

# Automatisation du raisonnement Mathématiques et Démonstration automatique de théorèmes

*Dominique Pastre*

Université René Descartes - Paris 5  
UFR de mathématiques et informatique  
Crip5  
Cours DEA MIASH - 2002

Chapitre 2 : Méthodes graphiques

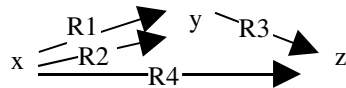
# Méthodes graphiques

Dominique Pastre - DEA MISAH - Université René Descartes (Paris 5) - 2002

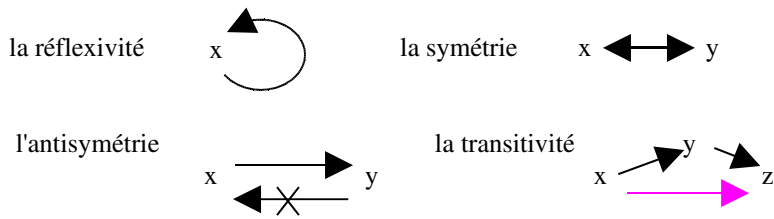
Des méthodes graphiques ont été utilisées dans divers démonstrateurs ou résolveurs de problèmes. On en verra des exemples dans différents domaines, les limites de telles représentations, mais aussi leur efficacité liée à la réalisation informatique qui a pu en être faite. Si certaines réalisations ont surtout un intérêt historique, les méthodes du programme de Mérialdo décrites en section 3 sont très efficaces et permettent en particulier de découvrir des objets intermédiaires utiles, ce qui évite de devoir donner certains lemmes au programme.

## 1 Relations binaires

a) Si on a deux relations binaires R1 et R2, les propriétés  $x R1 y$ ,  $x R2 y$ ,  $y R3 z$  et  $x R4 z$  peuvent être représentées (et mémorisées) sur le graphe étiqueté suivant :



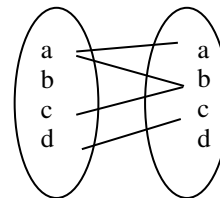
On peut représenter facilement certaines propriétés :



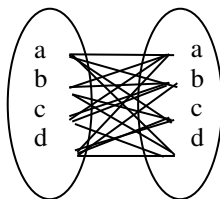
### 1.1 Utilisation de graphes pour la résolution automatique de problèmes combinatoires

Le système ALICE de Laurière [LAU 76, 78] est un système général de résolution de problèmes qui utilise, comme représentation interne de certaines contraintes, la représentation sagittale des relations (qu'il appelle fonctions, éventuellement multivoques), c'est-à-dire un graphe réparti. On a un ensemble E (fini) de sommets de départ et un ensemble F (fini) de sommets images et des arcs de certains sommets de E vers certains sommets de F.

Voici par exemple le diagramme sagittal de la relation R définie sur  $\{a,b,c,d\} \times \{a,b,c,d\}$  par  $R(a,a)$ ,  $R(a,b)$ ,  $R(c,b)$  et  $R(d,c)$



Sans indication particulière, tous les sommets de E sont reliés à tous les sommets de F.



Lorsque des contraintes sont imposées, elles sont manipulées sur le graphe. Par exemple si f est une fonction et si  $f(i)=j$  on supprime tous les arcs issus de i qui vont vers un sommet autre que j. Si f est injective on supprime également tous les arcs arrivant en j issus de sommets autres que i. Si f est

surjective et si un sommet de F n'a pas d'antécédent, la branche de l'arbre de recherche où l'on se trouve est une impasse. Plus généralement on traite aussi les cas où des nombres minimum et maximum sont imposés pour les antécédents ou les images.

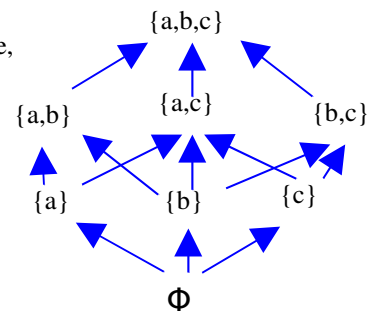
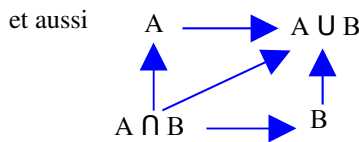
De plus un hypergraphe est construit sur les sommets de E pour traduire des disjonctions, c'est-à-dire des contraintes établissant qu'un sous-ensemble de E, appelé clique de disjonction, doit avoir, dans son ensemble, des images toutes différentes, par exemple pour des problèmes de coloriage ou de planning. Une procédure spéciale cherche l'ensemble de disjonctions le plus économique possible pour recouvrir E par construction de familles maximales et traiter plus rapidement les contraintes.

L'intérêt de cette représentation graphique réside dans son efficacité.

## 1.2 Théorie des ensembles, inclusion et appartenance

1.2.1 L'inclusion est une relation d'ordre partiel formant un treillis.

Si la flèche bleue représente l'inclusion, on a le graphe ci-contre, complété par sa fermeture,



Cette représentation des inclusions est utilisée pour travailler en *topologie générale* par Ballantyne dans [BAL 73] "Graphing methods for topological proofs"

L'ensemble vide est relié à tous les ensembles et tous les ensembles sont reliés à l'espace entier. Les ensembles qui interviennent dans un problème sont créés sur le graphe s'il n'existe pas déjà un ensemble ayant les propriétés requises, et dans tous les cas les liens sont ajoutés. Le programme manipule les opérations ensemblistes élémentaires (union, intersection, complémentaire, intérieur, adhérence, frontière) et les adhérences et frontières. Il ne reçoit pas leur définition formelle mais leurs propriétés, par exemple

- $A \subset A \cup B$
- $\overline{A \cup B} = \overline{A} \cap \overline{B}$  où  $\overline{A}$  désigne l'adhérence de A
- $\overline{A \cap B} \subset \overline{A} \cap \overline{B}$
- $A \subset C \wedge B \subset C \Rightarrow A \cup B \subset C$
- ...

Ces propriétés sont appliquées dès qu'une union, une intersection, ou une adhérence sont introduites. De plus la transitivité de l'inclusion

- $A \subset B \wedge B \subset C \Rightarrow A \subset C$

est appliquée directement sur le graphe (clôture transitive).

L'intérêt de cette représentation repose sur l'efficacité en exécution puisqu'on a immédiatement accès à tous les ensembles connus sous-ensembles ou sur-ensembles d'un ensemble.

Exemple de théorème démontré :

- $Fr(\overline{A}) \subset Fr(A)$  où  $Fr(A)$  désigne la frontière de A c'est-à-dire  $\overline{A} \cap \overline{CA}$ , et  $CA$  le complémentaire de A.

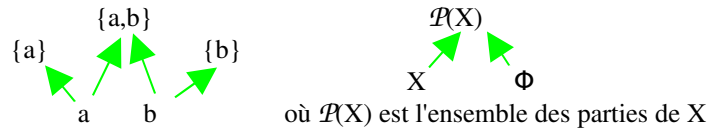
Cette représentation graphique est aussi utilisée pour la recherche de d'exemples et de contre-exemples [BAL 73, 75, 82].

Par exemple, on a  $\overline{A \cap B} \subset \overline{A} \cap \overline{B}$  mais on n'a pas  $\overline{A \cap B} = \overline{A} \cap \overline{B}$  et le programme le démontre en en trouvant un contre-exemple.

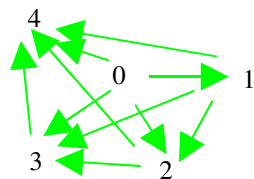
1.2.2 L'appartenance peut aussi être représentée graphiquement.

$a \in b$  est représenté par  $a \rightarrow b$

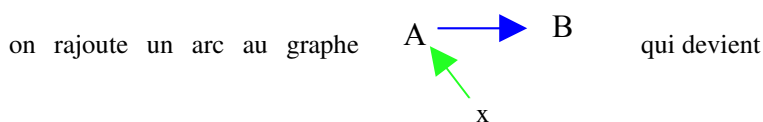
On a



Avec les ordinaux, définis comme  $0 = \Phi$ ,  $1 = \{\Phi\}$ ,  $2 = \{0,1\}$ , ... ,  $n+1 = n \cup \{n\}$ , on a

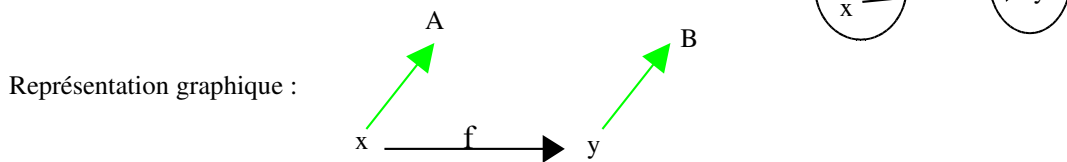


Pour traduire la propriété  
si  $x \in A$  et  $A \subset B$  alors  $x \in B$



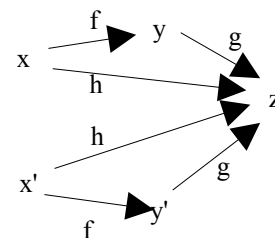
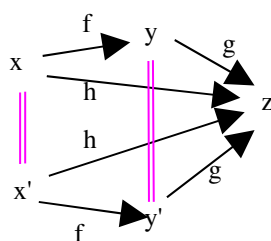
1.2.3 Applications

Soit  $f$  est une application de  $A$  dans  $B$ ,  $x \in A$  et  $y = f(x) \in B$ .



Pour montrer que si  $f$  et  $g$  sont bijectives, alors  $h = gof$  est aussi bijective ( $f$  de  $A$  dans  $B$ ,  $g$  de  $B$  dans  $C$ ), on montre d'abord que  $h$  est injective puis que  $h$  est surjective.

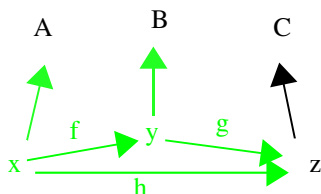
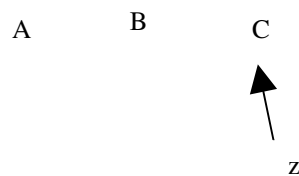
$h$  injective :  
soient  $x$  et  $x'$  dans  $A$  tels que  $h(x) = h(x') = z$ . Est-ce que  $x=x'$  ?



De  $g$  injective, on déduit que  $y=y'$ , puis de  $f$  injective que  $x=x'$ .  
(Les appartenances à  $A$ ,  $B$  et  $C$  n'ont pas été dessinées.)

$h$  surjective :

soit  $z \in C$ . Est-ce qu'il existe  $x \in A$  tel que  $h(x) = z$  ?



$g$  surjective donne  $y$  dans  $B$ , puis  $f$  surjective donne  $x$  dans  $A$ .  
 $z = g(f(x)) = h(x)$ .

1.2.4 DATTE [PAS 78] est un programme qui travaille en théorie des ensembles, naïve et axiomatique. Il utilise des méthodes de déduction naturelle, généralisant celles de [BLE 71]. Il comporte des règles, simples réécritures ou actions conditionnelles mais aussi des parties programmées. Ainsi toutes les hypothèses qui sont des relations binaires (beaucoup plus nombreuses que les autres hypothèses dans le domaine traité) sont mémorisées par un graphe et sont traitées différemment des autres hypothèses. Tous les objets concernés sont les sommets du graphe (ce qui donne en plus l'ensemble des objets mathématiques qui ont été introduits). Si deux objets  $x$  et  $y$  vérifient la relation  $R(x,y)$ , ceci est traduit par un arc étiqueté  $R$ . Ceci concerne les relation d'appartenance  $x \in y$ , d'inclusion  $x \subset y$ , d'égalité  $x = y$ , d'inégalité  $x < y$  ou  $x \leq y$ , la relation entre antécédent  $x$  et image  $y = f(x)$ , des relations quelconques pour les théorèmes sur les relations d'ordre ou d'équivalence, mais aussi les relations de non-égalité  $x \neq y$  et de non-appartenance  $x \notin y$ . En effet, ces dernières relations ont été jugées suffisamment importantes pour être considérées non plus comme des négations qui seraient plus difficiles à traiter mais comme de nouvelles relations, imitant ainsi le mathématicien qui utilise fréquemment les symboles  $\neq$  et  $\notin$ . Deux objets peuvent bien sûr être reliés par plusieurs arcs, par exemple  $y = f(x)$ ,  $x = f^{-1}(y)$ ,  $x \neq y$ ,  $x < y$ ,  $x \leq y$ . La représentation interne d'un théorème comporte trois parties : la liste des hypothèses qui ne sont pas des relations binaires, la conclusion à démontrer et un pointeur vers le graphe des relations binaires. Les règles comportent également une partie formelle et une partie graphique. Les règles ne comportant qu'une partie graphique, par exemple toutes les règles exprimant les propriétés de l'égalité, sont appliquées en priorité et jusqu'à saturation contrairement aux autres règles dont l'application est heuristique.

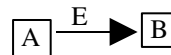
L'intérêt de ce graphe réside dans sa réalisation informatique qui permet des vérifications et des actions très rapidement faites.

MUSCADET [PAS 84, 89] est un système à base de connaissances développé par la suite. Ses premières versions ont repris de nombreuses propriétés de DATTE en particulier cette séparation des hypothèses en relations binaires et autres hypothèses. Mais le langage d'expression des connaissances de MUSCADET ne permettait plus une implémentation efficace du graphe. De plus, de nouvelles connaissances étaient ajoutées, certaines devaient être dupliquées pour les deux types d'hypothèses. Enfin le système devait fréquemment tester de quel type d'hypothèse il s'agissait. Après que la base de connaissances ait atteint une certaine taille, le gain apporté par la séparation des deux types d'hypothèses est devenu inférieur à la perte de temps occasionnée par sa gestion et cette séparation a été définitivement abandonnée. Néanmoins il me semble que l'idée d'utiliser des représentations machines efficaces pour certains types d'hypothèses devrait être repris un jour. Mais ceci devrait être fait d'une manière plus automatique de façon d'une part à ce que l'expert ou l'utilisateur qui donne les connaissances n'ait pas à s'en soucier et d'autre part que la représentation machine puisse être d'autant plus efficace qu'elle ne serait utilisée que par la machine.

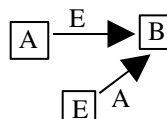
## 2 Opérateurs arithmétiques : "Doing arithmetic with diagrams"

Dans le but de simuler le comportement humain en résolution de problèmes, Bundy [BUN 73] a réalisé un démonstrateur de théorèmes en arithmétique qui utilise des graphes pour représenter les propriétés arithmétiques.

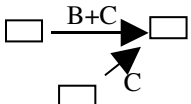
L'égalité  $B = A + E$  est représentée par le graphe suivant qui représente aussi la relation  $A \leq B$  dans  $\mathbb{N}$ .

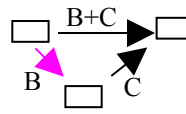


Le graphe suivant exprime la commutativité de l'addition.



La règle suivante en traduit l'associativité :

si on a  alors on rajoute la flèche B



Ces savoir-faire graphiques, ainsi que d'autres savoir-faire concernant en particulier la multiplication, les inégalités et le maximum de deux nombres remplacent axiomes et règles d'inférence classiques et le système SUMS (a System which Understands Mathematical Symbols) a démontré 70 théorèmes d'arithmétique dont la plupart sont difficiles à démontrer à partir des axiomes de Peano et a échoué pour 17 qui nécessiteraient d'autres savoir-faire.

Exemples de théorèmes démontrés :

$$\begin{array}{|l} A * (B + C) = A * B + A * C & \text{(distributivité de * par rapport à +)} \\ A \leq B \wedge B \leq C \Rightarrow A \leq C & \text{(transitivité de } \leq \text{)} \end{array}$$

Néanmoins, cette approche a des limites et un autre auteur, Brown, la conteste dans un autre article intitulé par opposition "Doing arithmetic without diagrams" [BRO 77]. Brown, admettant l'idée que des connaissances pourraient être incorporées dans des démonstrateurs mais rejetant l'idée que l'utilisation de diagrammes serait nécessaire, a réalisé un démonstrateur pour la théorie élémentaire des nombres qui représente les théorèmes comme des listes et applique des transformations conservant les valeurs de vérité. Toutes les expressions manipulées sont de la forme  $\phi_1, \dots, \phi_n \rightarrow \psi_1, \dots, \psi_m$  qui signifie  $\phi_1 \wedge \dots \wedge \phi_n \Rightarrow \psi_1 \vee \dots \vee \psi_m$ . Les connaissances sont toutes exprimées par des équivalences ou des égalités. Ce sont soit des connaissances logiques (qui seront utilisées plus tard dans un démonstrateur de théorèmes en théorie des ensembles ayant de bonnes performances [BRO 78]), soit des connaissances arithmétiques. Un exemple de connaissance est le remplacement de  $x < y$  par l'équivalence

$$x < y \Leftrightarrow \neg \exists u \ y + u = x \quad \text{ou par l'équivalence} \quad x < y \Leftrightarrow \exists u \ x + (u + 1) = y$$

selon que la sous-expression à évaluer est une hypothèse ou un but. Brown montre la supériorité de son système sur SUMS en démontrant 86 théorèmes sur les 87 essayés par SUMS.

Exemple de théorème démontré par son système et non par SUMS

$$A \leq B \wedge C \leq D \Rightarrow A + C \leq B + D$$

## 3 Représentation des ensembles en démonstration automatique

Le programme de Merialdo [MER 79] travaille en théorie des ensembles (et applications) et en topologie générale (ouverts, voisinages, intérieurs, adhérences, densité, frontières, continuité). Il a démontré plus de cent théorèmes.

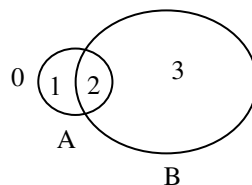
Grâce en particulier à une représentation des ensembles originale, il est d'une grande efficacité et il est capable de découvrir des résultats intermédiaires là où d'autres démonstrateurs doivent les recevoir comme lemmes ou les obtenir interactivement. Il possède de plus un mécanisme de recherche non

rigoureuse qui permet d'introduire des objets intermédiaires indispensables. Il utilise d'autres techniques devenues classiques ou au contraire améliorées depuis qui ne seront pas décrites ici. Je ne décrirai que les points qui en font aujourd'hui encore un travail original.

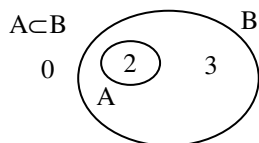
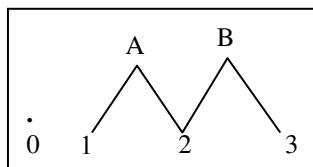
### 3.1 Une représentation qui simule et surpasse les diagrammes de Venn

De façon à avoir une perception visuelle des ensembles, on utilise souvent les diagrammes de Venn en théorie naïve des ensembles. Cette représentation permet de visualiser facilement les opérations et les relations ensemblistes (intersection, union, complémentaire, inclusion, différence). Merialdo [MER 79] a réalisé un démonstrateur en topologie algébrique qui utilise une représentation des ensembles sous forme d'un graphe qui simule les diagrammes de Venn. Cette représentation présente les mêmes avantages que les dessins faits par les êtres humains et elle est même supérieure d'une part parce qu'elle est capable de manipuler sans problème un plus grand nombre d'ensembles et d'autre part parce qu'elle est, en plus, rigoureuse.

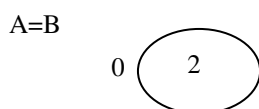
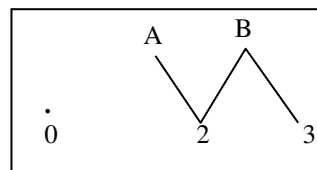
Le diagramme de Venn suivant permet de visualiser deux ensembles A et B. Il partage le plan en quatre régions 0, 1, 2 et 3.



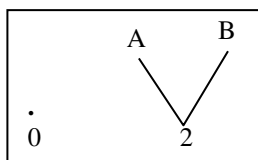
Le graphe de Merialdo est composé de *nœuds* et d'*atomes* reliés entre eux. Les nœuds sont les ensembles (ici A et B). Un atome *représente* les éléments qui appartiennent à tous les éléments auxquels il est relié et n'appartiennent pas aux autres. Dans le graphe ci-contre, l'atome 2 représente les éléments de  $A \cap B$ , l'atome 1 représente ceux de  $A - B$ , l'atome 3 représente ceux de  $B - A$ , et l'atome 0 représente ceux de  $C(A \cup B)$



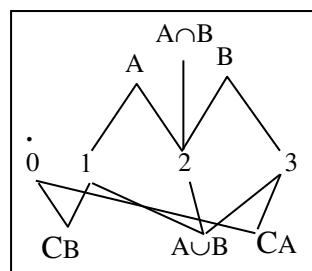
est exprimé par



par

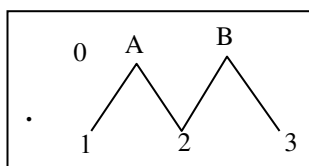
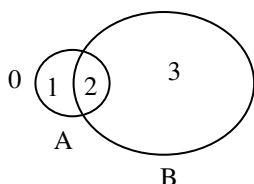


D'une manière générale, parmi un nombre quelconque d'ensembles,  
 $A \cap B$  est relié aux atomes qui sont reliés à la fois à A et à B.  
 $A \cup B$  est relié aux atomes qui sont reliés à A ou à B.  
 $CA$  est relié aux atomes qui ne sont pas reliés à A.<sup>1</sup>

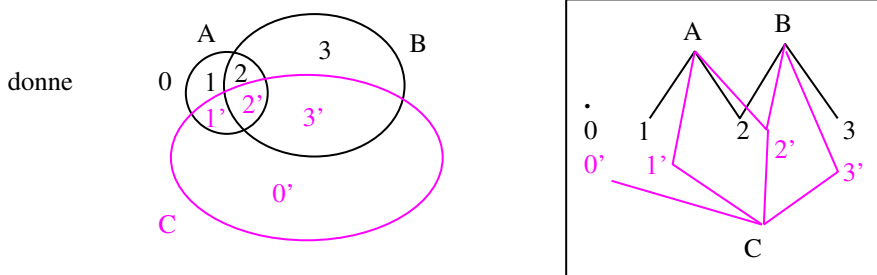


A chaque fois qu'on ajoute un nouvel ensemble, on duplique les atomes déjà existant et on relie ces nouveaux atomes au nouvel ensemble.

Ainsi, ajouter C à



<sup>1</sup> Il s'agit ici d'un complémentaire « naïf ». En théorie axiomatique on définirait le complémentaire d'un ensemble dans un autre,  $C_E A$  serait relié aux atomes qui sont reliés à E mais pas à A.



Au début, on a seulement l'atome 0. Ajouter A puis B donne le premier graphe donné ci-dessus comme premier exemple.

Le programme reçoit les propriétés concernant les relations d'inclusion et d'égalité :

$A \subset B$  si et seulement si tous les atomes reliés à A, le sont aussi à B.

$A = B$  si et seulement si A et B sont reliés exactement aux mêmes atomes.

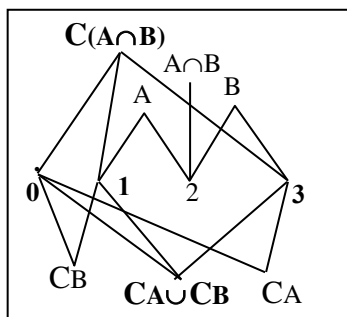
Ceci est utilisé dans les deux sens :

Pour *imposer* une relation, le programme modifie le graphe (suppression d'atomes et de liens).

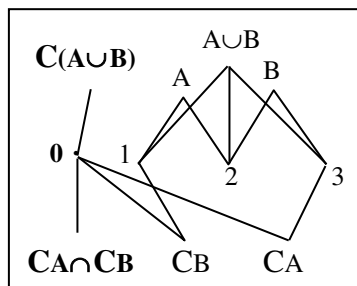
Sur l'exemple, pour imposer  $A \subset B$  il a supprimé l'atome 1 et le lien à A.

Pour *vérifier* (ou *découvrir*) une relation, il étudie si les propriétés précédentes sont vérifiées.

Premier exemple : démonstration des lois de Morgan :



$C(A \cap B) = CA \cup CB$  car reliés à 0, 1, 3



$C(A \cup B) = CA \cap CB$  car reliés à 0

La démonstration de la distributivité de l'intersection (resp. union) par rapport à l'union (resp. intersection) se démontre également aisément (c'est un peu plus compliqué pour l'être humain, sans problème pour le programme).

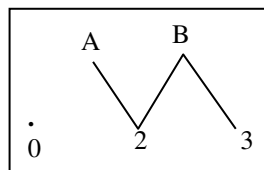
Ce graphe réalise en fait une *forme normale* pour les ensembles par les atomes auxquels ils sont liés.

Deuxième exemple : démonstration de la transitivité de l'inclusion

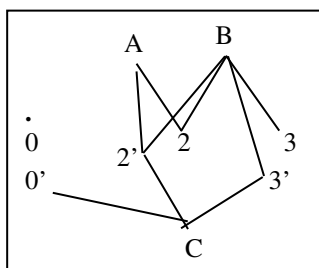
$$A \subset B \wedge B \subset C \Rightarrow A \subset C$$

Le programme construit le graphe correspondant aux hypothèses  $A \subset B$  et  $B \subset C$  puis vérifie la conclusion  $A \subset C$ .

- introduction des ensembles A et B tels que  $A \subset B$



- ajout de l'ensemble C

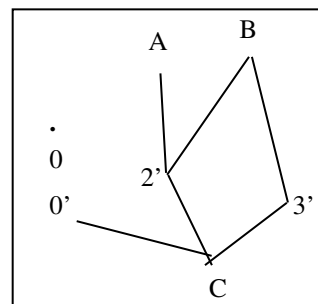




*Remarque* : on ne crée l'ensemble C qu'après avoir imposé  $A \subset B$  de façon à limiter le nombre d'atomes dupliqués.

- on impose  $B \subset C$ , ce qui se fait en supprimant les atomes 2 et 3 qui sont liés à B mais pas à C

- on vérifie que  $A \subset C$  : en effet seul l'atome 2' est lié à A et il est aussi lié à C



Non seulement ce théorème est immédiatement démontré, mais il pourra être utilisé comme propriété intermédiaire sans devoir être donné comme lemme comme dans la plupart des démonstrateurs à base de règles. En effet, une règle ayant comme condition une inclusion  $A \subset B$  qui n'a pas encore été déduite ne peut être appliquée que si le démonstrateur a un mécanisme sophistiqué de démonstration de conjectures, comme dans [SPA 01], qu'il est difficile de rendre général. Il est alors nécessaire de donner la propriété comme lemme, ce qui est tout à fait légitime pour une propriété aussi classique que la transitivité de l'inclusion mais le serait moins pour d'autres propriétés telles que celles rencontrées en topologie par ce programme.

Comme pour les ensembles, cette représentation réalise en fait une *forme normale* pour les relations ensemblistes.  $A \subset B$ ,  $A \cap CB = \emptyset$ ,  $CB \subset CA$  se traduisent en effet de la même façon sur le graphe.

Le programme n'a ainsi aucune connaissance formelle sur les ensembles mais toutes les connaissances ensemblistes sont incluses dans la construction et la manipulation du dessin.

L'inclusion et l'égalité sont les deux relations fondamentales parmi les relations ensemblistes traitées, elles sont facilement imposées ou reconnues. De plus périodiquement, le programme examine si des inclusions ou des égalités d'ensembles sont apparues. Cela peut se comparer au regard que l'être humain porte régulièrement sur son dessin entre d'autres étapes de son raisonnement.

Le but du programme de Mérialdo n'était pas la théorie des ensembles mais la topologie. Il comprend bien sûr d'autres méthodes très intéressantes, la gestion du dessin n'étant qu'une de ces méthodes. Mais c'est grâce en particulier à cette représentation des ensembles que les théorèmes de topologie nécessitant de découvrir des propriétés intermédiaires ont pu être démontrés.

*En voici quelques exemples* : si E est un espace topologique, on sait que E et  $\emptyset$  sont ouverts, pour montrer que E est fermé, c'est-à-dire que  $C_E E$  est ouvert, il suffit de découvrir que  $C_E E = \emptyset$  ce que le programme « voit » sur le dessin. De même pour montrer que l'intersection de deux fermés  $F1 \cap F2$  est fermée, on essaie de montrer que son complémentaire  $C(F1 \cap F2)$  est ouvert (définition). Comme d'autre part, les complémentaires de F1 et F2 sont ouverts, donc que leur union est un ouvert, il suffit de *découvrir* que  $C(F1 \cap F2) = CF1 \cup CF2$ , ce que le programme « voit » sur le dessin.

*Autre exemple* : Soit à montrer que  $A \subset B \Rightarrow \overset{\circ}{A} \subset \overset{\circ}{B}$  où  $\overset{\circ}{A}$  désigne l'intérieur de A.

On a  $A \subset B$  par hypothèse. Par définition de  $\overset{\circ}{A}$ ,  $\overset{\circ}{A}$  est ouvert et on a  $\overset{\circ}{A} \subset A$ . Le programme « voit » sur le dessin que  $\overset{\circ}{A} \subset B$  et en déduit que  $\overset{\circ}{A} \subset \overset{\circ}{B}$  par définition de  $\overset{\circ}{B}$  ( $\overset{\circ}{B}$  est le plus grand ouvert inclus dans B). S'il n'avait pas « vu » l'inclusion  $\overset{\circ}{A} \subset B$  il aurait dû recevoir le lemme de la transitivité de l'inclusion :  $X \subset Y \wedge Y \subset Z \Rightarrow X \subset Z$

*Autre exemple* : Montrer que  $A \overset{\circ}{\cup} B \subset \overset{\circ}{A \cup B}$

Les ensembles A, B,  $\overset{\circ}{A}$ ,  $\overset{\circ}{B}$ ,  $A \cup B$  et  $\overset{\circ}{A \cup B}$  sont introduits sur le graphe car ils apparaissent dans l'énoncé du théorème. On a  $\overset{\circ}{A} \subset A$  et  $\overset{\circ}{B} \subset B$  par définition de l'intérieur. Le programme « voit » alors sur le dessin que  $\overset{\circ}{A \cup B} \subset A \cup B$  et déduit alors le résultat cherché par définition de  $\overset{\circ}{A \cup B}$ . S'il n'avait pas « vu » l'inclusion  $\overset{\circ}{A \cup B} \subset A \cup B$  il aurait dû recevoir les lemmes suivants :

$$A \subset A \cup B$$

$$B \subset A \cup B$$

$$X \subset Z \wedge Y \subset Z \Rightarrow X \cup Y \subset Z$$

ainsi que la transitivité de l'inclusion

$$X \subset Y \wedge Y \subset Z \Rightarrow X \subset Z$$

Ces propriétés sont donc découvertes par le programme de Mérialdo alors qu'elles étaient données dans le programme de Ballantyne [BAL 73] vu dans la section 1.2.1.

Le théorème  $\text{Fr}(\bar{A}) \subset \text{Fr}(A)$

mentionné également dans la section 1.2.1 est également démontré facilement par le programme de Mérialdo, sans qu'il soit nécessaire de lui donner des lemmes. Pour ce théorème, dans [BAL 73] il fallait en plus donner le lemme  $A \subset B \Rightarrow CB \subset CA$ .

La démonstration du programme de Mérialdo est la suivante :

$$A \subset \bar{A} \text{ par définition de } \bar{A}$$

$$\overline{C\bar{A}} \subset \overline{CA} \text{ est « vu » sur le dessin}$$

$$\overline{C\bar{A}} \subset \overline{C\bar{A}} \text{ par définition de l'adhérence}$$

$$\bar{\bar{A}} = \bar{A} \text{ est « vu » sur le dessin}$$

$$\bar{A} \cap \overline{C\bar{A}} \subset \bar{A} \cap \overline{CA} \text{ est « vu » sur le dessin, c'est la propriété qui était à démontrer après remplacement des frontières par leur définition.}$$

Cette représentation est efficace pour une première raison qui est que toutes les égalités et inclusions déductibles sont connues. Elles sont connues car périodiquement le programme les cherche. La deuxième raison de l'efficacité est que cette recherche est très rapide grâce à la réalisation informatique spécialisée qui est faite et que le temps passé à la faire périodiquement est négligeable par rapport au reste de la démonstration.

Ce phénomène est apparu en particulier quand on a simulé quelques-unes des stratégies du programme de Mérialdo par le démonstrateur MUSCADET [PAS 84] qui est un système à base de connaissances. Ce travail a donné lieu à deux stages de DEA [LAU 86] et [ARD 87]. Ces deux réalisations ont eu en commun de simuler le graphe et sa gestion dans le langage général d'expression des connaissances de MUSCADET, sous forme de règles. Mais d'une part l'écriture des règles a été difficile car ces stratégies sont procédurales et le langage de MUSCADET a été conçu pour exprimer des connaissances déclaratives. De nombreuses super-actions<sup>2</sup> ont donc dû être écrites qui traduisent des procédures. Elles sont aussi difficiles à lire que dans un langage de programmation classique et ont été coûteuses en temps d'exécution car interprétées par le moteur d'inférence de MUSCADET non compilé ni optimisé.

En conclusion, l'efficacité du programme de Mérialdo repose sur le graphe mais *aussi* sur sa représentation interne et les procédures rapides qui y étaient associées. L'existence du graphe et le fait que toutes les inclusions déductibles y sont automatiquement mises à jour *rapidement* est très utile.

On peut comparer l'utilité de ce graphe à l'utilité d'un « dessin » dans le raisonnement humain. Celui-ci nous aide en nous permettant de découvrir *rapidement* des propriétés importantes.

On remarquera de plus que le graphe de Mérialdo fonctionne comme une boîte noire, les relations découvertes ne sont pas justifiées, comme c'est le cas pour ce que nous « voyons » sur un dessin. Par contre, **cette représentation est meilleure que les dessins que nous faisons** sur papier car le programme la gérant est **rigoureux**.

### 3.2 Extensions

Le graphe est également étendu pour traiter les éléments des ensembles et les applications, en particulier les images et images réciproques de sous-ensembles. Mais il n'est pas possible de l'étendre aux ensembles ou familles d'ensembles, par exemple à l'ensemble des parties d'un ensemble ou à des familles infinies d'ensembles.

<sup>2</sup> Les super-actions peuvent apparaître dans la partie <actions> d'une règle en MUSCADET. Elles sont définies par des paquets de règles.

### 3.2.1 Extension aux points

Pour représenter les points (éléments des ensembles), un nouveau type de lien est introduit sur le graphe. A priori un point est relié à un atome.

Pour *imposer*  $x \in A$ , on supprime tous les liens vers des atomes non liés à A.

Pour *vérifier*  $x \in A$ , on vérifie que tous les atomes liés à x sont aussi liés à A.

Pour *imposer*  $x=y$ , on lie x et y aux mêmes atomes.

On ne peut pas *vérifier* que  $x=y$ .

Ces nouveaux liens ont permis au progamme de démontrer des théorèmes comme

$$x \in A \wedge x \in B \Leftrightarrow x \in A \cap B$$

Un prétraitement est d'abord effectué et les deux sous-théorèmes

$$x \in A \wedge x \in B \Rightarrow x \in A \cap B$$

$$\text{et } x \in A \cap B \Rightarrow x \in A \wedge x \in B$$

sont démontrés sur le graphe.

### 3.2.2 Extension aux applications

Soit f une application de X dans Y, avec X et Y disjoints

Si  $A \subset X$  on crée l'ensemble image de A :

$$f(A) = \{y \in Y \mid \exists x \in A \ y = f(x)\}$$

Si  $B \subset Y$  on crée l'ensemble image réciproque de B :

$$f^{-1}(B) = \{x \in A \mid f(x) \in B\}$$

On sature mais on ne crée  $f^{-1}(A)$  et  $f(B)$  que s'ils sont explicites dans l'énoncé.

On a les règles de consistance suivantes :

(1)  $f(A) \subset Y$

(2)  $f^{-1}(B) \subset X$

(3) si l'atome a est lié à X alors

- si a lié à A alors les images possibles sont dans f(A)
- si a lié à  $f^{-1}(B)$  alors les images possibles sont dans B
- si a non lié à  $f^{-1}(B)$  alors pas d'image dans B

On examine toutes les parties A et B possibles et on marque les atomes qui doivent ou qui ne doivent pas contenir des images d'éléments de a. Si aucun atome de Y ne satisfait à ces conditions (être lié à f(A) [resp. B] si a est lié à A [resp.  $f^{-1}(B)$ ], ne pas être lié à B si a non lié à  $f^{-1}(B)$ ), on peut supprimer l'atome a.

(4) si l'atome b lié à Y

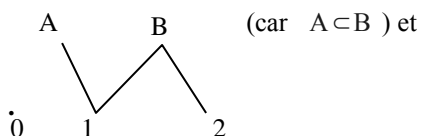
- si b lié à f(A) alors il existe un antécédent dans a
- si b non lié à f(A) alors pas d'antécédent dans A'
- si b non lié à B alors pas d'antécédent dans  $f^{-1}(B)$

Il faut vérifier que pour chaque A tel que b soit lié à f(A) il existe un atome lié à A qui ne soit lié à aucun A' [resp.  $f^{-1}(B)$ ] si b non lié à f(A) [resp. B]. Si cela n'est pas vérifié, on peut supprimer l'atome b.

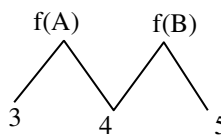
Exemple : Montrer que  $A \subset B \Rightarrow f(A) \subset f(B)$

(On ne dessinera que les parties caractéristiques du graphe)

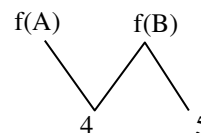
On a



(car  $A \subset B$ ) et



L'atome 3 est lié à f(A) et n'est pas lié à f(B). Mais il n'existe pas d'atome lié à A et non à B. Donc on supprime l'atome 3 du graphe.



L'inclusion  $f(A) \subset f(B)$  est alors vérifiée.

### 3.3 Enrichissement

L'enrichissement consiste à appliquer un axiome, un théorème connu, une définition ou une hypothèse universelle pour rajouter de nouvelles hypothèses (chainage avant) ou modifier la conclusion (chainage arrière). Cet enrichissement peut se faire en trois phases, correspondant à des priorités différentes.

phase 1 : à partir des hypothèses universelles  $\forall x(P(x) \Rightarrow Q(x))$

phases 2 et 3 : à partir des axiomes, définitions, lemmes donnés au programme, la distinction est faite par l'utilisateur, par exemple

$$\forall A \forall B \forall C (A \text{ et } B \text{ ouverts} \wedge C = A \cup B \Rightarrow \text{Couvert})$$

$$\forall A \forall B \forall C (A \text{ et } B \text{ ouverts} \wedge C = A \cap B \Rightarrow \text{Couvert})$$

$$\forall X (X \in \mathcal{F} \Rightarrow X \text{ ouvert}) \Rightarrow \cup \mathcal{F} \text{ ouvert}$$

$$\forall X (X \in \mathcal{F} \Rightarrow X \text{ ouvert}) \text{ et } \mathcal{F} \text{ finie} \Rightarrow \cap \mathcal{F} \text{ ouvert}$$

sont en phase 2 (non expansives)

alors que

$$\forall A \forall B (A \text{ et } B \text{ ouverts} \Rightarrow A \cup B \text{ ouvert})$$

$$\forall A \forall B (A \text{ et } B \text{ ouverts} \Rightarrow A \cap B \text{ ouvert})$$

ne sont qu'en phase 3 (possiblement expansives).

### 3.4 Activation des applications

Si une application est définie implicitement, le programme l'explique, ce qui lui permet ensuite de la traiter comme une application, c'est-à-dire, entre autres, de créer des images ou des images réciproques.

Deux situations sont possibles :

- à partir d'une expression existentielle, on crée une fonction de Skolem  $f$

$$\forall x(x \in A \Rightarrow \exists y Q(x, y)) \text{ devient } (\forall x(x \in A \Rightarrow Q(x, f(x))))$$

- à partir d'une opération, par exemple le complémentaire, on pose  $f(x) = C_E x$

$$\forall x(x \in A \Rightarrow Q(x, C_E x)) \text{ devient } (\forall x(x \in A \Rightarrow Q(x, f(x))))$$

Alors dans les deux cas, si  $Q(x, y)$  est de la forme  $U(y) \wedge V(x, y)$  les hypothèses suivantes sont ajoutées

$$\forall x(x \in A \Rightarrow f(x) \in f(A))$$

$$\forall y(y \in f(A) \Rightarrow \exists x(x \in A \wedge y = f(x)))$$

$$\forall y(y \in f(A) \Rightarrow U(y))$$

$$\forall x(x \in A \Rightarrow V(x, f(x)))$$

### 3.5 Règle du doublet

Une famille infinie  $\mathcal{F}$  d'ensembles ne peut pas être représentée sur le graphe. Ses propriétés et celles de ses éléments seront alors traitées comme des propriétés non ensemblistes et ne bénéficient donc pas de l'efficacité de la représentation graphique. Pour pallier ce manque Mérialdo a introduit la technique de la *règle du doublet*.

La famille va (provisoirement) n'être constituée que de deux éléments  $E1$  et  $E2$ , soit  $\mathcal{F} = \{E1, E2\}$ , c'est-à-dire vérifier les propriétés

$$\forall X (X \in \mathcal{F} \Rightarrow X = E1 \vee X = E2)$$

$$\cup \mathcal{F} = E1 \cup E2$$

$$\cap \mathcal{F} = E1 \cap E2$$

$E1, E2, E1 \cup E2$  et  $E1 \cap E2$  figurent sur le graphe et si on a par exemple

$$\forall X (X \in \mathcal{F} \Rightarrow X \subset A)$$

le graphe sera enrichi de la relation

$$\cup \mathcal{F} = A$$

La démonstration ne sera pas correcte, mais elle aura pu permettre d'introduire un objet utile, et, si c'est le cas, proposer une piste pour démontrer correctement les résultats ainsi obtenus <sup>3</sup>.

### 3.5 Exemples d'utilisation de la règle du doublet

3.5.1 *théorème à démontrer*  $A \text{ ouvert} \Leftrightarrow \forall x (x \in A \Rightarrow A \text{ voisinage de } x)$

(1) *premier sous-théorème*  $A \text{ ouvert} \Rightarrow \forall x (x \in A \Rightarrow A \text{ voisinage de } x)$   
 qui devient  $A \text{ ouvert} \wedge x \in A \Rightarrow A \text{ voisinage de } x$   
 la conclusion est remplacée par sa définition  $\exists v (v \text{ ouvert} \wedge v \subset A \wedge x \in v)$   
 le programme essaie plusieurs valeurs pour  $v$ ,  $\Phi$ ,  $E$  puis  $A$  qui convient.

(2) *deuxième sous-théorème*  $\forall x (x \in A \Rightarrow A \text{ voisinage de } x) \Rightarrow A \text{ ouvert}$   
 Remplacer la définition de *voisinage* donne

$$\forall x (x \in A \Rightarrow \exists w (w \text{ ouvert} \wedge w \subset A \wedge x \in w)) \Rightarrow A \text{ ouvert} \quad 4$$

la formule existentielle est skolémisée et la fonction de Skolem est activée (section 3.4).

Avec  $U(w) = w \text{ ouvert} \wedge w \subset A$  et  $V(x, w) = x \in w$  on obtient les nouvelles hypothèses

$$\begin{aligned} \forall x (x \in A \Rightarrow f(x) \in f(A)) \\ \forall y (y \in f(A) \Rightarrow \exists x (x \in A \wedge y = f(x))) \\ \forall y (y \in f(A) \Rightarrow y \text{ ouvert} \wedge y \subset A) \\ \forall x (x \in A \Rightarrow x \in f(x)) \end{aligned}$$

L'enrichissement en phase 2 ajoute alors l'hypothèse  $\cup f(A) \text{ ouvert}$   
 Avec la règle du doublet, le graphe permet de déduire (proposer) que  $\cup f(A) \subset A$   
 Par contre, le programme ne peut pas déduire que  $A \subset \cup f(A)$   
 Néanmoins, le programme a eu l'idée d'introduire et de travailler sur l'objet utile  $\cup f(A)$ .

3.5.2 *Montrer que toute intersection de fermés est un fermé*

$$\forall X (X \in \mathcal{F} \Rightarrow X \text{ fermé}) \Rightarrow \cap \mathcal{F} \text{ fermé}$$

Remplacer la définition de *fermé* donne <sup>5</sup>

$$\forall X (X \in \mathcal{F} \Rightarrow C_E X \text{ ouvert}) \Rightarrow \cap \mathcal{F} \text{ fermé}$$

L'enrichissement en phase 2 définit le *fermé* de la conclusion

$$\forall X (X \in \mathcal{F} \Rightarrow C_E X \text{ ouvert}) \Rightarrow C_E(\cap \mathcal{F}) \text{ ouvert}$$

Comme il a été vu en section 3.4, le complémentaire, considéré comme une application  $f$  sur  $\mathcal{F}$  est activé et  $f(\mathcal{F})$  est introduit.

Avec  $U(y) = (y \text{ ouvert})$ ,  $V(x, y) = \text{vrai}$ , les hypothèses suivantes sont ajoutées

$$\begin{aligned} \forall X (X \in \mathcal{F} \Rightarrow C_E X \in f(\mathcal{F})) \\ \forall Y (Y \in f(\mathcal{F}) \Rightarrow \exists X (X \in \mathcal{F} \wedge Y = C_E X)) \\ \forall Y (Y \in f(\mathcal{F}) \Rightarrow Y \text{ ouvert}) \end{aligned}$$

L'enrichissement en phase 2 ajoute l'hypothèse

$$\cup f(\mathcal{F}) \text{ ouvert}$$

Avec la règle du doublet, le graphe permet de déduire (proposer) que  $\cup f(\mathcal{F}) = C_E(\cap \mathcal{F})$

$C_E(\cap \mathcal{F})$  est donc ouvert.

Remarque : la règle du doublet peut aussi, mais sans aucune utilité, s'appliquer sur l'énoncé initial

$$\forall X (X \in \mathcal{F} \Rightarrow X \text{ fermé}) \Rightarrow \cap \mathcal{F} \text{ fermé}$$

La règle du doublet donne  $\mathcal{F} = \{E1, E2\}$ ,  $\cup \mathcal{F} = E1 \cup E2$ ,  $\cap \mathcal{F} = E1 \cap E2$ .

<sup>3</sup> Dans la thèse, le programme s'arrête au « provisoire », il montre seulement la faisabilité, il resterait à effectuer ensuite la démonstration rigoureuse utilisant les nouveaux objets créés.

<sup>4</sup> Le programme de Mérialdo ne peut pas appliquer une définition à l'intérieur d'une hypothèse, échoue donc. Mérialdo effectue le remplacement à la main et propose ce nouvel énoncé au programme. Faire faire le remplacement automatiquement par le programme serait une amélioration possible.

<sup>5</sup> Comme précédemment, le programme ne peut pas remplacer *fermé* par sa définition à l'intérieur de l'hypothèse. Il reçoit donc directement l'énoncé

L'enrichissement donne  $E1$  et  $E2$  fermés, puis  $\cup \mathcal{F}$  et  $\cap \mathcal{F}$  fermés. On a donc obtenu la conclusion cherchée, mais aucun nouvel élément n'a été introduit, donc pas d'aide pour trouver une démonstration correcte. (D'ailleurs on a aussi obtenu la conclusion  $\cup \mathcal{F}$  fermé, fausse si  $\mathcal{F}$  est infinie).

## Bibliographie

- [ARD 87] F. Arditti, Rapport de stage du DEA IARFAG, Paris 6, 1987
- [BAL 73] M. Ballantyne, W. Bennett, Graphing methods for topological proofs, University of Texas at Austin, Math. Dept. Memo ATP 7, 1973
- [BAL 75] M. Ballantyne, Computer generation of counterexamples in topology, University of Texas at Austin, Math. Dept. Memo ATP 24, 1975
- [BAL 82] A.M. Ballantyne, W.W. Bledsoe, On generating and using examples in proof discovery, *Machine Intelligence*, vol. 10, 1982, p. 3-39
- [BLE 71] W.W. Bledsoe, Splitting and reduction heuristics in automatic theorem proving, *Artificial Intelligence*, vol. 2, 1971, p. 55-77
- [BRO 77] F.M. Brown, Doing arithmetic without diagrams, *Artificial Intelligence*, vol. 8, 1978, p. 281-316
- [BRO 78] F.M. Brown, Towards the automation of set theory and its logic, *Artificial Intelligence*, vol. 10, 1977, p. 175-200
- [BUN 73] A. Bundy, Doing arithmetic with diagrams, *IJCAI*, 1973, p. 130-138
- [LAU 76] J-L. Laurière, Un langage et un programme pour énoncer et résoudre des problèmes combinatoires, Thèse d'état, Paris 6, 1976
- [LAU 78] J-L. Laurière, A language and a program for stating and solving combinatorial problems, *Artificial Intelligence*, vol. 10, 1978, p. 29-127
- [LAU 86] P. Laublet, Rapport de stage du DEA IARFAG, Paris 6, 1986
- [MER 79] B. Mérialdo, Représentation des ensembles en démonstration automatique, thèse de 3<sup>ème</sup> cycle, Paris 6, 1979
- [PAS 78] D.Pastre, Automatic theorem proving in set theory, *Artificial Intelligence*, vol. 10, 1978, p. 1-27
- [PAS 84] D. Pastre, MUSCADET : un système de démonstration automatique de théorèmes utilisant connaissances et métaconnaissances en mathématiques, thèse d'état, Paris 6, 1984
- [PAS 89] D. Pastre, MUSCADET : an automatic theorem proving system using knowledge and metaknowledge in mathematics, *Artificial Intelligence*, vol. 38 (3), 1989, p. 257-318
- [SPA 01] J-P. Spagnol, Automatisation du raisonnement et de la rédaction de preuves en géométrie de l'enseignement secondaire, thèse de l'Université René Descartes - Paris 5, 2001