Structures mathématiques du langage

La grammaire de Montague et la théorie de Heim et Kratzer

Résumé

Ce qui suit est un court résumé des principes essentiels de la Grammaire de Montague, souvent appelée "grammaire catégorielle" car elle se fonde en partie sur l'idée de résiduation vue au chapitre précédent. Surtout, elle cherche à établir un homomorphisme entre deux systèmes : l'un, syntaxique, basé sur les catégories syntaxiques (T, VI, VT, N, ... etc.) et l'autre, sémantique, basé sur les types primitifs e et t et tous les types qu'on peut obtenir au moyen d'eux à l'aide de la flèche. La question de *l'intensionnalité* n'est pas abordée dans ce chapitre, ce qui explique que, par rapport aux oeuvres originales de Montague et de ses commentateurs, on n'y trouve pas mention des opérateurs $^\wedge$ et $^\vee$ (intensionnalisation et extensionnalisation). Le chapitre est complété par un aperçu de la théorie de Heim et Kratzer, qui tente d'adapter la démarche compositionnelle suivie par Montague au cadre de la théorie générative contemporaine : il s'agit d'interpréter sémantiquement les opérations fondamentales *merge* et *move* et de montrer comment l'opération de déplacement est liée à l'intervention de *lieurs* grâce auxquels il est possible de rendre compte de divers phénomènes de liage dans la langue.

Table des matières

Références :

L.T.F. Gamut, *Logic, Language and Meaning*, Volume II, Intensional Logic and Logical Grammar, paragraphes 6.3.5 à 6.3.8

Attention : à des fins de simplification, nous avons éliminé toute référence aux intensions (absence d'opérateurs $^\wedge$ *et* $^\vee$ *dans tout ce qui suit, ainsi que du paramètre* s *dans la formulation des types)*

1 Principes de traduction

1.1 Homomorphisme

L'ensemble des catégories syntaxiques est défini comme suit :

1. S, CN, IV $\in CAT$
2. Si A, B $\in CAT$ alors $A/B \in CAT$

La grammaire est basée sur un *lexique* : une attribution d'une catégorie syntaxique à chaque mot ou expression de base. L'ensemble des expressions de base d'une catégorie A est noté B_A. Par exemple, B_{CN} contient la liste {*homme, femme, langue, licorne, éléphant, reine, parc...* }. Pour chaque entier n, les variables syntaxiques $il_n, elle_n, le_n, la_n$ etc. ont la catégorie syntaxique des termes, autrement dit appartiennent à B_T.
L'ensemble de toutes les expressions (pas seulement les basiques) qui appartiennent à une catégorie A est noté P_A.
On admet deux fonctions, l'une établissant un lien entre les catégories syntaxiques et les types sémantiques, et l'autre, qui sera définie récursivement, construisant une traduction des expressions de surface ("phonologiques") vers des expressions exprimant des formes logiques.

- un *homomorphisme* de types : $f \;:\; CAT \longrightarrow$ **Typ**
  - $f(S)$ = **t**
  - $f(CN) = f(VI)$ = **e** $\rightarrow$ **t**
  - $f(A/B) = f(B) \rightarrow f(A)$
- une fonction de traduction, notée $(\;)^t$ des expressions "phonologiques" vers des expressions de LI

pour l'instant, LI = FOL (langage de la "first order logic")

Exemple 1 *Exemples : verbes transitifs*

- $VT = VI/T$
- $f(VI)$ =**e** $\rightarrow$ **t**
- $f(T)$ = (**e** $\rightarrow$ **t**) $\rightarrow$ **t**
- $f(VT) = f(T) \rightarrow f(VI)$
- *donc* $f(VT)$ = ((**e** $\rightarrow$ **t**) $\rightarrow$ **t**) $\rightarrow$ (**e** $\rightarrow$ **t**)

1.2 Règles

1.2.1 Règle de base

S1 : tout élément lexical de catégorie A est une expression de catégorie A
(autrement dit : $B_A \subset P_A$ pour toute catégorie A)
T1(a) : Les éléments du lexique sont traduits sous forme de **constantes** ayant le type approprié
Exemples :

- marche : **marche** de type $\mathbf{e} \to \mathbf{t}$
- rencontre : **rencontre** de type $((\mathbf{e} \to \mathbf{t}) \to \mathbf{t}) \to (\mathbf{e} \to \mathbf{t})$
- enfant : **enfant** de type $\mathbf{e} \to \mathbf{t}$

1.2.2 Règle d'application fonctionnelle

S2 : si $\delta \in P_{VI}$ et $\alpha \in P_T$, alors $F_1(\alpha, \delta) \in P_S$ où F_1 est définie par $F_1(\alpha, \delta) = \alpha\delta$', où δ' est le résultat du remplacement du verbe principal dans δ par sa forme conjuguée à la troisième personne du singulier
T2 : $F_1(\alpha, \delta)^t = \alpha^t(\delta^t)$

Exemple 2 John travaille *:*

$\lambda P.P(\mathbf{john}^*)(\mathbf{travaille})$
S
John travaille

α *: John* — δ *: travailler*
T — VI
α^t *:* $\lambda P.P(\mathbf{john}^*)$ — δ^t *:* **travaille**

La β-réduction donne : $\lambda P.P(\mathbf{john}^*)(\mathbf{travaille}) \to \mathbf{travaille}(\mathbf{john}^*)$
Remarque : il est important de noter que, pour Montague, les termes (T) sont tous de type élevé, et non de type e. Ceci résulte de ce que le type des termes, T est tel que $T = S/VI$ et de l'homorphisme postulé plus haut qui traduit la catégorie syntaxique en le type sémantique $(e \to t) \to t$. On note alors que si **a** est une constante de type e, elle peut être transformée en un terme de type $(e \to t) \to t$ dans le cadre de la théorie simple des types au moyen de la preuve suivante.

$$\begin{array}{l|l}
1 & a : e \\
2 & \quad \begin{array}{|l} X : e \to t \\ \hline \end{array} \\
3 & \quad \begin{array}{|l} X(a) : t \end{array} \\
4 & \lambda X.X(a) : (e \to t) \to t
\end{array}$$

1.2.3 Règles de formation de termes

S3 si $\zeta \in P_{CN}$ alors $F_2(\zeta) \in P_T$ où $F_2(\zeta) = chaque\ \zeta$
T3 $F_2(\zeta)^t = \lambda Q.\forall x(\zeta^t(x) \Rightarrow Q(x))$
S4 si $\zeta \in P_{CN}$ alors $F_3(\zeta) \in P_T$ où $F_3(\zeta) = l(e)(a)(es)\ \zeta$
T4 $F_3(\zeta)^t = \lambda Q.\exists x(\forall y(\zeta^t(y) \Leftrightarrow (x = y)) \wedge Q(x))$
S5 si $\zeta \in P_{CN}$ alors $F_4(\zeta) \in P_T$ où $F_4(\zeta) = un(e)\ \zeta$
T5 $F_4(\zeta)^t = \lambda Q.\exists x(\zeta^t(x) \wedge Q(x))$

Variante :
S3' si $\sigma \in P_{T/CN}$ et $\zeta \in P_{CN}$ alors $F_2'(\sigma, \zeta) \in P_T$ où $F_2'(\sigma, \zeta) = \sigma\zeta$
T3' $F_2'(\sigma, \zeta)^t = \sigma^t(\zeta^t)$
exemple : chaque$^t = \lambda Q.\lambda P.\forall x(Q(x) \Rightarrow P(x))$

1.2.4 Les règles S1, S2, S3’

Exemple 3 chaque enfant dort *:*

chaque enfant dort, S, S2
$\forall x\ \mathbf{enfant}(x) \Rightarrow \mathbf{dort}(x)$

chaque enfant, T, S3’
$\lambda P.\forall x\ \mathbf{enfant}(x) \Rightarrow P(x)$

dormir, VI
dort

chaque, T/CN

enfant, CN

1.2.5 La règle S7

S7 : Si $\delta \in P_{VT}$ et $\alpha \in P_T$, alors $F_6(\delta, \alpha) \in P_{VI}$ où $F_6(\delta, \alpha) = \alpha'\delta$, où α' est la forme accusative de α si α est une variable syntaxique, $= \delta\alpha$ sinon
T7 : $F_6(\delta, \alpha)^t = \delta^t(\alpha^t)$

Exemple 4

cherche une licorne, $e \rightarrow t$, T7
$\mathbf{cherche}(\lambda P.\exists x\ \mathbf{licorne}(x) \wedge P(x))$

cherche,
cherche
$((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t),$
T1(a)

une licorne,
$\lambda P.\exists x.\mathbf{licorne}(x) \wedge P(x)$
$(e \rightarrow t) \rightarrow t,$ T5

lecture *de dicto*
commentaire : **chercher** ne traduit pas une relation avec un individu donné mais avec un ensemble de propriétés (l’ensemble de toutes les propriétés vraies d’au moins une licorne)
Cette lecture n’implique donc pas l’existence d’une licorne (*chercher* y est vu comme un verbe *intensionnel*, à la différence d’un verbe comme *toucher* qui serait nécessairement *extensionnel*, impliquant l’existence de ce que l’on touche). S’il n’y a pas de licorne, l’ensemble de propriétés en question est nécessairement vide ($\emptyset$) mais... $\emptyset$ **est** un ensemble !
Problème : on ne fait donc pas de différence avec *cherche un dragon* ou *chercher un cercle carré* puisque dans tous les cas on aurait une relation avec $\emptyset$. Ceci requiert l’introduction de l’intensionnalité (car si, dans le monde actuel, licorne = dragon = cercle carré, en revanche si on prend en compte les “mondes possibles”, nul ne doute que là où existent des licornes n’existent pas forcément des dragons, autrement dit l’ensemble des mondes possibles où vivent des licornes n’est en général pas identique à l’ensemble des mondes possibles où vivent des dragons !).

1.3 Ambiguïtés de lecture

1.3.1 Verbes intensionnels vs verbes extensionnels

La solution précédente permet de **ne pas** déduire l’existence de licornes à partir d’une proposition comme *Pierre cherche une licorne* :

- le verbe **chercher** est *intensionnel*

mais de *Pierre* **rencontre** *une licorne*, je peux en principe déduire qu’une licorne existe !

- le verbe **rencontrer** est *extensionnel*

1.3.2 Postulats de signification

- *Pierre cherche une licorne* $\rightarrow$ $\mathbf{cherche}(\mathbf{pierre}, \lambda P.\exists x\ \mathbf{licorne}(x) \wedge P(x)) \not\rightarrow \exists x.\mathbf{licorne}(x) \wedge \mathbf{cherche}_*(\mathbf{pierre}, x)$
- *Pierre rencontre une licorne* $\rightarrow$ $\mathbf{rencontre}(\mathbf{pierre}, \lambda P.\exists x\ \mathbf{licorne}(x) \wedge P(x)) = \exists x.\mathbf{licorne}(x) \wedge \mathbf{rencontre}_*(\mathbf{pierre}, x)$

Nous supposons que pour tout prédicat δ de type $((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t)$, il existe un prédicat unique δ_* de type $e \rightarrow (e \rightarrow t)$ tel que $\delta_*(x, y) = \delta(x, \lambda X.X(y))$
Remarque : ceci n'est pas arbitraire. On peut en effet prouver au moyen de la théorie simple des types que tout objet γ de type $((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t)$ peut-être transformé en un objet γ_* de type "abaissé" $e \rightarrow (e \rightarrow t)$
Démonstration :

$$
\begin{array}{l|l}
1 & \gamma : ((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t) \\
2 & \quad y : e \\
3 & \quad \lambda X.X(y) : (e \rightarrow t) \rightarrow t \\
4 & \quad \gamma(\lambda X.X(y)) : e \rightarrow t \\
5 & \qquad x : e \\
6 & \qquad \gamma(\lambda X.X(y))(x) : t \\
7 & \quad \lambda x.\gamma(\lambda X.X(y))(x) : e \rightarrow t \\
8 & \lambda y.\lambda x.\gamma(\lambda X.X(y))(x) : e \rightarrow (e \rightarrow t)
\end{array}
$$

On vérifie bien en effet que, pour tout x et tout y :
$\gamma_*(x, y) = \gamma_*(y)(x) = [\lambda y.\lambda x.\gamma(\lambda X.X(y))(x)](y)(x)$ qui se réduit à : $\gamma(\lambda X.X(y))(x)$.
Notes : ceci appelle quelques petites remarques :

- à la ligne 3, on utilise ce théorème (cf. plus haut) selon lequel tout objet α de type a peut être transformé en un objet α' de type $(a \rightarrow b) \rightarrow b$ pour n'importe quel b. En ce cas, $\alpha' = \lambda \Xi.\Xi(\alpha)$
- par "curryfication", f de type $A \times B \rightarrow C$ est équivalente à f' de type $B \rightarrow (A \rightarrow C)$ définie par $(f'(x^B))(x^A) = f(x_A, x_B)$, c'est pourquoi on écrit $\gamma(\lambda X.X(y))(x)$ aussi bien que $\gamma(x, \lambda X.X(y))$

Postulat de signification : pour certains verbes (**aimer, rencontrer, embrasser, trouver, toucher...**) :

$$\forall x \forall X\ \delta(x, X) \Leftrightarrow X(\lambda y.\delta_*(x, y))$$

cela n'est pas vrai pour d'autres verbes (**chercher, attendre ...**)
Remarque : ceci est indépendant de l'existence de δ_* qu'on vient de prouver. Par ce postulat de signification, on ajoute une propriété caractéristique des verbes extensionnels : celle selon laquelle, pour eux, toute assertion $\delta(x, X)$, exprimant une relation entre un individu x et un ensemble de propriétés X est équivalente à une assertion exprimant une relation entre deux individus. L'équivalence précécente dit que, la propriété pour un individu y quelconque d'être en relation δ_* avec x fait partie de l'ensemble des propriétés avec lesquelle x est en relation par δ. Par exemple si X est l'ensemble des propriétés vraies d'au moins une licorne, x *touche* X si et seulement si la propriété d'être touché par x est vraie d'au moins une licorne !

Exemple 5 *à* **aime** *correspond* $\mathbf{aime}_*$
on a :

$$\mathbf{aime}(\mathbf{pierre}, \lambda P.\exists x.femme(x) \wedge P(x))$$

$$= \lambda P.\exists x.femme(x) \wedge P(x)(\lambda y.\mathbf{aime}_*(\mathbf{pierre}, y))$$
$$= \exists x.femme(x) \wedge \lambda y.\mathbf{aime}_*(\mathbf{pierre}, y)(x)$$
$$= \exists x.femme(x) \wedge \mathbf{aime}_*(\mathbf{pierre}, x)$$

1.3.3 Conséquences

- La construction donnée par la règle S7 permet de donner sa lecture de dicto à la phrase *Pierre cherche une licorne*
- dans le cas de verbe extensionnel, on obtient la lecture de re qui est la seule qui existe, grâce au **postulat de signification**
- pour l'instant, on ne sait pas obtenir la lecture de re de la phrase *Pierre cherche une licorne*

1.3.4 Cas où un ensemble de propriétés désigne un individu spécifique

On a dit :
Postulat de signification : pour certains verbes (**aimer, rencontrer, embrasser, trouver, toucher...**) :

$$\forall x \forall X\ \delta(x, X) \Leftrightarrow X(\lambda y.\delta_*(x, y))$$

cela n'est pas vrai pour d'autres verbes (**chercher, attendre ...**) **sauf si** X est l'ensemble des propriétés d'un individu bien déterminé
Exemple : si $X = \lambda P.P(\mathbf{marie})$ (*Pierre cherche Marie* s'analysera en fait comme *Pierre aime Marie*).
C'est le cas des **noms propres** et des **variables**. Par exemple il est tout le temps le cas que :

$$\delta(x, \lambda P.P(x_k)) \Leftrightarrow \lambda P.P(x_k)(\lambda y.\delta_*(x, y))$$

1.3.5 Règle de quantification

S8, n : Si $\alpha \in P_T$ et $\phi \in P_S$, alors $F_{7,n}(\alpha, \phi) \in P_S$ où $F_{7,n}(\alpha, \phi) = \phi'$ où ϕ' résulte de ϕ par la substitution suivante :

(i) si α n'est pas une variable syntaxique il_k ou $elle_k$, alors remplacer la première occurrence de il_n/$elle_n$ ou de le_n/la_n par α et les autres occurrences par le pronom anaphorique approprié (si le_n/la_n, permuter l'ordre complément-verbe)

(ii) si $\alpha = il_k/elle_k$, alors remplacer toute occurrence de $il_n/elle_n$ par $il_k/elle_k$ et de le_n/la_n par le_k/la_k

T8, n : $F_{7,n}(\alpha, \phi)^t = \alpha^t(\lambda x_n.\phi^t)$

Exemple 6

$[\lambda P.\exists x\ licorne(x) \wedge P(x)](\lambda x_1.\mathbf{cherche}_*(x_2)(x_1))$
$\rightarrow \exists x\ licorne(x) \wedge [\lambda x_1.\mathbf{cherche}_*(x_2)(x_1))](x)$
$\rightarrow \exists x\ licorne(x) \wedge \mathbf{cherche}_*(x_2)(x)$
une licorne le_2 *cherche*, S, *S8, 1*

une licorne, T — $elle_1$ le_2 *cherche*, S
$\lambda P.\exists x\ licorne(x) \wedge P(x)$ — $\mathbf{cherche}_*(x_2)(x_1)$

lecture *de re*

Exemple 7

$[\lambda P.\exists x\ licorne(x) \wedge P(x)](\lambda x_2.\mathbf{cherche}_*(x_2)(x_1))$
$\rightarrow \exists x\ licorne(x) \wedge [\lambda x_2.\mathbf{cherche}_*(x_2)(x_1))](x)$
$\rightarrow \exists x\ licorne(x) \wedge \mathbf{cherche}_*(x)(x_1)$

elle$_1$ cherche une licorne, S, S8, 2

une licorne, T — $\lambda P.\exists x\ licorne(x) \wedge P(x)$

elle$_1$ la$_2$ cherche, S — $\mathbf{cherche}_*(x_2)(x_1)$

lecture *de re*

1.3.6 Pluralité de constructions

Exemple 8 *chaque enfant admire un héros, lecture $\forall\exists$:*

chaque enfant admire un héro, S, S2
- *chaque enfant, T, S3'*
  - *chaque, T/CN*
  - *enfant, CN*
- *admire un héro, VI, S7*
  - *admire, VT*
  - *un héro, T, S3'*
    - *un, T/CN*
    - *héro, CN*

$\rightarrow \forall x.\mathbf{enfant}(x) \Rightarrow \mathbf{admire}(\lambda Q.\exists y.\mathbf{hero}(y) \wedge Q(y))(x)$ =
$\forall x.\mathbf{enfant}(x) \Rightarrow \mathbf{admire}(x, \lambda Q.\exists y.\mathbf{hero}(y) \wedge Q(y))$ =
$\forall x.\mathbf{enfant}(x) \Rightarrow \exists y.\mathbf{hero}(y) \wedge \mathbf{admire}_*(x, y)$

Exemple 9 *lecture $\exists\forall$:*

chaque enfant admire un héro, S, S8, 2
- *un héro, T, S3'*
  - *un, T/CN*
  - *héro, CN*
- *chaque enfant le$_2$ admire, S, S2*
  - *chaque enfant, T, S3'*
    - *chaque, T/CN*
    - *enfant, CN*
  - *le$_2$ admire, VI, S7*
    - *le$_2$, T*
    - *admire, VT*

$\rightarrow \exists y.\mathbf{hero}(y) \wedge \forall x.(\mathbf{enfant}(x) \Rightarrow \mathbf{admire}_*(y)(x))$

1.4 Conclusion

Nous avons vu :
- comment produire une lecture de dicto pour une phrase comportant un verbe transitif intensionnel (règle S7)
- comment produire une lecture de re pour une phrase comportant un verbe transitif intensionnel (règle S8,n)
- comment éliminer cette différence dans le cas où elle n'est pas pertinente (postulat de signification et association d'un δ_* de type "plus bas" à chaque δ)

– comment utiliser la pluralité de règles de construction pour produire des lectures variées pour un même énoncé

Le point de vue de Montague s'est donc révélé particulièrement fécond. Nous n'avons pas, dans cette section, pris en compte l'intensionnalité, c'est pourtant quelque chose qu'il convient de faire si nous souhaitons entrer dans des distinctions plus subtiles. Un autre problème concerne la place de la théorie montagovienne aujourd'hui, au regard des théories syntaxiques contemporaines. Nous regarderons donc la manière dont I. Heim et A. Kratzer ont adapté des idées très proches de celles de la grammaire de Montague dans une conception syntaxique plus moderne fournie par la grammaire générative.

2 Aperçu sur la théorie de Heim and Kratzer

2.1 Sémantique formelle et grammaire générative

La grammaire de Montague date des années 1960, époque où la grammaire générative était encore fondée sur un modèle "à base de règles" (et de transformations). Des notions fondamentales comme celles de *projection*, ou bien les opérations de *merge* (fusion) et de *move* (déplacement) n'avaient pas encore été dégagées. Beaucoup de phénomènes similaires (comme extrapositions et questions, ou bien portée des quantificateurs en position sujet ou en position objet) n'était pas traités de manière uniforme. Heim et Kratzer, en tant que générativistes, ont mis la sématique formelle "au goût du jour" en l'adaptant à la syntaxe générative "moderne". Leur approche est un peu différente de celle de Montague, par ailleurs, en ce qu'elles visent à donner directement aux phrases d'une langue une interprétation sémantique véri-conditionnelle, sans passer par un langage logique intermédiaire, comme l'est LI (logique intensionnelle) pour Montague. Nous ne suivrons pas exactement leur perspective sur ce point, la traduction en formules nous paraissant être un avantage : on peut vouloir une représentation abstraite de la signification sans nécessairement souhaiter *interpréter* les formules en termes de vérité.

Nous continuerons donc, comme dans le chapitre sur Montague, à associer aux noeuds des arbres syntaxiques, non pas des interprétations véri-conditionnelles mais des λ-termes, selon les règles suivantes :

1. noeuds terminaux :
   Si α est un noeud terminal, alors son contenu sémantique α' est fourni par le lexique,
2. noeuds non branchants :
   Si α est un noeud non branchant et si β est son noeud fille, alors $\alpha' = \beta'$,
3. noeuds branchants :
   Si α est un noeud branchant et si β et γ sont ses neuds filles, alors si β' est une fonction applicable à γ', $\alpha' = \beta'(\gamma')$.

Lorsque nous disons que β' est *applicable* à γ', nous faisons évidemment référence implicitement à la notion de *type*, comme dans la grammaire de Montague, autrement dit, nous voulons dire que β' est de type $a \rightarrow b$ et γ' de type a, mais il peut exister des cas où cette condition n'est pas suffisante (cf. le cas des articles définis).

2.2 Règles particulières : modification et abstraction

2.2.1 Modification de prédicat

Dans les approches standard de type montagovienne ou catégorielle, les modifieurs (comme les adjectifs et les adverbes) sont des expressions qui prennent en argument des expressions d'une catégorie donnée (par exemple les verbes pour les adverbes et les noms pour les adjectifs) afin de rendre en sortie une expression de la même catégorie (verbe ou nom). Heim et Kratzer ont une attitude différente, reposant sur l'opération de *conjonction*. A partir de deux prédicats, par exemple *homme* et *en bleu*, on peut fabriquer un nouveau prédicat *homme en bleu* alors que les trois expressions sont toutes de type $e \rightarrow t$. Ainsi, en un tel cas, bien que α soit un noeud branchant, aucun de ses noeuds filles ne s'applique à l'autre, d'où l'introduction d'une règle dite *de conjonction* :

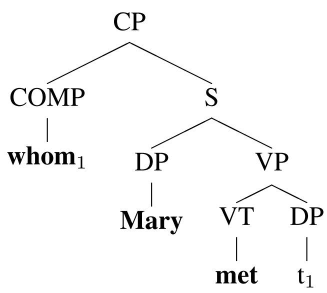
- Modification de prédicat :
  Si α est un noeud branchant et si β et γ sont ses noeuds filles, et si β' et γ' sont tous deux de type $e \rightarrow t$, alors α' = $\lambda x^e.(\beta'(x) \wedge \gamma'(x))$.

2.2.2 Variables et liage

Introduisons maintenant des opérations de déplacement dans la syntaxe, ce qui suppose l'apparition de nouveaux objets, communément appelés *traces*, dans la théorie chomskyenne. Une relative, par exemple, comme **whom Mary met**, est obtenue par une opération de déplacement (*move*) qui antépose l'objet du verbe en laissant une trace de ce déplacement à la position normale de l'objet. Dans un SN (ou DP) comme **the man whom Mary met**, **man** et **whom Mary met** sont deux propriétés qui se combinent exactement comme dans la règle de modification de prédicat. **Whom Mary met** a la représentation :

$$\textbf{whom Mary met}' = \lambda x^e.met(Mary, x) \tag{1}$$

Il reste à savoir d'où vient la variable x. En fait, les deux questions, la question syntaxique concernant *Move* et la sémantique, concernant l'introduction d'une variable x, ont une solution qui repose sur le même mécanisme.
Admettons l'analyse syntaxique suivante (1) :



La trace se traduira par une variable. D'où une règle supplémentaire concernant les noeuds :

1. noeuds terminaux non lexicaux :
   Si α est une trace t_i, alors son contenu sémantique α' est une variable nouvelle x, de sorte que s'il existe plusieurs occurrences de t_i dans l'expression, toutes soient associées à la même variable.

L'étape suivante consiste à définir le rôle de **whom**. Pour cela, Heim et Kratzer introduisent une nouvelle règle : *Abstraction de prédicat.*

Abstraction de prédicat :
Si α est un noeud branchant dont les noeuds filles sont un pronom relatif et un noeud dont le contenu sémantique dépend d'une variable x_i associée à une trace t_i que nous noterons $\beta'(x_i)$, alors α' = $\lambda x^e.\beta'(x)$

Nous pouvons alors calculer le terme associé à **whom Mary met** :

$$whom\ Mary\ met' \quad = \quad \lambda x^e.met(Mary, x)$$

A la dernière étape, nous pouvons interpréter l'expression **man whom Mary met** par *Modification de prédicat* obtenant ainsi :

$$\lambda x^e.(man(x) \wedge met(Mary, x)) \tag{2}$$

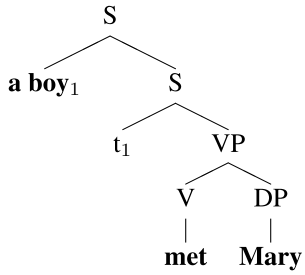


FIG. 1 – a boy met Mary

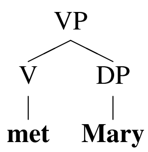
Whom est dit être un *lieur* de variable. Nous pouvons donner une définition de cette notion différente de celle donnée par Heim et Kratzer :

Définition 1 *Un* lieur *d'une variable x dans une expression est tout noeud syntaxique intervenant dans l'arbre de cette expression dont l'action sémantique se résume en une λ-abstraction sur cet x.*

On laisse au lecteur ensuite le soin d'arriver à la traduction suivante de **the man whom Mary met** :
$\iota x.(man(x) \wedge met(Mary, x))$

2.2.3 D'autres lieurs

Comme nous venons de le voir, les traductions de respectivement *the boy whom Mary met* et *Mary met a boy* sont très similaires : $\iota x.boy(x) \wedge met(Mary, x)$ et $\exists x.boy(x) \wedge met(Mary, x)$. Pour le SN (ou DP), ceci est du à l'utilisation de la règle de modification de prédicat, rendue possible par le fait que $boy(x)$ and $met(Mary, x)$ sont deux propriétés appliquées au même individu x. Dans $met(Mary, x)$, l'occurrence de x vient d'une trace en position d'objet du verbe, mais de quelle trace la variable x peut bien provenir dans le cas de la phrase ? Nous ne le voyons pas immédiatement. C'est pour introduire ce type d'analyse que R. May et R. Fiengo ont introduit l'idée que les phrases avec quantifieurs ont elles aussi (bien que ce ne soit pas apparent) ont une forme logique résultant de transformations. Ce sont les opérations de *Quantifier Raising* (montée du quantifieur). Elles reviennent à déplacer le syntagme quantifié vers une position plus haute dans la phrase, de sorte qu'il s'adjoigne au noeud S (ou au noeud IP) : voir fig. 1. Mais ce n'est pas tout, si nous comparons l'arbre de la figure 1 avec celui associé au SN (ou DP) sur la figure 2, nous pouvons voir que sur ce dernier, **whom** est le lieur, coindexé avec la trace. C'est la présence de **whom**, comme *lieur de variable*, qui transforme $Mary\ met\ t_1$ en le terme clos $\lambda x.met(Mary, x)$. Si nous voulons avoir quelque chose de semblable pour la phrase avec quantifieur, nous devons avoir un élément similaire dans son arbre, de sorte que nous puissions avoir deux termes $\lambda x.boy(x)$ and $\lambda x.met(Mary, x)$ et que le quantifieur **a**, s'appliquant aux deux donne $\exists x.boy(x) \wedge met(Mary, x)$, mais qu'est-ce qui peut jouer ce rôle de lieur dans ce cas ?
Heim et Kratzer proposent alors *d'insérer un lieur coindexé avec la trace pour tout déplacement d'un syntagme quatificationnel*, produisant ainsi l'arbre de la figure 3. Grâce à cette solution, nous voyons facilement comment la signification est obtenue :

(

$)^t = \lambda x.met(x, Mary)$

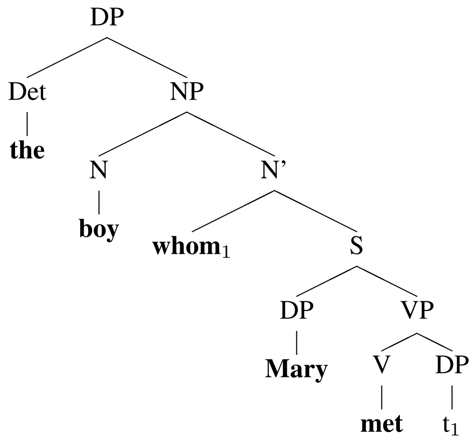


FIG. 2 – the boy whom Mary met

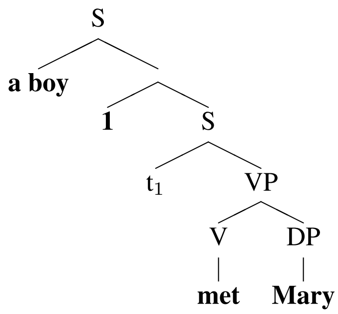
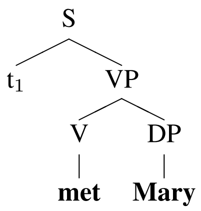
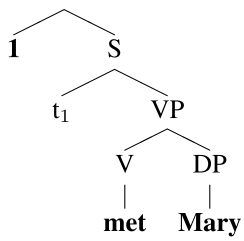
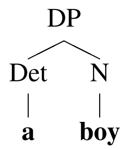


FIG. 3 – Le lieur **1**

( $)^t = met(x_1, Mary)$

Par Abstraction de prédicat :

( $)^t = \lambda x.met(x, Mary)$

( $)^t = \lambda g^{<e,t>}.\exists x.(boy(x) \wedge g(x))$

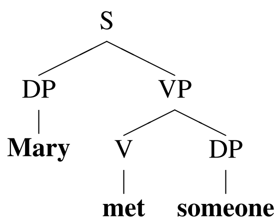
et finalement, par Application fonctionnelle :

$$\begin{aligned}
&a\ boy\ met\ Mary' = \\
&[\lambda g^{<e,t>}.\exists y.(boy(y) \wedge g(y))](\lambda x.met(x, Mary)) = \\
&\exists y.(boy(y) \wedge (\lambda x.met(x, Mary))(y)) = \\
&\exists y.(boy(y) \wedge met(y, Mary))
\end{aligned}$$

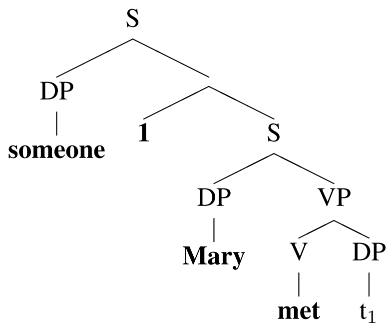
Evidemment, la règle d'Abstraction de prédicat est généralisée :

Définition 2 (Abstraction de prédicat) *Si α est un noeud branchant dont les noeuds filles sont un lieur qui porte l'indice i et un noeud de contenu sémantique $\beta(x_i)$, alors $\alpha' = \lambda x.\beta'(x)$*

Cette solution a de bons avantages. Par exemple, considérons un cas où le syntagme quantifié est en position objet. Dans un tel cas, nous avons en principe un *clash* de types :



où **someone** est de type $(e \to t) \to t$ et **met** de type $e \to (e \to t)$, empêchant ainsi toute application de la règle d'application fonctionnelle. *Quantifier raising*, avec cette introduction d'un lieur "artificiel" donne une solution correcte :



avec la signification attendue (pourvu que nous considérions que la trace t soit toujours de type e).

2.3 Structure de Surface et Forme Logique

2.3.1 Principe de liage

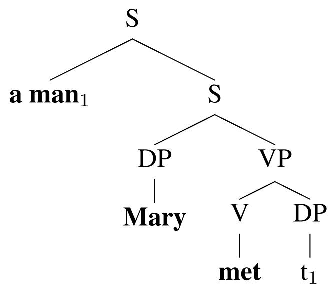
Notons que le point de vue de Heim et Kratzer se base sur une stricte distinction entre deux niveaux syntaxiques : la structure de surface (SS) et la forme logique (FL) :

– le niveau de la Structure de Surface (SS) est le niveau *syntaxique* proprement dit,
– et le niveau de la forme logique (FL) apparaît comme seulement légitimé par la transformation *QR* de montée du quantifieur, le produit de cette transformation n'étant pas visible en forme phonétique (FP)

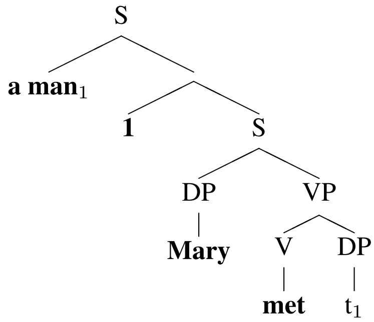
Ceci est plutôt un inconvénient de cette théorie. S'il y a en effet deux niveaux, il faut préciser rigoureusement les liens qu'ils entretiennent et définir par exemple ce qu'on entend par liage aux deux niveaux. Heim et Kratzer montrent qu'en effet le liage peut être défini à chaque niveau et elles postulent, dans leur *Principe de Liage* que les deux notions se correspondent :

> *Principe de Liage* :
> Soit α et β des DPs, où β n'est pas phonétiquement vide (autrement dit n'est pas une trace),
> alors α lie β syntaxiquement à SS si et seulement si α lie β sémantiquement à FL.

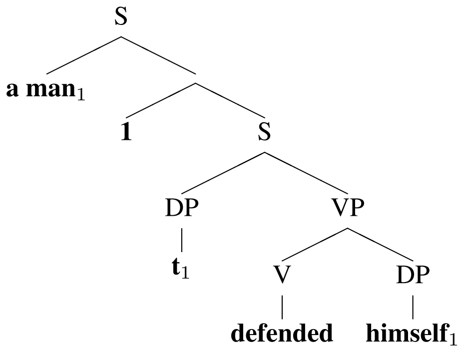
Classiquement, un DP α lie un autre DP, β, s'ils sont coindexés et si α *c-commande* β. Pour que le principe s'applique, il faut donc adapter la notion de liage sémantique. Considérons l'exemple **Mary met a man**. Au niveau de surface syntaxique, on a :



En forme logique, nous pouvons garder cette coindexation, mais nous ajoutons un noeud supplémentaire pour le lieur :



On dira que, dans cette configuration, le DP α (ici **a man**) lie sa trace t, même si c'est, en fait, le lieur **1** qui le fait. [Noter que cet abus de langage est plus clairement motivé si nous considérons des phrases avec réflexif comme **a man defended himself**, où les principes de liage usuels imposent à **himself** d'être lié par **a man**[1]. Dans ce cas, la forme logique est :



où il apparait que **a man** et **himself** sont liés par le même lieur de variable, ce qui est la seule manière de donner la lecture attendue.]
On donne donc la définition suivante du liage *sémantique* (niveau FL) entre deux DPs non vides :

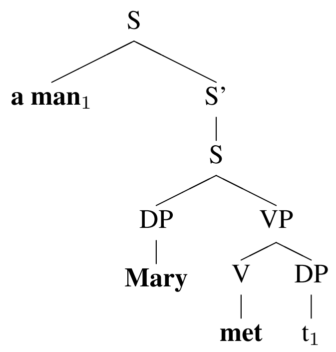
Définition 3 (Lieur sémantique) *Un DP α lie sémantiquement un DP β si et seulement si β et la trace de α sont sémantiquement liés par le même lieur de variable.*

La nécessité de *QR* serait alors une conséquence de ce principe : le quantifieur ne pourrait lier un β qu'en mettant le syntagme quantifié α dans une position où il *c-commande* β, donc en le déplaçant "vers le haut".
La question que l'on se pose alors est celle de savoir comment on pourrait se passer de cette distinction de deux niveaux (comme cela semble souhaitable dans le cadre des formulations les plus récentes du programme minimaliste). Si le niveau FL est créé seulement pour les besoins d'une transformation "invisible" QR, nous pouvons être conduits à un doute sur la nécessité des deux notions : FL et QR...
De fait, nous arrivons à une situation où l'analogie entre dérivations syntaxiques et preuves formelles peut procurer des idées intéressantes.

2.3.2 Vers une conception "théorie de la preuve" du liage

Supposons qu'au lieu d'insérer un nouveau noeud **1** en position de spécifieur de S, nous créions simplement une branche unaire :

[1]Rappelons que les principes de liage de la théorie générative contiennent en particulier le principe (A) selon lequel tout réflexif doit être lié dans sa catégorie gouvernante, ici le S minimal. Dans l'exemple, cela contraint à ce que *himself* soit lié par la trace de *a man*, d'où la coindexation.



et effectuons une rotation de 180^o de la configuration obtenue, alors nous obtenons une figure très proche d'un arbre de preuve en déduction naturelle :

$$\dfrac{\dfrac{a\ man_1}{DP} \qquad \dfrac{\dfrac{\dfrac{Mary}{DP} \quad \dfrac{\dfrac{met}{V} \quad \dfrac{t_1}{DP}}{VP}}{S}}{S'}}{S}$$

Supposons alors que les règles grammaticales soient exprimées comme des règles de déduction :

– Règles lexicales :

$$\frac{Mary}{DP} \qquad\qquad \frac{met}{V} \qquad\qquad \frac{a\ man}{DP}$$

– Règles grammaticales :

$$\frac{V\ DP}{VP} \qquad\qquad \frac{DP\ VP}{S} \qquad\qquad \frac{DP\ S'}{S}$$

Les traces sont vues comme des *hypothèses*, et la règle de déchargement de cette hypothèse est alors interprétée sémantiquement comme celle qui permet le liage de la variable introduite dans l'hypothèse :

$$\frac{A \qquad \begin{array}{c}[DP : x]\\ \vdots \\ B : \gamma\end{array}}{DP \multimap B : \lambda x.\gamma}$$

L'arbre est alors lu comme arbre de preuve.

$$\dfrac{\dfrac{a\ man_1}{DP}\,Lex \qquad \dfrac{\dfrac{\dfrac{Mary}{DP}\,Lex \quad \dfrac{\dfrac{met}{V}\,Lex \quad \dfrac{t_1 : x}{DP}\,Hyp}{VP}\,R1}{S}\,R2}{S'}\,[\,dechargement]}{S}\,R3$$

où, en fait, $S' = DP \multimap S$. Nous utiliserons $\multimap$ dans ce système dans la mesure où, à chaque pas, se trouve déchargée une hypothèse qui a été utilisée une et une seule fois (ou, en tout cas, nous voulons garder le contrôle sur le nombre de fois où cette hypothèse a été utilisée). Ceci ouvre la voie à une approche logique de la syntaxe et de la sémantique que nous examinerons dans la suite. On notera que le principe de liage s'obtient gratuitement puisque niveau de surface et niveau de forme logique se construisent ensemble comme se construisent ensemble les preuves d'un système intuitionniste et les termes d'un calcul (isomorphisme de Curry-Howard). Modulo quelques précautions à prendre concernant la linéarité, nous pouvons arriver à un principe qui peut s'énoncer ainsi :

Définition 4 (liage sémantique-2) *Un DP α lie sémantiquement un DP β sémantiquement représenté par une variable x si et seulement si α s'applique à un λ-terme $\lambda x.\Phi$ qui code une preuve dont le dernier pas consiste en l'introduction du connecteur $\multimap$.*