Jean Rohmer

PROLOG en son temps : hier, aujourd'hui, demain. Un point de vue industriel, économique et géopolitique

Volume 5, nº 2-3 (2024), p. 39-49.

https://doi.org/10.5802/roia.71

La Revue Ouverte d'Intelligence Artificielle est membre du Centre Mersenne pour l'édition scientifique ouverte
www.centre-mersenne.org
e-ISSN : 2967-9672

Revue Ouverte d'Intelligence Artificielle
Volume 5, nº 2-3, 2024, 39-49

PROLOG en son temps : hier, aujourd'hui, demain. Un point de vue industriel, économique et géopolitique

Jean Rohmer[a]

[a] Institut Fredrik Bull. DVRC Pôle Léonard de Vinci, Paris La Défense (France)
E-mail : jean.rohmer55@gmail.com.

Résumé. — Nous examinons dans quel contexte au sens large le langage Prolog est apparu en 1972, en dehors des aspects scientifiques et académiques. Ceci concerne l'état des technologies informatiques matérielles et logicielles, l'état des entreprises informatiques et de l'économie en général, et aussi des considérations géopolitiques. Nous considérons ces points de vue dans les décennies qui ont suivi son arrivée, jusqu'à aujourd'hui. On s'aperçoit que beaucoup d'aspects plus généraux que le seul champ de l'intelligence artificielle sont concernés. On voit aussi comment Prolog et son environnement général se sont mutuellement influencés.

Mots-clés. — Prolog.

Introduction

Comme pour tout évènement important, il est intéressant d'étudier les causes et les conséquences de l'avènement de Prolog. Dans quel contexte est-il survenu, en quoi est-il le produit de ce contexte, comment ce contexte l'a accueilli. Et en retour, comment a-t-il modifié cet environnement. Il faut mener cette réflexion à toutes les époques depuis le début des années 1970. Nous ne parlerons pas ici des aspects scientifiques et académiques, mais essentiellement des aspects industriels, économiques et même géopolitiques, dit autrement, comment Prolog a-t-il été accueilli en dehors du monde académique. Nous considérons uniquement le Prolog d'origine, et non la programmation par contraintes. Une caractéristique forte de Prolog est que c'est un objet assez simple, descriptible en quelques pages, et qui pourtant concentre un nombre inhabituel de propriétés intéressantes. Autant cette richesse a dans une certaine mesure déboussolé, voire heurté, la communauté scientifique, autant, pour un ingénieur, un entrepreneur, un responsable public ou économique, ce potentiel a pu être perçu comme une grande source d'opportunités. Nous allons examiner successivement :

- Les caractéristiques uniques de Prolog, qui ont été perçues comme des avantages pertinents par des décideurs techniques et économiques.
- Le contexte technique dans lequel il est apparu, et au sein duquel il a évolué.

- Le contexte professionnel : comment a-t-il été accueilli et utilisé.
- Le contexte industriel et économique, qui, depuis les origines a soutenu ou freiné son développement.

La nature unique de Prolog : un concentré d'une douzaine de capacités intéressantes, dans un « emballage » très simple

En approfondissant la réflexion sur Prolog, on est surpris de constater combien il possède de facettes, chacune intéressante en elle-même. Voici une liste des points saillants de Prolog, certainement incomplète, et très brièvement explicitée. Ce qui est important, c'est que chacun de ces points est très parlant pour un décideur, qu'il soit utilisateur ou producteur de biens informatiques.

Représentation des informations et des connaissances

Prolog est le seul langage qui permet au sens propre de « traiter de l'information », à travers la notion de clause ou de prédicat. En Prolog, filiation avec la logique oblige, on manipule des énoncés qui sont vrais ou faux. Ce n'est pas le cas des langages de programmation habituels, qui manipulent finalement juste des chaines de caractères ou de bits, des morceaux d'inscription. Par exemple, un langage classique manipule « 23 » « octobre » « 1987 » « Paris », etc. Ces choses ne sont ni des données, ni des informations. Prolog manipule des choses du genre « Le 23 octobre 1987 à Paris, la température minimale était de 7 degrés ». C'est le seul langage de traitement de l'information. « Octobre » n'est pas une information, une liste de mots n'est pas une information. Par ailleurs, les constituants des clauses ou prédicats peuvent être des arbres, qui sont des structures extrêmement présentes dans la vie courante : toutes les listes, toutes les hiérarchies... Le monde réel est plus facile à décrire avec des arborescences qu'avec des matrices ou des tableaux.

Manipulation des informations

À travers le seul concept d'unification d'arbres, Prolog offre de quoi extraire une partie d'un tout, de rassembler deux choses pour en faire une autre, d'apparier deux choses, d'explorer des structures complexes, de rechercher un élément dans un ensemble. Ceci reposant sur la notion très naturelle de comparaison entre deux descriptions structurées. Toutes choses qui sont vitales pour la plupart des applications informatiques, dont l'essentiel des activités consiste à retrouver, déplacer, recombiner des informations.

Parallélisme et systèmes distribués

Les notions de séquences d'instructions et d'affectation de variables étant inexistantes en Prolog, le parallélisme s'exprime par défaut (« by design » comme on dit en anglais), sans effort particulier avec toutes les restrictions que l'on connaît, et qui se sont relativement corrigées avec le temps. À toutes les époques, que le matériel

soit rare et couteux ou abondant et bon marché, les constructeurs et les utilisateurs mettent en avant l'intérêt de pouvoir exploiter des architectures multiprocesseurs, et en ce sens Prolog les a intéressés. Il est par ailleurs assez simple de réaliser des systèmes distribués en Prolog : grâce à la métacircularité (identité de représentation des données et des programmes) du langage, il suffit de construire un réseau d'interpréteurs Prolog qui échangent entre eux des clauses représentant des données, des programmes ou des requêtes.

Base de Données

L'ensemble des clauses d'un programme constitue une base de données. Voilà enfin – Lisp avait montré la voie – un langage qui abat la barrière entre les programmes et les données, qui confond la manière de représenter d'un côté des données au sein du programme (« en mémoire ») et de l'autre les données dans des serveurs (« sur un disque »). Ceci est révolutionnaire, et nous débarrasse de tout le fardeau de la conversion permanente entre les deux formats, sans parler de la synchronisation nécessaire entre ces deux espaces. Si l'on ajoute à cela l'aspect déductif et récursif des clauses, Prolog offre une capacité de généralisation des notions de fichiers et de tables quasi infinie. Tout cela donna naissance à la discipline dite Datalog au milieu des année 1980.

Idée de machine spécialisée

Au moment de l'apparition de Prolog et pendant la vingtaine d'année qui a suivi, le matériel était rare et couteux. Les microprocesseurs ont mis du temps à s'imposer comme le moyen de calcul puissant et universel que nous connaissons aujourd'hui. Les architectures de machines, leurs performances, évoluaient donc lentement. Il fallait à tout prix optimiser leur fonctionnement. L'idée en cours était de créer des machines spécialement adaptées aux langages. Et donc, plus un langage était riche, puissant, plus le gain à trouver des astuces matérielles pour son exécution serait spectaculaire. D'où le concept de « Machine Langage » : on voulait « câbler » les aspects caractéristiques d'un langage, remplacer de nombreuses instructions élémentaires par des opérateurs matériels puissants dédiés. Certains ont même dû rêver à des « transistors spécialisés dans l'inférence logique ou l'unification »... Prolog étant vu à l'époque comme le plus puissant, le plus riche des langages disponibles, les espoirs placés dans des « Machines Prolog » étaient les plus grands. C'est ce qui a justifié le grand Programme d'Ordinateurs de 5ème Génération japonais, et d'autres projets similaires partout dans le monde.

Aspects Système d'Exploitation

Un système d'exploitation est le programme qui, « au fond » de la machine, supervise l'ensemble de ses activités, pilote les échanges avec le monde extérieur, les réseaux, les périphériques, l'utilisateur et ses écrans, et qui s'arrange pour lui faire faire plusieurs choses à la fois, avec les ressources matérielles disponibles, dont les multiples cœurs, les multiples processeurs. On semble loin de la programmation

logique, mais cette dernière apporte aussi des solutions. D'une part, son parallélisme inhérent s'adapte bien aux notions de multitâches, multitraitement, la capacité de métacircularité et de réflexivité de Prolog (une donnée peut devenir un programme et réciproquement) peut faire merveille pour décrire les mécanismes des superviseurs, de la gestion des interruptions, de la virtualisation. D'autre part, la puissance de Prolog en représentation des informations complexes convient bien à la description des ressources, des tâches, des utilisateurs et de leurs droits, de tout ce que doit gérer un système d'exploitation.

Capacités pour le Génie Logiciel

La discipline du génie logiciel a été très critique vis-à-vis de Prolog en tant qu'outil de développement d'applications : pas de typage, difficulté de compilation, pas de notion de modularité. Ces critiques, justifiées du point de vue des principes du génie logiciel, ont beaucoup freiné l'adoption du langage comme outil de développement de base, et ont contribué à le cantonner à des niches. Le paradoxe est que les propriétés de Prolog en font aussi un très bon outil pour développer des environnements de génie logiciel – et ceci est aussi une niche !. Beaucoup de travaux ont été consacrés à ce sujet, comme une requête sur les moteurs de recherche le prouve aisément. Un environnement de génie logiciel est une application complexe, manipulant des objets variés, très reliés entre eux, incluant des outils de compilation et de vérification de programmes, domaines dans lequel Prolog excelle. De fait, Prolog est un excellent outil pour développer des compilateurs de langages qui, dans leurs principes, sont à l'opposé des siens. Et cela n'a pas échappé aux esprits ouverts.

Prolog donne la main sur les données

Étonnamment, on peut faire un parallèle entre Prolog et l'outil de traitement de l'information le plus répandu et le plus pratiqué de manière quotidienne, le tableur Excel. Tous les deux rendent visibles et accessibles à la fois des « programmes » et des « données », respectivement les formules et les cases d'un tableau dans le cas d'Excel, et les deux types de clauses dans Prolog, avec ou sans partie droite. Dans beaucoup de cas, les formules de calcul de Excel et les clauses de Prolog présentent la même simplicité : l'arithmétique sur les nombres, les concaténations, les tris et les appartenances dans Prolog. La mise en œuvre dans Prolog, de par son caractère interprétatif, résolveur de problèmes, la capacité d'écrire des requêtes, présente la même immédiateté que celle d'un usage habituel de Excel. Enfin, et ce n'est pas le moins important, les deux permettent, de manière native, de visualiser, d'afficher les données sur lesquelles ils travaillent. Ce qui, on l'oublie en général, n'est pas du tout le cas des langages classiques. Visualiser utilement, de manière standardisée, des objets manipulés en C++ ou Java demande un supplément de travail spécifique et complexe. Cette propriété de mettre les données immédiatement au premier plan fait qu'il est possible en Prolog de réaliser en quelques dizaines de minutes des petits programmes utiles, tout comme en Excel. Et ceci sans nuire à la capacité de ces deux outils de développer des applications extrêmement sophistiquées. L'idée de réaliser un tableur

logique vient alors naturellement, et a connu plusieurs avatars, comme le projet Visilog de Bull au milieu des années 1980.

Filiation avec le langage naturel : le dialogue Homme Machine

Le fait que Prolog est né dans le contexte de la traduction automatique et du dialogue en langage naturel avec une machine a intéressé beaucoup de monde dès l'origine, puisque ce dialogue est une exigence importante de beaucoup d'applications.

Prolog n'est pas juste un outil de construction , c'est un système actif

Un langage classique est un ensemble de matériaux, de composants et de règles pour les assembler en vue de construire un système. Prolog, c'est déjà un système, c'est un moteur -– dit d'inférence –, qui agit et réagit, c'est en lui-même une application qui fait quelque chose.

Capacité à s'attaquer à des systèmes complexes

La conjonction de toutes ces propriétés intéressantes a laissé des décideurs envisager qu'avec Prolog on pourrait s'attaquer à des problèmes hors de portée des solutions informatiques classiques. Ainsi, le projet japonais d'ordinateurs de 5ème génération se proposait de résoudre des problèmes liés au système de santé, au vieillissement de la population, et à tous les domaines de la conception assistée par ordinateur. Un exemple d'application complexe largement basée sur le langage Prolog fut le système de gestion de la sécurité d'État des Jeux Olympiques d'Abertville de 1992, et un autre de nos jours est le programme NEXSIS de l'Agence Numérique de la Sécurité Civile, visant à créer un système national de gestion des opérations des pompiers.

Métacircularité et réflexivité

En Prolog, programmes, données, requêtes (questions) et réponses, c'est la même chose. Même si ces notions n'étaient pas faciles à percevoir par des non-scientifiques, des décideurs constataient bien que des problèmes difficiles et concrets étaient résolus en Prolog car ce langage avait permis d'écrire simplement au-dessus de lui-même des moteurs spécialisés dans la résolution du problème, dans l'extension ou la spécialisation de formes de raisonnement, d'expression de connaissances bien adaptées. Citons par exemple les applications de configuration technique.

Le contexte technique dans lequel Prolog est arrivé

Un univers informatique qui évoluait lentement

Prolog commença à être connu au tout début des années 1980, quand le gouvernement japonais décida de lancer son programme d'ordinateur de 5ème génération. Dans les années qui ont précédé, l'industrie informatique était structurée autour d'une

demi-douzaine de constructeurs d'ordinateurs, et largement dominée par IBM. Ils tiraient l'essentiel de leurs revenus de la vente des « mainframes », grosses machines destinées aux grandes administrations, grandes banques, assurances et sociétés industrielles. Les prix étaient très élevés, avec des marges étonnantes aujourd'hui, de l'ordre de 80%. Même si la loi de Moore avait démarré en 1971 (microprocesseur Intel 4004 à mots de 4 bits), la puissance des circuits VLSI limitait leur usage à des ordinateurs individuels « jouets », l'Apple II date de 1977, ou à des calculateurs d'automatismes industriels. Leur usage professionnel avec l'IBM PC ne date que de 1981. Les technologies et les architectures des mainframes n'évoluaient que lentement, cadencées plus par les stratégies commerciales d'exploitation des parcs de clients que par les avancées technologiques. Côté logiciels, il n'y avait pas non plus de grandes innovations après le feu d'artifice conceptuel du tout début des années 1970 avec les bases de données relationnelles, le réseau Arpanet, la programmation orientée objet et... Prolog. Et cette salve d'innovations n'avait pas encore été digérée par la profession. Bref, tout ronronnait, sans être encore réveillé par la microinformatique. Dans les milieux académiques eux-mêmes, Prolog était peu connu. L'auteur a passé 6 ans à l'I(N)RIA, le plus important institut de recherche français en informatique, de 1974 à 1980 sans jamais en entendre parler. De fait, ce n'est pas le milieu académique qui a fait connaître Prolog.

La mode des Machines Langages, une fausse bonne idée

Techniquement, l'idée -– le fantasme ? – de « machine langage » était un des thèmes les plus innovants du moment. Cela s'est avéré être une mauvaise idée. En effet, les langages de programmation, Prolog y compris, sont faits pour traiter des problèmes généraux. Dès lors, comment imaginer des machines spécialisées dans le traitement de problèmes généraux ? De fait, pendant le temps nécessaire à de petites équipes pointues pour mettre au point leur architecture spécifique, Intel sortait des processeurs VLSI généraux toujours plus rapides qui annulaient les avantages de la machine spécialisée. Ce qui détermine in fine la performance d'une application, c'est l'accès à la mémoire. Et à mémoire égale, avec les mêmes hiérarchies de caches, il y a peu de différence entre un processeur général et un processeur spécialisé. Et si l'on reporte l'idée de spécialisation sur la mémoire, le concept de mémoire associative, dont on pourrait rêver pour Prolog, n'a jamais pu être réalisé de manière économiquement viable. D'ailleurs, l'équipe de Prolog, au début des années 1980, quand on leur demandait ce que serait la machine Prolog idéale, répondait « une machine avec une très grande mémoire ». Et, comme ils en étaient privés, ils se sont fabriqués cette machine, en créant une machine virtuelle logicielle sur Apple II (limité à 64 K) en l'étendant sur le lecteur de disque souple. Cette solution, qui ne pouvait apparaître que comme pure folie aux yeux des techniciens chevronnés, s'avéra pourtant capable d'exécuter des applications avec des temps de réponse honorables. Prolog relança cette idée de Machine Langage, en premier lieu au sein du projet japonais d'ordinateurs de 5ème génération, et aussi à l'ECRC, centre de recherche commun à Bull, ICL et Siemens, avec le projet KCM (Knowledge Crunching Machine).

Le contexte professionnel dans lequel Prolog a évolué

Par contexte professionnel, nous entendons la manière dont Prolog a été reçu par les ingénieurs chargés de développer des applications.

Prolog et le Génie Logiciel

En période de faible innovation, les professionnels s'intéressent aux méthodes de travail, ils optimisent l'usage de technologies qui elles-mêmes restent stables. C'était le cas autour de 1980. (À l'inverse l'irruption d'Internet et l'arrivée des microprocesseurs puissants vers 1995 ont engendré une phase de grand « bricolage » généralisé pour construire à marche forcée la première génération assez frustre des logiciels du web). Les années 1970 ont vu apparaître le concept de « crise du logiciel », le génie logiciel, étant censé y répondre avec la même rigueur et la même solidité que le génie civil, le génie mécanique, le génie maritime ... Les règles du génie logiciel s'imposèrent à la fin des années 1970, avec entre autres, l'apparition des ateliers de génie logiciel, environnements informatiques destinés à gérer les développements des applications, donc les programmes, avec la volonté d'industrialiser les métiers de l'informatique. Comme expliqué plus haut, Prolog, arrivant dans ce paysage, recueillait peu de considération aux yeux de cette discipline naissante. Il était aussi difficile à cerner, entre d'un côté sa prétention à attaquer des problèmes complexes comme la traduction automatique, et d'un autre côté le fait que des enfants, sur leur Apple II, étaient tout à fait capables d'écrire des programmes qui faisaient quelque chose d'intéressant. Cet écart entre deux visions de l'informatique freina en permanence l'adoption de Prolog par la profession. Et il fallait avoir du courage ou des circonstances particulières pour se lancer dans de grandes réalisations en Prolog, lesquelles pourtant en général furent très convaincantes. Des environnements de développement d'applications Prolog reprenant toutes les exigences des ateliers de génie logiciel furent néanmoins commercialisés, comme le produit SP-Prolog de Bull, et tournaient sur de puissantes stations de travail graphiques.

La perception des faiblesses de Prolog

Aux yeux des inconditionnels, il faudrait plutôt dire : « comment certains aspects de Prolog ont été perçus comme des faiblesses ». De fait, beaucoup d'obstacles se sont présentés et continuent de se présenter pour un usage de Prolog à grande échelle, parmi lesquels on peut citer :

- Difficulté de trouver des programmeurs compétents.
- Difficulté de comprendre et prévoir ce que fait le « moteur » de Prolog, qui peut partir en boucle, ou engendrer des temps de réponses très longs et difficiles à prévoir.
- Difficulté d'interfacer Prolog avec des programmes développés de manière classique.
- Certes on peut faire des prouesses avec Prolog, comme écrire des moteurs de raisonnement, d'inférence, qui exhibent des propriétés logiques plus riches

ou plus maîtrisées que le langage de base, mais cela exige des programmeurs virtuoses.
- Dans ce même sens, les professionnels sont désorientés face à un langage qui peut être vu à la fois comme une Ferrari, réservée à une élite, ou une Dinky Toys, que l'on peut faire rouler sur le parquet.
- Finalement, une vision de Prolog comme un « couteau suisse », certes utile dans des circonstances particulières rares, mais qui sera toujours surpassé par de solutions classiques face à un problème stable et bien identifié.

La question des améliorations à apporter à Prolog

Dès le départ, beaucoup de travaux universitaires et industriels ont visé à amplifier les bonnes propriétés de Prolog ou bien à en corriger les défauts. On peut distinguer à ce sujet trois grandes écoles qui se sont révélées au cours du temps :

- Une école « canal historique » : on ne touche pas à la pureté du langage, on fait avec, tel qu'il est, car on est encore loin d'en avoir tiré toute la substantifique moelle, surtout si l'on s'applique à exploiter la puissance des machines d'aujourd'hui, meilleures en rapport performance / prix d'un ordre supérieur à un milliard. Certains vont même encore plus loin : face à une question en anglais comme « How to teach the "cut" » – une des fonctions de Prolog pas très propre et pas très claire – ils répondent juste : « Don't ! »
- Une école « assimilatrice », qui consiste à ajouter à Prolog tout ce qui lui manque par rapport aux autres langages : des types, de la programmation objet, des modules, de la programmation structurée, des liaisons efficaces avec les bases de données, les formats de données standard, des interfaces graphiques, du calcul arithmétique efficace, etc. Beaucoup d'efforts ont été faits en ce sens dans les Prolog successifs et en particulier dans ceux qui ont survécu. D'un côté, cela permet au langage de s'adapter à l'évolution des environnements informatiques, de l'autre, cela conduit certains à déplorer un affadissement, un alourdissement du langage d'origine, perdu au milieu d'une documentation devenue aussi épaisse et aussi rugueuse que celle des langages les plus répandus.
- Enfin, une école « pragmatique » préfère ne garder de Prolog que quelques principes de base et les enfouir dans des librairies appelées en Java, Python ou autre. Les structures de Prolog deviennent des classes prédéfinies de ces langages, et les opérations (unification, résolution) sont des méthodes sur ces classes. Cela a du sens aussi, et permet en particulier un bon contrôle du fonctionnement d'un tel moteur livré en « pièces détachées ». La grande contrepartie en est la disparition physique du langage, on ne le voit plus. Il n'en subsiste qu'une documentation souvent absconse d'une librairie en annexe du manuel de référence. Adieu l'émerveillement des premiers pas en Prolog.

Prolog, le dernier des Mohicans pour les systèmes à base de connaissances

Beaucoup d'entreprises sont aujourd'hui légitimement attirées par les prouesses de l'IA connexionniste. Mais elles découvrent vite que les problèmes qu'elles veulent traiter relèvent aussi pour une bonne part de l'IA symbolique, de la gestion de connaissances métier, de la programmation par règles, bref, de tout ce que concrétisait la notion de système expert. Or tous leurs très bons environnements de développement graphiques sur stations de travail développés autour des années 1990 ont disparu, et n'ont été remplacés par rien d'approchant. Aujourd'hui, le seul outil disponible pour se lancer dans un tel développement qui nous reste est... Prolog, avec toute sa fragilité existentielle, quand on considère les faibles effectifs des équipes qui continuent à développer et maintenir ses implémentations solides. Un paradoxe de plus autour de la programmation logique.

Le contexte économique, géopolitique dans lequel Prolog est né et s'est développé

Pour synthétiser, on peut distinguer trois grandes périodes -– et peut-être une quatrième à venir – dans lesquelles le phénomène Prolog a vécu : les années 1970, les années 1980, la longue période de 1992 à 2022, et enfin le futur proche. Nous avons analysé plus haut le contexte technique des années 1970 : un monde informatique plutôt stable, où la naissance de Prolog est passée assez inaperçue.

Les années 1980 : la première fièvre « High-Tech » au niveau mondial, dont Prolog, via le Japon, a été un des catalyseurs

Ce furent les dix glorieuses de la technologie informatique. Tout commença par la vision stratégique du gouvernement japonais, qui, dans son effort entrepris après la défaite de 1945, se lança dans une série de plans nationaux concernant la sidérurgie, l'automobile, les produits de grande consommation et les composants électroniques, puis considéra dans les années 1970 que l'avenir était au traitement de l'information. Ils décidèrent alors de lancer leur industrie nationale, avec l'aide de scientifiques du monde entier, dans un une révolution technologique de rupture — la 5^{e} génération –, mêlant les machines multiprocesseurs et les approches de l'intelligence artificielle. Et ils choisirent donc Prolog comme le cœur conceptuel, l'ADN, en quelque sorte le langage machine de cette révolution. On peut raisonnablement conjecturer qu'ils choisirent Prolog car ce n'était pas Lisp, le langage par excellence de tout ce qui se passait en IA aux USA. Bien sûr ils furent eux aussi séduits par le concentré unique de propriétés intéressantes que constituait Prolog, mais la volonté d'adopter une voie originale qui se démarque du mainstream made in USA les conforta dans le choix de prendre ce pari audacieux, qui était totalement inédit à l'échelle de la planète. On sait que cette initiative provoqua des réactions et répliques dans le monde entier, avec le projet MCC aux USA, Alvey en Grande-Bretagne, et Esprit en Europe. L'initiative européenne eut une vigueur particulière. Initiée par douze industriels de l'informatique et de

l'électronique, elle inaugura l'effort de recherche concertée de l'Union Européenne, avec une continuité totale dans le temps, dont le programme Horizon Europe est le descendant en ligne directe. Dans le même ordre d'idée, les constructeurs d'ordinateurs Bull, ICL et Siemens créèrent un important centre de recherche commun à Munich, l'ECRC, dont le programme de travail était totalement centré sur la programmation logique. Cet afflux de moyens financiers rencontra un terrain technique favorable, avec l'apparition des stations de travail graphiques des constructeurs comme Sun et Apollo, qui offrirent aux chercheurs et ingénieurs des moyens de travail inédits et fort couteux. Toutes les grandes entreprises mondiales créèrent leurs équipes d'intelligence artificielle, et réalisèrent des applications très significatives, en particulier des systèmes experts, vus comme le meilleur moyen d'amplifier les savoir-faire et la compétitivité des entreprises et des administrations. En même temps, les effectifs d'étudiants et de chercheurs formés à l'informatique augmentèrent suffisamment pour alimenter toutes ces initiatives, dont Prolog fut un déclencheur déterminant.

De 1992 à 2022 : l'effacement progressif

Contrairement à ce qui est communément dit, le point de départ de cette chute n'est pas l'échec du Plan 5ème Génération Japonais, – dû en partie à une trop grande focalisation sur le matériel – mais la grave crise économique mondiale des années 1992 et 1993, aujourd'hui oubliée. En effet, qu'ils soient réalisés en Prolog ou avec d'autres outils, les systèmes experts de l'époque donnaient satisfaction. Simplement, leur développement était très couteux et mobilisait beaucoup de matière grise dans les entreprises, qui, dans ce contexte de crise économique, eurent vite ensuite d'autres priorités et d'autres terrains d'investissement : le passage au tout VLSI, la montée d'Internet, la multiplication des données, l'apprentissage profond, les réseaux sociaux, les téléphones mobiles. L'IA fut vite oubliée dans les entreprises, et de moins en moins enseignée à l'Université.

À partir de 2022 : un nouveau printemps ?

Peut-il arriver à Prolog ce qui est arrivé aux réseaux de neurones ? Ces derniers, issus de principes eux aussi assez simples et assez anciens, n'ont trouvé leur utilité que le jour où on a su et pu les alimenter, en mode apprentissage, par suffisamment de données et de puissance de calcul. La moindre des choses serait d'imaginer et de tenter des expériences d'usage de la programmation logique qui sauraient tirer parti de l'existence d'un univers numérisé qui n'avait pas le moindre début de réalisation il y a 40 ou 50 ans. Le changement d'échelle en termes de données et de puissance de calcul disponibles – pensons à tous les téléphones mobiles – se mesure sans doute pour un coût constant en milliards de fois. À comparer à la liaison à 30 caractères par seconde que le GIA de Luminy utilisait pour se connecter en temps partagé à l'IBM 360-67 de l'Université de Grenoble... Ce qui a été osé et réussi récemment avec les grands modèles de langages à base de réseaux de neurones dans des systèmes comme ChatGPT3, peut-il être transposé bientôt avec de la programmation logique ? D'autant que Prolog visait déjà à l'origine le dialogue en langage naturel avec des machines. Et

on peut imaginer que ChatGPT3 ouvre paradoxalement la voie à un retour en force de la logique, à une envie de logique. En effet la promesse – pour reprendre une expression contemporaine – de cet outil est de permettre un diaLOGue avec un robot omniscient. Spontanément, nous attendons de lui qu'il raisonne. Or, il ne raisonne pas. Les tenants de la programmation logique sauront-ils s'engouffrer dans cette brèche ? L'avenir nous le dira.

Abstract. — We consider the origins, development and possible future of Prolog, not from a scientific or academic point view, but we focus on the general context – from the 70's until today – in which Prolog emerged, how it was received, what were its interaction with this context, from various points of view: technical, industrial, economical, and even geopolitical. What where the other available technologies, how professional developers considered Prolog, what was the attitude of the main IT companies and their customers. We examine how Prolog was the product of this context, and reciprocally how it influenced it. Depending on the general climate of economy, the capabilities of the available hardware, the current concerns of the sofware community, Prolog experienced a favorable climate or not. We analyze also the numerous specific facets of Prolog which made it such an appealing concept for many people and institutions, entailing very significant intiatives and projects.

Keywords. — Prolog.

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