Macro-opérateurs revisités en apprentissage relationnel

Érick Alphonse

MIG - UR1111/INRA
78000 Jouy en Josas CEDEX FRANCE
ealphons@jouy.inra.fr

Résumé :

Depuis une dizaine d'années, un important travail de recherche a été consacré aux techniques dites de propositionnalisation. Ces techniques changent la représentation d'un problème relationnel en un problème attribut-valeur afin de déléguer l'apprentissage à des algorithmes plus simples, bien connus. Les approches par propositionnalisation ont été appliquées avec succès sur plusieurs problèmes, mais sont toujours considérées dans la littérature comme des techniques ad hoc. Dans cet article, nous étudions ces approches dans le contexte plus large des macro-opérateurs, comme un moyen d'améliorer la recherche heuristique. Le paradigme des macro-opérateurs nous permet d'une part de proposer une vue unifiée de la propositionnalisation, et d'autre part, de présenter ces limitations actuelles. Nous montrons qu'une famille plus générale d'approches peut être développée en apprentissage relationnel, qui étend l'idée des changements de représentation vers des langages plus adaptés à l'apprentissage. Particulièrement, nous proposons un nouveau langage de représentation, appelé multi-table, qui permet un meilleur compromis que les techniques de propositionnalisation actuelles entre le coût de construction des macro-opérateurs et celui de l'apprentissage. Ce nouveau langage est plus expressif que l'attribut-valeur, mais moins que le multi-instance. De plus, il est PAC-apprenable sous une contrainte faible. Finalement, nous suggérons que l'apprentissage relationnel peut tirer parti à la fois des communautés en résolution de problèmes et en apprentissage attribut-valeur, en se concentrant sur la définition de techniques efficaces de recherche de macro-opérateurs.

1 Introduction

Après (Mitchell, 1982), l'apprentissage de concept est défini comme une recherche : étant donné un espace d'hypothèses défini a priori, identifié par son langage de représentation, chercher dans cet espace une hypothèse cohérente avec les données. Cet article fondateur, assimilant l'apprentissage à une recherche dans un espace d'états, a permis à l'Apprentissage Automatique de s'enrichir de techniques de recherche issues de la Combinatoire, de la Recherche Opérationnelle et de la Résolution de Problèmes

(*Problem Solving*). Cependant, la recherche est NP-complète pour une grande variété de langages d'intérêt (par exemple (Haussler, 1989; Cohen, 1993; Kearns & Vazirani, 1994)) et la recherche heuristique est cruciale pour l'efficacité. Si celle-ci s'est montrée efficace dans des langages d'attribut-valeur, il s'est avéré très tôt que l'apprentissage dans les langages relationnels, connu depuis plus de dix ans sous le nom de Programmation Logique Inductive (PLI), doit faire face à d'importants *phénomènes de plateau* (Quinlan, 1991; Silverstein & Pazzani, 1991; Richards & Mooney, 1992) : la fonction d'évaluation, utilisée pour privilégier le développement de tel ou tel nœud dans le graphe de raffinement, est constante par endroit, la recherche se déroulant alors à l'aveugle. Ces phénomènes de plateau représente le cas pathologique de la recherche heuristique (exhaustive ou non) (Pearl, 1985).

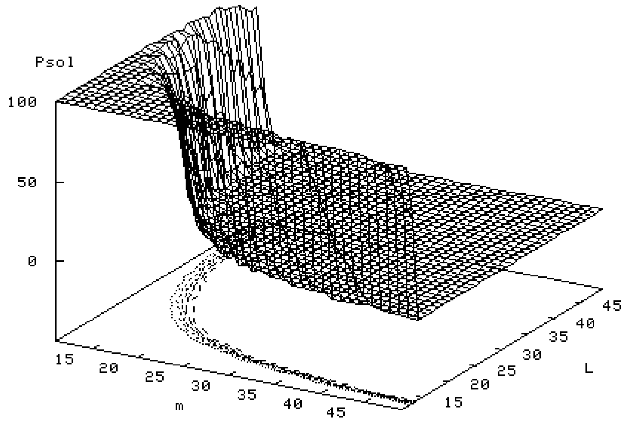

FIG. 1 – (Giordana & Saitta, 2000) Probabilité P_{sol} de couverture d'un exemple comprenant L constantes par une hypothèse comprenant m littéraux (construits sur 10 variables). Le contour projeté sur le plan (m, L) correspond à l'épaisseur de la zone où la probabilité de trouver une solution est entre 0.99 et 0.01

Une explication peut être donnée après le travail de (Giordana & Saitta, 2000), qui ont étudié le test de couverture en PLI dans le contexte de la transition de phase. Comme illustré dans à la figure 1, le test de couverture est NP-complet et exhibe donc une transition de phase nette dans sa probabilité de couverture (Cheeseman *et al.*, 1991). Si nous examinons la probabilité de couverture d'un exemple quelconque de taille fixe par une hypothèse quelconque, en fonction de la taille de celle-ci, nous distinguons trois zones bien identifiées comme le montre la figure 1 : une zone, appelée "yes", où la probabilité de couverture est proche de 1, un zone, appelée "no", où la probabilité est proche de 0, et enfin la transition de phase où un exemple peut ou pas être couvert. Comme la valeur heuristique d'une hypothèse prend en compte le nombre d'exemples couverts (positifs et négatifs), nous voyons que les différentes régions "yes" et "no" représentent des plateaux qu'il faut traverser pendant la recherche sans être guidé de façon heuristique.

Un état de l'art sur les fonctions d'évaluation en apprentissage (voir par exemple (Fürnkranz & Flach, 2003)) révèle que toutes reposent sur deux paramètres principaux qui sont le taux d'exemples positifs et négatifs couverts. Comme ces paramètres découlent directement de la formalisation du problème de cohérence à résoudre, il apparaît peu probable qu'une solution consiste à développer de nouvelles heuristiques.

Ce problème a été bien étudié dans la communauté de résolution de problèmes (Korf, 1985) et la solution proposée est fondée sur l'utilisation de macro-opérateurs. Les macro-opérateurs sont des opérateurs de raffinement définis par composition des opérateurs de raffinement élémentaires de l'espace de recherche initial. Ils peuvent appliquer plusieurs opérateurs élémentaires à la fois (ajoutant ou enlevant plusieurs littéraux dans le cas de l'apprentissage relationnel) et sont susceptibles alors de traverser les plateaux non informatifs. L'utilisation des macro-opérateurs en PLI n'est pas nouvelle et a été étudiée par exemple dans (Quinlan, 1991; Silverstein & Pazzani, 1991; Richards & Mooney, 1992; Peña Castillo & Wrobel, 2002). Dans ces systèmes, les macro-opérateurs *sont ajoutés* à la liste des opérateurs de raffinement initiaux et sont alors classiquement exploités pour parcourir le graphe de raffinement. En général, ce type de changement de représentation en apprentissage est connu sous le nom d'Induction Constructive (Michalski, 1978), ou Invention de Prédicats en PLI. Classiquement, ces approches *enrichissent* le langage des hypothèses dans le but de faciliter l'apprentissage. Cependant, en pratique l'augmentation de l'espace des hypothèses s'accompagne d'une détérioration des performances (Khan *et al.*, 1998).

Contrairement à ces approches, nous voyons les macro-opérateurs comme un moyen de *simplifier* l'espace de recherche en ne considérant seulement les macro-opérateurs comme opérateurs élémentaires. Autrement dit, nous ne nous intéressons pas au nouveau graphe de raffinement enrichi par l'ajout de macro-opérateurs, mais seulement au sous-graphe engendré par les macro-opérateurs. Nous proposons alors d'analyser une approche par macro-opérateurs en fonction de la complexité du langage des hypothèses associé à ce sous-graphe. Cette technique est similaire aux travaux de (Amarel, 1968) et peut être placée dans le contexte de l'Abstraction (voir par exemple (Zucker, 2001)).

Nous allons montré dans cet article qu'un grand nombre d'approches en PLI, appelées *propositionnalisation* après (Kramer *et al.*, 1998), utilisent implicitement les macro-opérateurs pour sélectionner un sous-graphe dont le langage de représentation est équivalent au langage attribut-valeur (AV). De cette façon, ils peuvent déléguer le problème d'apprentissage simplifié aux algorithmes AV bien connus (Turney, 1995; Geibel & Wysotzki, 1995; Kramer *et al.*, 1998; Cumby & Roth, 2000; Lavrač & Flach, 2001; Krogel & Wrobel, 2001; Roth & Yih, 2001; Kramer & de Raedt, 2001; Liquière, 2001). L'intérêt de cette formalisation est triple. Premièrement, elle permet de clarifier et de motiver la propositionnalisation qui est toujours considérée dans la littérature comme une collection de méthodes ad hoc. Deuxièmement, en tissant le lien avec les macro-opérateurs, elle permet à la PLI de s'enrichir des techniques développées dans ce cadre par la communauté en Résolution de Problèmes pour la définition de méthodes efficaces de construction de macro-opérateurs (voir par exemple (Hernádvölgyi, 2001). Enfin et surtout, cette analyse mets en lumière un certain nombre d'extensions qui paraissent prometteuses. Si les méthodes actuelles ciblent le langage AV qui est intéressant parce que les méthodes d'apprentissage appliquées sont efficaces, le coût de construction des macro-opérateurs est prohibitif et équivalent au coût de résolution des systèmes «classiques» de PLI sur des cas très simples. Nous proposons de relaxer la contrainte sur l'expressivité du langage d'hypothèse induit par les macro-opérateurs pour offrir un meilleur compromis entre le coût de construction des macro-opérateurs et celui de l'apprentissage.

À la section suivante, nous illustrons l'utilisation des macro-opérateurs (que nous abrégerons parfois en macros) pour la résolution des problèmes de PLI et nous montrons comment nous pouvons associer un langage au sous-espace d'hypothèses engendré par les macros. Un ensemble de langages qui peuvent être choisis au moyen de macros sera présenté. Nous montrerons alors à la section 3 que le langage déterminé actuellement utilisé par les approches courantes de propositionnalisation ne permet pas de déléguer efficacement l'apprentissage aux algorithmes AV. À la section, 4, nous discuterons du langage des k-locales, proposé par (Cohen, 1993), qui apparaît comme la première relaxation prometteuse du langage des hypothèses. Nous montrerons que ce langage permet de reformuler le problème de PLI initial dans un nouveau langage de représentation que nous appelons *multi-table*. Ce langage est plus simple que le multi-instance (Dietterich *et al.*, 1997). De plus, dans une forme plus contrainte du langage, nous verrons qu'un monôme est PAC-apprenable, ce qui suggère que des algorithmes efficaces existent. À la section 5, nous généralisons le langage du sous-espace d'hypothèse aux langages k-libre et k-indéterminé et discutons rapidement des propriétés de la propositionnalisation dans ces langages. Finalement, nous conclurons et argumenterons que l'important effort réalisé en propositionnalisation ces dernières années peut être poursuivi dans des langages plus riches.

2 Résolution de problèmes de PLI par macro-opérateurs

Comme indiqué précédemment, les macro-opérateurs ont déjà été utilisés en PLI. Nous illustrons leur apport sur un exemple simple exhibant un plateau à la première étape de raffinement. Par manque de place, nous considérons le lecteur familier avec la PLI (voir par exemple (Nienhuys-Cheng & de Wolf, 1997)).

Exemple 1
Nous appliquons une algorithme descendant glouton à la FOIL (Quinlan, 1991) sur le problème suivant de type train de Michalski :

Exemples	*Connaissance du domaine*			
e_1^+ $ouest(t1)$	$voiture\_1(t1, v11).$	$courte(v11).$	$voiture\_2(t1, v12).$	$toit(v12).$
e_2^+ $ouest(t2)$	$voiture\_1(t2, v21).$	$courte(v21).$	$voiture\_2(t2, v22).$	$courte(v22).$
e_1^- $ouest(t3)$	$voiture\_1(t3, v31).$	$toit(v31).$	$voiture\_2(t3, v32).$	$toit(v32).$

Il a deux exemples positifs et un exemple négatif. La sémantique est classique et par exemple le premier exemple positif se lit : le train va vers l'ouest, sa première voiture est courte et sa seconde voiture a un toit. L'hypothèse suivante est cohérente avec ce problème :

$$ouest(T) \leftarrow voiture\_1(T, V1), courte(V1)$$

La recherche démarre avec une hypothèse dont le corps est vide et considère l'ensemble des littéraux possibles à lui ajouter. Les deux littéraux $voiture\_1(T, V1)$ *et*

$voiture\_2(T, V2)$ *seuls ne discriminent pas les exemples positifs des négatifs (chaque train possédant deux voitures) et ont ainsi la même valeur heuristique (un gain d'information nul). L'algorithme d'apprentissage a un choix à faire et par conséquent peut ajouter* $voiture\_2(T, V2)$*, qui ne conduit à aucune solution, alors que la conjonction* $voiture\_1(T, V1), courte(V1)$ *discrimine les classes. La solution alors, pour pallier le manque d'information heuristique, consiste à définir un macro-opérateur pour spécialiser l'hypothèse par l'ajout de cette conjonction, traversant ainsi le plateau.*

La solution que nous adoptons pour définir les macro-opérateurs, qui est discutée également dans (Silverstein & Pazzani, 1991), et plus généralement est l'approche de l'Induction Constructive, est de créer un nouveau littéral dont la définition correspond à la composition des opérateurs élémentaires. En enrichissant la connaissance du domaine avec ce littéral, les opérateurs de raffinement initiaux peuvent faire usage de ce littéral pour raffiner une hypothèse. Nous définissons un macro-opérateur par rapport au langage d'hypothèse, sa définition correspondant à une séquence valide des opérateurs de raffinement.

Définition 2.1 (macro-opérateur)
Un macro-opérateur est un prédicat m*. La définition du macro-opérateur* m *est un ensemble de clauses* $\{D_i\}$ *ayant toutes pour tête un atome construit sur* m*. Chaque clause est construite sur le langage des hypothèses* $\mathcal{L}_h$*, c'est-à-dire :*

$$\forall i, \exists h \in \mathcal{L}_h \text{ tq } corps(D_i) = corps(h)$$

Exemple 2
Nous utilisons la notation prolog de définition de prédicats pour définir les deux macros suivants :

$$voiture\_1\_courte(T) \quad :- \quad voiture\_1(T, V1), courte(V1).$$
$$voiture\_2\_toit(T) \quad :- \quad voiture\_2(T, V1), toit(V1).$$

Les nouveaux littéraux sont ajoutés à la connaissance du domaine si leur définition y est vraie, ce qui nous donne une nouvelle représentation du problème d'apprentissage :

Exemples	*Connaissance du domaine*			
e_1^+ $ouest(t1)$	$v\_1(t1, v11).$	$courte(v11).$	$v\_2(t1, v12).$	$toit(v12).$
e_2^+ $ouest(t2)$	$v\_1(t2, v21).$	$courte(v21).$	$v\_2(t2, v22).$	$courte(v22).$
e_1^- $ouest(t3)$	$v\_1(t3, v31).$	$toit(v31).$	$v\_2(t3, v32).$	$toit(v32).$
	$\mathbf{v\_1\_courte(t1)}.$	$\mathbf{v\_2\_toit(t1)}.$	$\mathbf{v\_1\_courte(t2)}.$	$\mathbf{v\_2\_toit(t3)}.$

Ici, nous proposons de simplifier la représentation en utilisant seulement les nouveaux littéraux introduits. Autrement dit, nous ne nous intéressons pas au nouveau graphe de raffinement enrichi par l'ajout de macro-opérateurs, mais seulement au sous-graphe engendré par les macro-opérateurs. Si nous considérons à nouveau l'exemple au-dessus avec cette approche nous avons la nouvelle représentation :

Exemples	Connaissance du domaine	
e_1^+ $ouest(t1)$	$voiture\_1\_courte(t1).$	$voiture\_2\_toit(t1).$
e_2^+ $ouest(t2)$	$voiture\_1\_courte(t2).$	$voiture\_2\_toit(t3).$
e_1^- $ouest(t3)$		

Nous pouvons noter maintenant que seules les constantes des exemples sont présentes dans la connaissance du domaine. Ce langage est remarquable car c'est un langage déterminé et il est connu pour être aussi expressif que le langage attribut-valeur. (Dzeroski *et al.*, 1992; Lavrač & Džeroski, 1994) ont montré que l'espace d'hypothèses associé est isomorphe à l'espace d'hypothèses AV et que la recherche pouvait être déléguée à un algorithme AV en reformulant les exemples d'apprentissage. Cette reformulation est illustrée dans le tableau 1.

TAB. 1 – Transformation du nouveau problème déterminé en un problème propositionnel

	Variables	Attributs	
	T	$voiture\_1\_courte(T)$	$voiture\_2\_toit(T)$
e_1^+	$t1$	V	V
e_2^+	$t2$	V	F
e_1^-	$t3$	F	V

Cet exemple est représentatif de la résolution par macro-opérateurs que nous proposons et nous allons donner plus loin un algorithme général (section 2.2). Autant que nous sachions, les travaux sur cette approche, même si ils ne font pas référence aux macro-opérateurs sont dû à (Turney, 1995; Geibel & Wysotzki, 1995). Un nombre important de travaux leur ont fait suite (Kramer *et al.*, 1998; Cumby & Roth, 2000; Lavrač & Flach, 2001; Krogel & Wrobel, 2001; Roth & Yih, 2001; Kramer & de Raedt, 2001; Liquière, 2001) avec le même principe de définition de macro-opérateurs pour simplifier le problème en un problème déterminé, puis de reformulation du nouveau problème d'après (Dzeroski *et al.*, 1992), pour déléguer l'apprentissage à des algorithmes AV. Nous ne discutons pas les différentes techniques de construction de ce que nous appelons les macro-opérateurs, étant donné que cela est hors-sujet dans cet article. Une revue détaillée de ces approches peut être trouvée dans (Alphonse, 2003).

La formalisation de ces approches en deux étapes ouvre des perspectives intéressantes. D'une part, elle unifie l'ensemble de ces techniques en mettant l'accent sur la façon dont elles construisent les macro-opérateurs. De plus, elle en propose une motivation, celles-ci étant encore vues dans la littérature comme des approches ad hoc, comme un moyen d'améliorer la recherche heuristique en PLI. D'autre part, elles apparaissent être un cas particulier de résolution des problèmes de PLI par macro-opérateurs. Elles définissent des macros pour cibler le langage déterminé, mais nous pouvons développer de nouvelles approches en ciblant des langages plus expressifs. Se faisant, nous obtenons différents compromis entre le coût de construction des macros et le coût d'apprentissage. Particulièrement, nous allons montrer à la section suivante que la construction de macros pour le langage déterminé est aussi coûteux que les approches ILP «classiques» dans des cas très simples.

Bien que l'ensemble des langages entre le langage déterminé, actuellement utilisé, et les restrictions de la logique du premier ordre peuvent être utilisés comme langage cible, nous nous restreignons uniquement aux langages relationnels où des reformulations sous forme de vecteurs, c'est-à-dire AV et multi-instance, ont été proposées. Ces reformulations, étudiées par plus chercheurs (Cohen, 1993; Zucker & Ganascia, 1994;

Fensel *et al.*, 1995; Sebag & Rouveirol, 2000) sont des extensions de la reformulation montrée dans le tableau 1 à des langages plus expressifs. La restriction du choix de langage à ces langages particuliers est motivée par le fait que des algorithmes d'apprentissage dans ce cadre sont bien connus et efficaces. Nous pouvons espérer alors obtenir de meilleurs compromis, comme le démontre empiriquement les approches de propositionnalisation discutées plus haut, qui se sont montrées compétitives par rapport aux systèmes de PLI «classiques».

Par manque de place, nous n'allons que brièvement formaliser ces approches (voir (Alphonse, 2003) pour une présentation détaillée). Nous nous référerons au changement de représentation en utilisant le terme de propositionnalisation. Notons que nous utilisons le terme propositionnalisation de façon plus spécifique que (Kramer *et al.*, 1998), étant donné qu'il ne fait seulement référence qu'au changement de représentation, indépendamment de l'utilisation des macro-opérateurs.

2.1 Propositionnalisation d'un problème de PLI

Un pré-requis pour déléguer la résolution d'un problème de PLI à un algorithme d'apprentissage AV ou multi-instance est que l'espace de recherche de PLI initial soit isomorphe à celui de l'algorithme AV ou multi-instance. Dans ce qui suit, nous représentons cette condition en définissant l'espace d'hypothèses par ce que nous appelons une base d'hypothèses (par analogie avec une base vectorielle), notée $\mathcal{B}$.

Définition 2.2 (base d'hypothèses)
Une base d'hypothèses, notée $\mathcal{B}$, est de la forme :

$$\mathcal{B} = (tc \leftarrow < l_1, \ldots, l_n, v_1, \ldots, v_m >)$$

avec $l_1, \ldots, l_n$ un ensemble de littéraux et $v_1, \ldots, v_m$ un ensemble de variables contraintes dans $l_1, \ldots, l_n$. Chaque hypothèse a pour tête tc et son corps est représenté par un vecteur dans $\mathcal{B}$.

Par construction, il existe un isomorphisme entre une hypothèse de l'espace de recherche de PLI et une hypothèse dans l'espace AV. Pour effectuer la recherche dans l'espace AV, l'espace des instances a besoin d'être reformulé afin d'émuler le test de couverture. Le changement de représentation, comme montré dans le tableau 1, câble le test de couverture entre les hypothèses et les exemples d'apprentissage en calculant l'ensemble des appariements possibles entre $\mathcal{B}$ et les exemples. Si le langage est déterminé, il y aura seulement un appariement et ainsi un exemple sera reformulé en un seul vecteur AV. Dans le cas d'un langage indéterminé, un exemple sera reformulé en autant de vecteurs qu'il y a d'appariements possibles. Le fait que le nouvel espace des instances dans ce dernier cas soit un problème multi-instance vient de la définition du test de couverture en PLI : pour couvrir un exemple positif nous devons trouver un appariement et pour rejeter un exemple négatif, nous ne devons trouver aucun appariement possible.

2.2 Approche générale par macro-opérateurs

Nous donnons maintenant un algorithme général pour la résolution de problèmes de PLI par macro-opérateurs :

1. Choisir le langage $\mathcal{L}_{\mathcal{B}}$ dans lequel sera exprimée la base des hypothèses $\mathcal{B}$.
2. Construire un ensemble de macro-opérateurs pour définir $\mathcal{B}$ dans $\mathcal{L}_{\mathcal{B}}$.
3. Réaliser la propositionnalisation des exemples d'apprentissage
4. Appliquer un algorithme d'apprentissage dédié à la représentation du nouveau problème d'apprentissage obtenu.
5. Réécrire la théorie du domaine dans le langage des hypothèses du problème de PLI initial[1].

Le gain que peuvent apporter les langages plus expressifs que le langage déterminé à l'approche par macro-opérateurs est compensé par le coût de reformulation des problèmes de PLI par propositionnalisation. En effet, la propositionnalisation des langages indéterminés engendre des problèmes multi-instances de taille exponentielle étant donné que le test de couverture est NP-complet dans ces langages. Un contrôle du coût de propositionnalisation doit être fait en utilisant des langages avec un indéterminisme borné. Nous utilisons ceux proposés par (Cohen, 1993). Les différents compromis attendus entre le coût de construction des macro-opérateurs et celui de l'apprentissage délégué par propositionnalisation sont présentés à la figure 2.

À un extrême des cas possibles, nous avons les systèmes de PLI «classiques». En effet, nous pouvons voir une théorie clausale comme un ensemble de définitions de macro-opérateurs[2], et une propositionnalisation peut être faite. L'apprentissage est alors trivial, un nouvel exemple étant classé positif si au moins un de ces attributs est vrai, c'est-à-dire si au moins une des clauses le couvre, et négatif dans le cas contraire. À l'opposé, nous avons les approches de (Fensel *et al.*, 1995; Sebag & Rouveirol, 2000) qui ne construisent aucun macro-opérateur, ou plutôt, dans notre cadre, associent directement un macro à un littéral. Le coût d'apprentissage est alors entièrement délégué à un algorithme multi-instance qui doit travailler sur des données de taille exponentielle dans le cas général.

Entre ces deux extrêmes, nous voyons différentes classes d'approches qui dépendent de l'expressivité du langage de représentation de la base des hypothèses (construites à partir des macros). Dans la suite de l'article, nous discutons d'abord des limitations du langage déterminé, et ensuite évaluons l'intérêt des langages que nous proposons, et particulièrement le langage k-locale qui est la première relaxation prometteuse du langage déterminé.

[1]Cette transformation n'est pas toujours possible en fonction du type d'algorithme utilisé, comme les réseaux de neurones par exemple. Le classement des futures instances doit alors se faire dans l'espace propositionnalisé.

[2]Cette approche a d'ailleurs été utilisée en Invention de Prédicats au début de la PLI, et raffinée récemment par (Khan *et al.*, 1998) (*repeat learning framework*).

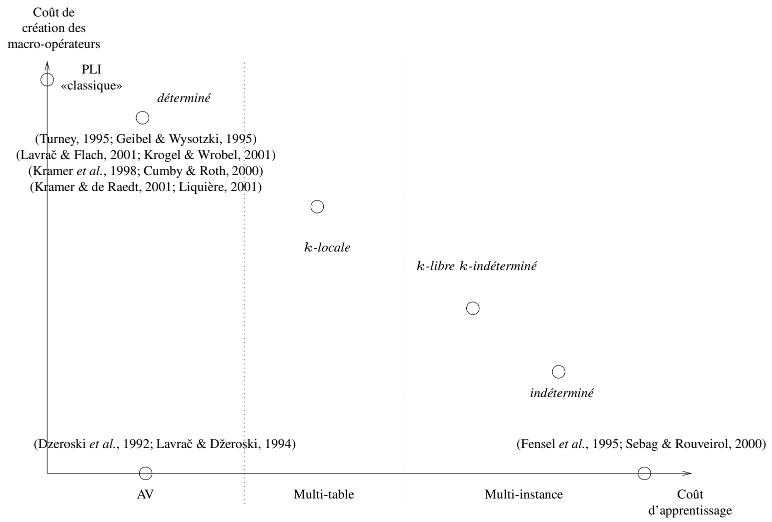

FIG. 2 – Compromis entre coût de création des macro-opérateurs et coût d'apprentissage

3 Limitations du langage déterminé

Intuitivement, le coût de construction de macro-opérateurs pour obtenir une représentation cohérente des données dans le langage déterminé est très élevé. Il vient du fait que la nouvelle représentation obtenue est centrée sur l'exemple, vu comme un tout (ce que (Lavrač & Flach, 2001) a appelé *individual-centered representation*), c'est-à-dire que les macros doivent représenter des propriétés globales de l'exemple. Si nous regardons un exemple de macro pour le langage déterminé :

voiture_courte_toit(T) :− voiture(T,V1),courte(V1),toit(V1).

Nous voyons qu'il exprime une propriété du train, de l'exemple vu comme un tout : le train a la propriété d'avoir une voiture courte qui a un toit. Cette restriction a des impacts importants sur le coût du calcul des macro-opérateurs.

3.0.1 Complexité de l'apprentissage numérique

Une limitation importante est la capacité d'apprentissage numérique. Dans la représentation «centrée sur l'individu», nous pouvons avoir un macro décrivant le nombre total de charges portées par le train, mais pas celui d'une voiture quelconque du train. Autrement dit, la partie de l'apprentissage numérique qui est déléguée à l'algorithme d'apprentissage ne peut reposer que sur des propriétés de l'exemple en entier. Par exemple, le système RELAGGS (Krogel & Wrobel, 2001) calcule un certain nombre de statistiques comme la moyenne et le maximum des valeurs numériques apparaissant dans l'exemple.

Pour apprendre un concept contenant des contraintes sur les valeurs numériques de parties de l'exemple, comme les voitures du train, l'algorithme de PLI doit être capable d'engendrer un macro de la forme :

$$voiture\_2+\_charges(T):-voiture(T,V1), nb\_charges(V1,N), N \geq 3.$$

La solution à ce problème est de déléguer la complexité de l'apprentissage numérique aux algorithmes multi-instance qui bénéficient de techniques plus avancées et robustes que la PLI (Zucker & Ganascia, 1994; Sebag & Rouveirol, 1994). Les variables numériques sont alors introduites dans la base des hypothèses et l'ensemble de leurs instanciations possibles parmi les exemples sont collectées sous forme tabulaire.
À titre d'exemple, l'apprentissage de contraintes sur le nombre de charge d'une voiture est permis par la définition du macro-opérateur :

$$charges\_voiture(T,N):-voiture(T,V1), nb\_charges(V1,N)$$

La base associée $\mathcal{B} = (ouest(T) \leftarrow (charges\_voiture(T,N), N))$ donne la reformulation suivante sur un problème hypothétique :

	Variables	Attributs	
	T	$charges\_voiture(T,N)$	N
e^+	$t1$	V	1
	$t1$	V	3
	$t1$	V	2
e^-	$t2$	V	1
	$t2$	V	2
	$t2$	V	2

Un algorithme multi-instance peut alors inférer la définition de concept :

$$ouest(T) \leftarrow charges\_voiture(T,N), N \geq 3$$

3.1 Taille des macro-opérateurs

Une deuxième limitation est que certains concepts ne peuvent être appris par l'algorithme AV et doivent donc être représentés par un unique macro. Le coût de calcul des macros devient alors *identique* au coût d'un algorithme de PLI classique. Considérons que nous avons à apprendre le concept suivant :

$$ouest(T) \leftarrow v(T,V1), rect(V1), v(T,V2), rect(V2), V1 \neq V2$$

Le concept est qu'un train a deux voitures rectangulaires différentes. Nous pouvons voir qu'il ne peut être appris en définissant des macros ayant comme définition un sous-ensemble des littéraux du concept. Le seul moyen pour prévenir les instanciations des deux voitures sur une même voiture d'un exemple est de former l'unique macro :

$$2\_voitures\_rect(T):-v(T,V1), rect(V1), v(T,V2), rect(V1), V1 \neq V2.$$

En relaxant la contrainte de déterminisme, nous pouvons créer deux macros moins complexes, définissant chacun une voiture rectangulaire, et obtenir un problème multi-instance par propositionnalisation.

Nous voyons que dans ces deux cas l'utilisation d'un langage de représentation plus riche de la base permet une meilleure exploitation des techniques d'apprentissage et ainsi une simplification de la créations des macros.

4 Le langage k-locale

Dans ce langage, la base des hypothèses est une k-locale.

Définition 4.1 (base k-locale)
Soit la base des hypothèses k-locale :

$$\mathcal{B} = (cc \leftarrow (LOC_1, \ldots, LOC_n))$$

Nous avons $\forall i \in \{1, \ldots, n\}, LOC_i$ *est une locale et* $k \geq vars(LOC_i)$

Une locale est définie après (Cohen, 1993), comme un ensemble de littéraux où leurs variables libres (existentielles) n'apparaissent pas dans d'autres locales. Ce langage permet d'introduire un indéterminisme bornée au niveau de chaque locale de $\mathcal{B}$. Par exemple, $\mathcal{B}$ est une 2-locale avec ses 2-locales soulignées :

$$\mathcal{B} = (ouest(T) \leftarrow \quad (\underline{voiture\_courte(T, V1), toit(C1)}, \\ \underline{voiture\_courte(T, V2), nb\_roues(V2, N), N}))$$

Par définition de la localité, lors de la propositionnalisation d'un problème de PLI avec la base $\mathcal{B}$, l'instanciation de la variable $V1$ n'influence pas l'instanciation de la variable $V2$. Il est alors plus efficace de représenter le nouvel espace des instances par une représentation produit. Considérons le problème des trains suivant :

Exemples	Connaissance du domaine		
e^+ $ouest(t1)$	$voiture(t1, v11)$.	$toit(v11)$.	$rect(v11)$.
e^- $ouest(t2)$	$voiture(t1, v12)$.	$courte(v12)$.	$nb\_roues(v12, 2)$.
	$voiture(t2, v21)$.	$courte(v21)$.	$toit(v21)$.
	$voiture(t2, v22)$.	$rect(v22)$.	$nb\_roues(v22, 3)$.

La propositionnalisation doit prendre en compte les k-locales et engendrer pour chacune d'elles un problème multi-instance propre comme nous le montre la figure 3. Cette représentation du problème n'a que deux lignes au maximum par exemple, au lieu de quatre dans sa représentation développée qui est obtenue par l'algorithme de propositionnalisation naïf de la section 2.1. Cette représentation obtenue par décomposition d'un problème multi-instance est une représentation plus simple que la représentation multi-instance (Dietterich *et al.*, 1997). Nous appelons cette nouvelle représentation la *représentation multi-table*, chaque table étant un problème multi-instance. Aucun algorithme d'apprentissage n'a encore été développé pour cette représentation, mais il est facile d'extrapoler étant donné qu'elle est plus simple que la représentation multi-instance : l'espace d'hypothèse est le même que pour l'apprentissage AV, seul le test de couverture change. Les algorithmes de type «générer-tester» descendants peuvent être facilement adaptés à ce type de langage (Zucker & Chevaleyre, 1998; de Raedt, 1998).

	Variables	Attributs	
	$V1$	$v\_c(T,V1)$	$toit(V1)$
e^+	$v11$	F	V
	$v12$	V	F
e^-	$v21$	V	V
	$v22$	F	F

$\times$

	Variables	Attributs		
	$V2$	$v\_c(T,V2)$	$nb\_roues(V2,N)$	N
e^+	$v11$	F	F	-
	$v12$	V	V	2
e^-	$v21$	V	F	-
	$v22$	F	V	3

	Variables		Attributs				
	$V1$	$V2$	$v\_courte(T,V1)$	$toit(V1)$	$v\_courte(T,V2)$	$nb\_r(V2,N)$	N
e^+	$v11$	$v11$	F	V	F	F	-
	$v11$	$v12$	F	V	V	V	2
	$v12$	$v11$	V	F	F	F	-
	$v12$	$v12$	V	F	V	V	2
e^-	$v21$	$v21$	V	V	V	F	-
	$v21$	$v22$	V	V	F	V	3
	$v22$	$v21$	F	F	V	F	-
	$v22$	$v22$	F	F	F	V	3

FIG. 3 – Une représentation multi-table et sa représentation développée

Une restriction particulièrement intéressante de la représentation multi-table est de borner le nombre d'attributs des tables par une constante l (le nombre de table étant toujours quelconque). Cette restriction correspond, dans le cas de la propositionnalisation, à borner la taille des locales de la base des hypothèses. Dans ce langage, nous montrons que l'apprentissage d'un monôme est PAC, et donc que des algorithmes efficaces existent. Suivant (Cohen, 1993), ce langage de concept conjonctif est noté $\mathcal{L}^1_{l-MULTI-TABLE}$.

Théorème 1
Pour tout l fixé, la famille de langages $\mathcal{L}^1_{l-MULTI-TABLE}$ avec n tables, n quelconque, est PAC-apprenable.

Preuve 1

Lemme 1 (d'après (Pitt & Warmuth, 1990))
Soient $\mathcal{L}_{h1}$ et $\mathcal{L}_{h2}$ deux langages d'hypothèses et $\mathcal{L}_{e1}$ et $\mathcal{L}_{e2}$ leur langage des exemples associés, $C \in \mathcal{L}_{h1}$ le concept cible. Soient les fonctions de réduction $f_i : \mathcal{L}_{e1} \rightarrow \mathcal{L}_{e2}$ et $f_c : \mathcal{L}_{h1} \rightarrow \mathcal{L}_{h2}$.

Si $\mathcal{L}_{h1}$ se réduit à $\mathcal{L}_{h2}$ et $\mathcal{L}_{h2}$ est PAC-apprenable et de plus ${f_c}^{-1}(C)$ est calculable en temps polynomial alors $\mathcal{L}_{h1}$ est PAC-apprenable.

En nous appuyant sur le lemme 1, nous montrons que pour n'importe quel l fixé $\mathcal{L}^1_{l-MULTI-TABLE}$ se réduit à $\mathcal{L}_{MONÔME}$, $\mathcal{L}_{MONÔME}$ le langage PAC-apprenable des monômes, et que f_c est inversible en temps polynomial, ce qui conclut la preuve.

Nous considérons que toutes les tables ont l attributs, éventuellement complétées par des attributs booléens neutres pour l'apprentissage dont la valeur est la même pour *tous les exemples. Nous construisons une représentation booléenne du problème multi-table en associant à chaque hypothèse de l'espace de recherche un attribut booléen. Comme l'espace des hypothèses est un produit direct des espaces d'hypothèses de chaque table, la représentation booléenne a* $n \times 2^{2l}$ *attributs. Pour cela, pour chaque table* T_i *contenant les* l *attributs booléens* $t_{i1}, \ldots, t_{il}$, *nous définissons un ensemble de* $2l$ *attribut booléens* $A_i = \{a_{i1}, \ldots, a_{i2l}\}$, *chaque attribut représentant un des* $2l$ *attributs* $t_{i1}, \ldots, t_{il}, \neg t_{i1}, \ldots, \neg t_{il}\}$. *Pour chaque ensemble* A_i, *nous définissons un ensemble* B_i *de* 2^{2l} *attributs, chaque attribut correspondant à une hypothèse de* T_i *construite sur les attributs de* A_i.

f_i : *Chaque exemple est redécrit avec les* $n \times 2^{2l}$ *attributs* $(B_1, \ldots, B_n)$ *où un attribut prend la valeur vraie si l'hypothèse correspondante couvre l'exemple et fausse sinon.*

f_c : *Un concept est de la forme* $(c_1, \ldots, c_n)$. *Par construction pour chaque* c_i *il existe un attribut* $b \in B_i$ *qui lui correspond et* c_i *couvre l'exemple ssi* b *prend la valeur vraie.* f_c *préserve l'extension du concept et de plus est facilement inversible en temps polynomial.*

5 Langages k-libre et l-indéterminé

Deux autres extensions naturelles sont les langages k-libre et l-indéterminé. Dans le premier langage la taille du problème multi-instance est exponentiel en k ($\mathcal{B}$ peut être vue comme une seule locale de taille k), dans le second la taille est linéaire en l.

Ces deux langages sont d'une utilisation complémentaire. Le langage k-libre donne une borne théorique sur la taille, alors que le langage l-indéterminé permet de prendre en compte la connaissance du domaine. Ainsi, l'utilisateur peut tirer parti du typage des variables afin d'obtenir une borne plus fine.

À titre d'exemple, une utilisation simple, et efficace à notre avis, de ces langages, fondée sur les langages de biais proposés dans le cadre déterminé, serait d'introduire l'indétermination uniquement au niveau des variables numériques (dans le cas des macros, elles participent à la complexité de l'appariement).

6 Conclusion

Il est connu que la PLI doit faire face à des phénomènes de plateau pendant la recherche heuristique et les macro-opérateurs sont bien motivés pour réduire ces problèmes. Nous avons proposé un paradigme de résolution des problèmes de PLI par macro-opérateurs. Contrairement aux approches précédentes, les macros ne sont pas employés pour enrichir le graphe de raffinement mais pour sélectionner seulement un sous-graphe pertinent. Nous avons montré que nous pouvions associer un langage de représentation au sous-graphe et étudier une approche par macro-opérateur en fonction de l'expressivité de celui-ci. Nous avons montré qu'un grand nombre de travaux en PLI (Turney, 1995; Geibel & Wysotzki, 1995; Kramer *et al.*, 1998; Cumby & Roth, 2000; Lavrač & Flach, 2001; Krogel & Wrobel, 2001; Roth & Yih, 2001; Kramer & de

Raedt, 2001; Liquière, 2001) était un cas particulier du paradigme. Ils définissent des macros pour reformuler les problèmes de PLI dans un langage déterminé et délèguent l'apprentissage par propositionnalisation à des algorithmes AV, adaptés aux langages déterminés. Cependant, leur coût de construction des macros est dans le pire des cas aussi élevé de celui de la résolution du problème de PLI initial par des systèmes de PLI «classiques». Nous avons alors proposé un panel de langages plus expressifs, mais d'indéterminisme borné, empruntées à (Cohen, 1993), qui permettent de meilleurs compromis entre le coût de construction des macros et celui de l'apprentissage.

La première extension prometteuse est le langage k-locale qui reformule les problèmes de PLI dans un nouveau langage de représentation que nous avons appelé *multi-table*. C'est une représentation produit d'un problème multi-instance qui représente la même information de manière plus compacte. De plus, nous avons montré que dans une restriction faible, où le nombre d'attributs par table est fixé, les monômes sont PAC-apprenable, ce qui suggère que des algorithmes efficaces existent. Une telle restriction apparaît naturellement dans le cadre de la propositionnalisation si nous limitons la taille de chaque locale dans l'espace des hypothèses construit à partir des macros. Nous pensons que la représentation multi-table dépasse le cadre de son utilisation en PLI et peut être employée en tant que langage de représentation en apprentissage.

De notre point de vue, l'important effort mis dans les techniques de propositionnalisation durant les dix dernières années peut être poursuivi dans des langues plus riches qui repoussent leurs limitations actuelles (par exemple en apprentissage numérique). La PLI peut tirer parti de développements réalisés dans la communauté de Résolution de Problèmes en réutilisant les techniques de constructions automatiques de macro-opérateurs et en déléguant l'apprentissage à des algorithmes plus simples travaillant sur les représentations AV, multi-table ou multi-instance. Afin de valider cette idée, nous travaillons sur un algorithme d'apprentissage multi-table et nous projetons de développer des techniques de construction de macro-opérateurs pour le langage k-locale.

Remerciements

Nous remercions Céline Rouveirol pour les nombreuses discussions sur les techniques de propositionnalisation et de construction des macro-opérateurs en PLI. Également, nous tenons à remercier Claire Nédellec pour son soutient tout au long de la rédaction de cet article, ainsi que les relecteurs pour leur remarques et conseils.

Références

ALPHONSE E. (2003). *Macro-opérateurs et Sélection Relationnelle en Programmation Logique Inductive : théorie et algorithmes*. PhD thesis, Thèse de doctorat de l'Université Paris-Sud.

AMAREL S. (1968). *On Representations of Problems of Reasoning about Actions*, In *Machine Intelligence 3*, p. 131–171. American Elsevier : New York, NY.

CHEESEMAN P., KANEFSKY B. & TAYLOR W. M. (1991). Where the really hard problems are. In R. MYOPOULOS, JOHN ; REITER, Ed., *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, p. 331–340, Sydney, Australia : Morgan Kaufmann.

COHEN W. W. (1993). Learnability of restricted logic programs. In S. MUGGLETON, Ed., *Proceedings of the 3rd International Workshop on Inductive Logic Programming*, p. 41–72 : J. Stefan Institute.

CUMBY C. & ROTH D. (2000). Relational representations that facilitate learning. In A. G. COHN, F. GIUNCHIGLIA & B. SELMAN, Eds., *Proceedings of the Conference on Principles of Knowledge Representation and Reasoning (KR-00)*, p. 425–434, S.F. : Morgan Kaufman Publishers.

DE RAEDT L. (1998). Attribute-value learning versus inductive logic programming : The missing link. In D. PAGE, Ed., *Proc. of the 8th International Workshop on Inductive Logic Programming*, p. 1–8 : Springer Verlag. Extended abstract.

DIETTERICH T. G., LATHROP R. H. & LOZANO-PÉREZ T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, **89**(1–2), 31–71.

DZEROSKI S., MUGGLETON S. & RUSSELL S. J. (1992). PAC-learnability of determinate logic programs. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory (COLT-92)*, Pittsburgh, Pennsylvania : ACM Press.

FENSEL D., ZICKWOLFF M. & WIESE M. (1995). Are substitutions the better examples ? Learning complete sets of clauses with Frog. In *Proceedings of the 5th International Workshop on Inductive Logic Programming*, p. 453–474.

FÜRNKRANZ J. & FLACH P. (2003). *An Analysis of Rule Learning Heuristics*. Rapport interne CSTR-03-002, Department of Computer Science, University of Bristol.

GEIBEL P. & WYSOTZKI F. (1995). Learning context dependent concepts. In L. D. RAEDT, Ed., *Proceedings of the 5th International Workshop on Inductive Logic Programming*, p. 323–337 : Department of Computer Science, Katholieke Universiteit Leuven.

GIORDANA A. & SAITTA L. (2000). Phase transitions in learning relations. *Machine Learning*, **41**, 217–25.

HAUSSLER D. (1989). Learning conjunctive concepts in structural domains. *Machine Learning*, **4**(1), 7–40.

HERNÁDVÖLGYI I. T. (2001). Searching for macro operators with automatically generated heuristics. *Lecture Notes in Computer Science*, **2056**, 194– ? ?

KEARNS M. J. & VAZIRANI U. V. (1994). *An Introduction to Computational Learning Theory*. Cambridge, Massachusetts : The MIT Press.

KHAN K., MUGGLETON S. & PARSON R. (1998). Repeat learning using predicate invention. In C. PAGE, Ed., *Proc. of the 8th International Workshop on Inductive Logic Programming (ILP-98)*, LNAI 1446, p. 165–174, Berlin : Springer-Verlag.

KORF R. E. (1985). Macro-operators : a weak method for learning. *Artificial Intelligence, 1985*, **26**, 35–77.

KRAMER S. & DE RAEDT L. (2001). Feature construction with version spaces for biochemical applications. In *Proc. 18th International Conf. on Machine Learning*, p. 258–265 : Morgan Kaufmann, San Francisco, CA.

KRAMER S., PFAHRINGER B. & HELMA C. (1998). Stochastic propositionalization of non-determinate background knowledge. In D. PAGE, Ed., *Proc. of the 8th International Workshop on Inductive Logic Programming*, p. 80–94 : Springer Verlag.

KROGEL M.-A. & WROBEL S. (2001). Transformation-based learning using multirelational aggregation. In *Proceedings of the 11th International Conference on Inductive Logic Programming*, volume 2157 of *LNAI*, p. 142–155 : Springer-Verlag.

LAVRAČ N. & DŽEROSKI S. (1994). *Inductive Logic Programming : techniques and Applications*. Ellis Horwood.

LAVRAČ N. & FLACH P. A. (2001). An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic*, **2**(4), 458–494.

LIQUIÈRE M. (2001). Du structurel au propositionnel, une approche formelle. In *CAP*, Grenoble.

MICHALSKI R. S. (1978). *Pattern Recognition as Knowledge-Guided Computer Induction*. Rapport interne 927, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

MITCHELL T. M. (1982). Generalization as search. *Artificial Intelligence*, **18**, 203–226.

NIENHUYS-CHENG S.-H. & DE WOLF R. (1997). *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

PEÑA CASTILLO L. & WROBEL S. (2002). Macro-operators in multirelational learning : a search-space reduction technique. In *Proceedings of the 13th European Conference on Machine Learning*, volume 2430 of *Lecture Notes in Artificial Intelligence*, p. 357–368 : Springer-Verlag.

PEARL J. (1985). *Heuristics*. Reading : Addison-Wesley.

PITT L. & WARMUTH M. K. (1990). Prediction preserving reducibility. *J. of Comput. Syst. Sci.*, **41**(3), 430–467. Special issue of the for the *Third Annual Conference of Structure in Complexity Theory* (Washington, DC., June 88).

QUINLAN J. R. (1991). Determining literals in inductive logic programming. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, p. 746–750, Sydney, Austalia.

RICHARDS B. & MOONEY R. (1992). Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, p. 723–738.

ROTH D. & YIH W. (2001). Relational learning via propositional algorithms : An information extraction case study. In B. NEBEL, Ed., *Proceedings of the seventeenth International Conference on Artificial Intelligence (IJCAI-01)*, p. 1257–1263, San Francisco, CA : Morgan Kaufmann Publishers, Inc.

SEBAG M. & ROUVEIROL C. (1994). Induction of maximally general clauses consistent with integrity constraints. In S. WROBEL, Ed., *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237 of *GMD-Studien*, p. 195–216 : Gesellschaft für Mathematik und Datenverarbeitung MBH.

SEBAG M. & ROUVEIROL C. (2000). Resource-bounded relational reasoning : Induction and deduction through stochastic matching. *Machine Learning*, **38**(1/2), 41–62.

SILVERSTEIN G. & PAZZANI M. J. (1991). Relational cliches : Constraining constructive induction during relational learning. In L. BIRNBAUM & G. COLLINS, Eds., *Proceedings of the 8th International Workshop on Machine Learning*, p. 203–207 : Morgan Kaufmann.

TURNEY P. (1995). Low size-complexity inductive logic programming : The East-West challenge considered as a problem in cost-sensitive classification. In *Proceedings of the 5th International Workshop on Inductive Logic Programming*, p. 247–263.

ZUCKER J.-D. (2001). Changements de représentation, abstractions et apprentissages. HDR.

ZUCKER J.-D. & CHEVALEYRE Y. (1998). Comprendre et résoudre les problèmes d'apprentissage multi-instances et multi-parties. In *Journées Francophone d'Apprentissage*.

ZUCKER J.-D. & GANASCIA J.-G. (1994). Selective reformulation of examples in concept learning. In *Proc. 11th International Conference on Machine Learning*, p. 352–360 : Morgan Kaufmann.