Université de Nantes
Institut de Recherche en Informatique de Nantes

Habilitation à Diriger des Recherches

Logique linéaire et syntaxe des langues

Christian RETORÉ
(Université de Nantes & INRIA)

soutenue le 4 janvier 2002 devant le jury composé de

M.	Alexandre	DIKOVSKY	*président*
MM.	Gérard	HUET	*rapporteurs*
	Joachim	LAMBEK	
	Michael	MOORTGAT	
Mme	Claire	GARDENT	*examinateurs*
MM	Gilles	KAHN	
	Jean-Xavier	RAMPON	
	Olivier	RIDOUX	

Organisation du manuscrit

Ce manuscrit comprend une partie **Présentation** composée de deux chapitres :

- *La thématique: logique linéaire, syntaxe des langues* qui présente les questions auxquelles nous nous sommes intéressés ainsi que nos motivations
- *État de l'art et contributions* qui résume très succintement nos contributions en les situant dans un panorama du domaine.

Les deuxième et troisième parties présentent des résumés de nos travaux.

La deuxième, **Graphes et logique linéaire** se compose de quatre chapitres :

- *Graphes*
- *Réseaux de démonstration*
- *Sémantique dénotationnelle*
- *Calcul mixte et réseaux de Petri*

et la troisième, **Modèles grammaticaux**, comporte trois chapitres dont les deux derniers sont issus de ma collaboration avec Alain Lecomte:

- *L'interface entre syntaxe et sémantique*
- *Modules ordonnés et grammaires*
- *Grammaires minimalistes catégorielles*

La **Conclusion** présente un bilan et ouvre quelques perspectives.

Remerciements

Ce travail a en grande partie été réalisé à l'INRIA (Sophia-Antipolis, LORIA et IRISA) dans les projets Meije (Sophia), Calligramme (Nancy), Aïda et Paragraphe (Rennes). Il a été achevé au sein de l'équipe TALN de l'IRIN à l'Université de Nantes. J'ai, dans ces divers endroits, bénéficié de conditions de travail idéales,tant matérielles qu'intellectuelles, et j'en remercie les responsables et les collègues. Mentionnons plus particulièrement Gérard Berry et Gérard Boudol (Sophia), Denis Bechet, Philippe de Groote, François Lamarche, Jean-Yves Marion et Guy Perrier (Nancy), Didier Caucal, Philippe Darondeau, Annie Foret, Daniel Herman, Jacques Nicolas, Sophie Pinchinat et Olivier Ridoux (Rennes), Béatrice Daille, Alexandre Dikovsky, Chantal Enguehard, Emmanuel Morin, Jean-Xavier Rampon (Nantes).

Outre les collègues, je remercie les jeunes chercheurs que j'ai eu la chance d'encadrer. Leur communiquer de l'enthousiasme pour un domaine que l'on affectionne est toujours une excellente stimulation et bien souvent une occasion de se remettre en question : Sylvain Pogodalla (INPL & Xerox), Elena Maringelli (U. di Roma tre), Yannick Le Nir et Roberto Bonato (U. Rennes 1), Erwan Moreau, Véronique Moriceau et Guillaume Pasquier (U. de Nantes). L'encadrement de leurs travaux a été l'occasion de collaborations avec Marc Dymetman (Xerox, Grenoble), Alain Lecomte (U. Grenoble II), Gianluigi Bellin (U. Verona), Jacques Nicolas (INRIA, Rennes), Annie Foret (U. Rennes 1), Alexandre Dikovsky (U. Nantes).

Mes recherches ont été fortement marquées par les travaux et le point de vue de Jean-Yves Girard (C.N.R.S., Paris, Marseille) qui a encadré ma thèse et m'a fait découvrir la logique linéaire, et, des années après, je lui en suis encore gré. Concernant cette même période, je remercie vivement Michel Parigot (C.N.R.S., Paris) qui en quelques jours m'a sorti de l'embarras en me trouvant un post-doc à l'INRIA, où je suis ensuite devenu chercheur.

Parmi les personnes mentionnées certaines ont directement contribué aux travaux que nous décrivons ici et je les en remercie plus particulièrement: Denis Bechet, Philippe de Groote, François Lamarche (Nancy), Edward Stabler (Los Angeles),et Nissim Francez (Tel-Aviv) avec lequel un livre de cours est en préparation. Bien que Philippe Darondeau (INRIA Rennes) ne soit pas à

proprement parler un co-auteur, ce travail et en particulier la partie sur les réseaux de Petri doit beaucoup à nos conversations ; j'ai beaucoup apprécié l'étendue de son savoir et la profondeur de sa réflexion.

Alain Lecomte (U. Grenoble II) est plus qu'un co-auteur. C'est lui qui m'a fait découvrir les applications de la logique à la linguistique, et toute la partie sur les modèles grammaticaux est le fruit de notre collaboration. Je le remercie bien sûr de m'avoir ouvert les yeux sur ce domaine passionnant, par son enthousiasme, son inventivité et sa culture ; je souhaite vivement que cette collaboration perdure.

N'étant pas linguiste de formation, je tiens à remercier les collègues qui ont utilement et agréablement complété ma formation mathématique, en particulier à l'occasion des cours proposés par les *European Summer School in Logic, Language and Information* tels ceux de Daniel Büring, Matthew Crocker, Edward Keenan, Edward Stabler, Tanya Reinhardt, ou de discussions à bâtons rompus avec Claire Beyssade (U.Paris III), Emmanuelle Chardon-Lechêne (U. Paris III), Claire Gardent (CNRS, Nancy), Evelyne Jacquey (U. Nancy II), Mark Johnson (Brown U.) et Aravind Joshi (Philadelphia).

Parmi les équipes étrangères qui ont influencé et permis ce travail, je dois mentionner en premier lieu celle de Michael Moorgat (U. Utrecht) où lui et ses étudiants,Raffaella Bernardi, Christophe Costa-Florencio, Esther Kraak, Richard Moot, Quintijn Puite, Willemijn Vermaat mènent des recherches de premier plan sur ce sujet, dans une ambiance particulièrement conviviale. Les *Roma Workshops* organisés par Claudia Casadio (U. Chieti) sur les applications de la logique linéaire à la linguistique ont aussi été des rencontres très stimulantes, où j'ai en particulier profité de discussions avec Glyn Morrill (UPC, Barcelona), Dick Oehrle (Tucson), et tout particulièrement avec Michele Abrusci (U. Roma tre).

Le moment venu, la première étape fut d'écrire un manuscrit.

Celui-ci a été relu, dans un délai très bref par Jean-Yves Marion (INPL, Nancy), Yannick Le Nir (U. Rennes 1) et Sylvain Pogodalla (Xerox, Grenoble), et a bénéficié de corrections de dernière minute de Sophie Pinchinat (U. Rennes 1). L'aide la plus substantielle dont j'ai bénéficié dans la rédaction du manuscrit est assurément celle d'Evelyne Jacquey (U. Nancy II), qui a considérablement amélioré certaines argumentations. Je les remercie tous vivement de leur aide.

Ce manuscrit ne serait pas devenu une habilitation sans l'intervention de rapporteurs et d'un jury, et je me réjouis d'avoir pu réunir dans cette dernière épreuve des experts pour lesquels j'ai la plus grande estime.

Je suis content que l'une des premières personnes que j'ai connues dans la communauté de la linguistique informatique, et assurément l'une des plus compétentes, Claire Gardent (C.N.R.S. LORIA, Nancy), m'ait fait l'honneur de ses commentaires et de sa présence le jour fatidique.

Je dois beaucoup à Gilles Kahn (INRIA-Rocquencourt) qui a favorisé ce travail en soutenant l'approche logique du traitement des languesdéveloppée par le projet Calligramme, puis l'action de recherche GRACQ. La lecture du manuscrit et l'appréciation de mon travail par un informaticien aussi complet m'ont fait grand plaisir.

Ce travail comporte une partie sur les graphes, et j'ai la chance davoir connu à Nantes un expert, Jean-Xavier Rampon (U. de Nantes), qui a bien voulu se pencher sur ce travail, et a su proposer des perspectives prometteuses.

Converser avec Olivier Ridoux (U. Rennes 1) est toujours enrichissant ; j'ai beaucoup profité de l'étendue de sa culture en maints domaines différents de l'informatique, domaines entre lesquels qu'il suggère souvent des connexions, parfois inattendues et toujours pertinentes.

Alexandre Dikovsky (U. de Nantes) a bien voulu présider le jury, et je l'en remercie: sa vaste culture en logique et linguistique en font un collègue dont la fréquentation est toujours enrichissante, d'autant que son point de vue, celui des grammaires de dépendance, diffère de celui que nous avons développé.

Michael Moortgat (U. Utrecht), expert des grammaires catégorielles dont j'ai beaucoup appris, m'a fait le plaisir d'évaluer ce travail ; l'approche multimodale qu'il développe est certes voisine mais aussi concurrente, ce qui fait de lui un relecteur parfait.

Joachim Lambek (U. McGill, Montreal) a aussi accepté d'être rapporteur de ce travail, ce qui m'honore puisqu'il est le père fondateur de cette approche logique (il préfèrerait sans doute algébrique) de la syntaxe des langues, et qui a aussi contribué à bien d'autres domaines — par exemple son livre de théorie des modules m'avait beaucoup plu lorsque j'étais étudiant. L'élégance de ses articles est pour moi un exemple dont je suis encore loin.

Cette habilitation n'eut certainement pas vu le jour de si tôt sans les encouragements et les conseils de Gérard Huet (INRIA-Rocquencourt), qui a bien voulu mettre sa grande compétence de logicien, et celle plus nouvelle mais tout aussi dynamique de linguiste computationnel, au service d'une relecture avisée, source de discussions fructueuses et amicales. Je l'en remercie tout particulièrement.

Première partie

Présentation

Chapitre 1

La thématique : logique linéaire, syntaxe des langues

Le point de départ de nos recherches est la logique linéaire et en particulier ses variantes non commutatives. Depuis quelques années, nous essayons de proposer des modèles de la syntaxe des langues fondés sur ces systèmes déductifs. L'objectif, peut-être trop ambitieux, de ce mémoire est de convaincre informaticiens, linguistes et logiciens de l'intérêt de cette nouvelle convergence entre logique et grammaire à travers la présentation de nos travaux dans ce domaine. L'introduction que voici et le document qui suit possèdent donc deux parties correspondant aux deux communautés concernées.

La logique linéaire, introduite par Jean-Yves Girard voici une quinzaine d'années [1], est tout à la fois une branche de la théorie de la démonstration classique et une théorie mathématique de certains aspects de l'informatique [2,3,4]. Le plus souvent, les problèmes informatiques abordés par la logique linéaire sont l'évaluation des langages fonctionnels avec partage de calculs, suivant le paradigme *proofs-as-programs*, ou la programmation logique et la modélisation de la concurrence, suivant le modèle *proof-search-as-computation*.

La principale application considérée ici relève de l'informatique linguistique : le calcul logique y est utilisé pour représenter les mécanismes calculatoires de

---

[1] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[2] Jean-Yves Girard. Linear logic: its syntax and semantics. In Girard et al. [GLR95], pages 1–42.

[3] Max Kanovitch. Linear logic as a logic of computation. *Annals of pure and applied logic*, 67:183–212, 1994.

[4] Anne Sjerp Troelstra. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. CSLI, 1992. (distributed by Cambridge University Press).

la syntaxe, et, dans une moindre mesure, de la sémantique des langues. Cette approche calculatoire de la syntaxe des langues a été initiée par Noam Chomsky dans les années cinquante [1] et continûment développée depuis en une théorie appelée « grammaire générative » [2,3]. On notera que dès cette époque Joachim Lambek a proposé un modèle logique des calculs impliqués dans la syntaxe des langues [4], dont la logique n'est autre qu'un fragment de la logique linéaire non commutative. Malheureusement, ces grammaires n'ont pas connu le développement qu'elles méritaient ; d'une part ce calcul logique était sans doute isolé dans la théorie de la démonstration de l'époque, et, d'autre part, les théories linguistiques du moment ne semblaient guère priser les attraits de cette approche : lexicalisme, lien avec la sémantique prédicative, apprentissage,. . . .

Les nouveaux modèles grammaticaux que nous avons proposés avec Alain Lecomte sont issus du calcul de Lambek. Ils tirent notamment profit de la structure combinatoire qu'ont les démonstrations de la logique linéaire : celles-ci sont des graphes appelés réseaux de démonstration. De ce fait, une partie du mémoire relève de la théorie des graphes, que nous avons rendue aussi standard que possible : couplages parfaits, cographes, etc. La mise au point des calculs logiques nous a aussi conduit à quelques détours par la sémantique dénotationnelle des espaces cohérents : il s'agit là encore de graphes, mais infinis cette fois.

Ces nouvelles grammaires catégorielles visent à étendre les grammaires de Lambek pour en augmenter la capacité générative et pour les rendre plus réalistes linguistiquement. Pour cela on s'inspire des grammaires d'arbres adjoints d'Aravind Joshi [5] et surtout des grammaires minimalistes d'Edward Stabler [6] qui formalisent les récents travaux de Noam Chomsky [7], en exploitant la richesse des structures offertes par la logique linéaire. On étudie également ce que deviennent les deux atouts principaux des grammaires catégorielles que sont leur interface aisée avec la sémantique, et l'existence d'algorithmes d'inférence grammaticale.

Le point de convergence entre les aspects mathématiques et linguistiques de ce travail est donc le calcul de Lambek ou, plus généralement, la logique linéaire.

---

[1] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, IT2:113–124, 1956.

[2] Jean-Yves Pollock. *Langage et cognition: le programme minimaliste de la grammaire générative*. Presses Universitaires de France, Paris, 1997.

[3] Ray Jackendoff. *The Architecture of the Language Faculty*. Number 28 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1995.

[4] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[5] Aravind Joshi, Leon Levy, and Masako Takahashi. Tree adjunct grammar. *Journal of Computer and System Sciences*, 10:136–163, 1975.

[6] Edward Stabler. Derivational minimalism. In Retoré [Ret97a], pages 68–95.

[7] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

En amont, sont établis des résultats combinatoires sur les graphes qui sont les lemmes nécessaires à l'établissement de propriétés logiques. En aval, des modèles grammaticaux issus des calculs logiques sont proposés. Comme le suggèrent les définitions rappelées ci-dessous, les différents aspects de ce travail peuvent être regroupés comme un travail sur la syntaxe : syntaxe graphique pour la logique, syntaxe des langues. L'objectif est un traitement algorithmique d'informations structurées, qu'elles soient de nature grammaticale ou logico-informatique.

Informatique : *n. f. et adj. (1962), mot créé par Ph. Dreyfus, sur le modèle de* mathématique, électronique, *et qui a lui même servi de modèle pour de nombreux dérivés analogiques (*bureautique, robotique, *etc.). Le mot désigne la science et l'ensemble des techniques automatisées relatives aux informations (collecte, mise en mémoire, utilisation etc.) et l'activité économique mettant en jeu cette science et ces techniques. L'informatique est en rapport avec les notions de calcul, de classement et d'ordre (*cf. ordinateur*) et, avec la gestion des données des informations au moyen de logiciels et du matériel approprié. [. . .]*

Logique : *n. f. a été emprunté (v. 1245) au latin* logica *« science des lois du raisonnement », lui-même emprunté au grec* logikê *(sous-entendu* tekhnê*) substantivation de l'adjectif* logikos *« qui concerne la raison » également « qui concerne la parole ». [. . .]* Logos, *forme de type ancien et de grande importance, signifie « propos, parole » et, en ionien-attique « récit, compte, explication ,considération, raisonnement, raison » ainsi que « parole ». Il a fini par désigner la raison immanente, et, dans la théologie catholique, la seconde personne de la Trinité, ou Dieu. Il est dérivé de* legein *« rassembler, cueillir,choisir » d'où « dire » et « compter, dénombrer » qui correspond au latin* legere, *de la même racine indoeuropéenne, et qui a donné le verbe lire.*

Grammaire : *n. f. [. . .] Mais le mot se spécialise aussi très tôt (v. 1200) au sens d'« étude systématique des éléments constitutifs d'une langue » [. . .] Aujourd'hui en linguistique, grammaire désigne l'ensemble des structures et des règles qui permettent de produire tous les énoncés corrects d'une langue : il est alors voisin de* syntaxe.

Graphe : *n. m. attesté en 1926 (certainement antérieur) est un emprunt à l'anglais* graph, *abréviation de* graphic formula *« formule graphique »,* graphic, *comme le français graphique, provenant du grec* graphein, *« écrire ». [. . .]*

Syntaxe : *n. f. réflection graphique d'après le latin (1640) de* sintaxe *(1572, Ramus) est emprunté au bas latin grammatical* syntaxis *« ordre, arrangement des mots », qui reprend le grec* suntaxis *« ordre, arrangement, disposition » d'où plusieurs sens qui conservent l'idée de mise en ordre, par exemple « ordre de bataille », « organisation d'un État », « composition d'un ouvrage », et, tardivement, « construction grammaticale ». Ce nom dérive du verbe* suntassein *(ou* suntattein*) « ranger ensemble », « arranger » (→ syntagme). [...] Les grammairiens latins utilisaient l'hellénisme* syntaxis *ou* constructio *(→ construction) pour la partie de la grammaire qui traitait de l'arrangement des mots mais aussi de l'emploi des mots, des cas et des modes, cet arrangement étant « réalisé par la construction d'une oraison parfaite » (Priscien, vi*^{e} *siècle). [...] Meigret (1550,* Tretté de la grammere françoise*) n'emploie pas* syntaxis *mais « bâtiment ou construccion ou ordonance bone de parolles » ; c'est Ramus qui introduit* syntaxe *pour désigner l'arrangement des mots, la construction des propositions ainsi que l'étude des règles qui les régissent. [...] Au xx*^{e} *siècle, le mot s'emploie surtout en linguistique descriptive, la pédagogie des langues préférant nommer* grammaire *la syntaxe normative.*

(Dictionnaire Historique de la Langue Française, sous la direction d'Alain Rey, Éditions Le Robert, 1994)

1.1 La logique linéaire

Ce genre de document m'impose de donner une présentation informelle de la logique linéaire, et de situer son rôle dans l'informatique ainsi que sa place dans la logique. N'étant pas philosophe de formation, ces lignes risquent d'être prétentieuses, naïves, et de contenir des contresens épistémologiques[a].

La logique est une discipline âgée d'au moins deux mille ans[b], associée dès ses débuts à la grammaire[c], et qui a beaucoup évolué durant le siècle dernier. Dès la fin du 19^{e}siècle elle s'est mathématisée, lors de la crise des fondements des mathématiques, et depuis le milieu de ce siècle l'éclosion de l'informatique lui a fournit un domaine privilégié d'application, assez distinct des mathématiques. L'informatique est pour la logique une source de renouveau, y compris dans ses thèmes : la notion de calcul est maintenant centrale en logique. La logique est traditionnellement définie comme l'étude des principes valides de « raisonnement », et de la « vérité ». Par conséquent elle traite aussi du langage qui exprime les énoncés qu'elle manipule et met en rapport. Pour reprendre les termes d'un ancien débat [4,5,6], elle peut être vue comme universelle (*lingua characterica*) ou comme un calcul (*calculus ratiocinator*). Selon la première vision, rien ne saurait exister ou s'exprimer en dehors de la logique, tandis que suivant la seconde, on peut envisager divers « mondes » où interpréter les énoncés logiques et où calculer leur « vérité ». On retrouve à mon avis cette opposition dans l'utilisation informatique qui est faite de la logique linéaire. Néanmoins, plus encore que dans le cas de la logique intuitionniste, les démonstrations jouent un rôle central, et c'est plutôt à

a. Merci à Michele Abrusci : les discussions que j'ai eues avec lui ont sans doute amélioré cette présentation.

b. Une référence standard pour l'histoire de la logique est [1]

c. La grammaire de Denys le Thrace est certainement l'un des ouvrages où c'est le plus patent, comme expliqué dans, par exemple [1,2,3]

[1] William Kneale and Martha Kneale. *The development of logic*. Oxford University Press, 3rd edition, 1986.

[2] Marc Baratin and Françoise Desbordes. *L'analyse linguistique dans l'antiquité classique – I Les théories*. Klincksieck, 1981.

[3] Bertil Malmberg. *Histoire de la linguistique de Sumer à Saussure*. Fondamental. Presses Universitaires de France, 1991.

[4] Jaakko Hintikka. La vérité est-elle ineffable? In *« La vérité est-elle ineffable? » et autres essais*, collection Tiré à part, pages 9–47. Editions de l'éclat, 1994. Traduit de l'anglais par Antonia Soulez et François Schmitz.

[5] Jean van Heijenoort. Logic as calculus and logic as language. In *Selected Essays* [vH85c], pages 11–16.

[6] Jean van Heijenoort. Absolutism and relativism in logic. In *Selected Essays* [vH85c], pages 75–83.

leur niveau qu'on retrouvera cette distinction en logique linéaire :

- Perçue comme un calcul relatif, la logique linéaire est utilisée comme un langage de spécification de processus. Le sens intuitif des connecteurs incite à de telles interprétations : opérateurs de choix, de composition, de duplication etc. Dans cette interprétation, une démonstration est vue comme une exécution d'un processus spécifié par la formule de la logique linéaire démontrée. Selon cette approche que les anglophones nomment *proof-search-as-computation*, le calcul est la recherche de démonstrations.
- Lorsque la logique linéaire est perçue comme le langage des processus, ce sont les démonstrations elles-mêmes qui sont vues comme des processus, et le mécanisme de calcul est alors interne à la logique : l'élimination des coupures. Cette vision est issue de la correspondance entre démonstrations en logique intuitionniste et programmes fonctionnels. Selon cette approche, les démonstrations sont elles-mêmes des programmes à évaluer : c'est le paradigme nommé *proofs-as-programs* en anglais.

La seconde approche est sans doute la plus développée, et elle est en tout cas à l'origine de la logique linéaire, aussi commencerons nous par elle.

1.1.1 Un premier pas vers la logique linéaire : la logique intuitionniste

La logique intuitionniste, introduite par Brouwer au début de ce siècle, voir par exemple [1], se distingue de la logique classique par l'absence du tiers exclu (*tertium non datur*) a déjà, mathématiquement, beaucoup des traits caractéristiques de la logique linéaire. Pour des raisons historiques évidentes, l'objectif poursuivi par Brouwer n'était pas de prendre en compte des phénomènes informatiques, et de plus il semblait même opposé à toute formalisation de la logique. La formalisation, indispensable à l'utilisation informatique de la logique, fut entreprise par Heyting dans les années trente, et la logique intuitionniste a eu un succès certain en informatique : le calcul des séquents ou la déduction naturelle de Gentzen [7,8] ont permis de comprendre et d'enrichir le contenu calculatoire de la logique intuitionniste, et de la mettre en œuvre dans la programmation fonctionnelle typée. En effet, l'absence du tiers exclus donne un contenu calculatoire aux démonstrations, notamment en présence de disjonctions ou de quantificateurs existentiels : on peut, en logique intuitionniste, extraire la proposition qui valide la disjonction

---

[7] Gehrard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39:176–210, 1934. Traduction Française de R. Feys et J. Ladrière: Recherches sur la déduction logique, Presses Universitaires de France, Paris, 1955.

[8] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

démontrée ou la valeur d'un témoin satisfaisant le corps de la formule existentielle établie. L'intérêt informatique de cette propriété saute aux yeux : les objets dont les démonstrations intuitionnistes affirment l'existence sont effectivement calculables.

Dans le calcul des séquents, le passage de la logique classique à la logique intuitionniste, qui modifie le sens de la disjonction et du quantificateur existentiel, ne s'obtient pas par une modification des règles introduisant ces connecteurs, mais par une modification des règles dites structurelles gérant le contexte. Les démonstrations manipulent des séquents, c'est-à-dire des expressions de la forme :

$$H_1, \ldots, H_n \vdash C_1, \ldots, C_p$$

où les H_i et C_i sont des formules, et un séquent est valide lorsque la conjonction de ses hypothèses $H_1 \wedge \cdots \wedge H_n$ entraîne la disjonction $C_1 \vee \ldots \vee C_p$ de ses conclusions. En logique intuitionniste, la structure du séquent est modifiée : il y a au plus une conclusion, $p \leq 1$, et cela change la règle structurelle de contraction. En logique intuitionniste, il devient impossible de contracter, c'est-à-dire d'identifier, deux conclusions identiques, ce que la logique classique autorise par la règle suivante :

$$\frac{\Gamma \vdash \Delta, A, A, \Delta'}{\Gamma \vdash \Delta, A, \Delta'}\ contraction$$

Quel est le sens intuitif d'un énoncé démontrable en logique intuitionniste ? Il est certainement vrai en logique classique, mais qu'apporte la nature intuitionniste de sa démonstration ? En termes de « vérité » et de « raisonnement », on peut dire que la logique intuitionniste est plus une logique de la connaissance que de la vérité : ainsi savoir que « *Apple déposera son bilan ou continuera.* » n'est pas valide d'un point de vue intuitionniste, puisque l'information véhiculée est vide. Par contre si l'on savait laquelle des deux propositions justifie cette disjonction, alors l'information ne serait pas vide, et la proposition serait valide d'un point de vue intuitionniste. Pour prendre un exemple célèbre, la loi de Pierce, c'est-à-dire $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$, n'est pas valide en logique intuitionniste : en fait la seule démonstration possible de cette loi consiste à supposer que A est vrai puis que A est faux et de conclure ; en d'autres termes, chaque implication de cette formule n'exprime pas de conséquence, mais simplement que son antécédent est faux ou sa conclusion vraie quelle que soit la valeur, vrai ou faux, des variables propositionnelles.

Autant dire que les démonstrations sont au cœur de la logique intuitionniste, dont l'interprétation la plus jolie est assurément :

> *Un énoncé s'interprète comme l'ensemble de ses démonstrations, et une implication $A \Rightarrow B$ est vue comme l'ensemble des fonctions qui transforment les démonstrations de A en démonstrations de B.*

Bien sûr, on ne souhaite pas distinguer toutes les démonstrations d'une formule, mais seulement celles qui ne sont pas équivalentes. La notion d'équivalence entre démonstrations retenue est la clôture symétrique et transitive de la réduction ou normalisation des démonstrations, entre d'autres termes la β-réduction, expliquée dans la figure 1.1.1.

Voilà un mécanisme de calcul inhérent aux démonstrations de la logique intuitionniste, mais qui ne permet d'interpréter les formules par leurs démonstrations que si la logique est la logique intuitionniste : en logique classique toutes les démonstrations sont équivalentes[d]. On peut donc dire qu'une démonstration de A est un programme de type A, et que l'évaluer c'est le normaliser. C'est l'isomorphisme de Curry-Howard [2,3] dont sont issus des langages de programmation fonctionnels typés (par exemple CaML, [4], pour citer un produit « maison »). C'est l'exemple type d'une notion de calcul interne à la logique dont l'informatique a su tirer parti.

1.1.2 La logique linéaire : une logique du calcul

La logique linéaire peut se voir comme un raffinement et une symétrisation de la logique intuitionniste. En effet, dans le passage de la logique classique à la logique intuitionniste, qui permet de voir les démonstrations comme des programmes, nous avons perdu la négation involutive et les lois de De Morgan [5,6], principes qui font la simplicité de la logique classique. La logique linéaire réintroduit la négation en décomposant les connecteurs de la logique intuitionniste en des connecteurs plus atomiques. On obtient alors une logique mathématique possédant toutes les propriétés que l'on peut espérer d'une logique. L'article original

d. Voir par exemple [1] où l'argument classique de Joyal est repris. Aujourd'hui de nombreux travaux utilisent néanmoins la logique classique, par exemple ce même article [1] en contrôlant son non déterminisme.

[1] Jean-Yves Girard. A new constructive logic: classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, November 1991.

[2] William A. Howard. The formulae-as-types notion of construction. In J. Hindley and J. Seldin, editors, *To H.B. Curry: Essays on Combinatory Logic, λ-calculus and Formalism*, pages 479–490. Academic Press, 1980.

[3] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[4] Pierre Weis, Maria-Virginia Aponte, Alain Laville, Michel Mauny, and Ascànder Suárez. *The CAML Reference Manual*. INRIA-ENS, 1987.

[5] William Kneale and Martha Kneale. *The development of logic*. Oxford University Press, 3rd edition, 1986.

[6] François Rivenc and Philippe de Rouilhan, editors. *Anthologie de Logique et Fondements des Mathématiques*. Editions Payot, 1992.

FIG. 1.1 – *La réduction ou normalisation en déduction naturelle*

En déduction naturelle, la réduction est définie comme suit :

– soit d une démonstration de $A \Rightarrow B$ sous l'hypothèse A obtenue à partir de d'une démonstration de B, utilisant un certain nombre de fois l'hypothèse A;

$$d : \begin{cases} \dots [A]^{\alpha} \dots [A]^{\alpha} \dots A \dots [A]^{\alpha} \dots \\ \vdots\, d' \\ B \\ \hline A \Rightarrow B \end{cases} \Rightarrow_i^{\alpha}$$

– soit a une démonstration de A ;

$$a : \begin{cases} \vdots\, a \\ A \end{cases}$$

– soit D la démonstration suivante, obtenue à partir de a et d par *modus ponens*;

$$D : \begin{cases} \dots [A]^{\alpha} \dots [A]^{\alpha} \dots A \dots [A]^{\alpha} \dots & \\ \vdots\, d' & \\ B & \vdots\, a \\ \hline A \Rightarrow B & A \\ \hline B \end{cases} \Rightarrow_i^{\alpha} \quad \Rightarrow_e$$

– alors la réduction de D en une étape donne

$$D^{\circ} : \begin{cases} \vdots\, a & \vdots\, a & & \vdots\, a & \\ \dots\ A & \dots\ A & \dots A \dots & A & \dots \\ & & \vdots\, d' & & \\ & & B & & \end{cases}$$

On obtient donc une démonstration plus élémentaire, mais en général plus longue, dès que l'hypothèse A, qui peut se voir comme un lemme, a été utilisée plusieurs fois.

de Girard [1], établit entre autres les propriétés suivantes :

- La logique linéaire possède une sémantique dite des phases pour laquelle elle est complète.
- La logique linéaire satisfait l'élimination des coupures. L'algorithme d'élimination des coupures conduit toujours à une démonstration normale. Celle-ci n'est pas unique, mais les diverses démonstrations obtenues sont très semblables[e], et toute formule d'une telle démonstration est sous-formule de l'une des conclusions. Cet algorithme peut être implanté de manière à ce que la normalisation effectue le plus possible de partage de calculs, ce qui résulte de la syntaxe parallèle en réseaux.
- La logique linéaire possède des sémantiques dénotationnelles, ce qui montre que la normalisation n'identifie pas toutes les démonstrations, et permet d'étudier son comportement calculatoire.
- La logique linéaire permet une traduction fidèle de la logique intuitionniste, et de manière plus subtile, de la logique classique.
- La logique linéaire admet une nouvelle syntaxe graphique pour représenter les démonstrations. Nous reviendrons sur cet aspect ultérieurement.

Comme dans le passage de la logique classique à la logique intuitionniste mentionné précédemment, la logique s'obtient à partir de la logique intuitionniste par une restriction des règles dites structurelles qui gèrent le contexte, et non par la modification des règles qui définissent les connecteurs logiques. Les règles de contraction et d'affaiblissement de la logique classique données ci-dessous sont supprimées.

$$\frac{\Delta,A,A,\Delta' \vdash \Gamma}{\Delta,A,\Delta' \vdash \Gamma}\ \text{contraction}_g \qquad \frac{\Gamma \vdash \Delta,A,A,\Delta'}{\Gamma \vdash \Delta,A,\Delta'}\ \text{contraction}_d$$

$$\frac{\Delta,\Delta' \vdash \Gamma}{\Delta,A,\Delta' \vdash \Gamma}\ \text{affaiblissement}_g \qquad \frac{\Gamma \vdash \Delta,\Delta'}{\Gamma \vdash \Delta,A,\Delta'}\ \text{affaiblissement}_d$$

En d'autres termes, c'est la structure même du contexte, c'est-à-dire la structure sur les formules hypothèses et conclusions d'une démonstration qui détermine en grande partie la logique — alors que pour chacune des règles définissant les connecteurs, la plupart de ces formules sont inactives et simplement recopiées

e. En utilisant un formalisme plus riche, [2] a obtenu une forme normale réellement unique.

[1] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[2] Lorenzo Tortora di Falco. *Réseaux, cohérence et expériences obsessionnelles*. Thèse de doctorat, spécialité logique et fondements de l'informatique, Université Paris 7, 1999.

en l'état. On a déjà mentionné que la restriction conduisant de la logique classique à la logique intuitionniste consistait à n'autoriser qu'une seule conclusion. Néanmoins la logique intuitionniste admet, tout comme la logique classique, une formulation où les contextes sont des ensembles d'hypothèses. Par contre, en logique linéaire, cela n'est pas possible : les contextes sont nécessairement des *multi-ensembles* d'hypothèses et de conclusions ; le nombre d'occurrences de chaque hypothèse ou conclusion est explicitement pris en compte.

Toutefois la logique qui en résulte a de quoi surprendre : en général, les hypothèses ne sont pas utilisables plusieurs fois au cours d'une démonstration. C'est là une grande différence avec les logiques classique et intuitionniste : habituellement les hypothèses peuvent être utilisées *ad libitum*, et s'il en reste d'inutilisées, l'énoncé établi l'est *a fortiori*. Il s'ensuit que des règles usuellement équivalentes pour la conjonction et la disjonction ne le sont plus : on voit apparaître deux disjonctions et deux conjonctions. Intuitivement, la conjonction et la disjonction dites additives correspondent à des choix, tandis que la conjonction et la disjonction dites multiplicatives correspondent à l'assemblage des formules. Afin de recouvrer le pouvoir d'expression de la logique intuitionniste ou classique, la logique linéaire possède deux connecteurs unaires appelés modalités ou exponentiels, « ? » et « ! », qui autorisent les règles structurelles usuelles :

$$\frac{\Delta,!A,!A,\Delta' \vdash \Gamma}{\Delta,!A,\Delta' \vdash \Gamma}\ \text{contraction}_g \qquad \frac{\Gamma \vdash \Delta,?A,?A,\Delta'}{\Gamma \vdash \Delta,?A,\Delta'}\ \text{contraction}_d$$

$$\frac{\Delta,\Delta' \vdash \Gamma}{\Delta,!A,\Delta' \vdash \Gamma}\ \text{affaiblissement}_g \qquad \frac{\Gamma \vdash \Delta,\Delta'}{\Gamma \vdash \Delta,?A,\Delta'}\ \text{affaiblissement}_d$$

Tandis que l'usage contrôlé des règles structurelles complexifie le calcul logique, et par exemple conduit à un calcul propositionnel indécidable, il permet néanmoins une plus fine analyse des opérations de duplication et d'effacement, et c'est sur cela que repose l'analyse des logiques classique et intuitionniste.

Avant d'aborder la place de la logique linéaire en informatique, on peut se demander en quoi elle est une logique au sens usuel, c'est-à-dire en termes de « raisonnement » ou de « vérité ». Il semble qu'il y ait là un glissement de sens qui surprend toujours les collègues de formation plus philosophique : il est bien difficile de donner une présentation intuitive de la logique linéaire sans faire référence à des processus calculatoires. Pour essayer de répondre à ces collègues, divers arguments viennent à l'idée.

– Comme nous l'avons vu, la logique linéaire satisfait les propriétés que l'on peut attendre d'une logique. Elle est donc au moins une logique au sens mathématique, au même titre qu'un entier non standard est appelé un entier

parce qu'il partage les propriétés des entiers usuels. De plus, elle permet d'obtenir des résultats sur les logiques classique et intuitionniste.

– On peut également répondre sans quitter le terrain philosophique : le lien entre logique et calcul n'est pas une nouveauté. En suivant l'histoire de la logique, par exemple [1], on pourra penser à l'ancienne volonté de ramener la logique à un calcul (Leibniz), au calcul propositionnel (Boole), à la théorie de la calculabilité (Church, Turing, Kleene), et, plus récemment, aux notions de calcul interne à la logique intuitionniste (l'isomorphisme de Curry-Howard entre démonstrations intuitionnistes et λ-termes typés). Dans ces conditions, pourquoi ne pas considérer la notion de calcul comme l'objet ou au moins l'un des objets de la logique ? Il ne s'agit en fait que d'élargir le domaine de la logique, et de plus, on y retrouve les logiques usuelles, qui parlent effectivement de « raisonnement » et de « vérité ».

Et finalement, qu'importe l'étiquette ? D'autant que ce glissement de sens, peut-être intrigant pour le philosophe, profite assurément à l'informatique : lorsque les hypothèses sont vues comme des ressources, par exemple si elles correspondent à la présence de jetons dans une place d'un réseau de Petri, il est normal qu'elles ne soient pas inépuisables. Effectivement, la logique linéaire peut directement représenter leur consommation, leur production, c'est-à-dire les changement d'états ou transitions par des implications, ce que même la logique intuitionniste ne peut faire.

Voici donc une logique qui semble adéquate pour décrire le calcul, mais jusqu'où va cette analogie ? Comme on l'a dit plus haut, il y a deux manières d'envisager le calcul en logique linéaire : soit on utilise la notion interne de calcul, l'élimination des coupures, soit on se sert de la logique linéaire pour décrire le calcul. Précisons cela.

1.1.3 Logique linéaire et informatique

Comme en témoignait une récente rencontre intitulée « *Linear logic and applications* » [2], si l'on souhaite organiser les utilisations informatiques de la logique linéaire, on peut obtenir la classification suivante, aux frontières perméables :

1. Logique linéaire et programmation
   (a) Programmation fonctionnelle, en utilisant la notion de calcul interne, c'est-à-dire l'élimination des coupures. Le modèle obtenu prend naturellement en compte le partage de calculs lors de l'évaluation.

---

[1] William Kneale and Martha Kneale. *The development of logic*. Oxford University Press, 3rd edition, 1986.

[2] Valeria de Paiva and Eike Ritter, editors. *Linear logic and applications*, 1999. Dagstuhl seminar – forthcoming proceedings.

(b) Programmation logique, en utilisant la recherche de démonstrations comme mécanisme de calcul, et la logique linéaire comme langage de spécification. C'est dans cette catégorie que je range la modélisation de la concurrence : π-calcul, réseaux de Petri, etc.

2. Logique linéaire et grammaire.

Le dernier aspect mérite quelques précisions : il peut surprendre, et de plus, l'essentiel de nos travaux concrètement utilisables lui sont consacrés. On notera que nous avons préféré ici le mot « grammaire » aux expressions « linguistique informatique », « linguistique computationnelle », « traitement automatique des langues » : d'une part il s'agit surtout de syntaxe, et d'autre part, je n'aime guère le clivage hexagonal entre étude des grammaires formelles, qui relève de l'informatique théorique, de la logique ou de l'algèbre, et traitement automatique des langues, qui relève de l'intelligence artificielle et du génie logiciel. Néanmoins, nous utiliserons aussi les autres expressions, surtout dans un contexte plus linguistique, car le terme « grammaire » y désigne plutôt l'ensemble des règles normatives de la syntaxe, peu concernées par ses structures et ses mécanismes.

Ce thème utilise la logique linéaire comme langage de spécification, et pourrait donc être regroupé avec le précédent point 1(b). Mais d'une part, on va également utiliser les démonstrations elles-mêmes (et non de simples formules) comme des spécifications, et d'autre part, le traitement automatique des langues est un sous-domaine assez particulier : on s'appuie sur une autre discipline, la linguistique. Surtout, à la différence des calculs de processus, on ne peut proposer de modifier le modèle décrit, puisqu'il n'est pas un modèle mais une réalité non modifiable, la langue. Techniquement néanmoins, cette application de la logique linéaire est semblable aux autres. Il faut cependant signaler une étonnante particularité : cette application de la logique linéaire a existé avant la logique linéaire. Le calcul de Lambek, qui formalise logiquement les grammaires catégorielles date de la fin des années cinquante [1,2], et se trouve être un fragment de la logique linéaire non commutative [3,4,5]. Il s'agit donc plutôt d'un renouveau des grammaires catégorielles auquel la logique linéaire participe que d'une totale nouveauté.

---

[1] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[2] Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, 1961.

[3] Vito Michele Abrusci. Phase semantics and sequent calculus for pure non-commutative classical linear logic. *Journal of Symbolic Logic*, 56(4):1403–1451, December 1991.

[4] Dirk Roorda. *Resource logic: proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam, 1991.

[5] Christian Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.

Nous allons donc décrire brièvement les différents aspects informatique de la logique linéaire, et plus particulièrement l'utilisation de la logique linéaire comme langage de spécification. En effet ce dernier aspect est plus proche des applications linguistiques, et celles-ci nous intéressant plus particulièrement, elles feront l'objet d'un paragraphe propre.

Jusqu'à présent, la logique linéaire me semble avoir obtenu plus de résultats frappants pour le modèle de calcul interne qu'elle offre, que comme langage de spécification du calcul, mais la seconde utilisation prend de l'ampleur. Du reste nous développerons davantage notre présentation de la logique linéaire comme langage de spécification, car c'est dans cette tendance que s'inscrivent les applications grammaticales de la logique linéaire.

(a) *Logique linéaire et évaluation des langages fonctionnels*
Cette utilisation de la logique linéaire exploite le plongement fidèle de la logique intuitionniste dans la logique linéaire. Plus spécifiquement il s'agit de décomposer l'implication intuitionniste en deux connecteurs linéaires. On commence par considérer $!A$ qui autorise la formule A à être dupliquée ou effacée, puis on utilise l'implication linéaire : $(A \to B)^{\#} = (!A^{\#}) \multimap B^{\#}$ où $^{\#}$ désigne la traduction. En utilisant les subtilités de l'élimination des coupures en logique linéaire, et en particulier son fonctionnement dans les réseaux de démonstration, on obtient une procédure d'évaluation avec partage des λ-termes (celle de John Lamping [6]) dont on sait montrer qu'elle est optimale [7]. Sans trop de risque, on peut affirmer que c'est ce genre d'application de la logique linéaire, basée sur l'étude fine de l'élimination des coupures, qui est le plus abouti. La portée informatique de ces résultats repose surtout sur la logique linéaire intuitionniste : les symétries classiques qu'expriment la négation et la disjonction y restent des outils et n'acquièrent pas vraiment de sens intuitif.

(b) *La logique linéaire comme langage de spécification*
Comme on l'a dit, le comportement des connecteurs de la logique linéaire ne se formule intuitivement qu'en termes de processus. Par suite il est normal de l'utiliser comme langage de spécification, en donnant corps à cette intuition. Par exemple, l'usage contrôlé des règles structurelles de contraction et d'affaiblissement permet de modéliser une transition, c'est-à-dire un changement d'état par l'implication linéaire, ce dont aucune autre logique bien faite ne sait rendre compte. En effet, de A et $A \multimap B$ on déduit, comme

[6] John Lamping. An algorithm for optimal lambda-calculus reductions. In *Proceedings of the Seventeenth ACM Symposium on Principles of Programming Languages*, pages 16–30. ACM, ACM Press, January 1990.

[7] Georges Gonthier, Jean-Jacques Lévy, and Martin Abadi. The geometry of optimal lambda reduction. In *Principles Of Programming Languages*, pages 15–26, 1992.

d'habitude, B, mais, à la différence de ce qui se produit dans les logiques usuelles, l'hypothèse A n'est plus valide après la transition. C'est une autre source de son succès en informatique théorique, et nombre de travaux utilisent la logique linéaire comme formalisme de description du calcul parallèle : planification, réseaux de Petri, π-calcul ... [1,2,3,4]. On obtient alors des résultats satisfaisants où cette fois le processus est décrit par une formule, tandis que son exécution correspond à la construction d'une démonstration de la dite formule. Les travaux en ce sens abondent, et on pourra se référer aux exemple repris dans [5], ou, dans le cas des réseaux de Petri, à l'article de synthèse [3]. On trouvera dans ce mémoire au chapitre 6 une modélisation des réseaux de Petri différente, basée sur une extension de logique linéaire qui mêle connecteurs commutatifs et non commutatifs.

C'est à cette famille qu'il convient à mon avis de rattacher les travaux de Jean-Marc Andreoli [6] ou de Dale Miller [7] sur la programmation logique en logique linéaire. Partant du constat que la logique linéaire admet des démonstrations uniformes (où les règles sont appliquées dans un certain ordre), on voit que la logique linéaire peut se voir, presque sans modification, comme un langage de programmation logique. On peut ainsi représenter divers calcul de processus, et la logique linéaire sert alors de langage de spécification.

Cependant, il arrive que la formulation en logique linéaire reste un codage ou que la formalisation en logique linéaire et le modèle qu'elle est censée décrire divergent. Il se peut que la logique linéaire n'aide pas à résoudre les questions posées par le modèle ; par exemple, à quel type de démonstration correspond l'exécution la plus rapide d'un réseau de Petri formalisé en logique linéaire ? Réciproquement, le modèle à décrire peut ne pas être une bonne interprétation de la logique linéaire, par exemple s'il identifie des

---

[1] Marcel Masseron, Christophe Tollu, and Jacqueline Vauzeilles. Generating plans in linear logic: I. actions as proofs. *Theoretical Computer Science*, 113:349–370, 1993.

[2] Vijay Gehlot and Carl Gunter. Nets as tensor theories. In G. De Michelis, editor, *Applications of Petri nets*, 1989.

[3] Uffe Engberg and Glynn Winskel. Completeness results for linear logic on Petri nets. *Annals of Pure and Applied Logic*, 86(2):101–135, 1997.

[4] Gianluigi Bellin and P. J. Scott. On the π-calculus and linear logic (with an introduction by S. Abramsky). Technical report, University of Edimbourgh, 1992.

[5] Anne Sjerp Troelstra. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. CSLI, 1992. (distributed by Cambridge University Press).

[6] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 1992.

[7] Dale Miller. Forum: a multiple conclusion specification language. *Theoretical Computer Science*, 165:201–232, 1996.

connecteurs différents, ou valide des lois fausses. Dans ce cas, on se trouve dans une sorte d'impasse : on ne peut modifier le système logique sans en perdre les qualités, et il n'est pas facile de modifier les principes déterminés par l'expérience, l'intuition et le savoir informatique — à moins d'avoir pleinement saisi le sens intuitif de la logique et de proposer un autre modèle sur cette base. On notera que le logicien préfère le plus souvent que les démonstrations soient finies, et par conséquent les comportements infinis, chers au parallélisme, restent hors d'atteinte. Tout comme dans l'utilisation du modèle de calcul interne que la logique linéaire offre, son utilisation comme langage de spécification fait encore peu référence à la logique linéaire *classique*, pourtant plus élégante mathématiquement.

1.1.4 Logique linéaire et grammaire

Comme on l'a dit, cette application se rapproche plus de la logique linéaire comme langage de spécification. Des formules, ou éventuellement des démonstrations partielles, spécifient le comportement des terminaux de la grammaire formelle, c'est-à-dire des mots de la langue. Analyser une phrase dans ce cadre, c'est construire une démonstration du symbole initial de la grammaire à partir des spécifications en logique linéaire du comportement des mots de la phrase. Cette application est très naturelle si l'on admet que la logique linéaire peut se voir comme un modèle général du calcul : tout comme dans l'étude de la programmation, le traitement des langues met en œuvre un calcul symbolique sur des données structurées. Il s'agit là d'une hypothèse linguistique à laquelle nous souscrivons, hypothèse selon laquelle le langage est structuré et que ce sont ses structures mêmes qui permettent la communication.

La différence provient évidemment de la nature linguistique des données : c'est une réalité extérieure dont les structures échappent encore. Aussi faut-il prendre en compte les descriptions de la langue que nous fournissent les linguistes, et les résultats des recherches linguistiques influencent donc le choix des structures et algorithmes associés. Par exemple, il faut d'une part un calcul suffisamment général pour que seules les spécifications des mots, *et non les règles*, dépendent de la langue considérée, mais d'autre part il faut un calcul efficace au moins pour le fragment logique utilisé. En effet, si on les compare aux déductions et aux autres tâches que nous accomplissons, les mécanismes calculatoires mis en œuvre dans les langues sont spécifiques et particulièrement efficaces. Dans cette faculté de langage dont nous sommes dotés, il ne faut pas omettre, aux côtés de l'analyse et de la production d'énoncés, l'apprentissage de la langue, c'est-à-dire la spécialisation des règles et l'acquisition du lexique, car le processus d'acquisi-

tion doit aussi se traduire par des algorithmes efficaces [1,2,3,4].

Comme la logique linéaire décompose les connecteurs usuels en des connecteurs plus élémentaires, elle autorise divers fragments et variantes, notamment non commutatives [5,6,7,8,9,10,11]. Cette souplesse, et la généralité du modèle de calcul qu'elle propose, en font un bon candidat pour isoler et étudier le calcul qu'utilise le traitement de la syntaxe des langues. Il s'agit là d'une application de la logique linéaire à la linguistique théorique, mais les modèles construits proposent un traitement algorithmique et suggèrent de ce fait des applications concrètes : analyse syntaxique et construction de représentations sémantiques, génération de textes à partir de représentations sémantiques, inférence grammaticale etc.

1.1.5 Une vision particulière de la logique linéaire

Le point de vue sur la logique linéaire que nous avons présenté n'est peut-être pas standard, même s'il est développé par plusieurs équipes de recherche, notamment à Nancy et à Utrecht.

Il est motivé par l'utilisation grammaticale de la logique linéaire que nous avons en vue, et l'objet principal de cette présentation de la logique linéaire était de justifier son emploi dans la formalisation de la syntaxe des langues. C'est pourquoi la seconde partie de ce chapitre consistera à voir si, en partant de considé-

---

[1] Steven Pinker. *The Language Instinct*. Penguin Science, 1994.

[2] Jean-Yves Pollock. *Langage et cognition: le programme minimaliste de la grammaire générative*. Presses Universitaires de France, Paris, 1997.

[3] Ray Jackendoff. *The Architecture of the Language Faculty*. Number 28 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1995.

[4] Juan Uriegereka. *Rhyme and reason: an introduction to minimalist syntax*. MIT Press, 1998.

[5] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[6] Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, 1961.

[7] Vito Michele Abrusci. Phase semantics and sequent calculus for pure non-commutative classical linear logic. *Journal of Symbolic Logic*, 56(4):1403–1451, December 1991.

[8] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

[9] Paul Ruet. *Logique non-commutative et programmation concurrente*. Thèse de doctorat, spécialité logique et fondements de l'informatique, Université Paris 7, 1997.

[10] Vito Michele Abrusci and Paul Ruet. Non-commutative logic I: the multiplicative fragment. *Annals of Pure and Applied Logic*, 1999. http://www.uniroma3.it/reseaux9798.ps.

[11] Christian Retoré. *Réseaux et Séquents Ordonnés*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, février 1993.

rations linguistiques, on peut également justifier l'utilisation grammaticale de la logique linéaire.

1.2 Syntaxe formelle du langage naturel

La linguistique informatique étant un sujet extrêmement varié dans ses objectifs et ses méthodes, il convient de délimiter assez précisément notre domaine d'étude. Nos travaux en ce domaine sont une application de la logique, à la linguistique informatique, et comme souvent lorsqu'il s'agit d'applications à la linguistique, le mot « application » renvoie à deux sens assez différents, mais qui ne s'excluent pas l'un l'autre :

- Il s'agit d'une part de concevoir ou de développer des outils pour le traitement automatique des langues, les modèles et les algorithmes étant issus de la théorie qu'on applique.
- Il s'agit d'autre part de formaliser des théories linguistiques, afin de confirmer ou infirmer certains choix faits par les linguistes, par exemple s'ils sont faits sur des critères algorithmiques.

Je ne suis pas linguiste de formation, et à part des cours suivis à ESSLLI en 97 et 99, et à Paris 7 dans ma jeunesse, le point de vue ici présenté est essentiellement issu de lectures et de conversations avec des collègues linguistes ou informaticiens-linguistes, en particulier Michael Moortgat, Edward Stabler, et surtout Alain Lecomte, puisque mes travaux en ce domaine sont essentiellement le fruit de notre collaboration.

1.2.1 Le rôle central de la syntaxe

La linguistique, même informatique, est un domaine plus que vaste : entre l'étude statistique du vocabulaire dans l'œuvre d'un écrivain et l'interprétation des syntagmes nominaux référentiels, il y a peut-être des rapports, mais les méthodes ne sont certainement pas les mêmes. Dans la division traditionnelle « phonétique, morphologie, syntaxe, sémantique, pragmatique », voir par exemple [12], on se concentrera sur la syntaxe, en gardant un œil sur la sémantique. Il est communément admis que chacun de ces différents aspects de la langue dépend de ses voisins, et par conséquent, une application réelle, par exemple un logiciel de traduction assisté par ordinateur, ou une étude purement linguistique se situent dans une vision générale de la langue.

Un autre problème auquel le traitement des langues se trouve presque toujours confronté est la nécessité de prendre en compte le discours dans sa totalité, et non

[12] John Lyons. *Introduction to theoretical linguistics*. Cambridge university press, 1968.

de simples phrases. Du point de vue linguistique, au contraire, l'étude du discours relève de recherches de nature différente, orthogonales à la classification traditionnelle donnée ci-dessus, même si la pragmatique est un des éléments de l'étude du discours. Comme souvent en syntaxe on n'abordera pas ces questions, même si des modèles utilisés en syntaxe ont été appliqués au discours, par exemple [1,2].

Le sujet de nos travaux est donc la syntaxe, et cela s'accorde avec le rôle central que lui attribue la linguistique de l'école de Chomsky, d'ailleurs appelée *grammaire* générative. Précisons que l'adjectif *central* doit ici être pris dans son sens spatial : la syntaxe est le point d'articulation entre la forme phonologique que l'on perçoit ou produit, et la forme logique que l'on comprend ou exprime, ce qui correspond à une position médiane dans la classification traditionnelle donnée ci-dessus[f]. L'originalité de notre approche, celles des grammaires catégorielles est d'utiliser la logique pour modéliser la syntaxe des langue. En effet, le lecteur, s'il est linguiste, s'attend plutôt à des utilisations de la logique pour la *sémantique*... mais les logiques considérées ici nous permettent, parce qu'elles sont sensibles aux ressources et non commutatives, de traiter de la *syntaxe* des langues. Bien entendu, l'un des intérêts de l'approche logique de la syntaxe est le lien avec la sémantique. Il s'établit aisément grâce aux étroites relations entre les logiques des ressources et les logiques plus habituelles, et nous serons à même de définir une interface aisée avec une sémantique simplifiée. On notera que la seule sémantique considérée jusqu'ici par notre communauté est la structure prédicative des énoncés, et non ce qu'on appelle sémantique lexicale qui met en relation les sémantiques des diverses entrées lexicales. Une telle sémantique logique est très voisine de la syntaxe et en fait même partie dans la grammaire générative.

L'autre discipline avec laquelle la syntaxe doit composer est la morphologie inflectionnelle, qui régit les phénomènes d'accord. Nous passerons cet aspect sous silence, pour deux raisons.

D'une part, dans les grammaires catégorielles classiques, les traits d'accord (genre, nombre, personne, temps, mode etc.) sont gérés comme dans les autres formalismes grammaticaux par des traits qui s'unifient. On notera cependant un atout des grammaires catégorielles : les traits sont toujours atomiques, et les algorithmes sont *a priori* plus efficaces en l'absence de structures de traits récursives. Cette approche qui s'inspire des travaux sur les grammaires d'unification a été

f. Cependant nombre de grammairiens génératives s'accommoderaient fort bien de l'autre sens de l'adjectif *central*.

[1] Claire Gardent and Bonnie Webber. Describing discourse semantics. In *Proceedings of the 4th TAG+ Workshop*, University of Pennsylvania, Philadelphia, 1998.

[2] Denys Duchier and Claire Gardent. Tree descriptions, constraints and incrementality. In H. Bunt, R. Muskens, and E. Thijsse, editors, *Computing Meaning, Volume 2*, Studies in Linguistics and Philosophy. Kluwer Academic Publishers, 2001.

menée à bien par Dirk Heylen [3]. On peut également envisager, comme le fait Nissim Francez [4] de décrire les opérations catégorielles à l'intérieur des structures de traits. Ces bonnes paroles ne doivent pas cacher un écueil : puisque les grammaires catégorielles disposent de non-terminaux structurés, les catégories, comment les traits se répartissent-ils sur les catégories participant à une catégorie composée ? En d'autres termes, quelle est la manière *spécifique* dont les grammaires catégorielles peuvent traiter du lien entre morphologie inflectionnelle et syntaxe ? À ce jour je n'ai pas croisé de réponse qui me satisfasse pleinement, mais je n'ai pas non plus de solution à proposer.

D'autre part, ces phénomènes sont également absents des grammaires inspirées du minimalisme. Pour l'instant nous n'avons pas abordé ces questions, mais elles pourraient l'être selon la manière minimaliste, qui consiste à introduire des catégories, c'est-à-dire des non-terminaux, pour l'accord. Par exemple, l'accord du verbe avec son sujet et avec son objet est régi par les catégories Agr_s et Agr_o, dont les traits, ici aussi atomiques, doivent s'unifier lors des déplacements.

Nous n'emploierons pas non plus toutes les méthodes possibles. Parmi les publications en linguistique informatique, on relève deux types d'approches : calcul symbolique et traitement statistique. On n'envisagera que le premier mode d'appréhension, qui, par nature, est plus accessible à la logique. C'est un peu à regret qu'on n'inclut pas de traitement statistique. A notre avis, la syntaxe est essentiellement un phénomène calculatoire, que l'on décrira logiquement, mais qui est augmenté de régularisations et d'automatismes de nature statistique. Mais malheureusement, à l'heure actuelle personne ne sait combiner calcul symbolique et traitement statistique, malgré quelques tentatives dans le cas des réseaux de neurones, et, plus récemment, dans les grammaires catégorielles [5].

1.2.2 Applications envisagées

Un traitement de la seule composante syntaxique de la langue ne peut donner lieu à des applications réelles telles la traduction automatique. Il permet néanmoins de réaliser des modules identifiables qui coopèrent dans les applications réelles : analyse syntaxique produisant des représentations sémantiques, génération d'énoncés à partir de représentations sémantiques, et acquisition automatique de grammaires. Le dernier point est un module un peu à part venant compléter l'analyse syntaxique. Le principal défaut des grammaires logiques utilisées, par

[3] Dirk Heylen. *Types and Sorts – Resource Logic for Feature Checkimg.* PhD thesis, Universiteit Utrecht, 1999.

[4] Nissim Francez. On fibring feature logics with concatenation logics. In Lecomte et al. [LLP99], pages 200–211.

[5] Guillaume Bonfante and Philippe de Groote. Stochastic lambek grammars. In *Formal Grammar – Mathematics of Language*, 2001.

exemple par rapport aux grammaires de dépendance [1,2], est qu'elles ne sont pas robustes et échouent sur beaucoup d'énoncés dans un corpus réel. L'une des raisons est que n'importe quel corpus contient des mots non prévus ou incorrects : un module d'acquisition permet alors de compléter automatiquement le lexique et de reconnaître et d'analyser les énoncés du corpus. Notons que les grammaires utilisées admettent de bons algorithmes d'apprentissage, ce qui n'est pas si courant.

Analyse syntaxique et production de représentations sémantiques

L'analyse syntaxique seule est de peu d'intérêt : la seule correction d'un énoncé n'est guère utile, hormis pour des exercices immédiats lors de l'apprentissage d'une langue. Construire la structure syntaxique est déjà plus intéressant, car cela permet d'extraire la structure prédicative, donc une des composantes de la sémantique de l'énoncé. On notera que, dans les formalismes les plus répandus, DCG, TAG, HPSG[g], voir par exemple [3,4], ce passage à une représentation sémantique est ardu : il n'existe pas d'algorithme uniforme s'appliquant récursivement à la structure de l'analyse pour produire la structure prédicative. Par contre dans les formalismes que nous utilisons ce calcul des représentations sémantiques est toujours automatisable, et pour nous l'analyse syntaxique inclura toujours la construction d'une structure syntaxique permettant le calcul d'une représentation sémantique.

Génération

L'inverse de l'analyse syntaxique est la génération automatique d'énoncés à partir de représentations sémantiques. Si cela est tout à fait possible et fonctionne bien, c'est, d'un point de vue applicatif, un peu moins satisfaisant que l'analyse syntaxique pour la raison suivante : les représentations sémantiques utilisées sont très proches des structures syntaxiques puisque des constantes logiques associées aux mots modélisent leur sens. Le choix des mots, ingrédient important de la

g. DCG: *Definite Clause Grammars*.
TAG: *Tree Adjoining Grammars*.
HPSG: *Head-Driven Phrase Structure Grammars*.

[1] Lucien Tesnière. *Éléments de syntaxe structurale*. Éditions Klincksieck, 1959. 5e édition: 1988.

[2] Igor Mel'čuk. *Dependency syntax – theory and practice*. Linguistics. State University of New York Press, 1988.

[3] Carl Pollard and Ivan A. Sag. *Head Driven Phrase Structure Grammar*. Center for the Study of Language and Information, Stanford, CA, USA, 1987. (distributed by Cambridge University Press).

[4] Anne Abeillé. *Les nouvelles syntaxes*. Armand Colin, 1993.

génération doit donc être traité en aval de ce que notre approche permet. Il en est de même pour la construction de représentations sémantiques à partir des données que l'on souhaite exprimer.

Acquisition automatique de grammaires ou inférence grammaticale

Ce module retient depuis peu notre attention. D'un point de vue pratique, ce module est une étape préliminaire à l'analyse syntaxique, en ce sens que, sans lui, l'analyse syntaxique ne peut être robuste sur des corpus réels, ceux-ci comportant souvent des néologismes et des erreurs. La construction automatisée de grammaires, ou inférence grammaticale, est aussi un des aspects essentiels de l'application de nos techniques à la modélisation des théories linguistiques.

En effet, nous nous situons dans une perspective chomskyenne et l'apprentissage de sa langue par l'enfant est la raison essentielle de nombreux choix et partis pris de la grammaire générative. Cette dernière a cherché à expliquer le paradoxe de l'apprentissage, c'est-à-dire le fait qu'un enfant apprend rapidement une grammaire très complexe à partir d'un nombre assez restreint d'exemples [5,6,7,8] ; ce constat conduit à supposer l'existence d'une grammaire universelle dont les grammaires des différentes langues sont des instances définies par la valeur de paramètres. Outre l'aspect pratique mentionné précédemment, les nombreuses études psycholinguistiques sur ce sujet, qui utilisent des arguments algorithmiques pour justifier le modèle d'apprentissage, sont une raison supplémentaire pour formaliser le mécanisme d'apprentissage grammatical : on peut ainsi espérer valider ou infirmer les arguments utilisés par les linguistes.

1.2.3 Grammaires catégorielles et calcul de Lambek

Venant de la logique linéaire, le lien le plus standard avec la linguistique est assurément le calcul de Lambek [9,10]. Ce calcul, qui existait bien avant la logique

---

[5] Steven Pinker. *The Language Instinct*. Penguin Science, 1994.

[6] Steven Pinker. Language acquisition. In Gleitman and Liberman [GL95], chapter 6, pages 135–182.

[7] Steven Pinker. Why the child holded the baby rabbits. In Gleitman and Liberman [GL95], chapter 5, pages 107–133.

[8] L.R. Gleitman and E.L. Newport. The invention of language by children: Environmental and biological influences on the acquisition of language. In Gleitman and Liberman [GL95], chapter 1, pages 1–24.

[9] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[10] Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, 1961.

linéaire, peut être utilisé comme une grammaire ainsi : un lexique associe à chaque mot m un ensemble fini de formules logiques $\mathrm{Lex}(m)$, et un énoncé $m_1 \cdots m_n$ appartient au langage lorsqu'il existe pour chaque mot m_i un type $T_i \in \mathrm{Lex}(m_i)$ de sorte que le séquent

$$T_1, \ldots, T_n \vdash S$$

soit démontrable dans le calcul de Lambek. Une analyse syntaxique de l'énoncé est une telle démonstration. Ces grammaires formelles se distinguent de leurs consœurs par les avantages et inconvénients suivants : d'une part elles bénéficient d'un lien aisé avec la sémantique de Montague, elles sont totalement lexicalisées et se formalisent élégamment ; mais d'autre part leur faible capacité générative et leur absence de souplesse sont un réel handicap.

Qualités formelles

Qu'il y a-t-il de séduisant dans l'approche catégorielle ? Pour le logicien, le calcul de Lambek est un calcul parfait, et de surcroît fort élégant de par son minimalisme. Pour le linguiste, le calcul de Lambek est une formalisation logique des grammaires catégorielles qui sont une sorte d'alternative aux grammaires syntagmatiques. Initialement conçues par Ajdukiewicz [1], pour la sémantique des langues et du langage de la logique, elles s'appuient sur la notion philosophique de catégorie. Les grammaires catégorielles ont été ensuite adaptées à la syntaxe des langues comme un calcul non commutatif de fractions par Bar-Hillel [2]. C'est Joachim Lambek [3,4] qui les étend et les présente comme un calcul logique : tous les principes discutés pour étendre les grammaires catégorielles (montée de type, loi de Geach) sont alors inclus puisqu'ils sont tous des conséquences du calcul logique. Lorsqu'elles sont mentionnées par Chomsky dès 1963 [5] les grammaires de Lambek sont vues comme une alternative *formelle* aux grammaires syntagmatiques. Il conjecture d'ailleurs qu'elles sont faiblement équivalentes aux grammaires non contextuelles, ou algébriques, ce qui sera effectivement, et fort joli-

[1] Kazimierz Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935. (English translation in [McC67], pages 207–231).

[2] Yehoshua Bar-Hillel. A quasi arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

[3] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[4] Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, 1961.

[5] Noam Chomsky. Formal properties of grammars. In *Handbook of Mathematical Psychology*, volume 2, pages 323 – 418. Wiley, New-York, 1963.

ment, établi par Pentus en 92 [6].

Lien avec la sémantique, en particulier de Montague

Outre les algorithmes d'apprentissage dont nous reparlerons, le principal avantage des grammaires de Lambek est leur interface aisée avec la sémantique de Montague [7]. La sémantique de Montague n'est pas, à mon avis, une sémantique au sens plein. En effet, le contenu lexical n'est pas pris en compte, les sèmes sont sans aucune relation : ce sont de simples constantes logiques. Il n'y a rien de comparable aux structures conceptuelles ou à l'organisation du sens dont parle Ray Jackendoff [8] ni même de la formalisation d'une partie de ces idées par James Pustejovsky [9]. Ce dernier cherche à articuler sémantique lexicale et sémantique prédicative, ce qui ouvre la voie à des travaux de sémantique formelle plus substantiels, comme par exemple ceux d'Evelyne Jacquey [10]. Dans une approche issue des grammaires de dépendances plutôt que de la grammaire générative, un programme similaire est proposée par la théorie « Sens-Texte » d'Igor Mel'čuk[11].

Néanmoins la sémantique de Montague, toute critiquable qu'elle soit, fait bien ce qu'elle est censée faire, et c'est déjà beaucoup : la structure prédicative des énoncés, la coréférence, la portée des quantificateurs, et même les phénomènes d'intentionnalité sont aisément calculés, et on observera au passage que ces aspects de la sémantique font, selon Chomsky, partie de la syntaxe. Les grammaires de Lambek, comme cela sera expliqué au chapitre suivant, permettent de suivre la construction de la représentation sémantique règle à règle et la correspondance est un simple plongement d'une logique dans une autre.

Rapports avec d'autres formalismes

LFG, GPSG, HPSG Les grammaires catégorielles sont relativement inconciliables avec les grammaires d'unification basées sur des architectures hors-contexte,

---

[6] Mati Pentus. Lambek grammars are context-free. In *Logic in Computer Science*. IEEE Computer Society Press, 1993.

[7] Richmond Thomason, editor. *The collected papers of Richard Montague*. Yale University Press, 1974.

[8] Ray Jackendoff. *The Architecture of the Language Faculty*. Number 28 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1995.

[9] James Pustejovsky. *The generative lexicon*. M.I.T. Press, 1995.

[10] Evelyne Jacquey. *Ambiguïtés lexicales et traitement automatique des langues: modélisation de la polysémie logique et application aux déverbaux d'action ambigus en français*. Thèse de doctorat, spécialité informatique, Université Nancy 2, décembre 2001.

[11] Igor Mel'čuk. Vers une linguistique sens-texte. leçon inaugurale, collège de france, 1997. http://www.fas.umontreal.ca/ling/olst/melcuk/.

telles LFG, GPSG et HPSG [1]. C'est un peu regrettable, car ces grammaires sont très populaires et de nombreuses données sont disponibles. L'unification en soi ne pose aucun problème, et d'ailleurs les grammaires catégorielles font elles aussi appel à des traits qu'elles unifient pour prendre en compte la morphologie inflectionnelle. C'est l'architecture générale de ces grammaires qui ne correspond pas aux grammaires catégorielles. Si l'on envisageait de représenter de telles grammaires dans les grammaires catégorielles, on ne verrait pas comment traiter les structures de traits récursives. Ceci est l'expression concrète d'un problème plus général : les grammaires HPSG sont complètes au sens de Turing (ou de type 0 dans la hiérarchie de Chomsky) et on ne voit pas comment les représenter dans un formalisme décidable, ni quel fragment isoler. Cette différence est plutôt à l'avantage des grammaires catégorielles, car il est linguistiquement clair que la correction d'un énoncé est décidable en temps fini, voire linéaire, par un locuteur. Une autre distinction s'oppose au rapprochement des grammaires d'unification et des grammaires catégorielles : l'ordre des mots est sans rapport avec la structure en constituants même si le « / » (*slash*) des HPSG rappelle le « / » des grammaires catégorielles. Cela est discutable linguistiquement : selon Noam Chomsky [2] et Richard Kayne [3], le lien est très étroit, tandis que selon Tesnière [4], Mel'čuk [5] le lien est bien plus souple.

TAG Les TAGs [6,1] sont bien plus proches des grammaires catégorielles, surtout lorsqu'on considère non seulement les formules ou types, mais aussi et surtout les démonstrations ou arbres d'analyse. Ce rapprochement récent et fructueux a deux objectifs : du point de vue des TAGs, il s'agit d'obtenir une interface plus simple avec la sémantique de Montague, et du point de vue des grammaires catégorielles, il s'agit d'introduire l'opération d'adjonction qui augmente considérablement le pouvoir d'expression.

---

[1] Anne Abeillé. *Les nouvelles syntaxes*. Armand Colin, 1993.

[2] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

[3] Richard Kayne. *The Antisymmetry of Syntax*. Number 25 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1994.

[4] Lucien Tesnière. *Éléments de syntaxe structurale*. Éditions Klincksieck, 1959. 5e édition: 1988.

[5] Igor Mel'čuk. *Dependency syntax – theory and practice*. Linguistics. State University of New York Press, 1988.

[6] Aravind Joshi, Leon Levy, and Masako Takahashi. Tree adjunct grammar. *Journal of Computer and System Sciences*, 10:136–163, 1975.

Le rôle des réseaux de démonstration

Le choix du formalisme dans lequel on décrit une théorie linguistique a son importance. Comme il s'agit ici de syntaxe logique, on dispose du calcul des séquents, de systèmes d'axiomes à la Hilbert, de la déduction naturelle, et des réseaux de démonstration. Puisque l'idée sous-jacente à ces modèles grammaticaux est qu'une analyse syntaxique, c'est une démonstration, il convient d'avoir des structures où la structure en constituants se voit, c'est-à-dire des structures d'arbres. Il est également préférable qu'il n'y ait pas plusieurs analyses formelles pour une seule et même structure en constituants. Les déductions à la Hilbert n'ont pas une claire structure d'arbre, et le calcul des séquents, outre qu'il est fortement redondant, distingue de nombreuses démonstrations équivalentes. Seuls restent la déduction naturelle et les réseaux de démonstration. Une remarque s'impose : pour les systèmes intuitionnistes utilisés en linguistique, il n'y a presque pas de différence. Par exemple, pour le calcul de Lambek, si la règle d'élimination du produit n'est pas utilisée, alors il y a correspondance bijective entre déductions naturelles et réseaux. Donc l'un et l'autre sont des systèmes satisfaisants, mais les réseaux permettent de conserver cette unicité de l'analyse formelle lorsqu'on étend le système logique, que ce soit pour y inclure le produit ou pour considérer des variantes classiques.

Mentionnons ici que la syntaxe des réseaux a eu quelque succès dans la communauté linguistique[h]. Le premier exemple se rapporte à un domaine assez naturel en logique : il concerne les techniques de recherche de démonstration (d'analyse syntaxique) que permettent ce formalisme, notamment avec des techniques de programmation dynamique [8,9]. Le second exemple est plus surprenant : il concerne une question de *performance*, à savoir la mesure de la complexité qu'il y a à analyser un énoncé. Les axiomes reliant des atomes situés de part et d'autre de deux conclusions consécutives du réseau représentent les dépendances non résolues, c'est-à-dire les syntagmes attendus lorsque le locuteur se trouve à cet endroit de la phrase. Cela a permis tout d'abord à Mark Johnson [10] de donner une mesure

h. C'est essentiellement à travers les travaux de Dirk Roorda (Amsterdam) [7] et de l'équipe Calligramme (Nancy) que des linguistes ont fait connaissance avec ce formalisme.

[7] Dirk Roorda. *Resource logic: proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam, 1991.

[8] Glyn V. Morrill. Memoisation of categorial proof nets: parallelism in categorial processing. In Abrusci and Casadio [AC96].

[9] Philippe de Groote. A dynamic programming approach to categorial deduction. In *Conference on Automated Deduction, CADE'99*, Lecture Notes in Artificial Intelligence. Springer-Verlag, July 1999.

[10] Mark E. Johnson. Proof nets and the complexity of processing center-embedded constructions. In Retoré [Ret98], pages 433–447.

de la complexité des relatives imbriquées au centre :

(1.1) Le chat a tué un rat.

(1.2) Le rat que le chat a tué avait mangé du blé.

(1.3) Le blé que le rat que le chat a tué avait mangé était empoisonné.

puis à Glyn Morrill [1] d'étudier d'autres questions de psycholinguistique plus spécifiques à l'anglais. Pour ma part, le fait d'obtenir de bons résultats de ce type, dont les diverses techniques d'analyses syntaxiques conçues jusqu'ici ne peuvent rendre compte, me convainc de la pertinence linguistique des réseaux de démonstration.

Limites et extensions des grammaires de Lambek

Comme on l'a dit plus haut, les grammaires de Lambek ne décrivent que des langages algébriques : ce n'est donc pas tout à fait suffisant, car les linguistes s'accordent à dire que les langages humains ne sont pas algébriques mais restent parmi les langages faiblement contextuels [2]. Il convient néanmoins de nuancer ce résultat : la richesse des langages décrits n'est pas la seule notion pertinente, la richesse d'analyse importe aussi. Si la notion de langage d'arbres d'analyse est claire pour les grammaires algébriques, elle l'est moins pour les grammaires de Lambek. Dans une première version, due à Wojciech Buszkowski [3,4], tous les arbres d'analyse sont permis : ce n'est pas exaltant linguistiquement, et cela s'apparente à un contresens sur la notion même d'arbre d'analyse. Fort heureusement, on peut trouver une notion plus adéquate d'arbre d'analyse dans les grammaires de Lambek. Dans une version plus fine, récemment développée par Hans-Jörg Tiede [5], ce dernier étudie la structure des arbres de déduction naturelle du calcul de Lambek — on notera que ce sont bel et bien des arbres, puisque l'hypothèse déchargée est clairement identifiable d'après le nom de la règle : s'il s'agit de « $/$ », l'hypothèse déchargée est la plus à droite qui soit libre à cet instant de la démonstration, et s'il s'agit de « $\backslash$ », c'est la plus à gauche. Ces arbres donnent bien une structure en constituants, ce qui est clairement plus intéressant puisque tous

[1] Glyn Morrill. Incremental processing and acceptability. *Computational Linguistics*, 26(3):319–338, 2000. preliminary version: UPC Report de Recerca LSI-98-46-R, 1998.

[2] Aravind Joshi, K. Vijay-Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, S. Schieber, and T. Wasow, editors, *Fundational issues in natural language processing*. MIT Press, 1991.

[3] Wojciech Buszkowski. Generative power of categorial grammars. In R. Oehrle, E. Bach, and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*. Reidel, Dordrecht, 1988.

[4] Wojciech Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [vBtM97], chapter 12, pages 683–736.

[5] Hans-Jörg Tiede. Lambek calculus proofs and tree automata. In Moortgat [Moo01a].

les parenthésages de l'énoncé analysé ne sont pas permis. En utilisant cette définition de l'arbre d'analyse, les langages d'arbres engendrés par les grammaires de Lambek peuvent être des langages d'arbres algébriques non réguliers : la capacité générative forte des grammaires de Lambek est donc au-delà de celles des grammaires algébriques, pourtant faiblement équivalentes[i].

Malgré cette richesse d'analyse, les grammaires de Lambek restent trop rigides pour décrire nombre de constructions syntaxiques. Le lien entre fonction et argument ne suffit pas à décrire correctement les dépendances structurelles entre constituants, l'ordre linéaire est trop lié à la structure fonctionnelle, et souvent un ordre linéaire s'avère trop rigide. Cela semble abstrait mais a de fâcheuses conséquences bien tangibles : comment décrire des constituants discontinus (la négation en français), ou les extractions médianes

(1.4) La voiture que$_i$ [j'ai croisée t_i à l'instant] était rouge.[j]

Pour cela, diverses extensions du calcul de Lambek ont été conçues. Le système le plus achevé est certainement celui développé par Michael Moortgat [6] et poursuivi par Glyn Morrill [7] : les grammaires catégorielles multimodales. Celles-ci gardent l'idée de base selon laquelle le lexique associe des formules aux mots. C'est la logique qui est modifiée. Le système de base est le calcul de Lambek non associatif [8], Plusieurs copies de ce calcul peuvent cohabiter, en distinguant leurs connecteurs respectifs par des indices, puisqu'ils satisfont les mêmes règles. Des modalités introduites par paires, qui elles aussi satisfont les mêmes règles, permettent d'assouplir le calcul en créant ou ouvrant des îlots. Ce sont alors des postulats, ou axiomes extra-logiques, qui régissent et distinguent le comportement des connecteurs et des modalités. En particulier les modalités permettent d'autoriser ou non les propriétés structurelles de la composition de syntagmes, telles la commutation et le reparenthésage des constituants [9].

i. Deux grammaires sont dites faiblement équivalente lorsqu'elles produisent les mêmes chaînes de terminaux, et fortement équivalentes lorsqu'elles produisent les mêmes structures d'analyse, celles-ci étant généralement des arbres. Deux classes de grammaires sont dites faiblement (respectivement fortement) équivalentes lorsque toute grammaire d'une classe est faiblement (respectivement fortement) équivalente à une grammaire de l'autre classe et réciproquement.

j. Nous utilisons ici les notations de la grammaire générative. Le symbole t_i désigne l'emplacement naturel du complément d'objet, ici « *que* » qui s'est déplacé. L'indice i coindexant la trace et le constituant sert à indiquer ce déplacement.

[6] Michael Moortgat. *Categorial Investigations*. Foris, Dordrecht, 1988.

[7] Glyn V. Morrill. *Type Logical Grammar*. Kluwer Academic Publishers, Dordrecht and Hingham, 1994.

[8] Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, 1961.

[9] Natasha Kurtonina and Michael Moortgat. Structural control. In P. Blackburn and

Nous avons plutôt opté pour un lexique associant aux mots non une simple formule, mais une démonstration partielle, un « morceau » de réseau de démonstration. Cela permettait aussi d'utiliser le calcul ordonné introduit dans ma thèse, avec lequel on peut exprimer qu'un constituant doit en précéder un autre, mais sans nécessairement que ces constituants soient contigus. Le premier aspect n'était pas si original que nous l'aurions cru : à peu près au même moment, Aravind Joshi et Seth Kulick dans [1,2] ont introduit les arbres de déduction partiels (du calcul de Lambek) pour faire bénéficier les TAGs d'une interface avec la sémantique de Montague. Plutôt que d'étendre la logique, nous avons cherché à tirer profit de la structure des démonstrations, et les conditions, par exemple celles de la grammaire générative, s'expriment comme des conditions sur les démonstrations à construire.

1.2.4 Faut-il prendre en compte des considérations psycholinguistiques ?

Pour reprendre une discussion que j'ai souvent eue avec des collègues, la modélisation linguistique doit-elle s'inspirer de la manière dont les êtres humains utilisent le langage, ou du moins de ce que l'on en sait ? Cela dépend un peu du sens que l'on donne au mot « application » dans l'expression « applications linguistiques ». Si l'on cherche à réaliser des outils de traitement automatique des langues, il n'est pas sûr que ce soit une bonne approche. Pour prendre un exemple historique, ce n'est pas en imitant les oiseaux que l'on a construit des avions ! Notre approche fait le choix de s'inspirer du comportement humain, et seules nos avancées pourrons dire si cela était fondé.

En revanche, s'il s'agit de vérifier la pertinence des théories linguistiques, alors le modèle formel se doit de suivre la manière dont procèdent les êtres humains, puisque c'est cela qui est étudié. La formalisation permet notamment de donner un sens et de vérifier des affirmations selon lesquelles telle option est préférable *parce que* sa complexité algorithmique est moindre. Puisque du point de vue des outils pour le traitement des langues il n'est pas clair que l'imitation du fonctionnement humain doive être mise de coté, et que du point de vue linguistique seul compte la modélisation de la faculté de langage, nous chercherons à être en accord avec ce qui est connu de notre utilisation quotidienne du langage. Autant dire clairement notre position : à mon avis la théorie linguistique la plus

---

M. de Rijke, editors, *Specifying Syntactic Structures*, pages 75–113. CSLI, 1997. Distributed by Cambridge University Press.

[1] Aravind Joshi and Seth Kulick. Partial proof trees as building blocks for a categorial grammar. In Morrill and Oehrle [MO95], pages 138–149.

[2] Aravind Joshi and Seth Kulick. Partial proof trees, resource sensitive logics and syntactic constraints. In Retoré [Ret97a], pages 21–42.

élaborée est celle développée autour de Noam Chomsky au MIT depuis quarante ans, et c'est aussi la seule à afficher une ambition explicative [3,4,5]. D'abord assez ardu, elle ne peut ensuite que séduire.

D'emblée, les similarités entre programme minimaliste et grammaires catégorielles sont remarquables. On ne suppose pas que le mécanisme de calcul varie d'une langue à l'autre. Même lors d'une modélisation assez fine à l'aide des grammaires multimodales, ce n'est pas la logique (le mécanisme de calcul) qui varie d'une langue à l'autre, mais les postulats, la logique devenant une sorte de grammaire universelle. Ce postulat commun est pour nous essentiel, et c'est celui qui explique le paradoxe de l'apprentissage, sur lequel repose toute la théorie chomskyenne. Celui-ci s'énonce simplement : comment se fait-il que sur la base d'exemples positifs exclusivement l'enfant acquiert aussi rapidement une grammaire si complexe que nul livre ne la décrit [6,7] ?

Par exemple tout le monde s'accorde sur la possible coréférence du nom propre *Chomsky* et du pronom anaphorique *il* dans l'énoncé 1.5, et sur son impossibilité dans 1.6.

(1.5) Combien de livres que Chomsky a écrit a-t-il aimé ?

(1.6) Il a aimé trois livres que Chomsky a écrit.

Cette observation pourtant absente des grammaires écrites et enseignées, est néanmoins acquise par tout enfant – et toutes les langues comportent de tels phénomènes.

Chomsky résout ce paradoxe par l'hypothèse d'une grammaire universelle donnée biologiquement, qui se spécialise en la grammaire de la langue apprise par une suite de choix booléens [8,9]. Néanmoins, le formalisme choisi jusqu'à présent par les grammairiens génératives était assez incompatible avec la vision catégorielle : règles syntagmatiques, transformations (déplacements de constituants), utilisation d'éléments vides. Si quelques différences persistent, le récent programme minimaliste nous a maintenant permis de formaliser la parenté entre grammaires

---

[3] Noam Chomsky. Beyond explanatory adequacy. Reading room copy, 2001.

[4] Jean-Yves Pollock. *Langage et cognition: le programme minimaliste de la grammaire générative*. Presses Universitaires de France, Paris, 1997.

[5] Ray Jackendoff. *The Architecture of the Language Faculty*. Number 28 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1995.

[6] Steven Pinker. Language acquisition. In Gleitman and Liberman [GL95], chapter 6, pages 135–182.

[7] L.R. Gleitman and E.L. Newport. The invention of language by children: Environmental and biological influences on the acquisition of language. In Gleitman and Liberman [GL95], chapter 1, pages 1–24.

[8] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

[9] Juan Uriegereka. *Rhyme and reason: an introduction to minimalist syntax*. MIT Press, 1998.

génératives et grammaires catégorielles, et donc d'aborder des questions linguistiques sérieuses qui, jusque là, nous semblaient hors d'atteinte.

1.2.5 Grammaire, logique et informatique

Heureusement et malheureusement la communauté linguistique sur laquelle nos travaux s'appuient est le lieu de multiples débats. L'un des plus anciens et fameux est le lien entre logique et linguistique. Tantôt perçues comme indissociables voire identiques, notamment dans l'antiquité [1], ou dans la grammaire de Port-Royal [2], ces deux disciplines sont aussi souvent opposées, par exemple dans [3], où Noam Chomsky s'oppose au traitement logique des variables dans la structure prédicative.

Donnons un exemple des premières observations liant logique et grammaire qui perdurent. Le caractère verbal (V) et le caractère nominal (N) des parties du discours, qu'on le retrouve dans [4] en 1970 et encore dans [5] en 1997, sont bien sûr à rapprocher de la distinction entre prédicats et individus en logique. Si l'on note $x : [+X, -Z]$ le fait que la catégorie x présente le caractère X mais pas le caractère Z on obtient alors la classification suivante :

(1.7)
$$\begin{array}{rcl} \text{verbe} & : & [+V, -N] \\ \text{nom} & : & [-V, +N] \\ \text{adjectif} & : & [+V, +N] \\ \text{préposition} & : & [-V, -N] \end{array}$$

On observera qu'effectivement les parties du discours qui partagent un même caractère ont des propriétés communes. Par exemple en latin, seuls les adjectifs et les noms, qui ont le caractère nominal se déclinent ; en anglais seuls les catégories portant le caractère verbal, verbes et adjectifs, admettent le préfixe *un* — la même remarque vaut pour le préfixe *dé* en français. Bien sûr, il faut encore d'autres caractères pour distinguer, par exemple, le fait d'être un auxiliaire, mais ce genre de connexion a été observé dès l'antiquité ce qui valut aux adjectifs d'être tantôt classés avec les verbes, tantôt avec les noms [1].

---

[1] Marc Baratin and Françoise Desbordes. *L'analyse linguistique dans l'antiquité classique – I Les théories*. Klincksieck, 1981.

[2] Antoine Arnauld and Claude Lancelot. *Grammaire générale et raisonnée*. Le Petit, 1660. Appelée *Grammaire de Port-Royal*. Rééditée en 1997 par les Éditions Allia.

[3] Noam Chomsky. *Langue, Linguistique, Politique – dialogues avec Mitsou Ronat*. Flammarion, 1991.

[4] Noam Chomsky. Remarks on nominalization. In R. A. Jacob and P. S. Rosenbaum, editors, *Readings in English transformational grammar*, pages 184–221, Waltham, Mass., 1970. Ginn.

[5] Andrew Radford. *Syntactic theory and the structure of English — a minimalist approach*. Cambridge University Press, 1997.

La logique a surtout été utilisée pour modéliser le sens des énoncés, souvent présenté comme une formule du calcul des prédicats. Son utilisation en syntaxe est plus originale : il fallait pour cela avoir des logiques sensibles aux ressources, aux changements d'état, c'est-à-dire des logiques pour l'informatique. Ce siècle a vu d'abord la logique se mathématiser, puis s'informatiser en ce sens que la notion de calcul est devenu centrale, que cette notion soit interne à la logique (par exemple le λ-calcul) ou que la logique serve à décrire le calcul (par exemple la logique de Hoare). Lorsqu'on veut décrire les processus calculatoires de la langue, les algorithmes d'apprentissage, de génération et d'analyse syntaxique, il est donc naturel d'utiliser des outils logiques.

En sémantique la tradition logique aboutit à la sémantique de Montague [6,7, 8] qui implante l'idée selon laquelle le sens d'un composé est déterminé par le sens de ses composants — ce qui est généralement vrai, mais parfois inexact, notamment à cause des pronoms, voir [9]. La DRT [10] étend ce genre d'approche au discours et on peut préférer à un modèle du monde un dodèle des points de vue sur le monde des agents impliqués [11]. La sémantique des situations [12] et celle des jeux [13] sont aussi venues enrichir les perspectives logiques en sémantique formelle.

Dans le domaine de la syntaxe, on ne peut parler de tradition logique, si ce n'est à un niveau très général : toutes les grammaires formelles donnent une description finie de l'infinité d'énoncés corrects, par des règles de production et un lexique. Ce genre de travaux initiés par Zelig Harris [14], et, surtout développés par Noam Chomsky [15] dans sa fameuse hiérarchie, ont connu un grand succès

---

[6] Richmond Thomason, editor. *The collected papers of Richard Montague*. Yale University Press, 1974.

[7] L. T. F. Gamut. *Logic, Language and Meaning – Volume 2: Intensional logic and logical grammar*. The University of Chicago Press, 1991.

[8] Barbara H. Partee. Montague grammar. In van Benthem and ter Meulen [vBtM97], chapter 1, pages 5–91.

[9] Theo M. V. Janssen. Compositionnality. In van Benthem and ter Meulen [vBtM97], chapter 7, pages 417–473.

[10] Jan van Eijck and Hans Kamp. Representing discourse in context. In van Benthem and ter Meulen [vBtM97], chapter 3, pages 179–237.

[11] Claire Beyssade. *Sens et savoirs — des communautés épistémiques dans le discours*. Langue et discours. Presses Universitaires de Rennes, 1998.

[12] Jerry Sligman and Lawrence Moss. Situation theory. In van Benthem and ter Meulen [vBtM97], chapter 4, pages 239–309.

[13] Jaakko Hintika and Gabriel Sandu. Game-theoretical semantics. In van Benthem and ter Meulen [vBtM97], chapter 6, pages 361–410.

[14] Zellig Harris. *Methods in structural linguistics*. University of Chicago Press, Chicago, 1951.

[15] Noam Chomsky. Three models for the description of language. *IRE Transactions on Infor-*

en informatique. Ils portaient initialement l'étiquette « logique » parce qu'issus des systèmes de Post, mais ils sont bien vite devenus un des classiques de l'informatique et de son enseignement, car ils servent à la description de la syntaxe des langages de programmation, des classes inductives d'objets etc.

Les logiques utilisées pour la syntaxe des langues sont trop nouvelles pour constituer une tradition. La volonté qu'ont les logiques sensibles aux ressources de décrire le calcul donne aux grammaires logiques deux aspects informatiques. L'un est une application à la linguistique : quel fragment du mécanisme universel de calcul est nécessaire aux divers aspects du traitement des langues ? Sachant qu'il y a des processus particuliers au langage, l'une des tâches de la linguistique chomskyenne est de déterminer cette spécificité. L'autre est une application au traitement des langues. Sur ce genre de modèles, quels outils de traitement des langues sommes-nous capables de développer ? Est-ce plus ou moins efficace que ceux conçus sur d'autres approches ?

Autant l'avouer, je n'ai pas consacré beaucoup de temps au développement de tels outils, et faute d'excuses voici quelques raisons. Les modèles proposés laissent des choix quant à leur implantation, il existe déjà des outils qui réalisent certaines parties : démonstration automatique en logique linéaire, normalisation de preuves. Surtout, Sylvain Pogodalla (Xerox RCE, Grenoble) dont je co-encadre la thèse implémente certains aspects en Objective CaML : les réseaux de démonstration bicolores, leur assemblage, leur normalisation. Les considérations algorithmiques qui sont en soi une part du travail d'implantation sont très présentes dans la conception de ces modèles et dans la description de leur possible utilisation. Enfin ce type de réalisations nécessite la saisie d'un lexique, contenant pour chaque mot les informations associées, ce qui est fastidieux d'autant que je ne connais pas de base de données qui soit à la fois librement utilisable et adaptée aux modèles catégoriels.

1.2.6 Évaluation des modèles grammaticaux

Voilà une délicate question ! Autant il est relativement aisé de dire si un résultat mathématique est juste ou non, autant il est difficile de dire si un modèle grammatical décrit bien la grammaire d'une langue. Les critères d'évaluation retenus incluent les suivants :

1. critères linguistiques
   (a) absence de surgénération
   (b) couverture empirique
2. critères algorithmiques
   (a) simplicité formelle

mation Theory, IT2:113–124, 1956.

(b) économie algorithmique

Critères linguistiques

Surgénération Les ambiguïtés fallacieuses ne posent pas de problèmes en soi, il existe des moyens ’éviter qu’une grammaire logique ne produise plusieurs analyses non justifiées linguistiquement. L’utilisation de calculs non associatifs et de démonstrations partielles sont des moyens bien connus d’éviter de telles ambiguïtés.

En revanche, vérifier la correction de chaque phrase que la grammaire peut engendrer est impossible. Cette impossibilité est notamment due au nombre infini de phrases possibles ; même si l’on suppose à tort que les mots sont en nombre fini, et que pour des raisons de performance la taille des phrases est bornée[k], les phrases restent bien trop nombreuses pour être vérifiées une à une. Le problème est en fait bien pire que ne le laissent supposer ces hypothèses irréalistes : d’une part ce n’est pas la *performance* mais la *compétence* que l’on souhaite modéliser, et d’autre part le langage est une activité créatrice, où le lexique et même les constructions syntaxiques s’enrichissent constamment.

Couverture empirique La couverture empirique semble *a priori* plus accessible. Cependant les modèles que nous développons ne sont pas suffisamment robustes pour affronter des corpus réels. L’une des raisons est la présence récurrente de néologismes, et de fautes de syntaxe. Une autre raison est l’impossibilité de construire un corpus représentatif d’une langue donnée. Conséquemment, lorsqu’une structure n’apparaît pas dans un corpus, on ne peut pour autant en déduire que cette structure n’appartient pas à la langue. C’est l’une des raisons pour lesquelles la vérification sur corpus d’un modèle syntaxique n’est pas adaptée à des fins d’évaluation. L’utilité des études syntaxiques sur corpus est néanmoins incontestable, puisqu’elle apporte des données complémentaires sur les phénomènes étudiés.

Enfin, il ne suffit pas de s’assurer que notre grammaire rend compte *indépendamment* de plusieurs phénomènes syntaxiques, il faut aussi s’assurer qu’elle en rend compte *simultanément*: une même phrase peut contenir des occurrences de phénomènes différents.

Ce constat négatif suggère, pour le type de grammaires assez nouvelles que nous étudions, une approche beaucoup plus conservatrice. Il faut avoir présents à l’esprit tous les exemples types que les linguistes ont répertoriés, et en rendre

k. L’analogue informatique de cette supposition erronée consisterait à étudier la calculabilité sur les automates et non sur les machines de Turing sous prétexte que les ordinateurs ont une mémoire bornée.

compte, en sachant que ce n'est pas parce que l'on rend compte indépendamment de diverses constructions syntaxiques, que l'on sait en rendre compte simultanément, d'autant que ces constructions peuvent effectivement cohabiter.

Il est bien connu que les pronoms clitiques, constamment utilisés, restent un phénomène d'une grande complexité, non encore totalement résolu[l] :

(1.8) Je répare ma voiture.
(1.9) Je la répare.
(1.10) Je la fais réparer.
(1.11) Je sais la réparer.
(1.12) Je te le dis.
(1.13) Je le lui dit.
(1.14) Elle dit à Marie de partir.
(1.15) Elle le lui dit.
(1.16) Je laisse Anne le lui dire.
(1.17) Je la laisse le lui dire.

Supposons qu'on sache, par ailleurs, rendre compte de la négation — constituant discontinu en français, ce qui est plus facile. Ce succès ne laisse en rien présager de la faculté de la grammaire à traiter simultanément de ces deux phénomènes. Il est possible que la grammaire produise les exemples suivants :

(1.18) Je ne répare pas ma voiture.
(1.19) * Je la ne répare pas.
(1.20) * Je ne la fais réparer pas.
(1.21) * Je sais la ne réparer pas.
(1.22) * Je te ne le dis pas.
(1.23) * Je le ne lui dis pas.
(1.24) * Elle dit à Marie de ne partir pas.
(1.25) * Elle le lui ne dit pas.

l. Mentionnons cependant les travaux tout récents de Daniela Bargelli, Claudia Casadio et Joachim Lambek sur ce sujet [1,2] dans un formalisme voisin des grammaires catégorielles, les *prégroupes*, introduits il y a peu par Joachim Lambek [3,4].

[1] Daniela Bargelli and Joachim Lambek. An algebraic approach to french sentence structure. In de Groote et al. [dGMR01], pages 62–78.

[2] Claudia Casadio and Joachim Lambek. An algebraic analysis of clitic pronouns in Italian. In de Groote et al. [dGMR01], pages 110–124.

[3] Joachim Lambek. Type grammar revisited. In Lecomte et al. [LLP99], pages 1–27.

[4] Joachim Lambek. Type grammars as pregroups. *Grammars*, 4(1):21–39, 2001.

(1.26) * Je laisse Anne le lui ne dire pas.

(1.27) * Je ne la laisse le lui dire pas.

La solution que nous avons choisie pour remédier aux difficultés d'évaluation est de s'inspirer de formalismes existants, par exemple des TAGs, et, surtout, de s'appuyer sur des théories linguistiques, en particulier sur la grammaire générative. C'est l'une des raisons de renforcer le rapprochement entre grammaires minimalistes et grammaires catégorielles.

Nous remplaçons les critères donnés au départ par les suivants, plus adaptés à nos objectifs :

1. critère linguistique: être justifié par une théorie linguistique
2. critères algorithmiques
   (a) simplicité formelle
   (b) économie algorithmique

Critères algorithmiques

Le critère d'économie algorithmique est bien défini, ce qui ne signifie pas que nous sachions toujours le vérifier : la complexité des algorithmes d'analyse, de génération ou d'apprentissage, n'est pas toujours aisément calculable.

Par contre, le critère de simplicité formelle est plus délicat à évaluer. Par exemple, quelle est la bonne définition de la taille d'une grammaire lexicalisée, notion essentielle aux algorithmes d'apprentissage ? Est-ce la taille maximale ou moyenne d'une entrée lexicale, le nombre de types atomiques, etc. ?

1.3 En guise de conclusion

En mathématiques, ce qui est établi perdure, même si l'intérêt peut passer. En revanche, les modèles linguistiques, tout comme les théories linguistiques dont ils sont issus, sont en mouvement perpétuel. C'est en les utilisant et en les affinant que se révèlent leurs défauts et leurs limites, ce qui conduit généralement à de meilleurs modèles.

Au fil de nos travaux se dégagent néanmoins des constantes telle la compositionalité de la sémantique prédicative, ou la présence de parallélisme dans la vérification des traits syntaxiques, la correspondance entre structure de l'analyse et forme logique, la calculabilité de l'ordre des mots à partir de cette structure logique.

Chapitre 2

État de l'art et contributions

Après avoir esquissé dans le premier chapitre nos thèmes de recherche, nous donnons dans ce chapitre-ci un rapide aperçu des travaux qui ont permis les nôtres et situons nos contributions par rapport aux travaux existants et aux perspectives du domaine.

2.1 Graphes et logique linéaire

Jusqu'à présent nous n'avons pas expliqué d'où provient le mot graphe souvent utilisé dans de ce mémoire. La syntaxe naturelle de la logique linéaire est celle des réseaux de démonstration, comme [1] l'établit, et ceux-ci sont des graphes. Dans nos utilisations grammaticales de la logique linéaire, les structures syntaxiques ainsi que les représentations sémantiques seront des réseaux de démonstration et donc des graphes.

Une relative originalité de nos travaux est de placer l'étude des réseaux de démonstration dans la théorie des graphes standard.

2.1.1 Les réseaux de démonstration : une syntaxe graphique pour la logique linéaire

Les démonstrations de la logique linéaire peuvent se voir comme des graphes, en particulier pour la plupart des fragments que nous avons considérés. Cette particularité a été intensivement étudiée dès l'article fondateur de Jean-Yves Girard

[1] Jean-Yves Girard. Proof-nets: The parallel syntax for proof-theory. In A. Ursini and P. Agliano, editors, *Logic and Algebra*, number 150 in Lecture Notes in Pure and Applied Mathematics, pages 97–124. Marcel Dekker, Inc, New York, 1996.

[1]. Dans l'approche qui est la nôtre, la généralisation qu'il propose dans [2] ainsi que la simplification apportée par Vincent Danos et Laurent Regnier [3] constituent deux étapes importantes à souligner. Depuis, de nombreux travaux sur cette question ont été réalisés, et nous ne citerons que les plus voisins [4,5,6,7,8,9]. Expliquons brièvement comment sont faits ces graphes dans le cas de la logique linéaire multiplicative, à laquelle l'essentiel de nos travaux est consacré. Un tel graphe est fait de deux parties :

- L'arbre des sous formules de la formule démontrée — suivant certains auteurs les sous formules sont des sommets et les connecteurs des arêtes, et suivant d'autres c'est l'inverse. [a].
- Un ensemble d'arêtes qui relient des feuilles de sorte que :
  - Les deux extrémités sont la négation l'une de l'autre.
  - Deux telles arêtes ne sont jamais adjacentes.
  - Toutes les feuilles sont incidentes à une telle arête.

Tous les graphes ainsi constitués ne sont pas des démonstrations, sinon pour peu qu'une formule ait autant de a que de $\neg a$, noté $a^{\perp}$, elle serait démontrable en logique linéaire, ce qui n'est certainement pas le cas de $a \wedge (\neg a)$, noté $a \otimes a^{\perp}$. Il y a donc un critère qui distingue les graphes qui représentent des démonstrations, appelés réseaux de démonstration, parmi les préréseaux, c'est-à-dire parmi les graphes qui ont la structure d'une démonstration sans forcément en être une.

a. Ce singulier est une simplification : il peut y en avoir plusieurs, dès que la formule est démontrée sous certaines hypothèses, celles-ci donnant lieu à autant d'arbres des sous formules, de la négation de ces hypothèses. De même dans un système classique où plusieurs conclusions sont possibles, chacune d'entre elles correspond à un arbre des sous formules.

[1] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[2] Jean-Yves Girard. Multiplicatives. *Rendiconti del Seminario dell'Universitá é Politecnico Torino*, October 1986. Special issue on Logic and Computer Science.

[3] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[4] Dirk Roorda. *Resource logic: proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam, 1991.

[5] Arnaud Fleury and Christian Retoré. The mix rule. *Mathematical Structures in Computer Science*, 4(2):273–285, June 1994.

[6] François Lamarche. Proof nets for intuitionistic linear logic: Essential nets. 35 page technical report available by FTP from the Imperial College archives, April 1994.

[7] Philippe de Groote. An algebraic criterion for intuitionnistic multiplicative proof nets. *Theoretical Computer Science*, 1999.

[8] Vito Michele Abrusci and Elena Maringelli. A new correctness criterion for cyclic multiplicative proof-nets. In Retoré [Ret98].

[9] Vito Michele Abrusci and Paul Ruet. Non-commutative logic I: the multiplicative fragment. *Annals of Pure and Applied Logic*, 1999. http://www.uniroma3.it/reseaux9798.ps.

Ce critère, dit de correction, comporte toujours une condition exprimant l'absence de certains cycles ; suivant le calcul considéré, il peut comporter une autre condition qui exprime une forme de connexité. Pour pouvoir substituer la syntaxe des réseaux à celle du calcul des séquents, il faut alors montrer que les démonstrations du calcul des séquents se transforment en des réseaux, et que tout réseau correspond à une démonstration du calcul des séquents ; cette seconde propriété, nommée *séquentialisation*, donne aux réseaux une structure inductive, tandis que le critère de correction les définit globalement.

Quels sont les avantages d'une telle description des démonstrations ? Comparée au traditionnel calcul des séquents, ou dans une moindre mesure à la déduction naturelle de [10,11] (voir [12] pour une excellente présentation plus récente, ou les ouvrages [13,14] pour une vision légèrement différente) la traduction des démonstrations du calcul des séquents en réseaux identifient de nombreuses démonstrations. Deux démonstrations associées à un même réseau ne diffèrent l'une de l'autre que par des permutations de règles, et elles ont le même comportement lorsqu'elles sont vues comme des programmes fonctionnels typés. Les réseaux nous rapprochent donc de l'essence même des démonstrations, et donnent à ces dernières une structure mathématique plus agréable. Effectivement, des résultats significatifs ont été obtenus en utilisant les réseaux de démonstration :

- La normalisation des réseaux de démonstration se définit comme une réduction locale pouvant être optimisée [15,16,17] ; on obtient ainsi un évaluateur du λ-calcul typé qui partage les calculs de manière optimale [18].
- Les réseaux induisent des techniques de démonstration automatique très ef-

---

[10] Gehrard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39:176–210, 1934. Traduction Française de R. Feys et J. Ladrière: Recherches sur la déduction logique, Presses Universitaires de France, Paris, 1955.

[11] Gehrard Gentzen. Untersuchungen über das logische Schließen II. *Mathematische Zeitschrift*, 39:405–431, 1934. Traduction française de J. Ladrière et R. Feys: Recherches sur la déduction logique, Presses Universitaires de France, Paris, 1955.

[12] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[13] Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

[14] Jean H. Gallier. *Logic for computer science*. Harper & Row Publishers, New York, 1986.

[15] Jean-Yves Girard. Geometry of interaction, I: interpretation of system F. In *Proceedings of the ASL meeting held in Padova*, August 1989.

[16] Vincent Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation et principalement du λ-calcul*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, juin 1990.

[17] Laurent Regnier. *Lambda calcul et Réseaux*. Thèse de doctorat, spécialité mathématiques, Université Paris 7, Janvier 1992.

[18] Georges Gonthier, Jean-Jacques Lévy, and Martin Abadi. Linear logic without boxes. In *Logic in Computer Science*, 1992.

ficaces [1,2,3].

– Lorsque les démonstrations sont vues comme des analyses syntaxiques d'énoncés, les réseaux donnent des mesures de la complexité instantanée des énoncés, qui s'accordent avec les données psycholinguistiques. Ceci a été montré par Mark Johnson et Glyn Morrill [4,5], notamment dans le cas de relatives imbriquées au centre :

  (2.1) Le loup a dévoré la chèvre.

  (2.2) La chèvre que le loup a dévoré avait mangé le chou.

  (2.3) ? Le chou que la chèvre que le loup a dévoré avait mangé appartenait au passeur.

  (2.4) ?? Le passeur auquel le chou que la chèvre que le loup a dévoré avait mangé appartenait possède plusieurs bateaux.

  (2.5) ??? Les bateaux que le passeur auquel le chou que la chèvre que le loup a dévoré avait mangé appartenait possède sont des barges.

  À un instant donné, c'est-à-dire entre deux mots de l'énoncé considéré, la complexité de la phrase, qui fait qu'on ne peut plus saisir un tel énoncé, s'exprime en fonction des axiomes du réseau de démonstration ; en effet celui-ci donne immédiatement accès aux constituants attendus à un instant donné, et permet de prendre en compte les paramètres de cette complexité : le nombre de constituants attendus, le nombre de constituants de même nature attendus, et leur degré d'imbrication dans la catégorie qui les attend.

2.1.2 Couplages parfaits, cographes et réseaux de démonstration

Si les réseaux sont des graphes, cela ne signifie pas pour autant qu'ils soient des graphes auxquels s'appliquent la théorie des graphes usuelle. En effet les graphes utilisés jusqu'ici manquent à mon goût de netteté mathématique. Si ce

---

[1] Philippe de Groote. Linear logic with Isabelle: pruning the proof search tree. In *4^th Workshop on theorem proving with analytic tableaux and related methods*, Lecture Notes in Artificial Intelligence. Springer-Verlag, March 1995.

[2] Glyn V. Morrill. Memoisation of categorial proof nets: parallelism in categorial processing. In Abrusci and Casadio [AC96].

[3] Philippe de Groote. A dynamic programming approach to categorial deduction. In *Conference on Automated Deduction, CADE'99*, Lecture Notes in Artificial Intelligence. Springer-Verlag, July 1999.

[4] Mark E. Johnson. Proof nets and the complexity of processing center-embedded constructions. In Retoré [Ret98], pages 433–447.

[5] Glyn Morrill. Incremental processing and acceptability. *Computational Linguistics*, 26(3):319–338, 2000. preliminary version: UPC Report de Recerca LSI-98-46-R, 1998.

sont des graphes, aucune propriété ne les distingue : ce ne sont pas vraiment des arbres, il ne sont pas réguliers et surtout le nom des sommets intervient sans cesse dans leur propriétés.

Cependant on a souvent besoin de résultats fins sur leur structure. J'ai donc cherché à en donner une description à l'aide de notions de théorie des graphes qui soient le plus standard possible, pour des raisons mathématiques et esthétiques, mais aussi pour pouvoir établir certaines propriétés logiques ou étendre les logiques considérées. Une première étape a été de simplifier le critère. Au lieu de traduire formules et sous-formules comme des sommets, celles-ci se voient traduites par des arêtes d'une couleur différente, qui définissent un couplage parfait du graphe[b] — un sujet très classique en théorie des graphes [6,7]. Les connecteurs correspondent pour leur part à des arêtes hors du couplage reliant les arêtes du couplage entre elles ; les connecteurs sont distingués par l'arête liant les deux prémisses entre elles. On peut alors définir aisément la correction du réseau sans faire appel à l'étiquette logique de la connexion : un réseau est correct si et seulement s'il ne contient pas de cycle qui alterne les arêtes des deux couleurs. On obtient alors une définition des préréseaux assez satisfaisante en termes de théorie des graphes, et le critère de correction distinguant les réseaux est aussi naturel. Le critère requiert l'absence de cycles élémentaires alternants, ce qui équivaut à l'unicité du couplage parfait dans le graphe sous-jacent [8]. Un résultat sur la structure des graphes munis d'un unique couplage parfait permet d'obtenir la séquentialisation et a pour corollaires les autres démonstrations du même genre de [9] et [10]. Ces premiers résultats se trouvent dans [8].

L'étape suivante a été de structurer les arêtes n'appartenant pas au couplage, c'est-à-dire celle qui décrivent la formule démontrée. Une formule peut se voir comme un cographe, c'est-à-dire un graphe de la plus petite classe contenant tous les graphes à un sommet et sans aucune arête et close par somme disjointe et complémentation. Les cographes admettent à la fois une caractérisation univer-

b. Un ensemble d'arêtes qui sont deux à deux non adjacentes et couvrent tous les sommets du graphe

[6] Claude Berge. *Graphes*. Gauthier-Villars, 1979.

[7] Làszlò Lovàsz and Michael David Plummer. *Matching Theory*, volume 121 of *Mathematics Studies*. North-Holland, 1986. Annals of Discrete Mathematics 29.

[8] Christian Retoré. Perfect matchings and series-parallel graphs: multiplicative proof nets as R&B-graphs. In J.-Y. Girard, M. Okada, and A. Scedrov, editors, *Linear'96*, volume 3 of *Electronic Notes in Theoretical Science*. Elsevier, 1996. (available from http://www.elsevier.nl/).

[9] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[10] Vincent Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation et principalement du λ-calcul*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, juin 1990.

selle par exclusion de certains sous-graphes et une définition inductive — voir par exemple [1] pour un excellent article de synthèse sur les cographes, les ordres séries-parallèles et ce type de structures. Un *réseau abstrait* a pour seuls sommets les occurrences de variables propositionnelles dans la formule démontrée. Les arêtes du couplage parfait décrivent les axiomes qui les relient. Les autres arêtes forment un cographe qui décrit la formule démontrée. La difficulté est alors d'exprimer le critère de correction. Dans [2] il s'agit d'une configuration présente sur tout cycle alternant. J'ai par la suite montré que ce critère équivaut à la présence d'une corde sur tout cycle alternant [3,4]. On obtient alors une description des réseaux tout à fait standard du point de vue de la théorie des graphes, et qui, du point de vue logique, a un grand avantage : les propriétés algébriques des connecteurs sont interprétées par l'égalité, tandis que le traditionnel arbre des sous-formules fait la différence entre, par exemple, $A \otimes (B \otimes C)$ et $(A \otimes B) \otimes C$.

Finalement, en utilisant l'inclusion des cographes que nous avons axiomatisée avec Denis Bechet et Philippe de Groote [5], j'ai obtenu une axiomatique de la logique linéaire comme un système de réécriture [3,4].

Nombre de ces résultats s'adaptent lorsqu'on augmente la logique linéaire par le connecteur « précède » associatif, non commutatif et auto-dual. Il faut pour cela caractériser les cographes orientés et axiomatiser leur inclusion [5], définir la correction et étudier le comportement de la réécriture par rapport à la correction, ce qui est fait dans [6].

---

[1] Rolf H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, volume 255 of *NATO ASI series C*, pages 105–194. Kluwer, 1989.

[2] Christian Retoré. Perfect matchings and series-parallel graphs: multiplicative proof nets as R&B-graphs. In J.-Y. Girard, M. Okada, and A. Scedrov, editors, *Linear'96*, volume 3 of *Electronic Notes in Theoretical Science*. Elsevier, 1996. (available from http://www.elsevier.nl/).

[3] Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 1999. To appear.

[4] Christian Retoré. Handsome proof-nets: R&B-graphs, perfect matchings and series-parallel graphs. Rapport de Recherche RR-3652, INRIA, March 1999. http://www.inria.fr/.

[5] Denis Bechet, Philippe de Groote, and Christian Retoré. A complete axiomatisation of the inclusion of series-parallel partial orders. In H. Comon, editor, *Rewriting Techniques and Applications, RTA'97*, volume 1232 of *LNCS*, pages 230–240. Springer Verlag, 1997.

[6] Christian Retoré. Pomset logic as a calculus of directed cographs. In V. M. Abrusci and C. Casadio, editors, *Dynamic Perspectives in Logic and Linguistics: Proof Theoretical Dimensions of Communication Processes,Proceedings of the 4th Roma Workshop*, pages 221–247. Bulzoni, Roma, 1999. Also available as INRIA Rapport de Recherche RR-3714 http://www.inria.fr/.

2.1.3 Réseaux de démonstration et systèmes logiques

Puisque les réseaux de démonstration sont la syntaxe naturelle de la logique linéaire, les résultats ci-dessus permettent une étude fine des systèmes logiques correspondants. Par exemple les réseaux de démonstration abstraits nous débarrassent des questions d'associativité, et la réécriture des cographes donne une démonstration particulièrement simple de l'élimination des coupures. Cette question mentionnée dans [3,4] est totalement résolue dans [6]. La structure et les propriétés des réseaux de démonstration, décrites au paragraphe précédent, ont permis d'étudier les démonstrations de divers calculs de la logique linéaire, en particulier pour les logiques linéaires non commutatives, celles qu'utilisent les applications linguistiques.

Calcul ordonné

Le calcul ordonné, introduit dans ma thèse [7], se simplifie considérablement en utilisant les réseaux abstraits : on peut alors le voir comme un système de réécriture [6]. Ce calcul, issu de la sémantique cohérente, possède la plupart des propriétés attendues d'une logique bien définie, telles l'élimination des coupures, un calcul des réseaux, une sémantique dénotationnelle. Il s'étend aux modalités et additifs, et le connecteur « précède » possède une modalité auto duale qui lui correspond. Néanmoins, je ne dispose toujours pas de calcul des séquents qui lui corresponde parfaitement : je connais certes des calculs des séquents qui ne donnent que des réseaux corrects, mais aucun jusqu'à ce jour ne rend compte de tous les réseaux. Néanmoins, en étudiant la réécriture des cographes orientés correspondant à ce calcul, j'ai récemment pu reformuler plus simplement cette question.

Dès mes premiers travaux, Andrea Asperti a remarqué que ce connecteur correspond à la composition séquentielle lorsque les démonstrations sont vues comme des processus qu'on exécute en les parcourant [8,9]. Ce calcul a aussi été utilisé par Alessio Guglielmi dans [10] pour modéliser des calculs de processus en

---

[7] Christian Retoré. *Réseaux et Séquents Ordonnés*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, février 1993.

[8] Andrea Asperti. A linguistic approach to dead-lock. Technical Report LIENS 91-15, Dép. Maths et Info, Ecole Normale Supérieure, Paris, October 1991.

[9] Andrea Asperti and Giovanna Dore. Yet another correctness criterion for multiplicative linear logic with mix. In A. Nerode and Yu. Matiyasevich, editors, *Logical Fundations of Computer Science*, volume 813 of *Lecture Notes in Computer Science*, pages 34–46, St. Petersburg, July 1994. Springer Verlag.

[10] Alessio Guglielmi. Concurrency and plan generation in a logic programming language with a sequential operator. In Pascal van Hentenryck, editor, *International Conference on Logic Programming*, pages 240–254, Genova, 1994. M.I.T. Press.

programmation logique en logique linéaire telle que Dale Miller l'a définie [1] ; son dernier travail [2,3], plus audacieux, s'inspire de la description que nous donnons des réseaux avec des cographes orientés, et considère des calculs des séquents généralisés. Mais pour notre part, nous avons préféré explorer les applications linguistiques et revenir aux calculs non commutatifs plus traditionnels.

Calculs non commutatifs

Le calcul de Lambek, introduit par Joachim Lambek en 1958 [4], correspond au fragment multiplicatif de la logique linéaire non commutative de Michele Abrusci [5] ou de David Yetter [6]. Dirk Roorda a certainement été le premier à s'intéresser aux réseaux de démonstration pour le calcul de Lambek [7]. Ceux-ci se sont avérés particulièrement utiles pour la démonstration automatique [8,9] et pour leur pertinence comme structure d'analyse syntaxique dans les grammaires de Lambek évoquées ci-après. A l'occasion d'un article de synthèse [10], j'ai étudié la correspondance entre calcul de Lambek et logique linéaire, et j'ai complété des travaux existants par des résultats que la communauté croyait établis mais qui n'étaient publiés nulle part. Le calcul de Lambek s'obtient par restrictions successives à partir de la logique linéaire multiplicative :

1. L'échange par transposition est supprimé, ce qui équivaut à la non commutativité.

---

[1] Dale Miller. Forum: a multiple conclusion specification language. *Theoretical Computer Science*, 165:201–232, 1996.

[2] Alessio Guglielmi. A calculus of order and interaction. Technical Report WV-99-04, Dresden University of Technology, 1999.

[3] Alessio Guglielmi and Lutz Straßburger. Non-commutativity and MELL in the calculus of structures. In L. Fribourg, editor, *CSL 2001*, volume 2142 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, 2001.

[4] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[5] Vito Michele Abrusci. Phase semantics and sequent calculus for pure non-commutative classical linear logic. *Journal of Symbolic Logic*, 56(4):1403–1451, December 1991.

[6] David N. Yetter. Quantales and (non-commutative) linear logic. *Journal of Symbolic Logic*, 55:41–64, 1990.

[7] Dirk Roorda. *Resource logic: proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam, 1991.

[8] Glyn V. Morrill. Memoisation of categorial proof nets: parallelism in categorial processing. In Abrusci and Casadio [AC96].

[9] Philippe de Groote. A dynamic programming approach to categorial deduction. In *Conference on Automated Deduction, CADE'99*, Lecture Notes in Artificial Intelligence. Springer-Verlag, July 1999.

[10] Christian Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.

2. Le calcul est intuitionniste, ce qui est dans ce cas une simple restriction de langage.
3. Ce calcul n'admet aucune tautologie, ce qui, grammaticalement, signifie que la séquence vide n'appartient à aucune catégorie.

Un point important est que ces trois restrictions commutent entre elles.

La restriction (3) a peu été étudiée d'un point de vue logique, car c'est dans les applications grammaticales qu'elle trouve tout son sens. Cependant, elle n'affecte pas les propriétés du système logique, telles la propriété de la sous-formule et l'élimination des coupures, et elle se traduit par une condition uniforme sur les réseaux.

Si on applique la restriction (1) on obtient le calcul cyclique de David Yetter [6]. Si on applique la restriction (2) on obtient le fragment multiplicatif de la logique linéaire intuitionniste.

En fait, notre principal apport est d'avoir montré que la non commutativité équivalait au fait que les axiomes définissent un bon parenthésage sur les variables propositionnelles. En effet, Dirk Roorda [7] ne montre que la nécessité de cette condition. Qu'elle soit non seulement nécessaire mais aussi suffisante a été établi dans notre article [10]. Dans cet article nous utilisons des réseaux bicolores avec des liens logiques, sans utiliser les réseaux abstraits ou cographes.

Poursuivant notre étude, Michele Abrusci et Elena Maringelli [11,12] ont trouvé dans le cadre des réseaux bicolores un critère élégant qui correspond à la non commutativité, formulé en termes de chemins plutôt qu'en termes de bons parenthésages. Leur travail permet de traiter simplement des réseaux avec coupures, pour lesquels le critère en termes de bons parenthésages s'adapte, mais devient très compliqué, en particulier pour établir l'élimination des coupures.

Mettre au point un critère de correction pour les préréseaux du calcul partiellement commutatif de Philippe de Groote [13] est un de nos objectifs prioritaires. Dans [14], nous avons utilisé ce calcul pour modéliser l'exécution des réseaux de Petri, et dans [15] nous l'avons utilisé pour rendre compte des grammaires mini-

---

[11] Elena Maringelli. *Grafi e logica lineare: una nuova definizione delle reti dimostrative non commutative*. Tesi di laurea, Università di Bari, December 1996. (Graphs and linear logic: a new definition of non-commutative proof-nets.).

[12] Vito Michele Abrusci and Elena Maringelli. A new correctness criterion for cyclic multiplicative proof-nets. In Retoré [Ret98].

[13] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

[14] Christian Retoré. A description of the non-sequential execution of Petri nets in partially commutative linear logic. In Jan van Eijck, Vincent van Oostrom, and Albert Visser, editors, *Logic Colloquium '99 (Utrecht)*, Lecture Notes in Logic. Association for Symbolic Logic & A. K. Peters, Ltd, 2001. Complete version: RR-INRIA 4288 http://www.inria.fr/.

[15] Alain Lecomte and Christian Retoré. Extending Lambek grammars: a logical account of minimalist grammars. In *Proceedings of the 39th Annual Meeting of the Association for*

malistes d'Edward Stabler [1]. Les réseaux de démonstration seront les bienvenus dans un cas comme dans l'autre : ils expriment exactement la consommation des jetons dans les réseaux de Petri ou celle des des traits linguistiques dans les grammaires minimalistes. Nous sommes en train d'étudier cette question.

Calcul mixte de Philippe de Groote et réseaux de Petri

Comme on l'a dit, la logique linéaire a dès ses débuts été utilisée pour modéliser les réseaux de Petri — voir [2,3,4] pour des synthèses sur ces questions. Si l'on souhaite traiter de questions comme la synthèse de réseaux de Petri, ces codages ne sont pas pleinement satisfaisants : les événements sont absents du codage, ce qui empêche leur description logique d'accéder aux notions essentielles que sont le langage d'un réseau ainsi que la description de la différence d'efficacité entre exécutions.

Le calcul partiellement commutatif de Philippe de Groote [5] m'a permis d'y remédier, ou plus précisément une extension de ce calcul. Ce calcul superpose le calcul de Lambek et la logique linéaire intuitionniste (commutative). Pour ce faire, les hypothèses d'un séquent se trouvent munies d'un ordre série-parallèle, qui peut augmenter suivant les règles de réécriture dont nous avons déjà parlé — c'était une autre raison de s'intéresser à cette question. En s'inspirant de la définition des grammaires de Lambek, j'ai pu donner une description de l'exécution dans les réseaux de Petri. Les places sont vues comme des variables propositionnelles, et un marquage est un produit *commutatif* de places. Un événement est décrit par $m \setminus n$, l'implication *non commutative* d'un marquage partiel n par un marquage partiel m. Ce jeu entre connecteurs commutatifs et non commutatifs permet de rendre compte de l'exécution parallèle des réseaux de Petri :

- Soit ϕ un multi-ensemble ordonné dont l'ordre soit série-parallèle et dont les éléments soient les événements e_i,$i \in I$;
- alors la possibilité d'exécuter les événements e_i,$i \in I$ à partir du marquage M, en respectant les contraintes exprimées par ϕ

Computational Linguistics, ACL 2001, pages 354–361, Toulouse, July 2001. ACL.

[1] Edward Stabler. Derivational minimalism. In Retoré [Ret97a], pages 68–95.

[2] Anne Sjerp Troelstra. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. CSLI, 1992. (distributed by Cambridge University Press).

[3] François Girault. *Formalisation en logique linéaire du fonctionnement des réseaux de Petri*. Thèse de doctorat, spécialité informatique industrielle, Université Paul Sabatier, Toulouse, décembre 1997.

[4] Uffe Engberg and Glynn Winskel. Completeness results for linear logic on Petri nets. *Annals of Pure and Applied Logic*, 86(2):101–135, 1997.

[5] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

équivaut à la prouvabilité du séquent $(M;\phi) \vdash N$, où le marquage N se calcule immédiatement à partir de M et ϕ.

On a donc une représentation fidèle de l'exécution des réseaux de Petri lorsque les contraintes entre événements sont décrites par un ordre série-parallèle [6].

2.1.4 Sémantique dénotationnelle

Comme la logique intuitionniste, la logique linéaire admet deux types de sémantiques : d'une part des sémantiques de la prouvabilité qui associent des valeurs de vérités aux formules, et, d'autre part, des sémantiques dénotationnelles qui associent aux démonstrations des objets que préservent l'élimination des coupures. Ayant davantage étudié les démonstrations que les formules, j'ai seulement étudié les sémantiques dénotationnelles de la logique linéaire.

Ce type de modèle se rattache aux sémantiques dénotationnelles de Dana Scott et Christopher Strachey [7] via les modèles dénotationnels de la logique intuitionniste ou sémantiques catégoriques. Chaque formule est interprétée par un objet de la catégorie (au sens mathématique cette fois) et les morphismes de A dans B interprètent les démonstrations de $A \vdash B$ — en particulier les morphismes de l'objet terminal, noté **1**, dans B interprètent les démonstrations de B. L'interprétation ne varie pas lorsqu'on normalise la démonstration ; si l'on voit la démonstration de $A \vdash B$ comme un programme fonctionnel typé de type $A->B$, qui peut lui même contenir des applications de fonctions à des arguments, cela revient à dire que l'interprétation du programme n'est pas modifiée par l'évaluation, d'où l'adjectif « dénotationnel ».

Parmi les modèles dénotationnels, nous nous sommes plus particulièrement intéressés aux espaces cohérents introduits par Jean-Yves Girard [8]. Ce sont des domaines d'information où tous les points sont totalement déterminés par le deuxième niveau de l'ordre ; ceux-ci forment un modèle de la logique intuitionniste, et c'est leur étude qui a conduit à la découverte de la logique linéaire.

On peut les définir comme des graphes ordinaires mais qui, à la différence des graphes considérés précédemment, peuvent être infinis. Les variables propositionnelles sont interprétées par des espaces cohérents quelconques et cela détermine,

---

[6] Christian Retoré. A description of the non-sequential execution of Petri nets in partially commutative linear logic. In Jan van Eijck, Vincent van Oostrom, and Albert Visser, editors, *Logic Colloquium '99 (Utrecht)*, Lecture Notes in Logic. Association for Symbolic Logic & A. K. Peters, Ltd, 2001. Complete version: RR-INRIA 4288 http://www.inria.fr/.

[7] J. E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. The MIT Press, 1977. The MIT Press Series in Computer Science.

[8] Jean-Yves Girard. The system F of variable types: Fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

pour toute formule l'espace cohérent qui lui est associé : chaque connecteur logique correspond à une opération sur les espaces cohérents. Les démonstrations d'une formule A sont interprétées par des cliques[c] de l'espace cohérent A. Cette sémantique qui suit de très près la logique linéaire, ne permet pourtant pas d'obtenir un résultat de complétude (*full completeness*) : toute clique, même maximale ne correspond pas forcément à une démonstration. Il faut enrichir subtilement le modèle pour obtenir de tels résultats, ce qui même pour de petits fragments n'est pas trivial [1]. C'est l'une des raisons du succès de la sémantique des jeux : elle permet d'obtenir de tels modèles [2,3,4].

Le connecteur « précède »

J'ai utilisé une première fois cette sémantique comme un guide : le connecteur « précède », étudié dans ma thèse, est issu de la sémantique cohérente, celle-ci ayant servi à mettre au point un calcul en réseaux incluant ce connecteur. Pour présenter ce connecteur en deux mots, disons qu'il est le seul connecteur covariant en ses deux arguments et non commutatif présent dans les espaces cohérents. Il a la particularité d'être à la fois associatif, non commutatif et auto-dual. La syntaxe des réseaux de démonstration s'étend simplement à ce nouveau connecteur, par contre il est plus difficile d'établir un calcul des séquents qui l'inclue.

Sémantique cohérente et réseaux de démonstration

Depuis, nous avons exploré le lien entre sémantique cohérente et réseaux de démonstration. Comme Jean-Yves Girard l'a montré dans son article original [5], on peut définir la sémantique d'une démonstration de manière non inductive, directement sur le réseau de démonstration. Il montre alors qu'on obtient bien une clique de l'espace cohérent correspondant à la conclusion.

Je me suis rendu compte qu'on peut définir la sémantique cohérente d'un préréseau, c'est-à-dire d'un réseau non nécessairement correct, ou préréseau. J'ai

c. Une clique d'un graphe est un ensemble de sommets deux à deux adjacents.

[1] Ralph Loader. Linear logic, totality and full completeness. In *LICS: IEEE Symposium on Logic in Computer Science*, 1994.

[2] Andreas Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56(1-3):183–220, 1994.

[3] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543 – 574, June 1994.

[4] Jean-Yves Girard. Locus solum. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.

[5] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

alors montré l'équivalence entre la correction du préréseau et le fait que son interprétation soit une clique, c'est-à-dire un objet sémantique valide. Ce résultat[6] vaut aussi pour le calcul ordonné, mais il est plus difficile à établir en l'absence d'une définition inductive des réseaux [7].

Une modalité auto-duale pour le connecteur « précède »

Une autre question posée par Jean-Yves Girard était de trouver une modalité correspondant à ce connecteur, autorisant la contraction positive et négative :

$$\triangleleft A < \triangleleft A \vdash \triangleleft A \qquad \text{et} \qquad \triangleleft A \vdash \triangleleft A < \triangleleft A.$$

Elle existe, est auto-duale, et les morphismes sont même des isomorphismes [8,7]. La syntaxe d'une telle modalité reste à mettre au point, et, du moins pour les réseaux de démonstration, cela semble faisable. De plus, elle a un sens en termes de concurrence : avec le sens temporel de « précède » elle signifie que A est présent à tout instant. Néanmoins je n'ai pas développé ce travail car c'est avec un calcul des séquents pour « $<$ » qu'elle deviendrait utile et convaincante — quoiqu'elle s'intègre sans doute dans les travaux d'Alessio Guglielmi [9,10].

2.2 Modèles grammaticaux issus de la logique linéaire

Le point de vue que nous suivons, avec Alain Lecomte, dans notre recherche[d] de modèles grammaticaux pour la syntaxe des langues est à peu près celui de Noam Chomsky, décrit dans les ouvrages [11,12,13] tandis que les formalismes uti-

d. Le terme « quête » serait peut-être plus pertinent que le terme « recherche », car nous doutons qu'il existe un modèle syntaxique qui satisfasse tous les critères donnés à la fin du chapitre précédent. Du reste, les modèles définis sont en constante évolution.

[6] Christian Retoré. A semantic characterisation of the correctness of a proof net. *Mathematical Structures in Computer Science*, 7(5):445–452, 1997.

[7] Christian Retoré. On the relation between coherence semantics and multiplicative proof nets. Rapport de Recherche RR-2430, INRIA, décembre 1994. http://www.inria.fr/.

[8] Christian Retoré. Une modalité autoduale pour le connecteur "précède". In Pierre Ageron, editor, *Catégories, Algèbres, Esquisses et Néo-Esquisses*, Publications du Département de Mathématiques, Université de Caen, pages 11–16, September 1994.

[9] Alessio Guglielmi. A calculus of order and interaction. Technical Report WV-99-04, Dresden University of Technology, 1999.

[10] Alessio Guglielmi and Lutz Straßburger. Non-commutativity and MELL in the calculus of structures. In L. Fribourg, editor, *CSL 2001*, volume 2142 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, 2001.

[11] Jean-Yves Pollock. *Langage et cognition: le programme minimaliste de la grammaire générative*. Presses Universitaires de France, Paris, 1997.

lisés font partie de la famille des grammaires catégorielles [1,2,3,4]. Il faut, à mon avis, bien distinguer une théorie linguistique d'un formalisme linguistique, qui n'en est qu'une implantation.

La théorie sous-jacente à nos travaux relève de la grammaire générative parce qu'elle en partage les fondations :

- Le rôle central de la syntaxe, une syntaxe au sens large qui inclut la structure prédicative des énoncés, le principal objet d'étude étant, par conséquent *la phrase*.
- La conception calculatoire de la langue, et en particulier de la syntaxe.
- La volonté de rendre compte, par un petit nombre de principes simples et indépendants de la langue considérée, non seulement des mécanismes d'analyse et de production d'énoncés suivant la grammaire de la langue, mais aussi du processus d'acquisition de cette grammaire.
- Le parti pris d'avoir des grammaires lexicalisées, qui réduisent les variations entre les langues à une variation du lexique, tandis que les règles grammaticales sont communes à toutes les langues.

Se réclamer de la grammaire générative dans des formalismes catégoriels peut surprendre. En effet les formalismes catégoriels sont a priori rétifs aux déplacements de constituants pourtant chers aux théories chomskyennes successives. Cette divergence se résout de deux manières : d'une part les déplacements semblent un ingrédient moins fondamental que ceux précités ; d'autre part il se retrouvent sous une autre forme et, qui plus est, il est possible de rendre compte des déplacements de constituants dans des formalismes catégoriels, comme on le verra ci-après.

2.2.1 Le renouveau des grammaires catégorielles

Les grammaires catégorielles sont issues de la logique en ce sens qu'elles sont apparues comme un formalisme pour décrire la correction des énoncés logiques au

---

[12] Ray Jackendoff. *The Architecture of the Language Faculty*. Number 28 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1995.

[13] Andrew Radford. *Syntactic theory and the structure of English — a minimalist approach*. Cambridge University Press, 1997.

[1] Michael Moortgat. *Categorial Investigations*. Foris, Dordrecht, 1988.

[2] Glyn V. Morrill. *Type Logical Grammar*. Kluwer Academic Publishers, Dordrecht and Hingham, 1994.

[3] Michael Moortgat. Categorial type logic. In van Benthem and ter Meulen [vBtM97], chapter 2, pages 93–177.

[4] Wojciech Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [vBtM97], chapter 12, pages 683–736.

moyen d'un calcul de fractions, ce dont rendent compte les travaux d'Ajdukiewicz [5]. Bar-Hillel a ensuite traité de la syntaxe des langues par un calcul directionnel de fractions [6]. Joachim Lambek a alors complété ce calcul, aussi bien au sens courant qu'au sens logique du terme « complété », et en a fait une logique digne de ce nom [7,8].

On remarquera qu'à ce moment là il n'y a nulle opposition entre la grammaire générative et les grammaires de Lambek. Noam Chomsky dans [9] présente d'ailleurs ces dernières comme une alternative aux grammaires syntagmatiques, peu après l'apparition du calcul de Lambek. Par la suite le succès des grammaires syntagmatiques a oblitéré l'intérêt pour les grammaires catégorielles.

Leur retour sur la scène, dans les années 70, passe par les travaux de Richard Montague [10], lequel utilise, pour calculer des représentations sémantiques, des grammaires catégorielles. Cette interface aisément calculable entre analyse syntaxique et structure prédicative est une des raisons du juste renouveau des grammaires catégorielles. En effet, établir qu'une phrase est grammaticale ou non n'est pas réellement un but en soi : obtenir une représentation sémantique, généralement conçue comme une formule logique, fusse-t-elle assez pauvre, est un objectif important de l'analyse syntaxique[e]. Cependant, pour les formalismes plus usuels utilisés dans le traitement automatique de la syntaxe, obtenir de telles représentations sémantiques est ardu, et c'est une des raisons qu'a la communauté TAGs de se rapprocher des grammaires catégorielles comme l'a fait Aravind Joshi avec les arbres de preuve partiels [11,12,13].

e. Notons que le calcul de la structure prédicative et des coréférences relève, dans la tradition générative, de la *syntaxe*.

[5] Kazimierz Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935. (English translation in [McC67], pages 207–231).

[6] Yehoshua Bar-Hillel. A quasi arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

[7] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[8] Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, 1961.

[9] Noam Chomsky. Formal properties of grammars. In *Handbook of Mathematical Psychology*, volume 2, pages 323 – 418. Wiley, New-York, 1963.

[10] Richmond Thomason, editor. *The collected papers of Richard Montague*. Yale University Press, 1974.

[11] Aravind Joshi and Seth Kulick. Partial proof trees as building blocks for a categorial grammar. In Morrill and Oehrle [MO95], pages 138–149.

[12] Aravind Joshi and Seth Kulick. Partial proof trees, resource sensitive logics and syntactic constraints. In Retoré [Ret97a], pages 21–42.

[13] Aravind Joshi and Seth Kulick an Natasha Kurtonina. An LTAG perspective on categorial

Mais c'est aux travaux logiques qu'il convient d'attribuer le renouveau dans lequel notre travail se situe. Le calcul de Lambek, qui consiste à voir les simplifications de fractions comme des *modus ponens* et à adjoindre les règles d'introduction des implications, donne effectivement un système logique. Celui-ci est à l'époque éloigné des systèmes logiques usuels, et la théorie de la démonstration du moment a d'autres intérêts. C'est tout d'abord Johan van Benthem et Michael Moortgat [1,2] qui étudient le lien entre cette logique et la logique pertinente (*relevant*) puis considèrent certaines extensions qui la relient à la logique modale.

Comme en témoigne le titre de ce document, c'est plutôt les liens avec la logique linéaire qui ont retenu notre attention. Pour être plus précis mais réducteur, disons que la logique linéaire apporte selon nous aux grammaires catégorielles deux atouts : d'une part une meilleure compréhension de l'interface entre syntaxe et sémantique, d'autre part la syntaxe des réseaux de démonstration. Le premier point s'explique facilement. Les analyses syntaxiques sont des démonstrations dans le calcul de Lambek, celui-ci étant le fragment multiplicatif de la logique linéaire intuitionniste non commutative ; ce sont donc en particulier des démonstrations dans la logique linéaire intuitionniste. Quant aux représentations sémantiques, elles sont décrites par des λ-termes simplement typés sur les types de base e (individus) et t (valeurs de vérité) et ce sont donc aussi des démonstrations en logique intuitionniste, et donc en logique linéaire intuitionniste. La logique linéaire permet donc de n'avoir qu'une seule sorte de structures tant pour les analyses syntaxiques que pour les représentations sémantiques : le passage des unes aux autres s'en trouve clarifié et automatisé, comme nous l'avons montré avec Philippe de Groote dans [3].

Le second point, les réseaux de démonstration, ouvre bien plus de perspectives. Si les analyses dans les grammaires de Lambek sont des démonstrations formelles, il importe de savoir comment deux analyses diffèrent l'une de l'autre. Or les formalismes classiques pour les démonstrations formelles ont le fâcheux inconvénient de proposer maintes démonstrations différentes pour une même analyse syntaxique, c'est-à-dire pour une même consommation des valences. Les réseaux de démonstration justifient le slogan *parsing-as-deduction*, où l'analyse syntaxique est vue comme une déduction. Dans le formalisme des réseaux, une déduction est effectivement une analyse syntaxique, en ce sens que deux déduc-

---

inference. In Moortgat [Moo01a], pages 90–105.

[1] Johan van Benthem. *Language in Action: Categories, Lambdas and Dynamic Logic*, volume 130 of *Sudies in logic and the foundation of mathematics*. North-Holland, Amsterdam, 1991.

[2] Michael Moortgat. *Categorial Investigations*. Foris, Dordrecht, 1988.

[3] Philippe de Groote and Christian Retoré. Semantic readings of proof nets. In Geert-Jan Kruijff, Glyn Morrill, and Dick Oehrle, editors, *Formal Grammar*, pages 57–70, Prague, August 1996. FoLLI.

tions diffèrent si et seulement si elles décrivent des analyses syntaxiques différentes. Mettre en correspondance bijective la consommation des valences et la notion d'analyse synatxique nous rapporche des grammaires de dépendance, mêmes si la notion de valence des grammaires de dépendance est différente. De manière plus prospective, les réseaux de démonstration suggèrent des extensions des grammaires catégorielles. Nous avons exploré l'une d'entre elles qui consiste à associer aux mots des démonstrations incomplètes, et un autre modèle proposé est issu des réseaux de démonstration, même si ceux-ci font encore défaut pour la logique considérée. On détaillera cela ci-après.

Plus récemment, j'ai noté un parallèle entre les réseaux de démonstration et les deux modes de description des énoncés grammaticaux. Chomsky propose une distinction entre théories *représentationnelles* et théories *dérivationnelles*. Les premières décrivent la classe des énoncés possibles ou plutôt la classe de leur structures d'analyse par un ensemble de contraintes qui doivent être satisfaites pour que la structure soit grammaticale. Les secondes donnent un ensemble de règles de production qui engendrent exactement toutes les structures d'analyse qui sont grammaticales. Aux théories représentationnelles, correspondent la définition des réseaux de démonstration par l'ensemble des graphes satisfaisant le critère de correction. Aux théories dérivationnelles, fait écho la définition inductive des réseaux de démonstration, qui suit la structure d'une démonstration dans les formalismes plus usuels, calcul des séquents ou déduction naturelle. L'équivalence de ces deux types de descriptions est donnée par le théorème de séquentialisation, mais on remarquera que, dans une théorie représentationnelle, il n'y a qu'une structure d'analyse pour un objet donné, tandis que dans une théorie dérivationnelle diverses dérivations peuvent *a priori* conduire à la même structure d'analyse. Le parallèle est en fait plus subtil. Pour obtenir les analyses syntaxiques et non des démonstrations générales, il est nécessaire dans un cas d'ajouter des contraintes additionnelles de nature non logique, et dans l'autre, de contraindre les règles de dérivation, en particulier l'ordre selon lequel elles sont appliquées.

Le modèle le plus proche de ceux que nous avons développés est assurément celui des grammaires catégorielles multimodales de Michael Moortgat [2,4] poursuivi par Glyn Morrill [5]. Bien que partageant une philosophie commune, celle des grammaires catégorielles, nos travaux s'en distinguent néanmoins. L'idée de base des grammaires multimodales est la suivante. Le calcul de Lambek non associatif est le noyau de ce calcul, et il y a en général plusieurs copies de ce calcul. Des modalités, introduites par paires et qui satisfont toutes les mêmes règles lo-

---

[4] Michael Moortgat. Categorial type logic. In van Benthem and ter Meulen [vBtM97], chapter 2, pages 93–177.

[5] Glyn V. Morrill. *Type Logical Grammar*. Kluwer Academic Publishers, Dordrecht and Hingham, 1994.

giques étendent ce noyau, contrôlent la gestion des ressources et gèrent le passage d'un type de composition à un autre [1]. Ce sont alors des postulats, ou axiomes extra logiques, qui régissent et distinguent le comportement des modalités et des connecteurs ; en effet, les règles logiques ne différencient pas. Notre approche s'en écarte en ce sens que pour enrichir le calcul, nous n'avons pas recours à des postulats. Pour étendre la capacité générative, nous nous sommes intéressés à deux extensions concurrentes. D'une part en considérant que des démonstrations partielles, et non de simples formules, peuvent être associées aux mots par le lexique. D'autre part en utilisant des contraintes sur les dérivations, et en calculant l'ordre des mots à partir des démonstrations, de manière un peu plus subtile que dans les grammaires catégorielles usuelles. Ces extensions sont formellement issues de la logique linéaire, et ce choix est relativement naturel : si la logique (linéaire) est un modèle universel du calcul, alors il est raisonnable d'essayer de rendre compte de la faculté de langage, supposée être un processus calculatoire spécifique, grâce aux notions et outils fournis par la logique.

Un formalisme courant dont nous nous sommes également rapprochés est celui des grammaires d'arbres adjoints, TAGs, [2,3]. On souhaite augmenter la capacité générative des grammaires catégorielles et celle des TAGs est linguistiquement pertinente : les TAGs décrivent tous les langages faiblement contextuels, ce qui, du dire de nombreux linguistes est une classe raisonnable pour décrire les langages humains, et de plus, les TAGs bénéficient d'algorithmes d'analyse polynomiaux. Ce rapprochement est possible parce que les TAGs sont, tout comme les grammaires catégorielles, des grammaires lexicalisées, et que l'une des deux opérations des TAGs, la substitution, est déjà présente dans les grammaires catégorielles.

2.2.2 Un modèle grammatical issu du calcul ordonné

Un modèle classique

Le premier modèle que nous ayons conçu [4] étend les grammaires catégorielles dans deux directions :

– D'une part des modules, c'est-à-dire des parties de réseaux de démonstra-

[1] Natasha Kurtonina and Michael Moortgat. Structural control. In P. Blackburn and M. de Rijke, editors, *Specifying Syntactic Structures*, pages 75–113. CSLI, 1997. Distributed by Cambridge University Press.

[2] Aravind Joshi, Leon Levy, and Masako Takahashi. Tree adjunct grammar. *Journal of Computer and System Sciences*, 10:136–163, 1975.

[3] Anne Abeillé. *Les nouvelles syntaxes*. Armand Colin, 1993.

[4] Alain Lecomte and Christian Retoré. Pomset logic as an alternative categorial grammar. In Morrill and Oehrle [MO95], pages 181–196.

tion sont associées aux mots, et non de simples formules.

– D'autre part la logique sous-jacente n'est pas le calcul de Lambek mais le calcul ordonné développé dans ma thèse.

La motivation de ce changement est la trop grande rigidité du calcul de Lambek, dont les formules n'expriment l'ordre des mots que pour des constituants contigus. Le calcul ordonné permet pour sa part d'exprimer qu'un constituant doit suivre où précéder un autre constituant, mais sans nulle nécessité que les constituants impliqués soient contigus. Il n'est pas possible d'utiliser ce calcul comme un analogue du calcul de Lambek avec $A^{\perp} < B$ à la place de $A \setminus B$, car par exemple $A^{\perp} < A$ n'est pas démontrable. De plus, l'ordre sur les mots ne peut plus se traduire par un ordre sur les conclusions : deux conclusions sont nécessairement incomparables dès qu'il y a un lien entre les deux, notamment lorsqu'une valence est fournie par une conclusion à une autre. Cela nous a conduit à définir l'ordre sur les mots comme un ordre entre les axiomes apparaissant dans la démonstration, et il a alors fallu associer aux mots non de simples formules mais des modules qui comporte des axiomes associés aux mots. L'ordre des mots obtenu est un ordre partiel, et toutes les linéarisations obtenues sont considérées comme autant d'énoncés corrects. Cet ordre entre axiomes est défini dans un réseau de démonstration, comme la clôture transitive de la relation suivante :

> *Étant donnés deux axiomes* $ax_1 : a - a^{\perp}$ *et* $ax_2 : b - b^{\perp}$ *on a* $ax_1 < ax_2$ *lorsque l'une des conclusions du réseau est* $F[U < V]$ *et que* U *contient* a *ou* $a^{\perp}$ *tandis que* V *contient* b *ou* $b^{\perp}$.

Le fait que cette relation soit acyclique provient bien entendu de la correction du dit réseau.

On arrive alors à décrire des constructions syntaxiques qui jusque là échappaient au calcul de Lambek. Citons par exemple :

– L'ordre relativement libre des mots comme avec les verbes de perception en français :

(2.6) Pierre entend Marie chanter.

(2.7) Pierre entend chanter Marie.

– Les constituants discontinus, telle la négation *ne…pas* en français :

(2.8) Pierre *ne* dort *pas*.

Le travail de modélisation dans ce cadre a été poursuivi notamment par Irene Schena [5] qui a pu rendre compte de la topicalisation, ou de l'extraposition en italien :

(2.9) Maria, Piero ama.

---

[5] Irene Schena. Pomset logic and discontinuity in natural language. In Retoré [Ret97a], pages 386–405.

et du déplacement de constituants, comme les questions (wh-movement) même doubles :

(2.10) Che cosa dice la radio che fa Mario.

Un modèle intuitionniste

Le modèle classique précédent avait un avantage à nos yeux qui nous a valu certaines objections : les deux constituants composés, ou les deux modules leur correspondant, jouent des rôles totalement symétriques alors que la plupart des théories syntaxiques attribuent un rôle privilégié à l'un des deux constituants lorsqu'ils se composent, et certains tels Richard Kayne, voient même dans cette antisymétrie une propriété caractéristique de la syntaxe [1]. Par ailleurs, on souhaite un formalisme qui décrive les mécanismes de composition, mais qui soit néanmoins le plus restreint possible . Clairement, le modèle classique proposé est très vaste.

C'est pourquoi, dans les articles suivants [2,3], nous avons restreint la forme des modules qu'il est possible d'associer à un mot. Nous avons alors utilisé des modules intuitionnistes, qui ont une conclusion privilégiée, ce qui nous a rapproché des modèles existants, et en particulier des TAGs.

Une autre raison pour se restreindre à un sous calcul intuitionniste était la correspondance avec la sémantique de Montague. En effet, celle-ci repose sur l'application de fonctions à des arguments, ce qui est contraire au mode de composition symétrique que nous avions proposé. De fait, le calcul des représentations sémantiques s'avérait délicat, même si nous y sommes toujours arrivés pour les exemples étudiés. Cela résultait de la forme des modules utilisés, qui possédaient toujours une conclusion privilégiée. Par suite nous nous sommes restreints à des modules intuitionnistes, sans que cela entrave notre travail de modélisation. Effectivement le modèle proposé permet de calculer les représentations sémantiques en suivant des chemins qui s'apparentent à ceux définis par François Lamarche [4]. Le connecteur « précède » se voit alors attribuer des polarités, comme les connecteurs intuitionnistes.

---

[1] Richard Kayne. *The Antisymmetry of Syntax*. Number 25 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1994.

[2] Alain Lecomte and Christian Retoré. Words as modules and modules as partial proof nets in a lexicalised grammar. In Abrusci and Casadio [AC96], pages 187–198.

[3] Alain Lecomte and Christian Retoré. Words as modules: a lexicalised grammar in the framework of linear logic proof nets. In Carlos Martin-Vide, editor, *Mathematical and Computational Analysis of Natural Language — selected papers from ICML'96*, volume 45 of *Studies in Functional and Structural Linguistics*, pages 129–144. John Benjamins publishing company, 1998.

[4] François Lamarche. Proof nets for intuitionistic linear logic: Essential nets. 35 page technical report available by FTP from the Imperial College archives, April 1994.

Nous en avons profité pour expliciter le lien avec les TAGs. Des portes sur les liens axiomes correspondent précisément à l'adjonction, tandis que la substitution correspond au branchement de modules. Sylvain Pogodalla a effectivement montré que le modèle intuitionniste basé sur le calcul ordonné décrit toutes les grammaires TAGs [5].

A travers les notions que nous avons pu dégager dans ce modèle nous avons constaté que les déplacements des théories chomskyennes correspondaient à certains chemins dans les réseaux de démonstration [6].

2.2.3 Une formalisation logique du programme minimaliste de Chomsky

Le programme minimaliste et ses aspects logiques

Le programme minimaliste de Noam Chomsky [7,8,9] est à ce jour le dernier modèle proposé par les grammaires génératives, et il propose de retrouver par une grammaire lexicalisée les principes dégagés dans la théorie du gouvernement et du liage (GB) [10]. Pour une présentation plus accessible, on se référera au cours d'Andrew Radford [11] ou de Jean-Yves Pollock [12], ou encore à l'ouvrage plus général de Juan Uriegereka [13].

GB est une théorie stratifiée par plusieurs niveaux de représentations (structure profonde, structure de surface) auxquels s'appliquent des modules qui rendent compte de phénomènes spécifiques à chaque strate. Reprenons un exemple dont nous avons déjà parlé en 1.5 et 1.6 au chapitre précédent. Dans la phrase « *Combien de livres que Chomsky a écrit a-t-il aimé ?* » la coréférence entre le pronom anaphorique *il* et le nom propre *Chomsky* est possible, mais ne l'est pas dans la

---

[5] Sylvain Pogodalla. Lexicalized proof-nets and TAGs. In Moortgat [Moo01a], pages 230–250.

[6] Alain Lecomte and Christian Retoré. Logique des ressources et réseaux syntaxiques. In D. Genthial, editor, 4 ème conférence sur le Traitement automatique du langage naturel, TALN'97, pages 70–83, Grenoble, 1997.

[7] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

[8] Noam Chomsky. Minimalist inquiries: the framework. Reading room copy, 1998.

[9] Noam Chomsky. Beyond explanatory adequacy. Reading room copy, 2001.

[10] Noam Chomsky. *La nouvelle syntaxe*. Seuil, Paris, 1987. Traduction de [Cho82].

[11] Andrew Radford. *Syntactic theory and the structure of English — a minimalist approach*. Cambridge University Press, 1997.

[12] Jean-Yves Pollock. *Langage et cognition: le programme minimaliste de la grammaire générative*. Presses Universitaires de France, Paris, 1997.

[13] Juan Uriegereka. *Rhyme and reason: an introduction to minimalist syntax*. MIT Press, 1998.

phrase « *Il a aimé trois livres que Chomsky a écrit.* », même si les structures profondes de ces phrases sont les mêmes. Cela tient à la position dans la structure de surface du pronom anaphorique *il*. Dans le premier cas, *Chomsky* gouverne *il*, ce qui n'est plus le cas dans la seconde phrase, et la notion de gouvernement ne s'applique qu'à la structure de surface.

Un autre exemple classique est le suivant :

(2.11) Il semble que Chomsky travaille beaucoup.

(2.12) Chomsky semble travailler beaucoup.

(2.13) *Il semble que Chomsky travailler beaucoup.

(2.14) *Il semble Chomsky travailler beaucoup.

Les deux derniers exemples sont incorrects, avec ou sans le la conjonction de subordination « *que* » parce que tout groupe nominal référentiel, ici « *Chomsky* », doit recevoir un cas, nominatif, et que celui-ci n'est fourni que par les verbes conjugués, ce qui impose à *Chomsky*, lorsque le verbe est à l'infinitif, de se déplacer à la place de *il* qui est un emplacement vide.

De tels principes sont séduisants, et si, comme dans le programme minimaliste, ils s'obtiennent par des grammaires lexicalisées, où aussi bien les compositions (*merge*) que les déplacements (*move*) sont dûs à la nécessité que des traits s'annulent, le rapprochement avec les grammaires catégorielles et la logique linéaire est tentant.

Un modèle logique du programme minimaliste

Edward Stabler a proposé en 96 [1] une formalisation du programme minimaliste de Noam Chomsky sous la forme de grammaires d'arbres. Ce sont des grammaires totalement lexicalisées, où chaque entrée est une liste de traits, et où l'assemblage des constituants (*merge*, opération binaire) ainsi que leur déplacement à l'intérieur d'un arbre déjà constitué (*move*, opération unaire) sont gérés par l'annulation de traits de polarité opposée. Cela nous a permis de préciser et de formaliser le lien entre grammaires catégorielles et programme minimaliste, en faisant un autre usage grammatical du calcul de Lambek que celui proposé initialement [2]. Les entrées lexicales d'Edward Stabler sont traduites par des formules du calcul de Lambek, et l'on doit obtenir une démonstration dans le calcul de Lambek du symbole initial de la grammaire. Bien sûr, la difficulté est de rendre compte des déplacements de constituants ; cela est fait en étiquetant la démonstration. Les étiquettes sont des chaînes qui comportent à la fois les informations phonologiques, mots ou traits d'accord, et des informations sémantiques que l'on rencontre aussi

[1] Edward Stabler. Derivational minimalism. In Retoré [Ret97a], pages 68–95.

[2] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

dans la sémantique de Montague : variables, quantificateurs,... Lors de la règle d'élimination du produit, l'étiquette $e : A \bullet B$ de la formule principale détermine quelles chaînes doivent remplacer les deux variables $x : A$ et $y : B$ que cette règle lie ; suivant la nature forte ou faible du trait A soit $x := e$ et $y := \epsilon$ ou $x := (e)$ (la partie sémantique de e) et $y := /e/$ la partie phonologique de e. On notera que les étiquette sont complètement calculables à partir de la démonstration formelle et ne guident en rien sa construction.

Ces travaux intéressent les deux domaines rapprochés : d'une part les grammaires catégorielles peuvent ainsi profiter de la profondeur d'analyse de la théorie chomskyenne, et d'autre part les grammaires minimalistes acquièrent ainsi une interface plus aisée avec la sémantique, par exemple la sémantique de Montague. Le premier travail publié [3] présente les idées de base. Le second [4] étend la correspondance à des mouvements plus compliqués (dits de tête) en utilisant une extension conservative du calcul de Lambek : la logique linéaire intuitionniste partiellement commutative de Philippe de Groote [5]. Le mouvement de tête est par exemple nécessaire pour rendre compte des langues VSO, telles l'arabe, qui nécessitent un déplacement du verbe:

(2.15) yondhoro Latifa Malik
regarde Latifa Malik
Verbe Sujet Objet
« Latifa regarde Malik »

Nous avons également abordé le calcul de la sémantique de Montague dans ce cadre [6]. Nous devons reconnaître que c'est bien moins facile que pour le calcul de Lambek original. Il faut transformer les règles, en remplaçant une règle par des règles interprétables en termes d'applications et d'abstraction, pour donner à la démonstration une forme dont le λ-terme s'extrait facilement, par l'isomorphisme de Curry-Howard.

Une des critiques que l'on adresse fréquemment aux grammaires catégorielles est l'absence de rapport entre la tête d'un syntagme et le constituant qui porte le

---

[3] Alain Lecomte and Christian Retoré. Towards a minimal logic for minimalist grammars: a transformational use of Lambek calculus. In *Formal Grammar, FG'99*, pages 83–92. FoLLI, 1999.

[4] Alain Lecomte and Christian Retoré. A logical formulation of the minimalist program. In *Third Tbilisi Symposium on Language, Logic and Computation*. FoLLI, 1999.

[5] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

[6] Alain Lecomte and Christian Retoré. Extending Lambek grammars: a logical account of minimalist grammars. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, ACL 2001*, pages 354–361, Toulouse, July 2001. ACL.

type fonctionnel.

(2.16)

green	ideas
$n \,/\, n$	n
argument	function

(2.17)

Pierre	dort
sn	$sn \setminus S$
argument	fonction

Dans le premier exemple c'est le nom, c'est-à-dire l'argument, qui est la tête du groupe nominal, tandis que dans le second c'est le verbe, c'est-à-dire la fonction qui est la tête de la phrase[f]. Cette critique avait amené Michael Moortgat et Glyn Morrill à dédoubler les implications, suivant que la fonction est ou non la tête du syntagme résultant [2]. Le modèle catégoriel des grammaires minimalistes que nous avons proposé échappe à ces critiques : en effet nous retrouvons bien sûr les notions de spécifieurs, de tête et de compléments d'un syntagme de la grammaire générative. Celles-ci ne satisfont pas la communauté des grammaires de dépendance, ce à quoi nous objecterons que leur notion de tête est plutôt sémantique et qu'il n'y a pas de raison que la tête syntaxique d'un syntagme coïncide avec sa tête sémantique. Par exemple dans les théories chomskyennes actuelles, c'est le déterminant qui est la tête du syntagme nominal, même s'il est absent dans certaines langues.

(2.18) [[ε Cats] [chase [ε rats]]]

La connexion entre calcul de Lambek et grammaires minimalistes n'est qu'un aspect du rapprochement entre la grammaire générative et les grammaires logiques. C'est pourquoi avec Edward Stabler nous avons organisé un workshop et écrit un article de synthèse sur cette convergence [3].

f. On retrouve ce même phénomène dans HPSG [1] : le modifieur est la tête parce que ses traits sémantiques sont propagés à l'élément modifié, mais il n'est pourtant pas la tête syntaxique.

[1] Carl Pollard and Ivan A. Sag. *Head Driven Phrase Structure Grammar*. Center for the Study of Language and Information, Stanford, CA, USA, 1987. (distributed by Cambridge University Press).

[2] Michael Moortgat and Glyn Morrill. Heads and phrases – Type calculus for dependency and constituent structure. Technical report, OTS, Utrecht, 1991.

[3] Christian Retoré and Edward Stabler, editors. *Resource Logics and Minimalist Grammars*, European Summer School in Logic Language and Information, Utrecht, 1999. FoLLI. Forthcoming special issue of *Language and Computation*, featuring a twenty-page introduction by the editors (INRIA Research Report RR-3780).

2.2.4 Applications au traitement automatique des langues

Tandis que les modèles proposés sont assez abstraits, on remarquera qu'on produit des modèles du fonctionnement de la langue, en s'inspirant des théories linguistiques les plus récentes, et dans un cadre général calculatoire. On peut donc envisager de mettre en pratique la vision calculatoire de la faculté de langage que la logique fournit, mais comme nous en avons débattu au chapitre précédent, il n'est pas sûr que la simulation d'un comportement humain, ou la formalisation des théories linguistiques, qui est notre parti pris, soit des plus efficaces. En ce sens on peut dire qu'il s'agit d'intelligence artificielle, expression pourtant peu admise dans la communauté de la logique linéaire, puisqu'on simule un comportement humain.

Il est néanmoins tout à fait légitime d'envisager la conception de logiciels basés sur ces principes : analyseurs syntaxiques, générateurs automatiques, voire des traducteurs assistés par ordinateur... L'un des attraits des modèles proposés est l'interface aisée avec la sémantique : par exemple la reformulation dans la langue source permet à l'utilisateur de préciser un énoncé ambigu, de lui demander de choisir au travers de reformulations qui lui seraient proposées, l'énoncé qu'il souhaite traduire, notamment en cas d'ambiguïté réelle [g], un exemple classique étant :

(2.19) La petite brise la glace

Bien sûr, nous sommes conscients que la sémantique d'une simple phrase est quelque chose d'excessivement complexe. Mais même une vision très simplifiée de la sémantique, avec une interface aisée avec la syntaxe, permet d'améliorer considérablement les logiciels que l'on rencontre sur Internet — par xemple les traductions de critiques de disques que proposait le site *All Music Guide*.

Outre les algorithmes d'analyse syntaxique produisant les représentations sémantique que nous avons décrits, notre étudiant Sylvain Pogodalla a obtenu de bons résultats en génération d'énoncés à partir de représentations sémantiques

g. Une ambiguïté réelle s'oppose à une ambiguïté fallacieuse qui n'existe que pour le modèle et non pour le locuteur. Un exemple d'ambiguïté sémantique fallacieuse est *Les avocats mûrissent dans les pays chauds.*, et un exemple d'ambiguïté syntaxique fallacieuse est *[Jean [mange une]] pomme.*, et un mélange des deux est *[Il regarde la passante] avec une robe rouge*. Les exemples du type *[Jean [mange une]] pomme.* peuvent être produits part des grammaires de Lambek si l'on prend comme parenthésage induit celui proposé par Wojciech Buszkowski[4], car alors tout parenthésage est le fruit d'une analyse. Fort heureusement, avec la notion introduite par Hans Jörg Tiede[5], ce n'est pas le cas.

[4] Wojciech Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [vBtM97], chapter 12, pages 683–736.

[5] Hans-Jörg Tiede. *Deductive Systems and Grammars: Proofs as Grammatical Structures*. PhD thesis, Illinois Wesleyan University, 1999.

[1,2,3,4]. Autant le dire, une partie du travail est éludée, si l'on part de représentations sémantiques à la Montague : l'une des difficultés de la génération est de partir de représentations sémantiques accessibles à partir des données à exprimer, et l'autre est le choix des mots et des structures de l'énoncé. Néanmoins, il a su montrer que les méthodes formelles pouvaient traiter de cette question restreinte mais importante.

On remarquera que le développement de telles applications est aussi une question de moyens et de temps. Par exemple l'équipe de Michael Moortgat, et en particulier Richard Moot, ont su développer un analyseur opératoire GRAIL[5,6] basé sur des techniques relativement similaires et donc aussi complexes et qui a atteint une certaine ampleur, notamment en incluant des modules pour les pronomsclitiques du français et de l'italien.

2.2.5 Acquisition des grammaires catégorielles

Une des étapes préalables à la réalisation de tout outil d'analyse ou de génération est l'écriture d'une grammaire, et donc d'un lexique. Si les mots grammaticaux, ou vides selon Lucien Tesnière [7], sont peu nombreux et connus d'avance, les mots imprévus qu'ils s'agisse d'erreurs, de néologismes ou d'entrées lexicales non saisies, sont fatals aux algorithmes d'analyse catégoriels. C'est pourquoi l'acquisition automatique d'entrées lexicales, est non seulement l'occasion de tester les hypothèses chomskyennes à ce propos, mais aussi un outil de traitement des langues. Nous avons montré que les grammaires minimalistes, ou plutôt leur codage dans le calcul partiellement commutatif sont apprenables à la limite au sens

---

[1] Sylvain Pogodalla. Generation with semantic proof nets. Research Report 3878, INRIA, January 2000. http://www.inria.fr/.

[2] Sylvain Pogodalla. Generation in the Lambek calculus framework: an approach with semantic proof nets. In *proceedings of NAACL 2000*, May 2000.

[3] Sylvain Pogodalla. Generation, Lambek calculus, montague's semantics and semantic proof nets. In *proceedings of Coling 2000*, August 2000.

[4] Sylvain Pogodalla. *Réseaux de preuve et génération pour les grammaires de types logiques*. Thèse de doctorat, spécialité informatique, INPL, Nancy, 2001.

[5] Richard Moot. Grail: An automated proof assistant for categorial grammar logics. In R.C. Backhouse, editor, *Proceedings of the 1998 User Interfaces for Theorem Provers Conference*, pages 120–129, 1998.

[6] Richard Moot. Automated deduction for categorial grammar logics: the grail prover. In E. Kraak and R. Wassermann, editors, *ACCOLADE'97*. ILLC, Universiteit van Amsterdam, 1997.

[7] Lucien Tesnière. *Éléments de syntaxe structurale*. Éditions Klincksieck, 1959. 5^{e} édition: 1988.

de Gold [8]. Les algorithmes d'apprentissage par unification des grammaires catégorielles classiques, initiées par Wojciech Buszkowsky et Gerald Penn [9] et la preuve de convergence de Makoto Kanazawa [10] s'étendent aux grammaires de Lambek, comme l'a montré Roberto Bonato [11].

En exploitant sa démonstration limpide nous avons montré dans [12] que les grammaires catégorielles minimalistes que nous avons définies constituent une classe apprenable suivant le paradigme de Mark Gold [8]. D'une part il faut disposer de structures d'analyse comme entrées, ce que des analyses dans les grammaires de dépendance, plus robustes peuvent fournir, mais qu'il faut traduire. D'autre part tel qu'il est rédigé, cet algorithme est non déterministe, et de plus, il utilise l'unification modulo commutativité et associativité, qui est d'une complexité rédhibitoire. La solution est assurément de prendre en compte la forme des types que l'unification doit produire, celle-ci étant *a priori* connue. Il se pourrait même, comme l'estiment certains auteurs, que seuls une trentaine de types soient possibles [13].

Grâce à notre action de recherche coopérative de l'INRIA *Acquisition des grammaires catégorielles*, nous avons pu employer Yoann Arsicaut, stagiaire de maîtrise de l'ENSAI, qui a réalisé un environnement de programmation pour programmer les algorithmes d'apprentissage des grammaires catégorielles. Les données ou exemples sont décrites comme des graphes orientés sans cycle (DAGs) en XML, les structures sont affichées en Tcl/Tk, et les algorithmes sont en OCaML.

Cette réalisation est poursuivie par Erwan Moreau, Yannick Le Nir dont nous co-encadrons les thèses, ainsi que par Jérôme Besombes et Jean-Yves Marion qui se sont penchés avec succès sur l'apprentissage des grammaires de dépendances [14].

Cet intérêt pour l'inférence grammaticale se développe aussi à l'étranger comme en témoignent les travaux récents de Michael Moortgat sur l'apprentissage des

---

[8] E. Mark Gold. Language identification in the limit. *Information and control*, 10:447–474, 1967.

[9] Wojciech Buszkowski and Gerald Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454, 1990.

[10] Makoto Kanazawa. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI, 1998. distributed by Cambridge University Press.

[11] Roberto Bonato. Uno studio sull'apprendibilità delle grammatiche di Lambek rigide — a study on learnability for rigid Lambek grammars. Tesi di Laurea & Mémoire de D.E.A, Università di Verona & Université Rennes 1, 2000.

[12] Roberto Bonato and Christian Retoré. Learning rigid lambek grammars and minimalist grammars from structured sentences. In Popelìnskỳ and Nepil [PN01], pages 23–34.

[13] Edward Stabler. Acquiring languages with movement. *Syntax*, 1:72–97, 1998.

[14] Jérôme Besombes and Jean-Yves Marion. Identification of reversible dependency tree languages. In Popelìnskỳ and Nepil [PN01], pages 11–22.

grammaires catégorielles multimodales,[1] et ceux, en cours d'Edward Stabler sur les grammaires minimalistes.

2.3 Conclusion

Jusqu'ici nous avons donné une synthèse très générale de nos travaux depuis la fin de notre thèse jusqu'à ce jour, qui essaie de les situer face aux travaux existants et de dégager des perspectives de recherches.

Parmi ces perspectives, nous tenons à souligner les questions techniques suivantes :

- Mise au point de réseaux de démonstration pour les calculs mixtes.
- Utilisation des réseaux de Petri pour l'analyse syntaxique dans les grammaires catégorielles mixtes.
- Définition d'un algorithme d'apprentissage effectif pour les grammaires catégorielles minimalistes.
- Automatisation du passage entre grammaires minimaliste et grammaires catégorielles.

ainsi que les directions plus prospectives que voici :

- Recherche d'un paradigme d'apprentissage plus réaliste linguistiquement que celui de Gold.
- Extension de la sémantique de Montague à certains aspects de la sémantique lexicale, peut-être par le biais de la sémantique dénotationnelle.
- Mises au point de calculs logiques plus souples, pour rendre compte du calcul ordonné comme un calcul de structures.

Les chapitres suivants vont maintenant résumer plus précisément, les recherches menées et celles envisagées, en les regroupant par thème:

- Graphes et logique linéaire.
  1. Graphes
  2. Réseaux de démonstration
  3. Sémantique dénotationnelle
  4. Calcul mixte et exécution des réseaux de Petri
- Modèles grammaticaux
  7. L'interface entre syntaxe et sémantique
  8. Un modèle issu du calcul ordonné
  9. Grammaires catégorielles minimalistes

[1] Michael Moortgat. Structural equations in language learning. In de Groote et al. [dGMR01], pages 1–16.

Deuxième partie

Graphes et logique linéaire

Chapitre 3

Graphes

Résumé : *Dans ce chapitre nous présentons les notions et les résultats combinatoires utiles aux réseaux de démonstration, qui concernent deux sujets indépendants de la théorie des graphes.*

La premier sujet concerne des classes inductives de graphes obtenus par des compositions simples définies sur la somme disjointe des sommets des deux graphes composés, tels les ordres séries-parallèles et les cographes. Ces graphes se décrivent par des termes linéaires dont les variables sont les sommets, et admettent une caractérisation universelle en termes de sous-graphes exclus. On donnera notamment un système de réécriture sur les termes qui axiomatise l'inclusion d'un graphe dans un autre. Ce travail a été réalisé avec Denis Bechet (Université Paris 13) et Philippe de Groote (INRIA Lorraine) [1].

Le second sujet concerne les couplage parfaits : nous donnons une définition inductive des graphes munis d'un couplage parfait et sans cycle élémentaire alternant ou, ce qui revient au même, des graphes munis d'un unique couplage parfait.

Finalement, on considérera les graphes munis d'un couplage parfait dont le complémentaire est un cographe orienté. C'est la structure dans laquelle se situent nos travaux sur les réseaux de démonstration. Le premier critère de correction [2]*, s'est ensuite simplifié en présence de cordes sur tout cycle élémentaire alternant* [3] *— un rapport donne tous les détails* [4].

---

[1] Denis Bechet, Philippe de Groote, and Christian Retoré. A complete axiomatisation of the inclusion of series-parallel partial orders. In H. Comon, editor, *Rewriting Techniques and Applications, RTA'97*, volume 1232 of *LNCS*, pages 230–240. Springer Verlag, 1997.

[2] Christian Retoré. Perfect matchings and series-parallel graphs: multiplicative proof nets as R&B-graphs. In J.-Y. Girard, M. Okada, and A. Scedrov, editors, *Linear'96*, volume 3 of *Electronic Notes in Theoretical Science*. Elsevier, 1996. (available from http://www.elsevier.nl/).

[3] Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 1999. To appear.

[4] Christian Retoré. Handsome proof-nets: R&B-graphs, perfect matchings and series-parallel graphs. Rapport de Recherche RR-3652, INRIA, March 1999. http://www.inria.fr/.

3.1 Graphes orientés et non orientés : définitions

Pour ces notions usuelles on pourra éventuellement consulter [1] ou l'introduction de [2], mais ces rappels devraient suffire. Ils sont surtout destinés à lever les ambiguïtés, car ces définitions sont sujettes à toutes sortes de variations.

Un **graphe orienté** $\mathcal{G} = (V; A)$ est défini par l'ensemble V de ses **sommets** et par un multi-ensemble A d'**arcs**, où un arc est un couple de sommets *distincts*. Un arc de source x et de but y sera noté (x,y). Si les arcs (x,y) sont toujours en même nombre que les arcs (y,x) on dira que le graphe est symétrique. Deux arcs (x,y) et (y,x) sont dits opposés.

Un **graphe non orienté** $\mathcal{G} = (V; E)$ consiste en un ensemble V de **sommets** et un multi-ensemble E d'**arêtes**, une arête étant une paire d'arcs opposés $(x,y),(y,x)$, parfois notée xy. Étant donné un graphe orienté $\mathcal{G} = (V; A)$ son **graphe non orienté sous-jacent** $|\mathcal{G}|$ est défini par $|\mathcal{G}| = (V; E)$ où E est défini comme suit : si la multiplicité de (x,y) dans A est p et celle de (y,x) est q alors la multiplicité de $\{x,y\}$ est $\max(p,q)$. Un graphe orienté (resp. non orienté), dont le multi-ensemble d'arcs (resp. d'arête) est un ensemble, est dit **simple**, c'est-à-dire sans arête multiple. Un graphe orienté est alors une relation binaire antiréflexive et un graphe non orienté une relation binaire antiréflexive et symétrique.

Un **chemin** dans un graphe orienté (resp. non orienté) est une suite alternant sommets et arcs, commençant et finissant par un sommet, et telle qu'entre deux points consécutifs x et y on trouve un arc xy (resp. une arête $\{x,y\}$). Lorsque le premier et le dernier sommet sont égaux, on parle de **circuit** (resp. **cycle**). Un chemin, un circuit ou un cycle est dit **élémentaire** lorsque tous ses sommets sont distincts (sauf le premier et le dernier dans le cas d'un cycle ou d'un circuit). Une **corde** d'un chemin, d'un cycle ou d'un circuit est un arc (x,y) reliant deux sommets du cycle tel que ni (x,y) ni (y,x) ne figure dans le chemin, cycle ou circuit.

Un sous-ensemble des sommets d'un graphe non orienté (resp. orienté) $\mathcal{G}$ est dit **connexe** (resp. **fortement connexe**) s'il existe toujours un chemin de x et y (resp. un chemin de x à y et un de y à x). Une **composante connexe** de $\mathcal{G}$ (resp. une composante fortement connexe) est un sous-ensemble de sommets connexe (resp. fortement connexe) maximale pour l'inclusion. Un **isthme** est une arête (resp. une paire d'arcs opposés) dont la suppression augmente le nombre de composantes connexes de $\mathcal{G}$.

---

[1] Claude Berge. *Graphes*. Gauthier-Villars, 1979.

[2] Làszlò Lovàsz and Michael David Plummer. *Matching Theory*, volume 121 of *Mathematics Studies*. North-Holland, 1986. Annals of Discrete Mathematics 29.

3.2 Ordres séries-parallèles, cographes et cographes orientés

3.2.1 Définitions

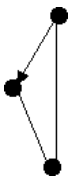
Ces structures ont été maintes fois (re)découvertes, le premier article les introduisant pour l'étude des réseaux électriques [3], les suivants pour celle de problèmes d'ordonnancement et pour celle des graphes parfaits. Une référence classique sur la question est [4], mais nous rappelons ici tout ce qui nous sera utile.

Les graphes envisagés dans ce paragraphe sont tous *orientés* et *simples* et on appelle $\mathsf{1}$ la classe de tous les graphes orientés à un sommet et sans arc. Soient $\mathcal{G} = (V; R)$ et $\mathcal{G}' = (V'; S)$, avec $V \cap V' = \emptyset$, deux graphes orientés sans sommet commun ; définissons leur composition en série $\mathcal{G} \mathbin{\widehat{<}} \mathcal{G}'$, leur composition en parallèle $\mathcal{G} \mathbin{\widehat{\wp}} \mathcal{G}'$ et leur composition symétrique $\mathcal{G} \mathbin{\widehat{\otimes}} \mathcal{G}'$ comme les graphes de sommets $V \uplus V'$ dont les arcs sont respectivement :

- $R \mathbin{\widehat{\wp}} S = R \uplus S$ — **composition en parallèle**
- $R \mathbin{\widehat{<}} S = R \uplus S \uplus (V \times V')$ — **composition en série**
- $R \mathbin{\widehat{\otimes}} S = R \uplus S \uplus (V \times V') \uplus (V' \times V)$ — **composition (en série) symétrique**

Par exemple si on considère les graphes :

$\mathcal{G}$



$\mathcal{G}'$

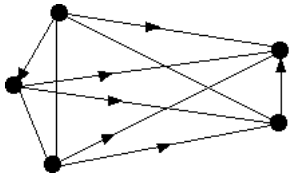
[3] John Riordan and Claude E. Shannon. The number of two-terminal series-parallel networks. *Journal of Mathematical Physics, Massachussets Institute of Technology*, 21:83–93, 1942.

[4] Rolf H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, volume 255 of *NATO ASI series C*, pages 105–194. Kluwer, 1989.

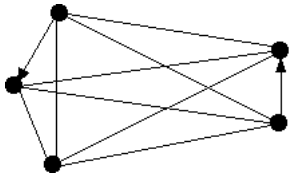
alors leurs composés sont :

$\mathcal{G} \mathbin{\widehat{\wp}} \mathcal{G}'$

$\mathcal{G} \mathbin{\widehat{<}} \mathcal{G}'$



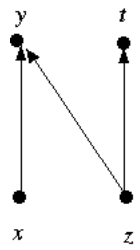
$\mathcal{G} \mathbin{\widehat{\otimes}} \mathcal{G}'$



On remarque aisément que ces compositions sont associatives, et qu'à l'exception de la composition en série elles sont commutatives.

La plus petite classe de graphes orientés contenant $\mathbf{1}$ et close par les compositions en série et en parallèle est la classe des **ordres séries parallèles** qui, comme leur nom l'indique, sont des ordres. D'après leur définition même, ces graphes admettent une écriture comme un terme linéaire sur V, qui est unique modulo l'associativité de $\widehat{\wp}$ et $\widehat{<}$ et la commutativité de $\widehat{\wp}$. Ces graphes ou relations sont caractérisés par le fait d'être sans N, c'est-à-dire que la restriction du graphe à quatre sommets n'est jamais $\{(x,y),(z,y),(z,t)\}$ — s'il y a une telle configuration, le graphe contient toujours au moins un autre arc dont les extrémités sont parmi x,y,z,t.

Le sous graphe exclu :

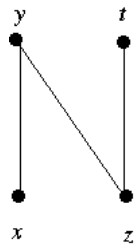


La plus petite classe de graphes orientés contenant $\mathsf{1}$ et close par les compositions en série symétrique et en parallèle est la classe des graphes séries parallèles ou **cographes**, qui sont bien sûr symétriques. D'après leur définition même, ces graphes admettent aussi une écriture comme un terme linéaire sur V, qui est unique modulo l'associativité et la commutativité de $\widehat{\wp}$ et $\widehat{\otimes}$. Ces graphes ou relations sont caractérisés par le fait d'être sans P_4, c'est-à-dire que la restriction du graphe à quatre sommets n'est jamais

$$\{xy,yz,zt,\} = \{(x,y),(y,x),(y,z),(z,y),(z,t),(t,z)\}$$

— s'il y a une telle configuration, le graphe contient toujours au moins une autre arête dont les extrémités sont parmi x,y,z,t.

Le sous graphe exclu:



La plus petite classe de graphes orientés contenant $\mathsf{1}$ et close par les trois types de composition envisagés n'a pas, à ma connaissance, été étudiée. Appelons ces graphes des **cographes orientés**. D'après leur définition même, ces graphes admettent eux aussi une écriture comme un terme linéaire sur V, qui est unique modulo l'associativité et la commutativité de $\widehat{\wp}$ et $\widehat{\otimes}$, et l'associativité de $\widehat{<}$. Étant donnée une relation $R \subset V^2$ définissons sa partie antisymétrique $R^{\uparrow}$ et sa partie symétrique $R^{\updownarrow}$ par : $R^{\uparrow} = \{(x,y) \in R \mid (y,x) \notin R\}$ et $R^{\updownarrow} = \{(x,y) \in R \mid (y,x) \in R\}$. On a bien sûr $R = R^{\uparrow} \uplus R^{\updownarrow}$.

Pour asseoir le lien avec la logique linéaire, déjà bien présente dans les notations, définissons l'**orthogonal** d'un cographe par :

$$\begin{array}{rcl} (x)^{\perp} & = & x^{\perp} \\ (x^{\perp})^{\perp} & = & x \\ (\phi \;\widehat{\wp}\; \phi')^{\perp} & = & (\phi)^{\perp} \;\widehat{\otimes}\; (\phi')^{\perp} \\ (\phi \;\widehat{<}\; \phi')^{\perp} & = & (\phi)^{\perp} \;\widehat{<}\; (\phi')^{\perp} \\ (\phi \;\widehat{\otimes}\; \phi')^{\perp} & = & (\phi)^{\perp} \;\widehat{\wp}\; (\phi')^{\perp} \end{array}$$

En notant $\overline{S}$ le graphe complémentaire d'un graphe symétrique S, l'orthogonal d'un cographe est donc une sorte de complément : $R^{\perp}$ est $R^{\uparrow} \uplus \overline{R}^{\updownarrow}$ dans lequel les noms des sommets sont échangés ($x := x^{\perp}$ et $y^{\perp} := y$).

3.2.2 Caractérisation des cographes orientés

Dans [1] nous avons donné une caractérisation universelle des cographes orientés :

Théorème 3.1 *Un graphe orienté $\mathcal{G} = (V; R)$ est un cographe orienté si et seulement si :*

- $R^{\uparrow}$ *est un ordre série-parallèle*
- $R^{\updownarrow}$ *est un cographe*
- R *est* ***faiblement transitive*** *:*
  $\forall x,y \in V \qquad \big((x,y) \in R \wedge (y,z) \in R^{\uparrow}\big) \Rightarrow (x,z) \in R$
  $\forall x,y \in V \qquad \big((x,y) \in R^{\uparrow} \wedge (y,z) \in R\big) \Rightarrow (x,z) \in R$

Remarque 3.2 *Les trois classes de graphes envisagées (ordres séries-parallèles, cographes, cographes orientés) sont stables par restriction — cela découle immédiatement de leur caractérisation universelle.*

Quelle que soit la classe de graphes considérée, parmi les trois mentionnées, étant donné un graphe $\mathcal{G}$ de cette classe on appellera **coterme** un terme linéaire sur V décrivant $\mathcal{G}$, et co-arbre l'unique arbre fini n-aire représentant $\mathcal{G}$.

3.2.3 Un système de réécriture complet pour l'inclusion des cographes

Étant donnés deux cographes $\mathcal{G} = (V; R)$ et $\mathcal{G}' = (V; S)$, comment voir sur des cotermes r et s associés respectivement à R et S que $R \subset S$? Dans [1] nous avons proposé un système $\multimap\!\bullet$ de réécriture modulo la commutativité de $\widehat{\wp}$, $\widehat{\otimes}$ et

[1] Denis Bechet, Philippe de Groote, and Christian Retoré. A complete axiomatisation of the inclusion of series-parallel partial orders. In H. Comon, editor, *Rewriting Techniques and Applications, RTA'97*, volume 1232 of *LNCS*, pages 230–240. Springer Verlag, 1997.

l'associativité de $\widehat{\wp}$, $\widehat{<}$, $\widehat{\otimes}$, qui axiomatise cette inclusion.

groupe	nom	règle
$(\otimes<)$	$(\otimes<4)$	$(X<Y) \otimes (U<V) \multimap (X\otimes U) < (Y\otimes V)$
	$(\otimes<l3)$	$(X<Y) \otimes U \multimap (X\otimes U) < Y$
	$(\otimes<r3)$	$Y \otimes (U<V) \multimap U < (Y\otimes V)$
	$(\otimes<2)$	$Y \otimes U \multimap U < Y$
$(\otimes\wp)$	$(\otimes\wp 4)$	$(X\wp Y) \otimes (U\wp V) \multimap (X\otimes U) \wp (Y\otimes V)$
	$(\otimes\wp 3)$	$(X\wp Y) \otimes U \multimap (X\otimes U) \wp Y$
	$(\otimes\wp 2)$	$Y \otimes U \multimap U \wp Y$
$(<\wp)$	$(<\wp 4)$	$(X\wp Y) < (U\wp V) \multimap (X<U) \wp (Y<V)$
	$(<\wp l3)$	$(X\wp Y) < U \multimap (X<U) \wp Y$
	$(<\wp r3)$	$Y < (U\wp V) \multimap U \wp (Y<V)$
	$(<\wp 2)$	$Y < U \multimap U \wp Y$

Ce système peut sembler un peu compliqué, mais toutes les règles peuvent se voir comme les instanciations possibles d'un unique schéma :

$$(R\triangledown S)\blacktriangle(R'\triangledown S') \multimap (R\blacktriangle R')\triangledown(S\blacktriangle S')$$

où $(\blacktriangle,\triangledown) \in \{(\otimes,<),(\otimes,\wp),(<,\wp)\}$ — $\blacktriangle$ est une composition plus forte que $\triangledown$, engendrant plus d'arêtes.

Les diverses instanciations s'obtiennent à partir de ce schéma général en remplaçant un ou plusieurs des quatre cotermes R,S,R',S' par le coterme vide (correspondant au cograph vide), qui est élément neutre de tous les types de compositions envisagées. Certaines des règles de réécriture obtenues ainsi étant identiques modulo les propriétés algébriques des connecteurs, les règles données ci-dessus suffisent.

Théorème 3.3 *Étant donnés $\mathcal{G} = (V; R)$ et $\mathcal{G}' = (V; S)$ deux*

1. *cographes*
2. *cographes orientés*
3. *ordres séries-parallèles*

de mêmes sommets on a $R \subset S$ si et seulement si $R \multimap S$, où $\multimap$ est la clôture réflexive et transitive des règles de réécriture

1. $(\otimes\wp)$
2. $(\otimes<)$, $(<\wp)$, $(\otimes\wp)$
3. $(<\wp)$

modulo l'associativité de $\widehat{\wp}$, $\widehat{<}$,$\widehat{\otimes}$ et la commutativité de $\widehat{\wp}$ et $\widehat{\otimes}$.

3.3 Graphes munis d'un unique couplage parfait : une définition inductive

Un **couplage** dans un graphe non orienté $\mathcal{G} = (V; E)$ est un *ensemble* d'arêtes $B \subset E$, deux à deux non adjacentes. Le couplage est dit **parfait** lorsque tout sommet est incident à une arête du couplage.

Par la suite nous aurons à considérer des couplages (parfaits) dans des graphes orientés. La définition est toute naturelle : un couplage dans un graphe orienté $\mathcal{G} = (V; A)$ est un **ensemble** d'arcs $B \subset A$, tel que $xy \in B$ entraîne $yx \in B$ et tel que deux arcs du couplage ne sont pas adjacents à moins d'être opposés ; en d'autres termes, c'est l'image réciproque d'un couplage du graphe non orienté sous-jacent. Le couplage est dit parfait lorsque tout sommet est incident à un arc du couplage, et par conséquent à exactement deux arcs (opposés) du couplage.

Étant donné un graphe (resp. un graphe orienté) $\mathcal{G} = (V; B,R)$ muni d'un couplage, un **chemin alternant** est un chemin dont les arêtes (ou les arcs, si $\mathcal{G}$ est orienté) sont alternativement dans B et dans R. Un cycle (resp. un circuit) est dit alternant s'il est un chemin alternant d'extrémités confondues *de longueur paire*. Si le chemin, cycle ou circuit alternant est élémentaire, c'est-à-dire si tous ses sommets sont distincts, sauf éventuellement le premier et le dernier dans le cas d'un cycle ou d'un circuit, alors on parlera d'un **chemin, cycle ou circuit alternant élémentaire**, ce qu'on abrègera en chemin, cycle ou circuit Æ .

Soit R&B$^+$ la plus petite classe de graphes R&B contenant le graphe vide et close par :

$$\left[\begin{array}{c} \mathcal{G} = (V; B,R) \in \text{R\&B}^+ \\ \mathcal{G}' = (V'; B',R') \in \text{R\&B}^+ \\ V \cap V' = \emptyset \\ x,x' \notin V \cup V' \\ V_1 \subset V \\ V_1' \subset V' \end{array}\right] \Longrightarrow \left(\begin{array}{l} V \uplus V' \uplus \{x,x'\}; \\ B \uplus B' \uplus \{\{x,x'\}\}, \\ R' \uplus R \uplus (V_1 \mathbin{\widehat{\otimes}} \{x\}) \uplus (V_1' \mathbin{\widehat{\otimes}} \{x'\}) \end{array}\right) \in \text{R\&B}^+$$

Cette construction se trouve illustrée par la figure qui suit.

Dans [1] nous avons montré que :

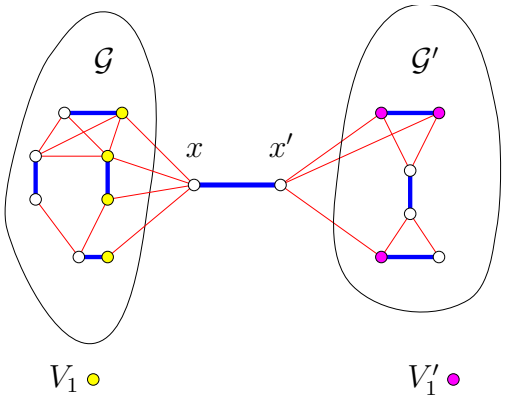
Théorème 3.4 *Étant donné un graphe* R&B *non orienté* $\mathcal{G} = (V; B,R)$*, les trois propriétés suivantes sont équivalentes :*

1. *$\mathcal{G}$ ne contient pas de cycle élémentaire alternant.*

---

[1] Christian Retoré. Perfect matchings and series-parallel graphs: multiplicative proof nets as R&B-graphs. In J.-Y. Girard, M. Okada, and A. Scedrov, editors, *Linear'96*, volume 3 of *Electronic Notes in Theoretical Science*. Elsevier, 1996. (available from http://www.elsevier.nl/).

FIG. 3.1 – *Construction d'un graphe sans cycle élémentaire alternant à partir de deux tels graphes*



2. *B est l'unique couplage parfait du graphe sous-jacent $\mathcal{G} = (V; B \uplus R)$*
3. *$\mathcal{G} \in$ R&B$^+$*

En fait la partie $3 \Leftrightarrow 1$ avait déjà été établie par Anton Kotzig[2,3,4], mais sans donner un algorithme qui calcule l'isthme. C'est l'existence d'un isthme B qui nous servira pour séquentialiser les réseaux usuels, c'est-à-dire la conséquence immédiate suivante :

Corollaire 3.5 *Tout graphe* R&B *sans cycle Æ contient un isthme* B *, qu'on peut trouver en $O(n^2)$ où n est le nombre de sommets du graphe.*

3.4 Cographes R&B

Un **cographe R&B (orienté)** est simplement un graphe R&B $\mathcal{G} = (V; B,R)$ tel que R soit un cographe (orienté) — on omettra parfois l'adjectif « orienté » , cela ne devrait pas causer d'ambiguïté.

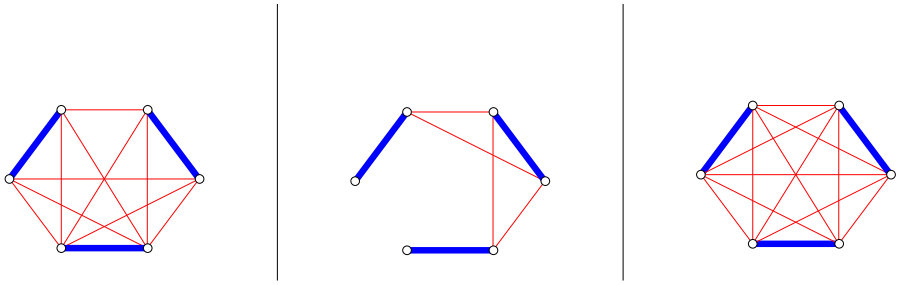
Un cographe R&B est dit **correct** si tout circuit Æ contient une corde ; si de plus il contient un chemin Æ sans corde de tout sommet à tout autre sommet, alors il est dit **strictement correct**. Par exemple en figure 3.4 on trouvera trois

[2] Anton Kotzig. Z teorie konečnỳch grafov s lineàrym faktorom II. *Matematicko-Fyzikàlny Časopis SAV*, IX(3):136 – 159, 1959. (On the theory of finite graphs with a linear factor II).

[3] Béla Bollobás. *Extremal Graph Theory*. Academic Press, 1978.

[4] Làszlò Lovàsz and Michael David Plummer. *Matching Theory*, volume 121 of *Mathematics Studies*. North-Holland, 1986. Annals of Discrete Mathematics 29.

FIG. 3.2 – *Trois cographes* R&B *non orientés: incorrect, correct mais pas strictement correct et strictement correct.*



cographes non orientés : le plus à gauche n'est pas correct, celui du milieu est correct sans être strictement correct, et celui de droite est strictement correct.

3.4.1 Réécriture et cographes R&B

Appelons $K_{2n}^{\mathrm{R\&B}}$ le graphe R&B complet sur $2n$ sommets :

$$K_{2n}^{\mathrm{R\&B}} = (\{x_1,x'_1,\ldots,x_n,x'_n\};\{\{x_1,x'_1\},\ldots,\{x_n,x'_n\}\},\widehat{\otimes}_{1\leq i\leq n}\,(x_i\,\widehat{\wp}\,x'_i)\,)$$

Clairement, tout graphe R&B correct $\mathcal{G} = (V;B,R)$ est inclus dans $K_{2n}^{\mathrm{R\&B}}$ — sinon on aurait un arc $x_ix'_i$ qui, avec l'arête B x'_ix_i constituerait un circuit Æ sans corde, et donc $\widehat{\otimes}_{1\leq i\leq n}(x_i\,\widehat{\wp}\,x_n)$ se réécrit en R. Le paragraphe 3.2.3 nous affirme que tout graphe R&B correct s'obtient par réécriture à partir de $K_{2n}^{\mathrm{R\&B}}$. Néanmoins comment pourrait-on obtenir tous les graphes R&B corrects, ou strictement corrects, *et rien que ceux-là* ? Un système de réécriture a notamment l'avantage de fournir une définition inductive des graphes considérés.

Nous avons totalement répondu à cette question dans le cas de R&B cographes symétriques, c'est-à-dire quand le cographe R ne contient pas de $\widehat{<}$. S'il en contient, nous avons une idée de la solution, la conjecture 3.8 ci-après, ainsi que certains des lemmes y conduisant.

Commençons par remarquer que, parmi les règles de réécriture données en 3.2.3, la règle $(\otimes\wp 4)$ ne préserve pas la correction, et que $(\otimes\wp 2)$ ne préserve pas la stricte correction.

Théorème 3.6 *A l'exception de* $(\otimes\wp 4)$*, toutes les règles de réécriture du paragraphe 3.2.3 préservent la correction des cographes* R&B .

Si le R&B *cographe est symétrique (*$\widehat{<}$ *n'est pas utilisée) alors parmi les règles* $(\otimes\wp)$ *seule la règle de réécriture* $(\otimes\wp 3)$ *préserve la stricte correction.*

3.4.2 Une définition inductive des cographes R&B

On considère ici les cographes R&B où R ne contient pas de $\widehat{<}$ — le graphe est donc symétrique ou non orienté. Le système de réécriture est alors restreint aux règles concernant la composition symétrique en série $\widehat{\otimes}$, et la composition en parallèle $\widehat{\wp}$, c'est-à-dire que seules les règles de réécriture $(\otimes\wp 4)$, $(\otimes\wp 3)$ et $(\otimes\wp 2)$ sont retenues. On a alors le résultat suivant :

Théorème 3.7 *Les cographes* R&B *symétriques corrects à $2n$ sommets sont exactement ceux qui s'obtiennent par les réécritures $(\otimes\wp 3)$ et $(\otimes\wp 2)$ à partir de $K_{2n}^{\mathrm{R\&B}}$.*

Les cographes R&B *symétriques strictement corrects à $2n$ sommets sont exactement ceux qui s'obtiennent par la réécriture $(\otimes\wp 3)$ à partir de $K_{2n}^{\mathrm{R\&B}}$.*

Pour l'instant, nous ne pouvons que conjecturer un résultat analogue pour les cographes R&B orientés :

Conjecture 3.8 *Les cographes* R&B *corrects à $2n$ sommets sont exactement ceux qui s'obtiennent à partir des $K_{2n}^{\mathrm{R\&B}}$ au moyen des règles de réécriture données au paragraphe 3.2.3 à l'exception de $(\otimes\wp 4)$.*

3.5 Critiques et perspectives

3.5.1 Extensions au calculs non commutatifs

Si l'on s'autorise à partitionner les arêtes R , on devrait pouvoir étendre ces résultats aux graphes sous-jacents aux réseaux de Lambek de [1], et aux calculs mixtes dont nous parlerons aux chapitres 6 et 9.

L'idée est que les « $\widehat{\wp}$ » créent aussi des arêtes, et c'est une direction de recherche dont je pense qu'elle sera productive à court terme.

3.5.2 Cographes R&B orientés

Comme on l'aura sans doute compris cette machinerie a été introduite pour le cas orienté qui correspond au calcul ordonné. Néanmoins, malgré l'aide de collègues, nos tentatives d'obtenir une définition inductive des cographes R&B orientés, comme celle énoncée à la conjecture 3.8 sont restées sans succès, tandis que toutes les autres propriétés sont semblables à celles des cographes R&B non orientés.

---

[1] Christian Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.

Chapitre 4

Réseaux de démonstration

Résumé : *Dans ce chapitre nous expliquons ce que sont les réseaux de démonstration, et donnons les résultats obtenus. Une grande partie de notre travail porte sur ce sujet, et ce chapitre est un plus long que les autres. Si ce n'est l'habitude qu'a le logicien d'utiliser le calcul des séquents, la syntaxe des réseaux de démonstration est mathématiquement plus élégante, du moins pour la partie multiplicative de la logique linéaire.*

En fait, le seul aspect utile du calcul des séquents est de fournir une définition inductive des démonstrations. C'est seulement à ce propos que nous présentons les calculs des séquents considérés. Nous avons essayé d'organiser ce chapitre ainsi : plutôt que de faire un paragraphe par calcul logique, comme nombre de résultats sont similaires, nous avons plutôt fait un paragraphe par type de définition ou de résultats.

La première présentation des réseaux qui est donnée est la nôtre, mais, si ce n'est le caractère plus standard des concepts combinatoires employés, elle reste voisine des présentations concurrentes.

Elle permet néanmoins d'aboutir à la seconde, les réseaux abstraits, qui se démarquent nettement de ce qui existe. Les liens, les briques dont sont faits les réseaux ont disparu, et il en résulte que ceux-ci sont des graphes définis par des propriétés standard, et que tous les tels graphes sont des préréseaux.

Ce chapitre est essentiellement de mon fait [1,2,3,4,5,6].

---

[1] Christian Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.

[2] Christian Retoré. Pomset logic: a non-commutative extension of classical linear logic. In Philippe de Groote and James Roger Hindley, editors, *Typed Lambda Calculus and Applications, TLCA'97*, volume 1210 of *LNCS*, pages 300–318, 1997.

[3] Christian Retoré. Perfect matchings and series-parallel graphs: multiplicative proof nets as R&B-graphs. In J.-Y. Girard, M. Okada, and A. Scedrov, editors, *Linear'96*, volume 3 of *Electronic Notes in Theoretical Science*. Elsevier, 1996. (available from http://www.elsevier.nl/).

[4] Christian Retoré. Pomset logic as a calculus of directed cographs. In V. M. Abrusci and C. Casadio, editors, *Dynamic Perspectives in Logic and Linguistics: Proof Theoretical Dimensions of Communication Processes,Proceedings of the 4th Roma Workshop*, pages

4.1 Les langages de la logique linéaire : formules, lois de De Morgan

On se donne un ensemble dénombrable de variables propositionnelles P. Le langage de base de la logique linéaire multiplicative est le suivant :

$$\mathcal{L} ::= P \mid P^{\perp} \mid \mathcal{L} \parr \mathcal{L} \mid \mathcal{L} \otimes \mathcal{L}$$

Notre étude porte également sur la logique linéaire augmentée d'un connecteur multiplicatif non-commutatif « précède » noté « $<$ » .

$$\mathcal{L}^{<} ::= P \mid P^{\perp} \mid \mathcal{L}^{<} \parr \mathcal{L}^{<} \mid \mathcal{L}^{<} \otimes \mathcal{L}^{<} \mid \mathcal{L}^{<} < \mathcal{L}^{<}$$

Pour simplifier l'exposé, on n'autorise pas la négation à porter sur des formules composées : cela ne change rien car en vertu des lois de De Morgan qui suivent, toute formule est équivalente à une formule de ce langage restreint, et, les divers calculs considérés possèdent tous la propriété dite de η-expansion[a], qui affirme qu'un axiome $\vdash A, A^{\perp}$ ou $A \vdash A$ peut se déduire d'axiomes portant sur les formules atomiques.

4.1.1 Lois de De Morgan

La logique linéaire possède une négation semblable à celle de la logique classique, qui valide les lois de De Morgan suivantes :

$$\begin{array}{rcl} (A^{\perp})^{\perp} & \equiv & A \\ (A \parr B)^{\perp} & \equiv & (B^{\perp} \otimes A^{\perp}) \\ (A \otimes B)^{\perp} & \equiv & (B^{\perp} \parr A^{\perp}) \\ (A < B)^{\perp} & \equiv & (A^{\perp} < B^{\perp}) \end{array}$$

On prendra garde à la dernière ligne : $(A < B)^{\perp} \equiv (A^{\perp} < B^{\perp})$

a. Cette propriété est effectivement l'analogue de la η expansion du lambda calcul simplement typé.

221–247. Bulzoni, Roma, 1999. Also available as INRIA Rapport de Recherche RR-3714 http://www.inria.fr/.

[5] Christian Retoré. Handsome proof-nets: R&B-graphs, perfect matchings and series-parallel graphs. Rapport de Recherche RR-3652, INRIA, March 1999. http://www.inria.fr/.

[6] Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 1999. To appear.

4.1.2 Autres connecteurs

On ne voit pas d'implications dans le langage $\mathcal{L}$ donné ci-dessus, et la négation est restreinte aux variables propositionnelles : ces formules sont en effet définissables par les lois de De Morgan.

- La négation d'une formule composée s'exprime en fonction de ses constituants et de leurs négations.
- L'implication est définie tout comme en logique classique, à ceci près que la disjonction n'est pas nécessairement commutative, ce qui permet d'avoir deux implications, notées $\multimap$ et $\circ\!\!-$ (ou $\setminus$ et $/$).

$$(A \multimap B) = (A \setminus B) = (A^{\perp} \parr B)$$

$$(B \circ\!\!- A) = (B \,/\, A) = (B \parr A^{\perp})$$

Les notations $A \setminus B$ et $B \,/\, A$ proviennent du calcul de Lambek, et ces fractions se justifient comme suit : $B \,/\, A, A \vdash B$ et $A, A \setminus B \vdash B$.

Si on le souhaite, on peut considérer le langage étendu suivant sans doute plus courant (et on pourrait même l'augmenter de $<$), dont toutes les formules sont équivalentes via les lois de De Morgan et la définition des implications à une unique formule de $\mathcal{L}$.

$$\mathcal{L}^{++} ::= P \mid \mathcal{L}^{++} \parr \mathcal{L}^{++} \mid \mathcal{L}^{++} \otimes \mathcal{L}^{++} \mid \mathcal{L}^{++} \multimap \mathcal{L}^{++} \mid \mathcal{L}^{++} \circ\!\!- \mathcal{L}^{++}$$

4.1.3 Formules intuitionnistes, polarités

Nous allons considérer ci-après des variantes intuitionnistes de la logique linéaire, qui sont uniquement des restriction de *langage*.

Les formules intuitionnistes se partagent en deux classes disjointes : les formules positives ou sorties (*ouput*) et les formules négatives ou entrées (*input*), qui sont définies par induction mutuelle :

$$\mathcal{L}^{\circ} ::= P \mid \mathcal{L}^{\bullet} \parr \mathcal{L}^{\circ} \mid \mathcal{L}^{\circ} \parr \mathcal{L}^{\bullet} \mid \mathcal{L}^{\circ} \otimes \mathcal{L}^{\circ}$$

$$\mathcal{L}^{\bullet} ::= P^{\perp} \mid \mathcal{L}^{\circ} \otimes \mathcal{L}^{\bullet} \mid \mathcal{L}^{\bullet} \otimes \mathcal{L}^{\circ} \mid \mathcal{L}^{\bullet} \parr \mathcal{L}^{\bullet}$$

Une formule est dans $\mathcal{L}^{\circ}$ si et seulement si sa négation est dans $\mathcal{L}^{\bullet}$, et, réciproquement, et certaines formules ne sont ni dans $\mathcal{L}^{\circ}$ ni dans $\mathcal{L}^{\bullet}$, telles $a \parr a$.

Modulo les lois de De Morgan, une formule positive est équivalente à une formule de $\mathcal{L}^{\mathcal{J}}$ qui est l'ensemble suivant de formules :

$$\mathcal{L}^{\mathcal{J}} ::= P \mid \mathcal{L}^{\mathcal{J}} \multimap \mathcal{L}^{\mathcal{J}} \mid \mathcal{L}^{\mathcal{J}} \circ\!\!- \mathcal{L}^{\mathcal{J}} \mid \mathcal{L}^{\mathcal{J}} \otimes \mathcal{L}^{\mathcal{J}}$$

4.2 Calculs des séquents

Nous donnons ici une version compacte du calcul des séquents de MLL+mix. Le calcul MLL s'obtient en omettant la règle mix. Le calcul MLL, donc sans mix, admet des restrictions de diverses natures, et qui commutent entre elles.

- Non-commutativité : la règle $(E.T.)$ est omise — les connecteurs cessent d'être commutatifs, et il est important que les lois de De Morgan soient : $(A \parr B)^\perp = (B^\perp \otimes A^\perp)$ et $(A \otimes B)^\perp = B^\perp \parr A^\perp$.
- Les calculs intuitionnistes : pour les calculs envisagés ici, il suffit d'imposer que les formules soient intuitionnistes — dans $\mathcal{L}^\circ \cup \mathcal{L}^\bullet$, cf. paragraphe 4.1.3. Il suffit donc d'éviter les *par* de formules positives et les *tenseurs* de formules négatives. On remarque aisément qu'alors il y a exactement une formule positive par séquent démontré, et il existe des calculs bilatères correspondants.
- Exclusion de la séquence vide : on peut imposer que tout séquent comporte au moins deux formules ; il suffit pour cela de n'appliquer la règle $\parr$ que lorsque le séquent prémisse a au moins deux formules.

En présence de $(E.T.)$ une seule des deux règles (cut_1) et (cut_2) suffit, et une seule des deux règles $(\otimes_1)$ et $(\otimes_2)$ suffit — les autres étant dérivables.

$$\left(\frac{\vdash \Gamma \qquad \vdash \Delta}{\vdash \Gamma,\Delta}\, mix\right)$$

$$\frac{\vdash \Gamma,A}{\vdash A,\Gamma}\,(E.C.) \qquad\qquad \frac{\vdash \Gamma,A,B,\Delta}{\vdash \Gamma,B,A,\Delta}\,(E.T.)$$

$$\frac{}{\vdash A,A^\perp}\, ax$$

$$\frac{\vdash \Gamma,A,\Delta \qquad \vdash A^\perp,\Delta'}{\vdash \Gamma,\Delta',\Delta}\, cut_1 \qquad \frac{\vdash \Gamma,A \qquad \vdash \Gamma',A^\perp,\Delta'}{\vdash \Gamma',\Gamma,\Delta'}\, cut_2$$

$$\frac{\vdash \Gamma,A,\Delta \qquad \vdash B,\Delta'}{\vdash \Gamma,A \otimes B,\Delta',\Delta}\, \otimes_1 \qquad \frac{\vdash \Gamma,A \qquad \vdash \Gamma',B,\Delta'}{\vdash \Gamma',\Gamma,A \otimes B,\Delta'}\, \otimes_2$$

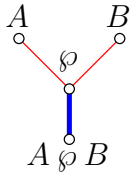
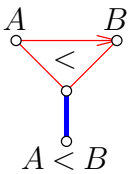
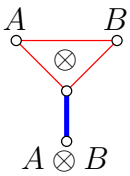
$$\frac{\vdash \Gamma,A,B,\Delta}{\vdash \Gamma,A \parr B,\Delta}\, \parr$$

4.3 Réseaux avec liens

Nous allons définir les préréseaux comme des graphes munis d'un couplage parfait.

4.3.1 Liens, préréseaux

Les briques dont sont faits les (pré)réseaux sont les graphes bicolores suivants, appelés liens et qui ont chacun des prémisses et des conclusions :

Nom	lien axiome	lien *par*	lien *précède*	lien *tenseur*
Prémisses	aucune	A et B	A et B	A et B
graphe R&B				
Conclusions	a et $a^\perp$	$A \wp B$	$A < B$	$A \otimes B$

Le lien axiome peut-être décomposé en une arête B pour a, une arête B pour $a^\perp$, et une arête R joignant les deux.

Ces graphes ne sont pas eux-mêmes des graphes R&B : les arêtes B constituent certes un couplage de ces graphes bicolores, mais il n'est pas parfait.

Une certaine latitude est laissée pour le lien coupure qui n'a pas de conclusion, mais deux prémisses, K et $K^\perp$: soit c'est un lien *tenseur* $K \otimes K^\perp$ signalé comme une coupure, et soit c'est une arête R, liant deux conclusions K et $K^\perp$, de couleur R — le dual d'un axiome. On utilisera l'une ou l'autre représentation de la coupure, chacune étant plus ou moins adapté à certains calculs.

L'axiome peut être restreint aux formules atomiques ou non. C'est sans grande importance, puisque, quel que soit le calcul logique considéré, un axiome $X - X^\perp$ sur une formule complexe X est toujours décomposable en un réseau correct ayant deux conclusions, X et $X^\perp$.

Un préréseau $G = (V; B, R)$ est un graphe bicolore qui est un assemblage de liens. La définition formelle de ces structure est un peu pénible, bien qu'il s'agisse d'objets tout simples. Aussi en donnons nous deux, évidemment équivalentes.

Définition standard des préréseaux

Un graphe R&B Π est un **préréseau** Π s'il existe une famille de liens $\ell = \{\ell_1,\ell_2,\ldots,\ell_k\}$ et morphisme de graphes Σ de $\biguplus \ell_i$ dans Π satisfaisant :

- Σ préserve les noms des sommets.
- Tout sommet de Π a un ou deux antécédent par Σ. S'il en a un seul alors c'est nécessairement la conclusion de l'un des ℓ_i, et s'il en a deux, l'un est la conclusion de ℓ_k et l'autre est la prémisse de ℓ_h avec $k \neq h$
- Tout arc de Π a un unique antécédent dans ℓ, et il est de même couleur.

Les **conclusions** du préréseau sont les extrémités des arêtes B qui n'ont qu'un seul antécédent par Σ.

On considérera aussi des **préréseaux enrichis**, qui s'obtiennent par ajout d'un cographe dont les sommets sont les conclusions du préréseau et les coupures, si ces dernières sont représentées par des conclusions de la forme $X \otimes X^{\perp}$. Les préréseaux enrichis sont en fait une étape intermédiaire vers les préréseaux abstraits : un préréseau abstrait est un préréseau enrichi dont les seuls liens sont des liens axiomes. En d'autres termes, ce sont les cographes R&B du chapitre 3.

On trouvera un exemple de réseau en figure 4.1.

Définition avec les arbres de sous formules

Soit F une formule ; on dit qu'un graphe R&B T est un arbre R&B de F si :

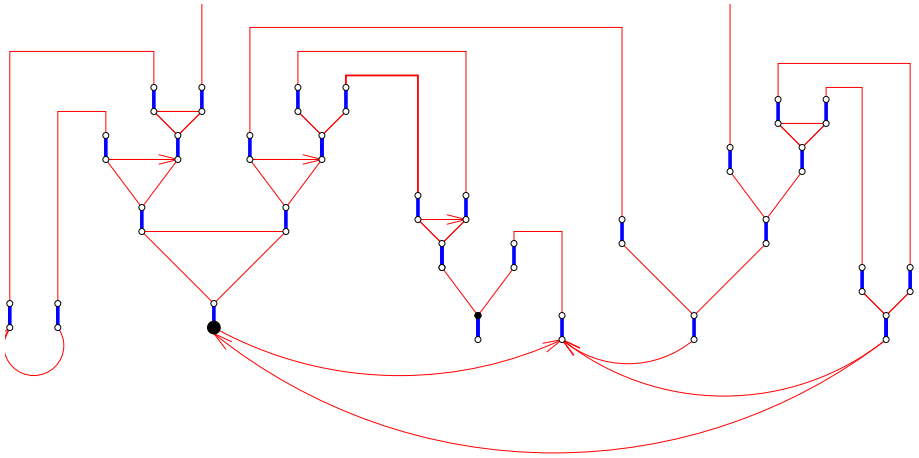
1. Pour toute formule F, le graphe réduit à un sommet nommé F est un arbre R&B de F. Sa racine et son unique feuille sont ce sommet F.
2. Pour toute formule *atomique* a, le graphe réduit à un sommet nommé a est un arbre R&B de F. Sa racine et son unique feuille sont ce sommet a.
3. Si T_1 et T_2 sont des arbres R&B de F_1 et F_2 alors alors le graphe R&B obtenu par ajout d'un lien $\diamond$ ayant pour prémisses les racines des arbres T_1 et T_2 est un arbre R&B de la formule$F_1 \diamond F_2$, où $\diamond$ désigne l'un des connecteurs.

Si la clause 1 n'est pas utilisée, alors une formule n'a qu'un seul arbre R&B et celui-ci est appelé l'arbre R&B des sous formules de F. On remarquera que le graphe sous-jacent d'un arbre R&B n'est pas un arbre, mais nous utilisons ce nom car il s'apparente à l'arbre des sous formules, d'où son nom.

Un préréseau de conclusions $F_1,\ldots,F_n$ et de coupures $K_1 \perp K_1^{\perp},\ldots,K_p \perp K_p$ est un graphe R&B , s'il est composé

1. d'un arbre R&B de chaque F_i et de chaque formule K_j et $K_j^{\perp}$, dont les sommets sont les seuls sommets du préréseau ;
2. d'une famille d'arêtes B non adjacentes entre feuilles d'arbres R&B de sorte que toute feuille soit incidente à un (et un seul de ces arcs), et que les noms des extrémités soient duaux l'un de l'autre ;

FIG. 4.1 – *Un exemple de réseau enrichi*



3. et d'une arête R entre chaque K_j et $K_j^\perp$ — ou un lien *tenseur* de prémisses K_j et $K_j^\perp$ pour tout j, signalé comme un lien coupure.

S'il s'agit d'un préréseau enrichi, on peut ajouter un cographe orienté dont les sommets sont les conclusions et les coupures décrites comme des liens *tenseurs*.

On trouvera un exemple de réseau en figure 4.1.

4.3.2 Réseaux de MLL et de MLL+mix

Les préréseaux de ces deux calculs ne font pas intervenir le lien « précède » : le langage est $\mathcal{L}$. Un préréseau est appelé un **réseau de MLL+mix** ou est dit **correct** s'il ne contient pas de cycle Æ (ou pas de circuit Æ , puisque ces graphes sont symétriques). Un préréseau est appelé un **réseau de MLL** ou est dit **strictement correct** s'il ne contient pas de cycle Æ .

Théorème 4.1 *Tout réseau de MLL+mix correspond à au moins une démonstration du calcul des séquent MLL+mix.*

Tout réseau de MLL correspond à au moins une démonstration de MLL.

Il y a diverses démonstrations de ce résultat, et de résultats analogues mais toutes reposent sur le théorème 3.4. Remarquons que la première partie du théorème est la plus importante, et pour ainsi dire, suffit. En effet, en vertu de résul-

tats obtenus en collaboration avec Arnaud Fleury pendant ma thèse [1], il est aisé de voir qu'*une* séquentialisation d'un réseau utilise n fois mix si et seulement si *toutes* les séquentialisations de ce réseau utilisent n fois la règle *mix*. On peut alors montrer qu'un réseau correct n'utilise pas *mix* si et seulement s'il est strictement correct.

Il reste donc à établir la première partie du théorème. On peut utiliser directement le théorème 3.4 pour montrer que le réseau contient un isthme « interne » . Par hypothèse d'induction, on peut alors trouver des séquentialisations correspondant aux deux morceaux de réseau que l'isthme relie. Il est alors aisé d'obtenir une séquentialisation du réseau à partir des séquentialisations correspondant aux parties du réseau situées de part et d'autre de l'isthme.

On peut aussi associer un autre graphe R&B à un préréseau, qui sera sans cycle Æ si et seulement si le préréseau est un réseau, et dont les arêtes B correspondent aux tenseurs. Le théorème 3.4 permet alors de trouver un isthme B , qui n'est autre qu'un *tenseur scindant* au sens de Jean-Yves Girard [2]. Cette propriété du tenseur scindant suffit à séquentialiser le réseau, c'est d'ailleurs ainsi que fonctionne la première démonstration de la séquentialisation de Jean-Yves Girard [2]. Mais ici cela fonctionne même avec mix, pour lequel on obtient en prime une petite propriété : dans un réseau avec *mix*, il y a un chemin Æ de n'importe quel point à un tenseur scindant.

On peut aussi associer au préréseau un autre graphe R&B , dont les arêtes B correspondent cette fois aux liens *par*. S'il s'agit d'un réseau, alors le graphe R&B obtenu est sans cycle Æ , et par le théorème 3.4 il contient un isthme B . Un tel isthme correspond à une *section* au sens de Vincent Danos [3] et permet de séquentialiser le réseau.

On peut aussi montrer que ces réseaux correspondent à ceux de Vincent Danos et Laurent Regnier, et que les critères se correspondent. Mais il est plaisant de constater que ces démonstrations apparemment indépendantes peuvent se voir comme les corollaires d'un même résultat à l'énoncé très simple sur les graphes munis d'un unique couplage parfait, le théorème 3.4, qui de plus traite aussi bien du calcul avec ou sans *mix*.

Le théorème suivant découle immédiatement de ce qui précède, une fois vérifié la préservation du critère par chaque étape d'élimination des coupures, ce qui est un peu laborieux, et n'a rien d'original :

Théorème 4.2 *Les réseaux de MLL et de MLL+mix satisfont l'élimination des*

---

[1] Arnaud Fleury and Christian Retoré. The mix rule. *Mathematical Structures in Computer Science*, 4(2):273–285, June 1994.

[2] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[3] Vincent Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation et principalement du λ-calcul*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, juin 1990.

coupures, qui est est une réécriture fortement normalisante et confluente.

4.3.3 Réseaux normaux pour MLL non commutative et pour le calcul de Lambek

Nous allons ici considérer diverses variantes de la logique linéaire multiplicative, qui s'obtiennent par les restrictions suivantes, que l'on peut cumuler, et qui permutent entre elles :

NC On peut souhaiter un calcul non commutatif, ce qui se fait en supprimant la règle d'échange par transposition (E.T.).

INTUI On peut souhaiter un calcul intuitionniste, ce qui s'obtient en contraignant les formules à être intuitionnistes, c'est-à-dire à être positives ou négatives au sens du paragraphe 4.1.3.

$\not{\varepsilon}$ On peut aussi souhaiter n'avoir aucune tautologie, c'est-à-dire contraindre tout séquent à avoir au moins deux conclusions.

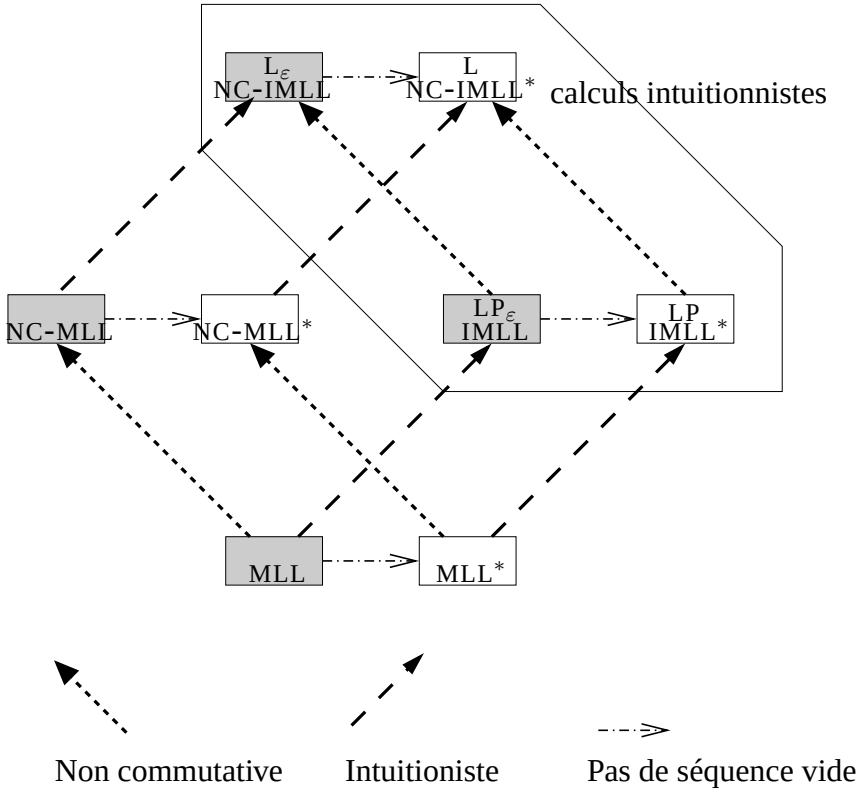
Certaines combinaisons de ces restrictions sont des calculs connus :

– Le *calcul de Lambek* correspond à MLL restreinte des trois manières susmentionnées.
– La première restriction seule conduit à la *logique linéaire cyclique* de David Yetter [4], CyMLL ou NC-MLL.
– La seconde restriction seule nous amène à la *logique linéaire intuitionniste*, IMLL, ou calcul de Lambek avec permutation et séquence vide LP, notamment utilisée pour modéliser les réseaux de Petri.

[4] David N. Yetter. Quantales and (non-commutative) linear logic. *Journal of Symbolic Logic*, 55:41–64, 1990.

On peut récapituler les variantes suivant les trois restrictions considérées ci-dessus, les variantes intuitionnistes ayant également un nom inspiré du calcul de Lambek L.

INTUI	NC	$\not\varepsilon$	nom linéaire	nom catégoriel
oui	oui	oui	NC-IMLL*	L
oui	oui	non	NC-IMLL	L_ε
oui	non	oui	IMLL*	LP
oui	non	non	IMLL	LP_ε
non	oui	oui	NC-MLL*	
non	oui	non	NC-MLL	
non	non	oui	MLL*	
non	non	non	MLL	
			unilatère	bilatère



Se restreindre aux formules intuitionnistes revient à ne jamais utiliser la règle *par* entre deux formules positives, ni de *tenseur* entre deux formules négatives. Les règles unilatères données ci-dessus se transforment alors une à une en règles bilatères de la logique intuitionniste. Par exemple :

$$\frac{\vdash \Gamma, A^{\circ}, B^{\bullet}, \Delta}{\vdash \Gamma, (A^{\circ} \wp B^{\bullet})^{\bullet}, \Delta} \qquad \frac{A^{\perp}, \Gamma^{\perp}, \Delta^{\perp} \vdash B}{\Gamma^{\perp}, \Delta^{\perp} \vdash A^{\perp} \setminus B \equiv A \wp B}$$

Cette restriction, aisément formulée dans les réseaux appelle une remarque (non publiée) : s'il y a des coupures, ils suffit que les *conclusions* soient intuitionnistes pour que le réseau le soit, il n'y a rien à demander pour les formules coupées. Cela se montre sans peine par induction sur le réseau en considérant un tenseur héréditairement scindant [1]. Le premier travail sur les polarités dans MLL est sans doute celui de Jacques van de Wiele (cité dans [2]), qu'on peut résumer ainsi :

Proposition 4.3 (van de Wiele) *Si une démonstration d ne contient que des formules intuitionnistes alors une seule formule de chaque séquent de d est positive. Si un réseau normal ne contient que des conclusions intuitionnistes alors il est séquentialisable en logique linéaire intuitionniste.*

La restriction qui correspond à l'absence d'échange par transposition est bien connue, du moins pour les réseaux normaux dans la communauté des grammaires catégorielles : les axiomes du réseau doivent constituer un bon parenthésage. Mais si la nécessité de ce critère apparaît dans [3] nous avons peut-être (car ce n'est pas bien dur) été les premiers à montrer que cette condition suffisait, et en tout cas à publier ce résultat [4].

Finalement, l'exclusion de la séquence vide dans le calcul de Lambek, qui surprend le logicien mais est nécessaire aux applications grammaticales, ne pose pas de grandes difficultés. En effet nous avons montré qu'*une* séquentialisation ne comporte pas de séquence vide si et seulement si *aucune* n'en comporte. Cela vaut pour tous les calculs multiplicatifs, même si cette restriction n'a de sens intuitif que pour le calcul de Lambek. On peut donc donner la caractérisation des réseaux normaux du calcul de Lambek. [4]

---

[1] Christian Retoré. *Réseaux et Séquents Ordonnés*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, février 1993.

[2] Gianluigi Bellin and P. J. Scott. On the π-calculus and linear logic (with an introduction by S. Abramsky). Technical report, University of Edimbourgh, 1992.

[3] Dirk Roorda. *Resource logic: proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam, 1991.

[4] Christian Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.

Théorème 4.4 *Un réseau normal de MLL est un réseau de CyMLL si et seulement si les axiomes définissent un bon parenthésage des atomes.*

Un réseau de MLL est un réseau de IMLL si et seulement si ses conclusions sont des formules intuitionnistes.

Un réseau est sans séquence vide si et seulement si il ne contient pas de sous-préréseau à une seule conclusion.

Un réseau normal correspond à une démonstration du calcul de Lambek si et seulement s'il satisfait les trois conditions précédentes.

Et qu'en est-il des réseaux avec coupures? On peut adapter ce qui précède, au prix de quelques complications. Les coupures peuvent être enfermées dans une face du réseau qui est planaire. Il faut alors trouver une place dans la suite de formules où noter les formules coupées, et une place canonique car on ne souhaite pas qu'une même démonstration du calcul des séquents correspondent à plusieurs réseaux. Une telle place existe, c'est l'unique symbole *par* qui est dans la face de la coupure enfermée. Il faut alors montrer que l'élimination des coupures préserve ce critère, et calculer les faces des coupures après réduction. Cela se fait également. Mais ce travail ne présente pas grand intérêt car Michele Abrusci et Elena Maringelli ont trouvé, en exploitant pleinement les particularités de nos réseaux bicolores, un critère simple qui fonctionne pour les réseaux non commutatifs avec coupures [1,2], qui simplifie beaucoup l'approche précédente [3].

4.3.4 Le calcul ordonné

Un préréseau ordonné est dit un réseau ordonné s'il ne contient pas de circuit élémentaire alternant. Dans un formalisme moins élégant, notre thèse montrait que ce critère est préservé par élimination des coupures, et que celle ci est une réduction fortement normalisante et confluente. Cela a été publié avec cette description des réseaux dans [4] mais nous n'en parlerons pas plus car c'est moins intéressant que dans les réseaux abstraits ordonnés du paragraphe suivant.

---

[1] Elena Maringelli. *Grafi e logica lineare: una nuova definizione delle reti dimostrative non commutative*. Tesi di laurea, Università di Bari, December 1996. (Graphs and linear logic: a new definition of non-commutative proof-nets.).

[2] Vito Michele Abrusci and Elena Maringelli. A new correctness criterion for cyclic multiplicative proof-nets. In Retoré [Ret98].

[3] Vito Michele Abrusci. Non- commutative proof nets. In Girard et al. [GLR95], pages 271–296.

[4] Christian Retoré. Pomset logic: a non-commutative extension of classical linear logic. In Philippe de Groote and James Roger Hindley, editors, *Typed Lambda Calculus and Applications, TLCA'97*, volume 1210 of *LNCS*, pages 300–318, 1997.

4.4 Réseaux abstraits

On donne ici une description des réseaux de démonstration qui identifie le plus possible de démonstrations. En particulier les propriétés algébriques des connecteurs logiques (associativité, commutativité) sont interprétées par l'égalité. Les disjonctions finales entre les conclusions du réseau sont aussi interprétées par l'égalité.

Pour des raisons de sémantique dénotationnelle, on sait qu'on ne peut identifier davantage de démonstrations sans que le système identifie toutes les démonstrations. Ce sont donc des réseaux qui conduisent l'idée fondatrice, l'identification des démonstrations équivalentes, le plus loin possible.

Pour notre part le principal intérêt de cette description est de faire des (pré)réseaux des graphes standard. Comme on l'a vu précédemment un (pré)réseau est un empilement de petits graphes qui a un sens d'un point de vue logique mais qui n'a pas de propriété caractéristique qui lui donnerait un statut en tant que graphe, c'est-à-dire indépendamment du sens logique qu'on lui accorde. Par contre tout graphe R&B est un préréseau abstrait — et le critère pour reconnaître les réseaux reste une propriété simple.

Nous pensons être ainsi arrivés à une telle description où le logicien retrouve ses objets familiers, mais qui ne déroute pas un théoricien des graphes. Cette description plus mathématique permet de plus d'établir des résultats qui sans elle aurait été trop compliqués à formuler ou à démontrer, en particulier pour le calcul ordonné.

Quand c'est possible, nous traitons simultanément les préréseaux de MLL, MLL+mix et du calcul ordonné, tirant parti des rapports simples existant entre ces calculs : un réseau du calcul ordonné sans « précède » est un réseau de MLL+mix, et un réseau de MLL, c'est un réseau de MLL+mix satisfaisant une condition supplémentaire.

4.4.1 Définitions

Un **préréseau abstrait** est un cographe R&B , c'est-à-dire un graphe muni d'un couplage parfait dont le complémentaire est un cographe possiblement orienté.

Si le cographe n'est pas orienté, c'est-à-dire s'il n'utilise pas la composition $\widehat{<}$ alors le cographe R&B sera dit symétrique.

Il reste à identifier les conclusions d'un préréseau abstrait. Décrivons, comme au chapitre 3, un cographe R&B $\mathcal{G}$ par un triplet $(V; B,R)$. Comme R est un cographe, il existe des cotermes sur V représentant R. En supprimant les « $\widehat{\ }$ » au dessus des connecteurs on obtient la conclusion du cographe R&B , mais on peut aussi étant donnée une partition des composantes connexes de R dire que chaque classe correspond à une conclusion, cela ne fait aucune différence, en particulier si

$R = R_1 \mathbin{\widehat{\wp}} \cdots \mathbin{\widehat{\wp}} R_k$ on peut bien sûr décider que c'est un préréseau de conclusions $R_1, \ldots, R_k$. Suivant le coterme choisi on obtient toutes les variantes de la conclusion qui sont équivalentes modulo les propriétés algébriques des connecteurs.

Quant aux coupures, elles sont des conclusions de la forme $S \mathbin{\widehat{\otimes}} S^{\perp}$ marquées comme telles — où $(\ldots)^{\perp}$ est définie comme au paragraphe 3.2.1.

Un préréseau abstrait est un **réseau abstrait** si tout circuit Æ contient une corde, c'est-à-dire s'il est un cographe R&B correct au sens du paragraphe 3.4.

En l'absence de la composition $\widehat{<}$, on parlera aussi de réseaux abstraits **strictement corrects** lorsqu'ils sont corrects et qu'il existe un chemin élémentaire alternant sans corde entre toute paire de sommets.

4.4.2 Des préréseaux aux graphes R&B via les préréseaux enrichis

Lorsque nous avons défini les préréseaux, nous avons envisagé que les conclusions soient munies d'un cographe. Nous allons exploiter cette possibilité pour transformer petit à petit un préréseau avec liens en un cographe R&B . Au vu de la structure des liens, le critère de correction pour les préréseaux avec liens peut également se formuler par « tout circuit Æ contient une corde ». En effet, dans un réseau avec liens, un circuit Æ ne peut contenir de corde : cela ferait une composante connexe de R à quatre points, ce qui n'est pas possible vue la structure des liens.

Si $\widehat{\diamond}$ désigne l'une des compositions $\widehat{<}$, $\widehat{\otimes}$, $\widehat{\wp}$ deux sommets x et y sont dits **$\widehat{\diamond}$-jumeaux** dans un cographe orienté R s'il existe un coterme t décrivant R de la forme $s[x \diamond y]$. Notons que tout cographe orienté contient au moins une paire de sommets $\widehat{\diamond}$-jumeaux, pour une opération $\widehat{\diamond}$, et qui si le cographe ne contient pas d'arcs, $\widehat{\diamond}$ n'est pas $\widehat{<}$.

Définissons maintenant une transformation entre préréseaux enrichis. Pour traiter simultanément tous les cas, soit $\diamond$ l'un des connecteurs, et $\widehat{\diamond}$ la composition de cographes lui correspondant. La transformation (réversible) consiste à remplacer un lien $\diamond$ final par deux prémisses $\widehat{\diamond}$-jumelles, qui prennent la place de la conclusion du lien dans le cographe entre conclusions. Réciproquement, il s'agit de placer un lien $\diamond$ entre deux conclusions $\widehat{\diamond}$-jumelles du cographes entre conclusions. Cette transformation et son inverse préservent la correction du préréseau. Plus formellement :

Théorème 4.5 *Soient*

$$\left|\begin{array}{lcccccc} \Pi_{\diamond} & = (& X \diamond Y & ,A_1,\ldots,A_n & ; & R[X \diamond Y] &) \\ \Pi_{\widehat{\diamond}} & = (& X,Y & ,A_1,\ldots,A_n & ; & R[X \mathbin{\widehat{\diamond}} Y] &) \end{array}\right.$$

$\Pi_{\diamond}$ est correct si et seulement si $\Pi_{\widehat{\diamond}}$ est correct.

En l'absence de composition $\widehat{<}$ et du connecteur $<$ on a aussi : $\Pi_\diamond$ est strictement correct si et seulement si $\Pi_{\widehat{\diamond}}$ est strictement correct.

On en déduit aisément le résultat suivant :

Théorème 4.6 *Tout cographe* R&B *symétrique correct correspond à une démonstration dans MLL+mix.*

Tout cographe R&B *symétrique strictement correct correspond à une démonstration dans MLL.*

Montrons ici comment l'on passe d'un réseau avec lien à un réseau abstrait:

4.4.3 Élimination des coupures

La preuve que l'élimination des coupures préserve la correction des réseaux est souvent laborieuse, mais, pour les réseaux abstrait, elle devient limpide: c'est encore une conséquence du théorème 3.6.

Théorème 4.7 *Soit $C = \phi \mathbin{\widehat{\otimes}} \phi^\perp$ une coupure d'un cographe* R&B

$$\mathcal{G} = (V; B, S \mathbin{\widehat{\wp}} (\phi \mathbin{\widehat{\otimes}} \phi^\perp))$$

Une étape d'élimination des coupures se définit comme suit, selon la nature de $\phi \mathbin{\widehat{\otimes}} \phi^\perp$:

axiome *si $\phi = x$ et $\phi^\perp = x^\perp$ (ϕ et $\phi^\perp$ sont réduit à deux sommets duaux) :*

$$\mathcal{G} \quad = (\quad V \uplus \{x, x^\perp\} \quad ; \quad B \uplus \{\underline{x}^\perp, x, x^\perp \underline{x}\} \quad , \quad S \mathbin{\widehat{\wp}} (x \mathbin{\widehat{\otimes}} x^\perp) \quad)$$

se réduit en

$$\mathcal{G}' \quad = (\qquad V \qquad ; \quad B \uplus \{\{\underline{x}^\perp, \underline{x}\}\} \quad , \qquad S \qquad)$$

Le sommet $\underline{x}$ (resp. $\underline{x}^\perp$) désigne l'unique sommet de $\mathcal{G}$ qui est B *-adjacent à $x^\perp$ (resp. x) et nommé x (resp. $x^\perp$). On a écrit ab à la place de $(a,b),(b,a)$ ce qui est cohérent : les arcs* B *sont toujours symétriques.*

tenseur / par *si $\phi = \phi_1 \mathbin{\widehat{\otimes}} \phi_2$ alors*

$$\mathcal{G} \quad = (\quad V \quad ; \quad B \quad , \quad S \mathbin{\widehat{\wp}} (\, (\phi_1 \mathbin{\widehat{\otimes}} \phi_2) \mathbin{\widehat{\otimes}} (\phi_2^\perp \mathbin{\widehat{\wp}} \phi_1^\perp \,)) \quad)$$

se réduit en

$$\mathcal{G}' \quad = (\quad V \quad ; \quad B \quad , \quad S \mathbin{\widehat{\wp}} (\, (\phi_1 \mathbin{\widehat{\otimes}} \phi_1^\perp) \mathbin{\widehat{\wp}} (\phi_2 \mathbin{\widehat{\otimes}} \phi_2^\perp) \,) \quad)$$

précède / précède *si $\phi = \phi_1 \mathbin{\widehat{<}} \phi_2$ alors*

$$\mathcal{G} \quad = (\quad V \quad ; \quad B \quad , \quad S \mathbin{\widehat{\wp}} (\, (\phi_1 \mathbin{\widehat{<}} \phi_2) \mathbin{\widehat{\otimes}} (\phi_1^\perp \mathbin{\widehat{<}} \phi_2^\perp) \,) \quad)$$

se réduit en

$$\mathcal{G}' \quad = (\quad V \quad ; \quad B \quad , \quad S \mathbin{\widehat{\wp}} (\, (\phi_1 \mathbin{\widehat{\otimes}} \phi_1^\perp) \mathbin{\widehat{<}} (\phi_2 \mathbin{\widehat{\otimes}} \phi_2^\perp) \,) \quad)$$

Chaque étape de réduction préserve la correction, c'est-à-dire l'absence de circuit Æ sans corde. Ces étapes de réduction définissent une réduction fortement normalisante et confluente.

Pour convaincre de l'intérêt de l'approche, nous allons donner la démonstration où $\multimap\!\bullet$ désigne la réécriture de cographe définie au paragraphe 3.2.3.

axiome En notant $B(u)$ l'unique sommet adjacent par une arête B à u, la réduction axiome consiste à remplacer une séquence de trois arêtes

$$B(x^\perp)x^\perp \in B \quad x^\perp x \in R \quad xB(x) \in B$$

avec x et $x^\perp$ non R -adjacents à d'autres sommets, par $B(x^\perp)B(x) \in B$. Ceci ne crée pas de nouveau circuit Æ , tandis que l'arc R $xx^\perp$ ne peut être une corde d'un circuit Æ (le degré R de ces points est 1).

tenseur contre par remarquons que

$$\begin{array}{rcl} S\,\widehat{\wp}\,((\phi_1\,\widehat{\otimes}\,\phi_2)\,\widehat{\otimes}\,(\phi_1^\perp\,\wp\,\phi_2^\perp)) & = & S\,\widehat{\wp}\,(\phi_2\,\widehat{\otimes}\,(\phi_1\,\widehat{\otimes}\,(\phi_1^\perp\,\wp\,\phi_2^\perp)) \\ & \multimap\!\bullet & S\,\widehat{\wp}\,(\phi_2\,\widehat{\otimes}\,((\phi_1\,\widehat{\otimes}\,\phi_1^\perp)\,\wp\,\phi_2^\perp)) \\ & \multimap\!\bullet & S\,\widehat{\wp}\,((\phi_1\,\widehat{\otimes}\,\phi_1^\perp)\,\wp\,(\phi_2\,\widehat{\otimes}\,\phi_2^\perp)) \end{array}$$

où les deux réécritures $\multimap\!\bullet$ sont deux applications de $(\otimes\wp 3)$. Le théorème 3.6 montre que Π' est correct (resp. strictement correct) lorsque Π l'est.

précède contre précède Remarquons que

$$S\,\widehat{\wp}\,((\phi_1\,\widehat{<}\,\phi_2)\,\widehat{\otimes}\,(\phi_1^\perp\,\widehat{<}\,\phi_2^\perp)) \quad \multimap\!\bullet \quad S\,\widehat{\wp}\,((\phi_1\,\widehat{\otimes}\,\phi_1^\perp)\,\widehat{<}\,(\phi_2\,\widehat{\otimes}\,\phi_2^\perp))$$

où la réécriture $\multimap\!\bullet$ est une une application de $(\otimes<4)$. Le théorème 3.6 montre que Π' est correct lorsque Π l'est.

La réduction termine car le nombre d'arcs R décroît à chaque étape. La confluence provient du fait que deux configurations à réduire sont toujours disjointes.

4.4.4 Axiomatisation par réécriture des calculs multiplicatifs

Comme on l'a vu au théorème 3.6 du chapitre 3, la correction des cographes orientés est préservée par les règles de réécriture axiomatisant l'inclusion de cographes, à l'exception de $(\otimes\wp 4)$. Dans le cas des cographes R&B symétriques la stricte correction est uniquement préservée par $(\otimes\wp 3)$.

La réciproque vaut aussi pour les cographes R&B symétriques, dans lesquels se traitent les calculs MLL et MLL+mix : tout cographe R&B symétrique dont chaque circuit Æ contient une corde, c'est-à-dire tout réseau abstrait, s'obtient à

partir du R&B graphe complet $K_{2n}^{\mathrm{R\&B}}$ sur $2n$ sommets par les réécritures $(\otimes\wp 3)$ et $(\otimes\wp 2)$. De plus, s'il est strictement correct la réécriture $(\otimes\wp 2)$ n'est pas utilisée.

Théorème 4.8 *Appelons axiomes les formules*

$$\bigotimes_{x\in V}(x \parr x^{\perp})$$

où V est un multi-ensemble de variables propositionnelles.

Les tautologies de MLL sont toutes les formules obtenues par la réécriture $(\otimes\wp 3)$ à partir d'un axiome.

Les tautologies de MLL+mix sont toutes les formules obtenues par les réécritures $(\otimes\wp 3)$ et $(\otimes\wp 2)$ à partir d'un axiome.

4.5 Critiques et perspectives

4.5.1 Séquentialisation du calcul ordonné

En fait, le matériel développé dans ce chapitre et le chapitre précédent avait pour principale ambition de trouver un calcul des séquents complets pour les réseaux ordonnés.

Malgré notre insistance et l'aide de quelques collègues, cette question n'est toujours pas résolue. L'axiomatique par réécriture reste la piste la plus prometteuse. Le lecteur pourra se demander pourquoi ne pas changer les réseaux de démonstration plutôt que de chercher en vain un calcul des séquents. La raison est simple : ce calcul en réseaux satisfait l'élimination des coupures et surtout suit de très près la sémantique cohérente, comme nous le verrons au chapitre suivant.

Il s'agit de montrer que les cographes R&B orientés, qui ne comportent pas de circuits Æ sans corde s'obtiennent tous à partir de $K_{2n}^{\mathrm{R\&B}}$ par les règles de réécritures du paragraphe 3.2.3 sauf $(\otimes\wp 4)$. Si tel est le cas, alors on a une axiomatique à la Hilbert du calcul ordonné, ou, si on préfère, un calcul des séquents complet. Celui-ci manipule des séquents dont les conclusions sont munies d'un cographe, noté par un coterme. Il admet, c'est sans doute un progrès par rapport à des tentatives précédentes, une formulation bilatère. Celle-ci pourra être très utile dans les applications linguistiques où pour l'instant seuls des réseaux d'allure intuitionniste sont utilisés — je dis d'allure intuitionniste car on ne connaît pas encore de système de polarités pour le connecteur « précède ». Voici donc un calcul des séquents dont on espère qu'il recouvre tous les cographes R&B dont tous les circuits Æ contiennent une corde. En tout cas :

Théorème 4.9 *Toute démonstration du calcul des séquents MLL< se traduit par un réseau correct, et possède donc une interprétation en sémantique cohérente.*

La première assertion découle de ce que les réécritures de cographe, qui introduisent les connecteurs implicitement, comme un cographe entre conclusions préservent la correction et de la validité immédiate des règles d'introduction des connecteurs — qui sont interprétées par l'égalité. La seconde résulte de la correspondance entre les réseaux avec liens et les réseaux abstraits : nous avions déjà établi que les réseaux avec liens étaient interprétables dans la sémantique cohérente.

Une autre direction serait de suivre les récents travaux d'Alessio Guglielmi sur les calculs de structures, qui ont fait suite à nos travaux sur les cographes. Il a considérablement enrichi la notion de calcul des séquents afin de rendre compte de calculs de processus, et son approche offre de nouvelles perspectives [1,2].

4.5.2 Réseaux non commutatifs abstraits

Nous souhaitons obtenir une formulation abstraite des réseaux pour le calcul de Lambek, ainsi que pour les calculs mixtes à la Philippe de Groote dont nous parlerons aux chapitres 6 et 9. Il faut pour cela faire apparaître des arcs correspondants aux liens *par*, qui soient d'une couleur différentes de ceux qu'on associent aux liens *tenseur*. Les structures ainsi définies devraient correspondent au déploiement des réseaux non commutatifs avec liens de Michele Abrusci et Elena Maringelli [3].

---

[1] Alessio Guglielmi. A calculus of order and interaction. Technical Report WV-99-04, Dresden University of Technology, 1999.

[2] Alessio Guglielmi and Lutz Straßburger. Non-commutativity and MELL in the calculus of structures. In L. Fribourg, editor, *CSL 2001*, volume 2142 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, 2001.

[3] Vito Michele Abrusci and Elena Maringelli. A new correctness criterion for cyclic multiplicative proof-nets. In Retoré [Ret98].

Chapitre 5

Sémantique dénotationnelle

Résumé : *Dans ce chapitre nous présentons quelques travaux de sémantique dénotationnelle dans le cadre des espaces cohérents, introduits par Jean-Yves Girard* [1]*, et dont est issue la logique linéaire* [2]*.*

Un premier travail [3,4] *montre qu'un préréseau est un réseau si et seulement si lorsqu'on l'interprète on obtient un objet qui a un sens dans la sémantique des espaces cohérents.*

Un second travail [5,6] *montre que les espaces cohérents admettent une modalité autoduale* $\neg$ *qui rend isomorphes les espaces* $\neg A < \neg A$ *et* $\neg A$ *où* $<$ *est le connecteur « précède ». Comme ces morphismes permettent d'envisager la contraction d'hypothèses et de conclusions, cette modalité, initialement conçue pour le codage de la logique classique sans affaiblissement a aussi sans doute son utilité dans les récents travaux d'Alessio Guglielmi sur les calculs de processus* [7]*.*

---

[1] Jean-Yves Girard. The system F of variable types: Fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[2] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[3] Christian Retoré. On the relation between coherence semantics and multiplicative proof nets. Rapport de Recherche RR-2430, INRIA, décembre 1994. http://www.inria.fr/.

[4] Christian Retoré. A semantic characterisation of the correctness of a proof net. *Mathematical Structures in Computer Science*, 7(5):445–452, 1997.

[5] Christian Retoré. Une modalité autoduale pour le connecteur “précède”. In Pierre Ageron, editor, *Catégories, Algèbres, Esquisses et Néo-Esquisses*, Publications du Département de Mathématiques, Université de Caen, pages 11–16, September 1994.

[6] Christian Retoré. A self-dual modality for "before" in the category of coherence spaces and in the category of hypercoherences. Rapport de Recherche RR-2432, INRIA, décembre 1994. http://www.inria.fr/.

5.1 Présentation

Comme on l'a dit au début du document, la logique linéaire est plutôt une logique « absolue » au sens de [1,2]. Sans être aussi interne que les sémantiques des jeux récemment proposées [3,4,5], la sémantique dénotationnelle des espaces cohérents [6,7,8] qui interprète les démonstrations, suit de très près la logique linéaire. Cette dernière est d'ailleurs issue des espaces cohérents comme on le voit dans l'article original [7] qui, originellement, avaient été conçus comme modèles du lambda calcul typé du second ordre [6].

Quelle peut être l'utilité de tels modèles ? Ils assurent que les démonstrations peuvent se voir comme des programmes typés, et permettent d'étudier la normalisation des démonstrations, c'est-à-dire l'évaluation de programmes fonctionnels. Lorsqu'on souhaite enrichir un calcul, ils sont alors un garde-fou : si le modèle dénotationnel contient une certaine construction, alors il est raisonnable de chercher à l'inclure dans la syntaxe. Techniquement, le modèle fournit des informations sur le comportement syntaxique de ces constructions. Le calcul ordonné[9] abordé dans les précédents chapitres vient d'une telle remarque de Jean-Yves Girard, et la syntaxe en réseaux pour ce connecteur est issue de sa sémantique cohérente.

Le premier résultat exposé ici concerne le lien entre sémantique cohérente et réseaux de démonstration. Constatant que la sémantique d'un préréseau non correct peut être définie, mais qu'elle n'est pas toujours un « bon » objet sémantique, on montre que la correction syntaxique d'un préréseau est équivalent au fait que sa sémantique soit effectivement un « bon » objet sémantique. Dans le même genre d'idée on caractérise sémantiquement sans évaluer effectivement le préréseau le

---

[1] Jean van Heijenoort. Absolutism and relativism in logic. In *Selected Essays* [vH85c], pages 75–83.

[2] Jaakko Hintikka. La vérité est-elle ineffable? In *« La vérité est-elle ineffable? » et autres essais*, collection Tiré à part, pages 9–47. Editions de l'éclat, 1994. Traduit de l'anglais par Antonia Soulez et François Schmitz.

[3] Andreas Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56(1-3):183–220, 1994.

[4] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543 – 574, June 1994.

[5] Jean-Yves Girard. Locus solum. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.

[6] Jean-Yves Girard. The system F of variable types: Fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[7] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[8] Jean-Yves Girard. Linear logic: its syntax and semantics. In Girard et al. [GLR95], pages 1–42.

[9] Christian Retoré. *Réseaux et Séquents Ordonnés*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, février 1993.

fait que sa réduction termine.

Le second résultat donne une réponse à une question naturelle sur le connecteur « précède », également posée par Jean-Yves Girard, dans la perspective de l'étude de la logique classique. Le connecteur « précède » étant autodual, on peut imaginer qu'il lui corresponde une modalité qui permette de dupliquer (mais pas d'effacer) les formules tant en hypothèse (entrées) qu'en conclusion (sorties). C'est effectivement le cas dans la sémantique. On peut donc envisager d'étendre le calcul ordonné à cette modalité.

5.2 Interprétation dans les espaces cohérents

La sémantique dénotationnelle d'un calcul logique associe à toute démonstration d'une formule F un objet d'un espace F^{∇}. À toute variable propositionnelle p on associe un espace cohérent arbitraire, et noté P^{∇}. Ensuite, toute formule se trouve associée à un espace cohérent, car les connecteurs (unaires ou binaires) de la logique correspondent à des opérations (unaires ou binaires) sur les espaces cohérents. Il s'agit alors d'interpréter les démonstrations d'une formule F par des objets de l'espace cohérent F^{∇}, de sorte que si une démonstration se réduit en une autre par élimination des coupures, l'interprétation, aussi appelée sémantique de la démonstration, reste inchangée. C'est l'analogue de la sémantique dénotationnelle du lambda calcul typé, un lambda terme de type A étant effectivement une démonstration de A en logique intuitionniste.

Un **espace cohérent** $A = (|A|; \frown)$ est un graphe simple (sans boucle ni arêtes multiples) non orienté, possiblement infini, dont l'ensemble des sommets est appelé la **trame**, et l'adjacence la cohérence stricte. Deux points x et y d'un espace cohérent $A = (|A|; \frown)$ sont dits **strictement cohérents** $x \frown y[A]$ s'ils sont distincts et adjacents ; ils sont dits cohérents s'ils sont adjacents ou égaux. Ils sont dit **strictement incohérents** $x \smile y[A]$ s'ils sont non adjacents et distincts ; ils sont dits incohérents s'ils sont non adjacents ou égaux.

Les objets de l'espace cohérent par lesquels les démonstrations sont interprétées sont les **cliques** de ce graphe, c'est-à-dire les ensembles de points deux à deux adjacents. Les morphismes entre deux espaces cohérents A et B que nous considérons ici sont les applications linéaires (qui représentent les implications linéaires). Une application linéaire f de A dans B est une relation entre $|A|$ et $|B|$ qui satisfait la propriété suivante : étant donnés deux couples distincts $(a,b),(a',b') \in f \subset |A| \times |B|$ si $a \frown a'[A] \vee a = a'$ alors $b \frown b'[B]$. Une application linéaire de A dans B peut aussi se voir comme une application des cliques de A dans les cliques de B en posant $f(C) = \{f(c), c \in C\}$ pour toute clique $C \subset |A|$. Cette notion d'application linéaire fait des espaces cohérents une catégorie notée COH.

La négation linéaire est un endofoncteur contravariant.

$$(|A|; \frown_A)^\perp = (|A|; (\frown_A)^\perp)$$

où $\{x,y\} \in (\frown_A)^\perp$ si et seulement si $\{x,y\} \notin (\frown_A)$; en d'autres termes, la négation transforme un graphe en son graphe complémentaire.

Un connecteur logique covariant en ses deux arguments s'interprète comme un bi-foncteur covariant en ses deux arguments. La logique linéaire a distingué deux types de connecteurs : les multiplicatifs, dont la trame est le produit cartésien des trames, et les additifs, dont la trame est la somme disjointe des trames. La négation, unaire, est à la fois additive et multiplicative.

Nous nous sommes principalement intéressés aux multiplicatifs. Pour définir un connecteur multiplicatif, il faut dire quand deux couples (a,a') et (b,b') de $|A| \times |B|$ sont strictement cohérents en fonction de la cohérence dans A de a et a' et de la cohérence dans B de b et b'.

Pour chaque connecteur $*$, présentons la cohérence selon $A * B$ en fonction de celles sur A et sur B sous forme d'une table qui se lit ainsi : s'il y a un signe ω à l'intersection de la ligne α et de la colonne β, cela signifie que lorsque les premières projections des deux couples sont α dans A et les secondes projections des deux couples sont β dans B alors les deux couples sont ω dans $A * B$.

Du fait de la covariance, sept des neuf cases sont déjà remplies :

$A * B$	$\smile$	$=$	$\frown$
$\smile$	$\smile$	$\smile$	
$=$	$\smile$	$=$	$\frown$
$\frown$		$\frown$	$\frown$

L'égalité étant exclue dans ces deux cases, cela fait 4 possibilités, dont deux commutatives :

$A \otimes B$	$\smile$	$=$	$\frown$
$\smile$	$\smile$	$\smile$	$\smile$
$=$	$\smile$	$=$	$\frown$
$\frown$	$\smile$	$\frown$	$\frown$

$A \wp B$	$\smile$	$=$	$\frown$
$\smile$	$\smile$	$\smile$	$\frown$
$=$	$\smile$	$=$	$\frown$
$\frown$	$\frown$	$\frown$	$\frown$

Les deux autres ne sont que les symétriques l'une de l'autre :

$A < B$	$\smile$	$=$	$\frown$
$\smile$	$\smile$	$\smile$	$\frown$
$=$	$\smile$	$=$	$\frown$
$\frown$	$\smile$	$\frown$	$\frown$

$A > B$	$\smile$	$=$	$\frown$
$\smile$	$\smile$	$\smile$	$\frown$
$=$	$\smile$	$=$	$\frown$
$\frown$	$\smile$	$\frown$	$\frown$

Dans les espaces cohérents il n'y a donc qu'un seul connecteur multiplicatif supplémentaire possible : c'est le précède introduit dans [1] et dont nous avons déjà parlé dans ce mémoire. Il est non commutatif, comme on le voit sur les tables de cohérence, associatif, et auto-dual ($A < B)^\perp \equiv (A^\perp < B^\perp)$).

5.3 Interprétation des (pré)réseaux

On peut bien sûr définir inductivement l'interprétation des démonstrations du calcul des séquents. Ce n'est pas ce que nous ferons ici. On va directement interpréter les (pré)réseaux de démonstration suivant la méthode des expériences de [2].

Pour définir l'interprétation d'un (pré)réseau de démonstration, on se fixe une interprétation des variables propositionnelles, ∇, ce qui fixe l'interprétation de toute formule. Pour simplifier les notations, l'espace cohérent associé à une formule F sera noté F au lieu de F^∇. On étiquette chaque (occurrence de) formule du réseau par un point de la trame de l'espace cohérent qui lui est associé. On procède ainsi : pour chaque axiome $A - A^\perp$ du réseau on choisit arbitrairement un point de la trame de A (qui est aussi celle de $A^\perp$), et l'on propage les valeurs ainsi : pour tout lien $*$ l'étiquette de sa conclusion est (u,v) si les étiquettes des prémisses sont u et v — comme la trame de $A * B$ est le produit cartésien des trames de A et B, (u,v) est bien un point de la trame de $A * B$. Un tel étiquetage est appelé une **expérience**. Si les étiquettes des deux prémisses de chaque lien *coupure* sont égales, alors l'expérience est dite **réussie**. Le **résultat d'une expérience réussie** est le n-uplet des étiquettes de ses conclusions. Si un réseau Π a pour conclusions $C_1, \ldots, C_n$, l'interprétation ou **sémantique de** Π est définie comme l'ensemble des résultats des expériences réussies de π.

Dans [2] il est montré que la sémantique d'un réseau de MLL est toujours une clique de $C_1 \wp \cdots \wp C_n$, c'est-à-dire du *par* des conclusions du réseau, et que la sémantique d'un réseau est préservée par élimination des coupures — d'où l'adjectif « dénotationnelle ». Par suite la sémantique d'un réseau n'est jamais triviale : elle contient toujours au moins $\#A_1 \times \cdots \times \#A_n$ points si la forme normale du réseau comporte n axiomes de types respectifs $A_1 - A_1^\perp \ldots A_n - A_n^\perp$. Dans [1,3] il est montré que cela vaut aussi du calcul ordonné.

---

[1] Christian Retoré. *Réseaux et Séquents Ordonnés*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, février 1993.

[2] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[3] Christian Retoré. Pomset logic: a non-commutative extension of classical linear logic. In Philippe de Groote and James Roger Hindley, editors, *Typed Lambda Calculus and Applications, TLCA'97*, volume 1210 of *LNCS*, pages 300–318, 1997.

5.4 Une sorte de complétude dénotationnelle pour les réseaux de démonstration

On remarque que la définition de l'interprétation d'un réseau ne suppose en rien que ce soit effectivement un réseau. On peut tout à fait définir la sémantique d'un préréseau. Mais sera-t-elle un objet sémantique, c'est-à-dire une clique de $C_1 \parr \cdots \parr C_n$? En général, non. Du reste, la démonstration qu'il s'agit d'une clique utilise fortement la correction du réseau, c'est-à-dire l'absence de circuit élémentaire alternant. Il est donc naturel de montrer qu'en fait, l'interprétabilité dans les espaces cohérents est équivalente à la correction du préréseau. Il faut néanmoins faire deux concessions : d'une part admettre la règle « mix » qui est validée par les espaces cohérents, et d'autre part ne considérer que les réseaux normaux. En effet, certains préréseaux incorrects se réduisent en des réseaux corrects et la sémantique est préservée par élimination des coupures : il y a donc des préréseaux incorrects qui admettent pour sémantique une clique.

Dans [1,2] nous avons montré que la correction sémantique d'un préréseau est équivalente à sa correction syntaxique, et pour MLL+mix, et pour MLL<. Qui plus est, il n'est nul besoin de faire varier l'interprétation, et l'une des deux expériences peut être fixée arbitrairement pour s'assurer de la correction ou exhiber un circuit Æ . Plus précisément :

Théorème 5.1 *Soit ∇ l'interprétation qui interprète toute variable propositionnelle par N l'espace cohérent* $(\{a,b,c,d); \{\{a,b\}; \{b,c\}; \{c,d\}\})$.

Soit Π un préréseau normal de conclusions $C_1, \ldots, C_n$ et $\mathcal{E}_1$ une expérience de Π.

Π est correct si et seulement si toute autre expérience $\mathcal{E}_2 \neq \mathcal{E}_1$ de Π satisfait $|\mathcal{E}_1| \frown |\mathcal{E}_2|[C_1 \parr \cdots \parr C_n]$.

Ce résultat vaut pour MLL+mix et MLL<.

On remarquera qu'on obtient ainsi un critère de correction, mais celui-ci est d'une complexité exponentielle.

Il est également possible d'utiliser la sémantique pour savoir si un préréseau se réduit en réseau correct, sans le réduire :

Théorème 5.2 *Soit ∇ l'interprétation qui interprète toute variable propositionnelle par N l'espace cohérent* $(\{a,b,c,d); \{\{a,b\}; \{b,c\}; \{c,d\}\})$.

Soit Π un préréseau normal de conclusions $C_1, \ldots, C_n$ et $\mathcal{E}_1$ une expérience réussie de Π.

---

[1] Christian Retoré. A semantic characterisation of the correctness of a proof net. *Mathematical Structures in Computer Science*, 7(5):445–452, 1997.

[2] Christian Retoré. On the relation between coherence semantics and multiplicative proof nets. Rapport de Recherche RR-2430, INRIA, décembre 1994. http://www.inria.fr/.

Π *se réduit en un réseau correct si et seulement si toute autre expérience* $\mathcal{E}_2 \neq \mathcal{E}_1$ *de* Π *satisfait* $|\mathcal{E}_1| \neq |\mathcal{E}_2|$.

Ce résultat vaut pour MLL+mix et MLL<.

5.5 Une modalité autoduale pour le connecteur « précède »

Le calcul MLL< est issu de la découverte du connecteur précède dans les espaces cohérents. Pour les multiplicatifs usuels $\otimes$ et $\parr$, il existe deux modalités leur correspondant : $?$ et $!$. Elles permettent de faire un usage contrôlé de la contraction et de l'affaiblissement, et sont donc utilisées pour représenter la logique intuitionniste et la logique classique en logique linéaire. Ces modalités satisfont :

$$!A \multimap \mathbf{1} \qquad !A \multimap (!A \otimes !A) \qquad !A \multimap A$$

$$\mathbf{1} \multimap ?A \qquad (?A \parr ?A) \multimap ?A \qquad A \multimap ?A$$

Vu que le connecteur « précède » est un connecteur autodual, on peut se demander s'il existe une modalité autoduale $\lhd$ lui correspondant, qui permette la contraction dans les deux sens : $\lhd A \multimap \lhd A < \lhd A$ et $\lhd A < \lhd A \multimap \lhd A$, avec une relation avec A dans les deux sens : $\lhd A \multimap A$ et $A \multimap \lhd A$. La réponse est oui, elle existe, et la contraction est même un isomorphisme linéaire. On remarquera l'absence de relation avec $\mathbf{1}$, qui correspond en logique à l'absence d'affaiblissement. S'il existait aussi une application linéaire de $\mathbf{1}$ dans $\lhd A$ et de $\lhd A$ dans $\mathbf{1}$, alors on aurait un modèle dégénéré, puisqu'il existerait une application linéaire de $\mathbf{1}$ dans A et de A dans $\mathbf{1}$ pour tout A !

Pour définir cette modalité, on considère $\mathbf{2}^\omega$ muni de la topologie produit de la topologie discrète sur $\mathbf{2} = \{0,1\}$. Cette modalité est définie ainsi :

- Les points de la trame $|\lhd A|$ sont les fonctions continues de $\mathbf{2}^\omega$ dans $|A|$ pour les topologies suivantes :
  - La topologie sur $\mathbf{2}^\omega$ est la topologie produit de la topologie discrète sur $\mathbf{2}$.
  - La topologie sur $|A|$ est la topologie discrète.

  Les applications continues de $\mathbf{2}^\omega$ dans $|A|$ s'identifient avec les arbres binaires finis dont les feuilles sont étiquetées par des éléments de $|A|$ de sorte que deux feuilles sœurs n'ont jamais la même étiquette. Il s'ensuit que si $|A|$ est dénombrable, $|\lhd A|$ l'est aussi.
- La cohérence est définie par $f \frown g[\lhd A]$ lorsqu'il existe $w \in \mathbf{2}^\omega$ tel que $f(w) \frown g(w)$ et pour tout $w' > w$ (suivant l'ordre lexicographique sur $\mathbf{2}^\omega$) $f(w') = g(w)$ — cette cohérence généralise celle du connecteur « précède ».

En posant $<_k A = A < \ldots < A$ (k fois), $\lhd A$ est la limite inductive d'une suite d'isomorphismes linéaires partiels d_n de $<_{2^n} A$ dans $<_{2^{n+1}} A$ définis par

$$d_n = \{((x_0,\ldots,x_{2^n-1}),(x_0,x_0,\ldots,x_{2^n-1},x_{2^n-1}))\} \subset |<_{2^n} A| \times |<_{2^{n+1}} A|$$

C'est aussi la limite projective des b_n qui sont les isomorphismes partiels inverses des d_n.

Théorème 5.3 *La modalité $\lhd$ est un endofoncteur de la catégorie COH.*
La modalité $\lhd A$ est auto-duale : $(\lhd A)^\perp \equiv \lhd A^\perp$.
Les espaces cohérents $(\lhd A < \lhd A)$ *et $\lhd A$ sont linéairement isomorphes.*
L'espace cohérent A est une rétraction linéaire de $\lhd A$.
Bien sûr, il n' y a pas de morphisme canonique de ***1*** *dans $\lhd A$ ni de $\lhd A$ dans* 1.

5.6 Critiques et perspectives

À part l'étude sémantique de la correction des réseaux, ces travaux sont essentiellement dévolus au connecteur « précède ». Ils n'ont pas été poursuivis, notamment parce qu'un calcul des séquents fait toujours défaut, comme on l'a expliqué précédemment. Néanmoins ils ouvrent quelques pistes.

5.6.1 Modalité autoduale et parallélisme

Cette modalité nous semble très prometteuse en particulier pour les applications de la logique linéaire aux calculs de processus. Si l'on voit le connecteur « précède » comme la composition séquentielle, $\lhd A$ signifie que A est présent à chaque instant. Néanmoins nous n'avons pas encore développé de syntaxe pour cette modalité, attendant d'avoir un calcul des séquents pour ce connecteur. Après une rapide étude, une telle modalité se rapproche beaucoup de la modalité $\natural$ de [1]. Elle trouverait sa place dans les calculs de structures d'Alessio Guglielmi [2,3] qui fournissent effectivement une interprétation temporelle au connecteur « précède ».

---

[1] Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, 1998.

[2] Alessio Guglielmi. A calculus of order and interaction. Technical Report WV-99-04, Dresden University of Technology, 1999.

[3] Alessio Guglielmi and Lutz Straßburger. Non-commutativity and MELL in the calculus of structures. In L. Fribourg, editor, *CSL 2001*, volume 2142 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, 2001.

5.6.2 Linguistique computationnelle et sémantique dénotationnelle

Si la communauté des grammaires catégorielles s'accorde à trouver bien restreinte la sémantique de Montague, il serait judicieux d'étudier ce que l'on peut tirer de la sémantique dénotationnelle. En effet, les analyses syntaxiques dans les grammaires catégorielles sont des démonstrations et ont donc une sémantique dénotationnelle. De plus les trames et la compatibilité d'atomes d'information rejoignent certains aspects de la sémantique lexicale. Néanmoins à ce jour personne n'a encore su en tirer parti.

Chapitre 6

Les réseaux de Petri en logique linéaire mixte

Résumé : *Le travail présenté ici* [1] *repose sur le calcul partiellement commutatif de Philippe de Groote (INRIA, Nancy)* [2]*, ou, plus précisément, sur une extension de celui qu'il a publié, que permet l'axiomatisation de l'inclusion des ordres séries-parallèles obtenue ultérieurement* [3]*. Il a aussi beaucoup bénéficié de suggestions de Philippe Darondeau (INRIA, Rennes), qui m'ont permis d'améliorer substantiellement une première connexion assez limitée.*

Ce travail ouvre sans doute des perspectives pour la communauté des réseaux de Petri et de la logique linéaire, notamment pour les réseaux mobiles ou d'ordre supérieur, mais aussi du point de vue des applications linguistiques : on espère obtenir ainsi une machine parallèle pour certains types de grammaires catégorielles, de même que les automates à pile sont les machines séquentielles associées aux grammaires algébriques.

---

[1] Christian Retoré. A description of the non-sequential execution of Petri nets in partially commutative linear logic. In Jan van Eijck, Vincent van Oostrom, and Albert Visser, editors, *Logic Colloquium '99 (Utrecht)*, Lecture Notes in Logic. Association for Symbolic Logic & A. K. Peters, Ltd, 2001. Complete version: RR-INRIA 4288 http://www.inria.fr/.

[2] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

[3] Denis Bechet, Philippe de Groote, and Christian Retoré. A complete axiomatisation of the inclusion of series-parallel partial orders. In H. Comon, editor, *Rewriting Techniques and Applications, RTA'97*, volume 1232 of *LNCS*, pages 230–240. Springer Verlag, 1997.

6.1 Réseaux de Petri et logique linéaire

Le lien entre réseaux de Petri et logique linéaire a été décelé dès les débuts de la logique linéaire [1,2,3,4,5,6]. L'idée est fort simple. Si l'on associe à chaque place une variable propositionnelle, un marquage peut se représenter par une conjontion multiplicative : par exemple la présence de deux jetons dans la place A et de trois jetons dans la place B correspond à la formule $A \otimes A \otimes B \otimes B \otimes B$.

Un événement se représente alors par une implication linéaire. Par exemple un événement consommant deux jetons dans la place A et un dans la place B et produisant un jeton dans la place C et deux dans la place D est représenté par une implication linéaire $A \otimes A \otimes B \multimap C \otimes D \otimes D$, ou par un axiome propre $A \otimes A \otimes B \vdash C \otimes D \otimes D$.

L'accessibilité se traduit alors par la prouvabilité en logique linéaire :

Théorème 6.1 (Gehlot & Gunter) *Un marquage N est accessible à partir d'un marquage M dans un réseau de Petri comportant les évènements $M_i \rightarrow N_i$ pour $i \in [1,n]$ si et seulement si le séquent $M \vdash N$ est démontrable en logique linéaire multiplicative en utilisant les axiomes propres $M_i \vdash N_i$, ou, ce qui revient au même, si le séquent $M,!(M_1 \multimap N_1),\ldots,!(M_n \multimap N_n) \vdash N$ est démontrable en logique linéaire intuitionniste multiplicative-exponentielle.*

Ce codage présente les inconvénients suivants :

- Tandis qu'il s'agit d'un phénomène purement multiplicatif, on est contraint
  - de se placer dans la logique linéaire avec exponentiels (dont la décidabilité est inconnue, tandis que l'accessibilité des réseaux de Petri est décidable)
  - ou d'utiliser des axiomes propres, qui nuisent aux propriétés formelles du calcul
- La notion d'exécution optimale, ou de parallélisme maximale échappe à ce

[1] Vijay Gehlot and Carl Gunter. Nets as tensor theories. In G. De Michelis, editor, *Applications of Petri nets*, 1989.

[2] Vijay Gehlot and Carl Gunter. Normal process representatives. In *L.I.C.S.*, 1990.

[3] Vijay Gehlot. *A proof-theoretic approach to semantics of concurrency*. PhD thesis, University of Pennsylvania, 1992.

[4] Anne Sjerp Troelstra. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. CSLI, 1992. (distributed by Cambridge University Press).

[5] Uffe Engberg and Glynn Winskel. Petri nets as models of linear logic. In *CAAP'90*, volume 431 of *LNCS*, pages 147–161. Springer, 1990.

[6] Uffe Engberg and Glynn Winskel. Completeness results for linear logic on Petri nets. *Annals of Pure and Applied Logic*, 86(2):101–135, 1997.

genre de codage. Par exemple Luis Künzle [7,8] a trouvé un contre-exemple à la tentative, pourtant sophistiquée, de Vijay Gehlot [1,2,3] d'une notion de preuve normale qui corresponde à une exécution optimale.

– Ce genre de codage a l'inconvénient d'abstraire du séquent démontré toute trace de l'exécution. Outre l'étude et la comparaison des exécutions, ce codage interdit aussi de considérer le langage d'un réseau (les séquences d'événements possibles), qui est pourtant nécessaire à l'étude de la synthèse de réseaux de Petri à partir de spécifications fournies par des langages formels.

Nous avons remarqué que le calcul mixte ou partiellement commutatif de Philippe de Groote [9] permettait, un fois un peu étendu, de remédier à ces inconvénients.

6.2 Une extension du calcul mixte de Philippe de Groote

Nous allons ici présenter le calcul mixte de Philippe de Groote [9] ou plus précisément une extension de la version publiée, mais qui reste fidèle à ce que son auteur souhaitait faire ; c'est le résultat obtenu avec Denis Bechet et Philippe de Groote [BdGR97] qui manquait au moment de la publication de ce calcul.

L'idée de départ de ce calcul est très naturelle, et répond aussi bien à des préoccupations logiques que linguistiques : comment faire cohabiter les connecteurs commutatifs de la logique linéaire et leurs contreparties non commutatives du calcul de Lambek ? Le langage d'un tel calcul est donc le suivant :

$$\mathcal{F} ::= \mathcal{P} \mid \mathbf{1} \mid \mathcal{F} \otimes \mathcal{F} \mid \mathcal{F} \bullet \mathcal{F} \mid \mathcal{F} \multimap \mathcal{F} \mid \mathcal{F} \setminus \mathcal{F} \mid \mathcal{F} / \mathcal{F}$$

Nous nous limiterons à la version intuitionniste de ce genre de calcul. Il suffit de structurer les contextes, c'est-à-dire d'avoir deux « virgules », l'une commutative, notée $\{\ldots,\ldots\}$ et l'autre non commutative, notée $(\ldots;\ldots)$ pour assembler les hypothèses. Ces dernières sont donc munies d'un ordre série-parallèle, struc-

---

[7] Luis Allan Künzle. *Raisonnement temporel basé sur les réseaux de Petri pour des systèmes manipulant des ressources*. Thèse de doctorat, spécialité informatique industrielle, Université Paul Sabatier, Toulouse, septembre 1997.

[8] Brigitte Pradin-Chézalviel, Luis Allan Künzle, F.rançois Girault, and Robert Valette. Calculating duration of concurrent scenarios in time Petri nets. *APII - Journal Européen des Systèmes Automatisés*, 33(8-9):943–958, 1999.

[9] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

ture déjà rencontrée au paragraphe 3.2, ce qui conduit à la définition suivante des contextes :

$$\mathcal{C} ::= \mathcal{F} \mid \{\mathcal{C},\mathcal{C}\} \mid (\mathcal{C};\mathcal{C})$$

Le calcul s'obtient en juxtaposant les règles des connecteurs commutatifs et celles des connecteurs non commutatifs, et il en résulte un calcul logique qui possède les propriétés usuelles : élimination des coupures, sémantique complète, etc.

Nous donnons ici ces règles. Les deux premières colonnes ainsi que la règle de coupure fournissent le calcul des séquents. La deuxième et la troisième colonnes donnent la formulation de ce calcul en déduction naturelle, que nous utiliserons au chapitre 9.

Implications		
$\dfrac{\Gamma[B] \vdash C \quad \Delta \vdash A}{\Gamma[\{\Delta,A \multimap B\}] \vdash C} \multimap_h$	$\dfrac{\{A,\Gamma\} \vdash C}{\Gamma \vdash A \multimap C} \multimap_i$	$\dfrac{\Delta \vdash A \quad \Gamma \vdash A \multimap B}{\{\Delta,\Gamma\} \vdash B} \multimap_e$
$\dfrac{\Gamma[B] \vdash C \quad \Delta \vdash A}{\Gamma[(\Delta; A \setminus B)] \vdash C} \setminus_h$	$\dfrac{(A;\Gamma) \vdash C}{\Gamma \vdash A \setminus C} \setminus_i$	$\dfrac{\Delta \vdash A \quad \Gamma \vdash A \setminus B}{(\Delta;\Gamma) \vdash B} \setminus_e$
$\dfrac{\Gamma[B] \vdash C \quad \Delta \vdash A}{\Gamma[(B \,/\, A; \Delta)] \vdash C} /_h$	$\dfrac{(\Gamma; A) \vdash C}{\Gamma \vdash C \,/\, A} /_i$	$\dfrac{\Delta \vdash B \,/\, A \quad \Gamma \vdash A}{(\Delta;\Gamma) \vdash B} /_e$

Produits / conjonctions		
$\dfrac{\Gamma[(A;B)] \vdash C}{\Gamma[A \bullet B] \vdash C} \bullet_h$	$\dfrac{\Delta \vdash A \quad \Gamma \vdash B}{(\Delta;\Gamma) \vdash A \bullet B} \bullet_i$	$\dfrac{\Delta \vdash A \bullet B \quad \Gamma[(A;B)] \vdash C}{\Gamma[\Delta] \vdash C} \bullet_e$
$\dfrac{\Gamma[\{A,B\}] \vdash C}{\Gamma[A \otimes B] \vdash C} \otimes_h$	$\dfrac{\Delta \vdash A \quad \Gamma \vdash B}{\{\Delta,\Gamma\} \vdash A \otimes B} \otimes_i$	$\dfrac{\Delta \vdash A \otimes B \quad \Gamma[\{A,B\}] \vdash C}{\Gamma[\Delta] \vdash C} \otimes_e$

Coupure
$\dfrac{\Gamma \vdash A \quad \Delta[A] \vdash B}{\Delta[\Gamma] \vdash B}\, coupure$

Néanmoins on souhaite qu'il y ait un rapport entre la conjonction non commutative et la conjonction commutative, par exemple que l'une entraîne l'autre, ce qui correspond à des règles structurelles. On pourra s'en étonner mais l'un comme l'autre est possible. Dans ce chapitre-ci nous ne considérerons que la version selon laquelle le produit non commutatif entraîne le produit commutatif, ce qui correspond à une règle structurelle qui permet de renforcer l'ordre série-parallèle sur les hypothèses. Par contre, dans le chapitre 9, nous utiliserons l'autre version de ce calcul.

Dans la version publiée par Philippe de Groote [1], seule la transformation du produit commutatif (ou de $\{...,...\}$) en un produit non commutatif (en $(...,...)$) est possible, et ici nous considérons une extension : si l'on a démontré un séquent $\Gamma \vdash C$ alors on peut en déduire $\Gamma' \vdash C$ où Γ' et Γ ont les mêmes formules, mais sont dotés d'ordres séries-parallèles Γ'^{SP} et Γ^{SP} satisfaisant $\Gamma'^{\mathrm{SP}} \supseteq \Gamma^{\mathrm{SP}}$. C'est en fait le calcul auquel souhaitait aboutir Philippe de Groote, mais à ce moment là nous ne disposions pas de l'axiomatisation de l'inclusion des ordres séries-parallèles vue au paragraphe 3.2.3, qui permet de décrire par des règles précises cette inclusion.

La règle peut donc se formuler ainsi :

$$\frac{\Gamma \vdash C}{\Gamma' \vdash C}\, aug(mentation) \qquad \boxed{\text{si } |\Gamma| = |\Gamma'| \text{ et } \Gamma'^{\mathrm{SP}} \supseteq \Gamma^{\mathrm{SP}}}$$

En vertu de l'axiomatisation de l'inclusion des ordres séries-parallèles du paragraphe 3.2.3, cette règle équivaut à l'ensemble des règles suivantes, où $\Psi[\Delta]$ désigne un SP-terme (un contexte) contenant le SP-terme (le contexte) Δ :

---

[1] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

égalités SP

$$\frac{\Psi[((\Gamma;\Delta);\Theta)] \vdash C}{\Psi[(\Gamma;(\Delta;\Theta))] \vdash C}\,(asso;r)$$

$$\frac{\Psi[(\Gamma;(\Delta;\Theta))] \vdash C}{\Psi[((\Gamma;\Delta);\Theta)] \vdash C}\,(asso;l)$$

$$\frac{\Psi[\{\{\Gamma,\Delta\},\Theta\}] \vdash C}{\Psi[\{\Gamma,\{\Delta,\Theta\}\}] \vdash C}\,(asso,r)$$

$$\frac{\Psi[\{\Gamma,\{\Delta,\Theta\}\}] \vdash C}{\Psi[\{\{\Gamma,\Delta\},\Theta\}] \vdash C}\,(asso,l)$$

$$\frac{\Psi[\{\Gamma,\Delta\}] \vdash C}{\Psi[\{\Delta,\Gamma\}] \vdash C}\,(com,)$$

inclusions SP

$$\frac{\Psi[\{(\Gamma;\Gamma'),(\Delta;\Delta')\}] \vdash C}{\Psi[(\{\Gamma,\Delta\};\{\Gamma',\Delta'\})] \vdash C}\,\big((<\wp 4)\big)$$

$$\frac{\Psi[\{\Gamma,(\Delta;\Delta')\}] \vdash C}{\Psi[(\{\Gamma,\Delta\};\Delta')] \vdash C}\,\big((<\wp l3)\big)$$

$$\frac{\Psi[\{(\Gamma;\Gamma'),\Delta\}] \vdash C}{\Psi[(\Gamma;\{\Delta,\Gamma'\})] \vdash C}\,\big((<\wp r3)\big)$$

$$\frac{\Psi[\{\Gamma,\Delta\}] \vdash C}{\Psi[(\Gamma;\Delta)] \vdash C}\,\big((<\wp 2)\big)$$

On peut alors se limiter aux règles suivantes pour l'élément neutre **1** commun aux deux conjonctions. En effet en utilisant la règle $(aug.)$ on peut ainsi insérer le **1** n'importe où dans le contexte, tandis qu'en l'absence de ces règles structurelles il faudrait spécifier sa place.

Règles du neutre	
$\dfrac{\Gamma \vdash C}{\{\Gamma,\mathbf{1}\} \vdash C}\,\mathbf{1}_h$	$\dfrac{}{\vdash \mathbf{1}}\,\mathbf{1}_i$

Paul Ruet a développé une version classique de ce calcul [1] en remplaçant les ordres séries-parallèles par des ordres cycliques séries-parallèles, mais il faut pour cela avoir une règle, baptisée *seesaw* qui affirme qu'au niveau le plus extérieur il n'y a pas de différence entre composition série et ce composition parallèle : chacun entraîne l'autre. L'étude de ces calculs mixtes a été poursuivie par Michele Abrusci et Paul Ruet [2], par Akim Demaille [3], et ce dernier a montré l'incompa-

[1] Paul Ruet. *Logique non-commutative et programmation concurrente*. Thèse de doctorat, spécialité logique et fondements de l'informatique, Université Paris 7, 1997.

[2] Vito Michele Abrusci and Paul Ruet. Non-commutative logic I: the multiplicative fragment. *Annals of Pure and Applied Logic*, 1999. http://www.uniroma3.it/reseaux9798.ps.

[3] Akim Demaille. *Logiques linéaires hybrides et leurs modalités*. Thèse de doctorat, spécialité informatique, Ecole Nationale Supérieure des Télécommunications de Paris, juin 1999.

tibilité entre la règle *seesaw* et l'inclusion des ordres (cycliques) séries-parallèles : on perd alors l'élimination des coupures.

L'inclusion des ordres séries-parallèles n'est donc possible que pour les calculs intuitionnistes. Elle est importante du point de vue qui nous importe dans ce chapitre : elle permet d'avoir la loi de distributivité du parallélisme (*interchange law*) $(a;b)|(x;y) \rightarrow (a|x);(b|y)$ comme une réécriture. Pour exécuter a avant b et, en parallèle x avant y, une manière de procéder est d'exécuter a et x en parallèle, puis b et y en parallèle. C'est précisément le principe qui axiomatise l'inclusion des ordres séries-parallèles. Par contre, on ne souhaite pas identifier les deux modes d'exécution afin de décrire du parallélisme vrai (*true concurrency*) où l'exécution simultanée de deux évènements est possible.

6.3 Représentation des exécutions non séquentielles

Afin de traiter de l'exécution non séquentielle des réseaux de Petri nous avons considéré des ordres (séries-parallèles) d'évènements, qui expriment les contraintes temporelles sur l'exécution des événements.

À chaque étape on exécute un sous-ensemble des événements minimaux de l'ordre, possiblement tous les événements minimaux, jusqu'à ce que tous les événements aient été exécutés, ce qui est une forme de transition par paquet (*step transition*), où l'on peut de surcroît imposer un ordre entre les événements.

Utilisons d'ores et déjà les notations logiques pour décrire les réseaux de Petri :

- Les places sont des variables propositionnelles.
- Un marquage est un élément du monoïde commutatif libre sur les places. Étant donné deux marquages M et N on notera $M \sqsupset N$ le fait que M a au moins autant de jetons que N en chaque place P — l'exposant de P dans M est au moins égal à celui de P dans N. Dans ce cas il existe un unique marquage noté $M \oslash N$ tel que $M = (M \oslash N) \otimes N$.
- Un événement e consommant le marquage M et produisant le marquage N sera noté $M \setminus N$. Lorsque seul e est connu M sera noté $\mathrm{Pre}[e]$ et N $\mathrm{Post}[e]$, et on étend cette notation aux multi-ensembles d'événements :

$$\mathrm{Pre}[Z] = \otimes_{e \in Z} \mathrm{Pre}[e] \qquad \mathrm{Post}[Z] = \otimes_{e \in Z} \mathrm{Post}[e]$$

On peut alors formaliser la possibilité d'exécuter un multi-ensemble ordonné d'événements. Considérons X un multi-ensemble ordonné d'événements. Étant donné un sous-ensemble de X clos par prédécesseur, notons $\mathrm{Front}(Y)$ les éléments minimaux du complémentaire de Y dans X. Par définition X est dit **exécutable** à partir du marquage M_0 lorsque, pour tout sous-ensemble Y de X clos par prédécesseur, on a

$$M_0 \otimes \mathrm{Post}[Y] \oslash \mathrm{Pre}[Y] \sqsupset \mathrm{Pre}[\mathrm{Front}(Y)]$$

On note cela $M_0 -[\, X \,]\longrightarrow$, et le marquage résultant est bien sûr $M_0 \otimes \mathrm{Post}[X] \oslash \mathrm{Pre}[X]$.

On remarquera que la possibilité d'exécuter toutes les linéarisations d'un ordre partiel X ne garantit en rien que X soit exécutable. On s'en convaincra aisément en considérant un réseau à une place A et à deux événements $a : (A \setminus A)$ et $b : (A \otimes A) \setminus (A \otimes A)$. Si initialement deux jetons sont présents dans la place A alors $\{a,b\}$ n'est pas exécutable tandis que ses deux linéarisations $(a; b)$ et $(b; a)$ le sont. Pour cette notion d'exécution, dès qu'un multi-ensemble ordonné est exécutable à partir d'un marquage, il existe un marquage minimum à partir duquel il est exécutable.

La proriété essentielle à notre codage concerne le remplacement d'une occurrence d'un événement $e : P \setminus Q$ par un ordre partiel d'événements Y exécutable à partir du marquage P et conduisant au marquage Q.

Proposition 6.2 *Si $P -[\, Y \,]\longrightarrow Q$ et si X contient une occurrence de l'événement $e : P \setminus Q$ alors on a*

$$\Big(M -[\, X \,]\longrightarrow\Big) \quad \Rightarrow \quad \Big(M -[\, X[e := Y] \,]\longrightarrow\Big)$$

De plus, si P est le marquage minimum de Y, la réciproque est vraie.

6.4 Le codage

Qualifions d'*exécutif* un séquent $\Gamma \vdash C$ lorsque C est un marquage et Γ un contexte ne comportant que des marquages et des événements, c'est-à-dire un séquent qui a un sens du point de vue des réseaux de Petri. Du fait de l'élimination des coupures et de la propriété de la sous-formule, que nous avons redémontrée pour ce calcul, et dont une démonstration sémantique figure dans [1], on a :

Proposition 6.3 *Tout séquent démontrable l'est sans utiliser la règle de* coupure *et dans une telle démonstration toutes les formules sont sous-formules d'une formule du séquent démontré.*

Proposition 6.4 *Une démonstration sans coupure d'un séquent exécutif ne comporte que des séquents exécutifs.*

On obtient alors le résultat suivant :

Théorème 6.5 *Un séquent exécutif $\Gamma \vdash C$ est démontrable si et seulement si*

$$M_0 -[\, X \,]\longrightarrow$$

[1] Akim Demaille. *Logiques linéaires hybrides et leurs modalités*. Thèse de doctorat, spécialité informatique, Ecole Nationale Supérieure des Télécommunications de Paris, juin 1999.

où M_0 est la conjonction de tous les marquages de Γ et X la restriction de Γ^{SP} aux événements de Γ (qui est aussi série-parallèle).

En particulier, si X est un multi-ensemble d'événements muni d'un ordre série-parallèle on a $M_0 -[\,X\,]\longrightarrow$ si et seulement si $(M_0; X) \vdash N$

On remarquera qu'on établit ainsi l'équivalence entre un énoncé universel et un énoncé existentiel plus simple : pour tout sous-ensemble clos par prédécesseur… il existe une démonstration de…

6.5 Critiques et perspectives

6.5.1 Réseaux de Petri étiquetés

Le modèle, tel qu'on l'a décrit ci-dessus, ne rend pas compte des réseaux étiquetés, où deux événements portant des noms différents peuvent avoir le même comportement. La modification à apporter pour en traiter est mineure : on procède comme dans les grammaires de Lambek, c'est-à-dire qu'un lexique associe à chaque nom d'événement un comportement, et même, possiblement, un événement peut avoir plusieurs comportement possibles. Un multi-ensemble ordonné d'événements est alors exécutable si et seulement s'il est possible de remplacer chaque événement par l'un de ses comportements de sorte que le séquent correspondant soit démontrable.

6.5.2 Ordres partiels généraux

Bien évidemment la communauté des réseaux de Petri serait ravie que l'on sache traiter d'ordres partiels qui ne soient pas forcément séries-parallèles. Malheureusement la description logique s'adaptera difficilement à ce cas trop général. En effet la décomposition des ordres est un sujet ardu (voir par exemple [2]) et de plus il faudrait être capable d'associer à tout ordre premier un connecteur logique qui n'endommage pas les propriétés du calcul. C'est sans doute trop demander, mais pour les ordres quasi-séries-parallèles de [3], c'est envisageable.

---

[2] Jean-Xavier Rampon. *Structures et algorithmiques des ensembles ordonnés*. Mémoire d'habilitation à diriger des recherches, Université de Rennes 1, 1996.

[3] Rolf H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, volume 255 of *NATO ASI series C*, pages 105–194. Kluwer, 1989.

6.5.3 Réseaux de Petri d'ordre supérieur

On remarque que cette description logique inclut naturellement des réseaux d'ordre supérieur, ou mobiles : par exemple une formule

$$((A \otimes A \otimes B \setminus B \otimes D) \otimes C) \setminus ((A \setminus E) \otimes D)$$

signifie qu'en présence d'un jeton dans la place C il est possible de transformer l'événement $A \otimes A \otimes B \setminus B \otimes D$ en un événement $A \setminus E$ et de produire un jeton dans D. Cela permet une définition propre de la mobilité, mais il faut assurément limiter l'ordre des formules car sinon les transformations décrites dépassent l'entendement. On notera néanmoins que la propriété de la sous-formule est un atout, car elle garantit que pour exécuter un réseau mobile comportant des transformations d'événements (formule du deuxième ordre) on n'a pas à considérer de formule d'ordre plus grand, du reste difficilement intelligibles en termes de réseaux de Petri.

6.5.4 Synthèse de réseaux de Petri

Une des questions classiques dans la communauté des réseaux de Petri est la suivante : étant donné un langage, existe-t-il un réseau de Petri dont les séquences exécutable d'événements soient précisément ce langage ? cette question a été résolu par Philippe Darondeau pour les langages algébriques déterministes [1].

Notre codage offre de nouvelles perspectives en ce domaine. En effet, si le langage est décrit par une grammaire de Lambek, alors la question se ramène à de l'unification de types.

6.5.5 Réseaux de démonstration pour la logique linéaire partiellement commutative

Comme me l'a fait remarquer l'un des relecteurs de l'article, les consommations de jetons seraient bien mieux représentées dans les réseaux de démonstration que dans les démonstration du calcul des séquents. Malheureusement à l'heure actuelle il n'existe pas de réseaux de démonstration pour ce calcul. Pour le calcul classique de Paul Ruet, Michele Abrusci et lui en ont trouvé, mais il nous faut ici un critère qui utilise fortement le caractère intuitionniste de ce calcul, en vertu des remarques faites au paragraphe 6.2.

---

[1] Philippe Darondeau. Deriving unbounded Petri nets from formal languages. Rapport de Recherche 3365, INRIA, février 1998. http://www.inria.fr/.

6.5.6 Perspectives linguistiques

Nous avons utilisé ce même calcul, et ces mêmes formules du premier ordre pour représenter les grammaires minimalistes d'Edward Stabler, mais avec la version de ce calcul où l'ordre série parallèle du contexte peut diminuer. Néanmoins, malgré cette différence qui doit se ramener à une dualité connue, nous espérons extraire de ce travail une notion de machine pour les grammaires minimalistes, qui serait à ces grammaires ce que sont les automates à pile aux grammaires algébriques, avec un parallélisme bienvenu dans la vérification des traits formels.

Troisième partie

Modèles grammaticaux

Chapitre 7

Syntaxe et sémantique prédicative dans les grammaires de Lambek

Résumé : *Ce chapitre concerne un aspect classique des grammaires catégorielles.*[1,2].

On y montre comment les réseaux du calcul de Lambek permettent de passer très simplement de l'analyse syntaxique d'une phrase à sa sémantique de Montague. Ces travaux découlent du plongement du calcul de Lambek dans la logique linéaire, et du plongement de la logique linéaire dans la logique intuitionniste. On clarifie ainsi les étiquetages qui jusque là calculaient les représentations sémantiques, et les réseaux de démonstrations optimisent les réductions nécessaires.

Ce travail a été réalisé avec Philippe de Groote (INRIA, Nancy). [3].

---

[1] Christian Retoré. Systèmes déductifs et traitement des langues: un panorama des grammaires catégorielles. *Technique et Science Informatique*, 20(3):301–336, 2000. Numéro spécial *Traitement Automatique du Langage Naturel* sous la direction de D. Kayser et B. Levrat. Version préliminaire RR-3917 http://www.inria.fr/.

[2] Christian Retoré. The logic of categorial grammars. Lecture notes, ESSLLI 2000, 2000. 68 page manuscript available from http://www.cs.bham.ac.uk/ esslli/.

[3] Philippe de Groote and Christian Retoré. Semantic readings of proof nets. In Geert-Jan Kruijff, Glyn Morrill, and Dick Oehrle, editors, *Formal Grammar*, pages 57–70, Prague, August 1996. FoLLI.

7.1 Sémantique de Montague et grammaires de Lambek

Lorsqu'on parle de sémantique de Montague, surtout dans un contexte logique, on s'attend à ce qu'on parle d'intentionnalité. Précisons d'emblée que nous restreignons à la sémantique de Montague sans intentionnalité, cette dernière permettant notamment de distinguer les adjectifs intentionnels, comme *bon* des adjectifs extensionnels comme *jeune*.

(7.1)	Les		étudiants	sont	des		conducteurs.
(donc)	Les	jeunes	étudiants	sont	des	jeunes	conducteurs.
**(donc)*	Les	bons	étudiants	sont	des	bons	conducteurs.

Cette restriction n'est qu'une simplification de présentation. En effet, les opérateurs d'intentionnalité, « $\hat{}$ » et « $\check{}$ » ne sont jamais que des constantes logiques et n'interfèrent pas avec ce que nous présentons : il suffirait d'adjoindre aux règles de normalisation la règle $\check{}\,\hat{}\,a \rightarrow a$. La sémantique d'un énoncé sera donc pour nous une simple formule du calcul des prédicats. L'interprétation d'un énoncé sera donc quelque chose d'assez plat, surtout d'un point de vue linguistique.

L'important est pour nous que l'interprétation soit compositionnelle : la sémantique d'un syntagme est déterminée par celle de ses constituants, en les composant suivant les règles syntaxiques appliquées pour la constitution du syntagme.

Cette sémantique est basée sur la représentation due à Alonso Church des formules du calcul des prédicats par des λ-termes simplement typés sur deux types : e le type des entités (ou individus, ou termes du premier ordre) et t le type des valeurs de vérité — généralement $\{0,1\}$, mais ce choix n'est pas imposé. Les opérations logiques correspondent alors à des constantes du λ-calcul dont les types sont les suivants:

Opération logique	Type
$\forall$	$(e \rightarrow t) \rightarrow t$
$\exists$	$(e \rightarrow t) \rightarrow t$
$\wedge$	$t \rightarrow t \rightarrow t$
$\vee$	$t \rightarrow t \rightarrow t$
$\Rightarrow$	$t \rightarrow t \rightarrow t$

Le λ-calcul est là pour gérer proprement la substitution du calcul des prédicats. Les connecteurs et quantificateurs de la logique sont des constantes du point de vue du λ-calcul. Les énoncés clos sont des λ-termes de type t, et un énoncé à n variables libres d'individus est un λ-terme de type $e \rightarrow e \cdots \rightarrow t$, avec n fois e : si on lui fournit n entités, alors on obtiendra une proposition.

Le calcul de la sémantique la construit à partir de la sémantique de chaque mot, donnée dans le lexique sous la forme d'un λ-terme, et de l'analyse syntaxique de l'énoncé. Les articles originaux de Richard Montague [Tho74] utilisent pour la syntaxe des grammaires syntagmatiques, et pour la sémantique des grammaires catégorielles. Chaque règle syntagmatique se voit associer son pendant sémantique, de nature catégorielle.

Il est donc *a priori* plus simple de calculer la sémantique d'un énoncé en partant d'une grammaire catégorielle, et en particulier d'une grammaire de Lambek, et c'est là l'un des avantages des grammaires catégorielles, comme on l'explique dans [1].

7.1.1 Les grammaires de Lambek

Dans le chapitre 4 nous avons défini le calcul de Lambek, dans une version unilatère, mais nous n'avons pas explicité son fonctionnement grammatical. Il est très simple. Les formules du calcul de Lambek sont définies comme suit, à partir d'un ensemble de variable propositionnelles P :

$$L ::= P \mid L \setminus L \mid L / L \mid L \bullet L$$

On trouvera dans la figure 7.1.1 le calcul de Lambek, exprimé en déduction naturelle et en calcul des séquents.

Une grammaire de Lambek est la donnée d'un lexique qui à chaque mot associe ses types, qui sont des formules de Lambek. Un énoncé $m_1 \cdots m_n$ appartient au langage lorsqu'il existe pour chaque mot m_i un type T_i du mot m_i tel que le séquent $T_1, \ldots, T_n \vdash S$ soit démontrable dans le calcul de Lambek. Une analyse syntaxique de l'énoncé est une telle démonstration.

7.2 Sémantique de Montague pour les grammaires de Lambek

Les types syntaxiques des grammaires de Lambek sont usuellement construits à partir des types de base suivants :

- S le type des phrase (correspondant au symbole initial de la grammaire générative correspondante)
- np le type des syntagmes nominaux et des noms propres

[1] Christian Retoré. Systèmes déductifs et traitement des langues: un panorama des grammaires catégorielles. *Technique et Science Informatique*, 20(3):301–336, 2000. Numéro spécial *Traitement Automatique du Langage Naturel* sous la direction de D. Kayser et B. Levrat. Version préliminaire RR-3917 http://www.inria.fr/.

Calcul des séquents		
$\dfrac{}{A \vdash A}\, axiome$		
$\dfrac{\Gamma,B,\Gamma' \vdash C \quad \Delta \vdash A}{\Gamma,\Delta,A \setminus B,\Gamma' \vdash C} \setminus_h$	$\dfrac{A,\Gamma \vdash C}{\Gamma \vdash A \setminus C} \setminus_i \quad \Gamma \neq \epsilon$	$\dfrac{\Delta \vdash A \quad \Gamma \vdash A \setminus B}{\Gamma \vdash B} \setminus_e$
$\dfrac{\Gamma,B,\Gamma' \vdash C \quad \Delta \vdash A}{\Gamma,B \,/\, A,\Delta,\Gamma' \vdash C} /_h$	$\dfrac{\Gamma,A \vdash C}{\Gamma \vdash C \,/\, A} /_i \quad \Gamma \neq \epsilon$	$\dfrac{\Delta \vdash B \,/\, A \quad \Gamma \vdash A}{\Gamma \vdash B} /_e$
$\dfrac{\Gamma,A,B,\Gamma' \vdash C}{\Gamma,A \bullet B,\Gamma' \vdash C} \bullet_h$	$\dfrac{\Delta \vdash A \quad \Gamma \vdash B}{\Delta,\Gamma \vdash A \bullet B} \bullet_i$	$\dfrac{\Delta \vdash A \bullet B \quad \Gamma,A,B,\Gamma' \vdash C}{\Gamma,\Delta,\Gamma' \vdash C} \bullet_e$
$\dfrac{\Gamma \vdash A \quad \Delta[A] \vdash B}{\Delta[\Gamma] \vdash B}\, coupure$	**Déduction naturelle**	

FIG. 7.1 – *Les règles du calcul de Lambek.*

– n le type des noms communs

À chaque type syntaxique x on va associer inductivement un type sémantique x^* ainsi :

– $S^* = t$ une phrase s'interprète comme une proposition
– $np^* = e$ un groupe nominal correspond à un individu
– $n^* = e \to t$ un nom commun à une propriété des individus
– $(b \,/\, a)^* = (a \setminus b)^* = a^* \to b^*$

Le lexique associe à chaque type T_i d'un mot un λ-terme t_i dont le type est T_i^* est la traduction sémantique du type syntaxique. Celui ci contient éventuellement d'autres conclusions, qui correspondent aux types des constantes logiques utilisées dans l'interprétation du mot.

(7.2) Certains énoncés parlent d'eux-mêmes.

Plutôt que des définitions formelles, que l'on trouvera dans [1], voyons sur l'exemple ci-dessus comment se calcule la réprésentation sémantique d'un énoncé à partir de la sémantique des mots le composant et de l'analyse syntaxique.

mot	***Type syntaxique*** u ***Type sémantique*** u^* ***Représentation sémantique :*** λ***-terme de type*** u^* x^v ***signifie que la variable ou la constante*** x ***est de type*** v
certains	$(S \,/\, (sn \setminus S)) \,/\, n$ $(e \to t) \to ((e \to t) \to t)$ $\lambda P^{e\to t}\, \lambda Q^{e\to t}\, (\exists^{(e\to t)\to t}\, (\lambda x^e (\wedge^{t\to(t\to t)} (P\ x)(Q\ x))))$
énoncés	n $e \to t$ $\lambda x^e(\mathsf{enonce}^{e\to t}\ x)$
parlent_de	$(sn \setminus S) \,/\, sn$ $e \to (e \to t)$ $\lambda x^e\ \lambda y^e\ ((\mathtt{parler\_de}^{e\to(e\to t)}\ x)y)$
eux-mêmes	$((sn \setminus S) \,/\, sn) \setminus (sn \setminus S)$ $(e \to (e \to t)) \to (e \to t)$ $\lambda P^{e\to(e\to t)}\ \lambda x^e\ ((P\ x)x)$

Commençons par montrer l'appartenance de l'énoncé « *Certains énoncés parlent d'eux-mêmes.* » au langage engendré par ce lexique. Pour cela construisons une déduction naturelle de conclusion :

[1] Christian Retoré. The logic of categorial grammars. Lecture notes, ESSLLI 2000, 2000. 68 page manuscript available from http://www.cs.bham.ac.uk/ esslli/.

$$(S \,/\, (sn \setminus S)) \,/\, n \;,\; n \;,\; (sn \setminus S) \,/\, sn \;,\; ((sn \setminus S) \,/\, sn) \setminus (sn \setminus S) \vdash S$$

en abrégeant en partie droite les types syntaxiques par $C\ E\ P\ X$:

$$\dfrac{\dfrac{C \vdash (S/(sn\backslash S))/n \quad E \vdash n}{C,E \vdash (S/(sn\backslash S))}\,/_e \quad \dfrac{P \vdash (sn\backslash S)/sn \quad X \vdash ((sn\backslash S)/sn)\backslash(sn\backslash S)}{P,X \vdash (sn\backslash S)}\,\backslash_e}{C,E,P,X \vdash S}\,/_e$$

En utilisant l'homomorphisme qui transforme les types syntaxiques en types sémantiques on obtient la démonstration intuitionniste suivante, où C^*,E^*,P^*,X^* sont des abréviations des types sémantiques associés ‡ C,E,P,X :

$$\dfrac{\dfrac{C^* \vdash (e \to t) \to (e \to t) \to t \;\; E^* \vdash e \to t}{C^*,E^* \vdash (e \to t) \to t}\to_e \quad \dfrac{P^* \vdash e \to e \to t \;\; X^* \vdash (e \to e \to t) \to e \to t}{P^*,X^* \vdash e \to t}\to_e}{C^*,E^*,P^*,X^* \vdash t}\to_e$$

Le λ-terme codant cette démonstration est tout simplement ((c e) (x p)) de type t où c,e,x,p sont des variables de types respectifs C^*,E^*,X^*,P^*.

Remplaçons ces variables par les λ-termes de mêmes types associés aux mots par le lexique — en omettant les types à l'intérieur des λ-termes, qui figurent dans le lexique ci-dessus. On obtient le λ-terme de type t suivant (écrit sur les deux premières lignes), que l'on réduit :

$$\Big((\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e(\wedge(P\ x)(Q\ x)))))(\lambda x^e(\texttt{enonce}^{e\to t}\ x))\Big)$$
$$\Big((\lambda P^{e\to(e\to t)}\ \lambda x^e\ ((P\ x)x))(\lambda x^e\ \lambda y^e\ ((\texttt{parler\_de}^{e\to(e\to t)}\ x)y))\Big)$$

$$\downarrow \beta$$

$$(\lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e(\wedge^{t\to(t\to t)}(\texttt{enonce}^{e\to t}\ x)(Q\ x)))))$$
$$(\lambda x^e\ ((\texttt{parler\_de}^{e\to(e\to t)}\ x)x))$$

$$\downarrow \beta$$

$$(\exists^{(e\to t)\to t}\ (\lambda x^e(\wedge(\texttt{enonce}^{e\to t}\ x)((\texttt{parler\_de}^{e\to(e\to t)}\ x)x))))$$

Ce terme réduit représente la formule du calcul des prédicats suivante :

$$\exists x : e\ (\texttt{enonce}(x) \wedge \texttt{parler\_de}(x,x))$$

C'est bien la représentation sémantique de l'énoncé précédemment analysé.

7.3 Calcul de Lambek et logique linéaire

Voici le lien avec le chapitre 4 où le langage considéré était celui de la logique linéaire :

$p^{\#} = p$

$(A \setminus B)^{\#} = (A^{\#})^{\perp} \wp B^{\#}$

$(B / A)^{\#} = B^{\#} \wp (A^{\#})^{\perp}$

$(A \bullet B)^{\#} = (A^{\#}) \otimes B^{\#}$ (les produits sont les mêmes, et, du reste, nous ne les utiliserons pas ici)

Réciproquement, si F est une formule de la logique linéaire multiplicative intuitionniste, c'est-à-dire de $F^{\circ} \uplus F^{\bullet}$ du paragraphe 4.1.3 alors elle admet un équivalent dans le calcul de Lambek.

Cette traduction préserve les démonstrations : si d est une démonstration de $F_1, \ldots, F_n \vdash C$ dans le calcul de Lambek, alors elle correspond règle à règle à une démonstration de $\vdash F_n, \ldots, F_1, C$ dans CyMLL intuitionniste et sans séquence vide, et réciproquement. Pour plus de détails on pourra consulter [1].

En vertu du chapitre 4 une démonstration, et donc une analyse syntaxique, peut être représentée par un réseau de démonstration, et cela présente plusieurs avantages.

- Des analyses syntaxiques identiques, qui produisent la même structure en constituants, correspondent à un seul et même réseau — tandis que des démonstrations différentes dans le calcul des séquents peuvent correspondre à une même analyse syntaxique.
- Un autre avantage est que la structure des réseaux est linguistiquement une structure porteuse de sens : c'est par exemple dans ce formalisme qu'on a pu rendre compte de la complexité instantanée de compréhension de relatives imbriquées.
- L'avantage qui nous importera le plus ci, est que la réduction nécessaire au calcul des représentation sémantiques est optimisée par la traduction en logique linéaire, et aussi par la réduction des réseaux.
- Finalement cette vision du calcul de Lambek suggère des extensions linguistiquement plus riches, utilisant plus les démonstrations que les formules, comme le modèle que nous proposons au chapitre 8.

---

[1] Christian Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.

7.4 Réseaux de démonstration pour le λ-calcul simplement typé

Nous n'avons pas parlé des modalités de la logique linéaire, qui permettent d'y plonger la logique intuitionniste [1,2]. En effet, nous n'avons pas travaillé directement sur cette question, mais il convient d'en dire deux mots pour ce chapitre qui les utilise. Le langage considéré dans cette partie est le suivant :

$$E ::= P \mid P^{\perp} \mid E \parr E \mid E \otimes E \mid !E \mid ?E$$

Ces connecteurs unaires sont gérés par les règles suivantes :

$$\frac{\vdash ?\Delta, A, ?\Delta'}{\vdash ?\Delta, !A, ?\Delta'} \text{ promotion}$$

$$\frac{\vdash \Delta, ?A, ?A, \Delta'}{\vdash \Delta, ?A, \Delta'} \text{ contraction}_d$$

$$\frac{\vdash \Delta, \Delta'}{\vdash \Delta, ?A, \Delta'} \text{ affaiblissement}_d$$

Les formules intuitionnistes sont les suivantes :

$$E^O ::= P \mid E^O \parr E^I \mid E^I \parr E^O \mid E^I \parr E^I \mid E^O \otimes E^O \mid !E^O$$

$$E^I ::= P^{\perp} \mid E^I \otimes E^O \mid E^O \otimes E^I \mid E^O \otimes E^O \mid E^I \parr E^I \mid ?E^I$$

En se restreignant aux formules intuitionnistes dans le calcul des séquents présenté ci dessus, on obtient une formulation unilatère de la logique linéaire intuitionniste. En vertu des lois de De Morgan déjà rencontrées, toute formule admet une forme normale négative, où la négation ne porte que sur les variables propositionnelles.

Pour traduire la logique intuitionniste, restreinte à l'implication notée $\rightarrow$, en logique linéaire, il suffit de poser :

$$(A \rightarrow B)^{\#} = ?(A^{\#})^{\perp} \parr B$$

---

[1] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[2] Jean-Yves Girard. Linear logic: its syntax and semantics. In Girard et al. [GLR95], pages 1–42.

On obtient alors une correspondance règle à règle entre une démonstration de $F_1, \ldots, F_n \vdash C$ en logique intuitionniste et une démonstration en logique linéaire de $\vdash F_n^{\#}, \ldots, F_1^{\#}, C$.

Nous n'avons pas envisagé de réseaux de démonstration pour les modalités. Nous utiliserons la formulation avec « boîtes » des réseaux intuitionnistes parce quelle est la plus simple. En fait cette description n'est pas pleinement satisfaisante, et [3] montre qu'on peut se passer des boîtes dans le cas intuitionniste ; mais cette amélioration est indépendante de notre propos, qui pourrait se faire avec la description de François Lamarche. Plus récemment Sylvain Pogodalla a obtenu une modélisation sans boîte des modalités dans le cadre des réseaux bicolores avec liens [4], qui serait aussi pertinente.

Nous ajouterons aux préréseaux les liens « déréliction » , « contraction » et « promotion » . La règle de promotion donnera lieu à une boîte, c'est-à-dire une partie du préréseau, dont toutes les conclusions sont de la forme $?C_i$ à l'exception de la prémisse du lien promotion. Un préréseau est appelé un réseau ou dit correct lorsqu'il satisfait les propriétés (récursives) suivantes :

- Les boîtes sont bien emboîtées, c'est-à-dire qu'étant données deux boîtes soit l'une est incluse dans l'autre soit elles sont disjointes.
- En remplaçant chaque boîte de profondeur 0 par toutes les arêtes B XY avec X et Y conclusion de la boîte on obtient un graphe sans cycle Æ et avec un chemin Æ entre toute paire de point.
- L'intérieur de chaque boîte de profondeur 0 est correct.

Les réseaux ainsi définis correspondent exactement aux démonstrations de MELL. Si de plus leurs conclusions sont des formules intuitionnistes, alors ils correspondent aux démonstrations de IMELL, et donc aux démonstration de la logique intuitionniste. Comme un λ-terme est une démonstration en logique intuitionniste de son type avec pour hypothèses les types de ses variables libres, un λ-terme simplement typé est un réseau de démonstration de IMELL.

La traduction en logique linéaire des formules d'un réseau de Lambek donne un réseau de IMELL : c'est immédiat, et c'est pourtant ce résultat qui va nous donner un algorithme très simple pour le calcul de la sémantique d'un énoncé.

---

[3] François Lamarche. Proof nets for intuitionistic linear logic: Essential nets. 35 page technical report available by FTP from the Imperial College archives, April 1994.

[4] Sylvain Pogodalla. *Réseaux de preuve et génération pour les grammaires de types logiques*. Thèse de doctorat, spécialité informatique, INPL, Nancy, 2001.

7.5 Calcul de la sémantique d'un énoncé

Au vu de la section précédente, on peut traduire les types sémantiques en des formules de la logique linéaire intuitionnistes avec modaliés (ou exponentielles).

- $S^* = t$
  $S^\ell = t$
- $np^* = e$
  $np^\ell = e$
- $n^* = e \to t$
  $n^\ell = ?e^\perp \multimap t$
- $(b \,/\, a)^* = (a \setminus b)^* = a^* \to b^*$
  $(b \,/\, a)^\ell = (a \setminus b)^\ell = ?(a^\ell)^\perp \multimap b^\ell$

Un λ-terme de type T_i^* sera vu comme un réseau de conclusion T_i^ℓ et dont les autres conclusions sont la traduction en logique linéaire des types des constantes logiques. Soit maintenant un réseau Π donnant l'analyse syntaxique d'une phrase $m_1 \cdots m_n$. Ce réseau de Lambek a pour conclusions $(T_1^\#)^\perp, \ldots, (T_n^\#)^\perp, S^\#$. Si on lui applique l'homomorphisme de types, alors on obtient un réseau Π^* dont les conclusions sont $(T_1^\ell)^\perp, \ldots, (T_n^\ell)^\perp, t$.

Considérons alors le réseau de IMELL obtenu en branchant par coupure chaque entrées $(T_i^\ell)^\perp$ avec la sortie de type T_i^ℓ de l'interprétation sémantique. On obtient ainsi un réseau de IMELL correct. Les réseaux de IMELL se normalisant, l'élimination des coupures nous donne un réseau de sortie t et dont les seules entrées correspondent aux constantes logiques. Ce réseau correspond à un λ-terme de type t dont les seules variables libres sont en fait des constantes, les connecteurs et quantificateurs. Il représente une formule du calcul des prédicats qui est l'interprétation de la phrase.

7.6 Critiques et perspectives

Ici nous n'avons envisagé que le coté analyse d'un énoncé : à partir d'un lexique de Lambek, on analyse syntaxiquement un énoncé, et, à partir de l'analyse syntaxique et de la sémantique de chaque item lexical, on produit la sémantique de l'énoncé tout entier. En fait, ce travail établit une relation entre diverses données : le lexique qui contient pour chaque entrée son type syntaxique et sa représentation sémantique, les analyses syntaxiques et les représentations sémantiques des énoncés corrects. On pourrait aussi supposer que l'on se donne le lexique et la représentation sémantique d'un énoncé et chercher à reconstituer un ou des énoncés dont la sémantique soit celle qu'on s'est donnée. Cette question est une forme particulière de la génération automatique de textes — très restreinte puisqu'il s'agit

d'énoncés et non de textes entiers, et qu'on part de la sémantique de Montague qui est elle-même très proche de la syntaxe. On pourrait également supposer qu'on se donne un ensemble fini d'énoncés supposés corrects, leur sémantique et la partie sémantique du lexique ; on chercherait alors à construire la partie syntaxique du lexique. Il s'agit là d'une question d'apprentissage, et plus précisément d'inférence grammaticale, à partir d'exemples positifs uniquement : on remarquera que pour une grammaire lexicalisée, on n'a pas à apprendre de règles, mais juste à trouver l'information lexicale associée à chaque mot.

7.6.1 Une amélioration formelle

Dans le passage du calcul de Lambek à la logique linéaire que nous avons décrit dans ce chapitre, on peut regretter qu'on omette la non commutativité du calcul de Lambek. Il serait sans doute plus élégant de la conserver, en se plaçant dans un calcul où cohabitent connecteurs commutatifs et non commutatifs, et où ce soit l'élimination des coupures qui force la commutativité — les λ-termes obtenus n'ont pas de raison d'être non commutatifs. La thèse d'Akim Demaille [1] devrait permettre de répondre à ces questions, puisqu'il a étudié les calculs mixtes en présences de modalités.

7.6.2 Catégories contractables

La traduction donnée ici peut s'optimiser si l'on prend en compte le fait suivant: seules certaines catégories linguistiques sont susceptibles d'être contractées au sens logique du terme. En effet, s'il est clair que les syntagmes nominaux sont contractés, ne serait-ce que par les pronoms réfléchis, les syntagmes complexes, et surtout les syntagmes incomplets, ne le sont pas. Cela signifie que dans les représentations sémantiques des mots, seules certaines variables sont contractées. On peut alors traduire un type $a \to b$ dans lequel la catégorie a n'est pas contractable, par la formule linéaire $a^{\#} \multimap b\# \equiv (a^{\#}) \parr b$, et l'absence de la modalité $?$, assure une réduction plus rapide.

Néanmoins il faut de solides connaissances linguistiques pour définir la classe de ces catégories.

---

[1] Akim Demaille. *Logiques linéaires hybrides et leurs modalités*. Thèse de doctorat, spécialité informatique, Ecole Nationale Supérieure des Télécommunications de Paris, juin 1999.

7.6.3 Génération d'énoncés à partir de représentations sémantiques

Ce travail a été mené par Sylvain Pogodalla (Xerox RCE, Grenoble) sur la base du travail ici présenté. Utilisant les graphes R&B (cf. chapitre 4) et une représentation matricielle, notamment pour l'élimination des coupures, adaptée de [1,2] ce dernier arrive à engendrer l'analyse syntaxique, et donc l'énoncé, à partir de la représentation sémantique, de la sémantique des mots et de leur type syntaxique. Bien que ces résultats soient excellents par rapport à ce qui existe, notamment par leur précision, on notera, qu'il n'y a néanmoins pas de miracle : la sémantique de Montague est très proche de la syntaxe, et dans ce premier travail il s'est restreint au cas n'impliquant pas la règle de contraction (en son absence, on ne peut identifier deux syntagmes nominaux, et en particulier les pronoms réfléchis sont exclus).

7.6.4 Un projet : apprentissage de grammaires de Lambek

Partant d'hypothèses linguistiques fondées selon lesquelles l'apprentissage de la syntaxe par l'enfant nécessite la connaissance de la sémantique des mots et de la sémantique des énoncés (corrects) entendus, comme cela est établi dans [3,4], Isabelle Tellier a cherché des algorithmes qui permettent de trouver le type syntaxique des mots [5,6]. Nous projetons d'utiliser les résultats de ce chapitre pour pousser plus loin cette approche, qui en est encore à ses débuts.

---

[1] Jean-Yves Girard. Towards a geometry of interaction. In *Categories in Logic and Computer Science—Boulder, June 1987*, volume 92 of *Contemporary Mathematics*, pages 68–109. AMS, 1990.

[2] Jean-Yves Girard. Geometry of interaction, I: interpretation of system F. In *Proceedings of the ASL meeting held in Padova*, August 1989.

[3] L.R. Gleitman and E.L. Newport. The invention of language by children: Environmental and biological influences on the acquisition of language. In Gleitman and Liberman [GL95], chapter 1, pages 1–24.

[4] Steven Pinker. Language acquisition. In Gleitman and Liberman [GL95], chapter 6, pages 135–182.

[5] Isabelle Tellier. Meaning helps learning syntax. In *Fourth International Colloquium on Grammatical Inference, ICG'98*, volume 1433 of *LNAI*, pages 25–36, 1998.

[6] Daniela Dudau-Sofronie, Isabelle Tellier, and Marc Tommasi. From logic to grammars via types. In Popelìnskỳ and Nepil [PN01], pages 35–46.

Chapitre 8

Modules ordonnés et grammaires

Résumé : *Ce modèle est issu des grammaires de Lambek. On en retrouve l'idée essentielle : une analyse syntaxique est une démonstration. Néanmoins, le fonctionnement des grammaires est généralisé en ce sens qu'on associe aux mots du lexique non une simple formule mais une démonstration partielle qui exprime son comportement. Plus techniquement, on fera usage de la procédure d'élimination des coupures tandis que les démonstrations normales suffisent dans le calcul de Lambek. Ce faisant on verra naturellement apparaître des opérations de composition de syntagmes qui s'apparentent à celles des grammaires d'arbres adjoints.*

Ce modèle grammatical, ou plutôt ce type de modèles grammaticaux, a été conçu avec Alain Lecomte (Université de Grenoble II) [1,2,3,4].

---

[1] Alain Lecomte and Christian Retoré. Pomset logic as an alternative categorial grammar. In Morrill and Oehrle [MO95], pages 181–196.

[2] Alain Lecomte and Christian Retoré. Words as modules and modules as partial proof nets in a lexicalised grammar. In Abrusci and Casadio [AC96], pages 187–198.

[3] Alain Lecomte and Christian Retoré. Words as modules: a lexicalised grammar in the framework of linear logic proof nets. In Carlos Martin-Vide, editor, *Mathematical and Computational Analysis of Natural Language — selected papers from ICML'96*, volume 45 of *Studies in Functional and Structural Linguistics*, pages 129–144. John Benjamins publishing company, 1998.

[4] Alain Lecomte and Christian Retoré. Logique des ressources et réseaux syntaxiques. In D. Genthial, editor, 4 ème conférence sur le Traitement automatique du langage naturel, TALN'97, pages 70–83, Grenoble, 1997.

8.1 Principes de ce modèle

Nous avons expliqué au chapitre 7 comment procède une grammaire de Lambek. A chaque mot est associé une formule, et analyser un syntagme de type U c'est démontrer U sous les hypothèses correspondant aux mots du syntagme. Néanmoins, pour jolies qu'elle soient, ce type de grammaires est limité : formellement, on sait qu'elle ne décrivent que des grammaires algébriques, mais surtout des constructions courantes leur échappent, telles les constituants discontinus, l'extraction, l'ordre des mots relativement libre :

(8.1) Je `ne` sais `pas`.

(8.2) `Flaubert`, j'adore.

(8.3) Pierre entend `Marie chanter`. Pierre entend `chanter Marie`.

Je ne crois pas qu'il puisse exister de démonstration possible de toutes les propriétés du genre : *tel type de grammaires ne peut rendre compte de telle construction linguistique* dans la mesure où l'on sous-entend toujours que la description doit avoir un sens du point de vue linguistique : ce n'est pas seulement le langage reconnu qui importe, mais aussi les analyses obtenues. Nous avons donc souhaité étendre les grammaires de Lambek de manière à rendre compte correctement de certaines constructions syntaxiques jusque là absentes de ce formalisme.

Pour ce faire, nous avons enrichi les grammaires de Lambek dans deux directions : d'une part la logique que nous avons utilisée est le calcul ordonné, d'autre part nous nous sommes placés au niveau des démonstrations plutôt qu'à celui des formules. Le cas d'une simple formule A correspond alors un axiome $A - A^{\perp}$. Au niveau logique, souhaitant nous départir des ordres linéaires qui s'avèrent très contraignants dans le calcul de Lambek, nous avons travaillé dans le calcul ordonné qui mélange les connecteurs usuels (commutatif) et le connecteur non commutatif et autodual « précède ». On dispose en outre ainsi d'un calcul classique, qui, à la différence du calcul non commutatif classique de David Yetter [1], ne valide pas la règle d'échange cyclique, douteuse d'un point de vue linguistique.

La logique est réservée aux mécanismes de composition des syntagmes, et les propriétés autres qu'il faut bien prendre en compte, tels l'accord, ou l'existence de domaines de localité sont prises en compte par des calculs faits sur le réseau de démonstration, mais n'affectent pas la structure du réseau. Soit ce sont des étiquettes qui se calculent sur le réseau, soit ce sont des propriétés supplémentaires demandées au réseau pour qu'il soit considéré comme une analyse linguistique correcte.

---

[1] David N. Yetter. Quantales and (non-commutative) linear logic. *Journal of Symbolic Logic*, 55:41–64, 1990.

8.2 Un modèle grammatical issu du calcul ordonné

Le lexique associe à chaque mot un réseau partiel ou module. Analyser une phrase consiste à assembler les réseaux partiels en un réseau complet. Un réseau partiel est simplement un réseau correct (sans circuit élémentaire alternant) mais dont certaines feuilles ne sont pas conclusion d'un axiome : celles-ci sont appelées les hypothèses du réseau partiel.

On dispose de deux modes d'assemblage de réseaux partiels : par coupure et par axiome, dont on peut user simultanément.

Le branchement par axiome est défini comme suit. Soient Π_1 et Π_2 des pré-réseaux de conclusions respectives $C_1^1,\ldots,C_1^{k_1}$ et $C_2^1,\ldots,C_2^{k_2}$, et d'hypothèses respectives $H_1^1,\ldots,H_1^{h_1}$ et $H_2^1,\ldots,H_2^{h_2}$. On se donne un ensemble de couples (H,C) d'occurrences de la même formule, tels que H soit une hypothèse de Π_i et C une conclusion de Π_j avec $\{i,j\} = \{1,2\}$. Le branchement des deux modules par axiome consiste simplement à identifier H et C pour tous les couples retenus. On notera que c'est une forme de composition totalement symétrique : on ne peut dire que l'un soit la fonction et l'autre l'argument.

Le branchement par coupure consiste simplement à prendre une conclusion de l'un et et une conclusion de l'autre qui soient la négation l'une de l'autre et à les relier par un lien coupure qu'on élimine ensuite.

Concernant le branchement par coupure on fera deux observations.

D'une part c'est pour ainsi dire la première fois que la coupure est utilisée dans une grammaire de type logique. L'élimination des coupures est absolument nécessaire à l'obtention d'une structure d'analyse. En effet, tant que les coupures ne sont pas éliminées, l'ordre des mots n'est pas déterminé. Plus précisément, si une coupure est le seul lien entre deux modules correspondant à l'analyse de deux syntagmes, on aura d'une part un ordre entre les mots de l'un et d'autre part un ordre entre les mots de l'autre, mais aucun ordre entre les mots d'un syntagme et ceux de l'autre.

D'autre part on remarque qu'une forme restreinte de branchement par coupure correspond à l'*adjonction*, ce mode de composition linguistique que l'on retrouve aussi bien dans les TAGs que dans la grammaire générative, et qui correspond typiquement à l'application d'un modifieur, qui sont de type $X \setminus X$ ou X / X dans les grammaires de Lambek.

(8.4)

Un	**beau**	film
$(sn \,/\, n)$	$(\mathbf{n} \,/\, \mathbf{n})$	n

(8.5)

Un	**très**	beau	film
$(sn \,/\, n)$	$((\mathbf{n} \,/\, \mathbf{n}) \,/\, (\mathbf{n} \,/\, \mathbf{n}))$	$(n \,/\, n)$	n

En effet, si on définit une *porte* comme un lien $X \otimes X^{\perp}$ reliant deux axiomes

$X^{\perp} - X$ et $X^{\perp} - X$ on a l'équivalent d'un nœud adjoignable dans une grammaire TAGs.

La porte peut s'éliminer, étant elle même une coupure (il est bien connu qu'une coupure se comporte comme un lien $X \otimes X^{\perp}$). Les modifieurs sont adjoints en branchant par coupure une porte avec un modifieur dont la sortie est de la forme $X \wp X^{\perp}$, et c'est seulement après élimination des coupures que l'ordre des mots, exprimé par les étiquettes, place le modifieur et le syntagme auquel il est adjoint.

On peut ainsi rendre compte de constructions syntaxiques qui jusque là échappaient aux grammaires catégorielles.

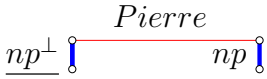
Les différents ordres des mots, dans le cas d'ordre des mots relativement libre, comme dans l'exemple classique des verbes de perception en français donné ci-dessous, sont décrits par une même analyse.

(8.6) Pierre entend Marie chanter

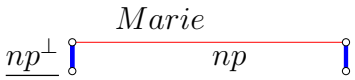
(8.7) Pierre entend chanter Marie

FIG. 8.1 – *modules: Pierre ; entend ; chanter; Marie*

– Pierre



– Marie



– chanter

– entend

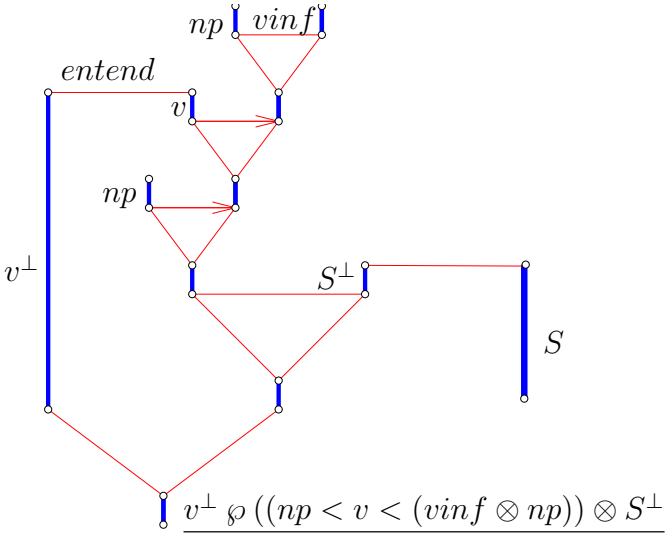
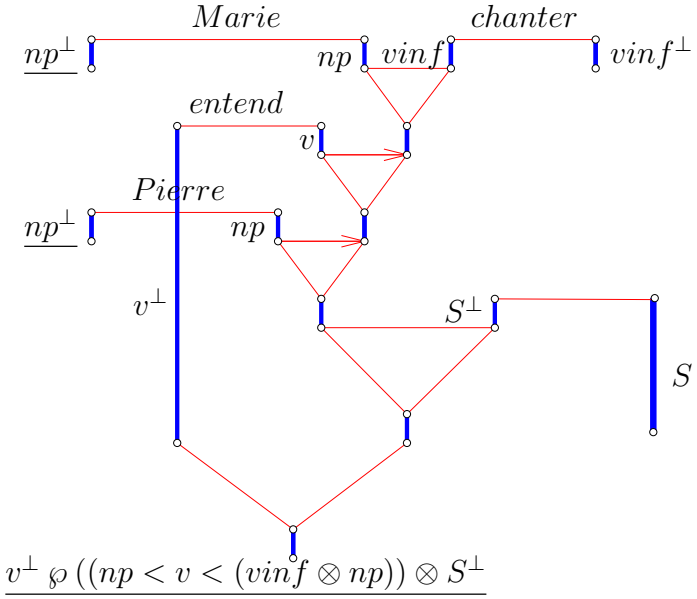


FIG. 8.2 – *Pierre < entend < {chanter,Marie}*

– Pierre entend chanter Marie



On peut également traiter des constituants discontinus, par exemple de la négation en français. Pour ce faire, l'entrée lexicale associée à *ne...pas* comporte deux axiomes, l'un étiqueté *ne* et l'autre *pas*.

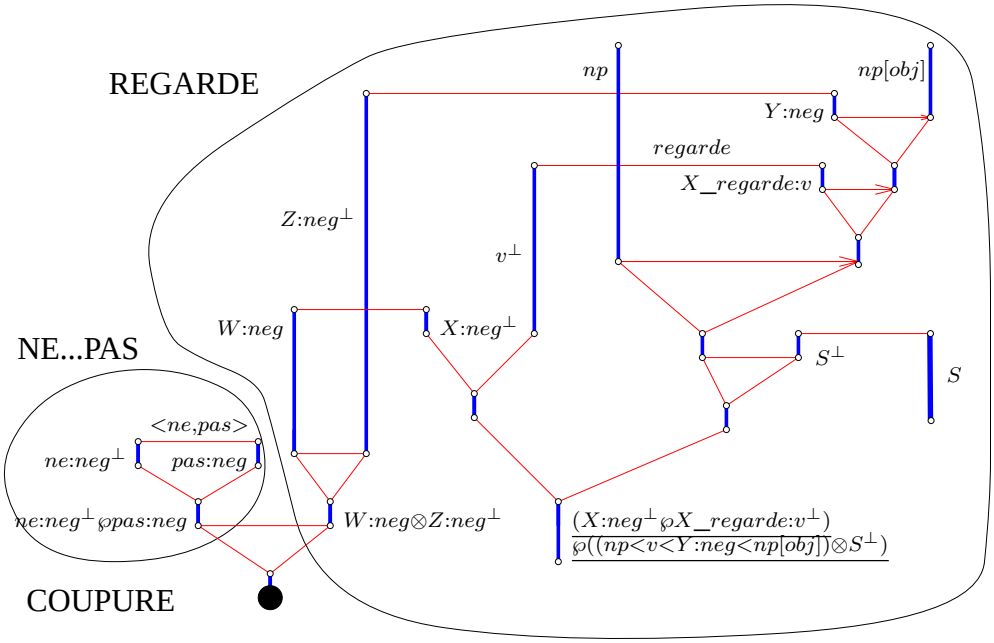
(8.8) *ne* regarde *pas*

8.3 Raffinements intuitionnistes

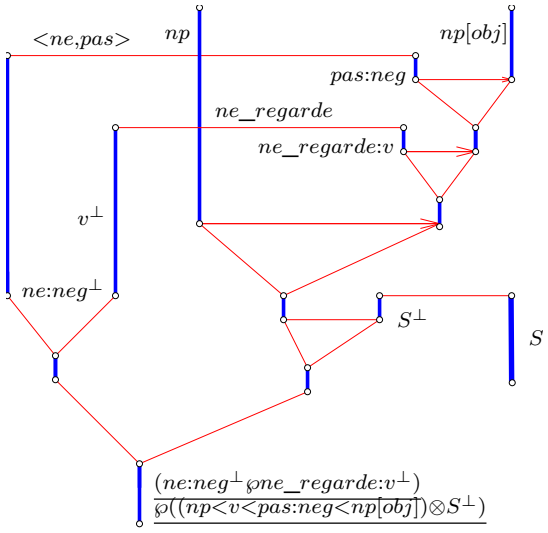
Nous avons ensuite restreint ce modèle en associant aux mots des modules que nous avons appelés intuitionnistes, parce qu'ils ont une conclusion privilégiée à laquelle le mot est attaché. Il y a plusieurs raisons à cela. D'une part la communauté linguistique s'insurgeait, et peut-être à juste titre, de la totale symétrie entre deux syntagmes composés. D'autre part l'ordre des mots, défini comme un ordre entre les axiomes n'était pas si facile à calculer. Pour l'analyse syntaxique, ce modèle devenait d'une complexité dramatique : comment engendrer à partir des modules associés aux mots de la phrase un réseau dont l'ordre des mots soit précisément celui de la phrase analysée? Cela est d'autant plus problématique que l'ordre entre axiomes résultant de la composition de deux modules n'est pas

FIG. 8.3 – *ne regarde pas*

– ne regarde pas (avant élimination des coupures)



– ne regarde pas (résultat de l'élimination des coupures)



une fonction simple des deux ordres entre axiomes que définissent les modules composés ? Finalement afin de pouvoir calculer des représentations sémantiques il était également impératif de retrouver une fonction et des arguments lors de la composition de modules.

Ces remarques nous ont conduit à modifier le modèle précédent de deux manières. D'une part on restreint la forme des modules associés aux mots, d'autre part on remplace l'ordre entre axiomes, assez difficilement calculable de manière incrémentale, par un étiquetage des nœuds du graphe par des ordres séries-parallèles, qui sont, compte tenu de la forme des modules associés aux mots, prévisibles lorsque l'on compose deux démonstrations partielles. L'ordre des mots correspondant à la phrase est l'étiquette de la sortie de la preuve partielle.

Décrivons ici la forme des modules associés aux mots :

- il y a deux conclusions :
  - b (sortie, type du syntagme résultant lorsque les arguments ont été fournis)
  - une conclusion de type

    $$a^{\perp} \wp (X_1 \otimes Y_1) \wp \cdots \wp (X_n \otimes Y_n)$$

    où aucun X_i ne contient de connecteur $\otimes$
- le module contient un axiome $a^{\perp} - a$ et a est dans l'un des X_i
- $a^{\perp}$ est étiqueté par le mot correspondant, et les formules $(X_i \otimes Y_i)$ sont étiquetées par l'ordre vide.

Voyons maintenant la propagation des étiquettes qui donnent l'ordre des mots.

- Les deux conclusions d'un lien axiome portent la même étiquette.
- Une des deux prémisses d'un lien $\otimes$ est la somme disjointe des deux ordres étiquetant respectivement l'autre prémisse et la conclusion de ce même lien.
- La conclusion d'un lien $\wp$ est la somme disjointe des ordres étiquetant les deux prémisses.
- La conclusion d'un lien $<$ est étiquetée par la somme ordinale des ordres étiquetant respectivement les deux prémisses.

On observera qu'en raison de la forme des modules choisis, et du critère de correction, les étiquettes sont totalement déterminées par le réseau, et aussi que les étiquettes n'interviennent en rien dans la construction du réseau.

L'adjonction dont nous avons déjà parlé au paragraphe précédent devient ici une réelle adjonction car en raison de la forme des modules il n'est plus possible qu'un module soit à la fois branché par coupure et par axiome à un même module.

8.4 Capacité générative

Comme on l'a dit ce modèle permet de décrire des constructions qui échappent au calcul de Lambek. Il est néanmoins difficile de déterminer exactement le pouvoir d'expression au sens usuel du terme, c'est-à-dire la classe des langages reconnus par de telles grammaires : en effet celles-ci reconnaissent des ordres partiels (séries-parallèles) sur les terminaux et les phrases reconnues sont toutes les projections de tels ordres. Néanmoins on peut aisément affirmer quelques propriétés : l'appartenance d'une phrase au langage est décidable, c'est en général un problème NP-complet (puisque cela inclut la recherche de démonstration dans MLL). Ce modèle est assez proches des LTAGs, et plus encore de leur présentation en Arbres de Démonstration Partiels de Joshi et Kulick [1,2,3]. Le branchement par axiomes correspond à la substitution et celui par coupure à l'adjonction. Cela a été montré très précisément par Sylvain Pogodalla [4]. Une sous classe de ce modèle, utilisant des modules d'une forme très particulière, correspond exactement aux LTAGs. Ce modèle décrit donc tous les langages légèrement contextuels (*mildly context-sensitive*).

8.5 Critiques et perspectives

8.5.1 Un modèle hybride

En suivant le parallèle établi au paragraphe 2.2.1 nous ne disposons ni d'une théorie représentationnelle, ni d'une théorie dérivationnelle. En effet on construit certes la structure syntaxique de l'énoncé par assemblage de modules, procédure dérivationnelle, mais encore faut-il que le résultat soit un réseau de démonstration, ce qui est une condition dans le style des théories représentationnelles. Cette critique renvoie à l'absence de calcul de séquents pour ce calcul. En effet si l'on disposait d'un calcul des séquents on aurait alors une théorie dérivationnelle, dans le même genre que les arbres de preuve partiels de Joshi. Les modules de base seraient des arbres de preuves partiels du calcul des séquents (avec des variables de contexte), la substitution correspondrait à l'ajout d'une démonstration au des-

[1] Aravind Joshi and Seth Kulick. Partial proof trees as building blocks for a categorial grammar. In Morrill and Oehrle [MO95], pages 138–149.

[2] Aravind Joshi and Seth Kulick. Partial proof trees, resource sensitive logics and syntactic constraints. In Retoré [Ret97a], pages 21–42.

[3] Aravind Joshi and Seth Kulick an Natasha Kurtonina. An LTAG perspective on categorial inference. In Moortgat [Moo01a], pages 90–105.

[4] Sylvain Pogodalla. Lexicalized proof-nets and TAGs. In Moortgat [Moo01a], pages 230–250.

sus d'une feuille d'une autre démonstration, et l'adjonction à l'insertion d'une démonstration partielle à la place d'un séquent.

8.5.2 Calcul des représentations sémantiques

Une autre critique est la suivante : nous clamons depuis le début de ce mémoire que l'un des atouts de cette famille de modèles est la correspondance aisément calculable entre analyse syntaxique et structure prédicative. Or dans ce modèle, si elle reste calculable, c'est tout à fait empiriquement : en polarisant le connecteur précède comme on le fait des connecteur multiplicatifs usuels, on peut calculer une représentation sémantique à la Montague. Mais si l'on compare avec le calcul de Lambek, c'est bien moins évident, et nous n'avons pas su formuler par un algorithme simple la translation des analyses syntaxiques en structure sémantique qu'il suffit de composer avec la représentation sémantique des mots pour obtenir la représentation sémantique de l'énoncé.

8.5.3 Lien avec la grammaire générative

Ce type de grammaires nous a ouvert d'autres pistes dont l'une seulement a été explorée. En étudiant les chemins des réseaux obtenus nous nous sommes rendus compte qu'on trouvait la trace des mouvements présents dans la grammaire générative, à ceci près que les condition d'économies telles celle qui impose à un constituant de se déplacer à l'emplacement le plus proche étaient assez délicates à exprimer. Comme à cette même époque Edward Stabler a introduit les grammaires minimalistes [1], cela nous a conduit à rendre compte de ces grammaires, et cette fois dans un calcul plus habituel (pour obtenir un modèle dérivationnel). C'est l'objet du chapitre qui suit.

8.5.4 Des formules aux réseaux

Une autre perspective ouverte parce genre de modèle est de retenir le principe d'associer des modules aux mots, sans forcément conserver la logique ordonnée. En particulier on devrait pouvoir retrouver les arbres de preuves partiels de Joshi [2], défini dans un système logique propre, à savoir le calcul partiellement commutatif de Philippe de Groote dont nous avant déjà parlé [3] au chapitre 6. En effet ce

[1] Edward Stabler. Derivational minimalism. In Retoré [Ret97a], pages 68–95.

[2] Aravind Joshi and Seth Kulick. Partial proof trees as building blocks for a categorial grammar. In Morrill and Oehrle [MO95], pages 138–149.

[3] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

système de Joshi souffre à ce jour de la bizarrerie suivante : les modules associés aux mots sont dans le calcul de Lambek (non commutatif) mais lorsqu'on adjoint un module à un autre la non commutativité peut être enfreinte. Un calcul mixte rendrait compte plus proprement de la présence de connecteurs commutatifs et non commutatifs.

8.5.5 Contraintes à distance

Finalement, avec le recul, je trouvais satisfaisant d'avoir des contraintes de précédence non forcément immédiates, comme dans les grammaires de dépendances non bornées, d'Alexandre Dikovsky [4]. Pour cela trouver un calcul des séquents correspondant serait vraiment une bonne chose, comme expliqué ci-dessus. Il semble que récemment Alessio Guglielmi avec ses calculs de structures [5,6], très proches et d'ailleurs issus de nos derniers travaux sur le calcul ordonné [7], ait un cadre général qui laisse quelque espoir. Il s'agit de calculs abstraits sur des ensembles structurés de formules, qu'il a défini pour des modèles logiques de la concurrence.

8.5.6 Élimination des coupures

Une des particularité de ce modèle est d'utiliser réellement le mécanisme d'élimination des coupures. En effet, dans le calcul de Lambek, on peut ne considérer que des démonstration sans coupures. Ici, s'il est vrai que les analyses syntaxiques sont des démonstrations sans coupures, l'assemblage des mots fait apparaître des démonstrations avec coupures, mais l'ordre entre les mots est alors bien incomplet, et c'est l'élimination des coupures qui produit l'ordre correct des mots.

---

[4] Alexandre Dikovsky. Polarized non-projective dependency grammars. In de Groote et al. [dGMR01], pages 139–157.

[5] Alessio Guglielmi. A calculus of order and interaction. Technical Report WV-99-04, Dresden University of Technology, 1999.

[6] Alessio Guglielmi and Lutz Straßburger. Non-commutativity and MELL in the calculus of structures. In L. Fribourg, editor, *CSL 2001*, volume 2142 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, 2001.

[7] Christian Retoré. Pomset logic as a calculus of directed cographs. In V. M. Abrusci and C. Casadio, editors, *Dynamic Perspectives in Logic and Linguistics: Proof Theoretical Dimensions of Communication Processes,Proceedings of the 4th Roma Workshop*, pages 221–247. Bulzoni, Roma, 1999. Also available as INRIA Rapport de Recherche RR-3714 http://www.inria.fr/.

Chapitre 9

Grammaires minimalistes catégorielles

Résumé : *Ce chapitre présente des travaux récents réalisés avec Alain Lecomte (Université Grenoble II)* [1,2,3]*. Il doit aussi beaucoup à Edward Stabler (University of California at Los Angeles) : bien qu'il n'ait pas participé directement aux nouveautés décrites dans ce chapitre, il est l'auteur de la formalisation du programme minimaliste de Chomsky que nous utilisons, et j'ai d'ailleurs écrit avec lui un article de synthèse sur la convergence entre grammaires minimalistes et logique des ressources* [4].

Le modeste résultat d'apprenabilité de ces grammaires [5] *doit beaucoup au travail de DEA de Roberto Bonato* [6].

---

[1] Alain Lecomte and Christian Retoré. Towards a minimal logic for minimalist grammars: a transformational use of Lambek calculus. In *Formal Grammar, FG'99*, pages 83–92. FoLLI, 1999.

[2] Alain Lecomte and Christian Retoré. A logical formulation of the minimalist program. In *Third Tbilisi Symposium on Language, Logic and Computation*. FoLLI, 1999.

[3] Alain Lecomte and Christian Retoré. Extending Lambek grammars: a logical account of minimalist grammars. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, ACL 2001*, pages 354–361, Toulouse, July 2001. ACL.

[4] Christian Retoré and Edward Stabler, editors. *Resource Logics and Minimalist Grammars*, European Summer School in Logic Language and Information, Utrecht, 1999. FoLLI. Forthcoming special issue of *Language and Computation*, featuring a twenty-page introduction by the editors (INRIA Research Report RR-3780).

[5] Roberto Bonato and Christian Retoré. Learning rigid lambek grammars and minimalist grammars from structured sentences. In Popelìnskỳ and Nepil [PN01], pages 23–34.

[6] Roberto Bonato. Uno studio sull'apprendibilità delle grammatiche di Lambek rigide — a study on learnability for rigid Lambek grammars. Tesi di Laurea & Mémoire de D.E.A, Università di Verona & Université Rennes 1, 2000.

9.1 Motivations

Comme expliqué dans les deux chapitres introductifs, nous pensons qu'à l'heure actuelle, dans le domaine de la syntaxe des langues, les travaux de la grammaire générative, conduits par Noam Chomsky et ses élèves, sont les plus développés et les plus convaincants. Leurs études couvrent maintenant de nombreuses langues, ce qui a conduit à de profondes modifications du modèle initial.

La substantielle réorganisation du modèle des années 90, appelée programme minimaliste [1], s'est rapprochée des grammaires catégorielles :

- Tous les principes du gouvernement et du liage doivent se déduire de deux opérations indépendantes de la langue considérée, et la variation grammaticale entre les langues est supposée être purement lexicale.
- Les deux opérations sont déclenchées par des paramètres de polarités opposées qui s'annulent.

L'opération appelée fusion (*merge*) ressemble beaucoup aux règles des grammaires AB, c'est-à-dire aux règles d'élimination des implications dans le calcul de Lambek. En revanche, il est plus délicat de rendre compte du déplacement (*move*).

Les propositions atomiques se trouvent considérablement enrichies par rapport aux grammaires catégorielles, non seulement en nombre, mais aussi dans leur nature même. On les regroupe sous le terme générique de **traits** (*features*) dont certains sont dits formels ou ininterprétables, et d'autres categoriels ; ces derniers correspondent bien sûr aux catégories élémentaires ou parties du discours dans une terminologie plus ancienne, et gèrent la consommation des valences, tandis que les premiers régissent les déplacements.

traits catégoriels

c complémenteur (*complementizer*) qui correspond au S des grammaires de Lambek

n nom commun

v verbe

d déterminant

…

traits fonctionnels

k,K le cas, qui peut être faible ou fort

wh interrogatif, qui peut être faible ou fort

…

---

[1] Noam Chomsky. A minimalist program for linguistic theory. In K. Hale and S.J. Keyser, editors, *The view from building 20*, pages 1–52. MIT Press, Cambridge, MA, 1993. (Reprinted in [Cho95]).

Une autre particularité des grammaires minimalistes est la lecture des chaînes qui superposent forme phonologique (PF) et forme logique (LF). En effet dans ce type de grammaire, ces deux aspects d'un constituant peuvent subir un traitement différent lors d'un déplacement. Soit les formes phonologique et logique d'un constituant se déplacent, soit une seule partie, en général la partie sémantique, se déplace, dans le but d'obtenir une forme logique acceptable.

Étant donné un constituant, sa forme logique sera notée *`constituant`* (en italique) et sa forme phonologique **`constituant`** (en gras), tandis que la superposition des deux sera notée ***`constituant`*** (en italique et en gras).

Nous rendons compte des déplacements par la répétition du constituant déplacé, comme dans [2] (*chains, copy theory*). La deuxième fois qu'un constituant est rencontré, il n'est pas lu : ceci n'est pas aussi extravagant qu'il y paraît, puisqu'une trace possède les mêmes caractéristiques que le constituant qui lui est associé. On aura donc des énoncés de la forme :

(9.1) [Quel livre] Chomsky a-t-il écrit [quel livre] ?

Comme les analyses fournissent une structure d'arbre aux mots, décrites par des crochets avec parenthésage implicite à droite[a], nous produirons des chaînes parenthésées, c'est-à-dire des arbres, dont les feuilles sont les formes phonologiques, logiques ou la supersposition des deux.

(9.2)	(chaîne)	[***c***	***g***]	[***c***	***g***]	[*p*	*l*]	***e***	[***p***	***l***]
	(PF)	[**c**	**g**]					**e**	**p**	**l**
	(LF)	[*c*	*g*]			[*p*	*l*]	*e*		

(9.3)	(chaîne)	[***q***	***g***]	[***q***	***g***]	[***m***	***l***]	***c***	[***m***	***l***]
	(PF)	**q**	**g**			[**m**	**l**]	**c**		
	(LF)	*q*	*g*			[*m*	*l*]	*c*		

9.2 Le principe du modèle

Pour rendre compte des déplacements (*move*) on peut utiliser l'introduction des implications (*hypothetical reasoning*) dans les grammaires multimodales de Michael Moortgat [3] comme l'ont fait Thomas Cornell [4] et Willemijn Vermaat

a. Une expression [***c g***] [***c g***] [*p l*] ***e*** [***p l***] se lit donc comme [[***c g***] [[***c g***] [[*p l*] [***e*** [***p l***]]]]]

[2] Michael Brody. *Lexico-logical form: a radically minimalist theory*. Number 27 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1994.

[3] Michael Moortgat. Categorial type logic. In van Benthem and ter Meulen [vBtM97], chapter 2, pages 93–177.

[4] Thomas Cornell. Derivational and representational views of minimalist transformational grammar. In Lecomte et al. [LLP99], pages 92–111.

[1]. Les modalités sont utilisées pour positionner convenablement les constituants déplacés et leurs types.

Notre modèle est bien plus simple, et l'idée de base peut se présenter dans le *calcul* de Lambek, sans même l'étendre, mais en définissant différemment la *grammaire* associée à ce calcul logique.

- La notion de lexique reste inchangée : à chaque mot est associé une ou plusieurs formules du calcul de Lambek *avec produit*.
- La production d'un énoncé est quasi inchangée : à partir d'un multi-ensemble de mots on construit une démonstration du symbole terminal c (au lieu de S) à partir du type des mots concernés, ces dernières étant vue comme des axiomes propres. En fait, en vertu des types présents dans le lexique, seules les règles d'élimination sont utilisées.
- La chaîne des mots se calcule à partir de la démonstration en étiquetant les sommets de la démonstration. *On observera que les dites étiquettes ne guident pas la démonstration mais sont simplement une information recalculable à partir de la démonstration.*
  - S'il s'agit d'un axiome propre associé à un mot, l'étiquette est le mot lui même.
  - S'il s'agit d'une hypothèse alors l'étiquette est une variable de chaîne — qui sera instanciée par la suite.
  - S'il s'agit de règles d'élimination des implications, alors la chaîne résultante est la concaténation des chaînes étiquetant les prémisses.

$$\frac{\Delta \vdash a : A \qquad \Gamma \vdash f : A \setminus B}{\Delta,\Gamma \vdash af : B} \qquad\qquad \frac{\Gamma \vdash f : B \,/\, A \qquad \Delta \vdash a : A}{\Delta,\Gamma \vdash fa : B}$$

  - S'il s'agit d'une règle d'élimination du produit, alors :
    1. On calcule l'étiquette e associée à la prémisse principale $A \bullet B$ de cette règle.
    2. On calcule l'étiquette c de la prémisse auxiliaire C de cette règle, qui fait intervenir x et y les étiquettes respectives de A et B, les deux hypothèses annulées par cette règle.
    3. La conclusion C de la règle reçoit alors l'étiquette $c(x := f(e),y := g(e))$ — les fonctions f et g seront décrites ci-après.

$$\frac{\Gamma \vdash e : A \bullet B \qquad \Delta,x : A,y : B,\Delta' \vdash c(x,y) : C}{\Delta,\Gamma,\Delta' \vdash c(x := f(e),y := g(e)) : C}$$

---

[1] Willemijn Vermaat. *Controlling Movement: Minimalism in a Deductive Perspective*. Doctorandus thesis, Universiteit Utrecht, 1999.

Les fonctions f et g permettent de distinguer la forme logique (ou sémantique) du constituant déplacé, de sa forme phonologique. Dans un déplacement visible (*overt move*), $f(\boldsymbol{e}) = g(\boldsymbol{e}) = \boldsymbol{e}$ tandis que dans un déplacement furtif (*covert move*), seule la partie sémantique est déplacée: $f(\boldsymbol{e}) = e$ et $g(\boldsymbol{e}) = \boldsymbol{e}$). Les déplacement visibles ont lieu lorsque le trait A est fort, et les déplacements furtifs lorsqu'il est est faible.

C'est ainsi que la variation entre langages est prise en compte. Par exemple, que ce soit pour un ordre SVO ou SOV, l'analyse restera la même, comme le montre les deux exemples parallèles suivants, l'un en latin et l'autre en français. C'est la différence de la force du trait $\bar{k}$ qui produit la différence dans l'ordre des mots.

entrée	abréviation	type
certains	c	$(\bar{k} \bullet d) \,/\, n$
plusieurs	p	$(\bar{k} \bullet d) \,/\, n$
langues	l	n
grammairiens	g	n
étudient	e	$(\bar{k} \setminus (d \setminus v)) \,/\, d$
(temps)		$(\bar{k} \setminus t) \,/\, v$
(comp)		$c \,/\, t$

entrée	abréviation	type
quidam	c	$(\bar{k} \bullet d) \,/\, n$
multas	p	$(\bar{k} \bullet d) \,/\, n$
linguas	l	n
grammatici	g	n
cognoscunt	e	$(\bar{K} \setminus (d \setminus v)) \,/\, d$
(temps)		$(\bar{k} \setminus t) \,/\, v$
(comp)		$c \,/\, t$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\bar{k} \bullet d) \,/\, n \quad n}{\bar{k} \bullet d}\,/_e \quad \dfrac{(\bar{k} \setminus (d \setminus v)) \,/\, d \quad [d]^1}{\bar{k} \setminus (d \setminus v)}\,/_e \quad [\bar{k}]^1}{(d \setminus v)}\setminus_e}{d \setminus v}\otimes_e^1}{v}\setminus_e}{(\bar{k} \setminus t) \,/\, v}\,/_e \quad [d]^2}{\bar{k} \setminus t}\setminus_e \quad [\bar{k}]^2}{t}\otimes_e^2}{d \otimes \bar{k}}\,/_e \quad n}{(\bar{k} \bullet d) \,/\, n}\,/_e \quad n}{t}\,/_e}{c \,/\, t}\,/_e$$

(9.4) (chaîne) [quidam grammatici] [quidam grammatici] [multas linguas] cognoscunt [multas linguas]
(PF) quidam grammatici [multas linguas] cognoscunt
(LF) quidam grammatici [multas linguas] cognoscunt
« Certains grammairiens étudient plusieurs langues »

(9.5) (chaîne) **[*certains grammairiens*] [*certains grammairiens*]** [*plusieurs langues*] **[*étudient*] [*plusieurs langues***]
(PF) **certains grammairiens** **étudient plusieurs langues**
(LF) [*Certains grammairiens*] [*plusieurs langues*] *étudient*
« Certains grammairiens étudient plusieurs langues »

L'unique différence est qu'en latin le cas accusatif fourni par le verbe conjugué est fort, ce qui n'est pas le cas en français.

Cette idée de départ amène quelques commentaires et critiques.

– L'utilisation des mots comme des axiomes propres est déplaisante, mais elle est nécessaire pour pouvoir annuler des hypothèses A et B séparées par des mots, comme on vient de le faire. Cela nous a conduit à utiliser un calcul plus souple, le calcul mixte du chapitre 6, comme on le verra ci-après.
– Les mouvements ne peuvent se croiser, alors que cela peut arriver, notamment dans le cas de mouvement de tête. Le calcul mixte du chapitre 6 permet de répondre à ces objections.
– Un passage de la forme logique à la Chomsky à une représentation sémantique plus intelligible serait le bienvenu.

9.3 Raffinements

Pour remédier aux inconvénients susmentionnés, nous avons modifié cette première idée en utilisant le calcul mixte du chapitre 6 mais dans sa version où l'ordre sur les formules en hypothèse peut diminuer.

En utilisant le produit commutatif, il n'est plus nécessaire que les hypothèses A et B annulées lors d'une règle d'élimination du produit soient contiguës, mais ils suffit qu'elles soient jumelles dans l'ordre sur les hypothèses. Par affaiblissement de l'ordre on peut toujours se ramener à ce cas-là, y compris lorsqu'une hypothèse, par exemple correspondant à un mot, se trouve entre A et B dans l'arbre de déduction.

Le produit commutatif permet également de rendre compte des déplacements de tête, qui sont nécessaires pour traiter des langues VSO (langues sémitiques, par exemple).

Les types considérés pour traduire les grammaires minimalistes sont de la forme :

(9.6) $(F_2 \setminus (F_3 \setminus (\cdots \setminus (F_{n-1}(\backslash F_n \setminus (G_1 \otimes G_2 \otimes \cdots G_1 \otimes A)))))) \,/\, F_1$

On notera qu'on s'éloigne un peu de la formulation donnée par Edward Stabler des grammaires minimalistes dans [1]. Les catégories et traits formels ne sont plus ordonnées mais sont un simple ensemble, c'est du reste ce que propose Chomsky dans son article de 98 [2].

Il faut bien sûr limiter la machinerie logique pour circonscrire la partie nécessaire à la syntaxe des langues, dont on sait qu'elle est plus spécifique — par

[1] Edward Stabler. Derivational minimalism. In Retoré [Ret97a], pages 68–95.

[2] Noam Chomsky. Minimalist inquiries: the framework. Reading room copy, 1998.

exemple en termes de complexité algorithmique. Plutôt que d'utiliser des postulats sur les connecteurs à la manière des grammaires multimodales, nous contraignons la forme des démonstrations. Dans la mesure où la production d'énoncés se ramène à la construction d'une démonstration, on restreint considérablement la combinatoire des règles : c'est donc un gain algorithmique.

Outre la contrainte sur les types mentionnée ci-dessus, les contraintes sur la forme des démonstration sont les suivantes :

- À part les règles d'entropie, seules les règles d'élimination sont utilisées, ce qui en fait découle de la forme des types.
- La règle d'entropie est utilisée systématiquement après les règles d'élimination des implications de sorte que l'ordre sur le contexte est toujours vide.

$$\dfrac{\dfrac{\Gamma \vdash B \,/\, A \qquad \Delta \vdash A}{(\Gamma;\Delta) \vdash B}\,\backslash_e}{\Gamma,\Delta \vdash B}\,entropie$$

- A tout moment de la démonstration, on a au plus une occurrence d'une même hypothèse : la démonstration ne contient jamais de séquent de la forme $\Gamma,\tau,\Gamma',\tau,\Gamma'' \vdash U$
- Toute hypothèse active est annulée dès que possible :
  - Si l'on a établi un séquent : $\Gamma,T,\Gamma' \vdash T' \setminus C$
  - et que le lexique contient un séquent $\mathcal{S}\ \Delta \vdash T' \otimes T$ (qui correspond à un mot si $\Delta = \varepsilon$ ou, si T est lui-même un produit, à une nouvelle hypothèse, et que le lexique contient le type $T' \otimes T$)
  - alors il faut introduire une hypothèse T' afin d'annuler au plus vite T et T' avec $\mathcal{S}$ par la règle $\otimes_e$.

$$\dfrac{\dfrac{\dfrac{T' \vdash T' \qquad \Gamma,T,\Gamma' \vdash T' \setminus C}{(T';\{\Gamma,T,\Gamma'\}) \vdash C}\,\backslash_e}{T',\Gamma,T,\Gamma' \vdash C}\,entropie \qquad \Delta \vdash T' \otimes T}{\Delta,\Gamma,\Gamma' \vdash C}\,\otimes_e$$

On rend ainsi compte de la grammaticalité de 9.9 et de l'agrammaticalité de 9.7 et de 9.8 :

(9.7) *Qui$_1$ connais-tu lesquels$_2$ t$_1$ connaît t_2 ?

(9.8) *Lesquels$_2$ connais-tu qui$_1$ t_1 connaît t_2 ?

(9.9) Tu connais ceux_que$_i$ Marie connaît t$_i$.

Les déplacements de tête nécessitent des types plus complexes, et les catégories de base peuvent également être fortes ou faibles. De plus ces déplacements peuvent croiser les déplacements de syntagmes complets, mais ne peuvent se croiser entre eux. Pour en rendre compte, on associe aux verbes transitifs des langues VSO le type $V \otimes ((\bar{k} \setminus (d \setminus v)) \,/\, d)$.

entrée	abréviation	type
yondhoro	y	$V \otimes ((\bar{k} \setminus (d \setminus v)) \,/\, d)$
Latifa	L	$\bar{k} \otimes d$
Malik	M	$\bar{k} \otimes d$
(tense)	$((T \otimes ((\bar{K} \setminus t) \,/\, v)) \,/\, V)$	
(comp)	$((c \,/\, t) \,/\, T$	

(9.10)

yondhoro	Latifa	Malik
regarde	Latifa	Malik
Verbe	Sujet	Objet

« Latifa regarde Malik »

$$\dfrac{\overset{\text{h}'2}{(\bar{k} \backslash (d \backslash v)) / d} \qquad \overset{\text{h}'1}{d}}{\overset{\text{h}'1\ \text{h}'2}{\bar{k} \backslash (d \backslash v)}} /_e$$

$$\dfrac{\overset{\text{h}2}{\bar{k}} \qquad \overset{\text{h}'1\ \text{h}'2}{\bar{k} \backslash (d \backslash v)}}{\overset{\text{h}2\ [\text{h}'1\ \text{h}'2]}{d \backslash v}} \backslash_e$$

$$\dfrac{\overset{\boldsymbol{M}}{\bar{k} \otimes d} \qquad \overset{\text{h}2\ [\text{h}'1\ \text{h}'2]}{d \backslash v}}{\overset{M\ \text{h}'1\ \mathbf{M}}{d \backslash v}} \otimes_e$$

$$\dfrac{\overset{\text{h}'3}{d} \qquad \overset{M\ \text{h}'1\ \mathbf{M}}{d \backslash v}}{\overset{\text{h}'3\ M\ \text{h}'1\ \mathbf{M}}{v}} \backslash_e$$

$$\dfrac{\overset{\text{h}'4}{(\bar{K} \backslash t) / v} \qquad \overset{\text{h}'3\ M\ \text{h}'1\ \mathbf{M}}{v}}{\overset{\text{h}'4\ \text{h}'3\ M\ \text{h}'1\ \mathbf{M}}{\bar{K} \backslash t}} /_e$$

$$\dfrac{\overset{\text{h}3}{\bar{K}} \qquad \overset{\text{h}'4\ \text{h}'3\ M\ \text{h}'1\ \mathbf{M}}{\bar{K} \backslash t}}{\overset{\text{h}3\ \text{h}'4\ \text{h}'3\ M\ \text{h}'1\ \mathbf{M}}{t}} \backslash_e$$

$$\dfrac{\overset{\boldsymbol{L}}{\bar{k} \otimes d} \qquad \overset{\text{h}3\ \text{h}'4\ \text{h}'3\ M\ \text{h}'1\ \mathbf{M}}{t}}{\overset{\boldsymbol{L}\ \text{h}'4\ \boldsymbol{L}\ M\ \text{h}'1\ \mathbf{M}}{t}} \otimes_e$$

$$\dfrac{\overset{\varepsilon}{(c / t) / T} \qquad \overset{\text{h}4}{T}}{\overset{\varepsilon\ \text{h}4}{c / t}}$$

$$\dfrac{\overset{\varepsilon\ \text{h}4}{c / t} \qquad \overset{\boldsymbol{L}\ \text{h}'4\ \boldsymbol{L}\ M\ \text{h}'1\ \mathbf{M}}{t}}{\overset{[\varepsilon\ \text{h}4]\ \boldsymbol{L}\ \text{h}'4\ \boldsymbol{L}\ M\ \text{h}'1\ \mathbf{M}}{c}} /_e$$

$$\dfrac{\overset{\varepsilon}{(T \otimes ((\bar{K} \backslash t) / v)) / V} \qquad \overset{\text{h}1}{V}}{\overset{\varepsilon\ \text{h}1}{T \otimes ((\bar{K} \backslash t) / v)}}$$

$$\dfrac{\overset{\varepsilon\ \text{h}1}{T \otimes ((\bar{K} \backslash t) / v)} \qquad \overset{[\varepsilon\ \text{h}4]\ \boldsymbol{L}\ \text{h}'4\ \boldsymbol{L}\ M\ \text{h}'1\ \mathbf{M}}{c}}{\overset{\text{h}1\ [\varepsilon\ [\varepsilon \text{h}1]]\ \boldsymbol{L}\ [\varepsilon \text{h}1]\ \boldsymbol{L}\ M\ \text{h}'1\ \mathbf{M}}{c}} \otimes_e$$

$$\dfrac{\overset{\boldsymbol{y}}{V \otimes ((\bar{k} \backslash (d \backslash v)) / d)} \qquad \overset{\text{h}1\ [\varepsilon\ [\varepsilon \text{h}1]]\ \boldsymbol{L}\ [\varepsilon \text{h}1]\ \boldsymbol{L}\ M\ \text{h}'1\ \mathbf{M}}{c}}{\overset{[\varepsilon\ [\varepsilon\ \mathbf{y}]]\ \boldsymbol{L}\ [\varepsilon\ \mathbf{y}]\ \boldsymbol{L}\ M\ y\ \mathbf{M}}{c}} \otimes_e$$

(9.11)

(chaîne)	**yondhoro**	***Latifa***	**yondhoro**	***Latifa***	*Malik*	*yondhoro*	**Malik**
(PF)	**yondhoro**	**Latifa**					**Malik**
(LF)		*Latifa*			*Malik*	*yondhoro*	

Néanmoins comme on le voit cette solution conduit à des types complexes, dont la structure est plus délicate à appréhender. Ce n'est pas si gênant : en effet si certains linguistes restent attachés à ce type de déplacements, ils peuvent être simulés par des déplacements de constituants (ceux que nous avons précédemment décrits) dont une partie s'est déjà déplacée [1]. Ce genre de déplacements, appelés déplacement de résidus (*remnant movement*), ne pose pas de problème particulier à notre approche.

9.4 Extraction de représentations sémantiques

Comme on s'en doute, l'un des intérêts d'ainsi modéliser les grammaires minimalistes consiste à extraire une représentation sémantique des analyses syntaxiques, en se laissant guider par la structure déductive de l'analyse syntaxique. Une des difficultés provient du halo qui nimbe la forme logique dans les ouvrages de Chomsky, dont l'une des rares propriétés observables est qu'elle ordonne correctement les quantificateurs.

Malgré l'hostilité affirmée envers le λ-calcul par Noam Chomsky [2], si l'on remplace les parties sémantiques par des lambda termes, on voit que l'on peut systématiquement remplacer chaque règle de notre calcul syntaxique par une succession de quelques règles. Dans la théorie du minimalisme la forme logique est produite par une succession de déplacements furtifs.

La procédure est la suivante :

- On ne conserve que les feuilles ayant un contenu sémantique dans l'arbre d'analyse.
- En vertu de la successions d'applications à gauche, on est conduit à élever le type des constituants (*type raising*), et les branchements de l'arbre restreint aux feuilles sémantique pleines correspondent alors à autant de règles d'applications qui nécessite des montées de types (*type raising*).

---

[1] Edward Stabler. Remnant movement and structural complexity. In Gosse Bouma, Erhard Hinrichs, Geert-Jan M. Kruijff, and Richard Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Sema ntics*, pages 299–326. CSLI, 1999. distributed by Cambridge University Press.

[2] Noam Chomsky. *Langue, Linguistique, Politique – dialogues avec Mitsou Ronat*. Flammarion, 1991.

9.5 Acquisition de grammaires minimalistes à partir d'énoncés structurés

A l'occasion de son mémoire de DEA [1] Roberto Bonato a étendu aux grammaires de Lambek l'algorithme de Wojciech Buszkowski et Gerald Penn [2] d'apprentissage des grammaires AB. Surtout, il a considérablement simplifié la preuve de convergence au sens de Gold [3] de ce genre d'algorithmes, initialement due à Makoto Kanazawa [4].

On rappelle qu'un algorithme d'apprentissage est une fonction qui associeune grammaire à un ensemble fini d'énoncés , et cette fonction est dite convergente au sens de Gold lorsque, quel que soit le langage de la classe à apprendre, et quelle que soit son énumération, il existe un entier à partir duquel la fonction d'apprentissage ne varie plus et associe aux exemples rencontrés une grammaire équivalente à celle ayant produit les exemples.

En suivant la présentation de Roberto Bonato, il n'a pas été bien difficile de traiter similairement des grammaires minimalistes. Si l'on suppose que l'on connaît la structure d'analyse des phrases, alors on peut inférer le type des mots. Néanmoins, comme nous utilisons un calcul qui contient à la fois des connecteurs commutatifs et des connecteurs non commutatifs, il n'y a plus d'unificateur le plus général, mais seulement un ensemble fini d'unificateurs minimaux, et l'algorithme doit choisir le bon unificateur minimal qui permet à la preuve de convergence de fonctionner. En cela il s'agit d'un algorithme non déterministe et nous espérons corriger rapidement ce défaut qui nous avait échappé.

On dira qu'une grammaire H est **subincluse** dans une grammaire G s'il existe une substitution qui, lorsqu'on l'applique à H ne produit que des assignation de types contenues dans G.

L'algorithme d'apprentissage des grammaire minimalistes à partir de squelettes d'arbres d'analyse fonctionne comme suit :

– On calcule le type principal des structures de démonstration données en exemples. Lorsqu'on rencontre une règle d'élimination du produit, il faut typer en premier la prémisse auxiliaire puis la prémisse principale $A \otimes B$ ou $A \bullet B$.

---

[1] Roberto Bonato. Uno studio sull'apprendibilità delle grammatiche di Lambek rigide — a study on learnability for rigid Lambek grammars. Tesi di Laurea & Mémoire de D.E.A, Università di Verona & Université Rennes 1, 2000.

[2] Wojciech Buszkowski and Gerald Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454, 1990.

[3] E. Mark Gold. Language identification in the limit. *Information and control*, 10:447–474, 1967.

[4] Makoto Kanazawa. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI, 1998. distributed by Cambridge University Press.

– Lorsqu'un ensemble d'exemples est effectivement produit par une grammaire, alors si on rassemble les types correspondants aux différents exemples il est possible de trouver un ensemble fini d'unificateurs minimaux — tels que tout autre unificateur se factorise par l'un de ces unificateurs.
– Pour l'une de ces substitutions, la grammaire obtenue par substitution est une grammaire rigide subincluse dans la grammaire recherchée.

Les lemmes suivants entraînent aisément la convergence :

Proposition 9.12 *1. Il n'y a qu'un nombre fini de grammaires subincluses dans une grammaire donnée.*

2. *Si G est subincluse dans G_0 alors les arbres d'analyse produits par G sont également produits par G_0.*
3. *Si tous les exemples de D sont des arbres d'analyse de G alors la grammaire $GF(D)$ obtenue par typage et collecte de types sur chacun des arbres de D est subincluse dans G, par une substitution σ*
4. *Si $GF(D)$ est subincluse dans G alors les arbres de D sont des arbres d'analyse de G.*
5. *On définit alors $RMG(D)$ comme suit. On calcule l'ensemble fini d'unificateurs minimaux modulo la commutativité et l'associativité de de $\bullet$ — l'existence et la calculabilité de cet ensemble fini d'unificateurs minimaux est dû à François Fages [5,6,7]. L'un d'entre eux correspond à une substitution σ_0 qui factorise τ du point 3 : $\sigma = \tau\sigma_0$. La grammaire associée par l'algorithme d'apprentissage est $RMG(D) = \sigma_0(GF(D))$.*
6. *Si $RMG(D)$ existe et est subincluse dans G alors les exemples structurés de D sont des arbres d'analyse de G.*
7. *Si les arbres d'analyse de D sont produits par la grammaire G alors $RMG(D)$ existe et $RMG(D)$ est subincluse dans G.*

Cette approche est très limitée, car l'algorithme proposé n'est pas déterministe (il faut choisir l'un des unificateurs minimaux), et fonctionne sur des données structurées qui sont en pratique inexistantes. Le principal résultat est donc un résultat d'apprenabilité. Ce premier pas suggère néanmoins des améliorations plus satisfaisantes d'un point de vue pratique, notamment parce que les types obtenus doivent avoir la forme donnée en 9.6.

Théorème 9.13 *La classe des grammaires catégorielles minimalistes est appre-*

---

[5] François Fages. Associative-commutative unification. In R. Shostak, editor, *Proceedings of the 7th International Conference on Automated Deduction, CADE*, volume 170 of *Lecture Notes in Computer Science*, pages 194–208. Springer-Verlag, 1984.

[6] François Fages and Gérard Huet. Complete sets of unifiers and matchers in equational theories. *Theoretical computer science*, 43(1):189–200, 1986.

[7] Claude Kirchner and Hélène Kirchner. Rewriting, solving, proving, 2001. forthcoming book available from http://www.loria.fr/ ckirchne.

nable par identification à la limite à partir de données structurées : l'algorithme donné précédemment converge.

9.6 Critiques et perspectives

9.6.1 Quel est l'intérêt d'une description logique ?

Si l'on croit que les logiques sensibles aux ressources sont un formalisme adapté au calcul, et que la syntaxe est effectivement un type particulier de calcul, alors il est naturel de chercher à rendre compte de théories linguistiques assez sophistiquées dans un cadre logique.

La principale critique que le logicien fera est sans doute la ressemblance entre notre modèle et une usine à gaz : obtenus par ajouts et restrictions ces modèles manquent d'élégance. Mais qui connaît une telle description de la syntaxe d'une langue ? Admet-elle une description simple et si possible, explicative ?

Malgré ce défaut on se satisfait tout de même d'avoir un très petit nombre d'opérations, même si certaines opérations correspondent à plusieurs règles logiques. On remarquera que la logique permet également d'exprimer les conditions de minimalité, comme des conditions sur les démonstrations qui viennent restreindre l'espace des recherches, ce qui est cohérent avec la présupposée efficacité des algorithmes d'analyse et de génération.

Plus concrètement, l'apport de la logique est manifeste dans le calcul de représentations sémantiques et les algorithmes d'apprentissage. Le len avec la sémantique prédicative provient de la similarité entre la structure d'analyse et le calcul des prédicats, qui peuvent toutes deux se voir comme des démonstrations formelles. Nous sommes certes bien loin d'une correspondance aussi aisée que dans le cadre du calcul de Lambek présenté au chapitre 7. Mais en poursuivant dans cette voie nous espérons que la correspondance s'éclaircira, et pour oser une critique, c'est déjà plus convaincant que le flou entourant la notion de forme logique dans les articles sur le programme minimaliste. En particulier la transformation présentée qui conduit toujours à des types élevées se simplifierait sans doute en utilisant une application inversée : $u(\lambda xt) \rightarrow t[x := u]$.

La possibilité de formaliser le processus d'acquisition grammaticale provient de celui des grammaires catégorielles, où la l'unfication de types est cruciale. Nous reviendrons dessus ci-après.

9.6.2 Réseaux de démonstration

Comme le lecteur familier de la logique linéaire l'aura sans doute remarqué, ce modèle est issu de celui développé au chapitre 8 où nous manipulions des

réseaux de démonstration. De plus le formalisme des réseaux de démonstration serait un réel gain pour les connecteurs utilisés ici, où, à cause des permutations possibles des règles d'élimination des produits $\otimes$ et $\bullet$, plusieurs déductions naturelles peuvent correspondre à une même analyse syntaxique — celle-ci se définissant par la consommation des traits présents dans les entrées lexicales. Définir une notion de réseaux de démonstration pour les calculs mixtes est donc une priorité, comme nous l'avions dit au chapitre 6.

9.6.3 Analyse syntaxique parallèle

Au chapitre 6 nous avons établi une correspondance fidèle entre l'exécution parallèle des réseaux de Petri et les démonstration du calcul mixte limité aux formules du premier ordre. Nous utilisons exactement ces même formules dans notre modélisation des grammaires minimalistes, mais avec le calcul dual où l'ordre entre les hypothèses peut croître au cours d'une démonstration. Si nous arrivions à établir une correspondance entre l'exécution des réseaux de Petri et l'analyse syntaxique dans les grammaires catégorielles minimalistes, alors nous serions ravis. En effet il est connu que les divers traits syntaxiques s'annulent en parallèle, mais jusqu'à ce jour aucun modèle formel n'en rend compte.

9.6.4 Apprentissage

Le petit résultat d'apprentissage mentionné ci-dessus n'est satisfaisant que par les perspectives qu'il ouvre. En effet aussi bien d'un point de vue pratique que d'un point de vue linguistique il est très critiquable.

D'un point de vue pratique, nous sommes encore très loin de pouvoir acquérir un lexique minimaliste à partir de corpus, et le non déterminisme ou la complexité de l'algorithme que nous avons évoqués ne sont pas les seules raisons. Les structures qu'on suppose sur les exemples sont bien trop proches de l'information que l'on souhaite acquérir, les types des mots, et de fait il n'y a pas de données de ce genre disponibles. De plus on voit mal comment en définir de manière automatique, à partir de ces corpus structurés existants, par exemple à partir d'arbres de dépendance. L'une des raisons principales est la présence d'éléments vides dans la grammaire générative, qu'elles soient des traces ou des constituants réellement vides tel le temps, le cas etc.

D'un point de vue linguistique, certains aspects concordent avec ce que l'on sait du mécanisme d'acquisition grammatical : nous n'utilisons que des exemples positifs, l'interface avec la sémantique pourrait être utilisée, etc. Mais les algorithmes proposés et fondés sur l'unification étendent à chaque exemple la grammaire apprise jusque là alors que ce n'est pas ainsi que fonctionne le mécanisme d'acquisition grammaticale, puisque l'instanciation de paramètres consiste plutôt

à restreindre la grammaire universelle en la spécialisant à la grammaire produisant la langue avec laquelle on est en contact. Lorsque notre modèle se précisera on pourra alors envisager de rendre compte de ce mode d'apprentissage ce qui conduira vraisemblablement à une révision du modèle de Gold utilisé pour la convergence des algorithmes d'inférence grammaticale.

Quatrième partie

Conclusion

Chapitre 10

État des lieux et perspectives

10.1 Graphes et logique linéaire

Concernant les réseaux de démonstration et les structures combinatoires, nous pensons que l'essentiel de notre objectif est atteint avec [1,2]. En effet, les structures décrites aux chapitres 3 et 4, relèvent enfin de la théorie des graphes la plus standard, et unifient les résultats du domaine. Les réseaux abstraits sont ultimes puisqu'ils identifient le plus possible de démonstrations équivalentes, ce qui est l'objectif de la syntaxe des réseaux de démonstration. De plus la structure combinatoire sous-jacente est extrêmement simple (un graphe muni d'un couplage parfait dont le complémentaire est un cographe), ainsi que le critère (tout cycle élémentaire alternant contient une corde). Cela ne signifie pas pour autant qu'il n'y ait plus rien à faire en ce domaine.

Concernant les réseaux bicolores avec liens, il faudrait les étendre à toute la logique linéaire. C'est en grande partie chose faite, puisque Sylvain Pogodalla a traité les modalités dans ce cadre [3]. Les calculs non commutatifs de Yetter et de Lambek ont également été décrits dans ce cadre par Elena Maringelli et Michele Abrusci [4,5]. Un de nos objectifs prioritaires est le traitement des calculs mixtes

[1] Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 1999. To appear.

[2] Christian Retoré. Handsome proof-nets: R&B-graphs, perfect matchings and series-parallel graphs. Rapport de Recherche RR-3652, INRIA, March 1999. http://www.inria.fr/.

[3] Sylvain Pogodalla. *Réseaux de preuve et génération pour les grammaires de types logiques*. Thèse de doctorat, spécialité informatique, INPL, Nancy, 2001.

[4] Elena Maringelli. *Grafi e logica lineare: una nuova definizione delle reti dimostrative non commutative*. Tesi di laurea, Università di Bari, December 1996. (Graphs and linear logic: a new definition of non-commutative proof-nets.).

[5] Vito Michele Abrusci and Elena Maringelli. A new correctness criterion for cyclic multiplicative proof-nets. In Retoré [Ret98].

comme celui de Philippe de Groote [1] que nous avons utilisé aux chapitres 6 et 9.

Les réseaux abstraits, que je qualifierais volontiers de « parfaits », nécessitent pour leur part d'être adaptés aux calculs non-commutatifs et mixtes. C'est une réelle priorité, car ces calculs sont ceux utilisés par les modèles grammaticaux. L'inclusion des modalités dans ce formalisme ne pourra malheureusement se faire qu'au détriment de la simplicité mathématique des réseaux abstraits multiplicatifs.

Quant au calcul ordonné, qui m'a beaucoup occupé, je pense que ce sont les calculs de processus d'Alessio Guglielmi [2] qui sont le plus à même d'en donner une description inductive. Le cas échéant, la souplesse de ce calcul, qui décrit l'ordre non immédiat entre les processus ou entre les constituants d'une phrase, est un réel atout, tant pour le parallélisme que pour les grammaires issues de ce calcul.

10.2 Modèles grammaticaux

L'approche catégorielle et logique qui est la nôtre n'est pas forcément la panacée. Les formalismes issus de la théories des langages classique et en particulier les grammaires d'arbres, que ce soit les TAGs d'Aravind Joshi [3,4], ou les grammaires minimalistes d'Edward Stabler [5] sont une réelle concurrence. En particulier ces formalismes offrent des algorithmes d'analyse syntaxique polynomiaux, tandis que les grammaires catégorielles sont généralement dans NP. De plus, ces formalismes sont moins contraignants et décrivent nombre de constructions syntaxiques délicates.

Les deux avantages principaux des grammaires logiques sont l'interface aisée avec la sémantique à la Montague, et l'existence d'algorithmes d'inférence grammaticale, qui échappent aux formalismes grammaticaux issus de la théorie des langages classique. Il convient donc d'étendre la couverture des formalismes catégoriels, mais aussi de le faire avec précaution : si nous perdons les avantages propres aux formalismes logiques, alors les grammaires d'arbres sont nettement plus intéressantes.

La voie que nous suivons est d'intégrer les travaux récents sur le minimalisme, car nous bénéficions ainsi de quarante ans de recherche sur la structure syntaxique

---

[1] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

[2] Alessio Guglielmi. A calculus of order and interaction. Technical Report WV-99-04, Dresden University of Technology, 1999.

[3] Aravind Joshi, Leon Levy, and Masako Takahashi. Tree adjunct grammar. *Journal of Computer and System Sciences*, 10:136–163, 1975.

[4] Anne Abeillé. *Les nouvelles syntaxes*. Armand Colin, 1993.

[5] Edward Stabler. Derivational minimalism. In Retoré [Ret97a], pages 68–95.

des langues, ce qui nous évite de développer un modèle pour découvrir *a posteriori* son incohérence linguistique. La volonté de traiter des principes communs aux langues humaines et de les spécialiser ensuite à telle ou telle langue est aussi un avantage de cette approche.

Pour ce faire nous devons consolider les atouts des grammaires catégorielles, par exemple simplifier le calcul de représentations sémantiques des grammaires minimalistes catégorielles, améliorer les algorithmes d'inférence grammaticale, et, ce qui semble possible au vu des types particuliers utilisés, donner un algorithme efficace d'analyse syntaxique — ou, mieux encore, donner une machine parallèle qui réalise l'analyse syntaxique., en exploitant notre travail sur les réseaux de Petri [6]

À travers les modèles décrits dans les deux derniers chapitres, se dégage des principes sur lesquels nos méthodes s'appuient, et qui persiterons même si les modèles varient. D'une part la logique (multiplicative) semble adéquate à décrire la composition des constituants d'un énoncé, ce qu'on appelait, dans les débuts de la grammaires générative, *la structure profonde*. D'autre part la structure logique de l'analyse syntaxique permet de reconstituer les déplacements de constituants, et de rejoindre la grammaire générative, notamment le programme minimaliste [7] tel qu'il est décrit par Edward Stabler [5,8] mais aussi l'interprétation sémantique, ce que nous faisons en étiquetant les formules apparaissant dans la démonstration formelle.

Comme on l'a dit, la structure du lexique autorise des algorithmes d'apprentissage, et ce sujet assez nouveau pour le traitement des langues nous semble très prometteur. D'un point de vue pratique, il augmente la robustesse de ce genre de grammaire, et, d'un point de vue théorique, il permet de voir à l'œuvre l'un des arguments-clefs de la grammaire générative. Sur ce sujet, afin d'étendre les réulstats désormais classiques de Buszkowski et Penn [9], de Kanazawa [10], plusieurs pistes se présentent. On peut étendre ces résultats à des formalismes plus riches, tels les grammaires minimalistes, mais aussi revoir le processus même d'appren-

---

[6] Christian Retoré. A description of the non-sequential execution of Petri nets in partially commutative linear logic. In Jan van Eijck, Vincent van Oostrom, and Albert Visser, editors, *Logic Colloquium '99 (Utrecht)*, Lecture Notes in Logic. Association for Symbolic Logic & A. K. Peters, Ltd, 2001. Complete version: RR-INRIA 4288 http://www.inria.fr/.

[7] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

[8] Edward Stabler. Remnant movement and structural complexity. In Gosse Bouma, Erhard Hinrichs, Geert-Jan M. Kruijff, and Richard Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Sema ntics*, pages 299–326. CSLI, 1999. distributed by Cambridge University Press.

[9] Wojciech Buszkowski and Gerald Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454, 1990.

[10] Makoto Kanazawa. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI, 1998. distributed by Cambridge University Press.

tissage. D'une part, même si les algorithmes sont relativement réalistes parce qu'ils n'utilisent que des exemples positifs, ils restent éloignés de la réalité parce qu'ils augmentent le langage engendré jusqu'au langage cible, alors qu'il faudrait plutôt qu'ils réduisent le langage jusqu'au langage cible. Procéder par instanciation de paramètres, comme cela est suggéré dans [Cho95] serait une meilleure approche. D'autre part le modèle de Gold n'est pas pleinement satisfaisant, parce qu'il ne donne pas de borne sur le nombre d'exemples nécessaires pour apprendre une grammaire donnée, et ne permet pas de considérer des approximations, ni de mesurer l'écart entre le langage cible et la grammaire produite. Finalement nous devons, comme certains ont commencé à le faire [1,2], prendre en compte les informations sémantiques dont on sait qu'elles sont nécessaire à l'acquisition de la syntaxe. Techniquement, les algorithmes existant gagneront sans doute en efficacité par l'utilisation d'une relation englobant unification et déduction, comme le fait Annie Foret [3] et en munissant les exemples de structures correspondant à des démonstrations d'une forme particulière comme celle définie par Yannick Le Nir [4].

Concernant la sémantique, un de nos souhaits les plus chers mais les plus prospectifs est de combiner sémantique prédicative et sémantique lexicale comme on le voit dans [5] mais en conservant une interface aisément calculable avec la syntaxe. Pour ce faire, l'utilisation de sémantique dénotationnelle semble une piste à explorer.

---

[1] Isabelle Tellier. Meaning helps learning syntax. In *Fourth International Colloquium on Grammatical Inference, ICG'98*, volume 1433 of *LNAI*, pages 25–36, 1998.

[2] Daniela Dudau-Sofronie, Isabelle Tellier, and Marc Tommasi. From logic to grammars via types. In Popelìnskỳ and Nepil [PN01], pages 35–46.

[3] Annie Foret. On mixing deduction and substitution in lambek categorial grammars. In de Groote et al. [dGMR01], pages 158–174.

[4] Yannick Le Nir. From proof trees in lambek calculus to ajdukiewicks bar-hillel elimination binary trees. *Language and Computation*, 2001. To appear.

[5] James Pustejovsky. *The generative lexicon*. M.I.T. Press, 1995.

Bibliographie

[Abe93] Anne Abeillé. *Les nouvelles syntaxes*. Armand Colin, 1993.

[Abr91] Vito Michele Abrusci. Phase semantics and sequent calculus for pure non-commutative classical linear logic. *Journal of Symbolic Logic*, 56(4):1403–1451, December 1991.

[Abr95] Vito Michele Abrusci. Non- commutative proof nets. In Girard et al. [GLR95], pages 271–296.

[AC96] Vito Michele Abrusci and Claudia Casadio, editors. *Third Roma Workshop: Proofs and Linguistics Categories – Applications of Logic to the analysis and implementation of Natural Language*. Bologna:CLUEB, 1996.

[AD94] Andrea Asperti and Giovanna Dore. Yet another correctness criterion for multiplicative linear logic with mix. In A. Nerode and Yu. Matiyasevich, editors, *Logical Fundations of Computer Science*, volume 813 of *Lecture Notes in Computer Science*, pages 34–46, St. Petersburg, July 1994. Springer Verlag.

[AJ94] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543 – 574, June 1994.

[Ajd35] Kazimierz Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935. (English translation in [McC67], pages 207–231).

[AL60] Antoine Arnauld and Claude Lancelot. *Grammaire générale et raisonnée*. Le Petit, 1660. Appelée *Grammaire de Port-Royal*. Rééditée en 1997 par les Éditions Allia.

[AM98] Vito Michele Abrusci and Elena Maringelli. A new correctness criterion for cyclic multiplicative proof-nets. In Retoré [Ret98].

[And92] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 1992.

[AR99] Vito Michele Abrusci and Paul Ruet. Non-commutative logic I: the multiplicative fragment. *Annals of Pure and Applied Logic*, 1999. http://www.uniroma3.it/reseaux9798.ps.

[Asp91] Andrea Asperti. A linguistic approach to dead-lock. Technical Report LIENS 91-15, Dép. Maths et Info, Ecole Normale Supérieure, Paris, October 1991.

[BD81] Marc Baratin and Françoise Desbordes. *L'analyse linguistique dans l'antiquité classique – I Les théories*. Klincksieck, 1981.

[BdG01] Guillaume Bonfante and Philippe de Groote. Stochastic lambek grammars. In *Formal Grammar – Mathematics of Language*, 2001.

[BdGR97] Denis Bechet, Philippe de Groote, and Christian Retoré. A complete axiomatisation of the inclusion of series-parallel partial orders. In H. Comon, editor, *Rewriting Techniques and Applications, RTA'97*, volume 1232 of *LNCS*, pages 230–240. Springer Verlag, 1997.

[Ber79] Claude Berge. *Graphes*. Gauthier-Villars, 1979.

[Bey98] Claire Beyssade. *Sens et savoirs — des communautés épistémiques dans le discours*. Langue et discours. Presses Universitaires de Rennes, 1998.

[BH53] Yehoshua Bar-Hillel. A quasi arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

[BL01] Daniela Bargelli and Joachim Lambek. An algebraic approach to french sentence structure. In de Groote et al. [dGMR01], pages 62–78.

[Bla94] Andreas Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56(1-3):183–220, 1994.

[BM01] Jérôme Besombes and Jean-Yves Marion. Identification of reversible dependency tree languages. In Popelìnskỳ and Nepil [PN01], pages 11–22.

[Bol78] Béla Bollobás. *Extremal Graph Theory*. Academic Press, 1978.

[Bon00] Roberto Bonato. Uno studio sull'apprendibilità delle grammatiche di Lambek rigide — a study on learnability for rigid Lambek grammars. Tesi di Laurea & Mémoire de D.E.A, Università di Verona & Université Rennes 1, 2000.

[BP90] Wojciech Buszkowski and Gerald Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454, 1990.

[BR01] Roberto Bonato and Christian Retoré. Learning rigid lambek grammars and minimalist grammars from structured sentences. In Popelìnskỳ and Nepil [PN01], pages 23–34.

[Bro94] Michael Brody. *Lexico-logical form: a radically minimalist theory*. Number 27 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1994.

[BS92] Gianluigi Bellin and P. J. Scott. On the π-calculus and linear logic (with an introduction by S. Abramsky). Technical report, University of Edimbourgh, 1992.

[Bus88] Wojciech Buszkowski. Generative power of categorial grammars. In R. Oehrle, E. Bach, and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*. Reidel, Dordrecht, 1988.

[Bus97] Wojciech Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [vBtM97], chapter 12, pages 683–736.

[Cho56] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, IT2:113–124, 1956.

[Cho63] Noam Chomsky. Formal properties of grammars. In *Handbook of Mathematical Psychology*, volume 2, pages 323 – 418. Wiley, New-York, 1963.

[Cho70] Noam Chomsky. Remarks on nominalization. In R. A. Jacob and P. S. Rosenbaum, editors, *Readings in English transformational grammar*, pages 184–221, Waltham, Mass., 1970. Ginn.

[Cho82] Noam Chomsky. *Some concepts and consequences of the theory of Government and Binding*. MIT Press, Cambridge, MA, 1982.

[Cho87] Noam Chomsky. *La nouvelle syntaxe*. Seuil, Paris, 1987. Traduction de [Cho82].

[Cho91] Noam Chomsky. *Langue, Linguistique, Politique – dialogues avec Mitsou Ronat*. Flammarion, 1991.

[Cho93] Noam Chomsky. A minimalist program for linguistic theory. In K. Hale and S.J. Keyser, editors, *The view from building 20*, pages 1–52. MIT Press, Cambridge, MA, 1993. (Reprinted in [Cho95]).

[Cho95] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

[Cho98] Noam Chomsky. Minimalist inquiries: the framework. Reading room copy, 1998.

[Cho01] Noam Chomsky. Beyond explanatory adequacy. Reading room copy, 2001.

[CL01] Claudia Casadio and Joachim Lambek. An algebraic analysis of clitic pronouns in Italian. In de Groote et al. [dGMR01], pages 110–124.

[Cor99] Thomas Cornell. Derivational and representational views of minimalist transformational grammar. In Lecomte et al. [LLP99], pages 92–111.

[Dan90] Vincent Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation et principalement du λ-calcul*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, juin 1990.

[Dar98] Philippe Darondeau. Deriving unbounded Petri nets from formal languages. Rapport de Recherche 3365, INRIA, février 1998. http://www.inria.fr/.

[Dem99] Akim Demaille. *Logiques linéaires hybrides et leurs modalités*. Thèse de doctorat, spécialité informatique, Ecole Nationale Supérieure des Télécommunications de Paris, juin 1999.

[dG95] Philippe de Groote. Linear logic with Isabelle: pruning the proof search tree. In *4^th Workshop on theorem proving with analytic tableaux and related methods*, Lecture Notes in Artificial Intelligence. Springer-Verlag, March 1995.

[dG96] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [AC96], pages 199–208.

[dG99a] Philippe de Groote. An algebraic criterion for intuitionnistic multiplicative proof nets. *Theoretical Computer Science*, 1999.

[dG99b] Philippe de Groote. A dynamic programming approach to categorial deduction. In *Conference on Automated Deduction, CADE'99*, Lecture Notes in Artificial Intelligence. Springer-Verlag, July 1999.

[DG01] Denys Duchier and Claire Gardent. Tree descriptions, constraints and incrementality. In H. Bunt, R. Muskens, and E. Thijsse, editors, *Computing Meaning, Volume 2*, Studies in Linguistics and Philosophy. Kluwer Academic Publishers, 2001.

[dGMR01] Philippe de Groote, Glyn Morrill, and Christian Retoré, editors. *Logical Aspects of Computational Linguistics, LACL'2001*, number 2099 in LNCS/LNAI. Springer-Verlag, 2001.

[dGR96] Philippe de Groote and Christian Retoré. Semantic readings of proof nets. In Geert-Jan Kruijff, Glyn Morrill, and Dick Oehrle, editors, *Formal Grammar*, pages 57–70, Prague, August 1996. FoLLI.

[Dik01] Alexandre Dikovsky. Polarized non-projective dependency grammars. In de Groote et al. [dGMR01], pages 139–157.

[dPR99] Valeria de Paiva and Eike Ritter, editors. *Linear logic and applications*, 1999. Dagstuhl seminar – forthcoming proceedings.

[DR89] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[DSTT01] Daniela Dudau-Sofronie, Isabelle Tellier, and Marc Tommasi. From logic to grammars via types. In Popelìnskỳ and Nepil [PN01], pages 35–46.

[EW90] Uffe Engberg and Glynn Winskel. Petri nets as models of linear logic. In *CAAP'90*, volume 431 of *LNCS*, pages 147–161. Springer, 1990.

[EW97] Uffe Engberg and Glynn Winskel. Completeness results for linear logic on Petri nets. *Annals of Pure and Applied Logic*, 86(2):101–135, 1997.

[Fag84] François Fages. Associative-commutative unification. In R. Shostak, editor, *Proceedings of the 7th International Conference on Automated Deduction, CADE*, volume 170 of *Lecture Notes in Computer Science*, pages 194–208. Springer-Verlag, 1984.

[FH86] François Fages and Gérard Huet. Complete sets of unifiers and matchers in equational theories. *Theoretical computer science*, 43(1):189–200, 1986.

[For01] Annie Foret. On mixing deduction and substitution in lambek categorial grammars. In de Groote et al. [dGMR01], pages 158–174.

[FR94] Arnaud Fleury and Christian Retoré. The mix rule. *Mathematical Structures in Computer Science*, 4(2):273–285, June 1994.

[Fra99] Nissim Francez. On fibring feature logics with concatenation logics. In Lecomte et al. [LLP99], pages 200–211.

[Gal86] Jean H. Gallier. *Logic for computer science*. Harper & Row Publishers, New York, 1986.

[Gam91] L. T. F. Gamut. *Logic, Language and Meaning – Volume 2: Intensional logic and logical grammar*. The University of Chicago Press, 1991.

[Geh92] Vijay Gehlot. *A proof-theoretic approach to semantics of concurrency*. PhD thesis, University of Pennsylvania, 1992.

[Gen34a] Gehrard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39:176–210, 1934. Traduction Française de R. Feys et J. Ladrière: Recherches sur la déduction logique, Presses Universitaires de France, Paris, 1955.

[Gen34b] Gehrard Gentzen. Untersuchungen über das logische Schließen II. *Mathematische Zeitschrift*, 39:405–431, 1934. Traduction française de J. Ladrière et R. Feys: Recherches sur la déduction logique, Presses Universitaires de France, Paris, 1955.

[GG89] Vijay Gehlot and Carl Gunter. Nets as tensor theories. In G. De Michelis, editor, *Applications of Petri nets*, 1989.

[GG90] Vijay Gehlot and Carl Gunter. Normal process representatives. In *L.I.C.S.*, 1990.

[Gir86a] Jean-Yves Girard. Multiplicatives. *Rendiconti del Seminario dell'Universitá é Politecnico Torino*, October 1986. Special issue on Logic and Computer Science.

[Gir86b] Jean-Yves Girard. The system F of variable types: Fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[Gir89] Jean-Yves Girard. Geometry of interaction, I: interpretation of system F. In *Proceedings of the ASL meeting held in Padova*, August 1989.

[Gir90] Jean-Yves Girard. Towards a geometry of interaction. In *Categories in Logic and Computer Science—Boulder, June 1987*, volume 92 of *Contemporary Mathematics*, pages 68–109. AMS, 1990.

[Gir91] Jean-Yves Girard. A new constructive logic: classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, November 1991.

[Gir95] Jean-Yves Girard. Linear logic: its syntax and semantics. In Girard et al. [GLR95], pages 1–42.

[Gir96] Jean-Yves Girard. Proof-nets: The parallel syntax for proof-theory. In A. Ursini and P. Agliano, editors, *Logic and Algebra*, number 150 in Lecture Notes in Pure and Applied Mathematics, pages 97–124. Marcel Dekker, Inc, New York, 1996.

[Gir97] François Girault. *Formalisation en logique linéaire du fonctionnement des réseaux de Petri*. Thèse de doctorat, spécialité informatique industrielle, Université Paul Sabatier, Toulouse, décembre 1997.

[Gir98] Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, 1998.

[Gir01] Jean-Yves Girard. Locus solum. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.

[GL95] L.R. Gleitman and M. Liberman, editors. *An invitation to cognitive sciences, Vol. 1: Language*. MIT Press, 1995.

[GLA92a] Georges Gonthier, Jean-Jacques Lévy, and Martin Abadi. The geometry of optimal lambda reduction. In *Principles Of Programming Languages*, pages 15–26, 1992.

[GLA92b] Georges Gonthier, Jean-Jacques Lévy, and Martin Abadi. Linear logic without boxes. In *Logic in Computer Science*, 1992.

[GLR95] Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors. *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes*. Cambridge University Press, 1995.

[GLT88] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[GN95] L.R. Gleitman and E.L. Newport. The invention of language by children: Environmental and biological influences on the acquisition of language. In Gleitman and Liberman [GL95], chapter 1, pages 1–24.

[Gol67] E. Mark Gold. Language identification in the limit. *Information and control*, 10:447–474, 1967.

[GS01] Alessio Guglielmi and Lutz Straßburger. Non-commutativity and MELL in the calculus of structures. In L. Fribourg, editor, *CSL 2001*, volume 2142 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, 2001.

[Gug94] Alessio Guglielmi. Concurrency and plan generation in a logic programming language with a sequential operator. In Pascal van Hentenryck, editor, *International Conference on Logic Programming*, pages 240–254, Genova, 1994. M.I.T. Press.

[Gug99] Alessio Guglielmi. A calculus of order and interaction. Technical Report WV-99-04, Dresden University of Technology, 1999.

[GW98] Claire Gardent and Bonnie Webber. Describing discourse semantics. In *Proceedings of the 4th TAG+ Workshop*, University of Pennsylvania, Philadelphia, 1998.

[Har51] Zellig Harris. *Methods in structural linguistics*. University of Chicago Press, Chicago, 1951.

[Hey99] Dirk Heylen. *Types and Sorts – Resource Logic for Feature Checkimg*. PhD thesis, Universiteit Utrecht, 1999.

[Hin94] Jaakko Hintikka. La vérité est-elle ineffable? In *« La vérité est-elle ineffable? » et autres essais*, collection Tiré à part, pages 9–47. Editions de l'éclat, 1994. Traduit de l'anglais par Antonia Soulez et François Schmitz.

[How80] William A. Howard. The formulae-as-types notion of construction. In J. Hindley and J. Seldin, editors, *To H.B. Curry: Essays on Combinatory Logic, λ-calculus and Formalism*, pages 479–490. Academic Press, 1980.

[HS96] Jaakko Hintika and Gabriel Sandu. Game-theoretical semantics. In van Benthem and ter Meulen [vBtM97], chapter 6, pages 361–410.

[Jac95] Ray Jackendoff. *The Architecture of the Language Faculty*. Number 28 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1995.

[Jac01] Evelyne Jacquey. *Ambiguïtés lexicales et traitement automatique des langues: modélisation de la polysémie logique et application aux déverbaux d'action ambigus en français*. Thèse de doctorat, spécialité informatique, Université Nancy 2, décembre 2001.

[Jan96] Theo M. V. Janssen. Compositionnality. In van Benthem and ter Meulen [vBtM97], chapter 7, pages 417–473.

[JaNK01] Aravind Joshi and Seth Kulick an Natasha Kurtonina. An LTAG perspective on categorial inference. In Moortgat [Moo01a], pages 90–105.

[JK95] Aravind Joshi and Seth Kulick. Partial proof trees as building blocks for a categorial grammar. In Morrill and Oehrle [MO95], pages 138–149.

[JK97] Aravind Joshi and Seth Kulick. Partial proof trees, resource sensitive logics and syntactic constraints. In Retoré [Ret97a], pages 21–42.

[JLT75] Aravind Joshi, Leon Levy, and Masako Takahashi. Tree adjunct grammar. *Journal of Computer and System Sciences*, 10:136–163, 1975.

[Joh98] Mark E. Johnson. Proof nets and the complexity of processing center-embedded constructions. In Retoré [Ret98], pages 433–447.

[JVSW91] Aravind Joshi, K. Vijay-Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, S. Schieber, and T. Wasow, editors, *Fundational issues in natural language processing*. MIT Press, 1991.

[Kan94] Max Kanovitch. Linear logic as a logic of computation. *Annals of pure and applied logic*, 67:183–212, 1994.

[Kan98] Makoto Kanazawa. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI, 1998. distributed by Cambridge University Press.

[Kay94] Richard Kayne. *The Antisymmetry of Syntax*. Number 25 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1994.

[KK86] William Kneale and Martha Kneale. *The development of logic*. Oxford University Press, 3rd edition, 1986.

[KK01] Claude Kirchner and Hélène Kirchner. Rewriting, solving, proving, 2001. forthcoming book available from http://www.loria.fr/ ckirchne.

[Kle52] Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

[KM97] Natasha Kurtonina and Michael Moortgat. Structural control. In P. Blackburn and M. de Rijke, editors, *Specifying Syntactic Structures*, pages 75–113. CSLI, 1997. Distributed by Cambridge University Press.

[Kot59] Anton Kotzig. Z teorie konečnỳch grafov s lineàrym faktorom II. *Matematicko-Fyzikàlny Časopis SAV*, IX(3):136 – 159, 1959. (On the theory of finite graphs with a linear factor II).

[Kün97] Luis Allan Künzle. *Raisonnement temporel basé sur les réseaux de Petri pour des systèmes manipulant des ressources*. Thèse de doctorat, spécialité informatique industrielle, Université Paul Sabatier, Toulouse, septembre 1997.

[Lam58] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[Lam61] Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, 1961.

[Lam90] John Lamping. An algorithm for optimal lambda-calculus reductions. In *Proceedings of the Seventeenth ACM Symposium on Principles of Programming Languages*, pages 16–30. ACM, ACM Press, January 1990.

[Lam94] François Lamarche. Proof nets for intuitionistic linear logic: Essential nets. 35 page technical report available by FTP from the Imperial College archives, April 1994.

[Lam99] Joachim Lambek. Type grammar revisited. In Lecomte et al. [LLP99], pages 1–27.

[Lam01] Joachim Lambek. Type grammars as pregroups. *Grammars*, 4(1):21–39, 2001.

[LLP99] Alain Lecomte, François Lamarche, and Guy Perrier, editors. *Logical aspects of computational linguistics: Second International Conference, LACL '97, Nancy, France, September 22–24, 1997; selected papers*, volume 1582. Springer-Verlag, 1999.

[LN01] Yannick Le Nir. From proof trees in lambek calculus to ajdukiewicks bar-hillel elimination binary trees. *Language and Computation*, 2001. To appear.

[Loa94] Ralph Loader. Linear logic, totality and full completeness. In *LICS: IEEE Symposium on Logic in Computer Science*, 1994.

[LP86] Làszlò Lovàsz and Michael David Plummer. *Matching Theory*, volume 121 of *Mathematics Studies*. North-Holland, 1986. Annals of Discrete Mathematics 29.

[LR95] Alain Lecomte and Christian Retoré. Pomset logic as an alternative categorial grammar. In Morrill and Oehrle [MO95], pages 181–196.

[LR96] Alain Lecomte and Christian Retoré. Words as modules and modules as partial proof nets in a lexicalised grammar. In Abrusci and Casadio [AC96], pages 187–198.

[LR97] Alain Lecomte and Christian Retoré. Logique des ressources et réseaux syntaxiques. In D. Genthial, editor, 4 ème conférence sur le Traitement automatique du langage naturel, TALN'97, pages 70–83, Grenoble, 1997.

[LR98] Alain Lecomte and Christian Retoré. Words as modules: a lexicalised grammar in the framework of linear logic proof nets. In Carlos Martin-Vide, editor, *Mathematical and Computational Analysis of Natural Language — selected papers from ICML'96*, volume 45 of *Studies in Functional and Structural Linguistics*, pages 129–144. John Benjamins publishing company, 1998.

[LR99a] Alain Lecomte and Christian Retoré. A logical formulation of the minimalist program. In *Third Tbilisi Symposium on Language, Logic and Computation*. FoLLI, 1999.

[LR99b] Alain Lecomte and Christian Retoré. Towards a minimal logic for minimalist grammars: a transformational use of Lambek calculus. In *Formal Grammar, FG'99*, pages 83–92. FoLLI, 1999.

[LR01] Alain Lecomte and Christian Retoré. Extending Lambek grammars: a logical account of minimalist grammars. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, ACL 2001*, pages 354–361, Toulouse, July 2001. ACL.

[Lyo68] John Lyons. *Introduction to theoretical linguistics*. Cambridge university press, 1968.

[Mal91] Bertil Malmberg. *Histoire de la linguistique de Sumer à Saussure*. Fondamental. Presses Universitaires de France, 1991.

[Mar96] Elena Maringelli. *Grafi e logica lineare: una nuova definizione delle reti dimostrative non commutative*. Tesi di laurea, Università di Bari, December 1996. (Graphs and linear logic: a new definition of non-commutative proof-nets.).

[McC67] Storrs McCall, editor. *Polish Logic, 1920-1939*. Oxford University Press, 1967.

[Mel88] Igor Mel'čuk. *Dependency syntax – theory and practice*. Linguistics. State University of New York Press, 1988.

[Mel97] Igor Mel'čuk. Vers une linguistique sens-texte. leçon inaugurale, collège de france, 1997. http://www.fas.umontreal.ca/ling/olst/melcuk/.

[Mil96] Dale Miller. Forum: a multiple conclusion specification language. *Theoretical Computer Science*, 165:201–232, 1996.

[MM91] Michael Moortgat and Glyn Morrill. Heads and phrases – Type calculus for dependency and constituent structure. Technical report, OTS, Utrecht, 1991.

[MO95] Glyn Morrill and Richard Oehrle, editors. *Formal Grammar*, Barcelona, August 1995. FoLLI.

[Möh89] Rolf H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, volume 255 of *NATO ASI series C*, pages 105–194. Kluwer, 1989.

[Moo88] Michael Moortgat. *Categorial Investigations*. Foris, Dordrecht, 1988.

[Moo96] Michael Moortgat. Categorial type logic. In van Benthem and ter Meulen [vBtM97], chapter 2, pages 93–177.

[Moo97] Richard Moot. Automated deduction for categorial grammar logics: the grail prover. In E. Kraak and R. Wassermann, editors, *ACCOLADE'97*. ILLC, Universiteit van Amsterdam, 1997.

[Moo98] Richard Moot. Grail: An automated proof assistant for categorial grammar logics. In R.C. Backhouse, editor, *Proceedings of the 1998 User Interfaces for Theorem Provers Conference*, pages 120–129, 1998.

[Moo01a] Michael Moortgat, editor. *Logical Aspects of Computational Linguistics, LACL'98, selected papers*, number 2014 in LNCS/LNAI. Springer-Verlag, 2001.

[Moo01b] Michael Moortgat. Structural equations in language learning. In de Groote et al. [dGMR01], pages 1–16.

[Mor94] Glyn V. Morrill. *Type Logical Grammar*. Kluwer Academic Publishers, Dordrecht and Hingham, 1994.

[Mor96] Glyn V. Morrill. Memoisation of categorial proof nets: parallelism in categorial processing. In Abrusci and Casadio [AC96].

[Mor00] Glyn Morrill. Incremental processing and acceptability. *Computational Linguistics*, 26(3):319–338, 2000. preliminary version: UPC Report de Recerca LSI-98-46-R, 1998.

[MTV93] Marcel Masseron, Christophe Tollu, and Jacqueline Vauzeilles. Generating plans in linear logic: I. actions as proofs. *Theoretical Computer Science*, 113:349–370, 1993.

[Par96] Barbara H. Partee. Montague grammar. In van Benthem and ter Meulen [vBtM97], chapter 1, pages 5–91.

[PCKGV99] Brigitte Pradin-Chézalviel, Luis Allan Künzle, F.rançois Girault, and Robert Valette. Calculating duration of concurrent scenarios in time Petri nets. *APII - Journal Européen des Systèmes Automatisés*, 33(8-9):943–958, 1999.

[Pen93] Mati Pentus. Lambek grammars are context-free. In *Logic in Computer Science*. IEEE Computer Society Press, 1993.

[Pin94] Steven Pinker. *The Language Instinct*. Penguin Science, 1994.

[Pin95a] Steven Pinker. Language acquisition. In Gleitman and Liberman [GL95], chapter 6, pages 135–182.

[Pin95b] Steven Pinker. Why the child holded the baby rabbits. In Gleitman and Liberman [GL95], chapter 5, pages 107–133.

[PN01] Luboš Popelìnskỳ and Miloslev Nepil, editors. *Proceedings of the third workshop on Learning Language in Logic, LLL 01*, number FI-MU-RS-2001-08 in FI MU Report series, Strabourg, September 2001. Faculty of Informatics – Masaryk University.

[Pog00a] Sylvain Pogodalla. Generation in the Lambek calculus framework: an approach with semantic proof nets. In *proceedings of NAACL 2000*, May 2000.

[Pog00b] Sylvain Pogodalla. Generation, Lambek calculus, montague's semantics and semantic proof nets. In *proceedings of Coling 2000*, August 2000.

[Pog00c] Sylvain Pogodalla. Generation with semantic proof nets. Research Report 3878, INRIA, January 2000. http://www.inria.fr/.

[Pog01a] Sylvain Pogodalla. Lexicalized proof-nets and TAGs. In Moortgat [Moo01a], pages 230–250.

[Pog01b] Sylvain Pogodalla. *Réseaux de preuve et génération pour les grammaires de types logiques*. Thèse de doctorat, spécialité informatique, INPL, Nancy, 2001.

[Pol97] Jean-Yves Pollock. *Langage et cognition: le programme minimaliste de la grammaire générative*. Presses Universitaires de France, Paris, 1997.

[PS87] Carl Pollard and Ivan A. Sag. *Head Driven Phrase Structure Grammar*. Center for the Study of Language and Information, Stanford, CA, USA, 1987. (distributed by Cambridge University Press).

[Pus95] James Pustejovsky. *The generative lexicon*. M.I.T. Press, 1995.

[Rad97] Andrew Radford. *Syntactic theory and the structure of English — a minimalist approach*. Cambridge University Press, 1997.

[Ram96] Jean-Xavier Rampon. *Structures et algorithmiques des ensembles ordonnés*. Mémoire d'habilitation à diriger des recherches, Université de Rennes 1, 1996.

[RdR92] François Rivenc and Philippe de Rouilhan, editors. *Anthologie de Logique et Fondements des Mathématiques*. Editions Payot, 1992.

[Reg92] Laurent Regnier. *Lambda calcul et Réseaux*. Thèse de doctorat, spécialité mathématiques, Université Paris 7, Janvier 1992.

[Ret93] Christian Retoré. *Réseaux et Séquents Ordonnés*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, février 1993.

[Ret94a] Christian Retoré. On the relation between coherence semantics and multiplicative proof nets. Rapport de Recherche RR-2430, INRIA, décembre 1994. http://www.inria.fr/.

[Ret94b] Christian Retoré. A self-dual modality for "before" in the category of coherence spaces and in the category of hypercoherences. Rapport de Recherche RR-2432, INRIA, décembre 1994. http://www.inria.fr/.

[Ret94c] Christian Retoré. Une modalité autoduale pour le connecteur "précède". In Pierre Ageron, editor, *Catégories, Algèbres, Esquisses et Néo-Esquisses*, Publications du Département de Mathématiques, Université de Caen, pages 11–16, September 1994.

[Ret96a] Christian Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.

[Ret96b] Christian Retoré. Perfect matchings and series-parallel graphs: multiplicative proof nets as R&B-graphs. In J.-Y. Girard, M. Okada, and A. Scedrov, editors, *Linear'96*, volume 3 of *Electronic Notes in Theoretical Science*. Elsevier, 1996. (available from http://www.elsevier.nl/).

[Ret97a] Christian Retoré, editor. *Logical Aspects of Computational Linguistics, LACL'96*, volume 1328 of *LNCS/LNAI*. Springer-Verlag, 1997.

[Ret97b] Christian Retoré. Pomset logic: a non-commutative extension of classical linear logic. In Philippe de Groote and James Roger Hindley, editors, *Typed Lambda Calculus and Applications, TLCA'97*, volume 1210 of *LNCS*, pages 300–318, 1997.

[Ret97c] Christian Retoré. A semantic characterisation of the correctness of a proof net. *Mathematical Structures in Computer Science*, 7(5):445–452, 1997.

[Ret98] C. Retoré, editor. *Special Issue on Recent Advances in Logical and Algebraic Approaches to Grammar*, volume 7(4) of *Journal of Logic Language and Information*. Kluwer, 1998.

[Ret99a] Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 1999. To appear.

[Ret99b] Christian Retoré. Handsome proof-nets: R&B-graphs, perfect matchings and series-parallel graphs. Rapport de Recherche RR-3652, INRIA, March 1999. http://www.inria.fr/.

[Ret99c] Christian Retoré. Pomset logic as a calculus of directed cographs. In V. M. Abrusci and C. Casadio, editors, *Dynamic Perspectives in Logic and Linguistics: Proof Theoretical Dimensions of Communication Processes,Proceedings of the 4th Roma Workshop*, pages 221–247. Bulzoni, Roma, 1999. Also available as INRIA Rapport de Recherche RR-3714 http://www.inria.fr/.

[Ret00a] Christian Retoré. The logic of categorial grammars. Lecture notes, ESSLLI 2000, 2000. 68 page manuscript available from http://www.cs.bham.ac.uk/ esslli/.

[Ret00b] Christian Retoré. Systèmes déductifs et traitement des langues: un panorama des grammaires catégorielles. *Technique et Science Informatique*, 20(3):301–336, 2000. Numéro spécial *Traitement Automatique du Langage Naturel* sous la direction de D. Kayser et B. Levrat. Version préliminaire RR-3917 http://www.inria.fr/.

[Ret01] Christian Retoré. A description of the non-sequential execution of Petri nets in partially commutative linear logic. In Jan van Eijck, Vincent van Oostrom, and Albert Visser, editors, *Logic Colloquium '99 (Utrecht)*, Lecture Notes in Logic. Association for Symbolic Logic & A. K. Peters, Ltd, 2001. Complete version: RR-INRIA 4288 http://www.inria.fr/.

[Roo91] Dirk Roorda. *Resource logic: proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam, 1991.

[RS42] John Riordan and Claude E. Shannon. The number of two-terminal series-parallel networks. *Journal of Mathematical Physics, Massachussets Institute of Technology*, 21:83–93, 1942.

[RS99] Christian Retoré and Edward Stabler, editors. *Resource Logics and Minimalist Grammars*, European Summer School in Logic Language and Information, Utrecht, 1999. FoLLI. Forthcoming spe-

cial issue of *Language and Computation*, featuring a twenty-page introduction by the editors (INRIA Research Report RR-3780).

[Rue97] Paul Ruet. *Logique non-commutative et programmation concurrente*. Thèse de doctorat, spécialité logique et fondements de l'informatique, Université Paris 7, 1997.

[Sch96] Irene Schena. Pomset logic and discontinuity in natural language. In Retoré [Ret97a], pages 386–405.

[SM96] Jerry Sligman and Lawrence Moss. Situation theory. In van Benthem and ter Meulen [vBtM97], chapter 4, pages 239–309.

[Sta97] Edward Stabler. Derivational minimalism. In Retoré [Ret97a], pages 68–95.

[Sta98] Edward Stabler. Acquiring languages with movement. *Syntax*, 1:72–97, 1998.

[Sta99] Edward Stabler. Remnant movement and structural complexity. In Gosse Bouma, Erhard Hinrichs, Geert-Jan M. Kruijff, and Richard Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Sema ntics*, pages 299–326. CSLI, 1999. distributed by Cambridge University Press.

[Sto77] J. E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. The MIT Press, 1977. The MIT Press Series in Computer Science.

[TdF99] Lorenzo Tortora di Falco. *Réseaux, cohérence et expériences obsessionnelles*. Thèse de doctorat, spécialité logique et fondements de l'informatique, Université Paris 7, 1999.

[Tel98] Isabelle Tellier. Meaning helps learning syntax. In *Fourth International Colloquium on Grammatical Inference, ICG'98*, volume 1433 of *LNAI*, pages 25–36, 1998.

[Tes59] Lucien Tesnière. *Éléments de syntaxe structurale*. Éditions Klincksieck, 1959. 5^{e} édition: 1988.

[Tho74] Richmond Thomason, editor. *The collected papers of Richard Montague*. Yale University Press, 1974.

[Tie99] Hans-Jörg Tiede. *Deductive Systems and Grammars: Proofs as Grammatical Structures*. PhD thesis, Illinois Wesleyan University, 1999.

[Tie01] Hans-Jörg Tiede. Lambek calculus proofs and tree automata. In Moortgat [Moo01a].

[Tro92] Anne Sjerp Troelstra. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. CSLI, 1992. (distributed by Cambridge University Press).

[Uri98] Juan Uriegereka. *Rhyme and reason: an introduction to minimalist syntax*. MIT Press, 1998.

[vB91] Johan van Benthem. *Language in Action: Categories, Lambdas and Dynamic Logic*, volume 130 of *Sudies in logic and the foundation of mathematics*. North-Holland, Amsterdam, 1991.

[vBtM97] Johan van Benthem and Alice ter Meulen, editors. *Handbook of Logic and Language*. North-Holland Elsevier, Amsterdam, 1997.

[vEK96] Jan van Eijck and Hans Kamp. Representing discourse in context. In van Benthem and ter Meulen [vBtM97], chapter 3, pages 179–237.

[Ver99] Willemijn Vermaat. *Controlling Movement: Minimalism in a Deductive Perspective*. Doctorandus thesis, Universiteit Utrecht, 1999.

[vH85a] Jean van Heijenoort. Absolutism and relativism in logic. In *Selected Essays* [vH85c], pages 75–83.

[vH85b] Jean van Heijenoort. Logic as calculus and logic as language. In *Selected Essays* [vH85c], pages 11–16.

[vH85c] Jean van Heijenoort. *Selected Essays*. Bibliopolis, Napoli, 1985.

[WAL+87] Pierre Weis, Maria-Virginia Aponte, Alain Laville, Michel Mauny, and Ascànder Suárez. *The CAML Reference Manual*. INRIA-ENS, 1987.

[Yet90] David N. Yetter. Quantales and (non-commutative) linear logic. *Journal of Symbolic Logic*, 55:41–64, 1990.

Table des matières