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

La présidente de la session : Nous allons tout d'abord accueillir Bob Kowalski, que l'on peut considérer comme l'un des pères de `Prolog`. Son travail sur l'interprétation procédurale des clauses de Horn lui confère une certaine paternité, et il est par ailleurs un ami de longue date d'Alain Colmerauer. C'est d'ailleurs lui qui a beaucoup insisté pour que cette paternité soit pleinement reconnue au niveau international. Bob, la parole est à vous.

Robert Kowalski : On parle d'Alain Colmerauer comme du père de `Prolog`, à l'époque où ce langage a acquis une grande notoriété. C'était entre 1972 et 1973, lorsque les gens ont commencé à programmer avec. C'était l'anniversaire de ma rencontre avec Alain et Colette chez eux à Marseille en 1971.

Ce délai de trois ans a donc finalement été, en quelque sorte, à mon avantage. Alors, merci de me permettre de partager mes souvenirs de cette époque, et des jours qui les ont immédiatement précédés, car ils ont tellement enrichi nos échanges. Je vais donner cet exposé bien volontiers.

Je vais donc commencer par mes séjours à Marseille durant les étés 1971 et 1972.

(*Voix de robot* : Connexion impossible. *Bob reprend :* Connexion impossible, m'a-t-on dit. Bonjour, vous m'entendez ? *Réponse de tous en chœur* : Oui. *Bob reprend à nouveau* : D'accord. On vient de m'informer que la connexion a échoué. Si cela se reproduit, soyons prêts.)

Quoi qu'il en soit, tout a commencé pour moi avec ces deux visites à Marseille, mais avant cela, il y a eu un contexte important que je vais vous présenter aujourd'hui. Après nos premières visites communes, j'ai obtenu un financement de l'OTAN pour un autre échange, ce qui a permis d'intensifier les échanges entre Édimbourg et Marseille.

S'en est suivie une période durant laquelle nous étions tous deux très occupés par d'autres projets, notamment le projet japonais de cinquième génération, un projet de dix ans visant à utiliser des techniques de programmation logique pour exploiter les architectures informatiques hautement parallèles dans le cadre d'applications d'intelligence artificielle. Durant cette période, l'activité était intense en Europe et dans le monde entier. Le projet `Compulog`, qui s'est déroulé à la fin des années 1980, était un projet de six ans auquel Alain et moi avons participé. Je voudrais toutefois terminer par quelques spéculations sur l'avenir, que je crois très prometteur pour `Prolog`, qui était bien en avance sur son temps.

Comme vous le savez déjà, Alain a soutenu sa thèse de doctorat à Grenoble en 1967, environ trois ans avant que je ne soutienne la mienne à Édimbourg. Ses travaux portaient sur les langages formels, appliqués à des applications pratiques pour les compilateurs de langages de programmation. Ces travaux ont été présentés à un public international dans la prestigieuse revue *Journal of the Association for Computing Machinery (JACM)*, qui était alors la revue de référence, et ont suscité

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

un vif intérêt.

En particulier, ils ont attiré l'attention de Floyd à Stanford, qui a invité Alain à venir leur en parler plus en détail. Il l'a également invité à postuler à un poste à Stanford, mais Alain devait alors choisir entre rester en France ou accepter cette prestigieuse offre. Il a finalement opté pour la création de son propre groupe à Marseille, ce qui, je pense, a été bénéfique pour tous. Alain n'était pas très connu pour ses publications. Il était bien plus concentré sur l'exécution des tâches que sur la publication de ses travaux, mais il était également tiraillé entre le français et l'anglais. Il devait publier en français pour obtenir des financements, sa principale préoccupation, mais il devait aussi publier en anglais s'il voulait toucher un public international plus large. Voici une sélection.

Il s'agit presque de l'intégralité de ses premières publications. On constate que ses premières publications étaient généralement en français, suivies plus tard par des publications en anglais, dont certaines n'ont paru que dans des recueils d'articles imprimés a posteriori. J'ai soutenu ma thèse de doctorat environ trois ans plus tard, bien que nous soyons nés la même année, en 1941. J'avais donc trois ans de retard sur Alain, mais mes travaux portaient sur un domaine très différent : la démonstration de théorèmes, avec pour objectif la démonstration automatique de théorèmes mathématiques.

La technique que nous développions était basée sur le principe de résolution de Alan Robinson [1]. J'ai principalement collaboré avec Pat Hayes, et nous avons publié l'article assez tôt dans ma carrière. J'ai ensuite travaillé avec un ancien professeur, Donald Kuehner, et nous avons mis au point une version améliorée de la résolution, ce qui a attiré l'attention d'Alain et m'a conduit à l'inviter à lui rendre visite en 1971. Ma visite de 1971 était donc motivée par la lecture par Alain de l'article avec Kuehner.

Alain m'a invité à passer quatre jours et quatre nuits chez lui et Colette, dans leur petit appartement marseillais. Ma famille était bien sûr invitée à se joindre à nous. Nous dormions tous les cinq sur le sol du salon, sauf lorsque Alain et moi travaillions tard dans la nuit, échangeant avec enthousiasme des idées : les siennes issues des grammaires et de la linguistique formelles, les miennes de la démonstration de théorèmes. Durant cette courte période, nous avons observé que le type de démonstration de théorèmes descendante proposé par SL-résolution correspondait aux approches descendantes d'analyse syntaxique et de génération de langage utilisant des grammaires formelles. Par ailleurs, nous avons également constaté qu'il était possible d'obtenir des analyseurs syntaxiques ascendants pour des grammaires exprimées en logique en appliquant d'autres types de démonstrateurs de théorèmes par résolution.

Ce fut une découverte extrêmement stimulante qui a suscité et motivé de nombreuses interactions au cours de l'année suivante. Alain m'a même invité à revenir pour un séjour de deux mois. Colette et lui nous ont trouvé un bel appartement à Cassis, et mes enfants ont pu visiter les écoles locales et y séjourner. C'est durant cette période que, d'une manière ou d'une autre, en nous concentrant sur l'exécution descendante via un démonstrateur de théorèmes SL-résolution, l'idée de la programmation logique et les prémices de `Prolog`, tel que je le comprends, ont émergé.

1. John Alan Robinson (1930-2016).

Alors, quelle était cette idée géniale qui nous est venue en 1972 ? C'était que l'écriture d'axiomes pour un démonstrateur de théorèmes, que ce soit la SL-résolution, l'hyper-résolution ou tout autre type de démonstrateur de théorèmes, revient à écrire des programmes pour un ordinateur. Mais programmer pour un ordinateur, c'est comme conduire une voiture : on peut avoir une voiture formidable sans savoir la conduire.

Vous êtes peut-être un excellent conducteur, mais vous n'avez peut-être pas de voiture. Pour écrire des axiomes pour un démonstrateur de théorèmes et obtenir des résultats efficaces, il faut écrire les bons axiomes et utiliser le bon démonstrateur de théorèmes. C'est cette combinaison – comment écrire des axiomes pour qu'ils se comportent comme des programmes et comment identifier le type de démonstrateur de théorèmes nécessaire pour qu'il se comporte comme un ordinateur et que le résultat soit l'exécution d'un programme informatique – qui a posé problème.

Je voudrais maintenant aborder le contexte de ma collaboration, à savoir les controverses apparues ces trois ou quatre dernières années concernant les représentations déclaratives et procédurales de la connaissance. À gauche de l'écran, nous avons l'approche de résolution de problèmes basée sur la résolution. Et à droite, l'approche procédurale de la résolution de problèmes en intelligence artificielle à cette époque.

De mon point de vue, tout a commencé avec l'article novateur d'Alan Robinson sur le principe de résolution, une contribution majeure à la logique et à l'intelligence artificielle de l'époque. Robinson y concevait un système de logique formelle reposant sur un seul principe d'inférence : le principe de résolution. Ce principe, comme nous l'avons découvert par la suite, pouvait être utilisé de manière ascendante, descendante, ou une combinaison des deux. À ses débuts, il était difficile à maîtriser, car son fonctionnement restait souvent inconnu. Vers 1968, Cordell Green, à Stanford, découvrit comment exploiter ce principe pour répondre à des questions par démonstration de théorèmes de résolution.

Il développa ensuite plusieurs applications pratiques, en dehors du domaine de la démonstration de théorèmes mathématiques, qui avait initialement inspiré le principe de résolution. Ces travaux eurent une influence considérable et suscitèrent un vif intérêt parmi les chercheurs en logique. Ils furent également une source d'inspiration majeure pour mes travaux avec Donald Kuehner.

Et en particulier, cela nous a incités à réfléchir à son application à des situations plus concrètes et à comparer notre approche avec des approches plus heuristiques, parmi lesquelles un planificateur, que je vais aborder maintenant, commençait à susciter l'intérêt. Ainsi, tandis que cette approche logique prospérait, une approche alternative émergeait du MIT, soutenue en partie par Marvin Minsky et Seymour Papert, ainsi que par leur doctorant de l'époque, Carl Hewitt. Carl a développé un système qu'il a appelé `Planner`, qui semblait utiliser une forme de logique.

Comme vous pouvez le constater, il utilise la terminologie d'un système déductif, mais il précise que ce système est subordonné au contrôle hiérarchique, qui était en quelque sorte procédural. En effet, les travaux de Carl portaient sur cette combinaison de contrôle procédural et de systèmes déductifs. Et aujourd'hui encore, il subsiste, à mon avis, une certaine controverse quant à ses véritables intentions et à la manière dont ses idées ont été correctement interprétées.

En particulier, ses travaux ont influencé le développement d'autres chercheurs du MIT, notamment d'autres doctorants de l'époque : Jerry Sussman [2], Terry Winograd et Gene Charniak [3]. Ils ont implémenté un sous-langage de `Planner` appelé `Microplanner`, qui a connu une influence considérable. Cette influence s'est notamment exercée grâce à la promotion, par Terry Winograd, de ces travaux appliqués à la compréhension du langage naturel.

De même qu'Alain à Marseille s'intéressait à la réponse aux questions en français, Winograd s'intéressait également à d'autres domaines, mais pas tant à la réponse aux questions elle-même. C'est peut-être là que leurs approches ont divergé. Il s'intéressait davantage au contrôle procédural, par exemple, d'un robot manipulant un univers de blocs. C'est de là que provient le concept d'"univers de blocs", issu de la thèse de doctorat de Terry Winograd.

Ce travail a eu une influence considérable. Sa thèse de doctorat a été publiée sous forme de livre en 1972 et comptait 4299 citations hier, et probablement davantage aujourd'hui. Son influence fut telle qu'au Royaume-Uni, lorsqu'une commission fut créée par l'éminent mathématicien Lighthill [4] pour étudier les perspectives de l'intelligence artificielle, cette étude fut reportée d'un an afin d'examiner les résultats de la thèse de Winograd. À Édimbourg, tous s'accordaient à dire que ces résultats influenceraient fortement les conclusions de l'étude et favoriseraient le développement de la recherche en intelligence artificielle.

Ces deux approches s'opposaient. L'approche logique était en contradiction avec l'approche procédurale promue au MIT, considérée comme la méthode la plus appropriée pour concevoir et mettre en œuvre les applications de l'intelligence artificielle. Voilà le contexte de ma visite à Marseille, et cela préoccupait tout le monde à Édimbourg à l'époque, et cela me préoccupait aussi lors de mes rencontres avec Alain. Je lui ai fait part de ces problèmes, et il est revenu avec des solutions très pratiques pour y remédier.

Comme je l'ai dit, l'idée principale est que programmer un démonstrateur de théorèmes revient à programmer un ordinateur, un peu comme conduire une voiture. Il faut avoir une bonne voiture et savoir la conduire.

Personne ne voudra voyager avec vous si vous êtes un mauvais conducteur. C'est ainsi que `Prolog` a vu le jour durant l'été 1972. Initialement conçu comme un démonstrateur de théorèmes pour la SL-résolution, il a rapidement évolué vers un domaine bien plus vaste lorsqu'il a été appliqué à la compréhension du langage naturel et à la réponse à des questions en français.

Vous pouvez d'ailleurs consulter les rapports qu'Alain et son équipe ont rédigés dès 1972, et qui n'ont été traduits en anglais que bien plus tard. Lors de mon séjour à Marseille durant l'été 1972, cette idée passionnante a émergé et j'ai répondu. À cette époque, internet n'existait pas. Il n'y avait pas de courrier électronique. Je devais utiliser le système postal. J'ai écrit à Édimbourg pour résumer les idées qui émergeaient à cette époque, et les réactions ont été mitigées.

2. Gerald Jay Sussman (1947-).
3. Eugene Charniak (1946-2023).
4. Sir Michael James Lighthill (1998-2024).

Le monde de l'intelligence artificielle était en pleine effervescence, les représentations déclaratives et procédurales étant perçues comme contradictoires. À tel point que certains chercheurs d'Édimbourg, je dirais même, étaient ouvertement hostiles. Mais, plus important encore à mes yeux, les figures les plus influentes d'Édimbourg étaient très enthousiastes. Mon directeur de thèse, Bernard Meltzer, était très enthousiaste et encourageait les autres membres de notre groupe à explorer les pistes que nous avions découvertes.

Donald Michie, considéré comme le chef de file de l'intelligence artificielle à Édimbourg, a convaincu deux de ses doctorants, enfin, l'un d'eux, David Warren, d'orienter ses recherches doctorales vers ce nouveau domaine passionnant. Il a également encouragé l'un de ses postdoctorants, Martin van Emden, à collaborer avec moi sur les aspects plus théoriques de ces idées. Bernard a quant à lui persuadé Boyer et Moore, également membres de notre groupe, d'examiner ces idées et d'envisager d'y apporter leur contribution.

J'ai exposé ma compréhension de cette idée dans un article que j'ai soumis à la conférence IFIP en 1974. Parmi les références citées à la fin de cet article, on trouve, outre le groupe d'Alain, la thèse de doctorat de Philippe Roussel, qui a apporté une contribution importante au traitement de l'égalité. On y trouve également les travaux que j'ai menés un peu plus tard avec Martin sur la sémantique, travaux qu'Alain a eu l'amabilité d'encenser.

On peut observer Boyer et Moore dans leur utilisation d'une représentation des clauses par partage de structure pour la démonstration générale de théorèmes, représentation qui a ensuite été intégrée aux implémentations de `Prolog`. Mais on peut aussi voir d'autres tentatives pour concilier les représentations déclaratives et procédurales de la connaissance. Pat Hayes, qui était mon meilleur ami – nous étions doctorants ensemble à Édimbourg – avait une vision différente de la manière dont le calcul et la déduction devaient être appréhendés l'un par rapport à l'autre.

Elcock et Foster, travaillant à Aberdeen, avaient développé leur propre langage de programmation déclaratif, dont ils affirmaient qu'il avait des applications informatiques. Eric Sandoval, quant à lui, avait sa propre conception de l'utilisation de la logique à des fins informatiques. C'était une période passionnante, et il était évident que nous devions poursuivre notre collaboration. En effet, l'effervescence qui régnait à Marseille était telle qu'Édimbourg, qui était alors sans doute un centre de référence en intelligence artificielle en Europe, était peut-être trop influencée par ses liens avec les centres de recherche américains.

Nous aspirions à une interaction plus étroite avec Marseille. J'hésitais à solliciter la bourse de l'OTAN, dont Bernard Meltzer avait découvert l'existence, car j'étais engagé dans le mouvement de protestation contre la guerre du Vietnam aux États-Unis. Mais je pensais qu'il valait mieux utiliser les fonds militaires à des fins intellectuelles plutôt qu'à des fins qui m'inquiétaient lorsque j'étais étudiant aux États-Unis avant de venir à Édimbourg.

Nous avons sollicité plusieurs référents potentiels pour appuyer notre proposition, et il est intéressant de noter que l'un des chercheurs les plus éminents du domaine, qui avait collaboré avec Cordell Green, a refusé de la soutenir. Après avoir examiné notre demande de subvention, il a estimé que

notre approche avait déjà été tentée sans succès. Il voulait dire par là que les travaux avaient été menés en utilisant la résolution pour effectuer non seulement des tâches de réponse à des questions, mais aussi des tâches liées à la programmation informatique, sous une forme ou une autre. Son collaborateur avait d'ailleurs considéré cette tentative comme un échec.

Cependant, une fois la subvention accordée pour un an, les échanges entre Édimbourg et Marseille ont porté leurs fruits. David Warren, qui a effectué un séjour prolongé à Marseille, est notamment revenu avec des résultats remarquables obtenus grâce au compilateur `Prolog` qu'Alain et son équipe développaient sur place. Et il a travaillé de son côté au développement de son propre type de compilateur appelé `WAM`, pour Warm Abstract Machine, qui, d'une certaine manière, est devenu plus connu que le compilateur de Marseille, tout simplement parce qu'il a été publié en anglais dès le début.

Il en va de même pour les travaux d'Alain sur les grammaires métamorphiques. Si je vous montre, vous verrez qu'Alain a finalement écrit ses travaux, enfin, tout d'abord, en français. Il a publié ses idées sur l'écriture des grammaires sous forme logique en 1970. J'ai un peu de mal à me souvenir de la suite : en 1975 en français, puis en anglais en 1978. Je voulais simplement souligner cette asymétrie entre les travaux menés à Marseille et ceux diffusés ailleurs. D'un côté, on a les travaux d'Alain, publiés en anglais dès 1978, avec un nombre de citations modeste, mais relativement respectable. Mais lorsque David Warren et Fernando Pereira ont retravaillé cette approche, elle a reçu un nombre de citations beaucoup plus élevé, ce qui me semble assez typique de l'asymétrie qui a joué en défaveur d'Alain et aussi d'autres chercheurs à Marseille.

Comme je le disais, j'ai quitté Édimbourg pour Londres en 1975 et j'ai créé mon propre groupe, ce qui a entraîné un éloignement de nos contacts avec Marseille. Cependant, nous avons profité d'une visite de Marseille pour organiser le premier atelier de programmation logique en 1976, et d'autres ateliers ont été organisés sur des sujets connexes liés à la logique et aux bases de données. Alain a organisé la première conférence de programmation logique à Marseille.

Le projet japonais de cinquième génération a découvert `Prolog` et a décidé d'investir des ressources considérables pour développer l'utilisation de la programmation logique avec `Prolog`. Malheureusement, l'utilisation d'une forme concurrente de programmation logique n'a pas donné les résultats escomptés. Quoi qu'il en soit, le projet de cinquième génération a engendré une intense activité internationale, notamment au sein de l'Union européenne, impliquant le groupe d'Alain, notre groupe et treize autres groupes à travers l'Europe. Le Centre européen de recherche informatique de Munich, dirigé par Hervé Gallaire, s'est également intéressé aux approches de programmation logique pour les applications de bases de données et a contribué au développement de l'approche de programmation logique concurrente, dont Alain a été l'un des pionniers.

Alors, où en sommes-nous aujourd'hui ? *(Robert Kowalski projette une reproduction du célèbre tableau montrant la Tour de Babel de Brueghel.)* Si j'ai bien compris, c'est la photo préférée d'Alain. Il me l'envoyait régulièrement chaque Noël et je me demande souvent ce qu'il avait en tête. Il me semble qu'il pensait peut-être à l'état actuel de l'informatique. Mais quel avenir nous réserve-t-il ? Eh bien, je pense que le langage `Pure Prolog`[5] a un avenir immense, grâce à des raccourcis syn-

5. autre nom pour le langage `Prolog` dans la norme Iso.

taxiques qui le transforment en langage naturel, une forme logique d'anglais et une forme logique de français.

`Pure Prolog` est bien plus proche du langage humain que n'importe quel autre langage informatique. `Pure Prolog` est également proche d'un langage juridique bien rédigé. Si l'on examine des documents juridiques, on constate qu'ils peuvent être considérés comme des programmes destinés à être exécutés par des personnes et sont donc orientés vers l'humain plutôt que vers la machine.

Et ces idées, je crois, ne sont pas très différentes de celles qu'Alain poursuivait à Marseille à ses débuts. C'est ainsi que j'aime me souvenir de lui. Envisageant l'avenir avec enthousiasme, fort de ses réalisations passées et conscient qu'il restait encore beaucoup à faire, non seulement par lui-même, mais aussi par d'autres.

J'invite donc tous ceux qui souhaitent partager cette vision d'avenir qu'Alain, comme vous le voyez, envisage encore aujourd'hui. Merci beaucoup. Merci beaucoup.

Avons-nous le temps pour une question ?

Quel est le lien avec la vérification des programmes informatiques ? Je peux aborder ce sujet dans le contexte juridique. Si l'on considère les programmes écrits pour représenter des documents juridiques en anglais logique, en français logique, dans la syntaxe simplifiée que d'autres et nous avons développée, on pourrait affirmer, et certains l'ont affirmé, qu'ils sont auto-vérifiables. Auto-vérifiables en ce sens qu'ils sont isomorphes aux documents originaux en langage naturel.

D'un autre côté, il est vrai que le code juridique lui-même doit être validé. Cependant, lorsqu'on implémente des applications juridiques dans d'autres langages, les professionnels de ces domaines, qui n'utilisent pas de langages logiques, affirment généralement que la vérification est importante, d'une manière qui, à mon avis, ne serait pas nécessaire pour les approches logiques, du fait de leur proximité du langage naturel. Par ailleurs, il est vrai que les spécialistes du domaine logique ont envisagé différents axiomes pour une même application.

Ainsi, certains axiomes très efficaces peuvent sembler très différents d'autres axiomes plus proches de la spécification. Dans ce cas, il faut prouver que les axiomes efficaces sont, d'une certaine manière, équivalents à des axiomes moins efficaces, mais plus manifestement corrects. Donc, oui, une forme de vérification est requise, mais tout cela est réalisable dans une perspective purement logique. Il n'est pas nécessaire d'adopter une perspective informatique, dans le sens où l'on ne peut confier ces problèmes à un logicien ignorant tout de l'informatique.