Robert Kowalski

The Marseille-Edinburgh Connection

Volume 5, nᵒ 2-3 (2024), p. 31-37.

https://doi.org/10.5802/roia.70

Revue Ouverte d'Intelligence Artificielle
Volume 5, n° 2-3, 2024, 31-37

The Marseille-Edinburgh Connection

Robert Kowalski[a]

[a] Department of computing Imperial College, London (UK)
E-mail: r.kowalski@imperial.ac.uk.

RÉSUMÉ. — L'article rappelle le contexte scientifique de l'émergence de Prolog et évoque les échanges fructueux entre Alain Colmerauer à Marseille et Robert Kowalski à Edimbourg qui ont donné naissance à la Programmation Logique.

MOTS-CLÉS. — Prolog, Programmation Logique.

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 compilers 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.

Soon after his arrival in Marseille, Alain attracted a strong team of research students, including Robert Pasero and Philippe Roussel, to follow up the work he started on automated question-answering in Montreal. Robert worked on the natural language side of the system, and Philippe worked on automated reasoning for question-answering.

In contrast, although I was the same age as Alain, I did not complete my PhD until 1970. I arrived at the University of Edinburgh as a PhD student in the Metamathematics Unit, headed by Bernard Meltzer, in October 1967, and I started to work on automated theorem-proving. I had already studied some mathematical logic as a postgraduate student at Stanford University, but I had virtually no background or interest in computer science. Fortunately, Alan Robinson, the inventor of the resolution method of theorem-proving [17], was spending his sabbatical in Bernard's Unit, and I had privileged access to all the latest and most relevant research in the field.

At that time, the University of Edinburgh was possibly the most active centre of research in Artificial Intelligence outside of the USA. Most of the activity was in the Department of Machine Intelligence and Perception, headed by Donald Michie, who also founded the Machine Intelligence series of Workshops and Proceedings. Donald and Bernard collaborated as editors of the Proceedings, and Bernard started the Journal of Artificial Intelligence in 1970, becoming its first editor-in-chief.

The Machine Intelligence workshops attracted a wide range of AI researchers, mainly from the UK and USA. However, the 4th Machine Intelligence Workshop in 1969 was exceptional. It contained papers on automated theorem-proving by Alan Robinson and by such other leading researchers in the field as Dag Prawitz, Donald Loveland and Larry Wos. It also included the famous paper, by John McCarthy and Pat Hayes, introducing the situation calculus [14].

At the workshop, McCarthy's PhD student, Cordell Green, presented his work on the use of resolution for question-answering [7]. Foster and Elcock presented their assertional programming language Absys [5]. Pat Hayes and I presented the application of semantic trees as a method for "finding efficient rules of proof for mechanical theorem-proving" [11]. Pat Hayes went on to focus on the further development of semantic trees for his PhD thesis.

I was fortunate to meet and have discussions with many of these famous researchers. I corresponded with Cordell Green about the proofs in his paper, with Donald Loveland about the relationship between his model generation theorem-proving method and linear resolution, and with Larry Wos about Horn clauses. But I worked most closely with Pat Hayes, who joined the Metamathematics Unit as a PhD student at the same time as I did. We started to write a book based on lecture notes for a postgraduate course on automated theorem-proving [8]. The lecture notes were 105 pages long and were published internally in March 1971 as Memo 40 of the Department of Computational Logic, School of Artificial Intelligence, University of Edinburgh. Looking back at the lecture notes now, I am amazed to see that they did not contain any examples, other than purely symbolic examples without any meaning.

In the early days of research on automated theorem-proving, the focus was on proving mathematical theorems. However, Cordell Green in his 1969 PhD thesis, and in his 1969 IJCAI paper [6], showed that resolution theorem-proving could also be applied to question-answering, robot planning and "automatic programming". Unfortunately, the theorem-proving systems available at that time were opaque and very inefficient.

Despite the inefficiency of Cordell's resolution theorem-prover, his applications attracted a lot of interest in the AI research community. However, they also created a lot of opposition from researchers at MIT, who were advocating procedural, as opposed to declarative, logic-based representations of knowledge and problem-solving. The opposition was led by Seymour Papert, one of the developers of the Logo programming language, and Marvin Minsky, the founding director of the MIT AI Lab. As a PhD student under their supervision, Carl Hewitt developed the programming language Planner, as a procedural representation of knowledge [9].

Planner became very influential through its use by another MIT PhD student, Terry Winograd, who developed a natural language understanding program, SHRDLU, for a toy blocks world, where a user could both ask questions about the state of the world and issue commands to move blocks and change the state of the world [21]. SHRDLU was implemented in a combination of Planner, Lisp and PROGRAMMAR, a parsing system which interpreted grammars written in terms of programs [20]. Research on resolution-based theorem-proving went into world-wide, sharp decline.

Pat Hayes was greatly influenced by these new developments. He came back from one of his visits to John McCarthy at Stanford, disillusioned with resolution theorem-proving. Among other outcomes of his visit, he wrote a joint paper with Bruce Anderson on "The Logicians Folly" against the resolution uniform proof procedure paradigm [1]. He also wanted to rewrite most of our book. We agreed to abandon it instead.

I was not convinced that resolution was dead, believing that SL-resolution (linear resolution with selection function) addressed many of the same problems with resolution theorem-proving that had been identified by the advocates of procedural knowledge representation. As Donald Kuehner and I wrote in the conclusion section of our 1971 SL-resolution paper [12], "the amenability of SL-resolution to the application of heuristic methods suggests that, on these grounds alone, it is at least competitive with theorem-proving procedures designed solely from heuristic considerations".

Although I was convinced that the goal-oriented approach of SL-resolution could achieve similar behaviour to procedural approaches, I was struggling to understand how it could compete with procedural approaches to parsing and natural language understanding. Moreover, I couldn't understand why SHRDLU needed a separate language, PROGRAMMAR, for representing and parsing grammars, when PLANNER was supposed to be so powerful and so general. As a result, I began to investigate the possibility of representing grammars in logical form and using SL-resolution for parsing. These investigations proved to be useful when I later met Alain in the summer of 1971.

Alain knew about Cordell Green's work on question-answering; and, when he learned about my work with Donald Kuehner on SL-resolution, he decided to investigate its use for the reasoning component of the natural language question-answering system being developed in his group.

I was visiting relatives in Poland with my wife and three young daughters, when Alain sent an invitation to Edinburgh, inviting me to visit him in Marseille. Donald opened the letter and volunteered to go in my place. But he forwarded the letter to me in Poland, and I 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 exchanged ideas about theorem-proving and parsing in natural and formal languages.

I thought I knew a lot about resolution theorem-proving. But Alain knew all there was to know about parsing formal and natural languages. Back in Edinburgh, I had been working on a crude representation of grammars with explicit axioms of associativity for string concatenation, inspired by the way a mathematician might define an algebra. Alain immediately recognised the inefficiency inherent in using associativity to reason about concatenation. He suggested, instead, formalising a graph representation of strings, like the one used in his Q-Systems [3].

In our excited exchange of ideas, we discovered amazing parallels between resolution theorem-proving and parsing grammars: Both employ declarative representations of knowledge. Resolution employs logic, while parsers employ grammars. Both can solve problems bottom-up (forwards) or top-down (backwards). Even more amazing, when formal grammars are represented in formal logic, then hyper-resolution [16] behaves as a bottom-up parser/generator, and SL-resolution behaves as a top-down parser/generator. It is because of these discoveries in the summer of 1971 that 1971 is sometimes given as the year that Prolog was born.

Our exchange of ideas lasted four days and much of four nights. It was not only a meeting of minds, but a bonding of 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. 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 a 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, partly because Alain was busy with a heavy teaching load, it was during these two months that the idea of using logic as a computer programming language was born.

I had been asked by Alain to serve as external examiner for Philippe Roussel's PhD, which dealt with "formal equality" (characterised by the single axiom $x = x$) [18]. Philippe showed that many applications involving equality for which the traditional axioms of equality are intolerably inefficient can be implemented more efficiently using formal equality instead. This reminded me of the discussions I had with Alain the previous summer, and it motivated me to look for other cases where a change of representation could also lead to improved efficiency. Philippe, in turn, met with Alain and reported back ideas that arose during their own conversations. By the end of the summer of 1972, Alain's group 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 [10].

It is impossible for me to disentangle our different contributions to the idea of programming in logic. 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'm sure 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.

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. Although the reaction among several of my colleagues was enthusiastic, Pat Hayes was not very happy. He believed that I was taking credit for the thesis that "computation is controlled deduction," which he had been advocating in Edinburgh before my second visit to Marseille. On the other hand, he did not like the Prolog approach of achieving efficiency by choosing an appropriate logical representation of a problem, given the fixed behaviour of a given theorem prover. He advocated instead the opposite approach of choosing a fixed logical specification and obtaining the desired efficiency by varying the directions given to control the behaviour of the theorem-prover.

In contrast, Donald Michie and Bernard Meltzer were both very enthusiastic. Donald 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. David was primarily interested in using the Marseille implementation of Prolog and on developing it further. Maarten was interested in studying the theoretical foundations of Prolog and of logic programming more generally.

Maarten suggested that we investigate Dana Scott's fixed point semantics for functions [19] and see if we could adapt it to the semantics of logic programs [4]. When we sent the paper to Alain, he commented that the three equivalent semantics defined in the paper captured his own intuitions about the semantics of Prolog. It was one of the nicest things that Alain ever said to me.

With the support of Bernard Meltzer, I obtained a small NATO research grant to fund 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" [15], which is how they are better known today.

I also benefited from the opportunity to discuss ideas with other researchers in Edinburgh, including Alan Bundy, Rod Burstall, Michael Gordon, Robin Milner and

Gordon Plotkin, and with visitors such as Aaron Sloman, Danny Bobrow and even Carl Hewitt. Some of the visitors were especially attracted to the logic programming idea. They included Luis Pereira from Lisbon, Sten Ake Tarnlund from Stockholm, Peter Szeredi from Budapest and Maurice Bruynooghe from Leuven. I also travelled in Europe, giving talks about these new developments.

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.

Bibliography

[1] B. Anderson & P. Hayes, "The logician's folly", in *the (European) AISB Bull.*, British Comput. Soc., 1972.

[2] R. Boyer & J. Moore, "The sharing of structure in theorem proving programs", *Machine Intelligence* **7** (1972), p. 101-116.

[3] A. Colmerauer, "Les Systèmes Q ou un Formalisme pour Analyser et Synthétiser des Phrases sur Ordinateur", Tech. report, Mimeo, Montréal, 1969.

[4] M. van Emdem & R. Kowalski, "The Semantics of Predicate Logic as a Programming Language", *Journal of the ACM* **23** (1976), no. 4, p. 733-742.

[5] J. M. Foster & E. W. Elcock, "Absys1: An incremental compiler for assertions: An introduction", *Machine Intelligence* **4** (1969), p. 423-429.

[6] C. Green, "Application of theorem-proving to problem-solving", in *Proceedings of First International Joint Conference on Artificial Intelligence, IJCAI'69, Washington D.C.*, 1969, p. 219-239.

[7] ———, "Theorem proving by resolution as a basis for question-answering systems", *Machine Intelligence* **4** (1969), p. 183-205.

[8] P. J. Hayes & R. A. Kowalski, "Lecture notes on automatic theorem-proving", 1971, Metamathematics Unit Memo 40.

[9] C. Hewitt, "PLANNER: a language for proving theorems in robots", in *Proceedings of First International Joint Conference on Artificial Intelligence, IJCAI '69,Washington D.C.*, 1969, p. 295-301.

[10] R. A. Kowalski, "The Predicate Calculus as a Programming Language (abstract)", in *Proceedings of the First MFCS Symposium, Jablonna, Poland*, 1972.

[11] R. A. Kowalski & P. J. Hayes, "Semantic trees in automated theorem-proving", *Machine Intelligence* **4** (1969), p. 87-102.

[12] R. A. Kowalski & D. Kuehner, "Linear resolution with selection function", *Artif. Intell.* **2** (1971), no. 3, p. 227-260.

[13] D. W. Loveland, "Mechanical theorem-proving by model elimination", *Journal of the ACM* **15** (1968), no. 2, p. 236-251.

[14] J. P. McCarthy & J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence", *Machine Intelligence* **4** (1969), p. 463-502.

[15] F. C. Pereira & D. H. Warren, "Definite clause grammars for language analysis – a survey of the formalism and a comparison with augmented transition networks", *Artif. Intell.* **13** (1980), no. 3, p. 231-278.

[16] J. A. Robinson, "Automatic deduction with hyper-resolution", *International Journal of Computing and Mathematics* **1** (1965), p. 227-234.

[17] ———, "A machine-oriented logic based on the resolution principle", *Journal of the ACM* **12** (1965), no. 1, p. 23-41.

[18] P. Roussel, "Définition et traitement de l'égalité formelle en démonstration automatique", thèse de 3e cycle, Groupe Intelligence Artificielle, Faculté des Sciences de Luminy, Université Aix-Marseille II, France, 1972.

[19] D. Scott, "Outline of a Mathematical Theory of Computation", in *Proceedings of the the Fourth Annual Princeton Conference on Information Sciences and Systems*, 1970, p. 169-176.

[20] T. Winograd, "PROGRAMMAR: A language for writing grammars", in *AI Memo 181*, MIT, Cambridge, 1969.

[21] ———, "Understanding natural language", *Cognitive psychology* **3** (1972), no. 1, p. 1-191.

Abstract. — The article recalls the scientific context of Prolog's emergence and recalls the fruitful exchanges between Alain Colmerauer in Marseille and Robert Kowalski in Edinburgh, which gave birth to Logic Programming.

Keywords. — Prolog, Logic Programming.

Manuscrit reçu le 27 mai 2024, accepté le 12 juillet 2024.