Amine : une plate-forme pour le développement de systèmes et d'agents intelligents

Adil Kabbaj

INSEA, BP 6217, Rabat, Maroc

akabbaj@insea.ac.ma

Karim Bouzouba

EMI, Université Mohamed V - Agdal, Av. Ibn Sina, B-P 765, Rabat, Maroc

karim.bouzoubaa@emi.ac.ma

Résumé

Cet article présente Amine, une plate-forme open-source et multicouche, implémentée en Java et dédiée au développement de systèmes intelligents et au développement d'agents (http://sourceforge.net/projects/amine-platform). La plate-forme Amine est composée de quatre couches : une couche « noyau » qui permet la création, l'édition, la mise à jour et la manipulation d'ontologies multilingues, une couche algébrique qui offre un ensemble de classes (structure + opérations) ; une couche de programmation et une quatrième couche permettant le développement d'agents et de systèmes multi-agents. La couche de programmation fournit trois paradigmes de programmation : un paradigme de programmation basée sur la mémoire, un paradigme de programmation à base de règles incorporé dans le langage PROLOG+CG et un paradigme de programmation visuelle à base d'activation et de propagation, incorporé dans le langage SYNERGY.

Abstract

This paper presents an overview of Amine; a multi-layer and open-source platform, implemented in Java and dedicated to the development of intelligent systems and agents. Amine is composed of four layers : a) a kernel layer that enables the creation, edition, update and manipulation of multi-lingua ontologies, b) an algebraic layer that offers a set of elementary data types, structured types and various matching-based operations, c) a programming layer that provides three programming paradigms: i) an ontology or memory-based programming paradigm which is concerned by incremental and automatic integration of knowledge in an ontology (or agent memory), ii) a pattern-matching and rule-based programming paradigm, embedded in PROLOG+CG language, and iii) an activation and propagation-based programming paradigm, embedded in SYNERGY language, and d) an agent and multi-agent systems layer that enables the development of agent-based applications.

Mots-clés

plate-forme de développement, systèmes intelligents, ontologie, graphe conceptuel, extension Prolog, agent, Java

Keywords

development platform, intelligent systems, ontology, conceptual graph, Prolog, agent, Java

1. Introduction

Un système intelligent est un programme informatique présentant une certaine forme d'intelligence. Le développement de tels systèmes a débuté vers les années 50. Nous pouvons identifier quatre périodes dans le développement de tels systèmes, chaque période a permis d'explorer différents types de systèmes intelligents :

- Exploration de processus intelligents et développement des premiers systèmes intelligents,
- Systèmes à bases de connaissances et de méta-connaissances, et systèmes intentionnels,
- Systèmes intelligents autonomes,
- Agents autonomes intégrés.

Durant ces périodes, différents outils de développement ont vu le jour : les langages de programmation dédiés tels que Lisp et Prolog, les formalismes de représentation (frames, scripts, réseau sémantique), les coquilles de développement de systèmes experts et les coquilles de développement d'agents intelligents. Chaque outil de développement ou formalisme de représentation répond à un besoin spécifique.

A titre d'exemple, Amzi Prolog <http://www.amzi.com> et WinProlog <http://www.lpa.co.uk/win.htm> sont des outils qui permettent d'éditer, d'analyser et d'interpréter un programme prolog. Ils permettent également de faire appel à ces programmes à partir de langages compilés tels que C. Ils ne permettent cependant pas d'engager de raisonnement à partir d'ontologies existantes et n'intègrent pas de formalisme de représentation des connaissances.

Par ailleurs, pour le raisonnement sémantique, les graphes conceptuels (GC) sont des formalismes de représentation de connaissance très largement utilisés (Sowa 1984, 2000). CoGITaNT <http://cogitant.sourceforge.net> et CharGer <http://www.cs.uah.edu/~delugach/CharGer/> sont des outils d'édition et de manipulation des GC. Cependant, ces outils ne fournissent pas de mécanisme d'inférence afin de mener des raisonnements.

Une ontologie représente l'ensemble des concepts (ou catégories) et des relations utilisés pour formuler des propositions à propos d'un domaine donné. Une grande variété d'ontologies ont été développées : une ontologie linguistique comme WordNet (www.cogsci.princeton.edu/~wn), une ontologie encyclopédique comme CYC (www.cyc.com), et des milliers d'ontologies pour différents domaines. L'importance des ontologies a été révélée avec le développement du web sémantique. Le web sémantique permet de structurer le contenu sémantique des documents web et ainsi permettre à des programmes et à des agents d'interpréter leur contenu et de raisonner à partir de ces documents.

De nombreux langages pour le web sémantique ont ainsi été définis ces dernières années. Des outils d'édition et de gestion de documents ont été développés afin de valider ces langages. A titre d'exemple, on peut citer les langages XOL (XML based ontology exchange language) (www.ai.sri.com/~pkarp/xol/), Topic Maps (un standard ISO pour la description de représentations de connaissances) (www.topicmaps.org), RDF/RDFS (Resource Description Framework et RDF Schema) (www.w3.org/RDF/), DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer) (www.daml.org), et OWL (Web Ontology Language) (www.w3.org/TR/owl-features/).

Il est également possible de raisonner sur les ontologies en utilisant un moteur d'inférence général tel que JESS (herzberg.ca.sandia.gov/jess/), ou des outils d'inférence caractéristiques du web sémantique fondés sur des logiques de description (dl.kr.org/) tels que RACER (www.sts.tu-harburg.de/~r.f.moeller/racer/). Il existe aussi des éditeurs d'ontologies comme

Protégé-2000 (protege.stanford.edu/) qui permet de construire une ontologie selon les langages précités et d'intégrer des moteurs d'inférence comme JESS ou RACER.
L'apport de la plate-forme Amine dans le domaine d'ontologies se résume par les points suivants :

- un méta-modèle fondé sur les structures conceptuelles de la théorie des graphes conceptuels (GC) ;
- l'emploi des GC comme langage de représentation de connaissances ;
- un processus d'intégration automatique de nouvelles connaissances à l'ontologie ;
- des processus de base pour l'exploration d'une ontologie ;
- l'usage d'une ontologie à partir du langage Prolog+CG (qui permet l'inférence) ;
- L'utilisation des outils sur les ontologies à une plate-forme intégrée

Pour la construction de systèmes multi-agents, SMA, plusieurs plates-formes ont vu le jour. JADE <http://jade.cselt.it/> ou Jason <http://jason.sourceforge.net> en sont des exemples. Cependant, ces outils ne gèrent que la communication entre agents sans possibilité de manipulation des graphes conceptuels ou d'utilisation d'une ontologie particulière.
Ainsi, disposer d'un environnement de développement intégré reste un souci constant pour le développement de divers systèmes intelligents. La plate-forme Amine vise à offrir un tel environnement. Elle comprend des outils de définition et de manipulation d'ontologies pouvant être exprimés par plusieurs formalismes de représentation et par des moteurs d'inférence incluant un langage de programmation logique et conceptuel. Il s'agit d'une plate-forme open-source[1] et multicouche, implémentée en Java et dédiée au développement de systèmes intelligents et au développement d'agents (Kabbaj et al. 2005). La figure 1 montre l'architecture de la version courante, Amine 3. Elle est composée de quatre couches. La première constitue le noyau et concerne la création et la manipulation d'ontologies multilingues. La deuxième couche algébrique fournit plusieurs types de données élémentaires, et composées et plusieurs opérations associées. La troisième couche offre des mécanismes d'inférence. La quatrième offre la possibilité d'implémenter les concepts d'agents et systèmes multi-agents.

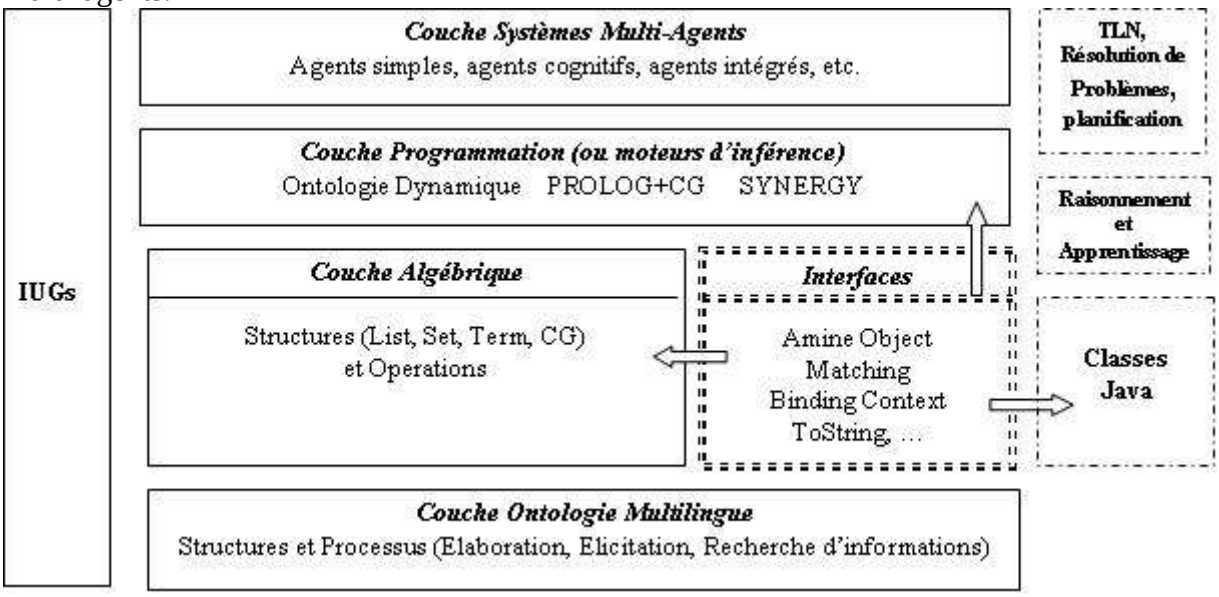

Figure 1. L'architecture de la plate-forme Amine

[1] http://sourceforge.net/projects/amine-platform

Les quatre couches forment une hiérarchie : chaque couche est construite sur la base et utilise les couches inférieures. Pour offrir une conception et une implémentation d'Amine avec un haut niveau de modularité chaque couche peut être utilisée de manière indépendante.
Amine peut être vu comme un environnement de programmation basé sur l'ontologie ou comme la base d'une machine virtuelle intelligente. La sémantique des catégories (types) utilisées par le système/agent est précisée dans l'ontologie grâce à la couche noyau. Différentes structures (AmineList, Term, CG) peuvent être utilisées par le système ainsi que plusieurs opérations basées sur l'appariement -matching- grâce à la couche algébrique. Les éléments de ces structures peuvent être des catégories mentionnées dans l'ontologie. La couche programmation fournit trois paradigmes de programmation où l'ontologie, les structures et les opérations peuvent être utilisées. L'ontologie, les structures, les opérations basées sur l'appariement et les paradigmes de programmation sont les ingrédients permettant la formulation et le développement de plusieurs stratégies d'inférence (induction, déduction, abduction, analogie) et de processus cognitifs. Ils sont aussi des ingrédients nécessaires au développement de la mémoire dynamique. Et tel que énoncé par Michalski, "Apprentissage = Mémoire + Inférences" (Michalski 1994), tous ces ingrédients, avec les stratégies d'inférence, les capacités de mémoire et d'apprentissage, forment les composantes fondamentales d'une machine virtuelle intelligente.
Amine fournit également plusieurs interfaces utilisateurs graphiques (IUG). La figure 2 en présente la fenêtre principale « Amine Suite Panel » qui permet l'accès à toutes les IUG, ainsi que l'accès à quelques exemples d'ontologie et à des tests qui illustrent l'utilisation des structures fournies et leur API.

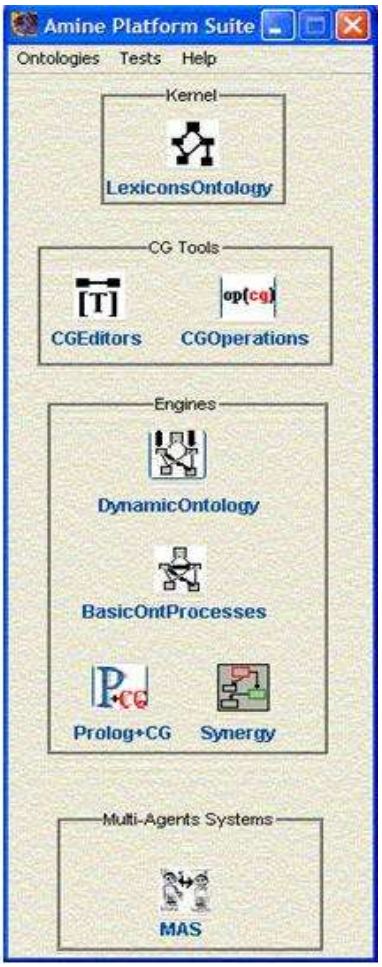

Figure2. Fenêtre principale d'Amine « Amine Suite Panel »

La suite de cet article est organisée comme suit. La deuxième section introduit brièvement la couche noyau. La troisième section présente la couche algébrique. La section quatre présente la dernière version de Prolog+CG (les composants DynamicOntology (Bouzouba et Kabbaj 2004) et SYNERGY (Kabbaj 1999) ne sont pas décrits dans cet article). La cinquième section décrit notre approche du développement d'applications fondées sur les agents. La sixième section conclut et expose quelques orientations pour nos travaux futurs.

2. La couche noyau

La couche noyau permet la création et la manipulation d'ontologies multilingues (Bouzouba et Kabbaj, 2004). Rappelons qu'une ontologie représente l'ensemble des concepts (ou des catégories) et des relations utilisées pour formuler des propositions à propos d'un domaine donné (Sowa 2000). Ce catalogue de catégories et de relations est organisé en général dans une hiérarchie appelée hiérarchie de types. Cette définition d'ontologie peut être étendue en fournissant à chaque type toutes les connaissances spécifiques acquises par le système. Ces connaissances peuvent être organisées en termes de structures conceptuelles (SC) à l'aide d'une définition, d'un canon (description des contraintes respectées par le type) et de schémas (description de situations) associés. Les individus (instances) peuvent être associés à leur type. Ainsi, une ontologie dans Amine est un graphe de nœuds qui représentent les structures conceptuelles. Actuellement, cinq types de nœuds sont définis.

- Nœud type : le nœud représente un type (une catégorie). Ce nœud peut contenir la définition du type ou le canon du type.
- Nœud Individu : le nœud représente un individu (ou instance) et peut contenir sa propre description.
- Nœud Situation : le nœud contient la description d'une situation. Une situation peut être spécifique ou générique et peut être indexée sous plusieurs types ou individus.
- Nœud contexte : le nœud contient la description d'un contexte. Ce type de nœud est utilisé pour la représentation et l'intégration de descriptions composées (tel que les GCs composés).
- Nœud métaphore : le nœud représente une métaphore.

La figure 3 fournit une image d'écran d'une ontologie éditée avec Amine. L'ontologie est visualisée dans la fenêtre principale avec une structure hiérarchique. Trois nœuds représentent des SC de l'ontologie (chaque type de nœud est représenté par une couleur différente). Trois types de relations sont possibles :

- lien de spécialisation noté `-s->` : un type peut être spécialisé par d'autres types ;
- lien individuel noté `-i->` : un type peut être relié à plusieurs individus ;
- lien d'utilisation noté `-u->` : un type peut être relié à plusieurs situations.

Le contenu d'un nœud est visualisé, si cela est souhaité, dans une autre fenêtre. La figure 3 montre d'une part le contenu du type "Homme" : le nœud contient une définition "personne de sexe mâle" et ne contient pas de canon et, d'autre part deux situations associées à ce type : la situation "Homme qui boit de l'eau" et la situation "Homme qui conduit la voiture". Amine permet aussi de visualiser et d'explorer graphiquement une ontologie.

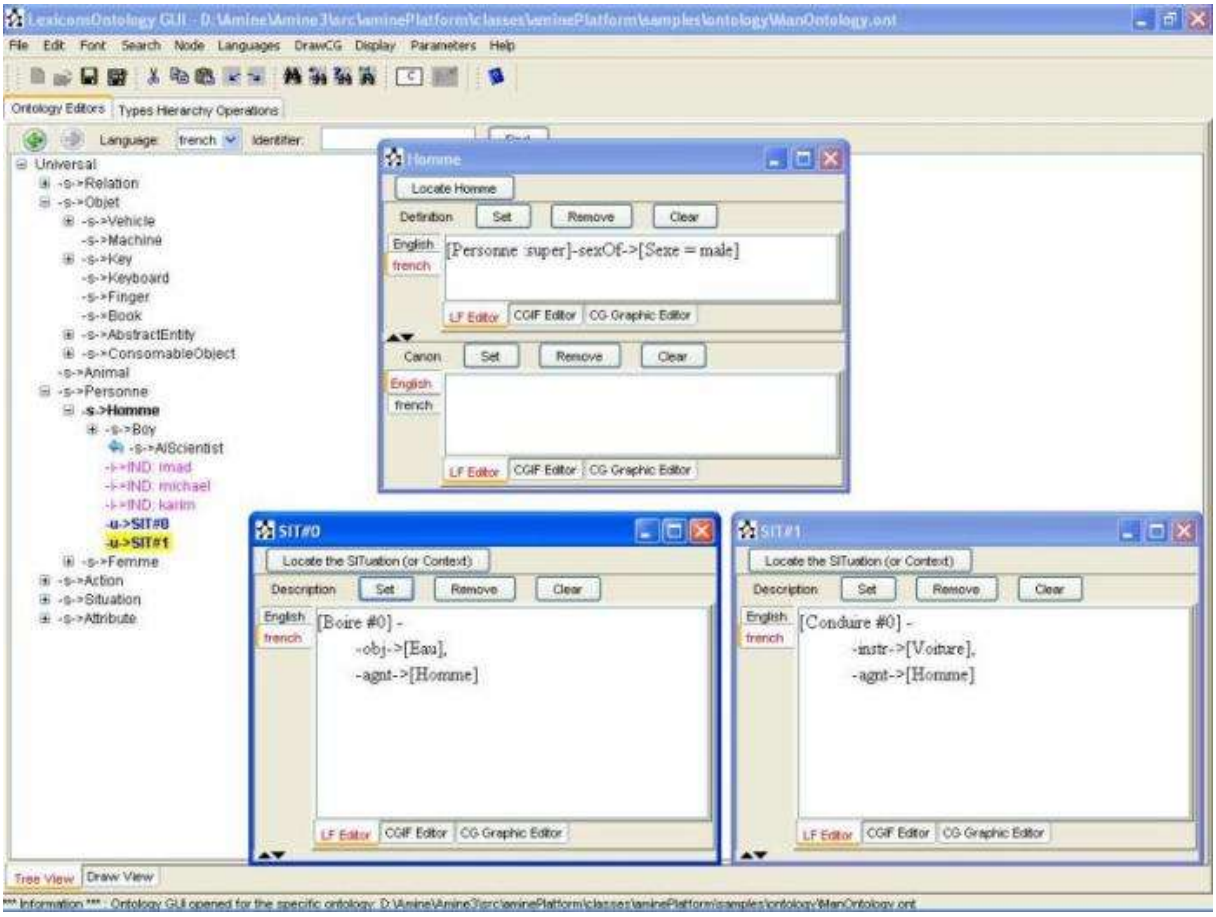

Figure 3: Un exemple d'une ontologie dans Amine

3. La couche algébrique

Cette couche fournit différents types de structure et d'opération.
- Structure : en plus des objets Java élémentaires et des objets élémentaires propres à Amine, les structures de collection[2] et les structures complexes[3] sont fournies.
- Opération : en plus des opérations particulières de chaque type de structure, un ensemble d'opérations de base[4] et un ensemble d'opérations d'appariement[5] sont fournis.
Ces opérations sont définies respectivement dans AmineObject et dans des interfaces Java implémentées par les structures d'Amine. Toute classe de Java qui implémente AmineObject et les interfaces d'appariement deviendra un membre entier de la plate-forme ce qui améliore l'ouverture d'Amine envers Java.

4. Re-ingénierie et intégration de Prolog+CG

Des versions antérieures de Prolog+CG ont déjà été présentées dans (Kabbaj et Moulin 2001, Kabbaj et al. 2001, Kabbaj et Janta 2000). Rappelons trois caractéristiques clés de Prolog+CG :

[2] AmineList qui étend la classe ArrayList de Java et AmineSet qui étend la classe HashSet de Java
[3] Term, Concept, Relation et Conceptual Graph
[4] clear(), clone(), toString()
[5] match(), equal(), unify(), subsume(), maximalJoin(), generalize()

- un GC, simple ou composé, est une structure primitive dans Prolog+CG, tels qu'une liste ou un terme. Un GC peut être utilisé comme une structure ou comme une représentation d'un but ;
- par un mécanisme d'indexation supplémentaire des règles. Prolog+CG offre une extension de Prolog fondée sur l'objet ;
- Prolog+CG offre une interface avec Java. Les objets Java peuvent être créés et les méthodes peuvent être appelées à partir d'un programme Prolog+CG. Aussi, Prolog+CG peut être activé à partir de classes Java.

Ces trois caractéristiques clés sont toujours présentes dans la version actuelle de Prolog+CG mais la ré-ingénieurie de Prolog+CG, qui était nécessaire pour son intégration dans Amine, a induit quelques changements dans le langage. Voici les quatre modifications les plus intéressantes.

- La hiérarchie des types et les SC n'est plus exprimée à l'intérieur d'un programme Prolog+CG. Les programmes Prolog+CG sont interprétés selon une ontologie courante. La hiérarchie des types et les SC sont lisibles directement à partir de cette ontologie (qui les contient). Aussi, Prolog+CG tente d'interpréter chaque identifiant dans un programme selon le lexique courant de l'ontologie courante. Si l'identifiant n'est pas trouvé, alors il est considéré comme un simple identifiant (sans aucune sémantique associée) ;
- La notion de projet est introduite. L'utilisateur peut consulter plusieurs programmes, et non plus un seul, à la fois ;
- Prolog+CG hérite des deux premières couches d'Amine. Toutes les structures et les opérations définies dans Amine le sont également dans Prolog+CG. L'utilisateur peut manipuler l'ontologie courante et les lexiques associés selon leur API. Les opérations, implémentées en tant que méthodes, peuvent être utilisées grâce à la nouvelle interface entre Prolog+CG et Java ;
- L'interface entre Prolog+CG et Java est plus simple. L'appel d'une méthode dans Prolog+CG est très similaire à son appel dans Java comme illustré par les exemples suivants.

```
length(L, s) :-          // Supposons que la valeur de L est une AmineList,
s is L:size().           // appel de la méthode Java size() et assignation du
                         // résutat à la variable s.
maxJoin(G1, G2, G3) :-   // appel de la méthode maximalJoin sur G1 et G2, et
G3 is G1:maximalJoin(G2).// assignation du résultat à G3
```

5. La couche agents et systèmes multi-agents

Au lieu de développer notre propre environnement de développement agent en Java, nous avons opté pour l'intégration d'un environnement open-source dans Amine. Nous explorons actuellement[6] l'utilisation de JADE (jade.tilab.com), un environnement de développement agent open-source basé sur Java. Il offre la possibilité de créer des agents avec différents types de comportement et différentes capacités de communication. La première application développée porte sur l'utilisation de JADE et Amine dans le but de simuler une situation de tutorat où des étudiants interagissent avec leur enseignant. Les étudiants posent des questions et l'enseignant répond. L'enseignant a un accès à une ontologie et connaît quelles primitives (méthodes du noyau) utiliser pour répondre à des questions particulières. Une autre application en cours de développement teste le cas d'un système multi-agents qui possède des agents cognitifs simulant un conte.

[6] Dans le contexte du DESA intitulé "L'utilisation d'Amine dans le développement d'applications basées sur les agents" de ElHari

6. Conclusion

Cet article fournit un aperçu de la plate-forme Amine : plate-forme open source, implémentée en Java et dédiée au développement de systèmes intelligents et au développement d'agents. Amine possède une architecture multicouche composée de quatre couches et de plusieurs IUG. Ces couches forment une hiérarchie, chaque couche est construite à partir des couches inférieures Cependant, une couche peut être utilisée sans les couches supérieures.
Le travail futur aura trait à l'amélioration de la plate-forme et au développement d'applications dans différents domaines, notamment le traitement du langage naturel, la résolution de problèmes et la planification, les systèmes à base de cas et les systèmes multi-agents.

Références

Bouzouba K., Kabbaj A. (2004) Construction et manipulation d'ontologies multi-lingue, sémantique et dynamique dans Amine Platform, *Conférence Plénière sur les Technologies de l'Information et de la Communication COPSTIC*, Rabat, Morocco, Janvier.
Kabbaj A. (1999) Synergy: a conceptual graph activation-based language, in Proc. Of the 7th *International Conference on Conceptual Structures* ICCS.
Kabbaj, A. et al. (2001) Uses, Improvements and Extensions of Prolog+CG: Case studies, in Proc. Of the *International Conference on Conceptual structures* ICCS, San Francisco, August.
Kabbaj, A., Bouzouba, K. Soudi, A. (2005) Amine Platform : an Artificial Intelligence Environment for the Development of Intelligent Systems, *Information and Communication Technologies International Symposium*, Tetuan, Maroc, June.
Michalski R. S. (1994) Inferential Learning Theory : Developing Theoretical Foundations for Multistrategy Learning" in *Machine Learning: A Multistrategy Approach*, Vol. 4, R. S. Michalski and G. Tecuci (Eds.), San Mateo, CA, Morgan Kaufmann.
Sowa, J. F. (2000) *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks Cole Publishing.