Intervention de Robert Kowalski à la journée d'hommage, à Marseille en 2021, à Alain Colmerauer, le père de `Prolog`.

Session Chair : We will first welcome Bob Kowalski, who we can give as one of the fathers of `Prolog`. He, with his work on the procedural interpretation of the Horn clauses, had a certain paternity, and at the same time, he is a very good friend of Alain Colmerauer for a very long time. And it is also him who insisted a lot that this paternity be really recognized at the international level. Bob, floor is yours.

Robert Kowalski : We speak of Alain Colmerauer as the father of `Prolog`, when it became very well known as a language. It was in the years 1972-1973, when people started programming with it. It was the anniversary of my meeting Alain and Colette in their home in Marseille in 1971.

So, this three-year delay has been, as it were, to my own advantage. So, thank you for allowing me to present my memories of those days, and of the days immediately preceding them, because they contributed so much to our interactions together. Now, I'm doing fine, yes.

So, I will start with my visits to Marseille in the summers of 1971 and 1972. Connection failed. Connection failed, I have been told.

Yes, you are screen sharing. Hello, can you hear me ? Yes. Okay.

I was just told that the connection failed, so if that happens again, let's be prepared. Anyway, so everything started for me with these two visits to Marseille, but before that, there was an important background to my side of the story, which I'll be presenting to you today. Following the initial visits that we had together, there was another exchange which I obtained funding for through a NATO grant, and that enabled further exchanges between Edinburgh and Marseille as a whole.

That was followed by a period during which both of us were very busy with other issues, in particular the Japanese fifth generation project, which was a 10-year project focused on using logic programming techniques to exploit highly parallel computer architectures for artificial intelligence applications. During that period, there was a lot of activity in Europe and throughout the world, and the Compulogue project, which took place at the end of the 1980s, was a six-year project during which Alain and I both participated. I want to finish, however, with some speculations about the future, which I believe is very bright for `Prolog`, which was far ahead of its time.

As you already know, Alain completed his PhD in Grenoble, and this was in 1967, approximately three years before I completed my own PhD in Edinburgh. His work was in the field of formal languages, but applied to practical applications for compilers in programming languages. This work was presented to an international audience in the very prestigious Journal of the Association for Computing Machinery, which was the premier journal in those days, and it attracted an enormous amount of attention.

Texte obtenu automatiquement par logiciel de captation de licence libre Audacity, transcrit en anglais par Turboscribe : Denise Vella-Chemla, novembre 2025.

In particular, it attracted the attention of Floyd at Stanford, who invited Alain to come and tell them more about the details of Alain's work. He invited Alain to apply for a position at Stanford, but Alain then had to decide whether to stay in France or to accept this prestigious appointment in Stanford, and he indeed decided to set up his own group in Marseille, which I think was to the advantage of everyone overall. Alain was not well known for his publications. He was much more focused on getting things done rather than on reporting his work, but he was also conflicted between reporting his work in French and reporting his work in English. He needed to report his work in French in order to obtain funding, which was his primary concern, but he needed also to publish in English if he wanted to reach a wider international audience. And here is a selection.

It's almost an entire collection of his earliest publications, and you can see that his earliest publications were typically in French followed by publications later in English, and some of those English publications only took place in an assembly of papers that were printed, published retrospectively. My own PhD was approximately three years later, although we were born in the same year, 1941. I was typically three years behind Alain, but my work was in quite a different field, the field of theorem proving with the intention of proving mathematical theorems automatically.

The technique that we were developing was based on Alan Robinson's resolution principle. I interacted primarily with Pat Hayes, and we published the paper quite early in my career, so to speak, and I then also worked with a former teacher, Donald Kuehner, and we developed an improved version of resolution, which was what attracted Alain's attention and resulted in his invitation for me to visit him in 1971. So my visit to 1971 was motivated by Alain-Philippe Roussel having seen the paper with Donald Kuehner.

Alain invited me for a four-day and four-night visit to stay with him and Colette in their small apartment in Marseille and invited the rest of my family to join me, of course, and we slept, the five of us, on their living room floor when Alain and I weren't busy working deep into the night, excitedly comparing ideas, his ideas coming from formal grammars and formal linguistics and mine coming from theorem proving. And in that short period, we observed how the kind of top-down theorem proving that SL-resolution provided corresponded to top-down approaches to parsing and generating language using formal grammars. On the other hand, we also observed that you could obtain bottom-up parsers for grammars expressed in logic if you applied other kinds of resolution theorem provers.

This was an enormously exciting discovery and it provoked and motivated further interactions over the following year. Some were so that Alain invited me to come back for a longer period of two months and he and Colette arranged to find us a beautiful apartment in Cassis and my children were able to visit and stay in some of the local schools in Cassis as well. It was during that period that, one way or another, focusing on top-down execution through an SL-resolution theorem prover, the idea of logic programming and the beginnings of `Prolog` as I understand it emerged.

So what was that big idea that came to us in 1972? It was that writing axioms for a theorem prover, SL-resolution or Hyper Resolution or for that matter, any kind of theorem prover, is like writing programs for a computer. But writing programs for a computer is like driving a car. You can have a great car, but you may not know how to drive.

You might be a great driver, but you might not have a car. In order to write axioms for a theorem prover and have the results be effective, you need to write the right kind of axioms and you need to have the right kind of theorem prover. And it was that combination of how to write axioms so that they behaved like programs and how to identify the kind of theorem prover needed to behave like a computer so that the result was the execution of a computer program.

Well, I now want to turn to the background from my end of the collaboration that we had, which was the controversies that were arising over the previous three, four years or so concerning declarative versus procedural representations of knowledge. So we had on my left-hand side of the screen, we have the resolution-based approach to problem solving. And on the right-hand side of my screen, we have procedural approach to problem solving in artificial intelligence in those days.

So from my point of view, everything started with Alan Robinson's groundbreaking paper on the resolution principle, which was a tremendous contribution to both logic and artificial intelligence at that time because he devised a system of formal logic which only involved one inference principle, the resolution principle. And this resolution principle, as we later discovered, could be used either bottom-up or top-down or in a combined top-down, bottom-up manner and it was quite hard to control because you never knew what it was doing in the earliest days. That was followed around 1968 or so by Cordell Green at Stanford, discovering how to drive that car in order to answer questions by resolution theorem proving.

So he developed a number of applications which were practical applications outside of the field of mathematical theorem proving, which was the initial inspiration for the resolution principle. That work was tremendously influential and it attracted a great deal of attention among logic-based and logically inclined researchers. It also was much of the inspiration for the work that I did with Donald Kuehner.

And in particular, it inspired us to think about the application to more practical situations and to compare our approach with more heuristically-based approaches of which a planner, which I'll turn to now, was beginning to receive the attention. So while this logic-based approach was flourishing, there was an alternative approach emerging from MIT sponsored to some extent by Marvin Minsky and Seymour Papert and their PhD student, Carl Hewitt at that time. Carl developed a system he called `Planner`, which seemed to use the form of logic.

You can see he uses the terminology deductive system, but he says that it's subordinate to the hierarchical control, which was in some sense procedural. So indeed, the focus in Carl's work was on this combination of procedural control over deductive systems. And to this day, there's still, I would say, some controversy about what he really intended and to what extent his ideas were properly interpreted.

In particular, his work became influential to the development of others working at MIT, other PhD students at the time, Jerry Sussman, Terry Winograd, and Gene Charniak. They implemented a sub-language of `Planner` called `Microplanner`, and it became tremendously influential. In particular, it was influential through Terry Winograd's promotion of the work applied to understanding

natural language.

In the same way that Alain Marseille was focused on French question answering, Winograd was also interested, not so much in question answering, and maybe this is where the two approaches diverged. He was interested more in procedural control over, for example, a robot manipulating a world of blocks. That's where the blocks world comes from, from Terry Winograd's PhD thesis.

That work was enormously influential. You could see his PhD thesis was published as a book in 1972, and it has 4,299 citations as of yesterday and probably more as of today. It was so influential that in the United Kingdom, when there was a commission in order to investigate the prospects of artificial intelligence by a distinguished mathematician, Lighthill [1], that investigation was delayed by a year in order to see the results of Winograd's thesis, which everyone in Edinburgh argued would enormously influence the conclusions of the investigation to the benefit of further support for research in artificial intelligence.

Well, those two approaches clashed. The logic-based approach clashed, and to some extent, the procedural approach being promoted at MIT was very much seen to be the more appropriate way to implement and to think about artificial intelligence applications. That was the background to my visit to Marseille, and it was troubling everyone in Edinburgh at the time, and it was troubling me as I had my meetings with Alain, and I brought those troubles to him, and he came back with very practical ways of addressing those troubles.

As I said, the big idea is that writing axioms for a theorem prover is writing programs for a computer, and it's like driving a car. You have to have a good car. You have to know how to drive it.

No one is going to want to travel with you if you're a lousy driver. So `Prolog` emerged in that summer of 1972. It started out as a theorem prover for SL-resolution, but it rapidly developed into something much, much greater when it was applied to natural language understanding question answering tasks in French.

And you can see the reports that Alain and his group wrote to already in 1972, and only written up in English at a much later date. Well, when I was in Marseille in the summer of 1972, this exciting idea emerged, and I wrote back. In those days, we didn't have the internet. We didn't have email. I had to use the post system. I wrote to Edinburgh to summarize the ideas that were emerging at that time, and there was a mixed reaction.

There was so much turmoil in the world of artificial intelligence with declarative and procedural representations seen to be in conflict with one another that some of the researchers in Edinburgh, I would say, were outright hostile. But more importantly, from my point of view, the more influential leaders in Edinburgh were very enthusiastic. My supervisor, Bernard Meltzer, was very excited and encouraged others in our group to follow the directions that we had discovered.

Donald Mickey, who was widely regarded as the leader of artificial intelligence in Edinburgh, convinced two of his, well, one of his PhD students, David Warren, to change the direction of his PhD

1. Sir Michael James Lighthill (1998-2024).

research into this new, exciting field. He also encouraged one of his postdoctoral fellows, Martin Van Emden, to work with me on more theoretical issues concerning these ideas. Bernard convinced Boyer and Moore, who were also working in our group, to look at these ideas and see if they could make some contributions to them.

I wrote up my understanding of the idea in a paper which I submitted to the IFIP conference in 1974. And to show you a selection of the references at the end of the paper, you can see, along with Alain's group, Philippe Roussel's PhD, which made an important contribution in the treatment of equality. You can see the work I did a little bit later with Martin on the semantics, which Alain very kindly spoke highly of.

You can see Boyer and Moore in their use of the kind of structure-sharing representation of clauses for general theorem proving, which was later incorporated into the prologue implementations. But you can also see alternative attempts to reconcile declarative and procedural representations of knowledge. Pat Hayes, who was my best friend, we were PhD students together in Edinburgh, had an alternative view of how computation and deduction should be understood in relationship to one another.

Elcock Foster, working in Aberdeen, had their own declarative programming language, which they argued had computational uses. Eric Sandoval had his own understanding of how logic could be used for computing purposes. Well, these were such exciting times, and there was no doubt that we needed to continue our collaboration, because there was so much happening in Marseille that Edinburgh, which at that time was maybe a leading center for artificial intelligence in Europe, was nonetheless overly influenced, maybe, by its connections with American research centers.

And we were desperate to have a deeper interaction with Marseille. I was reluctant to apply for the NATO grant, which Bernard Meltzer discovered might be available, because I was active in the protest movement against the war in Vietnam when I was in the United States. But I thought it may be better to employ military funding for intellectual purposes than for the purposes which caused my concerns as a student in the United States before coming to Edinburgh.

We approached several potential referees to endorse our proposal, and it was interesting that one of the more prominent researchers in the field who had actually worked with Cordell Green refused to endorse the proposal because he said that, having seen our grant application, he said that what we were trying to do had already been tried and failed. And what he meant by that was that Cordell's work had been tried using resolution to perform not only question-answering tasks, but also tasks associated with computer programming in some form or other. And his collaborator deemed it to have been a failure.

But on the positive side, once the grant had been approved for a single year, the interactions between Edinburgh and Marseille produced all of the things we had been hoping for. In particular, David Warren, who stayed in an extended visit in Marseille, came back with tremendous results that he learned from the `Prolog` compiler that Alain and his group were implementing in Marseille. And he did his own work developing his own kind of compiler called the WAM, Warm Abstract Machine, which in a way became better known than the Marseille compiler, simply because it was

published in English at an early stage.

And the same thing is true of Alain's work on metamorphose grammars. If I show you, you see Alain eventually wrote his, well, first of all, he wrote in French. He published his ideas about writing grammars in logical form in 1970.

I'm having trouble, in 1975 in French, and then again in English in 1978. This was a book who was edited by Leona Boltz, a Polish researcher. We are a bit over time, but I don't know how many time you need.

I need a maximum of five minutes. I'm nearly finished. Okay, thank you.

So I just wanted to point out that there was this asymmetry between the work in Marseille and the work going elsewhere. On the one hand, you have Alain's work even published in English in 1978 with this modest, but relatively respectable number of citations. But then when David Warren and Fernando Pereira reworked the approach, it received a much higher number of citations, which I think is quite typical of the asymmetry which worked to the disadvantage of Alain and also of other researchers in Marseille.

Now I'm having trouble. Oh, I just lost my screen. Let's see.

Okay, let me try again. Can you see my screen ? Yes, yes. All right.

Okay, so as I say, I moved from Edinburgh to London in 1975 and started a group of my own as a result of which our contacts with Marseille grew less intense. However, we took advantage of a visit from Marseille to hold the first logic programming workshop in 1976 and other workshops were being held for related developments in logic and databases. Alain organized the first logic programming conference in Marseille.

The Japanese Fifth Generation Project discovered `Prolog` and decided to invest enormous resources into furthering the use of logic programming in `Prolog`, but unfortunately the use of a concurrent form of logic programming which didn't have the same results as might have been expected. Anyway, the Fifth Generation Project resulted in an enormous amount of international activity including activity in the European Union which involved Alain's group, our group, and a total of 13 groups throughout Europe. The European Computer Research Center in Munich which was led by HervÃ© Gallaire also focused on logic programming approaches to database applications and helped to develop the concurrent logic programming approach which Alain also spearheaded.

Well, so where are we today ? This is Alain's favorite picture if I understand correctly. He would regularly send it to me every Christmas and I often wonder, you know, what did he have in mind ? And it seems to me that maybe he was thinking about the state of computing today. But what does the future look like ? Well, I think Pure `Prolog` has an enormously an enormous future ahead of it with syntactic sugar that turns it into natural language a logical form of English and a logical form of French.

Pure `Prolog` is much closer to human language than any other computer language. Pure `Prolog` is also close to well-written legal language. If you look at legal documents you can see that they can be thought of as programs that are intended to be executed by people and are therefore people-oriented rather than machine-oriented.

And those ideas, I think, are not very different from the ideas that Alain pursued back in Marseille in his early days. And this is how I like to remember him. Looking forward into the future excited about what he'd already achieved and knowing that more was to be done not only by himself but by others.

So I welcome others to join in this vision of the future which Alain, you see, is still looking forward to even today. Thank you very much. Thank you very much.

Perhaps we have time for one question.

What is the relationship with the verification of computer programs ? I can say something about that in the legal context. If you look at programs written for representing legal documents in logical English, logical French, in the sugared up syntax that others and we have been developing, you could argue, and people have argued, that they are self-verifying. Self-verifying in the sense that they are isomorphic to the original natural language documents.

On the other hand, it is true that the legal code itself needs to be validated. But nonetheless, if you implement legal applications in other languages, then typically people working in those fields, not using logic-based languages, do say that verification is important, in ways which I think would not be necessary for logic-based approaches, because of their closeness. On the other hand, it is true, and people in the logic-based world, have been looking at different axioms for the same application.

So some axioms, which are very efficient, may look quite different from other axioms, which are closer to the specification, in which case you have to prove that efficient axioms are equivalent, in some manner, to less efficient, more obviously correct axioms. So, yes, there is a form of verification required, but that's all doable within a purely logic-based perspective. There's no need for a computer science perspective, in the sense that you can hand these problems to a logician who knows nothing about computing.