

Curriculum Vitae de Alain Colmerauer

Dernière mise à jour, novembre 2007

Etat civil

Alain Marie Albert Colmerauer, né le 24 janvier 1941 à Carcassonne, de nationalité française, marié à Colette Coursaget, trois enfants, habitant à : 3 impasse des Iris, 13009 Marseille. Téléphone 04 91 40 65 47, courrier alain.colmerauer@free.fr, site <http://alain.colmerauer.free.fr>.

Diplômes

Ingénieur en Informatique de l'Institut Polytechnique de Grenoble en 1963. Doctorat d'état en Informatique en 1967 à l'Université de Grenoble : *Précédences, analyse syntaxique et langages de programmation*, sous la direction de Louis Bolliet et Jean Kuntzman.

Postes occupés

- 1963–1966. Chercheur contractuel à l'Université de Grenoble dans l'équipe de Louis Bolliet.
- 1966–1967. Attaché de recherche au CNRS à l'Université de Grenoble dans l'équipe de Louis Bolliet.
- 1967–1970. Professeur Assistant d'informatique à l'Université de Montréal
- 1970–1979. Professeur 2ème classe (Maître de conférence ancien régime) à la Faculté des Sciences de Luminy, Université II d'Aix-Marseille.
- 1979–1988. Professeur 1ère classe à la Faculté des Sciences de Luminy, Université II d'Aix-Marseille.
- 1988–2006 Professeur classe exceptionnelle à la Faculté des Sciences de Luminy, Université II d'Aix-Marseille.
- 2000–2006 Professeur classe exceptionnelle à la Faculté des Sciences de Luminy, Université II d'Aix-Marseille, Institut Universitaire de France.

Directions de laboratoires

- 1967–1970. Directeur du projet de Traduction Automatique de l'Université de Montréal (TAUM).
- 1973–1985. Directeur du Groupe d'Intelligence Artificielle (GIA), URA Cnrs 816, Faculté des Sciences de Luminy.
- 1991–1993. Directeur du Groupe d'Intelligence Artificielle (GIA), URA Cnrs 816, Faculté des Sciences de Luminy.
- 1993–1995. Directeur du Laboratoire d'Informatique de Marseille (LIM), URA Cnrs 1787, Université de Provence et Université de la Méditerranée.

Prix et distinctions

- Lauréat de la Pomme d'Or du Logiciel français 1982, prix décerné par la compagnie Apple France à H. Kanoui, M. Van Caneghem et lui-même.
- Lauréat de l'année 1984 du Conseil Régional Provence Alpes et Côte d'Azur.
- Prix Michel Monpetit 1985, décerné par l'Académie des Sciences.
- Chevalier de la Légion d'Honneur en 1986.
- Correspondant à l'Académie des Sciences en mathématiques depuis 1986.
- AAAI Fellow (American Association for Artificial Intelligence) en 1991 : "In recognition of Contributions to Natural Language Processing and to the Origination of the Prolog Language and Extensions to Handle Constraints."

1963–1967 Analyse syntaxique

Mes premières activités de recherche ont commencé en 1963 à Grenoble lors de la préparation de ma thèse de doctorat d'état. Je me suis intéressé à l'analyse des grammaires « context-free » dans le cadre de la compilation et plus spécialement dans le but de trouver en un seul passage le maximum d'erreurs syntaxiques dans un programme. Je me rappelle que mon analyseur remontant à précedence totale me permettait de prendre un paquet de cartes représentant un programme Algol, de le couper plusieurs fois comme des cartes à jouer et de demander à la machine de trouver les séquences déplacées [2].

1967–1970 Langages de programmations pour la traduction automatique

Etant devenu spécialiste d'analyse syntaxique, je me suis alors tourné vers des langages à syntaxe plus compliquée que celle des langages de programmation : les langages naturels. Dans le cadre du projet de Traduction Automatique de l'Université de Montréal j'ai tout d'abord écrit en 1968 un analyseur et un synthétiseur général pour W-grammaires [22], le formalisme de A. van Wijngaarden pour définir Algol68. Ceci a permis aux linguistes du projet d'aborder les problèmes divers de la chaîne complète de traduction de l'anglais vers le français : morphologie de l'anglais, analyse de l'anglais, transfert, synthèse du français, morphologie du français. Fort de cet expérience je développais en 1969 un formalisme mieux adapté au traitement de données linguistiques : les systèmes-Q [41, 11]. Il s'agissait de règles de réécriture du type : « la sous-suite d'arbres de telle forme se

réécrit dans la suite d'arbres de telle forme ». L'application de ces règles se faisait en parallèle et nécessitait la mise en œuvre d'un algorithme d'unification non-déterministe, tenant compte de l'associativité de la concaténation de listes mais présupposant toujours que l'un des deux termes à unifier ne contienne aucune variable. Les systèmes-Q peuvent être considérés comme l'ancêtre de Prolog [21]. L'expérience des systèmes-Q me renforça dans l'idée qu'il fallait développer des langages de programmation de très haut niveau, même si les temps d'exécution qu'ils impliquaient pouvaient sembler effarants !

Ces systèmes-Q furent utilisés pour construire une chaîne complète de traduction automatique anglais-français : la morphologie de l'anglais fut écrite par Brian Harris, Richard Kittredge écrivit une importante grammaire pour l'analyse de l'anglais, Gilles Stewart écrivit la phase de transfert, Jules Danserau écrivit la grammaire pour la synthèse du français et Michel van Caneghem produisit une morphologie complète du français [42]. Après mon retour en France les systèmes-Q furent aussi utilisés pour produire le système Meteo qui, au Canada, traduit quotidiennement les bulletins météorologiques de l'anglais au français.

1970–1974 Naissance de Prolog

De retour en France en 1970 je m'intéressais davantage à faire des déductions sur des textes qu'à les traduire dans une autre langue. Avec Jean Trudel (un étudiant québécois nourri de logique bien avant moi) et Philippe Roussel nous avons étudié ce qui se faisait en démonstration automatique et plus particulièrement l'article d'Alan Robinson sur le « Resolution Principle ». Nous avons eu des échanges très fructueux avec Robert Kowalski qui était alors chercheur à Edimbourg. Son travail sur la « SL-resolution » nous impressionna beaucoup et servit de base comme premier modèle théorique de Prolog. Cependant mon but n'était pas de créer un nouveau langage de programmation mais de pouvoir décrire, en français, un univers à l'ordinateur afin que celui-ci puisse répondre à des questions concernant cet univers. Nous fîmes un embryon d'un tel système [43]) et c'est à cette occasion que fut développé l'outil Prolog qui servait aussi bien pour faire la partie analyse-synthèse du français que la partie raisonnement pour répondre aux questions posées [21].

L'interpréteur Prolog écrit en W-Algol par Philippe Roussel fut revu un an après. J'introduisis notamment la fameuse opération de coupure de l'espace de recherche « slash », et Philippe, suite à des discussions avec R. Boyer et J. Moore (à l'époque à Edimbourg), conçut le premier interpréteur Prolog moderne à « structures partagées ». Il fut programmé en Fortran par H. Méloni et G. Battani, alors étudiants de DEA. C'est cette version de Prolog qui se répandit partout : tout d'abord à Edimbourg par l'intermédiaire de David Warren venu faire un stage chez nous, puis, comme nous l'apprîmes bien plus tard, en Hongrie, au SRI à Stanford et de là au Japon [21].

1974–1979 Travaux sur le langage naturel

Pour ma part j'ai continué à travailler sur les langues naturelles et en particulier sur la sémantique du Français

[3, 17]. Pour résoudre des problèmes de présupposition j'ai conçu différentes logiques à trois valeurs [16]. Cependant j'ai compris que l'application la plus intéressante de mes travaux était l'interrogation de bases de données. J'ai écrit une grammaire assez conséquente du français qui sera le point de départ de plusieurs applications. Veronica Dahl s'en servira pour faire un interface avec un configurateur (écrit en Prolog) pour la gamme d'ordinateurs Solar (nous faisons des systèmes experts sans le savoir). Dans le cadre de sa thèse de 3ème cycle, elle s'en inspirera aussi pour écrire une grammaire de l'espagnol. Jean François Pique, Paul Sabatier et Ch. Giraud utiliseront la même grammaire pour écrire un système d'interrogation de la hiérarchie militaire de l'infanterie française et ceci dans le cadre d'un contrat avec la Cap Sogeti. David Warren et Fernando Pereira partiront des travaux de Veronica Dahl pour écrire des systèmes d'interrogation en anglais et en portugais.

En 1976, sous l'impulsion de Michel Van Caneghem nous avons acheté notre premier micro-ordinateur : l'outil de développement de Motorola pour 6800 avec deux unités de disques souples et 48k (Exorciser). C'était la solution à tous nos problèmes de machines trop chères et inconnues en dehors de la France. Encore fallait-il y implanter Prolog ! Le défi fut relevé par H. Kanoui, M. Van Caneghem et moi-même en simulant une machine virtuelle comportant une mémoire virtuelle sur disque souple.

1979–1982 Prolog II et les premiers micro-ordinateurs

Le travail que nous avons fait sur l'Exorciser nous avait redonné le goût de développer Prolog. D'autre part les premiers Apple II que nous avons achetés dès 1977 disposaient de 64k de mémoire. Pourquoi ne pas concevoir un Prolog amélioré sur la machine la moins chère du monde ?

Pour spécifier ce nouveau Prolog j'ai remplacé la notion d'unification par celle de résolution d'équations dans un domaine donné. Ceci m'a permis d'introduire les arbres infinis et aussi des diséquations du type \neq : la nouvelle unification ne risquait plus de tourner en rond et il était enfin possible, sans recourir à l'opération de coupure « slash », de tester que deux objets étaient différents [5, 23].

Tout ce travail donnera naissance à Prolog II [7] qui tournera effectivement sur un Apple II avec une mémoire virtuelle sur disque souple adressée par des mots de 3 octets ! C'est ce logiciel qui nous a valu la Pomme d'Or 1982 d'Apple.

1982–1990 Prolog III et les contraintes

D'octobre 1982 à octobre 1983 j'ai été détaché au Centre Mondial d'Informatique à Paris, ce qui m'a permis de démarrer des recherches toutes nouvelles sur des extensions des mécanismes de base de Prolog. Pour définir Prolog II, qui intégrait les arbres infinis et la relation \neq , j'avais été amené à remplacer le concept d'unification par celui de résolution de contraintes dans un domaine précis muni d'opérations et de relations précises. Pourquoi ne pas poursuivre dans cette voie avec un domaine plus riche ?

C'est ainsi que naîtra Prolog III [10] qui intègre au niveau de l'algorithme d'unification : (1) une manipulation plus fine des arbres, qui peuvent être infinis, avec un traitement spécifique pour les listes, (2) un traitement complet de l'algèbre de Boole, (3) un traitement numérique en précision infinie comprenant l'addition, la multiplication par une constante et les relations $<$, \leq , \geq , $>$, (4) un traitement général des relations $=$, \neq .

Le développement de Prolog III fut une tâche difficile. (1) J'ai d'abord défini les grandes lignes du langage et des algorithmes utilisés et j'ai écrit plusieurs exemples de programmes. (2) Michel Henrion, Frédéric Benhamou, Jean Marc Boï et Touraïvane, tous des étudiants en thèse [92, 93, 96], ont alors affiné ces algorithmes et conçu les différents modules pour résoudre les contraintes de type équations et diséquations dans les différents domaines algébriques intervenant dans ce nouveau Prolog. (3) Touraïvane a alors intégré tous ces modules en un seul programme. (4) J'ai alors terminé les spécifications du langage en y ajoutant notamment le concept de multiplication et de concaténation introduit par Touraïvane.

Les tâches (2) et (3) furent effectuées en deux ans et fin 1987 nous disposons d'un premier prototype d'interpréteur Prolog III écrit en Pascal. Il faudra deux ans de plus à la compagnie PrologIA, que nous avons créée en janvier 1984, pour transformer ce prototype en un produit commercial écrit en C.

Il est à signaler que le développement de prolog III bénéficia d'un soutien financier important de la CEE et du CNET (Centre National d'Etude des Télécommunications).

1990–1996 Prolog IV et les contraintes

La notion de *programmation par contraintes* se précise. Programmer par contraintes consiste à formuler un problème en terme d'inconnues soumises à une contrainte. Cette dernière se présente sous forme d'une conjonction de contraintes élémentaires et, d'une façon plus générale, sous forme d'une formule du premier ordre faisant intervenir des opérations et des relations définies dans un domaine donné. Résoudre la contrainte, et par la même le problème posé, revient alors à trouver les valeurs qu'il faut attribuer aux variables libres de la formule pour la rendre vraie. En fait il s'agit d'une vision unifiée de la programmation logique et de la programmation mathématique (comme on la connaît en Recherche Opérationnelle).

Je travaille, d'une part, sur les algorithmes de résolutions de contraintes et, d'autre part, sur le développement de systèmes complets de programmation utilisant ces algorithmes. Le résultat final sera Prolog IV [51], sorti en juillet 96.

Ce langage se caractérise par un jeu très vaste de contraintes : plus d'une centaine de contraintes élémentaires, allant des contraintes sur les listes et les arbres aux contraintes numériques traitées en précision infinie par les algorithmes de Gauss et du type simplex, en passant par les contraintes traitées par réduction d'intervalles flottants, s'appliquant aussi bien aux réels qu'aux cas particuliers des entiers et des booléens.

Voici un exemple de résolution de contraintes qui résume

bien les capacités du langage :

$$\exists u \exists v \exists w \exists x \left[\begin{array}{l} y \leq 5 \\ \wedge v_1 = \cos v_4 \\ \wedge \text{size}(u) = 3 \\ \wedge \text{size}(v) = 10 \\ \wedge u \bullet v = v \bullet w \\ \wedge y \geq 2 + (3 \times x) \\ \wedge x = (74 > \lfloor 100 \times v_1 \rfloor) \end{array} \right]$$

devient

$$y = 5$$

Pour information, ici x, y sont des réels, u, v, w des vecteurs, v_1, v_4 la première et quatrième composante de v et $(74 > \lfloor 100 \times v_1 \rfloor)$ vaut 1 ou 0 suivant que 74 est ou n'est pas plus grand que la partie entière de $100 \times v_1$.

Plusieurs années ont été nécessaires pour aboutir aux spécifications précises de Prolog IV [50]. Les deux problèmes majeurs étaient (1) de préciser la structure mathématique π_4 très hétérogène sur laquelle était bâti le langage et (2) de caractériser la nature des différentes incomplétudes des algorithmes de résolutions de contraintes.

La structure π_4 choisie est la structure d'arbre mais enrichie de 124 relations. Les seules opérations sont donc les constructeurs. Une grande partie des relations sert à exprimer des opérations numériques, en assimilant un nombre à un arbre d'un seul nœud étiqueté par ce nombre.

Pour caractériser l'incomplétude des algorithmes de résolutions j'ai dégagé les 25 propriétés mathématiques de la structure π_4 qui sont systématiquement utilisées. Ces propriétés ont été énoncées sous forme de schémas d'axiomes du premier ordre et définissent donc la théorie \mathcal{T}_4 dans laquelle on se place lorsque l'on résout des contraintes Prolog IV.

Pour concevoir Prolog III et Prolog IV j'ai étudié et intégré toutes sortes d'algorithmes de résolution de contraintes : algorithme de Gauss, simplex, réduction et propagation d'intervalles, résolution d'équations et diséquations sur les arbres etc. Après la sortie de Prolog IV j'ai éprouvé et j'éprouve toujours le besoin d'étudier à fond et séparément quelques algorithmes ponctuels de résolution de contraintes.

1996–2000 Résolution de contraintes par réduction d'intervalles

Essentiellement on s'intéresse à résoudre des conjonctions de contraintes atomiques en résolvant itérativement des sous-contraintes élémentaires qui se chevauchent et ce jusqu'à aboutir à un point fixe. Chaque sous-contrainte élémentaire est formée d'une contrainte atomique centrale et d'un ensemble de contraintes atomiques attribuant un domaine à chaque variable figurant dans la contrainte centrale. La résolution d'une contrainte élémentaire, souvent qualifiée de *locale*, consiste à réduire au mieux ces domaines qui sont généralement des intervalles.

Problèmes combinatoires abordés J'ai terminé l'encadrement de la thèse de Ian Gambini [105] qui portait sur le découpage d'un carré en des carrés plus petits et tous de tailles différentes.

Avec Bruno Giletta nous avons travaillé sur le problème qui consiste à remplir une boîte de dimensions $6 \times 5 \times 4$ cubes élémentaires par 12 pentaminos chacun composé de 5 cubes élémentaires. C'est ici que la méthode de résolution de contraintes par réduction d'intervalles a commencé à faire ses preuves. Ce travail s'est terminé par la soutenance d'une thèse à la fin de l'année 2000 [107].

Contrainte de tri Dans un domaine totalement ordonné comme celui des réels ou des entiers, je m'intéresse à résoudre des contraintes de la forme

$$(x_1, \dots, x_n) \in r_n \wedge x_1 \in a_1 \wedge \dots \wedge x_n \in a_n, \quad (1)$$

où les a_i sont des intervalles et r_n une relation simple mais définie pour des n aussi grands que l'on veut. Comme exemple de relation r_n on peut prendre l'ensemble des n -uplets d'entiers distincts ou bien l'ensemble des n -uplets de la forme $(x_1, \dots, x_m, y_1, \dots, y_m)$, où le m -uplet (y_1, \dots, y_m) est obtenu par tri du m -uplet (x_1, \dots, x_m) , ou encore une propriété géométrique d'un objet représenté par les x_i .

Résoudre la contrainte (1) consiste, non seulement à déterminer si elle admet au moins une solution, mais aussi à calculer les plus petits (au sens de l'inclusion) intervalles b_i qui, substitués aux intervalles a_i , ne modifient pas l'ensemble de ses solutions. En fait il s'agit de calculer, au sens de l'inclusion, le plus petit produit cartésien d'intervalles de la forme $b_1 \times \dots \times b_n$ qui contient l'ensemble $r_n \cap a_1 \times \dots \times a_n$ de n -uplets.

L'objectif est d'isoler les bonnes contraintes globales qui peuvent être résolues par des algorithmes de complexité polynomiale en n . Avec Noëlle Bleuzen-Guernalec, nous avons développé un algorithme pour résoudre la contrainte de tri en $\mathcal{O}(n \log n)$ [12]. Pour $n = 2 \times 5$, cet algorithme permet, par exemple, d'opérer la réduction suivante du vecteurs d'intervalles $(a_1, \dots, a_5, a_6, \dots, a_{10})$ de la contrainte (1) :

$$\begin{pmatrix} [0, 13], [6, 10], [10, 11], [4, 16], [4, 6], \\ [1, 3], [5, 10], [6, 9], [11, 17], [10, 15] \end{pmatrix} \\ \downarrow \\ \begin{pmatrix} [1, 3], [6, 9], [11, 11], [11, 15], [5, 6], \\ [1, 3], [5, 6], [6, 9], [11, 11], [11, 15] \end{pmatrix}$$

J'ai réussi à résoudre cette contrainte, dans des temps raisonnables, sur des exemples tirés au hasard, avec $2n$ allant jusqu'à 10 000 000.

Une retombée intéressante de cet algorithme est de permettre de résoudre la contrainte être n entiers tous différents en $\mathcal{O}(n \log n)$ [53].

Espaces d'approximation En rédigeant mon cours de DEA j'ai senti la nécessité d'approfondir le concept même de résolution de contraintes par réduction d'intervalles. J'ai donc passé du temps à cerner ses aspects topologiques en introduisant et étudiant le concept d'espace d'approximation. J'ai notamment défini la notion de bonne relation n -aire et montré, à peu de choses près, que r est une bonne relation n -aire si et seulement si chacune des relations binaires qu'on peut extraire de r est une bonne relation binaire. J'ai présenté mes premiers résultats lors de trois conférences invitées [36] dont une à « l'International Conference on Logic Programming », sur l'île de Chypre, où j'ai eu la surprise et le plaisir de recevoir un numéro spécial de la revue *Theory*

and Practice of Logic programming, dédié à mon 60ième anniversaire [13].

2000–2004 Théories complètes du 1er ordre

Contraintes, du premier ordre complet, sur les arbres

Dans les langages de programmation classiques on rencontre la notion de donnée composée pour représenter des objets complexes variés, comme des suites, des suites de suites, des vecteurs, des matrices etc. Essentiellement une donnée composée est, soit une donnée simple comme un nombre réel ou un identificateur, soit une suite finie de pointeurs vers des objets composés. Une telle donnée peut être vue comme un arbre dont les nœuds sans fils sont étiquetés par les données simples et les autres nœuds par quelques identificateurs.

Il est donc vital de savoir résoudre des contraintes générales dans la théorie des arbres. Par contraintes générales, nous entendons des formules complètes du premier ordre, donc construites avec les quantificateurs \exists, \forall , les constantes logiques *vrai, faux* les connecteurs \neg, \wedge, \vee , la relation $=$ et des termes eux mêmes construits avec des variables et des constructeurs.

En prolongeant les algorithmes de résolution de Prolog II, Bich-Han Dao-Thi dont j'ai supervisé la thèse, a développé un algorithme qui simplifie une formule du premier ordre en une formule équivalente dans la théorie des arbres infinis. Cette formule simplifiée est soit la constante logique *vrai*, soit la constante *faux*, soit une formule vraie pour certaines valeurs des variables libres et fausses pour d'autres.

Thi Bich Hanh a conçu un algorithme exprimé en 11 règles de réécriture qui a fonctionné sur des exemples de contraintes faisant intervenir jusqu'à 100 quantifications alternées $\exists, \forall, \exists, \dots$. Ce travail s'est terminé par une soutenance de thèse fin 2000 [106] et un article dans une revue internationale [14].

Nous nous intéressons maintenant à la complexité de cet algorithme suivant les différentes formes de la formule p traitée. Rappelons que dans le pire des cas cette complexité est considérable puisqu'elle s'écrit sous forme d'une tour d'exponentielles dont la hauteur dépend linéairement de la taille de p .

Contraintes, du premier ordre complet, sur les arbres et les rationnels ordonnés Par la suite, avec Khalil Djelloul dans le cadre d'une thèse, nous avons amalgamé deux de ces théories, les arbres et les rationnels additifs ordonnés, en une seule théorie complète [108].

Théories complètes du 1er ordre J'ai commencé à établir un catalogue des théories complètes du 1er ordre les plus intéressantes pour l'informatique. J'ai présenté ce catalogue dans deux conférences invitées, [34, 35]. J'ai aussi organisé un colloque intitulé « Solving first-order constraints in various structures » au Centre International des Rencontres de Mathématiques à Marseille, début Mai 2001. Ce colloque portait sur trois thèmes : présentation d'algorithmes de résolutions de contraintes du 1er ordre, axiomatisation au premier ordre de structures intéressantes, étude du pouvoir d'expression de contraintes du 1er ordre et de la complexité

de leur résolution. Ont notamment participé à ce colloque : Volker Weispfenning, Universität Passau, Andrei Voronkov, The University of Manchester Hoon Hong, North Carolina State University, Jean-Pierre Jouannaud, Université Paris-sud.

2004–2006 Complexité des programmes universels

Chaque année, dans le cadre des travaux pratiques de mon cours de 1er cycle sur les machines universelles, je fais programmer et tester une machine de Turing universelle. J'ai d'ailleurs programmé un ensemble de logiciels à cet effet. En fait les étudiants ne programment pas une machine complète, ce serait une tâche trop difficile et fastidieuse, ils réécrivent un module ou deux d'une machine complète que j'ai préalablement testé et documenté. Ma première machine était très lente et il n'était absolument pas envisageable de la faire tourner sur elle-même. Au fil des années j'ai conçu des machines de plus en plus rapides et je dispose maintenant d'une machine très efficace pouvant tourner sur elle-même.

Ce travail m'a amené à définir et étudier la complexité d'un programme universel. Les résultats obtenus ont été présentés dans deux conférences invitées ainsi qu'au Colloque annuel de l'Institut Universitaire de France sur le thème de la complexité en mars 2003 [38, 39, 28].

Logiciels développés

- Analyseur et synthétiseur de langages définis par une W-grammaire ;
- Les systèmes-Q ;
- Prolog I (avec Ph. Roussel) ;
- Prolog II (avec M. Van Caneghem et H. Kanoui) ;
- Prolog III (avec plusieurs thésards et surtout Touraïvane) ;
- Prolog IV (avec PrologIA et surtout Touraïvane).

Autres activités

Etant le premier professeur d'informatique nommé à Marseille, j'ai développé cette discipline « ex nihilo » :

- en me préoccupant constamment de disposer de moyens de calcul ;
- en formant les collègues plus jeunes ;
- en dirigeant de nombreuses thèses ;
- en créant en 1973 le Groupe d'Intelligence Artificielle (GIA), URA CNRS 816, à la Faculté des Sciences de Luminy ;
- en participant activement à la création en 1994 du Laboratoire d'Informatique de Marseille (LIM), une URA CNRS associant deux universités : l'Université d'Aix-Marseille I et l'Université d'Aix-Marseille II ; ce laboratoire comprend maintenant plus de cent personnes (avec les doctorants) et regroupe l'essentiel des informaticiens de Marseille ;
- en créant dès 1975 un DEA d'Informatique qui a pris de l'ampleur et rassemble maintenant 5 Universités :

Avignon, Aix-Marseille I, Aix-Marseille II, Toulon et la Réunion ;

- en participant à la création de nombreux enseignements d'Informatique (licence, maîtrise, 2 DESS, département d'Informatique de l'école d'ingénieurs ESIL de Luminy) ;
- en passant de nombreux contrats avec l'industrie et des organismes de recherches ; entre autres, j'ai obtenu trois contrats ESPRIT dont deux en tant que contractant principal ;
- en créant en 1984 la SARL PrologIA (avec cinq de mes collègues). En plus de la diffusion de Prolog II, Prolog III et maintenant de Prolog IV, cette société a des activités de service, notamment dans le domaine bancaire et la planification pour des compagnies d'aviation.

Articles dans des revues

Références

- [1] Alain Colmerauer. Notions d'opérateurs dans une grammaire "context-free", *RIRO*, no 2, 1967.
- [2] Alain Colmerauer. Total Precedence Relations, *Journal of the ACM*, janvier 1970.
- [3] Alain Colmerauer. Un sous-ensemble intéressant du français, dans *RAIRO Informatique Théorique* 13, no 14, 1979.
- [4] Alain Colmerauer. Sur les bases théoriques de Prolog, dans *Groupe Programmation et Langages AFCET*, division théorie et technique de l'informatique, no 9, 1979.
- [5] Alain Colmerauer, Henry Kanoui et Michel Van Caneghem. Prolog, Bases théoriques et développements actuels, dans *TSI*, vol. 2, no 4 (AFCET-Bordas), août 1983.
- [6] Alain Colmerauer et Jean-François Pique. About natural logic, dans *Logique et Analyse*, Nauwelaerts Printing S.A., 101-111, septembre 1985. Note : il s'agit d'une republication de [16].
- [7] Alain Colmerauer. Prolog in 10 Figures, *Communications of the ACM*, vol. 28, num. 12, décembre 1985.
- [8] Alain Colmerauer. Opening the Prolog III Universe, revue *Byte*, août 1987.
- [9] Alain Colmerauer. Une Introduction à Prolog III, *Annales des Télécommunications*, 44, numéro 5-6, 1989.
- [10] Alain Colmerauer. An Introduction to Prolog III, *Communications of the ACM*, 33(7) : 68–90, 1990.
- [11] Alain Colmerauer. Les systèmes-Q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur. *Traitement Automatique des Langues*, Revue semestrielle de l'Association pour le Traitement Automatique des Langues, volume 33, Numéro 1–2, p 105–148, 1992. Il s'agit d'un rapport technique ancien de l'Université de Montréal qui n'avait jamais été publié.
- [12] Noëlle Bleuzen-Guernalec et Alain Colmerauer. Optimal narrowing of a block of sortings in optimal time, dans *Constraints*, Kluwer Academic publishers. janvier, pp.85–118.

- [13] Special issue in honor of Alain Colmerauer, on the occasion of his 60th birthday, *Theory and Practice of Logic programming*, volume 1, part 6, novembre 2001.
- [14] Alain Colmerauer et Thi-Bich-Hanh Dao. Expressiveness of Full First-Order constraints in the Algebra of Finite or Infinite Trees, dans *Constraints*, Kluwer Academic publishers. 2003, volume 8, issue 3.
- [26] Noëlle Bleuzen-Guernalec et Alain Colmerauer. Narrowing a block of sortings in $\mathcal{O}n \log n$. Dans les actes du congrès CP97, Linz, Autriche, octobre 97, *Principles and Practice of Constraint Programming, CP97*. Lecture Notes in Computer Science, no 1330, pages 2–16. Springer Verlag 1997.
- [27] Alain Colmerauer and Bruno Giletta. Solving the three-dimensional pentamino puzzle, 2000.
- [28] Alain Colmerauer. On the complexity of universal programs. In *itsshape Machine, Computations and Universality* (Saint-Petersburg 2004). Lecture Notes in Computer Science, pp 18–35, 2005.

Livres et articles dans des livres

Références

- [15] Alain Colmerauer. Metamorphosis grammars, dans *Natural Language Communication with Computers*, Lectures Notes in Computer Science n 63, édité par L. Bolc, Springer Verlag, 1978.
- [16] Alain Colmerauer et Jean-François Pique. About natural logic, dans *Advances in Data Base Theory 1*, édité par H. Gallaire, H. Minker et J.M. Nicolas, Plenum Press, 1981.
- [17] Alain Colmerauer. An interesting subset of natural language, dans *Logic Programming*, édité par K. L. Clark et J. A. Tarnlund, Academic Press, 1982.
- [18] Alain Colmerauer. Prolog and infinite trees, dans *Logic Programming*, édité par K. L. Clark et J. A. Tarnlund, Academic Press, 1982.
- [19] Frédéric Benhamou and Alain Colmerauer, editors. *Constraint Logic programming, Selected Research*. MIT Press, 1993.
- [20] Alain Colmerauer. Naive solving of non-linear constraints, In Frédéric Benhamou and Alain Colmerauer, editors, *Logic Constraint Programming, Selected Research*, MIT Press, Cambridge, USA, 1993.
- [21] Alain Colmerauer et Philippe Roussel. The birth of Prolog, dans *History of Programming Languages*, édité par Thomas J. Bergin et Richard G. Gibson, ACM Press/Addison-Wesley, 1996.

Conférences avec actes

Références

- [22] Alain Colmerauer et Guy de Chastellier. W-Grammars, congrès ACM, San Francisco, août 1970.
- [23] Alain Colmerauer. Equations and Inequations on Finite and Infinite Trees, papier présenté en tant que “invited lecture”, dans *Proceedings of the International Conference on Fifth Generation Computer Systems*, Tokyo, novembre 1984.
- [24] Alain Colmerauer. An Introduction to Prolog III, *Proceedings of the annual Esprit conference*, Bruxelles, septembre 1987.
- [25] Alain Colmerauer and Philippe Roussel. The birth of Prolog. In *History of Programming Languages Conference*, ACM SIGPLAN, Cambridge USA, avril 1993.

Conférences invitées

Références

- [29] Alain Colmerauer. Naive concatenation. Invited talk, Workshop on Principles and Practice of Constraint Programming, Newport, Rhode Island, USA, avril 1993.
- [30] Alain Colmerauer. A legal framework for discussing approximate solving of constraints. In *Proceedings of INTERVAL 94*, St. Petersburg, Russia, mars 1994. (présentation avec résumé dans des actes informels).
- [31] Alain Colmerauer. Résolution approchée de contraintes par produits cartésiens de sous-ensembles privilégiés. In *Ecole de Printemps d’Informatique Théorique, Châtillon sur Seine*, mai 1994. Conférencier invité, pas d’actes.
- [32] Alain Colmerauer. Toward an ideal unified constraint domain, Conférencier invité, exposé purement oral, (Pact 96, avril 96, Londres) *Practical Application of Constraint Technology*, Actes publiés par The Practical Application Company Ltd, ISBN 0 9525554 25.
- [33] Alain Colmerauer. Mixing constraints domain, the Prolog IV case, Conférencier invité, exposé purement oral, *Frontiers of Combining Systems, First International Workshop*, Munich, mars 1996, Actes édités par Franz Bader et Klaus U. Schultz, Applied Logic Series 3, Kluwer Academic publishers.
- [34] Colmerauer Alain, Négations et quantifications dans les contraintes, *Journées Francophones de Programmation en Logique et par contraintes (JFPLC 2000)*, conférence invitée, Marseille, juin 2000.
- [35] Colmerauer Alain, *Full first order constraints in various structure*, Conférence invitée, Institute of Software, Chinese Academy of Sciences, Pékin, septembre 2000.
- [36] Colmerauer Alain, Solving the multiplication constraint in several approximation spaces, conférence invitée, *Seventeenth International Conference on Logic Programming ICLP-2001*, Paphos, septembre 2001 (et aussi à l’Université de Passau sur l’invitation de Volker Weispfenning, novembre 2002).
- [37] Colmerauer Alain, Cinq années de recherche dans l’équipe de Louis Bolliet, 50 ans d’Informatique à Grenoble, Grenoble, novembre 2002.
- [38] Colmerauer Alain, Complexité d’un programme universel, conférence donnée au *Colloque annuel de l’Institut Universitaire de France sur le thème de la complexité*,

Ecole normale supérieure de Lyon, mars 2003. Aussi conférence invitée, *Séminaire du Laboratoire d'Informatique Fondamentale d'Orléans*, Université d'Orléans, octobre 2003.

- [39] Colmerauer Alain, Complexity of Universals Programs, Conférence invitée, Jornadas de Programación y Lenguajes, Alicante novembre 2003.

Publications internes

Références

- [40] Alain Colmerauer. Précédences, analyse syntaxique et langages de programmation. Thèse d'Etat. Université de Grenoble, septembre 1967.
- [41] Alain Colmerauer. *Les systèmes- q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur*. Publication interne n° 43, Département d'Informatique de l'Université de Montréal, septembre 1970.
- [42] Alain Colmerauer, Jule Dansereau, Brian Harris, Richard Kittredge, Gilles Steward et Michel Van Caneghem. *TAUM 71, rapport annuel du projet de Traduction Automatique de l'Université de Montréal*, janvier 1971.
- [43] Alain Colmerauer, Henry Kanoui, Robert Pasero et Philippe Roussel. *Un système de communication en français*. Rapport préliminaire, Groupe de Recherche en Intelligence Artificielle, Université II d'Aix-Marseille, octobre 1972.
- [44] Alain Colmerauer. *Final Specifications for Prolog III*. Projet Esprit P1219 (1106), Further Development of Prolog and its Validation by KBS in Technical Areas. Milestone II, février 1988.
- [45] Alain Colmerauer. *Une introduction à Prolog III*. Publication interne, LIM, 2000. Note : version française de [10].
- [46] Alain Colmerauer. *Résolution naïve de contraintes non linéaires*. Publication interne, LIM, 1992. Note : version française de [20].
- [47] Alain Colmerauer. *Prolog IV Specifications*, Esprit project 5246, Reference Manual WP7/R25,TRD Prince, janvier 1995.
- [48] Alain Colmerauer. *Combining constraint domains*. Esprit project 7195, Deliverable D2.11/3, Research Action Acclaim, juin 1995.
- [49] Alain Colmerauer et Philippe Roussel, *La naissance de Prolog*. Publication interne, LIM, 1996. Note : version française de [21].
- [50] A. Colmerauer. *Les bases de Prolog IV*. Publication interne, LIM, 1996.
- [51] F. Benhamou, P. Bouvier, A. Colmerauer, H. Garetta, B. Giletta, J.L. Massat, G.A. Narboni, S. N'Dong, R. Pasero, J.F. Pique, Touraïvane, M. Van Caneghem et E. Vétillard. *Le manuel de Prolog IV*. PrologIA, Marseille, juin 1996.
- [52] Noëlle Bleuzen-Guernalec et Alain Colmerauer. *Réduction d'un pavé de tris en $O(n \log n)$* . Publication interne, LIM, 1997. Note : version française de [26].

[53] Noëlle Bleuzen-Guernalec et Alain Colmerauer. *Narrowing a $2n$ -block of sortings in $O(n \log n)$, application to the "all different" relation*. Publication interne, LIM, 1998.

[54] Noëlle Bleuzen-Guernalec et Alain Colmerauer. *Réduction optimale d'un pavé de tris en temps optimal*. Publication interne, LIM, 1999. Note : version française de [12].

[55] Alain Colmerauer et Thi-Bich-Hanh Dao. *itshape* Pouvoir d'expression des contraintes du 1ier ordre dans l'algèbre des arbres finis ou infinis. Publication interne, LIF, 2003. Note : version française de [14].

[56] Colmerauer Alain, *Résolution de contraintes par réduction d'intervalles*, Notes de cours : DEA d'Informatique et DESS de Mathématiques et d'Informatique des Nouvelles Technologies, version 2004, Université de la Méditerranée.

[57] Alain Colmerauer. *Sur la complexité de programmes universelles*. Publication interne, LIF, 2005. Note : version française de [28].

Thèses encadrées, toutes à l'Université d'Aix-Marseille II

Références

- [58] Ph. Roussel. Définition et traitement de l'égalité formelle en démonstration automatique, thèse de 3ème cycle, mai 1972.
- [59] R. Pasero. Représentation du français en logique du premier ordre en vue de dialoguer avec un ordinateur, thèse de 3ème cycle, juin 1973
- [60] H. Kanoui. Application de la démonstration automatique aux manipulations algébriques et à l'intégration formelle sur ordinateur, thèse de 3ème cycle, octobre 1973.
- [61] M. Bergman. Résolution par la démonstration automatique de quelques problèmes en intégration symbolique sur calculateur, thèse de 3ème cycle, octobre 1973.
- [62] M. Joubert. Un système de résolution de problèmes à tendance naturelle, thèse de 3ème cycle, février 1974.
- [63] J. Gispert. Etude de l'optimisation et réalisation d'un éditeur de textes paginés, thèse de 3ème cycle, juin 1975.
- [64] G. Battani. Mise en œuvre des contraintes phonologiques, syntaxiques et sémantiques dans un système de compréhension automatique de la parole, thèse de 3ème cycle, juin 1975.
- [65] H. Meloni. Mise en œuvre des contraintes phonologiques, syntaxiques et sémantiques dans un système de compréhension automatique de la parole, thèse de 3ème cycle, juin 1975.
- [66] J. Guizol. Synthèse du français à partir d'une représentation en logique du premier ordre, thèse de 3ème cycle, octobre 1975.
- [67] V. Dahl. Un système déductif d'interrogation de banques de données en espagnol, thèse de 3ème cycle, novembre 1977.

- [68] M. Rodriguez. Un système patient d'Aide à la conception JOB, thèse de 3ème cycle, octobre 1978. (En grande partie dirigée par Ph. Roussel.)
- [69] Ph. Donz. Une méthode de transformation et d'optimisation de programmes Prolog. définition et implémentation, thèse de 3ème cycle, juin 1979.
- [70] P. Sabatier. Dialogues en Français avec un ordinateur, thèse de 3ème cycle, juin 1980. (En grande partie dirigée par R. Pasero.)
- [71] Ch. Giraud. Logique et conception assistée par ordinateur, thèse de 3ème cycle, mai 1980.
- [72] L. Périchaud. Consultation en français d'une banque de données sur fichier et mise en place du système Prolog nécessaire, thèse de 3ème cycle, avril 1981.
- [73] J. F. Pique. Sur un modèle logique du langage naturel et son utilisation pour l'interrogation des banques de données, thèse de 3ème cycle, décembre 1981.
- [74] P. Siegel. La saturation au secours de la Non-Monotonie, thèse de 3ème cycle, mai 1981.
- [75] G. Bossu. La saturation au secours de la Non-Monotonie, thèse de 3ème cycle, mai 1981.
- [76] H. Méloni. Etude et réalisation d'un système de reconnaissance automatique de la parole continue, thèse d'Etat, février 1982.
- [77] J. P. Parcy. Un système expert en diagnostic sur réacteurs à neutrons rapides, thèse de 3ème cycle, septembre 1982.
- [78] M. Hileyan. Modélisation des fins de parties d'échecs, thèse de 3ème cycle, septembre 1982.
- [79] M. Van Caneghem. L'Anatomie de Prolog II, thèse d'Etat, octobre 1984.
- [80] H. Garetta. Un compilateur Modula II écrit en Prolog, thèse de doctorat, juin 1985.
- [81] S. Himbault. Un système expert en diagnostic de panne pour un réacteur nucléaire à neutrons rapides, thèse de doctorat, octobre 1985.
- [82] C. Sedogbo. De la grammaire en chaîne du français à un système de questions-réponses, thèse d'Etat, juin 1987.
- [83] P. Siegel. Représentation et utilisation de la connaissance en calcul propositionnel, thèse d'état, juillet 1987.
- [84] R. Picca. Implantation de Concurrent Prolog, thèse de doctorat, décembre 1987.
- [85] H. Bellone. Concurrent Prolog : étude et utilisation, thèse de doctorat, décembre 1987.
- [86] A. Vergara-Bracquemond. Exther, un système de diagnostic en échanges thermiques convectifs, thèse de doctorat, janvier 1988.
- [87] C. Sabatier. Acquisition et interrogation de connaissances en langue naturelle, thèse de doctorat, avril 1988.
- [88] S. Grandcolas, Résolution d'équations sur les arbres et les listes, thèse de doctorat, mai 1989.
- [89] L. Hénocque, Un système logique pour le traitement du discours, thèse de doctorat, mai 1989.
- [90] J.L. Imbert, Simplification des systèmes de contraintes numériques linéaires, thèse de doctorat, mai 1989.
- [91] M. Rolbert, Résolution de formes pronominales dans l'interface d'interrogation d'une base de données, thèse de doctorat, mai 1989.
- [92] F. Benhamou, Le traitement des contraintes booléennes dans Prolog III, thèse de doctorat, novembre 1988.
- [93] J.M. Boi, Le traitement des contraintes booléennes dans Prolog III, thèse de doctorat, novembre 1988.
- [94] S. Coupet, Deux arguments pour les arbres infinis en Prolog, thèse de doctorat, novembre 1988.
- [95] L. Oxusoff, Evaluation sémantique en calcul propositionnel, thèse de doctorat, janvier 1989.
- [96] Touraivane, La récupération de mémoire dans les machines non déterministes, thèse de doctorat, novembre 1988.
- [97] J.J. Zotian. Prolog en informatique de gestion, thèse de doctorat, mai 1988.
- [98] Alfonso San Miguel Aguire. How to use symmetries in boolean constraint solving, thèse de doctorat, juin 1992.
- [99] Jean Luc Massat. Algorithmes énumératifs et résolutions de contraintes booléennes dans un langage de programmation logique avec contraintes, thèse de doctorat, mars 1993.
- [100] Jean-Louis Imbert. Les contraintes linéaires sur les nombres réels dans la cadre de la programmation en logique avec contraintes, habilitation, février 1993.
- [101] Olivier Bartheys. Calcul de plans d'actions : des méthodes déductives vers les méthodes algébriques, thèse de doctorat, 1994.
- [102] Jaam Jihad. Une étude sur les nombres de Ramsey classiques et multiples, binaires et ternaires, thèse de doctorat, février 1994.
- [103] Jianyang Zhou. Calcul de plus petits produits cartésien d'intervalles, application au problème d'ordonnement d'atelier, thèse de doctorat, mars 1997.
- [104] Christophe Aillaud. Résolution de contraintes par analyse de parties convexes dans \mathbf{R} , thèse de doctorat, juillet 1997.
- [105] Ian Gambini. Quant aux carrés carrelés, thèse de doctorat, 1999.
- [106] Thi Bich Hanh Dao. Résolution de contraintes du premier ordre dans la théorie des arbres finis ou infinis, thèse de doctorat, 2000.
- [107] Bruno Giletta. Placements des pentaminos par résolution de contraintes, thèse de doctorat, 2000.
- [108] Khalil Djelloul, Théories complètes autour des arbres, (Complete theories around trees), thèse de doctorat, juin 2006.