Christian Boitet

Souvenirs d'Alain Colmerauer : TAUM, Prolog, TA

Volume 5, nº 2-3 (2024), p. 51-63.

https://doi.org/10.5802/roia.72

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, n° 2-3, 2024, 51-63

Souvenirs d'Alain Colmerauer : TAUM, Prolog, TA

Christian Boitet[a]

[a] Université Grenoble Alpes (UGA) et CNRS Laboratoire d'Informatique de Grenoble (LIG), UMR5217 (France)
E-mail : Christian.Boitet@imag.fr.

Résumé. — Je rassemble ici quelques souvenirs éblouis d'Alain Colmerauer. Sans encore le connaître, j'ai étudié et utilisé ses « systèmes-Q » à TAUM (UdM), là où il les avait inventés, durant ma première année de thèse. Je retrace ses apports à la TA (par pivot conceptuel hybride à la Shaumyan), à la programmation logique (avec BenZaken, nous enseignions dans nos cours de logique « comment et pourquoi Prolog marche »), et à la linguistique computationelle (dont sa formalisation originale avec les « quantificateurs à trois branches »).

Mots-clés. — Alain Colmerauer, systèmes-Q, programmation logique, Prolog, traduction automatique.

1. 1970-77, avant Prolog

1.1. Premiers souvenirs

Mes premiers souvenirs d'Alain Colmerauer datent de 1970-71 : je venais d'arriver au CETA, frais émoulu de l'X, sorti dans la « botte recherche ». Le CETA était un labo propre du CNRS depuis 1960-61. Il était célèbre par son système de TA russe-français. Je m'intéressais déjà beaucoup aux langues (dont le russe), aux mathématiques et à l'informatique. Mais j'avais à peine été initié à l'informatique, alors que les étudiants de Grenoble avaient eu des cours/TD/TP déjà très avancés, en particulier ceux de l'ENSIMAG. J'avais simplement suivi les conseils de Laurent Schwartz (le meilleur professeur de mathématiques que j'ai eu, inventeur des distributions, etc., avec qui j'étais en option) et de Jean Train (professeur de russe à l'INaLCO et à l'X), tous deux connaissant et appréciant Bernard Vauquois, et pris contact avec ce dernier, qui me « recruta ».

Bref, en octobre 1970, j'arrivai au CETA comme attaché de recherche au CNRS. J'avais 7 ans pour préparer et soutenir une thèse d'État[(1)], la 1re année étant consacrée à un DEA, et la suivante à une année dans un labo à Moscou... qui se transforma en une année à Montréal où B. Vauquois fit une année sabbatique (1971-72).

Ma première année, à Grenoble, fut passionnante. En DEA, ce fut la découverte, grâce à des professeurs tous réellement géniaux,

(1) ce qui fut fait en 1976.

(1) des grammaires et des automates (Claude BenZaken),
(2) de la calculabilité (Georges Werner),
(3) des bases de données (Jean-Raymond Abrial)[2],
(4) du TALN (Bernard Vauquois),
(5) et de l'activité de recherche (séminaires de DEA de Jean Kuntzmann).

Au labo, ce fut l'initiation :

(1) à la TA : étude du système complet russe-français (Nicolas Nédobejkine, Lydie Torre, Monique Axtmeyer, Irina Poroxovthikova, Jacqueline Gaudey), analyseur de l'allemand (Roger Stiers) ;
(2) aux métalangages (grammaires en FNC + attributs, grammaires transformationnelles) – devenus LSPL (Langages Spécialisés pour la Programmation Linguistique) ;
(3) aux algorithmes sous-jacents[3] ;
(4) à la linguistique (travaux de I. Mel'tchuk, Gladkij, O.S. Kulagina, V. Rosenzweig, Barchtchov & Xomyakov, Apresyan ; N. Chomsky, D. Hayes, P. Sgall, etc.) ;
(5) et... aux systèmes-Q d'Alain Colmerauer [1], créés en 1967 à Montréal !

1.2. Pourquoi essayer d'utiliser les systèmes-Q à Grenoble en 1970-71 ?

En 1970-71, le CETA fit ses dernières expériences de TA russe-français sur IBM-7044. Cette machine fut donnée à l'université de Clermont-Ferrand, pour être remplacée au centre de calcul de Grenoble (CICG) par un IBM-360/67 sous VM/CMS. Les toutes dernières expériences se firent d'ailleurs quand l'IBM-7044 était déjà à Clermont-Ferrand.

Les résultats obtenus sur 1 600 pages du corpus de la Rand (400K mots), portant sur les moteurs de fusées et les satellites, étaient tout à fait excellents, par rapport au paradigme opérationnel, qui était la *TA pour l'accès à l'information*.

Le système du CETA avait le même objectif que celui du système russe-anglais du groupe GAT[4] [6], qui fut utilisé à Euratom (Ispra) jusqu'à 1973, et ensuite remplacé par Systran[5].

---

[2] Il faisait un cours ENSIMAG+DEA, ayant inventé 2 ans avant le concept de BD et étant en train de l'implémenter dans le projet SOCRATE.

[3]
- algorithme d'analyse de Cocke, généralisé aux grammaires FNC (en forme normale de Chomsky, avec des attributs synthétisés et des « variables véhiculaires » fonctionnant comme des attentes de dépendances à longue distance) et optimisé par Gérard Veillon ;
- algorithme de transformation vers des arbres de dépendances avec attributs ;
- algorithme de choix (désambiguïsation automatique) par préférences, dû à M. Berthod ;
- nouveau formalisme pour les grammaires de dépendances, dû à Jacques Courtin, utilisant des index négatifs et positifs pour représenter l'ordre linéaire.

[4] Georgetown Automatic Translation

[5] qui avait été développé en macro-assembleur 360.

Le GAT tournait sur une IBM-7044, elle aussi remplacée à Ispra par une IBM-360/67 vers 1968-69. Mais il n'y eut pas de portage, on fit simplement tourner le GAT sous l'émulateur de l'IBM-7044.

Les résultats du CETA (vers le français) étaient bien meilleurs et, rétrospectivement, le CETA aurait été bien inspiré de faire de même, *i.e.* de ne pas « tout jeter », mais de développer une exploitation opérationnelle (qu'espéraient les financeurs, CNRS, DRET et CEDOCAR), ainsi que des outils d'évaluation et d'amélioration. Bien entendu, il fallait développer en parallèle un nouveau système.

Malheureusement, vers 1968-70, le CETA subit le contrecoup de mai 68, et fut divisé en 3 URA (unités de recherche associées), l'une (avec G. Veillon et J. Courtin) se consacrant à l'IA, qui commençait juste en France, la seconde à l'indexation de documents et à la recherche d'information (J. Rouault), et la troisième, le GETA, à la TA automatique et assistée (TAO), toujours avec B. Vauquois.

En 1970, les objectifs et principes du CETA devinrent :

- TA du réviseur : atteindre une qualité « révisable », *i.e.* équivalente à celle du « premier jet » d'un traducteur junior ;
- Utilisation de méthodes d'IA (domaine tout nouveau en France)[(6)] ;
- Développement de LSPL permettant la programmation « heuristique » ;
- Transduction plutôt qu'analyse pour éviter le « tout ou rien » ;
- Structures linguistiques « multiniveau » (l'idée était de mettre les informations des différents « niveaux d'interprétation » linguistiques (morpho-syntaxiques, syntaxiques, logiques, sémantiques) sur un même graphe arborescent.
- Grâce aux systèmes-Q, on voulait aussi expérimenter un LSPL permettant d'écrire un système de TA complet. Cela commença par l'écriture par N.Nédobejkine, en systèmes-Q, d'un analyseur morphologique complet du russe.

1.3. Année à TAUM

J'arrivai à TAUM juste après le retour en France d'Alain Colmerauer, dans le cadre de la coopération franco-québecoise. Alain y avait passé 3 ans, comme Professeur Assistant d'informatique à l'Université de Montréal et directeur du projet TAUM, mais j'arrivai après son départ et ne le rencontrai pas.

À TAUM[(7)],

- il y avait en octobre 1971 Brian Harris, directeur (traductologue), succédant à Alain Colmerauer, Richard Kittredge (linguiste), Elliott Maclovitch (traducteur), Laurent Bourbeau (linguiste-informaticien), Jules Dansereau (linguiste), Gilles Stewart (informaticien-linguiste), et 2 ou 3 autres dont les noms m'échappent.

[(6)] Augustin Lux (même promo) fut sans doute le premier chercheur en IA à Grenoble !

[(7)] Traduction Automatique à l'Université de Montréal, groupe de recherche créé en 1965 par Guy Rondaud (?), puis dirigé par Alain Colmerauer de 1967 à 1970.

- Alain Colmerauer y était une vraie légende !
  - Il avait conçu le LSPL des « systèmes-Q » dès son arrivée en 1967 [1, 3],
  - il l'avait implémenté en 6 semaines en Algol-60 (sur CDC-6600),
  - et cela avec un exemple de système (maquette fr↔fr) réversible !

TAUM avait déjà fait :

- une 1re version de TA en→fr (AM[8] de Brian Harris, AS de Richard Kittredge, TLS de Gilles Stewart, GS de Jules Dansereau, GM de Michel Van Caneghem) ;
- une 2e version plus efficace des Systèmes-Q en Fortran (G.Stewart) ;
- une 3e version en assembleur CDC-6600 (toujours par G.Stewart), bien plus rapide, presque terminée.

Brian Harris me lança sur la « pidgin translation » ru → en+fr, avec comme modules :

- L'AM de N.Nédobejkine, qui produisait à partir de la n+1-ième occurrence russe un ou plusieurs arcs, de forme
  `-n- MOT(lemme russe, /, attributs russes) -n+1-`
- Puis un tranfert lexical vers l'anglais, produisant
  `-n- MOT(lemme anglais, /, attributs anglais) -n+1-`.
- Idem vers le français, avec en plus quelques réarrangements dans les groupes nominaux pour obtenir un ordre plus proche de celui du français,
- avec pour les deux une mise en page en SNOBOL.

L'idée était qu'on pourrait en tirer des présentations adaptées à la « lecture active ».

Exemple.

Это красная комната.

(C'est une chambre rouge.) donnait, en anglais et en français[9] :

This *PR+neu* red *AJ+fem+sin+nom* room *N+fem+sin+nom*.

Cela *PR+neu* chambre *N+fem+sin+nom* rouge *AJ+fem+sin+nom*.

1.4. Apports des systèmes-Q à la TA

Les systèmes-Q permirent d'écrire complètement TAUM-météo et de l'opérationnaliser, en 3 ans :

- 1975-77 : lancement du projet, avec Environnement Canada, l'objectif étant la TA en→fr des bulletins météo (20 000 mots/an).
- 1977 : John Chandioux quitta TAUM pour opérationnaliser TAUM-météo, et fut rejoint par Benoit Thouin (qui venait de faire son DEA à Grenoble en 1975-76, en travaillant sur la compilation du nouveau LSPL SYGMOR).

[8] AM = analyse morphologique, AS = analyse syntaxique, TLS = transfert lexical et syntaxique, GS = génération syntaxique, GM = génération morphologique.

[9] On ne produisait pas de couleurs à l'époque !

Ensuite, il y eut 2 évolutions.

- 1977-81 : TAUM se lança dans le projet TAUM-aviation. Les systèmes-Q furent un peu étendus (version optimisée pour les dictionnaires), et un nouvel outil, REZO, dérivé des ATN de Woods et travaillant sur des graphes-Q décorés par des attributs booléens, fut créé par G.Stewart.
- 1979-80 : on ajouta le sens fr→en à ce qui devint METEO. Et surtout, J.Chandioux fit évoluer les systèmes-Q vers un nouveau LSPL, GramR, sur les conseils d'Alain et de B. Vauquois, par ajout de typage (faible) et réduction/contrôle du non-déterminisme.
- 1985-2005 : METEO traduisait (sur PC) 30M mots/an (20 en en→fr, 10 en fr→en), avec une post-édition de 1mn/bulletin, épargnant ainsi 17 traducteurs seniors, et tous les traducteurs juniors qui auparavant faisaient un stage de plusieurs mois avec le pensum consistant à produire des traductions brutes d'une durée de vie de 4h. Non seulement c'était inintéressant et très répétitif, mais c'était difficile ! On peut rappeler que c'est à leur demande que le projet TAUM-météo fut lancé !

2. Après les débuts de Prolog

2.1. 1972-88 : Prolog vu de Grenoble

- L'IBM 360/67 du CICG (Grenoble) fut utilisée par l'équipe de Marseille, pour développer Prolog-I, qui sortit en 1972. Ensuite, elle passa à des mini- ou micro-ordinateurs (comme le Solar), puis à l'Apple II, sur lequel fut implémenté Prolog-II.
- Plusieurs groupes dont le GETA firent dès que possible des expériences avec Prolog pour de l'IA, en particulier pour du calcul formel, et on l'introduisit dans l'enseignement (BenZaken, Trilling, Boitet, Lévy).
- Au GETA, on avait abandonné l'idée d'un LSPL universel pour la TA, et Jacques Chauché créa 2 LSPL, immédiatement utilisés pour un nouveau système de TA ru→fr :
  - ATEF (transducteur à états finis non-déterministe) pour l'AM, muni de fonctions heuristiques,
  - CETA (systèmes transformationnels d'arbres, programmation heuristique).

    Puis l'équipe du GETA créa d'autres langages spécialisés (TRANSF/EXPANS pour les phases de transfert et d'expansion lexicales, SYGMOR pour la génération morphologique). ROBRA fut une extension de CETA (par moi-même) avec une nouvelle implémentation en PL360 (C n'était pas encore disponible).

  Cependant, nous fîmes avec B. Vauquois de nombreuses visites à Marseille (je rencontrai Alain pour la première fois fin 1972), et diverses utilisations de Prolog s'ensuivirent, dont :
  - le développement de GA (générateur d'analyseurs) et LT (langage de transcriptions), par Yves Lepage, vers 1986 ;

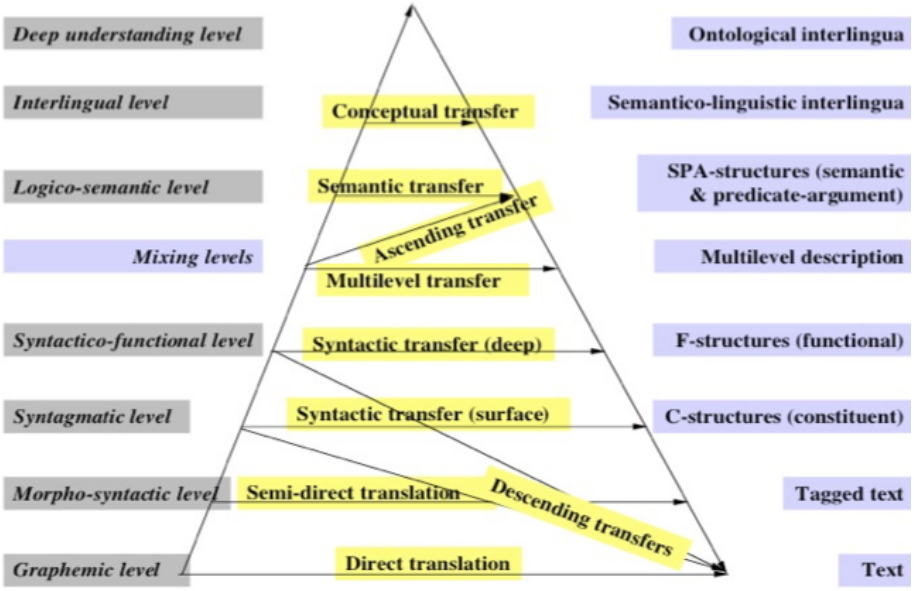
– un système de dictionnaires « furcoïdes » en Prolog-CRISS (augmenté de « grands caractères », préfigurant Unicode, pour traiter tous les systèmes d'écriture).

2.2. Et la TA ?

Alain se plaçait d'emblée dans une architecture linguistique à « transfert conceptuel », c'est-à-dire passant par un « pivot hybride » à la Shaumyan, proche du « Pivot-1 » du premier système du CETA.

Il y a 3 dimensions indépendantes dans les architectures des systèmes de TA complets : linguistiques, computationnelles, et opérationnelles

- Architectures linguistiques (voir le Triangle de Vauquois ci-dessous).



- Architectures computationnelles : méthodes pour passer d'une représentation à la suivante : empiriques, expertes, avec ou sans interaction ;
- Architectures opérationnelles, dépendant des buts (TA pour assimilation, pour dissémination avec haute qualité, pour dialogues...), des situations par rapport aux langues en jeu ($1 \rightarrow N, M \rightarrow 1, M \leftrightarrow N$), et des ressources disponibles (experts, données annotées...).

Que pensait Alain Colmerauer de la TA et qu'y a-t-il directement apporté ?

Lors d'un colloque à Moscou en 1977 (aussi mentionné ci-dessous), il m'asséna : « La TA est un gigantesque bricolage » !
Pourtant, il y a apporté directement des contributions théoriques réduisant ce « bricolage » :

- Représentations logico-sémantiques (cf. [3]) de forme

$$\text{Prédicat}(\text{Arg}_1, \text{Arg}_2, \text{Arg}_3, /, \text{Circ}_1 \ldots \text{Circ}_p)$$

- Quantificateurs à 3 branches [2]. Je reprends les 2 pages suivantes de [3].

Exemple. — Arthur possède une voiture

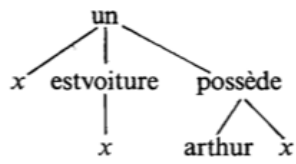
faire apparaître clairement. Nous proposons de remplacer l'énoncé par la paraphrase pseudo-française

pour un x tel que x est (une) voiture,
il est vrai qu'Arthur possède x

à laquelle nous associons la formule :

$$\text{un}(x,\text{estvoiture}(x),\text{possède}(\text{arthur},x))$$

qui se visualise mieux par l'arbre :



HYPOTHÈSE 2 : D'une façon générale à chaque article α correspondra un « quantificateur à trois branches » q qui, à partir d'une variable x et de deux formules $f1$ et $f2$, crée la nouvelle formule :

$$q(x,f1,f2)$$

correspondant à l'énoncé :

pour α x tel que $e1$, il est vrai que $e2$

où $e1$ et $e2$ sont les énoncés correspondant à $f1$ et $f2$.

Par article nous entendons des mots comme

"un", "chaque", "le", "la", "les",...

mais aussi des groupes de mots comme

"beaucoup de", "peu de", "aucun(ne)", "tous les",...

Nous avons introduit le terme de « quantificateur à trois branches » pour rappeler que ce type de quantificateur lie une variable à deux formules, contrairement aux quantificateurs classiques ∃ et ∀ qui ne lient une variable qu'à une seule formule. Nous indiquerons plus loin les cas où l'on peut se ramener directement à ces quantificateurs classiques.

3. Utilisation de Prolog en TA

Prolog fut d'abord utilisé en TA dans le système LMT (1984) d'IBM [7], « Logic-based Machine Translation », conçu par Mickael McCord, et basé sur ses Slot Grammars. Ses dictionnaires généraux et terminologiques comportaient environ 300 000 entrées en 2011. Dans les années 90, LMT fut distribué comme PT (Personal Translator) par Langenscheidt et amélioré par Linguatech (Heidelberg) and Lingenio (Munich), qui transformèrent son architecture linguistique, en passant d'un transfert descendant à un transfert horizontal.

C'est ce système qui fut utilisé pour produire « Talk & Translate », présenté à COLING-2000, le premier système grand public de TA de parole. L'analyseur de LMT fut utilisé dans Watson pour trouver et indexer les « factoïdes » dans 1M livres (50G mots) par le système qui gagna Jeopardy ! (le 14/2/2011).

Il faut aussi mentionner, vers 1985-90, la version Prolog de LINGOL (Hozumi Tanaka, Tokyo Institute of Technology), initialement développé en LISP. Cet environnement de développement de systèmes de TA fut utilisé par le groupe de Hiroshi Sakaki (KDD) pour développer le système KATE (utilisé à un moment par Nippon Steel).

Enfin, une approche intéressante, qui fut appliquée à la traduction de documents sur l'agriculture, fut le système « presque réversible » CRITTER [5], 1984-92.

4. Remarques diverses

4.1. Retour sur l'apport des systèmes-Q à la TA

En 1977, je rencontrai Alain pour un colloque sur la TA à Moscou. Partageant la même chambre, nous avons beaucoup discuté. Il était en train de travailler sur Prolog-II, et continuait à monter le nouveau département d'informatique à Aix-Marseille.

Il me dit une phrase souvent citée : « j'ai raté Prolog de peu quand j'ai créé les systèmes-Q ». Peut-être... mais ce détour fut un bien ! Certes, ils ne font que de l'instanciation, pas de l'unification comme Prolog-I-II-III, et encore moins de la résolution de contraintes comme Prolog-IV.

Cependant, ils ont permis d'écrire très vite des systèmes « pour sous-langages » (TAUM-météo, METEO) opérationnels, bien plus rapides que n'aurait été une implémentation en Prolog, et quasi-parfaits[(10)].

[(10)] Eh oui ! Plus précisément, le système METEO a tourné pendant près de 20 ans en produisant des TA de qualité exceptionnelle, avec typiquement 1 minute de post-édition par bulletin, soit moins de 5 minutes par page standard de 250 mots (1 400 caractères), faite par des traducteurs seniors du Bureau des Traductions d'Ottawa.

On a pu lire ces dernières années, même dans des articles et revues scientifiques, que les récentes méthodes empiriques de TA (statistiques, par l'exemple, neuronales) donnaient des résultats aussi bons sinon meilleurs que ceux des traducteurs professionnels. Mais ... c'est tout à fait faux.

Ce qui est vrai, c'est qu'un traducteur professionnel, s'il utilise un système de TA (adapté à son couple de langues, au type de document, et au domaine), pourra post-éditer le résultat en 15 à 20 minutes par page.

4.2. METEO : un système « expert » meilleur que de la TA statistique ou neuronale

Quant à la prétendue perfection des systèmes comme DeepL, Bing, GoogleTranslate, etc., voici la TA brute produite par DeepL sur le résumé de cet article.

"I'm gathering here a few dazzling memories of Alain Colmerauer. Without yet knowing him, I studied and used his 'Q-systems' at TAUM (UdM), where he invented them, during the first year of my thesis. I retrace his contributions to MT (by hybrid conceptual pivot à la Shaumyan). hybrid conceptual pivot), to logic programming (with BenZaken, we taught 'how and why Prolog works'), and to computational linguistics (including his linguistics (including its original formalization with 'three-pronged quantifiers')."

Les deux premières phrases sont parfaites (20/20), mais la troisième, un peu plus longue, ne va pas du tout (5/20). À cause de répétitions erratiques et d'occurrences aberrantes (comme « including his linguistics »), on est loin de ce que produirait un traducteur professionnel (que je fus occasionnellement depuis 1970 entre français, allemand et anglais).

4.3. Un chercheur éblouissant et une personnalité chaleureuse

Plusieurs fragments de ce qui suit sont tirés de [4], car comment les formuler mieux que lui ? Sur le fond, Alain était vraiment génial.

- Capacité à trouver des moyens simples (*a posteriori* !) pour mettre en œuvre des résultats théoriques de façon opérationnelle.
- Incroyable efficacité (« pondre » les systèmes-Q en 6 semaines...).
- Prolog-I (1972) : méthode de Robinson (preuve par résolution) avec unification et skolémisation et suppression du occurCheck pour gagner du temps (beaucoup).
- Prolog-II (1978) : remplacement de l'unification par la résolution d'équations dans un domaine donné, ce qui permit d'introduire les arbres rationnels, la relation $\neq$ et le « freeze » ;
- Prolog-III (1987) : intégration au niveau de l'algorithme d'unification :
  (1) d'une manipulation plus fine des arbres, qui peuvent être infinis, avec un traitement spécifique pour les listes ;
  (2) d'un traitement complet de l'algèbre de Boole ;
  (3) d'un traitement numérique en précision infinie comprenant l'addition, la multiplication par une constante et les relations $<, \leqslant, \geqslant, >$;
  (4) d'un traitement général des relations $=, \neq$.

Vers 2005, j'ai introduit la *mesure de qualité d'usage*, une note sur 20 donnée par la formule :

$$\text{Qualité d'usage} = (20 - 2/5 \times T_{\text{PostEdit}})/20$$

Pour 15 minutes, cela donne 14/20 (bien). Pour 20 minutes, cela donne 12/20 (assez bien). Pour METEO, cela donnait donc 18/20 (excellent).

- Prolog-IV (1996) : unification de la programmation logique et de la programmation mathématique via l'introduction de la *programmation par contraintes* et l'intégration de nombreux algorithmes de résolution de contraintes.

Il y aurait encore tant à dire...

Relire sa thèse d'État (9/67), ce que j'ai fait depuis cette journée d'hommage, permet de voir comment il cherchait – et trouvait.

Quant au plan personnel, Alain était « une crème » – mais c'est plutôt à d'autres d'en parler !

Remerciements

Merci d'abord à Marc Bergman pour avoir organisé cette journée d'hommage et m'y avoir invité, à Vincent Risch qui a mis en place la journée et le format LaTeX de ce volume, et à Odile Papini, pour sa patience et son aide.

Bibliographie

[1] A. Colmerauer, « Les systèmes-q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur », Publication interne n° 43, Département d'Informatique de l'Université de Montréal, 1970.

[2] ———, « Un sous-ensemble intéressant du français », *RAIRO. Informatique théorique* **13** (1979), n° 4, p. 309-336.

[3] ———, « Les systèmes-Q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur », *Traitement Automatique des Langues, Revue semestrielle de l'Association pour le Traitement Automatique des Langues* **33** (1992), n° 1-2, p. 105-148, Il s'agit du rapport technique n°43 (1970) de l'Université de Montréal, qui avait été publié seulement comme rapport technique auparavant.

[4] ———, « CV d'Alain Colmerauer avec présentation détaillée de toute sa recherche », novembre 2007, `http://alain.colmerauer.free.fr/alcol/ArchivesPublications/CurriculumVitae/Cvf06a4.pdf`.

[5] M. Dymetman & P. Isabelle, « Grammaires logiques réversibles pour la traduction automatique », Monographie Co28-1/28-1988E-PDF, Laval: Canadian Workplace Automation Research Centre, Department of Communications Canada, November, 1988, Existe aussi en version anglaise.

[6] B. Henisz-Dostert, R. R. Macdonald & M. Zarechnak, *Machine Translation*, De Gruyter., 2011 (Accessed: 14 October 2022), 278 pages.

[7] M. McCord, « A New Version of the Machine Translation System LMT », *Literary and Linguistic Computing* **4** (1989), n° 3, p. 218-229.

Annexe : petite introduction aux systèmes-Q par l'exemple

Tirée de mes cours/TD à Polytech (Grenoble) sur la « communication multilingue ».

I. Outils informatiques : systèmes-Q (extrait d'un examen de RICM en 2009)

On s'intéresse à la représentation de phrases comportant un sujet coordonné et un objet coordonné, formés de noms propres, avec un verbe du 1er groupe et un point final. Si le sujet et l'objet ont le même nombre d'éléments, et la même conjonction de coordination (et, ou), on dira que la phrase est de type PHC2PAR (coordination parallèle), et sinon qu'elle est de type PHC2GEN (coordination générale). Voici un exemple de PHC2GEN et un exemple de PHC2PAR sous forme de graphes-q.

```
Ex. 1: -01- *ANNE+ET+*CLIO+HABITENT+*ARLES+,+*PARIS+OU+*CHAMBERY+.-02-
Ex. 2: -01- *ANNE+,+*JEAN+ET+*CLIO+AIMENT+*ARLES+,+*PARIS+ET+*CHAMBERY+.-02-
```

On convient que "" en début d'étiquette indique une majuscule initiale d'un nom propre.*

1) *Soit le traitement-q T1 :*

```
-REQ- ** T1.
,  == :VIRG (,).
.  == :POINT(.).
A* == :COORD(A*)    / A* -DANS- ET, OU.    ** Produire :CCORD(ET) ou :COORD(OU).
A* == :NPRO (A*)    / *  -DANS- $$A*.      ** Nom propre si "*" dans l'étiquette.
A* == :VRB1 ($$A*) / A* -HORS- ET, OU -ET- *, ., , -HORS- $$A*. ** (5ème règle).
:VRB1 ($$A*, E, N, T) == :VRB2 ($$A*, E, R).
:VRB2 ($$A*)          == :VERBE(A*). ** AIMENT et AIMER donneront :VERBE(AIMER).
```

a) Interpréter la condition de la 5ème règle. Détailler l'exécution sur `-01-*AL+,+*JO+JOUENT+.-02-` et donner le résultat de T1 sur le premier exemple.

►Cette condition signifie que l'étiquette instanciant `A*` ne peut pas commencer par `'*'`, `'.'` ou `','`. En effet, les deux listes doivent avoir une intersection non vide, et ces caractères non alphnumériques ne peuvent apparaître qu'en tête d'une étiquette. Sur le premier exemple, T1 donne :

```
-01- :NPRO(*ANNE) + :COORD(ET) + :NPRO(*CLIO) + :VERBE(HABITER) + :NPRO(*ARLES)
+ :VIRG(,) + :NPRO(*PARIS) + :COORD(OU) + :NPRO(*CHAMBERY) + :POINT(.) -02-◄
```

b) Que se passe-t-il si on a un mot de type non prévu comme `AIMONS` ? Donner la forme générale du résultat de T1 sur une phrase (correcte) de type PHC2GEN.

►Sur `AIMONS`, on obtient `:VRB1(A,I,M,O,N,S)`. La forme générale est `-01-:NPRO(Nom1_1)+ :VIRG(,)+… +:COORD(Coord1)+:NPRO(Nom1_n)+:VERBE(Infinitif)+:NPRO(Nom2_1)+ :VIRG(,)+… +:COORD(Coord2)+:NPRO(Nom2_p)+:POINT(.)-02-`.◄

2) a) Écrire un traitement-q T2 pour transformer le résultat de T1 sur une phrase de type PHC2GEN en un graphe-q à un seul arc, de forme :

```
-01- :PHC2GEN(:GNC(:COORD(Coord1), :NPRO(Nom1_1) …, :NPRO(Nom1_n)),
:VERBE(Infinitif), :GNC(:COORD(Coord2), :NPRO(Nom2_1) …, :NPRO(Nom2_p))) -02-
```

```
►-REQ- ** T2.
      ** Construction des deux groupes nominaux par la droite.
:NPRO(A*) + :COORD(C*) + :NPRO(B*)  == :GNC(:COORD(C*), :NPRO(A*), :NPRO(B*).
:NPRO(A*) + :VIRG(,) + :GNC(I*, U*) == :GNC(I*, :NPRO(A*), U*).
      ** Construction finale de la phrase.
:GNC(U*) + :VERBE(A*) + :GNC(V*) + :POINT(.)
  == :PHC2GEN(:GNC(U*), :VERBE(A*), :GNC(V*)).  ** Construction finale.◄
```

b) *On veut produire un graphe vide si la phrase n'est pas du type prévu.* Ajouter 3 règles (de type "cul de sac" de Colmerauer) pour cela et dire pourquoi il vaut mieux les faire précéder d'un `-REQ-`.

```
►-REQ- ** Si l'arbre d'un arc n'a pas :PHC2GEN pour racine, vider le graphe.
A*(U*) == &CUL + &DE + &SAC  / A* -NE- :PHC2GEN.  ** Ces trois règles.
&CUL + &DE == &CULDE.      &DE + &SAC == &DESAC.  ** créent un "trou".
```

Si on ne sépare pas par un `-REQ-`, ces 3 règles s'appliqueront inutilement à tous les arbres du graphe, initiaux et construits, sauf au tout dernier si la phrase est correcte. Mais le résultat final sera le même.◄

3) *On veut maintenant produire une structure abstraite (sans fonctions syntaxiques et sans symboles de syntagmes) pour les phrases PHC2GEN et on propose de continuer avec T3 :*

```
-REQ- ** T3.
** Production d'une structure comme: habiter(et(Al, Jo), ou(Ay, Is, Eu)).
:PHC2GEN(I*, J*, K*, L*) == :PHC2GEN + I* + J* + K* + L* + :POINT(.).
:GNC(U*) == U*.
:COORD(A*, U*) + :NPRO(B*) == :COORD(A*, U*, B*).
:COORD(A*, U*) + :VERBE(B*) + :COORD(C*, V*) + :POINT(.) == B*(A*(U*), C*(V*)).
```

a) Une règle de T3 est syntaxiquement incorrecte. Pourquoi ? Écrire T3 après correction.

►":GNC(U*) == U*" est une règle incorrecte car on doit avoir un chemin de longueur connue en partie droite comme en partie gauche. Il faut donc "sortir" progressivement les éléments de la liste. On peut par exemple remplacer cette règle par les deux règles :

```
:GNC(U*,I*) == :GNC(U*)+I*. **Sortir U* par la droite.  :GNC+I* == I*. **Fin.◄
```

b) Donner le résultat de T3 corrigé sur l'exemple 1, et la forme générale du graphe résultat.

```
►-01- :PHC2GEN + HABITER(ET(*ANNE, *CLIO), OU(*ARLES, *PARIS, *CHAMBERY))-02-
-01- :PHC2GEN + Verbe(Coord1(Nom1_1,… Nom1_n), Coord2(Nom2_1,… Nom2_p) -02-◄
```

4) *On veut produire pour une phrase PHC2PAR* (Coord1=Coord2=Coord & n=p) *un graphe de forme :*

```
-01- :PHC2PAR + Verbe(,(Nom1_1, Nom2_1, ,(Nom1_2, Nom2_2, ,(… Coord(Nom1_n,
Nom2_n)…)))-02-.
```

On écrit pour cela deux traitements-q T4 et T5 successifs.

```
-REQ- ** T4.
** Production d'une structure comme: habiter(,(Al, Paris), et(Jo, Istres)).
:PHC2GEN + I* == :PHC2GEN(I*).   ** Mise en réserve du résultat précédent.
B*(C*(U*,D*), C*(V*,E*)) == B*(C*(U*), C*(V*), C*(D*,E*)). ** 1er couple.
      ** --> C'est ici que les 2 règles manquent: les spécifier et les écrire.
-REQ- ** T5: produire :PHC2PAR+cette structure ou garder l'entrée de T4.
:PHC2GEN(I*) + B*(J*)    == :PHC2PAR + B*(J*).
:PHC2GEN(I*) + B*(J*,U*) == :PHC2GEN + I*.
```

Compléter T4 par 2 règles adéquates, et corriger la seconde règle de T5.

►** Pas tout-à-fait ce qu'on veut! La 2ème règle de T4 vérifie qu'on a la même coordination et commence la construction par la droite. Il faut continuer jusqu'à arriver à Verbe(Coord, Coord, Coord(Nom1_1, Nom2_1),… Coord(Nom1_n, Nom2_n)). Si on n'y arrive pas, c'est que n≠p et il restera 3 arbres sous le verbe.

La seconde règle de T5 ne va pas car U* peut s'instancier à la liste vide.

```
-REQ- ** T4.
** Production d'une structure comme: habiter(,(Al, Paris), et(Jo, Istres)).
:PHC2GEN + I* == :PHC2GEN(I*).   ** Mise en réserve du résultat précédent.
B*(C*(U*,D*), C*(V*,E*)) == B*(C*(U*), C*(V*), C*(D*,E*)). ** 1er couple.
B*(C*(U*,D*), C*(V*,E*), I*) == B*(C*(U*), C*(V*), ,(C*(D*, E*), I*))
                              / -NON-(.U*, V* -EQ- -NUL-.).
B*(C*, C*, I*) == B*(I*)).
-REQ- ** T5: produire :PHC2PAR+cette structure ou garder l'entrée de T4.
:PHC2GEN(I*) + B*(J*)          == :PHC2PAR + B*(J*). ** Il reste un seul arbre.
:PHC2GEN(I*) + B*(J*, K*, L*) == :PHC2GEN + I*.     ** B*(J*, K*, U*) va aussi.
◄
```

ABSTRACT. — I gather here a few dazzling memories of Alain Colmerauer. Without yet knowing him, I studied and used his "Q-systems" at TAUM (UdM), where he invented them, during the first year of my PhD thesis. I retrace his contributions to MT (by hybrid conceptual pivot à la Shaumyan), to logic programming (with BenZaken, we taught "how and why Prolog works"), and to computational linguistics (including his original formalization with "three-pronged quantifiers").

KEYWORDS. — Alain Colmerauer, Q-systems, logic programming, Prolog, machine translation.

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