à "trous" `((eul -5 1)` ((vois4)(((nivg 200 256))))). La présence de la description du disque dans lequel vont être considérés les dés va limiter la zone de recherche de ceux-ci au cercle de centre (241,241) et de rayon 210. Ces dés seront alors décrits par (inter ((vois4) (((eul -5 1) ((vois4) (((nivg 200 256)))))) ((cercle 241 241 210))). De ce fait, dans l'image (b), les dés placés sur le côté, hors de la piste, ne seront pas pris en compte. Les points sont alors des zones non claires (nivg 0 200) se trouvant dans les dés pleins.

Par la validation et la compréhension des informations manipulées par le graphe d'opérateurs, l'objectif dont ce graphe est solution fournit les éléments nécessaire à sa réutilisation totale ou partielle pour l'élaboration d'autres problèmes. Ainsi, l'absence de description de la piste dans des termes faisant référence à des entités à extraire de l'image permet de faire le comptage des points à l'aide du même traitement dans des situations pour lesquelles cette même piste est, cette fois, virtuellement présente (c) et (d).

Enfin, le retrait de cette description de la piste permettra alors d'intégrer, dans l'espace de représentation ainsi défini, les entités *dés* préalablement écartées. Cet objectif, une fois simplifié est alors :

```
points_marques.τ = '()
points_marques.κ = 't
points_marques.ι = '(gl)
points_marques.σ = '(nbcc ((vois8)((et (((nivg 0 200)))((dans ((vois4)(((eul -262144 -5 1 262144)
      ((vois4)(((nivg 200 256))))) ((nivg 0 200))))(((eul -5 1)((vois4) (((nivg 200 256)))))))))))
points_marques.ω = '(0 131073)
```

Une fois planifiées, ces chaînes de traitement d'images sont exécutées. Contrairement à d'autres domaines -tels que le pilotage de robots- intégrant des modules de planification en environnement dynamique, le traitement d'images offre l'avantage de fournir des données stables, c'est à dire non modifiables au cours du processus d'exécution par les interventions d'entités distinctes des opérateurs qui les génèrent. Du fait de cette stabilité des données et des opérateurs de la bibliothèque de traitement d'images utilisée, des exécutions répétées de la même chaîne sur les mêmes données d'entrées fourniront les mêmes données de sorties. Cette stabilité dans la génération de résultats peut être exploitée pour confirmer la validation de la chaîne et fournir à un non-expert des programmes de traitement d'images spécifiques à son domaine. En effet, comme nous venons de le voir, la description en termes d'indices visuels de l'objectif à atteindre détermine à la fois sa portée -cas particuliers des objectifs résolus- et ses restrictions ce qui peut être considéré comme une preuve du résultat de l'exécution, garanti par la stabilité des résultats validés par l'utilisateur ou par un système d'analyse. Ainsi, les programmes développés sont décrits en termes d'indices visuels, prouvés par des validations sur des ensembles de données tests, et répondent à des objectifs précis d'inspection ou de traitement. Ce couple (objectif, programme) associé au plan symbolique générant la chaîne d'opérateurs du programme peut être étiqueté par un nom du langage de l'application pour fournir un élément de ce que nous appelons vocabulaire exécutable du domaine.

4.3 De la planification à l'exécution

Un programme de traitement d'images peut être considéré comme un plan -script- combinant des opérateurs de traitement d'images. Pour prendre en compte la variété des situations réelles, les conditions opératoires, le bruit, l'incomplétude et l'incertitude des données, ce plan devra être adaptable -s'adapter aux données- et ajustable pour être modifié si son exécution ne fournit pas les résultats attendus.

La phase d'adaptation consiste d'une part à intégrer des connaissances sur le contexte de l'acquisition des données manipulées et d'autre part à faire des mesures sur les résultats intermédiaires. Des traitements spécifiques relevant de processus d'ajustements de paramètres ou de choix d'opérateurs dans une même fonctionnalité peuvent être alors effectués. Le processus le plus complexe consiste à re-planifier des parties du plan au vu de ces résultats, c'est à dire à tenir compte de modifications dans les descriptions des concepts manipulés, comme nous l'avons vu précédemment dans l'exemple des dés (voir 4.2.4). En conséquence le plan lui-même est modifié car répondant à un objectif différent.

Les techniques de contrôle d'exécution de plans implémentent des mesures particulières afin de calculer les valeurs des paramètres des opérateurs. Ces valeurs seront déterminées en prenant en compte des informations caractérisant les résultats des opérateurs ou des séquences considérées par la supervision. De façon générale, on peut considérer le contrôle d'exécution d'un plan généré par un système de résolution d'objectif de deux façons différentes :

1. Comme le passage d'un système de prédiction -plan générique- à un système d'exécution -plan instancié- (fig.4.7). Cette généricité du plan impose des contraintes sur les valeurs retournées. Ainsi l'ensemble de ces valeurs appliquées au plan formeront alors une instance de ce plan résolvant l'objectif.

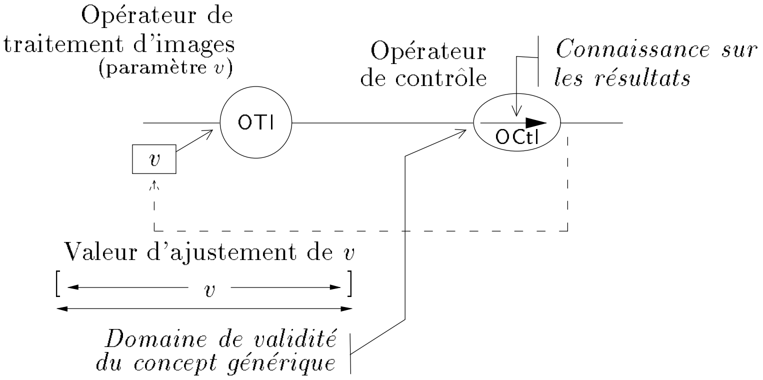


FIG. 4.7 - *Instanciation de plans génériques*

2. Comme le passage à une forme adaptée du plan généré (fig.4.8). Dans ce cas, les mesures effectuées en cours d'exécution ne sont pas contraintes et amènent à une généralisation ou spécialisation du concept utilisé dans la formulation de l'objectif.

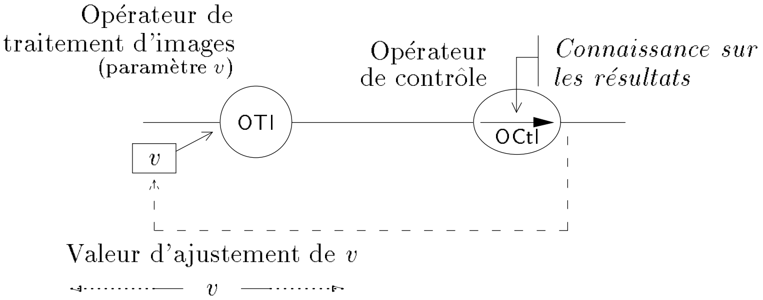


FIG. 4.8 - *Adaptation de plan*

4.3.1 Articulation Planification / Supervision / Contrôle

Nous pouvons établir une distinction entre la supervision de ce plan et le contrôle de son exécution. La supervision consiste en la gestion de la cohérence du plan par rapport aux données créées au cours de son exécution. Elle met en jeu des connaissances particulières sur l'utilisation de structures permettant d'effectuer des re-planifications de sous-objectifs ou des ajustements de plans ainsi que des processus gérant des boucles itératives et des choix de séquences d'opérateurs dans le cas de correction et de prise en compte de conditions opératoires.

Le contrôle d'exécution effectue quant à lui les mesures nécessaires pour valider les données produites par l'exécution d'une action du plan ou pour fournir au processus de supervision les éléments nécessaires à l'établissement d'une gestion complète et performante. Ceci passe par le contrôle de points d'arrêts pour des mesures spécifiques permettant de fournir des propositions d'instanciations possibles lors de choix d'opérateurs pour une fonctionnalité donnée ou lors d'ajustement de leurs paramètres.

L'exécution et la mise au point de méthodes ou de modules de contrôle peuvent se concevoir schématiquement de trois façons différentes selon la précision de l'objectif et la nature du planifieur, c'est à dire selon la généricité du plan lui-même. Cette précision pose ainsi le problème de la coopération entre planification, supervision et contrôle d'exécution proprement dit. Il existe une continuité dans l'implémentation de ces aspects de la gestion de l'exécution du plan, depuis une exécution totalement contrôlée jusqu'à une exécution totalement automatique. Mais la distinction qui est faite dans les trois modes de coopération suivants permet de montrer les rôles des processus mis en jeu.

1. *Pas de prédiction : planification et exécution au fur et à mesure*
   Ce procédé oblige à mettre en œuvre un puissant système de supervision de plan (fig.4.9) permettant de gérer *a priori* des entités (données, opérateurs) incertaines ou incomplètes. Cette supervision du plan par l'intermédiaire de mesures effectuées par des modules de contrôle sur les données permettra de générer et d'exécuter pas à pas des plans de résolution d'objectif.

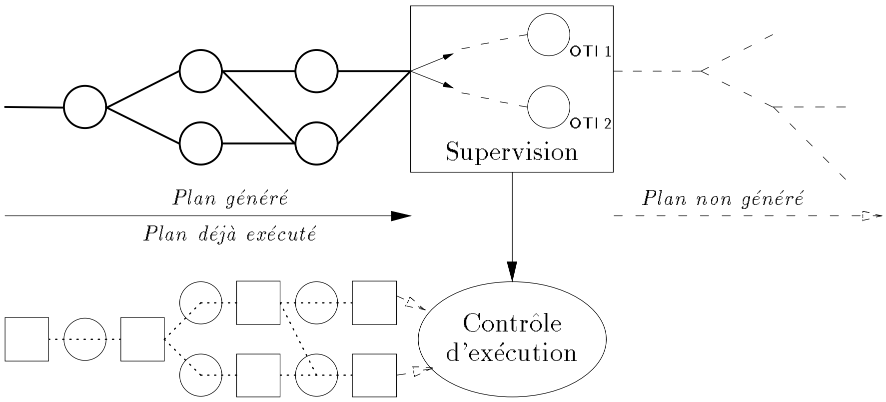


FIG. 4.9 - *Pas de prédiction*

2. *Prédiction partielle : planification, ajustement, validation des étapes*
   Ce procédé impose de mettre en œuvre un puissant système de contrôle d'exécution de plan (fig.4.10) permettant de fournir des ajustements précis des opérateurs en fonction des requêtes établies par la supervision du plan, de mesures effectuées sur les données produites et sur les conditions opératoires (mesures sur les données d'entrée).
   Le système de planification devra implémenter des connaissances sur les opérateurs de traitement de la base, des connaissances sur l'objectif à atteindre ainsi que des connaissances sur l'intégration de modules de contrôle dans le plan qui sera alors complètement exécutable et autonome dans le sens d'une non intervention extérieure au couple (plan,superviseur).
3. *Prédiction totale : planification et exécution sans contrôle.*
   Ce procédé (fig.4.11) doit mettre en œuvre des modèles de représentation des entités mani-

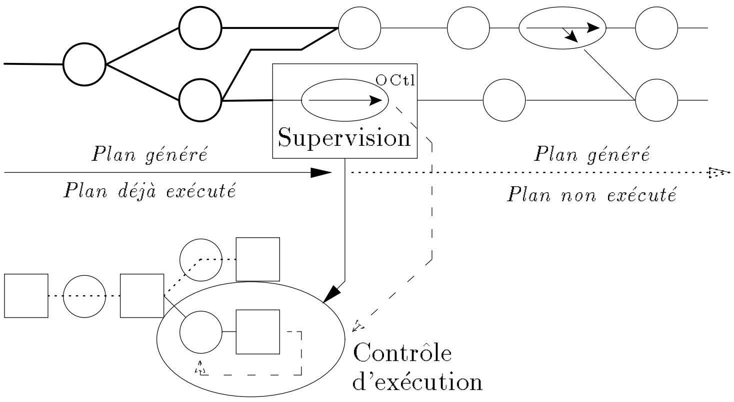


FIG. 4.10 - *Prédiction partielle*

pulées par le planifieur (objectifs, données, opérateurs) suffisamment précis et détaillés pour permettre de construire des chaînes d'opérateurs dont l'exécution directe fournira le résultat attendu.

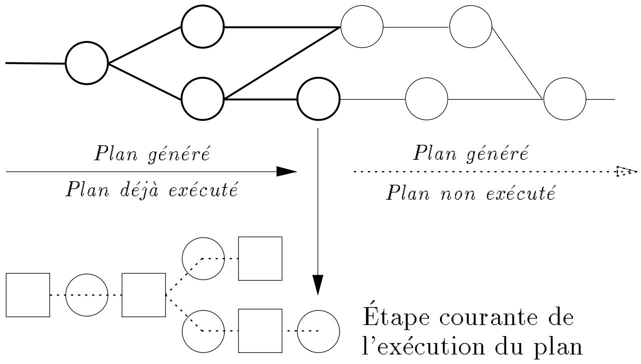


FIG. 4.11 - *Prédiction totale*

4.3.2 Deux modes de supervision

Dans [Hen90] J.Hendler présente les techniques les plus utilisées dans des systèmes de supervision et de contrôles d'exécution de plans. Dans [Tho93], M.Thonnat classe ces mêmes techniques de contrôle d'exécution en quatre ensembles :

1. intégration, dans le plan, d'actions d'attentes de la configuration prévue,
2. intégration, dans le plan, d'actions de tests et de correction,
3. intégration, dans le plan, d'actions de tests et d'alternatives,
4. intégration, dans le processus de planification, de processus de contrôle d'exécution.

Nous pouvons en fait les rassembler en deux groupes en faisant porter la responsabilité de l'échec de la résolution de l'objectif, soit sur l'exécution du plan, soit sur le planifieur lui-même. À ces deux groupes nous en ajoutons un troisième mettant en cause l'ambiguïté entre l'objet vu -représenté dans le modèle de la scène- et l'objet analysé -représenté dans le modèle de l'image- lors de l'expression de l'objectif planifié. Ces trois groupes reflètent les trois grandes raisons de l'échec de l'exécution d'un plan de résolution d'objectif par rapport à l'objectif à atteindre, de par la difficulté d'obtenir une solution totalement planifiée : mauvais modèle pour l'expression de l'objectif, mauvaise prise en compte des conditions opératoires ou mauvaise description des opérateurs de traitements.

4.3.2.1 Supervision par attente de la configuration prévue

Lorsqu'une mesure sur la donnée produite par une action met en évidence une différence entre la prédiction -résultat attendu- et ce résultat effectif, un premier type de supervision consiste à marquer un arrêt dans l'exécution du plan. Ce point d'arrêt peut être utilisé :

- pour attendre une configuration favorable du monde et poursuivre alors l'exécution. L'état du monde doit alors être modifié par une intervention externe à la base d'actions planifiables. Il lui faudra, par exemple attendre qu'un objet mobile soit complètement entré dans le champ d'observation pour l'analyser.
- pour atteindre une configuration correspondant à la prédiction en déclenchant des processus de mise au point de paramètres par itérations, sélections de plan, etc. Il lui faudra, par exemple, modifier un paramètre de segmentation pour obtenir une correspondance entre le nombre d'éléments extraits et le nombre, supposé connu, d'objets à analyser.

Dans ce cas les connaissances à intégrer concernent plus particulièrement l'influence que peut avoir la modification d'un élément du plan (paramètre, opérateur, ...) sur les données générées. Celles-ci doivent donc faire l'objet de mesures spécifiques reliant les sous-chaînes, les opérateurs ou les paramètres aux transformations qu'ils produisent sur les données.

4.3.2.2 Supervision par Re-planification

Dans le cas où la différence entre la prédiction et les résultats obtenus à l'exécution ne peut être traitée simplement (par ajustements, ...) la re-planification permet de déterminer une séquence d'actions à régénérer et exécuter. Généralement, ceci est dû à une mauvaise adéquation entre le modèle et la donnée ou à la formulation de l'objectif qui ne correspond pas exactement au concept. La re-planification peut aussi être exploitée lorsqu'on ne souhaite pas faire de planification alternative lors de la conception pour des raisons de coût (connaissance d'un faible pourcentage de cas rencontrés, ...-fig.4.12). La re-planification peut aussi être utilisée parce que les informations nécessaires ne seront disponibles qu'à l'exécution. Il sera possible de revenir à des étapes antérieures dans la chaîne d'opérateurs et de re-planifier à partir d'un point déterminé. Les connaissances à implémenter dans ce cas doivent permettre de déterminer la ou les séquences de plan à régénérer -l'utilisation de techniques comme la planification hiérarchique peut grandement aider à cette détermination. Il faut donc intégrer et mettre en relation des modèles de représentation d'opérateurs et de données permettant d'examiner ces dernières de telle sorte que l'analyse effectuée puisse indiquer, dans le plan lui-même, le point de départ et le sous-objectif de re-planification.

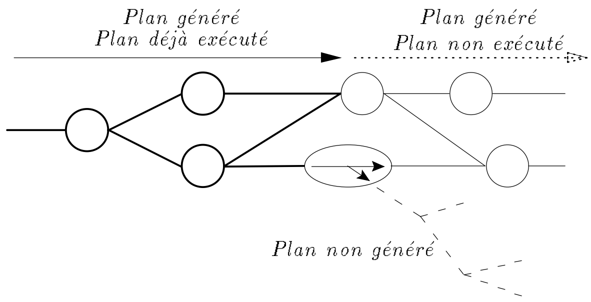


FIG. 4.12 - *Plans alternatifs non générés*

4.3.3 Représentation de la connaissance en traitement d'images

Dans notre approche de la planification d'opérateurs de traitement d'images, implémentée dans ExTI, les modèles utilisés que nous venons de développer -[Dej96e, Dej96d, Dej96f]- permettent de décrire les données produites et l'objectif à atteindre dans un formalisme spécifique introduisant la notion d'indices visuels : entités vues, mesurables, dans l'image de la scène.
Dans ce système, les opérateurs de la base sont considérés comme des outils de transformation d'informations et sont décrits dans les même termes que les données. La base ainsi formée ne contient que des opérateurs élémentaires ne pouvant pas être remis en cause lors de l'exécution du plan généré. Ainsi, l'échec de cette exécution peut porter sur une mauvaise définition du modèle -modèle mental et visuel, établi par l'utilisateur, des objets à reconnaître ou généricité insuffisante- ou à une mauvaise formulation de l'objectif c'est à dire sa traduction en termes d'indices visuels pouvant provenir d'une mauvaise appréciation du processus de formation de l'image -prise en compte des conditions opératoires. Enfin, si une chaîne d'opérateurs élémentaires -cette chaîne forme alors un macro-opérateur symbolique- est implémentée par un opérateur réel d'une bibliothèque, les erreurs peuvent provenir de la non concordance : opérateur symbolique / opérateur réel.

4.3.4 Réutilisation de programmes

Dans notre approche de résolution d'objectifs, la bibliothèque utilisée contient des opérateurs symboliques élémentaires décrits dans le formalisme choisi. Les résultats fournis sont des scripts d'opérateurs réels de traitement d'images associés à des plans d'opérateurs symboliques. Utiliser des macro-opérateurs d'une bibliothèque de programmes de traitement d'images pose alors le problème de la description de ces macro-opérateurs dans le même formalisme.

Si cette description ne peut-être effectuée de façon correcte, il est toutefois possible de générer un plan à l'aide des opérateurs symboliques de notre base. Ce dernier est approximatif dans le sens où les opérateurs qu'il utilise ressemblent, du point de vue des fonctionnalités mises en jeu, aux macro-opérateurs qu'ils sont sensés remplacer. Le travail consiste donc à essayer de mettre en correspondance les résultats obtenus par le script avec ceux obtenus par l'application des macro-opérateurs examinés. Deux possibilités peuvent alors être envisagées au niveau du contrôle d'exécution pour tenir compte de ces différences de résultats.

Dans le cas où le script correspond à un programme de la bibliothèque, on peut décider que le plan associé ne pourra pas être coupé, c'est à dire que lors de sa ré-utilisation, ses nœuds de fusion ne devront pas être pris en considération. Nous obtenons ainsi un nouvel élément pour notre base symbolique composé d'un plan non sécable traduisant les transformations effectuées par un programme de traitement d'images (fig.4.13). Son exécution se fera alors sans contrôle particulier comme tout autre opérateur de la base.

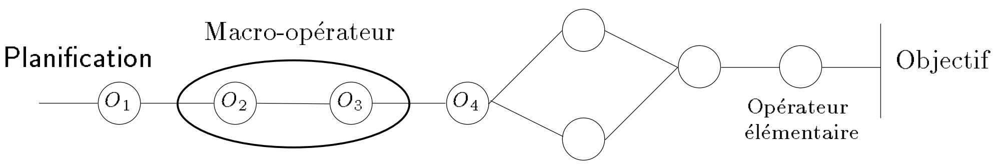


FIG. 4.13 - *Réutilisation de programme sans contrôle particulier*

Dans le cas où le script d'opérateurs élémentaires produit des résultats distincts du programme à ajouter dans la base (fig.4.14), l'intégration de celui-ci oblige à mettre en place une coopération étroite entre une description des transformations d'informations déterminées par le plan et une description du contrôle de son exécution.

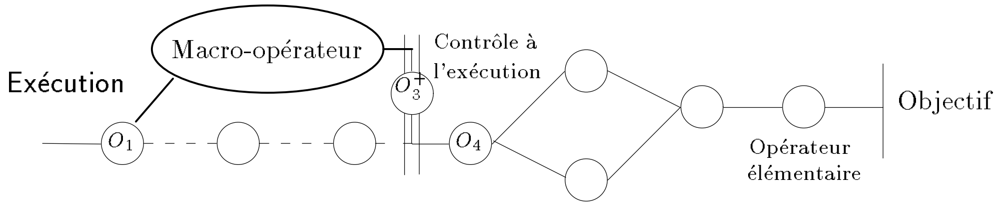


FIG. 4.14 - *Réutilisation de programme avec contrôle à l'exécution*

La notion de vocabulaire exécutable d'un domaine spécifique implanté dans le système ExTI, associant un concept du domaine de l'application, sa description en termes d'indices visuels, son plan d'opérateurs symboliques et son script associé, est utilisable pour construire ou intégrer une bibliothèque de programmes complexes de traitement d'images. En effet, lorsqu'une différence existe entre le résultat du programme à intégrer et la solution proposée par l'exécution du script associé au plan, deux cas peuvent être distingués pour l'intégration de ce programme dans la bibliothèque, selon que cette différence peut être décrite ou mesurée lors de son évaluation. Ainsi, dans le cas où la différence entre ces résultats peut être décrite dans le formalisme utilisé, nous obtenons alors la description des transformations à effectuer pour "*corriger*" directement la description du programme à ré-utiliser et pour ajuster l'objectif atteint (fig.4.15) ; nous obtenons alors une description complète du programme à réutiliser, sans contrôle particulier.

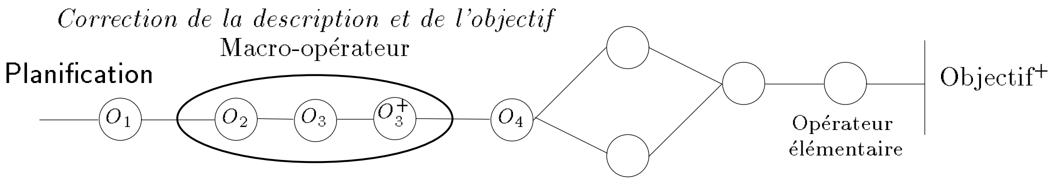


FIG. 4.15 - *Intégration de programme par correction de description*

Dans le cas où cette description ne peut être exprimée, la mesure mise en œuvre pour ajuster les résultats de l'exécution permettent de développer un module de contrôle d'exécution spécifique, associé à ce programme. Lors de l'exécution de la chaîne dans laquelle il sera ré-utilisé, ce module ajustera les résultats de cet opérateur (fig.4.16). Il s'agit là d'un contrôle d'exécution particulier, lié à l'opérateur.

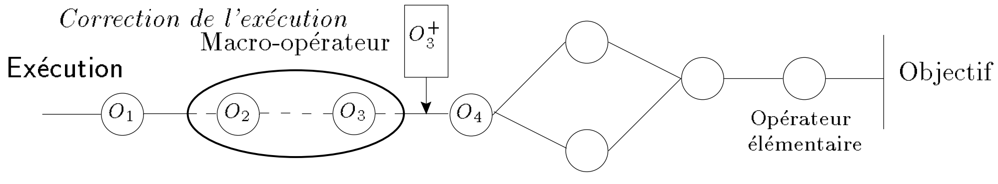


FIG. 4.16 - *Intégration de programme avec correction de l'exécution*

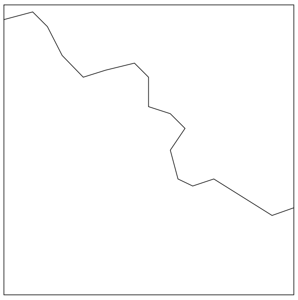
Une connaissance sur la ré-utilisation de ces programmes doit toutefois être implémentée pour permettre au processus de génération de plan d'intégrer des séquences particulières destinées au contrôle de son exécution. Cette connaissance dépendra de la base d'opérateurs élémentaires décrite et des plans associés à l'utilisation de ces programmes. Elle sera donc évolutive jusqu'à une éventuelle complétude et intégration dans la base d'opérateurs symboliques.

4.3.5 Un formalisme bien adapté

Les éléments que nous venons d'évoquer ne sont pas actuellement implémentés dans le système que nous maintenons. Cependant, le formalisme rigoureux des modèles présentés qui constitue une expression de la connaissance du traitement d'images permet et facilite l'intégration de cette supervision de plan. En effet, LEO permet d'exprimer un objectif de traitement d'images, mais est également utilisé pour décrire les données intermédiaires planifiées et générées. Comme nous l'avons montré précédemment, les expressions du langage que nous manipulons peuvent être représentées sous la forme d'un graphe conceptuel. Cette représentation va nous permettre de montrer plus facilement comment les mécanismes de supervisions proposés ci-dessus peuvent être liés à la représentation des objectifs, et donc comment introduire le contrôle d'exécution en exprimant les mesures à effectuer dans le langage LEO. Les processus de contrôle d'exécution sont nombreux, mais nous pouvons illustrer comment les descriptions des résultats intermédiaires peuvent faciliter leur intégration en prenant des exemples simples. La mise en évidence de la rivière de l'image suivante (fig.4.17), est déterminée par l'objectif `riviere_2` dont le graphe conceptuel est développé dans la figure 4.18.

Image

Représentation mentale de l'objectif



Plan généré par ExTI

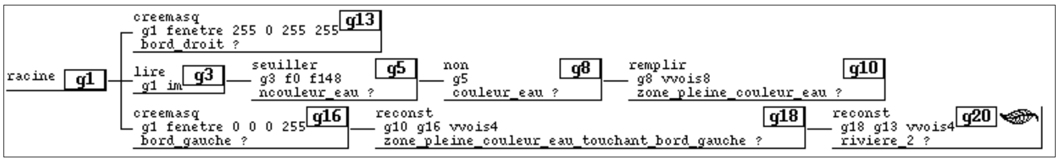


FIG. 4.17 - *Image et objectif* `riviere_2`

```
riviere_2.τ = '((0 256) (0 256))
riviere_2.κ = '((inter ((vois4)((inter ((vois4)(((nivg 148 256)) (dans ((vois8)(((nivg 0 148))))
      (((nivg 148 256)))))) ((fenetre 0 0 0 255))))) ((fenetre 255 0 255 255))))
riviere_2.ι = '(gl)
riviere_2.σ = 'vab
riviere_2.ω = '(1 2)
```

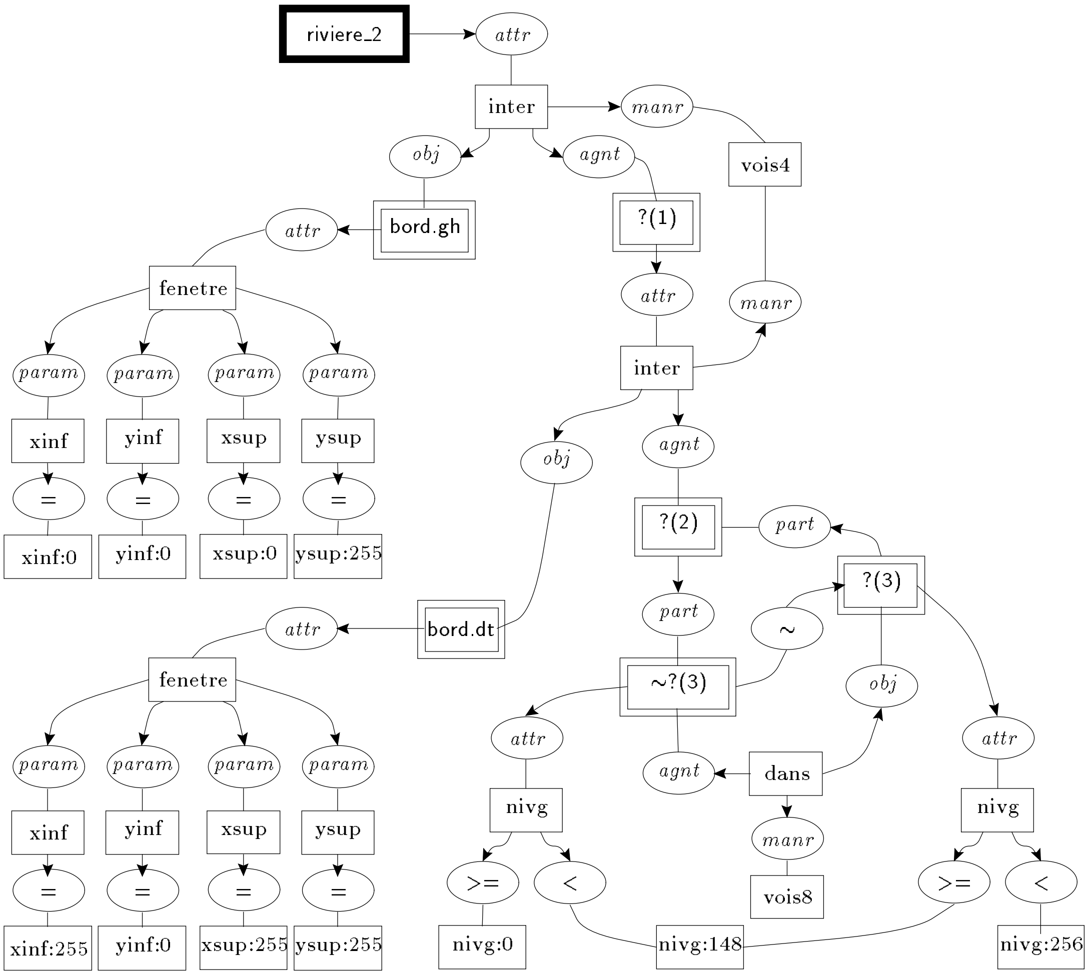


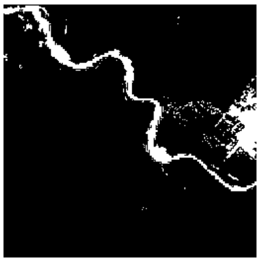
FIG. 4.18 - *Graphe conceptuel de* `riviere_2`

Le script permettant l'exécution de ce plan est le suivant :

```
#! /bin/csh
clear
echo SCRIPT MExTI
echo ===========
echo Image initiale: $1
rm -f *.dsc
echo: FS_LIRE $1 g3
cp $1 g3
echo: FS_SEUILLER g3 0 148 g5
oseuiller g3 g5 0 148
echo: FS_NON g5 g8
onot g5 g8
```

```
echo: FS_REMPLIR 2 g8 g10
oremplir 2 g8 g10
echo: FS_CREEMASQ g1 "(fenetre 255 0 255 255)" g13
ofenetre g13 256 256 255 0 255 255
echo: FS_CREEMASQ g1 "(fenetre 0 0 0 255)" g16
ofenetre g16 256 256 0 0 0 255
echo: FS_RECONST 1 g10 g16 g18
oreconst 1 g10 g16 g18
echo: FS_RECONST 1 g18 g13 g20
oreconst 1 g18 g13 g20
/home/elsa2/xv/xv-3.00/xv g20&
echo Execution terminee
```

Or l'exécution du plan généré fourni un résultat vide (image noire) ne contenant aucune composante. L'examen des images (fig.4.19) montre que le voisinage (4-voisins) utilisé à partir de la donnée `g10` pour générer la donnée `g18` n'était pas adapté.



Donnée
g10

Donnée
g18

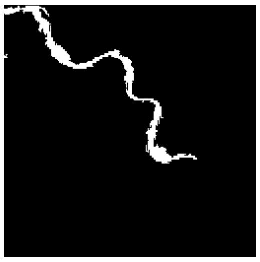


FIG. 4.19 - *Images des résultats intermédiaires*

Les descriptions de ces données nous donnent exactement l'élément à modifier dans l'objectif. En effet, le terme du voisinage, que nous avons souligné dans `g18`, doit être modifié au niveau de la description de l'objectif ou peut être ajusté au niveau du contrôle de l'exécution.

```
g10.τ = '((0 256) (0 256))
g10.κ = '(((nivg 148 256)) (dans ((vois8)(((nivg 0 148)))) (((nivg 148 256)))))
g10.ι = '(gl)
g10.σ = 'vab
g10.ω = '(1 2)
```

```
g18.τ = '((0 256) (0 256))
g18.κ = '((inter ((vois4)(((nivg 148 256)) (dans ((vois8)
      (((nivg 0 148)))) (((nivg 148 256)))))) ((fenetre 0 0 0 255)))))
g18.ι = '(gl)
g18.σ = 'vab
g18.ω = '(1 2)
```

La connaissance particulière exprimant la fusion des composantes connexes par le voisinage 8-voisins peut être alors utilisée pour résoudre ce problème par du contrôle d'exécution. Nous proposons ici une correction du script dont l'expression de la mesure à effectuer est exprimée dans le langage LEO. Cette correction consiste à mettre en œuvre une sélection de paramètre au regard du résultat d'une mesure sur le nombre de composantes déterminées.

Le script peut être alors modifié de la façon suivante[1] :

```
...
set v = 1
ajuste:
echo : FS_RECONST $v g10 g16 g18
oreconst $v g10 g16 g18
echo : FS_RECONST $v g18 g13 g20
oreconst $v g18 g13 g20
echo -----CONTROLE-------
echo -1- Mesure
olabel 1 g20 ExTImg1
```

```
ohisto ExTImg1 ExTImg2 1 256
omax_i ExTImg2 g23 1
rm -f ExTImg*
echo -2- Decision
if (`cat g23` == 0) then
set v = 2
echo Ajustement !
goto ajuste
endif
...
```

La donnée `g23` étant décrite par l'expression de l'objectif `nombre_cc`, nous pouvons ainsi reporter sur le graphe conceptuel de `riviere_2` la mesure employée pour le test du contrôle d'exécution (fig.4.20)

```
nombre_cc.τ = '()
nombre_cc.κ = 't
nombre_cc.ι = '(gl)
nombre_cc.σ = '(nbcc ((inter ((vois4)((inter ((vois4)(((nivg 148 256)) (dans ((vois8)
      (((nivg 0 148)))) (((nivg 148 256)))))) ((fenetre 0 0 0 255)))))
      ((fenetre 255 0 255 255))))
nombre_cc.ι = '(0 16385.)
```

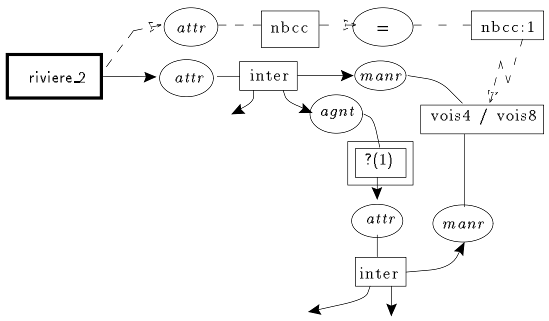


FIG. 4.20 - *Premier test pour le contrôle d'exécution*

Une seconde correction possible consiste à sélectionner une sous-chaîne en fonction d'un résultat préalable sur les boîtes englobantes des composantes connexes. La mesure à effectuer consiste alors à déterminer s'il existe une composante connexe dont les valeurs des mesures `xinf`érieur et `xsup`érieur de sa boîte sont égaux respectivement à 0 et 255 ; ce qui correspond à une valeur de ce que nous appelons `diam`ètre en `x` (`dx`) égale à 256. Une telle mesure est décrite par l'expression suivante :
`'((vmax dx) ((vois4)(((nivg 148 256)) (dans ((vois8)(((nivg 0 148)))) (((nivg 148 256)))))))`
Elle peut alors être reportée sur le graphe conceptuel de `riviere_2` (fig.4.21).

La modification du script à effectuer est alors la suivante :

1. Cette modification n'a pas été établie directement par le système. Elle résulte de l'intégration manuelle du script généré pour la mesure considérée.

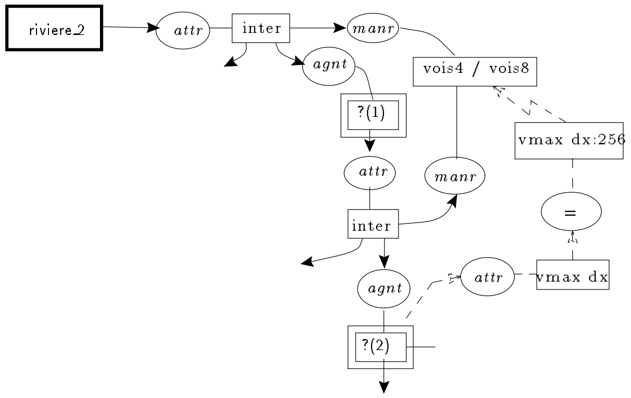


FIG. 4.21 - *Second test pour le contrôle d'exécution*

```
...
echo: FS_CREEMASQ g1 "(fenetre 0 0 0 255)" g16
ofenetre g16 256 256 0 0 0 255
echo -----CONTROLE-------
echo -1- Mesure
olabel 1 g10 g35
omesure g35 ExTImg 16
omax_v ExTImg g23
rm -f ExTImg
echo -2- Decision
if (`cat g23` == 256) then
echo Selection sequence 1
goto sequence1
else
echo Selection sequence 2
goto sequence2
endif
```

```
sequence1:
echo --Sequence 1--
echo: FS_RECONST 1 g10 g16 g18
oreconst 1 g10 g16 g18
echo: FS_RECONST 1 g18 g13 g20
oreconst 1 g18 g13 g20
echo --------------
goto fin
sequence2:
echo --Sequence 2--
echo: FS_RECONST 2 g10 g16 g18
oreconst 2 g10 g16 g18
echo: FS_RECONST 2 g18 g13 g20
oreconst 2 g18 g13 g20
echo --------------
fin:
/home/elsa2/xv/xv-3.00/xv g20&
```

Le dernier exemple que nous allons prendre consiste à réutiliser le script que nous venons de produire pour la résolution de l'objectif `riviere_2` pour tenter d'extraire de l'image de circuits, ceux traversant l'image (fig.4.22). L'expression initiale de cet objectif est équivalente à `riviere_2` mis à part une inconnue sur l'intensité des entités à représenter. Cette valeur égale à `148` pour le problème précédent doit être diminuée pour ces circuits. Nous pouvons donc envisager un processus de contrôle diminuant cette valeur jusqu'à obtenir un nombre de fils correct répondant à l'objectif. Ce nombre -5- fourni au processus de contrôle va permettre de déterminer le niveau de gris des pixels représentant les fils. Cette valeur servira ensuite à modifier l'objectif pour l'adapter au problème.

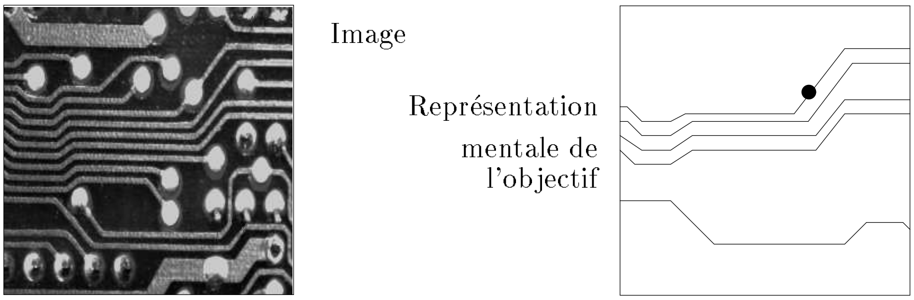


FIG. 4.22 - *Image et objectif* `conntrav0`

```
riviere_2.τ = '((0 256) (0 256))
riviere_2.κ = '((inter ((vois4) ((inter ((vois4) (((nivg 148 256)) (dans ((vois8)
      (((nivg 0 148)))) (((nivg 148 256)))))) ((fenetre 0 0 0 255))))) ((fenetre 255 0 255 255))))
riviere_2.ι = '(gl)
riviere_2.σ = 'vab
riviere_2.ω = '(1 2)
```

Cette valeur à ajuster apparaît trois fois dans l'expression ci-dessus. En fait, et nous pouvons le constater dans le graphe conceptuel, elle n'est liée qu'à un seul concept et peut alors être modifiée par une boucle décrémentant la valeur du seuil initial. La mesure utilisée pour le contrôle est, comme précédemment, un calcul du nombre de composantes extraites. Le script modifié, implémentant cette boucle, est alors :

```
#! /bin/csh
clear
echo SCRIPT MExTI
echo ===========
echo Image initiale: $1
rm -f *.dsc
echo: FS_LIRE $1 g3
cp $1 g3
echo: FS_CREEMASQ g1 "(fenetre 255 0 255 255)" g13
ofenetre g13 256 256 255 0 255 255
echo: FS_CREEMASQ g1 "(fenetre 0 0 0 255)" g16
ofenetre g16 256 256 0 0 0 255
echo -----BOUCLE DE CONTROLE-------
set nivg = 149
set vtest = 0
while ($vtest!= "5")
if ($vtest!= "5") then
set nivg = `expr $nivg - 1`
echo: FS_SEUILLER g3 0 $nivg g5
oseuiller g3 g5 0 $nivg
echo: FS_NON g5 g8
onot g5 g8
echo: FS_REMPLIR 2 g8 g10
oremplir 2 g8 g10
echo: FS_RECONST 2 g10 g16 g18
oreconst 2 g10 g16 g18
echo: FS_RECONST 2 g18 g13 g20
oreconst 2 g18 g13 g20
echo -----CONTROLE-------
echo -1- Mesure
olabel 2 g20 ExTImg1
ohisto ExTImg1 ExTImg2 1 256
omax_i ExTImg2 g23 1
rm -f ExTImg*
echo -2- Valeur de test
set vtest = `cat g23`
end
echo -----------------------------
echo Valeur de nivg determinee: $nivg
/home/elsa2/xv/xv-3.00/xv g20&
echo Execution terminee
```

L'exécution de ce script détermine la valeur à modifier au niveau de l'expression pour re-planifier correctement le nouvel objectif spécifique au domaine considéré et fournit alors la donnée de la figure 4.23 possédant la description suivante :

```
conntrav0.κ = '((inter ((vois4) ((inter ((vois4) (((nivg 88 256))
      (dans ((vois8)(((nivg 0 88)))) (((nivg 88 256))))))
      ((fenetre 0 0 0 255))))) ((fenetre 255 0 255 255))))
conntrav0.ι = '(gl)
conntrav0.σ = 'vab
conntrav0.ω = '(1 2)
```

FIG. 4.23 - *Résultat de l'exécution – conntrav0*

Exécution de ce script :

SCRIPT MExTI ============ Image initiale: circuit2.ras : FS_LIRE circuit2.ras g3 : FS_CREEMASQ g1 (fenetre 255 0 255 255) g13 : FS_CREEMASQ g1 (fenetre 0 0 0 255) g16 -----BOUCLE DE CONTROLE------- : FS_SEUILLER g3 0 148 g5 : FS_NON g5 g8 : FS_REMPLIR 2 g8 g10 : FS_RECONST 2 g10 g16 g18 : FS_RECONST 2 g18 g13 g20 -----CONTROLE------- -1- Mesure -2- Valeur de test : FS_SEUILLER g3 0 147 g5 : FS_NON g5 g8 : FS_REMPLIR 2 g8 g10 : FS_RECONST 2 g10 g16 g18 : FS_RECONST 2 g18 g13 g20 -----CONTROLE-------	-1- Mesure -2- Valeur de test ... : FS_SEUILLER g3 0 89 g5 : FS_NON g5 g8 : FS_REMPLIR 2 g8 g10 : FS_RECONST 2 g10 g16 g18 : FS_RECONST 2 g18 g13 g20 -----CONTROLE------- -1- Mesure -2- Valeur de test : FS_SEUILLER g3 0 88 g5 : FS_NON g5 g8 : FS_REMPLIR 2 g8 g10 : FS_RECONST 2 g10 g16 g18 : FS_RECONST 2 g18 g13 g20 -----CONTROLE------- -1- Mesure -2- Valeur de test ------------------------------ Valeur de nivg determinee: 88

Troisième partie

Mise en œuvre et exploitation

La mise en œuvre et l'intégration des modèles dans le système de résolution de problèmes ExTI ont été effectuées en respectant deux contraintes que nous nous étions posées à la suite d'une analyse du système existant qui a montré des limites sur le plan informatique -limites d'implémentation dûes aux priorités et au temps de développement- et des limites sur le plan théorique -problèmes posés lors de l'évaluation du système.

Tout d'abord il nous fallait assouplir l'utilisation du système et l'écriture des objectifs. Il s'agissait de d'étudier les problèmes de formalisme, d'interface, et de vérification des objectifs. Ces difficultés d'écriture d'un objectif provenaient essentiellement d'un manque de formalisme et d'une obligation d'application manuelle de règles de simplification entraînant une difficulté d'expression de certains objectifs. Ce premier aspect est résolu par les modèles que nous avons proposés et les outils, tels l'analyseur syntaxique, que nous avons pu développer.

Il nous fallait ensuite permettre au système de résoudre des objectifs complexes et pour cela pallier l'"explosion" rapide du moteur de planification. Ceci était essentiellement dû à des redondances dans les créations d'objets et dans les applications de ceratins opérateurs, un manque de simplification des expressions et un filtrage d'opérateurs peu performant ainsi qu'à un manque de stratégie de planification. Ce second aspect, nous le verrons dans cette partie, est résolu par la simplification des expressions produites, un filtrage d'opérateurs faisant intervenir des connaissances sur leurs propriétés en traitement d'images et un appariement d'expressions guidé par la syntaxe de ces expressions dans le langage LEO. Le module de planification est alors amélioré par l'exploitation de la notion de vocabulaire exécutable.

Chapitre 1

Données, opérateurs et planification

Dans ce premier chapitre sur la mise en œuvre et l'exploitation des modèles, nous allons montrer comment sont implantés les données et les opérateurs dans le système ExTI. À l'aide d'exemples, et en s'appuyant sur le formalisme proposé -en particulier sur les transformations d'informations- nous détaillerons une méthode d'implémentation de nouveaux opérateurs dans la base symbolique.

Dans un second temps, nous reviendrons sur le module de planification développé pour ce système en montrant les implications de ce nouveau formalisme sur ce planifieur. Nous proposerons ensuite le module de réutilisation de plans, pour finir sur la notion de base minimale d'opérateurs et ses conséquences sur la gestion de macro-opérateurs.

1.1 Implémentation

1.1.1 Le modèle des données

Pour implémenter le modèle des données dans le système ExTI, nous avons conservé la structure interne de représentation de ces données. Cette structure est constituée des sept champs suivants : le nom symbolique de la donnée (`nom`), la fonction génératrice identifiant l'opérateur produisant cette donnée (`generateur`), les structures manipulables (`s_m`), les représentations de ces structures (`repr`), les états accessibles (`e_a`), les opérateurs applicables (`o_a`) ainsi que les types des données accessibles (`d_a`).

Les cinq champs `nom`, `generateur`, `e_a`, `o_a` et `d_a` n'ont pas été modifiés par cette implémentation car ils contiennent des informations nécessaires au déroulement de la planification et n'interviennent pas dans la représentation symbolique des données.

Les champs `s_m` et `repr` ont quant à eux été modifiés. Dans le modèle précédent, il existait plusieurs valeurs possibles pour les structures manipulables qui avaient pour rôle d'affiner le typage des données représentées. Celles-ci étaient en effet regroupées sous cinq types prédéfinis : image multi-niveaux (`im`), images binaires multi-niveaux (`imb+`), images d'étiquettes (`label`), images binaires (`ib`), vecteurs (`vect`) et entiers ou réels (`fix`). Ces types permettaient de définir les opérateurs applicables aux données ainsi regroupées. Dans notre formalisme, la description associée au champ signal du modèle des données a remplacé les valeurs de ce champ `s_m`, en le complétant et en le précisant. Les structures manipulables de cette nouvelle représentation sont alors constituées de l'unique liste `(d do)` correspondant à l'identification des descriptions des données `(d)` et des complémentaires de ces descriptions `(do)` ; la donnée symbolique étant considérée comme formée des deux entités D et $\overline{D}$. Le champ `repr` d'une donnée est alors représentée par une liste de deux vecteurs de cinq éléments chacun. Le premier vecteur contient les descriptions des champs trame (τ), cache (κ), index (ι), signal (σ) et codage (ω) de la donnée. Le second vecteur contient les descriptions des champs trame de la donnée (τ), complémentaire du cache κ ($\overline{\kappa}$), index sur $\overline{\kappa}$ ($\overline{\iota}$), signal des points de $\overline{\kappa}$ ($\overline{\sigma}$), et leur codage ($\overline{\omega}$).

De par l'uniformisation de cette représentation interne des données, les opérateurs symboliques effectuant le même traitement sur des données de types différents ont été supprimés et remplacés par un seul opérateur manipulant une seule représentation des données. Néanmoins, ces types existent encore. En effet, la suppression de ceux-ci entraînerait des instanciations de paramètres étonnantes, pour la gestion des opérateurs applicables. Nous pourrions alors appliquer un opérateur de seuillage sur une donnée entière (élément de $\mathbb{N}$) avec des seuils faisant référence à une image et une séquence d'images ; cela reviendrait à un appel du genre `(fs_seuiller fix im im3)`, ce qui est totalement incorrect du point de vue du traitement d'images. Ainsi, dans le cas de la suppression de ces types, cette connaissance du traitement d'images, que nous mettons dans le système ExTI par le typage des données pour paramétrer correctement les opérateurs, devrait alors être remplacée par un processus de planification pourvu d'un module performant basé sur une stratégie ou un filtrage d'opérateurs particulier. En résumé, le typage des données n'est plus une nécessité en tant que représentation et description des données ; il est par contre important de le conserver tant qu'un module de planification tenant en compte cette caractéristique du modèle n'est pas développé.

L'implémentation de la donnée est alors définie par la structure suivante :

```
(defstruct #:objet:type
     (nom (gensym))
     (generateur)
     (s_m '( d do ))
     (repr '( #[()()()()()] #[()()()()()] )
     (e_a '( () () )
     (o_a '( () () )
     (d_a '( () () )))
```

Au niveau des types de données manipulées, nous avons fusionné les types `imb+` et `label` et rajouté le type `im3`. De ce fait les types correspondent essentiellement à des dimensions de données :

`im3` données de dimension 3.
`im` données de dimension 2, multi-niveaux.
`imb+` données de dimension 2, binaires multi-niveaux.
(c'est à dire établissant une partition entre les valeurs nulles et non-nulles)
`ib` données de dimension 2, binaires.
(la donnée n'est codée que sur deux valeurs)

`vect` données de dimension 1.

`fix` données de dimension 0.

1.1.2 Le modèle des opérateurs

Les schémas de représentation que nous construisons permettent de mettre en exergue les différentes étapes du processus de transformations d'information effectuées par les opérateurs. Ces étapes d'identification, de mesure, de partition, de codage et de regroupement étant déterminées pour les représenter graphiquement, ces schémas fournissent les informations essentielles, nécessaires à l'implémentation des opérateurs symboliques, au regard des opérations de transformation des champs de la donnée. Ils peuvent alors être considérés comme sont un préalable indispensable à l'implémentation des opérateurs qu'ils décrivent dans le système de planification automatique de chaînes de traitement d'images.

Ce système étant développé en langage Lisp [Lel91], les opérateurs symboliques sont alors implantés sous forme de fonctions ayant toutes la même structure : une en-tête composée du nom et des paramètres de la fonction, une déclaration et initialisation de variables locales permettant de récupérer les différents champs des données d'entrée et le constructeur de liste de deux vecteurs, chacun d'eux appelant les fonctions de transformation liées aux champs à modifier.

```
(df nom (paramètres)
  (lets ( variables )
    (list (vector τ^t() κ^t() ι^t() σ^t() ω^t() )
          (vector τ^t() κ^t() ι^t() σ^t() ω^t() ))))
```

Si nous prenons les exemples simples des opérateurs de seuillage et de dilatation que nous avons déjà utilisés, nous pouvons montrer (fig.1.1 et 1.2) comment développer l'opérateur symbolique à partir des éléments représentatifs des schémas de décompositions. L'exemple de l'opérateur de dilatation montre comment il serait implanté dans ExTI. En effet, cette implantation n'est pas effectivement faite dans ExTI car cet opérateur est un macro-opérateur ; il est donc planifié par le système (voir paragraphe suivant).

Opérateur de seuillage

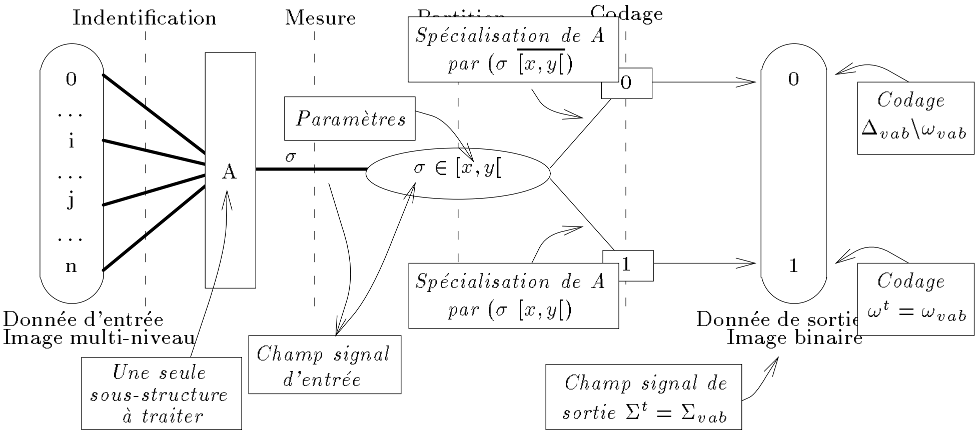


FIG. 1.1 - *Schéma symbolique augmenté de l'opérateur de seuillage*

```
01 (df fs_seuiller (img inf sup)
02    (lets ( (seuil1 (ds_val_repr inf))
03            (seuil2 (ds_val_repr sup))
04        (imd (car (send 'repr (eval img))))
05        (imd*t (vref imd 0))
06        (imd*k (vref imd 1))
07        (imd*i (vref imd 2))
08        (imd*s (vref imd 3))
09        (prop (ds_redom `(,imd*s ,seuil1 ,seuil2)))
10        (~prop (ds_redom `(,imd*s ,(ds_MinDom imd*s) ,seuil1 ,seuil2 ,(ds_MaxDom imd*s)))) )
11
12    (list (vector (Idt imd*t) (Kappa- prop imd*k imd*i) (Gamma-) (Sigma 'vab) (Omega 'vab) )
13          (vector (Idt imd*t) (Kappa- ~prop imd*k imd*i) () (Sigma 'vab) (~Omega 'vab)))))
```

Commentaires :
Dans un premier temps, la fonction symbolique implémentant l'opérateur de seuillage récupère les valeurs de seuil des données `inf` et `sup`, paramètres de cette fonction (lignes `02-03`), ainsi que la sous-structure A de la donnée image `im` (lignes `04`). Après avoir récupéré les champs trame (`imd*t`), cache (`imd*k`), index (`imd*i`) et signal (`imd*s`) de cette donnée (lignes `05-08`), la fonction construit les propriétés `prop` et `~prop` (lignes `09-10`) nécessaires à la partition de la sous-structure. Ces propriétés de spécialisation (`imd*s seuil1 seuil2`) et (`imd*s (ds_MinDom imd*s) seuil1 seuil2 (ds_MaxDom imd*s)`) correspondent ainsi à une partition des valeurs du signal σ=`imd*s` ; $\sigma \in$ [`seuil1`,`seuil2`[et $\sigma \in \Delta_\sigma \backslash$[`seuil1`,`seuil2`[. Elles sont simplifiées par la fonction de réduction des domaines (`ds_redom ...`). Les vecteurs de représentation sont alors produits (lignes `12-13`), établissant ainsi la description de la donnée `ib = fs_seuiller(im inf sup)`

Opérateur de dilatation binaire

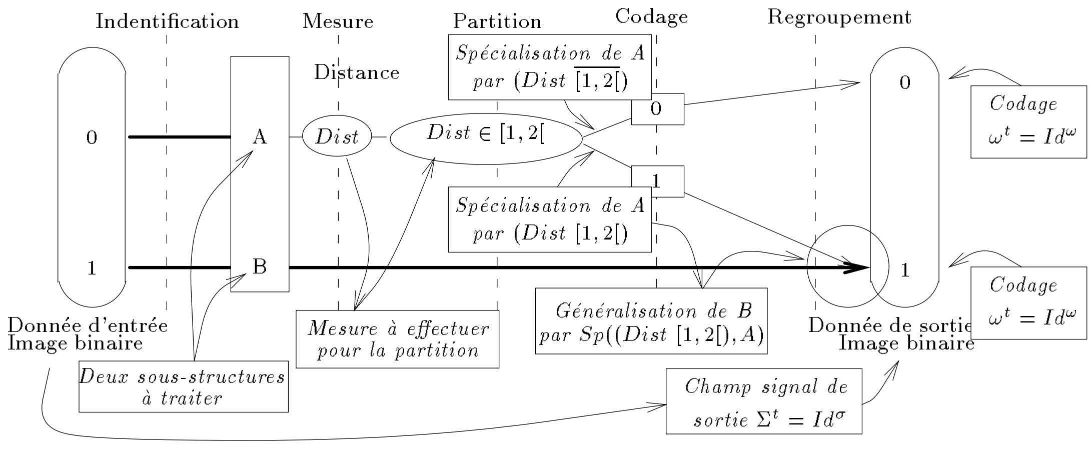


FIG. 1.2 - *Schéma symbolique augmenté de l'opérateur de dilatation*

```
(df fs_dilate (imgb)
   (lets ( (ibd (car (send 'repr (eval imgb))))
       (~ibd (cadr (send 'repr (eval imgb))))
       (ibd*t (vref ibd 0))
       (ibd*k (vref ibd 1))
       (~ibd*k (vref ~ibd 1))
       (ibd*i (vref ibd 2))
       (~ibd*i (vref ~ibd 2))
       (ibd*s (vref ibd 3))
       (~ibd*s (vref ~ibd 3))
```

```
      (ibd*o (vref ibd 4))
      (~ibd*o (vref ~ibd 4))
      (prop '(dist 1 2)))
      (~prop `(dist 2 ,(ds_MaxDom 'dist))) )
   (list (vector (Idt ibd*t) (Kappa+ prop ~ibd*k ibd*i ibd*k) (Idt ibd*i) (Idt ibd*s) (Idt ibd*o) )
         (vector (Idt ibd*t) (Kappa- ~prop ~ibd*k ibd*i) (Idt ~ibd*i) (Idt ~ibd*s) (Idt ~ibd*o)))))
```

Opérateur de dilatation multi-niveaux

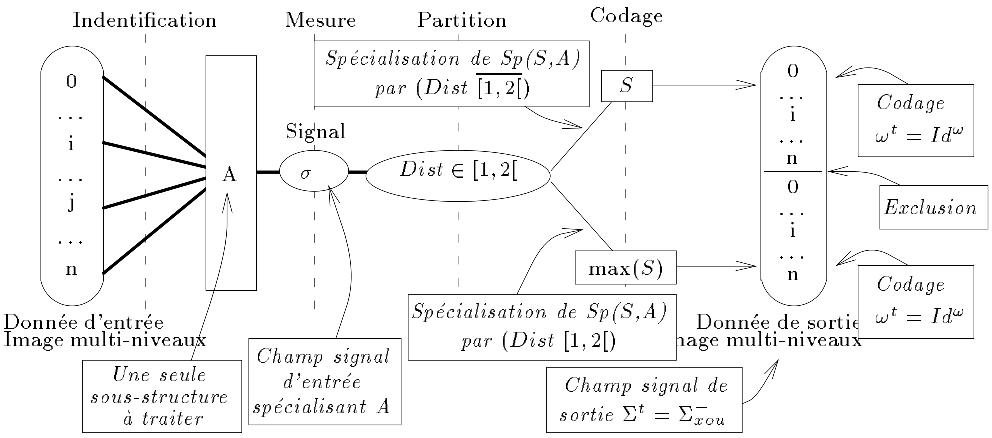


FIG. 1.3 - *Schéma symbolique augmenté de l'opérateur de dilatation multi-niveaux*

```
(df fs_dilate (img vsn iter)
   (lets ( (seuil (ds_val_repr iter))
      (imd (car (send 'repr (eval img))))
      (imd*t (vref imd 0))
      (imd*k (vref imd 1))
      (imd*i (vref imd 2))
      (imd*s (vref imd 3))
      (imd*c (vref imd 4))
      (prop1 `(dist ,seuil ,(ds_MaxDom 'dist)))
      (prop2 `(dist 1 ,seuil))
      (vrvs (ds_val_repr vsn))
      (vsd `(vois ,(ds_val_repr vsn))))
   (list (vector (Idt imd*t)
            (kexclusion (Kappa- prop1 (Kappa- imd*s imd*k imd*i) vsd)
               (Kappa- prop2 (Kappa- imd*s imd*k imd*i) vsd))
            (Gamma-- vrvs)
            (sexclusion imd*s (Sigma- 'max imd*s))
            (Idc imd*c) )
         ()) ) ))
```

1.2 Planification par ExTI

L'uniformisation de la gestion du typage des données -organisée selon la dimension de celles-ci- et le respect du formalisme quant à la représentation de ces données, nous a conduit à considérer celles-ci sous un autre point de vue et à développer des algorithmes de planification et d'appariement plus généraux que ceux déjà implémentés. L'introduction de ce formalisme a donc cinq conséquences principales sur l'ensemble du processus de planification automatique :

1. simplification d'expressions.
2. modification du module d'appariement.
3. développement d'un processus de filtrage d'opérateurs.
4. développement d'un module de planification spécifique.
5. augmentation du potentiel de recherche de solution.

Les deux premiers points évoqués seront détaillés dans les chapitres suivants traitant pour l'un de la simplification des expressions, et pour l'autre du module particulier d'appariement d'expressions. En effet, nous avons jugé intéressant de différer leur étude étant donné leur importance au niveau de la planification mais aussi au niveau de la manipulation des expressions du langage LEO ; ces manipulations pouvant être exploitées dans des systèmes autres que la planification : compréhension, enseignement, interface, etc.

1.2.1 Filtrage d'opérateurs

Lorsqu'une donnée symbolique est créée par un opérateur, sa liste d'opérateurs applicables - champ `o_a`- est mise à jour en fonction du type de cette donnée. Cette liste comprend des éléments tels que `(fs_seuiller img inf sup)` où `img` désigne une donnée de type `im` ou `imb+` et `inf` et `sup` désignent des données de type `fix`. Hormis le paramètre correspondant au type de la donnée courant qui est directement instancié, les autres paramètres sont considérés comme des paramètres génériques. L'objectif du module d'appariement, pour des raisons de performance et de généralisation de son utilisation, n'étant plus de retourner des valeurs d'instanciations, a conduit à déterminer celles-ci par une procédure différente. Ainsi, les termes génériques, paramètres des opérateurs symboliques, seront-ils remplacés par toutes les données de ce type, disponibles, qu'elles soient issues de traitements précédents ou de l'analyse syntaxique, en particulier pour les valeurs numériques. À ces dernières s'ajouteront les valeurs des domaines de validité des descripteurs reconnus dans l'expression de l'objectif. La liste de ces opérateurs applicables est alors filtrée pour éliminer les configurations de paramètres incorrects. Ce filtrage fait ainsi intervenir des connaissances du traitement d'images particulières que nous avons ajouté au système ExTI. Ces connaissances sont mises sous forme de règles d'acceptation d'opérateurs en fonction des descriptions de leurs paramètres.

Exemple : Si nous prenons l'objectif `test` suivant, les valeurs attribuées par l'analyse syntaxique aux descripteurs `nivg` et `dist` sont :

```
test
     '((0 256) (0 256))
     '(((nivg 50 150))((dist 1 6) ((vois4)(((nivg 100 200))))))
     '(gl)
     'vab
     '(1 2)
   nivg.lfix: (0 50 100 150 200 256)
   dist.lfix: (1 6 128)
```

La création d'une donnee symbolique de type `im` fournira alors l'entité suivante :

```
-------------------------
g3 : #:objet:im
generateur : ()
- - - - - - - - - - - - -
s_m : repr
d : #[() () () () ()]
do : #[() () () () ()]
- - - - - - - - - - - - -
s_m : o_a -->d_a
d :
(fs_seuiller g3 fix? fix?) -->#:objet:ib
(fs_masque g3 ib?) -->#:objet:imb+
do :
-------------------------
```

Cette donnée est une instance du type `im` dans le contexte de l'objectif. En effet les champs `o_a` et `d_a` sont mis à jour en fonction des éléments sémantiques de l'objectif récupérés par l'analyseur. Cependant, ces champs sont encore non instanciés. Ceci est effectué en considérant cette donnée comme produite par un opérateur.

Ainsi pour l'opérateur de seuillage, l'instance (`fs_seuiller gx` $\texttt{fix?}_1$ $\texttt{fix?}_2$) sera acceptée si et seulement si une des règles suivantes peut être appliquée :
- $\texttt{fix?}_1$ ou $\texttt{fix?}_2$ ne sont pas des valeurs mais des résultats de mesures du type `fix` requis,
- $\texttt{fix?}_1 < \texttt{fix?}_2$, $\texttt{fix?}_1$ et $\texttt{fix?}_2$ ont été attribués par l'analyseur syntaxique au descripteur relatif au signal de la donnée `gx` et l'intervalle [$\texttt{fix?}_1$,$\texttt{fix?}_2$[ne définit pas le domaine de validité du descripteur précédent [1].

Exemple : Sur l'exemple précédent, et compte tenu des objets accessibles de type `fix` et `ib`, l'instanciation complète de la donnée `g3` est :

```
#:objet:fix : (f0 f1 f6 f50 f100 f128 f150 f200 f256)
#:objet:ib : ()
 -------------------------
 g3 : #:objet:im
 generateur : (fs_lire g1 im)
 - - - - - - - - - - - - -
 s_m : repr
 d : #[((0 256) (0 256)) t (gl) nivg (0 256)]
 do : #[((0 256) (0 256)) () () () ()]
 - - - - - - - - - - - - -
 s_m : o_a -->d_a
 d :
 (fs_seuiller g3 f0 f50) -->#:objet:ib
 (fs_seuiller g3 f0 f100) -->#:objet:ib
 (fs_seuiller g3 f0 f150) -->#:objet:ib
 (fs_seuiller g3 f0 f200) -->#:objet:ib
 (fs_seuiller g3 f50 f100) -->#:objet:ib
 (fs_seuiller g3 f50 f150) -->#:objet:ib
 (fs_seuiller g3 f50 f200) -->#:objet:ib
 (fs_seuiller g3 f50 f256) -->#:objet:ib
 (fs_seuiller g3 f100 f150) -->#:objet:ib
 (fs_seuiller g3 f100 f200) -->#:objet:ib
 (fs_seuiller g3 f100 f256) -->#:objet:ib
 (fs_seuiller g3 f150 f200) -->#:objet:ib
 (fs_seuiller g3 f150 f256) -->#:objet:ib
 (fs_seuiller g3 f200 f256) -->#:objet:ib
 do :
 -------------------------
```

Il faut alors noter la disparition de l'opérateur de masque (`fs_masque g3 ib?`) dû au fait qu'à cet instant de la planification aucune donnée de type `ib` n'est disponible. Néanmoins, nous retrouverons cet opérateur lors de la création d'une donnée de ce type :

```
 -------------------------
 g5 : #:objet:ib
 generateur : ()
 - - - - - - - - - - - - -
```

1. Le pourquoi de cette contrainte est expliqué dans le chapitre suivant sur la simplification des expressions.

```
s_m : repr
d : #[() () () () ()]
do : #[() () () () ()]
- - - - - - - - - - - - -
s_m : o_a -->d_a
d :
(fs_distance g5 vois?) -->#:objet:imb+
(fs_label g5 vois?) -->#:objet:imb+
(fs_masque im? g5) -->#:objet:imb+
(fs_ou g5 ib?) -->#:objet:ib
(fs_ou ib? g5) -->#:objet:ib
(fs_et g5 ib?) -->#:objet:ib
(fs_et ib? g5) -->#:objet:ib
do :
(fs_non g5) -->#:objet:ib
-------------------------
```

Celle-ci, après instanciation, permettra à nouveau d'accéder à cet opérateur de masque :

```
-------------------------
g5 : #:objet:ib
generateur : (fs_seuiller g3 f50 f100)
- - - - - - - - - - - - -
s_m : repr
d : #[((0 256) (0 256)) (((nivg 50 100))) (gl) vab (1 2)]
do : #[((0 256) (0 256)) (((nivg 0 50 100 256))) () vab (0 1)]
- - - - - - - - - - - - -
s_m : o_a -->d_a
d :
(fs_distance g5 vvois4) -->#:objet:imb+
(fs_label g5 vvois4) -->#:objet:imb+
(fs_masque g3 g5) -->#:objet:imb+
do :
(fs_non g5) -->#:objet:ib
-------------------------
```

D'autres règles, en particulier celles relatives aux opérateur logiques -`fs_ou`, `fs_et`, `fs_non`- font intervenir la fonction génératrice de la donnée courante -champ `generateur` de la donnée. Nous pouvons de ce fait déduire la pertinence d'un opérateur non plus en fonction de ses paramètres mais également en fonction de son utilisation dans la chaîne de traitement. Si nous prenons l'opérateur `(fs_et gx gy)`, les règles de filtrage qui lui sont associées sont alors fonction du `generateur` de `gx` : `(fs_? param)`. Cet opérateur `(fs_et gx gy)` sera rejeté si et seulement si :
- les valeurs `x` et `y` des numéros des données paramètres sont telles que `x` < `y`.
Cette règle gère la commutativité de l'opérateur.
- `fs_?`=`fs_et` et `gy` est un paramètre du générateur de `gx`, c'est à dire `gy`∈`param`.
Cette règle gère la propriété logique $gx \vee gy = (g? \vee gy) \vee gy = g? \vee gy = gx$.
- `fs_?`=`fs_ou` et `gy` est un paramètre du générateur de `gx`, c'est à dire `gy`∈`param`.
Cette règle gère la propriété de traitement d'images $gx \vee gy = (g? \wedge gy) \vee gy = gy$.
- `fs_?`=`fs_non` et `gy` est le paramètre du générateur de `gx`, c'est à dire `gx`=`fs_non gy`.
Cette règle gère la propriété logique $gx \vee gy = \sim gy \vee gy = \lambda$.

Notre formalisme et le principe du système ExTI permettant d'accéder à toutes les données produites ainsi qu'à leurs descriptions, nous pouvons ainsi élaborer des règles de filtrage d'opérateurs complexes établies sur une connaissance des opérateurs et de leur enchaînement.

1.2.2 Planification par réutilisation de plan

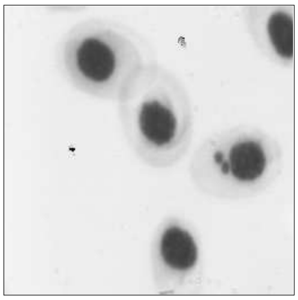
Le formalisme que nous proposons cherche à exploiter au maximum les connaissances de l'expert dans son domaine en lui permettant d'exprimer ses objectifs non plus en termes d'images à traiter, mais en termes d'indices visuels, c'est à dire en termes d'éléments spécifiques à extraire de ces images. Ces objectifs sont alors identifiables par un terme générique ou propre à son domaine. Nous avons vu que, par l'utilisation de règles transformationnelles, nous pouvons modéliser l'expansion[2] d'un de ces termes dans une description d'objectif en manipulant le concept qui lui est associé. Nous avions jusqu'à présent utilisé uniquement les entités *nom* et *objectif* du quadruplet que nous avons défini comme étant un élément du vocabulaire exécutable du domaine d'application. Le rôle du module de planification par réutilisation de plan consiste alors à approfondir l'exploitation de ce quadruplet -(*nom*, *objectif*, *plan*, *programme*)- en amenant le planifieur à tenir compte de ces entités déjà générées et validées. L'exploitation complète, intégrant le quatrième élément -*programme*- ne pourra être envisagé de façon pertinente que lorsque des processus de supervision et de contrôle d'exécution auront été mis en place. Il s'agira alors de maintenir ces contrôles *a priori* et *a posteriori* en les ajustant et en les intégrant dans le script final.
Comme nous l'avons évoqué précédemment, la planification des objectifs que nous manipulons ne peut être menée avec des méthodes classiques de planification non-linéaire. En effet, la décomposition en sous-objectifs de nos expressions étendues est difficile voire impossible à effectuer, si nous voulons rester attaché à la représentation sous forme de transformations d'information des opérateurs de notre base. Ceux-ci, décrivant des réalités du traitement d'images, génèrent des données dont les descriptions symboliques ne peuvent être toujours décomposables en plusieurs sous-objectifs. La réutilisation de plan, profite de la décomposition de cet objectif à atteindre pour le décomposer à l'aide des sous-objectifs exprimés. La comparaison avec les systèmes de planification tels TWEAK ne peut pas être faite car il ne s'agit pas ici de planification par disjonction de buts sinon de fusion de plans déjà développés.

1.2.2.1 Principe de la réutilisation

Il ne s'agit en aucun cas de reconnaissance de plan par l'intermédiaire d'expressions d'objectifs, mais plutôt de fusionner les plans associés aux étiquettes de concepts d'un domaine d'application spécifique, présents dans l'expression de l'objectif ; un plan étant constitué par une suite de couples formés par les éléments `nom` et `generateur` de la structure implémentant la donnée.

Exemple cellules.plan :

```
g3
(fs_lire g1 im)
g8
(fs_seuiller g3 f0 f113)
g10
(fs_borderkill g8 vvois4)
g5
(fs_seuiller g3 f0 f226)
g13
(fs_reconst g5 g10 vvois4)
```



Lorsque le système de planification fonctionne en mode réutilisation, cinq processus correspondant à des phases distinctes de traitement sont mis en œuvre.

2. Nous parlons d'expansion ou d'étendue d'un objectif étiqueté lorsque l'étiquette de celui-ci est remplacée par sa description

Phase d'initialisation
Pour un objectif donné, il s'agit de déterminer, avant expansion des sous-objectifs étiquetés, quels sont ceux pour lesquels un plan a été développé et validé. La récupération des étiquettes est effectué au cours de l'analyse syntaxique de l'objectif principal, qui, dans un même temps, effectue l'expansion.

Exemple sur un cache :
objectif `cellules*k`:
```
'((inter ((vois4)(cytoplasmes*k)) (noyaux_ou_micro-noyaux*k)))
```
liste des sous-objectifs utilisés : {`cytoplasmes`,`noyaux_ou_micro-noyaux`}
objectif totalement étendu `cellules*k`:
```
'((inter ((vois4) (((nivg 0 226)))) ((ninter ((vois4) (((nivg 0 113))))
((nfenetre 1 1 254 254))))))
```
Il suffit alors d'extraire de cette liste des sous-objectifs utilisés, les étiquettes pour lesquelles il n'existe pas de plan. Ceci est effectué par une simple recherche dans la base des plans.

Afin d'illustrer complètement cette décomposition en plusieurs phases, nous allons prendre un exemple générique de planification d'un objectif O. Cet objectif est décrit à l'aide de trois sous-objectifs étiquetés O_1, O_2 et O_3 ; les deux premiers eux-mêmes décrits en utilisant respectivement les sous-objectifs étiquetés A et B. Les sous-objectifs O_1 et O_2 possédent des plans que nous allons devoir réutiliser.

Phase 0 de régénération de plans
Cette phase consiste à développer les plans à réutiliser en parcourant, depuis la racine, le plan principal de résolution. Les opérateurs déjà utilisés pour la construction de ce graphe correspondent à des arcs déjà existants. Ces arcs sont alors suivis, seuls les opérateurs non encore utilisés doivent créer de nouveaux arcs.
Dans la figure 1.4 nous avons montré les deux types de génération d'un objectif O_2. Le schéma de gauche présente une génération aveugle, sans tenir compte des opérateurs déjà utilisés. Le schéma de droite présente le principe adopté où la génération de O_2 se fait en parcourant le plan de résolution de O_1.

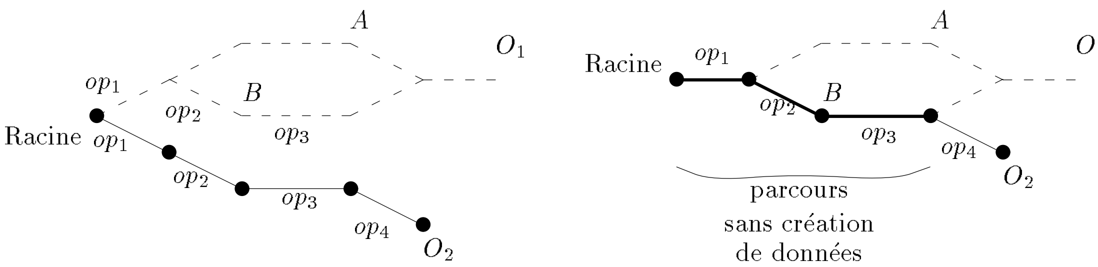


FIG. 1.4 - *Réutilisation sans et avec fusion des plans*

Cette phase est effectuée en utilisant le module de planification simple en tant que processus de création de plan, les étapes de filtrage et d'appariement de description n'étant pas déclenchées.

Phase 1 de planification sur les plans générés
Il s'agit cette fois d'une véritable planification de l'objectif principal (O) par le module simple. Or, contrairement à la planification classique, à partir de la racine du graphe, le processus est déclenché sur les données résultats, produites lors de la phase précédente, qui résolvent les sous-objectifs (ici O_1 et O_2). Cette phase 1 a donc pour effet de tenter d'exploiter ces sous-objectifs pour allonger le graphe de résolution.

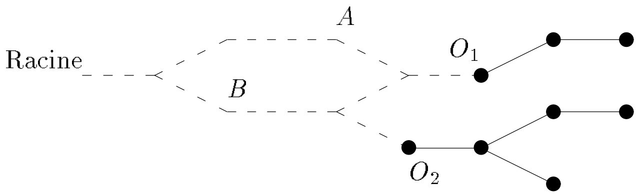


FIG. 1.5 - *Planification sur les plans générés*

L'objectif du module de réutilisation de plans étant d'exploiter au maximum le vocabulaire du domaine utilisé pour étiqueter les objectifs, les éléments de ce domaine rencontrés lors de la phase de régénération de plans devront être conservés pour procéder à la phase 2 du processus.

Phase 2 de planification sur les sous-objectifs étiquetés
Comme sa dénomination l'indique, cette phase consiste à déclencher le processus de planification sur les sous-objectifs étiquetés reconnus (ici A et B) lors de la phase de reconstruction des plans (ici O_1 et O_2). Ceci a pour effet de donner un certain privilège aux données représentant des concepts du domaine de l'application.

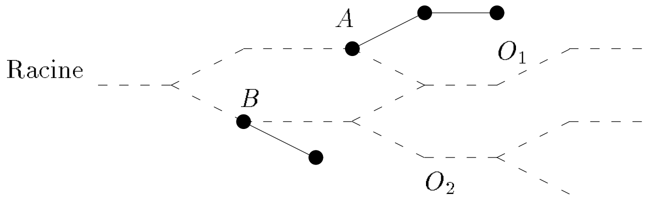


FIG. 1.6 - *Planification sur les sous-objectifs étiquetés*

Si nous nous ramenons à l'expression initiale de l'objectif général, cette phase tient compte de l'utilisation de sous-sous-objectifs étiquetés.

Exemple sur le cache précédent :
objectif `cellules*k` :
`'((inter ((vois4)(cytoplasmes*k)) (noyaux_ou_micro-noyaux*k)))`
liste des sous-objectifs utilisés : {`cytoplasmes`,`noyaux_ou_micro-noyaux`}
objectif partiellement étendu `cellules*k` :
`'((inter ((vois4)(((nivg 0 226)))) ((ninter ((vois4)(n+mn*k)) (bord_image*k)))))`
liste des sous-sous-objectifs : {`n+mn`,`bord_image`}

Phase 3 de planification sur les sous-objectifs non étiquetés
Cette phase consiste à tenter de résoudre l'objectif principal en déclenchant le processus de planification simple sur les données intermédiaires, produites lors de la phase 0 et ne correspondant pas à un sous-sous-objectif étiqueté.

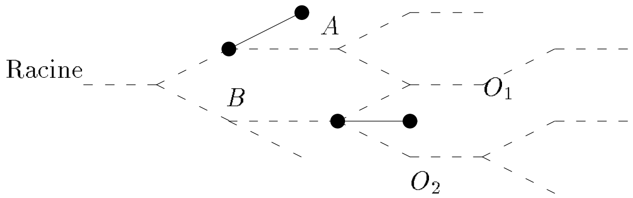


FIG. 1.7 - *Planification sur les sous-objectifs étiquetés*

Phase 4 de planification sur la racine

La dernière phase consiste à planifier l'objectif à partir de la racine. Il s'agit donc d'une planification simple, avec cependant un graphe de résolution partiel, déjà constitué.

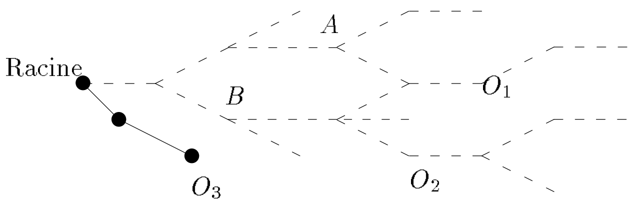


FIG. 1.8 - *Planification sur la racine*

L'obtention du résultat est garanti par cette dernière phase correspondant, dans le cas le plus extrême à une planification sans réutilisation du graphe de résolution partiel.

Cependant, le graphe optimal -avec réutilisation maximale des concepts du domaine- est obtenu du fait du parcours des plans générés pour déclencher des processus de planification sur des données intermédiaires. Toutes les données étant connues et accessibles au cours de la planification, il ne peut y avoir d'omission durant ce parcours.

1.2.2.2 Résultats

Les quatre planifications suivantes montrent comment l'algorithme que nous venons de développer s'intègre dans le processus de planification. L'exemple choisi consiste à planifier la résolution de l'objectif `cellules_avec_micro-noyaux` suivant :

```
cellules_avec_micro-noyaux
    '((0 256) (0 256))
    '((inter ((vois4)(cellules*k)) (micro-noyaux*k)))
    '(gl)
    'vab
    '(1 2)
cellules
    '((0 256) (0 256))
    '((inter ((vois4)(cytoplasmes*k)) (noyaux_ou_micro-noyaux*k)))
    '(gl)
    'vab
    '(1 2)
micro-noyaux
    '((0 256) (0 256))
    '(((srf 1 100) ((vois4)(noyaux_ou_micro-noyaux*k))))
    '(gl)
    'vab
    '(1 2)
```

Le premier plan est généré sans utiliser le module de réutilisation de plans. Pour générer le second, le système de planification utilise le fait que l'objectif `cellules` a été planifié et validé. Il réutilise donc ce plan. La troisième planification tient compte des plans associés aux objectifs `cellules` et `micro-noyaux`. Enfin la dernière utilisation du module consiste à régénérer le plan validé de l'objectif.

Les résultats au niveau temps d'exécution de ces planifications, effectuées sur une station SPARC1, sont rassemblés dans le tableau suivant. Celui-ci montre la diminution considérable du nombre d'opérateurs testés qui correspondent aux opérateurs filtrés que le planifieur a tenté d'appliquer sur les données pour résoudre l'objectif. C'est en grande partie cette diminution du nombre d'opérateurs qui est à l'origine de cette amélioration de la durée de planification. Toutes ces valeurs sont fournies à la fin de chaque planification.

Plans réutilisés	Nombre d'opérateurs testés	Durées de Résolution	Temps par opérateur
aucun	165	204,76s	1,24s
`cellules`	114	110,04s	0,96s
`cellules , micro-noyaux`	11	15,30s	1,39s
`cellules_avec_micro-noyaux`	8	13,48s	1,68s

Première planification

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : ((inter ((vois4) ((inter ((vois4) (((nivg 0 226)))) ((ninter
((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254 254))))))) (((srf 1
100) ((vois4) ((ninter ((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254
254))))))))
| Index : (g1)
| Signal : vab
| Codage : (1 2)

g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f113)
          [n+mn]
g7 = (fs_borderkill g5 vvois4)
          [noyaux_ou_micro-noyaux]
g9 = (fs_reconst g5 g7 vvois4)
g11 = (fs_reconst g5 g9 vvois4)
          (fs_reconst g5 g9 vvois4) -->g9
g13 = (fs_label g7 vvois4)
g15 = (fs_filtre g13 f1 f100 srf)
          [micro-noyaux]
g17 = (fs_reconst g5 g15 vvois4)
g19 = (fs_reconst g5 g17 vvois4)
          (fs_reconst g5 g17 vvois4) -->g17
g21 = (fs_label g17 vvois4)
g23 = (fs_reconst g9 g15 vvois4)
g25 = (fs_reconst g9 g23 vvois4)
          (fs_reconst g9 g23 vvois4) -->g23
g27 = (fs_label g15 vvois4)
g29 = (fs_label g5 vvois4)
g31 = (fs_filtre g29 f1 f100 srf)
g33 = (fs_reconst g15 g31 vvois4)
          [micro-noyaux]
          (fs_reconst g15 g31 vvois4) -->g15
g35 = (fs_label g31 vvois4)
g37 = (fs_seuiller g3 f0 f226)
          [cytoplasmes]
g39 = (fs_reconst g37 g15 vvois4)
g41 = (fs_reconst g37 g39 vvois4)
          (fs_reconst g37 g39 vvois4) -->g39
g43 = (fs_label g39 vvois4)
g45 = (fs_reconst g37 g7 vvois4)
          [cellules]
g47 = (fs_reconst g45 g15 vvois4)
          [cellules_avec_micro-noyaux]
Le but est atteint par egalite stricte
Temps d'execution (CPU) de la planification : 3mn 24.76001s
Nombre d'operateurs testes : 165
Temps moyen de test par operateur : 1.24097s

Solutions : (g47)
```

Deuxième planification

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : ((inter ((vois4) ((inter ((vois4) (((nivg 0 226)))) ((ninter
((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254 254))))))) (((srf 1
100) ((vois4) ((ninter ((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254
254))))))))
| Index : (g1)
| Signal : vab
| Codage : (1 2)

Generation du plan :
cellules sous /home/dejean/.../cellules.plan
====================
#[((0 256) (0 256)) ((inter ((vois4) (((nivg 0 226)))) ((ninter
((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254 254)))))) (g1) vab (1
2)]
g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f113)
          [n+mn]
g7 = (fs_borderkill g5 vvois4)
          [noyaux_ou_micro-noyaux]
g9 = (fs_seuiller g3 f0 f226)
          [cytoplasmes]
g11 = (fs_reconst g9 g7 vvois4)
          [cellules]
Le but est atteint par egalite stricte

Planification du but principal :
===============================
Phase No1 : sur les plans generes
Objet : g11
g14 = (fs_reconst g9 g11 vvois4)
          [cellules]
          (fs_reconst g9 g11 vvois4) -->g11
g16 = (fs_label g11 vvois4)
g18 = (fs_non g11)
Phase No2 : sur les sous-buts etiquettes reconnus
Objet : g9
g21 = (fs_reconst g9 g11 vvois4)
          [cellules]
          (fs_reconst g9 g11 vvois4) -->g11
g23 = (fs_reconst g11 g9 vvois4)
          [cellules]
          (fs_reconst g11 g9 vvois4) -->g11
g25 = (fs_reconst g7 g9 vvois4)
          [noyaux_ou_micro-noyaux]
          (fs_reconst g7 g9 vvois4) -->g7
g27 = (fs_reconst g5 g9 vvois4)
          [n+mn]
          (fs_reconst g5 g9 vvois4) -->g5
g29 = (fs_label g9 vvois4)
g31 = (fs_et g18 g9)
g33 = (fs_label g31 vvois4)
g35 = (fs_ou g31 g11)
          [cytoplasmes]
          (fs_ou g31 g11) -->g9
Objet : g7
g38 = (fs_reconst g9 g7 vvois4)
          [cellules]
          (fs_reconst g9 g7 vvois4) -->g11
g40 = (fs_reconst g5 g7 vvois4)
g42 = (fs_reconst g5 g40 vvois4)
          (fs_reconst g5 g40 vvois4) -->g40
g44 = (fs_label g7 vvois4)
g46 = (fs_filtre g44 f1 f100 srf)
          [micro-noyaux]
g48 = (fs_reconst g11 g46 vvois4)
          [cellules_avec_micro-noyaux]
Le but est atteint par egalite stricte
Temps d'execution (CPU) de la planification : 1mn 50.04004s
Nombre d'operateurs testes : 114
Temps moyen de test par operateur : .9652635s

Solutions : (g48)
```

Troisième planification

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : ((inter ((vois4) ((inter ((vois4) (((nivg 0 226)))) ((ninter
((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254 254))))))) (((srf 1
100) ((vois4) ((ninter ((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254
254)))))))))
| Index : (gl)
| Signal : (vab)
| Codage : (1 2)

Generation du plan :
cellules sous /home/dejean/.../cellules.plan
====================
#[((0 256) (0 256)) ((inter ((vois4) (((nivg 0 226)))) ((ninter
((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254 254)))))) (gl) (vab) (1
2)]
g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f113)
         [n+mn]
g7 = (fs_borderkill g5 vvois4)
         [noyaux_ou_micro-noyaux]
g9 = (fs_seuiller g3 f0 f226)
         [cytoplasmes]
g11 = (fs_reconst g9 g7 vvois4)
         [cellules]
Le but est atteint par egalite stricte

Generation du plan :
micro-noyaux sous /home/dejean/.../micro-noyaux.plan
====================
#[((0 256) (0 256)) (((srf 1 100) ((vois4) ((ninter ((vois4) (((nivg
0 113)))) ((nfenetre 1 1 254 254))))))) (gl) (vab) (1 2)]
g3 = (fs_lire g1 im)
          Re-utilisation de l'objet : g3
g5 = (fs_seuiller g3 f0 f113)
          [n+mn]
          Re-utilisation de l'objet : g5
g7 = (fs_borderkill g5 vvois4)
          [noyaux_ou_micro-noyaux]
          Re-utilisation de l'objet : g7
g16 = (fs_label g7 vvois4)
g18 = (fs_filtre g16 f1 f100 srf)
          [micro-noyaux]
Le but est atteint par egalite stricte

Planification du but principal :
================================
Phase No1 : sur les plans generes
Objet : g11
g21 = (fs_reconst g11 g18 vvois4)
          [cellules_avec_micro-noyaux]
Le but est atteint par egalite stricte
Temps d'execution (CPU) de la planification : 15.30005s
Nombre d'operateurs testes : 11
Temps moyen de test par operateur : 1.390913s

Solutions : (g21)
```

Quatrième planification

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : ((inter ((vois4) ((inter ((vois4) (((nivg 0 226)))) ((ninter
((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254 254))))))) ((
(srf 1 100) ((vois4) ((ninter ((vois4) (((nivg 0 113)))) ((nfenetre 1
1 254 254)))))))))
| Index : (gl)
| Signal : (vab)
| Codage : (1 2)

Generation du plan :
cellules_avec_micro-noyaux sous
/home/dejean/.../cellules_avec_micro-noyaux.plan
====================
#[((0 256) (0 256)) ((inter ((vois4) ((inter ((vois4) (((nivg 0
226)))) ((ninter ((vois4) (((nivg 0 113)))) ((nfenetre 1 1 254
254))))))) ((
(srf 1 100) ((vois4) ((ninter ((vois4) (((nivg 0 113)))) ((nfenetre 1
1 254 254))))))))) (gl) (vab) (1 2)]
g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f113)
          [n+mn]
g7 = (fs_borderkill g5 vvois4)
          [noyaux_ou_micro-noyaux]
g9 = (fs_label g7 vvois4)
g11 = (fs_filtre g9 f1 f100 srf)
          [micro-noyaux]
g13 = (fs_seuiller g3 f0 f226)
          [cytoplasmes]
g15 = (fs_reconst g13 g7 vvois4)
          [cellules]
g17 = (fs_reconst g15 g11 vvois4)
          [cellules_avec_micro-noyaux]
Le but est atteint par egalite stricte
Temps d'execution (CPU) de la planification : 13.48s
Nombre d'operateurs testes : 8
Temps moyen de test par operateur : 1.685s

Solutions : (g17)
```

1.2.3 Potentiel de résolution

Du fait d'une simplification des expressions manipulées, d'un allègement du processus d'appariement et de l'utilisation d'un algorithme de planification performant et adapté à notre formalisme, nous avons pu augmenter le potentiel de résolution du système ExTI en repoussant le point d'explosion combinatoire. Ceci était un des soucis principaux -avec la minimisation des temps de planification- qui ont guidé nos choix lors de l'implémentation et le développement des modules.

Parallèlement, de par le regroupement de types de données et l'emploi de la même représentation pour les différentes valeurs numériques (valeurs de l'objectif, domaine de validité, et données intermédiaires) et malgré un filtrage et un appariement plus rigoureux, nous avons laissé plus de liberté au module de planification. En effet, l'appariement ne fournit plus les valeurs nécessaires pour instancier les expressions génériques produites par les opérateurs, mais valide des descriptions complètes, générées par ces mêmes opérateurs. L'avantage est donc de créer des données supplémentaires, susceptibles de correspondre à des concepts du domaine d'application, non accessibles par l'ancien formalisme.

1.3 Base minimale et macro-opérateurs

1.3.1 Principe d'élimination

La notion de base minimale a été introduite lors de l'étude des performances et du potentiel de résolution du planifieur de ExTI. La décomposition rigoureuse et fine des opérateurs nous a conduit en effet à émettre l'hypothèse qu'il existait une base minimale d'opérateurs symboliques pour notre système. Même si l'existence d'une telle base est difficile à démontrer et à concevoir matériellement, cette notion doit toutefois être évoquée lorsqu'un nouvel opérateur doit être introduit dans la base d'opérateurs symboliques.

Le principe de cette notion repose sur le fait que le module de planification, à l'aide d'opérateurs d'une même base, est capable de développer un graphe de résolution permettant d'accéder à toutes les solutions résolvant un problème. Si un de ces opérateurs peut se décomposer en une suite d'opérateurs de la même base, le graphe comporte alors plusieurs séquences solutions correspondant au même traitement. Ainsi donc, nous avons jugé inutile de disposer de tels opérateurs pour planifier l'objectif. Nous pourrons voir dans le dernier chapitre, relatif aux résultats, que l'introduction de tels opérateurs entraîne également un degré de complexité supplémentaire, augmentant de ce fait les temps d'exécution de la planification.

1.3.2 Restitution dans le script

Si des opérateurs sont supprimés de la base, ils doivent cependant être utilisés afin d'écrire un script de traitement d'images correct et performant correspondant aux opérateurs réels -par opposition à symbolique- de la bibliothèque de traitement d'images. Le problème qui se pose est de déterminer la séquence d'opérateurs du plan à remplacer et à traduire par le "*macro-opérateur*". Il s'agit donc d'un problème de reconnaissance de plans relativement délicat étant donné la non unicité des plans à reconnaître et l'imbrication sur plusieurs niveaux de ces macro-opérateurs. Ainsi, l'ouverture morphologique est décrite comme une combinaison des opérateurs de dilatation et d'érosion, eux-même considérés comme des macro-opérateurs; l'expression de la transformation par l'opérateur d'érosion intervenant même dans la description de la dilatation.

Un autre point de difficulté consiste à vérifier, après reconnaissance, et avant substitution, s'il n'existe pas de sous-graphe engendré à partir d'éléments du plan reconnus ; les opérateurs générant ces éléments devant alors être conservés. Ainsi, sur la figure 1.9, la chaîne dérivant le macro-opérateur ne pourra être remplacée simplement par celui-ci car cela interdirait le développement du sous-graphe issu de la donnée D.

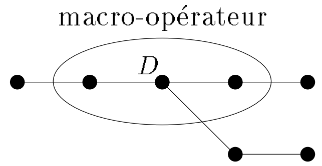


FIG. 1.9 - *Macro-opérateur et plan généré*

Dans le système ExTI, la prise en compte de ces macro-opérateurs pour la génération du script n'en est encore qu'au stade de l'étude des problèmes. Cependant, nous trouverons au chapitre 4 les résultats de premiers essais portant sur des opérateurs et des objectifs simples.

Chapitre 2

Intérêts et principes de la simplification d'expressions

Dans ce chapitre nous allons tout d'abord étudier le double intérêt d'une simplification des expressions de LEO pour la planification d'objectif : pour l'écriture et le développement plus souple des objectifs et pour une planification plus fiable. La simplification d'une expression a pour but de réduire la complexité de son analyse (planification, interprétation, lecture, etc) tout en conservant sa structure. Elle est ainsi liée à la mise sous une forme canonique d'une expression pour optimiser sa reconstruction par le planifieur.
Ensuite nous rappellerons l'ensemble des éléments de manipulation d'expressions qui sont utilisés par les opérateurs symboliques pour générer de nouveaux concepts. Nous compléterons cet ensemble, puis nous étudierons les configurations d'expressions pour lesquelles la simplification doit avoir lieu, en proposant une forme canonique pour chaque élément caractérisé.

2.1 Intérêts fondamentaux

La simplification des expressions possède un double intérêt. Le premier consiste à permettre l'écriture des objectifs en utilisant les noms de leurs étiquettes. Si cette possibilité était utilisée déjà dans [Ing92], certaines configurations d'écriture ne pouvaient être planifiées par manque, justement de simplification. Le second intérêt découle de cette remarque. En effet, le module de planification génère des expressions qu'il doit pouvoir comparer à celle représentant l'objectif.

2.1.1 Intérêt pour l'écriture des expressions

Un objectif développé par un utilisateur via une interface spécialisée possède des caractéristiques particulières témoignant des méthodes qu'il a mises en œuvre pour exposer son problème. Un même but peut alors être exprimé de diverses façons à l'aide de LEO. En effet, l'utilisation des descriptions génériques et / ou complémentaires (voir seconde partie §2.1.2) ainsi que des termes spécifiques du domaine de l'application employés pour étiqueter des buts déjà écrits autorise l'écriture d'objectifs sous différentes expressions, toutes équivalentes ; l'emploi de termes spécifiques est rendu possible par le développement de règles particulières d'une grammaire transformationnelle effectuant les substitutions des noms d'objectifs pour obtenir une expression correcte au sens de LEO.

Un exemple simple consiste à considérer deux descriptions établissant une partition des entités considérées. La disjonction de ces deux descriptions doit alors décrire ces entités sans faire aucune référence aux propriétés de partition. Ceci s'illustre par l'ensemble des objectifs suivants, et où l'on constate qu'après simplification de `stylo2`, que `stylo1` et `stylo2` sont deux descriptions totalement équivalentes ; le domaine de validité de l'angle d'incidence, appelé aussi cosinus directeur, (angle de l'axe principal d'inertie avec l'axe horizontal) étant $\Delta_{angi} = [-90, 90]$.

```
stylo1
          '((0 256) (0 256))
          '(((prmt 350 65536.) ((vois4) (objets*k))))
          '(gl)
          'vab
          '(1 2)
stylo1_horizontal
          '((0 256) (0 256))
          '(((angi -45 45) ((vois4) (stylo1*k))))
          '(gl)
          'vab
          '(1 2)
stylo1_vertical
          '((0 256) (0 256))
          '(((angi -90 -45 45 90.001) ((vois4) (stylo1*k))))
          '(gl)
          'vab
          '(1 2)
stylo2
          '((0 256) (0 256))
          '(stylo1_horizontal*k stylo1_vertical*k)
          '(gl)
          'vab
          '(1 2)
```

Preuve :

stylo2.τ = `'((0 256) (0 256))` = stylo2.τ

stylo2.ι = `'(gl)` = stylo2.ι

stylo2.σ = `'vab` = stylo2.σ

stylo2.ω = `'(1 2)` = stylo2.ω

stylo2.κ = `'(stylo1_horizontal*k stylo1_vertical*k)`

```
    = '(((angi -45 45) ((vois4)(stylo1*k)))
        ((angi -90 -45 45 90.001) ((vois4)(stylo1*k))))
```

```
= '(((angi -90 -45 -45 45 45 90.001) ((vois4)(stylo1*k))))
= '(((angi -90 90.001) ((vois4)(stylo1*k))))
= 'stylo1*k = stylo1.κ
```

Des simplifications plus complexes vont faire intervenir des connaissances particulières sur des descripteurs. Celles-ci constituent une forme de connaissance experte en traitement d'images. Ainsi, nous trouverons parmi elles les caractéristiques liées au calcul des ordres[1] d'une composante connexe et de son complémentaire qui fait l'objet d'une règle de manipulation de concepts[2]. Ces caractéristiques permettent d'exprimer en terme d'indices visuels des propriétés spécifiques de certains opérateurs de traitement d'images comme l'idempotence des opérateurs morphologiques d'ouverture et de fermeture.

2.1.2 Intérêt pour la planification

L'intérêt pour la planification est sans doute le plus important. En effet, comme nous l'avons dit en introduction, le module de résolution d'objectif génère des descriptions de données à partir des opérations de transformations associées aux opérateurs symboliques mis en jeu et de la description du but à atteindre. Il faut donc considérer deux caractéristiques particulières. La première porte sur l'appariement entre les descriptions construites et celle de l'objectif, tandis que la seconde porte sur la cohérence du plan construit.

2.1.2.1 Pour un bon appariement

Nous pouvons concevoir l'appariement entre les deux types de description selon les trois façons rapportées sur la figure 2.1

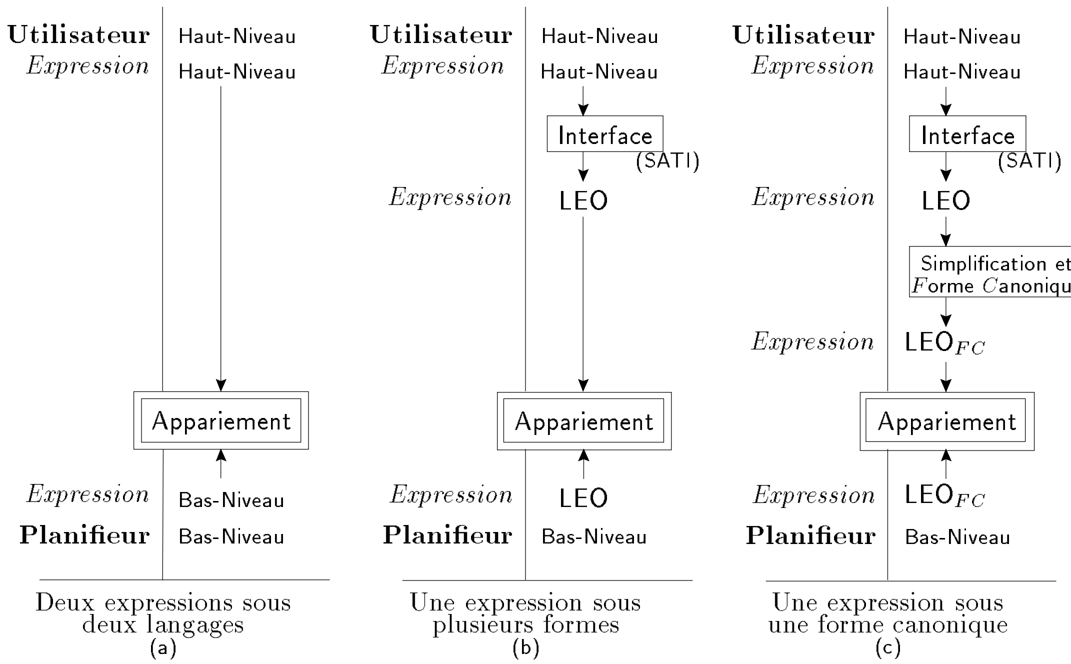


FIG. 2.1 - *Différents appariements envisageables*

1. distance au bord de la composante connexe
2. règle 5 – Annexe F

Dans le premier cas (a), le travail du module d'appariement des expressions doit établir des correspondances entre deux expressions écrites dans deux langages de haut et bas-niveau, tout à fait différents. Ce problème est classiquement résolu (b) en proposant un langage de bas-niveau d'expression et de description des données construites par planification. LEO a été écrit dans ce sens et permet, via une interface utilisateur, d'exprimer des objectifs en termes d'indices visuels. Les expressions construites et celles posant le problème étant générées par deux systèmes distincts, le module d'appariement doit, dans ce cas, effectuer le lien entre ces expressions en tenant compte de caractéristiques telles que l'organisation des concepts dans une disjonction, la redondance de termes, etc. Ceci impose de concevoir un module robuste de reconnaissance de ces expressions.

Le principe de la simplification et de la mise en forme canonique des expressions (c) répond précisément à ce problème en proposant, au module d'appariement, deux expressions respectant des contraintes d'écritures. D'un côté, la description de l'objectif est analysée, simplifiée et réduite dès le problème soumis au processus de résolution. De l'autre côté, les transformations d'informations sont effectuées par les opérateurs symboliques au regard de règles de simplification et de réduction. Ainsi, le système d'appariement est allégé de ces phases de traduction et de mise en forme d'expressions, ce qui diminuera d'autant le temps de planification.

2.1.2.2 Pour un plan cohérent

Le module de simplification d'expressions permet en outre d'améliorer la lisibilité des expressions générées automatiquement et, de ce fait, de mieux appréhender les entités représentées. Ainsi la description de la donnée `g161` générée par le module de planification implanté dans ExTI ([Ing92]) peut s'exprimer plus simplement en `g23`, donnée issue de la planification du même objectif à l'aide d'un module intégrant la simplification d'expression.

```
#:objet:ib:--g161--
<vab=1>=(((intersecte (((nivg >= 100))
                        (dans (((nivg < 100))) (((nivg >= 100)))))
                       (((mom_xx < 1) (surface >= 6) (((nivg < 100))))))))
<vab=0>=(((hors (((nivg < 100))) (((nivg >= 100))))
          (intersecte (((nivg >= 100)) (dans(((nivg < 100))) (((nivg >= 100)))))
                      (((mom_xx < 1) (((nivg < 100))))))
         ((hors (((nivg < 100))) (((nivg >= 100))))
          (intersecte (((nivg >= 100)) (dans (((nivg < 100))) (((nivg >= 100)))))
                      (((mom_xx < 1) (surface >= 6) (((nivg < 100)))))))
         ((intersecte (((nivg >= 100)) (dans (((nivg < 100))) (((nivg >= 100)))))
                      (((mom_xx < 1) (((nivg < 100))))))
          (nintersecte (((nivg >= 100)) (dans (((nivg < 100))) (((nivg >= 100)))))
                       (((mom_xx < 1) (surface >= 6) (((nivg < 100)))))))
         ((intersecte (((nivg >= 100)) (dans (((nivg < 100))) (((nivg >= 100)))))
                      (((mom_xx < 1) (surface >= 6) (((nivg < 100))))))
          (nintersecte (((nivg >= 100)) (dans(((nivg < 100))) (((nivg >= 100)))))
                       (((mom_xx < 1) (((nivg < 100)))))))
         (hors (((nivg < 100))) (((nivg >= 100))))
         (nintersecte (((nivg >= 100)) (dans (((nivg < 100))) (((nivg >= 100)))))
                      (((mom_xx < 1) (surface >= 6) (((nivg < 100))))))))
-------------------------
g23 : #:objet:ib
d : #[((0 256) (0 256))
```

```
      ((inter ((vois4) (((nivg 100 256))
                         (dans ((vois4) (((nivg 0 100)))) (((nivg 100 256))))))
               (((mxx 0 1) (srf 6 2000) ((vois4) (((nivg 0 100))))))))
      (gl) vab (1 2)]
do : #[((0 256) (0 256))
       ((hors ((vois4) (((nivg 0 100)))) (((nivg 100 256))))
        (ninter ((vois4) (((nivg 100 256))
                           (dans ((vois4) (((nivg 0 100)))) (((nivg 100 256))))))
                 (((mxx 0 1) (srf 6 2000) ((vois4) (((nivg 0 100))))))))
       () vab (0 1)]
```

Enfin, une conséquence plus gênante d'absence de simplification et de réduction est la génération de deux descriptions distinctes d'un même objet. Lorsque de tels événements interviennent en cours de planification, les deux entités sont prises comme point de départ potentiel d'une recherche de solution dans le graphe d'opérateurs (fig.2.2). Ceci aura pour effet de surcharger la résolution.

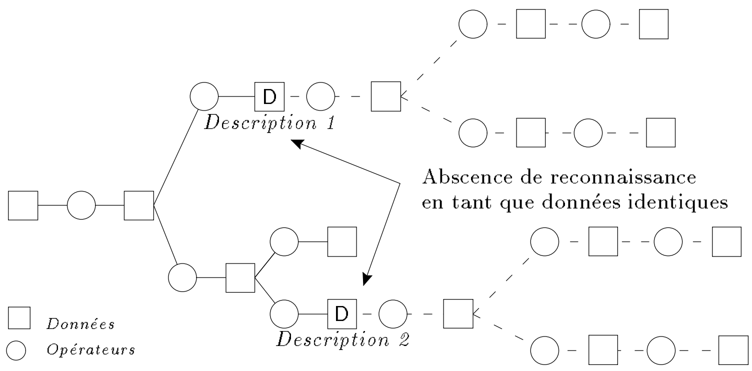


FIG. 2.2 - *Absence de simplification dans un graphe d'opérateurs*

2.2 Distances spatiales

2.2.1 Rappel de définitions

2.2.1.1 Rayon d'Examen d'un attribut

Le Rayon d'Examen (RE) d'un attribut est le rayon maximum de la zone à examiner pour déterminer la valeur de l'attribut pour le type considéré. Ce rayon est exprimé en pixels et est mesuré suivants les axes horizontaux et verticaux. Des exemples simples sont :

- $RE(nivg) = 0$: $nivg$ qualifie l'entité pixel
- $RE(surface) = 1$: $surface$ qualifie l'entité composante connexe
- $RE(distance) = 1$: $distance$ qualifie l'entité pixel dans un voisinage 3x3

2.2.1.2 Distance Spatiale d'un concept

La notion de Distance Spatiale (DS) a été introduite par J.M. Inglebert [Ing92] afin de déterminer les conditions spatiales de structuration des concepts et les règles relatives à ces structurations. Cette distance est mesurée suivant les axes horizontaux et verticaux, est exprimée en pixels et correspond à la distance minimale séparant deux instances quelconques du concept considéré.

Voici deux petits exemples pour illustrer cette notion :

1. Soit le concept [[$nivg$ 0 128]].
   Son type[3] est le pixel.
   Deux instances (pixels) de ce concept sont séparées au minimum d'aucun pixel.
   Donc $DS([nivg\ 0\ 128]) = 0$.
2. Soit le concept [[$surface$ 1000 2000] & [$nivg$ 0 128]].
   Son type est la composante connexe.
   Deux instances (composantes connexes) de ce concept sont séparées au minimum d'un pixel.
   Donc $DS([[surface\ 1000\ 2000]\ \&\ [nivg\ 0\ 128]]) = 1$.

2.2.1.3 Distance Spatiale d'une Disjonction

Alors que la DS est relative à deux instances d'un même concept, la distance spatiale d'une disjonction (DSD) fait référence pour sa part de la distance entre deux instances de deux concepts distincts. Elle traite ainsi les intersections totalement ou partiellement recouvrantes qui peuvent se produire dont nous rappelons les définitions :

Intersections totalement recouvrantes :
Si deux concepts possèdent une intersection non vide et que seules des entités entières appartenant à l'un quelconque des concepts regroupés constituent l'intersection, alors nous dirons que ces concepts possèdent une intersection totalement recouvrante.

La règle générale est alors :
$DSD(C_1 \vee C_2) = DSD(C_1 \vee C_2\ /\ (C_1 \cap C_2))$ avec $(C_1 \cap C_2)$ totalement recouvrante.

Intersections partiellement recouvrantes :
Lorsque l'intersection de deux concepts est constituée d'entités n'appartenant à aucun des concepts initiaux, nous dirons que ces concepts s'intersectent partiellement.

La règle générale est alors :
$DSD(C_1 \vee C_2) = 0$ avec $(C_1 \cap C_2)$ partiellement recouvrante.

2.2.1.4 Intérêt pour la simplification

Les notions de Distance Spatiale et de Distance Spatiale d'une Disjonction ne peuvent suffire pour la simplification d'expressions car elles mettent en jeu des instances particulières des concepts qu'elles mesurent, alors que, dans le cadre de la simplification d'expressions, nous souhaitons travailler sur les concepts eux-même.
L'exemple cité dans [Ing92][4] montre bien le problème de leur utilisation dans les simplifications.
Soient les concepts :

$$\begin{array}{lcll} K_3 & = & [surface(cc) >= v_2]\text{«}[nivg < v_1]\text{»} & DS(K_3) = 1 \\ K_4 & = & [surface(cc) < v_3]\text{«}[nivg < v_1]\text{» avec } v_3 \leq v_2 & DS(K_4) = 1 \\ K_5 & = & [surface(cc) < v_3]\text{«}[nivg >= v_1]\text{»} & DS(K_5) = 1 \end{array}$$

La Distance Spatiale de la Disjonction de ces concepts est théoriquement nulle alors que si aucun pixel de valeur inférieure à v_1 n'est voisin d'aucun pixel de valeur supérieure à v_1 la DSD est égale à 1 : $DSD(K_3 \vee K_4) = 1, DSD(K_3 \vee K_5) = 0, DSD(K_4 \vee K_5) = 0$

3. Le type d'un concept correspond au type de l'entité caractérisée par ce concept
4. pages 118 et 119 ; les notations utilisées sont reportées en annexe F

Se référer aux instances de concepts afin de simplifier leurs expressions pose donc quelques problèmes. De ce fait, bien qu'elles soient importantes et incontournables, les cinq règles de manipulation de concepts telles ont été proposées dans [Ing92], et que nous avons rappelé en annexe F, se trouvent difficilement exploitables pour mettre en pratique la simplification des expressions.

2.2.2 Distance Théorique

Nous allons introduire la notion de Distance Théorique (DT) entre deux concepts en étendant la notion de Rayon d'Examen au niveau du vecteur de propriétés (RE_v) puis au niveau du concepts (RE_c).

RE_v : Le Rayon d'Examen d'un vecteur de propriété sera égal au maximum des rayons d'examen des attributs des propriétés qui le composent.

RE_c : Plusieurs Rayons d'Examen d'un concept sont à envisager selon comment se dérive la catégorie syntaxique Cpt décrivant les concepts. Ainsi, dans le cas d'un vecteur de propriétés $Cpt \longrightarrow (Ppt^{+}\{\lambda \mid Dst\})$, ce rayon RE_c est égal au maximum des rayons d'examen des vecteurs de propriétés qui le compose, tandis que le RE_c d'une relation -$Cpt \longrightarrow (\rho\ Dst\ Bse^{+})$- est égal à celui de la base déstructurée Dst, élément principal du concept car spécialisé par la relation ρ au regard des bases Bse^{+}. Enfin, dans le cas d'une jonction $Cpt \longrightarrow (\mu\ Bse\ Bse)$, il est égal au maximum des RE_c des bases le composant, alors que celui d'une figure -$Cpt \longrightarrow (\phi\ Mes^{+})$- est toujours égal à un.

Exemples :

$C = [[nivg\ 100\ 200]]$

$$\begin{aligned} RE_c(C) &= RE_v([nivg\ 100\ 200]) \\ &= RE(nivg) \\ &= 0 \end{aligned}$$

$C = [[surface\ 1000\ 2000]\ \&\ [[nivg\ 100\ 200]]]$

$$\begin{aligned} RE_c(C) &= max(RE_v([surface\ 1000\ 2000]), RE_v([nivg\ 100\ 200])) \\ &= max(RE(surface), RE(nivg)) \\ &= max(1, 0) \\ &= 1 \end{aligned}$$

Le Rayon d'Examen d'un concept (RE_c) correspond donc à la Distance Spatiale (DS) théorique de ce concept et peut être calculé à partir des attributs de l'expression de ce concept.

2.3 Principes de la simplification d'expressions

Chaque champ de la donnée va devoir être soumis à des règles de simplifications. Ces dernières vont en fait être réalisées au niveau de certaines catégories syntaxiques du LDC, celles faisant intervenir plusieurs catégories de même nature. Si nous reprenons ces catégories, nous pouvons dégager sous ce critère, les sept dérivations suivantes qui vont faire l'objet des règles de simplification :

<fonction_sur_des_mesures>	::=	(opérateur *<mesure>* *<mesure>*)
<domaine>	::=	[*<mesure>* *<mesure>*]*
<concept>	::=	(*<propriété>*+)
	|	(*<propriété>*+ *<déstructuration_d'une_base>*)
	|	(relation *<déstructuration_d'une_base>* *<base>*+)
	|	(jonction *<base>* *<base>*)
<base>	::=	(*<concept>**)

2.3.1 Spécificités de notre langage d'expression

Une grande partie des éléments manipulés dans LEO est empruntée au calcul des prédicats. L'idée première pour gérer les simplifications de description était d'implanter des outils classiquement connus de minimisation d'expressions logiques.
Mais qu'il s'agisse des méthodes par les tables dites de Karnaugh, ou celles basées sur les représentations de Horn [Hor51] comme la méthode de Quine-McCluskey ([Qui52],[McC56]), les entités manipulées sont binaires et telles que si $x \vee y$ est une tautologie, alors $y = \overline{x}$ et $\forall(u, v), y \neq u \vee v$.

Or, nous manipulons des entités plus complexes pouvant partitionner l'élément global `t` (tautologie) en n parties disjointes. Ceci peut se rapprocher de travaux en logique multi-valuée introduite par E.Post (1921) [Gin91].
Exemple : n=4
`((nivg 0 50))`$\vee$`((nivg 50 100))`$\vee$`((nivg 100 200))`$\vee$`((nivg 200 256))` $\models$ `((nivg 0 256))` $\models$ `t`
avec toujours les relations sur les complémentaires :
`((nivg 50 100))` $\vee$ `C` $\models$ `t` $\Longrightarrow$ `C = ((nivg 0 50 100 256))`

De fait, les méthodes de la logique classique, bien qu'intéressantes et robustes ne peuvent cependant gérer qu'une partie des problèmes qui sont posés par la simplification des expressions de LEO. Une autre raison de la difficulté d'adapter ces outils ainsi que ceux proposés en logique multi-valuée, est dûe à la présence des opérations de structuration et de déstructuration. Ces opérations ont étés rajoutées au calcul des prédicats annotés et donnent la possibilité de manipuler des entités et sous-entités de traitement d'images de types différents : pixels, composantes connexes, etc.

2.3.2 Simplification d'une fonction sur des mesures

La simplification d'une fonction sur des mesures va dépendre des opérateurs effectifs de ce calcul. Les simplifications seront celles de l'arithmétique classique avec l'évaluation de termes.

```
(+ 1 2 (* 1 2) a b (* 20 (* (+ 5 5) d)) c 4)
                                ⊨ (+ (* 200 d) (+ 9 (+ a (+ b c))))
(- 1 2 (* 1 2) a b (* 20 (* (+ 5 5) d)) c 4)
                                ⊨ (+ (- (* 200 d)) (+ (- a) (+ (- b) (+ (- c) -7))))
(+ 5 a (- c (+ 1 2 3) 5))
                                ⊨ (+ -6 (+ a c))
```

Après la simplification de l'expression, la forme canonique des fonctions sur des mesures est déterminée pour obtenir une expression constituée d'opérations binaires ordonnées alpha-numériquement.
Ainsi les deux expressions suivantes se simplifient et se réduisent en la même expression.
Simplification :

```
         (+ (+ 2 b) (+ (+ (* 20 (* (* 5 5) d)) (+ c 4)) a) 2)   ⊨   (+ 8 b (* 500 d) c a)
 (+ 2 (+ (/ 4 2) (+ a (+ b (+ c (+ 4 (* 20 (* (* 5 5) d))))))))   ⊨   (+ 8 a b c (* 500 d))
```

Forme canonique :

```
 (+ 8 b (* 500 d) c a)   ⊨   (+ (* 500 d) (+ 8 (+ a (+ b c))))
 (+ 8 a b c (* 500 d))   ⊨   (+ (* 500 d) (+ 8 (+ a (+ b c))))
```

Remarque : Nous n'avons pas tenu compte dans ces règles de simplification des problèmes liés à la factorisation et la distributivité. En effet comme nous l'avons évoqué en parlant du contexte de l'application, le module de planification est développé de telle sorte que lors de l'analyse de l'objectif, les différents paramètres numériques utilisés sont créés comme des données de traitement

d'images de dimension 0. Ces données serviront à l'instanciation des opérateurs symboliques lors du choix des opérateurs, dans une phase de la planification. Le fait de ne pas réduire ces expressions factorisables comme les deux ci-dessous, n'empêchera nullement d'obtenir un résultat. Ce dernier ne sera pas optimum du fait que l'expression initiale n'était pas simplifiée. Le principal étant que la forme canonique soit générée, elle seule garantissant l'obtention de ce résultat.
Défaut de simplification :

`(+ (* a 6) (* a 7))` $\models$ `(+ (* 6 a) (* 7 a))`

`(* a 42)` $\models$ `(* 42 a)`

2.3.3 Simplification des domaines

2.3.3.1 Organisation spatiale des domaines de propriétés

Cette nouvelle notion a pour but d'exploiter les caractéristiques ensemblistes des domaines de deux propriétés ayant le même sélecteur (attribut). L'organisation spatiale des domaines de propriétés (OSD) correspond en fait à une adaptation des relations de Allen développées pour le maintient de la connaissance dans les intervalles temporels. Dans [All83, All84] il énonce les treize relations gérants ces intervalles :

Relation	*Symbol*	*Symbol for Inverse*	*Pictorial Example*
X before Y	$<$	$>$	XXX YYY
X equal Y	$=$	$=$	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

Les notions de début et de fin d'intervalle n'ont pas autant d'importance pour les propriétés que nous manipulons que dans les problèmes liés au temps. Ainsi, nous considérons les relations s, f et si, fi comme des cas particuliers des relations d et di. À l'inverse, si la notion de temps implique des intervalles n'ayant ni début ni fin, les domaines des propriétés sont tous inclus dans les domaines de validité qui sont eux-même des intervalles. Les relations d'Allen notées o, oi, m et mi doivent donc être dupliquées afin de mettre en exergue la complétude des domaines considérés par rapport au domaine de validité. De plus, le temps est une notion continue alors que nos domaines peuvent être constitués d'une union d'intervalles.

Pour tenir compte de toutes ces spécificités, nous proposons donc une nouvelle classification des relations d'intervalles basée sur leur union et leur intersection. Les trois critères de l'intersection seront l'ensemble vide $\emptyset$, l'intervalle $[x, y[$, et l'union d'intervalles $\cup[x, y[$. Les trois critères de l'union seront l'intervalle $[x, y[$, le domaine de validité Δ_p -qui est un cas particulier du premier- et l'union d'intervalles $\cup[x, y[$. Dans le tableau suivant[5], ainsi que dans la figure 2.3, nous présentons les neuf Organisations Spatiales des Domaines en précisant les relations de Allen équivalentes.

5. lignes : critères de l'intersection — colonnes : critères de l'union

Types	$d_1 \cup d_2 = [x, y[$	$d_1 \cup d_2 = \Delta_p$	$d_1 \cup d_2 = \cup[x, y[$
$d_1 \cap d_2 = \emptyset$	1 (m, mi)	2	3 $(<, >)$
$d_1 \cap d_2 = [x, y[$	4 $(o, oi, =, d, di)$	5	6
$d_1 \cap d_2 = \cup[x, y[$	7	8	9

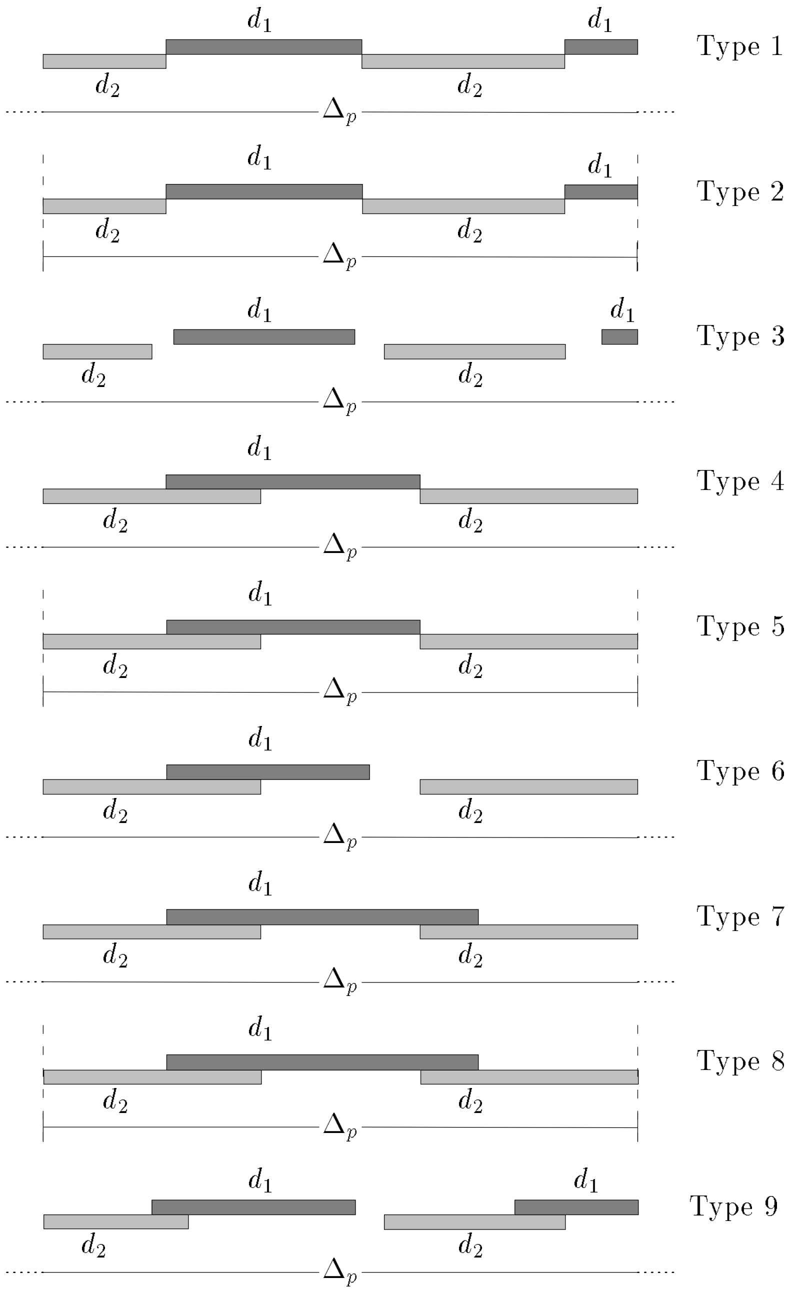


FIG. 2.3 - *Organisation Spatiale des Domaines de propriétés*

2.3.3.2 Règles de simplification et de manipulation

La simplification des domaines -union d'intervalles-, comme la détermination de leur complémentaire, ou de leur intersection est extrèmement facilité par notre choix initial d'intervalles semi-ouverts à droite et par les domaines de propriété auxquels ils sont liés lors de leur utilisation dans des catégories syntaxiques de plus haut niveau d'imbrication.

Union de deux domaines :
Soient $D_k = \bigcup_{i=1}^{k}[x_i, y_i[$ et $D_l = \bigcup_{j=1}^{l\leq k}[a_j, b_j[$ deux domaines syntaxiquement corrects. L'union $D_k \cup D_l$ de ces deux domaines est déterminée par modification de certains intervalles D_l par ceux de D_k. Ces modifications faisant références aux intervalles $[x_i, y_i[$ et $[a_j, b_j[$ sont données par les règles suivantes :

1. Si $x_i \leq a_j$ et $a_j \leq y_i \leq b_j$ alors $[x_i, b_j[$ (relations de Allen : s,si,f,o,m)
2. Si $x_i \leq a_j$ et $b_j \leq y_i$ alors $[x_i, z[$ (relation de Allen : di)
   avec $z = y_i$ si $y_i < a_{j+u}$ quel que soit $u > 0$
   $z = b_{j+u}$ s'il existe un $u > 0$ minimal tel que $a_{j+u} < y_i \leq b_{i+u}$
3. Si $a_j < x_i < b_j$ et $y_i \leq b_j$ alors $[a_j, b_j[$ (relations de Allen : d,fi)
4. Si $a_j < x_i < b_j$ et $b_j < yi$ alors $[a_j, z[$ (relations de Allen : oi,mi)
   avec z déterminé comme précédemment.

Intersection de deux domaines :
Soient $D_k = \bigcup_{i=1}^{k}[x_i, y_i[$ et $D_l = \bigcup_{i=1}^{l\leq k}[a_i, b_i[$ deux domaines syntaxiquement corrects. L'intersection $D_k \cap D_l$ de ces deux domaines est déterminée en ne conservant que les valeurs des domaines telles que :

v est borne inférieure d'un intervalle de D_k et d'un intervalle de D_l
v est borne supérieure d'un intervalle de D_k et d'un intervalle de D_l
$v \in D_k$ (respectivement D_l) et v compris entre les bornes d'un intervalle de D_l (respectivement D_k)

Complémentaire d'un domaine :
Soient $D_k = \bigcup_{i=1}^{k}[x_i, y_i[$. Le complémentaire de D_k au domaine de validité Δ est déterminé en établissant tout d'abord la liste ordonnée de ses valeurs d'intervalles. Ensuite, si la valeur minimale du domaine de validité auquel le domaine est lié est égale à la première valeur de la liste alors cette valeur est retirée de la liste, sinon elle est rajoutée à la liste. Le même traitement est effectué pour la valeur maximale du domaine de validité en considérant cette fois la valeur de fin de liste. Chaque couple de la liste alors constituée détermine un intervalle du complémentaire.

$$\sim D_{k_{|\Delta}} = [min\{\Delta\}, x_1[\ \cup \ \left\{ \bigcup_{i=1}^{k-1} [y_i, x_{i+1}[\right\} \ \cup \ [y_k, max\{\Delta\}[$$

2.3.4 Simplification des vecteurs de propriétés

Les simplifications de vecteurs de propriétés peuvent être réparties en deux groupes ; l'un mettant en jeu l'élément déstructurant du concept (*<propriété>*$^+$*<déstructuration_d'une_base>*), l'autre traitant plus particulièrement des propriétés elle-mêmes (*<propriété>*$^+$). Dans ce dernier cas, il peut être convenu qu'il existe une base déstructurée $\mathcal{B}$ décrivant toute entité ; de fait $\mathcal{B}$ est une déstructuration quelconque de la base Bse ::=t.

Les simplifications que nous abordons ici considèrent uniquement un ensemble de propriétés, celles ci étant organisées de façon conjonctive.

$$P_1 \ \& \ P_2 \ \& \ \ldots \ \& \ P_m \ ::> \ V$$

2.3.4.1 Notions particulières sur les propriétés

Soit P une propriété, nous désignerons par $Desc(P)$ son descripteur -sélecteur- et par $Dom(P)$ son domaine. Ainsi :

$$P = (p\ x_1\ y_1\ \dots\ x_n\ y_n) \text{ et } Desc(P) = p\ ,\ Dom(P) = \cup_{i=1}^{n}[x_i, y_i[$$

Pour faciliter les écritures des règles de simplification, nous allons définir les trois notions particulières suivantes :

Propriété vide

Quel que soit un descripteur p, il existe une infinité de propriété vide p^o liée à p. Une propriété P est dite vide si elle possède un domaine tel que $Dom(P) = [x, x[= \emptyset$.

$$p^o = (p\ \{\cup_{i=1}^{n}[x_i, x_i[\})\ ,\ \forall i \in \mathbb{N}$$

Propriété pleine

Quel que soit un descripteur p, il existe une unique propriété pleine p^Δ liée à p. Une propriété P est dite pleine si elle posséde un domaine tel que $Dom(P) = \Delta_{Desc(P)}$.

$$p^\Delta = (p\ \Delta_p)$$

Propriété complémentaire

Quelle que soit une propriété P, il existe une unique propriété complémentaire à P. Une propriété $\sim P$ est dite complémentaire de P si elle posséde le même descripteur et le domaine complémentaire du domaine de P.

$$\sim P = (Desc(P)\ \sim Dom(P))$$

2.3.4.2 Principe de simplification

L'objectif de la simplification d'un vecteur de propriétés est d'obtenir une suite de propriétés à descripteurs distincts. La forme canonique liée à un tel vecteur est donnée par arrangement alphanumérique de ces propriétés. Un vecteur de propriétés ne peut se simplifier que s'il contient des propriétés vides, pleines ou à descripteurs identiques. Sa simplification est obtenue en appliquant les règles suivantes :

Règle 1 : Simplification par suppression des propriétés pleines.

Une propriété pleine p^Δ dans un vecteur de propriétés ne détermine aucune partition des éléments ainsi spécialisés car elle envisage toutes les valeurs possibles -Δ_p- pour la mesure p.
La simplification d'un tel vecteur va donc consister en la suppression des propriétés pleines.

Si $P_1\ \&\ \dots\ \&\ P_{i-1}\ \&\ P_i\ \&\ P_{i+1}\ \&\ \dots\ \&\ P_m\ ::>\ V$
et si $P_i = Desc(P_i)^\Delta$
alors $V\ \models\ P_1\ \&\ \dots\ \&\ P_{i-1}\ \&\ P_{i+1}\ \&\ \dots\ \&\ P_m$

Exemple :
Sur une image de taille 512×512, $\Delta_{diametre\_x} = [1, 513[$

((diametre_x 1 513) (surface 10 120) ((vois4) (((nivg 100 200)))))
$\models$ ((surface 10 120) ((vois4) (((nivg 100 200)))))

Règle 2 : Simplification par annulation du vecteur.

La présence d'une propriété vide dans un vecteur témoigne d'une impossibilité d'affecter des mesures de partition au descripteur désigné. Ceci aura donc pour effet d'annuler la totalité du vecteur, quelles que soient les autres propriétés.

Si $P_1\ \&\ \dots\ \&\ P_{i-1}\ \&\ P_i\ \&\ P_{i+1}\ \&\ \dots\ \&\ P_m\ ::>\ V$
et si $P_i = Desc(P_i)^o$
alors $V\ \models\ \lambda$

Exemple : ((diametre_x 20 20) (surface 10 120) ((vois4) (((nivg 100 200))))) $\models$ ()

Règle 3 : Simplification par fusion de propriétés à descripteur identique.

La simplification de deux propriétés à descripteur identique présentes dans un même vecteur de propriétés va s'effectuer en les fusionnant en une nouvelle propriété P de même descripteur. Le domaine de P est alors constitué de l'intersection des domaines des propriétés initiales. Les règles 1 et 2 s'appliqueront à nouveau sur le vecteur nouvellement simplifié.

2.3.5 Simplification des relations

S'il n'existe pas *a priori* de règles génériques permettant de gérer les simplifications de concepts relationnels, chaque descripteur de relation est toutefois lié à un ensemble de règles spécifiques témoignant d'une connaissance experte en traitement d'images. Ces ensembles de règles seront développés au regard du langage d'expression d'objectif et des descripteurs pouvant être mis en jeu dans ces concepts. Elles auront la forme de fonction de manipulation d'expressions du langage.
L'introduction d'une nouvelle relation dans le système s'accompagnera alors de la constitution de cet ensemble de règles ainsi que de l'enrichissement éventuel des ensembles attachés aux autres relations prenant en compte le nouveau descripteur. Il faut toutefois noter que, généralement, il s'agit en fait d'un couple de descripteurs de relation, étant donnée la dualité de la description de la donnée et de l'obligation de décrire une entité et son complémentaire.

Une relation $R = (\rho\ Dst\ Bse)$ est une spécialisation de la disjonction de concepts déstructurée Dst par une caractéristique relationnelle décrite par la base Bse. Comme toute entité issue d'une partition d'un concept de plus haut niveau, le concept spécialisé est présent dans le concept initial. Nous parlerons alors plus facilement d'inclusion. Ainsi, le concept relationnel R est inclus dans la base déstructurée Dst qu'il spécialise.

$$R = (\rho\ Dst\ Bse) \subset Dst$$

Exemples : soit $d = ((\zeta)\mathcal{B})$
Règles liées au descripteur intersection :
(inter d b) $\models \mathcal{B}$ si $\mathcal{B} \trianglerighteq b$
(inter d b) $\models \lambda$ si $DS(\mathcal{B}, b) = 0$
Règles liées au descripteur non intersection :
(ninter d b) $\models \lambda$ si $\mathcal{B} \trianglerighteq b$
(ninter d b) $\models d$ si $DS(\mathcal{B}, b) = 0$

2.4 Simplification des concepts disjonctifs et des bases

2.4.1 Remarques sur ces simplifications

Il existe de nombreuses façons de considérer les disjonctions [Jen94]. Certaines, plus proches de spécificités propres aux langages naturels, doivent toutefois être prises en compte lors d'acquisition, par un module d'interaction, des requêtes formulées par un utilisateur.
Les deux types de disjonction (ou,xou) présents dans LEO sont des représentations classiques rencontrées en logique et définies par :

$x \in (a\ ou\ b)$ si une au moins des deux assertions suivantes est vrai :

$$x \in a\ ,\ x \in b$$

$x \in (a\ xou\ b)$ si une et une seule des deux assertions suivantes est vrai :

$$x \in (a \cup b)_{|b}\ ,\ x \in (a \cup b)_{|a}$$

Le concept établissant la disjonction exclusive de deux bases ne peut être simplifié car il effectue une partition du cache de la donnée. Les autres jonctions (disjonction, conjonction) sont régies par le même principe général de simplification. Ce principe consiste en la minimisation du nombre d'éléments manipulés.

Le cas de la disjonction non exclusive est particulier à deux titres. Tout d'abord, parce que l'élément disjonctif d'un tel concept n'apparaît que lorsque l'expression provient du remplacement de noms de concepts du domaine d'application par leur description, ensuite parce que la forme unique et canonique d'une base est la disjonction de concepts. La simplification d'un concept disjonctif va donc consister en la suppression de son caractère disjonctif propre en remontant les concepts des bases de cette jonction au niveau de celle la contenant. Nous obtenons ainsi une base qu'il faut alors simplifier par des règles spécifiques.

Exemple :

$$(C_1 \ \ldots \ C_i \ (\texttt{ou} \ (C_{a_1} \ \ldots \ C_{a_l}) \ (C_{b_1} \ \ldots \ C_{b_m})) \ C_j \ \ldots \ C_n)$$
$$\models (C_1 \ \ldots \ C_i \ C_{a_1} \ \ldots \ C_{a_l} \ C_{b_1} \ \ldots \ C_{b_m} \ C_j \ \ldots \ C_n)$$

La simplification la plus complexe est celle gérant la réduction d'une base dont la structure est, rappelons-le, la disjonction de concepts. Le concept pouvant être formé de divers éléments -vecteur de propriétés, fonction sur des mesures, figure, relation ou jonction-, la simplification de la disjonction dépend donc de la structure des concepts la composant. Le principe général de la simplification réside toutefois dans la réduction d'une disjonction de deux concepts à un unique concept ; la simplification d'une disjonction de n concepts se faisant par examen de ses concepts deux à deux.

Dans tous les cas faisant intervenir des caractéristiques de déstructuration de bases, ces éléments ne pourront être simplifiés lorsqu'ils sont manipulés avec d'autres bases -*Dst* ou *Bse*- que si les terminaux déstructurant sont identiques. Cette règle fondamentale étant énoncée, nous nous permettrons, dans la suite du chapitre, de faire abstraction de ce terminal en supprimant les catégories $Dst = ((\zeta)\mathcal{B})$ pour les remplacer par de simples bases $Bse = \mathcal{B}$.

2.4.2 Extension de la notion d'organisation spatiale

La notion d'organisation spatiale des domaines de propriétés peut être étendue aux concepts en considérant, cette fois, les propriétés d'inclusion -généralisation /spécialisation-, d'intersection et de complétude de ceux-ci. Par analogie avec l'OSD, l'Organisation Spatiale des Concepts (OSC) possède des valeurs de type ; les définitions qui leur sont attribuées étant quelque peu modifiées pour rendre compte de leurs spécificités.

Type 1: l'OSC de deux concepts est de type 1 s'ils sont disjoints -c'est à dire qu'aucun éléments d'une instance de l'un n'appartient à une instance décrite par l'autre- et si la Distance Théorique entre ces concepts est nulle.

$$C_1 \wedge C_2 = \lambda \ , \ DT(C_1, C_2) = 0$$

Type 2: l'OSC de deux concepts est de type 2 s'ils sont disjoints et si la Distance Théorique entre ces concepts est non nulle.

$$C_1 \wedge C_2 = \lambda \ , \ DT(C_1, C_2) > 0$$

Type 3: l'OSC de deux concepts est de type 3 s'ils sont de type 1 et complémentaires.

$$C_1 \wedge C_2 = \lambda \ , \ DT(C_1, C_2) = 0 \ , \ C_1 \vee C_2 = \texttt{t}$$

Type 4: l'OSC de deux concepts est de type 4 si $C_1 \Join\!\!< C_2$.
Type 5: l'OSC de deux concepts est de type 5 si $C_1 \rhd C_2$.
Type 6: l'OSC de deux concepts est de type 6 si $C_1 = C_2$.

2.4.3 Simplification d'une disjonction de vecteurs de propriétés

L'éventualité d'une simplification de disjonction de vecteurs de propriétés se produit lorsque les bases spécialisées par ces vecteurs sont soit identiques, soit telles que l'une est incluse dans l'autre. Ainsi, l'unique concept issu de cette simplification aura pour base, la disjonction des bases spécialisées.
Un vecteur de propriétés généré par l'expression régulière $Cpt \longrightarrow (Ppt^{+}\ Dst)$ est articulé sur une base Bse composante de Dst dont la structure est déterminante pour gérer la simplification. Nous pouvons, en effet, distinguer deux cas pour la simplification. Le premier considère les vecteurs pour lesquels les bases déstructurées sont identiques ou absentes (voir plus haut), le second considérant les couples de vecteurs à bases déstructurées distinctes.

2.4.3.1 Cas 1 : Les concepts de la disjonction ont des bases identiques

Dans ce cas, la simplification de la disjonction de concepts se fera à l'examen des propriétés propres à chaque concept. Cette simplification se déroule en quatre phases :

1. Retrait des propriétés identiques.
   Un cas particulier pouvant se produire consiste à rencontrer, dans chaque concept, une même propriété -même sélecteur, même domaine. Lorsque cette éventualité se produit, la ou les propriétés en question sont retirées de chaque concept afin de limiter l'espace de travail. On appellera *liste des propriétés identiques* (notée ci-après L^{id}) l'ensemble des propriétés ainsi retirées.
   Exemple :
   C_1 =((angi -30 30) (dx 10 257) (srf 500 2000) ((vois4) (((nivg 0 100))))) et
   C_2 =((angi -30 30) (dx 100 150) (dy 1 10) (srf 500 2000) ((vois4) (((nivg 0 100))))).
   On extrait de ces concepts la liste des propriétés identiques :
   L^{id} = {(angi -30 30),(srf 500 2000)}
2. Équilibrage des listes de propriétés.
   Lorsque les propriétés d'un concept ne sont pas présentes dans l'autre concept examiné[6], nous introduirons les propriétés pleines p^{Δ} correspondantes dans ce dernier. Ainsi, pour chaque concept de la disjonction, nous possèderons une liste de propriétés de même descripteurs. Ces listes sont dites équilibrées.
   Exemple:
   C_1 =((dx 10 257) ((vois4) (((nivg 0 100))))) et
   C_2 =((dx 100 150) (dy 1 10) ((vois4) (((nivg 0 100))))).
   On extrait de ces concepts les listes équilibrées de propriétés:
   $L^{eq}_{C_1}$ = {(dx 10 257),(dy 1 257)}
   $L^{eq}_{C_2}$ = {(dx 100 150),(dy 1 10)}
3. Fusion des listes équilibrées.
   Cette phase va consister à essayer de construire une nouvelle liste de propriétés à partir des listes de propriétés équilibrées. Ceci est obtenu en effectuant la disjonction des propriétés à même descripteur. Cette phase est la plus délicate à mettre en œuvre et fait donc l'objet du paragraphe suivant.
   Exemple :
   $L^{eq}_{C_1}$ = {(dx 10 257),(dy 1 257)}
   $L^{eq}_{C_2}$ = {(dx 100 150),(dy 1 10)}
   $Fusion(L^{eq}_{C_1}, L^{eq}_{C_2})$ = {(dx 10 257),(dy 1 257)}

6. Une propriété P est présente dans un concept si une des propriétés de ce concept a pour descripteur Desc(P).

4. Construction du vecteur.
La construction du vecteur simplifié dépend du résultat de la fusion. En effet, cet élément, issu d'une disjonction, peut être mono-terme dans le cas d'une simplification totale ou multi-terme lorsqu'il ne peut y avoir simplification. Lorsque la fusion est mono-terme, le vecteur simplifié est alors constitué de celle-ci et de la liste des propriétés identiques, le tout spécialisant la base déstructurée commune : (F & L^{id} & $\mathcal{B}$) avec $F = Fusion(L^{eq}_{C_1}, L^{eq}_{C_2})$.
Exemple :
$C \;=\; C_1 \vee C_2$
$=$ ({(dx 10 257) (dy 1 257)} {(angi -30 30) (srf 500 2000)} ((vois4) (((nivg 0 100)))))
$\models$ ((angi -30 30) (dx 10 257) (srf 500 2000) ((vois4) (((nivg 0 100)))))
$= C_1$

2.4.3.2 Disjonctions de propriétés pour la fusion de vecteurs

Nous définissons l'opération commutative de disjonction de deux propriétés possédant un même descripteur de la façon suivante :

$$\sqcup : \mathcal{P} \times \mathcal{P} \longrightarrow \mathcal{P}$$
$$P_1, P_2 \longmapsto P$$

avec $P_1 = (p\ Dom(P_1)), P_2 = (p\ Dom(P_2))$
et $P = (p\ \{Dom(P_1) \cup Dom(P_2)\})$

Listes équilibrées de propriétés à un seul élément
Le domaine de la propriété issue de la disjonction de propriétés est l'union des domaines des propriétés en question. Soient deux propriétés P_1 et P_2 ayant même descripteur p, si nous examinons les unions de domaines au regard de leur organisation spatiale, nous pouvons alors distinguer les trois cas suivants concernant les résultats de l'opération $P_1 \sqcup P_2$ (nous excluons la cas trivial $P_1 = P_2$ qui n'apparaît pas dans les listes équilibrées). Ces trois cas établiront les règles de simplification.

- Un des domaines est inclus dans l'autre, au sens purement ensembliste (cas particulier de l'OSD de type 4).
- L'union des domaines correspond au domaine de validité Δ_p (OSD de types 2,5 et 8).
- Les autres cas : l'union proprement dite est effectuée.

Première loi de simplification
Lorsque deux concepts C_1 et C_2 ont des listes de propriétés équilibrées L_1^{eq} et L_2^{eq} n'ayant qu'un élément (respectivement P_1 et P_2), on peut simplifier la disjonction de ces concepts en un nouveau concept C en respectant les règles suivantes :

Règle 1 :
L'OSC de C_1 et C_2 est de type 4 (5 par réciprocité) :
si $Dom(P_1) \subset Dom(P_2)$ alors
$C_1 \vee C_2 = C_2 \vee C_1 = C = ((Desc(P_2)\ Dom(P_2))\ \&\ L_{id}\ \&\ \mathcal{B})$
Règle 2 :
L'OSC de C_1 et C_2 est de type 3 :
si $Dom(P_1) \cup Dom(P_2) = \Delta_p$ alors
$C_1 \vee C_2 = C_2 \vee C_1 = C = (L_{id}\ \&\ \mathcal{B})$
Règle 3 :
si les règles précédentes ne peuvent s'appliquer alors
$C_1 \vee C_2 = C_2 \vee C_1 = C = ((Desc(P_1)\ \{Dom(P_1) \cup Dom(P_2)\})\&\ L_{id}\ \&\ \mathcal{B})$

avec toujours L^{id} désignant la liste des propriétés identiques et $\mathcal{B}$ la base commune aux concepts.

Listes de propriétés à deux éléments

Examinons la simplification de la disjonction des deux concepts $C_1 = (S_1P_1Q_1 \ldots Q_n \ \& \ \mathcal{B})$ et $C_2 = (S_2P_2Q_1 \ldots Q_n \ \& \ \mathcal{B})$. On détermine sur ces concepts la liste $L^{id} = \{Q1, \ldots, Qn\}$ des propriétés identiques, et $L^{eq}_{C_1} = \{S_1, P_1\}$ et $L^{eq}_{C_2} = \{S_2, P_2\}$ les listes des propriétés équilibrées à descripteurs s et p distincts.

Pour déterminer les conditions de simplification de $C_1 \vee C_2 = C_2 \vee C_1$ en C, il faut examiner les applications des règles 1 et 3 -la règle 2 est un cas particulier de la règle 3- de la première loi de simplification pour chaque couple de propriétés (S_1, S_2) et (P_1, P_2). Pour faciliter les démonstrations, la relation d'inclusion de domaines de propriétés est étendue aux propriétés par la relation de spécialisation : $P' \lessdot P$ si et seulement si $Dom(P') \subset Dom(P)$.

–Hypothèse 1 : $Dom(S_1) \subset Dom(S_2)$ (règle 1) et $Dom(P_1) \subset Dom(P_2)$ (règle 1)

$$\begin{aligned} Dom(S_1) \subset Dom(S_2) &\iff S_1 \lessdot S_2 \text{ (OSC de type 4)} \\ &\iff (S_1P_1) \lessdot (S_2P_1) \\ Dom(P_1) \subset Dom(P_2) &\iff P_1 \lessdot P_2 \text{ (OSC de type 4)} \\ &\iff (S_2P_1) \lessdot (S_2P_2) \end{aligned}$$

Des deux points précédents nous déduisons : $(S_1P_1) \lessdot (S_2P_1) \lessdot (S_2P_2)$

Conclusion :

Si $Dom(S_1) \subset Dom(S_2)$ et $Dom(P_1) \subset Dom(P_2)$
alors $Fusion((S_1P_1), (S_2P_2)) = (S_2P_2)$

–Hypothèse 2 : $Dom(S_1) \subset Dom(S_2)$ (règle 1) et $Dom(P_2) \subset Dom(P_1)$ (règle 1)

$Dom(S_1) \subset Dom(S_2) \iff S_1 \lessdot S_2$

$S_1 \lessdot S_2 \iff (S_1P_1) \lessdot (S_2P_1)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

$S_1 \lessdot S_2 \iff (S_1P_2) \lessdot (S_2P_2)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

$Dom(P_2) \subset Dom(P_1) \iff P_2 \lessdot P_1$

$P_1 \lessdot P_2 \iff (S_1P_1) \lessdot (S_1P_2)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

$P_1 \lessdot P_2 \iff (S_2P_1) \lessdot (S_2P_2)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

Conclusion :

Si $Dom(S_1) \subset Dom(S_2)$ et $Dom(P_2) \subset Dom(P_1)$
alors $Fusion((S_1P_1), (S_2P_2))$ ne se simplifie pas.

–Hypothèse 3 : $Dom(S_1) \subset Dom(S_2)$ (règle 1) et $Dom(P_1) \cup Dom(P_2) = \bigcup[x_p, y_p[$ (règle 3)

$Dom(S_1) \subset Dom(S_2) \iff S_1 \lessdot S_2$

$S_1 \lessdot S_2 \iff (S_1P_1) \lessdot (S_2P_1)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

$S_1 \lessdot S_2 \iff (S_1P_2) \lessdot (S_2P_2)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

$Dom(P_1) \cup Dom(P_2) = \bigcup[x_p, y_p[\iff P_1 \sqcup P_2 = P$

$Fusion((S_1P_1), (S_1P_2)) = (S_1P)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

$Fusion((S_2P_1), (S_2P_2)) = (S_2P)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

Conclusion :

Si $Dom(S_1) \subset Dom(S_2)$ et $Dom(P_1) \cup Dom(P_2) = \bigcup[x_p, y_p[$
alors $Fusion((S_1P_1), (S_2P_2))$ ne se simplifie pas.

–Hypothèse 4 : $Dom(S_1) \cup Dom(S_2) = \bigcup[x_s, y_s[$ (règle 3) et $Dom(P_2) \cup Dom(P_1) = \bigcup[x_p, y_p[$ (règle 3)

$Dom(S_1) \cup Dom(S_2) = \bigcup[x_x, y_x[\iff S_1 \sqcup S_2 = S$

$Fusion((S_1P_1), (S_2P_1)) = (SP_1)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

$Fusion((S_1P_2), (S_2P_2)) = (SP_2)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

$Dom(P_1) \cup Dom(P_2) = \bigcup [x_p, y_p[\Longleftrightarrow P_1 \sqcup P_2 = P$

$Fusion((S_1P_1),(S_1P_2)) = (S_1P)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

$Fusion((S_2P_1),(S_2P_2)) = (S_2P)$ donc il n'existe aucun lien entre (S_1P_1) et (S_2P_2)

Conclusion :

Si $Dom(S_1) \cup Dom(S_2) = \bigcup [x_s, y_s[$ et $Dom(P_2) \cup Dom(P_1) = \bigcup [x_p, y_p[$
alors $Fusion((S_1P_1),(S_2P_2))$ ne se simplifie pas.

Les hypothèses que nous venons de formuler font la synthèse des configurations possibles pour les OSD des couples de propriétés étudiées. Ces configurations sont reportées sur la figure 2.4 où l'on peut noter les deux seules possibilités de simplification. Ainsi donc, la disjonction de concepts de $C_1 \vee C_2 = C_2 \vee C_1$ ne se simplifie en C que lorsque les propriétés S_1, P_1 et S_2, P_2 des listes équilibrées sont telles que :

$Dom(S_1) \subset Dom(S_2)$ et $Dom(P_1) \subset Dom(P_2)$ et réciproquement.

Dans cet unique cas la fusion des listes équilibrées de propriétés est possible et le concept simplifié construit de la façon suivante :
$C = (S_2 \;\&\; P_2 \;\&\; L^{id} \;\&\; \mathcal{B})$ avec L^{id} la liste des propriétés identiques et $\mathcal{B}$ la base déstructurée commune.

Listes équilibrées de propriétés à n éléments

Le cas où les listes des propriétés ont n éléments ($n > 1$) n'est qu'une extension du cas précédent. La disjonction de concepts ne se simplifie que si toutes les propriétés de la liste équilibrée de l'un sont incluses dans les propriétés de la liste équilibrée de l'autre, au sens de l'inclusion des propriétés définie plus haut. Nous pouvons alors énoncer la seconde loi de simplification.

Deuxième loi de simplification

Lorsque deux concepts C_1 et C_2 ont des listes équilibrées de propriétés $L^{eq}_{C_1}$ et $L^{eq}_{C_2}$ ayant plus d'un élément, on peut simplifier la disjonction $C_1 \vee C_2 = C_2 \vee C_1$ de ces concepts en un nouveau concept C si et seulement si toutes les propriétés de $L^{eq}_{C_1}$ ont un domaine inclus dans le domaine de leur homologue[7] dans $L^{eq}_{C_2}$.

Le concept C s'écrit alors : $C_1 \vee C_2 = C_2 \vee C_1 = C = (L^{eq}_{C_2} \;\&\; L^{id} \;\&\; \mathcal{B})$

2.4.3.3 Cas 2 : les concepts de la disjonction ont des bases distinctes

Lorsque les bases déstructurées des concepts sont différentes, la première analyse est faite sur l'ensemble des entités décrites par ces bases. Deux cas peuvent alors se rencontrer :

1. Les types des bases sont différents :
   Lorsque les types des bases sont différents, cela signifie que l'opération de structuration appliquées aux concepts afin de produire les bases travaille sur des parties distinctes de l'espace des concepts, c'est à dire que leur structure de représentation est de nature différente. En aucun cas nous ne pouvons appliquer le premier principe de simplification : impossibilité de supprimer l'effet de l'une des bases de par leur différence de type et impossibilité d'effectuer la disjonction des bases pour la même raison.
2. Les types des bases sont identiques :
   Les bases étant de même type, les entités qu'elles décrivent peuvent se retrouver dans la même structure de représentation. Ainsi la base déstructurée du concept issu de la simplification est du type des concepts de la disjonction ; ce qui permet l'application des principes de simplification.

7. P_2 de $L^{id}_{C_2}$ est dite homologue de P_1 dans $L^{id}_{C_1}$ si Desc(P_1)=Desc(P_2)

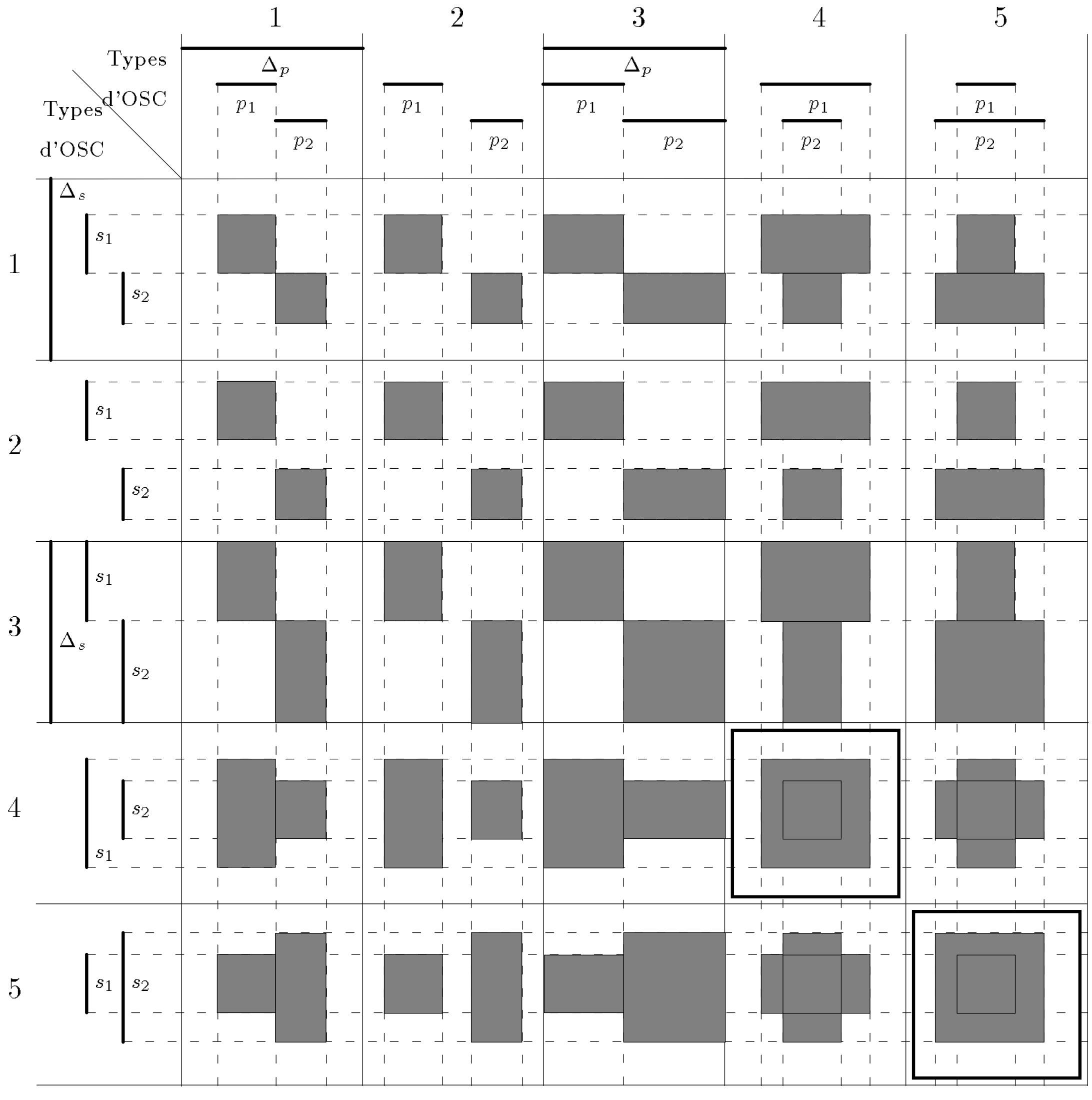


FIG. 2.4 - *Configuration de couples d'intervalles pour la simplification*

Si nous considérons les bases déstructurées sur le même plan de spécialisation que les propriétés utilisées dans les vecteurs, nous pouvons appliquer la seconde loi de simplification. De fait, si nous considérons les concepts $C_1 = (S_1P_1Q_1 \ldots Q_n \ \& \ \mathcal{B}_1)$, $C_2 = (S_2P_2Q_1 \ldots Q_n \ \& \ \mathcal{B}_2)$ et $C = (SPQ_1 \ldots Q_n \ \& \ \mathcal{B})$ le résultat simplifié de la disjonction $C_1 \vee C_2$, nous pouvons affirmer que le concept C_1 (respectivement C_2) est inclus dans le concept C_2 (respectivement C_1). Ceci nous amène à établir une troisième loi de simplification que l'on peut considérer comme une généralisation de la seconde loi.

Troisième loi de simplification :
Lorsque deux concepts C_1 et C_2 sont tels que l'un est inclus dans l'autre (leurs bases déstructurées et les propriétés homologues de leurs listes équilibrées sont incluses l'une dans l'autre), on peut simplifier la disjonction de ces concepts en un nouveau concept C. Ce dernier sera égal au concept le plus englobant.

2.4.4 Simplification d'une disjonction de relations

Comme dans le cas de la simplification de relations, des caractéristiques de réduction peuvent être déterminées pour la disjonction de concepts dont un, au moins, est une relation. De façon analogue les règles gérant ces simplifications sont liées aux descripteurs ou couples de descripteurs relationnels mis en jeu. Ceci ne permet donc pas de généraliser systématiquement les différents principes de simplification ; les seuls principes toujours applicables étant ceux issus de la caractéristique $R = (\rho \ ((\zeta)\mathcal{B}_1) \ \mathcal{B}_2) |\!\!< \mathcal{B}_1$ et de la dualité de la donnée :

1. Si G est tel que $G |\!\!> \mathcal{B}_1$ alors $(\rho \ ((\zeta)\mathcal{B}_1) \ \mathcal{B}_2) \vee G \models G$
2. $\forall \rho, (\rho \ ((\zeta)\mathcal{B}_1) \ \mathcal{B}_2) \ \vee (\sim \rho \ ((\zeta)\mathcal{B}_1) \ \mathcal{B}_2) \models \mathcal{B}_1$

Exemple de simplifications spécifiques :
Règle liée au descripteur intersection :
$(\mathsf{inter\ d\ b_1}) \vee (\mathsf{inter\ d\ b_2}) \models (\mathsf{inter\ d}\ \{\mathsf{b_1} \vee \mathsf{b_2}\})$

2.4.5 Simplification d'une disjonction de conjonctions de concepts

Le principe général de simplification et de mise en forme canonique tend à réduire le nombre de termes disjonctifs tout en gardant cette même forme disjonctive.

Dans le cas d'une disjonction de concepts dont l'un au moins est une conjonction, le seul moyen de pouvoir la simplifier est de rechercher un terme commun aux conjonctions et de «factoriser» l'expression pour regrouper des termes n'appartenant pas à un même concept conjonctif :

$$((\mathsf{et\ X\ A})\ (\mathsf{et\ X\ B})) \models (\mathsf{et\ X\ C}) \text{ avec } (\mathsf{A\ B}) \models \mathsf{C}$$

La simplification ne pourra être prise en compte que si le concept non disjonctif C existe, conformément à la régle de mise en forme canonique des expressions.

2.4.6 Simplification d'une disjonction de figures

Comme les règles de simplification liées aux descripteurs de relation, les simplifications de figures doivent tenir compte de caractéristiques trop spécifiques pour être généralisées ; les simplifications classiques pouvant être utilisées et liées à la complétude ou à l'inclusion de concepts n'étant que des cas particuliers de simplifications plus générales. Elles seront donc elles aussi gérées par des règles liées aux descripteurs mis en jeu.

2.5 Simplification des concepts conjonctifs

2.5.1 Simplification d'une conjonction de disjonctions de concepts

La spécification dans une description des accumulations de concepts peut être assurée par le rapprochement lors de la construction de deux descriptions. Ainsi les entités décrites par un tel concept C correspondent aux sous-entités décrites à la fois par la base B_1 et par la base B_2. Cette conjonction de concepts que l'on notera $C = B_1 \wedge B_2$ peut être exprimée dans LEO par la jonction (`et` B_1 B_2) et doit aussi faire l'objet de simplifications. Le travail de simplification d'une conjonction de bases se fait alors en trois étapes :

1. Factorisation des disjonctions :
   Exemple : soient $B_1 = (C_1 \vee C_2)$ et $B_2 = (C_3 \vee C_4)$
   $B_1 \wedge B_2 = (C_1 \vee C_2) \wedge (C_3 \vee C4) = (C_1 \wedge C_3) \vee (C_1 \wedge C4) \vee (C_2 \wedge C_3) \vee (C_2 \wedge C4)$
2. Simplification des conjonctions de concepts $(C_i \wedge C_j)$
3. Simplification de la disjonction de concepts résultante.

Comme nous pouvons exprimer une conjonction en deux concepts dans le langage LEO, la factorisation des disjonctions (première étape) ne pose aucun problème. La simplification de la disjonction de concepts (troisième étape) ayant été abordée, il ne reste plus qu'à déterminer les conditions de simplifications de la conjonction de deux concepts (deuxième étape) dont le principe général est, comme dans le traitement de la disjonction, de se réduire à un unique concept.

2.5.2 Simplification d'une conjonction de vecteurs de propriétés

Comme précédemment, nous allons distinguer deux cas en tenant compte de la nature des bases des concepts de la conjonction.

2.5.2.1 Cas 1 : les concepts de la conjonction ont des bases identiques

À nouveau nous examinons les ensembles de propriétés et passons par les deux phases déjà présentées : retrait des propriétés identiques et équilibrage des listes de propriétés des concepts.

Conjonction de propriétés
Nous définissons l'opération commutative de conjonction sur les propriétés de la façon suivante :

$$\begin{aligned} \sqcap :\ & \mathcal{P} \times \mathcal{P} \longrightarrow \mathcal{P} \\ & P_1, P_2 \longmapsto P \end{aligned}$$

avec $P_1 = (p\ Dom(P_1)), P_2 = (p\ Dom(P_2))$
et $P = (p\ \{Dom(P_1) \cap Dom(P_2)\})$

Contrairement à $\sqcup$, cette opération peut être facilement étendue aux listes équilibrées de propriétés en respectant la règle de réécriture ci-dessous :
$\forall i \in \{1, \ldots, n\}\ /\ Desc(P_i) = Desc(P'_i)$,

$$\{(P_1, P_2, \ldots, P_n)\} \sqcap \{(P'_1, P'_2, \ldots, P'_n)\} \models \{(P_1 \sqcap P'_1), (P_2 \sqcap P'_2), \ldots, (P_n \sqcap P'_m)\}$$

Principe de simplification
Les bases étant identiques, les entités décrites par le concept issu de la conjonction $C_1 \wedge C_2$ ont, si elles existent, un ensemble de caractéristiques formé par deux sous-ensembles de propriétés de C_1 et C_2. Le premier d'entre eux est la liste des propriétés communes L^{id} déterminées comme précédemment. Le deuxième est constitué du résultat de l'application de l'opération $\sqcap$ étendue aux propriétés des listes équilibrées. L'existence d'une telle description tient dans le fait que ce dernier

sous-ensemble ne peut contenir de propriétés à domaine vide. Dans un tel cas, la conjonction des concepts est nulle.

Loi de simplification
Lorsque deux concepts C_1 et C_2 ont des listes de propriétés équilibrées $L_{C_1}^{eq}$ et $L_{C_2}^{eq}$, on peut simplifier la conjonction de ces concepts en un nouveau concept C en respectant les règles suivantes :

Règle 1: soient $P_1 \in L_{C_1}^{eq}$ et $P_2 \in L_{C_2}^{eq}$ tels que $Desc(P_1) = Desc(P_2)$
Si $Dom(P_1) \cap Dom(P_2) = \emptyset$ alors $C_1 \wedge C_2 = C_2 \wedge C_1 = C = \lambda$, C n'existe pas.

Règle 2: si la règle précédente ne peut s'appliquer alors
$C_1 \wedge C_2 = C_2 \wedge C_1 = C = (\ \&_{i=1}^{n}(Desc(P_{1,2i})\{Dom(P_{1i}) \cap Dom(P_{2i}\})) \ \&\ L^{id}\ \&\ \mathcal{B}\ )$

Application :
Soient $C_1 =$`((p 10 20) (s 25 50) &` $\mathcal{B}$`)` et $C_2 =$`((q 100 200) (s 25 50) &` $\mathcal{B}$`)`, la simplification de la conjonction $C = C_1 \wedge C_2$ se fera de la façon suivante :

$$
\begin{array}{rcl}
C & = & C_1 \wedge C_2 \\
 & = & \texttt{(et (((p 10 20) (s 25 50) \& } \mathcal{B}\texttt{)) (((q 100 200) (s 25 50) \& } \mathcal{B}\texttt{)))} \\
 & = & \texttt{((p 10 20) (s 25 50) \& B)} \wedge \texttt{((q 100 200) (s 25 50) \& } \mathcal{B}\texttt{)} \\
 & = & \texttt{(\{(p 10 20) q}^{\Delta}\texttt{\} \{(s 25 50)\} } \mathcal{B}\texttt{)} \wedge \texttt{(\{p}^{\Delta}\texttt{ (q 100 200)\} (s 25 50) \& } \mathcal{B}\texttt{)} \\
 & = & \texttt{(\{(p 10 20) (q } \Delta_q\texttt{)\}} \sqcap \texttt{\{(p } \Delta_p\texttt{) (q 100 200)\} \{(s 25 50)\} \& } \mathcal{B}\texttt{)} \\
 & = & \texttt{(\{(p 10 20)} \sqcap \texttt{(p } \Delta_p\texttt{)) ((q } \Delta_q\texttt{)} \sqcap \texttt{(q 100 200))\} \{(s 25 50)\} \& } \mathcal{B}\texttt{)} \\
 & = & \texttt{(\{(p [10,20[} \cap \Delta_p\texttt{) (q } \Delta_q \cap \texttt{[100 200[)\} \{(s 25 50)\} \& } \mathcal{B}\texttt{)} \\
 & = & \texttt{(\{(p 10 20) (q 100 200)\} \{(s 25 50)\} \& } \mathcal{B}\texttt{)} \\
 & = & \texttt{((p 10 20) (q 100 200) (s 25 50) \& } \mathcal{B}\texttt{)}
\end{array}
$$

2.5.2.2 Cas 2 : les concepts de la conjonction ont des bases distinctes

Lorsque les bases des concepts de la conjonction sont distinctes, le concept résultat, s'il existe, est alors formé de la conjonction des bases de C_1 et C_2 spécialisée par l'ensemble des propriétés identiques et le résultat de l'application de la fonction $\sqcap$ sur les listes équilibrées de propriétés de C_1 et C_2. Comme dans le cas précédent, si une des intersections est vide, le concept résultat de la conjonction n'existe pas.

Principe de simplification
Ainsi, soient les concepts équilibrés $C_1 = (L_{C_1}^{eq} \& L^{id} \& \mathcal{B}_1)$ et $C_2 = (L_{C_2}^{eq} \& L^{id} \& \mathcal{B}_2)$, nous pouvons distinguer trois éventualités pour exprimer la conjonction $C = C_1 \wedge C_2$, selon le résultat de la conjonctions de leurs bases :

1. si $\mathcal{B}_1 \wedge \mathcal{B}_2 = \lambda$ alors $C = C_1 \wedge C_2 = \lambda$, C n'existe pas
2. si $L_{C_1}^{eq} \sqcap L_{C_2}^{eq} = \lambda$ alors $C = C_1 \wedge C_2 = \lambda$, C n'existe pas
3. sinon $C = C_1 \wedge C_2 = (\{L_{C_1}^{eq} \sqcap L_{C_2}^{eq}\} \& L^{id} \& (\mathcal{B}_1 \wedge \mathcal{B}_2))$

Loi de simplification
Lorsque deux concepts C_1 et C_2 ont des listes de propriétés équilibrées $L_{C_1}^{eq}$ et $L_{C_2}^{eq}$ et des bases $\mathcal{B}_1$ et $\mathcal{B}_2$, on peut simplifier la conjonction de ces concepts en un nouveau concept C en respectant les règles suivantes :

Règle 1:
si $\mathcal{B}_1 \wedge \mathcal{B}_2 = \lambda$ alors $C_1 \wedge C_2 = C_2 \wedge C_1 = C = \lambda$, C n'existe pas

Règle 2: soit $P_1 \in L^{eq}_{C_1}$ et $P_2 \subset L^{eq}_{C_2}$ tels que $Desc(P_1) = Desc(P_2)$
si $Dom(P_1) \cap Dom(P_2) = \emptyset$ alors $C_1 \wedge C_2 = C_2 \wedge C_1 = C = \lambda$, C n'existe pas

Règle 3:
si les règles précédentes ne peuvent s'appliquer alors
$C_1 \wedge C_2 = C_2 \wedge C_1 = C = (\ \&^n_{i=1}(Desc(P_{1,2i})\{Dom(P_{1i}) \cap Dom(P_{2i}\})) \ \&\ L^{id}\ \&\ (\mathcal{B}_1 \wedge \mathcal{B}_2))$

Application :
Soient C_1 =`((p 10 20) & ((S 2 5)))` et C_2 =`((q 1 2) & ((T 25 50)))`, `(S 2 5)` et `(T 25 50)` étant les bases[8] de C_1 et C_2, la simplification de la conjonction $C = C_1 \wedge C_2$ se fera de la façon suivante :

$$
\begin{aligned}
C &= C_1 \wedge C_2 \\
&= \texttt{(et (((p 10 20) \& ((S 2 5)))) (((q 1 2) \& ((T 25 50)))))} \\
&= \texttt{((p 10 20) \& ((S 2 5)))} \wedge \texttt{((q 1 2) \& ((T 25 50)))} \\
&= \texttt{(\{(p 10 20) q}^\Delta\texttt{\} \& (\{(S 2 5) T}^\Delta\texttt{\}))} \wedge \texttt{(\{p}^\Delta\texttt{ (q 1 2)\} \& (\{S}^\Delta\texttt{ (T 25 50)\})))} \\
&= \texttt{(\{(p 10 20) q}^\Delta\texttt{\}} \sqcap \texttt{\{p}^\Delta\texttt{ (q 1 2)\} \& (\{((S 2 5) T}^\Delta\texttt{)\}} \sqcap \texttt{\{(S}^\Delta\texttt{ (T 25 50))\}))} \\
&= \texttt{(\{(p 10 20)} \sqcap \texttt{p}^\Delta\texttt{\} \{q}^\Delta \sqcap \texttt{(q 1 2)\} \& (\{((S 2 5)} \sqcap \texttt{S}^\Delta\texttt{)\} \{(T}^\Delta \sqcap \texttt{(T 25 50))\}))} \\
&= \texttt{((p 10 20) (q 1 2) \& ((S 2 5) \& (T 25 50)))}
\end{aligned}
$$

2.5.3 Simplification d'une conjonction de relations

Compte tenu des remarques faites précédemment sur les relations, la simplification des conjonctions où l'un des concepts est une relation -$G \wedge (\rho\ ((\zeta)\mathcal{B})\ Bse)$- se réduit pour les simplifications génériques à l'examen de l'inclusion de la base $\mathcal{B}$ dans le concept G.
Règle: Si G est tel que $G \rhd \mathcal{B}$ alors $G\ \wedge\ (\rho\ ((\zeta)\mathcal{B})\ Bse) \models (\rho\ ((\zeta)\mathcal{B})\ Bse)$ Comme précédemment, d'autres règles de simplifications spécifiques liées aux types et donc aux descripteurs de relations peuvent être mises en œuvre.

2.5.4 Simplification d'une conjonction de figures

Des règles de simplification spécifiques sont aussi à mettre en œuvre afin de gérer les conjonctions de figures. Elles sont développées au regard des autres règles afin d'assurer une bonne réduction des expressions : simplification et mise sous forme canonique.

2.6 Autres formes de simplification

D'autres formes de simplification existent et sont implantées sous forme de règles spécifiques déclenchées sur les expressions fournies au système ou générées par celui-ci. Ces simplifications font référence à une forme particulière de connaissance du traitement d'images directement liée aux descripteurs utilisés. Elles permettent de traiter des propriétés connues de simplification et manipulent de ce fait plusieurs entités de LEO que sont les descripteurs ou concepts de plus haut niveau de construction d'expression : vecteurs de propriétés, concepts relationnels, bases, etc. Il s'agit donc de règles de réécriture d'expression reposant sur un double traitement : une première phase consistant en la une reconnaissance d'une configuration particulière des éléments de la description analysée, une seconde phase consistant en la substitution cette configuration par une autre garantissant la même description initiale mais lui conférant une forme plus simple au regard des principes énoncés.

Nous pouvons ainsi prendre un exemple lié aux opérateurs morphologique d'érosion $Ero(x)$ et de dilatation $Dil(x)$ sur une composante connexe x. Les caractéristiques que nous pouvons traduire

8. nous faisons abstraction de l'élément déstructurant

par $Ero(Dil(Ero(x))) = Ero(x)$ et $Dil(Ero(Dil(x))) = Dil(x)$ -qui découlent de [Zhu86], proposition 5[9]- peuvent être implantées par des recherches de configurations des propriétés à sélecteur `distance` présentes, dans la description du cache de la donnée, sous la forme de vecteurs de propriétés à un seul élément. Ce vecteur spécialise une base déstructurée spécifique correspondant à l'entité manipulée que nous noterons x =((ζ)$\mathcal{B}$) avec $\sim x$ =(($\sim \zeta$)$\sim \mathcal{B}$) :

Reconnaissance de $Ero(x)$: `(((dist 2 128)` x`))`

Reconnaissance de $Dil(x)$: `(((dist 1 2)` $\sim x$`)` $\mathcal{B}$`)`

Règles de substitutions :

Si $C = Ero(Dil(Ero(x)))$ alors $C \models Ero(x)$

Si $C = Dil(Ero(Dil(x)))$ alors $C \models Dil(x)$

Une conséquence de ces règles est de pouvoir traduire par extrapolation des caractéristiques particulières d'autres opérateurs comme, dans l'exemple présent, l'idempotence des opérateurs d'ouverture et de fermeture :

Ouverture $Ouv(x)$:

$$\begin{array}{lcl} \text{Si } C & = & Ouv(Ouv(x)) \text{ alors} \\ C & \models & Dil(Ero(Dil(Ero(x)))) \\ & \models & Ero(Dil(Ero(x))) \\ & \models & Dil(Ero(x)) \\ & \models & Ouv(x) \end{array}$$

Fermeture $Fer(x)$:

$$\begin{array}{lcl} \text{Si } C & = & Fer(Fer(x)) \text{ alors} \\ C & \models & Ero(Dil(Ero(Dil(x)))) \\ & \models & Dil(Ero(Dil(x))) \\ & \models & Ero(Dil(x)) \\ & \models & Fer(x) \end{array}$$

9. *Proposition 5 : if* $S = A \oplus J$ *then* $S = S \circ J = S$

Chapitre 3

Appariement d'expressions

Le problème de l'appariement d'expression consiste à juger de l'interêt de l'utilisation d'un opérateur pour la résolution d'un objectif. Ce module peut donc être considéré comme une des composantes les plus importantes du système. Nous pouvons reprendre le schéma établis par J.Lemaire ([Lem95]) sur la mise en correspondances de la DSA -description de la scène attendue- et la DSP -description de la scène perçue- pour mettre exergue les erreurs d'interprétation susceptibles d'intervenir au cours de l'appariement. En complétant ce schéma, qui ne prenait pas en compte la possibilité de valider l'appariement d'une erreur de description de la DSA et d'une erreur de perception (DSP), nous pouvons visualiser les spécifications que doit respecter notre module d'appariement.

Dans la figure suivante (fig.3.1), nous avons mis en italique les termes utilisés par Lemaire. Ils correspondent à des notions de notre propre domaine de recherche : les *éléments perçus* correspondent aux éléments produits par nos opérateurs, les *éléments décrits* correspondant à l'objectif à atteindre. Cette figure montre comment interpréter l'analyse des réponses de l'appariement en mettant en jeu la scène, la description par l'expert de l'objectif à atteindre et les descriptions des données produites par les opérateurs. Ainsi nous voyons qu'un appariement que nous pouvons juger incorrect peut être dû à des erreurs de description d'objectif et de description d'opérateurs (a). Parallèlement, l'une de ces deux erreurs de description entraînera un non appariement que nous jugeons incorrect (b,c).

Un tel schéma de présentation de l'appariement dans le contexte de l'application peut être exploité de diverses façons. Il peut aider l'utilisateur d'un système intégrant un tel module en lui fournissant des moyens de comprendre les erreurs produites, soit dans la conception de l'objectif, soit dans l'utilisation des opérateurs -manuelle par sélection, ou automatique par planification. Il peut aussi aider à l'intégration d'un macro-opérateur en permettant de mettre au point la description

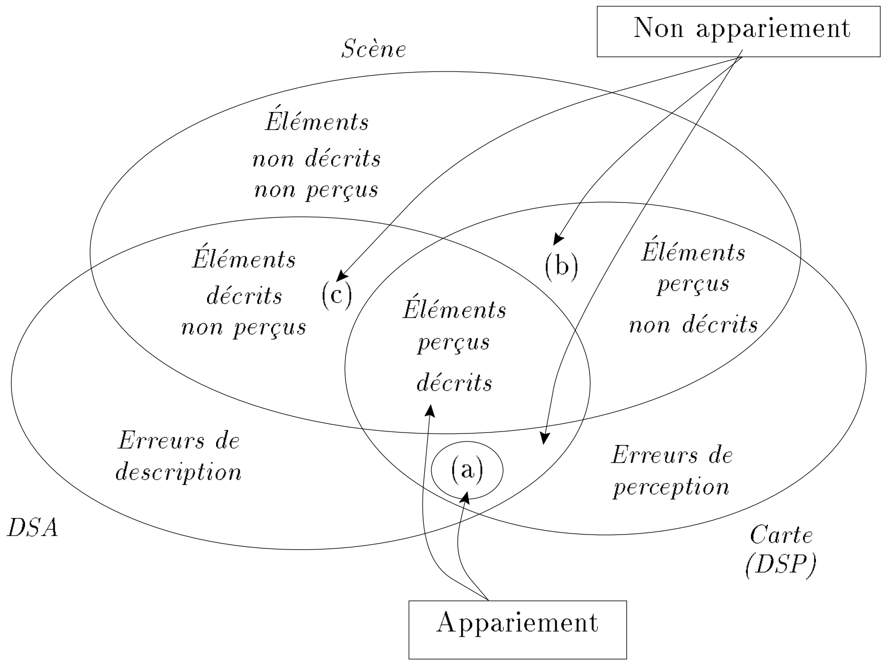


FIG. 3.1 - *Scène, objectif, donnée produite et appariement*

de la chaîne à planifier pour pouvoir le réutiliser, l'intégrer et contrôler son exécution.

3.1 Notion de distance à l'objectif

Nous avons généralement parlé du module d'appariement comme étant un des éléments essentiels, avec le filtrage d'opérateurs, du processus de planification. Or, tout comme la notion de simplification d'expressions, l'appariement peut être considéré comme un module indépendant, exploitant le formalisme que nous avons développé. En effet, son rôle est de donner une réponse quant à la "ressemblance" d'une expression par rapport à une autre.

Dans le cas de la planification, cette ressemblance est exploitée pour déterminer la pertinence de la donnée décrite -et donc de son opérateur `generateur`- par rapport à l'objectif à atteindre. Nous allons toutefois décrire cet appariement au regard de la planification d'objectif, étant donné son implantation concrète dans le système ExTI. La hiérarchisation des concepts manipulés, passant par un regroupement des descripteurs n'étant pas encore définie -des travaux sont en cours concernant l'étude des mesures et de leurs descriptions-, nous étudierons donc l'appariement sous l'angle syntaxique de l'expression et non pas sous son aspect sémantique.

L'avancement du plan tend à diminuer le degré de généralité de la donnée initiale pour se rapprocher de l'objectif à atteindre, même s'il est parfois impératif de "s'éloigner" de cet objectif pour y parvenir par un chemin plus prometteur selon le critère de l'heuristique employée [Win77].

C'est ce rapprochement de l'état courant par rapport au but à atteindre qui est évalué par le module d'appariement. Actuellement, cette réponse est booléenne, validant ou invalidant la donnée produite en vue de son intégration dans le plan. Or, ce processus fait plus que constater la différence entre deux expressions car si tel était le cas, la description d'une donnée intermédiaire aurait la même réponse à l'appariement que celle d'une donnée inintéressante pour le plan.

Il nous semble ainsi possible de définir une valeur témoigant de cette différence ou une expression la décrivant. C'est pour cette raison que nous pouvons parler de "distance" à l'objectif, même si ce terme de distance paraît impropre dans son sens mathématique strict ; cependant, il fait toutefois référence aux deux notions de différence et de mesure que nous venons d'évoquer.

Comme dans le cadre de l'étude des simplifications d'expressions, l'appariement va manipuler des descriptions définies à l'aide du LDC de LEO. Pour des raisons liées au temps de développement et à l'implantation de la grammaire, cet appariement est effectué en fonction des syntaxes des expressions fournies et est donc construit autour de la catégorie syntaxique *Bse* du langage ; cette catégorie qui définit la *base* manipulant toutes les autres par imbrications successives. Le champ signal σ de la donnée, dans le cas où il décrit des mesures sur une base, sera alors traité comme une base ; le seul traitement particulier étant fait sur son descripteur de mesure.

Les principes que nous présentons ici permettent de définir le module d'appariement en fonction de la syntaxe tel qu'il est implanté dans le système ExTI. Or, maintenant que le langage d'expression est établi et formalisé, il nous paraît plus performant pour le planifieur d'étudier les moyens d'effectuer cet appariement à partir des arbres de décomposition syntaxique de ces mêmes expressions. La caractérisation de la distance à l'objectif pourra alors se faire plus simplement à l'aide d'outils de manipulation d'arbres syntaxiques. Si une telle méthode n'est actuellement qu'une idée concernant l'exploitation du langage d'expression, les principes que nous allons décrire sont néanmoins suffisamment généraux pour caractériser les deux méthodes. Ces principes sont à l'origine d'une réécriture complète de ce module d'appariement en fonction de la syntaxe.

3.2 Principes d'appariement

3.2.1 Appariement général

Le processus d'appariement *-appariement général-* entre une expression, issue d'un élément de description d'une donnée produite E, et une expression correspondante de la description de l'objectifs O, va s'effectuer en deux phases que nous appellerons *appariement global*, et *appariement local*. Cette seconde phase n'est en fait développée que lorsque la première a échouée. Nous noterons cet appariement général $O \approx E$. Si nous notons alors $\asymp$ l'appariement global et $\succ$ l'appariement local, nous pouvons établir la règle d'appariement général de la façon suivante :

$$O \approx E = O \asymp E \ \ || \ \bigvee_{i=1}^{|O|} (O_i \succ E)$$

3.2.2 Appariement global

Comme nous l'avons indiqué en introduction, l'appariement global, comme l'appariement général, manipule deux expressions E et O produites par la catégorie syntaxique *Bse*. Le principe de cet appariement est simple et basé sur la forme normale disjonctive des expressions produites par *Bse*. Trois cas, établis sur le nombre de concepts des expressions O et E, peuvent être considérés :

1. Si le nombre de concepts -catégorie *Cpt*- de O est supérieur à celui de E, un appariement spécial est effectué sur les éléments de O et de E. Nous cherchons ici à mettre en correspondance tous les concepts E_j avec des O_i. Ces mises en correspondances, que nous noterons $O_i \equiv E_j$, relatives à des entités de descriptions aboutiront à ce que nous appelons des *décisions locales*. En parcourant les éléments de E en parallèle avec ceux de O -les concepts

sont rangés par ordre alphanumériques selon la forme canonique-, nous rencontrons ainsi les quatre cas suivants :

(a) $O_i \equiv E_j$ est un succès. Il faut alors tester la correspondance $O_{i+1} \equiv E_{j+1}$
(b) $O_i \equiv E_j$ est un échec. Il nous faut tester la correspondance $O_{i+1} \equiv E_j$
(c) E a été parcouru en totalité. Dans ce cas, il s'agit d'un succès de l'appariement spécial et donc global.
(d) O a été parcouru en totalité (mais pas E). Il s'agit cette fois d'un échec des appariements spécial et global.

2. Si les nombres de concepts de O et de E sont identiques, les décisions locales sont effectuées sur tous les éléments de O et de E.

   Si nous restons dans une optique de planification, une erreur d'appariement intervenant sur une de ces décisions locales $O_i \equiv E_i$ provoque l'échec de l'appariement global. Si, par contre, l'objectif de l'appariement est de déterminer la différence entre les deux expressions, alors des mises en correspondances peuvent être développées comme précédemment.
3. Si, enfin, le nombre d'éléments de O est inférieur à celui de E, l'appariement, toujours dans le cas d'une réponse pour le planifieur, est voué à l'échec. Le processus de parcours des expressions peut alors être effectué pour donner des informations complémentaires.

Ainsi, nous pouvons définir l'appariement global par :

$$O \asymp E = \bigwedge_{j=1}^{|E|} (O_i \equiv E_j)\,, \forall i \in [1, \mid O \mid]$$

3.2.3 Appariement local

Après échec de l'appariement global, nous traitons, par ces appariements locaux, le fait que l'expression E peut être un composant des concepts C présents dans O. Toujours en dirigeant notre recherche par la syntaxe des expressions, nous pouvons rencontrer trois cas selon la nature de ces éléments C :

1. Si C est une relation ou une jonction, nous effectuons alors un appariement général sur chaque base B_i avec l'expression E. Selon le type de réponse souhaitée, la recherche peut être arrêtée dès le premier appariement correct -ce qui est le cas dans le module implanté dans ExTI. L'expression d'un tel appariement peut s'écrire alors :

$$C \succ E = \bigvee_{i=2}^{|C|} (C_i \approx E)$$

2. Si C est un vecteur de propriétés ou un vecteur de propriétés appliqué à une base déstructurée, il faut effectuer un appariement général de chaque élément x avec l'expression E de la façon suivante : si x est une propriété, on détermine le filtrage général de chaque élément du domaine de la propriété, sinon x est une base déstructurée, on effectue alors l'appariement général de la base de cet élément syntaxique par l'expression E. Ainsi, C étant un vecteur de propriétés, $C = (P_1 \ldots P_i \ldots P_n)$, avec $P_i = (Desc(P_i)\ d_{i_1} \ldots\ d_{i_j} \ldots d_{i_m})$ nous pouvons établir la formule de l'appariement comme suit :

$$C \succ E = \bigvee_{i=1}^{|C|} \left(\bigvee_{j=1}^{|Dom(P_i)|} \left(d_{i_j} \approx E \right) \right)$$

dans le cas où ce vecteur est appliqué à une base déstructurée B, nous avons :

$$C \succ E = \bigvee_{i=1}^{|C|} \left(\bigvee_{j=1}^{|Dom(P_i)|} \left(d_{i_j} \approx E \right) \right) \vee (B \approx E)$$

3. Sinon, C est une base issue d'une mesure du champ signal de la donnée. On effectue alors l'appariement général de cette description avec l'expression E initiale.

3.2.4 Décisions locales

Les décisions locales consistent à fournir une réponse quant à l'appariement de deux expressions selon leur structure syntaxique : liste Lisp, valeur entière, etc. Cet ensemble de fonctions de décisions forme ainsi l'élément bas-niveau du module d'appariement. Si les décisions locales autres que celles traitant des atomes Lisp sont simples, celles appliquées à deux listes sont d'une complexité non négligeable. Ici encore, plusieurs cas sont observés selon la nature des expressions : accumulation de propriétés, propriétés, relations, etc. C'est cette complexité qui nous a amené à proposer une amélioration de ce module fondamental par l'utilisation de manipulations des grammaires et des arbres syntaxiques.

Le principe de détermination de ces décisions est équivalent à celui gérant l'appariement global, exception faite de la possibilité de passer outre un échec d'appariement dû à un nombre insuffisant d'éléments mis en correspondance.

3.3 Espace de résolution

Cette caractéristique de l'appariement global que nous venons de voir est actuellement utilisée pour développer un processus de prise en compte d'expressions plus spécialisées que l'objectif. Cet élément est important car il va permettre de planifier des chaînes plus complexes du point de vue de la compréhension des transformations effectuées.

Nous pouvons en effet représenter (fig.3.2) l'espace de résolution d'objectif par une structure triangulaire convergeant, via un axe de spécialisation, vers cet objectif. Cet espace de résolution correspond aux données accessibles et pertinentes pouvant être produites au cours de l'élaboration d'un graphe de résolution. Cet espace correspond à une partie de l'espace des données plus généralisées que l'objectif. Ainsi, la construction du graphe de résolution ne peut se faire que selon cet axe de spécialisation des données ; les seuls éléments plus spécialisés que l'objectif et accessibles étant les concepts mêmes de cet objectif.

Par l'introduction de cette caractéristique dans le module d'appariement d'expressions qui autorise un certain degré de liberté dans l'espace valide de résolution, l'axe de spécialisation selon lequel le planifieur évolue peut être rompu. Ainsi, comme nous l'avons schématisé sur la figure 3.3, l'espace de résolution peut être étendu aux concepts spécialisés de l'objectif, ce qui augmente les possibilités de détermination des solutions.

Cette notion est exploitée dans le système grace à la possibilité de simplifier les expressions des descriptions de données produites par les opérateurs. Ainsi, une fois que le système a basculé dans l'espace des concepts spécialisés, les simplifications d'expressions permettent de rester dans l'axe de généralisation des concepts en évitant de rentrer dans l'espace des données non pertinentes.

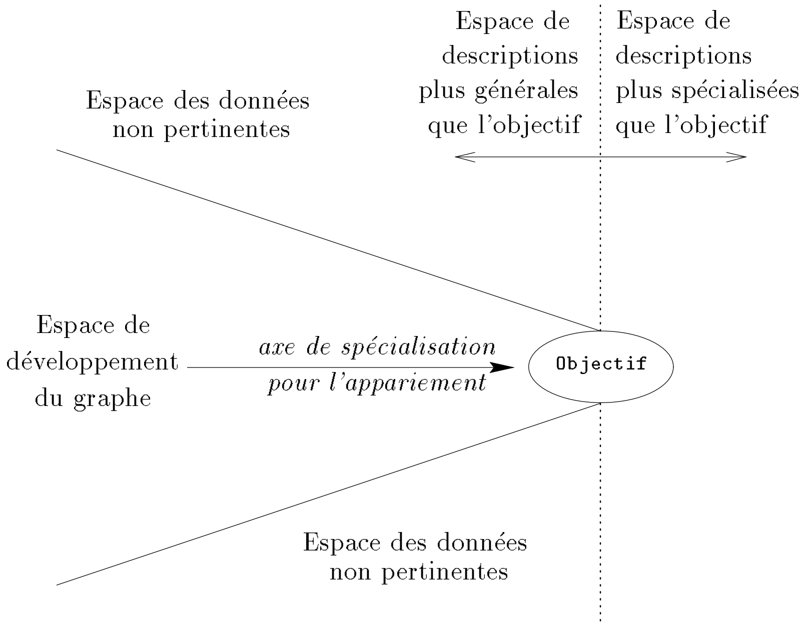


Fig. 3.2 - *Résolution selon l'axe de spécialisation*

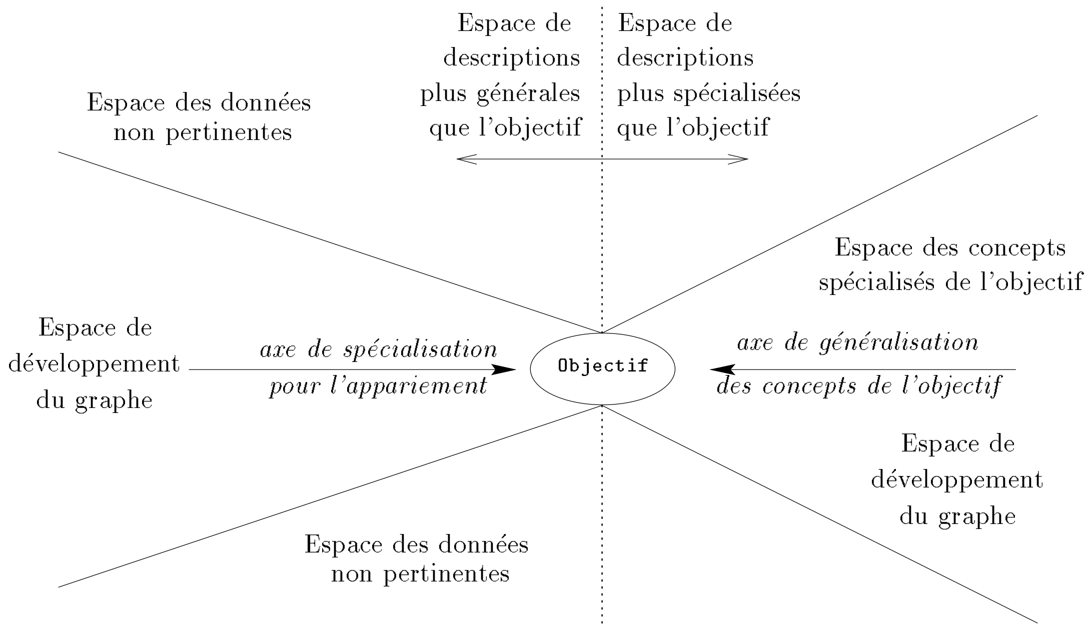


Fig. 3.3 - *Double axe de résolution*

Chapitre 4

Résultats

L'objectif de ce chapitre est de montrer, sur un exemple, les capacités du système ExTI et plus particulièrement celles du module de planification par réutilisation de plans. Nous allons ainsi montrer les éléments accessibles à la suite d'une planification : graphe d'opérateurs, descriptions de toutes les données produites et programme exécutable.

Mis à part le fait qu'il intègre la notion importante de concept d'un domaine d'application, le module de planification par réutilisation de plan permet d'améliorer considérablement les temps de planification. Nous verrons alors, par des comparaisons de ces temps, cet autre intérêt d'utiliser ce module.

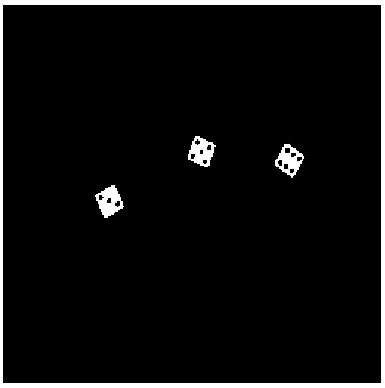
4.1 Planification de l'objectif `rel5`

Nous allons étudier l'objectif rel5 ([Ing92] Annexe 1) qui consiste à extraire les dés d'une image d'objets.

La présentation des résultats va consister en un rappel de l'expression selon l'ancienne syntaxe ainsi que dans le langage LEO. Nous rapporterons ensuite les traces des planifications selon les deux modes automatiques -simple et par réutilisation de plan- ainsi que les graphes de résolution, les actions symboliques mises en œuvre et le programme exécutable appelé aussi script.

Le graphe de résolution est représenté par un arbre du système Aïda [AID92] sur lequel figurent les noms des données symboliques auxquelles sont associés l'opérateur et les paramètres qui ont généré cette donnée ainsi que le nom du concept décrit.

Image d'objets



Dés

Les actions symboliques constituent une autre représentation du graphe de résolution, en mettant en valeur les descriptions `d` et `do` -complémentaire de `d`- des données créées.

Le script correspond à un fichier `shell UNIX` écrit à partir du plan et contenant les appels aux opérateurs réels de traitement d'images de notre base.

4.1.1 Expressions de l'objectif et des sous-objectifs utilisés

4.1.1.1 Ancien formalisme de description

```
  objets   '(((nivg >= 100)))
n_objets   '(((nivg < 100)))
    cc10   '(((mom_xx < 1) (surface >= 6) (surface < 11) (n_objets)))
    rel1   '(objets (dans (n_objets) (objets))
    rel4   '((intersecte (rel1) (cc10)))
    rel5   '((intersecte (objets) (rel4)))
```

4.1.1.2 Description d'après **LEO**

```
objets
   '((0 256) (0 256))
   '(((nivg 100 256)))
   '(gl)
   'vab
   '(1 2)

cc10 ;(petits -mxx,srf- trous des objets)
   '((0 256) (0 256))
   '(((mxx 0 1)(srf 6 11)((vois4)(~objets*k))))
   '(gl)
   'vab
   '(1 2)

rel1 ;(objets pleins)
   '((0 256) (0 256))
   '(objets*k (dans ((vois4)(~objets*k))(objets*k)))
   '(gl)
   'vab
   '(1 2)
```

```
rel4 ;(objets pleins à petits trous)
   '((0 256) (0 256))
   '((inter ((vois4)(rel1*k))(cc10*k)))
   '(gl)
   'vab
   '(1 2)

rel5 ;(objets à petits trous)
   '((0 256) (0 256))
   '((inter ((vois4)(objets*k))(rel4*k)))
   '(gl)
   'vab
   '(1 2)
```

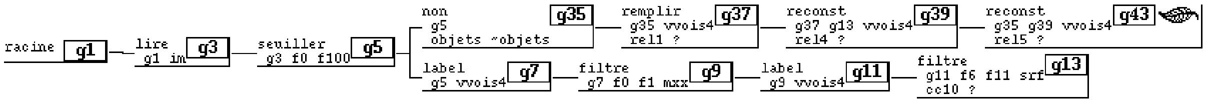
4.1.2 Planification en mode simple

4.1.2.1 Trace de la planification

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : ((inter ((vois4) (((nivg 100 256))))
((inter ((vois4) (((nivg 100 256)) (dans ((vois4)
(((nivg 0 100)))) (((nivg 100 256)))))) (((mxx 0 1)
(srf 6 11) ((vois4) (((nivg 0 100)))))))))
| Index : (gl)
| Signal : vab
| Codage : (1 2)

g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f100)
g7 = (fs_label g5 vvois4)
g9 = (fs_filtre g7 f0 f1 mxx)
g11 = (fs_label g9 vvois4)
g13 = (fs_filtre g11 f6 f11 srf)
       [cc10]
g15 = (fs_reconst g5 g13 vvois4)
g17 = (fs_reconst g5 g15 vvois4)
        (fs_reconst g5 g15 vvois4) -->g15
g19 = (fs_label g15 vvois4)
g21 = (fs_label g13 vvois4)
g23 = (fs_filtre g7 f6 f11 srf)
g25 = (fs_reconst g13 g23 vvois4)
       [cc10]
        (fs_reconst g13 g23 vvois4) -->g13
```

```
g27 = (fs_label g23 vvois4)
g29 = (fs_filtre g27 f0 f1 mxx)
       [cc10]
        (fs_filtre g27 f0 f1 mxx) -->g13
g31 = (fs_et g23 g9)
       [cc10]
        (fs_et g23 g9) -->g13
g35 = (fs_non g5)
       [objets]
g37 = (fs_remplir g35 vvois4)
       [rel1]
g39 = (fs_reconst g37 g13 vvois4)
       [rel4]
g41 = (fs_reconst g37 g39 vvois4)
       [rel4]
        (fs_reconst g37 g39 vvois4) -->g39
g43 = (fs_reconst g35 g39 vvois4)
       [rel5]
Le but est atteint par egalite stricte
Temps d'execution de la planification : 2mn 41.35999s
Nombre d'operateurs testes : 121
Temps moyen de test par operateur : 1.333554s

Solutions : (g43)
```

4.1.2.2 Graphe de résolution



4.1.3 Planification par réutilisation de plans

4.1.3.1 Trace de la planification

Cette première trace de planification consiste à construire le plan de résolution de `rel5` à partir de ces sous-objectifs `objets` et `rel4` déjà résolus :

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : ((inter ((vois4) (((nivg 100 256))))
((inter ((vois4) (((nivg 100 256)) (dans ((vois4)
(((nivg 0 100)))) (((nivg 100 256)))))) (((mxx 0 1)
(srf 6 11) ((vois4) (((nivg 0 100))))))))))
| Index : (gl)
| Signal : vab
| Codage : (1 2)

Generation du plan :
objets sous /home/dejean/.../objets.plan
====================
#[((0 256) (0 256)) (((nivg 100 256))) (gl) vab (1
2)]
g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f100)
g7 = (fs_non g5)
        [objets]
Le but est atteint par egalite stricte

Generation du plan :
rel4 sous /home/dejean/.../rel4.plan
====================
#[((0 256) (0 256)) ((inter ((vois4) (((nivg 100
256)) (dans ((vois4) (((nivg 0 100)))) (((nivg 100
256)))))) (((mxx 0 1) (srf 6 11) ((vois4) (((nivg 0
100)))))))) (gl) vab (1 2)]
g3 = (fs_lire g1 im)
        Re-utilisation de l'objet : g3
g5 = (fs_seuiller g3 f0 f100)
        Re-utilisation de l'objet : g5
g11 = (fs_label g5 vvois4)
g13 = (fs_filtre g11 f0 f1 mxx)
g15 = (fs_label g13 vvois4)
g17 = (fs_filtre g15 f6 f11 srf)
        [cc10]
g5 = (fs_seuiller g3 f0 f100)
        Re-utilisation de l'objet : g5
g7 = (fs_non g5)
        [objets]
        Re-utilisation de l'objet : g7
g22 = (fs_remplir g7 vvois4)
        [rel1]
g24 = (fs_reconst g22 g17 vvois4)
        [rel4]
Le but est atteint par egalite stricte

Planification du but principal :
================================
Phase No1 : sur les plans generes
Objet : g7
g27 = (fs_reconst g7 g24 vvois4)
        [rel5]
Le but est atteint par egalite stricte
Temps d'execution de la planification : 33.75s
Nombre d'operateurs testes : 13
Temps moyen de test par operateur : 2.596154s

Solutions : (g27)
```

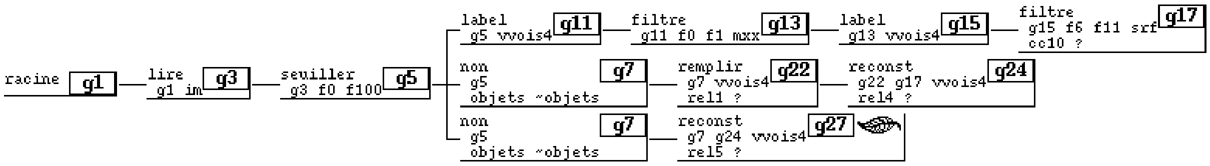
Cette seconde trace montre comment le plan de l'objectif `rel5` peut être rappelé et ses données symboliques re-générées :

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : ((inter ((vois4) (((nivg 100 256))))
((inter ((vois4) (((nivg 100 256)) (dans ((vois4)
(((nivg 0 100)))) (((nivg 100 256)))))) (((mxx 0 1)
(srf 6 11) ((vois4) (((nivg 0 100))))))))))
| Index : (gl)
| Signal : vab
| Codage : (1 2)

Generation du plan :
rel5 sous /home/dejean/.../rel5.plan
====================
#[((0 256) (0 256)) ((inter ((vois4) (((nivg 100
256)))) ((inter ((vois4) (((nivg 100 256)) (dans
((vois4) (((nivg 0 100)))) (((nivg 100 256))))))
(((mxx 0 1) (srf 6 11) ((vois4) (((nivg 0 100))))))
)))) (gl) vab (1 2)]
g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f100)
g7 = (fs_label g5 vvois4)
g9 = (fs_filtre g7 f0 f1 mxx)
g11 = (fs_label g9 vvois4)
g13 = (fs_filtre g11 f6 f11 srf)
        [cc10]
g17 = (fs_non g5)
        [objets]
g19 = (fs_remplir g17 vvois4)
        [rel1]
g21 = (fs_reconst g19 g13 vvois4)
        [rel4]
g23 = (fs_reconst g17 g21 vvois4)
        [rel5]
Le but est atteint par egalite stricte
Temps d'execution (CPU) de la planification :
30.53003s
Nombre d'operateurs testes : 10
Temps moyen de test par operateur : 3.053003s

Solutions : (g23)
```

La diminution du nombre d'opérateurs utilisés, de 13 dans la première trace à 10 dans la seconde, vient du fait que lors de la construction du plan -première planification-, trois opérateurs ont été réutilisés. Si nous comparons maintenant les temps d'exécution des deux modules de planification, nous pouvons évaluer à environ 80% le gain de temps par l'utilisation du module de réutilisation de plan. Si nous ajoutons à ce temps ceux nécessaires à la résolution des sous-objectifs étiquetés de `rel5`, de leurs sous-objectifs, etc, le gain global est de 57%. Enfin, si nous restons dans les comparaisons autour de cet objectif, le module initial de planification simple développé par J.-M.

Inglebert, développait 63 couples (donnée,noeud du graphe) alors que notre module simple n'en développe que 22, le module par réutilisation de plan n'en développant, lui, que 12. Cette différence s'explique par une gestion rigoureuse des simplifications d'expressions et un filtrage d'opérateur plus performant.

4.1.3.2 Graphe de résolution



4.1.4 Éléments disponibles après planification

4.1.4.1 Actions symboliques

```
OBJECTIF = #[((0 256) (0 256)) ((inter ((vois4) (((nivg 100 256)))) ((inter ((vois4) (((nivg 100 256)) (dans
((vois4) (((nivg 0 100)))) (((nivg 100 256)))))) (((mxx 0 1) (srf 6 11) ((vois4) (((nivg 0 100))))))))) (gl)
vab (1 2)]
 0: g1 = (fs_racine)
     |? {d}: ()
     |? {do}: ()
 1: g3 = (fs_lire g1 im)
     |? {d}: #[((0 256) (0 256)) t (gl) nivg (0 256)]
     |? {do}: #[((0 256) (0 256)) () () () ()]
 2: g5 = (fs_seuiller g3 f0 f100)
     |? {d}: #[((0 256) (0 256)) (((nivg 0 100))) (gl) vab (1 2)]
     |? {do}: #[((0 256) (0 256)) (((nivg 100 256))) () vab (0 1)]
 3: g11 = (fs_label g5 vvois4)
     |? {d}: #[((0 256) (0 256)) (((nivg 0 100))) (etiq vois4) etiq (1 16385.)]
     |? {do}: #[((0 256) (0 256)) (((nivg 100 256))) (gl vois8) vab (0 1)]
 4: g13 = (fs_filtre g11 f0 f1 mxx)
     |? {d}: #[((0 256) (0 256)) (((mxx 0 1) ((vois4) (((nivg 0 100)))))) (gl) vab (1 2)]
     |? {do}: #[((0 256) (0 256)) (((mxx 1 10837.68) ((vois4) (((nivg 0 100))))) ((nivg 100 256))) () vab (0 1)]
 5: g15 = (fs_label g13 vvois4)
     |? {d}: #[((0 256) (0 256)) (((mxx 0 1) ((vois4) (((nivg 0 100)))))) (etiq vois4) etiq (1 16385.)]
     |? {do}: #[((0 256) (0 256)) (((mxx 1 10837.68) ((vois4) (((nivg 0 100))))) ((nivg 100 256))) (gl vois8) vab
           (0 1)]
 6: g17 = (fs_filtre g15 f6 f11 srf)
     | cc10 {d}: #[((0 256) (0 256)) (((mxx 0 1) (srf 6 11) ((vois4) (((nivg 0 100)))))) (gl) vab (1 2)]
     |? {do}: #[((0 256) (0 256)) (((mxx 0 1) (srf 1 6 11 65536.) ((vois4) (((nivg 0 100))))) ((mxx 1 10837.68)
           ((vois4) (((nivg 0 100))))) ((nivg 100 256))) () vab (0 1)]
 3: g7 = (fs_non g5)
     | objets {d}: #[((0 256) (0 256)) (((nivg 100 256))) (gl) vab (1 2)]
     | ~objets {do}: #[((0 256) (0 256)) (((nivg 0 100))) () vab (0 1)]
 4: g22 = (fs_remplir g7 vvois4)
     | rel1 {d}: #[((0 256) (0 256)) (((nivg 100 256)) (dans ((vois4) (((nivg 0 100)))) (((nivg 100 256)))))
           (gl) vab (1 2)]
     |? {do}: #[((0 256) (0 256)) ((hors ((vois4) (((nivg 0 100)))) (((nivg 100 256))))) () vab (0 1)]
 5: g24 = (fs_reconst g22 g17 vvois4)
     | rel4 {d}: #[((0 256) (0 256)) ((inter ((vois4) (((nivg 100 256)) (dans ((vois4) (((nivg 0 100))))
           (((nivg 100 256)))))) (((mxx 0 1) (srf 6 11) ((vois4) (((nivg 0 100)))))))) (gl) vab (1 2)]
     |? {do}: #[((0 256) (0 256)) ((hors ((vois4) (((nivg 0 100)))) (((nivg 100 256)))) (ninter ((vois4)
           (((nivg 100 256)) (dans ((vois4) (((nivg 0 100)))) (((nivg 100 256)))))) (((mxx 0 1) (srf 6 11)
           ((vois4) (((nivg 0 100)))))))) () vab (0 1)]
 3: g7 = (fs_non g5)
     | objets {d}: #[((0 256) (0 256)) (((nivg 100 256))) (gl) vab (1 2)]
     |  objets {do}: #[((0 256) (0 256)) (((nivg 0 100))) () vab (0 1)]
```

```
4: g27 = (fs_reconst g7 g24 vvois4)
   | rel5 {d} : #[((0 256) (0 256)) ((inter ((vois4) (((nivg 100 256)))) ((inter ((vois4) (((nivg 100 256))
         (dans ((vois4) (((nivg 0 100)))) (((nivg 100 256)))))) (((mxx 0 1) (srf 6 11) ((vois4)
         (((nivg 0 100)))))))))) (gl) vab (1 2)]
   |? {do} : #[((0 256) (0 256)) (((nivg 0 100)) (ninter ((vois4) (((nivg 100 256)))) ((inter ((vois4)
         (((nivg 100 256)) (dans ((vois4) (((nivg 0 100)))) (((nivg 100 256)))))) (((mxx 0 1) (srf 6 11)
         ((vois4) (((nivg 0 100)))))))))) () vab (0 1)]
```

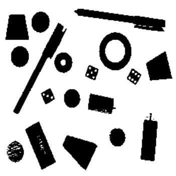
Si nous reprenons un des éléments produits par la planification nous pouvons constater que nous avons accès à la totalité des informations de description qui lui sont associées. Pour exemple, la donnée g22 nous fourni l'opérateur qui l'a générée -(`fs_remplir g7 vvois4`)- le concept qu'elle décrit -`rel1`- ainsi que les descriptions des deux vecteurs `d` et `do` de représentation de cette donnée.

4.1.4.2 Programme exécutable

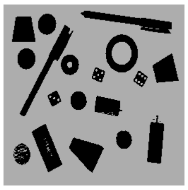
```
clear
echo SCRIPT MExTI
echo ============
echo Image initiale : $1
rm -f *.dsc
echo : FS_LIRE $1 g3
cp $1 g3
echo : FS_SEUILLER g3 0 100 g5
oseuiller g3 g5 0 100
echo : FS_NON g5 g7
onot g5 g7
echo : FS_NON g5 g7
onot g5 g7
echo : FS_LABEL 1 g5 g11
olabel 1 g5 g11
echo : FS_FILTRE g11 mxx 0 1 g13
omesure g11 ExTImg 10
ofiltre g11 ExTImg g13 0 1
rm -f ExTImg
echo : FS_LABEL 1 g13 g15
olabel 1 g13 g15
echo : FS_FILTRE g15 srf 6 11 g17
omesure g15 ExTImg 5
ofiltre g15 ExTImg g17 6 11
rm -f ExTImg
echo : FS_REMPLIR 1 g7 g22
oremplir 1 g7 g22
echo : FS_RECONST 1 g22 g17 g24
oreconst 1 g22 g17 g24
echo : FS_RECONST 1 g7 g24 g27
oreconst 1 g7 g24 g27
/home/elsa2/xv/xv-3.00/xv -na Solution g27&
echo Execution terminee
```

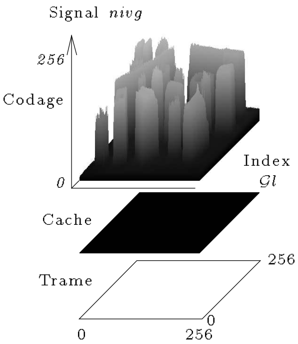
4.1.4.3 Données produites après exécution de ce script

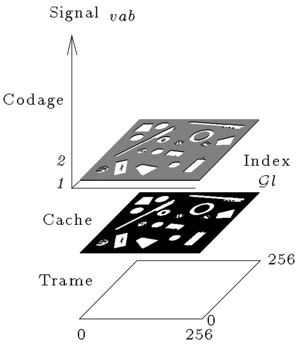

g3 = (fs_lire g1 im)

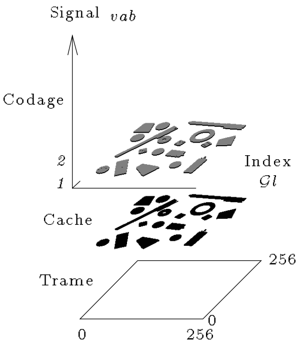

g5 = (fs_seuiller g3 f0 f100)

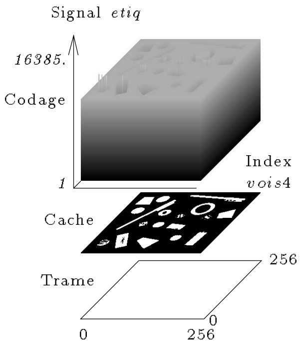

g7 = (fs_non g5)


g11 = (fs_label g5 vvois4)

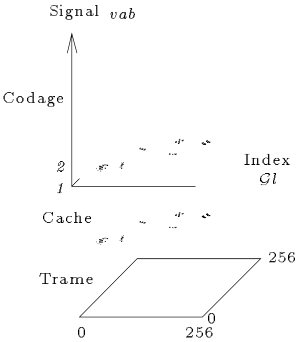




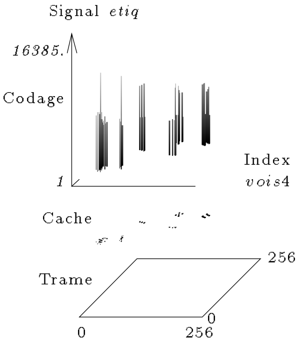


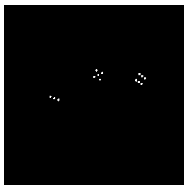


g13 = (fs_filtre g11 f0 f1 mxx)

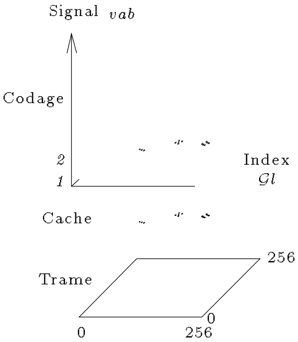


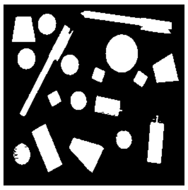
g15 = (fs_label g13 vvois4)



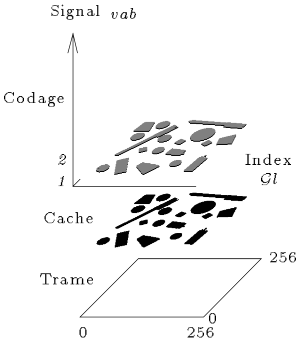


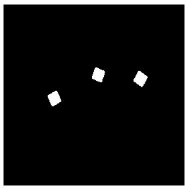
g17 = (fs_filtre g15 f6 f11 srf)



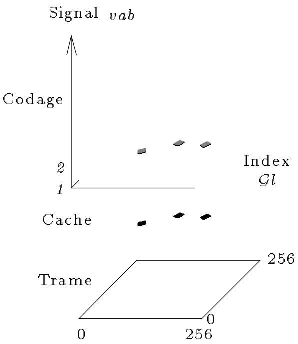


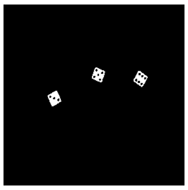
g22 = (fs_remplir g7 vvois4)



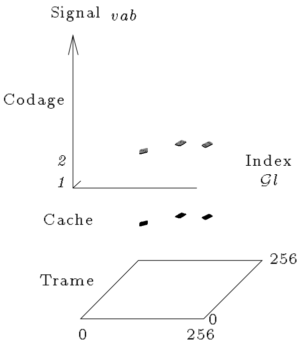


g24 = (fs_reconst g22 g17 vvois4)





g27 = (fs_reconst g7 g24 vvois4)

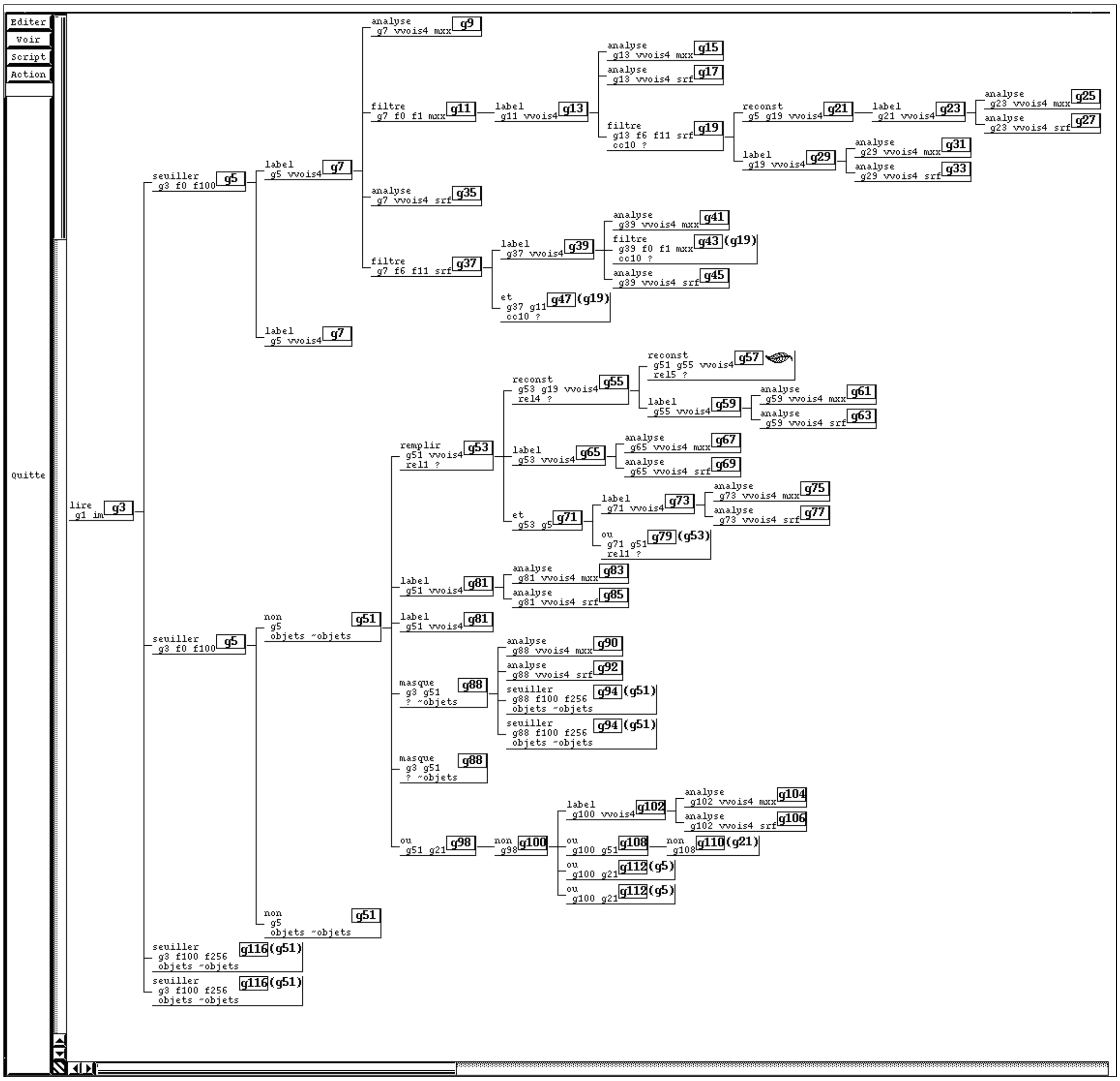


4.1.5 Graphe d'une planification totale

Le graphe suivant correspond à une utilisation du module de planification simple de ExTI pour une recherche de toutes les solutions permettant de résoudre l'objectif `rel5` à partir de la base d'opérateurs que nous possédons. Les noms des données, mis entre parenthèses, désignent une entité équivalente, déjà générée -le module de planification arrête alors le développement du graphe pour la donnée courante.

Nous pouvons donc distinguer quatre solutions équivalentes :

[g3 , g5 , g7 , g11 , g13 , g19 , g51 , g53 , g55 , g57]
[g3 , g5 , g7 , g37 , g39 , g43 , g51 , g53 , g55 , g57]
[g3 , g5 , g7 , g11 , g13 , g19 , g116 , g53 , g55 , g57]
[g3 , g5 , g7 , g37 , g39 , g43 , g116 , g53 , g55 , g57]



4.2 De l'emploi des macro-opérateurs

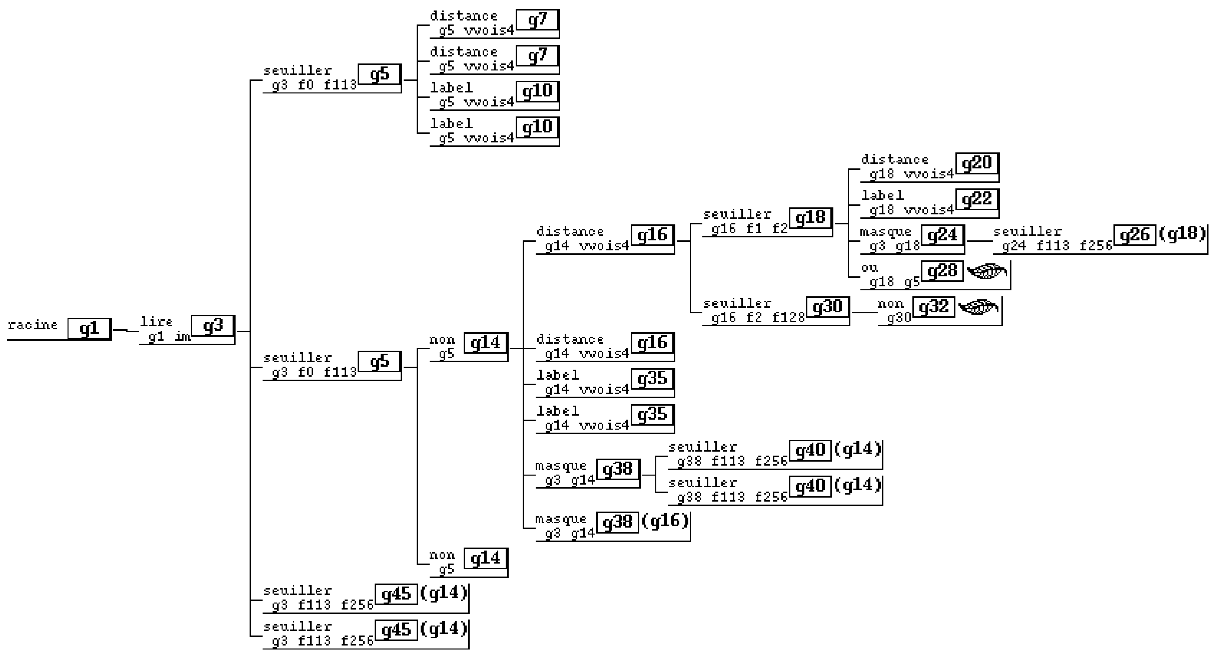
Cette partie des résulats va montrer l'intérêt de supprimer les opérateurs décomposables de la base d'opérateurs symboliques, servant à la planification. Nous montrons alors comment ils doivent être re-pris en compte lors de l'écriture du script de résolution.

Nous proposons ainsi la résolution d'un objectif simple permettant d'utiliser les opérateurs de dilatation et d'érosion morphologique. Trois planifications simples sont montrées. La première met en jeu ces deux opérateurs, la seconde uniquement l'opérateur de dilatation, enfin la troisième témoigne de l'état de la base actuelle dans laquelle les deux opérateurs sont retirés.

Première planification

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : (((dist 1 2) ((vois4) (((nivg 113 256)))))
((nivg 0 113)))
| Index : (g1)
| Signal : vab
| Codage : (1 2)

g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f113)
        [ncpt]
g7 = (fs_distance g5 vvois4)
g10 = (fs_label g5 vvois4)
g14 = (fs_non g5)
        [cpt]
g16 = (fs_distance g14 vvois4)
g18 = (fs_seuiller g16 f1 f2)
g20 = (fs_distance g18 vvois4)
g22 = (fs_label g18 vvois4)
g24 = (fs_masque g3 g18)
g26 = (fs_seuiller g24 f113 f256)
        (fs_seuiller g24 f113 f256) -->g18
g28 = (fs_ou g18 g5)
        [dilatation]
```

```
Le but est atteint par egalite stricte
g30 = (fs_seuiller g16 f2 f128)
        [erosion]
g32 = (fs_non g30)
        [dilatation]
        (fs_non g30) -->g28
g35 = (fs_label g14 vvois4)
g38 = (fs_masque g3 g14)
g40 = (fs_seuiller g38 f113 f256)
        [cpt]
        (fs_seuiller g38 f113 f256) -->g14
        (fs_seuiller g38 f113 f256) -->g14
        (fs_masque g3 g14) -->g16
g45 = (fs_seuiller g3 f113 f256)
        [cpt]
        (fs_seuiller g3 f113 f256) -->g14
        (fs_seuiller g3 f113 f256) -->g14
Temps d'execution (CPU) de la planification :
10.26001s
Nombre d'operateurs testes : 28
Temps moyen de test par operateur : .3664289s

Solutions : (g28 g32)
```



À partir de cette planification, nous pouvons écrire au moins deux scripts résolvant l'objectif par les données g28 et g32. Les scripts suivants correspondent à ces programmes. Ils ont été générés avec et sans l'utilisation de macro-opérateurs. Nous pouvons ainsi remarquer les deux plans à reconnaître pour être remplacés par le macro-opérateur.

Scripts par g28 et g32 sans macro-opérateur :

```
clear
echo SCRIPT MExTI
echo ============
echo Image initiale: $1
rm -f *.dsc
echo : FS_LIRE $1 g3
cp $1 g3
echo : FS_SEUILLER g3 0 113 g5
oseuiller g3 g5 0 113
echo : FS_NON g5 g14
onot g5 g14
echo : FS_DISTANCE 1 g14 g16
oordre 1 g14 g16
echo : FS_SEUILLER g16 1 2 g18
oseuiller g16 g18 1 2
echo : FS_OU g18 g5 g28
oou g18 g5 g28
/home/elsa2/xv/xv-3.00/xv -na Solution
g28&
echo Execution terminee
```

```
clear
echo SCRIPT MExTI
echo ============
echo Image initiale: $1
rm -f *.dsc
echo : FS_LIRE $1 g3
cp $1 g3
echo : FS_SEUILLER g3 0 113 g5
oseuiller g3 g5 0 113
echo : FS_NON g5 g14
onot g5 g14
echo : FS_DISTANCE 1 g14 g16
oordre 1 g14 g16
echo : FS_SEUILLER g16 2 128 g30
oseuiller g16 g30 2 128
echo : FS_NON g30 g32
onot g30 g32
/home/elsa2/xv/xv-3.00/xv -na Solution
g32&
echo Execution terminee
```

Script par g28 et g32 avec macro-opérateur :

```
clear
echo SCRIPT MExTI
echo ============
echo Image initiale: $1
rm -f *.dsc
echo : FS_LIRE $1 g3
cp $1 g3
echo : FS_SEUILLER g3 0 113 g5
oseuiller g3 g5 0 113
# MACRO-OPERATEUR
echo : FS_DILATE 1 g5 g28 1
odilate 1 g5 g28 1
/home/elsa2/xv/xv-3.00/xv -na Solution
g28&
echo Execution terminee
```

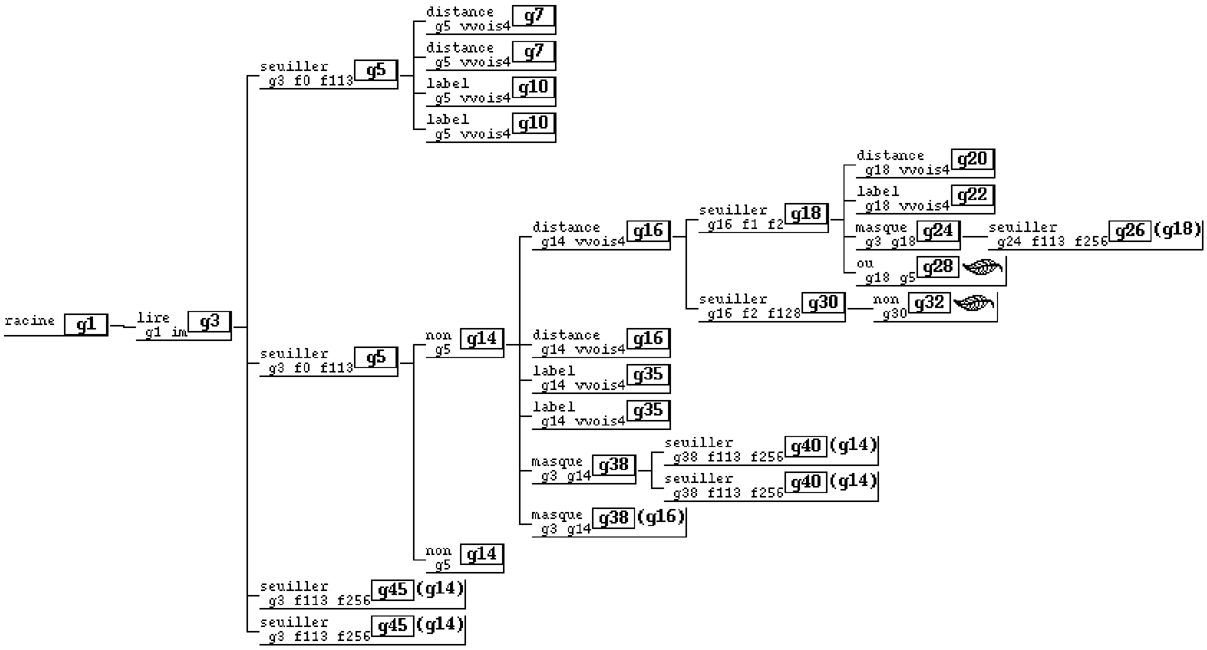
```
clear
echo SCRIPT MExTI
echo ============
echo Image initiale: $1
rm -f *.dsc
echo : FS_LIRE $1 g3
cp $1 g3
echo : FS_SEUILLER g3 0 113 g5
oseuiller g3 g5 0 113
# MACRO-OPERATEUR
echo : FS_DILATE 1 g5 g32 1
odilate 1 g5 g32 1
/home/elsa2/xv/xv-3.00/xv -na Solution
g32&
echo Execution terminee
```

Il est à noter que dans ce cas, le couple d'opérateurs (g16,g30) définissant l'érosion de g14 ne doit pas être remplacé comme dans le script suivant, résolvant g30 :

```
clear
echo SCRIPT MExTI
echo ============
echo Image initiale: $1
rm -f *.dsc
echo : FS_LIRE $1 g3
cp $1 g3
echo : FS_SEUILLER g3 0 113 g5
oseuiller g3 g5 0 113
echo : FS_NON g5 g14
onot g5 g14
# MACRO-OPERATEUR
echo : FS_ERODE 1 g14 g30 1
oerode 1 g14 g30 1
echo Execution terminee
```

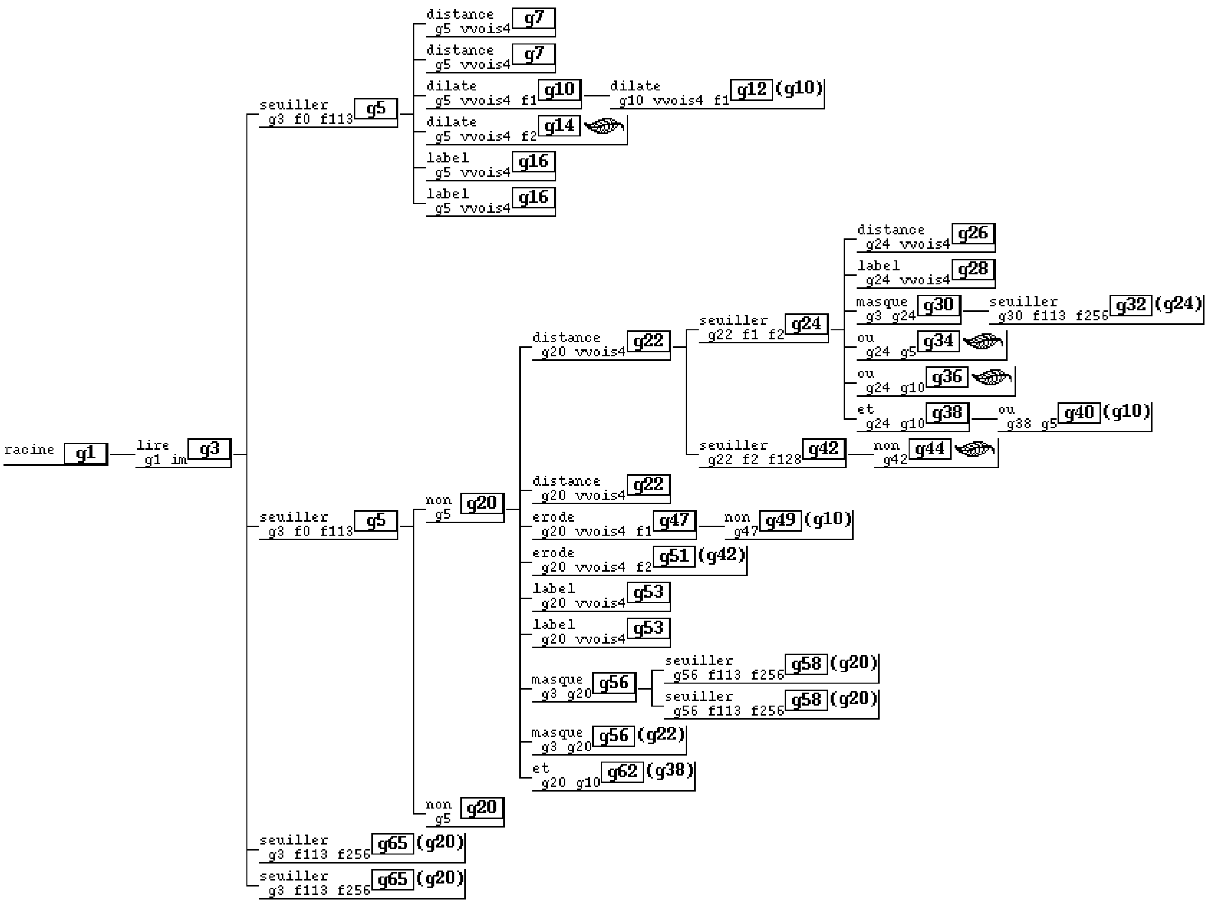
Deuxième planification

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : (((dist 1 2) ((vois4) (((nivg 113 256)))))
((nivg 0 113)))
| Index : (g1)
| Signal : vab
| Codage : (1 2)

g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f113)
        [ncpt]
g7 = (fs_distance g5 vvois4)
g10 = (fs_dilate g5 vvois4 f1)
g12 = (fs_dilate g10 vvois4 f1)
        (fs_dilate g10 vvois4 f1) -->g10
g14 = (fs_dilate g5 vvois4 f2)
        [dilatation]
Le but est atteint par egalite stricte
g16 = (fs_label g5 vvois4)
g20 = (fs_non g5)
        [cpt]
g22 = (fs_distance g20 vvois4)
g24 = (fs_seuiller g22 f1 f2)
g26 = (fs_distance g24 vvois4)
g28 = (fs_label g24 vvois4)
g30 = (fs_masque g3 g24)
g32 = (fs_seuiller g30 f113 f256)
        (fs_seuiller g30 f113 f256) -->g24
g34 = (fs_ou g24 g5)
        [dilatation]
        (fs_ou g24 g5) -->g14
g36 = (fs_ou g24 g10)
        [dilatation]
        (fs_ou g24 g10) -->g14
g38 = (fs_et g24 g10)
g40 = (fs_ou g38 g5)
        (fs_ou g38 g5) -->g10
g42 = (fs_seuiller g22 f2 f128)
        [erosion]
g44 = (fs_non g42)
        [dilatation]
        (fs_non g42) -->g14
g47 = (fs_label g20 vvois4)
g50 = (fs_masque g3 g20)
g52 = (fs_seuiller g50 f113 f256)
        [cpt]
        (fs_seuiller g50 f113 f256) -->g20
        (fs_seuiller g50 f113 f256) -->g20
        (fs_masque g3 g20) -->g22
g56 = (fs_et g20 g10)
        (fs_et g20 g10) -->g38
g59 = (fs_seuiller g3 f113 f256)
        [cpt]
        (fs_seuiller g3 f113 f256) -->g20
        (fs_seuiller g3 f113 f256) -->g20
Temps d'execution (CPU) de la planification :
48.61011s
Nombre d'operateurs testes : 74
Temps moyen de test par operateur : .6568934s

Solutions : (g14 g34 g36 g44)
```



Troisième planification

```
OBJECTIF =
| Trame : ((0 256) (0 256))
| Cache : (((dist 1 2) ((vois4) (((nivg 113 256))))) ((nivg 0 113)))
| Index : (g1)
| Signal : vab
| Codage : (1 2)

g3 = (fs_lire g1 im)
g5 = (fs_seuiller g3 f0 f113)
         [ncpt]
g7 = (fs_distance g5 vvois4)
g10 = (fs_dilate g5 vvois4 f1)
g12 = (fs_dilate g10 vvois4 f1)
         (fs_dilate g10 vvois4 f1) -->g10
g14 = (fs_dilate g5 vvois4 f2)
         [dilatation]
Le but est atteint par egalite stricte
g16 = (fs_label g5 vvois4)
g20 = (fs_non g5)
         [cpt]
g22 = (fs_distance g20 vvois4)
g24 = (fs_seuiller g22 f1 f2)
g26 = (fs_distance g24 vvois4)
g28 = (fs_label g24 vvois4)
g30 = (fs_masque g3 g24)
g32 = (fs_seuiller g30 f113 f256)
         (fs_seuiller g30 f113 f256) -->g24
g34 = (fs_ou g24 g5)
         [dilatation]
         (fs_ou g24 g5) -->g14
g36 = (fs_ou g24 g10)
         [dilatation]
         (fs_ou g24 g10) -->g14
g38 = (fs_et g24 g10)
g40 = (fs_ou g38 g5)
          (fs_ou g38 g5) -->g10
g42 = (fs_seuiller g22 f2 f128)
          [erosion]
g44 = (fs_non g42)
          [dilatation]
          (fs_non g42) -->g14
g47 = (fs_erode g20 vvois4 f1)
g49 = (fs_non g47)
          (fs_non g47) -->g10
g51 = (fs_erode g20 vvois4 f2)
          [erosion]
          (fs_erode g20 vvois4 f2) -->g42
g53 = (fs_label g20 vvois4)
g56 = (fs_masque g3 g20)
g58 = (fs_seuiller g56 f113 f256)
          [cpt]
          (fs_seuiller g56 f113 f256) -->g20
          (fs_seuiller g56 f113 f256) -->g20
          (fs_masque g3 g20) -->g22
g62 = (fs_et g20 g10)
          (fs_et g20 g10) -->g38
g65 = (fs_seuiller g3 f113 f256)
          [cpt]
          (fs_seuiller g3 f113 f256) -->g20
          (fs_seuiller g3 f113 f256) -->g20
Temps d'execution (CPU) de la planification : 1mn 19.44006s
Nombre d'operateurs testes : 105
Temps moyen de test par operateur : .756572s

Solutions : (g14 g34 g36 g44)
```



Nous pouvons comme précédemment établir un tableau comparatif des temps d'exécution de ces planifications :

Macro-opérateurs introduits	Nombre d'opérateurs testés	Durées de Résolution	Temps par opérateur
aucun	28	10,26s	0,37s
dilatation	74	48,61s	0,66s
dilatation , érosion	105	79,44s	0,76s

Il montre ainsi l'augmentation assez significative du temps de planification lorsque de tels macro-opérateurs sont introduits dans la base d'opérateurs symboliques.

4.3 Temps d'exécutions et complexité

Sur les graphiques suivants, nous avons voulu montrer le comportement du planifieur au niveau des temps d'exécution. Le nombre d'opérateurs du graphe solution n'étant pas un critère fiable puisque très inférieur au nombre total d'opérateurs testés, nous avons choisi ce dernier pour évaluer la complexité des modules simples et par réutilisation de plans. Nous avons donc calculé ces temps de planification sur des ensembles d'objectifs relatifs à différents types de domaines, en utilisant pour chacun d'eux les deux modes de planification. Les deux premiers graphiques (fig.4.1) montrent cette évolution sur l'ensemble résultats.

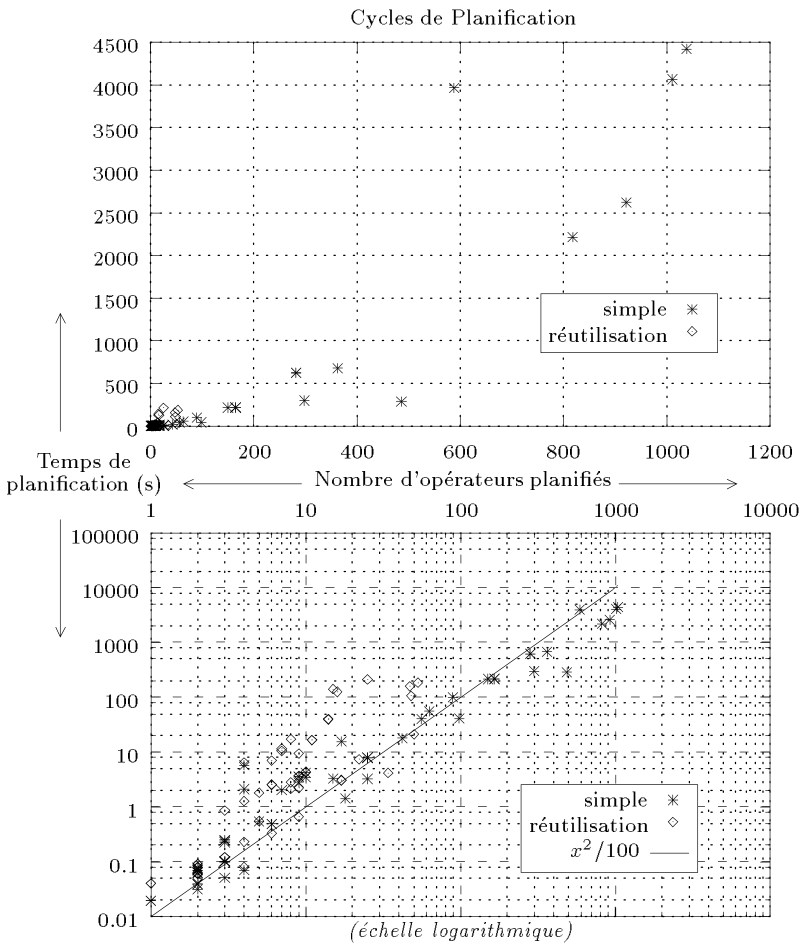


FIG. 4.1 - *Totalité des résultats*

Les deux suivants (fig.4.2) ne montrent que la partie "basse" des traitements pour laquelle nous avons conservé les plans dans lesquels le nombre d'opérateurs testés était inférieur à 100. Nous pouvons ainsi constater que la totalité des objectifs planifiés à l'aide du module de réutilisation de plans se trouve dans cette partie basse.

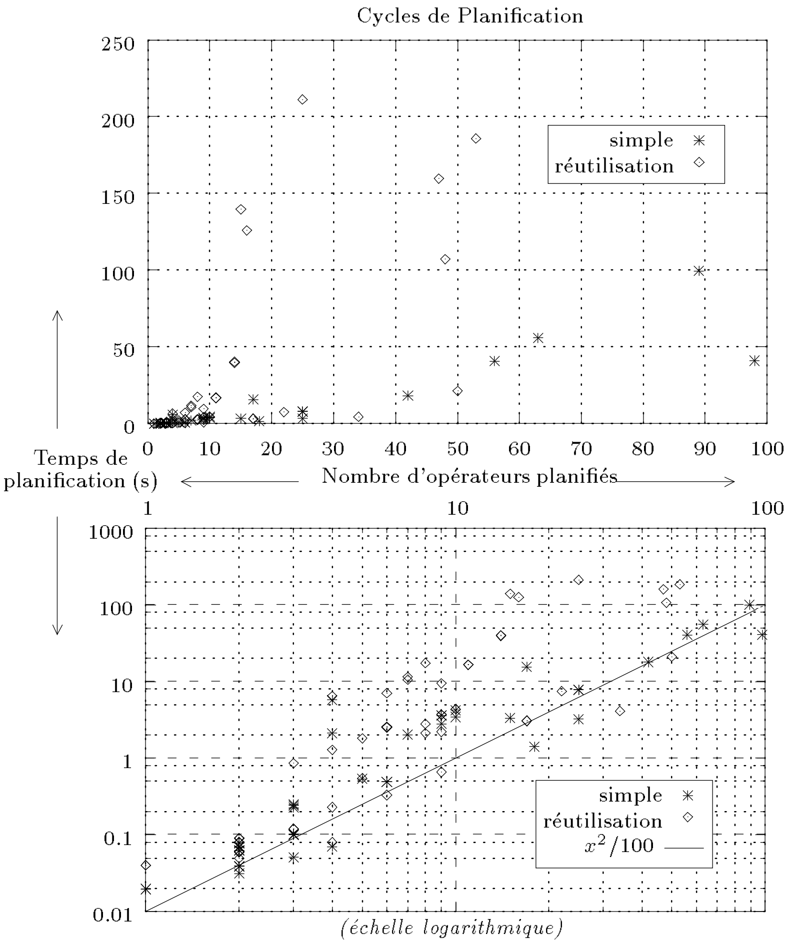


FIG. 4.2 - *Résultats partiels*

Enfin pour être complet, nous avons représenté sur le graphique suivant (fig.4.3), et pour chaque objectif planifié, la différence entre la durée d'une reconstruction -valeur inférieure du segment tracé- et la durée d'une planification par réutilisation de plan -valeur supérieure du segment tracé.

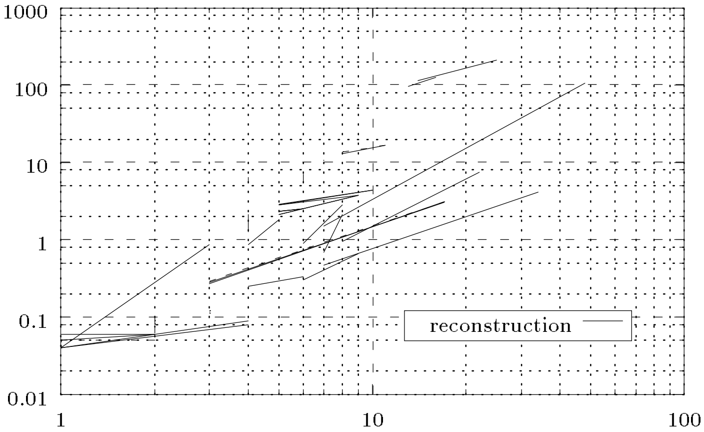


FIG. 4.3 - *Reconstruction et réutilisation*

Conclusion

L'étude que nous avons menée sur le formalisme des entités de traitement et de l'analyse des images a abouti au développement d'un modèle de la donnée exprimée en termes d'indices visuels. Ainsi, toute donnée manipulée sera considérée sous la forme d'une double structure définissant une partition entre les éléments relatifs aux zones d'intérêt et leurs complémentaires. Chaque structure est alors décomposée en cinq champs : la trame, le cache, l'index, le signal et le codage.

Partant de ce modèle de la donnée nous nous sommes appuyé sur les travaux de J.-M. Inglebert pour développer et caractériser les grammaires permettant de définir le langage d'expression d'objectifs LEO. L'implémentation de ce modèle et de ces grammaires dans le système de planification automatique de chaînes d'opérateurs de traitement d'images ExTI, nous a contraint à définir et à caractériser les fonctions de transformation mises en œuvre par les opérateurs. Ainsi nous avons repris le modèle de ces opérateurs de traitement d'images, basé sur une représentation en termes de transformation d'information ou de combinaisons de ces transformations, pour le décomposer en un ensemble de fonctions de transformations des champs de la donnée.

La construction automatique d'un graphe d'opérateurs, résolvant un objectif de traitement d'images, pose essentiellement quatre problèmes relatifs à la planification de la chaîne, à l'exécution et sa supervision, à la validité des résultats produits et à l'explication des éléments mis en œuvre et des propriétés du traitement d'images.

Le formalisme que nous proposons donne des réponses à ces problèmes ou, pour le moins, fournit des éléments nécessaires à la mise en œuvre d'outils appropriés à leur résolution.

Planification

Dans le cadre de la planification, ce formalisme permet d'exprimer des objectifs mais aussi, par les deux modèles rigoureux des données et des opérateurs, de reconstruire les concepts présents dans cet objectif. Pour augmenter les capacités du système ExTI qui exploite ces notions, nous avons développé, à partir des grammaires formelles reconnaissant les expressions de LEO, un analyseur syntaxique chargé de contrôler l'expression fournie comme objectif à atteindre et de simplifier et réduire celle-ci sous une forme canonique que nous avons définie -langage LEO_{FC}. Les représentations cohérentes des opérateurs et des données fournissent des descriptions des résultats intermédiaires exprimées dans le langage de description de données LEO_{FC}. Le module d'appariement et de filtrage détermine alors de façon précise la pertinence des données intermédiaires générées au regard de l'objectif.

Parallèlement, un module spécifique de planification a été développé. Il intègre la notion de réutilisation de plans étiquetés correspondant à des descriptions de concepts du domaine de l'application, dont la résolution a déjà été effectuée et validée. Le quadruplet, alors défini par le terme désignant le concept, son expression dans le langage LEO, le plan de résolution et le script de traitement d'images, forme alors un élément de ce que nous avons appelé le vocabulaire exécutable du domaine d'application.

Cet ensemble de modèles et de processus spécifiques définit un noyau complet de planification symbolique d'objectifs de traitement d'images. L'aspect le plus intéressant de ce système est de fournir des descriptions, exploitables par une interface *Homme-Machine*, de toutes les données créées et manipulées pendant l'exécution du plan.

Nous pouvons alors envisager d'intégrer un tel ensemble dans un système de manipulation des opérateurs physiques de traitement d'images. Via une interface adaptée implémentant, par exemple, des processus de la *Réalité Augmentée*, ceci permettrait de proposer, à l'utilisateur de tels systèmes, les descriptions des résultats produits. Ainsi, il disposerait d'éléments autre que la simple visualisation des données produites pour estimer la pertinence de ces dernières, au regard du problème qu'il doit résoudre. La faisabilité de telles interfaces a été établie [Cap95] et de nouveaux développements sont en cours.

Exécution

Depuis le début de nos recherches, nous avons souhaité ne pas intégrer de contrôle d'exécution dans le système de planification. La raison essentielle était de pouvoir développer un système de manipulation d'opérateurs symboliques rigoureux, complet et cohérent, fournissant tous les éléments descriptifs nécessaires. Cet ensemble de modèles a été ainsi développé tout en gardant bien en vue l'extrême nécessité de la mise en œuvre d'un processus de supervision et de contrôle de l'exécution des plans générés. La décomposition de la représentation de la donnée en deux structures de cinq caractéristiques, et le langage LEO_{FC} permettent de déterminer avec précision les éléments des données intermédiaires à contrôler.

Nous pouvons donc nous appuyer sur ce formalisme pour développer des mécanismes de contrôle d'exécution. Celui-ci peut alors être *a priori*, avec par exemple et dans le cas d'ajustement de paramètres ou de plan, le développement d'un langage spécifique basé sur des règles de type transformationnelles, ou *a posteriori*, par l'adaptation du modèle de données symboliques à un modèle de données réelles intégrant, par exemple, des mesures sur les domaines des descripteurs. Ce nouveau modèle faciliterait alors la mise en correspondance des schémas symboliques -plan, données-attendus et les données effectivement créées, pour ainsi vérifier la validité du traitement en cours d'exécution. Des recherches concernant le contrôle d'exécution basé sur nos modèles sont en cours [Gre96].

Validation

Dans le cadre de la validation des résultats obtenus, la cohérence et la rigueur de l'ensemble des modèles est à nouveau l'élément le plus important du système. Les descriptions des données produites fournissent ainsi les domaines de représentativité ou de résolution des programmes générés. De ce fait, la validation de ces descriptions offre la possibilité d'une réutilisation des programmes pour d'autres domaines d'application par ajustement des plans et scripts associés.

Explication

C'est peut-être une des caractéristiques les plus spécifiques à cet ensemble de modèles et ce langage d'expression. En effet, la recherche d'un formalisme précis décrivant les transformations d'information effectuées par les opérateurs nous a conduit à proposer une écriture particulière de ceux-ci sous la forme d'un ensemble de neuf fonctions ou compositions de fonctions strictement définies. Ainsi, une chaîne de traitement peut-être exprimée sous une forme littérale. Cette écriture permet ainsi d'appréhender les éléments spécifiques mis en jeu par les opérateurs, mais aussi de pouvoir classer ces derniers selon de nouveaux critères : champ d'application, fonction de transformation utilisée, etc.

Ainsi, nous disposons de nouvelles représentations pour parler du traitement d'images et de ce fait pour l'enseigner. Ce formalisme a effectivement pu être utilisé dans le cadre d'un enseignement en DESS d'Informatique, Reconnaissance des formes et Robotique pour décrire les opérateurs de traitement d'images et permettre aux étudiants de faire le lien entre l'étude des opérateurs et la résolution d'un problème. D'autre part, ces nouvelles classifications peuvent être à la base d'outils de comparaison d'effets produits permettant de développer des modules d'aide à la configuration manuelle d'application.

Nous pensons que le formalisme utilisé pour représenter les données de traitement d'images doit être de très bas niveau, à la fois pour décrire de façon très fine le traitement effectué par les opérateurs et pour éviter d'établir *a priori* des liens entre opérateurs et objectifs de traitement d'images.

Le modèle de description des opérateurs symboliques s'appuie sur ce formalisme et permet de propager les transformations opérées par une chaîne de traitements. La description de ces opérateurs en termes d'effets produits permet d'introduire une expertise en traitement d'images (liens opérateurs-caractéristiques produites). Enfin la description des relations entre objets du domaine et ces indices visuels constitue l'expertise du domaine. Notre modèle évite d'introduire ces expertises dans le formalisme de base. Par contre, il permet de les introduire à l'aide d'environnement de conception pour produire des applications génériques ou dédiées, tout en gardant le lien avec le traitement d'images. Ceci permet d'une part de formuler des objectifs de traitement d'images et d'autre part de configurer le système dédié en planifiant la chaîne qui reconstruit la description de ces objectifs.

La base de notre étude a été un système de planification automatique de chaîne de traitement d'images (thèse de J.-M. Inglebert, 1992) qui montrait la faisabilité d'une construction de séquence d'opérateurs résolvant un objectif exprimé en termes d'indices visuels. Nous avons souhaité dans ce travail revenir sur les notions implémentées dans ce système afin de développer un formalisme précis des entités du traitement d'images. Nous avons ainsi approfondi ces notions et proposé de nouveaux modèles plus représentatifs des manipulations effectuées au cours du processus d'expression d'objectifs, de planification et de façon indirecte, d'exécution. L'intégration de ce formalisme dans ce système nous a alors permis d'augmenter les performances de celui-ci, tant sur le plan de la complexité des problèmes résolvables, que sur le plan des temps de planification avec un gain non négligeable.

Annexes

Annexe A

Notations

D	Donnée de traitement d'images.
n	Dimension de la donnée.
R_D	Repère de la donnée D.
π	Point d'une donnée.
Υ^n	Référentiel de dimension n.
R^n	Repère du référentiel de dimension n.
τ^n	Trame de dimension n
$\mathcal{T}()$	Taille d'une trame
T_n	Ensemble des trames de dimension n
κ_δ^n	Cache de dimension n à description δ
δ	Description d'un cache
f_δ	Fonction cache à description δ
$\Lambda_\Upsilon^n(\pi)$	Description d'un point
∇_Υ^n	Ensemble des descriptions valides
F_Υ^n	Ensemble des fonctions caches à descriptions valides
Δ_p	Domaine de validité d'un sélecteur p
K_n	Ensemble des caches de dimension n
ι	Index affecté au cache de la donnée
$\mathcal{G}l$	Critère d'index global
$I_{\mathcal{G}l}$	Ensemble des index de globalité
$\mathcal{I}n$	Critère d'indépendance d'index
$I_{\mathcal{I}n}$	Ensemble des index d'indépendance
I_χ	Ensemble des index à critère spécifique
I	Ensemble de tous les index

σ	Signal d'une donnée
Sgn^n	Ensemble des signaux définis sur une dimension n
ω	Codage d'une donnée
$\sim$x,$\overline{x}$	Complémentaire de x
$\bar{\iota}$	Index affecté au complémentaire du cache de la donnée
$\overline{\sigma}$	Signal du complémentaire de la donnée
υ	Ensemble des valeurs numériques
ς	Ensemble des sélecteurs
$\check{\varsigma}$	Ensemble des descripteurs de sous-sélecteurs
θ	Ensemble des descripteurs d'opération arithmétique
ρ	Ensemble des descripteurs de relation
ϕ	Ensemble des descripteurs de figure
ζ	Ensemble des descripteurs de déstructuration
$\hat{\zeta}$	Ensemble des descripteurs de pointeurs de déstructuration
η	Ensemble des descripteurs modificatif de descripteurs
μ	Ensemble des descripteurs de jonction de concepts
ν	Ensemble des descripteurs de jonction de sélecteurs
Sel	Catégorie syntaxique <*sélecteur*>
Mes	Catégorie syntaxique <*mesure*>
Fct	Catégorie syntaxique <*fonction_sur_des_mesures*>
Dom	Catégorie syntaxique <*domaine*>
Ppt	Catégorie syntaxique <*propriété*>
Cpt	Catégorie syntaxique <*concept*>
Bse	Catégorie syntaxique <*base*>
Dst	Catégorie syntaxique <*déstructuration_d'une_base*>
Trm	Catégorie syntaxique <*trame*>
Cch	Catégorie syntaxique <*cache*>
Ind	Catégorie syntaxique <*index*>
Sgn	Catégorie syntaxique <*signal*>
Cdg	Catégorie syntaxique <*codage*>
$\mathcal{S}p$	Ensemble générique des fonctions de spécialisation
$\mathcal{G}n$	Ensemble générique des fonctions de généralisation
$\mathcal{D}s$	Ensemble générique des fonctions de déstructuration
$\mathcal{S}t$	Ensemble générique des fonctions de structuration
τ^t	Ensemble des fonctions de transformation des trames
F^η	Ensemble des fonctions de transformation de dimension de trame
$F^{\mathcal{T}}$	Ensemble des fonctions de transformation de taille de trame
$\mathcal{I}d^{\mathcal{T}}$	Transformation identique ou identité.
Φ_d	Ensemble des fonctions à tailles de trame déterminées
Φ_c	Ensemble des fonctions à tailles de trame calculées
$\mathcal{I}d^t$	Transformation identique
$\mathcal{T}^-$	Spécialisation par diminution de la taille de la trame
η_G^-	Spécialisation par diminution de la dimension de la trame
$\eta_{\mathcal{T}_\chi}^-$	Structuration de la trame par une caractéristique
$\mathcal{T}^+$	Généralisation par augmentation de la taille de la trame
η_G^+	Généralisation par augmentation de la dimension de la trame
$\eta_{\mathcal{T}_\chi}^-$	Déstructuration de la trame par une caractéristique
κ^t	Ensemble de fonctions de transformation des caches

$\mathcal{I}d^{\kappa}$	Transformation pseudo-identique
1^t	Transformation de globalité
$\mathcal{I}d^{\delta}$	Transformation identique
$\mathcal{I}d^{\chi}$	Deuxième forme de transformation identique
κ_{χ}^{-}	Spécialisation de la description de la fonction cache
κ_{χ}^{+}	Généralisation de la description de la fonction cache
ι^t	Ensemble de fonctions de transformation des index
$\mathcal{I}d^{\iota}$	Transformation identique
ι_{χ}^{-}	Spécialisation d'index par une caractéristique
ι_{χ}^{+}	Généralisation d'index par une caractéristique
Γ^{-}	Structuration de globalité
Γ_{χ}^{-}	Structuration par une caractéristique
Γ^{+}	Déstructuration d'indépendance
Γ_{χ}^{+}	Déstructuration par une caractéristique
σ^t	Ensemble de fonctions de transformation des signaux
$\mathcal{I}d^{\sigma}$	Transformation identique
Σ^{+}	Déstructuration du signal
Σ^{-}	Structuration du signal
σ_{χ}^{+}	Généralisation du signal par une caractéristique
σ_{χ}^{-}	Spécialisation du signal par une caractéristique
ω^t	Ensemble de fonctions de transformation des codages
$\mathcal{I}d^{\omega}$	Transformation identique
ω^{σ}	Transformation par fixation du codage
ω_{χ}^{-}	Spécialisation du codage par une caractéristique
ω_{χ}^{+}	Généralisation du codage par une caractéristique

Annexe B

Analyseurs syntaxiques de LEO

Pour la construction des analyseurs syntaxiques de LEO, nous allons considérer les noms des catégories syntaxiques utilisées dans la catégorie étudiée non plus comme un terminal de cette dernière -ceci nous avait permis de caractériser LEO-, mais cette fois comme un non terminal. Ceci va nous permettre, entre autre de construire les tables d'analyse prédictive avec récupération sur erreur, éléments de base des analyseurs.
Dans chacune de ces tables d'analyse prédictive, relative à la grammaire G_x d'une catégorie syntaxique x, nous utiliserons le symbole $\bullet$ pour désigner tout terminal de l'ensemble T_{LEO} n'appartenant pas à T_x et non déjà référencé dans la table.

$$T_{LEO} = \{(,), \upsilon, \varsigma, \check{\varsigma}, \theta, \rho, \phi, \zeta, \hat{\zeta}, \eta, \mu, \nu, \mho\}$$

B.1 Les catégories syntaxiques du LDC

B.1.1 Catégorie syntaxique Sel

Expression : $Sel = \varsigma \mid Mes \mid (\eta^{+}\{\varsigma|Mes|\check{\varsigma}\{\varsigma|Mes\}\}\ ) \mid (\check{\varsigma}\{\varsigma|Mes\}\})$
Catégorie syntaxique déterministe.
Grammaire régulière LL(2) :

$$G_{Sel} = [\ \{Sel_0, Sel_1, Sel_2, Sel_3, Mes_0\}\ ,\ \{(,), \varsigma, \check{\varsigma}, \eta, \lambda\}\ ,\ \mathcal{P}_{Sel}\ ,\ Sel_0\ ]$$

$$\text{avec } \mathcal{P}_{Sel} = \left|\begin{array}{lcl} Sel_0 & \longrightarrow & Sel_1 \\ Sel_0 & \longrightarrow & (Sel_2) \\ Sel_1 & \longrightarrow & \varsigma \\ Sel_1 & \longrightarrow & Mes_0 \\ Sel_2 & \longrightarrow & \check{\varsigma}\ Sel_1 \end{array}\right| \begin{array}{lcl} Sel_2 & \longrightarrow & \eta\ Sel_3 \\ Sel_3 & \longrightarrow & \eta\ Sel_3 \\ Sel_3 & \longrightarrow & Sel_1 \\ Sel_3 & \longrightarrow & \check{\varsigma}\ Sel_1 \\ & & \end{array}$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	ς	$\check{\varsigma}$	η	υ	$\bullet$	λ	$(\theta,(\varsigma$	$(\check{\varsigma},(\eta$
Sel_0	–	s_1	Sel_1	s_2	s_2	Sel_1	s_2	s_1	Sel_1	$(\ Sel_2\ )$
Sel_1	Mes_0	s_1	ς	s_2	s_2	υ	s_2	s_1	–	–
Sel_2	s_3	s_1	s_3	$\check{\varsigma}\ Sel_1$	$\eta\ Sel_3$	s_3	s_3	–	–	–
Sel_3	Sel_1	s_4	Sel_1	$\check{\varsigma}\ Sel_1$	$\eta\ Sel_3$	Sel_1	s_4	–	–	–

Avec les erreurs :
s_1 : Une sélection ne peut être vide.
s_2 : Ce n'est pas un descripteur de sélecteur.
s_3 : Un descripteur modificatif ou de sous-sélecteur est attendu.
s_4 : Un descripteur modificatif, de sous-sélecteur, de sélecteur ou une fonction sur des mesures est attendu.

B.1.2 Catégorie syntaxique Mes

Expression : $Mes = \upsilon \mid (Sel\ Dst) \mid Fct$
Catégorie syntaxique déterministe.
Grammaire régulière LL(2) :

$$G_{Mes} = [\ \{Mes_0, Mes_1, Mes_2, Sel_0, Dst_0, Fct_0\}\ ,\ \{(,), \upsilon, \lambda\}\ ,\ \mathcal{P}_{Mes}\ ,\ Mes_0\ ]$$

$$\text{avec } \mathcal{P}_{Mes} = \left| \begin{array}{lcl} Mes_0 & \longrightarrow & \upsilon \\ Mes_0 & \longrightarrow & Fct_0 \\ Mes_0 & \longrightarrow & (Mes_1) \\ Mes_1 & \longrightarrow & Sel_0\ Mes_2 \\ Mes_2 & \longrightarrow & Dst_0 \end{array} \right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	υ	ς	$\bullet$	λ	$(\theta$	$(\varsigma,(($
Mes_0	–	m_1	υ	m_2	m_2	m_1	Fct_0	$(\ Mes_1\ )$
Mes_1	$Sel_0\ Mes_2$	m_1	m_3	$Sel_0\ Mes_2$	m_3	–	–	–
Mes_2	Dst_0	m_4	m_4	m_4	m_4	–	–	–

Avec les erreurs :
m_1 : Une mesure ne peut être vide.
m_2 : Ce n'est pas une valeur numérique.
m_3 : Ce n'est pas un descripteur de sélection.
m_4 : Une description de déstructuration de base est attendue.

B.1.3 Catégorie syntaxique Fct

Expression : $Fct = (\theta\ Mes\ Mes)$
Catégorie syntaxique déterministe.
Grammaire régulière LL(1) :

$$G_{Fct} = [\ \{Fct_0, Fct_1, Fct_2, Fct_3, Mes_0\}\ ,\ \{(,), \theta, \lambda\}\ ,\ \mathcal{P}_{Fct}\ ,\ Fct_0\ ]$$

$$\text{avec } \mathcal{P}_{Fct} = \left| \begin{array}{lcl} Fct_0 & \longrightarrow & (\ Fct_1\ ) \\ Fct_1 & \longrightarrow & \theta\ Fct_2 \\ Fct_2 & \longrightarrow & Mes_0\ Fct_3 \\ Fct_3 & \longrightarrow & Mes_0 \end{array} \right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	θ	υ	$\bullet$	λ
Fct_0	$(\ Fct_1 \ )$	f_1	f_1	f_1	f_1	f_1
Fct_1	f_2	f_1	$\theta \ Fct_2$	f_2	f_2	–
Fct_2	$Mes_0 \ Fct_3$	f_3	f_3	$Mes_0 \ Fct_3$	f_3	–
Fct_3	Mes_0	f_3	f_3	Mes_0	f_3	–

Avec les erreurs :
f_1 : Une fonction sur des mesures est une liste non vide.
f_2 : Un descripteur de fonction arithmétique est attendu.
f_3 : Une description de mesure est attendue.

B.1.4 Catégorie syntaxique Dom

Expression : $Dom = \{Mes \ Mes\}^*$
Catégorie syntaxique déterministe.
Grammaire régulière LL(1) :

$$G_{Dom} = [\ \{Dom_0, Dom_1, Dom_2, Mes_0\} \ , \ \{\lambda\} \ , \ \mathcal{P}_{Dom} \ , \ Dom_0 \ ]$$

$$\text{avec } \mathcal{P}_{Dom} = \left| \begin{array}{lcl} Dom_0 & \longrightarrow & Mes_0 \ Dom_1 \\ Dom_0 & \longrightarrow & \lambda \\ Dom_1 & \longrightarrow & Mes_0 \ Dom_2 \\ Dom_2 & \longrightarrow & Mes_0 \ Dom_1 \\ Dom_2 & \longrightarrow & \lambda \end{array} \right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	υ	$\bullet$	λ
Dom_0	$Mes_0 \ Dom_1$	λ	$Mes_0 \ Dom_1$	m_2	λ
Dom_1	$Mes_0 \ Dom_2$	d_1	$Mes_0 \ Dom_2$	m_2	d_1
Dom_2	$Mes_0 \ Dom_1$	λ	$Mes_0 \ Dom_1$	m_2	λ

Avec les erreurs :
d_1 : Un domaine possède un nombre pair d'éléments
m_2 : Ce n'est pas une valeur numérique (voir catégorie syntaxique Mes).

B.1.5 Catégorie syntaxique Ppt

Expression : $Ppt = (Sel \ Dom)$
Catégorie syntaxique déterministe.
Grammaire régulière LL(1) :

$$G_{Ppt} = [\ \{Ppt_0, Ppt_1, Ppt_2, Dom_0\} \ , \ \{(,),\lambda\} \ , \ \mathcal{P}_{Ppt} \ , \ Ppt_0 \ ]$$

$$\text{avec } \mathcal{P} = \left| \begin{array}{lcl} Ppt_0 & \longrightarrow & (\ Ppt_1 \ ) \\ Ppt_1 & \longrightarrow & Sel_0 \ Ppt_2 \\ Ppt_2 & \longrightarrow & Dom_0 \end{array} \right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	ς	υ	$\bullet$	λ
Ppt_0	$(\ Ppt_1 \ )$	p_1	p_1	p_1	p_1	p_1
Ppt_1	$Sel_0 \ Ppt_2$	p_2	$Sel_0 \ Ppt_2$	p_2	p_2	–
Ppt_2	Dom_0	p_3	p_3	Dom_0	p_3	–

Avec les erreurs :
p_1 : Une propriété est une liste non vide
p_2 : Une description de sélecteur est attendue.
p_3 : Une description de domaine est attendue.

B.1.6 Catégorie syntaxique Cpt

Expression : $Cpt = (Ppt^+ \ \{\lambda | Dst\}) | (\rho \ Dst \ Bse^+) | (\phi \ Mes^+) | (\nu \ Bse \ Bse^+) | \mho$
Catégorie syntaxique déterministe.
Grammaire LL(3) :

$$G_{Cpt} = [\ \{Cpt_0, Cpt_1, Cpt_2, Cpt_3, Cpt_4, Cpt_5, Cpt_6, Cpt_7, Cpt_8, Ppt_0, Dst_0, Bse_0, Mes_0\} \ ,$$

$$\{(,), \rho, \phi, \mu, \mho, \lambda\} \ , \ \mathcal{P}_{Cpt} \ , \ Cpt_0 \ ]$$

$$\text{avec } \mathcal{P}_{Cpt} = \left| \begin{array}{lcl} Cpt_0 & \longrightarrow & (Cpt_1) \\ Cpt_0 & \longrightarrow & \mho \\ Cpt_1 & \longrightarrow & Ppt_0 \ Cpt_2 \\ Cpt_1 & \longrightarrow & \rho \ Cpt_3 \\ Cpt_1 & \longrightarrow & \phi \ Cpt_5 \\ Cpt_1 & \longrightarrow & \mu \ Cpt_7 \\ Cpt_2 & \longrightarrow & Ppt_0 \ Cpt_2 \\ Cpt_2 & \longrightarrow & Dst_0 \\ Cpt_2 & \longrightarrow & \lambda \end{array} \right| \begin{array}{lcl} Cpt_3 & \longrightarrow & Dst_0 \ Cpt_4 \\ Cpt_4 & \longrightarrow & Bse_0 \ Cpt_4 \\ Cpt_4 & \longrightarrow & \lambda \\ Cpt_5 & \longrightarrow & Mes_0 \ Cpt_6 \\ Cpt_6 & \longrightarrow & Mes_0 \ Cpt_6 \\ Cpt_6 & \longrightarrow & \lambda \\ Cpt_7 & \longrightarrow & Bse_0 \ Cpt_8 \\ Cpt_8 & \longrightarrow & Bse_0 \\ & & \end{array}$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	ρ	ϕ	μ	$\mho$	υ	$\bullet$	λ	$(\varsigma\upsilon, (\varsigma(, ((\breve{\varsigma}, ((($	$((\zeta$
Cpt_0	$(\ Cpt_1 \ )$	c_1	c_2	c_2	c_2	$\mho$	c_2	c_2	c_1	-	-
Cpt_1	$Ppt_0 \ Cpt_2$	c_1	ρCpt_3	ϕCpt_5	μCpt_7	c_3	c_3	c_3	c_1	-	-
Cpt_2	–	λ	c_4	c_4	c_4	c_4	c_4	c_4	λ	$Ppt_0 \ Cpt_2$	Dst_0
Cpt_3	$Dst_0 \ Cpt_4$	c_5	n_1	n_1	n_1	n_1	n_1	n_1	c_5	-	-
Cpt_4	$Bse_0 \ Cpt_4$	λ	b_1	b_1	b_1	b_1	b_1	b_1	λ	-	-
Cpt_5	$Mes_0 \ Cpt_6$	c_6	m_1	m_1	m_1	m_1	$Mes_0 \ Cpt_6$	m_1	c_6	-	-
Cpt_6	$Mes_0 \ Cpt_6$	λ	m_1	m_1	m_1	m_1	$Mes_0 \ Cpt_6$	m_1	λ	-	-
Cpt_7	$Bse_0 \ Cpt_8$	c_7	b_1	b_1	b_1	b_1	b_1	b_1	c_7	-	-
Cpt_8	Bse_0	c_7	b_1	b_1	b_1	b_1	b_1	b_1	c_7	-	-

Avec les erreurs :
c_1 : Un concept ne peut être vide.
c_2 : Un concept a une structure de liste ou est une étiquette.
c_3 : Ce n'est pas un descripteur de relation, de figure ni de jonction.
c_4 : Propriété ou déstructuration de base attendues.
c_5 : Une relation repose sur au moins une base déstructurée.
c_6 : Une figure possède au moins un paramètre.
c_7 : Une jonction repose sur deux bases.
n_1 : Une déstructuration de base a une structure de liste non vide (voir catégorie syntaxique Dst).
b_1 : Une base est formée d'une liste de concepts (voir catégorie syntaxique Bse).
m_2 : Ce n'est pas une valeur numérique (voir catégorie syntaxique Mes).

B.1.7 Catégorie syntaxique Bse

Expression : $Bse = (Cpt^*)$
Catégorie syntaxique déterministe.
Grammaire régulière LL(1) :

$$G_{Bse} = [\ \{Bse_0, Bse_1, Cpt_0\}\ ,\ \{(,), \lambda\}\ ,\ \mathcal{P}_{Bse}\ ,\ Bse_0\ ]$$

$$\text{avec } \mathcal{P}_{Bse} = \left| \begin{array}{lll} Bse_0 & \longrightarrow & (Bse_1) \\ Bse_1 & \longrightarrow & Cpt_0\ Bse_1 \\ Bse_1 & \longrightarrow & \lambda \end{array} \right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	$\mho$	$\bullet$	λ
Bse_0	(Bse_1)	b_1	b_1	b_1	b_1
Bse_1	$Cpt_0\ Bse_1$	λ	$Cpt_0\ Bse_1$	c_2	λ

Avec les erreurs :
b_1 : Une base est formée d'une liste de concepts.
c_2 : Un concept a une structure de liste ou est une étiquette (voir Catégorie syntaxique Cpt).

B.1.8 Catégorie syntaxique Dst

Expression : $Dst = ((\zeta)\ Bse)$
Catégorie syntaxique déterministe.
Grammaire régulière LL(1) :

$$G_{Dst} = [\ \{Dst_0, Dst_1, Dst_2, Bse_0\}\ ,\ \{(,), \zeta, \lambda\}\ ,\ \mathcal{P}_{Dst}\ ,\ Dst_0\ ]$$

$$\text{avec } \mathcal{P}_{Dst} = \left| \begin{array}{lll} Dst_0 & \longrightarrow & (Dst_1) \\ Dst_1 & \longrightarrow & (Dst_2)\ Bse_0 \\ Dst_2 & \longrightarrow & \zeta \end{array} \right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	ζ	$\bullet$	λ
Dst_0	(Dst_1)	n_1	n_1	n_1	n_1
Dst_1	(Dst_2) Bse_0	n_2	n_2	n_2	n_2
Dst_2	n_3	n_2	ζ	n_4	n_2

Avec les erreurs :
n_1 : Une déstructuration de base a une structure de liste non vide
n_2 : Un élément déstructurant est constitué d'une liste non vide
n_3 : Un descripteur de déstructuration de base est attendu
n_4 : Ce n'est pas un descripteur de déstructuration de base.

B.2 Les catégories syntaxiques du LDD

B.2.1 La Trame : Trm

Expression : $Trm = (\{(Mes\ Mes)\}^*)$
Catégorie syntaxique déterministe.
Grammaire LL(1) :

$$G_{Trm} = [\ \{Trm_0, Trm_1, Trm_2, Mes_0\}\ ,\ \{(,), \lambda\}\ ,\ \mathcal{P}_{Trm}\ ,\ Trm_0\ ]$$

$$\text{avec } \mathcal{P}_{Trm} = \left| \begin{array}{lcl} Trm_0 & \longrightarrow & (Trm_1) \\ Trm_1 & \longrightarrow & (Trm_2)\ Trm_1 \\ Trm_1 & \longrightarrow & \lambda \\ Trm_2 & \longrightarrow & Mes_0\ Mes_0 \end{array} \right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	v	$\bullet$	λ
Trm_0	(Trm_1)	t_1	t_1	t_1	t_1
Trm_1	(Trm_2) Trm_1	λ	t_2	t_2	λ
Trm_2	$Mes_0\ Mes_0$	m_4	$Mes_0\ Mes_0$	m_2	m_4

Avec les erreurs :
t_1 : La structure de la trame est la liste
t_2 : La trame est une liste soit vide soit formée de sous-listes

B.2.2 Le Cache : Cch

Expression : $Cch = t|Bse$
Catégorie syntaxique déterministe.
Grammaire LL(1) :

$$G_{Cch} = [\ \{Cch_0\}\ ,\ \{t, \lambda\}\ ,\ \mathcal{P}_{Cch}\ ,\ Cch_0\ ]$$

$$\text{avec } \mathcal{P}_{Cch} = \left| \begin{array}{lcl} Cch_0 & \longrightarrow & t \\ Cch_0 & \longrightarrow & Bse_0 \end{array} \right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	t	$\bullet$	λ
Cch_0	Bse_0	t	k_1	k_2

Avec les erreurs :
k_1 : La structure du cache non global est la liste.
k_2 : Descripteur de cache attendu.

B.2.3 L'Index : Idx

Expression : $Idx = ()|(gl)|(\hat{\zeta}\ \zeta)$
Catégorie syntaxique déterministe.
Grammaire LL(2) :

$$G_{Idx} = [\ \{Idx_0, Idx_1, Idx_2\}\ ,\ \{(,), gl, \hat{\zeta}, \zeta, \lambda\}\ ,\ \mathcal{P}_{Idx}\ ,\ Idx_0\ ]$$

$$\text{avec } \mathcal{P}_{Idx} = \left| \begin{array}{lcl} Idx_0 & \longrightarrow & (Idx_1) \\ Idx_1 & \longrightarrow & \hat{\zeta}\ Idx_2 \\ Idx_1 & \longrightarrow & gl \\ Idx_1 & \longrightarrow & \lambda \\ Idx_2 & \longrightarrow & \zeta \end{array} \right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	gl	$\hat{\zeta}\zeta$	ζ	$\bullet$	λ
Idx_0	(Idx_1)	i_1	–	i_1	i_1	i_1	i_1
Idx_1	i_2	λ	gl	$\hat{\zeta}\ Idx_2$	i_3	i_3	λ
Idx_2	n_3	λ	–	n_3	ζ	n_3	λ

Avec les erreurs :
i_1 : La structure de l'index est la liste.
i_2 : Descripteur d'index attendu.
i_3 : Ce n'est pas un descripteur d'index.

B.2.4 Le Signal : Sgn

Expression : $Sgn = Sel|(\nu\ Sgn\ Sgn)$
Catégorie syntaxique déterministe.
Grammaire LL(2) :

$$G_{Sgn} = [\ \{Sgn_0, Sgn_1, Sgn_2, Sgn_3, Sel_0\}\ ,\ \{(,),\nu,\lambda\}\ ,\ \mathcal{P}_{Sgn}\ ,\ Sgn_0\ ]$$

$$\text{avec } \mathcal{P}_{Sgn} = \left|\begin{array}{lcl} Sgn_0 & \longrightarrow & Sel_0 \\ Sgn_0 & \longrightarrow & (Sgn_1) \\ Sgn_1 & \longrightarrow & \nu\ Sgn_2 \\ Sgn_2 & \longrightarrow & Sgn_0\ Sgn_3 \\ Sgn_3 & \longrightarrow & Sgn_0 \end{array}\right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	ν	ς	υ	$\bullet$	λ	$(\nu$	$(\eta$,$(\check{s}$,((,$(\varsigma$,$(\theta$
Sgn_0	–	g_1	g_1	Sel_0	Sel_0	g_1	g_1	(Sgn_1)	Sel_0
Sgn_1	g_2	g_2	$\nu\ Sgn_2$	g_2	g_2	g_2	g_2	–	–
Sgn_2	$Sgn_0\ Sgn_3$	g_3	g_3	$Sgn_0\ Sgn_3$	$Sgn_0\ Sgn_3$	g_3	g_3	–	–
Sgn_3	Sgn_0	g_4	g_4	Sgn_0	Sgn_0	g_4	g_4	–	–

Avec les erreurs :
g_1 : Le signal a une structure de liste non vide ou est un sélecteur ou une valeur numérique.
g_2 : Un descripteur de jonction est attendu.
g_3 : Une description de mesure ou de sélecteur est attendue.
g_4 : Une description de sélecteur est attendue.

B.2.5 Le Codage : Cdg

Expression : $Cdg = (Dom)$
Catégorie syntaxique déterministe.
Grammaire LL(1) :

$$G_{Cdg} = [\ \{Cdg_0, Dom_0\}\ ,\ \{(,)\}\ ,\ \mathcal{P}_{Cdg}\ ,\ Cdg_0\ ]$$

$$\text{avec } \mathcal{P}_{Cdg} = \left|\ Cdg_0 \longrightarrow (Dom_0)\right.$$

Table d'analyse prédictive avec récupération sur erreur :

	(	)	$\bullet$	λ
Cdg_0	(Dom_0)	o_1	o_1	o_1

Avec les erreurs :
o_1 : La structure du codage est la liste non vide.

Annexe C

Descripteurs et Domaines de validité

C.1 Descripteurs

Dans cette annexe nous trouverons l'ensemble des descripteurs -sélecteurs- utilisés actuellement dans ExTI. Comme nous l'avons vu dans la seconde partie en 2.2.4 (page 82), ces ensembles sont à rapprocher des six *classes de caractéristiques* établies par J.-M. Inglebert ([Ing92]–annexe 2.3). Mais, alors que ces classes déterminaient une partition de l'ensemble des symboles terminaux selon les entités manipulées, les ensembles que nous proposons les regroupent selon leur rôle dans les descriptions. Ces ensembles peuvent donc être disjoints. Ce fait a été souhaité pour donner au système de planification une plus grande souplesse dans les recherches de solutions étant ainsi moins "bridé" par l'association opérateur-descripteur (exemple : *valeur_max*, *max_cc* et *max* sont réunis sous l'unique terminal *max*. Ainsi, des descripteurs ont disparu -parce que regroupés sous une même terminologie- tandis d'autre descripteurs ont étés implantés dans le système. Les noms des terminaux provenant des anciennes versions ont étés modifiés (raccourcis) afin d'obtenir des affichages plus réduits.

Le tableau ci-après reprend la totalité des terminaux classés par ordre alphabétique en indiquant :

- les ensembles d'appartenance (T_{LEO}).
- l'ancien nom dans le cas d'un descripteur inchangé.
- le regroupement de noms dans le cas de descripteurs disparus (avec la référence des classes d'appartenance).
- le rôle du descripteur.

Descripteurs	Ens.	Ancien nom	Regroup.	Rôle
λ				index nul
+	θ	+1		descripteur arithmétique de somme
+	ν			jonction de sélecteurs pour la sommation
-	θ	-1		descripteur arithmétique de différence
-	ν			jonction de sélecteurs pour la différence
*	θ			descripteur arithmétique de produit
/	θ			descripteur arithmétique de quotient
abs,sgn	$\check{\varsigma}$			sous-signaux (valeur absolue,signe) du signal de différence
angi	ς			angle de plus grande inertie d'une composante connexe
cercle	ϕ	cercle		descripteur de la figure disque
cgx,cgy	ς			coordonnées du centre géodésique d'une composante connexe
cix,ciy	ς			coordonnées du centre d'inertie d'une composante connexe
coul	ς			signal d'image couleur
cpt	ς			degré de compacité d'une composante connexe
cxpi,cypi	ς	coordonnee		coordonnées en x et y d'un *pi*xel
dans,hors	ρ	dans, hors		composante connexe de type trou (exactement inclue) -fond (exactement exclue)-
dist	ς	ordre		plus petite distance au bord de la forme
dx	ς	diamx		plus grand diamètre en x
dy	ς	diamy		plus grand diamètre en y
et	μ			jonction accumulative de concepts
etiq	ς	nom		étiquette (label)
etiq	$\hat{\zeta}$			index d'étiquetage
eul	ς		euler(3)	
			euler(5)	mesure du nombre d'Euler
fenetre	ϕ	fenetre		descripteur de la figure parallélogramme plein
fman	ς			signal du codage de Freeman
`gl`	$\hat{\zeta}$			index de globalité
inter,ninter	ρ	intersecte		composante connexe possédant au moins un -aucun- pixel en commun avec l'autre entité considérée
	ρ	nintersecte		
lgg	ς			longueur géodésique d'une composante connexe
max	η		max_cc(2)	
			max(2)	
			valeur_max(5)	modificateur de descripteur maximum
min	η		min_cc(2)	
			min(2)	
			valeur_min(5)	modificateur de descripteur minimum
moy	η	moyenne		modificateur de descripteur moyenne
mx	ς	mom_x		moment d'inertie d'ordre 1 en x
mxx	ς	mom_xx		moment d'inertie d'ordre 2 en x
mxy	ς	mom_xy		moment d'inertie d'ordre 1 en xy
my	ς	mom_y		covariance des moments d'inertie d'ordre 2
myy	ς	mom_yy		moment d'inertie d'ordre 2 en y

Descripteurs	Ens.	Ancien nom	Regroup.	Rôle
nbcc	ς	nb_zones		nombre de composantes connexes positives dans l'image
nbpx	ς	nb_pixels		nombre de pixels à valeur positive dans l'image
nbpx,dir	$\breve{\varsigma}$			sous-signaux (nombre de pixels, orientation) du signal du codage de Freeman
nbvsn	ς	nb_voisins		nombre de pixels non nuls dans le voisinage indiqué
ncercle	ϕ			descripteur du complémentaire de la figure disque
nfenetre	ϕ			descripteur du complémentaire de la figure parallélogramme plein
nivg	ς	nivg		niveau de gris
opp	η			modificateur de descripteur valeur opposée
ou	μ			jonction disjonctive de concepts
ox	ς	orig_x		abscisse du premier pixel de la composante connexe rencontrée lors d'un balayage gauche-droite, haut-bas
oy	ς	orig_y		ordonnée du premier pixel de la composante connexe rencontré lors d'un balayage gauche-droite, haut-bas
pix	η	nb_pixels		modificateur de descripteur du nombre de pixels d'une valeur de signal
prmt	ς	périmètre		nombre de pixels du périmètre extérieur de la composante connexe
rg,vt,bl	$\breve{\varsigma}$			sous-signaux (rouge,vert,bleu) du signal d'image couleur
ryg	ς			rayon géodésique d'une composante connexe
sobel	η	sobel		gradient d'un signal par sobel
srf	ς	surface		nombre de pixels de la composante connexe
vab	ς			valeur binaire
vois4	ζ			déstructuration par les 4-voisins
vois8	ζ			déstructuration par les 8-voisins
xinf,yinf	ς			coordonnées du pixel haut-gauche du rectangle englobant d'une composante connexe
xou	ν			jonction de sélecteurs pour une donnée multi-signal
xou	μ			jonction disjonctive exclusive de concepts
xsup,ysup	ς			coordonnées du pixel bas-droit du rectangle englobant d'une composante connexe

C.2 Domaines de validité

De tous ces ensembles de descripteurs, seuls les ensembles ς des sélecteurs et $\breve{\varsigma}$ des descripteurs de sous-descripteurs possèdent des domaines de validité. Il faut toutefois noter que les descripteurs modificatifs -ensemble η- peuvent modifier le domaine du descripteur sur lequel ils sont appliqués.

Les domaines de validité vont dépendre de paramètres témoignant des conditions opératoires (voisinages) ou des conditions d'acquisition des données. Actuellement, toute cette connaissance sur le contexte de l'application est caractérisée par les quatre paramètres suivants :

1. n: nombre de niveaux de gris (contexte d'acquisition)
2. x: nombre de colonnes de la donnée initiale (contexte d'acquisition)
3. y: nombre de lignes de la donnée initiale (contexte d'acquisition)
4. v: voisinage utilisé (conditions opératoires) $v \in \{4, 8\}$

Pour les domaines marqués d'une astérisque (*), une démonstration ou un complément d'information est donné à la fin de ce mémoire.

C.2.1 Domaines dépendants de n

$$\begin{array}{lcl}
\Delta_{nivg} & = & [\ 0\ ,\ n\ [\\
\Delta_{sobel} & \subseteq & [\ 0\ ,\ n\ [\\
\Delta_{rg} & = & [\ 0\ ,\ n\ [\\
\Delta_{vt} & = & [\ 0\ ,\ n\ [\\
\Delta_{bl} & = & [\ 0\ ,\ n\ [\\
\Delta_{coul} & = & [\ min\{\ \Delta_{rg}\ ,\ \Delta_{vt}\ ,\ \Delta_{bl}\ \}\ ,\ max\{\ \Delta_{rg}\ ,\ \Delta_{vt}\ ,\ \Delta_{bl}\ \}\ [\\
 & = & [\ 0\ ,\ n\ [
\end{array}$$

C.2.2 Domaines dépendants uniquement de v

$$\Delta_{nbvsn} = [\ 0\ ,\ 1+v\ [$$

C.2.3 Domaines dépendants de x

$$\begin{array}{lcl}
\Delta_{dx} & = & [\ 1\ ,\ 1+x\ [\\
\Delta_{ox} & = & [\ 0\ ,\ x\ [\\
\Delta_{cgx} & = & [\ 0\ ,\ x\ [\\
\Delta_{cix} & = & [\ 0\ ,\ x\ [\\
\Delta_{cxpi} & = & [\ 0\ ,\ x\ [\\
\Delta_{xinf} & = & [\ 0\ ,\ x\ [\\
\Delta_{xsup} & = & [\ 0\ ,\ x\ [\\
\Delta_{mx} & = & [\ 0\ ,\ x\ [\\
\Delta_{mxx} & = & \left[\ 0\ ,\ (x-1)\left[\dfrac{(x-3)}{12}+\dfrac{xy}{6(x+y-2)}\right]+\epsilon\ \right[\ \ (*)
\end{array}$$

C.2.4 Domaines dépendants de y

$$\begin{array}{lcl}
\Delta_{dy} & = & [\ 1\ ,\ 1+y\ [\\
\Delta_{oy} & = & [\ 0\ ,\ y\ [\\
\Delta_{cgy} & = & [\ 0\ ,\ y\ [\\
\Delta_{ciy} & = & [\ 0\ ,\ y\ [\\
\Delta_{cypi} & = & [\ 0\ ,\ y\ [\\
\Delta_{yinf} & = & [\ 0\ ,\ y\ [\\
\Delta_{ysup} & = & [\ 0\ ,\ y\ [\\
\Delta_{my} & = & [\ 0\ ,\ y\ [\\
\Delta_{myy} & = & \left[\ 0\ ,\ (y-1)\left[\dfrac{(y-3)}{12}+\dfrac{xy}{6(x+y-2)}\right]+\epsilon\ \right[\ \ (*)
\end{array}$$

C.2.5 Domaines dépendants de x et de y

Soit $\mathcal{Z}(i,j) = \left(i - 2\left\lceil \frac{i}{2} \right\rceil\right)\left(j - 2\left\lceil \frac{j}{2} \right\rceil\right)$

$$\begin{array}{lcl}
\Delta_{dist} & = & \left[\, 1 \ , \ \frac{1}{2}\, min\{\ x \ , \ y\ \} \,\right[\\
\Delta_{prmt} & \subset & [\, 1 \ , \ 1 + xy \,[\\
\Delta_{srf} & = & [\, 1 \ , \ 1 + xy \,[\\
\Delta_{lgg} & = & \left[\, 1 \ , 1 + \max\{(y-2)\left\lceil \frac{x}{2}\right\rceil + 2\left\lceil \frac{3x}{4}\right\rceil \ , \ (x-2)\left\lceil \frac{y}{2}\right\rceil + 2\left\lceil \frac{3y}{4}\right\rceil\} \,\right[\\
\Delta_{ryg} & = & \left[\, 1 \ , 1 + \frac{1}{2}\min\{(y-2)\left\lceil \frac{x}{2}\right\rceil + 2\left\lceil \frac{3x}{4}\right\rceil \ , \ (x-2)\left\lceil \frac{y}{2}\right\rceil + 2\left\lceil \frac{3y}{4}\right\rceil\} \,\right[\\
\Delta_{nbcc} & = & \left[\, 0 \ , \ 1 + \frac{1}{8}\left((8-v)xy + v\mathcal{Z}(x,y)\right) \,\right[\ (*) \\
\Delta_{etiq} & = & \left[\, 1 \ , \ 1 + \frac{1}{8}\left((8-v)xy + v\mathcal{Z}(x,y)\right) \,\right[\ (*) \\
\Delta_{eul} & = & \left[\, 1 - \frac{1}{8}\left((8-v)xy + v\mathcal{Z}(x-2,y-2)\right) \ , \ 1 + \frac{1}{8}\left((8-v)xy + v\mathcal{Z}(x,y)\right) \,\right[\ (*) \\
\Delta_{fman} & = & [\, min\{\ \Delta_{nbpx} \ , \ \Delta_{dir}\ \} \ , \ max\{\ \Delta_{nbpx} \ , \ \Delta_{dir}\ \} \,[\\
 & = & [\, 0 \ , \ max\{\ x \ , \ y\ \} \,[\\
\Delta_{nbpx} & = & [\, 0 \ , \ 1 + xy \,[\\
\Delta_{nbpx} & = & [\, 1 \ , \ max\{\ x \ , \ y\ \} \,[\ (freeman) \\
\Delta_{mxy} & : & \text{Ce domaine est obtenu en minimisant et maximisant l'équation :} \\
 & & \dfrac{1}{k}\displaystyle\sum_{i=1}^{k} (x_i - Mx)(y_i - My)
\end{array}$$

avec (Mx, My) le centre d'inertie de la composante de k pixels considérée. (*)

Nous avons pu montrer que les valeurs absolues des bornes de ce domaine Δ_{mxy} sont égales. Les expérimentations que nous avons pu faire au sujet de ce domaine nous ont amené à conclure, sans pouvoir toutefois le prouver, que ces bornes étaient obtenues pour des composantes connexes correspondant à des portions d'hyperboles reliées par une droite.

C.2.6 Domaines indépendants

$$\begin{array}{lcl}
\Delta_{sgn} & = & [\, 0 \ , \ 3 \,[\\
\Delta_{angi} & = & [\, -90 \ , \ 90 + \epsilon \,[\\
\Delta_{cpt} & = & [\, \epsilon \ , \ 1 + \epsilon \,[\ \subset [0 \ , \ 1.0001 \,[\\
\Delta_{vab} & = & [\, 0 \ , \ 2 \,[\\
\Delta_{dir} & = & [\, 0 \ , \ 8 \,[\ (freeman)
\end{array}$$

C.2.7 Domaines de sous-descripteurs

Alors que l'application d'un descripteur arithmétique appliqué avec une valeur numérique sur une donnée n'aura pour effet que de modifier l'intervalle du codage de la donnée mais non le domaine de validité du signal de celle-ci, les sous-descripteurs qui leur sont liés auront des domaines de validité dépendant des domaines des signaux ; soit $\Delta = [\delta_i, \delta_s[$ $(\delta_i \geq 0)$ un tel domaine. En fait, la création de ces sous-descripteurs permet de "normaliser" les valeurs associées aux résultats de ces opérations afin qu'elle soient toujours comprises dans l'intervalle $[0, \delta_s[$.

C.2.7.1 Sous-descripteurs de $diff$

Si nous définissons le domaine Δ_{diff}, dépendant du domaine $\Delta = [\delta_i, \delta_s[$ par :

$$\Delta_{diff} \ = \ [\ \delta_i - \delta_s \ , \ \delta_s - \delta_i \ [$$

les sous-signaux de $diff$ qui sont abs et sgn seront alors définis par
$\Delta_{abs} \;=\; [\,0\,,\,\delta_s - \delta_i\,[\; \subseteq [\,0\,,\,\delta_s\,[$
$\Delta_{sgn} \;=\; [\,0\,,\,3\,[$ *(déjà défini dans les domaines indépendants)*
avec 0 : valeurs négatives
avec 1 : valeurs nulles
avec 2 : valeurs positives

C.2.7.2 Sous-descripteurs de *somm*

Sur un schéma analogue du précédent, le domaine Δ_{somm}, dépendant du domaine Δ est défini par :
$\Delta_{somm} \;=\; [\,2\delta_i\,,\,2\delta_s\,[$
les sous-signaux de $somm$ qui sont val et mod seront alors définis par
$\Delta_{val} \;=\; [\,0\,,\,\delta_s\,[$ *correspondant au reste de la somme par* δ_s
$\Delta_{mod} \;=\; [\,0\,,\,2\,[$ *correspondant au quotient de la somme par* δ_s

Annexe D

Opérations de transformations

D.1 Transformations de la trame : τ^t

1. La transformation identique $\mathcal{I}d^t$

$$\mathcal{I}d^t = \left(\; \mathcal{I}d^\eta, Id^\mathcal{T} \; \right)$$

2. Les transformations par réduction de trame

   (a) Les transformations de spécialisation $\mathcal{S}p^t$

$$\mathcal{S}p^t = \left\{ \begin{array}{c} \mathcal{T}^- = \left\{ \begin{array}{c} \mathcal{T}^-_\mathcal{T} \\ \\ \mathcal{T}^-_\chi \end{array} \right\} = \left\{ \begin{array}{l} \left(\; \mathcal{I}d^\eta, f^\mathcal{T} \; \right) \text{par diminution fixée de la taille.} \\ \left(\; \mathcal{I}d^\eta, f^\chi \; \right) \text{par diminution calculée de la taille.} \end{array} \right\} \\ \eta^-_G = \left(\; f^{i,j<i}, f^G \; \right) \text{par sélection d'entité géométrique.} \end{array} \right\}$$

   (b) Les transformations de structuration $\mathcal{S}t^t$

$$\mathcal{S}t^t = \eta^-_{\mathcal{T}_\chi} = \left(\; f^{i,j<i}, f^{\mathcal{T}_\chi} \; \right)$$

3. Les transformations par augmentation de trame

   (a) Les transformations de généralisation $\mathcal{G}n^t$

$$\mathcal{G}n^t = \left\{ \begin{array}{c} \mathcal{T}^+ = \left\{ \begin{array}{c} \mathcal{T}^+_\mathcal{T} \\ \\ \mathcal{T}^+_\chi \end{array} \right\} = \left\{ \begin{array}{l} \left(\; \mathcal{I}d^\eta, f^\mathcal{T} \; \right) \text{par augmentation fixée de la taille.} \\ \left(\; \mathcal{I}d^\eta, f^\chi \; \right) \text{par augmentation calculée de la taille.} \end{array} \right\} \\ \eta^+_G = \left(\; f^{i,j>i}, f^G \; \right) \text{par expansion d'entité géométrique.} \end{array} \right\}$$

(b) Les transformations de déstructuration $\mathcal{D}s^t$

$$\mathcal{D}s^t = \eta^{\pm}_{\mathcal{T}_\chi} = \left(\ f^{i,j>i}, f^{\mathcal{T}_\chi}\ \right)$$

D.2 Transformations du cache : κ^t

1. Les transformations avec variation de la dimension

(a) Une transformation pseudo identique $\mathcal{I}d^\kappa$

$$\mathcal{I}d^\kappa \left| \begin{array}{l} K_\tau \longrightarrow K_{\tau'=\eta^{\pm}_{G}(\tau)} \\ \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \kappa' \left| \begin{array}{l} \tau' \\ f_{\delta'=\mathcal{S}p(\varepsilon,\delta)} \end{array} \right. \end{array} \right.$$

(b) La transformation de globalité $\kappa^t = 1^t$

$$1^t \left| \begin{array}{l} K_\tau \longrightarrow K_{\tau'=\eta^{\pm}_{\mathcal{T}_\chi}(\tau)} \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \kappa' \left| \begin{array}{l} \tau' \\ f_{\delta_o} \end{array} \right. = \{\pi' \in \tau'\} \end{array} \right.$$

2. Les transformations sans variation de la dimension

(a) La transformation identique $\mathcal{I}d^\delta$

$$\mathcal{I}d^\delta \left| \begin{array}{l} K_\tau \longrightarrow K_\tau \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \kappa \left| \begin{array}{l} \tau \\ f_{\delta=\mathcal{I}d^\nabla(\delta)} \end{array} \right. \end{array} \right.$$

(b) Deuxième forme de transformation identique $\mathcal{I}d^\chi$

$$\mathcal{I}d^\chi \left| \begin{array}{l} K_\tau \longrightarrow K_\tau \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \tau \\ f_{\delta'=\mathcal{S}p(\chi_1,\delta)\cup\mathcal{S}p(\chi_2,\delta)\cup\ldots\cup\mathcal{S}p(\chi_n,\delta)} \end{array} \right.$$

(c) La spécialisation de la description de la fonction cache κ^-_χ

$$\kappa^-_\chi \left| \begin{array}{l} K_\tau \longrightarrow K_\tau \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \kappa' \left| \begin{array}{l} \tau \\ f_{\delta'=\mathcal{S}p(\chi,\delta)} \end{array} \right. \end{array} \right.$$

(d) La généralisation de la description de la fonction cache κ^+_χ

$$\kappa^+_\chi \left| \begin{array}{l} K_\tau \longrightarrow K_\tau \\ \kappa \left| \begin{array}{l} \tau \\ f_\delta \end{array} \right. \longmapsto \kappa' \left| \begin{array}{l} \tau \\ f_{\delta'=\mathcal{G}n(\chi,\delta)} \end{array} \right. \end{array} \right.$$

D.3 Transformations de l'index : ι^t

1. Une Transformation identique

$$\mathcal{I}d^{\iota} \left| \begin{array}{l} I \longrightarrow I \\ \iota \longmapsto \iota \end{array} \right.$$

2. Transformation de spécialisation d'index

$$\iota_{\chi}^{-} \left| \begin{array}{l} I \longrightarrow I \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{S}p(\chi, \mathcal{C})\ [\ldots]) \end{array} \right.$$

3. Transformation de généralisation d'index

$$\iota_{\chi}^{+} \left| \begin{array}{l} I \longrightarrow I \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{G}n(\chi, \mathcal{C})\ [\ldots]) \end{array} \right.$$

4. Transformation de structuration d'index

$$\Gamma^{-} \left| \begin{array}{l} I \longrightarrow I_{\mathcal{G}l} \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{G}l\ [1]) \end{array} \right.$$

5. Transformation de déstructuration d'index

$$\Gamma^{+} \left| \begin{array}{l} I \longrightarrow I_{\mathcal{I}n} \\ \iota = (\mathcal{C}\ [\ldots]) \longmapsto \iota' = (\mathcal{I}n\ [\ldots]) \end{array} \right.$$

D.4 Transformations du signal : σ^t

1. Transformations indépendante d'une variation de dimension

   (a) Transformation identique

$$\mathcal{I}d^{\sigma} \left| \begin{array}{l} Sgn \longrightarrow Sgn \\ \sigma \longmapsto \sigma \end{array} \right.$$

   (b) Transformation par fixation du signal

$$\Sigma_{\varsigma} \left| \begin{array}{l} Sgn^{i} \longrightarrow Sgn^{i \pm k} \\ \sigma \longmapsto \varsigma \end{array} \right.$$

2. Transformations sans variation de dimension

   (a) Transformation de déstructuration du signal

$$\forall n \in \mathsf{IN},\ \Sigma^{+} \left| \begin{array}{l} Sgn^{n} \longrightarrow \{\sigma_i \in \Sigma^i / i < n, card(\sigma) \leq n\} \\ \sigma \longmapsto \{\sigma_i \in \Sigma^i\} \end{array} \right.$$

   (b) Transformation de structuration du signal

$$\Sigma^{-} \left| \begin{array}{l} \{\sigma_i \in \Sigma^i / i < n, card(\sigma) \leq n\} \longrightarrow Sgn^{n} \\ \{\sigma_i \in \Sigma^i\} \longmapsto \sigma \end{array} \right.$$

   (c) Transformation de généralisation du signal

$$\sigma_{\chi}^{+} \left| \begin{array}{l} Sgn^{n} \longrightarrow Sgn^{n} \\ \sigma \longmapsto \mathcal{G}n(\chi, \sigma) \end{array} \right.$$

(d) Transformation de spécialisation du signal

$$\sigma_\chi^- \left| \begin{array}{l} Sgn^n \longrightarrow Sgn^n \\ \sigma \longmapsto \mathcal{S}p(\chi, \sigma) \end{array} \right.$$

3. Transformations avec variation de dimension

(a) Transformation de spécialisation

$$\forall i \in \mathsf{IN},\ \sigma_\chi^- \left| \begin{array}{l} Sgn^i \longrightarrow Sgn^{i-k} \\ \sigma \longmapsto \mathcal{S}p(\chi, \sigma) \end{array} \right.$$

(b) Transformation de généralisation

$$\forall i \in \mathsf{IN},\ \sigma_\chi^+ \left| \begin{array}{l} Sgn^i \longrightarrow Sgn^{i+k} \\ \sigma \longmapsto \mathcal{G}n(\chi, \sigma) \end{array} \right.$$

D.5 Transformations du codage : ω^t

1. Transformation identique

$$\mathcal{I}d^\omega \left| \begin{array}{l} \mathcal{X} \longrightarrow \mathcal{X} \\ \omega \longmapsto \omega \end{array} \right.$$

2. Transformation par fixation du codage

$$\omega_\sigma \left| \begin{array}{l} \mathcal{X} \longrightarrow \mathcal{X} \\ \omega \longmapsto \Delta_\sigma \setminus 0 \end{array} \right.$$

3. Transformation de spécialisation

$$\omega_\chi^- \left| \begin{array}{l} \mathcal{X} \longrightarrow \mathcal{X} \\ \omega \longmapsto \mathcal{S}p(\chi, \omega) \end{array} \right.$$

4. Transformation de généralisation

$$\omega_\chi^+ \left| \begin{array}{l} \mathcal{X} \longrightarrow \mathcal{X} \\ \omega \longmapsto \mathcal{G}n(\chi, \omega) \end{array} \right.$$

Annexe E

Schéma symbolique des opérateurs

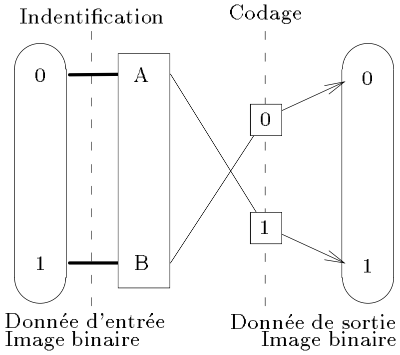
Pour chacun des opérateurs, il est nécessaire de bien définir les cinq éléments pré-cités : identification, mesure, partition, codage, regroupement.

• **Opérateur de complémentation binaire**
Opérateur `vnot`
Script KHOROS : `vnot -i X -o R -l 255`:

Donnée d'entrée :	image binaire :	`-i`: *input image (infile)*
Donnée de sortie :	image binaire :	`-o`: *resulting output image (outfile)*
Pamaètre fixé :	255 :	`-l`: *level to subtract input image from (float)*

Identification : deux sous-structures A et B correspondant aux valeurs binaires 0 et 1.
Mesure : aucune.
Partition : aucune.
Codage : valeur 1 pour la classe A et valeur 0 pour la classe B.
Regroupement : aucun.

$$Non(X) = \left(\begin{bmatrix} \tau^t = \mathcal{I}d^{\tau} \\ \kappa^t = \mathcal{I}d^{X.\overline{\delta}} \\ \iota^t = \Gamma^- \\ \sigma^t = \mathcal{I}d^{\sigma} \\ \omega^t = \mathcal{I}d^{\omega} \end{bmatrix}, \begin{bmatrix} \kappa^t = \mathcal{I}d^{X.\delta} \\ \iota^t = 0 \\ \sigma^t = \mathcal{I}d^{\sigma} \\ \omega^t = \mathcal{I}d^{\omega} \end{bmatrix} \right)$$

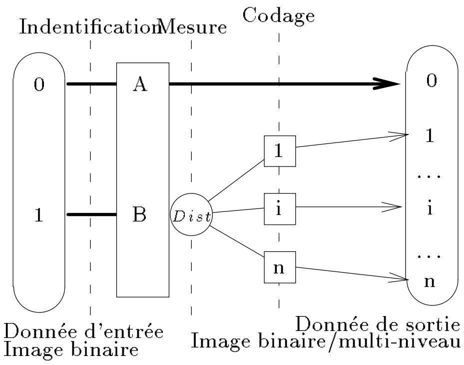


• **Opérateur de calcul des distances**
Opérateur `vdistance`
Script KHOROS : `vdistance -i X -o R`

Donnée d'entrée :	image binaire :	`-i` : *input image (infile)*
Donnée de sortie :	image binaire/multi-niveau :	`-o` : *output image (outfile)*

Identification : deux sous-structures A et B correspondant aux valeurs 0 et x>0.
Mesure : distance.
Partition : aucune.
Codage : valeur inchangée pour la classe A et valeur de la mesure pour la classe B.
Regroupement : aucun.

$$Dst(X) = \left(\begin{bmatrix} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \mathcal{I}d^\delta \\ \iota^t = \mathcal{I}d^\iota \\ \sigma^t = \Sigma_{dist} \\ \omega^t = \omega_{dist} \end{bmatrix}, \begin{bmatrix} \kappa^t = \mathcal{I}d^\delta \\ \iota^t = \mathcal{I}d^\iota \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{bmatrix} \right)$$



• **Opérateur de seuillage**

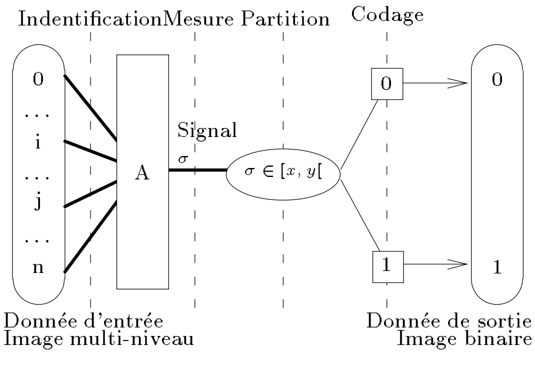
Opérateur `vthresh`

Script KHOROS : `vthresh -i X -l m -u (M-1) -o R`:

Donnée d'entrée :	image multi-niveaux :	`-i` : *input image (infile)*
Donnée de sortie :	image binaire :	`-o` : *output image (outfile)*
Paramètres de seuil :	entiers / réels :	`[-l]` `[-u]` : *lower/upper threshold value (float)*

Identification : une seule sous-structure A.
Mesure : aucune.
Partition : des valeurs du signal selon le paramètre intervalle fourni.
Codage : valeur 1 pour la sous-classe de A issue du résultat de la partition tel que "signal $\in$ intervalle" et valeur 0 pour l'autre sous-classe issue de la partition.
Regroupement : aucun.

$$Sll(X, m, M) = \left(\begin{bmatrix} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \kappa^-_{\{X.\sigma \in [m,M[\}} \\ \iota^t = \Gamma^- \\ \sigma^t = \Sigma_{vab} \\ \omega^t = \omega_{vab} \end{bmatrix}, \begin{bmatrix} \kappa^t = \kappa^+_{\{X.\sigma \in (\Delta_{X.\sigma} \setminus [m,M[)\}} \\ \iota^t = 0 \\ \sigma^t = \Sigma_{vab} \\ \omega^t = \Delta_{vab} \setminus \omega_{vab} \end{bmatrix} \right)$$



• **Opérateur de disjonction binaire**

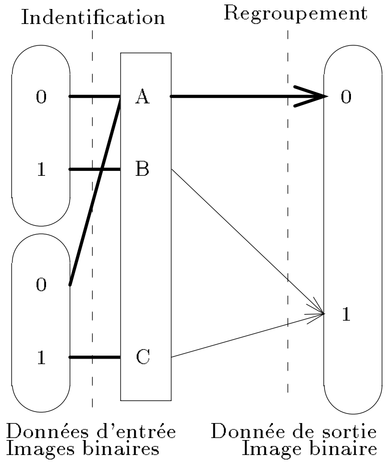
Opérateur `vor`

Script KHOROS : `vor -i1 X -i2 X' -o R`

Données d'entrée :	deux images binaires :	`-i1/-i2` : *first/second input image (infile)*
Donnée de sortie :	image binaire :	`-o` : *resulting output image (outfile)*

Identification : une sous-structure A correspondant aux valeurs 0 communes aux deux données, une sous-structure B correspondant aux valeurs 1 d'une donnée et une sous-structure C correspondant aux valeurs 1 de l'autre donnée.
Mesure : aucune.
Partition : aucune.
Codage : valeur inchangée pour les classes issues de A, B et C.
Regroupement : des classes issues de B et C.

$$Ou(X, X') = \left(\begin{bmatrix} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \kappa^+_{X'.\delta} \\ \iota^t = \Gamma^- \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{bmatrix}, \begin{bmatrix} \kappa^t = \kappa^-_{X'.\delta} \\ \iota^t = 0 \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{bmatrix} \right)$$

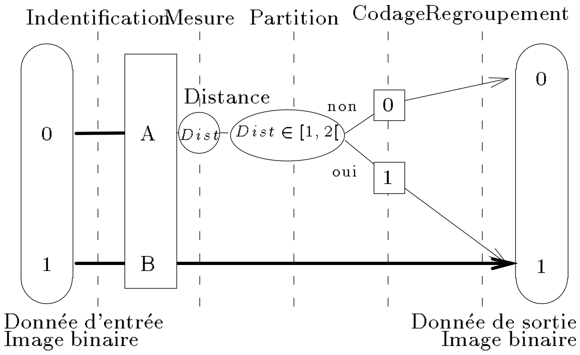


• **Opérateur de dilatation**
Opérateur `vmdilate`
Script KHOROS : `vmdilate -i X -k ERay1 -o R`

Données d'entrée :	image binaire :	`-i` : *input image (infile)*
Paramètre fixé :	Élément structurant :	`-k` : *input kernel (infile)*
Donnée de sortie :	image binaire :	`-o` : *output image (outfile)*

Identification : deux sous-structures A et B correspondant aux valeurs binaires 0 et 1.
Mesure : de distance sur la sous-structure A.
Partition : des valeurs de la mesure selon le paramètre intervalle $[1, 2[$ fourni.
Codage : valeur 1 pour la sous-classe de A issue d'une partition positive ($Dist \in [1, 2[$) et valeur 0 pour la sous-classe issue d'une partition négative.
Regroupement : des classes issues de B et de la sous-classe issue de la partition positive de A.

$$Dil(X, i) = \left(\begin{array}{l} \begin{bmatrix} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \mathcal{G}n(\mathcal{S}p((dist\ 1\ (i+1)), X.\overline{\delta}), X.\delta) \\ \iota^t = \mathcal{G}l \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{bmatrix}, \\ \begin{bmatrix} \kappa^t = \mathcal{S}p((dist\ (i+1)\ min(\mathcal{T}(X.\tau))/2), X.\overline{\delta}) \\ \iota^t = 0 \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{bmatrix} \end{array} \right)$$



• **Opérateur d'érosion**

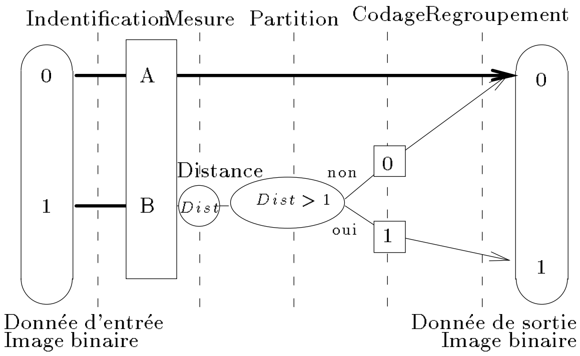
Opérateur `vmerode`

Script KHOROS : `vmerode -i X -k ERay1 -o R`

Données d'entrée :	image binaire :	`-i` : *input image (infile)*
Paramètre fixé :	Élément structurant :	`-k` : *input kernel (infile)*
Donnée de sortie :	image binaire :	`-o` : *output image (outfile)*

Identification : deux sous-structures A et B correspondant aux valeurs binaires 0 et 1.
Mesure : de distance sur la sous-structure B.
Partition : des valeurs de la mesure selon le paramètre intervalle $[2, max(\Delta_{Dist})[$ fourni.
Codage : valeur 1 pour la sous-classe de A issue d'une partition positive $(Dist \in [2, max(\Delta_{Dist}[)$ et valeur 0 pour la sous-classe issue d'une partition négative.
Regroupement : des classes issues de A et de la sous-classe issue de la partition négative de B.

$$Ero(X,i) = \left(\begin{array}{l} \left[\begin{array}{l} \tau^t = \mathcal{I}d^\tau \\ \kappa^t = \mathcal{S}p((dist\ (i+1)\ min(\mathcal{T}(X.\tau))/2), X.\overline{\delta}) \\ \iota^t = \mathcal{G}l \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{array} \right], \\ \left[\begin{array}{l} \kappa^t = \mathcal{G}n(\mathcal{S}p((dist\ 1\ (i+1)), X.\overline{\delta}), X.\delta) \\ \iota^t = 0 \\ \sigma^t = \mathcal{I}d^\sigma \\ \omega^t = \mathcal{I}d^\omega \end{array} \right] \end{array} \right)$$



Annexe F

Règles de manipulation de concepts

F.1 Notations

En plus des symboles employés dans le calcul des prédicats annotés de Michalski les notations suivantes sont utilisées pour la manipulation d'expressions.

$[signal(e)]$	sélecteur "représentationnel" des entités de type e, non sélectif.
S_e	sélecteur quelconque défini à partir d'un attribut qualifiant l'entité e.
V_e	vecteur de propriétés dont tous les attributs qualifient le même type d'entité e.
$S_e \& V_e$	spécialisation du vecteur de propriétés V_e le sélecteur S_e.
$V^* = V«*»$	vecteur de propriétés définissant un sous ensemble des entités décrites par l'expression $*$.
$«D_1 \vee D_2»$	disjonction quelconque de concepts décrits par les expressions D_1 et D_2.
$«D_1 \wedge D_2»$	intersection quelconque de concepts décrits par les expressions D_1 et D_2.

F.2 Règles

Conditions spatiales de structuration :

Règle R1

$$
\begin{array}{lcll}
[attribut(e)=l]\ \&\ V_e^* & = & [attribut(e)=l]«V_e^*» & \text{si } attribut(e) \text{ est un attribut variable.}\\
[attribut(e)=l]\ \&\ V_e^* & = & [attribut(e)=l]«V_e^*» & \text{si } attribut(e) \text{ est un attribut constant}\\
 & & & \text{et si } RE(attribut(e)) > DS(V_e).\\
[attribut(e)=l]\ \&\ V_e^* & = & V_e^{'*} & \text{si } attribut(e) \text{ est un attribut constant}\\
 & & & \text{et si } RE(attribut(e)) \leq DS(V_e).
\end{array}
$$

Règle R2

$$
[attribut(e)=l]\ \&\ V_{e'}^* \quad = \quad [attribut(e)=l]«V_{e'}^*» \quad \text{si } e\#e'.
$$

Conditions spatiales de distributivité :

Règle R3

Si $RE(attribut) \leq DSD(D_1 \vee D_2)$ alors

$[attribut(e)=l]«D_1 \vee D_2»\longleftrightarrow«[attribut(e)=l]\ \&\ D_1\ \vee\ [attribut(e)=l]\ \&\ D_2$

Conditions spatiales de factorisation :

Règle R4

Si $RE(attribut_1) \leq DS([attribut2(e')=l_2]«B»)$ alors

$[attribut_1(e)=l_1]«B» \wedge [attribut_2(e')=l_2]«B» \longleftrightarrow [attribut_1(e)=l_1]\ \&\ [attribut_2(e')=l_2]«B»$

Factorisation d'expression possédant un ordre :

Règle R5

Si l'expression $S_1\&S_2\&\ldots\&S_j\&S_{j+1}\&\ldots\&S_N«B»$ vérifie les conditions :

$$
\begin{array}{rcl}
RE(attribut_1) & \leq & DS(S_2\&\ldots\&S_j\&S_{j+1}\&\ldots\&S_N«B»)\\
RE(attribut_2) & \leq & DS(S_3\&\ldots\&S_j\&S_{j+1}\&\ldots\&S_N«B»)\\
\ldots & & \\
RE(attribut_j) & \leq & DS(S_{j+1}\&\ldots\&S_N«B»)\\
RE(attribut_{j+1}) & \leq & DS(S_{j+2}\&\ldots\&S_N«B»)\\
\ldots & & \\
RE(attribut_N) & \leq & DS(«B»)
\end{array}
$$

alors

s'il existe j position la plus à droite telle que

$$
\begin{array}{rcl}
RE(attribut_j) & \leq & DS(\&S_e\&S_{j+1}\&\ldots\&S_N«B»)\\
RE(attribut_e) & \leq & DS(S_{j+1}\&\ldots\&S_N«B»)
\end{array}
$$

alors

$S_e«B» \wedge S_1\&\ldots\&S_j\&S_{j+1}\&\ldots\&S_N«B» \longleftrightarrow S_1\&\ldots\&S_j\&S_e\&S_{j+1}\&\ldots\&S_N«B»$

Bibliographie

[Aho91] A. Aho, R. Sethi et J. Ullman. – *COMPILATEURS Principes, techniques et outils.* – IIA Inter-Editions, 1991. trad. P.Boullier, Ph.Deschamp, M.Jourdan, B.Lorho et M.Mazaud (INRIA).

[AID92] *Aïda version 1.65.* – I.L.O.G. 2, Av Galliéni BP85 – 94253 Gentilly (France), 1992.

[All83] J.F. Allen. – Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, vol. 26, n° 11, 1983, pp. 832–843.

[All84] J.F. Allen. – Toward a General Theory of Action and Time. *Artificial Intelligence*, vol. 23, n° 2, 1984, pp. 123–154.

[Bas96] Ch. Bastié et P. Régnier. – Intégration de la planification et du contrôle d'exécution en environnement dynamique: le système SPEEDY". *In : 10e RFIA, Rennes (France)*, janvier 1996, pp. 879–888.

[Bel85a] A. Belaïd. – Un langage de traitement d'images fondé sur des concepts de types abstraits. *Techniques et Sciences Informatiques*, vol. 4, n° 3, 1985, pp. 309–323.

[Bel85b] A. Belaïd et R. Mohr. – Système de génération de menus, application au traitement d'images. *In : 5e RFIA, Grenoble (France)*, novembre 1985, pp. 527–540.

[Bod95] R. Bodington. – A sotfware environment for the automatic configuration of inspection systems. *In : KBUP95, Sophia Antipolis (France)*, novembre 1995, pp. 100–108.

[Bok93] C Boksenbaum, B. Carbonneill, O. Maemmerlé et Th. Libourel. – Conceptual Graphs for Relational Databases. *In : Lectures Notes in Artificial Intelligence 699 – Conceptual Graphs for Knowledge Representation – ICCS'93 proceedings*, pp. 142–161. – S.K. Chang, Plenum Press, 1993.

[Cap92] O. Capdevielle, P. Dalle, P. Gurdjos et J.-M. Inglebert. – Langage visuel pour la formulation d'objectifs de traitements d'image. *In : IHM, Paris (France)*, 1992.

[Cap94] O. Capdevielle et P. Dalle. – Image Processing chain construction by interactive goal specification. *In : ICIP, Austin Texas (U.S.A.)*, 1994.

[Cap95] O. Capdevielle. – *Formulation d'objectifs de traitement d'images : Une exploitation interactive d'un système de planification automatique de chaînes d'opérateurs.* – Thèse de 3e cycle, Université P. Sabatier de Toulouse (France), 1995.

[Cha86] S.K. Chang. – Visual Languages and Iconic Languages. *In : Visual Languages*, pp. 1–7. – S.K. Chang, T. Ichikawa, P.A. Ligomenides, Plenum Press, 1986.

[Cha87a] S.K. Chang, Q.-Y. Shi and C.-W. Yan. – Iconic Indexing by 2-D Strings. *Pattern Analysis and Machine Intelligence*, vol. PAMI-9, n° 3, mai 1987, pp. 413–428.

[Cha87b] D. Chapman. – Planning for Conjunctive Goals. *Artificial Intelligence*, vol. 32, n° 3, 1987, pp. 333–377.

[Cle93] V. Clement et M. Thonnat. – A knowledge-based approach to integration of image processing procedures. *CVGIP : Image Understanding*, vol. 57, n° 2, mars 1993, pp. 166–183.

[Clo93] R. Clouard, Ch. Porquet et M Revenu. – Resolution of image processing problems by dynamic planning within framework of the blackboard model. *SPIE : Intelligent Robots and Computer Vision XII*, vol. 2056, 1993. – Boston.

[Clo94] R. Clouard. – *Raisonnement incrémental et opportuniste appliqué à la construction dynamique de plans de Traitement d'Images.* – Thèse de 3^{e} cycle, Université de Caen (France), 1994.

[Clo95] R. Clouard, M. Revenu, A. Elmoataz et Ch. Porquet. – Un Atelier Logiciel pour la conception d'applications de segmentation d'images. *In : Quinzième Colloque GRETSI, Juan-les-Pins (France)*, 1995, pp. 577–580.

[Dal91] P. Dalle et J.-M. Inglebert. – Modèle de description d'opérateurs de traitement d'images. *In : 8^{e} RFIA, Lyon-Villeurbanne (France)*, 1991, pp. 257–266.

[Dav93] J.L. Davidson. – Classification of Lattice Transformations in Image Processing. *CVGIP : Image Understanding*, vol. 57, n° 3, mai 1993, pp. 283–306.

[Dej96a] Ph. Dejean. – *LEO, Le Langage d'Expression d'Objectifs implanté dans ExTI.* – Rapport de recherche n° 96/02/R, Institut de Recherche en Informatique de Toulouse, 1996.

[Dej96b] Ph. Dejean. – Une modélisation du Traitement d'Images. *In : Support de cours du DESS IRR – Université P.Sabatier, Toulouse III*, 1996.

[Dej96c] Ph. Dejean et P. Dalle. – Comprendre et utiliser le traitement d'images. *In : CARI'96 (Colloque Africain sur la Recherche en Informatique), Libreville (Gabon)*, octobre 1996.

[Dej96d] Ph. Dejean et P. Dalle. – Image analysis operators as concept constructors. *In : IEEE Southwest Symposium on Image Analysis and Interpretation, San Antonio, Texas (USA)*, avril 1996, pp. 66–70.

[Dej96e] Ph. Dejean et P. Dalle. – Modèle symbolique de la donnée de traitement d'images. *In : 10^{e} RFIA, Rennes (France)*, 1996, pp. 746–755.

[Dej96f] Ph. Dejean et P. Dalle. – Un langage de description de concepts pour la formulation d'objectifs d'analyse. *In : Journées ORASIS'96 – Pôle vision du GDR-PRC 'CHM', Clermont-Ferrand (France)*, mai 1996, pp. 219–224.

[Den95] M. Denis, M.-R. Gonçalves et D. Memmi. – Mental scanning of visual images generated from descriptions : towards a model of image accuracy. *Neuropsychologia*, vol. 33, n° 11, 1995, pp. 1511–1530.

[Die86] T.G. Dietterich et R.S. Michalski. – Learning to predict sequences. *In : Machine learning : an artificial intelligent approach*, pp. 63–106. – Berlin : Springer-Verlag, 1986.

[Ebe91] D. Eberly et D. Wenzel. – Adaptation of Group Algebras to Signal and Image Processing. *CVGIP : Graphical Models and Image Processing*, vol. 53, n° 4, juillet 1991, pp. 340–348.

[Fic96] V. Ficet. – Construction interactive d'un modèle conceptuel d'application d'images. *In : Journées des jeunes chercheurs en IA, Nantes (France)*, 1996.

[Fik71] R.E. Fikes et N.Nilson. – STRIPS : A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, vol. 2, n° 3/4, 1971, pp. 189–208.

[Fli90] M. Flicker, M. Lavin et S.Das. – An object-oriented langage for image and vision executive (olive). *In : 10^{e} ICPR, Atlantic city, New Jersey*, 1990, pp. 561–571.

[Gaz79] G. Gazdar. – Phrase structure grammar. *In : The nature of syntactic representation*, pp. 131–186. – Jacobson & Pullum, Dorderecht : Reidel(82), 1979.

[Gin91] M.L. Ginsberg. – *Multivalued logics : A uniform approach to inference in artificial intelligence.* – Rapport technique, Department of Computer Science, Stanford University, 1991.

[Gon95] L. Gong et C.A.Kulikowski. – Composition of Image Analysis Processes Through Object-Centered Hierarchical planning. *Pattern Analysis and Machine Intelligence*, vol. PAMI-17, n° 10, octobre 1995, pp. 997–1009.

[Gre96] S. V. Grenier. – Conception d'Application de Traitement d'Images : Étude du Module de Contrôle d'Exécution, 1996.

[Haa90] V. Haarslev et R. Möller. – VIPEX : VIsual Programming of EXperimental systems. *In : Visual Languages and visual programming*, pp. 185–212. – S.K. Chang, Plenum Press, 1990.

[Has86] J. I. Hasegawa, H. Kubota et J.–I Toriwaki. – Automated Construction of Image Processing Procedures by sample-Figure Presentation. *In : 8e ICPR, Paris (France)*, 1986, pp. 586–588.

[Hen90] J. Hendler, A. Tate et M. Drummond. – AI Planning : Systems and techniques. *AI Magazine*, été 1990, pp. 61–77.

[Hor51] A. Horn. – On sentences which are true on direct unions of algebras. *Journal of Symbolic Logic*, vol. 16, n° 1, 1951, pp. 14–21.

[Hor87] G. Horrocks. – *Generative Grammar.* – Fifth impression 1993, Longman Inc. New York, 1987.

[IMA] IMASYS, 68 R. Carnot, 92150 Suresnes (France). – *OPTIMA 5.*

[Ing90] J. M. Inglebert et P. Dalle. – Traitement d'images : Acquisition des connaissances opératoires. *In : Journées Acquisition des connaissances, Lannion (France)*, 1990, pp. 131–142.

[Ing92] J. M. Inglebert. – *Planification automatique de chaînes de traitement d'images.* – Thèse de 3e cycle, Université P. Sabatier de Toulouse (France), 1992.

[Ing96] J. M. Inglebert. – Un langage de description de concepts de traitement d'images. *In : 10e RFIA, Rennes (France)*, 1996.

[IUE94] *Image Understanding Environment, Program – IUE Overview Document.* – ftp.aai.com (192.190.241.40), 1994.

[Jen94] R.E. Jennings. – *The Genealogy of Disjunction.* – Oxford University Press, Oxford, New York, 1994.

[Jos90] D. Joslin et J. Roach. – A Theorical Analysis of Conjunctive-Goal Problems. *Artificial Intelligence*, vol. 41, n° 1, 1990, pp. 189–208.

[Jun93] E. Jungert et S.K. Chang. – An Image Algebra for Pictorial Data Manipulation. *CVGIP : Image Understanding*, vol. 58, n° 2, septembre 1993, pp. 147–160.

[Kar94] P.D. Karp, J.D. Lowrance, T.M. Strat, et D.E. Wilkins. – The Grasper-CL Graph Management System. *LISP and Symbolic Computation: An International Journal*, vol. 7, 1994, pp. 245–273.

[KBU95] First international workshop on Knowledge-Based systems for the (re)Use of Program libraries. INRIA - Sophia Antipolis, novembre 1995.

[Kno94] C.A. Knoblock et Q. Yang. – Analyzing the Efficiency of Partial-Order Planning Algorithms. *In : Canadian Artificial Intelligence Conference*, 1994. – Banff, Alberta, (Canada), 1994.

[Kno95] C.A. Knoblock et Q. Yang. – Relating the Performance of Partial-Order Planning Algorithms to Domain Features. *SIGART Bulletin, Special Issue on Evaluating Plans, Planners, and Planning Agents*, vol. 6, n° 1, 1995.

[Lee91] S.Y. Lee et F.J. Hsu. – Picture algebra for spatial reasoning of iconic images represented in 2D C-string. *Pattern Recognition Letters*, vol. 12, juillet 1991, pp. 425–435.

[Lel91] *LE-LISP version 15.25.* – I.N.R.I.A. Domaine de Voluceau Rocquencourt – 78153 Le Chesnay (France), 1991.

[Lem95] J. Lemaire. – *Développement d'un système de reconnaissance de scènes à partir de descriptions imprécises.* – Rapport de recherche n° 200/95/RA, DCN/STSN-CERT/DERA, 1995.

[Lem96] J. Lemaire. – Utilisation de Descriptions Imprécises et Gestion de l'Incertitude dans un Système de Reconnaissance de Scènes. *In : Journées ORASIS'96 – Pôle vision du GDR-PRC 'CHM', Clermont-Ferrand (France)*, mai 1996, pp. 31–36.

[Lie86] C. E. Liedtke et M. Ender. – A knowledge based vision system for a automated adaptation to new scene contents. *In : 8e ICPR, Paris (France)*, 1986, pp. 795–797.

[Lie92] C. E. Liedtke et A. Blömer. – Architecture of the knowledge based configuration system for image analysis CONNY. *In : 11e ICPR, La Hague (Pays-bas)*, 1992, pp. 375–378.

[Lig94] G. Ligozat. – *Représentation des connaissances et linguistique.* – Armand Collin, 1994.

[Mat86] T. Matsuyama. – LLVE : An expert system for top-down image segmentation. *J. IPS Japan*, vol. 27, 1986, pp. 191–204. – [japonais].

[Mat89] T. Matsuyama. – Expert systems for image processing : knowledge-based composition of image analysis processes. *CVGIP*, vol. 48, n° 1, octobre 1989, pp. 22–49.

[Mat90] T. Matsuyama et V. Hwang. – *SIGMA : a knowledge-based aerial image understanding system.* – Plenum Press, 1990.

[McC56] E.J. McCluskey. – Minimization of Boolean Functions. *Bell Systems Technical Journal*, vol. 35, 1956, pp. 1417–1444.

[Mey92] B. Meyer. – *Introduction à la théorie des langages de programmation.* – IIA InterEditions, 1992. trad. J.-F. Picardat.

[Mic83] R.S. Michalski. – A theory and methodology of inductive learning. *Artificial Intelligence*, vol. 20, n° 2, 1983, pp. 111–161.

[Mic84] R.S. Michalski, J.G. Carbonell et T.M. Mitchell. – A theory and methodology of inductive learning. *In : Machine learning : an artificial intelligent approach*, pp. 83–134. – Berlin : Springer-Verlag, 1984.

[Miy95] J. Miyao et S.K. Chang. – A Framework of a Visual Language with Dynamic Specification. *In : IEEE Symposium on Visual Languages, Darmstadt (Allemagne)*, 1995, pp. 140–147.

[Moi95a] S. Moisan, F. Florimont, M. Thonnat, V. Clement et R. Vincent. – *Document de l'Interface du Logiciel OCAPI Version 2.0.* – Rapport de recherche n° 0186, INRIA, novembre 1995.

[Moi95b] S. Moisan, R. Vincent, J.van den Elst et F. van Harmelen. – Towards an intelligent failure handling mechanisms in program supervision. *In : KBUP95, Sophia Antipolis (France)*, novembre 1995, pp. 110–118.

[Moi95c] S. Moisan, R. Vincent, M. Thonnat, V. Clement et J. van den Elst. – *Manuel de référence du Logiciel OCAPI Version 2.0.* – Rapport de recherche n° 0183, INRIA, novembre 1995.

[Nil80] N.J. Nilson. – *Principles of Artificial Intelligence.* – Tioga Publishing Company PaloAlto, California, 1980.

[Noe92] Noesis, 13 Av M Saulnier, 78140 Vélisy. – *Visilog 4 Reference Guide*, 1992.

[Per80] F.C.N. Pereira and D.H.D. Warren. – Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence*, vol. 13, 1980, pp. 231–278.

[Pol94] C. Pollard et I.A. Sag. – *Head-Driven Phrase Structure Grammar.* – The University of Chicago Press, 1994.

[Qui52] W.V. Quine. – The Problem of Simplifying Truth Functions. *American Mathematical Monthly*, vol. 59, 1952, pp. 521–531.

[Rit90] G.X. Ritter, J.N. Wilson et J.L. Davidson. – Image Algebra : An Overview. *CVGIP*, vol. 49, n° 3, septembre 1990, pp. 297–331.

[Sab88] G. Sabah. – *L'intelligence artificielle et le langage, Vol. 1 : Représentation des connaissances.* – Hermes, 1988.

[Sac75] E.D. Sacerdoti. – The non linear nature of plans. *In : IJCAI*, 1975.

[Sch89] M. Schmitt. – Mathematical Morphology and Artificial Intelligence : An automatic programming system. *Signal Processing*, vol. 16, n° 4, 1989, pp. 389–401.

[Sch92] M. Schmitt. – *La morphologie mathématique en analyse d'images.* – Rapport technique n° 4, Thomson-CSF, décembre 1992.

[Ser86] J. Serra. – Introduction to Mathematical Morphology. *CVGIP*, vol. 35, 1986, pp. 283–305.

[Ser89] V. Serfaty–Dutron, B. Hoeltzener–Douarin et B. Zavidovique. – Exemple de conception mécanique d'Opérateurs de T.I. *In : 7e RFIA, Paris (France)*, 1989, pp. 75–88.

[Sow84] J.F. Sowa. – *Conceptual Structures, Information Processing in Mind and Machine.* – Addison-Wesley Publishing Company, Inc., 1984.

[The92] C. Theot E. Gallier et D. Mischler. – *VSDE un environnement de développement automatisé de systèmes de vision.* – Rapport technique n° 4, Thomson-CSF, décembre 1992.

[Tho93] M. Thonnat, V. Clement et J. van den Elst. – *Supervision of Perception Tasks for Autonomous Systems : the OCAPI Approach.* – Rapport de recherche n° 2000, INRIA, juin 1993.

[Uni92] University of New Mexico. – *Khoros Reference Manual*, 1992.

[van92] R. van den Boomgaard et R. van Balen. – Methods for Fast Morphological Image Transforms Using Bitmapped Binary Images. *CVGIP : Graphical Models and Image Processing*, vol. 54, n° 3, mai 1992, pp. 252–258.

[Vig92] J.R. Vigouroux et D. Mischler. – *Allegory. Un environnement de programmation pour le traitement d'images.* – Rapport technique n° 4, Thomson-CSF, décembre 1992.

[Voy87] R. Voyer. – *Moteurs de Systèmes Experts.* – Eyrolles, 1987.

[Wil95] D.E. Wilkins et K.L. Myers. – A common knowledge representation for plan generation and reactive execution. *In : Journal of Logic and Computation*, décembre 1995, pp. 731–761.

[Win77] P.H. Winston. – *Artificial Intelligence.* – Addison-Wesley Publishing Compagny, 1977.

[Yan90] Q. Yang et J.D. Tenenberg. – ABTWEAK : abstracting a nonlinear, least commitment planner. *In : IJCAI*, 1990.

[Zhu86] X. Zhuang, R.M. Haralick. – Morphological Structuring Element Decomposition. *CVGIP*, vol. 35, n° 3, septembre 1986, pp. 370–382.

Table des matières

Table des figures

Notes

• Extrema des descripteurs *nbcc*, *etiq* et *euler* :

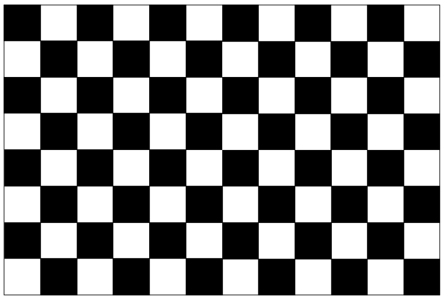
Le nombre maximum des composantes connexes que l'on peut constituer sur une image de taille $X \times Y$ est identique au nombre de composantes étiquetées. De même la valeur maximale du nombre d'euler que l'on peut obtenir est atteinte lorsque le nombre de composantes connexes est maximal ($max(nbcc)$)) et le nombre de trous minimal (0). De ce fait :

$$max(nbcc) = max(etiq) = max(euler)$$

La configuration permettant d'obtenir ces valeurs maximales est donnée, dans le cas d'un voisinage de 4-voisins, par un damier et par analogie, dans le cas des 8-voisins par une donnée du type de celle rapportée dans dans la figure suivante.

x=12

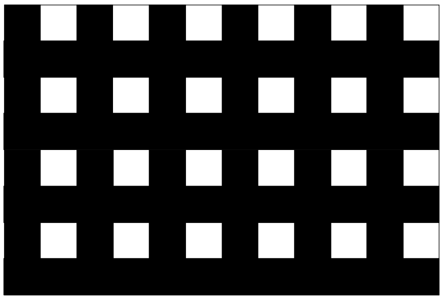
y=8



4-voisins

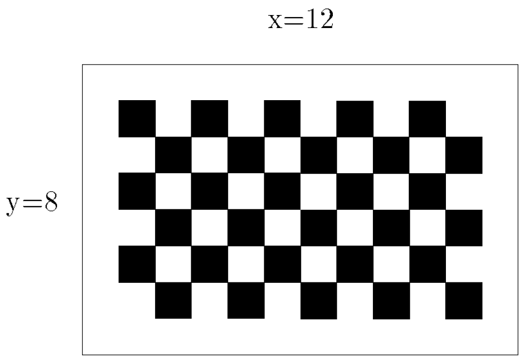
x=12

y=8

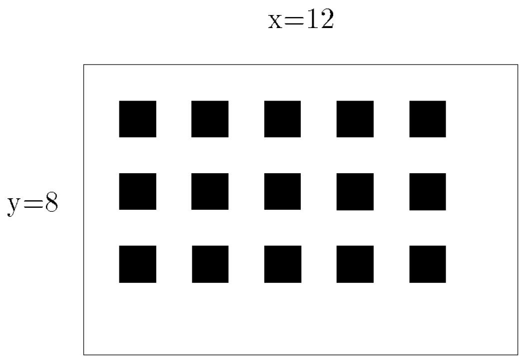


8-voisins

La valeur minimale de ce même nombre d'euler est atteinte lorsque le nombre de composantes connexes étiquetées est minimal (1) et le nombre de trous maximal. Cette configuration correspond aux composantes de la figure suivante.



4-voisins



8-voisins

• Extrema des moments d'inertie d'ordre 2 :

Le problème de la minimisation et la maximisation des moments d'inertie d'ordre 2 consiste à déterminer les composantes connexes pour lesquelles la mesure des moments sera extrémale.

○ Moment d'inertie d'ordre 2 en x ou y

Formules du moment d'ordre 1

$$Mx = \frac{1}{k}\sum_{n=1}^{k} x_n \qquad\qquad My = \frac{1}{k}\sum_{n=1}^{k} y_n$$

Formules du moment d'ordre 2

$$Mxx = \frac{1}{k}\sum_{n=1}^{k}(x_n - Mx)^2 \qquad\qquad Myy = \frac{1}{k}\sum_{n=1}^{k}(y_n - My)^2$$

Minimum

Le minimum du moment d'inertie -toujours nul puisqu'il s'agit de somme de carrés- est obtenu lorsque le terme $(x_n - Mx)$ (respectivement $(y_n - My)$) est égal à 0.

$$Min(Mxx) = Min(Myy) = 0$$

Maximum (axe des abscisses)

Analysons la formule. Il s'agit de la somme, pour tous les pixels, de la distance projetée en x élevée au carré -donc un terme positif- du pixel considéré de la composante à son centre d'inertie Mx divisé par le nombre de ces points. Le maximum de ce terme est atteint pour les points les plus distants du centre ; c'est à dire les bords gauches et droits, mais ceux-ci déterminent alors deux composantes connexes. L'ensemble des points devant être connexe, c'est à dire ne former qu'une seule composante, celle-ci sera donc constituée par le cadre intérieur de l'image (fig.F.1).
Remarque : Nous retrouvons là un fait connu en mécanique qui veut que le moment d'inertie d'un anneau soit plus élevé que celui du disque de même rayon.

Le moment d'inertie d'ordre 1 est alors :

$$Mx = \frac{1}{k}\sum_{n}^{k} x_n \text{ avec } k \text{ nombre pixels du cadre : } k = 2(x + y - 2)$$

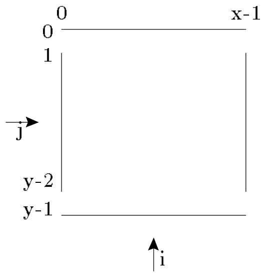


FIG. F.1 - *Composante considérée pour max(mxx)*

$$\begin{aligned}
&= \frac{1}{2(x+y-2)}\left[\left(\underbrace{2\sum_{i=0}^{x-1} i}_{bords\ sup.\ et\ inf.}\right) + \underbrace{\sum_{j=1}^{y-2} 0}_{bord\ gauche\ :\ x_n=0} + \underbrace{\sum_{j=1}^{y-2}(x-1)}_{bord\ droit\ :\ x_n=(x-1)}\right] \\
&= \frac{1}{2(x+y-2)}\left(2\frac{x(x-1)}{2} + 0 + (y-2)(x-1)\right) \\
&= \frac{1}{2(x+y-2)}\left((x-1)(x+y-2)\right) \\
Mx &= \frac{x-1}{2}
\end{aligned}$$

Le moment d'inertie d'ordre 2 :

$$\begin{aligned}
Mxx &= \frac{1}{k}\sum_{n=1}^{k}(x_n - Mx)^2 \\
&= \frac{1}{k}\left(\sum_{n=1}^{k} x_n^2 + \sum_{n=1}^{k} Mx^2 + \sum_{n=1}^{k}(-2)Mx.x_n\right) \\
&= \frac{1}{k}\left[\left(2\sum_{i=0}^{x-1} i^2\right) + \sum_{j=1}^{y-2} 0^2 + \sum_{j=1}^{y-2}(x-1)^2 + \right. \\
&\quad \sum_{n=1}^{k}\left(\frac{x-1}{2}\right)^2 - \\
&\quad \left. 2Mx\left(\left(2\sum_{i=0}^{x-1} i\right) + \sum_{j=1}^{y-2} 0 + \sum_{j=1}^{y-2}(x-1)\right)\right] \\
&= \frac{1}{2(x+y-2)}\left[\ 2\left(\frac{(x-1)x(2x-1)}{6}\right) + 0 + (y-2)(x-1)^2 + \right. \\
&\quad k\left(\frac{x-1}{2}\right)^2 - \\
&\quad \left. 2\left(\frac{x-1}{2}\right)\left(2\left(\frac{x(x-1)}{2}\right) + 0 + (y-2)(x-1)\right)\right] \\
&= \frac{(x-1)\left(x(2x-1) + 3(y-2)(x-1)\right)}{6(x+y-2)} + \\
&\quad \frac{(x+y-2)(x-1)^2}{4(x+y-2)} -
\end{aligned}$$

$$
\begin{aligned}
& \frac{(x-1)\left(x(x-1)+(y-2)(x-1)\right)}{2(x+y-1)} \\
= & \frac{(x-1)\left(2x(2x-1)+6(y-2)(x-1)+3(x+y-2)(x-1)-6x(x-1)-6(y-2)(x-1)\right)}{12(x+y-2)} \\
= & \frac{(x-1)\left(2x(2x-1)+3(x+y-2)(x-1)-6x(x-1)\right)}{12(x+y-2)} \\
= & \frac{(x-1)\left((x-3)(x+y-2)+2xy\right)}{12(x+y-2)} \\
Mxx = & (x-1)\left[\frac{(x-3)}{12}+\frac{xy}{6(x+y-2)}\right]
\end{aligned}
$$

∘ Moment d'inertie d'ordre 2 en xy

Formules du moment d'ordre 2

$$Mxy = \frac{1}{k}\sum_{n=1}^{k}(x_n - Mx)(y_n - My)$$

La détermination des valeurs extrémales de ces moments est complexes étant donnée la présence des deux coordonnées des pixels des composantes. Nous pouvons toutefois avancer quelques éléments de réflexions autour de l'équation, éléments confirmés par le calcul expérimental. Cette équation correspond à la somme des produits des distances des coordonnées des points par rapport au centre d'inertie de la composante. Les valeurs minimales et maximales seront donc égales en valeur absolue.

Étudions le maximum de ces valeurs. Il est atteint pour des valeurs de produit, toutes positives. Rapportons à présent la somme Mxy au centre d'inertie de la composante connexe $\mathcal{C}$ étudiée. Nous obtenons l'équation :

$$Mxy = \frac{1}{k}\sum_{n=1}^{k} x_n . y_n$$

La composante $\mathcal{C}$ est donc constituée de pixels appartenant aux zones $\oplus$ (fig.F.2) et telles que si le pixel $P_{Mx,My}(x_i, y_i)$ appartient à $\mathcal{C}$, alors $\overline{P}_{Mx,My}(-y_i, -x_i)$ appartient aussi à $\mathcal{C}$. Ceci permet de garantir l'inertie autour du point (Mx, My). Il faut noter qu'une rotation de -90^o ou 90^o de la composante autour de ce point fournira la valeur minimale de Mxy.

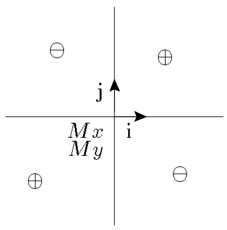


FIG. F.2 - *Quadrants autour de Mxy*

Caractéristiques de $\mathcal{C}$

Soit $\mathcal{C}_k$ une composante à k pixels et $Mxy(\mathcal{C}_k)$ sa mesure du moment d'ordre 2 en xy. L'ajout de deux pixels $p(x,y)$ et $\overline{p}(-y,-x)$ à cette composante va maximiser $Mxy(\mathcal{C}_k)$ si et seulement si :

$$
\begin{aligned}
& \frac{1}{k}\sum_{n=1}^{k} x_n . y_n < \frac{1}{k+2}\left(\sum_{n=1}^{k} x_n . y_n + 2xy\right) \\
\Longleftrightarrow \quad & 2xy > \sum_{n=1}^{k} x_n . y_n \left(\frac{2}{k(k+2)}\right) \\
\Longleftrightarrow \quad & xy > \frac{1}{(k+2)} Mxy(\mathcal{C}_k)
\end{aligned}
$$

Cette inégalité montre que la composante $\mathcal{C}$ fournissant le maximum des valeurs de Mxy est formée de deux portions d'hyperboles. Or, pour une raison de connexité de $\mathcal{C}$, il nous faut rajouter la plus petite composante connexe reliant des deux portions d'hyperbole. Cette plus petite composante est la droite $y = (1+\epsilon)x$ reliant ces entités et telle que $\mid \epsilon \mid$ soit minimal. La famille de composantes est donc donnée par la figure F.3.

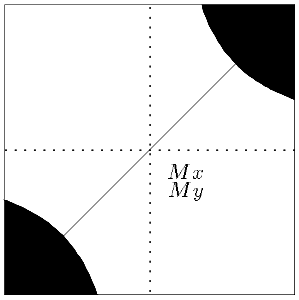


FIG. F.3 - *Famille de composantes pour Mxy*

Toulouse, le 18 décembre 1996