Un hommage à Alain Colmerauer

Jacques Cohen

Prélude

En tant que contributeur invité à ce recueil d'hommages à Alain Colmerauer, je me sens tenu de présenter non seulement ses principales contributions à la recherche, mais aussi mon point de vue sur les motivations qui les sous-tendent. J'espère que cela permettra au lecteur d'entrevoir comment un esprit concentré, tenace, rigoureux et inventif comme celui d'Alain choisit des problèmes de recherche et s'attelle à les résoudre. L'histoire de `Prolog`, le langage qui demeure l'une des réalisations majeures d'Alain, est bien documentée. Son article sur la "Naissance de `Prolog`", coécrit avec Philippe Roussel [4], est un exposé vivement recommandé des circonstances qui ont conduit au développement de `Prolog`. Bob Kowalski [12] présente son point de vue sur les débuts de `Prolog` du point de vue de la démonstration automatique de théorèmes. Enfin, mon propre article sur le sujet [1] contient des éléments complémentaires aux récits d'Alain et de Bob.

Plutôt que de reformuler des documents historiques déjà disponibles, j'ai choisi de présenter ici un récit plus personnel des contributions d'Alain, tout en reconnaissant d'avance le biais individuel inhérent à un tel compte rendu d'événements anciens.

Une première rencontre : les années soixante

J'ai eu la chance de travailler en étroite collaboration avec Alain Colmerauer pendant près de quatre décennies. Nous nous sommes rencontrés au début de l'automne 1963, alors qu'Alain avait une vingtaine d'années, venait de terminer ses études de premier cycle à l'Institut Polytechnique de Grenoble et envisageait un doctorat.

J'avais été attiré par Grenoble en raison de l'essor que connaissait le nouveau domaine de l'informatique, un essor fondé sur les mathématiques appliquées. À cette époque, l'enseignement des mathématiques appliquées mettait l'accent principalement sur l'analyse numérique et l'algèbre de Boole. L'Institut de Mathématiques Appliquées de Grenoble (IMAG) était dirigé par le professeur Jean Kuntzmann, spécialiste de l'algèbre de Boole ; son plus proche collaborateur était le professeur Noël Gastinel, expert en analyse numérique. Outre les professeurs Kuntzmann et Gastinel, deux jeunes chercheurs se distinguaient au sein du corps professoral de l'Institut : Louis Bolliet, programmeur expérimenté des premiers ordinateurs, et Bernard Vauquois, astronome de formation, qui s'est intéressé à la théorie des langages formels et des automates et à son application à la traduction automatique du langage naturel.

Deux événements ont fait de Grenoble, dans les années 1960, un lieu stimulant pour la recherche en informatique. Le premier fut un effort continu mené par des chercheurs européens et américains pour concevoir et implémenter un langage informatique standard, `Algol` 60, s'appuyant sur l'expérience acquise lors du développement de langages antérieurs tels que `Fortran`. Le second fut la mise à disposition d'un ordinateur central (IBM 7044) qui offrit aux chercheurs de l'IMAG des

Traduction du texte ici : https://arxiv.org/pdf/cs/0402058.

possibilités de calcul supérieures à cette époque.

La plupart de ce que nous avons appris à Grenoble à cette époque consistait en des techniques novatrices récemment proposées et publiées dans la revue *Communications* et dans le *Journal* de l'ACM, ou dans des monographies et thèses contemporaines. Je me souviens que Bernard Vauquois, membre de l'équipe de conception initiale d'`Algol` 60, et qui dirigeait alors un groupe de recherche en traduction automatique, donnait un cours sur la théorie des langages et des automates. Ce cours était purement théorique, et il nous incombait de travailler à l'application de ces théories à l'implémentation des langages sur les ordinateurs existants.

L'un des principaux objectifs de l'équipe d'informatique de l'IMAG était de développer un compilateur `Algol` 60 pour la machine IBM nouvellement acquise.

Jean-Claude Boussard était le doctorant chargé du développement du compilateur. Ce logiciel serait parmi les premiers compilateurs `Algol` développés en France.

`Algol` 60 possédait une définition rigoureuse de sa syntaxe, utilisant ce que l'on appelle aujourd'hui la forme de Backus-Naur ou BNF. À cette époque, les compilateurs dirigés par la syntaxe étaient étudiés au niveau de la recherche, mais étaient considérés comme inefficaces pour une implémentation sur les machines disponibles. Néanmoins, les doctorants du groupe de compilation de Louis Bolliet, dont Alain et moi-même, étions fascinés par la possibilité d'utiliser des règles syntaxiques comme modèles pour la construction de compilateurs.

Premier projet de recherche d'Alain

Le projet qui allait constituer provisoirement la thèse d'Alain consistait à concevoir et implémenter un programme de détection d'erreurs destiné à servir d'interface à un compilateur `Cobol`. La syntaxe de `Cobol` était disponible en BNF, mais l'objectif d'Alain était de concevoir un programme général dirigé par la syntaxe. L'implémentation elle-même devait être écrite en `Algol` 60 à l'aide du compilateur développé à Grenoble.

À cette époque, il existait très peu de publications sur les compilateurs. L'approche dominante était celle du modèle de machine à pile de Dijkstra, qui fut finalement présenté dans l'ouvrage de Randell et Russell [17]. Grâce à Louis Bolliet, nous avons obtenu un article intéressant de Robert W. Floyd, qui travaillait alors chez `Computer Associates`, près de Boston. L'article, intitulé "Analyse syntaxique et priorité des opérateurs", avait paru dans un numéro récent (1963) du Journal de l'ACM [7]. En résumé, Floyd avait trouvé un moyen de générer automatiquement les priorités des opérateurs de la pile de Dijkstra pour les langages présentant une forme restreinte particulière de règles grammaticales.

Je me souviens qu'Alain s'est passionné pour cet article et a décidé d'utiliser les grammaires de précédence de Floyd dans son détecteur d'erreurs syntaxique pour `Cobol`. C'était plus facile à dire qu'à faire. Quiconque connaît cette approche sait que la syntaxe de `Cobol` ne se prête pas facilement à une forme de précédence. Il y avait une multitude de conflits de précédence qu'il fallait résoudre

"manuellement", c'est-à-dire au cas par cas. De plus, les grammaires de précédence, étant déterministes, n'autorisaient pas les règles avec des seconds membres identiques et cette caractéristique était courante dans les définitions BNF existantes pour divers langages.

Ainsi, le problème initial d'Alain était plus difficile qu'il ne l'avait anticipé. Le lecteur se rendra vite compte que la manière dont il a contourné ce problème est typique de la réaction d'Alain face à un obstacle.

J'ouvre une parenthèse pour évoquer deux traits de caractère d'Alain qui éclairent sa créativité et sa persévérance. Nous avions l'habitude de sillonner les ruelles étroites de Grenoble et des villes environnantes. Alain empruntait rarement deux fois le même chemin : sa curiosité innée le poussait souvent à trouver de nouveaux itinéraires.

Ce tempérament lui était manifestement très utile pour résoudre les problèmes. Je me souviens aussi qu'Alain avait loué un studio sur un boulevard de Grenoble. Il adorait et adore toujours le sport, la voile étant l'un de ses passe-temps favoris. Je me rappelle qu'il avait décidé de construire un petit voilier dans son atelier ; une fois le bateau terminé, mes collègues et moi l'avons aidé à le sortir par une fenêtre. Je suis certain qu'il avait pris les précautions nécessaires en mesurant soigneusement la pièce, avant d'entreprendre ce projet inhabituel. Là encore, il semblait qu'Alain avait le don de générer des problèmes ingénieux et de les résoudre. (Peut-être même que son choix d'habiter dans une rue nommée *Impasse des Iris* n'est pas complètement aléatoire !)

Pour revenir au problème des grammaires de précédence auquel Alain s'était consacré, un article de Griffiths et Petrick a également attiré notre attention [10]. Il portait sur la conception d'une machine de Turing (MT) à deux piles afin d'estimer l'efficacité de diverses méthodes d'analyse syntaxique hors contexte, notamment les approches descendantes et ascendantes. Les auteurs avaient astucieusement simulé différents analyseurs syntaxiques existants à l'aide d'ensembles d'instructions de MT. Je me souviens qu'Alain avait lu cet article avec grand intérêt. La notion de non-déterminisme était implicite dans les instructions de la MT ; l'efficacité des différents analyseurs syntaxiques était estimée en simulant la MT sur ordinateur et en déterminant le nombre d'étapes nécessaires pour analyser des chaînes représentatives générées par des grammaires typiques. L'ingéniosité d'Alain pour contourner les difficultés liées aux conflits de priorité représentait la généralisation des analyseurs syntaxiques de Floyd afin de pouvoir traiter des langages plus généraux que ceux préconisés par Floyd.

En résumé, les analyseurs ascendants et par décalage-réduction remplacent le membre de droite d'une règle par son membre de gauche. Lors d'une analyse de gauche à droite, l'élément au sommet de la pile est comparé à l'élément courant de la chaîne analysée. Cela introduit une asymétrie, car seule la pile peut contenir des non-terminaux permettant à l'analyseur de les manipuler.

En utilisant deux piles, à la manière de Griffiths et Petrick, la symétrie est rétablie puisque la chaîne d'entrée analysée est placée dans une seconde pile, et les réductions peuvent avoir lieu dans l'une ou l'autre. Cette extension permet l'analyse syntaxique d'une classe de langages beaucoup plus générale que ceux définis par la simple précédence. Il est alors devenu possible de gérer l'analyse syntaxique et la détection d'erreurs à partir des règles syntaxiques `Cobol` existantes, pratiquement

sans intervention manuelle.

Une prémonition de Prolog : la fin des années soixante

L'article d'Alain sur la précédence totale, publié dans le *JACM*, résume sa thèse et donne un aperçu de l'ingéniosité, de la simplicité et de la rigueur qu'il a appliquées à la résolution d'un problème d'informatique assez complexe [2]. On peut également considérer que cette dissertation contient des éléments qui sont apparus plus tard dans le développement de Prolog (par exemple, l'analyse syntaxique et le non-déterminisme).

Le langage Algol 68 était en cours de perfectionnement à peu près au moment de l'achèvement de la thèse d'Alain. Il manifesta un grand intérêt pour les grammaires à deux niveaux proposées par van Wijngaarden pour définir la syntaxe de ce nouveau langage [18]. Là encore, ce formalisme présentait une ressemblance intéressante avec celui qui allait devenir les règles de Prolog. (Ceci s'explique par le fait que les grammaires à deux niveaux peuvent représenter un nombre potentiellement infini de règles hors contexte). Vers 1967, l'équipe de compilation de Grenoble se familiarisa avec un autre article de Robert Floyd ; dans cet article, il proposait des annotations à un langage informatique permettant à son processeur de gérer des situations non déterministes [8]. Floyd proposa également une implémentation de ses idées à l'aide d'organigrammes. Un groupe d'étudiants de troisième cycle, dont Alain, a effectivement implémenté une version non déterministe d'Algol 60 qui s'est avérée efficace pour décrire succinctement la solution de problèmes combinatoires.

Je partage ces souvenirs des précurseurs de Prolog car je suis convaincu que les travaux de Floyd, Griffiths et Petrick, et van Wijngaarden ont joué un rôle déterminant dans l'établissement d'un état d'esprit qui a préparé Alain à la "découverte" de Prolog. Si je me souviens bien, lors d'une réunion de conception d'Algol 68, Alain avait suggéré à van Wijngaarden d'intégrer des constructions non déterministes dans Algo68 ; ce dernier avait répondu quelque chose comme : "Attendez, jeune homme, on ne devrait pas introduire une fonctionnalité dans un langage simplement parce qu'elle est astucieuse." Il semble qu'il ait été judicieux pour Alain de faire précisément cela une décennie plus tard.

Le séjour à Montréal

Vers 1967, après avoir obtenu son doctorat à Grenoble, Alain passa trois ans à l'Université de Montréal. L'Université de Montréal et d'autres universités canadiennes étaient des options intéressantes pour les jeunes Français effectuant leur service militaire, dont Alain.

À Montréal, Alain décida de concentrer ses recherches sur le traitement automatique du langage naturel et l'intelligence artificielle (IA). Ses interactions se faisaient à la fois avec des informaticiens et des linguistes, et sa décision d'inclure les deux disciplines dans ses recherches doit être due en grande partie à sa femme, Colette, qui est une linguiste accomplie.

Durant cette période, Alain a développé les systèmes Q, aujourd'hui considérés comme le précurseur de la sémantique opérationnelle de Prolog. Essentiellement, il s'agit d'un ensemble de règles

spécifiant qu'une séquence d'arbres peut être réécrite en une autre séquence d'arbres ; une version de ce modèle a ensuite été utilisée par Alain pour définir rigoureusement la sémantique des versions ultérieures de `Prolog`.

À mon avis, Alain continuait de généraliser ses travaux sur les analyseurs syntaxiques hors contexte afin d'y inclure le non-déterminisme. C'est une caractéristique essentielle du traitement automatique du langage naturel, où la gestion des ambiguïtés est indispensable. De plus, la syntaxe étant insuffisante à elle seule pour traiter la sémantique, Alain s'est lancé dans une étude approfondie des techniques de démonstration de théorèmes et a pris connaissance du célèbre article d'Alan Robinson sur la résolution et l'unification. Cet article avait été publié deux ans auparavant [16]. C'est Cordell Green qui, en 1969, avait proposé d'utiliser la démonstration de théorèmes comme approche de la résolution de problèmes [6].

Lorsqu'Alain envisagea de rentrer en France en 1969, il avait plusieurs possibilités. Grâce au parrainage de Robert Floyd, Alain passa un entretien pour un poste à l'Université de Stanford. Son choix final, Marseille, est tout à fait conforme à ce que l'on attend d'Alain. Il aurait facilement pu obtenir un poste dans une université française au sein d'une équipe déjà établie en informatique, mais il préféra créer son propre département. Il devait également être fasciné par la beauté naturelle des villes voisines de Provence, comme Aix-en-Provence et Cassis. Bon randonneur et navigateur, les calanques de Cassis ont dû exercer une forte attraction sur lui.

Retour en France et l'aube de `Prolog` : les années 70

De retour en France et installé à Marseille, Alain se vit confier la tâche ardue de créer un nouveau département d'informatique sur le campus de Luminy. Il s'entoura d'étudiants brillants, parmi lesquels Philippe Roussel, et concentra ses recherches sur la démonstration de théorèmes et la linguistique informatique. Les infrastructures informatiques à Marseille étaient minimales, ce qui dut être décevant compte tenu du matériel performant disponible à Grenoble et à Montréal. La difficulté à obtenir des ordinateurs adéquats fut un problème auquel Alain dut faire face tout au long de son mandat à la tête du département d'informatique naissant de Luminy.

À la fin des années 60 et au début des années 70, le groupe d'intelligence artificielle d'Édimbourg figurait parmi les meilleurs d'Europe. Une équipe y explorait le potentiel des techniques de démonstration automatique de théorèmes pour la résolution de problèmes. Bob Kowalski, doctorant à Édimbourg, avait démontré comment réduire considérablement l'espace de recherche des démonstrateurs de théorèmes basés sur la résolution [11]. Alain obtint des fonds pour inviter Bob à un séjour à Marseille et la coopération entre Édimbourg et Marseille se développa. Comme je l'ai mentionné précédemment, l'histoire de cette coopération est bien documentée.

Il est juste d'affirmer qu'avant l'arrivée d'Alain Colmerauer dans le domaine de la conception des langages de programmation, il existait essentiellement deux paradigmes : l'un représentant les langages impératifs (comme `Algol` ou `Fortran`) et l'autre les langages fonctionnels (comme `Lisp`). L'intuition remarquable d'Alain et de Bob a été d'"inventer" ou de "découvrir" un troisième paradigme, connu sous le nom de langages de programmation logique, représenté par `Prolog`. La

simplicité et les fondements logiques de `Prolog` ont contribué à son succès et à son adoption mondiale. L'une des preuves les plus marquantes de cette adoption mondiale est sans doute l'adoption de `Prolog` comme langage principal du projet informatique japonais de cinquième génération.

Les nombreuses contributions d'Alain à l'utilisation élégante et aux fonctionnalités fondamentales de `Prolog` restent pertinentes aujourd'hui. Elles comprennent l'utilisation généralisée des grammaires de métamorphose, ou leurs équivalents, les grammaires de clauses définies, la suggestion des annotations de contrôle (par exemple, la coupure), l'utilisation de l'évaluation paresseuse, etc. Même la première utilisation du prédicat de concaténation désormais omniprésent "*append*" est due à Alain. Comme pour `Lisp`, cette procédure permet à l'utilisateur d'effectuer un traitement de texte sophistiqué. De plus, grâce aux capacités de calcul inverse de `Prolog`, "*append*" peut simuler d'autres fonctions, notamment la recherche dans une table. Je dois mentionner ici Bob Pasero et Henri Kanoui, qui furent parmi les premiers doctorants d'Alain. Bob a étudié de près les problèmes de compréhension du langage naturel, et Henri a exploré les techniques de manipulation de formules symboliques qui exploitaient les capacités de calcul inverse de `Prolog`.

`Prolog II` : les années 80

Les contributions mentionnées ci-dessus, bien qu'importantes, n'étaient qu'un prélude aux fonctionnalités de conception plus ambitieuses qu'Alain a intégrées dans la première extension de `Prolog`, connue sous le nom de `Prolog II`. Il s'agissait de l'unification des arbres infinis et d'un nouveau prédicat pour tester la non-égalité de ces arbres. Ces développements ont eu lieu à la fin des années 70 et au début des années 80. Il est admirable qu'Alain et son collègue Michel van Caneghem aient pu non seulement concevoir les nouvelles fonctionnalités du langage, mais aussi les implémenter sur ce qui est aujourd'hui considéré comme un ordinateur personnel très primitif : un `Apple II`. On ne peut qu'admirer qu'Alain et Michel aient implémenté un système de mémoire virtuelle utilisant une disquette sur un ordinateur doté d'une mémoire vive (RAM) minuscule !

Je me souviens que l'une des prouesses de Michel et Alain dans le système `Apple II-Prolog II` était la possibilité d'interrompre un calcul à l'aide de la commande `Ctrl+C`, et de sauvegarder toutes les informations importantes avant l'exécution de cette commande. L'utilisation d'un système de mémoire virtuelle basé sur des disquettes peu fiables a dû être une source de frustration immense. Quoi qu'il en soit, cette implémentation a préfiguré ce qui existe aujourd'hui dans une implémentation PC de `Prolog`, avec toutes les fonctionnalités supplémentaires telles que les fonctions de débogage et le ramasse-miettes.

Au milieu des années 80, j'ai eu la chance d'être invité par Alain à donner un cours de compilation à Luminy, au même moment où John McCarthy de Stanford y était invité pour présenter des séminaires sur la logique non monotone. Je me souviens avec émerveillement des fois où nous avons eu l'occasion de dîner ensemble et de discuter de problèmes d'informatique. À cette époque, McCarthy avait récemment proposé le nouveau domaine de la logique non monotone et deux des meilleurs étudiants d'Alain rédigeaient leurs thèses sur ce sujet.

Je me souviens aussi que John McCarthy avait mentionné son appartenance à une société futuriste, en Californie, qui envisageait d'envoyer des scientifiques séjourner dans une colonie lunaire pour

étudier les capacités de résolution de problèmes dans des environnements planétaires. Il suggéra qu'Alain et Colette soient inclus parmi les candidats potentiels pour ce séjour lunaire. (Connaissant le côté aventurier d'Alain et Colette, je ne suis pas sûr qu'ils auraient complètement rejeté l'idée comme farfelue). Lors d'une autre conversation avec John, Alain mentionna qu'à la fin des années soixante, il avait été invité à rejoindre la faculté de Stanford. Ce à quoi John rétorqua : Si tu avais accepté cette offre, tu n'aurais probablement pas inventé `Prolog`!

Revenons aux nouveautés de `Prolog II`, à savoir les arbres infinis et "`diff`" (le prédicat de non-égalité). Ce sont sans aucun doute les précurseurs des contraintes telles qu'on les conçoit aujourd'hui dans le paradigme de la programmation logique par contraintes (PLC). Derrière ces fonctionnalités se cache la volonté d'étendre, de manière propre, les prédicats d'égalité et de non-égalité à de nouveaux domaines.

`CLP` et `Prolog III` : les années 90

L'ajout des arbres infinis et de la fonction `diff` a permis à Alain de réaliser une nouvelle conception, introduisant cette fois de nouveaux types de données et des opérateurs globaux. Pour effectuer des calculs d'égalités en algèbre linéaire avec la rigueur nécessaire, il faut introduire d'abord le domaine des rationnels, puis la possibilité de tester la satisfiabilité de systèmes d'équations, d'inéquations et de déséquations [1] linéaires.

Dans `Prolog III`, une conception harmonieuse consistait à fusionner le domaine des arbres infinis avec celui des rationnels, ainsi qu'avec deux domaines supplémentaires : les booléens et un nouveau domaine appelé listes linéaires. Avec `Prolog III`, le `Prolog` original devient simplement un cas particulier de `CLP`.

Il est essentiel de souligner le rôle important joué par les doctorants d'Alain dans le développement des versions ultérieures de `Prolog`.

L'implémentation de `Prolog III` est en soi une œuvre d'art en matière de programmation.

Il fallait parfaitement maîtriser l'admirable machine abstraite proposée par David H. Warren et l'étendre considérablement afin d'y intégrer des processeurs booléens de type Davis-Putnam, des solveurs de type simplex capables de détecter lorsqu'une variable est liée à une valeur spécifique, des ramasse-miettes spéciaux, etc.

Tout cela devait être réalisé en assurant une interaction fluide entre les quatre domaines : arbres infinis, nombres rationnels, booléens et listes linéaires. De plus, il existait une exigence implicite : face à un programme `Prolog` standard, le compilateur devait produire un code aussi efficace qu'un processeur `Prolog` non encombré par les nouvelles extensions. Dans le cadre de la conception des différents composants de `Prolog III`, Alain a encadré plusieurs thèses de doctorat qui ont exploré

1. *Note de la traductrice :* Ici, la distinction est à faire entre assertion contenant seulement des signes $=$, assertions contenant des signes $=, \leqslant, \geqslant$ et assertions contenant des $\neq$.

en profondeur les composants algorithmiques nécessaires au traitement de chaque domaine spécifique.

`Prolog III` a étendu les applications de `Prolog` au domaine des calculs numériques, qui sont au cœur des travaux en algèbre linéaire et en recherche opérationnelle (RO). Après avoir conçu `Prolog III`, Alain a concentré son intérêt sur les algorithmes combinatoires extrêmement complexes nécessaires à la résolution des problèmes d'ordonnancement en RO. À cette époque, Alain a également pris connaissance des travaux de William Older sur l'intégration des domaines d'intervalles à `Prolog` [15]. Alain y a vu une nouvelle opportunité d'étendre `Prolog III` pour traiter ce nouveau domaine.

Intervalles et `Prolog IV` : la fin des années 90

`Prolog IV` peut être considéré comme l'aboutissement des efforts d'Alain Colmerauer en matière de conception de langages ; bien sûr, chacune de ses réalisations majeures était perçue comme l'aboutissement de la précédente ! La conception équilibrée de `Prolog IV` surpasse celle de `Prolog III`.

L'introduction des variables d'intervalle permet non seulement une définition rigoureuse des nombres réels (les nombres à virgule flottante) orientée machine, mais elle englobe également les nombres rationnels, les entiers et les booléens. Essentiellement, les booléens sont un cas particulier de domaines finis, et ces derniers sont un cas particulier de variables réelles exprimées en notation à virgule flottante. L'introduction des variables d'intervalle a également ouvert la voie à la résolution de problèmes numériques non linéaires et a permis la résolution pratique de problèmes d'ordonnancement qui nécessitaient auparavant des heures de calcul. De plus, les variables d'intervalle permettent de démontrer des propositions affirmant la non-existence de solutions à des systèmes d'équations ou d'inéquations dans lesquels les variables doivent prendre des valeurs dans certains intervalles. Face à de telles situations, si un processeur pour un langage de contraintes d'intervalles répond "non", il fournit implicitement une preuve qu'aucune solution n'existe (c'est-à-dire, en supposant que l'interpréteur soit correct).

Un amour persistant pour les énigmes

Alain s'est toujours intéressé à la résolution d'énigmes mathématiques. Dans le domaine de l'arithmétique booléenne, il admirait les énigmes logiques de Lewis Carroll (et, fait intéressant, sa plus jeune fille s'appelait Alice). Il recherche constamment de nouvelles énigmes dans le quotidien français *Le Monde* et dans le *Scientific American*. Dans ce contexte, l'un des derniers problèmes combinatoires qui retiennent l'attention d'Alain est la recherche des carrés qui peuvent être recouverts par un ensemble de carrés plus petits différents [9]. Il a proposé l'un des programmes `Prolog` les plus concis et remarquables pour accomplir cette tâche efficacement.

Travaux en cours

Récemment, Alain s'est également intéressé au problème du tri des variables d'intervalle [3]. Donald Knuth, l'un des plus grands noms de l'informatique, a consacré un volume entier de ses œuvres

complètes au tri. L'intérêt d'Alain pour le tri apporte une perspective nouvelle à ce problème fondamental. De plus, ce type de tri s'est avéré primordial en ordonnancement, l'une des tâches les plus complexes de la recherche opérationnelle. L'approche d'Alain atteint la complexité du tri classique et, même si aucune application majeure du problème n'est connue à ce jour, il est fort probable qu'elle se développe à l'avenir. Enfin, au cours de l'année écoulée, Alain a également renouvelé son intérêt pour une extension de `Prolog` initialement proposée par Michael Maher [13]. Cette extension consiste à permettre à une utilisatrice d'intégrer des quantificateurs existentiels et universels aux clauses `Prolog`. Michael avait rédigé un exposé théorique et fourni des preuves de la validité de cette approche. Le dernier article d'Alain démontre que les idées de Maher sont réalisables en pratique et peuvent être efficacement utilisées pour résoudre des problèmes intéressants [5].

Postlude

On ne peut qu'admirer l'étendue et la portée des recherches d'Alain Colmerauer. Ses contributions couvrent un large éventail de domaines, de la linguistique informatique à la manipulation symbolique, en passant par la conception de langages, la logique symbolique et la recherche opérationnelle. Cette diversité s'accompagne d'analyses approfondies des solutions qu'il a trouvées pour des problèmes complexes. Dans l'histoire de l'informatique, la combinaison de théorie et de pratique présente dans les travaux d'Alain est extrêmement rare. Et qui sait, Alain nous réserve peut-être encore quelques surprises !

Ce paragraphe me rappelle une phrase typique des biscuits chinois Unix (ceux qu'on utilise pour se dire au revoir après chaque session) :

> Échec :
> → Travaillez dur pour vous améliorer.
> Succès :
> → Vous avez résolu le mauvais problème.
> → Travaillez dur pour vous améliorer.

La métaphore décrit parfaitement le comportement d'un interpréteur `Prolog` parcourant un arbre de recherche pour tenter de trouver toutes les solutions à un programme donné. Elle illustre également avec justesse les tribulations et les réussites d'une vie professionnelle. Peut-être que le choix, la génétique, ou une combinaison des deux, détermine la taille de nos propres arbres de recherche et le nombre de leurs nœuds d'échec et de succès.

Peut-être que le nombre de nœuds de succès pourrait mesurer notre perception des réalisations d'une personne. L'arbre de recherche d'Alain s'est avéré tout à fait remarquable ! On y trouve une multitude de nœuds de réussite, éloignés de la racine et représentant des accomplissements exceptionnels.

Et assurément, ses succès ont rendu possibles certains des nôtres.

Merci Alain.

Remerciements : Je tiens à remercier Colette Colmerauer et Krzysztof Apt pour leurs commentaires pertinents sur le manuscrit original.

Références

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

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

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

[4] Alain Colmerauer, 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.

[5] Alain Colmerauer, 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.

[6] 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.

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

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

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

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

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

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

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

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

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

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

[17] Brian Randell, Lawford J. Russell, 1964, `Algol` 60 Implementation, *Academic Press*.

[18] Adriaan van Wijngaarden, 1968, Final Draft Report on the Algorithmic Language `Algol` 68, *Mathematisch Centrum*, Amsterdam.