Partie I. Méthodes symboliques
Alexandre DIKOVSKY

Cours 3. Acquisition des types syntaxiques.

1 Grammaires des types syntaxiques

Nous allons voir trois classes des grammaires de types syntaxiques : les grammaires catégorielles, les grammaires de Lambek et les grammaires catégorielles de dépendances.

1.1 Grammaires catégorielles

Ayant été initiallement conçues par K. Ajdukiewicz [*Studia Philosophica*, 1935(1)] pour tester la correction des langages logiques, les grammaires classiques catégorielles (GC) ont été ensuite orientées par Y. Bar-Hillel [*Language*, 1953, 29(1)] vers les définitions de la syntaxe du langage naturel.

Les GC définissent chaque mot par les types syntaxiques requis à sa gauche (ou à sa droite) pour former un syntagme d'un autre type. Autrement dit, les catégories d'un mot décrivent explicitement ses valences des constituants gauches ou droits. Les types $\mathcal{T}$ de GC sont très simples :

P : Les types primitifs déterminent directement les rôles syntaxiques des mots et des syntagmes composés (par exemple, n - un nom, np - un syntagme nominal, s - une phrase).

$\mathcal{T}$: Les types composés sont construits à partir des types primitifs au moyen de deux opérateurs : $\backslash$ (pour les contextes gauches) et / (pour les contextes droits).

Plus précisement, $x\backslash y$ (y/x) est le type des syntagmes w tels que uw (respectivement, wu) a le type y pour tout syntagme u de type x. Par exemple, en anglais np/n est le type d'un déterminatif, $np\backslash s$ est le type d'un syntagme prédicatif, $(np\backslash s)/np$ est le type d'un verbe transitif, etc.

Cette sémantique des types donne deux règles universelles de leur dérivation :

$$x(x\backslash y) \vdash y \quad (\backslash_e), \qquad\qquad (y/x)x \vdash y \quad (/_e)$$

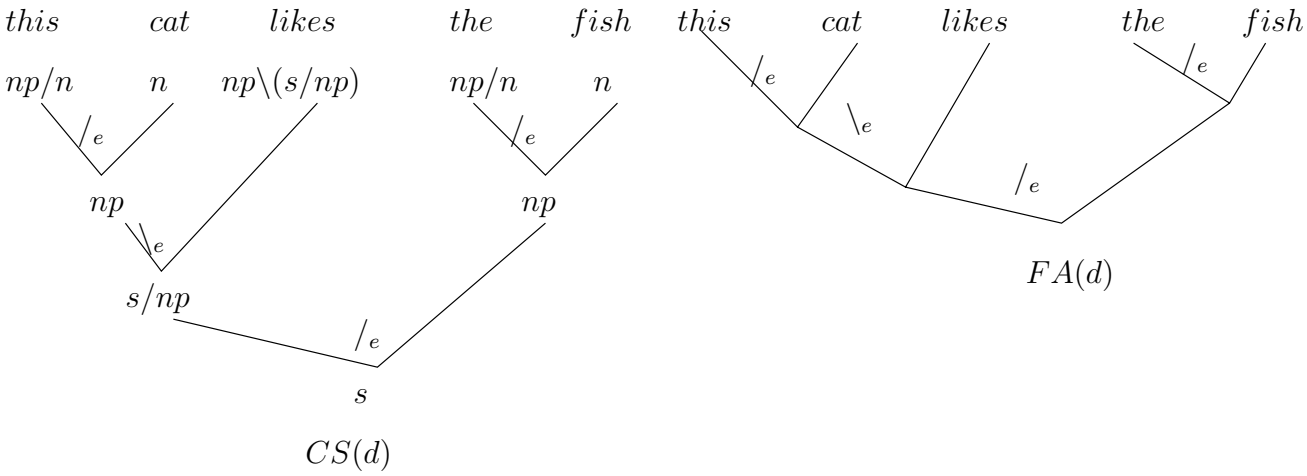
et ainsi les déductions sous la forme des arbres de réductions des types. Par exemple, la phrase *this cat likes the fish* appartient au langage déterminé par la GC dont l'interprétation lexicale g affecte à ses mots les types suivant :

$$\begin{aligned} cat &\mapsto n \\ fish &\mapsto n \\ the &\mapsto np/n \\ this &\mapsto np/n \\ likes &\mapsto np\backslash(s/np) \end{aligned}$$

Le séquent $r = ((np/n)n(np\backslash(s/np))(np/n)n \vdash s)$ est une formule qui exprime que la chaîne des types affectés à cette phrase se réduit à s. r a la dérivation d suivante :

$$\dfrac{\dfrac{\dfrac{}{np/n, n \vdash np}\,/_e}{np, np\backslash(s/np) \vdash (s/np) \quad np/n, n \vdash np}\,\backslash_e, /_e}{(s/np), np \vdash s}\,/_e$$

La dérivation d définit l'arbre unique de réduction $CS(d)$ qui est un système de constituants de la phrase. Si on supprime dans cet arbre les types, on obtient l'arbre correspondant de la structure Fonction-Argument (FA-structure) $FA(d)$:



Définition 1. *Une GC G est une substitution finie $g : W \to 2^C$ (*l'interprétation lexicale*). Le* langage de type *x déterminé par G est l'ensemble des phrases (chaînes de caractères sur W) dont une g-image se réduit à x : $L(G|x) = \{w \in W^+ \mid \exists X \in g(w)\ (X \vdash x)\}$, $L(G) = L(G|s)$ étant le* langage déterminé par *G. $CS(G)$ est l'ensemble des structures syntagmatiques (des* constituants*) affectées par G aux phrases du type s et $FA(G)$ est l'ensemble des FA-structures correspondantes.*

Bien évidemment, les GC déterminent des langages hors contexte. H. Gaifman [*I & C*, 1965, 8(3)] a prouvé que pour chaque grammaire hors contexte, il y a une GC faiblement équivalente [1].

Étant correctes par rapport à la sémantique de types, les GC ne sont pas complètes par rapport à elle. Par exemple, $u \in L(G|(p\backslash q)), v \in L(G|(q\backslash r))$ n'implique pas $uv \in L(G|(p\backslash r))$. Plusieurs complétions de GC ont été développées et étudiées, la première étant dûe à J. Lambek [*American mathematical monthly*, 1958].

[1] C'est-à-dire déterminant le même langage.

1.2 Grammaires catégorielles de Lambek

J. Lambek introduit le premier calcul logique **L** dont les règles pour $\backslash_e$ et $/_e$ sont les règles usuelles du calcul naturel de l'implication :

$$\frac{\Gamma \vdash A \quad \Delta \vdash A\backslash B}{\Gamma, \Delta \vdash B}\backslash_e \qquad \frac{[A]\Gamma \vdash B}{\Gamma \vdash A\backslash B}\backslash_i$$

$$\frac{\Delta \vdash B/A \quad \Gamma \vdash A}{\Delta, \Gamma \vdash B}/_e \qquad \frac{\Gamma, [A] \vdash B}{\Gamma \vdash B/A}/_i$$

Les grammaires de Lambek sont définies comme les GC d'Ajdukiewicz-Bar-Hillel. L'interprétation lexicale g d'une telle grammaire G associe à chaque mot (c'est-à-dire à un terminal) m_i un ensemble fini de type $g(m_i)$. Une suite de mots $s = m_1 \dots m_n$ est dans le langage $L(G)$ lorsque

$$\forall i \in [1,n]\ \exists T_i \in g(m_i) \quad (T_1 \dots T_n \vdash s).$$

Par exemple, la phrase nominale *le chat que Marie brosse* est dérivée par la grammaire de Lambek :

$$\begin{array}{rcl} chat & \mapsto & n \\ Marie & \mapsto & np \\ le & \mapsto & np/n \\ brosse & \mapsto & np\backslash(s/np) \\ que & \mapsto & (np\backslash np)/(s/np) \end{array}$$

En voici une preuve :

$$\dfrac{\dfrac{\dfrac{le}{np/n \vdash np/n}ax \quad \dfrac{chat}{n \vdash n}ax}{np/n, n \vdash np}/_e \quad \dfrac{que}{(np\backslash np)/(s/np) \vdash (np\backslash np)/(s/np)}ax \quad \dfrac{\dfrac{\dfrac{Marie}{np \vdash np}ax \quad \dfrac{brosse}{np\backslash(s/np) \vdash np\backslash(s/np)}ax}{np, np\backslash(s/np) \vdash (s/np)}\backslash_e \quad \dfrac{hypothèse}{np \vdash np}ax}{np, np\backslash(s/np), [np] \vdash s}/_e}{np, np\backslash(s/np) \vdash s/np}/_i}{(np\backslash np)/(s/np), np, np\backslash(s/np) \vdash np\backslash np}/_e}{np/n, n, (np\backslash np)/(s/np), np, np\backslash(s/np) \vdash np}\backslash_e$$

Etant donné que **L** est le fragment implicatif du calcul naturel, il est correct et complet par rapport à la conséquence sémantique des types. En particulier, $x/y, y/z \Rightarrow x/z$ et le séquent $x/y, y/z \vdash x/z$ qui exprime cette implication est dérivable dans **L**. Ainsi $u \in L(G|(p\backslash q)), v \in L(G|(q\backslash r))$ implique $uv \in L(G|(p\backslash r))$ dans les grammaires de Lambek.

Les grammaires de Lambek sont devenues la première classe des grammaires logiques qui sont complètement lexicalisées et dont les catégories sont interprétées par les formules. Par ailleurs, elles s'avèrent faiblement équivalentes aux grammaires hors contexte [W.Buszkowski, *Bull. Acad. Pol. Sci.(Math.)*. 1986 ; M. Pentus, *Proc. 8th IEEE Symp. on Logic in CS*,1993].

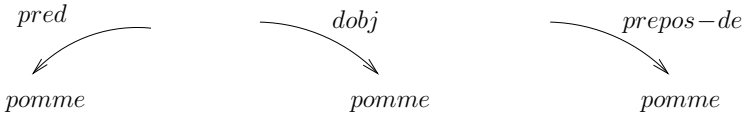
1.3 Grammaires catégorielles de dépendances

Types de dépendances. Les dépendances sont les relations binaires de surface entre les mots. $m_1 \xrightarrow{d} m_2$ veut dire $\ll m_1$ autorise ou réclame m_2 en fonction de la dépendance $d \gg$ (m_1 est le gouverneur, m_2 est le subordonné). Par exemple, $voit \xrightarrow{dobj} Marie$ est la

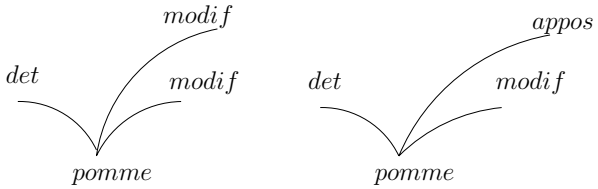
dépendance *dobj* entre le verbe *voit* et son objet direct *Marie*. Dans les langues naturelles, la majorité des structures de dépendances sont les arbres (arbres de dépendances).

Les types de dépendances d'un mot m sont définis en tenant compte de deux statuts de m : en tant que gouverneur (débuts des fléches) et en tant que subordonné (fins des fléches).

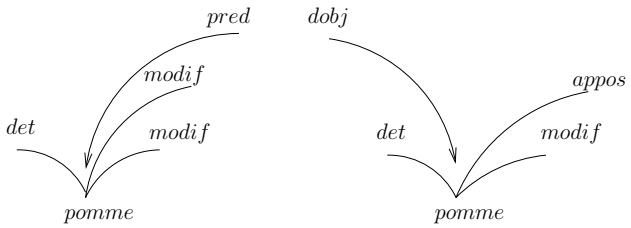
Par exemple, le mot français *pomme* en tant que subordonné peut avoir les dépendances :



Le même mot *pomme* en tant que gouverneur peut avoir les dépendances :

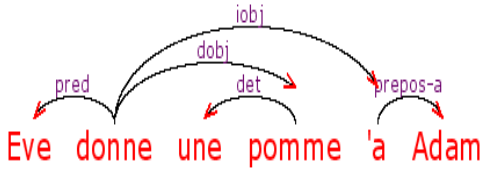


Les types de dépendances sont les combinaisons compatibles des deux statuts :



La première combinaison a le type $[det\backslash pred/modif*]$, la deuxième a le type $[det\backslash dobj/appos/modif*]$. Par exemple, le deuxième type réclame à gauche un subordonné avec la dépendance *det* (déterminant) et à droite il admet zero ou plusieurs subordonnés via la dépendance itérée *modif*$*$ (modifieur) et réclame un subordonné avec la dépendance *appos* (appositif), dans cet ordre, pour devenir un objet direct subordonné via la dépendance *dobj*.

Les types comme ça définissent les dépendances locales continues des mots.



Pour définir les dépendances distantes eventuellement discontinues (non projectives), on utilise les types (valences) polarisés.

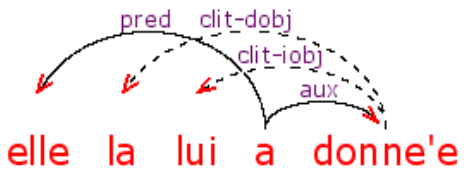
Polarités de dépendances distantes (discontinues).

Valence gauche négative : $\swarrow prepos$: celle d'un mot dont le gouverneur à travers la dépendance *prepos* ce trouve à droite,

Valence gauche positive : $\nearrow prepos$: celle d'un mot dont le subordonné à travers la dépendance *prepos* ce trouve à droite.

Valences duales ($\breve{v}$) : ($\swarrow prepos$) et ($\nwarrow prepos$), ($\nearrow prepos$) et ($\searrow prepos$). Ensemble, elles définissent la dépendance distante *prepos* à condition que l'une soit la première disponible

(*first available* : FA) pour l'autre.
Valences négatives ancrées (à un type t) : réclament les fins des dépendances distantes définies par les valences négatives voisines à t (à gauche ou à droite).
Par exemple, en français le type $\#^l(\swarrow clit{-}iobj)$ d'un objet indirect pronominalisé (un clitique) s'ancre (à gauche) à un type (au type d'un verbe auxiliaire ou principal) :



La grammaire catégorielle de dépendances qui définit cet arbre de dépendances affecte les types suivant aux mots de cette phrase :

$$\begin{array}{rcl} elle & \mapsto & [pred] \\ la & \mapsto & [\#^l(\swarrow clit{-}dobj)] \\ lui & \mapsto & [\#^l(\swarrow clit{-}iobj)] \\ a & \mapsto & [\flat^l(\swarrow clit{-}iobj)\backslash\flat^l(\swarrow clit{-}dobj)\backslash pred\backslash S/aux] \\ donne'e & \mapsto & [\nwarrow clit{-}iobj\backslash \nwarrow clit{-}dobj\backslash aux] \end{array}$$

Tout comme les GC, une grammaire catégorielle de dépendances (GCD) [A. Dikovsky, COLING Workshop on DG, 2004] est définie par une interprétation lexicale, c'est-à-dire par une substitution finie $g : W \to 2^{types(C)}$. Une phrase w appartient au langage $L(G)$, s'il y a une preuve $\alpha_1...\alpha_n \vdash S$ pour une chaîne de types $\alpha_1...\alpha_n \in g(w)$ dans le calcul suivant de dépendances.

Définition 2 (M. Dekhtyar, A. Dikovsky, 1st Int. Conf. on CG, 2004). [2].

Calcul de dépendances $\mathcal{D}^{\mathbf{FA}}$

Types $[L_1\backslash\cdots\backslash L_i\backslash C/R_j/\cdots/R_1]$:
C *: neutre, négatif* $(\swarrow C, \searrow C)$ *ou ancré* $(\#(\swarrow C), \#(\searrow C))$
L_i, R_j *: neutre, positif* $(\nwarrow C, \nearrow C)$ *ou hôte* $(\flat(\swarrow C), \flat(\searrow C))$

- $\mathbf{L^l}$. $C[C\backslash\beta] \vdash [\beta]$
- $\mathbf{I^l}$. $C[C^*\backslash\beta] \vdash [C^*\backslash\beta]$
- $\mathbf{\Omega^l}$. $[C^*\backslash\beta] \vdash \beta$
- $\mathbf{V^l}$. $[\alpha\backslash\beta] \vdash \alpha[\beta]$, $\alpha \in \{\nwarrow C, \flat^l(\swarrow C), \flat^l(\searrow C)\}$
- $\mathbf{A^l}$. $\#^l(\alpha)\flat^l(\alpha) \vdash \alpha$, $\alpha \in \{\swarrow C, \searrow C\}$
- $\mathbf{C^l}$. $\alpha\beta \vdash \beta\alpha$, $\alpha \in \{\swarrow C, \nwarrow C, \nearrow C, \searrow C\}$, *où* $\beta = \flat(v)$ *ou* β n'a pas d'occurrence de $\alpha, \breve{\alpha}, \#(\alpha), \flat(\alpha)$
- $\mathbf{D^l}$. $\swarrow C \nwarrow C \vdash \varepsilon$

Exemple 1. *CDG pour* $\{a^n b^n c^n \mid n > 0\}$

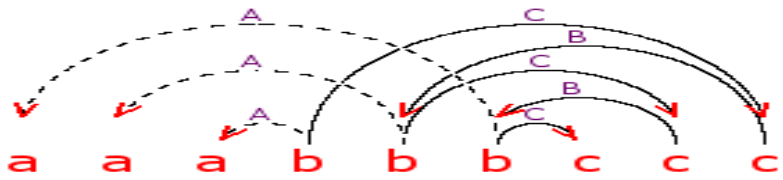
$$\left\{\begin{array}{rcl} a & \mapsto & \#(\swarrow A),\ [\flat(\swarrow A)\backslash\#(\swarrow A)] \\ b & \mapsto & [\nwarrow A\backslash B/C],\ [\flat(\swarrow A)\backslash \nwarrow A\backslash S/C] \\ c & \mapsto & C,\ [B\backslash C] \end{array}\right.$$

[2] On présente les règles pour les arguments gauches ; les règles des arguments droits sont symétriques.

Affectation des types *:*

$a^3b^3c^3 \mapsto \#(\swarrow A)[\flat(\swarrow A)\backslash\#(\swarrow A)][\flat(\swarrow A)\backslash\#(\swarrow A)][\flat(\swarrow A)\backslash\nwarrow A\backslash S/C][\nwarrow A\backslash B/C][\nwarrow A\backslash B/C]C[B\backslash C][B\backslash C]$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\#(\swarrow A),[\flat(\swarrow A)\backslash\#(\swarrow A)],[\flat(\swarrow A)\backslash\#(\swarrow A)] \qquad \flat(\swarrow A),\nwarrow A,\nwarrow A,\nwarrow A,S}{S}(\mathbf{V}^l)^+,(\mathbf{A}^r)^+}{\flat(\swarrow A),\nwarrow A,[S/C]}(\mathbf{V}^r,\mathbf{C}^l)^+ \qquad \dfrac{\dfrac{\dfrac{[\nwarrow A\backslash B/C]}{\nwarrow A,[B/C]}(\mathbf{V}^l) \qquad \dfrac{\dfrac{\dfrac{[\nwarrow A\backslash B/C]C}{[\nwarrow A\backslash B]}(\mathbf{L}^r)}{\nwarrow A,B}(\mathbf{V}^l) \qquad \dfrac{[B\backslash C]}{\nwarrow A,C}(\mathbf{L}^l)}{\nwarrow A,\nwarrow A,B}(\mathbf{C}^l,\mathbf{L}^r) \qquad [B\backslash C]}{\nwarrow A,\nwarrow A,C}(\mathbf{L}^l)}{\nwarrow A,\nwarrow A,\nwarrow A,C}(\mathbf{C}^l)^+,(\mathbf{L}^r)$$



Les GCD sont faiblement plus expressives que les grammaires hors contexte (par conséquent, que les GC). Par ailleurs, il existe un algorithme de l'analyse syntaxique des CDG en temps polynômial ($\mathcal{O}(n^3)$ en pratique) [M.Dekhtyar, A.Dikovsky, ibid]. Pour comparer : les GC ont la même complexité, mais elles sont moins expressives, tandis que l'analyse syntaxique des grammaires de Lambek est NP-complète [M. Pentus. 2003, *Tech. Rep. CUNY*].

2 Grammaires rigides et k-valuées des types

Certes, les GC, les grammaires de Lambek et les GCD ne sont pas P-apprenables, parce que les familles indexées correspondantes contiennent des points limites (langages infinis et ses sous-ensembles finis). En revanche, il existe les sous-familles intéressantes de ces familles qui sont P-apprenables à des conditions naturelles.

Définition 3. *Une grammaire de types G avec une interprétation lexicale g est* k-valuée *($k \geq 1$) si $|g(m)| \leq k$, quel que soit le mot $m \in W$. G k-valuée est* rigide *si $k = 1$.*

En considérant l'apprenabilité des grammaires de types, il est très utile d'utiliser les variables à la place des types primitifs. Alors, nous allons suivre la convention que l'ensemble C des types primitifs ci-dessous sont constitués des variables et d'une seule constante S : $C = \{S, x, y, z, \ldots\}$. Respectivement, les types sont construits de ces éléments : $S, x\backslash(y/z), [x/\nearrow y], [x\backslash S/\#(\searrow u)], \ldots$ etc.

L'intérêt de cette convention est qu'on peut appliquer les substitutions aux grammaires. Soit $\theta : C \to types(C)$. Alors

$$\begin{array}{rcl} \theta(S) & = & S, \\ \theta(A\backslash B) & = & \theta(A)\backslash\theta(B), \\ \theta(A/B) & = & \theta(A)/\theta(B), \end{array}$$

pour $A, B \in types(C)$.

Exemple 2. *Soient la substitution $\theta = \{y \mapsto x\backslash S\}$ et les grammaires*

$$G_1 : \left\{ \begin{array}{rcl} \textbf{fast} & \mapsto & y\backslash(x\backslash S), \\ \textbf{John} & \mapsto & x, \\ \textbf{walks} & \mapsto & x\backslash S,\ y \end{array} \right. \quad et \quad G_2 : \left\{ \begin{array}{rcl} \textbf{fast} & \mapsto & (x\backslash S)\backslash(x\backslash S), \\ \textbf{John} & \mapsto & x, \\ \textbf{walks} & \mapsto & x\backslash S \end{array} \right.$$

Alors, G_1 est 2-valuée, G_2 est rigide et $G_2 = \theta[G_1]$.

Lemme 1 (Buszkowski & Penn, *Stud. Log.*, 1990, 49).
1. Si $\theta[G_1] \subseteq G_2$, alors $FA(G_1) \subseteq FA(G_2)$.
2. $\theta[G_1] \subseteq G_2$ et $\theta[G_2] \subseteq G_1$ implique $G_1 \equiv G_2$ [3].

Vérifiez le point 1 pour G_1, G_2.

Définition 4. *1.Une substitution θ est* fidèle *pour G si pour tout mot $w \in W$ et tout types $A, B \in g(w)$, si $A \neq B$, alors $\theta(A) \neq \theta(B)$.*
2. $G_1 \sqsubseteq G_2$ veut dire qu'il existe θ **fidèle** *pour G_1 telle que $\theta[G_1] \subseteq G_2$.*
$G_1 \sqsubset G_2 =_{df} G_1 \sqsubseteq G_2 \ \wedge \ G_1 \not\equiv G_2$.

Lemme 2 (M.Kanazawa, ibid).
(i) $\sqsubset$ est un ordre bien fondé.
(ii) Si $G_1 \sqsubseteq G_2$ alors $|G_1| \leq |G_2|$.
(iii) $\{G' \mid G' \sqsubseteq G\}$ est fini.

Lemme 3 (M.Kanazawa, ibid).
Soient une GC rigide G_1 sans types inutiles [4] *et une GC G_2 rigide. Alors $FA(G_1) \subseteq FA(G_2)$ ssi $G_1 \sqsubseteq G_2$.*

Lemme 4 (M.Kanazawa, ibid). *Soient des GC rigides $G_0, G_1, \ldots, G_n$ sur W sans types inutiles qui forment la suite ascendante $G_0 \sqsubset G_1 \sqsubset \ldots \sqsubset G_n$. Alors $n \leq |W|$.*

2.1 P-acquisition des CG rigides à partir d'exemples structurés

Nouvelle idée : D'apprendre la GC cible G_c à partir d'énumérations de l'ensemble de ses structures foncteur-argument $FA(G_c)$.

Définition 5. *Un algorithme M* apprend *une GC cible G_c* à partir d'exemples structurés, *si pour toute séquence*

$$(2') \qquad D = f_1, f_2, f_3, \ldots \text{ telle que } \{f_1, f_2, f_3, \ldots\} = FA(G_c).$$

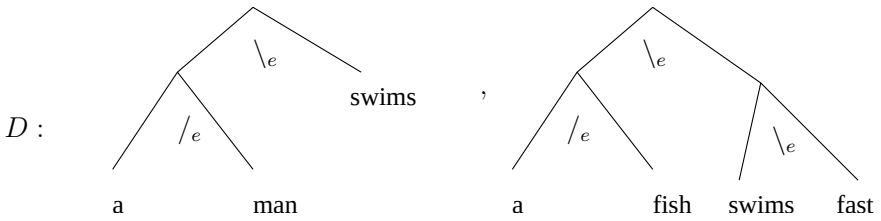
il existe un pas de stabilisation K tel que :

$$\forall\ k \geq K\ (M(D[k]) = G_{h_K} \ \wedge \ L(G_{h_K}) = L(G_c)).$$

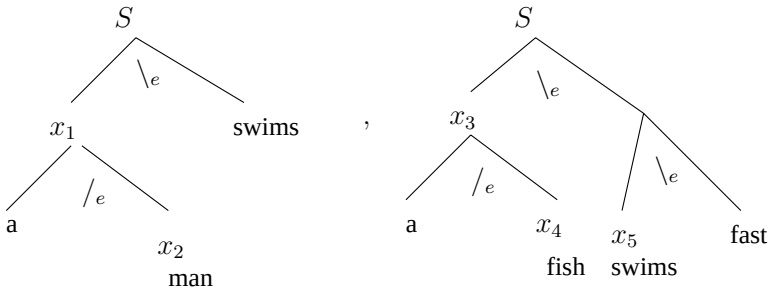
Algorithme RG d'apprentissage des GC rigides à partir d'exemples structurés [Buszkowski & Penn, ibid].
Entrée : une séquence d'exemples $D[k] \subseteq FA(G_c)$ pour une GC cible rigide G_c.
Sortie : une GC rigide G telle que $D \subseteq FA(G)$.

Nous allons définir **RG** sur la séquence :

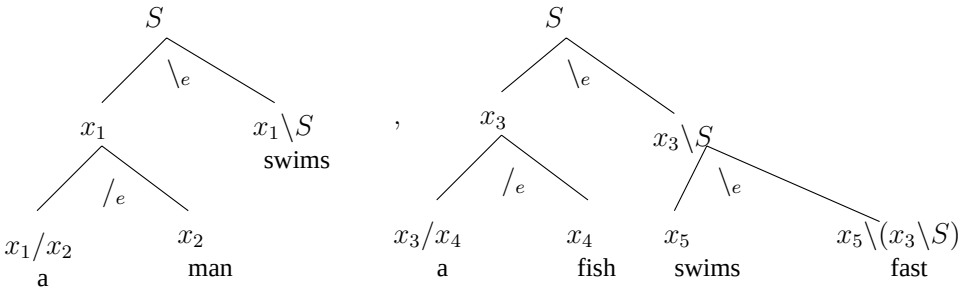
[3] Egales à un renommage de variables près. Certes, $G_1 \equiv G_2$ implique $L(G_1) = L(G_2)$.
[4] C'est-à-dire sans types non utilisés dans les dérivations $g(w) \vdash S$.



Phase 1. (a) Etiquetter les racines des arbres dans D par le type S.
(b) Affecter aux nœuds argumentaux [5] des variables deux-à-deux différentes :



(c) Calculer les types des nœuds fonctionnels correspondants :



Phase 2. Regrouper les affectations des types aux mêmes mots dans l'interprétation lexicale obtenues au pas (c) :

$$GF(D)=_{df}\begin{cases}\textbf{a} & \mapsto & x_1/x_2,\ x_3/x_4,\\ \textbf{fast} & \mapsto & x_5\backslash(x_3\backslash S),\\ \textbf{fish} & \mapsto & x_4\\ \textbf{man} & \mapsto & x_2\\ \textbf{swims} & \mapsto & x_1\backslash S,\ x_5.\end{cases}$$

Phase 3. (a) Pour tout mot m, regrouper dans l'ensemble A_m les types affectés au mot m : $A_m=_{df}\{t \mid GF(D) : m \mapsto t\}$,
(b) Calculer le MGU (l'unificateur le plus général) θ_m de A_m pour tout m et
(c) Composer les MGUs ainsi obtenus : $\theta=_{df}\theta_{m_1}\circ\ldots\circ\theta_{m_r}$.
Dans notre exemple : $\theta=\{x_3\mapsto x_1, x_4\mapsto x_2, x_5\mapsto x_1\backslash S\}$.

RG plante si θ n'existe pas.

Phase 4. RENDRE $RG(D)=_{df}\theta[GF(D)]$.

Dans notre exemple :

[5] Pour le père $\backslash_e$ c'est son fils gauche et pour le père $/_e$ c'est son fils droite.

$$RG(D) = \left\{ \begin{array}{rcl} \textbf{a} & \mapsto & x_1/x_2, \\ \textbf{fast} & \mapsto & (x_1\backslash S)\backslash(x_1\backslash S), \\ \textbf{fish} & \mapsto & x_2 \\ \textbf{man} & \mapsto & x_2 \\ \textbf{swims} & \mapsto & x_1\backslash S. \end{array} \right.$$

Propriétés de RG.

Lemme 5 (Buszkowski & Penn, ibid).
(1) Pour toute GC rigide G et tout ensemble fini D de FA-structures si $RG(D)$ est défini, alors $D \subseteq FA(RG(D))$.
(2) Si $RG(D)$ existe, alors elle est rigide et sans types inutiles.

Lemme 6 (Buszkowski & Penn, ibid).
Pour toute GC rigide G et tout ensemble fini D de FA-structures, $D \subseteq FA(G)$ ssi $RG(D)$ est défini et $RG(D) \sqsubseteq G$.

Corollaire 1. *Soit une GC rigide et deux ensembles finis D_1, D_2 de FA-structures. Si $D_1 \subseteq D_2 \subseteq FA(G)$ alors $RG(F_1)$ et $RG(F_2)$ sont définis et $RG(D_1) \sqsubseteq RG(D_2) \sqsubseteq G$.*

Théorème 1 (M.Kanazawa, ibid). *L'algorithme* **RG** *apprend la famille des GC rigides à partir d'exemples structurés.*

Preuve. Soit une GC cible rigide G_c. On peut supposer sans perte de généralité que G_c n'a pas de types inutiles.[6] Soit une énumération $D = \{f_1, f_2, \ldots\}$ de $FA(G_c)$. Alors, $D[i] \subseteq D[i+1] \subseteq FA(G_c), i \geq 1$. D'après le corollaire 1, $RG(D[i])$ et $RG(D[i+1])$ sont définis et $RG(D[i]) \sqsubseteq RG(D[i+1]) \sqsubseteq G_c$. Selon le lemme 2, l'ensemble des GC $RG(D[i]), i \geq 1$ est fini. C'est pourquoi RG converge sur D vers une GC G_K à partir d'un pas K : $RG(D[i]) = G_K$ pour tout $i \geq K$. $D[i] \subseteq FA(G_c)$ implique que $RG(D[i])$ est défini. D'après le lemme 5(1), $D[i] \subseteq FA(RG(D[i]))$. Ensuite, d'après le lemme 1(1), $RG(D[i]) \sqsubseteq RG(D[i+1])$ implique $FA(RG(D[i])) \subseteq FA(RG(D[i+1]))$, ce qui veut dire que $FA(G_c) = D \subseteq FA(RG(D[K])) = FA(G_K)$. G_c étant rigide sans types inutiles et $G_K = RG(D)$ étant rigide (voir le lemme 5(2)), d'après le lemme 3, $G_c \sqsubseteq G_K$. D'un autre côté, on a vu ci-dessus que $G_K = RG(D[K]) \sqsubseteq G_c$. C'est-à-dire, $G_c \equiv G_K$ et par conséquent (voir le lemme 1(2)), $L(G_K) = L(G_c)$. □

2.2 P-acquisition des GC rigides à partir de chaînes des caractères

Commencons par établir une élasticité finie de la famille $\mathcal{F}_r^{CG}$ des FA-langages des GC rigides.

Théorème 2 (M.Kanazawa, ibid). *$\mathcal{F}_r^{CG}$ a une élasticité finie.*

Preuve. Soit $F_0, F_1, F_2, \ldots \in \mathcal{F}_r^{CG}$ et une séquence de FA-structures $T_0, T_1, T_2, \ldots$ telles que pour tous $0 \leq i \leq n$,

$$T_i \notin F_i$$

[6]Pour toute GC il y a une GC sans types inutiles qui a le même langage FA.

et

$$\{T_0, \ldots, T_i\} \subseteq F_{i+1}.$$

Alors selon le corollaire 1 et le lemme 1(1), $RG(\{T_0, \ldots, T_i\})$ est défini et $FA(RG(\{T_0, \ldots, T_i\})) \subseteq F_{i+1}$ pour $0 \leq i \leq n$. Ainsi $T_{i+1} \notin FA(RG(\{T_0, \ldots, T_i\}))$. Cela veut dire, $RG(\{T_0, \ldots, T_i\}) \sqsubset RG(\{T_0, \ldots, T_{i+1}\})$ pour $0 \leq i \leq n$. Alors, selon le Lemme 4, $n + 1 \leq |W|$. L'élasticité de $\mathcal{F}_r^{CG}$ est donc finie. □

Remarque. La relation

$$x \; R_{FA} \; T \; \Leftrightarrow T \; est \; une \; FA-structure \; sur \; x$$

est F-valuée. Alors, d'après le Théorème 5 du cours 2, la famille

$$\{R_{FA}^{-1}(F) \mid F \in \mathcal{F}_r^{CG}\}$$

des langages engendrés par les GC rigides a une élasticité finie.

Corollaire 2 (M.Kanazawa, ibid). *Les GC rigides sont P-apprenables à partir de chaînes des caractères.*

2.3 P-acquisition des GC k-valuées

Remarque. Soit une GC k-valuée pour un $k > 1$ sur un alphabet W. Soit W_c l'alphabet des copies : $c^{(i)}, 1 \leq i \leq k$, pour tout $c \in W$. Alors, on peut définir l'homomorphisme $h(c^{(i)}) = c$ et pour toute GC k-valuée G sur W avec son interprétation lexicale g_G définir une GC rigide G^R sur W_c avec son interprétation lexicale g_G^R :

$$g_G^R(c^{(i)}) =_{df} A_i, \text{ où } g_G(c) = \{A_1, \ldots, A_k\}.$$

Evidémment, $FA(G) = h(FA(G^R))$. Par conséquent, selon le même Théorème 5 du cours 2, nous obtenons :

Théorème 3 (M.Kanazawa, ibid). *La famille $\mathcal{F}_{(k)}^{CG}$ des FA-langages des GC k-valuées a une élasticité finie.*

Corollaire 3 (M.Kanazawa, ibid). *Les GC k-valuées sont P-apprenables à partir de chaînes des caractères.*

2.4 P-acquisition des grammaires de Lambek k-valuées

Contrairement aux GC classiques, les grammaires de Lambek rigides ou k-valuées ne sont pas apprenables à partir de chaînes des caractères.

Théorème 4 (A.Foret & Y. Le Nir, *COLING'2002*). *La famille des grammaires de Lambek rigides a des points limites.*

Ceci est vrai en dépit du fait que cette famille est bien apprenable à partir d'exemples FA-structurés.

Théorème 5 (C.Retoré & R.Bonato, *3d Workshop on Learning Language in Logic*, **2001).** *La famille des grammaires de Lambek rigides est apprenable à partir d'exemples FA-structurés.*

Les structures FA des grammaires de Lambek sont similaires à celles des GC. La seule différence est que outre les étiquettes des nœuds d'élimination il y a aussi les étiquettes des nœuds d'introduction. Alors, on utilise pour l'apprentissage le même algorithme de Buszkowski. Pourtant, la réduction de Kanazawa ne marche plus, car pour les grammaires de Lambek il n'y a pas de relation F-valuée qui relie les langages FA et les langages des chaînes.