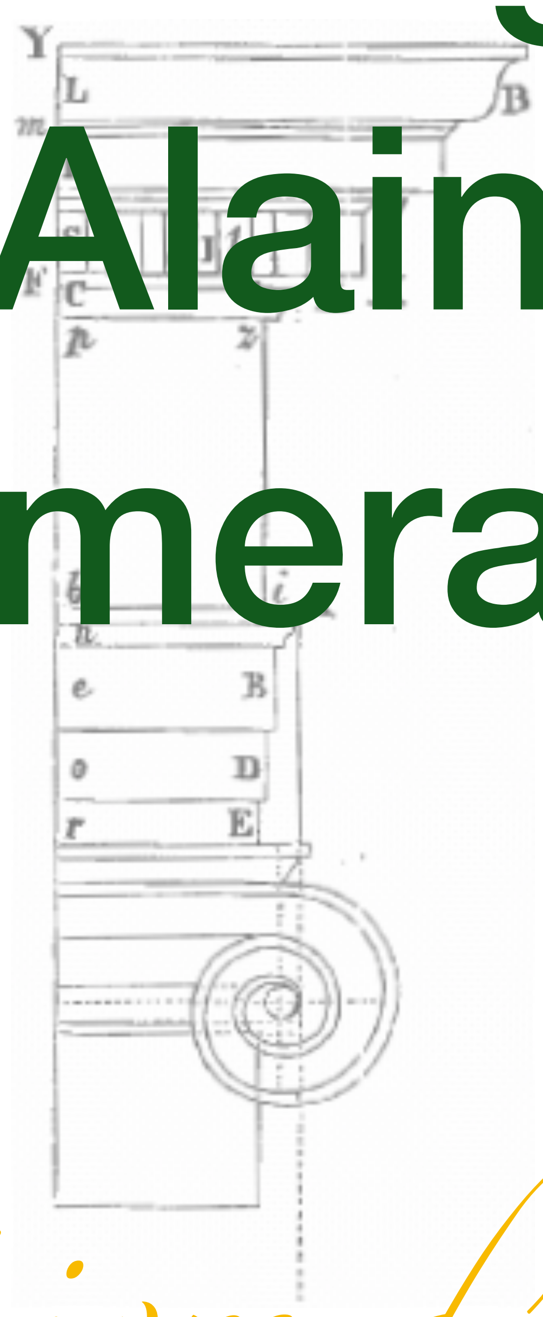
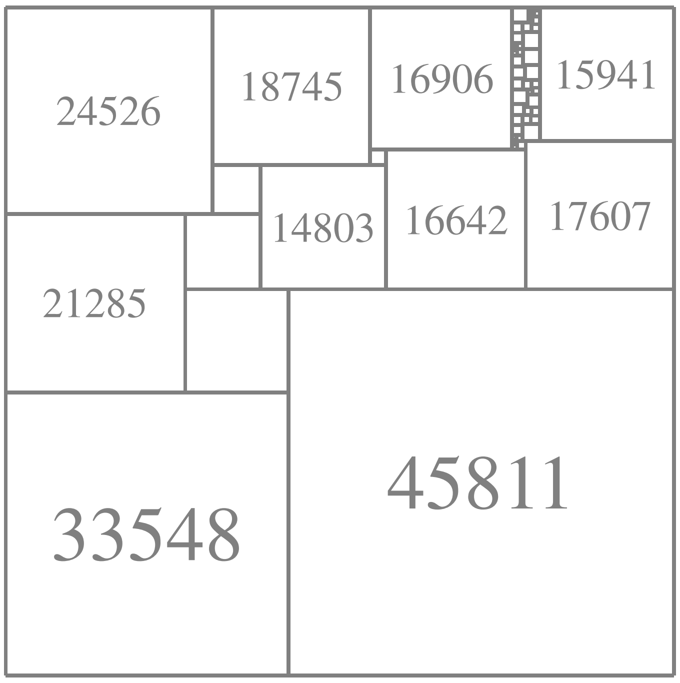
Association Prolog Heritage

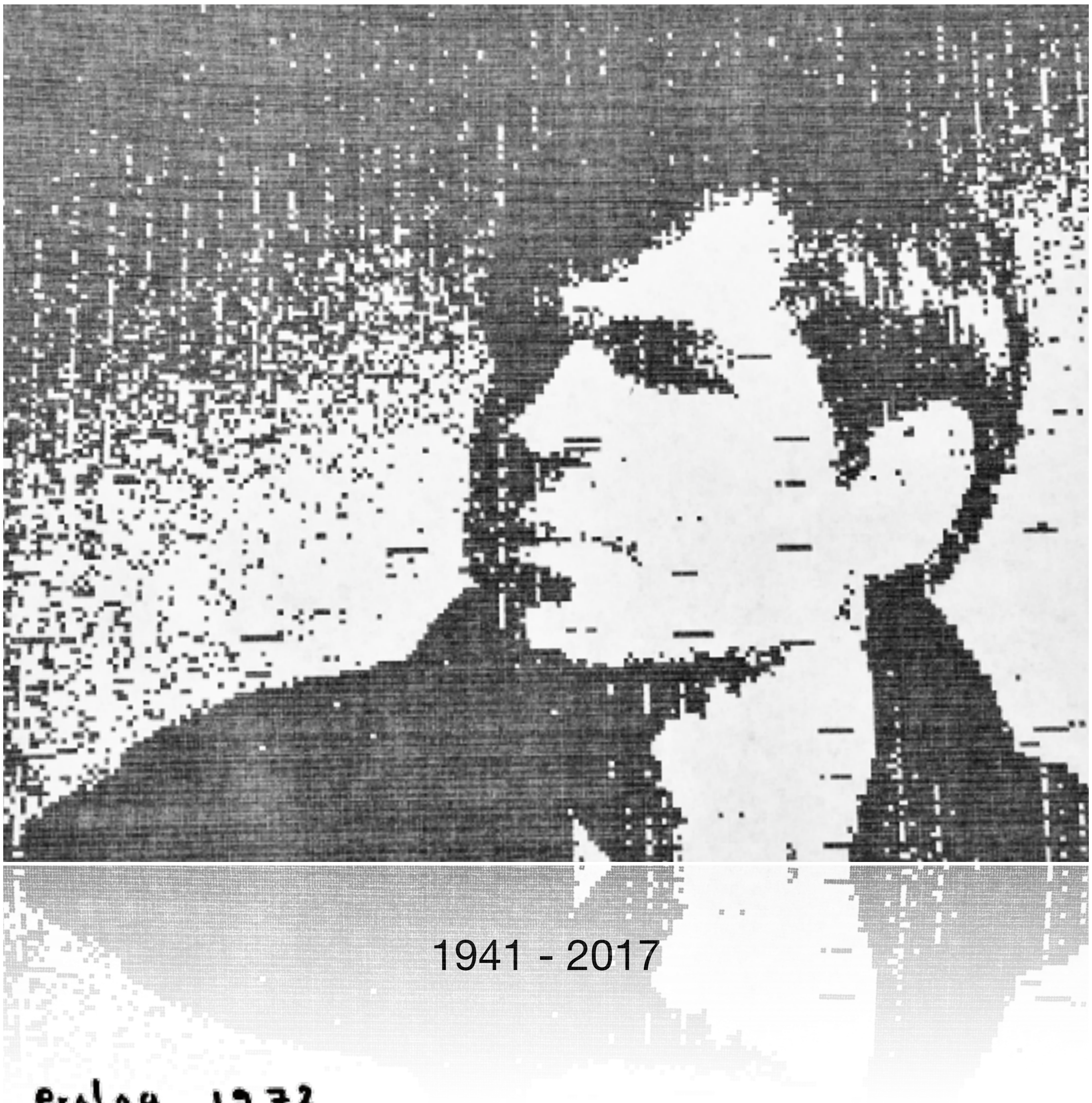
Hommage à Alain Colmerauer



Livre d'or

Edition 2020


24526
18745
16906
15941
14803
16642
17607
21285
33548
45811



1941 - 2017

Prolog 1972

```
factorielle (0,1) ⇐

factorielle (n,x) ⇐ plus(1,m,n) et factorielle (m,y) et fois (n,y,x)
```

Introduction

Vous trouverez ici les différents articles, messages personnels, anecdotes et souvenirs pris sur le vif de tous ceux qui ont bien voulu les partager dans notre livre d’or.

You will find here all the articles, private messages, stories and vivid memories shared by our visitors.

Contributeurs

Contributors

Pierre Lescanne

Bob Kowalski

François Fages

Veronica Dahl

Frédéric Gardi

Philippe Jorrand

Frédéric Benhamou

Jacques Cohen

François Bry

Janusz S. Bień

Mehmet Dincbas

Henry Kanoui

Brian Harris

Denis Lugiez

Pascal Van Hentenryck

Pierre Lescanne

À Alain Colmerauer

La première fois où j'ai entendu parler de Prolog, ça n'était pas par Alain Colmerauer, mais par Bob Kowalski un chercheur britannique d'origine étasunienne, au nom bien peu anglais [1] et ça n'était pas en France mais aux Pays-Bas en 1974, à une époque où les chercheurs ne voyageaient pas autant que maintenant [2]. En fait, ce fut lors de mon premier séjour scientifique à l'étranger. Avec Jean-Pierre Finance et Jean-Luc Rémy, nous avions assisté, à Amsterdam, à la première des écoles de printemps en informatique fondamentale, organisée au Mathematisch Centrum par Jaco de Bakker, école où étaient intervenus, Robin Milner, Erwin Engeler, Jaco de Bakker, lui-même, Erich Neuhold, Eugene Lawler et bien sûr Bob Kowalski. Pour l'anecdote, je me rappelle très bien que deux étudiants hollandais posaient plus de questions que les autres après les cours et s'appelaient Peter van Emde Boas et Willem-Paul de Roever. Ils nous impressionnaient et nous énervaient un peu parce qu'ils semblaient avoir tout compris et montraient qu'ils maîtrisaient parfaitement bien l'anglais quand ils posaient leurs questions. Nous avions sympathisé avec un étudiant, qui venait d'arriver tout juste de Pologne et qui collaborait à l'époque avec de Bakker [1] sur la sémantique des langages de programmation, c'était Krzysztof Apt. Il deviendra plus tard un des spécialistes de la programmation logique. Le soir, sous l'impulsion énergique de Jean-Luc Rémy, bien qu'épuisés par l'attention soutenue que réclamait l'écoute d'une langue qui nous était totalement étrangère, nous travaillions d'arrache-pied pour rattraper, grâce aux notes de cours, ce que nous n'avions pas compris à l'oral. Le cours que nous avons le plus travaillé, avait heureusement aussi le meilleur support écrit, en fait le premier document jamais rédigé sur la programmation logique, le fameux *Logic for problem solving* [3], cité dans [4]. Il avait le contenu le plus inattendu ; ce cours remarquable était celui de Kowalski et ce qu'il racontait était fascinant, pour nous les Nancéiens qui, autour de Claude Pair, tentions de fonder la programmation sur la logique. De plus, ce que Kowalski nous exposait, s'appuyait sur la réalisation d'un chercheur français dont Kowalski disait le plus grand bien. J'avais vraiment hâte de connaître ce fameux Alain Colmerauer ! C'est en particulier grâce à Kowalski que j'ai appris le concept fondamental d'*unification*, qui va jouer un rôle si important dans ma propre carrière .

[1] "Colmerauer", n'est pas quant à lui un nom à la consonance bien française quoiqu'il évoque une ville bien française ! Et c'est quelqu'un qui a passé sa jeunesse à Colmar qui le dit !

[2] Je n'avais étudié au collège et au lycée à Colmar que l'allemand, le latin et le grec, ce qui ne m'aidait pas beaucoup dans les échanges internationaux.

L'occasion de rencontrer « Colmé », m'en sera donnée en 1975, au centre de conférences des Prémontrés à Pont-à-Mousson, où l'IRIA avait organisé des rencontres sur le thème de *la logique et la programmation.* Il y avait tout le gratin de la recherche française en programmation de l'époque : Jacques Arsac, Maurice Nivat, Claude Pair, les jeunes loups du "bâtiment 8" de Rocquencourt et surtout Alain Colmerauer, que j'avais hâte d'écouter [3]. Lui, avec son allure sportive, à la Laurent Terzieff, qui détonnait parmi certains fumeurs compulsifs, venait avec une réalisation logicielle [4], fondée sur des principes logiques, ce qui était encore rare pour l'époque. Mais il n'était évidement pas question de faire une démo. Si l'on voulait voir le logiciel fonctionner, il fallait aller à Marseille où l'on entrerait son programme avec des cartes perforées. Je m'attendais à une salle qui partagerait mon enthousiasme et mon admiration et au lieu de cela, ce fut un feu roulant de critiques de la part des informaticiens qui étaient les plus versés en logique. Les objections les plus virulentes venaient de deux d'entre eux, le premier reprochant, entre autres, à la logique sous-jacente de ne pas être décidable, ce à quoi Claude Pair répondit que les mathématiciens sont parfois un peu doctrinaires [5], le second critiquant l'emploi d'une unification sans test d'occurrence et le slash qui brise la complétude. Mais devant cette avalanche, Alain Colmerauer garda son calme et, dans la prosodie que nous lui connaissions, maintint sa foi en la puissance de la programmation logique, en particulier pour l'analyse des langues naturelles. Clairement le courant ne passait pas entre les Parisiens et les Marseillais [6]. Plus tard cependant, Gilles Kahn fut un ardent avocat de Prolog qu'il voyait comme *le* langage d'implantation de la *sémantique naturelle* des langages de programmation [2]. Peut-être, est-ce qu'entre-temps, il avait rejoint le sud de la France à Sophia-Antipolis.

Du point de vue académique, je me suis un peu éloigné des thèmes du GIA [7], tout en gardant comme point commun l'unification. Cependant quand en 1982, j'ai été élu au comité national du CNRS, compte tenu de ma relative proximité thématique, tout en appartenant à une école différente, j'ai été amené à rapporter sur le GIA ; ce fut une tâche difficile, car Alain ne faisait pas en sorte que son laboratoire fût défendable du point de vue académique, ses membres ne publiaient pas dans les canaux universitaires traditionnels [8], diffusait mal leurs logiciels et lui-même ne faisait aucun lobbying, même pas le lobbying le plus éthique ; je dirais même, qu'il faisait presque le contraire. Mais bon ! Le GIA n'a pas été désassocié et a pu continuer à bénéficier d'une certaine forme d'appui du CNRS [9]. Avec une idée comme Prolog, il aurait pu prétendre à beaucoup plus. Heureux temps où l'on pouvait ne pas publier sans pour autant périr et où

[3] Il devait aussi y avoir Louis Nolin et Bernard Robinet, mais je ne m'en souviens pas.

[4] Le terme "logiciel" venait d'être officialisé par un arrêté du 12 janvier 1974.

[5] Pour nous qui subissions, à Nancy, à l'époque, la morgue des Bourbakistes, cette remarque n'était pas, à mon avis, élogieuse.

[6] *Nihil novi sub sole.* Et je ne parle pas de football, mais de science !

[7] L'acronyme du "Groupe d'intelligence artificielle" de Marseille a été porté depuis par un groupe nettement mois pacifique que celui de Colmerauer.

[8] On pourra par exemple, consulter sa propre page, sur DBLP

[9] J'ai déposé aux Archives Poincaré où les historiens pourront les consulter, les notes que j'avais prises à l'époque.

la folie comptable n'obnubilait pas les évaluateurs. Trop fier, Alain Colmerauer refusait de passer sous les fourches caudines des institutions.

En 1987, j'ai séjourné un mois à l'ICOT à Tokyo, dans la cadre du projet de cinquième génération d'ordinateurs. J'ai pu y découvrir la beauté du Japon et la gentillesse des Japonais, notamment de ses chercheurs très accueillants. Mais surtout j'ai pu voir l'aura qu'avaient Alain et Prolog qui ne correspondait pas à la façon dont ils étaient perçus en France.

J'ai revu Alain plus tard. Pour cela, j'ai dû aller à Marseille, car je crois qu'en dehors de Pont-à-Mousson, je ne l'ai presque jamais vu que là. La participation à des jurys de thèse étaient de telles occasions. Nous échangions. J'aimais l'écouter, car j'entendais les paroles d'un sage, qui analysait avec un peu d'aigreur le fonctionnement de la nomenclature et qui parfois était aussi déraisonnable dans certains de ses jugements. Je dois dire que je partageais, dans les grandes lignes, son point de vue. D'autres parleront sûrement des centres d'intérêt non académiques d'Alain, mais je comprenais clairement en l'écoutant que sa vie ne s'articulait pas seulement autour de la réussite académique, même si la sienne fut magnifique.

Pierre Lescanne
Professeur émérite
à l'École normale supérieure de Lyon

Références

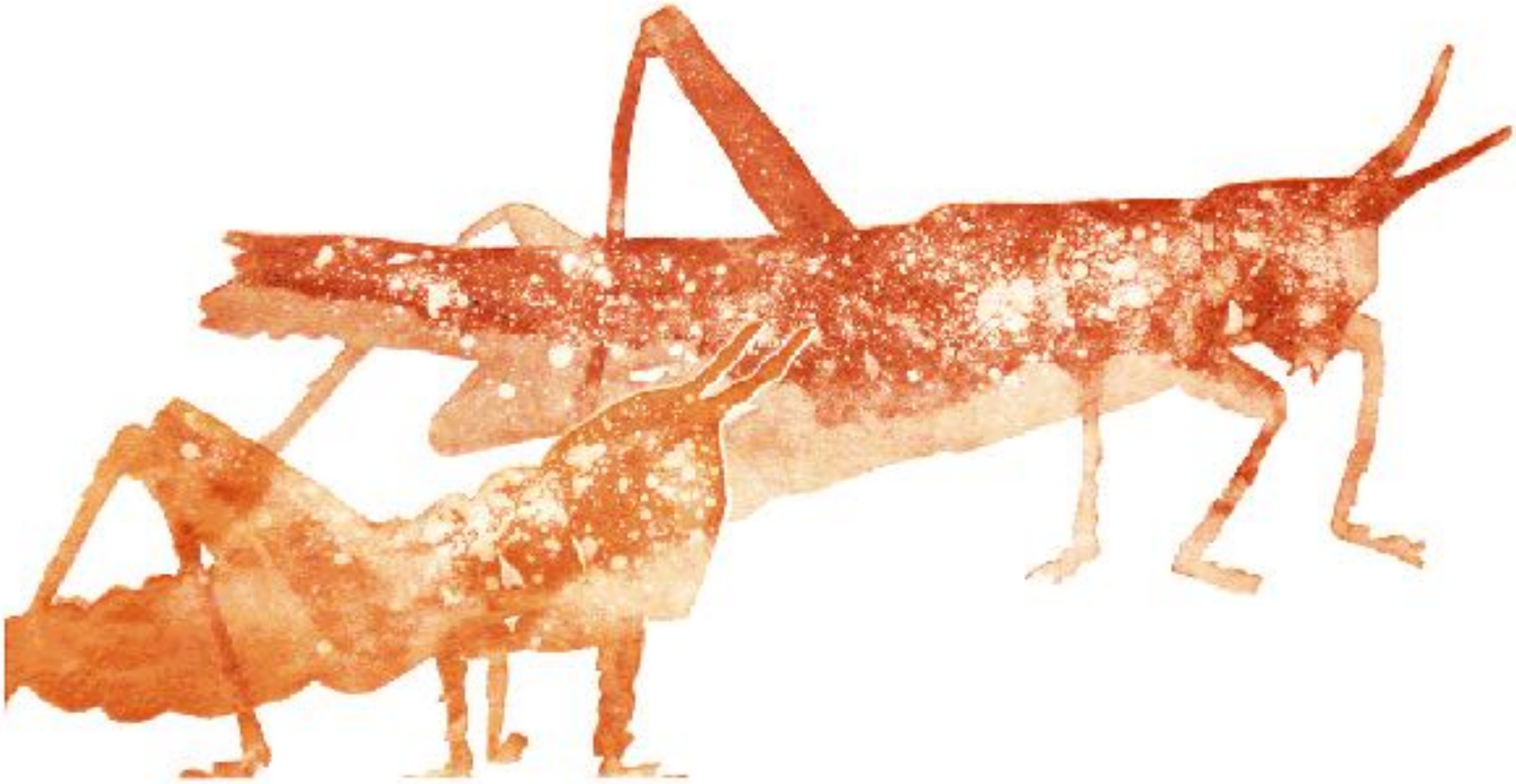
[1] Krzysztof R. Apt and J. W. de Bakker. Exercises in denotational semantics. In Antoni W. Mazurkiewicz, editor, *Mathematical Foundations of Computer Science 1976, 5th Symposium, Gdansk, Poland, September 6-10, 1976, Proceedings*, volume 45 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1976.

[2] Gilles Kahn. Natural semantics. In Franz-Josef Brandenburg, Guy Vidal-Naquet, and Martin Wirsing, editors, *STACS 87, 4th Annual Symposium on Theoretical Aspects of Computer Science, Passau, Germany, February 19-21, 1987, Proceedings*, volume 247 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 1987.

[3] Robert A. Kowalski. Logic for problem solving. School of Artifical Intelligence, Univ. of Edinburgh U. K., 1974.

[4] Robert A. Kowalski. The Early Years of Logic Programming. *Commun. ACM*, 31(1) :38–43, 1988.

François Bry



The Rationality of Codata versus the Irrationality of Infinite Terms

19 June 2020

François Bry
Institute for Informatics
Ludwig-Maximilian University of Munich, Germany

I met Alain Colmerauer between 1986 and 1988 at the European Computer-Industry Research Centre (ECRC) in Munich where I recently had joined a project on applications of logic programming to databases and automated reasoning. At ECRC, Alain Colmerauer reported on his work on rational trees. I did not understand the name “rational tree” but did not dare to publicly ask for the phrase's meaning. Indeed, I thought this meaning must be clear for most members of the audience. After the talk, I privately asked (in French) Alain Colmerauer what it means for trees to be rational and whether there also are irrational trees. He told me that the name had been chosen in reference to the fact that a real number is a rational number if and only if its decimal expansion is finite or repeating. I realized that the name "rational tree" appropriately stresses that, by Cantor’s diagonal argument, the set of all (rational and irrational) trees over a finite or denumerable alphabet is not denumerable while the set of rational trees of course is denumerable. “Exactement,” said Alain Colmerauer, “that's the reason for the name.”

Recently, I devoted some attention at coinduction which I so far had no understanding of. This led to a generalization of the coinduction proof principle. While reporting on that generalization [1], I realized that Colmerauer's rational trees had been an early, maybe even the earliest, proposal of what is now called codata. I read again the articles of the 80es of the 20th century on rational trees and related issues (like perpetual processes specified as non-terminating SLD derivations). Doing so, I noticed two puzzling facts. First, the articles on rational trees do not explain the name and do not even mention that the set of all infinite trees over a finite or denumerable alphabet is not denumerable. I remembered that such a “communication scarcity” was rather typical of Alain Colmerauer. I reflected that this trait is not unpleasant at all at a time where being loud about oneself is commonly seen as a sine qua non condition for a successful academic career. Second, Davide Sangiorgi's extensive account of the origins of coinduction, bisimulation and of related precursor concepts and methods in philosophical and mathematical logic, mathematics and computer science [2] does not mention Alain Colmerauer's rational trees, Prolog II [3], and codata in logic programming.

Nonetheless, Alain Colmerauer deserves to be remembered for the early proposal to use codata in programming and for stressing both, the rationality and denumerability of codata and the irrationality and non-denumerability of infinite terms. And Alain Colmerauer also deserves to be remembered for the pleasantness of his restraint in communicating.

References

[1] François Bry. 2020. Coinduction Plain and Simple - A Generalized Coinduction Proof Principle and its Application to Declarative Programming. Submitted for publication

[2] Davide Sangiorgi. 2009. On the origins of bisimulation and coinduction. ACM Transactions on Programming Languages and Systems, volume 31, number 4, pages 15:1–15

[3] Maarten H.van Emden and John W. Lloyd. 1984. A logical reconstruction of Prolog II. The Journal of Logic Programming, volume 1, issue 2, pages 143-149

Bob Kowalski

Alain Colmerauer left us three years ago, but he left behind an intellectual legacy that will influence future generations for years to come.

Alain arrived at the University of Aix-Marseille Luminy in 1970 as a Maître de Conferences, the equivalent of an American Associate Professor. Before arriving in Marseille, he had already achieved an international reputation for his PhD research on compliers in Grenoble, and for his machine translation work at the University of Montreal. Although he had an opportunity to join the Computer Science Department at Stanford University, he preferred instead to join and lead the new Department of Computer Science in Marseille. The decision was typical of his style: reaching out in new directions, starting from the beginning, and following through to the end.

Immediately upon his arrival in Marseille, Alain attracted a strong team of research students, to follow up the work he started on automated question-answering in Montreal. This led him to investigate Cordell Green's work on question-answering, using the resolution theorem-proving method of Alan Robinson. During the course of his investigations, he discovered my work with Donald Kuehner in Edinburgh on SL-resolution, and he sent me an invitation to visit him and his group in Marseille.

It was in the summer of 1971, when my family and I were staying with relatives in Poland, that the invitation reached me. We excitedly diverted our return journey to Edinburgh, driving in our Austin mini, with a detour via Marseille.

Alain and his wife, Colette, generously invited us to stay with them in their small apartment. When my family and I weren't sleeping on the floor of their living room, or partaking of Colette's generous hospitality, Alain and I talked excitedly, exchanging ideas about theorem-proving and parsing in natural and formal languages.

Before arriving in Marseille, Pat Hayes and I had been working in Edinburgh on logic-based automated theorem-proving using variants of the resolution method. But starting around 1969, resolution and "declarative" representations of knowledge were being attacked in MIT by advocates of "procedural" representations of knowledge. I was convinced that the goal-oriented approach of SL-resolution could achieve similar behaviour as procedural approaches, but I was struggling to understand, for example, how it could compete with procedural approaches to parsing and natural language understanding.

I thought I knew a lot about resolution theorem-proving. But Alain knew all there was to know about parsing formal and natural languages. In our excited exchange of ideas, we discovered amazing parallels between resolution theorem-proving and parsing: Both employ declarative representations of knowledge. Resolution employs logic, while parsers employ grammars. Both can solve problems in a forward or backward direction - resolution by reasoning forwards or backwards; and parsers by chaining forwards, bottom-up, or chaining backwards, top-down. Even more amazing, formal grammars can be represented in formal logic, and then hyper-resolution behaves as a bottom-up parser/generator, and SL-resolution behaves as a top-down parser/generator.

Our exchange of ideas lasted four days and much of four nights. It was not only a meeting of minds, but a bonding off spirits and the beginning of a lasting friendship. We were both born in 1941; but I was about three years behind him in my career, having completed my PhD in 1970, three years after he completed his in 1967. We both were married with three daughters, and over the years our friendship grew to include exchange visits between our families.

Alain invited me back to Marseille for a longer visit of approximately two months in the summer of 1972. This time he arranged nursery school for our two oldest daughters, and an apartment for us in the lovely, unspoiled village of Cassis on the other side of the mountain from the campus at Luminy.

Although our discussions this time did not have the same intensity as those of the previous year, it was during these two months that the idea of using logic as a computer programming language was born. It is impossible for me to disentangle our different contributions to the idea. But, in general terms, my ideas were more theoretical and perhaps more philosophical than Alain's. Alain once even referred to them as "poetical", in a sense that I think was intended to be complementary. Alain's ideas were more practical than mine, and were based on a deeper understanding of the computer science issues involved.

These days our joint discussions and discoveries would probably have resulted in joint publications. But in those days communication between researchers based in different countries were primarily conducted by post, which did not encourage the production of joint publications. Moreover, Alain was not driven by a need to publish his work. He was driven by a need to produce practical results underpinned by sound theoretical principles. By the end of the summer of 1972, Alain's group had developed the first version of Prolog, and used it to implement a natural language question-answering system, while I reported an abstract of my own findings at the Mathematical Foundations of Computer Science Conference in Poland.

To get a sense of Alain's approach to research and research publication, type his name into Google Scholar. With the exception of his 1973 technical report, Un systeme de communication homme-machine en francais, co-authored with Henri Kanoui, Robert Pasero and Philippe Roussel, his 1993 book, Constraint Logic Programming, co-edited with Frédéric Benhamou, and his 1996 article, The birth of Prolog, co-authored with Philippe, the remaining seven of his ten most highly cited publications are single-authored. Moreover, looking into the content of his publications, you will get an appreciation of his style, which was to drill deeply into the topic of his study, take all the related work he could find into consideration, and allow no further distractions.

When I returned to Edinburgh from my two-month visit to Marseille in the summer of 1972, I spread the word about logic programming. The reaction among several, but not all, of my colleagues was enthusiastic. Donald Michie, widely regarded after Alan Turing as the "father of Artificial Intelligence" in Britain, encouraged both his PhD student David H. D. Warren and his post-doctoral researcher Maarten van Emden to work with me on this exciting new development. With the support of my PhD adviser, Bernard Meltzer, who founded the Journal of Artificial Intelligence, I obtained a small NATO research grant to support exchange visits between Edinburgh and Marseille.

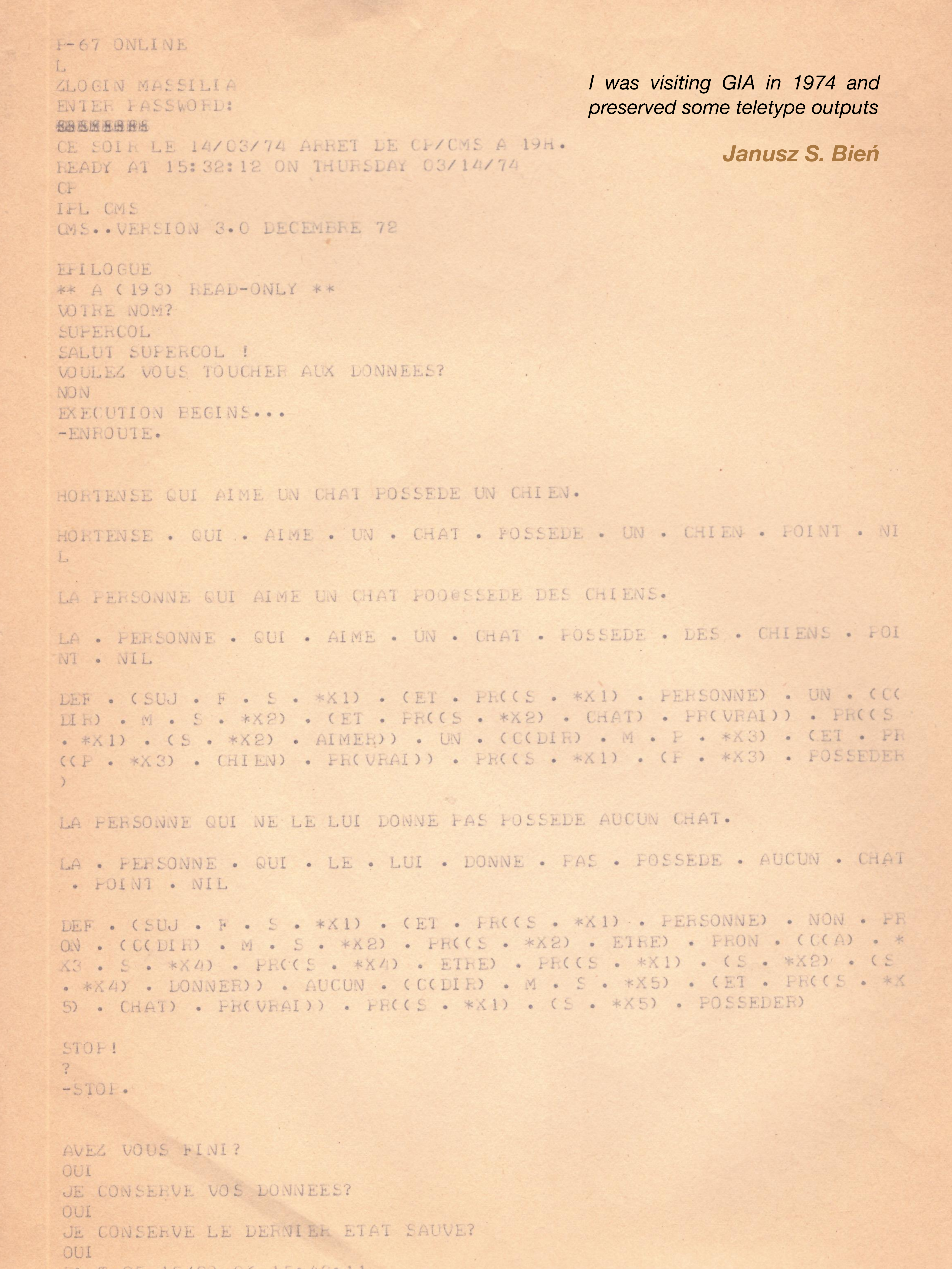
The NATO grant covered the one-year period from October 1973 to October 1974. It supported a further two month visit by me to Marseille, as well as visits by Philippe to Edinburgh and by David to Marseille. Philippe learned about the structure sharing implementation of resolution developed by Robert Boyer and J Moore in Edinburgh, and he incorporated a version of structure sharing into the Prolog system in Marseille. David, in turn, obtained a wealth of knowledge that allowed him to develop and implement the first Prolog compiler. He also further developed Alain's metamorphose grammars and with Fernando Pereira renamed them "definite clause grammars", which is how they are better known today.

The fact that Edinburgh was a hub for research in Artificial Intelligence helped to spread interest in Prolog throughout Europe. But my time in Edinburgh was coming to an end, and I left in January 1975, to become a Reader (the British equivalent of an Associate Professor) at Imperial College in London. It was my turn to set up a research group, following in Alain's footsteps. London's central location made a further contribution to the spread of Prolog.

As a step towards promoting Prolog and related work, I organised a workshop at Imperial College in May 1976, using the term "logic programming" to describe the topic of the workshop. The workshop lasted five days, and it had over 35 participants, including such luminaries as Alan Robinson from the USA, and Alain and Philippe from Marseille. This time, it was Alain who slept on my living room floor.

For me, the London workshop marked the transition of Prolog and logic programming from childhood to adolescence, together with all of the growing up pains that such transitions incur. Alain and I continued to collaborate, most notably during the first International Logic Programming Conference in 1982 in Marseille, and during the EU-supported Compulog Basic Research Action from 1989 to 1993. We also met from time to time both at research meetings and on purely social occasions.

Alain himself is no longer with us, but his memory remains: an intellectual giant, a towering leader, and a generous friend.



```
F-67 ONLINE
L
ZLOGIN MASSILIA
ENTER PASSWORD:
[illegible]
CE SOIR LE 14/03/74 ARRET DE CP/CMS A 19H.
READY AT 15:32:12 ON THURSDAY 03/14/74
CP
IPL CMS
CMS..VERSION 3.0 DECEMBRE 72

EPILOGUE
** A (193) READ-ONLY **
VOTRE NOM?
SUPERCOL
SALUT SUPERCOL !
VOULEZ VOUS TOUCHER AUX DONNEES?
NON
EXECUTION BEGINS...
-ENROUTE.


HORTENSE QUI AIME UN CHAT POSSEDE UN CHIEN.

HORTENSE . QUI . AIME . UN . CHAT . POSSEDE . UN . CHIEN . POINT . NI
L

LA PERSONNE QUI AIME UN CHAT POO@SSEDE DES CHIENS.

LA . PERSONNE . QUI . AIME . UN . CHAT . POSSEDE . DES . CHIENS . POI
NT . NIL

DEF . (SUJ . F . S . *X1) . (ET . PR((S . *X1) . PERSONNE) . UN . ((C(
DIR) . M . S . *X2) . (ET . PR((S . *X2) . CHAT) . PR(VRAI)) . PR((S
. *X1) . (S . *X2) . AIMER)) . UN . ((C(DIR) . M . P . *X3) . (ET . PR
((P . *X3) . CHIEN) . PR(VRAI)) . PR((S . *X1) . (P . *X3) . POSSEDER
)

LA PERSONNE QUI NE LE LUI DONNE PAS POSSEDE AUCUN CHAT.

LA . PERSONNE . QUI . LE . LUI . DONNE . PAS . POSSEDE . AUCUN . CHAT
 . POINT . NIL

DEF . (SUJ . F . S . *X1) . (ET . PR((S . *X1) . PERSONNE) . NON . PR
ON . ((C(DIR) . M . S . *X2) . PR((S . *X2) . ETRE) . PRON . ((C(A) . *
X3 . S . *X4) . PR((S . *X4) . ETRE) . PR((S . *X1) . (S . *X2) . (S
. *X4) . DONNER)) . AUCUN . ((C(DIR) . M . S . *X5) . (ET . PR((S . *X
5) . CHAT) . PR(VRAI)) . PR((S . *X1) . (S . *X5) . POSSEDER)

STOP!
?
-STOP.


AVEZ VOUS FINI?
OUI
JE CONSERVE VOS DONNEES?
OUI
JE CONSERVE LE DERNIER ETAT SAUVE?
OUI
```

I was visiting GIA in 1974 and preserved some teletype outputs

Janusz S. Bień

François Fages

Alain Colmerauer a laissé à la postérité le langage Prolog de "programmation en logique", toujours utilisé aujourd'hui, qu'il a créé au début des années soixante dix, d'abord comme un formalisme grammatical extrêmement puissant pour l'expression des langues naturelles, et immédiatement ensuite en collaboration avec Robert Kowalski, comme un langage de programmation à part entière, très original, fondé sur l'utilisation de relations plutôt que de fonctions, et plus précisément sur un fragment de la logique du premier ordre : les clauses de Horn.

En 1982, avec l'introduction des contraintes d'inégalité entre termes dans Prolog II (à laquelle je contribue modestement en corrigeant l'algorithme d'unification des termes infinis rationnels que me soumet Michel Van Caneghem lors d'un petit déjeuner mémorable au CIRM dont c'était le début, comme pour moi en recherche) Alain me pose la question de la théorie de Prolog II, de la sémantique logique des termes infinis et des contraintes d'inégalité. Je ne vois pas.

Ce sont Jean-Louis Lassez, Michael Maher et Joxan Jaffar qui feront la théorie générale de la programmation logique avec contraintes en 1984 en généralisant l'algèbre des termes à tout domaine de contraintes présenté par une théorie axiomatique complète. Ils créent le système CLP(R) avec contraintes linéaires sur les réels, concurremment à Prolog III qui inclut de la même manière les contraintes linéaires sur les nombres rationnels, mais aussi d'autres contraintes qui donneront naissance au concept de contrainte globale sur des structures de données particulières, et qu'Alain et Michel mettront en oeuvre dans Prolog IV au début des années 90.

En cette même année 1984, Mehmet Dincbas présente aux célèbres journées d'IA d'Avignon, le système CHIP issu de l'équipe de Hervé Gallaire au laboratoire ECRC de Münich. Ce système CHIP inclut dans Prolog, suivant le même schéma général de “programmation logique avec contraintes", les contraintes sur les domaines finis, et montre la résolution spectaculaire de problèmes combinatoires difficiles, dans la lignée des travaux pionniers de Jean-Louis Laurière en 1978 sur le système Alice.

Ce qui a toujours animé Alain est l'envie de faire faire à un ordinateur des choses radicalement nouvelles que personne n'avait jamais pu programmer auparavant. Au début, en traitement des langues naturelles, puis ensuite en résolution de puzzles logiques qu'Alain ne manquera jamais de considérer, rechercher et inclure dans ses cours et ses manuels de Prolog.

Alain n'avait pas un profil académique standard. On se disait souvent qu'il fallait 10 ans pour comprendre ce qu'il voulait dire. Il pouvait sembler ne pas beaucoup se préoccuper de communication à l'extérieur au travers de publications scientifiques bien écrites, mais en réalité il avait des intuitions extrêmement novatrices et je dirais géniales, qui demandaient nécessairement du temps et le travail d'autres personnes pour être vraiment comprises et communiquées.

C'est une grande chance pour moi d'avoir connu Alain, "la bande à Colmé" et Colette qui le soutenait toujours bien sûr. Contrairement aux apparences pour certains, Alain n'était pas indifférent à l'image qu'il renvoyait de lui-même et qui pouvait paraître assez énigmatique. Je me souviens de son stress la veille d'une évaluation CNRS du LIM où je me trouvais chez eux avec leur fille Alice qui nous faisait des tours de passe-passe le soir pour détendre l'atmosphère.

Alain nous montre qu'on pouvait être génial et hors norme. C'était une autre époque, une très belle époque de pionniers, qu'il est instructif de se remémorer et utile de transmettre.

François Fages
Directeur de recherche Inria

Mehmet Dincbas

Hommage à Alain COLMERAUER

Mehmet DINCBAS

Septembre 2020

J’ai eu la chance, avec mes anciens collègues H. Gallaire, J-M. Nicolas et C. Pradelles du Centre de Recherche ONERA-CERT, d’être parmi les premiers utilisateurs des travaux d'Alain Colmerauer et de PROLOG (dès les années 70). Ces travaux ont servi de base aussi bien dans ma carrière de chercheur que dans celle de chef d’entreprise à COSYTEC à travers les logiciels que nous avons développés pour nos clients industriels.

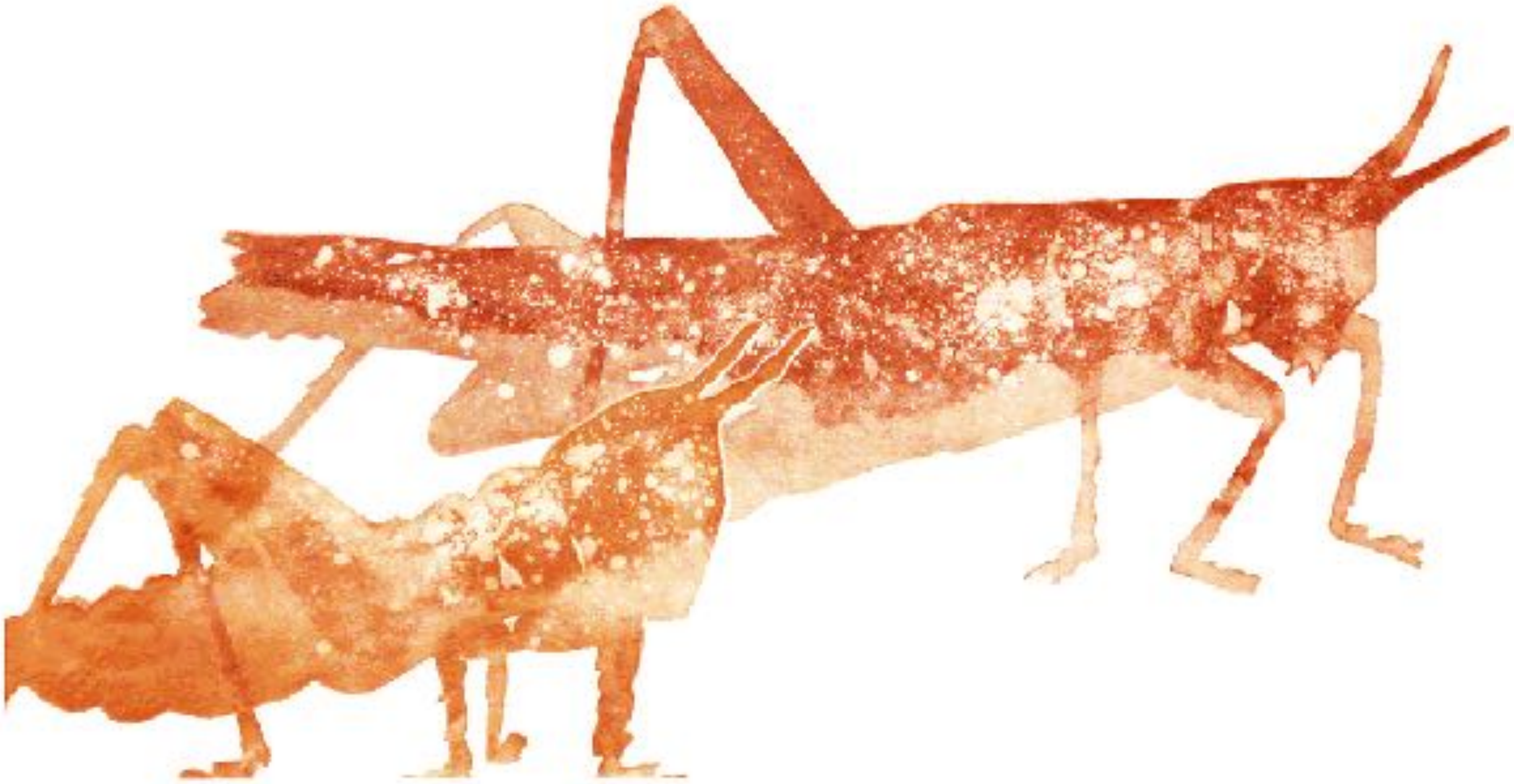
J’ai eu également la chance de rencontrer personnellement Alain à plusieurs reprises aussi bien à Marseille qu’à Toulouse, à Lannion ou à Munich, notamment dans le cadre des collaborations avec le GIA (Groupe Intelligence Artificielle) de Marseille.

Nous avons également eu le grand privilège de recevoir deux fois Alain pour le Séminaire de Programmation Logique de Trégastel organisé par le CNET pour la présentation publique, en avant-première, de « Prolog en 10 figures » en 1983 et « Note sur Prolog III » en 1986.

Alain était un chercheur « hors norme », doté d’une très grande créativité et des idées extrêmement originales. Ses travaux ont été à l’origine des domaines de recherche très fertiles sur la Programmation Logique et la Programmation par Contraintes.

Je peux témoigner que, près de 50 ans après sa naissance, son « bébé » PROLOG et ses multiples successeurs sont toujours (et plus que jamais) utilisés dans des applications opérationnelles, et sont même au cœur de l’organisation du « business process » de certaines entreprises dans les medias, les transports et les services d’intervention d’urgence.

Veronica Dahl



In Memoriam: Alain Colmerauer

If I were asked to distil into a single word the essence of what I learned from Alain, it would have to be: beauty.

Even when computers were much less efficient than today, and hacking to gain speed appeared as a must, Alain aimed always at concise and elegant formulations, betting that efficiency would follow largely from the resulting clarity– and so it typically did.

Alain's focus on beauty led him naturally to think in more general, more abstract terms than strictly required by the problem at hand. Thus, in his Grenoble University thesis dissertation on finding a program's syntactic errors, his goal of doing so elegantly in a single pass led him to a very general solution solidly resting on precedence relations theory.

Next, during his military (so called) service at Université de Montreal as "Coopérant Scientifique" in the late 60's, Alain solved an automatic translation problem by lifting it into the higher level framework of Q-systems, a formalism he developed for processing linguistic data through tree-rewriting rules. In his own words: "The experience gained using the Q-systems reinforced my feeling that to suitably solve a problem it was necessary to develop high level programming languages, even if their execution time would appear impracticably slow at that time."

Just how formidably his bet on beauty over speed paid off even in practical terms is firstly attested to by the fact that the TAUM METEO system, created only a few years later using an industrial version of Q-systems, was the first industrial application of Machine Translation.

Secondly and most importantly, this approach led Alain to develop Prolog, once back in France as Head of the Groupe d'Intelligence Artificielle he created at Université d'Aix-Marseille II.

Curiously, his aim was not to produce a programming language, but rather a tool for enabling computer input of concepts described in French that could later be queried in French.

Constantly tacking between theory and practice, he interleaved working with Robert Pasero on a French analyser and conducting numerous experiments with Philippe Roussel and Jean Trudel on automated theorem proving methods. This led him to adopt a refinement of Alan Robinson's resolution principle, SL-resolution, introduced by Bob Kowalski and Donald Kuehner. In the process, Prolog was born.

It soon became apparent that the new language, which integrates Alain's Q-systems approach with SL-resolution and with Bob Kowalski's procedural interpretation of Horn clauses, was nothing less than a revolution in scientific thinking about programming: it catapulted computing sciences from its old number-crunching, algorithm-focussed and mostly imperative paradigm into the new era of inferential engines. An era in which we no longer measure computation speed in terms of calculations per second, but in terms of inferences per second− a fantastic qualitative leap, with import well beyond the natural language processing uses for which it had been first conceived.

Prolog's semantic underpinnings were captured by Maarten van Emden and Bob Kowalski. Improvements made it competitive as an AI language: Alain's search space cut operation, his elegantly simple use of this operation for an implementation in two clauses of negation-as-failure (which Keith Clark then endowed with the first proper semantics), and techniques such as structure sharing (R. Boyer and J. Moore) and structure copying (Maurice Bruynooghe and Chris Mellish). When Prolog was chosen for the Fifth Generation Computing Project in Japan, it became famous worldwide.

Early implementations by Philippe Roussel, Gérard Battani and Henri Meloni and by David H.D. Warren, Fernando Pereira and Luis Pereira paved the way to Warren's Abstract Machine, which became the standard for Prolog compilers.

Commercial applications mushroomed in areas such as expert systems, automated translators, interpreters and compilers, automatic design, deductive databases, and computer-aided instruction. Alain's syntactic variant of Prolog for language processing− Metamorphosis Grammars, also known as Definite Clause Grammars− became a standard Prolog feature, as well as inspiring further logical variants of executable grammar.

In 1984, Alain founded a research company, PROLOGIA, with colleagues Geneviève Bossu, Henri Garetta, Henri Kanoui, Robert Pasero, Jean-Francois Pique and Michel van Caneghem, for commercializing Prolog and developing practical software solutions to complex problems, still in operation.

It took uncompromising determination and strength of character for Alain to materialise his ideas into practical form in the face of generalised disbelief at the time when Prolog was being conceived. When he visited Stanford with Philippe Roussel and Robert Pasero in 1972 to present his ideas, they were met with scepticism and almost laughter. Yet Alain's unwavering faith in his ideas and his happy and enterprising spirit kept him going. After history proved him right, John McCarthy himself showed up aboard the Queen Mary's First Workshop on Logic Programming, trying to learn more about the new paradigm and seeking our help to introduce it at Stanford.

Those early days in the aptly named Luminy, where I had the privilege of studying under Alain's supervision from 1975 to 1977, are indelibly imprinted in my mind as an improbable domain of peaceful and inspiring Mediterranean landscapes, relentless mistral, and meridional friendliness in Alain's uniquely creative lab. I suspect Alain's love of sailing in the blue waters dazzlingly framed in white calcareous rock provided him with additional source of inspiration.

Already in 1976, Alain felt constraints were the next giant leap needed in logic programming− a dream that spawned two most influential further contributions. The first was the move from unification to equations and inequations over infinite trees (late 70's/ early 80's), which materialised in award-winning software and paved the way for many of the accomplishments in Constraint LP that flourished as from the mid-80's.

Alain's second crucial contribution to CP were his highly innovative extensions of constraint solving and its semantic underpinnings into richer domains: from Prolog II's infinite trees and non-equality predicate − admirably implemented with Michel van Caneghem on a very primitive personal computer with floppy disk virtual memory−; to Prolog III's extension to Boolean algebra and real numbers; to the beautiful formalization of interval reasoning and the unifying presentation of the heterogeneous solvers in Prolog IV; to generalising the idea of constraint solving by intervals narrowing into general constraints.

These two achievements influenced in particular Joxan Jaffar and Jean-Louis Lassez in developing the theory underlying CLP. The second influenced Pascal van Hentenryck to leverage the work of Jean-Louis Lauriere and Alan Mackworth to design CP over finite domains.

Alain's subsequent achievements in CP involve a phenomenal ambition: solving non-restricted first-order formulae constraints in the theory of trees, as well as combining them with additive ordered rational numbers.

Remarkably, his most recent research was completed after he recovered from a severe stroke. It once more addresses an entirely different research field− the study of the complexity of universal programs− from a highly original and high level perspective. This was the formal notion of coefficient introspection, expressing the average number of instructions executed by a universal machine for simulating the execution of one of its own instructions.

As a supervisor, Alain was always available for chatting about research, and had a great sense of humour which made his students feel very comfortable around him. He provided them with the exact combination of guidance and independence needed.

He inspired generations of students, as attested by the fifty-one theses he supervised over his career, and also by many other people he helped out with the same dedication as if they had been his own students. In David H.D. Warren's words: "I still think of Alain as a young and vigorous and inspiring research leader. We had many interesting conversations, especially when I first visited Marseille. He gave me a lot of his time, considering I was just an inexperienced visiting PhD student."

Alain's inspiring creativity, his relentless pursuit of his ideas, his unassuming way of producing brilliant gems for our scientific culture, his generosity in sharing them, have all shaped the fields of Logic Programming, Natural Language Processing and Constraint Programming indelibly.

Personally, I feel very privileged and incredibly lucky to have known Alain, a being of so much light, genius, and open-minded humanity. I feel orphaned and at the same time very grateful for his presence all these years, and for all the good that came into my life from his having accepted me into his lab without knowing me, a student from distant Argentina who wrote to him out of the blue, with no connections to invoke.

My condolences to Colette and all of us, my everlasting gratitude to Alain, and my thought that his light is something we will always continue to share and also to spread, having had the luck to bask in it.

Veronica Dahl

Henry Kanoui

1982 – 1983 LE CENTRE MONDIAL INFORMATIQUE

Le Centre Mondial Informatique a été créé en 1981 par Jean-Jacques Servan-Schreiber.

JJSS était un agitateur d'idées et un visionnaire. Dès 1970 il prédit dans « Le Défi Mondial » que l'informatique serait partout, et décrit la société de l'internet avec 30 ans d'avance. Proche de Gaston Defferre, il convainc le président Mitterrand de financer un centre de recherche permettant de repérer et expérimenter les nouvelles technologies de l'informatique et ainsi se préparer à la révolution numérique.

Dès la création du CMI, il fait venir des USA des sommités de l'IA : Nicolas Négroponte et Seymour Papert, bientôt suivis par Raj Reddy pour le diriger.

A cette époque, la communauté informatique est agitée par le projet d'ordinateurs de 5ème génération lancé à grand fracas par les japonais. Le MITI (ministère de l'industrie du Japon) lance un programme sur 10 ans et mobilise des moyens énormes pour concevoir et développer une architecture basée uniquement sur le parallélisme et l'inférence logique. Prolog est choisi de préférence à Lisp pour constituer le noyau du système d'exploitation et de tous les programmes d'applications à venir.

Tout cela suscite un intérêt international grandissant pour Prolog et le Groupe Intelligence Artificielle de Luminy commence à connaître une certaine notoriété. De grandes entreprises comme Bull et Cap-Sogeti financent le portage de Prolog II sur diverses machines françaises. Début 1982, avec l'Apple II Prolog est pour la première fois disponible sur un petit ordinateur individuel, que tout le monde peut se procurer, avec des performances correctes et une utilisation bien plus confortable que sur les mini-ordinateurs poussifs dont on disposait jusque-là.

Et finalement, en juin 1982 on reçoit une invitation pour passer une année au CMI. Je fonce à Paris, je rencontre Négroponte et on fait affaire. Immédiatement après Alain décide de venir.

En septembre nous voilà à Paris. Le CMI, c'est le grand luxe : immeuble somptueux avenue Matignon et mobilier de designer. On nous demande de choisir bureaux et fauteuils dans les catalogues. Ça nous changeait de l'université ! Les ordinateurs étaient ce qui se faisait de mieux à l'époque, des machines Vax et PDP-10 qu'on n'avait jamais vues en France et qui étaient l'apanage des grandes universités américaines.

Au CMI on trouvait toutes sortes de gens : hackers chevelus, genre hippies, venus de France ou d'Amérique ; cadres de grandes entreprises mises à contribution, hauts fonctionnaires détachés, etc.

En fait le CMI n'avait pas vraiment de stratégie. Chacun travaillait sur ses propres sujets et selon ses propres intérêts. Alain peaufinait son article « Prolog en 10 figures » avec sa fameuse horloge. En même temps, il jetait les bases de Prolog III avec le concours de 2 étudiants de DEA invités au CMI.

Mais le plus intéressant, ce sont les rencontres, c'est d'être là où les choses se passent, à portée de main des opportunités. En fait tout le petit monde académique et industriel était très curieux de ce qui se passait au CMI. Nous recevions constamment des visites des uns ou des autres et nous étions invités à beaucoup de rencontres et de réunions. C'est ainsi que nous avons fait la connaissance de Geoffrey Staines, un éditeur qui nous incitera plus tard à écrire notre livre sur Prolog II. Fin 1982, Apple nous décerne la « Pomme d'Or ». Le prix nous est remis en grande cérémonie par le vice-président d'Apple.

Parmi tous ces gens, nous avions rencontré M. Paul (sic), un personnage un peu énigmatique du Ministère de la Recherche. On n'a jamais su ce qu'il faisait exactement, mais il s'était entiché de nous et venait nous voir à tout propos. Nous, nous commencions à penser sérieusement à fonder une société pour commercialiser Prolog II. M. Paul a foncé sur cette idée et il a fini par nous proposer un contrat pour porter Prolog II sur IBM-PC/MS-DOS. C'est comme ça que le business plan d'une société autour de Prolog a été monté. Ça a convaincu M. Paul et il nous a promis un contrat de 500 000F (75 000€).

En fait, en un an au CMI on a rencontré plus de gens influents qu'en 10 ans à Marseille. On a établi des liens solides avec de grandes compagnies comme Bull, Cap-Sogeti ou Thomson, des agences gouvernementales qui financent des projets, des banquiers, des représentants des médias, etc. Des années plus tard tous ces contacts se révèleront déterminants.

C'était une très belle année. Nous avions de beaux titres, des salaires confortables, et nous habitions dans le carré Faubourg Saint-Honoré – Champs-Elysées, à 2 pas de l'Etoile. L'ambiance du CMI était très détendue, autour d'un apéro ou la fête de Thanksgiving 82 organisé par les américains du CMI. La semaine, on allait dans les endroits qu'Alain avait fréquentés quand il était jeune (il aimait beaucoup la Rhumerie, Bd Saint-Germain). Ou alors on écumait les brasseries et les restaurants d'huitres du quartier. Chaque week-end on faisait l'aller-retour Paris Marseille en avion et taxi tous frais payés.

Toute cette agitation avait aiguisé nos ambitions et nous sommes revenus à Marseille avec des projets plein la tête. Dès notre retour nous avons travaillé à la mise en place de Prologia. Il fallait trouver des locaux (finalement la faculté nous a hébergés), recruter et s'assurer que le CNRS nous permettrait d'exploiter Prolog. Nous avons finalisé le contrat promis par M. Paul, mobilisé partenaires et clients potentiels et lancé la création de PrologIA (pour Prolog et Intelligence Artificielle) qui a été immatriculée en janvier 84.

Frédéric Gardi

Frédéric Gardi
Co-fondateur & directeur associé, LocalSolver
24 avenue Hoche 75008 Paris, France

fgardi@localsolver.com
06 77 82 37 87

À Paris, le 17 février 2021

Ancien étudiant en informatique de la Faculté des Sciences de Luminy (1998-2001), dernier doctorant de Michel Van Caneghem (2002-2005), je dois beaucoup à Alain Colmerauer et aux enseignants-chercheurs du département d’informatique. Ils m'ont fait confiance et m’ont formé ; ils m’ont aussi passionné et inspiré. Au-delà du langage lui-même, j'ai découvert auprès d’eux l'aventure qu’avait représenté Prolog. J’ai fouillé ce passé plus encore lorsque j'étais jeune ingénieur au sein de la société Prologia, passant de longues soirées à exhumer rapports de recherche, articles scientifiques, et autres livres que j’y trouvais. Cette idée d'un solveur mathématique « universel » m'est apparue comme un graal scientifique et technologique.

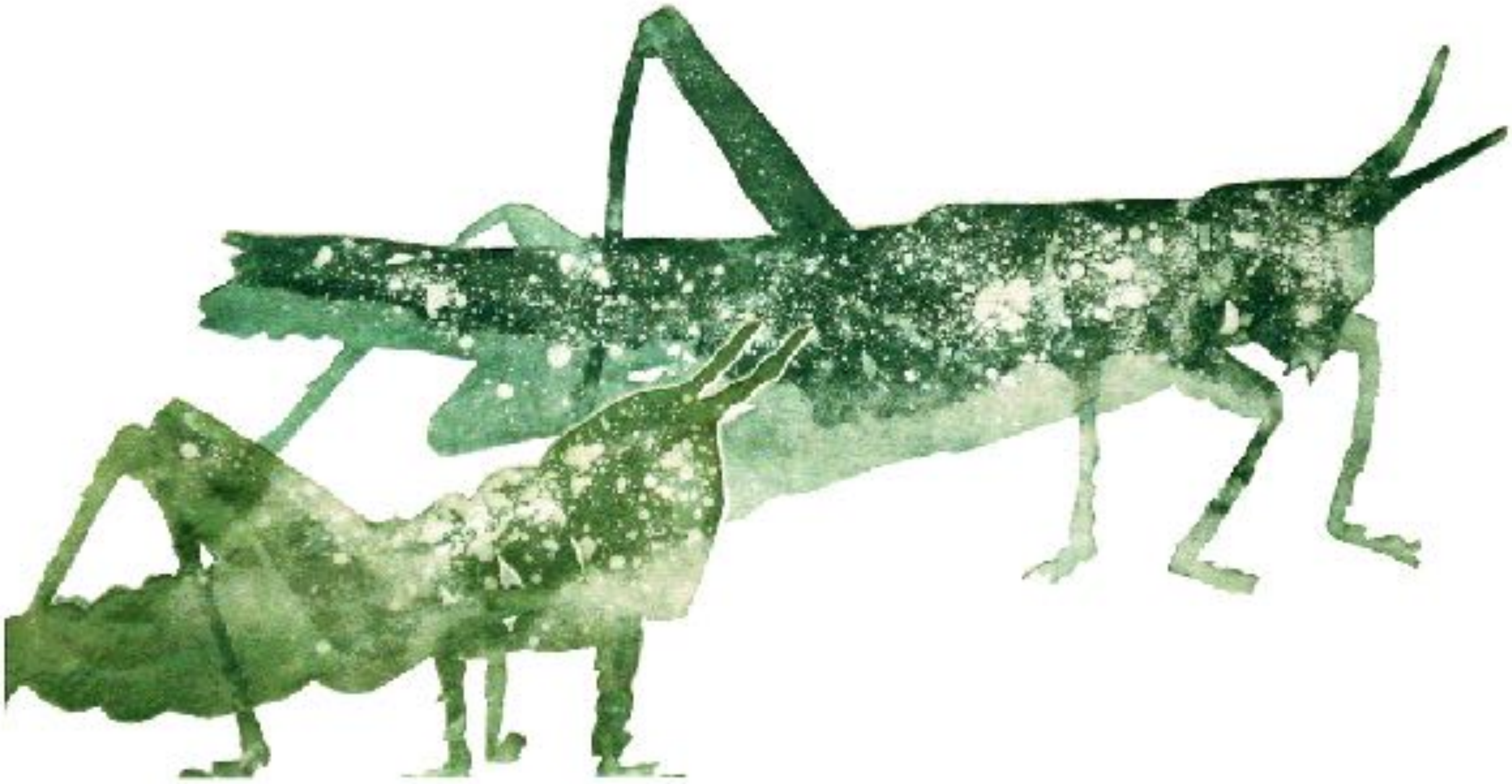
Ce n'est donc pas par hasard que quelques années plus tard je me suis lancé en quête de ce graal. Aidé de mes collègues marseillais Bertrand Estellon et Karim Nouioua, devenus enseignants-chercheurs au sein de cette même faculté, et grâce à l’adhésion de mes collègues du Bouygues e-lab, notamment Thierry Benoist, responsable du département de Recherche Opérationnelle, nous avons développé un solveur d’optimisation. Son originalité consistait précisément en l’intégration d’ingrédients heuristiques qui n’étaient pas d’actualité à l’époque du développement de Prolog.

Ce solveur, qui s’appelle à tort LocalSolver puisque c’est aujourd’hui un solveur global combinant des techniques de résolution exactes et heuristiques, doit donc beaucoup à Alain et à toute l’équipe du département d’informatique de Luminy. Sans eux, cette aventure et ce solveur n’auraient certainement jamais existé. On peut voir LocalSolver comme une des nombreuses suites et ramifications de cette fabuleuse épopée qu’a été Prolog. Or peu autour de nous le perçoivent. Sans doute même aucun, parmi nos collaborateurs, nos clients et utilisateurs.

Je suis heureux de pouvoir l’évoquer ici, aujourd’hui, modestement, en le souvenir d’Alain.

F Gardi

Brian Harris



Alain Colmerauer, Machine Translation Pioneer

TAUM group showing off a piece of Q-System output
circa 1970

My recent birthday, my 88th, was clouded over by news that somebody I'd worked for nearly 50 years ago had died. He was **Alain Colmerauer**, an outstanding French computer scientist, *Chevalier de la Légion d'Honneur* (the French equivalent of a knighthood), emeritus professor at the University of Marseille-Luminy. My work for him only lasted three years, from 1968 to 1971, but they were very formative years for me. Also for others; I've received messages from two other ex-colleagues saying they were influenced by him. All that was in the days before I conceived the notion of Natural Translation, when I was part of a Canadian group doing research on machine translation. There will be many obits and tributes to him, but I would like to add a few personal reminiscences.

In the late 1960s I was working as a linguistic research assistant in the machine translation project at the University of Montreal, a French-speaking university. We had acquired a linguistic model of the translation process from the leading research group in France, the one at the University of Grenoble. It was the dependency grammar of the French linguist Lucien Tesnière. But we didn't have software to implement it.

Then in 1968 Alain came to Canada and to the University of Montreal as a *coopérant*. The *coopérants* were young French university graduates who, under a scheme devised by De Gaulle's government, were sent to work for two years in developing countries in lieu of their compulsory military service. During that time they received only army pay. For diplomatic reasons, probably to favour relations with Quebec, Canada was included among the receiving countries. With Alain came at least two other coopérant computer graduates whom I came to know, **Michel van Caneghem** and **Guy de Chastellier**. They came from the University of Grenoble; it had a strong computer science department, but Alain's background was in mathematics. At the young age of 28 he had recently obtained a *Doctorat d'État*, a French superior, competitive doctorate that no longer exists. One day in his office later on he asked me if I would like to see his doctoral thesis. So he showed it to me. It had about 40 pages. I expressed surprise that he could obtain a *Doctorat d'État* with a thesis of a mere 40 pages. He smiled and replied, "Only in mathematics."

Though Grenoble had a well-known machine translation project, Alain wasn't in it and didn't come directly to our Montreal project. He came first to the computer science department. The university had a state-of-the-art computer centre wth a CDC mainframe and an encouraging engineer manager, **Jean Baudot**, who was interested in linguistics. But the head of the MT project, **Guy Rondeau**, was a good talent-spotter (after all, he recruited me!) and he didn't miss the opportunity to recruit Alain. And so we met. Then Rondeau left the university hurriedly in a huff and the university needed a credible replacement in order to safeguard its lucrative MT research contract with the National Research Council of Canada (NRC). So it appointed Alain, and that's how he became my boss.

One of the first things Alain did was stop the quarterly publication of our research papers. He said we should not publish until we had something really substantial to present. It taught me to look down on the 'publish or die' attitude so prevalent in our universities, which produces more minor articles and theses than people have time to read. We eventually waited two years.

He set about providing us with the software we needed. The leading linguistic paradigm of the time was transformational grammar (TG). Alain was well acquainted with the TG of Noam Chomsky through his wife, who was writing her PhD thesis on it. His first product was a TG program which he called a ***W-Grammar*** because it was inspired by the Algol programming language invented by the Dutch computer scientist A. van Wijngaarden. Indeed it was through Alain that I learnt about the European style of programming represented by Algol, more logical and transparent than the then current American languages like Fortran.

W-Grammar was usable for MT and so I wrote the first (and perhaps only) proof-of-concept piece of translation in it, just one sentence. Alain was a bit disappointed that I didn't use Chomskyan TG but the Tesnière dependency model. However he was very open-minded and later even allowed me time and resources to work on my own side-project, the Transformulator (a forerunner of translation memories). He was also sure of himself. Some computer science colleagues told him, on theoretical grounds, that the Q-Systems might not work; but he thought the danger was negligible and went ahead anyway.

I liked W-Grammar and would have continued with it, but something better soon came from him that rendered it obsolete. This was his much better known ***Q-Systems***. (The Q stood for Quebec.) There is no point in describing Q-Systems here, since there is a good article on them in *Wikipedia*. Alain was a hands-on computer scientist: he was proud that he programmed Q-Systems in Algol himself in the space of six intensive weeks.

Q-Systems were a high-level language, a revolutionary tool for us linguists. With them we were equipped to devise an elaborate English to French MT system. The task was too much for one person, so it was split up into stages and parcelled out. The chaining of programs in Q-Systems made this feasible. I got to design the English morphology analyser and programmed it with the aid of a student, Laurent Belisle. Alain once paid me what for me was a supreme compliment: "Brian, your morphology never fails."

By 1971 we were ready to make a presentation to the NRC and to publish. The publication is the volume ***TAUM 71***. (TAUM stands for Traduction automatique à l'Université de Montréal.) It's difficult to find today because it was only intended for the NRC, but it's a classic of the so-called rule-governed approach to MT. That paradigm was overturned by the invention of statistical MT in the late 1980s, so it might look as if we were barking up the wrong tree. However, the right tree wasn't available to us, because the computers of the time couldn't have handled the enormous data bases that are needed for statistical MT.

By 1971, with *TAUM 71* published and his *coopérant* obligations acquitted, Alain felt the tug of his home country and returned to France. One side-consequence was that he left me his spacious Montreal apartment on prestigious Nun's Island in the St. Lawrence river along with its antique furniture. But not long afterwards I myself left Montreal for Ottawa. Thereafter our interests diverged so widely, his towards computer programming and mine towards translation theory, that I had little contact with him.

I visited him once at his office at Marseille-Luminy University and was present there at a discussion in which the ever faithful Michel van Caneghem was urging him to switch to what was then the latest development in computing, a micro-computer. I attended Guy de Chastellier's wedding in the Montreal Basilica. But these days you can watch people's careers from afar on the internet. And, as you can judge from the above, those halcyon years in Montreal under Alain's leadership have remained bright in my memory.

References

Alain Colmerauer (ed.). *TAUM 71*. Montreal: Projet de Traduction Automatique de l'Université de Montréal, 1971. 223 p.

Click [here] or go to

https://books.google.es/books/about/
Projet_de_traduction_automatique_de_l_un.html?
id=4lL_nQEACAAJ&redir_esc=y.

Alain Colmerauer and Guy de Chastellier. *W-Grammar*. Département d'informatique, Université de Montréal, c1969, 8 p.

Click [here] or go to alain.colmerauer.free.fr/alcol/ArchivesPublications/Wgrammar/Wgrammar.pdf.

Q-systems. *Wikipedia*. 2016.

Brian Harris and Laurent Belisle. POLYGRAM grapho-morphology analyzer for English. In *TAUM 71*, pp. 46-105.

Image

Alain Colmerauer is holding the Q-System output. Far left with pipe is Michel van Caneghem. With long hair, looking over the output, is Jules Dansereau, a Canadian language analyst for French. Behind Jules may be Richard Kittredge, American linguist.

Photo by courtesy of Colette Colmerauer.

Philippe Jorrand

Prolog est orphelin

Alain Colmerauer était un ancien de l'IMAG (Institut d'Informatique et de Mathématiques Appliquées de Grenoble), devenu une personnalité scientifique de premier plan par le rayonnement international de l'œuvre majeure de sa recherche, le langage PROLOG.

Alain Colmerauer était un élève de la première promotion de l'ENSIMAG, diplômée en 1963.
Il a débuté sa recherche au Laboratoire de Calcul de l'Université de Grenoble, l'ancêtre du LIG. Dans sa thèse, soutenue en 1967, il développait les bases théoriques d'une méthode d'analyse syntaxique. Puis, pendant son séjour de deux ans à l'Université de Montréal, c'est en imaginant une utilisation originale des grammaires à deux niveaux (les « W-grammaires »), qu'il a établi les bases embryonnaires de ce qui allait devenir PROLOG.

De retour en France en 1970, il accomplit ensuite toute sa carrière à l'Université de Marseille, où il devient professeur. C'est là, au Groupe d'Intelligence Artificielle du campus de Luminy, qu'il forme avec détermination une petite équipe de jeunes doctorants, puis d'enseignants-chercheurs, pour développer la PROgrammation en LOGique. Sous sa direction, c'est ce petit groupe qui a élaboré les fondements théoriques de cette approche originale de la programmation, puis conçu et mis en œuvre les versions successives du langage qui allait connaître un succès international et être la source d'un courant de recherche fertile : PROLOG I, PROLOG II, PROLOG III où les contraintes linéaires venaient rejoindre la logique, puis Prolog IV avec une théorie d'approximation plus aboutie, des contraintes sur les intervalles et un solveur.

Par ailleurs, le langage Prolog sera adopté par le projet d'ordinateur de 5ème génération développé par le MITI au Japon dont l'objectif était de créer une industrie et les technologies de l'intelligence artificielle à la fin des années 80. Une entreprise PrologIA, distribuera le langage dans ses différentes versions.

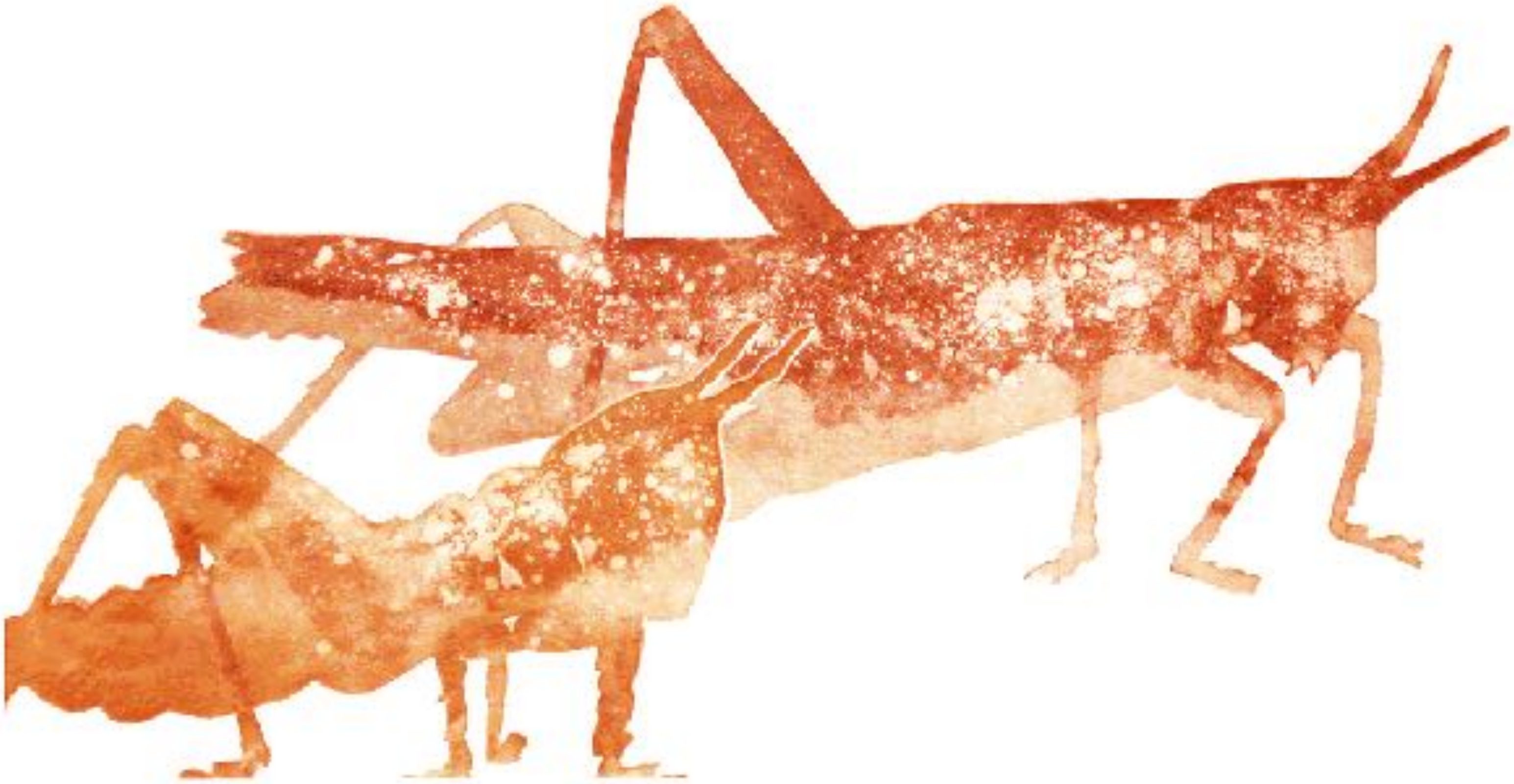
Alain Colmerauer a toujours été un esprit original. Il se défiait de tout ce qui peut ressembler à une pensée unique, et n'hésitait pas à exprimer des idées parfois iconoclastes, mais souvent fécondes. Il croyait à ce qu'il faisait, et sa ténacité lui a souvent été utile face à quelques difficultés institutionnelles et à l'incrédulité de collègues plus installés que lui dans les modes scientifiques. Pour ceux qui l'ont bien connu pendant de longues années, Alain était un ami solide.

Alain Colmerauer est décédé à Marseille, le vendredi 12 mai 2017.

Philippe Jorrand
Directeur de recherche émérite au CNRS

Texte publié dans Binaire par Pierre Paradinas le 18 mai 2017
https://www.lemonde.fr/blog/binaire/2017/05/18/prolog-est-orphelin/

Denis Lugiez



Hommage à Alain Colmerauer

Alain Colmerauer a sans doute été l'informaticien français le plus connu dans le monde dans les années 80, suite au choix de PROLOG par le MITI japonais pour le programme d'ordinateur de 5ème génération. Nul n'étant prophète en son pays, on cherchera en vain un amphithéâtre baptisé en son honneur. Pourtant, Alain Colmerauer a été un pionnier qui aurait pu cocher toutes les cases désormais imposées aux chercheurs qui visent l'excellence.

Choix d'étude : à l'époque où les ordinateurs s'appelaient calculateurs et n'existaient que dans quelques centres de calcul, il a été diplômé en 1963 de la première promotion de l'ENSIMAG, devenue l'école de référence pour la formation en informatique. Premier exemple d'originalité et de flair pour les thèmes d'avenir.

Thème de recherche : sa thèse au Laboratoire de calcul de l'université de Grenoble en 1967 porte sur des aspects théoriques de l'analyse syntaxique, une base de l'informatique et du traitement des langues naturelles. Là encore, un choix de thématique porteuse d'avenir. Les traducteurs automatiques de Google et autres sont les héritiers des travaux de cette époque.

Séjour de recherche à l'étranger : pendant deux ans à l'université de Montréal, il travaille sur le traitement du langage et ce séjour porte en germe le développement de PROLOG.

Création d'une équipe : en 1970, recruté à l'université d'Aix-Marseille II, il constitue une petite équipe d'ingénieurs, chercheurs et doctorants qui établissent le paradigme de la programmation logique et réalisent le premier compilateur PROLOG (une prouesse technologique compte-tenu du matériel de l'époque). Notez que ce paradigme repose sur une méthode de démonstration automatique de théorèmes, c'est de l'Intelligence Artificielle (IA) !

Valorisation des résultats : fin 80, la société prologIA - on ne disait pas start-up à l'époque -a été créée pour promouvoir PROLOG et ses applications et elle a été une pleine réussite.

Recherche appliquée et fondamentale : le paradigme de la Programmation logique a été défini par l'équipe d'Alain Colmerauer (et d'autres notamment Kowalski à Edinbourgh) et son équipe a prouvé que ce paradigme était opérationnel via la réalisation de PROLOG.

Savoir rebondir : Alain Colmerauer ne s'est pas limité à PROLOG, il est aussi un des inventeurs d'un autre paradigme, la programmation par contrainte qui a connu un énorme développement dans les années 90 et cela a conduit à la série des langages PROLOG II, PROLOG III et PROLOG IV avec l'idée que le programmeur se contente de décrire son problème qui est résolu par le solveur de contraintes associé au noyau PROLOG.

La personnalité d'Alain Colmerauer se caractérise par une indépendance en particulier vis à vis des modes, une originalité de pensée et une persévérance qui lui ont permis d'accomplir une œuvre scientifique de tout premier ordre qui a eu une très forte visibilité internationale et de former de nombreux élèves. Il convient de saluer sa mémoire avec tout le respect qui lui est dû. Alain Colmerauer est décédé le 12 mai dernier, à l'âge de 76 ans.

http://alain.colmerauer.free.fr/

Denis Lugiez
Professeur d'informatique
Directeur adjoint du Labex Archimède

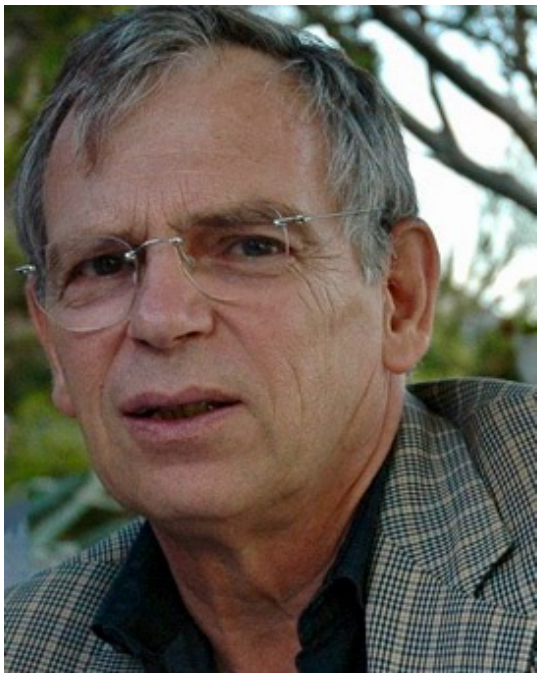
Frédéric Benhamou

Décès d'Alain Colmerauer, pionnier de l'intelligence artificielle

Frédéric Benhamou[1]

Prologue

Pionnier de l'intelligence artificielle, Alain Colmerauer restera dans l'histoire de l'informatique comme le créateur du langage de programmation Prolog, qu'il a développé avec son équipe au début des années 70 et sur lequel il a travaillé, littéralement, jusqu'à la fin de sa vie.



À la fois mathématicien et informaticien, chercheur infatigable et visionnaire, il a perçu très tôt l'intérêt d'inverser la logique de la programmation classique, où chaque action de l'ordinateur doit être programmée très précisément, pour créer la programmation logique. Cette programmation déclarative décrit un univers auquel on s'intéresse et l'utilisateur soumet au programme des requêtes qui sont analysées par un algorithme qui fournit les réponses déduites du modèle programmé. Cet algorithme a été inspiré de ceux développés par Alan Robinson, puis Robert Kowalski.

Dans ces années héroïques qui voient naître les fondements de l'intelligence artificielle, Prolog n'en est pourtant pas le seul langage. En effet, dix ans avant la création de Prolog, John McCarthy fonde le laboratoire d'intelligence artificielle de Stanford

[1] Professeur des universités, vice-président de l'université de Nantes.

et crée le langage LISP pour développer la robotique et l'intelligence artificielle. Les deux hommes se rencontreront et une amitié en naîtra, faite d'admiration et de respect mutuel.

Mais ni Alain, ni John n'auraient imaginé alors, qu'en janvier 2017, presque 50 ans plus tard, une « intelligence artificielle » nommée AlphaGo jouerait contre les meilleurs joueurs mondiaux du jeu de Go... et les battrait tous !

Prolog

Prolog a été initialement conçu pour traiter le langage naturel, celui des humains, à une époque où l'une des grandes ambitions de l'intelligence artificielle était déjà de programmer un ordinateur qui puisse traduire des textes entre plusieurs langues. Lors de son séjour à Montréal de 1967 à 1970, en 1968, son implication dans le projet de traduction automatique de l'université de Montréal l'amène de fait à développer un formalisme, les « systèmes-Q », qui lui permettra d'obtenir des résultats remarquables sur le plan linguistique, mais sera surtout le début de l'aventure Prolog.

De retour à Marseille, où il est nommé sur le premier poste de professeur d'informatique de l'université, Alain acquiert la conviction que l'avenir appartient à des langages de programmation de beaucoup plus haut niveau que Fortran, Cobol, ou même Algol 60. Les langages qu'il pressent seront logiques, capables de s'attaquer aux problèmes les plus difficiles. Le premier Prolog est créé en 1971 avec Philippe Roussel, Henri Meloni et Gérard Battani.

Prolog II

Cette première réussite académique, très remarquée, amène sur Prolog et son inventeur les projecteurs de nombreuses universités et le logiciel est diffusé dans le monde entier. L'attention portée à Prolog le conduit alors en 1979 à créer Prolog II, en renversant à nouveau complètement les acquis de la programmation logique pour définir une sémantique qui s'appuie sur une algèbre des arbres rationnels, qu'il invente, et des facilités de programmation multiples qui en font un véritable langage de programmation. Il a déjà en tête d'améliorer Prolog II pour qu'il soit un jour utilisé dans le monde de l'industrie informatique.

C'est à cette période, en 1982, qu'il est invité au tout nouveau Centre mondial informatique et ressource humaine, situé au 22 de l'avenue Matignon à Paris. Il y rencontre des chercheurs célèbres du MIT : Nicholas Negroponte, fondateur du Media Lab du MIT, Richard M. Stallman, créateur de la fondation pour le logiciel libre, Seymour Papert ou Alan Kay.

Cette époque est aussi celle où Prolog est soudainement projeté sur le devant de la scène médiatique, grâce à l'annonce par le MITI (Ministry of International Trade and Industry) japonais d'un projet d'ordinateur de 5^{e} génération conduit sur une décennie et s'appuyant sur les ordinateurs parallèles et sur un système d'exploitation basé sur le langage Prolog. À cette occasion Alain fera partie d'une visite officielle au Japon, où il voyagera dans l'avion du président Mitterrand.

Alain confiera par la suite que ce séjour d'un an au centre mondial lui avait surtout permis de s'isoler et de travailler d'arrache-pied sur la suite de son projet : la programmation par contraintes.

En effet, une fois émancipé de l'approche logique, à ses yeux mortifère, il voyait Prolog comme une machine à résoudre des problèmes d'algèbre. Sur le modèle des arbres rationnels, il pouvait alors étendre son langage à d'autres algèbres : les booléens, les listes, les nombres rationnels. La programmation logique par contraintes était née !

Prolog III

De retour à Marseille à la rentrée 1983, il démarre un nouveau grand projet : Prolog III, une extension de Prolog II qui permettait d'étendre de façon spectaculaire les capacités déductives du langage en introduisant mais surtout en mixant de façon extrêmement élégante des algèbres diverses, en introduisant des aspects continus (l'algèbre des nombres rationnels) et en créant la notion de solveur de contraintes. Le premier prototype de Prolog III est tout de suite un succès. La start-up PrologIA est créée en janvier 1984 par Alain et ses collaborateurs initiaux (Bob Pasero, Michel Van Caneghem, Henri Garetta), et fin 1989 la première version commerciale de Prolog III est enfin disponible. La première conférence internationale sur la programmation par contraintes est organisée par Alain en 1995 à Cassis près de Marseille.

Prolog IV

La dernière étape de cette histoire de Prolog est celle de la conception et du design de Prolog IV qui a occupé son équipe pendant cinq bonnes années. Prolog IV s'inscrit dans la continuité de Prolog III, avec une avancée fondamentale qui est celle d'intégrer et à nouveau de mixer élégamment deux univers distants que sont les mathématiques discrètes, à la base de l'informatique théorique, et les mathématiques continues, celles qui fondent la physique, par exemple. Alain a passé de très longs mois à concevoir mathématiquement la construction sémantique du langage qui était d'une très grande complexité en raison de la richesse des possibilités qui y étaient incluses et il en a tiré un long article théorique, qui accompagne le manuel d'utilisation du logiciel.

Cette dernière production, en partie inachevée, met un terme à la lignée des Prolog marseillais, l'œuvre de sa vie. Elle est pour moi très représentative du scientifique et de l'homme qu'il était. Elle met en évidence son opiniâtreté, son engagement, son humilité, la volonté constante d'apporter autant de soin à la théorie qu'à la mise en œuvre, de tout remettre en cause, toujours, de tout essayer, toujours, de tout expliquer, toujours, de tout maîtriser, toujours, tout en gardant le doute chevillé au corps. Je pense après toutes ces années que c'est ce qu'il nous faut garder de lui et je pense aussi que c'est ce qui fait de lui un immense scientifique et un homme attachant.

Alain Colmerauer était

— professeur d'informatique de classe exceptionnelle de l'université Aix-Marseille,

— correspondant de l'Académie des sciences, chevalier de la Légion d'honneur, et Fellow de l'American Association for Artificial Intelligence.

Ses derniers articles sont consultables sur son site `alain.colmerauer.free.fr`.

Il est décédé le 12 mai 2017 à Marseille.

Dates clés

24 janvier 1941 : naissance à Carcassonne
1968 : Directeur de TAUM (Traduction Automatique Université de Montréal)
1971 : Création de Prolog
1984 : Création de PROLOGIA
12 mai 2017 : décès à Marseille

Pascal Van Hentenryck

Special Issue in honor of Alain Colmerauer's sixtieth birthday - Guest Editors' Introduction

Frédéric Benhamou
IRIN, Université de Nantes
BP 92208. F-44322 Nantes
Cedex 03. France
(*e-mail:* Frederic.Benhamou@irin.univ-nantes.fr)

Pascal Van Hentenryck
Box 1910 Brown University
Providence, RI 02912, USA
(*e-mail:* pvh@cs.brown.edu)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Alain Colmerauer, father of Prolog and pioneer of constraint programming, will be sixty this year. It is a great honor and a great pleasure for us to edit a *festschrifft* celebrating this event on behalf of his contemporary fellow scientists.

We have known Alain for many years. Both of us are from a generation of researchers who were largely inspired by Alain's research on logic and constraint programming. We began our work in the same years, one as a PhD student in Alain's Prolog III project and one as a PhD student in the ECRC CHIP project. We have had many opportunities to meet Alain, to witness the emergence of his ideas in constraint (logic) programming, and to appreciate how much they contributed in creating and fostering a fascinating and challenging research area which is, in our opinion, a fundamental tool for computer science research.

The mid-eighties witnessed the start of three projects on the design and implementation of constraint logic Programming languages: Prolog III in the Groupe d'Intelligence Artificielle (Marseilles), CHIP in European Computer-industry Research Centre (Munich), and CLP(R) in IBM Watson Research Center (Yorktown Heights, NY). These three teams were led by Alain Colmerauer, Mehmet Dincbas, and Jean-Louis Lassez respectively. An entire chapter would be, and hopefully will be, necessary to describe the creative energy, the scientific contributions, the new and bold ideas, the human interactions, and the system developments which took place in these pioneering years. It is now largely accepted that most of these developments were initiated by Alain's seminal work on the semantics of Prolog II, which captures the actual behaviour of the interpreter using equations and disequations over infinite trees. This move from unification to constraints paved the way to incorporating constraint solving over various domains as the main operation of (constraint) logic programming.

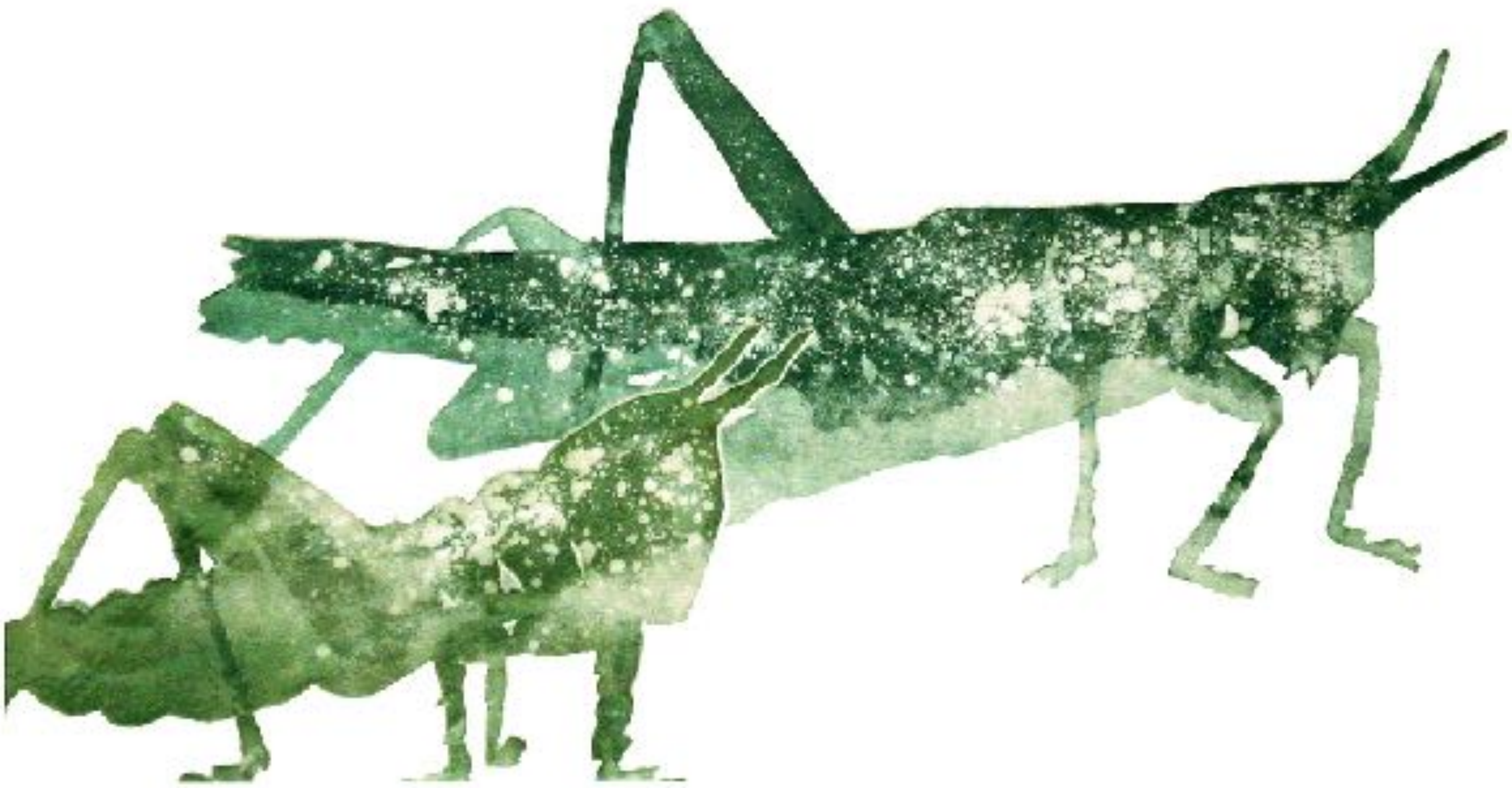
It is probably impossible to describe Alain's unique approach to computer science research; but we will use this opportunity to risk a couple of comments that may shed some light on why so many of us were inspired by his work and were seeking his advice. Probably most impressive is Alain's ability to strike the right balance between semantic simplicity and practical applicability in language design. Many of his seminal results were driven by a constant will of superposing mathematically elegant semantics on practical and innovative systems for fundamental application areas. To see this balance at work, it suffices to mention Prolog, the design and specification of Prolog-II with its rational trees and its constraint-based semantics, the design of Prolog III, and the introduction of a non-standard semantics for real numbers capturing delay-based multiplications. Alain also broke into new territories many times, crossing frontiers, exploring unusual theoretical realms, and exhibiting an amazing creativity over four decades. But, beyond his scientific papers and his systems, his ability of building elegant theories and practical innovative systems together remains his landmark and an inspiration for those of us who were fortunate enough to interact closely with him.

It was not easy to find the right format for this special issue. After some long discussions, we decided to go for a book and to edit part of it as a special issue in *Theory and Practice of Logic Programming*. This newly (re-)born forum for logic and constraint programming researchers is definitely a very appropriate medium for this event. Without unveiling the book project, we selected from its prospective content an introductory paper specially contributed for the event as well as several research articles. To meet the overall flavor of this special issue, we wanted these articles to be particularly prospective and to cover, as much as possible, the many research areas Alain has been interested in. We believe that the contributed papers of this issue fit nicely with our initial view.

We invited Jacques Cohen to contribute the first paper. Jacques is one of Alain's closest colleagues and friends. We thought he was the perfect person to give his own view on Alain's research path and achievements, starting from the very beginning, when they were both PhD students in Grenoble. We particularly thank him for this contribution. The idea of listing the numerous PhD theses supervised by Alain is also his.

...

Jacques Cohen



A Tribute to Alain Colmerauer

JACQUES COHEN
TJX/Feldberg Professor
Volen Center
Brandeis University
Waltham, MA 02454, USA
(*e-mail:* jc@cs.brandeis.edu)

July 10, 2001

Prelude

As an invited contributor to this Festschrift honoring Alain Colmerauer, I feel compelled to give not only an account of his main research contributions, but also of my perspective on the motivations behind them. I hope that this will provide the reader with a glimpse of how a focused, tenacious, rigorous, and inventive mind like Alain's picks research problems and proceeds to solve them. The history of Prolog, the language that remains one of Alain's major accomplishments, is well documented. His paper on the *"Birth of Prolog,"* co-authored with Philippe Roussel [Alain Colmerauer and Philippe Roussel, 1970], is a highly recommended account of the circumstances that led to the development of Prolog. Bob Kowalski [Robert Kowalski, 1988] presents his views of the early history of Prolog from the automatic theorem proving perspective. Finally, my own paper on the topic [Jacques Cohen, 1988] contains material complementing Alain's and Bob's narratives. Instead of recasting already-available historical material, I have opted to present here a more personal account of Alain's contributions, acknowledging in advance the individual bias inherent in such an accounting of long-past events.

An Early Encounter: the Sixties

I have been fortunate enough to work closely with Alain Colmerauer for almost four decades. We met in the early fall of 1963, when Alain was in his early twenties, had just completed his undergraduate studies at the Institut Polytechnique de Grenoble, France, and was contemplating a doctoral degree. I had been attracted to Grenoble by the expansion that was taking place in the development of the new field of computer science, an expansion based on applied mathematics. At that time, the education in applied mathematics emphasized mostly numerical analysis and Boolean algebra. The Institute of Applied Mathematics in Grenoble (known by its French acronym IMAG) was led by Professor Jean Kuntzmann, a specialist in Boolean algebra; his closest associate was Professor Noel Gastinel, an expert in numerical analysis. In addition to Professors Kuntzmann and Gastinel, two younger researchers were prominent in the faculty at the Institute: Louis Bolliet, an experienced programmer of the earlier computers, and Bernard Vauquois, an astronomer by training, who became involved in formal language and automata theory and its application to automatic natural language translation. There were two events that made 1960s Grenoble an exciting place for research in computer science. The first was an on-going effort by European and American researchers to design and implement a standard computer language, Algol60, based on the experience gained in developing earlier languages such as Fortran. The second was the availability of a mainframe computer (IBM 7044) that offered the IMAG researchers superior opportunities for computation in those early days. Most of what we learned in Grenoble at that time consisted of novel techniques that had been recently proposed and published in *Communications* and the *Journal* of the ACM, or in contemporary monographs and dissertations. I recall that Bernard Vauquois, a member of the Algol60 original design team, and who was then directing a group on mechanical translation, was teaching a course in languages and automata theory. The course was purely theoretical, and it was our responsibility to work toward application of those theories to language implementation using the existing computers. One of the main goals of the computer science team at IMAG was to develop an Algol60 compiler for the newly acquired IBM machine. Jean-Claude Boussard was the doctoral student responsible for developing the compiler. This software would be among the first Algol compilers developed in France. Algol60 had a rigorous definition of its syntax, using what is known today as Backus-Naur-Form or BNF. In those days syntax-directed compilers were studied at the research level, but were considered inefficient for implementation in the available machines. Nevertheless, the graduate students in the compiler group of Louis Bolliet, including Alain and myself, were fascinated by the possibility of using syntactic rules as templates for building compilers.

Alain's First Research Project

The project that would tentatively make up Alain's dissertation was to design and implement an error-detecting program to be used as a front-end for a Cobol compiler. Cobol's syntax was available in BNF, but Alain's goal was to design a general syntax-directed general program. The implementation itself was to be written in Algol60 using the compiler being developed in Grenoble. At that time, there were very few papers available on compilers. The prevalent approach was that of Dijkstra's stack-machine model which eventually became available in the book of Randell and Russell [Brian Randell and Lawford J. Russell, 1964]. Through Louis Bolliet we obtained an interesting new paper by Robert W. Floyd, who was at the time working at Computer Associates, near Boston. The paper was entitled "Syntactic Analysis and Operator Precedence," and had appeared in a recent (1963) issue of the *Journal* of the ACM [Robert W. Floyd, 1963]. Basically, Floyd had found a way of automatically generating Dijkstra's stack operator precedences for languages exhibiting a special restricted form of grammar rules.

I recall that Alain became extremely interested in that paper and decided to use Floyd's precedence grammars in his syntax-driven error-detector for Cobol. This was easier said than done. Anyone familiar with that approach realizes that Cobol's syntax does not easily conform to a precedence form. There were a multitude of precedence conflicts that would have to be resolved "manually," that is, case by case. In addition, precedence grammars, being deterministic, would not allow rules with identical right-hand-sides and that characteristic was common in the existing BNF definitions for various languages.

Thus, Alain's initial problem was more difficult than he had anticipated. The reader will soon realize that what happened in bypassing that problem is typical of Alain's reaction when confronted with an obstacle. I open a parenthesis to mention a couple of Alain's traits that shed some light onto his creativity and perseverance. We used to drive through the narrow old streets of Grenoble and surrounding towns. Alain seldom took the same route twice: his innate curiosity often led him to find new ways of going from one spot to another. That temperament obviously served him well when it came to problem solving. I also remember that Alain had rented a studio apartment in one of the boulevards of Grenoble. He loved and still loves sports, sailing being one of his favorite hobbies. I recall that he had decided to build a small sailing boat in his studio; when the boat was ready, colleagues and I helped Alain remove it through a window. I am sure that he had taken the necessary precautions by careful measurements of the room, prior to undertaking the unusual project. Again it seemed that Alain had the knack for generating clever problems and then surmounting them. (Perhaps even his choice of living in a street named *Impasse des Iris* is not completely random!) Now, to continue with the precedence grammar problem to which Alain had applied himself. A paper of Griffiths and Petrick also came to our attention [Timothy V. Griffiths and Stanley Petrick, 1963]. It involved the design of a two-stack Turing Machine (TM) to estimate the efficiency of various context-free parsing methods, in particular the approaches known as top-down and bottom-up. The authors had cleverly simulated various existing parsers using TM sets of instructions. I remember Alain avidly reading that paper. The notion of nondeterminism was implicit in the TM instructions; the efficiency of various parsers was estimated by simulating the TM in a computer and by determining the number of steps needed to parse representative strings generated by typical grammars. Alain's acumen in bypassing difficulties with precedence conflicts amounted to generalizing Floyd's precedence parsers to be able to process more general languages than those advocated by Floyd. Basically, bottom-up and shift-reduce parsers replace the right-hand-side of a rule by its left-hand-side. When parsing from left to right, the element on the top of the stack is compared with the current element in the string being parsed. That introduces asymmetry, as only the stack may contain non-terminals allowing the parser to manipulate them. By using two stacks, *à la* Griffiths and Petrick, symmetry is restored since the input string being parsed is placed in a second stack, and reductions may occur in either stack. This extension allows for the parsing of a significantly more general class of languages than those defined by simple precedence. It then became possible to handle parsing and error detection based on the existing Cobol syntactic rules, virtually without "manual" intervention.

A Premonition for Prolog: the Late Sixties

Alain's JACM paper on Total Precedence summarizes his dissertation and provides a preview of the ingeniousness, simplicity and rigor he applied to the solving of a fairly complex computer science problem [Alain Colmerauer, 1970]. The dissertation can also be viewed as containing ingredients that appeared later on in the development of Prolog (e.g., parsing and nondeterminism). The language Algol68 was being perfected at about the time of the completion of Alain's thesis. He showed great interest in the two-level grammars proposed by van Wijngaarden to define the syntax of that new language [Adriaan van Wijngaarden, 1968]. Again, that formalism had some intriguing resemblance with the one that later became Prolog rules. (This is because two-level grammars can represent a potentially infinite number of context-free rules.) Around 1967, Grenoble's compiler team familiarized itself with yet another paper by Robert Floyd; in this paper, he proposed annotations to a computer language that allowed its processor to deal with (*don't know*) nondeterministic situations [Robert W. Floyd, 1963]. Floyd also proposed an implementation of his ideas using flowcharts. A group of graduate students including Alain actually implemented a version of nondeterministic Algol60 that proved successful in describing succinctly the solution of combinatorial problems. I offer the above reminiscences of the precursors to Prolog because I firmly believe that the papers of Floyd, Griffiths & Petrick, and van Wijngaarden were pivotal in establishing a frame of mind that prepared Alain for the "discovery" of Prolog. If I recall correctly, in one of the Algol68 design meetings, Alain had suggested to van Wijngaarden the incorporation of nondeterministic constructs into Algo68; the latter replied with something like: "Wait young man, one should not introduce a feature into a language just because of it being nifty." It seems that it behooved to Alain to do precisely that a decade later.

The Stay in Montreal

Around 1967, upon finishing his doctoral degree at Grenoble, Alain spent three years at the University of Montreal. The University of Montreal and other Canadian universities were attractive options for the military service of young Frenchmen, including Alain. While in Montreal, Alain decided to concentrate his research on natural language processing and artificial intelligence (AI). His interactions were with both computer scientists and linguists, and his decision to include both disciplines in his research must have been due in no small part to his wife, Colette, who is an accomplished linguist. During that period Alain developed Q-systems, now considered the precursor of Prolog's operational semantics. Essentially it consists of a set of rules specifying that a sequence of trees can be rewritten into another sequence of trees; a version of that model was subsequently used by Alain to define rigorously the semantics of later versions of Prolog. In my view, Alain was continuing to generalize his work on context-free parsers to include nondeterminism. That is a key feature in natural language processing, where dealing with ambiguities is a must. In addition, since syntax *per se* is insufficient to deal with semantics, Alain embarked on an in-depth study of theorem-proving techniques and became aware of the famous paper by Alan Robinson on resolution and unification. That paper had been published two years earlier [J. Alan Robinson, 1965]. It was Cordell Green in 1969 who had proposed using theorem-proving as an approach to problem solving [Cordell Green, 1969]. When Alain was considering returning to France in 1969 he had multiple choices. Under the sponsorship of Robert Floyd, Alain had an interview for an appointment at Stanford University. The final choice of Marseille is typical of what one would expect from Alain. He could easily have had a position in a French university with an already-established group in computer science, but he preferred to start his own department from scratch. He must also have been fascinated by the natural beauty of the neighboring towns in Provence, like Aix-en-Provence and Cassis. As a good hiker and sailor, the calanques (fjord-like inlets) in Cassis must also have exerted a strong attraction.

Back to France and the Dawn of Prolog: the Seventies

Upon returning to France and settling in Marseilles, Alain had the challenging task of starting a new computer science department at the campus of Luminy. He surrounded himself by bright students, among them Philippe Roussel, and concentrated his research on theorem-proving and computational linguistics. The computing facilities in Marseilles were minimal, and this must have been anticlimactic considering the good equipment available in Grenoble and Montreal. The struggle to obtain adequate computers is one that Alain had to face throughout his tenure as chair of the budding CS department in Luminy. In the late sixties and early seventies the artificial intelligence group at Edinburgh was among the best in Europe. A team there was exploring the potential of automatic theorem-proving techniques to problem solving. A doctoral student from Edinburgh, Bob Kowalski, had shown how to reduce substantially the search space for resolution-based theorem-provers [Robert Kowalski and Donald Kuehner, 1971]. Alain obtained funds to invite Bob for a stay in Marseilles and the cooperation between Edinburgh and Marseilles flourished. As I mentioned earlier the history of that cooperation is well documented. It is fair to state that before Alain Colmerauer's entry into the field of programming language design, there were basically two applicable paradigms: one representing imperative languages (like Algol or Fortran) and the other functional languages (like Lisp). Alain's and Bob's remarkable insight was to "invent" or "discover" a third paradigm, known as logic programming languages and represented by Prolog. Prolog's simplicity and logical foundations contributed to its worldwide acceptance and success. Perhaps the definitive indication of the worldwide acceptance has been the adoption of Prolog as the main language for the Fifth Generation Japanese Computer Project. Alain's many contributions to the elegant usage and basic features of Prolog remain valid to this day. They include the widespread use of the so-called metamorphosis grammars or the equivalent definite clause grammars, the suggestion of control annotations (e.g., the cut), the use of lazy evaluation, and so forth. Even the first usage of the now ubiquitous concatenation predicate *append* is due to Alain. As in the case of Lisp, that procedure allows a user to perform clever text processing. Furthermore, because of Prolog's capabilities for inverse computations, *append* can simulate other functions including table-lookup. I should mention here Bob Pasero and Henri Kanoui, who were among Alain's first doctoral students. Bob closely examined the problems of natural language understanding, and Henri explored the symbolic formula manipulation techniques that made use of the inverse computation capabilities of Prolog.

Prolog II: the Eighties

The above contributions, even though major, were but a prelude to the more ambitious design features that Alain incorporated into the first extension to Prolog, known as Prolog II. They were the unification of infinite trees and a new predicate for testing non-equality of those trees. Those developments took place in the late seventies and early eighties. It is admirable that Alain and his colleague Michel van Caneghem were able not only to design the new language features but to implement them in what is now recognized as a very primitive personal computer: an Apple II. One can only marvel that Alain and Michel had implemented a virtual memory system using a floppy disk in a computer with a tiny fast (RAM) memory! I recall that one of the feats that Michel and Alain incorporated in the Apple II- Prolog II system was the ability to abort computation using a *control-C* command, and to manage to safeguard all the important information prior to issuing that command. It must also have been an immense source of frustration to have a virtual memory system based on fairly unreliable floppy disks. In any case, that implementation became a forerunner of what now occurs in a PC implementation of Prolog, with all the trimmings such as debugging features and garbage collection. In the mid-eighties I had the good fortune of being invited by Alain to teach a compiler course at Luminy at the same time that John McCarthy from Stanford had been invited there to present seminars in non-monotonic logic. I recall with amazement the times in which we had the opportunity to dine together and discuss problems in computer science. At that time McCarthy had recently proposed the new area of non-monotonic logic and two of Alain's top students were writing their dissertations on that topic.

I also recall that John McCarthy mentioned that he belonged to a futuristic society, in California, that was planning to have scientists spend time in a moon colony to study problem solving capabilities in planetary environments. He suggested that Alain and Colette be included among potential candidates for the lunar sojourn. (Knowing the adventurous side of Alain and Colette I am not sure that they would have completely dismissed the idea as farfetched.) In another conversation with John, Alain mentioned that in the late sixties he had been invited to join the Stanford faculty. To this John retorted: If you had accepted that offer you probably would not have come up with Prolog! Let me return to the novel features of Prolog II, namely infinite trees and *diff* (the non-equality predicate). They are definitely the precursors of constraints as understood now in the Constraint Logic Programming (CLP) paradigm. Behind these features is the desire to extend, in a clean manner, the equality and non-equality predicates to new domains.

CLP and Prolog III: the Nineties

The addition of infinite trees and *diff* enabled Alain to carry out yet another new design, this time introducing new data types and global operators. To perform equalities in linear algebra with the necessary rigor, one must introduce first the domain of rationals, and second the capability to test the satisfiability of systems of linear equations, inequations, and disequations. In Prolog III, a harmonious design included the blending of the domain of infinite trees with that of the rationals and also with two additional domains: Booleans and a new domain called linear lists. With Prolog III, the original Prolog becomes just a special case of CLP. It is essential to point out the significant role that Alain's doctoral students had in the development of the later Prologs. The implementation of Prolog III is in itself a work of programming art. One had to be thoroughly familiar with the admirable abstract machine proposed by David H. Warren and to extend it in a substantial manner in order to incorporate Boolean processors *à la* Davis-Putnam, simplex-like solvers capable of detecting when a variable becomes bound to a specific value, special garbage collectors, and so on. All this had to be done by providing a seamless interaction among the four domains: infinite trees, rationals, Booleans, and linear lists. Furthermore, there was the implicit requirement that, when confronted with a standard Prolog program, the compiler should produce code as efficiently as a Prolog processor unencumbered by the new extensions. In the quest of designing the various components of Prolog III, Alain mentored several doctoral dissertations that delved deeply into the algorithmic components necessary to process each specific domain. Prolog III extended Prolog's applications into the realm of numerical computations, which are the staple of work in linear algebra and in operations research (OR). After designing Prolog III, Alain concentrated his interest on the extremely difficult combinatorial algorithms needed to solve scheduling problems in OR. At that time Alain also became acquainted with the work of William Older on the incorporation of interval domains to Prolog [William Older, and André Vellino, 1990]. Alain saw a renewed opportunity to extend Prolog III to deal with this new domain.

Intervals and Prolog IV: the Late Nineties

Prolog IV can be viewed as the culmination of Alain Colmerauer's efforts in language design; of course, each one of Alain's major accomplishments was considered as the culmination of the previous one! Prolog IV's balanced design surpasses that of Prolog III. The introduction of interval variables not only enables a machine-oriented rigorous definition of reals (the floating-point numbers), but it also subsumes rationals, integers, and Booleans. Essentially, Booleans are a particular case of finite domains, and the latter are a particular case of real variables expressed in floating-point notation. The introduction of interval variables also paved the way for dealing with non-linear numerical problems and enabled the practical solution of scheduling problems that previously required hours of computations. In addition, interval variables allow for proving propositions asserting the nonexistence of solutions to systems of equations or inequations in which variables are required to have values within certain ranges. When confronted with such situations, if a processor for an interval constraint language replies "no," then it implicitly provides a proof that no solution exists (i.e., assuming that the interpreter is proven correct).

A Continued Love for Puzzles

Alain has always had an interest in solving mathematical puzzles. In the realm of Booleans, Alain admired the logical puzzles of Lewis Carroll (and not incidentally, Alain's youngest daughter was named Alice). He continually seeks out new puzzles in the French daily *Le Monde* and in *Scientific American.*

In this context, one of the latest combinatorial problems holding Alain's interest is finding the squares that can be covered by a set of different smaller square [Ian Gambini, 1999]. He has proposed one of the most succinct and remarkable Prolog programs to accomplish that task efficiently.

Current Work

Of late, Alain has also applied himself to the problem of sorting interval variables [Alain Colmerauer and Noëlle Bleuzen-Guernalec, 2000]. Donald Knuth, one of the world's leaders in computer science, has devoted a full volume of his collected works to sorting. Alain's involvement in sorting brings a fresh new approach to that basic problem. Furthermore, this type of sorting has proved to be of paramount importance in scheduling, one of the most difficult tasks in Operations Research. Alain's approach achieves the complexity of classical sorting and even though no large application of the problem is presently known, it is not unlikely that it will occur in the future.

Finally, in the past year or so Alain has also renewed his interest in an extension of Prolog originally proposed by Michael Maher [Michael J. Maher, 1988]. This extension consists of allowing a user to incorporate existential and universal quantifiers to Prolog clauses. Michael had written a theoretical account and provided proofs of the validity of that approach. Alain's most recent paper demonstrates that Maher's ideas are feasible in practice and can be effectively used for solving interesting problems [Alain Colmerauer and Dao Thi-Bich-Hanh, 2000].

Postlude

One can only marvel at the breadth and scope of Alain Colmerauer's research. His contributions range from computational linguistics, to symbolic manipulation, to language design, to symbolic logic, to operations research. This range is matched by in-depth analyses of the solutions he has found for complex problems. In the history of computer science, the combination of theory and practice present in Alain's work has been achieved only rarely. And who knows – Alain may still have a couple of good tricks up his sleeve! The above paragraph brings to mind a favorite Unix fortune cookie saying (those that are used to bid farewell after each session):

Failure:
Work hard to improve
Success:
You solved the wrong problem
Work hard to improve

The metaphor aptly describes the behavior of a Prolog interpreter traversing a search tree while attempting to find all solutions to a given program. The metaphor also aptly portrays the tribulations and breakthroughs in one's professional life. Perhaps choice, genetics, or a combination of both determines the size of our own search trees and their number of failure and success nodes. Perhaps the number of success nodes could measure our perception of a person's achievements. Alain's search

tree has proven to be quite remarkable! An abundance of success nodes, far removed from the root and representing unusual achievements, are present. And certainly, his successes have made some of ours possible.
Thank you Alain.

Acknowledgements

I wish to thank Colette Colmerauer and Krzysztof Apt for the thoughtful comments they made on the original manuscript.

References

[Jacques Cohen, 1988] Jacques Cohen (Jan. 1988), A View of the Origins and Development of Prolog, *Communications ACM*, vol 31, pp 26-36.

[Alain Colmerauer, 1970] Alain Colmerauer (Jan. 1970), Total Precedence Relations, *Journal ACM 1970*, vol 17, pp 14-30.

[Alain Colmerauer and Noëlle Bleuzen-Guernalec, 2000] Alain Colmerauer and Noëlle Bleuzen-Guernalec, (2000), Optimal Narrowing of a Block of Sortings in Optimal time, *Constraints: an International Journal*, Kluwer, vol 5, pp 85–118.

[Alain Colmerauer and Philippe Roussel, 1970] Alain Colmerauer and Philippe Roussel, (1996), The Birth of Prolog, in History of Programming Languages, edited by T.J. Bergin and R. G. Gibson, ACM Press and Addison Wesley, pp 331-367.

[Alain Colmerauer and Dao Thi-Bich-Hanh, 2000] Alain Colmerauer and Dao Thi-Bich-Hanh, (2000), Expressiveness of full first order constraints in the algebra of finite or infinite trees, *Lecture Notes in Computer Science*, vol 1894, Rina Dechter (Ed.), Principles and Practice of Constraint Programming - CP 2000.

[Cordell Green, 1969] Cordell Green, (1969), Application of theorem proving to problem solving, *Proceedings of the First International Joint Conference in Artificial Intelligence*, Washington DC, pp 219-239.

[Robert W. Floyd, 1963] Robert W. Floyd, (1963), Syntactic Analysis and Operator Precedence, *Journal ACM*, vol 10, pp 316-333.

[Robert W. Floyd, 1963] Robert W. Floyd, (Oct. 1967), Nondeterministic Algorithms, *Journal ACM*, vol 14, pp 636-644.

[Ian Gambini, 1999] Ian Gambini, (Dec. 1999) Quant aux carrés carrelés, in French, Doctoral Dissertation, Université de la Méditerranée.

[Timothy V. Griffiths and Stanley Petrick, 1963] Timothy V. Griffiths and Stanley Petrick, (1965) On the Relative Efficiencies of Context-Free Grammar Recognizers, *Communications ACM*, vol. 8, no. 5, pp 289-300.

[Robert Kowalski and Donald Kuehner, 1971] Robert Kowalski and Donald Kuehner, (1971) Linear Resolution with selection function, *Artificial Intelligence*, vol 2, 1971, pp 227-260.

[Robert Kowalski, 1988] Robert Kowalski,(Jan. 1988) The Early History of Logic Programming, *Communications ACM*, vol 31, pp 38-43.

[Michael J. Maher, 1988] Michael J. Maher: (1988) Complete Axiomatizations of the Algebras of Finite, Rational and Infinite Trees, *LICS 1988*: pp 348-357.

[Guy Narboni, 1999] Guy Narboni, From Prolog III to Prolog IV, (Dec. 1999) *Constraints: An International Journal*, Vol 4 ,Number 4, pp 313-335

[William Older, and André Vellino, 1990] William Older, and André Vellino, (1990) Extending Prolog with constraint arithmetic on real intervals, *Proceedings of the Canadian Conference on Computer & Electrical Engineering*, Ottawa, Canada.

[J. Alan Robinson, 1965] J. Alan Robinson, (Jan. 1965) A Machine-Oriented Logic Based on the Resolution Principle, *Journal ACM*, vol 12, pp 23-41.

[Brian Randell and Lawford J. Russell, 1964] Brian Randell and Lawford J. Russell (1964), Algol60 Implementation, *Academic Press*.

[Adriaan van Wijngaarden, 1968] Adriaan van Wijngaarden,*et al*, (1968) Final Draft Report on the Algorithmic Language Algol68, *Mathematisch Centrum*, Amsterdam.

[Alain Colmerauer, 2001] Alain Colmerauer, URL of his Web page:
`http://www.lim.univ-mrs.fr/~colmer/`

Prolog
HERITAGE