Programmation Logique Inductive et génération automatique d'un modèle des croyances de l'apprenant

MÉMOIRE

Soutenu le 4 septembre 1995

Pour l'obtention du

DEA de l'Université Henri Poincaré - Nancy I
(spécialité Informatique)

par

Nicolas VAN LABEKE

Table des Matières

Table des Figures

Introduction

Notre travail se place dans la cadre de la conception et de la réalisation du Tuteur d'Aide Logique pour la Construction de figures géométriques TALC. L'objectif de ce tuteur est de diagnostiquer la correction d'une figure construite par un élève vis-à-vis d'une spécification fournie par le professeur. Ses caractéristiques principales sont d'une part d'autoriser le professeur à définir lui-même les exercices, les connaissances géométriques requises et les éléments de construction autorisés pour les résoudre, d'autre part de disposer d'un démonstrateur automatique complet et conforme aux exigences de l'interactivité. TALC fournit actuellement trois types de diagnostics : la figure est correcte, la figure ne contient pas tous les objets et/ou propriétés spécifiées, la figure contient des propriétés non déductibles de la spécification (figure particulière).
Dans le cas d'une construction incorrecte, la difficulté consiste à fournir à l'élève des explications pertinentes sur les raisons de cette inadéquation. Afin de fournir de telles explications, plusieurs améliorations du diagnostic actuel sont envisageables.

La première consiste à générer un diagnostic dégageant précisément les objets et propriétés de la construction cause de l'incorrection. En effet le diagnostic actuel fait référence à la traduction interne de la construction qui ne correspond pas exactement à la représentation qu'a l'élève des objets de sa construction [Person 95].

Une autre amélioration possible, présentée dans ce document, est basée sur l'hypothèse didactique de la cohérences des raisonnements de l'élève et consiste à mettre en évidence les conceptions de la géométrie propres à l'élève. Du fait de la représentation des connaissances géométriques de référence dans TALC - et donc de la représentation des connaissances de l'élève - sous forme logique de clauses de Horn, cette modélisation des croyances de l'apprenant revient à trouver un modèle logique cohérent avec les productions de l'élève en vue de fournir une base d'explication et de négociation entre l'apprenant, le professeur et le système.
Pour automatiser la recherche d'un tel modèle, l'utilisation de l'apprentissage automatique à partir d'exemples s'avère approprié. La représentation logique des connaissances nous a amené plus particulièrement à nous intéresser à un sous-domaine de recherche de l'apprentissage automatique: la Programmation Logique Inductive (PLI).
La méthode utilisée pour ce travail est l'étude de la réutilisation d'un système de PLI aisément disponible pour l'inférence d'un modèle des croyances de l'apprenant.

Dans une première partie (*Le cadre de recherche*), nous proposons une brève présentation du contexte de la recherche, c'est-à-dire de l'Apprentissage Symbolique Automatique et des Environnements Interactifs d'Apprentissage avec Ordinateur en accentuant dans ce dernier domaine. Nous présentons aussi le cadre de recherche que constitue le Tuteur d'Aide Logique à la Construction de figure géométrique TALC.

Le deuxième chapitre de ce document (*Introduction à la Programmation Logique Inductive*) est une présentation théorique de la PLI. Le principe de base est présenté, ainsi que l'intérêt d'une telle approche de l'apprentissage automatique à partir d'exemples. Nous décrivons aussi quelques techniques de base, essentiellement fondées sur la notion d'ordre de généralité entre clauses, et la structure générale d'un algorithme d'induction.

Le troisième chapitre (*Evaluation de systèmes de Programmation Logique Inductive*) est consacré à une évaluation des caractéristiques de certains systèmes de PLI. Nous nous sommes intéressé plus aux aspects qualitatifs que quantitatifs de ces systèmes, notre objectif pour ce travail étant d'utiliser un système de PLI comme une *'boîte noire'* pour générer notre modèle. Nous présentons particulièrement quatre de ces systèmes.

Dans le quatrième et dernier chapitre de ce document (*PLI et génération d'un modèle des croyances de l'apprenant*), nous proposons une approche de la génération d'un modèle des croyances de l'apprenant dans TALC basée sur une application de la PLI. Nous présentons d'abord les principes et les objectifs d'un tel modèle

puis proposons une implémentation dans le cadre de TALC avec l'un des systèmes présenté dans le chapitre précédent. Nous concluons ce chapitre par une mise en évidence des problèmes posés par cette modélisation.

Par l'utilisation de la Programmation Logique Inductive (PLI) pour générer un tel modèle, notre travail se situe à la jonction de l'Apprentissage Symbolique Automatique (ASA) et des Environnements Interactifs d'Apprentissage avec Ordinateur (EIAO).

Le cadre de recherche

1 Introduction

Ce chapitre a pour objectif de replacer notre travail dans son contexte.
La première partie est consacrée à une présentation des différentes approches de l'Apprentissage Symbolique Automatique, de manière à identifier la place qu'y occupe la Programmation Logique Inductive.
Nous présentons ensuite les Environnements Interactifs d'Apprentissage avec Ordinateur. Nous y faisons apparaître les approches de la modélisation de l'apprenant par les techniques d'ASA.
Nous concluons ce chapitre par une présentation du cadre de travail que constitue TALC, Tuteur d'Aide Logique à la Construction de figures géométriques.

2 Apprentissage Symbolique Automatique.

Les sources bibliographiques de ce paragraphes proviennent de [Kod 88] et [MKCM 92].
Le terme d'Apprentissage Symbolique Automatique (ASA, traduction proposée par Y. Kodratoff [Kod 88] pour l'expression *"Machine Learning : an Artificial Intelligence Approach"*) regroupe les différents sous-domaines de l'Intelligence Artificielle dont l'objectif principal est l'acquisition automatique de connaissances. L'idée est de remplacer l'instructeur humain quant à l'introduction de ces connaissances (exemples, faits, descriptions, ...) dans les systèmes concernés.
Le grand attrait de l'ASA est que ce domaine offre une diversité considérable de tâches et de terrains pour l'expérimenter. Cette diversité est due au fait que l'apprentissage peut être nécessaire dans de multiples domaines de l'Intelligence Artificielle, et ceci quelle que soit la tâche principale de ces domaines (classification, résolution de problèmes, vision, ...).
Il faut signaler que l'Intelligence Artificielle n'est pas la seule approche de l'acquisition automatique de connaissances : les systèmes adaptatifs de l'automatique, l'inférence grammaticale, la reconnaissance des formes sont autant d'exemples d'un tel processus.

D'une manière générale, les méthodes d'ASA peuvent se repartir en deux grands groupes selon le but principal de l'apprentissage et le mode principal d'inférence utilisé dans cet objectif : il s'agit de l'apprentissage *analytique* et de l'apprentissage *synthétique*.

L'apprentissage *analytique* vise principalement à mettre les connaissances données sous une forme plus appropriée et utilise la déduction comme mode d'inférence. Il s'agit d'un apprentissage guidé par les exemples. La recherche dans cette branche se développe surtout sur l'*apprentissage à partir d'explications* (Explanation-Based Learning, [DM 86]) qui consiste à expliquer un exemple observé en termes des connaissances du domaine et à utiliser cette explication pour construire un description plus 'opérationnelle'.

L'apprentissage *synthétique* vise plutôt à créer des connaissances nouvelles ou meilleures en utilisant l'induction comme mode d'inférence. Il s'agit d'un apprentissage à partir d'exemples. Les connaissances du domaine y joue un rôle important et permettent de classer les méthodes synthétiques en deux sous-groupes. L'apprentissage inductif est dit *empirique* lorsqu'il s'appuie sur peu de connaissances du domaine (et généralement indépendantes du domaine comme les implications tautologiques) et *constructif* lorsqu'une grande importance est accordée aux connaissances du domaines. Les *réseaux de neurones* et les *algorithmes génétiques* sont des méthodes d'apprentissage empirique inductif car ils s'appuient sur des quantités relativement faibles de connaissances du domaine et le type principal d'inférence est l'induction. Ces deux méthodes sous souvent appelées systèmes d'apprentissages *subsymboliques* car le principe de compréhensibilité des résultats de l'apprentissage n'est pas considéré comme prioritaire dans leur implémentation. Les méthodes d'*inférence inductive* par arbres de décisions (ID3 de [Quinlan 79]) ou par apprentissage de règles (AQ de [Michalski 73]), à l'origine de la PLI, sont aussi des exemples d'induction empirique.

Cette catégorisation des méthodes d'ASA est générale et ne prend en compte que le mode principal d'inférence utilisé. Michalski [MKCM 92] signale que les systèmes d'apprentissage à venir ne doivent pas se concentrer sur une seule stratégie d'apprentissage pour acquérir les connaissances mais doivent en combiner plusieurs guidées par les buts

poursuivis. Un exemple d'un tel processus multistratégique est l'*apprentissage par analogie* (Case-Based Learning, [Carbonell 86]) qui permet le transfert de certaines propriétés des connaissances de la base vers les connaissances cibles de l'apprentissage. Il s'agit d'un exemple d'apprentissage combinant induction (généralisation des propriétés) et déduction (transfert des propriétés).

La Programmation Logique Inductive (voir chapitre suivant), en tant qu'extension des techniques d'induction empirique par prise en compte de la théorie du domaine et l'utilisation de la logique du premier ordre comme langage de description des connaissances, peut donc se classer dans la catégorie des processus d'induction *constructif*.

3 EIAO et modèle de l'apprenant

Le terme EIAO, apparu dans les années 70, fait référence à l'*Enseignement Intelligemment Assisté par Ordinateur* (traduction de l'anglais *Intelligent Computer Aided Instruction*). Il désigne des travaux mettant en oeuvre des techniques d'Intelligence Artificielle pour des applications à l'enseignement plus souples et plus interactives que des sytèmes d'EAO classiques. Les débuts de l'EIAO sont associés au système SCHOLAR [Car 73]. Plusieurs approches ont été explorées et la décennie 80-90 voit apparaître des *Systèmes Tutoriels Intelligents* (*Intelligent Tutoring Systems*), fortement liés au développement des SBC en IA comme GUIDON [Cla 82]. Aujourd'hui, l'accent est mis plus sur l'interaction entre apprenant et système que sur le coté 'intelligent' des EIAO : des applications ne comportant pas de capacités de raisonnements peuvent quand même servir d'environnement d'apprentissage. C'est le cas des micro-mondes dont LOGO [Pap 80] est l'ancêtre direct. Le terme EIAO se décline alors par *Environnements Interactif d'Apprentissage avec Ordinateur* (*Interactive Learning Environments*).
Une caractéristique importante de ce domaine de recherche est son caractère pluridisciplinaire.
Le sigle EIAO désigne aussi deux aspects complémentaires : les EIAO sont des systèmes qui visent à favoriser l'apprentissage d'un domaine de connaissances par un utilisateur (appelé apprenant) tandis que l'EIAO est le domaine de recherches portant sur ces systèmes et situé au confluent de l'informatique et des sciences de l'éducation.

D'une façon générale, un EIAO peut se représenter comme composé d'au moins quatre modules [NV 88] :

- le module *expert du domaine* est la composante qui contient les connaissances du domaine et les mécanismes de raisonnement.
- le module *pédagogique* qui gère l'apprentissage. Celui-ci peut être grossièrement de trois sortes : *directif* quand l'apprenant doit impérativement réaliser le travail donné par le professeur, *libre* quand l'élève à l'initiative, et à *initiative mixte* quand élève et professeur se partagent l'initiative de l'apprentissage.
- le module *interface* avec l'apprenant qui gère la communication entre celui-ci et le système. Cette communication peut se faire en langage naturel comme pour SCHOLAR [Car 73] ou être réduite à des dialogues par mot-clefs ou à des menus

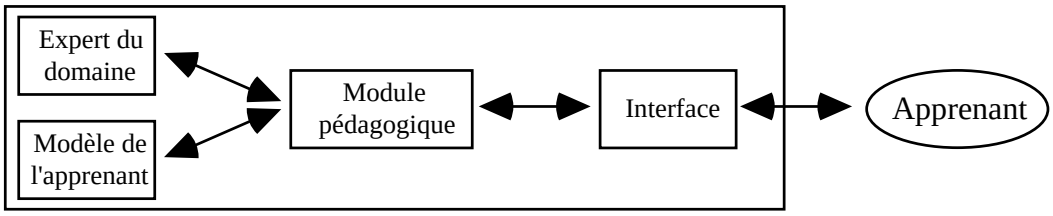


Figure 1 Les composants d'un EIAO

Les frontières entre les différents composants des EIAO ne sont pas figées et, d'un système à l'autre, les modules peuvent avoir d'autres noms, être découpés ou reliés entre eux différemment. Il s'agit cependant d'une présentation rendant compte des différents domaines de recherche en EIAO.
Si la représentation des connaissances du domaine est un problème assez bien maîtrisé, il n'en est pas de même pour celle relative à la pédagogie et à l'apprenant, sur lequel porte notre travail.

3.1 Modélisation de l'apprenant.

Dans un EIAO, l'un des utilisateurs est un *apprenant* dont l'objectif est d'acquérir un ensemble de connaissances relatives à un domaine précis. L'appellation 'modèle de l'apprenant' en EIAO regroupe les travaux dont l'objectif est de disposer d'un ensemble d'informations sur l'apprenant pour permettre une adaptation dynamique et personnalisée des interventions du système.
Ces informations peuvent être de nature diverse : connaissances conceptuelles dans le domaine étudié, connaissances procédurales (ou savoir-faire), stratégies d'apprentissage suivies par l'apprenant, erreurs couramment effectuées, informations sur le comportement de l'élève (assiduité, évolution du savoir, ...), identification de l'élève (âge, niveau social, ...).

La plupart des modèles de l'apprenant s'en tiennent cependant aux connaissances et aux erreurs de celui-ci, les autres informations étant plus facilement exploitables de manière automatique.

Un tel modèle peut servir à de multiples objectifs [Ded 86], [NV 88] :

- évaluer le savoir de l'élève dans le domaine concerné.
- fournir des explications et des conseils personnalisés.
- générer des problèmes et des exercices en vue de compléter et de corriger les connaissances de l'élève.
- aider à caractériser ses erreurs et ses méconnaissances.

La notion de *modèle de l'apprenant* est souvent liée à celle du *diagnostic* qui consiste à inférer les informations du modèle à partir du comportement de l'apprenant, c'est-à-dire à interpréter et à analyser les données recueillies par le système au cours de l'apprentissage. Ce sont les différentes approches de ce diagnostic que nous allons présenter dans le paragraphe suivant.

3.2 Les approches existantes.

Dans les systèmes d'EIAO, les méthodes de modélisation de l'apprenant les plus utilisées selon [NV 88] et [BV 95] sont :

- les modèles par *expertise partielle* (ou *recouvrement*, traduction de l'anglais *overlay*) où les connaissances de l'apprenant sont considérées comme un sous-ensemble de celles de l'enseignant. Cette méthode permet l'analyse d'erreurs en comparant leur savoirs respectifs mais est incomplète car seul le manque de connaissance de l'apprenant est modèlisable. GUIDON [Cla 82] utilise un tel modèle. Les modèles *différentiels* en sont une variante où les faiblesses de l'apprenant sont détectées par un raisonnement obtenu en utilisant les connaissances du système dans le contexte où se trouvait l'élève. C'est la cas de WEST [BB 82] par exemple
- les modèles par *déviation* ou *perturbation* qui consistent à coder les connaissances de l'expert et à les augmenter avec les méconnaissances probables de l'apprenant. Ces méconnaissances (en anglais *bugs*) représentent les déviations du modèle de l'expert et sont regroupées dans un 'catalogue' issu d'une observation et d'une étude des comportements de l'élève. De nombreux systèmes comme PROUST [Joh 83] ou GEOMETRY TUTOR [ABY 85] utilisent ainsi des bibliothèques d'erreurs.

Ces différentes approches partent toutes d'un modèle correct de la tâche étudiée, quel que soient le formalisme employé et la méthode utilisée pour perturber ce modèle. D'autres méthodes cherchent au contraire à modéliser les comportements indépendamment de la notion de correction de ce comportement. Ces méthodes utilisent généralement les techniques de l'ASA.

3.3 Apprentissage automatique et Modèle de l'apprenant.

Nous nous sommes intéressé à l'utilisation de techniques d'apprentissage automatique pour modéliser les connaissances de l'apprenant. Il apparaît en fait qu'il existe peu de réalisations à la jonction de ces deux domaines.
Une premiere approche est le 'diagnostic reconstructif' d'ACM [OL 88]. Un ensemble d'opérateurs de base (primitives), issus d'une décomposition du problème considéré, est défini. Le diagnostic est reconstructif dans le sens où il tente de décrire les observations à l'aide de ces opérateurs. Dans le cas d'ACM, la modélisation des connaissances de l'apprenant correspond à une recherche de 'chemins solutions' construits à partir de ces opérateurs entre l'énoncé du problème et son résultat. Cette construction se fait par apprentissage à partir d'exemples. Pour rendre la recherche praticable, ACM dispose d'un nombre restreint d'opérateurs et d'heuristiques adapté. Cette méthode peut se généraliser par l'utilisation de la PLI.
Les premiers travaux dans cette optique concerne la proposition de Kawai [KMKT 86] selon laquelle le modèle de l'apprenant est représenté par un ensemble de clauses de Horn. La modélisation des comportements de l'apprenant se fait alors par inférence inductive sur le langage de représentation du modèle à partir des observations de ce comportement, sans se préoccuper de sa correction.
Dans le cas de Sarah (Système d'Aide à la Rééducation des ApHasiques, [Sio 94]), cette idée est reprise mais l'apprentissage de concepts se fait à partir d'un seul exemple.

4 Le Cadre du travail.

Notre travail s'effectue dans le cadre du Tuteur d'Aide Logique à la Construction de figures géométriques TALC. Ce tuteur est lui-même basé sur le micro-monde géométrique Cabri-Géomètre. Dans ce paragraphe, nous présentons les objectifs et les caractéristiques de chacun de ces deux systèmes.

4.1 Cabri-Géomètre.

Cabri-Géomètre (*CAhier de BRouillon Interactif* pour la géométrie [Bel 92]) est un micro-monde permettant la construction, la manipulation et l'étude de figures géométriques dans le plan. Il fournit à un utilisateur quatre objets géométriques de base qui ne dépendent d'aucun autre objet : les points, les droites, les segments et les cercles. Ces objets de base permettent de construire des objets plus complexes munis de propriétés particulières, comme par exemple la construction d'une droite passant par deux points, l'intersection de deux objets ou encore la perpendiculaire à une droite.

La conception de l'interface de Cabri-Géomètre permet à l'utilisateur une manipulation directe des objets par la souris. Ce principe est important dans un EIAO car il permet à l'apprenant d'éviter l'association de ses connaissances à une représentation particulière et permet une prise en main rapide du système.
La manipulation directe dans Cabri-Géomètre facilite la création des objets géométriques mais surtout permet d'en modifier la position géographique dans le plan tout en conservant les propriétés géométriques données pendant la construction. Seuls les objets dont le degré de liberté est non nul peuvent être déplacés : les points et les cercles de base ont deux degrés de liberté, les droites de base et les points sur les objets possèdent un seul degré de liberté, les intersections d'objets aucun.

Par exemple, dans la construction suivante (figure 3.1), les points A et B sont des points de base à partir desquels la droite L et le milieu I du segment [AB] ont été construits. L'utilisateur peut 'saisir' un des deux points avec la souris et le déplacer dans le plan, la construction est alors redessinée dynamiquement. Quel que soit le déplacement des deux points, les propriétés de la construction sont conservées : le point I sera toujours au milieu du segment [AB] et la droite L passera toujours par A et B.

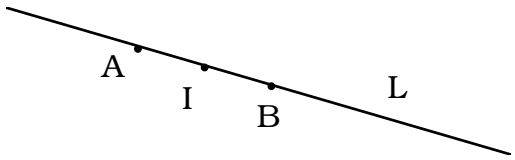


Figure 2 Exemple de construction géométrique sous Cabri-Géomètre

Parmi les caractéristiques de Cabri-Géomètre, nous pouvons noter :

- sa capacité à détecter un certain nombre d'incohérences syntaxiques et numériques.
- la définition de macro-constructions par l'utilisateur.
- un outil, appelé *oracle*, permettant, dans une certaine mesure, de tester un certain nombre de propriétés (appartenance, perpendicularité, parallélisme, alignement et égalité de longueurs).

Cependant Cabri-Géomètre ne dispose véritablement pas de capacités de raisonnement car les connaissances géométriques qu'il manipule sont locales à chaque objet construit. Notamment, il ne permet pas de valider la construction de l'utilisateur relativement à une spécification fournie au préalable. C'est dans cette optique que le TALC a été conçu.

4.2 TALC.

L'objectif de TALC (*Tuteur d'Aide Logique à la Construction* [Des 94]) est de diagnostiquer si une figure géométrique construite par l'apprenant est conforme ou non à un énoncé (ou spécification) fourni par le professeur dans le contexte d'une théorie géométrique représentant un ensemble de connaissances requises.
Ses caractéristiques fondamentales sont d'une part d'autoriser l'enseignant à définir lui-même ses exercices, les connaissances requises et les constructions autorisées pour les résoudre, d'autre part de disposer d'un démonstrateur automatique complet répondant aux exigences de l'interactivité.
TALC est utilisé comme client d'une version serveur de Cabri-Géomètre pour l'interface graphique et la construction des figures.
Le contrat didactique de TALC, c'est-à-dire les relations exigées sur l'objectif donné à l'apprenant et le contexte qui y est rattaché, exige que les conditions suivantes soient respectées :

- la construction doit satisfaire toutes les propriétés de la spécification.
- la construction doit satisfaire uniquement les propriétés de la spécification.
- la construction et la spécification doivent être cohérentes vis-à-vis de la théorie.
- la spécification doit être satisfaite en une seule construction et non en plusieurs disjointes.

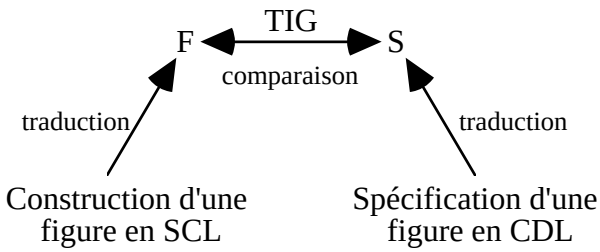


Figure 3 Principe du diagnostic de correction dans TALC

Pour cela, TALC dispose des éléments suivants:

- des langages de description permettant de décrire la spécification du professeur (CDL), la construction de l'apprenant (SCL)
- une Théorie Instrumentale de la Géométrie TIG permettant de représenter le contexte de l'apprentissage
- un démonstrateur complet permettant de vérifier les conditions du contrat.

4.2.1 Les langages de descriptions.

TALC dispose de deux langages d'interface entre les utilisateurs et le système:

- CDL (Classroom Description Language) qui permet la description de l'énoncé du professeur. Il dispose d'une syntaxe et d'une sémantique proches des langages utilisés dans les manuels de géométrie.
- Par exemple, la spécification d'un triangle (ABC) isocèle en A (voir figure 3.3) est décrite par la formule CDL suivante :

$$s1 = [A\ B]\ ,\ s2 = [B\ C]\ ,\ s3 = [C\ A]\ ,\ |A\ B| = |A\ C|$$

- SCL (Student Construction Language) utilisé pour exprimer la construction réalisée par l'apprenant. TALC étant client du micro-monde Cabri-Géomètre pour les constructions géométriques, les primitives de ce langage sont celles de Cabri-Géomètre avec des contraintes supplémentaires concernant les objets construits.

Afin de permettre la vérification de la correction d'une construction de l'apprenant vis-à-vis de l'énoncé du professeur, il est nécessaire de les traduire en deux formules exprimées dans un même langage: LDL (Logical Description Language).

LDL comprend deux catégories de prédicats:

- des prédicats de typage exprimant le type et les arguments d'un objet géométrique.

  Exemples:

  point(p) *p* est un point
  segment(s,p1,p2,l) *s* est un segment d'extrémités *p1* et *p2* et porté par la droite *l*
  cercle(c,p,d) *c* est un cercle de centre le point *p* et de rayon la distance *d*

- des prédicats de propriétés exprimant les relations géométriques entre les objets.

  Exemples:

  appcc(p,c) le point *p* appartient au cercle c
  distancep(d,p1,p2) *d* est la distance séparant les points *p1* et *p2*

Une présentation complète de LDL se trouve dans l'**Annexe B**.

Par exemple, la construction d'un triangle (ABC) isocèle en A (figure 3.3) est traduite par la formule logique LDL suivante:

Enoncé Cabri-Géomètre	**Traduction LDL**
B : point quelconque	point(B)
A : point quelconque	point(A)
segment [B A]	segment(v1,B,A,v2) droite(v2)
C1 : cercle de centre A passant par B	cercle(C1,A,v3) distance(v3) appcc(B,C1)
C : point du cercle C1	point(C) appcc(C,C1)
segment [B C]	segment(v4,B,C,v5) droite(v5)
segment [C A]	segment(v6,B,A,v7) droite(v7)

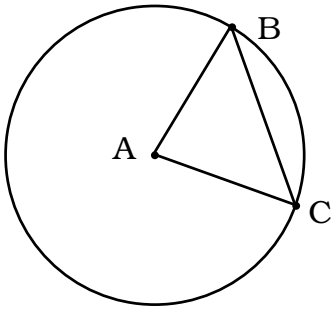


Figure 4 Construction géométrique et traduction LDL

4.2.2 La Théorie Instrumentale de la Géométrie TIG

Les connaissances requises à l'élève pour construire sa figure sont représentée par un ensemble d'axiomes construits à partir du langage LDL. Ces axiomes sont des clauses de Horn.
Par exemple, dans le cas de la construction du triangle isocèle précédent, TIG peut contenir l'axiomes suivant :

- "Un point appartient à un cercle de centre o et de rayon r s'il est à une distance r de o"

```
appcc(p,c) ->
     cercle(c,o,r)
     distancep(r,o,p) ;
```

4.2.3 Le diagnostic de correction

Disposant des traductions en LDL de la spécification du professeur S et de la construction de l'apprenant F, le diagnostic de la correction de figure se ramène alors à vérifier l'équivalence logique entre F et S (figure 3.4).

Plus exactement, il s'agit de trouver une extension de S qui soit équivalente à F et ceci relativement à la théorie de la géométrie TIG, ce qui s'exprime par

$$\exists S^* \text{ tq } TIG \vdash S^* \Leftrightarrow F$$

Cette extension S^* de S permet de prendre en compte les objets de la construction non décrits dans la spécification (le cercle C1 dans l'exemple précédent) tout en conservant la décidabilité.

Actuellement, le diagnostic de correction dans TALC permet de dégager trois situations :

- la construction satisfait toutes les hypothèses de la spécification et uniquement celles-ci. Le contrat est alors rempli.
- la construction ne satisfait pas toutes les hypothèses de la spécification, c'est-à-dire qu'il manque des éléments (objets ou propriétés).
- la construction satisfait toutes les hypothèses de la spécification mais présente des propriétés non déductibles de la spécification. La construction est alors un cas particulier.

5 Conclusion.

Le point important à retenir du cadre de cette recherche est qu'il existe aujourd'hui très peu d'approches de la modélisation de l'apprenant par des méthodes d'apprentissage automatique, et encore moins par la Programmation Logique Inductive (deux références uniquement).

Introduction à la Programmation Logique Inductive

1 Introduction

La Programmation Logique Inductive ou PLI (en anglais *Inductive Logic Programming*, ILP) peut se définir comme étant à l'intersection de l'apprentissage automatique et de la programmation logique.
De l'apprentissage automatique, et plus précisément de l'apprentissage inductif, la PLI hérite des objectifs : développer des outils et des techniques permettant d'induire des hypothèses à partir d'observations et de synthétiser de nouvelles connaissances à partir de l'expérience.
De la programmation logique, la PLI hérite d'un langage de représentation des connaissances palliant les limitations des approches non logique de l'induction (en particulier l'absence de prise en compte d'une théorie du domaine) et les techniques et théories bien établies de ce domaine.
La Programmation Logique Inductive avait à l'origine pour objectif l'inférence de programmes logiques à partir d'observations et relativement à une théorie du domaine. Son application s'étend aujourd'hui à l'apprentissage de concepts représentés sous forme de clauses de Horn.

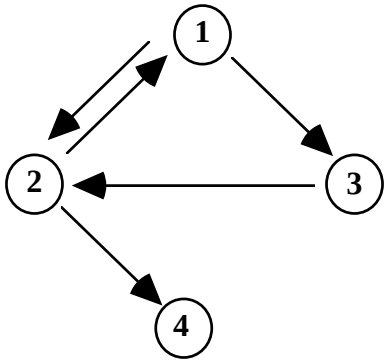
La première partie de ce chapitre est consacrée à une présentation des objectifs et des principes de la PLI. Nous présentons ensuite un ensemble d'outils et de mécanismes de base servant de 'boîte à outils' pour la PLI. La dernière partie détaille le processus d'induction.

2 Présentation de la PLI

2.1 Origines de la PLI

La représentation Attribut-Valeur est la plus répandue dans les systèmes d'induction non logiques (voir les systèmes ID3 de [Quinlan 1986] et AQ de [Michalski 1983]). Sa simplicité permet de traiter des volumes importants d'exemples et de contre-exemples. L'apprentissage inductif consiste à inférer une règle permettant de caractériser et de classifier des objets selon le concept auquel ils appartiennent. Chaque objet est décrit par un ensemble fixé d'attributs, chacun d'eux prenant une valeur dans un ensemble pré-spécifié de valeur. Un concept est décrit par une fonction des valeurs des attributs.

Prenons l'exemple d'un réseau représenté sur la figure 1.1 ([Qua 94]).
Nous décidons que chaque objet noeud va être caractérisé par trois attributs désignant les successeurs directs de celui-ci : S_1, S_2 et S_3 (pas plus de trois successeurs).



	Successeurs		
Noeuds	S_1	S_2	S_3
1	2	3	-
2	1	4	-
3	2	-	-
4	-	-	-

Figure 5 : Exemple de représentation Attribut-Valeur.

Nous pouvons constater plusieurs problèmes :

- figer les attributs entraîne une limite importante de la représentation d'un domaine (dans notre exemple, nous imposons au plus trois successeurs possibles).
- certains concepts sont peu aisés à exprimer. Par exemple, la notion de lien bidirectionnel entre deux noeud X et Y s'écrit :

  $lien\_bidirectionnel(X,Y) = (X.S_1 = Y \cup X.S_2 = Y \cup X.S_3 = Y) \cap (Y.S_1 = X \cup Y.S_2 = X \cup Y.S_3 = X)$
- les concepts récursifs sont impossibles à exprimer. Par exemple, un chemin de longueur quelconque entre deux noeuds.
- il est impossible d'adjoindre une théorie du domaine à une telle base de connaissances.

Ces limitations ont rendu nécessaire l'utilisation d'une représentation des connaissances plus riche (variables, symboles de fonctions, ...). La logique du premier ordre est le formalisme retenu dans la PLI.

Les premiers travaux en induction dans le cadre logique sont ceux de Plotkin [Plo 70] avec la notion de généralité entre clauses. Ses résultats négatifs dans une logique du premier ordre ont suggéré de restreindre le cadre de travail, notamment le langage de représentation des connaissances.
C'est ce que fait Shapiro dans MIS [Sha 81], premier système de PLI, en se restreignant à des clauses de Horn. Il introduit le principe de recherche de clauses dans un espace structuré par un ordre de généralité (voir définition paragraphe 2.1). Les travaux de Shapiro servent de base à des recherches théoriques sur la généralité en logique du premier ordre, en particulier la description d'opérateurs basés sur l'inversion de la résolution par Muggleton [MB 88]. Ces opérateurs permettent notamment à un système de PLI d'inventer de nouveaux prédicats non présents dans la théorie du domaine.
Depuis le début de cette décennie, l'intérêt pour la PLI va grandissant et, parallèlement à ces travaux théoriques, sont créés des systèmes à vocation plus pratique comme GOLEM [MF 92] ou FOIL [Qui 90] capables d'induire des hypothèses à partir d'un grand nombre d'observations et de théories du domaine.

De part son intérêt et la complexité de mise en oeuvre des notions sous-jacentes, la Programmation Logique Inductive (dont le terme *Inductive Logic Programming* est dû à Muggleton [Mug 92]) a aujourd'hui une place importante dans le domaine de l'apprentissage automatique. Preuves en sont les quatre workshops internationaux annuels (*International Workshop on Inductive Logic Programming*, qui se sont succédés depuis 1992) exclusivement dédiés à la PLI.

2.2 Principe de la PLI

L'objectif de la PLI est d'inférer une définition d'un concept (appelé *concept cible*) à partir d'observations. Nous allons préciser les objectifs d'un tel processus et les éléments nécessaires à sa réalisation.

L'ensemble des observations est divisé en ensemble des exemples positifs et ensemble des exemples négatifs. L'ensemble des exemples positifs (appelés simplement exemples) du concept est noté $\mathbf{E^+}$. Celui des exemples négatifs (ou contre-exemples) est noté $\mathbf{E^-}$. Généralement, les observations sont constitués d'atomes clos dont le symbole de prédicats est celui du concept cible.
La connaissance relative au domaine où se déroule l'apprentissage, pouvant être utilisé dans la définition du concept, est noté **B** (pour *Background Knowledge*, terme anglais généralement utilisé). Celle-ci est un programme logique constitué de définitions de prédicats (on parle de théorie du domaine intentionnelle) et de faits (théorie du domaine extensionnelle).
La définition du concept (ou hypothèse) que l'on cherche à obtenir est noté **H**.
Cette notation sera utilisée tout au long de ce travail.

Formellement, le principe de la PLI peut se définir de la façon suivante [MD 94] :

Étant donnés
- une théorie du domaine **B**
- un ensemble d'exemples $\mathbf{E^+}$
- un ensemble de contre-exemples $\mathbf{E^-}$

Trouver une hypothèse **H** vérifiant
- la complétude $\mathbf{B, H \vdash E^+}$
- la consistance $\mathbf{B, H \nvdash E^-}$

La première condition de l'apprentissage exige que H soit complète par rapport aux exemples positifs, c'est-à-dire qu'elle couvre la totalité de ceux-ci. La deuxième condition impose que H ne couvre aucun exemple négatif. Cette notion de couverture est précisée dans le paragraphe 3.2.
A ces deux conditions, nous pouvons rajouter la précondition évidente mais nécessaire **B** |-/- $\mathbf{E}^+$ qui permet de s'assurer que la théorie du domaine ne contient pas déjà une définition complète du concept.
Prenons un exemple simple : l'apprentissage du concept *grand-père* (tiré de [Sio 94]).
Nous disposons d'une théorie du domaine **B** contenant les clauses de Horn suivantes :

```
père(Jean, Patrick).
père(Louis, Nicole).
père(Louis, Jean).
mère(Nicole, Sophie).
mère(Nicole, Quentin).
parent(X, Y) ← mère(X, Y).
parent(X, Y) ← père(X, Y).
```

Les exemples $\mathbf{E}^+$ du concept *grand-père* sont :

```
grand-père(Louis, Patrick).
grand-père(Louis, Sophie).
grand-père(Louis, Quentin).
```

Des contre-exemples $\mathbf{E}^-$ peuvent être :

```
grand-père(Louis, Jean).
grand-père(Nicole, Nicole).
grand-père(Jean, Quentin).
......
```

Une hypothèse **H** possible est :

```
grand-père(X, Y) ← père(X, Z) , père(Z, Y).
grand-père(X, Y) ← père(X, Z) , mère(Z, Y).
```

ou même

```
grand-père(X, Y) ← père(X, Z) , parent(Z, Y).
```

Cet exemple met en évidence quelques problèmes de la PLI :

- - il existe un grand nombre d'hypothèses syntaxiquement correctes. En effet, l'ensemble des symboles de prédicats utilisables est fixé par la théorie du domaine mais les variables et le contenu du corps des clauses constituant cette hypothèse ne le sont pas. L'inférence d'une hypothèse correcte correspond de ce fait à un problème de recherche dans un espace constitué de l'ensemble de toutes les clauses syntaxiquement correctes. Il apparaît donc nécessaire de disposer d'outils permettant de restreindre cet espace.
- - plusieurs hypothèses peuvent satisfaire les conditions d'induction : il s'agit dans ce cas de faire un choix parmi celles proposées et donc de disposer d'outils permettant cette sélection.
-

Les langages utilisés pour exprimer les éléments de la PLI diffèrent suivant le domaine abordé et les objectifs visés. L'intérêt principal de la PLI est l'inférence d'hypothèses en logique du premier ordre mais les problèmes posés font apparaître le besoin de restreindre le langage de description utilisé, l'espace des théories possibles étant infini. Passer de la logique de premier ordre aux clauses de Horn s'avère opportun mais nécessite généralement d'autres restrictions. Ces restrictions (appelées *biais*) peuvent s'appliquer au vocabulaire du langage de description, à la forme de la théorie du domaine ou à celle des hypothèses induites.

2.3 Intérêts et applications de la PLI

L'originalité de la PLI réside principalement dans l'utilisation de la logique du premier ordre. Les techniques classiques d'apprentissage inductif se restreignent en effet à l'apprentissage de concepts exprimés dans des langages équivalents à la logique propositionnelle. La PLI bénéficie aussi des résultats théoriques et pratiques de la programmation logique (beaucoup d'outils de la PLI sont issus de techniques de base de la logique, comme la substitution, la résolution, ... voir paragraphes suivants).
Les travaux actuels de la PLI peuvent se partager grossièrement en trois grands groupes :

- - des travaux théoriques sur la généralisation, la spécialisation, la révision de théorie, ...
- - des travaux plus pratiques sur des méthodes efficaces de recherche, par exemple en combinant différentes approches de l'apprentissage automatique (apprentissages inductif et basé sur

l'explication - Explanation Based Learning - dans FOCL [PK 92]) ou en procédant à un parcours bi-directionnel de l'espace de recherche [Sio 94].

- - des travaux sur l'application de la PLI à des domaines réels. Nous y retrouvons des applications à l'acquisition de connaissances et à la découverte de connaissances dans des bases de données (*Knowledge Discovery in Databases*, KDD) [MD 94], à la synthèses de programmes logiques [ND 94] et des applications à des domaines physiques (voir les exemples de la conception de maillage fini [Mug 92]).

3 Mécanismes de base de la PLI

L'ensemble des formules qu'il est possible d'exprimer dans le langage constitue un espace de recherche dans lequel il s'agit de trouver celles qui formeront les hypothèses satisfaisant les conditions d'induction. Cet espace doit être structuré de façon à permettre une recherche la plus optimisée possible. En vue de cet objectif, une relation d'ordre sur les formules, appelé "ordre de généralité" est utilisé. Nous décrivons dans ce paragraphe des mises en oeuvre de la généralité, ainsi que celles d'opérateurs de généralisation et de spécialisation de clauses qui en dérivent et qui constituent une "boîte à outils" pour la Programmation Logique Inductive.

3.1 Une notion importante : la généralité

La notion intuitive de la généralité sur les formules logiques est la suivante [Sio 94] :

une formule logique C_1 est plus générale qu'une formule C_2 (noté $C_1 > C_2$) si elle permet de décrire plus d'objets que C_2.

Par exemple, la formule *père(Louis, X)* est plus générale que *père(Louis, Jean)* car *père(Louis, X)* s'applique aux objets *Jean* et *Nicole*.

En Programmation Logique Inductive, cette comparaison des formules est facilitée par l'utilisation d'une représentation unique pour les exemples et la théorie du domaine, à savoir les clauses de Horn, restriction de la logique du premier ordre. La comparaison de deux formules relativement à la théorie du domaine **B**, par l'utilisation de la dérivation logique (ou conséquence syntaxique), permet d'introduire la notion **sémantique** de la généralité.

- Définition de la généralité sémantique :

  C_1 est plus générale que C_2 ($C_1 > C_2$) si et seulement si
  $\mathbf{B}, C_1 \vdash C_2$

Or, nous savons que la mise en oeuvre d'une telle opération pose problème du fait de la semi-décidabilité de la dérivation logique. Beaucoup de relation de généralité se contentent donc d'utiliser la notion **syntaxique** de la généralité, c'est-à-dire en ne comparant que les formules indépendamment de la théorie du domaine. C'est la cas de la θ-subsomption [Plo 70].

- Définition de la θ-subsomption :

  C_1 θ-subsume C_2 si et seulement si il existe une substitution θ telle que $C_1\theta \subseteq C_2$ c'est-à-dire si
  $tête(C_1)\theta = tête(C_2)$
  $corps(C_1)\theta \subseteq corps(C_2)$
  (Par tête(X) et corps(X), nous désignons respectivement la tête et le corps de la clause X).

La clause C_1 est au moins aussi générale que la clause C_2 ($C_1 > C_2$) si C_1 θ-subsume C_2 et C_2 ne θ-subsume pas C_1.

Par exemple, soient les deux clauses

C_1 = grand-père(X,Z) ← père(X,Y)
C_2 = grand-père(A,C) ← père(A,B) , père(B,C)

On a C_1 θ-subsume C_2 avec θ={X/A, Y/B, Z/C}.

Cette relation d'ordre entre clauses est purement syntaxique et souffre donc de certaines limitations, notamment en ne prenant pas en compte la théorie du domaine. Par exemple, sachant que *parent(X,Y)* est vrai si *père(X,Y)* ou *mère(X,Y)* sont vrais, nous souhaiterions pouvoir dire que la clause C_1 = *grand-père(X,Z)* ← *père(X,Y)* , *parent(Y,Z)* est plus générale que la clause
C_2 = *grand-père(A,C)* ← *père(A,B)* , *père(B,C)*. Or C_1 ne θ-subsume pas C_2. Il faut pour cela introduire la théorie du domaine, notamment le prédicat *parent* .

Cependant, la θ-subsomption, en tant que relation d'ordre, possède un certain nombre de propriétés intéressantes [LD 94],[MD 94]:

- elle permet une structuration de l'espace de recherche.
- elle permet d'élaguer cet espace de recherche. En effet, si nous disposons d'un ordre de généralité entre clauses, alors nous pouvons facilement vérifier que si une clause couvre un exemple négatif, alors toutes les clauses plus générales le couvrent aussi et si une clause ne couvre aucun exemple positif, alors ses spécialisations n'en couvrent aucun non plus.
- elle introduit deux types d'opérateurs (ou règles d'inférence) importants qui constituent la base d'une boîte à outils pour la Programmation Logique Inductive : les opérateurs de généralisation ρ_G : $\rho_G(C_1) = \{ C_2 \mid C_2 > C_1 \}$ et les opérateurs de spécialisation (ou d'affinement) ρ_S : $\rho_S(C_1) = \{ C_2 \mid C_1 > C_2 \}$.

3.2 Opérateurs de généralisation

Les opérateurs de généralisation (ou règle d'inférence inductive) permettent un parcours ascendant (*Bottom-Up*) de l'espace de recherche, c'est-à-dire de la clause la plus spécifique autorisée par le langage de description à la plus générale.
La généralisation de clauses consiste à appliquer principalement deux opérations syntaxiques

- la substitution inversée,
  soit θ une substitution, il existe une substitution θ^{-1} telle que $c\theta\theta^{-1} = c$
  père(Louis, Albert) < père(X, Albert) *avec* θ^{-1} *= {louis/X}*
- le retrait de littéraux dans le corps d'une clause : *parent(X,Y)* ← *père(X,Y) < parent(X,Y)*

Nous présentons deux exemples d'opérateurs de généralisation intéressants :

- *Least general Generalization* (*lgg*, moindre généralisation) [Plo 70]
- *Inverse Resolution* (inversion de la résolution) [MB 88]

3.2.1 Least general Generalization

La θ-subsomption a amené Plotkin à définir la notion de "généralisation la plus spécifique" (*Least General Generalization* [Plo 70]) qui peut être prise comme notion duale de l'unification la plus générale (*Most General Unification*, *mgu*). Cette notion a une place importante dans le cadre de la PLI car elle sert de base à de nombreux algorithmes de généralisation comme GOLEM [MF 90].
Le *lgg* construit la plus grande borne inférieure de deux clauses sous la θ-subsomption.

- Définition du *lgg* :
  Termes :

  $lgg(f(s_1,...,s_n),f(t_1,...,t_n)) = f(lgg(s_1,t_1),...,lgg(s_n,t_n))$
  $lgg(f(s_1,...,s_n),g(t_1,...,t_n)) = V$ si $f \bullet g$ où V est une variable représentant le lgg de la paire des termes.

  Littéraux :

  $lgg(p(s_1,...,s_n),p(t_1,...t_n)) = p(lgg(s_1,t_1),...,lgg(s_n,t_n))$
  $lgg(p(s_1,...,s_n),q(t_1,...t_n)) = \emptyset$ si $p \bullet q$ (symboles ou signes)

  Clauses :

  $lgg(C_1,C_2) = \{lgg(l_i,l_j) \mid l_i \in C_1,\ l_j \in C_2\}$

Par exemple, le lgg des clauses

- *e_1 = grand-père(Louis, Patrick)* ← *père(Louis, Jean), parent(Jean, Patrick)*

- e_2 = *grand-père(Louis, Quentin)* ← *père(Louis, Nicole), parent(Nicole, Quentin)*

est la clause

- *lgg*(e_1, e_2) = *grand-père(Louis, X)* ← *père(Louis, Y), parent(Y,X),*

où X = lgg(Patrick, Quentin) et Y = lgg(Jean, Nicole).

Le *lgg* reste syntaxique et possède les mêmes limitations inhérentes à la θ-subsomption. D'autres approches plus générales ont étés étudiées, particulièrement le *rlgg* (*Relative Least general Generalization*) [MF 92]. Le rlgg prend en compte la théorie du domaine mais nécessite que celle-ci soit donnée sous forme intentionnelle (c'est-à-dire sous forme d'atomes clos) exclusivement. Soit K la conjonction de tous les faits de la théorie du domaine, le *rlgg* de deux exemples positifs L_1 et L_2 se définit par :

$$rlgg(L_1,L_2) = lgg(L_1 \leftarrow K,\ L_2 \leftarrow K)$$

Soient deux exemples positifs du concept *grand-père*

- L_1 = *grand-père(Louis, Patrick)*
- L_2 = *grand-père(Louis, Nicole)*

et la conjonction des faits de notre théorie du domaine précédente

K = *père(Jean, Patrick), père(Louis, Nicole), père(Louis, Jean), mère(Nicole, Sophie), mère(Nicole, Quentin)*

Nous avons alors rlgg(L_1, L_2) = lgg(L_1 ← K, L_2 ← K) qui produit la clause suivante :

grand-père(Louis , $V_{P,N}$) ←
père(Jean, Patrick), père(Louis, Nicole), père(Louis, Jean),
père($V_{J,L}$, $V_{P,N}$), père($V_{J,L}$, $V_{P,J}$), père(Louis, $V_{N,J}$),
mère(Nicole, Sophie), mère(Nicole, Quentin), mère(Nicole, $V_{S,Q}$)

où $V_{x,y}$ correspond à lgg(x,y). Nous remplaçons maintenant les termes $V_{x,y}$ par des variables distinctes pour obtenir la clause suivante :

grand-père(Louis , X) ← *père(Jean, Patrick), père(Louis, Nicole), père(Louis, Jean),*
père(Y, X), père(Y, U), père(Louis, Z),
mère(Nicole, Sophie), mère(Nicole, Quentin), mère(Nicole, W)

L'exemple montre que le *rlgg* produit des clauses qui sont relativement importantes mais dont certains littéraux peuvent être superflus; pour les supprimer, l'utilisation de contraintes syntaxiques est nécessaire, particulièrement sur les conditions d'introduction des nouvelles variables.

3.2.2 Inversion de la résolution

Dans le système d'apprentissage de concepts CIGOL [MB 88], Muggleton introduit des opérateurs de généralisation et de reformulation basés sur l'inversion du mécanisme de la résolution.

Dans la description suivante des opérateurs de l'inversion de la résolution, les minuscules désignent des atomes, les majuscules des conjonctions d'atomes. Les opérateurs sont représentés graphiquement sous la forme d'une étape de résolution (les données en haut et la résolvante en bas) de façon à mettre en évidence le principe de l'inversion.
Les clauses encadrées correspondent à celles engendrées par l'application de l'opérateur, les autres correspondent aux données de l'opérateur.
Les trois opérateurs sont les suivants :

- les opérateurs "V" qui inversent une étape de la résolution. Suivant la position du littéral commun dans la clause obtenue par application de l'opérateur, il s'agit de l'absorption (littéral dans la queue, figure 1.1.a) ou de l'identification (littéral dans la tête, figure 1.1.b). Seule l'absorption est un opérateur de généralisation. Muggleton et Buntine [MB 88] font l'hypothèse de séparabilité selon laquelle il n'existe dans chacune des clauses parents qu'un seul littéral unifiable.

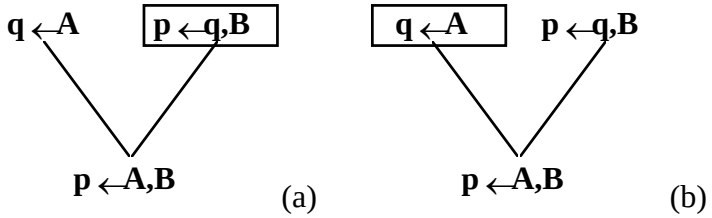


Figure 6 : Opérateurs "V" : Absorption et Identification.

- les opérateurs "W" qui combinent deux étapes d'inversion de la résolution. Il s'agit d'opérateurs de reformulation qui ne modifient pas le pouvoir d'expression des formules mais, l'inversion de la résolution formant un tout, nous les décrivons ici. Comme pour les opérateurs "V", la position du littéral commun définit deux opérateurs : l'intraconstruction (figure 1.2.a) et l'interconstruction (figure 1.2.b). Le résultat de l'application d'un opérateur "W" n'est pas unique, il faut utiliser des restrictions syntaxiques pour limiter le nombre de solutions possibles.

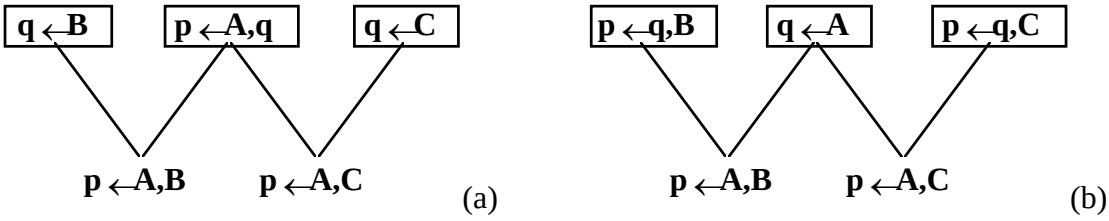


Figure 7 : Opérateurs "W" : Intraconstruction et Interconstruction.

L'inversion combinée de deux étapes de la résolution est un moyen d'introduire un nouveau prédicat. En effet, en combinant deux résolutions, le littéral *q* n'apparaît plus dans les deux résolvantes; en inversant le processus, il faut donc introduire ce littéral. Cette *invention de prédicat* est une caractéristique importante de la PLI qui sera détaillée plus loin dans ce chapitre. Par exemple, l'application de l'opérateur d'intraconstruction à notre exemple de problème peut donner le résultat suivant (**figure 1.3**) : le prédicat *new-pred(X,Y)* correspond ici au prédicat *parent(X,Y)*.

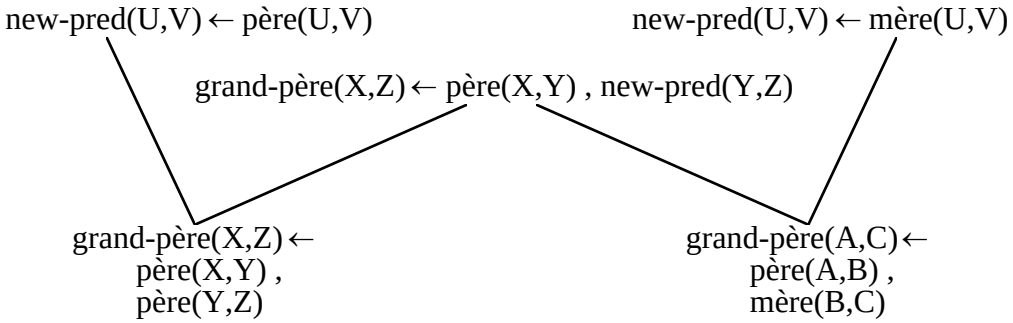


Figure 8 : Exemple d'application de l'Intraconstruction.

- la Troncation qui est un opérateur "V" ou "W" dont la ou les bases sont la clause vide. Dans CIGOL, le seul opérateur de troncation implémenté est celui qui correspond au *lgg* de Plotkin (voir paragraphe précédent).

3.3 Opérateurs de spécialisation

Les opérateurs de spécialisation (ou règles d'inférence déductive) permettent un parcours descendant (*Top-Down*) de l'espace de recherche.

Les opérateurs de spécialisation (appelés aussi opérateurs d'*affinement*) appliquent les deux opérations syntaxiques duales de la généralisation :

- - la substitution : *père(X,albert) > père(louis,albert)* *avec* $\theta = \{X/louis\}$
- - l'ajout de littéraux dans le corps d'une clause : *parent(X,Y) > parent(X,Y) ← père(X,Y)*
-

Un des opérateur classique est la résolution qui permet de dériver une résolvante R à partir de deux clauses C_1 et C_2. Selon le principe de la résolution, $\{C_1, C_2\}$ |-- R et R est donc bien plus spécifique que C_1 et C_2. Par exemple,

$$\frac{C_1 = parent(U,V) \blacklozenge père(U,V) \quad C_2 = grand\text{-}père(X,Z) \blacklozenge père(X,Y), parent(Y,Z)}{R = grand\text{-}père(X,Z) \blacklozenge père(X,Y), père(Y,Z)}$$

Dans [LD 93] et [Sio 94], les auteurs signalent qu'il n'existe pas d'opérateur d'affinement "idéal". selon eux, la pratique fait apparaître, pour des raisons d'efficacité, la nécessité de restreindre l'espace de recherche par des opérateurs d'affinements au problème abordé.

A titre d'exemple, nous donnons l'opérateur d'affinement utilisé dans FOIL [Qui 90] qui construit l'ensemble des spécialisations d'une clause.
Soit p le prédicat cible d'arité n et défini par un ensemble de faits clos.
Soit C une clause à spécialiser. $D \in \rho_{FOIL}(C)$ si [LD 94] :

- si C = $\square$ alors $D = p(X_1, X_2, ..., X_n)$
  où p est le symbole de prédicat cible et $X_1, X_2, ..., X_n$ des variables distinctes
- si $C = P \leftarrow Q$ alors D est obtenu en rajoutant dans le corps de C un littéral qui a l'une des formes suivantes :
  - $X_j = X_s$ ou not($X_j = X_s$)
    où X_j et X_s apparaissent dans C
  - $q_i(Y_1, Y_2, ..., Y_m)$ ou not($q_j(Y_1, Y_2, ..., Y_m)$)
    où $q_i \bullet p$ et au moins une des variables $Y_1, Y_2, ..., Y_m$ apparaît dans C
  - $p(Y_1, Y_2, ..., Y_n)$ ou not($p(Y_1, Y_2, ..., Y_n)$)
    où $Y_1, Y_2, ..., Y_n$ apparaissent dans C et il existe au moins une variable X_i de la tête de C telle que $X_i < Y_i$ appartient à O_c (ensemble de relations d'ordre imposé sur les variables de chaque clause et associé à celle-ci; les règles pour créer et mettre à jour cet ensemble sont décrites dans [Qui 91])

Par exemple, soit p = *grand-père* et $C = grand\text{-}père(X,Y) \leftarrow père(X,Z)$ avec $O_c = \{X<Z\}$.
Alors le littéral *parent(Z,Y)* peut être ajouté au corps de C (car Z et Y apparaissent dans C).
De même, le littéral *grand-père(Z,Y)* peut être rajouté car $X<Z \in O_c$. Mais *grand-père(W,Y)* ne le peut pas car $X < W \notin O_c$.

4 Processus d'induction

Dans le paragraphe précédent, nous avons vu qu'un ordre de généralité entre clauses et des opérateurs de généralisation et de spécialisation de clauses permettent de construire et de structurer un espace de recherche. Cela signifie qu'il existe potentiellement dans cet espace certains noeuds qui correspondent à des hypothèses vérifiant les conditions d'un apprentissage inductif. Le problème consiste à les trouver.
Dans un premier temps, nous allons décrire un modèle générique d'un algorithme de PLI, puis détailler deux aspects importants du processus d'induction : les problèmes de la construction des clauses de l'hypothèse et la justification de celles-ci. Nous concluons ce paragraphe en précisant le cadre et les problèmes de l'invention de prédicats, introduite par exemple avec les opérateurs "W" de l'inversion de la résolution (voir paragraphe 2.2.2).

4.1 Algorithme générique

Les algorithmes de Programmation Logique Inductive peuvent se classer en deux catégories :

- les algorithmes incrémentaux dans lesquels les exemples sont donnés les uns après les autres
- les algorithmes empiriques dans lesquels les exemples sont initialement fournis au système.

Dans le premier cas, la méthode générale du processus d'induction consiste à rajouter à l'hypothèse en construction une clause qui ne couvre aucun exemple négatif et a supprimer les exemples positifs couverts par cette clause de l'ensemble des observations à induire, ceci jusqu'à la complétude de l'hypothèse.
Un système incrémental doit procéder à une révision de l'hypothèse inférée à chaque nouvel exemple. A chaque exemple rajouté (positif ou négatif), tant que l'hypothèse couvre un exemple négatif, on supprime la clause fautive; tant que l'hypothèse ne couvre pas tous les exemples positifs, on lui rajoute une clause qui assure la couverture.

Les deux processus ne sont pas complètement similaire mais nous pouvons néanmoins donner un algorithme générique du processus d'induction [MD 94] (figure 3.1).
QH est la file des hypothèses candidates, H est l'hypothèse en cours de traitement.

```
QH ← Initialise()
Répéter
    Sortir H de QH
    Choisir les règles d'inférences r1, ..., rn de R à appliquer à H
    Appliquer les règles r1, ..., rn à H pour obtenir H1, ..., Hk
    Ajouter H1, ..., Hk à QH
    Élaguer QH
Jusqu'à critère d'arrêt(QH)
```

Figure 9 : Algorithme générique de PLI.

L'algorithme travaille de la façon suivante : il garde une trace de toutes les hypothèses candidates QH. De manière répétitive, il sort une hypothèse H de la queue QH et la développe de toutes les manières possibles en utilisant les opérateurs à sa disposition. Les hypothèses développées sont réinsérées dans QH qui peut être élagué en supprimant les hypothèses qui ne satisfont pas certains conditions. Le processus est répété jusqu'à obtention d'un critère d'arrêt.

L'algorithme précédent fait appel à des procédures génériques (termes soulignées) qui assurent les opérations suivantes :

- Initialise dénote l'ensemble des hypothèses initiales.
- Sortir influence la stratégie de recherche. Suivant le type d'accès à QH, l'algorithme peut réaliser des parcours de l'espace de recherche en profondeur (Sortir = LIFO), en largeur (Sortir = FIFO) ou meilleur d'abord.
- R dénote l'ensemble des règles d'inférence (généralisation, spécialisation et reformulation) applicables.
- Choisir permet de sélectionner certaines règles d'inférence.
- Élaguer détermine les hypothèses à supprimer de QH. Se fait généralement par heuristique sur les hypothèses de QH (voir paragraphes suivants) ou par intervention de l'utilisateur (systèmes interactifs).
- Critère d'arrêt détermine les conditions sous lesquelles l'induction s'arrête. Les critères les plus employés sont l'obtention d'une hypothèse H correcte ou, au contraire, son inexistence.

Il faut noter que cet algorithme recherche des solutions au niveau des hypothèses plutôt qu'au niveau des clauses, comme c'est la cas de la plupart des systèmes. Il tient compte en effet d'une approche la plus générale.
Parmi les problèmes de mise en oeuvre de cet algorithme, nous développons la construction et la justification des hypothèses.

4.2 Justification des hypothèses

Lors de la construction d'une hypothèse (ou d'une des clauses de celle-ci), il est nécessaire de s'assurer de sa cohérence : l'hypothèse doit "couvrir" un certain nombre d'exemples positifs et aucun exemple négatif.

- Définition de la couverture intentionnelle.

  Une clause C couvre un exemple e relativement à une théorie du domaine B si et seulement si B,C |-- e

Il s'agit de la notion de couverture intentionnelle car la théorie du domaine est intentionnelle (elle peut contenir aussi bien des règles que des faits). La méthode la plus utilisée pour vérifier cette couverture est la SLD-résolution (bornée ou non).

De manière à pouvoir apprendre des clauses indépendamment les unes des autres, sans se préoccuper du comportement général de l'hypothèse (par exemple dans le cas de définitions récursives ou d'apprentissage multi-conceptuel), une notion plus syntaxique a été introduite : la couverture extensionnelle. La couverture extensionnelle requiert que la théorie du domaine B soit transformée en modèle clos M.

- Définition de la couverture extensionnelle.

  Une clause C = T←Q couvre un exemple e relativement à un modèle clos M si et seulement si il existe une substitution θ telle que Tθ = e et Q$\theta \subseteq$ M

La méthode extensionnelle comporte néanmoins des défauts. Notamment, une hypothèse consistante peut être rejetée parce qu'un exemple correspondant à un but de la preuve par extension ne fait pas partie des exemples positifs. En conséquence, une justification extensionnelle ne peut se faire que si l'ensemble des exemples positifs est complet ou suffisant pour permettre d'inférer une hypothèse consistante. C'est pour cette raison que, d'une manière générale, les systèmes incrémentaux utilisent la couverture intentionnelle tandis que les systèmes empiriques utilisent la couverture extensionnelle.

4.3 Construction des hypothèses

Construire une hypothèse satisfaisant les conditions d'apprentissage revient, de manière répétitive, à construire une clause qui couvre un ensemble d'exemples positifs sans couvrir d'éventuels exemples négatifs. Ces derniers peuvent être utilisés pour contraindre la recherche à effectuer prioritairement sur les exemples positifs.
Cette recherche peut être caractérisée par la structure de l'espace, les stratégies de parcours utilisées et les heuristiques employées pour guider ce parcours.
En ce qui concerne la structure de l'espace, nous avons vu que l'utilisation d'un ordre de généralité sur les clauses permet une telle structuration.
Les stratégies de parcours sont généralement celles classiquement utilisées en intelligence artificielle pour le parcours d'arbre de résolution : largeur, profondeur ou meilleur d'abord.
C'est au niveau des heuristiques que peut se faire les différences entre différentes approches de la PLI.

4.4 Invention de prédicats

L'une des perspective les plus intéressantes de la PLI est certainement sa capacité à étendre automatiquement le vocabulaire du langage de description, désignée sous le terme d'*invention de prédicats*.
Nous désignons par vocabulaire d'un programme logique P l'ensemble de tous les symboles de prédicats trouvés dans la tête des clauses de P. Dans le cadre de la PLI, le vocabulaire du langage, fixé avant le début de l'apprentissage, provient de différentes sources [MD 94] :

- le vocabulaire observationnel obtenu à partir des observations $\mathbf{E^+}$ et $\mathbf{E^-}$
- le vocabulaire théorique obtenu exclusivement à partir de la théorie du domaine **B**

Sous ses définitions, l'invention de prédicats intervient lorsque le vocabulaire des hypothèses n'est pas équivalent à l'ensemble des deux autres.
Les opérateurs "W", décrits dans le paragraphe 2.2.2, sont une première approche de l'invention de prédicats. Leur application (voir exemple, figure 2.3) introduit bien un nouveau prédicat *new-pred(X,Y)* mais il ne s'agit que d'une reformulation qui rend la théorie plus compacte et compréhensible. Par contre, l'invention de prédicats devient intéressante lorsque la théorie du domaine n'est pas suffisante pour expliquer les exemples du concept.

Par exemple, dans la figure 2.3, le prédicat inventé *new-pred* n'est pas véritablement nécessaire à la définition de *grand-père* puisque nous disposons des définitions de *père* et *mère*. Par contre, en supposant que le prédicat *mère* n'existe pas dans la théorie du domaine, la définition de *grand-père* devient alors impossible à produire à partir des exemples du seul prédicat *père*. Un nouveau prédicat est donc nécessaire. Sa définition sera, dans ce cas simple, constitué de faits.
Dans un cas plus général, une définition intentionnelle pourrait être induite à partir de ces faits et de la théorie du domaine, de la même manière que pour le concept cible de l'induction.

L'invention de prédicats est un problème complexe mais très utile. Elle nécessite principalement de répondre à quatre questions [Sio 94], [MD 94], [Mug 94] :

- Quand inventer un prédicat ?
- Où placer l'appel du nouveau prédicat ?
- Quelle est l'arité du nouveau prédicat ?
- Quelle est la définition du nouveau prédicat ?

L'invention de prédicats n'étant pas l'objectif de ce travail, nous ne nous étendrons pas plus sur cet aspect intéressant de la PLI.

5 Caractéristiques des systèmes de PLI

Dans les paragraphes précédents, nous avons donné un aperçu des principales techniques et opérations de la Programmation Logique Inductive. Les systèmes de PLI qui mettent en oeuvre ces différents outils peuvent être classés selon différents aspects du point de vue d'un utilisateur :

- incrémentaux ou empiriques : les données d'induction sont respectivement entrées les unes après les autres ou déjà présentes dans le système.
- interactifs ou non : l'apprentissage nécessite l'intervention de l'utilisateur (ou oracle) pour confirmer ou infirmer les inférences du système. L'interactivité va souvent de pair avec l'incrémentalité, mais peut s'appliquer aussi uniquement à un certain type de problèmes dans un système empirique (par exemple révision de la théorie dirigée par l'utilisateur).
- mono ou multi-conceptuels : le système permet l'induction respectivement d'un unique concept ou de plusieurs simultanément.
- induction d'hypothèses ou révision de théories : l'apprentissage va consister dans le premier cas à induire un ou plusieurs concepts (exemple : induction de la définition du concept *grand-père*) alors que dans le deuxième, il va permettre en outre la révision d'une définition préalable incomplète et/ou incorrecte des concepts à induire.
- invention de prédicats : certains systèmes ont la capacité de pouvoir inventer de nouveaux prédicats utiles pour la définition du concept mais n'existant pas déjà dans la théorie du domaine.

Le chapitre suivant est consacré à une analyse d'un certain nombre de ces systèmes de PLI.

6 Conclusions

La Programmation Logique Inductive vise à étendre les capacités de l'apprentissage automatique, jusque là restreint par les formalismes de descriptions des connaissances utilisées.
Les principales caractéristiques de la PLI sont :

- un apprentissage inductif consistant à inférer la définition d'un ou plusieurs concepts à partir d'exemples de leurs comportements.
- la prise en compte d'une théorie du domaine, ensemble de connaissances relatives au problème.
- l'utilisation d'un formalisme unique pour représenter les exemples des concepts, la théorie du domaine associée au problème et les hypothèses induites.
- l'utilisation des clauses de Horn, restriction commode de la logique du premier ordre.
- sa capacité à "inventer" des éléments non existants dans le vocabulaire d'origine.

Évaluation des systèmes de Programmation Logique Inductive

1 Introduction

Dans le chapitre précédent, nous avons vu que les systèmes de PLI peuvent être caractérisés selon plusieurs critères:

- - apprentissage incrémental ou empirique.
- - apprentissage interactif ou non.
- - apprentissage mono ou multi-conceptuel.
- - induction d'hypothèses ou révision de théories.
- - invention de prédicats ou non.

Il existe actuellement un grand nombre de systèmes de PLI, tous à un stade plus ou moins expérimental ou académique. De ce fait, il est difficile d'en faire un état de l'art exhaustif, d'autant plus que bon nombre des systèmes de PLI cités dans la littérature ne sont pas disponibles aisément ou sont trop spécifiques pour une comparaison.
Quatre de ces systèmes ont été évalués afin d'envisager l'utilisation de l'induction pour construire un modèle des connaissances de l'apprenant dans TALC: il s'agit de FOIL, FOCL, GOLEM et CLINT.

La première partie de ce chapitre propose une étude des systèmes de PLI retenus. Dans la deuxième, nous présentons une synthèse de l'exploitation réalisée afin de mettre en évidence les caractéristiques intéressantes des systèmes de PLI.

2 Présentation des systèmes.

Dans cette partie, nous présentons sommairement les systèmes retenus pour ce document. La sélection des systèmes s'est faite essentiellement sur leur disponibilité via des sites ftp mais aussi de manière à avoir un éventail relativement complet des différentes approches de la PLI (apprentissage mono ou multi-conceptuel, apprentissage interactif ou incrémental, ...). Les systèmes non retenus et les raisons de cet abandon se trouve dans l'**Annexe A**.

2.1 FOIL (First-Order Inductive Logic).

FOIL ([Qui 90], [Qui 91] et [QC 93]) est un système qui étend le principe d'induction sur une représentation Attribut-Valeur des connaissances par l'utilisation de la logique du premier ordre sans symboles de fonctions (exception faite des constantes, symboles de fonction d'arité 0).

2.1.1 Restrictions sur l'espace de recherche.

FOIL est mono-conceptuel, il tente d'inférer la définition d'un unique concept (éventuellement récursif) à partir des prédicats de la théorie du domaine. Les définitions des prédicats du domaine sont données de manière exclusivement extensionnelle et doivent être closes (pas de variables libres). C'est une limitation importante de FOIL.

Il est empirique et ne nécessite pas obligatoirement de contre-exemples, ceux-ci étant automatiquement générés lors de la phase de prétraitement des exemples en utilisant l'hypothèse du monde clos (Close World Assumption, CWA). Pour permettre cette génération, les prédicats sont typés et les valeurs possibles de ces types sont données explicitement, de manière à travailler sur un univers de Herbrand fini.
Les données d'entrée du programme pour un domaine d'apprentissage sont :

- - un ensemble de types utilisés dans la définition des prédicats. Ces types peuvent être infinis (nombres entiers et réels usuels) ou énumérés auquel cas ils peuvent être ordonnés, non ordonnés

ou d'ordre inconnu. Cet ordre sur les valeurs des types est utilisé par FOIL lors de la généralisation des hypothèses pour préférer un littéral plutôt qu'un autre équivalent. Certaines valeurs des types peuvent être désignées comme pouvant être utile dans la définition du concept (par exemple pour les cas de base).

- - un ensemble de prédicats décrits de façon extensionnelle dont l'un est le prédicat cible de l'induction. Ces prédicats sont données par leur signature (nom du prédicat, type de chacune des variables) et un ensemble de faits clos correspondant aux instances positives ou négatives du prédicat.
-

Le mode d'instanciation des prédicats de la théorie du domaine peut être aussi donné. Il correspond aux combinaisons possibles des variables du prédicat servant soit de donnée, soit de résultat. Lorsque le mode d'instanciation n'est pas donné explicitement, toutes les combinaisons sont alors testée par FOIL. Sa présence permet, au même titre que le typage des variables, de restreindre l'espace de recherche.

Le mécanisme d'induction de FOIL peut être conditionné par un certain nombre de paramètres comme l'acceptation ou non de la négation, le nombre de nouvelles variables introduites, ...

2.1.2 Syntaxe et exemple d'induction.

FOIL est implémenté en C et ne bénéficie donc pas de l'interprète de la programmation logique. Particulièrement, il ne permet pas la manipulation des listes en tant qu'objet PROLOG et donc l'accès aux éléments de tête ou de queue comme par exemple dans *foo(A,[B|C]):- foo(B,C).*
Les listes sont donc définies comme constantes alphanumériques: la liste [1,2,3] par exemple est représentée par la chaîne de caractères [123].

Prenons l'exemple de l'induction de MEMBER tel qu'il est donné par QUINLAN [Qui 90].
Deux types sont définis: ELT et LIST, tous deux discrets et non ordonnés.

```
ELT: 1, 2, 3.
LIST: [111], [112], [113],  ... , [333], [33], [3], *[].
(énumération incomplète)
```

Le caractère * devant la chaîne vide [] indique qu'elle peut être utilisée dans les cas de base.
Suit la définition de la théorie du domaine :

- - du prédicat cible *MEMBER(E,L)* qui signifie "l'élément E appartient à la liste L".
- - d'un prédicat *COMPONENTS(L1,E,L2)* qui signifie "la liste L1 a pour tête l'élément E et pour queue la liste L2".

```
member(ELT,LIST)
1,[1]
3,[3]
1,[11]
1,[13]
3,[13]
                    ....
;                   fin   des
exemples positifs
1,[]
1,[3]
1,[33]
1,[333]
3,[]
3,[1]
                    ....
.                   fin   des
exemples négatifs
```

```
*components(LIST,ELT,LIST) #--
/-##
[1],1,[]
[2],2,[]
[3],3,[]
[11],1,[1]
[12],1,[2]
[13],1,[3]
[21],2,[1]
22],2,[2]
[23],2,[3]
....
.                   fin des
exemples positifs

pas d'exemples négatifs donnés
explicitement
```

La clef #--/-## après l'entête de COMPONENTS correspond au mode d'instanciation du prédicat et signifie que pour l'utiliser, soit la première variable, soient les 2 dernières doivent être instanciées.

Le résultat de l'induction est un ensemble de clauses cohérentes avec la théorie du domaine. Chaque clause est constituée d'une disjonction de littéraux sur des prédicats du domaine, uniquement d'atomes si on n'a pas imposé que la définition soit sous forme de clauses de Horn ou des prédicats prédéfinis de FOIL, notamment l'égalité syntaxique et les relations d'ordre mathématiques.

L'apprentissage de l'exemple précédent fournit par exemple la définition suivante sous la forme PROLOG:

```
member(A,B) :- components(B,A,C).
member(A,B) :- components(B,C,D), member(A,D).
```

2.2 FOCL (First-Order Combined Learner).

FOCL ([PK 92], [PBS 92]) est un système mono-conceptuel et empirique qui combine des extensions de l'apprentissage inductif appliqué dans FOIL et l'apprentissage basé sur l'explication (en anglais, Explanation-Based Learning ou EBL).

L'Explanation-Based Learning ([MKK 86], [DM 86] et [Ell 89]) est une technique d'apprentissage analytique (c'est-à-dire transformation d'une théorie du domaine existante sous une forme plus utile) qui tente de formuler une généralisation à partir d'un seul exemple et en s'appuyant fortement sur une théorie du domaine. On parle d'explication car il s'agit plus de recompilation que de création ou d'extension de connaissances.
Le terme d'EBL regroupe en fait une grande variété de méthodes différentes, mais peut être présenté par un processus en deux étapes. La première consiste à produire une explication de la fonction ou du comportement de l'exemple cible. Cette explication est destinée à faire ressortir le principe général contenu dans l'exemple. La deuxième étape consiste à analyser l'explication produite et l'exemple de façon à construire une généralisation du concept. Les caractéristiques et les contraintes relatives à l'exemple sont généralisées autant que possible, tant que l'explication reste valide. Cette généralisation englobera alors tous les exemples qui pourront être compris par cette explication.
Le résultat de la combinaison des deux techniques d'apprentissage dans FOCL permet une optimisation de la construction des hypothèses. Par l'utilisation de l'EBL, le système s'appuie sur la théorie du domaine (quand elle est disponible) pour contraindre la recherche des littéraux pouvant être obtenus par le processus inductif.

2.2.1 Restrictions sur l'espace de recherche.

FOCL étend les capacités de FOIL notamment par:

- - l'utilisation de contraintes inter-arguments sur les prédicats pour réduire l'espace de recherche (typage et mode d'instanciation comme dans FOIL mais aussi contraintes sur la commutativité et l'unicité des variables).
- - l'utilisation de prédicats définis intentionnellement dont l'ensemble constitue la théorie du domaine pour l'EBL.
- - l'acceptation d'une règle partielle et éventuellement incorrecte comme approximation initiale du concept à apprendre. Cette règle pourra être complétée et corrigée lors de la phase de révision de théorie de l'apprentissage. La révision de théorie, supervisée par le système, consiste généralement à supprimer des éléments non significatifs de la théorie ou à en rajouter
- - une représentation graphique (sous sa version Macintosh) des éléments du domaine par un arbre ET/OU, ce qui rend ce système plus convivial et plus manipulable que les autres.

L'intérêt de FOCL repose non seulement sur l'utilisation de L'EBL en renforcement des mécanismes classiques d'induction mais aussi sur la présence d'un certain nombre d'opérateurs prédéfinis facilitant l'apprentissage de concepts. Ces opérateurs (appelés en anglais Relational Clichés [PK 92]) sont souvent indispensables pour obtenir une définition correcte d'un concept mais présentent l'inconvénient de ne pas être paramètrables et d'être trop peu décrits formellement dans les articles concernant FOCL.
Lorsque le système construit le corps des clauses de la définition du concept recherché et qu'elles couvrent encore des exemples négatifs, il est nécessaire d'y rajouter un ou plusieurs littéraux de façon à avoir la correction. Les Clichés servent alors à éviter l'explosion combinatoire due à l'essai des différentes paires ou triplets de littéraux. Il s'agit d'un ensemble de modèles génériques qui imposent des restrictions sur les littéraux utilisés dans la conjonction et sur leur variabilisation.
Les Clichés diffèrent des modèles de règles (voir par exemple l'utilisation des séries de langages abstraits par CLINT) par leurs restrictions appliquées à la fois sur les littéraux et leurs arguments.

2.2.2 Syntaxe et exemple d'induction.

En ce qui concerne le langage de description du domaine, FOCL représente les concepts (faits et règles) dans un formalisme LISP, c'est-à-dire sous la forme

```
(nom-relation var1 var2 ... varn)

((nom-relation var1 var2 ... varn)
       (nom-relationk var1 var2 ... vari)
       (nom-relationl var1 var2 ... varj))
```

Reprenons l'exemple de l'apprentissage de MEMBER tel qu'il est défini dans FOIL.
La théorie du domaine se compose donc des deux faits MEMBER et COMPONENTS, représentés ci-dessous sous forme textuelle directement utilisable par FOCL:

```
(def-type :LIST
   [] [111] [112] [113] [11] [121] ..... [33] [3] )

(def-type :ATOM
   1 2 3 )

(def-pred COMPONENTS
  :vars  (?A ?R ?L)
  :type  (:LIST :ATOM :LIST)
  :mode  ()
  :pos ( ([1] 1 []) ([2] 2 []) ([3] 3 []) ([11] 1 [1])
([12] 1 [2]) ......... ([333] 3 [33]) )
  :neg   :computed
  :induction    T
)

(def-pred MEMBER
  :vars  (?A ?L)
  :type  (:ATOM :LIST)
  :mode  ()
  :pos  ( (1 [1]) (3 [3]) (1 [11]) (1 [13]) (3 [13])
........ (3 [333]) )
  :neg  ( (1 []) (1 [3]) (1 [33]) (1 [333]) ....... (1
[332]) )
  :induction    T
)
```

Qu'il s'agisse d'un fait ou d'une règle, la structure du prédicat est la même et donne les informations suivantes:

- - *:vars* désigne le nom des arguments du prédicat.
- - *:type* correspond au typage de chaque argument.
- - *:mode* décrit le mode d'instanciation du prédicat.
- - *:pos* et *:neg* donnent les ensembles d'exemples positifs et négatifs du prédicat. Lorsque les exemples négatifs sont omis ou désignés par *:computed*, ils sont alors automatiquement générés par hypothèse du monde clos comme dans le cas de FOIL.
- -*:induction* indique que le prédicat va pouvoir être utilisé dans la définition de l'hypothèse induite. Cette information peut être utilisée pour tester un processus d'induction avec une partie réduite de la théorie du domaine sans avoir à la retranscrire.

L'apprentissage fournit alors la définition représentée graphiquement sur la figure 1.1 et qui correspond au programme PROLOG suivant :

```
member__learned_rule(A,B) :- components(B,A,C).
member__learned_rule(A,B)      :-      components(B,C,D),
member__learned_rule(A,D).
```

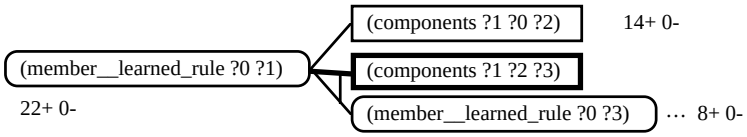


Figure 10 : Définition induite de MEMBER et couverture des exemples.

La figure 1.1 donne aussi des informations concernant la couverture des observations par la définition. Cette couverture est donnée à chaque noeud de la définition et est représentée par les symboles a+b-, a étant le nombre d'exemples positifs et b les négatifs. Dans le cas présent, la définition couvre les 22 exemples positifs de MEMBER dont 14 couverts par la première clause et 8 par la seconde.

Dans le cas d'une théorie du domaine intentionnelle, le concept à apprendre est constitué de deux éléments: un ensemble d'observations comme pour MEMBER et une approximation de sa définition. C'est la combinaison de ces deux éléments qui va permettre l'apprentissage du concept et la révision de la théorie incorrecte et/ou incomplète. Nous présentons dans l'évaluation de FOCL un tel exemple d'induction.

2.3 GOLEM.

GOLEM ([MF 92]) est un système empirique basé sur le principe d'inversion de la résolution introduit par S. Muggleton mais qui n'est pas capable d'inventer de nouveaux prédicats, l'invention de prédicats étant cependant la caractéristique principale de cette méthode. D'après les articles relatifs à GOLEM et à sa comparaison avec d'autres systèmes de PLI sur des problèmes particuliers ([MF 92], [Mug 92]), il semble qu'il soit le plus efficace lorsque l'on est en présence d'un nombre élevé d'exemples.

2.3.1 Restrictions sur l'espace de recherche.

Contrairement à FOIL ou à FOCL, GOLEM n'admet pas le typage des arguments des prédicats de la théorie du domaine. Cela entraîne l'incapacité d'utiliser l'hypothèse du monde clos sur un univers d'Herbrand fini pour générer automatiquement les contre-exemples à partir des exemples positifs: les exemples négatifs doivent donc être donnés explicitement. Pour obtenir une hypothèse induite sémantiquement correcte, l'absence de typage impose la présence soit de l'ensemble complet des exemples du concept (quand cela est possible bien évidemment), soit d'un sous-ensemble judicieusement choisi. Dans beaucoup de problèmes testés sur GOLEM, il suffit de supprimer certains exemples positifs ou négatifs pour que la définition induite ne correspond plus à nos attentes (généralement trop spécifique).

Deux outils sont principalement utilisés pour restreindre l'espace de recherche: le mode d'instanciation des prédicats et la détermination des prédicats de la théorie du domaine utilisés.
Comme dans le cas de FOIL, le mode d'instanciation correspond au choix fait sur chaque argument du prédicat (donnée ou résultat).
La détermination permet de désigner le prédicat cible et les prédicats qui pourront être utilisés dans la définition de l'hypothèse. De cette façon, nous pouvons forcer le système à induire une hypothèse récursive ou non et tester l'apprentissage sur une partie seulement de la théorie du domaine. Par défaut, toutes les combinaisons des modes d'instanciation sont testées, l'hypothèse peut être récursive et utilise tous les prédicats de la théorie du domaine.

2.3.2 Syntaxe et exemple d'induction.

Comme FOIL, la théorie du domaine et les observations du concept cible doivent être donnés de façon extensionnelle. Cependant, GOLEM autorise l'utilisation de symboles de fonction dans leur définition.
Un autre avantage de GOLEM par rapport à FOIL est qu'il utilise un interprète PROLOG et qu'il manipule donc directement ses objets. Particulièrement, les termes représentant les listes sont considérés comme tels par le système.

Prenons l'exemple de l'apprentissage de MEMBER défini dans FOIL.
La théorie du domaine est schématisée ci-dessous:

mode(member(-,+)).
determination(member/2, member/2).
determination(member/2, components/3).

exemples positifs	exemples négatifs
member(1,[1]). *member(3,[3]).* *.....* *member(1,[3,3,1]).* *member(3,[3,3,1]).* *member(3,[3,3,3]).* *components([1],1,[]).* *components([2],2,[]).* *.....* *components([3,3,2],3,[3,2]).* *components([3,3,3],3,[3,3]).*	*member(1,[]).* *member(1,[3]).* *member(1,[3,3]).* *member(1,[3,3,3]).* *member(3,[]).* *.....* *member(1,[3,2,2]).* *member(1,[3,2,3]).* `member(1,[3,3,2]).`

mode(member(-,+)) indique le mode d'instanciation du prédicat MEMBER (la première variable est un résultat alors que la deuxième est une donnée).
La relation determination indique sur quelle partie de la théorie du domaine GOLEM s'appuie pour générer les hypothèses. Dans ce cas, GOLEM tente d'apprendre le concept MEMBER de manière récursive - *determination(member/2, member/2)* - et en utilisant le prédicat COMPONENTS.
L'induction donne alors la définition suivante :

```
member(A,[A | B]).
member(A,[B, C | D]) :- member(A,[C | D]).
```

Nous pouvons remarquer que la capacité de GOLEM à manipuler directement les listes rend inutile l'emploi du prédicat COMPONENTS dans la définition du concept.

2.4 MacCLINT.

MacCLINT (version Macintosh de CLINT, [Der 91], [DB 92]) est le seul système étudié qui soit incrémental et multi-conceptuel, c'est-à-dire qu'il apprend plusieurs définitions de prédicats à partir d'exemples fournis les uns après les autres et de l'intervention de l'utilisateur (appelé aussi oracle).
Outre cet aspect, il diffère des autres systèmes par le fait qu'il accepte une description intentionnelle de la théorie du domaine, des symboles de fonction d'arité 1 et des contraintes d'intégrité (propriétés sur les connaissances de la base, par exemple le fait qu'un parent soit ou un père ou une mère). La vérification de ces contraintes peut amener CLINT à supprimer de sa base des concepts contradictoires (contraintes, faits ou même clauses). C'est l'une des caractéristiques de CLINT qui peut être mise en parallèle avec la révision de théorie de FOCL.

2.4.1 Restrictions sur l'espace de recherche.

Pour restreindre l'espace de recherche, CLINT utilise une série de langages abstraits de description des concepts. Les langages sont abstraits car indépendants de toute théorie du domaine mais deviennent spécifiques en les unifiant avec les noms des prédicats d'une théorie du domaine.
Toutes les clauses induites ou faisant partie de la théorie du domaine appartiennent à l'un de ces langages. Les langages abstraits L_i d'une série sont ordonnés selon l'expressivité croissante des clauses, c'est-à-dire qu'une clause pouvant être formulée par un langage L_i pourra l'être par les langages L_j , i<j, de la série. Quand CLINT découvre un concept non formulable dans un langage, il passe au langage suivant de la série.
Ces langages sont décrits formellement dans [Der 91] mais nous en donnons ici une idée intuitive.
Le premier langage L_1 requiert que toutes les variables présentes dans la tête d'une clause soient présentes aussi dans sa queue et réciproquement, par exemple *père(X,Y) :- homme(X) , parent(X,Y)* (voir l'exemple de théorie du domaine ci-dessous).
L_2 permet l'utilisation de variables existentielles telles qu'elles n'apparaissent au plus qu'une seule fois dans la clause et qu'au plus une soit présente dans chaque littéral de la clause. La clause est-un-père*(X) :- homme(X) , parent(X,Y)* peut être exprimée dans L_2 mais pas dans L_1. Le langage L_3 permet l'utilisation d'occurrences multiples de variables existentielles, par exemple la clause *grand-père(X,Y) :- père(X,Z), parent(Z,Y)* qui ne peut être exprimée ni dans L_1 ni dans L_2.
Les exemples de langages ci-dessus ne sont pas exhaustifs et il peut être préférable d'en définir de nouveaux pour certains problèmes.

2.4.2 Syntaxe et exemple d'induction.

La syntaxe utilisé est celle du PROLOG standard. Par exemple, dans le cas d'un problème d'apprentissage des liens de parenté (*grand-parent(X, Y)*, *père(X, Y)* , ...), la théorie du domaine et les contraintes suivantes peuvent être définies :

```
parent(alice, rose).                                parent(X, Y)
=> père(X, Y) , mère(X, Y).
parent(leon, rose).                                      true =>
personne(X).
parent(rose, luc).                                  mère(luc) =>
false.
     .......
ancêtre(X, Y):-ancêtre(X, Z),ancêtre(Z, Y).
```

Étudions un exemple d'induction en prenant encore une fois l'exemple de l'apprentissage de MEMBER défini pour FOIL.
Étant incrémental, CLINT ne dispose comme théorie du domaine que du prédicat COMPONENTS donné en extension. Les observations de MEMBER sont saisies par l'utilisateur et généralisées les unes après les autres par le système pour fournir une définition.

Le premier exemple introduit est déclaré positif: *member(1, [123])*
CLINT génère alors une clause couvrant cet exemple et trouve

```
member(X, Y) if
        components(Y, X, Z) .
```

L'oracle décide que cette définition n'est pas suffisante et saisit alors un deuxième exemple positif lui aussi:
member(2, [123])

La définition, après computation et vérification des contraintes (c'est-à-dire la couverture des exemples précédents), est alors :

```
member(X, Y) if
        components(Y, X, Z) .
member(X, Y) if
        member(Z, Y) and member(X, U) .
```

De nouveau insatisfait par la définition, l'oracle saisit alors l'exemple négatif *member(1, [23])*
CLINT continue à construire sa définition et dispose alors de suffisamment de matière pour rentrer dans une phase d'interaction avec l'oracle: il propose un certain nombre de faits dont la validation par l'oracle va lui permettre de corriger sa définition:
member(2, [23]) déclaré positif
member(1, [1]) déclaré positif

La définition résultant de cette apprentissage est alors

```
member(_X, _Y) if
        components(_Y, _X, _Z) .
member(_X, _Y) if
        components(_Y, _Z, _U) and member(_X, _U) .
```

Contrairement aux systèmes empiriques où le processus d'induction s'arrête lorsque tous les exemples ont été traités, CLINT nécessite l'intervention de l'oracle pour décider de la validité de l'hypothèse induite ce qui, sur des problèmes complexes, peut s'avérer délicat. Néanmoins, cette validité peut se déterminer par la convergence de l'hypothèse, c'est-à-dire que l'introduction de nouvelles observations ne la modifie plus.

3 Évaluation comparée des systèmes.

Le but de l'évaluation est de se faire une idée des capacités d'induction de chacun des 4 systèmes étudiés vis-à-vis de leurs restrictions respectives sur le langage de description utilisé et de leur paramétrisation. Ce ne sont pas

leurs capacités purement quantitatives (vitesse d'exécution, mémoire utilisée, ...) qui nous intéressent lors de cette évaluation mais plutôt qualitatives, c'est-à-dire leur habilité à fournir des réponses équivalentes (sémantiquement ou syntaxiquement) pour un même problème. De plus, nous nous intéressons particulièrement à l'obtention d'une définition qui soit la plus 'compréhensible' possible, c'est-à-dire la plus facile à interpréter syntaxiquement et sémantiquement. Nous donnons cependant dans l'**Annexe A** un résumé quantitatif de l'évaluation.

Cette évaluation des systèmes de PLI s'est faite en plusieurs étapes:

- - Prise en main des systèmes (syntaxe et restrictions des langages de description des connaissances, paramétrisation du processus d'induction).
- - Test et analyse des exemples fournis avec chaque système (et donc généralement conçus pour être efficaces).
- - Perturbation de ces exemples (changement du typage, suppression des contre-exemples et activation de l'hypothèse du monde clos, respécification de la théorie du domaine, ...) pour mettre en évidence leur résistance aux données incorrectes,
- - apprentissage d'un même exemple (après modification de la syntaxe) sur chacun des systèmes et comparaison (syntaxique dans le meilleur des cas, sémantique sinon) des résultats obtenus.

Les exemples d'application de la PLI, concrets ou abstraits, sont nombreux et divers: traitement des listes (MEMBER, QUICKSORT, REVERSE, ...), application à des domaines numériques (fonction Ackermann, multiplication, fonction combinatoire, ...), problèmes 'réels' (problème de la parenté, ...). Cependant, pour satisfaire aux tests multi-systèmes, l'un de ces exemples a été particulièrement traité: il s'agit du KRK-endgame.

3.1 Le problème du KRK-endgame.

Le problème du KRK-endgame (King-Rook-King, en français Roi-Tour-Roi) est un exemple très abordé dans la littérature ([Bai 92],[Qui 90]) et connu comme l'un des plus faciles des échecs. Son traitement par la PLI consiste à induire un programme logique capable de reconnaître une position illégale lorsque seuls le roi blanc (WK), la tour blanche (WR) et le roi noir (BK) sont sur le plateau et que c'est aux blancs de jouer. Un placement aléatoire des pièces sur l'échiquier donne $64^3 = 262144$ positions dont approximativement 1/3 sont illégales.
Une position illégale peut être définie formellement par les 3 faits suivants (cf la théorie du domaine de FOCL, figure 2.2):

- - 2 ou 3 pièces sont sur la même case.
- - le roi blanc met en échec le roi noir (pièces placées sur 2 cases adjacentes).
- - la tour blanche met en échec le roi noir (pièces placées sur une même ligne telles que le roi blanc ne se trouve pas entre la tour et le roi noir).

Le prédicat *illegal(WK_x,WK_y,WR_x,WR_y,BK_x,BK_y)* est utilisé pour représenter les positions (ligne et colonne) de chacune des 3 pièces. Deux exemples de positions sont représentées sur la figure 2.1.

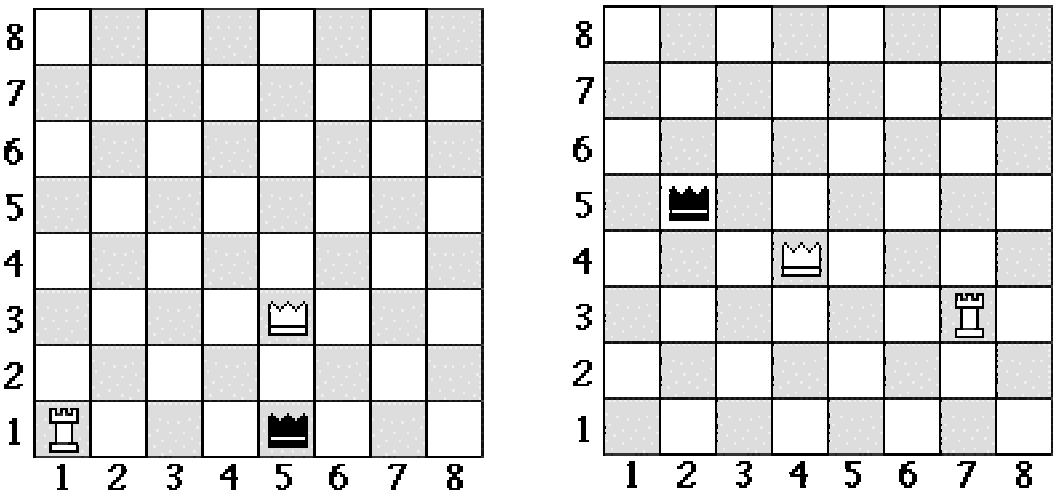


Figure 11 : Exemple de positions du problème KRK-endgame

Extraite de FOCL, la théorie du domaine associée contient les données suivantes (figure 2.2):

- - types ligne et colonne : entier de 1 à 8
- - *near-row(X,Y)* et *near-column(X,Y)* signifient respectivement "la ligne (colonne) X est adjacente à la ligne (colonne) Y" c'est-à-dire X=Y-1 ou X=Y+1
- - *equal-row(X,Y)* et *equal-column(X,Y)* signifient respectivement "la ligne (colonne) X et la ligne (colonne) Y sont égales" c'est-à-dire X=Y
- - *between-row(X1,X,X2)* et *between-column(X1,X,X2)* signifient respectivement "la ligne (colonne) X est strictement comprise entre les lignes (colonnes) X1 et X2" c'est-à-dire X1<X<X2
- - *king-attack-king, rook-attack-king* et *king-not-between-row* sont définis en intention avec les faits précédents et n'apportent pas d'autres informations que syntaxiques.

Ces prédicats n'ont que des exemples positifs correspondants à l'énumérations de tous les cas possibles. Il est évident que les relations sur les lignes sont redondantes avec celles sur les colonnes mais favorisent la compréhension de la règle *illegal*. Celle-ci contient 79 exemples positifs et 121 négatifs, soit moins de 0,1% de l'ensemble des positions possibles.

Le problème du KRK-endgame tel qu'il est présenté ici provient de FOCL. C'est pour cette raison que nous commençons l'évaluation par ce système, de façon à obtenir une hypothèse qui serve de base comparative pour les autres systèmes.
Décrivons maintenant le comportement de chaque système.

3.2 FOCL

La figure 2.2 représente, sous forme d'un graphe ET/OU, la théorie du domaine relative au problème du KRK-endgame. Y est indiqué le nombre d'exemples positifs et négatifs couverts par chaque noeud.
Nous pouvons instantanément identifier les branches qui ne couvrent aucun exemple positif (les noeuds grisés), ce qui montre que cette définition de ILLEGAL est sur-spécifiée. Elle est cependant correcte (aucun exemple négatif couvert) et complète (tous les exemples positifs le sont).

Après apprentissage et optimisation (suppression des termes non significatifs sémantiquement), nous obtenons l'hypothèse donnée graphiquement sur la figure 2.3.
Nous pouvons remarquer que les prédicats between-row et between-column ne sont pas utilisés dans la définition: le gain d'information que donne leur instanciation durant l'apprentissage n'est pas suffisant par rapport à ceux des autres termes pour qu'ils soient retenus dans une clause.

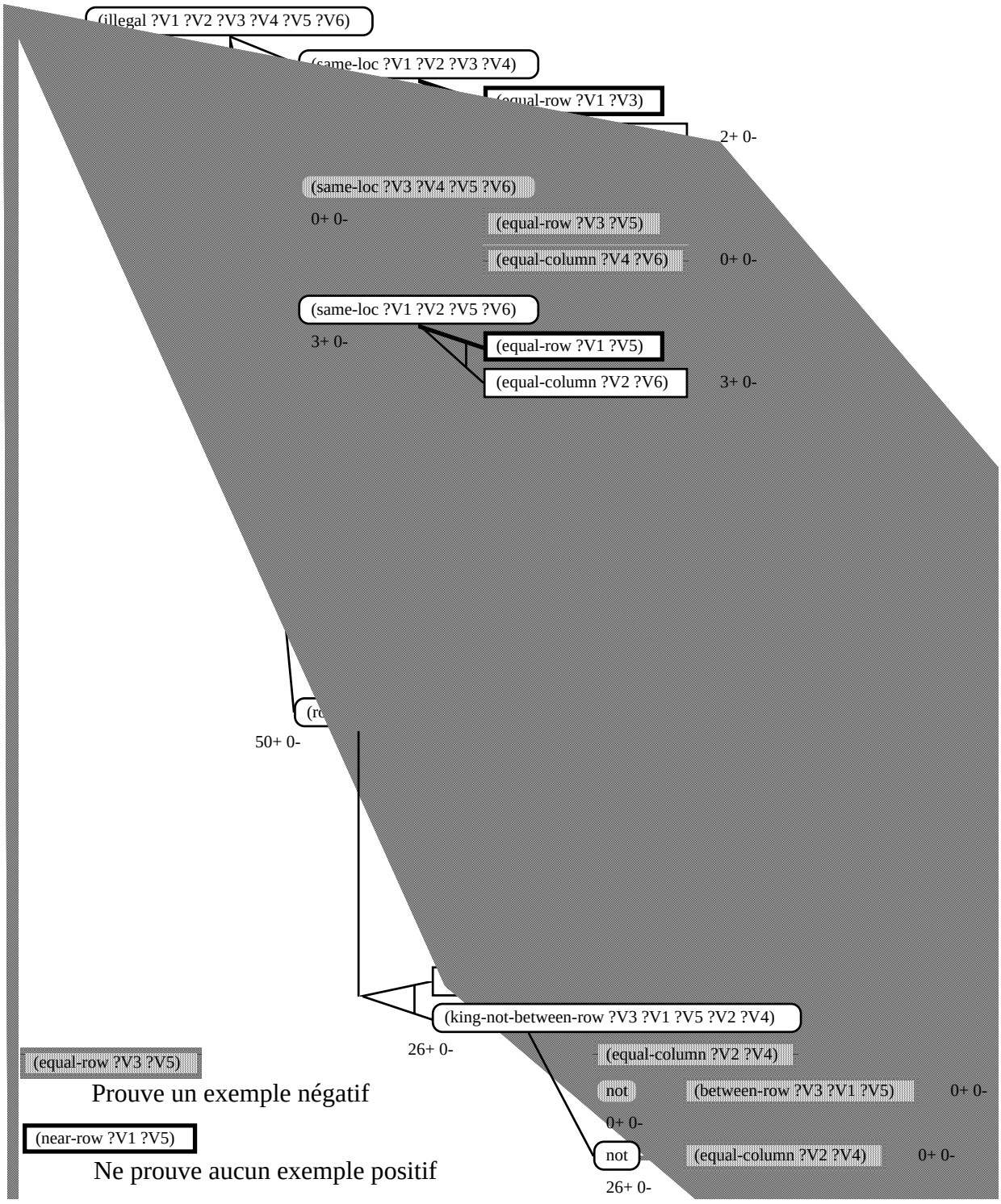


Figure 12 : Théorie du domaine et couverture de ILLEGAL

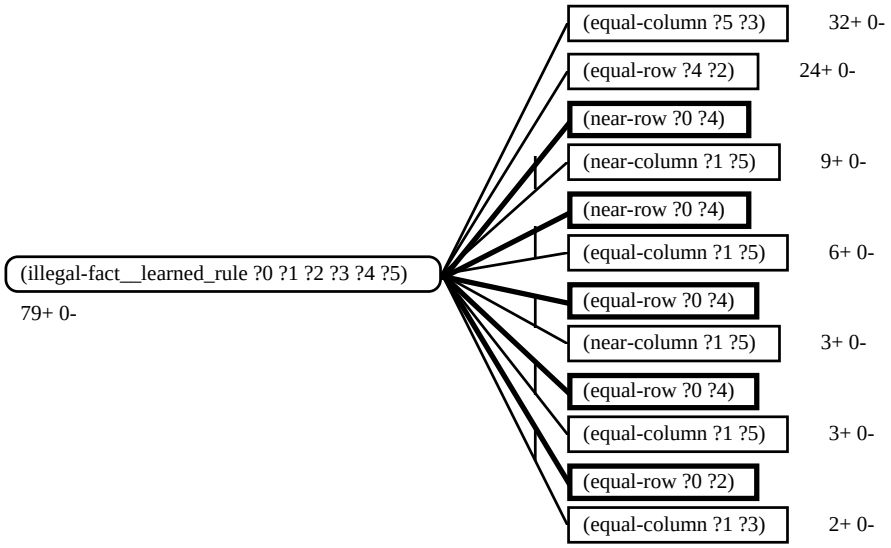


Figure 13 : Définition induite de ILLEGAL

La révision de la théorie proposée par FOCL (figure 2.4) revient dans ce cas à supprimer les termes de la règle ILLEGAL qui ne sont pas significatif sémantiquement c'est-à-dire, comme nous pouvions nous y attendre, les noeuds grisés de la théorie du domaine (figure 2.2).

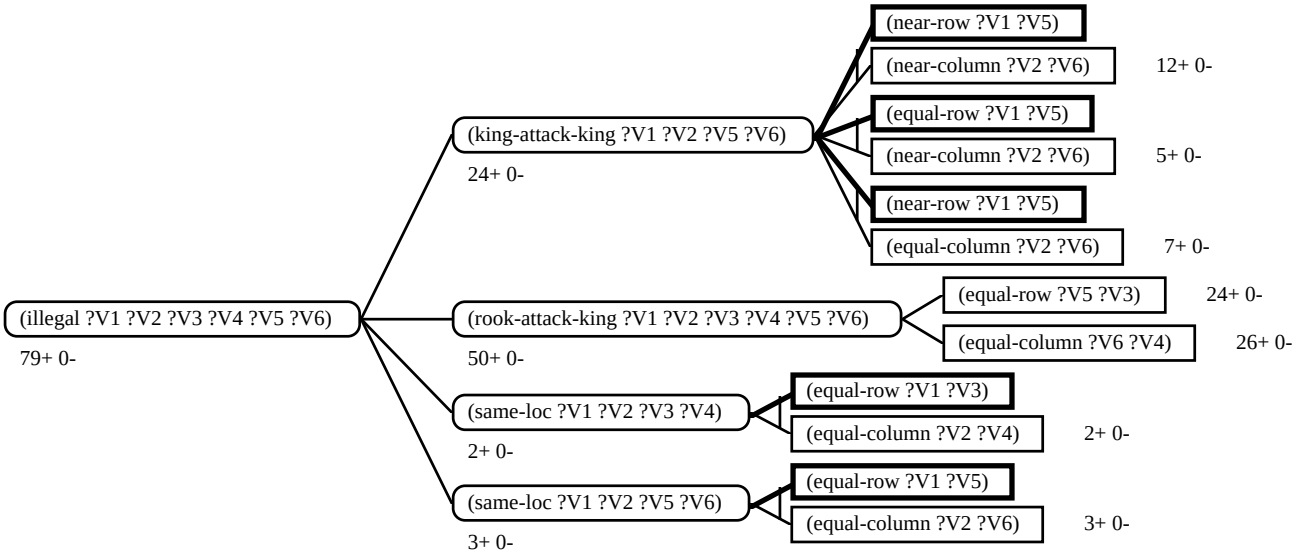


Figure 14 : Révision de la théorie de ILLEGAL

Analysons plus en détail les capacités de révision de théorie de FOCL.
Nous reprenons le même exemple mais nous rendons la théorie du domaine incorrecte et incomplète par négation de la première clause de ILLEGAL (figure 2.5). L'analyse de la couverture de ILLEGAL donne alors 77+120- (au lieu des 79 exemples positifs et 121 exemples négatifs précédents).

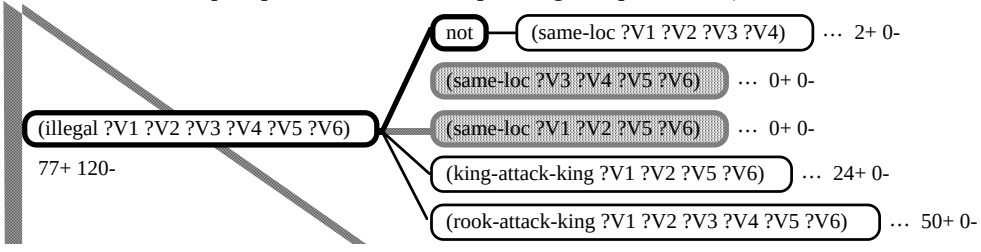


Figure 15 : Théorie incomplète et incorrecte de ILLEGAL

La définition induite est alors la même que précédemment (cf figure 2.3) mais la révision de la théorie se fait cette fois en deux étapes:

- - disjoindre les termes correspondant à la clause *(same-loc ?V1 ?V2 ?V3 ?V4)* supprimée et réobtenue durant l'apprentissage à la définition de ILLEGAL. Ils sont par défaut rattachés au sommet du graphe mais peuvent être placés ailleurs par l'utilisateur sans changer l'exactitude de l'hypothèse.
- - supprimer les termes sans signification sémantique.

La définition révisée correspond alors à la précédente, modulo la tête de la clause *same-loc*.

D'autres manipulation peuvent être faites sur la théorie et toutes, sous réserve de bien suivre les indications fournies par le système pour la révision, retourneront la même définition (au moins sémantiquement). Comme nous l'avons signalé dans la présentation de FOCL, cette partie de la révision de la théorie est à la seule charge de l'utilisateur et peut aboutir à une mauvaise reformulation de la théorie si les instructions du système ne sont pas suivies correctement.

3.3 FOIL

FOIL, comme nous l'avons précisé dans sa présentation, n'accepte pas de définition en intention de la théorie. Nous sommes donc obligé de supprimer tous les prédicats (qui ne servent en fait qu'à assurer une cohésion syntaxique) et de ne conserver que les faits.

Dès la première exécution de FOIL, les faiblesses du typage et l'influence des paramètres d'induction apparaissent.
FOIL utilise en effet le typage des prédicats pour restreindre l'espace de recherche par CWA et non pour assurer la cohérence des données. Dans le cas du KRK, les types colonnes et lignes, ainsi que les faits s'appliquant sur chacun des deux types sont redondants, ce qui amène FOIL, lors du calcul du gain d'information à les trouver identiques et à les confondre. Dans ces conditions, l'apprentissage de ILLEGAL se fait donc avec un seul type et une seule série de faits. Pour pallier à ce problème, nous aurions pu aussi changer l'identification des colonnes en a,b,....,h.

La première évaluation s'est faite sans aucun paramètre d'induction et le résultat s'avère relativement inattendu :

```
illegal(A,B,C,D,E,D).
illegal(A,B,C,D,C,F).
illegal(A,B,A,B,E,F).
illegal(A,B,C,D,E,F)        :-        not(between(A,G,E)),
not(between(B,G,F)).
```

Nous pouvons faire plusieurs constatations sur cette l'hypothèse :

- - l'utilisation de la négation qui n'est pas précisément conforme à une utilisation sous PROLOG
- - l'introduction de nouvelle variables non utilisées dans la tête de la clauses.

Cette hypothèse est peut être correcte sémantiquement mais laisse à désirer pour l'interprétation, à cause notamment de la dernière clause. En effet, les 3 premières clauses correspondent respectivement, dans la définition fournie par FOCL (cf figure 2.3), à

```
(equal-column ?V5 ?V3)
(equal-column ?V4 ?V2)
(equal-row ?V0 ?V2) , (equal-column ?V1 ?3)
```

Mais déterminer les équivalents dans FOCL de la dernière clause est plus délicat.
Pour remédier à cette faible définition, nous recommencons l'induction en inactivant l'utilisation de la négation. L'hypothèse obtenue est la suivante:

```
illegal(A,B,C,D,E,D).
illegal(A,B,C,D,C,F).
illegal(A,B,A,B,E,F).
```

```
illegal(A,B,C,D,E,F) :- near(A,E), near(B,F).
illegal(A,B,C,D,E,B) :- near(A,E).
illegal(A,B,C,D,A,F)  :-  between(C,A,G),  between(H,B,I),
between(F,J,G).
```

Nous retrouvons cette fois deux autres clauses présentes dans l'hypothèse fournie par FOCL mais la dernière pose encore un problème d'interprétation résolu cette fois en interdisant l'emploi de toute nouvelle variable. La définition obtenue est sémantiquement identique à celle fournie par FOCL :

```
illegal(A,B,C,D,E,D).
illegal(A,B,C,D,C,F).
illegal(A,B,C,D,E,F) :- near(A,E), near(B,F).
illegal(A,B,C,D,E,B) :- near(A,E).
illegal(A,B,C,D,A,B).
illegal(A,B,C,D,A,F) :- near(B,F).
illegal(A,B,A,B,E,F).
```

3.4 GOLEM

L'une des caractéristiques de GOLEM est son incapacité à utiliser l'hypothèse du monde clos (CWA) pour générer l'ensemble des exemples négatifs à partir du corpus positif. Dans un certain nombre de cas (ceux notamment où l'ensemble des exemples positifs est réduit par rapport à l'univers), cette caractéristique s'avère être une faiblesse mais dans le cas présent, les exemples positifs ne forment certainement pas tous les cas possibles de positions illégales et donc ne permettent pas une génération automatique des contre-exemples.

Lors de notre première session d'apprentissage, aucune restriction sur les relations à utiliser n'est imposée et GOLEM tente alors d'induire la définition de ILLEGAL de manière récursive. Une telle définition est peut être tout aussi correcte que celles non récursives fournies par les autres systèmes mais le processus d'induction prend alors une ampleur en durée telle qu'il est nécessaire de le stopper (plus de 30 mn de calcul sur un Sun sans détermination de la première clause). Aucune hypothèse récursive ne semble être inductible dans une durée raisonnable.

Pour éviter ce problème, nous restreignons la détermination des prédicats (voir présentation de GOLEM) à tous sauf à ILLEGAL. Cette fois, l'induction peut se faire correctement et fournit le résultat suivant:

```
illegal(1,8,3,5,2,8).
illegal(A,B,C,4,D,B).
illegal(A,B,C,D,A,B).
illegal(A,B,C,D,C,E).
illegal(A,B,C,D,E,D).
illegal(5,A,5,B,C,D) :- between(5,A,E).
illegal(A,3,B,C,7,D) :- near(7,A).
illegal(A,B,C,C,A,D) :- near(B,D).
illegal(A,B,C,D,E,F) :- near(A,E), near(B,F).
illegal(3,A,3,B,C,D)  :-  between(B,E,C)  ,  near(B,E)  ,
near(C,E).
illegal(A,4,B,C,D,E) :- near(4,E) , near(B,F) , near(D,F).
```

A la vue de cette définition, plusieurs remarques peuvent être faites. GOLEM, comme FOIL et CLINT, est capable d'utiliser une ou plusieurs constantes du domaine pour caractériser une forme particulière de la définition. Dans le cas présent, nous remarquons cependant que l'utilisation des constantes résulte du fait que GOLEM ne semble pas être capable de déterminer une définition suffisamment générale du concept. Preuve en est la première clause de la définition qui est un cas particulière de position non couverte par le reste des clauses.
De la même manière que FOIL donnait une définition difficile à interpréter avec l'utilisation de nouvelles variables et la négation de *between*, nous pouvons supputer la même cause ici, bien que GOLEM n'autorise pas l'utilisation de la négation. Le prédicat *between* semble alors être le seul responsable.
L'apprentissage est alors recommencé en l'excluant de la théorie du domaine et on obtient alors la définition de ILLEGAL suivante:

```
illegal(A,B,A,B,C,5).
```

```
illegal(A,B,C,4,D,B).
illegal(A,B,C,D,4,B) :- near(4,C).
illegal(A,B,C,D,A,B).
illegal(A,B,C,D,C,E).
illegal(A,B,C,D,E,D).
illegal(A,B,C,D,A,E) :- near(B,E).
illegal(A,B,C,D,E,B) :- near(A,E).
illegal(A,B,C,D,E,F) :- near(A,E), near(B,F).
```

Les 6 dernières clauses de ILLEGAL se retrouvent (à un renommage près) dans la définition fournie par FOIL ou FOCL mais les 3 premières n'ont pas leur équivalent. De la même façon que précédemment, l'utilisation de constantes du domaine n'apporte rien à la compréhension du programme logique. Il semble bien que GOLEM n'arrive pas à généraliser suffisamment les exemples fournis. Une vérification de la syntaxe de la théorie du domaine n'a pas mis en évidence une éventuelle erreur de transcription. Il apparaît donc que les exemples ne sont pas suffisamment représentatifs pour permettre une bonne généralisation du concept par GOLEM.

3.5 MacCLINT

CLINT, malgré ses résultats pour l'apprentissage de MEMBER, présente dans ce cas ses limites. Étant incrémental, les observations de ILLEGAL ne sont pas dans le système mais lors de la saisie du premier exemple (positif ou négatif), celui-ci commence le processus d'induction par rechercher les littéraux possibles généralisant cet exemple. Ce processus semble prendre plus de temps et surtout de mémoire que CLINT ne peut le supporter car il aboutit à un plantage généralisé, ceci quel que soit l'exemple saisi. Plusieurs versions réduites de la théorie du domaine (sans théories intentionnelles par exemple) ont été testées mais le même phénomène s'est reproduit.

Cela est certainement causé par l'inadéquation des séries de langages de description des concepts utilisées (voir présentation de CLINT) : les concepteurs du système [Der 91] recommande en effet de les adapter au problème étudié. Devant l'ampleur d'une telle redéfinition des séries de langages, nous avons préféré nous en abstenir dans le cadre de l'évaluation.

4 Conclusions.

Une synthèse de l'évaluation est représentée dans le tableau suivant. Nous y avons mis en évidence les caractéristiques comportementales (forme de l'apprentissage, capacité d'invention de prédicats, ...), les restrictions apportées au langage de description des connaissances (littéraux négatifs, théorie du domaine en intention ou en extension, ...) et les quelques éléments paramètrables par l'utilisateur pour restreindre l'espace de recherche (typage et mode d'instanciation des arguments).

	FOCL	**FOIL**	**CLINT**	**GOLEM**
Caractéristiques comportementales				
Apprentissage interactif			**X**	
Apprentissage incrémental			**X**	
Apprentissage multi-conceptuel			**X**	
Révision de théories	**X**		**X**	
Invention de prédicats	**X**		**X**	
Restrictions du langage de représentation				
Pas de littéraux négatifs	**P**	**P**		**X**
Pas de symboles de fonctions	**X**	**X**	**X**	
Théorie du domaine intentionnelle	**X**		**X**	
Restrictions de l'espace de recherche				
Modes d'instanciation des prédicats	**X**	**X**		**X**
Typage des arguments	**X**	**X**		

Figure 16 : Caractéristiques des systèmes évalués

Le symbole (X) signifie que le système possède cette propriété, le (P) signifie qu'il peut l'utiliser.

A partir de cette évaluation comparée de quelques systèmes de PLI, nous pouvons faire un certain nombre de remarques quant à l'apprentissage de concepts.

- - Importance des restrictions imposées au langage de description. Le typage des arguments des relations, le mode d'instanciation de celles-ci, ... tous ces paramètres vont permettre un élagage non négligeable de l'espace de recherche lors de l'induction. Bien entendu, ces restrictions peuvent être trop limitatives: dans certains cas, nous pouvons souhaiter utiliser par exemple des symboles de fonctions ou des termes non clos dans la théorie du domaine.
-
- - Taille du corpus d'exemples positifs et négatifs associé à un concept. Plus nous disposons d'un corpus important, plus la définition sera valide non seulement vis-à-vis de ces observations mais aussi dans le cas général. L'utilisation de l'hypothèse du monde clos (CWA) permet dans une certaine mesure de répondre à cette exigence. Mais cela impose la donnée explicite de tous les exemples positifs du concept et un typage des constantes (pour limiter l'univers de Herbrand), ce qui n'est pas toujours réalisable.
-
- - Révision de la théorie. Il existe une différence fondamentale entre l'apprentissage proposé par FOIL, GOLEM ou CLINT et celui proposé par FOCL. Dans le premier cas, le système se contente de trouver une définition à un concept en tenant compte d'éléments externes (la théorie du domaine) et internes (les observations) alors que dans le second, la définition en elle-même est relativement secondaire par rapport à la révision de la théorie introduite grâce à ces hypothèses.
-
- - Comportement de l'induction. La question se pose de l'avantage et de l'utilité d'un système incrémental par rapport à un système empirique. Nous ne pensons pas qu'on puisse y répondre en général: cela dépend énormément du contexte d'utilisation. Néanmoins, nous pouvons dire que dans le cas empirique, l'apprentissage est plus efficace particulièrement grâce à l'utilisation d'heuristiques comme le gain d'information de FOIL.

Programmation Logique Inductive et génération d'un modèle des croyances de l'apprenant

1 Introduction.

Dans les chapitres précédents, nous avons présenté les objectifs de la Programmation Logique Inductive et une évaluation de quelques systèmes de PLI. Nous allons maintenant étudier de quelle manière cette technique d'apprentissage automatique peut être utilisée pour générer automatiquement un modèle logique des croyances de l'apprenant dans le cadre du Tuteur d'Aide Logique à la Construction de figures géométriques TALC.

Rappelons brièvement le principe de la PLI (voir deuxième chapitre) : nous disposons d'une théorie du domaine **B**, d'un ensemble d'exemple **E+** et de contre-exemples **E-** illustrant un ou plusieurs concepts à apprendre et nous cherchons à construire une hypothèse **H** telle que **H** couvre tous les exemples positifs et ne couvre aucun exemple négatif, relativement à la théorie du domaine. Le lien entre génération d'un modèle logique cohérent avec les productions d'un apprenant et Programmation Logique Inductive est alors évident:

- - **B** correspond au domaine d'expertise.
- - **E+** et **E-** correspondent aux réponses (correctes ou incorrectes) de l'apprenant relatives aux problèmes posés.
- - **H** permet de caractériser les processus que l'apprenant met en oeuvre pour ses réponses.

Dans une première partie, nous présentons les objectifs d'un modèle des croyances de l'apprenant, sa définition et ses intérêts dans le cadre de TALC. La deuxième partie de ce chapitre propose une mise en oeuvre de ce modèle par la PLI. La troisième partie est consacré à la présentation des premiers résultats expérimentaux obtenus. Nous concluons ce chapitre par une mise en évidence des problèmes que pose cette approche.

2 Modèle des croyances de l'apprenant.

Par modélisation des croyances de l'apprenant, notre objectif est de disposer d'un modèle qui puisse rendre compte des conceptions de la géométrie que possède l'apprenant. Nous parlons de *croyances* car notre modèle doit englober aussi bien les conceptions correctes que les incorrectes. Cette modélisation des croyances de l'apprenant se concrétise en se plaçant sous l'hypothèse didactique de cohérences des raisonnements de celui-ci, c'est-à-dire que nous considérons que les réponses incorrectes à un problème le sont uniquement à cause de ses conceptions.
En utilisant les techniques de la PLI, notre objectif est de générer un modèle logique qui soit cohérent avec les observations du comportement de l'apprenant et qui utilise le même formalisme de représentation des connaissances que la théorie du domaine.

2.1 Définition du modèle des croyances.

Rappelons le principe du diagnostic d'erreur dans TALC.
Soit ES l'ensemble des spécifications fournies par l'enseignant au cours d'une session d'apprentissage.

$$ES = \{S_1, S_2, \ldots, S_n\}$$

Soit EF l'ensemble des constructions réalisées par l'apprenant durant cette même session d'apprentissage, chacune correspondant à l'une des spécifications de ES.

$$EF = \{F_{11}, \ldots, F_{1m_1}, \ldots, F_{n1}, \ldots, F_{nm_n}\}$$

TALC fournit un diagnostic sur ces constructions F_{ij} (construction correcte ou incorrecte). Nous disposons donc d'un sous-ensemble $EFF \subseteq EF$ qui contient les construction erronées de l'apprenant.
D'après le contrat didactique défini dans TALC (voir premier chapitre), nous savons que la correction d'une construction F vis-à-vis de sa spécification S et de la théorie géométrique TIG s'exprime logiquement par

$\forall F \in EF \quad \exists S^* \text{ tq } TIG \vdash S^* \Leftrightarrow F$ (S* étant une extension de S)

Le problème est maintenant de mettre en évidence les raisonnements effectués par l'apprenant (ou les connaissances erronées) et qui lui ont permis de réaliser les constructions de EF. Ces raisonnements doivent permettre de valider aussi bien les constructions correctes vis-à-vis de TIG que les incorrectes.
En d'autres termes, l'objectif de la modélisation des croyances de l'apprenant est de déterminer une Théorie de l'Apprenant en Géométrie (TAG) cohérente avec toutes les productions de l'apprenant, c'est-à-dire

$\forall F_{ij} \in EF \quad \exists S^* \text{ tq } TAG \vdash S^* \Leftrightarrow F_{ij}$ (S* étant une extension de S_{ij})

Afin d'illustrer cette génération d'une Théorie Géométrique de l'Apprenant TAG, nous allons prendre un exemple que nous traitons tout au long de ce travail : le cas de la symétrie orthogonale d'un segment.
Dans leurs travaux, D. GRENIER [Gre 88] et S. TAHRI [Tah 93] ont isolé un certain nombre de conceptions correctes ou incorrectes de l'élève liées à la symétrie orthogonales d'un segment par rapport à une droite. Ces conceptions forment un corpus intéressant pour exploiter notre modèle. Leurs traductions dans TALC sont présentées en détails dans l'**Annexe B** mais nous en donnons deux ici:

- (1) la conception "parallélisme" qui correspond à un amalgame fait par l'apprenant entre les notions de symétrie et de parallélisme (figure 1.1.A)
-
- (2) la conception "report perceptiblement orthogonal à l'axe" qui correspond à une construction semi-globale par l'apprenant (c'est-à-dire que la construction de la figure ne s'appuie pas entièrement sur d'autres objets géométriques). Dans ce cas, seule l'extrémité A' du segment est construite, B' étant placé au jugé sur une droite orthogonale à l'axe (figure 1.1.B).

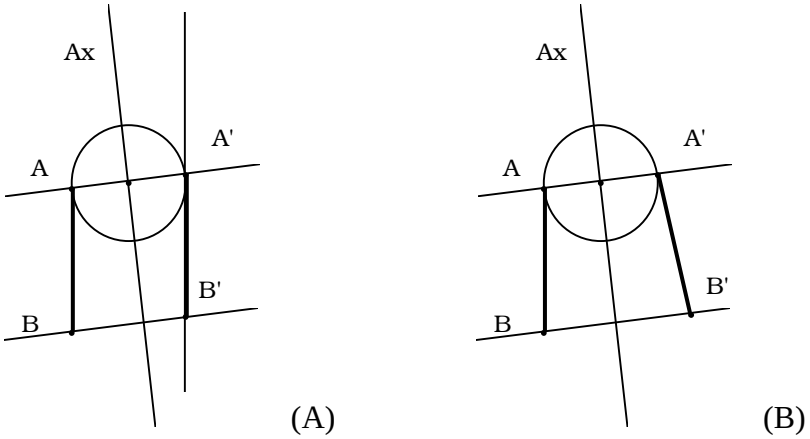


Figure 17 : Construction de la symétrie par procédures incorrectes.

La génération du modèle des croyances de l'apprenant va donc réviser la théorie instrumentale TIG par rapport à ces procédures. En supposant l'existence d'une théorie TIG suffisamment complète et cohérente par rapport à ce problème de la symétrie, nous souhaiterions par exemple obtenir les théories TAG suivantes :

- - le prédicat parallèle(X,Y) existant dans le théorie du domaine, le système pourrait être capable d'induire TAG_1 , qui représente bien le fait que l'apprenant confond les notions de symétrie et de parallélisme.

  $(1) \quad TAG_1 = TIG \cup symétrique([AB],[A'B']) \Leftarrow parallèle([AB],[A'B'])$

- - dans le deuxième cas, nous voulons faire ressortir le fait que la construction réalisée par l'apprenant est une construction semi-globale: les distances de l'axe à B et B' sont a peu près égale (B' est placé au jugé sur la droite orthogonale). L'utilisation d'un des aspects le plus intéressant de la PLI apparaît dans TAG_2 : l'invention d'un prédicat *a-peu-près-egale(X,Y)* qui n'existe pas dans la théorie du domaine associé à ce problème et qui permettrait de généraliser de façon plus efficace le concept.

  (2) TAG_2 = TIG ∪ symétrique([AB],[A'B']) ⇐ perpendiculaire([AB],Ax)

  perpendiculaire([A'B'],Ax)

  distance(A,Ax,d1)

  distance(A',Ax,d2)

  égale(d1,d2)

  distance(B,Ax,d3)

  distance(B',Ax,d4)

  à-peu-près-égale(d3,d4)

Les hypothèses induites ci-dessus ne sont pas des résultats expérimentaux mais des exemples de ce que nous voudrions qu'un système de PLI génère. De plus, la syntaxe utilisée n'est pas tout à fait celle de LDL mais une approximation afin de faciliter la compréhension des hypothèses.

2.2 Intérêts du modèle.

Le premier intérêt d'une telle modélisation des croyances de l'apprenant est que TAG est une base logique cohérente avec les productions de l'élève, qui permet au système de manipuler une représentation des croyances de l'apprenant. De plus, TAG utilise le même langage de description de TIG, ce qui lui donne un vocabulaire précis et limité (les prédicats de LDL) et permet à l'enseignant (et à l'élève bien sur) d'en obtenir une interprétation relativement aisée. Cette interprétation synthétique des croyances de l'apprenant est une condition importante dans notre cas et dans les EIAO en général.
Une autre propriété de TAG est qu'il constitue une base de négociation entre l'apprenant, le professeur et le système. Cette négociation peut prendre la forme d'une remise en question d'éléments du contrat didactique, d'une proposition par le professeur ou par le système de contre-exemples corrects géométriquement mais incorrects selon les conceptions de l'apprenant.
TAG permet enfin de proposer une modification semi-automatique de la théorie initiale TIG : dans le cas où TAG est correct géométriquement, il peut s'avérer approprié de l'introduire dans le théorie du domaine.

3 Mise en oeuvre du modèle de l'apprenant.

L'idée générale de la modélisation automatique des croyances de l'apprenant ayant été posée, il nous reste maintenant à définir de façon la plus complète possible l'utilisation des éléments de TALC par un système de PLI. La Programmation Logique Inductive nécessite un certain nombre d'éléments que nous allons identifier dans TALC. Puis nous décrivons un algorithme général intégrant ces données et construisant TAG.
Comme nous l'avons signalé dans le chapitre précédent, nous nous sommes exclusivement appuyé sur les exemples de la symétrie orthogonale (voir **Annexe B**) pour mettre en place notre modélisation. Cela nous permet de disposer d'un cadre relativement simple mais capable néanmoins de mettre en évidence les problèmes posés.

3.1 Besoins de TALC vis-à-vis de la PLI.

Dans l'objectif de générer un modèle logique cohérent avec les productions de l'élève par la PLI, nous avons besoin d'identifier dans TALC les éléments suivants : le langage de description, la théorie du domaine, les concepts cibles de l'induction et les observations de ces concepts.

3.1.1 Langages de description.

Le premier point à définir est le langage de description des connaissances.
Dans la présentation de TALC, nous avons vu que ce système dispose de trois langages de description différents : un langage de description des constructions (SDL), un langage de description des spécifications (CDL) et un langage logique (LDL).
C'est sur ce dernier langage que nous allons porter notre attention pour modéliser les croyances de l'apprenant. A ce choix, plusieurs raisons:

- - LDL est un langage logique. Une formule LDL bien formée représentant une construction géométrique est composée d'une conjonction de prédicats de type ou de propriété ou de la négation de prédicats de propriétés, tous clos et sans symboles de fonctions. Cette caractéristique s'avère primordiale pour une application à la PLI (voir la présentation des systèmes de PLI, chapitre précédent).
- - LDL servant de langage pour la vérification de la correction des énoncés du professeur et des constructions de l'apprenant, nous disposons directement de leurs traductions sous une forme logique exploitable par les systèmes de PLI (modulo les modifications syntaxiques exigées par ceux-ci).
- - la Théorie Instrumentale de la Géométrie (TIG) est constituée d'axiomes construits sur les prédicats de LDL.

3.1.2 Théorie du domaine.

Définir la théorie du domaine est peut être la partie du modèle la plus aisée à réaliser. En effet, TALC possède déjà tous les éléments nécessaires à sa mise en oeuvre (voir présentation du cadre de travail) : l'existence d'une Théorie Instrumentale de la Géométrie (TIG) qui représente, sous forme d'axiomes, les connaissances supposées maîtrisées par l'apprenant et requises pour résoudre le problème ainsi qu'un langage logique de description des objets géométriques (LDL). Nous proposons donc d'inclure dans la théorie du domaine ces deux données.
Aucune modification de ces éléments (autre qu'éventuellement syntaxique lors de la mise en oeuvre) ne semble être nécessaire dans un premier temps. Toutefois, certains systèmes de PLI comme FOIL ou FOCL utilise le typage des arguments des prédicats de la théorie du domaine pour restreindre l'espace de recherche et nous pouvons alors nous demander si, dans ce cas, ces informations ne sont pas redondantes avec les prédicats de typage de LDL (*point(X), cercle(C), ...*).
En effet, sous un des systèmes précédent, la formule LDL *point(a) distance(d) cercle(c1,a,d)* peut être réduite à *cercle(c1,a,d)* si nous avons au préalable défini

- - les types CERCLE, POINT et DISTANCE
- - le prédicat typé cercle(C : CERCLE, P : POINT, D : DISTANCE)

Cependant, le typage des arguments des prédicats reste transparent pour l'utilisateur, notamment dans la définition de l'hypothèse. A des fins de lisibilité, nous avons donc décidé de garder les prédicats de typage de LDL, même en utilisant le typage des arguments.

Un autre problème à régler concerne TIG. Cette théorie instrumentale est paramétrable par l'enseignant de façon à cadrer avec le problème à traiter. Lorsque nous parlons de TIG, il s'agit bel et bien de la partie fixée par l'enseignant et non de la totalité d'une théorie de la Géométrie. Cette distinction est importante car elle impose que la même théorie du domaine soit utilisée au cours d'une même session d'apprentissage, de façon à disposer d'un modèle cohérent des croyances de l'apprenant.

Par exemple, une théorie TIG suffisante pour un diagnostic de la symétrie orthogonale peut contenir les axiomes suivants:

"Un point appartient à un cercle de centre o et de rayon r si il est à une distance r de o"

```
appcc(p, c) -> cercle(c, o, r) distancep(r, o, p);
```

"La propriété distancep est symétrique"

```
distancep(r, o, p) -> distance(r) point(p) point(o)
distancep(r, p, o);
```

"Un point est à une distance r d'un point o s'il appartient au cercle de rayon r et de centre o"

```
distancep(r, o, p) -> cercle(c, o, r) distance(r) point(p)
appcc(p, c) ;
```

La définition complète de TIG se trouve dans l'**Annexe B**.

3.1.3 Concepts cibles.

Le deuxième élément à identifier concerne la désignation du ou des concepts à induire dans le cadre de la génération d'un modèle des croyances de l'apprenant par PLI.
Dans le premier chapitre, nous avons vu que le diagnostic de correction d'une construction dans TALC fournit trois types de réponse :

- - la figure est correcte
- - la figure ne possède pas tous les objets ou propriétés exigés par la spécification

- - la figure est particulière

Les corpus d'exemples sur les conceptions de la symétrie orthogonale nous permettent de nous placer uniquement dans le cas où le diagnostic fait apparaître l'absence de propriétés : nous excluons donc de cette modélisation les autres cas.
Telle que nous l'avons défini, cette génération du modèle consiste donc à redéfinir les propriétés géométriques erronées que l'apprenant utilise afin qu'elles soient conformes à ses constructions. Lorsque TALC diagnostique une erreur de construction, un certain nombre de propriétés requises par l'énoncé ne sont pas vérifiés. Ce sont ces propriétés qui vont servir de concepts cibles à l'induction d'une théorie TAG. Nous pouvons aussi utiliser les capacités de révision de théories de la PLI pour se focaliser sur des définitions incomplètes des concepts cibles, éventuellement obtenu lors d'une précédente génération de TAG.

Dans le cas des deux exemples précédents (figure 1.1), l'objectif de l'exercice proposé à l'apprenant est de vérifier sa compréhension du concept de symétrie orthogonale. Nous cherchons donc à obtenir une définition de la propriété géométrique *symétrique(S1,S2)* appliquée aux segments qui soit cohérente vis-à-vis des conceptions de l'apprenant.
Or il apparaît que ce prédicat n'existe pas dans la théorie du domaine TIG associée à ce problème et il doit donc être donné explicitement au système de PLI comme étant le concept à induire.
Ce problème permet de mettre en évidence un manque dans la conception de TALC : la possibilité qu'a l'enseignant de pouvoir définir ses propres *macro-définitions* (terme donné par analogie à la définition de macro-constructions dans Cabri-Géomètre, voir chapitre I) de manière à étendre le vocabulaire de la théorie géométrique.

3.1.4 Corpus d'observations.

Le corpus d'observation englobe deux ensembles: l'instanciation éventuelle de la théorie du domaine par des faits et les exemples et/ou contre-exemples associés aux concepts à apprendre.
L'instanciation de la théorie du domaine (atomes de typage, atomes de propriétés et éventuellement axiomes géométriques) va se faire simplement en utilisant les faits résultants des traductions LDL des constructions de l'apprenant.
En ce qui concerne les observations des concepts, l'une des hypothèses de travail (voir paragraphe 1) est que toutes les constructions de l'élève sont cohérentes vis-à-vis de ces conceptions géométriques. Cela signifie que nous ne disposons pas de contre-exemples des concepts à induire. Toutes les observations sont donc considérées comme exemples positifs. Or, un système de PLI nécessite la présence de contre-exemples afin de diriger l'induction de l'hypothèse. Ceux-ci ne pouvant pas être obtenu à partir des constructions de l'apprenant, ils doivent être inférés par le système. Notre idée est donc de se placer dans le cadre d'un apprentissage inductif incrémental dans lequel une interaction avec l'apprenant permet au système d'obtenir ces contre-exemples du concept cible.

Dans l'exemple précédent de la procédure "parallélisme" (figure 1.1), la traduction LDL de la construction de l'élève permet l'instanciation de la théorie du domaine suivante :

- atomes de typages :

```
cercle(VarC1,VarP3,var3)
distance(var3)
droite(Ax) droite(VarD2) droite(VarD3) droite(VarD4)
point(A') point(A) point(B') point(B) point(VarP3)
segment(var2,A,B,var1) segment(var5,A',B',var4)
```

- atomes de propriétés :

```
appcc(A',VarC1) appcc(A,VarC1)
appdr(A',VarD2)        appdr(A',VarD4)        appdr(A,VarD2)
appdr(B',VarD3) appdr(B',VarD4)
                                  appdr(B,VarD3)  appdr(VarP3,Ax)
appdr(VarP3,VarD2)
par(VarD4,var1)
perp(VarD2,Ax) perp(VarD3,Ax)
```

L'exemple positif associé au concept à induire est donc *symétrique([AB],[A'B'])*.

Un dernier problème à résoudre concerne l'utilisation les identificateurs. En effet, du point de vue de la PLI, les segments [AB], par exemple, définis dans deux constructions différentes ne correspondent pas à la même

constante logique. Nous devons donc faire attention à ce que les noms de tous les identificateurs identiques à travers plusieurs constructions soient uniques.

3.2 Proposition d'un algorithme.

Après avoir identifié les besoins de TALC relativement à la PLI, nous allons maintenant proposer une base algorithmique de notre processus de génération de TAG.

Dans le paragraphe précédent, nous avons signalé que, par l'hypothèse didactique de cohérence des raisonnements de l'apprenant, nous ne disposons pas de contre-exemples de ses conceptions.
Nous proposons donc de nous placer dans un environnement où ceux-ci sont générés par le système, au fur et à mesure de ses besoins pour généraliser ses hypothèses, et proposés pour validation à l'apprenant. La partie PLI de ce processus doit donc être incrémentale et interactive.

L'algorithme (figure 2.1) est lui même incrémental, de façon à pouvoir prendre en compte, dans une première couche, toutes les productions de l'apprenant relatives à une spécification de l'enseignant puis, dans une couche supérieure, toutes les spécifications données. Comme nous l'avons signalé dans le paragraphe 2, cette génération de TAG s'effectue toujours dans le cadre d'une théorie TIG donnée et immuable, de façon à conserver une cohérence des résultats.

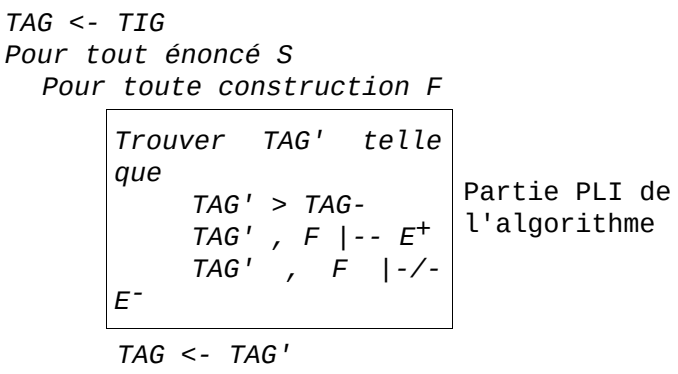


Figure 18 : Principe de génération automatique d'un modèle de l'apprenant

Procédant par révision de théorie à partir de la théorie instrumentale initiale TIG, l'algorithme généralise la théorie actuelle de façon à en obtenir une nouvelle qui permette d'expliquer les constructions de l'apprenant. Par TAG' > TAG, nous assurons la convergence de notre modèle tout au long de son évolution. A chaque étape de sa construction, TAG doit permettre d'expliquer plus d'erreurs de l'apprenant que les versions précédentes.
La partie PLI de l'algorithme se situe au niveau des vérifications de complétude (*TAG', F* |-- E^+) et de correction (*TAG', F* |-/- E^-) de TAG'. C'est durant cette phase que les contre-exemples peuvent être proposés à la validation par l'apprenant.

3.3 Choix d'un système de PLI.

Dans le chapitre III présentant l'évaluation des systèmes de PLI, nous avons résumé leurs caractéristiques principales dans un tableau de nous reproduisons ici (figure 2.2) en y rajoutant une colonne pour les besoins de la génération de TAG.

Pour le langage de description, les impératifs de TALC sont évidents: LDL n'admet ni symboles de fonctions, ni littéraux négatifs. De plus, la présence de TIG impose l'utilisation d'une théorie du domaine intentionnelle. Le typage des arguments des prédicats existe de manière indirecte sous LDL (axiomes de typage) et peut être considéré comme tel pour la restriction de l'espace de recherche. Par contre, le mode d'instanciation des prédicats ne semble pas être indispensable.

	FOCL	**FOIL**	**CLINT**	**GOLEM**	**TAG**
Caractéristiques comportementales					
Apprentissage interactif			**X**		**X**
Apprentissage incrémental			**X**		**X**
Apprentissage multi-conceptuel			**X**		**?**

Révision de théories	**X**		**X**		**X**
Invention de prédicats	**X**		**X**		**?**
Restrictions du langage de représentation					
Pas de littéraux négatifs	**P**	**P**		**X**	**X**
Pas de symboles de fonctions	**X**	**X**	**X**		**X**
Théorie du domaine intentionnelle	**X**		**X**		**X**
Restrictions de l'espace de recherche					
Modes d'instanciation des prédicats	**X**	**X**		**X**	
Typage des arguments	**X**	**X**			**?**

Figure 19 : Comparaison des caractéristiques des systèmes évalués et des besoins de TALC.

Le choix se fait donc surtout en ce qui concerne les caractéristiques comportementales des systèmes. Notre modélisation de TAG nécessite que le système puisse effectuer une révision de la théorie initiale. De plus, le mode incrémental de la génération de TAG et l'absence de contre-exemples des propriétés cibles semble nécessiter aussi un système incrémental et interactif.
Les questions de l'invention de prédicats et de l'apprentissage mono ou multi-conceptuel ne sont pas, dans un premier temps, handicapantes pour le choix du système mais peuvent fournir un plus quant à la qualité du modèle.

En conclusion de cette identification des besoins de la génération de TAG, il nous reste à proposer un système qui puisse servir de moteur à la génération de TAG.
Il semble bien que nous pouvons exclure des possibilités les systèmes FOIL et GOLEM, surtout du fait de leur théorie du domaine exclusivement extensionnelle. L'approche incrémentale et interactive de l'induction de TAG nécessite un tel système, ce qui réduit le choix au seul système CLINT. Malheureusement, l'évaluation de ce système a montré de gros problèmes liés à ses capacités machine et à la définition des méta-règles utilisées dans la restriction de l'espace de recherche. Nous pensons qu'une expérimentation basée sur CLINT n'a que peu de chance de s'avérer efficace, voir même utilisable.

4 Premiers résultats expérimentaux

Nous avons donc utilisé les corpus d'exemples des conceptions de la symétrie orthogonale (voir paragraphe 1.1) pour expérimenter la modélisation des croyances de l'apprenant. Les premiers essais se sont basés sur la conception "parallélisme" car le résultat attendu est relativement simple.
Les expérimentations se sont effectuées de façon indépendante des constructions sous TALC : les corpus ont été saisis à la mains et injectés dans les systèmes après éventuellement quelques corrections syntaxiques (renommage des variables, adaptation de la syntaxe des règles, ...).

La définition complète de cet exemple (théorie du domaine, spécification de l'exercice, traduction de la construction) se trouve dans l'**Annexe B** (paragraphe 4.1).
Le concept de symétrie orthogonale n'étant pas fixé dans LDL, nous avons donc défini le prédicat *symétrique(X,Y)* comme concept cible du processus d'induction. Un problème est tout de suite apparu : le choix du type des arguments *X* et *Y* . Le prédicat de typage LDL d'un segment se définit par :

```
segment(s1, p1, p2, l)   le segment s1 est défini par ses
extrémités p1 et p2 et par
                    sa droite support l
```

La théorie du domaine associée à ce problème comportant un grand nombre d'identificateurs de droites, nous avons choisi un prédicat portant sur les droites supports :

```
symétrique(l1, l2) les droites l1 et l2,   supports des
segments, sont symétriques
```

L'expérimentation sous CLINT a confirmé ses limites déjà mises en évidence lors de l'évaluation de ce système (voir chapitre III) : quelles que soient la taille de la théorie du domaine et la définition du concept cible, le

processus entraîne un temps d'induction trop important ou même un plantage du système. Dans les meilleures conditions de paramétrisation du système, le processus fournit la définition de la symétrie orthogonale suivante :

```
symétrique(l1,l2) :- segment (s1,p1,p2,l1)
segment(s2,p3,p4,l2).
```

Nous avons alors utilisé le système FOCL qui présente l'inconvénient de ne pas être incrémental : les contre-exemples du concept cible doivent être donnés. Cette limitation a été contourné en ne se basant que sur un seul exemple de la conception "parallélisme" et en autorisant le système à générer automatiquement les contre-exemples à partir de cet ensemble d'exemples positifs supposé complet. Les résultats varient selon la paramétrisation du système. La définition obtenue la plus conforme aux résultats attendus est la suivante :

```
symétrique(l1,l2) :- par(l3,l1) segment (s,p1,p2,l2)
appdr(p1,l3).
```

ce qui signifie que les droites l1 et l2 (support des deux segments) sont symétriques si l2 est le support du segment s=[p1 p2], si p1 appartient à la droite l3 et que cette droite l3 est parallèle à l1.
Cette définition n'est pas complètement correcte : il y a bien parallélisme de l1 et l2 uniquement si les droites l3 et l2 sont identiques.
Après analyse de l'exercice, il apparaît que cette particularité est dûe à la faiblesse de la théorie TIG choisie : lorsque l'on construit un segment passant par deux points déjà situés sur une droite, alors la droite support du segment est cette droite. Or la théorie TIG de cette exercice, qui permet pourtant le diagnostic de symétriques orthogonaux, n'est pas suffisante pour permettre à TALC d'opérer cette identification. Cette hypothèse est vérifiée en remplaçant l'identificateur de la droite support du segment [A' B'] dans la théorie du domaine par l'identificateur approprié : nous obtenons alors la définition attendue *symétrique(l1,l2) :- par(l2,l1)*.

Cette difficulté que nous avons rencontrée en nous mettant à la place de l'enseignant est un parfaite illustration des difficultés que peut avoir celui-ci à spécifier une théorie du domaine TIG suffisante pour un exercice donné (en particulier pour modifier TIG).
Cette première expérimentation n'a pu être conduite plus en avant, faute de robustesse des systèmes et confirme bien le fait que les systèmes de PLI présentés dans ce document ne permettent pas une bonne application à la génération d'un modèle logique des croyances de l'apprenant.

5 Limites du modèle.

Les problèmes posés par la mise en oeuvre de TAG sont multiples et, malheureusement, pas tous solubles dans l'état actuel de nos travaux.

L'inadéquation partielle des systèmes de PLI. Le paragraphe précédent à montré que les systèmes de PLI que nous avons évalués se prêtent assez mal à la génération de TAG telle que nous l'avons spécifiée: pas d'interactivité dans FOCL, manque d'efficacité de CLINT sur des domaines importants, ... Cela n'est pas un gros problème en soi car nous n'affirmons pas avoir sélectionné les meilleurs ou même tous les systèmes existants. De plus, il nous semble que la création d'un algorithme de PLI particulièrement adapté à nos besoins et à nos objectifs soit préférable à l'utilisation d'une plate-forme générale. L'existence de nombreuses implémentations des outils de la PLI (subsomption, rlgg, inversion de la résolution, ... voir chapitre II) permet quand même de ne pas avoir à redéfinir l'existant.

Les limites de l'automatisation. Il apparaît que de nombreuses opérations du processus de modélisation des croyances de l'apprenant ne peuvent être complètement prises en charge par le système : elles nécessitent l'intervention d'un utilisateur, que se soit l'apprenant ou l'enseignant.
Notre idée étant d'utiliser un processus d'induction incrémental et interactif, l'intervention de l'apprenant est requise notamment lors de la validation ou de l'infirmation des faits inférés par le système de PLI. L'intervention doit être parfaite afin que le processus de PLI puisse correctement généraliser ses hypothèses. Cette perfection est néanmoins difficile à obtenir.
Nous avons signalé dans le premier paragraphe que TAG constitue une base logique permettant une renégociation d'éléments du contrat didactique (voir présentation de TALC) ou une modification de la théorie TIG. Ces opérations ne peuvent évidemment pas être effectuées sans contrôle de l'enseignant.

Les limites du corpus d'exemples. Notre mise en oeuvre de la modélisation s'est basée essentiellement sur l'analyse de corpus concernant des conceptions correctes et incorrectes de la symétrie orthogonale. Ce corpus, suffisant pour étudier l'utilisation de la PLI pour cette approche, ne permet pas de traiter tous les cas de figure du diagnostic de TALC : nous n'avons pas abordé le cas des figures particulières, ni celui où il manque des objets dans la construction de l'apprenant. Il est nécessaire de valider notre modélisation sur des exemples plus complexes pour mettre en évidence ces cas d'erreur de diagnostic.

Nous donnons des éléments de réponses à ces limitations de la modélisation dans la conclusion et les perspectives de ce document.

6 Conclusions

La génération d'un modèle des croyances de l'apprenant par rapport aux réponses à des exercices ou à des problèmes nécessite la description et l'interprétation du raisonnement hypothétique effectué par celui-ci. L'utilisation de la Programmation Logique Inductive pour réaliser cette génération présente notamment deux avantages.
D'une part l'utilisation de la logique du premier ordre permet l'inférence de descriptions complexes. Le programme logique obtenu est exécutable et autorise la simulation des raisonnements étudiés.
D'autre part, l'apprentissage s'effectue par un mécanisme d'induction indépendant du domaine étudié. Seules une théorie du domaine logique et des informations sur les concepts cibles à inférer sont nécessaires lors d'une application à un problème donné.

Appliquée au diagnostic des erreurs de conceptions dans le cadre de TALC, cette approche prend tout son sens. La représentation des connaissances sous forme de clauses de Horn (le langage LDL de TALC), la présence d'une théorie du domaine logique (TIG, Théorie Instrumentale de la Géométrie), la génération du modèle sous forme de révision de cette dernière font de la PLI un outil particulièrement adéquat.
Cependant, cette approche comporte des inconvénients, ses principaux étant l'absence de contre-exemples et la présentation incrémentale des exemples qui nécessitent une intervention efficace de l'utilisateur.

Conclusion et perspectives

Les travaux présentés dans ce document sont à la croisée de deux domaines de recherche : la Programmation Logique Inductive et l'EIAO. Il s'agissait d'étudier comment la PLI pouvait être utilisée pour générer un modèle des croyances d'un élève dans le système TALC.
La démarche suivie a dégagé les étapes suivantes :

- - étude de l'existant à la jonction de ces deux domaines
- - étude des principes théoriques de la Programmation Logique Inductive
- - évaluation de quelques systèmes de PLI
- - étude de leur réutilisation pour la génération d'un modèle des croyances de l'apprenant.

Nous présentons ici les résultats significatifs de chacune de ces études et les perspectives qu'elles offrent.

1 Résultats.

Notre travail bibliographique a montré le trés faible nombre de réalisations visant à utiliser l'apprentissage automatique pour la modélisation de l'apprenant.

Nous avons extrait des principes théoriques de la PLI et de l'existant dans ce domaine une classification des systèmes de PLI selon des critères comportementaux. Le principe directeur était d'utiliser un tel système comme une boîte noire générant le modèle de l'apprenant. Cette classification nous a permis de sélectionner et évaluer un ensemble représentatif de systèmes de PLI candidats. De ce fait, parmi les systèmes cités dans la littérature, seuls ceux effectivement disponibles et opérationnels ont été retenus.

Notre contribution à l'amélioration de TALC a consisté à proposer la génération d'une *Théorie de l'Apprenant en Géométrie* (TAG). A partir de l'hypothèse didactique de cohérences des raisonnements de l'apprenant, cette théorie rend compte des conceptions géométriques de l'élève qu'il manifeste par l'ensemble de ses constructions. TAG est un modèle logique généré par induction constructive.
Cette approche s'est averée particulièrement adéquate. D'une part il existe une grande similitude des principes de la PLI et de la modélisation des croyances de l'apprenant; dans les deux cas, il s'agit d'inférer des hypothèses cohérentes avec des observations de concepts dans le cadre d'une théorie du domaine, sans se préoccuper de la correction vis-à-vis du domaine de telles hypothèses. D'autre part, la PLI a pour grand avantage qu'elle utilise le même formalisme de représentation des connaissances pour les données et les résultats et que c'est justement celui utilisé dans TALC.
Les expérimentations sur des cas réels ont montré qu'il est nécessaire de disposer de possibilités de conception de macro-définitions de concepts géométriques dans TALC, de façon à étendre le langage de description des énoncés par le professeur.

Nous avons aussi montré que les systèmes évalués dans ce document s'avèrent peu adaptés pour une réutilisation en boîte noire : sauf à engager un dialogue avec l'apprenant pour les obtenir, l'absence de contre-exemples des concepts cibles de l'apprentissage est la raison majeure de cette limite.

2 Perspectives.

Du fait de la mauvaise adéquation des quatre systèmes de PLI présentés dans ce document, il apparaît nécessaire, dans un premier temps, de rechercher et d'utiliser un système palliant aux limitations des précédents. A plus long terme, plutôt que de réutiliser tel quel un système de PLI, nous envisageons la conception d'un module de PLI adapté aux contextes structurels et langagiers de TALC et qui puisse s'intégrer dans la structure de ce système. Cette conception d'un module d'inférence inductive peut être facilitée par l'existence de librairies PROLOG implémentant la plupart des mécanismes de la PLI.

La modélisation des croyances de l'apprenant par la PLI présentée dans ce document s'est appuyée essentiellement sur un seul ensemble d'exemples relativement simples : la symétrie orthogonale. Il est souhaitable, une fois le choix du système fixé, de valider ce processus sur d'autres exemple plus complexes.

Comme nous l'avons déjà signalé, l'utilisation des contre-exemples des concepts cibles dans un système de PLI incrémental est importante pour guider la recherche des hypothèses. De façon à conserver une interaction compatible avec les exigences d'un EIAO, nous pensons que la mise en place d'un formalisme de présentation de ces contre-exemples sous la forme d'une figure géométrique est indispensable.

De la même manière que l'utilisateur de Cabri-Géomètre peut définir ses propres macro-constructions, il nous semble intéressant de permettre au professeur de définir et d'introduire lui-même dans TALC les macro-définitions dont nous avons déjà parlées. Pour cela, deux méthodes sont possibles : l'introduction de ces nouvelles propriétés comme axiomes de la Théorie Instrumentale de la Géométrie TIG ou leur définition par un processus de remplacement syntaxique (comme par exemple les *#define* du langage C).

D'une façon plus générale se pose la question de l'utilisation du modèle obtenu.
Il faudrait, par exemple, donner à l'enseignant un moyen de décrire les réactions du système en fonction des causes d'erreur obtenues par l'analyse de TAG : proposer à l'apprenant telle ou telle activité si telles ou telles conditions sont déductible du modèle.
La définition de TAG elle même peut être prise en considération en disposant d'un moyen de vérifier sa correction d'un point de vue géométrique où deux cas de figure peuvent se présenter. Dans le cas où TAG s'avère être correcte géométriquement (par exemple lorsque l'apprenant utilise des axiomes ou des théorèmes géométriques non spécifiés par l'enseignant), son adjonction à la théorie instrumentale TIG devient peut être envisageable, selon les besoins et les conditions fixés par l'enseignant. Dans le cas contraire, il semble essentiel de pouvoir présenter à l'apprenant un contre-exemple, c'est-à-dire une figure qui soit correcte géometriquement mais non selon ses propres conceptions, de manière à le convaincre de remettre en cause ses conceptions.

Bibliographie

[And 85] Anderson J.R. , Boyle C.F. , Yost G. , *The Geometry Tutor*, proc. of 9th Internation Joint Conference on Artificial Intelligence, Morgan Kaufmann (1985)

[Bai 92] Bain M. , *Experiments in Non-monotonic First-Order Induction,* in S. Muggleton (Ed.), Inductive Logic Programming, pp 423-435 (1992)

[Bar 94] Baron M. , *EIAO, quelques repères*, Terminal 64, L'Harmattan (1994)

[BV 95] Baron M. , Vivet M. , *Modélisation de connaissances pour des environnements interactifs d'apprentissage avec ordinateur*, actes des 5èmes journées nationales du PRC-GDR IA, Teknea, pp 239-262

[Bel 92] Bellemain F. , *Conception, réalisation et expérimentation d'un logiciel d'aide à l'enseignement de la géométrie*, Thèse, Université Joseph Fourier - Grenoble (1992)

[Bur 82] Burton R.R. , Brown J.S. , *An investigation of computer coaching for informal learning activities* , Intelligent Tutoring Systems, Sleeman & Brown (Eds), Academic Press, pp 79-98 (1982)

[Car 70] Carbonell J.R. , *AI in ICAI : an artificial intelligence approach to computer-assisted instructions*, IEEE trans. Man Machine systems , 11 (4), pp 190-202 (1970)

[Cla 82] Clancey W.J. , *Tutoring rules for guiding a case method dialogue*, Intelligent Tutoring Systems, Sleeman & Brown (Eds), Academic Press, pp 201-225 (1982)

[DB 92] De Raedt L. , Bruynooghe M. , *An Overview of the Interactive Concept-Learner and Theory Revisor CLINT*, in S. Muggleton (Ed.), Inductive Logic Programming, pp 163-191 (1992)

[DB 93] De Raedt L. , Bruynooghe M. , *A Theory of Clausal Discovery*, Proc. of 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann , pp 1058-1063 (1993)

[Der 91] De Raedt L. , *Interactive Theory Revision: an Inductive Logic Programming Approach*, Academic Press (1991)

[DLD 93] De Raedt L. , Lavrac N. , Dzeroski S. , *Multiple Predicat Learning*, Proc. of 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp 1037-1043 (1993)

[Ded 86] Dede C. , *A review and synthesis of recent research in intelligent computer-assisted instruction*, International Journal of Man-Machine Studies 24, pp 329-353 (1986)

[DM 86] DeJong G. , Mooney R. , *Explanation-Based Learning: An AlternativeView*, Machine Learning, 1, pp 145-176 (1986)

[Des 94] Desmoulins C. , *Etude et réalisation d'un système tuteur pour la construction de figures géométriques*, Thèse, Université Joseph Fourier - Grenoble I (1994)

[Des 95] Desmoulins C. , *La détéction de solutions particulières dans TALC : une approche logique basée sur des extensions de l'énoncé du problème*, in EIAO, 4èmes journées EAIO de Cachan, pp 161-172 (1995)

[Ell 89] Ellman T. , *Explanation-Based Learning: A Survey of Programs and Perspectives*, ACM Computing Surveys, Vol. 21, 2, pp 163-221 (1989)

[Fla 93] Flach P.A. , *Predicate Invention in Inductive Data Engineering*, Proc. of European Conference on Machine Learning '93, LNAI 667 (1993)

[Gre 88] Grenier D. , *Construction et étude du fonctionnement d'un processus d'enseignement sur la symétrie orthogonale en sixième*, Thèse, Université Joseph Fourier - LSD2-IMAG (1988)

[Joh 83] Johnson W.L. , *Knowledge-Based program understanding*, Technical Report YaleU/CSD/RD = 285, University of Yale, Dept. of Computer Science (1983)

[KMKT 86] Kawai K. , Mizoguchi R. , Kakusho O. , Toyoda J. , *A framework for ICAI systems based on Inductive Inference and Logic Programming*, Proc. of 3rd Logic Programming Conference, LNCS 225, pp 186-202 (1986)

[KW 92] Kietz J.U. , Wrobel S. , *Controlling the Complexity of Learning in Logic Through Syntactic and Task-Oriented Models*, in S. Muggleton (Ed.), Inductive Logic Programming, Academic Press (1992)

[Kod 86] Kodratoff Y. , *Leçons d'Apprentissage Symbolique Automatique*, Cepadues-Editions (1986)

[LD 94] Lavrac N. , Dzeroski S. , *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood (1994)

[MKCM 93] Michalski R.S. , Kodratoff Y. , Carbonell J.G. , Mitchell T. (Eds.) , *Apprentissage Symbolique : une approche de l'Intelligence Artificielle*, Cepadues-Editions (1993)

[MKK 86] Mitchell T. , Keller R. , Kedar-Cabelli S. , *Explanation-Based Generalization: A Unifying View*, Machine Learning, 1, pp 47-80 (1986)

[Mug 92] Muggleton S. (Ed.) , *Inductive Logic Programming*, Academic Press, London

[MB 88] Muggleton S. , Buntine W. (1988), *Machine Invention of First-order Predicates by Inverting Resolution*, Proc. of 5th International conference on Machine Learning, pp 339-352 (1992)

[MD 94] Muggleton S. , De Raedt L. , *Inductive Logic Programming: Theory and Methods*, Journal of Logic Programming 19-20, pp 629-679 (1994)

[MF 92] Muggleton S. , Feng C. , *Efficient Induction of Logic Programs*, in S. Muggleton (Ed.), Inductive Logic Programming, pp 281-298 (1992)

[Mug 94] Muggleton S. , *Predicate Invention and Utility*, Journal of Experimental and Theorical Artificial Intelligence, 6 (1), pp 127-130 (1994)

[NV 88] Nicaud J.F. , Vivet M. , *Les tuteurs intelligents : réalisation et tendances de recherche*, Technique et Science Informatiques, 7 (1), pp 21-45 (1988)

[OL 88] Ohlsson S. , Langley P. , *Psychological evaluation of path hypotheses in cognitive diagnosis*, Learning Issues for Intelligent Tutoring systems, Mandl & Lesgold (Eds), pp 42-62 (1988)

[Pap 80] Papert S. , *Jaillissement de l'esprit, ordinateur et apprentissage*, Flammarion (1980)

[PBS 92] Pazzani M. , Brunk C. , Silverstein G. , *An Information-Based Approach to Combining Empirical and Explanation-Based Learning*, in S. Muggleton (Ed.), Inductive Logic Programming, pp 373-394 (1992)

[PK 92] Pazzani M. , Kibbler D. , *The Utility of Knowledge in Inductive Learning*, Machine Learning, 9, pp 57-94 (1992)

[Per 95] Person M. , *Génération de diagnostic dans le tuteur de constructions géométriques TALC*, stage de DEA, CRIN - Université Henry Poincarré (1995)

[Plo 70] Plotkin G.D. , *A Note on Inductive Generalization*, Machine Intelligence 5, Meltzer et Michie (Ed.), pp 153-163 (1970)

[Qua 94] Quast M. , *Programmation Logique Inductive*, Examen probatoire du CNAM - Grenoble (CUEFA) (1994)

[QC 93] Quinlan J.R. , Cameron-Jones R.M. , *FOIL: a Midterm Report*, Proc. of European Conference on Machine Learning '93, LNAI 667, pp 3-20 (1993)

[Qui 91] Quinlan J.R. , *Determinate Literals in Inductive Logic Programming*, Proc. of 12th International Joint Conference on Artificial Intelligence, pp 746-750 (1991)

[Qui 90] Quinlan J.R. , *Learning Logical Definitions from Relations*, Machine Learning, 5, pp 239-266 (1990)

[Sio 94] Siou E. , *Programmation Logique Inductive et modélisation de l'apprenant; application à l'analyse des erreurs de raisonnement chez l'aphasique*, Thèse, Université de Rennes I (1994)

[STW 93] Stahl I. , Tausend B. , Wirth R. , *Two methods for improving inductive logic programming systems*, Proc. of European Conference on Machine Learning '93, LNAI 667 (1993)

[Tah 93] Tahri S. , *Modélisation de l'interaction didactique : un tuteur hybride sur Cabri-Géomètre pour l'analyse de décisions didactiques*, Thèse, Université Joseph Fourier - Grenoble I (1993)

Annexe A
Evaluation quantitative des systèmes de PLI

1 A propos de l'évaluation des systèmes de PLI

Dans cetet annexe, nous présentons succinctement les systèmes de PLI qui n'ont pas été évalués et les raisons de cet abandon ainsi qu'un résumé qualitatif des principaux problèmes étudiés sur les quatre systèmes retenus.

2 Systèmes non retenus.

Comme nous l'avons signalé dans le **chapitre III**, seuls les quatre systèmes FOCL [PK 92], FOIL [Qui 90], CLINT [Der 91] et GOLEM [MF 92] ont été choisis pour l'évaluation.
Les systèmes non retenus et les raisons de cet abandon sont les suivants :

- INDEX [Fla 93]. Sa syntaxe restée incompréhensible et l'absence d'une documentation technique et théorique suffisante m'ont amené à rejeter rapidement ce système.
- MOBAL [KW 92] pose un problème différent. Il s'agit plus d'un système d'acquisition de connaissances ou de découverte de connaissances (KDD) utilisant l'induction que d'un système de PLI: il est capable de redécouvrir des relations entre les éléments d'une base de données logique (clauses de Horn). Son intérêt réside dans le fait qu'il permet, pour l'acquisition de connaissances, un interfaçage avec des systèmes de PLI classiques (exploitation en commun des théories et des observations). Le problème est qu'il n'admet des liens qu'avec GOLEM et FOIL pour le moment, systèmes rapidement rejetés pour une application à la modélisation des connaissances de l'apprenant. De plus, son paramétrage est vraiment trop lourd.
- MILES de Stahl et Tausend (pas de reference directe au système mais voir [STW 93]) est un système de PLI vraiment intéressant dans un cadre théorique mais inutile dans la pratique : il s'agit en fait d'une plate-forme d'expérimentation paramétrable des différentes techniques, outils et principes de la PLI. Il est impossible de l'utiliser à des fins pratiques.
- CLAUDIEN (pas de références) est le dernier système en date de l'équipe de De Raedt, concepteur de CLINT. Encore en développement, il est néanmoins disponible à des fins académiques sur requête. Malgré deux tentatives de contact avec les concepteurs de ce système, nous n'avons reçu aucune réponse.
- FILP est le dernier système trouvé sur site ftp. Il nécessite l'utilisation d'un interpreteur Prolog indisponible en local et l'utilisation de librairies particulières empeche de le porter rapidement sous une autre syntaxe. De plus, je n'ai pas réussi à trouver des références bibliographiques à son propos.

Tous ces systèmes ont étés récupérés par ftp sur des serveurs dédiés à l'apprentissage automatique ou la Programmation Logique Inductive. C'est pour cette raison que des systèmes plus connus comme MIS de Shapiro [Sha 81] ou plus efficaces comme mFOIL de S. Dzeroski [LD 94] (qui étend les capacités de FOIL par une théorie du domaine intentionnelle et un traitement des données imparfaites) sont introuvables. En fait, dans le cas de MIS, nous avons appris que l'équipe EIAO de Rennes avait retapé le code d'après les articles correspondants.

3 Analyse quantitative.

Sélectionnés parmi les problèmes fournis avec chacun des systèmes, nous avons particulièrement traité et étudié les problèmes présentés dans le tableau suivant.

La colonne *Observation* indique le nombre d'exemples positifs et négatifs du concept cible. Des systèmes comme FOIL (voir chapitre III) sont capables de calculer automatiquement des contre-exemples à partir de l'ensemble complet des exemples positifs, ce qui explique que certains concepts cible n'ont pas d'exemples négatifs explicites. Lorsqu'ils doivent l'être, pour GOLEM par exemple, nous les avons fabriqués à partir de perturbation des exemples positifs . Ces cas sont désignés par le symbole (*).

La colonne *BK* indique le nombre de prédicats autre que le concept cible présent dans la théorie du domaine, ainsi que le nombre total d'instances de ceux-ci (dans le cas d'une théorie du domaine extensionnelle).

	Observations (pos./neg.)	**BK** (pred./nb.)	**Description**
Ackermann	51 / 9261 (*)	1 / 20	fonction récursive d'Ackermann
Animals	10 / 32	6 / 60	caractérisation des oiseaux, poissons, ...
Choose	23 / 31	3 / 230	fonction combinatoire C(n,p)
Family	20 / 61 (*)	3 / 18	apprentissage du concept d'*ancêtre*
Poker-Pair	10 / 10	5 / 292	détermination d'une paire au Poker
QuickSort	65 / 4225 (*)	3 / 382	tri sur les listes
Reverse	11 / 30	1 / 40	inversion de listes
Xmas-Present	2 / 6	1 / 8	induction du concept '*cadeau de Noël*'
KRK-end game et *Member* ont été présentés dans le **chapitre 3**			

Les résultats qualitatifs de l'évaluation des systèmes sont résumés dans le tableau ci-dessous.

Sauf indications contraires, les systèmes FOIL, FOCL et CLINT ont été évalués sur un Macintosh LC 475, GOLEM sur stations SUN.

Pour chaque système, nous précisons:

- - *Np* qui correspond au nombre approximatif d'inductions effectué (nombre de traces d'exécution effectivement conservées). Quand cela est approprié, nous avons modifié à chaque exécution les paramètres des systèmes (acceptation ou non de la négation, utilisation ou non de nouvelles variables, ...) ou la théorie du domaine (contraintes inter-arguments, génération automatique des contre-exemples, perturbations des exemples, ...).
- - *T* qui correspond au temps d'induction. Ce temps est fortement approximatif: la plupart des systèmes ne possèdent pas de chronomètre ou ne tiennent pas compte du temps d'affichage de la trace (notamment dans le cas de l'interface graphique de FOCL). De plus, l'influence des paramètres d'induction (surtout dans le cas de FOCL et FOIL) sur le temps de calcul est très importante (du simple au triple ou plus).
- En ce qui concerne CLINT, le temps indiqué est une évaluation du processus d'induction, sans prendre en compte l'interaction avec l'utilisateur.

	FOCL		**FOIL**		**CLINT**		**GOLEM**	
	Np	**T**	**Np**	**T**	**Np**	**T**	**Np**	**T**
Ackermann	2	> 30 mn	2	11,5 mn			4	1 mn (1)
Animals	4	3 mn			2	5 mn	2	30 s
Choose	1	12 mn	4	7 mn	2	15 mn	3	2 mn
Family	3	3 mn	2	2 mn	1	8 mn		
KRK-end game	5	3 mn	7	2 mn	4	>30 mn (2)	9	1 mn
Member	9	1,5 mn	10	1 mn	2	4 mn	5	20 s
Poker-Pair	5	5 mn	8	3 mn	1	8 mn	2	1 mn
QuickSort			1	15 mn	2	15 mn (2)	3	3 mn (1)
Reverse	3	4 mn	4	2 mn	2	6 mn	4	1 mn
Xmas-Present	3	3 mn	3	1 mn				30 s

- (1) évaluation effectuée sur GOLEM avec un ensemble réduit de contre-exemples.
- (2) convergence des hypothèses de CLINT non certaine (voir présentation de CLINT, chapitre III) mais temps d'induction trop important pour continuer le processus

Figure 20 : Analyse quantitative des systèmes étudiés

Annexe B
Définition de corpus d'exemples sur la symétrie orthogonale dans TALC

1 Introduction.

Dans leurs travaux, D. GRENIER [Gre 88] et S. TAHRI [Tah 93] ont étudié les erreurs de conceptions géométriques d'un élève, sous Cabri-Géomètre, lors de la construction du symétrique orthogonal d'un segment.
En jouant sur les variables directionnelles (car la conservation des distances est toujours prises en compte chez les élèves), ils ont mis en évidence des procédures de constructions erronées chez les élèves de 6ème :

- - le rappel vertical ou horizontal : l'élève trace les images du segment sur une direction verticale ou horizontale et non orthogonale à l'axe.
- - le parallélisme : l'élève conçoit aisément que le segment cible et son image doivent être parallèles. Pour la construction des extrémités, il peut faire appel à des procédures de type report horizontal, ...
- - le prolongement : l'élève construit l'image du segment dans le prolongement de celui-ci.
- - l'orthogonalité par rapport au segment et non à l'axe.
- - la demi symétrie, dans le cas où le segment coupe l'axe.
- - les procédures perceptives: procédure de construction globale (l'image du segment ne fait intervenir aucun objet géométrique autre que le segment produit) ou semi-globale (semi-analytique, une seule extrémité image est construite; le segment image est alors construit au jugé en s'appuyant sur cette extrémité).

Ces procédures sont utilisées dans les quatre conceptions de la symétrie suivantes :

- - la conception de "parallélisme" lié à une procédure de parallélisme. Le segment et son image sont parallèles et de même longueur.
- - la conception "symétrie oblique". Les distances à l'axe des points et de leurs images sont conservées, ainsi que les directions des supports. Seule l'orthogonalité est absente.
- - la conception "symétrie centrale" par rapport à un point sur l'axe. Le segment image est soit dans le prolongement du segment objet, soit parallèle et de sens inverse.
- - la conception de symétrie orthogonale par rapport à l'axe. C'est la seule conception correcte.

Ces travaux représentent un cadre intéressant pour tester une génération automatique d'un modèle représentant les connaissances de l'apprenant dans un EIAO de géométrie.

Pour réaliser un corpus d'expérimentation du modèle des connaissances de l'apprenant, nous avons pris une construction appartenant à chacune des quatre conceptions précédentes, plus une correspondant à une conception "perceptive".
Nous les décrivons dans les paragraphes suivants, en commençant par donner une théorie TIG suffisante pour le diagnostic d'erreurs, ainsi qu'une spécification de l'énoncé donné par le professeur. Nous présentons auparavant une descriprion complète du langage LDL qui sert de langage logique à la fois pour TALC et pour la modélisation des connaissances de l'apprenant par Programmation Logique Inductive.

2 Description du langage LDL

Comme nous l'avons signalé dans le chapitre I, les prédicats du langage LDL sont divisés en deux catégories [Des 94] : les prédicats de typage qui représentent le type et les attributs d'un objet géométrique et les prédicats de propriétés qui représentent les relations géométriques reliant ces objets. Nous les décrivons ici.

2.1.1 Prédicats de typage.

Atomes	Signification
point(p)	p est un point
droite(l)	l est une droite
semi-droite(h,p,l)	h est une demi-droite d'origine le point p et de support la droite l
segment(s,p1,p2,l)	s est un segment d'extrémités les points p1 et p2 et de support la droite l
	c est un cercle de centre le point p et de rayon la distance d
cercle(c,p,d)	d est une distance
distance(d)	

2.1.2 Prédicats de propriété.

Atomes	Signification
appdr(p,l)	le point p appartient à la droite l
appdd(p,h)	le point p appartient à la demi-droite h
appseg(p,s)	le point p appartient au segment s
appcc(p,c)	le point p appartient au cercle c
par(l1,l2)	les droites l1 et l2 sont parallèles
perp(l1,l2)	les droites l1 et l2 sont perpendiculaires
invsens(h1,h2)	les demi-droites h1 et h2 ont sens inverse
memesens(h1,h2)	les demi-droites h1 et h2 ont même sens
distancep(d,p1,p2)	d est la distance entre les points p1 et p2
infdis(d1,d2)	la distance d1 est inférieure à la distance d2
demidis(d1,d2)	la distance d1 est la moitié de la distance d2
sommdis(d,d1,d2)	la distance d est la somme des distances d1 et d2
egalpt(p1,p2)	les points p1 et p2 sont égaux
egaldr(l1,l2)	les droites d1 et d2 sont égales
egaldd(h1,h2)	les demi-droites h1 et h2 sont égales
egalseg(s1,s2)	les segments s1 et s2 sont égaux
egalcc(c1,c2)	les cercles c1 et c2 sont égaux
egaldis(d1,d2)	les distances d1 et d2 sont égales

3 Définition de la spécification de l'exercice

Nous définissons ici la spécification de l'exercice fourni à l'élève par le professeur. Les données nécessaires à l'exécution du problème sont :

- - une spécification sous forme d'énoncé
- - une théorie instrumentale de la géométrie TIG

3.1 Spécification du professeur

% Soit une droite Ax
% Soit un segment [A B]
% Construire le segment [A' B'] symétrique de [A B] par rapport à la droite Ax

Spécification (CDL)	Traduction LDL
droite(Ax),	droite(Ax)
s1 = [A B],	point(A) point(B) segment(s1,A,B,var1)
s2 = [A' B'],	point(A') point(B') segment(s2,A',B',var2)
s3 = (A A'),	droite(s3) appdr(A,s3) appdr(A',s3)
s4 = (B B'),	droite(s4) appdr(B,s4) appdr(B',s4)
s3 !- Ax, s4 !- Ax,	perp(s3,Ax) perp(s4,Ax)
p e Ax, p e s3,	point(p) appdr(p,Ax) appdr(p,s3)
q e Ax, q e s4,	point(q) appdr(q,Ax) appdr(q,s4)
|A p| = |p A'|,	distance(var4) distancep(var4,A,p) distancep(var4,p,A')
|B q| = |q B'|.	distance(var8) distancep(var8,q,B') distancep(var8,B,q)

3.2 Définition de TIG

11.1; "Un point appartient à un cercle de centre o et de rayon r si il est à une distance r de o"

```
appcc(p, c) ->
     cercle(c, o, r)
     distancep(r, o, p) ;
```

16.1; "La propriété distancep est symétrique"

```
distancep(r, o, p) ->
     distance(r)
     point(p)
     point(o)
     distancep(r, p, o);
```

16.2; "Un point est à une distance r d'un point o s'il appartient au cercle de rayon r et de centre o"

```
distancep(r, o, p) ->
     cercle(c, o, r)
     distance(r)
     point(p)
     appcc(p, c) ;
```

4 Constructions correctes

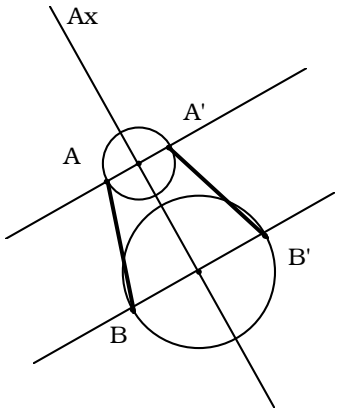
Les exemples de constructions ci-dessous appartiennent à la conception correcte de symétrie orthogonale par rapport à un axe. De façon à varier les axiomes mis en oeuvre dans le diagnostic de TALC, nous en avons choisis deux mettant en oeuvre des procédures différentes:

- - symétrique par un report orthogonal des distances.
- - symétrique par intersection des droites supports des segments.

Pour chaque construction, nous donnons l'énoncé Cabri-Géomètre associé et sa traduction en langage LDL.

4.1 Construction par report orthogonal des distances

L'élève construit le segment image en reportant les distances de chacune des extrémités du segment cible à l'axe sur les perpendiculaires à ce dernier.



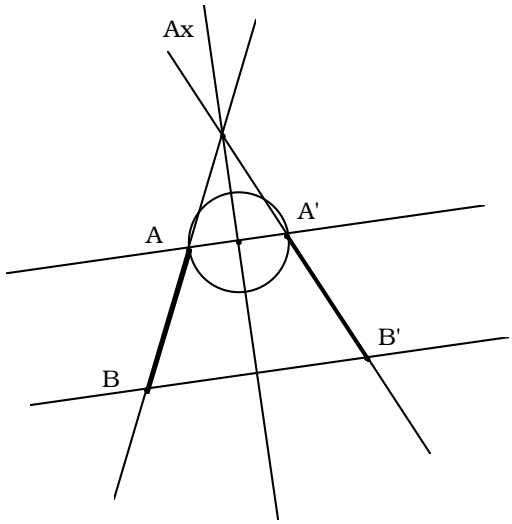
Enoncé Cabri-Géomètre	Traduction LDL
La figure comporte 13 objets.	
Ax : droite quelconque	droite(Ax)
A : point quelconque	point(A)
B : point quelconque	point(B)
segment [A B]	segment(var2,A,B,var1)
D#2 : droite passant par A et perpendiculaire à Ax	droite(VarD2) appdr(A,VarD2) perp(VarD2,Ax)
D#3 : droite passant par B et perpendiculaire à Ax	droite(VarD3) appdr(B,VarD3) perp(VarD3,Ax)
P#3 : intersection des 2 droites D#2 et Ax	point(VarP3) appdr(VarP3,VarD2) appdr(VarP3,Ax)
P#4 : intersection des 2 droites D#3 et Ax	point(VarP4) appdr(VarP4,VarD3) appdr(VarP4,Ax)
C#1 : cercle de centre P#3 passant par A	distance(var3) appcc(A,VarC1)
C#2 : cercle de centre P#4 passant par B	cercle(VarC1,VarP3,var3)
A' : intersection de la droite D#2 et du cercle C#1 (A est l'autre point)	distance(var4) appcc(B,VarC2)
B' : intersection de la droite D#3 et du cercle C#2 (B est l'autre point)	cercle(VarC2,VarP4,var4)
segment [A' B']	point(A') appdr(A',VarD2) appcc(A',VarC1)
	point(B') appdr(B',VarD3) appcc(B',VarC2)
	segment(var6,A',B',var5)

4.2 Construction par intersection des supports

L'élève construit le symétrique orthogonale de la première extrémité du segment cible puis obtient l'image de la deuxième extrémité par l'intersection des deux droites

Enoncé Cabri-Géomètre	Traduction LDL
La figure comporte 14 objets.	
Ax : droite quelconque	droite(Ax)
A : point quelconque	point(A)

B : point quelconque	point(B)
segment [A B]	segment(var2,A,B,var1)
D#2 : droite passant par A et perpendiculaire à Ax	droite(VarD2) appdr(A,VarD2) perp(VarD2,Ax)
D#3 : droite passant par B et perpendiculaire à Ax	droite(VarD3) appdr(B,VarD3) perp(VarD3,Ax)
D#4 : droite passant par A et B	droite(VarD4) appdr(A,VarD4) appdr(B,VarD4)
P#3 : intersection des 2 droites Ax et D#4	point(VarP3) appdr(VarP3,Ax) appdr(VarP3,VarD4)
P#4 : intersection des 2 droites D#2 et Ax	point(VarP4) appdr(VarP4,VarD2) appdr(VarP4,Ax)
C#1 : cercle de centre P#4 passant par A	cercle(VarC1,VarP4,var3) distance(var3)
A' : intersection de la droite D#2 et du cercle C#1 (A est l'autre point)	appcc(A,VarC1) point(A') appdr(A',VarD2) appcc(A',VarC1)
D#5 : droite passant par P#3 et A'	
B' : intersection des 2 droites D#3 et D#5	droite(VarD5) appdr(VarP3,VarD5) appdr(A',VarD5)
segment [A' B']	point(B') appdr(B',VarD3) appdr(B',VarD5) segment(var5,A',B',var4)



5 Constructions incorrectes

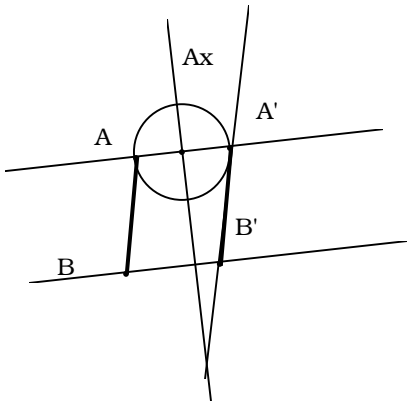
Les constructions suivantes mettent en oeuvre l'un des conceptions incorrectes de la symétrie orthogonale mises en évidence par S. TAHRI et présentées ci-dessus.
De la même façon que pour les constructions correctes, nous donnons pour chacune l'énoncé Cabri-Géomètre associé et sa traduction LDL.

5.1 Conception de "parallélisme"

L'image du segment est obtenue par une procédure de parallélisme.

Enoncé Cabri-Géomètre	Traduction LDL
La figure comporte 11 objets.	
Ax : droite quelconque	droite(Ax)
A : point quelconque	point(A)
B : point quelconque	point(B)
segment [A B]	segment(var2,A,B,var1)
D#2 : droite passant par A et perpendiculaire à Ax	droite(VarD2) appdr(A,VarD2) perp(VarD2,Ax)

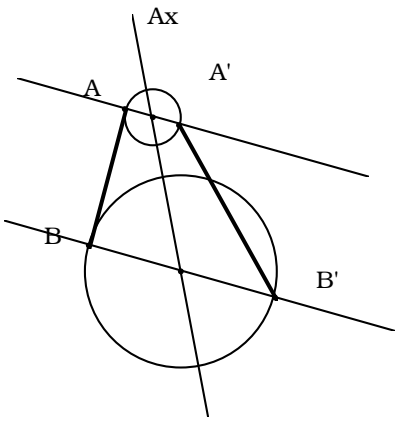
P#3 : intersection des 2 droites D#2 et Ax	point(VarP3) appdr(VarP3,VarD2) appdr(VarP3,Ax)
C#1 : cercle de centre P#3 passant par A	cercle(VarC1,VarP3,var3) distance(var3) appcc(A,VarC1)
D#3 : droite passant par B et perpendiculaire à Ax	droite(VarD3) appdr(B,VarD3) perp(VarD3,Ax)
A' : intersection de la droite D#2 et du cercle C#1 (A est l'autre point)	point(A') appdr(A',VarD2) appcc(A',VarC1)
D#4 : droite passant par A' et parallèle au segment [AB]	droite(VarD4) appdr(A',VarD4) par(VarD4,var1)
B' : intersection des 2 droites D#3 et D#4	point(B') appdr(B',VarD3) appdr(B',VarD4)
segment [A' B']	segment(var5,A',B',var4)



5.2 Conception de "symétrie oblique"

L'image du segment est obtenue par symétrie oblique, c'est-à-dire que toutes les propriétés de la figure sont exactes, sauf l'orthogonalité des droites (AA') et (BB') par rapport à l'axe Ax.

Enoncé Cabri-Géomètre	**Traduction LDL**
La figure comporte 13 objets.	
Ax : droite quelconque	droite(Ax)
A : point quelconque	point(A)
B : point quelconque	point(B)
segment [A B]	segment(var2,A,B,var1)
P#3 : point de la droite Ax	point(VarP3) appdr(VarP3,Ax)
D#2 : droite passant par A et P#3	droite(VarD2) appdr(A,VarD2) appdr(VarP3,VarD2)
D#3 : droite passant par B et parallèle à D#2	droite(VarD3) appdr(B,VarD3) par(VarD3,VarD2)
C#1 : cercle de centre P#3 passant par A	cercle(VarC1,VarP3,var3) distance(var3) appcc(A,VarC1)
P#4 : intersection des 2 droites D#3 et Ax	point(VarP4) appdr(VarP4,VarD3) appdr(VarP4,Ax)
C#2 : cercle de centre P#4 passant par B	cercle(VarC2,VarP4,var4) distance(var4) appcc(B,VarC2)
A' : intersection de la droite D#2 et du cercle C#1 (A est l'autre point)	point(A') appdr(A',VarD2) appcc(A',VarC1)
B' : intersection de la droite D#3 et du cercle C#2 (B est l'autre point)	point(B') appdr(B',VarD3) appcc(B',VarC2)
segment [A' B']	segment(var6,A',B',var5)



5.3 Conception de "symétrie centrale"

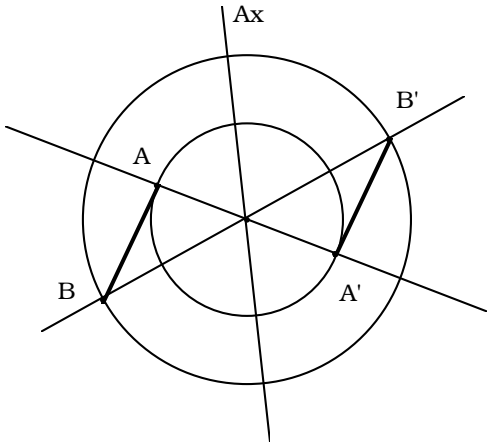
L'image du segment cible est obtenue par une symétrie centrale (par rapport à un point) et non par symétrie orthogonale (par rapport à un axe).

Deux types de symétrie centrale mettant en oeuvre des objets géométriques différents sont présentés:

- - une symétrie centrale par un point quelconque sur l'axe
- - une symétrie centrale par prolongement du segment cible.

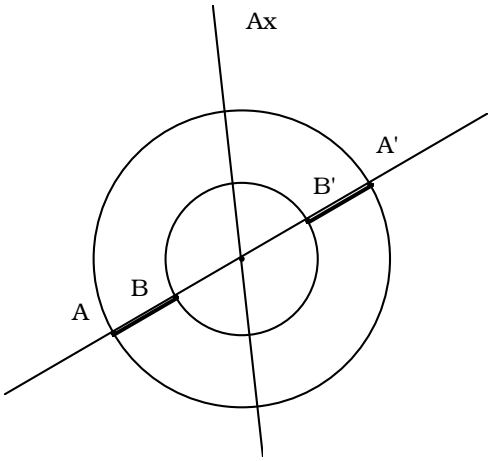
5.3.1 Symétrie centrale quelconque

Enoncé Cabri-Géomètre	**Traduction LDL**
La figure comporte 12 objets.	
Ax : droite quelconque	droite(Ax)
A : point quelconque	point(A)
B : point quelconque	point(B)
segment [A B]	segment(var2,A,B,var1)
P#3 : point de la droite Ax	point(VarP3) appdr(VarP3,Ax)
D#2 : droite passant par A et P#3	droite(VarD2) appdr(A,VarD2) appdr(VarP3,VarD2)
D#3 : droite passant par B et P#3	droite(VarD3) appdr(B,VarD3) appdr(VarP3,VarD3)
C#1 : cercle de centre P#3 passant par A	cercle(VarC1,VarP3,var3) distance(var3)
C#2 : cercle de centre P#3 passant par B	appcc(A,VarC1)
A' : intersection de la droite D#2 et du cercle C#1 (A est l'autre point)	cercle(VarC2,VarP3,var4) distance(var4) appcc(B,VarC2)
B' : intersection de la droite D#3 et du cercle C#2 (B est l'autre point)	point(A') appdr(A',VarD2) appcc(A',VarC1)
segment [A' B']	point(B') appdr(B',VarD3) appcc(B',VarC2) segment(var6,A',B',var5)



5.3.2 Symétrie centrale par prolongement

Enoncé Cabri-Géomètre	Traduction LDL
La figure comporte 11 objets.	
Ax : droite quelconque	droite(Ax)
A : point quelconque	point(A)
B : point quelconque	point(B)
segment [A B]	segment(var2,A,B,var1)
D#2 : droite passant par A et B	droite(VarD2) appdr(A,VarD2) appdr(B,VarD2)
P#3 : intersection des 2 droites D#2 et Ax	point(VarP3) appdr(VarP3,VarD2) appdr(VarP3,Ax)
C#1 : cercle de centre P#3 passant par B	cercle(VarC1,VarP3,var3) distance(var3)
C#2 : cercle de centre P#3 passant par A	appcc(B,VarC1)
A' : intersection de la droite D#2 et du cercle C#2 (A est l'autre point)	cercle(VarC2,VarP3,var4) distance(var4) appcc(A,VarC2)
B' : intersection de la droite D#2 et du cercle C#1 (B est l'autre point)	point(A') appdr(A',VarD2) appcc(A',VarC2)
segment [A' B']	point(B') appdr(B',VarD2) appcc(B',VarC1)
	segment(var6,A',B',var5)



5.4 Conception "perceptive"

Cette conception incorrecte de la symétrie orthogonale d'un segment est un peu différente des trois autres: il s'agit d'une construction globale ou semi-globale, ne faisant pas appel à la totalité des objets géométriques nécessaires, mais les propriétés géométriques exploitées par l'apprenant peuvent être tout à fait correctes. En

d'autres termes, nous sommes sur que la définition du concept ne pourra pas se faire de façon suffisante uniquement grâce aux éléments de la théorie du domaine. L'invention de prédicat de la PLI est ici certainement indispensable

Il existe un grand nombre de conceptions perceptives; l'exemple que nous donnons ci-dessous est une construction semi-globale correspondant à un report approximatif de la distance orthogonale de la deuxième extrémité du segment. Les deux figures correspondent à la même construction et montrent l'ambiguïté d'une construction perceptive : la première construction semble correcte.

Enoncé Cabri-Géomètre	**Traduction LDL**
La figure comporte 11 objets.	
Ax : droite quelconque	droite(Ax)
A : point quelconque	point(A)
B : point quelconque	point(B)
segment [A B]	segment(var2,A,B,var1)
D#2 : droite passant par A et perpendiculaire à Ax	droite(VarD2) appdr(A,VarD2) perp(VarD2,Ax)
D#3 : droite passant par B et perpendiculaire à Ax	droite(VarD3) appdr(B,VarD3) perp(VarD3,Ax)
P#3 : intersection des 2 droites Ax et D#2	point(VarP3) appdr(VarP3,Ax) appdr(VarP3,VarD2)
C#1 : cercle de centre P#3 passant par A	cercle(VarC1,VarP3,var3) distance(var3)
A' : intersection de la droite D#2 et du cercle C#1 (A est l'autre point)	appcc(A,VarC1) point(A') appdr(A',VarD2) appcc(A',VarC1)
B' : point de la droite D#3	
segment [A' B']	point(B') appdr(B',VarD3) segment(var5,A',B',var4)