Grammaires d'unification polarisées

Sylvain Kahane

Lattice, Université Paris 7
Modyco, Université Paris 10
sk@ccr.jussieu.fr

Résumé – Abstract

Cet article propose un formalisme mathématique générique pour la combinaison de structures. Le contrôle de la saturation des structures finales est réalisé par une polarisation des objets des structures élémentaires. Ce formalisme permet de mettre en évidence et de formaliser les mécanismes procéduraux masqués de nombreux formalismes, dont les grammaires de réécriture, les grammaires de dépendance, TAG, HPSG et LFG.

This paper proposes a generic mathematical formalism for the combination of structures. The control of saturation of the final structures is realized by a polarization of the objects of the elementary structures. This formalism allows us to bring to the fore and to formalize the hidden procedural mechanisms of numerous formalisms, including rewriting systems, dependency grammars, TAG, HPSG and LFG.

Mots Clés –Keywords

Grammaire formelle, unification, polarisation, grammaire de réécriture, grammaire de dépendance, TAG, HPSG, LFG, graphe, arbre, dag.
Formal grammar, unification, polarization, rewriting system, dependency grammar, TAG, HPSG, LFG, graph, tree, dag.

1 Introduction

Notre objectif est de donner un cadre général à la plupart des formalismes à base de règles utilisées en linguistique. Les productions langagières sont fortement structurées et par ailleurs la langue est compositionnelle, c'est-à-dire que la structure[1] d'unités langagières complexes peut

[1] La question de savoir si une production langagière, par exemple une phrase, a une ou des structures est une question qui dépend du point de vue. D'une part, il est clair qu'on peut associer à une phrase différentes structures selon que l'on considère le point de vue sémantique, syntaxique, morphologique ou phonologique. D'autre part, les différentes structures d'une phrase ne sont pas indépendantes les unes des autres et même si ce ne sont pas des structures sur les mêmes objets (par exemple les nœuds de la structure sémantique ne correspondent pas un à un aux nœuds de la structure syntaxique, qui eux correspondent aux mots), il existe des liens entre les différents objets de ces structures. En d'autres termes, en considérant de façon dissociée les différentes structures simples de la phrase, on ne rend pas compte de la structure complète de la phrase, puisqu'on perd les interrelations entre les structures simples.

être obtenue par composition de structures élémentaires associées aux unités élémentaires. Le plus simple des modes de composition de deux structures A et B est l'unification, à savoir la construction d'une structure C en superposant partiellement A et B, c'est-à-dire en identifiant une partie des objets de A avec ceux de B. Cette idée rejoint une idée ancienne, déjà présente chez Jespersen 1924, Tesnière 1934 ou Ajduckiewicz 1935, que la phrase serait comme une molécule dont les mots seraient les atomes, chaque mot étant doté d'une certaine valence (terme explicitement emprunté par la linguistique à la chimie) qui l'oblige ou lui permet de s'assembler à un certain nombre d'autres mots. Néanmoins, les grammaires d'unification (où l'unification est prise au sens strict du terme) sont incapables de rendre compte du fait que la structure de certaines unités linguistiques est insaturée et que celles-ci doivent obligatoirement se combiner avec d'autres unités pour former une unité stable. Par exemple, la forme verbale *voulait* ne peut former à elle seule une phrase et appelle un sujet et un complément. La plupart des formalismes à base de règles ne traitent pas directement dans le formalisme des règles la question de la saturation des structures générées. Par exemple, dans une grammaire de réécriture, ce ne sont pas les règles de la grammaire qui assurent qu'une suite générée par la grammaire soit uniquement composée de symboles terminaux, mais une condition séparée, vérifiée indépendamment et compliquant par conséquent la procédure de dérivation. En HPSG, la saturation est assurée par des conditions obligeant certains traits à avoir pour valeur une liste vide. L'utilisation de polarités permet d'incorporer le traitement de la saturation dans le formalisme des règles. Le processus de saturation va ainsi guider la composition des structures tout au long du processus de génération.

Dans cet article, nous présentons une nouvelle famille de formalismes, les *grammaires d'unification polarisées* (GUP). Les GUP prolongent les grammaires d'unification en proposant un contrôle très explicite de la saturation des structures par l'attribution à chaque objet d'une *polarité*. Certaines polarités sont *neutres*, d'autres non, mais une structure finale doit obligatoirement être complètement neutre. Deux objets non neutres peuvent s'unifier (c'est-à-dire en fait s'identifier) en formant un objet neutre, c'est-à-dire en se neutralisant. L'unification proprement dite ne permet rien d'équivalent. Le formalisme de Nasr (1995) est l'un des premiers à utiliser explicitement une polarisation des structures et Duchier & Thater 1999 est la première étude à ma connaissance à mettre en avant la question de la polarité (et à introduire les termes *polarité* et *neutralisation*), tandis que Perrier 2002 est le premier à proposer un formalisme, les grammaires d'interaction, entièrement basé sur ces idées.

Dans la Partie 2, nous présenterons le cadre général des GUP et notamment le système des polarités. La Partie 3 proposera plusieurs exemples de GUP et tout particulièrement la traduction en GUP de formalismes classiques comme les grammaires de réécriture, TAG, HPSG ou LFG.

2 Unification et polarités

2.1 Grammaires d'unification polarisées

Les grammaires d'unification polarisées sont des grammaires permettant de générer des ensembles de structures finies. Tous les types de structures sont a priori envisageables, mais dans cet article nous ne considérerons que des graphes orientés et notamment les cas particuliers des dags[2] et des arbres. Nous considérerons également des structures complexes obtenues en combinant plusieurs structures sur les mêmes objets.

Une structure repose sur des *objets*. Par exemple, pour un graphe (orienté), les objets sont les *nœuds* et les *arcs*. Ces deux types d'objets sont liés entre eux, ce qui fournit la structure proprement dite : si X est l'ensemble des nœuds et U l'ensemble des arcs, le graphe est défini par deux fonctions π_1 et π_2 de U dans X qui associent respectivement à un arc sa *source* et sa *cible*.

---

[2] On appelle *dag* un graphe orienté acyclique (angl. *directed acyclic graph*).

Une *structure polarisée* est une structure dont les objets sont polarisés, c'est-à-dire étiquetés par une valeur appartenant à un ensemble fini P de polarités. L'ensemble P est muni d'une opération notée ".", appelée *produit*. Le produit est commutatif et généralement associatif : (P,.) est appelé le *système des polarités*. Un sous-ensemble N de P contient les polarités dites *neutres*. Une structure polarisée est dite *neutre* si tous les objets de cette structure sont neutres.

Les structures peuvent être combinées par *unification*. L'unification de deux structures A et B donne une nouvelle structure A⊕B obtenue en « collant » ensemble ces structures par l'identification d'une partie des objets de la première structure avec ceux de la deuxième. Lorsque deux structures polarisées A et B sont unifiées, la polarité d'un objet de A⊕B obtenu par identification de deux objets de A et B est le produit de leurs polarités.

Une *grammaire d'unification polarisée* (GUP) est définie par une famille finie F de types d'objets (avec des fonctions attachées aux différents types d'objets), un système (P,.) de polarités, un sous-ensemble N de P de polarités neutres, et un ensemble fini de structures élémentaires polarisées, dont les objets sont décrits par F et dont une est marquée comme la structure initiale. Les structures *générées* par la grammaire sont les structures neutres obtenues par combinaison de la structure initiale et d'un nombre fini de structures élémentaires. Les polarités sont uniquement utilisées pour contrôler la saturation au cours du calcul et ne sont pas considérées lorsqu'on évalue la capacité générative de la grammaire. Bien que données sous une forme déclarative, les polarités jouent en fait un rôle essentiellement procédural.

2.2 Le système des polarités

Nous allons présenter un système des polarités P qui sera ensuite utilisé dans tous nos exemples. L'élément neutre (au sens mathématique du terme) de P est noté ◎ et appelé *gris* (□*x*□P, ◎.*x*=*x*). Le symbole ◎ est une polarité neutre (◎□N) et peut être interprété comme un *contexte facultatif*. Un *contexte obligatoire* est représenté par la symbole ○ appelé *blanc*. Il s'agit d'une polarité non neutre (○□N) qui se comporte comme un élément neutre sauf avec ◎ (□*x*≠◎, ○.*x*=*x*). Le symbole □ représente l'*échec* et est par conséquent l'élément absorbant de P (□*x*□P, □.*x*=□). Evidemment, □ n'est pas neutre et par l'impossibilité de le neutraliser, l'apparition de □ bloque tout espoir de produire une structure neutre. Le symbole ●, appelé *noir*, représente la *saturation*□ il s'agit d'une polarité neutre qui ne peut être combinée avec aucune autre polarité que les deux précédentes (□*x*≠◎/○, ●.*x*=□). Les symboles + et – représentent les polarités *positive* et *négative*□ ce sont des polarités non neutres qui, combinées entre elles, donnent la saturation et qui ne peuvent être combinées avec elles-mêmes ou avec la saturation (+.–=●, +.+=□, +.●=□, –.–=□ et –.●=□). Le symbole – peut être vu comme un *besoin* et + comme la *ressource* correspondante.

Le système {○,●,□} est utilisé par Nasr 1995, tandis que le système {◎,●,–,+,□} est considéré par Bonfante et al. 2003[3], qui montrent les avantages des polarités négatives et positives dans le préfiltrage des objets en analyse syntaxique (une famille de structures comportant des polarités négatives et positives ne pourront se combiner en une structure neutre que si la somme des polarités négatives de chaque type d'objet est égale à la somme des polarités positives).

Le système (P,.) que nous venons de présenter est commutatif et associatif. La commutativité est essentielle pour que la combinaison de deux structures ne soit pas procéduralement orientée. L'associativité permet de rendre la combinaison des objets également associative, c'est-à-dire que si un objet B doit se combiner avec A et C, il n'y a pas d'ordre de précédence entre la combinaison de A et B et celle de B et C : A⊕(B⊕C) = (A⊕B)⊕C.

---

[3] Ceux-ci utilisent les notations {=,□ ,□ ,□ ,□}.

Bien d'autres polarités peuvent encore être utiles. Par exemple, un *besoin facultatif*, noté ?, aura les mêmes propriétés que la polarité négative – (+.?=●, ?.?=□ et ?.●=□), mais sera neutre. Ou encore, l'*instanciation*, au sens des structures de traits, notée ::, aura les mêmes propriétés que la saturation, mais pourra se combiner avec elle-même (::.::=::). Dans la suite, nous utiliserons un système avec deux polarités neutres (◎, ●) et quatre polarités non neutres (○, +, –, □).

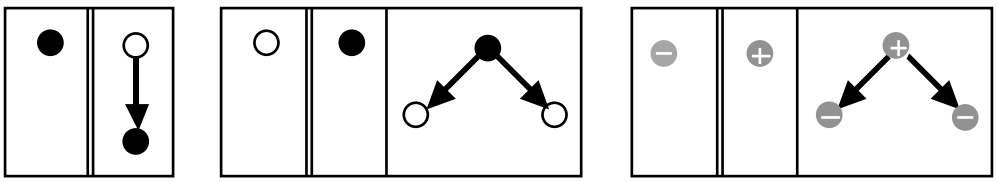
2.3 Monotonie

Tel que présenté jusqu'à maintenant, le formalisme des GUP est monotone aux polarités près : le résultat A⊕B de la combinaison de deux structures A et B par une GUP contient A et B comme sous-structures. On peut ajouter un ordre (partiel) sur P pour rendre le formalisme complètement monotone[4]. Notons ≤ cet ordre. Pour que le formalisme soit monotone, ≤ doit vérifier la *propriété de monotonie* suivante : $\forall x,y \in P\ x.y \geq x \sqcup y$, où $x \sqcup y$ est le maximum de x et y. Ceci nous impose d'avoir ◎ < ○ < +/–/? < ● < □ et ○ < :: < □. Une GUP construite sur un système ordonné (P,.,≤) vérifiant la propriété de monotonie est monotone. Ceci induit de bonnes propriétés computationnelles et permet par exemple de traduire l'analyse en GUP en un problème de résolution de contraintes (Duchier & Thater 1999).

3 Des exemples de GUP

3.1 Les grammaires d'arbres

Les premières grammaires d'arbre rentrant dans le paradigme des GUP ont, à ma connaissance, été proposées par Nasr 1995. La grammaire G1 suivante permet de générer tous les *arbres* finis (un arbre est un graphe orienté connexe dont tous les nœuds sont la cible d'au plus un arc) : les objets sont des nœuds et des arcs ; la structure initiale (placé dans le cadre de gauche) est réduite à un nœud noir ; la grammaire possède une seule structure élémentaire qui est composée d'un arc noir reliant un nœud blanc à un nœud noir. Chaque nœud blanc devra s'unifier avec un nœud noir pour être neutralisé et un nœud noir pourra s'unifier avec un nombre quelconque de nœuds blancs. On vérifie aisément que les structures générées par la grammaire sont bien des arbres, puisque tout nœud aura un et un seul gouverneur à l'exception du nœud introduit par la structure initiale qui sera la racine de l'arbre. Dans nos figures, la double barre sépare la structure initiale des autres structures élémentaires.



G1 G2 G3

La grammaire G1 ne permet pas de contrôler le nombre de fils de chaque nœud. Une grammaire du type de G2 permet de contrôler la valence de chaque nœud, mais elle n'assure pas que les structures générées sont des arbres puisque deux nœuds blancs peuvent s'unifier et un nœud peut donc avoir deux gouverneurs[5]. La grammaire G3 résout le problème en ne générant que des

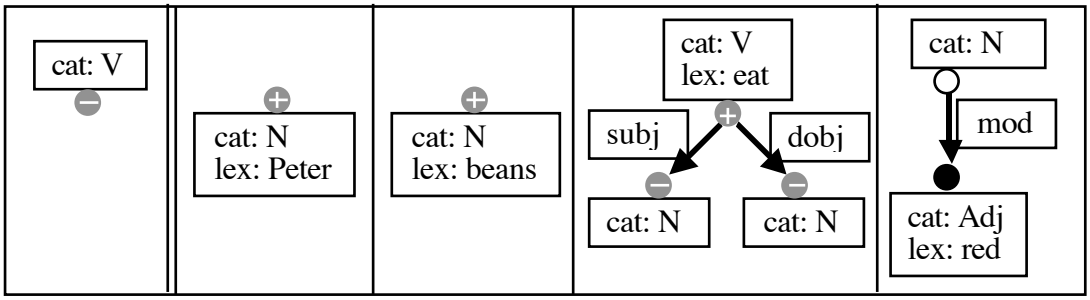
---

[4] Cette idée m'a été suggérée par Guy Perrier.

[5] Nasr 1995 propose une grammaire de ce type pour générer des arbres. Ceci est assuré par une clause procédurale qui oblige, lors de la combinaison de deux structures, la racine d'une des structures à se combiner avec un nœud de l'autre.

arbres. En fait, on peut voir G3 comme la superposition de G1 et G2. En effet, si P_0 = {○,●}, P_1 = {○,+,–,●} ≡ $P_0 \square P_0$, avec +=(○,●) et –=(●,○). La première polarité contrôle la structure d'arbre comme dans G1, tandis que la deuxième contrôle la valence comme dans G2.

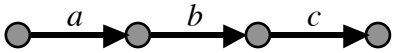
Avec les mêmes principes, on peut construire une *grammaire de dépendance* générant les arbres de dépendance syntaxique d'un fragmet de langue. La grammaire G4, directement inspirée de Nasr 1995, propose un fragment de grammaire pour l'anglais permettant de générer l'arbre syntaxique de *Peter eats red beans*. Les nœuds de cette grammaire sont étiquetés par des structures de traits qui devront s'unifier. Les traits et les valeurs des structures de traits sont égalements des objets qui peuvent (et doivent dans une GUP) être polarisés (cf. plus loin la représentation des structures de traits d'HPSG sous forme de dags). En attribuant des polarités positives et négatives aux traits /cat/ plutôt qu'aux nœuds, on obtient une grammaire dans l'esprit de Perrier 2002.



G4 (Grammaire de dépendance pour l'anglais)

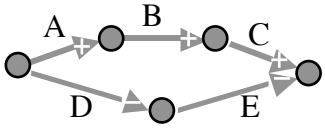
3.2 Grammaires de réécriture et arbres ordonnés

N'importe quelle *grammaire de réécriture* peut être simulée par une GUP. Nous reprenons ici des idées développées par Burroni 1993. Les GUP ont donc la capacité générative faible des machines de Turing. Une suite *abc* est représentée par une chaîne d'arcs étiquetés a, b et c :



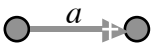
La catégorie initiale S de la grammaire donne la structure initiale :

Une règle de récriture ABC → DE donne la structure élémentaire :

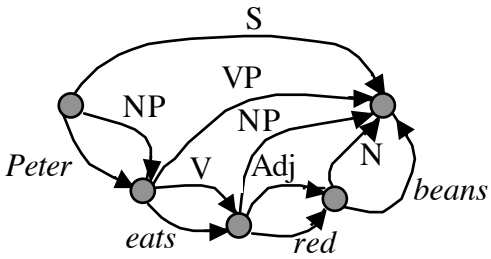


Cette structure élémentaire est une «□ellule□» dont le bord supérieur est une chaîne d'arcs positifs correspondant à la partie gauche de la règle, tandis que le bord inférieur est une chaîne d'arcs négatifs correspondant à la partie droite de la règle. Chaque arc positif doit s'unifier avec un arc négatif et vice-versa pour donner un arc noir. Les nœuds sont gris, c'est-à-dire qu'ils sont neutres et peuvent s'unifier entre eux.

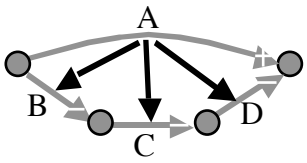
Un symbole terminal *a* correspond à un arc positif :



Les cellules vont venir s'unifier les unes aux autres pour donner une structure finale qui représente la structure de dérivation d'une suite, la suite elle-même étant le bord inférieur de cette structure. Nous donnons ci-après la structure de dérivation de la suite *Peter eats red beans* avec une grammaire syntagmatique classique, que le lecteur reconstituera de lui-même. Dans une telle représentation, les arcs représentent des syntagmes et correspondent à des intervalles de découpage de la suite et les nœuds aux bornes de ces intervalles.



Pour une *grammaire de réécriture hors-contexte*, la grammaire génère en fait l'arbre de dérivation , que l'on peut représenter de manière traditionnelle en ajoutant les branches de l'arbre. Les nœuds de l'arbre de dérivation seront les arcs des cellules et une règle du type A□ BCD donnera la structure :



Rappelons qu'un arbre de dérivation pour une grammaire de réécriture hors-contexte est un arbre ordonné. Un *arbre ordonné* combine deux structures sur un même ensemble de nœud : une structure d'arbre proprement dite et une relation de précédence sur les nœuds de l'arbre. Ici la relation de précédence est représentée de manière explicite (un «□nœud□» de l'arbre précède un autre «□nœud□» si la cible du premier est la source du deuxième). Il n'est pas possible, avec une GUP, de générer l'arbre de dérivation, relation de précédence comprise, de façon plus simple[6]. Notons que la représentation usuelle des arbres ordonnés (où la relation de précédence n'est pas explicite, mais seulement déductible de la planéité de la représentation) est fort trompeuse d'un point de vue computationnel. En calculant la relation de précédence, les analyseurs syntaxiques (du type CKY par exemple) calculent en fait une structure de données comme celle que nous présentons ici, dont les nœuds sont les débuts et les fins des syntagmes.

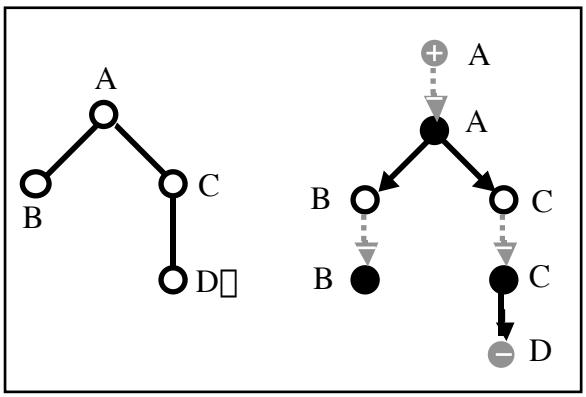
3.3 TAG (Grammaire d'adjonction d'arbres)

Les GUP doivent beaucoup aux grammaires TAG qui sont les premières grammaires de combinaisons de structures à avoir été étudiées en détail. On présente généralement les grammaires TAG comme des grammaires de combinaison d'arbres (ordonnés). En fait, en tant que grammaires d'arbres, les TAG ne sont pas monotones et ne peuvent donc être simulées par une GUP. Comme l'a montré Vijay-Shanker 1992, il faut pour obtenir un formalisme monotone considérer TAG comme une grammaire de combinaison de quasi-arbres. Intuitivement, un *quasi-arbre* est un arbre dont les nœuds sont scindés en deux et possèdent donc une partie supérieure et une partie inférieure, entre lesquelles peut venir s'intercaler un autre quasi-arbre (c'est la fameuse opération d'adjonction de TAG). Formellement, un quasi-arbre est un arbre dont les branches sont de deux types : des relations de dépendance et des relations de dominance (respectivement notés par un trait plein et un trait pointillé). Deux nœuds reliés par une relation de dominance sont considérés comme étant potentiellement les deux parties d'un même nœud ;

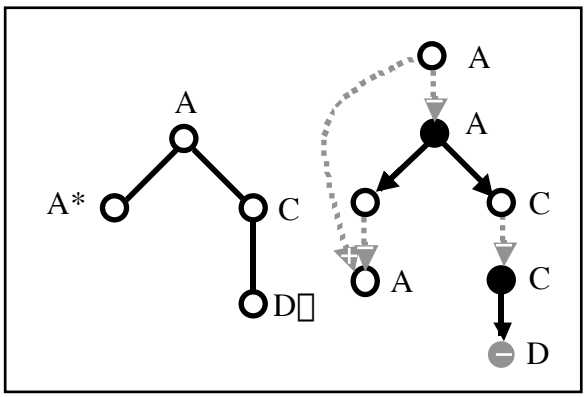
---

[6] L'idée la plus naturelle serait d'encoder une règle de réécriture par un arbre de profondeur 1 et la relation de précédence par des arcs allant d'un nœud à son successeur. La difficulté est ensuite de propager la relation d'ordre aux descendants de deux frères lorsqu'on leur applique une règle de réécriture par substitution d'un arbre de profondeur 1. La solution la plus simple est incontestablement celle présentée ici, consistant à un introduire des objets représentant les débuts et fins des syntagmes (nos nœuds gris) et d'indiquer les relations entre un syntagme, son début et sa fin en représentant le syntagme par un arc allant de son début à sa fin.

seule la partie inférieure peut avoir plusieurs fils. La figure ci-dessous donne un arbre □ (= à substituer) et un arbre □ (= à adjoindre) avec les structures de la GUP correspondante[7].



Un arbre □ et sa traduction GUP



Un arbre □ et sa traduction GUP

Un nœud de substitution (comme D□) donne un nœud négatif, qui devra s'unifier avec la racine d'un arbre □. Un arbre □ possède une racine et un nœud pied blancs qui devront s'unifier avec la partie haute et la partie basse d'un nœud scindé. Pour cette raison, la racine et le nœud pied sont reliés par un lien de dominance positif qui devra s'unifier avec le lien de dominance négatif reliant les deux parties du site d'adjonction. A la fin de la dérivation, la structure doit être un arbre et tous les nœuds doivent être reconstitués : pour cela nous introduisons la règle suivante, qui présente un lien de dominance positif reliant un nœud à lui même et qui en s'unifiant avec un lien de dominance négatif assurera la saturation de celui-ci et forcera l'unification de ses deux extrémités.

Cette dernière règle montre encore une fois l'avantage d'une GUP : la réunification des nœuds qui dans la présentation de Vijay-Shanker 1992 est donnée sous forme procédurale est ici assurée par une règle déclarative.

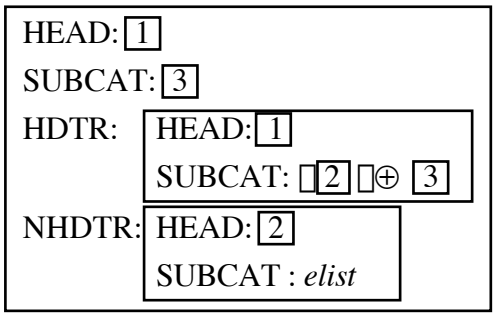
3.4 HPSG (Grammaire syntagmatique guidée par les têtes)

Il est bien connu que les structures de traits (ST) peuvent être vues comme des dags dont les arcs représentent les traits et les nœuds les valeurs (cf., par exemple, Kesper et Mönnich 2003). Les grammaires HPSG s'écrivent donc simplement dans le formalisme des GUP. Il est important de souligner au préalable une différence essentielle entre l'unification de ST et l'unification de structures dans les GUP : on considère généralement qu'une ST possède au plus un trait de chaque sorte et, par conséquent, lors de l'unification de deux ST, les traits de même sorte des deux ST doivent être identifiés et leurs valeurs unifiées. Il n'y a rien de tel pour les GUP : un nœud d'un dag peut très bien être la source de plusieurs arcs de même étiquette et seule la nécessité de neutraliser deux objets peut inciter à les identifier (il y a aussi des identifications qui résultent de contraintes structurelles : l'identification de deux arcs entraîne l'identification de leurs extrémités).

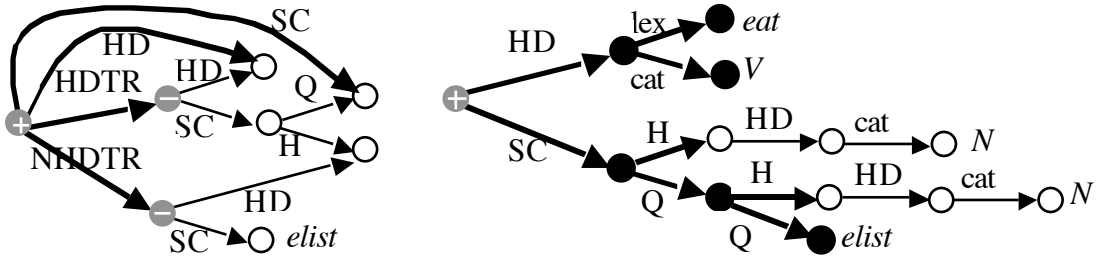
---

[7] Par souci de simplicité, nous laissons de côté dans notre présentation des grammaires TAG, l'encodage de la relation de précédence sur les fils d'un même nœud. Celle-ci devra être encodée, comme nous l'avons fait pour les grammaires de réécriture hors-contexte, en modélisant les semi-nœuds des arbres TAG par des arcs. Ceci ne pose aucune difficulté particulière, mais pourrait rendre les structures difficiles à comprendre.

Voyons donc comment traduire l'unification de ST en GUP. Considérons le schéma de base de HPSG, à savoir le schéma *head-daughter-phrase* de combinaison d'un constituant tête avec un sous-constituant fille sous-catégorisé[8] :



Cette structure de traits donne le dag suivant, où une liste est représentée récursivement en deux morceaux : sa tête (valeur de H) et sa queue (valeur de Q). Nous plaçons à la droite de ce dag l'entrée lexicale pour *eat* indiquant que *eat* est un V dont la SUBCAT contient deux constituants de catégorie N (nous traitons le sujet comme un complément sous-catégorisé).



Comme nous l'avons dit, l'identification de deux arcs de même étiquette et de même source n'est plus immédiate comme avec les ST. Elle est ici contrôlée par la polarisation : une partie du dag est une ressource (noir/gras ou positif) et une partie est un besoin (blanc/fin ou négatif) qui devra être neutralisé. En plus d'assurer l'unification complète des ST, la polarisation permet également de forcer l'instanciation des listes SUBCAT (et par conséquent de contrôler la saturation de la valence), ce qui n'est pas directement assuré par le formalisme habituel des règles en HPSG. Notons que le fragment de grammaire HPSG donné ici est équivalent à celui de G4 et inutilement plus compliqué. Remarquons qu'à la différence des arbres, on n'a pas de GUP qui génère tous les dags (on ne peut pas déceler des cycles de longueur potentiellement illimitée). Néanmoins, on peut assurer qu'une GUP ne génère que des dags, notamment en obligeant comme ici à respecter une structure d'arbre sous-jacente : cela signifie qu'un nœud ne pourra avoir deux gouverneurs dans une structure finale que si cela a été spécifié par une structure élémentaire.

3.5 LFG (Grammaire lexicale fonctionnelle) et grammaires synchrones

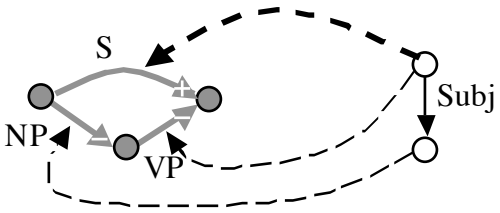
Le formalisme de LFG permet de synchroniser une grammaire syntagmatique hors-contexte avec une grammaire de dépendance. La grammaire de dépendance ne génère pas des arbres, mais des structures de traits réentrantes, appelées *structures fonctionnelles*, que nous représentons par

---

[8] Les numéros dans les boîtes indiquent des valeurs partagées par plusieurs traits. La valeur de SUBCAT est une liste (la liste des constituants sous-catégorisés). Le constituant fille (NHDTR) doit être saturé et avoir donc une liste SUBCAT vide (*elist*). La liste SUBCAT du sous-constituant tête (HDTR) est la concaténation, notée ⊕, de deux listes : une liste à un élément constituée de la description du sous-constituant fille et la liste SUBCAT du constituant complet. Le reste de la description de ce constituant (valeur de HEAD) est égal à celle de son sous-constituant tête.

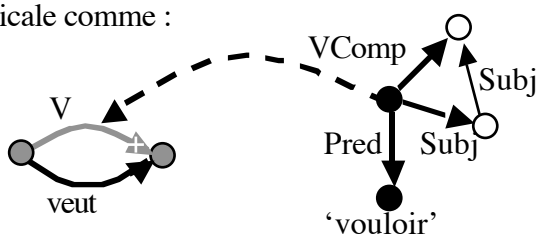
des dags, comme nous l'avons fait pour HPSG. Considérons une première règle LFG et sa traduction GUP :

S → NP VP
↓ = ↑Subj ↓ =



Les équations sous les catégories assurent la synchronisation de la structure syntagmatique avec la structure fonctionnelle. Chaque nœud syntagmatique est synchronisé avec un nœud «fonctionnel». Les expressions ↓ et ↑ renvoient respectivement au nœud fonctionnel synchronisé avec le nœud syntagmatique courant et à celui synchronisé avec le nœud syntagmatique père (ici S). L'équation ↓=↑ signifie donc que le nœud syntagmatique courant et son père sont synchronisés avec le même nœud fonctionnel. L'expression ↑Subj désigne le nœud fonctionnel dépendant de ↑ par la relation Subj. Dans la traduction, seul le lien de synchronisation vers le nœud syntagmatique père est saturé. La structure fonctionnelle est totalement blanche et elle sera saturée par une règle lexicale comme :

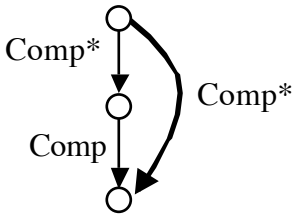
V → veut
↑Pred = 'vouloir⟨↑Subj,VComp⟩
↑Subj = ↑VCompSubj

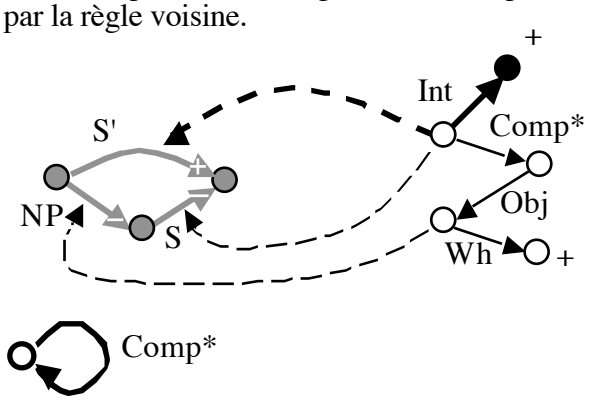


On notera que la règle lexicale est entièrement traduite par des éléments saturés, à l'exception des éléments de la valence et des éléments introduits par une équation fonctionnelle, comme ici la relation Subj entre les deux éléments de la valence introduite par ↑Subj = ↑VCompSubj.

Donnons encore la traduction d'une règle pour le traitement d'une interrogative indirecte avec extraction de l'objet. Notons que la distinction entre équation contrainte ($f =_c v$) et non contrainte ($f = v$) est facilement prise en compte par les polarités. La propagation de Comp* (qui correspond, comme son nom l'indique, à une séquence de dépendance Comp) est assurée par la règle en bas à gauche et l'arrêt de cette propagation par la règle voisine.

S' → NP S
↓ = ↑Comp*Obj ↑Int = +
↓Wh =$_c$ +





Encore une fois la traduction en GUP permet de mettre en évidence certaines composantes fondamentales des règles (comme les liens de synchronisation) et des mécanismes non explicités comme la propagation de Comp* ou le fait qu'une équation lexicale comme ↑Pred = 'vouloir⟨↑Subj,VComp⟩ introduit à la fois des ressources et des besoins.

4 Conclusion

L'introduction de la polarisation des objets dans une grammaire d'unification permet de contrôler la saturation des structures de manière explicite. Procéduralement, le formalisme des GUP est

extrêmement simple : il impose seulement que la combinaison de deux structures mette en jeu l'unification d'au moins un objet. L'obligation ou l'interdiction de combiner davantage d'objets est ensuite entièrement contrôlée par la polarisation des objets. La polarisation va donc guider la procédure de combinaison des structures élémentaires. Malgré sa simplicité, le formalisme des GUP est suffisamment puissant pour simuler élégamment la quasi-totalité des formalismes utilisés en linguistique formelle et en TAL[9]. Cela permet à la fois d'obtenir un éclairage nouveau sur ces formalismes, de mettre en évidence la nature exacte des structures qu'ils manipulent et d'extraire certains mécanismes procéduraux masqués par le formalisme. Mais surtout, le formalisme des GUP permet d'écrire séparément divers modules de la grammaire manipulant des structures différentes et de les coupler ensuite en un même formalisme, comme cela est par exemple fait en GUST (Kahane 2002). Les GUP fournissent ainsi une alternative aux formalismes d'unification basés sur les structures de traits en permettant une plus grande diversité des structures géométriques et un meilleur contrôle des ressources. Les propriétés computationnelles des GUP restent à étudier et notamment les restrictions qui permettent d'avoir une analyse en temps polynomial.

Remerciements

Je remercie Benoît Crabbé, Kim Gerdes, Piet Mertens, Guy Perrier, Alain Polguère et Benoît Sagot pour leurs nombreuses remarques et leurs commentaires éclairants.

Références

AJDUKIEWICZ K. (1935), Die syntaktische Konnexität, *Studia Philosophica,* Vol. 1, 1-27.

BONFANTE G., GUILLAUME B., PERRIER G. (2003), Analyse syntaxique électrostatique, *TAL.*

BURRONI A. (1993), Higher-dimensional word problems with applications to equational logic, *Theoretical Computer Sciences*, Vol. 115, 43-62.

DUCHIER D., THATER S., 1999, Parsing with tree descriptions: a constraint-based approach, *NLULP 1999 (Natural Language Understanding and Logic Programming)*, Las Cruces, NM.

KAHANE S. (2002), *Grammaire d'Unification Sens-Texte : vers un modèle mathématique articulé de la langue*, Habilitation à diriger les recherches, Université Paris 7, 82 p.

KESPER S., MÖNNICH U. (2003), Graph properties of HPSG feature structures, *Formal Grammar 2003*, 115-124.

NASR A. (1995), A formalism and a parser for lexicalised dependency grammars, *4th Int. Workshop on Parsing Tecnologies*, State Univ. of NY Press.

PERRIER G. (2002), Descriptions d'arbres avec polarités : les Grammaires d'Interaction, *TALN 2002*, Nancy.

ROGERS J., VIJAY-SHANKER K. (1992), Reasoning with descriptions of trees, *ACL 1992*, 72-80.

TESNIERE Lucien, 1934, Comment construire une syntaxe, *Bulletin de la Faculté des Lettres de Strasbourg,*Vol. 7, 12ème année, 219-229.

VIJAY-SHANKER K. (1992), Using description of trees in a Tree Adjoining Grammar, *Computational Linguistics*, Vol. 18, n°4, 481-517.

---

[9] Aux formalismes présentés ici, nous pouvons ajouter une partie des grammaires catégorielles, traduites en GUP par Perrier 2002 sous le nom de grammaires d'interaction. Rappelons encore une fois que la présente contribution doit beaucoup à ce travail précurseur.