# Génération à l'aide de réseaux de preuve sémantiques

Sylvain Pogodalla

# Génération à l'aide de réseaux de preuve sémantiques

Sylvain Pogodalla
Xerox Research Center Europe
6, chemin de Maupertuis
38240 Meylan
tél. : 04.76.61.51.76
Sylvain.Pogodalla@xrce.xerox.com

## 1 Introduction

La relation naturelle entre le calcul de Lambek et la sémantique de Montague a déjà été beaucoup étudiée pour l'analyse syntaxique. Paradoxalement, peu de choses ont été proposées pour la génération. En la comparant à une autre approche avec laquelle elle partage un certain nombre de traits formels (Merenciano & Morrill 1997), nous proposons une méthode préliminaire de génération basée sur les réseaux de preuve de la logique linéaire (Girard 1987).

En effet, il est connu que cette dernière, dans sa variante non commutative, contient le calcul de Lambek (Roorda 1991). D'autre part, ses preuves peuvent être exprimées au moyen d'une syntaxe particulière, les réseaux de preuve, qui quotiente les preuves inessentiellement différentes du calcul des séquents. La syntaxe des entrées lexicales peut alors s'exprimer dans ces réseaux.

Mais, tout comme la déduction naturelle pour la logique classique, ces réseaux de preuve permettent de coder des $\lambda$-termes. Aussi est-il possible, en s'inspirant de la sémantique de Montague, d'avoir à la fois des réseaux syntaxiques et des réseaux sémantiques attachés aux entrées lexicales.

Cet aspect, déjà développé pour l'analyse (de Groote & Retoré 1996), nous en proposons ici une utilisation pour la génération. Nous nous intéressons au problème de vérifier qu'avec des items lexicaux donnés, il est possible de générer une phrase syntaxiquement correcte ayant la forme sémantique donnée. Les propriétés calculatoires qui en découlent étant améliorées par rapport à l'approche de Merenciano & Morrill (1997). Par contre, la sélection des items lexicaux à employer reste en dehors de cette étude.

Le premier paragraphe rappelle brièvement les notations du calcul de Lambek et des réseaux de preuve que nous utiliserons par la suite. Le second paragraphe présente, sur un exemple, la méthode développée par Merenciano & Morrill (1997). Le dernier paragraphe introduit la notion de réseau sémantique et en montre l'utilisation en analyse. Il se poursuit par un exemple de la méthode de génération que nous proposons et en discute les propriétés.

## 2 Calcul de Lambek et réseaux

Dans ce paragraphe, nous évoquons le calcul de Lambek, la logique linéaire et les réseaux de preuve surtout pour en présenter les syntaxes que nous utiliserons. Pour une présentation plus précise, nous renvoyons le lecteur à (Lambek 1958, Moortgat 1997) pour le calcul de Lambek et à (Girard 1987, Retoré 1996b) pour la logique linéaire et les réseaux de preuve.

Le système que nous utilisons est le système d'inférence proposé par Lambek (1958) dans le cas non commutatif, avec simplement les connecteurs \ et /. Le langage formel est défini par :

$$L ::= P | L/L | L\backslash L$$

(c'est-à-dire le calcul de Lambek sans la concaténation), où *P* désigne les types de base (*S, NP, N…*).

Ce calcul est une grammaire catégorielle dans laquelle les réductions admises sont les séquents valides du système de Lambek (voir Table 1 pour la définition de ces règles, où les majuscules grecques représentent des multi-ensembles ordonnés de formules, et les majuscules romaines des formules). Elles permettent d'exprimer dans une logique le principe de base des grammaires catégorielles, à savoir que la juxtaposition syntaxique d'un mot de type A/B et celle d'un mot de type B (dans cet ordre) produit (ou dérive) un mot de type A. De même, la juxtaposition d'un mot de type B et celle d'un mot de type B\A produit un mot de type A. On voit ici le rôle de la non-commutativité.

Table 1 : Règles du calcul de Lambek

| | |
|---|---|
| $\dfrac{}{A \Rightarrow A}\,\mathrm{Ax}$ | $\dfrac{\Gamma, A, \Gamma' \Rightarrow C \quad \Delta \Rightarrow A}{\Gamma, \Delta, \Gamma' \Rightarrow C}\,\mathrm{cut}$ |
| $\dfrac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow B / A}\,\mathrm{R/}$ | $\dfrac{\Gamma, A, \Delta \Rightarrow C \quad \Gamma' \Rightarrow B}{\Gamma, A / B, \Gamma', \Delta \Rightarrow C}\,\mathrm{L/}$ |
| $\dfrac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A \setminus B}\,\mathrm{R}\setminus$ | $\dfrac{\Gamma, A, \Delta \Rightarrow C \quad \Gamma' \Rightarrow B}{\Gamma, \Gamma', B \setminus A, \Delta \Rightarrow C}\,\mathrm{L}\setminus$ |

Le problème de l'analyse d'une phrase se pose alors de la manière suivante : étant donné un lexique où à chaque entrée est associée une catégorie, pour une phrase $m_1 \ldots m_n$ donnée et des types $t_1, \ldots, t_n$ associés aux mots $m_1, \ldots, m_n$, peut-on dériver le type *S* spécifique des phrases correctes ?

L'un des aspects qui nous intéresse ici est celui du lien de ce calcul avec la logique linéaire (Girard 1987). Le fait que le calcul de Lambek soit un fragment d'une variante non commutative de la logique linéaire a été établi par exemple dans (Roorda 1991, Lamarche & Retoré 1996, Retoré 1996a). Les connecteurs de la logique linéaire que nous utiliserons, seront le *par* (+), le *tenseur* (*), les deux *implications* (\ et /).

Dans le calcul de Lambek, les séquents n'ont qu'une formule à droite, et un multi-ensemble (ordonné) de formules à gauche, et pas de négation. Pour pouvoir manipuler les réseaux de preuve dans de telles logiques dites intuitionnistes, on utilise la notion de polarité (Danos 1990) : une polarité positive $()^\circ$ (sortie) et une polarité négative $()^\bullet$ (entrée). Les polarités positives et négatives correspondent respectivement à la conclusion de droite du séquent et à toutes les conclusions de gauche. Deux mêmes formules de polarités différentes sont duales.

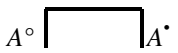
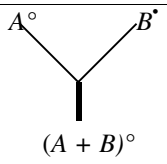
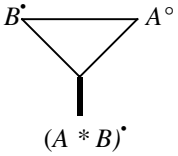
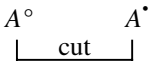
Mais l'intérêt de ce lien entre ces deux calculs tient entre autre à ce que l'on peut alors utiliser une syntaxe (initialement définie pour la logique linéaire) de réseaux de preuve pour ce calcul, ce qui permet, tout comme pour la logique linéaire, d'identifier les démonstrations inessentiellement différentes et d'éviter le problème des ambiguïtés fallacieuses (*spurious ambiguities*).

En considérant les liens de la Table 2, on définit les *préréseaux*, ou *structures de preuve*, comme des graphes constitués de ces liens tels que :

- chaque prémisse de chaque lien soit reliée à une et une seule conclusion d'un autre lien,
- chaque conclusion de chaque lien soit reliée à au plus une prémisse d'un autre lien.

On définit ensuite les réseaux comme des préréseaux vérifiant un critère de correction (par exemple il n'existe pas de cycle empruntant en alternance une arête grasse puis une arête fine et ne passant pas deux fois par le même sommet). Ces réseaux sont tels que leurs conclusions, sous forme de séquent, est prouvable dans le calcul des séquents. Un réseau définit une application $\xi$ sur ses atomes telle que $\xi(A) = B$ si et seulement si *A* et *B* sont reliés par un lien axiome.

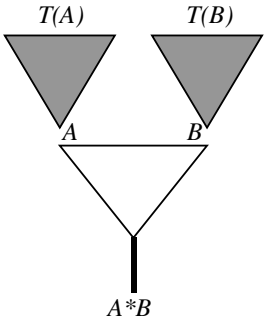
Table 2 : Définition des liens

| Nom | Axiome | + | * | Cut |
|---|---|---|---|---|
| Prémisses | Aucune | $A$ et $B$ | $A$ et $B$ | $A^\circ$ et $A^\bullet$ |
| Graphe |   |   |   |   |
| Conclusions | $A^\circ$ et $A^\bullet$ | $A+B$ | $A*B$ | aucune |

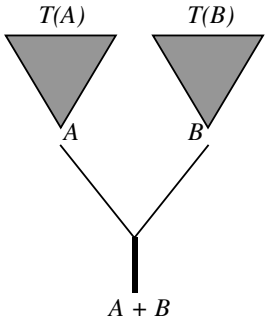
Nous définissons également l'arbre de sous-formules (on dit que l'on déplie l'arbre de sous-formules) d'une formule C comme le graphe défini inductivement comme suit :

- si C=α est atomique, alors T(C) est : α
- T (A*B) et T (A+B) sont définis à partir de T (A) et T (B) :

$T(A * B)$ :




T(A +B ) :




Bien entendu, pour pouvoir traiter de la non-commutativité, il est important de distinguer les prémisses droites et gauches de chaque lien. Au critère de correction habituel des réseaux s'ajoute alors celui de planarité : les liens axiomes ne doivent pas se couper. L'ordre des mots est alors considéré comme cyclique. Ainsi, le réseau de la Figure 1 est correct, alors que celui de la Figure 2 ne l'est pas. L'ordre des mots dans la phrase est de droite à gauche à partir de la conclusion positive $S^\circ$.

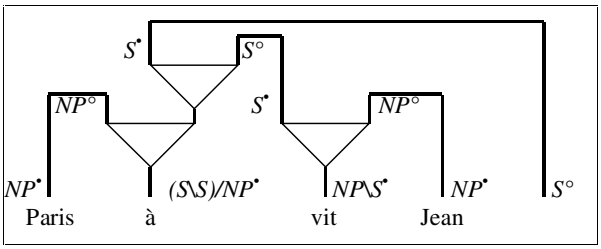



Figure 1 : Exemple de réseau correct

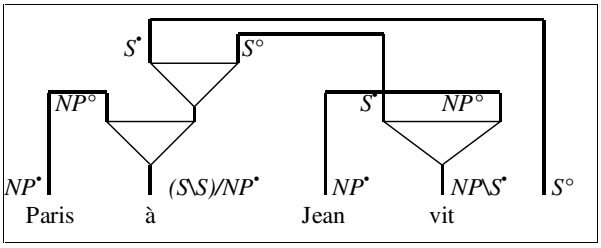
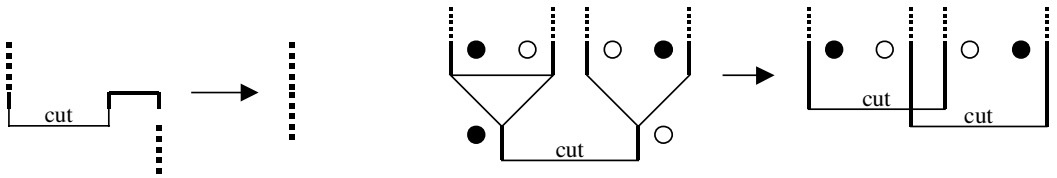


Figure 2 : Exemple de réseau incorrect du fait de l'ordre des mots

Pour ajouter à l'expression intrinsèque des réseaux, notons que la propriété fondamentale (pour une logique, mais aussi pour ce que nous présentons par la suite) de l'élimination des coupures peut se traduire directement sur le réseau, par simple réécriture :




## 3 Génération avec les réseaux étiquetés

Merenciano & Morrill (1997) ont proposé d'utiliser les propriétés déductives d'un système basé sur les réseaux de preuve pour faire de la génération dans un contexte liant calcul de Lambek et grammaires de Montague.

Le principe adopté est celui d'associer à chaque formule de type (ou de catégorie) un label consistant en une paire $(a, \alpha)$ où $a$ est la forme prosodique et $\alpha$ la forme sémantique ($\lambda$-terme). Les formules du réseau sont alors toutes étiquetées. En se restreignant au fragment implicationnel, on peut étiqueter les liens :

| | | | |
|---|---|---|---|
| $a-\alpha : A^\bullet \quad a+c-(\gamma\ \alpha) : B^\circ$ <br> $c-\gamma : A\backslash B^\circ$ | $c+a-(\gamma\ \alpha) : A^\circ \quad a-\alpha : B$ <br> $c-\gamma : A/B^\circ$ | $a+c-(\gamma\ \alpha) : B^\bullet \quad a-\alpha : A^\circ$ <br> $c-\gamma : A\backslash B^\bullet$ | $a-\alpha : B^\circ \quad c+a-(\gamma\ \alpha) : A^\bullet$ <br> $c-\gamma : A/B^\bullet$ |
| lien \° | lien /° | lien \• | lien /• |

On reconnaît ici les liens du calcul des séquents de la Table 1 transformés en réseaux en tenant compte de ce que les formules à gauche du signe => sont négatives, et celles à droite sont positives, et avec les relations entre par, tenseur et implication ($A\backslash B^\circ \equiv A^\bullet + B^\circ$, $A\backslash B^\bullet \equiv B^\bullet * A^\circ$, $A/B^\circ \equiv A^\circ + B^\bullet$, $A/B^\bullet \equiv B^\circ * A^\bullet$).

Bien sûr, le critère de correction énoncé au premier paragraphe reste valable. Néanmoins, G. Morrill et J. Merenciano préfèrent considérer le problème d'unification sur les labels $\cup\{(a=a', \alpha=\alpha') | a-\alpha : A^\circ \text{ et } a'-\alpha' : A^\bullet \text{ et } \xi(A^\circ)=A^\bullet\}$. Outre que la correction (c'est-à-dire la vérification qu'un préréseau est bien un réseau) est assurée par l'existence d'une solution à ce problème, cela évite lors de la recherche de preuve de construire tous les préréseaux et de tester *a posteriori* leur correction. Ici, la recherche de preuve est guidée par l'unification.

Cette procédure fonctionne aussi bien en analyse (unification des termes prosodiques) qu'en génération (unification des $\lambda$-termes). Nous donnons pour cette dernière l'exemple de la Figure 4 construit à l'aide du lexique de la Table 3 qui indique pour chaque entrée lexicale : la forme prosodique, la forme sémantique et la catégorie.

Table 3 : lexique

| | | |
|---|---|---|
| Jean – **j** | : | *NP* |
| Paris – **p** | : | *NP* |
| vit – **vit** | : | *N**S* |
| à – **a** | : | (*S**S*)/*N* |

La Figure 3 déplie les arbres des sous-formules associées à chacune des catégories des expressions. La partie recherche de preuve consiste alors à placer les liens axiomes entre les atomes duaux. Le principe utilisé par G. Morrill et J. Merenciano est, en partant depuis la forme sémantique d'entrée (ici (**a p**)(**vit j**) pour le symbole de la phrase correcte *S*°) de propager un système d'équations d'unification à résoudre, et sa résolution indiquera à la fois la correction du réseau et sa configuration finale. Ainsi, sur l'exemple, on part en fait d'un réseau sans lien axiome (Figure 3). Puis, on essaye de mettre le premier lien entre la sortie (conclusion positive, ici *S*°) étiquetée par la forme sémantique à exprimer, et un atome d'un arbre de sous formule. D'une part cet atome doit être de la même catégorie mais de polarité opposée (donc ici *S*•), et d'autre part pour savoir si ce lien a des chances d'être valide, on tente d'unifier les deux formes sémantiques. Ici, cela donne (**a p**)(**vit** j) = (**a** α)β (une autre possibilité, en considérant un lien avec l'autre atome S• aurait été d'essayer l'unification (**a p**)(**vit j**) = **vit** γ qui aurait tout de suite échouée ; ainsi on évite la construction de liens axiomes permettant des choses incorrectes).

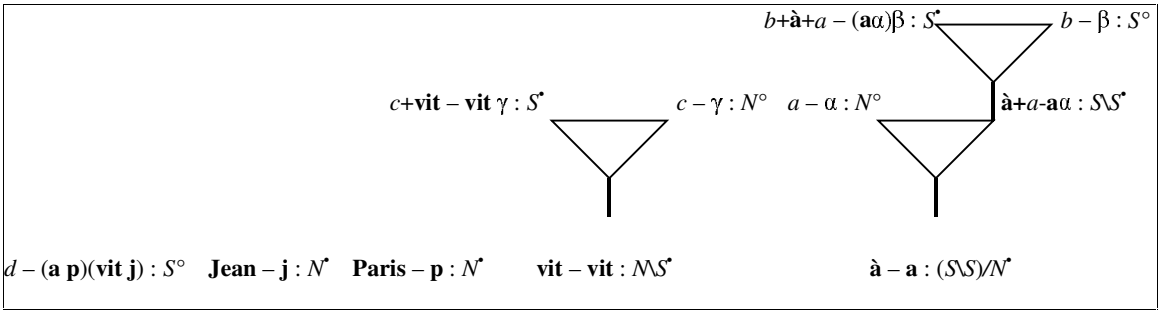



Figure 3 : Arbres des sous-formules

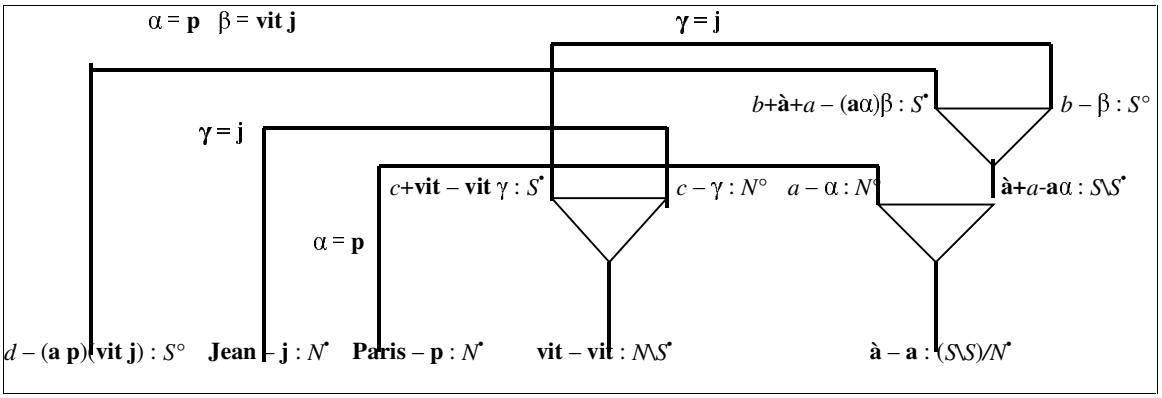



Figure 4 : Liens axiomes et unification des formes sémantiques

Sur l'exemple, on essaye d'unifier les formes sémantiques. Les différentes équations sont écrites Figure 4 au-dessus des liens axiomes qui les engendrent. On obtient dans ce cas un système unifiable, et les

liens axiomes proposés fonctionnent[1]. Cela permet, toujours par unification, de déduire la forme prosodique associée à la forme sémantique d'entrée.

En effet, on a les équations sur les formes prosodiques :

$c$ = **Jean**

$a$ = **Paris**

$b = c +$ **vit**

$e = b +$ **à** $+ a$

D'où finalement $e$ = **Jean vit à Paris**.

Néanmoins, l'unification de λ-termes, moteur de cette méthode, en fixe également les limites. En effet, comme le mentionnent eux-mêmes les auteurs, le problème de l'unification de λ-termes est en général difficile. S'il est décidable au premier ordre, au deuxième ordre cela n'est déjà plus le cas en général (Levy 1996). Or, avant d'arriver au problème d'unification lorsque tous les liens axiomes sont fixés, la méthode de recherche de preuve elle-même impose de résoudre une par une les équations d'unification. Cette résolution, dans le cas où elle serait impossible, peut empêcher la terminaison de l'algorithme d'unification. Mais même dans le cas où l'algorithme terminerait, le problème d'unification peut admettre une infinité de solutions incomparables (aucun des unifieurs n'étant plus général que les autres), ce qui empêche la propagation des solutions dans les équations à venir lors de l'étude des liens axiomes suivants.

Pour rester dans les termes manipulables (c'est-à-dire unifiables), il faut donc des λ-termes linéaires (chaque variable liée apparaît au plus une fois), avec variables libres apparaissant au plus deux fois dans le terme (mais au moins une fois), ce qui restreint le lexique utilisable.

Nous proposons donc une méthode de recherche de preuve également basée sur les réseaux mais pas sur les λ-termes, éludant la difficulté due à l'unification, tout en tachant d'éviter la construction de réseaux qui ne conduiraient pas à la forme sémantique choisie.

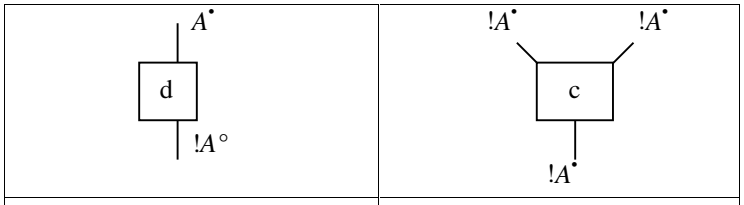
# 4 Génération par réseaux sémantiques

## 4.1 λ-termes et réseaux

Notre approche de la génération à l'aide des réseaux se base sur les travaux de de Groote & Retoré (1996). Ceux-ci montrent comment exprimer directement un λ-terme sous forme de réseau. Pour ceci, on utilise les réseaux intuitionnistes (avec les polarités) tels qu'ils ont été définis précédemment. Mais cette fois nous nous plaçons dans le cas commutatif ; et les connecteurs \ et / se fondent en un seul connecteur implicationnel →.

Nous ne détaillerons pas ici les méthodes pour coder un λ-terme dans un réseau ou pour lire un λ-terme depuis un réseau. Simplement, le principe est le même que lors de l'association des λ-termes aux formules de la déduction naturelle par l'isomorphisme de Curry-Howard (Girard, Lafont & Taylor 1988) ; on procède cette fois-ci avec les liens correspondant aux connecteurs de la logique linéaire multiplicative. Ce codage s'étend naturellement à la logique linéaire intuitionniste avec exponentielles. Ceci nous permettra de traiter le cas des λ-termes non linéaires. Cela nécessite l'ajout des deux liens : *contraction* et *déréliction.*

[1] Ici, la planarité n'est pas nécessaire, car la forme prosodique s'obtient directement dans les labels des liens du réseau (avec l'opération + sur les formes prosodiques). Ce sont les liens qui indiquent si la concaténation s'effectue dans un ordre ou dans l'autre.




Contrairement à Merenciano & Morrill (1997), le lexique se compose d'une forme prosodique et d'une catégorie, mais aussi d'un réseau sémantique. Ce dernier correspond à un λ-terme dont les types de base sont ceux des grammaires de Montague (Dowty, Wall & Peters 1981) : *e* et *t*. Les conclusions négatives (entrées) sont étiquetées par des constantes. Le type de l'unique conclusion positive (sortie) de chaque réseau sémantique est le type sémantique du mot. Ce dernier peut s'exprimer grâce à la règle de correspondance entre la composition syntaxique et la composition sémantique, résumée par l'homomorphisme suivant (bien défini si les types syntaxiques de base que l'on utilise sont *N*, *NP* et *S*) :

$$H(N) = (!e \rightarrow t) \qquad H(NP) = !e \qquad H(\mathrm{S}) = t$$
$$H(A\backslash B) = H(A) \rightarrow H(B) \qquad H(A/B) = H(B) \rightarrow H(A)$$

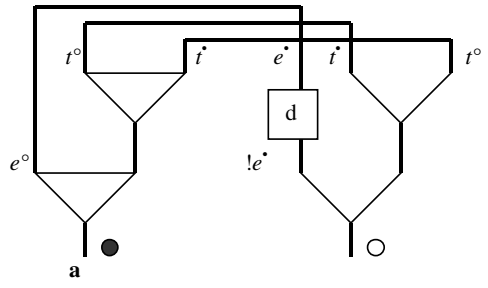
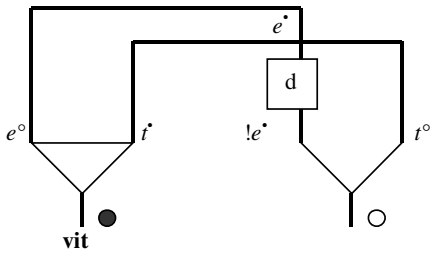
Ainsi, en reprenant l'exemple du paragraphe précédent, on a le lexique :

| Mot | Catégorie syntaxique | Catégorie sémantique | Réseau sémantique |
|---|---|---|---|
| Jean | *NP* | $!e$ | $\Pi_{\text{Jean}}$ |
| Paris | *NP* | $!e$ | $\Pi_{\text{Paris}}$ |
| vit | *NP**S* | $!e \rightarrow t$ | $\Pi_{\text{vit}}$ |
| à | (*S**S*)/*NP* | $(!e \rightarrow (t \rightarrow t))$ | $\Pi_{\text{à}}$ |

avec les types suivants pour les constantes :

**j** : $!e$ **p** : $!e$ **vit** : $e \rightarrow t$ **a** : $e \rightarrow (t \rightarrow t)$

Table 4 : Réseaux sémantiques des items lexicaux

| $\Pi_{\text{Jean}}$ | $\Pi_{\text{Paris}}$ |
|---|---|
|   |   |
| $\Pi_{\text{à}}$ | $\Pi_{\text{vit}}$ |
|   |   |

Afin d'expliquer plus facilement le processus de génération, nous donnons très rapidement celui de l'analyse. Comme toujours dans ce type de grammaire, l'analyse d'une phrase correspond à une preuve. Ici, il s'agit à partir du préréseau syntaxique de la Figure 5 (partie gauche) de trouver les liens axiomes qui garantissent la correction du graphe (avec la planarité) tel celui de la Figure 5 (partie droite) qui indique que

la phrase « Jean vit à Paris » est correcte (l'ordre des mots de la phrase est l'inverse de celui sur le graphe à cause d'une part de la non-commutativité, et d'autre part du fait que les réseaux représentent des séquents unilatères).

Ce réseau peut-être transformé, via l'homomorphisme *H*, en un réseau encore correct avec les types sémantiques. Dès lors, pour substituer à chacune des entrées sémantiques sa valeur (et donc son réseau sémantique), il suffit de faire un cut des entrées (conclusions négatives) avec chacun des réseaux sémantiques associé aux entrées lexicales et d'éliminer ce cut (Figure 6 et Figure 7). Par définition, ce cut est possible. On obtient alors un réseau qui est la forme sémantique de la phrase analysée (sa lecture donne bien comme λ-terme : (**a p**)(**vit j**)).

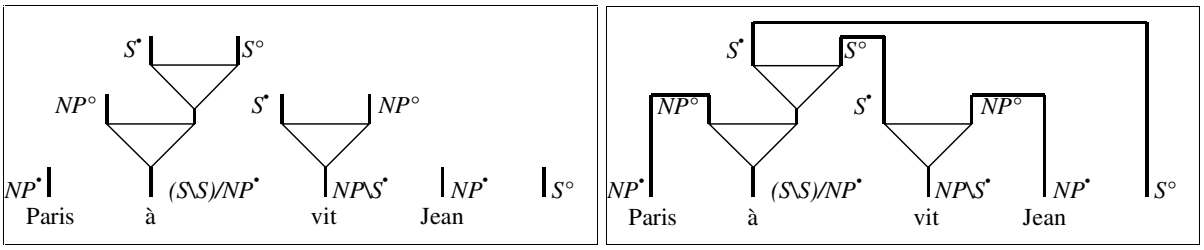



Figure 5 : Forêt syntaxique et réseau syntaxique

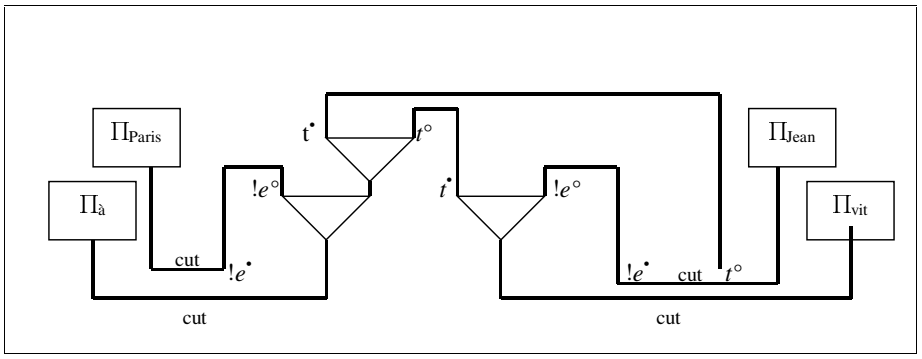



Figure 6 : Réseau sémantique (avant réduction)

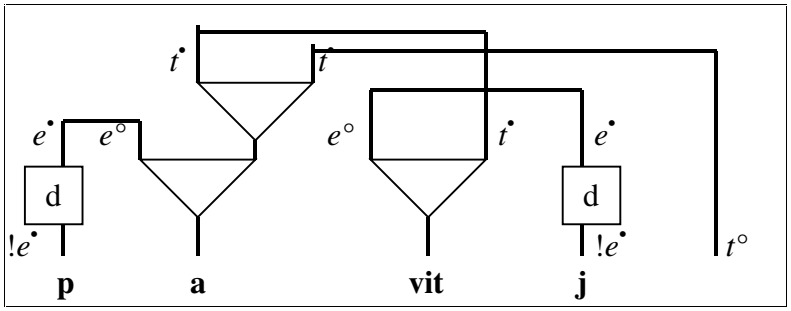



Figure 7 : Réseau sémantique (après réduction)

Ces figures illustrent l'analyse de la phrase « Jean vit à Paris ». Et ce dernier réseau (ou du moins ce type de réseau) sera le point de départ, la donnée à partir de laquelle commencera la procédure de génération (comme elle commençait, pour G. Morrill et J. Merenciano, par la donnée d'un λ-terme). Décrivons maintenant cette procédure.

## 4.2 Méthode de génération

La génération va s'attacher à faire le processus inverse de celui que l'on vient de décrire. Le problème peut alors s'énoncer ainsi : étant donnés un réseau $\Pi_{cible}$ (qui correspond à la forme sémantique de la phrase dont on cherche la forme prosodique) et un certain nombre de réseaux sémantiques du lexique[2], trouvons le réseau $\Pi$ qui a autant d'entrées que de réseaux sémantiques mis en jeux, dont la sortie soit bien typée (*t* dans le cas d'une phrase), planaire, et qui, lorsqu'on le lie par des cut aux réseaux sémantiques et qu'on élimine ces cut, donne $\Pi_{cible}$. Il est alors aisé, à partir d'un tel réseau, de déduire un réseau syntaxique pour la phrase.

Quant au réseau $\Pi$ lui-même, nous avons beaucoup d'informations pour le construire. En effet, la structure des arbres des sous-formules est complètement déterminée par la donnée des items lexicaux : ce sont les images par *H* des catégories syntaxiques. On peut déplier les arbres des sous-formules des entrées et de la sortie de $\Pi$. Il ne reste alors qu'à lier les atomes duaux de manière à former un graphe correct et planaire.

Nous pouvons noter qu'à ce stade, dès lors que les réseaux sémantiques lexicaux à utiliser ont été donnés, le problème de trouver un réseau $\Pi$ correct, planaire, et qui, après coupure, donne $\Pi_{cible}$ est décidable (puisqu'on a au pire un nombre fini d'atomes à apparier). Il nous reste à donner un algorithme exploitant cette propriété et qui construise le plus directement le réseau voulu, sans avoir à construire et tester tous les réseaux possibles. Pour ceci, nous allons bien entendu utiliser $\Pi_{cible}$.

Illustrons cette description par l'exemple de la Figure 7. Cet exemple fait partie des cas simples ($\lambda$-termes linéaires, présence de variables libres). On peut bien entendu traiter le cas général, mais la résolution est plus complexe. Dans le cas présent, nous séparons d'une part la phase d'appariement des atomes (réalisée par indexation), et la phase d'ordonnancement des mots dans la phrase.

Lorsque les arbres des sous-formules ont été dépliés, il faut judicieusement apparier les atomes duaux.

- nous relions par des cut les arbres dépliés et les réseaux sémantiques pour obtenir $\Pi_{inter}$ (Figure 9)[3] ;
- nous indexons chaque atome de $\Pi_{cible}$ avec une fonction $\iota$ telle que pour tout atome X;Y de $\Pi_{cible}$, si $X \in \{Y; \xi(Y)\}$ alors $\iota(X) = \iota(Y)$ et $\iota(X) \neq \iota(Y)$ sinon (Figure 8) ;
- chaque atome de $\Pi_{cible}$ est également feuille de chacun des arbres de sous-formule des conclusions. Or ces arbres, sauf la sortie, sont tous étiquetés par une constante typée. Comme ces constantes se retrouvent également dans le réseau en construction, on peut faire correspondre chacun des atomes des feuilles des arbres syntaxiques de conclusion négative de $\Pi_{inter}$ à un atome de $\Pi_{cible}$ (indexé). On peut alors lui donner le même index, ainsi qu'au lien axiome dont il est conclusion. Puis, pour rejoindre les atomes à lier, on suit les liens axiomes indexés et les cut (selon le chemin déterminé par l'élimination des coupures si on voulait la réaliser) jusqu'à parvenir aux atomes non liés de $\Pi_{inter}$ ;
- enfin, nous pouvons indexer l'atome où l'on vient d'arriver par le même index que celui du lien axiome traversé. Pour chaque lien axiome de $\Pi_{cible}$ relié à deux conclusions distinctes de $\Pi_{cible}$, en partant de chacune des conclusions correspondantes dans $\Pi_{inter}$, on peut indexer deux atomes dans $\Pi_{inter}$ avec le même index. Il ne reste alors qu'à mettre un lien axiome entre ces deux atomes (voir Figure 11) pour obtenir $\Pi$.

[2] Rappelons que nous ne nous préoccupons pas de la manière dont le choix des items lexicaux à utiliser a été fait.

[3] Il est important de les déplier comme on le ferait dans le cas non commutatif puisqu'on revient ensuite à la syntaxe.

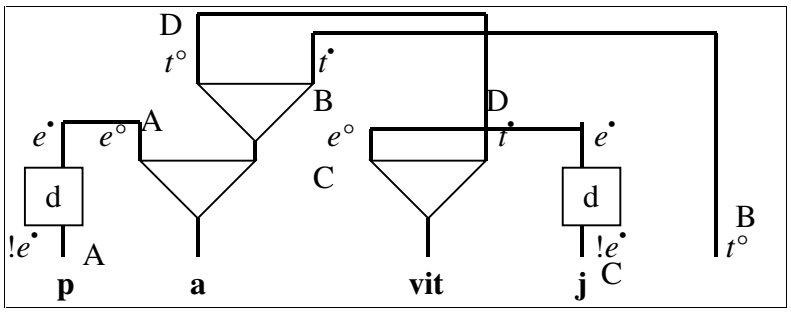



Figure 8 : Réseau cible indexé

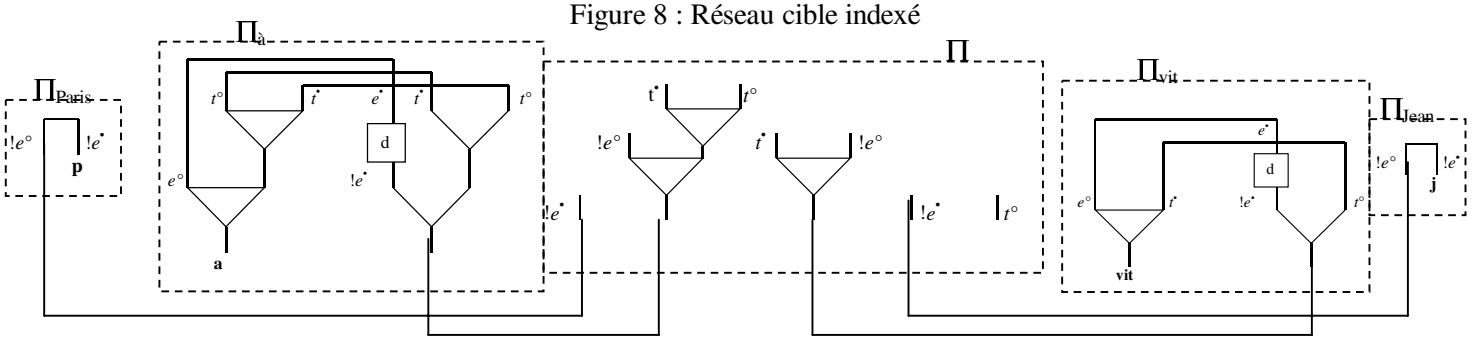



Figure 9 : Préréseau déplié

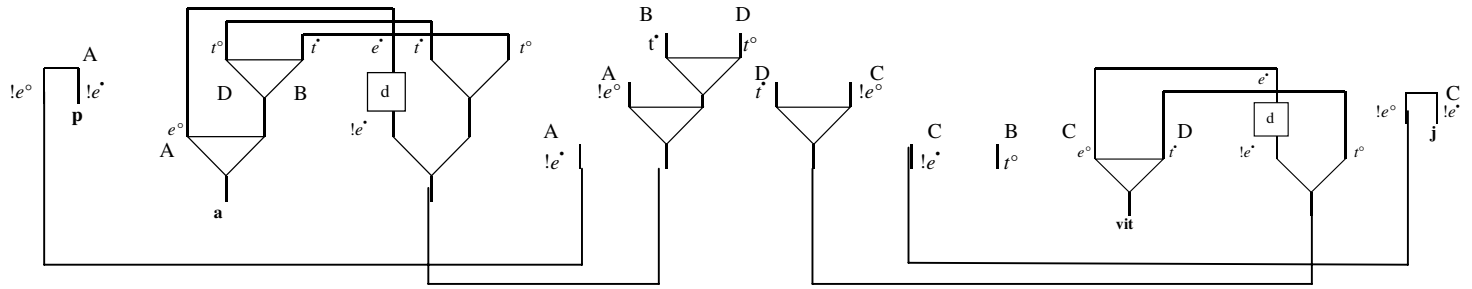



Figure 10 : Préréseau indexé

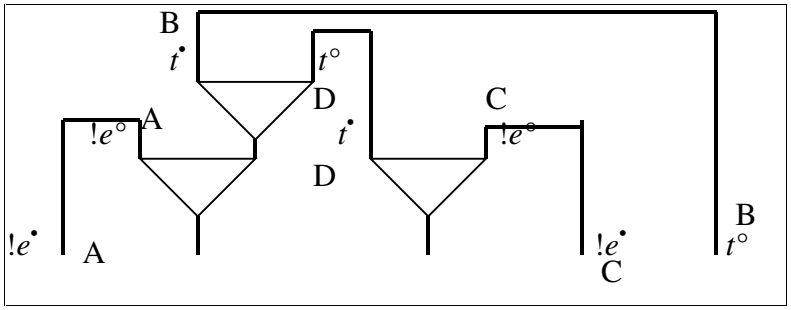



Figure 11 : Calcul et ajout des liens axiomes : Π

Lorsque les liens ont été définis, nous pouvons vérifier que ce réseau est correct dans le cas commutatif (cela se fait par exemple en temps quadratique avec un critère géométrique de recherche de cycle). Il reste alors à vérifier que l'on peut placer les mots de telle sorte qu'il reste correct dans le cas non commutatif. Pour ceci, il suffit de trouver un agencement des items lexicaux tels que les atomes forment un bon parenthèsage (avec X° et X• parenthèses). Cela correspond à ce que les liens axiomes ne se croisent pas.

Là encore, l'indexation des atomes nous suffit. En effet, nous nous trouvons (en considérant les index) avec cinq mots à concaténer de façon à ce que les atomes forment un bon parenthèsage (soit, dans l'exemple ci-dessus, les mots *ABD, C, DC, A* et *B*). Pour les placer les uns par rapport aux autres, nous prenons un de ces mots (de longueur strictement plus grande que un) au hasard (par exemple *DC*) et nous le plaçons sur un cercle (la description qui suit est illustrée Figure 12). Nous prenons ensuite un deuxième mot qui a au moins une lettre commune avec un des mots déjà placés (par exemple *C*) et nous le plaçons également sur le cercle. Nous considérons alors qu'un lien entre les deux *C* partage le plan en deux demi-plans. Si l'on veut ensuite ajouter le mot *ABD*, celui-ci doit alors se trouver dans le même demi-plan que le *D* déjà placé (sinon on a un croisement des liens axiomes). Un nouveau lien entre les deux *D* partage désormais le plan en trois, et les mots restant doivent être ajoutés de façon à ce que deux mêmes lettres se trouvent toujours dans la même partie du plan. Enfin, si on peut placer ainsi tous les mots, on a généré une phrase correcte (l'ordre des items lexicaux est lu dans l'ordre inverse des aiguilles d'une montre à partir du mot correspondant à l'unique conclusion positive (ici *B*), et l'ordre réel des mots dans la phrase est l'ordre inverse de celui des items lexicaux dans le réseau correct). Si on n'arrive pas à placer tous les mots, c'est qu'il n'existe pas de réalisation phonologique correspondant au réseau sémantique donné. Ici, on peut réaliser la phrase « Jean vit à Paris », correspondant à l'ordonnancement *B A ABD DC C* des items lexicaux qui réalise un bon parenthèsage.

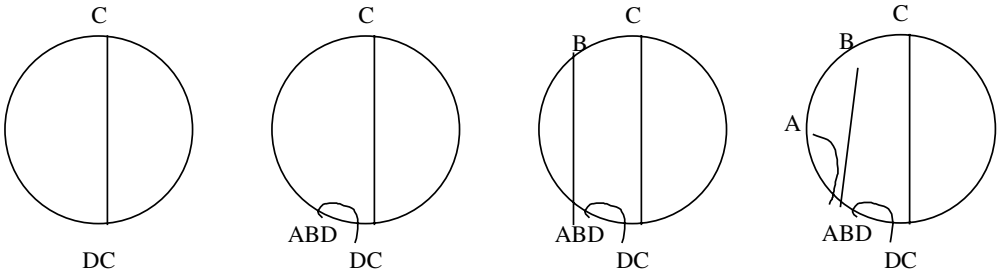



Figure 12 : Placement des mots les uns par rapport aux autres

Nous n'avons pas encore établi la complexité de cet algorithme dans tous les cas (indexation et recherche de l'ordre des mots). En particulier, si le cas simple où toutes les constantes sont distinctes et où tous les réseaux sémantiques ont au moins une entrée ne pose aucun problème (c'est le cas développé ci-dessus), les autres cas semblent moins évidents. En effet, l'indexation et la recherche de l'ordre des mots s'imbriquent et s'aident mutuellement.

De plus, par certains côtés (utilisation de la condition de planarité), on se rapproche du problème pour l'instant irrésolu de connaître la complexité de la recherche de preuve dans le calcul de Lambek.

Néanmoins, des cas connus comme posant problème pour Merenciano & Morrill (1997), l'algorithme les résout sans difficulté (unification du troisième ordre avec seek : $\lambda p.$(**try**($p$**find**), utilisation de himself : $\lambda P \lambda x.(P\ x)x$)

## 4.3 Remarques

La méthode de génération que nous avons présentée est étroitement liée à l'analyse dans le calcul de Lambek. En effet, le principe même est de trouver une preuve du calcul syntaxique (c'est-à-dire un bon appariement d'atomes) grâce à des données annexes (le réseau $\Pi_{\text{cible}}$). L'algorithme a donc la propriété de n'engendrer que des phrases correctes (donc analysables). Pour la propriété de ce que toute phrase correcte peut-être engendrée, elle correspond à la complétude de l'algorithme (car une solution du problème d'analyse est aussi une solution du problème de génération). Cette dernière est assurée par la finitude des possibilités (nombre d'appariements possibles, et différents ordres des mots possibles), mais est bien entendu un élément important de la complexité algorithmique qu'il reste à analyser.

On voit ainsi que ce processus de génération n'est lié qu'à l'expression sémantique, et non pas à son interprétation. Ainsi la partie syntaxique du lexique va déterminer ce qui est reconnu est engendrable par la

grammaire. Et la phrase « Jean vit à Jean », syntaxiquement correcte dans le lexique que nous avons donné (*Jean* et *Paris* ont le même type syntaxique :*NP*) et qui est analysée comme (**a j**)(**vit j**) ($\lambda$-terme bien typé, mais que l'on ne sait pas interpréter) peut donc être également engendrée.

Telle quelle, la capacité générative d'une grammaire dépend donc de la syntaxe associée à chaque item lexicale. Néanmoins, rien n'empêche d'imaginer une étape supplémentaire utilisant cette fois l'interprétation de la forme sémantique comme facteur limitant en analyse comme en génération.

# 5 Conclusion

Nous avons présenté une méthode de génération basée sur le calcul de Lambek et les grammaires de Montague. Plus précisément, nous avons utilisé les liens que ces systèmes entretiennent avec la logique linéaire et les réseaux de preuve.

Bien que de nombreux affinages restent à faire, en particulier en ce qui concerne la complexité algorithmique dans les cas un peu évolués, les écueils majeurs présents dans la méthode de Merenciano & Morrill (1997) (qui prend place dans le même cadre formel) peuvent être évités. Ceci en ne travaillant pas sur l'unification de $\lambda$-termes, mais sur leur représentation sous forme de réseau.

Enfin, comme nous l'avons vu, l'ordre correct des mots requiert un critère supplémentaire sur les réseaux (planarité). Nous savons aussi que la stricte concaténation du calcul de Lambek fixe des limites de modélisation linguistique. On peut alors envisager d'étendre la méthode présentée à des systèmes ayant une autre gestion de l'ordre des mots, mais manipulant également des réseaux de preuve (Lecomte & Retoré 1995).

# Références


Danos, V. (1990), Une Application de la Logique Linéaire à l'Étude des Processus de Normalisation (principalement du $\lambda$-calcul), PhD thesis, Université Paris VII.

de Groote, P. & Retoré, C. (1996), On the semantic readings of proof-nets, in G. M. Geert-Jan Kruijff & D. Oehrle, eds, 'Formal Grammar', FoLLI, Prague, pp. 57--70.

Dowty, D. R., Wall, R. E. & Peters, S. (1981), Introduction to Montague Semantics, Kluwer Academic Publishers.

Girard, J.-Y. (1987), `Linear logic', Theoretical Computer Science 50, 1--102.

Girard, J.-Y., Lafont, Y. & Taylor, P. (1988), Proofs and Types, Cambridge Tracts in Theoretical Computer Science 7, Cambridge University Press.

Lamarche, F. & Retoré, C. (1996), Proof-nets for the lambek calculus -- an overview, in V. M. Abrusci & C. Casadio, eds, 'Proceedings 1996 Roma Workshop. Proofs and Linguistic Categories', Editrice CLUEB, Bologna, pp. 241--262.

Lambek, J. (1958), 'The mathematics of sentence structure', American Mathematical Monthly 65(3), 154-170.

Lecomte, A. & Retoré, C. (1995), Pomset logic as an alternative categorial grammar, in 'Formal Grammar', Barcelona.

Levy, J. (1996), Linear second-order unification, in H. Ganzinger, ed., 'Proceedings of the 7th International Conference on Rewriting Techniques and Applications (RTA-96)', Vol. 1103 of LNCS, SpringerVerlag, Berlin, pp. 332--346.

Merenciano, J. M. & Morrill, G. (1997), Generation as deduction on labelled proof nets, in Retoré (1997), pp. 310--328.

Moortgat, M. (1997), Categorial type logics, in van Benthem & Meulen (1997).

Retoré, C. (1996a), 'Calcul de lambek et logique linéaire', Traitement Automatique des Langues 37(2), 39-70.

Retoré, C. (1996b), Perfect matchings and series-parallel graphs : multiplicatives proof nets as r&b-graphs, in J.-Y. Girard, M. Okada & A. Scedrov, eds, 'Linear Logic 96 Tokyo Meeting', Vol. 3 of Electronic Notes in Theoretical Computer Science.

Retoré, C., ed. (1997), Logical Aspects of Computational Linguistics : First International Conference, LACL '96, Nancy, France, September 23--25, 1996 : selected papers, Vol. 1328 of Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science, Springer-Verlag Inc., New York, NY, USA.

Roorda, D. (1991), Resource Logics : Proof-theoretical Investigations, PhD thesis, University of Amsterdam. van Benthem, J. F. A. K. & Meulen, A. T., eds (1997), Handbook of Logic and Language, MIT Press.