Les techniques du T.A.L.

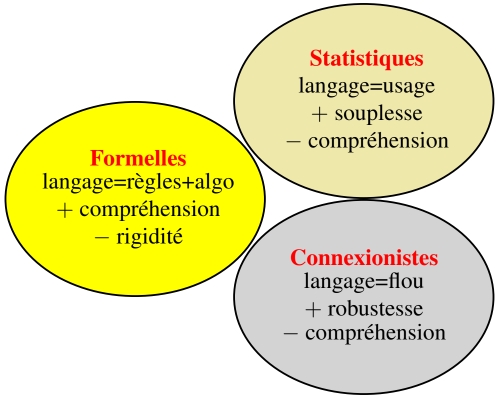
Cours ENST – TALN
Vendredi 8 Mars 2002

Éric de la Clergerie
Eric.De_La_Clergerie@inria.fr

Transparents disponibles sur http://atoll.inria.fr/~clerger ▶ Enseignement

Les approches

Les langues sont des objets difficiles à cerner ⇒ plusieurs approches



Les approches se combinent : formelle + statistique (grammaires stochastiques)

Quelques lignes de coupures

parole / écrit

analyse / génération

langue générale / langues spécialisées et langage contrôlé

De nombreux champs d'application

- correction orthographique ; numérisation avec correction
- correction grammaticale
- aide à la rédaction (stylistique ; langage contrôlé)
- reconnaissance vocale ; dictée vocale (mono ou multi locuteurs)
- synthèse vocale (avec prosodie)
- recherche d'information
- extraction d'information ; fouille de textes
- résumés
- analyse thématique
- recherche ruptures dans un document
- génération de textes
- communication homme-machine
- traduction (complète ou approximative)

Niveaux de traitement

Présentation des technologies informatiques utilisées pour certaines étapes du traitement linguistique.

Analyse lexicale **Segmenteurs** en phrases et mots.

Analyse Morphologique Recherche de la catégorie lexicale, la racine et la forme des mots.

Etiquetage Lexical Désambiguation sur les catégories lexicales (N,V,Det,Prep, ...).

La<Det N Pron> **belle**<Adj N> **montre**<N V> **le**<Det Pron> **chemin**<N>

Oracle pour les mots inconnus (noms propres, acronymes, ...)

Analyse syntaxique Recherche de la structures grammaticale des phrases.

« chunking » délimitation et identification des syntagmes non récursifs

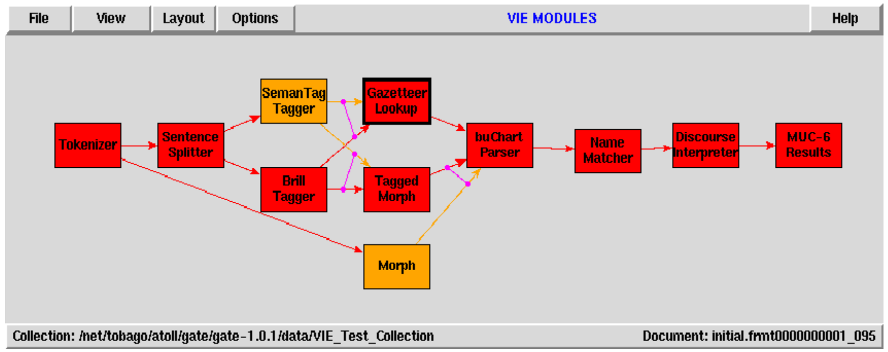
[Il]<SN> [a regardé]<SV> [un homme]<SN> [avec un télescope]<SP>

analyse profonde Structure grammaticale complète

Analyse Sémantique Sens des phrases

Analyse Pragmatique Prise en compte du contexte et de la connaissance du monde.

Chaîne de traitement linguistique



Capture d'écran du système **GATE** développé à l'université de Sheffield (UK).

Traitements locaux

Approches priviliégiant des traitements linguistiques

locaux actions prises en fonction du « mot » courant et des « mots » dans un contexte proche.

non récursifs Pas de régles de la forme $NP \longrightarrow NPPP$.

⇒ traitements en temps linéaire, en fonction de la longueur de la phrase

Technologies :

- Modèles de markov cachés (HMM) ou N-grammes
- Automates et transducers à états finis (FSA et FST), éventuellement pondérés
- Règles symboliques « locales »

Achitecture de traitement en cascade : le flux de sortie d'un traitement sert d'entrée au suivant.

Etiquetage par N-grammes

Les étiquettes sont déterminées selon un modèle probabiliste de **Chaîne cachée de Markow** prenant en compte des fenêtres de N mots.

Seuls les N-1 derniers mots et/ou étiquettes sont nécessaires pour calculer l'étiquette la plus probable du Nème mot.

En général, $N = 2$ (bigrammes) ou $N = 3$ (trigrammes).

$$P(w_{i+1} = \text{le}, e_{i+1} = \text{Pronom} \| e_i = \text{Verb}) \qquad P(w_{i+1} = \text{le}, e_{i+1} = \text{Art} \| e_i = \text{Verb})$$

Pour une phrase donnée, on calcule la séquence d'étiquettes $e^{1,n}$ qui maximise la probabilité de

$$\Pi_{i=1}^{i=n} P(w_{i+1}, e_{i+1} = e^{i+1} | e_i = e^i)$$

Cette séquence est calculable en $O(n)$ par un algorithme type Viterbi.

N-grammes et apprentissage

Les N-grammes sont extraits de corpus annotés
Exemples :

- Brown Corpus [500 Doc, 15 Genres, 1Mmots]
- British National Corpus [100 Mmots]
- Wall Street Journal, Le Monde

Problèmes :

1. Temps important d'annotation
2. Constitution des corpus
3. Problèmes des données rares (sparse data) : certains N-grammes peuvent ne pas être présents dans un corpus.
4. Caractère non linguistique des N-grammes.

Etiquetage par Règles de Transformation (Tagger de Brill)

1. Initialisation des mots avec l'étiquette la plus probable (indépendemment des autres mots).
2. Modification des étiquettes en appliquant dans l'ordre des règles de transformation.

Les règles de transformations sont produites par apprentissage sur corpus annotés et supervision de linguistes.

Elles portent sur les étiquettes et mots voisins, par exemple :

> Changer l'étiquette VERBE en NOM si le mot précédent est un ARTICLE (par exemple dans "je vois une<Det> **montre**<V>").

L'apprentissage donne un faible ensemble de transformations, motivées linguistiquement et l'étiquetage est de très bonne qualité.

Conserver l'ambiguité lexicale

Une autre approche consiste à conserver l'ambiguité lexicale et à passer un treillis de possibilité à l'analyse syntaxique.

la belle montre le chemin

Ce treillis est un cas particulier d'automate à état finis (non-déterministe).

Permet la représentation les mots inconnus, des trous ou d'associer des probabilités (texte parlé).

la belle mantre le chemin

Exemple : étiquetage

Sortie de Tree Tagger sur « La belle montre la porte »

```
La      DET:def   le
belle   ADJ       belle
montre  VER:pres  montrer
la      DET:def   le
porte   NOM       porte
.       PON:sep   .
```

Tree Tagger `http://www.ims.uni-stuttgart.de/Tools/DecisionTreeTagger.html` librement disponible pour Français, Anglais, Allemand et Italien.

Exemple : analyse morphologique

Sortie de Flemm sur « La belle montre la porte » étiquetée par TreeTagger :

```
La       DET(def):f:s    le
belle    ADJ:f:s         beau
montre   VER(pres):{{1|3}p:s:pst:{ind|subj}|2p:s:pst:imper}         montrer:1g
la       DET(def):f:s    le
porte    NOM:_:s         porte
.        PON:sep         .
```

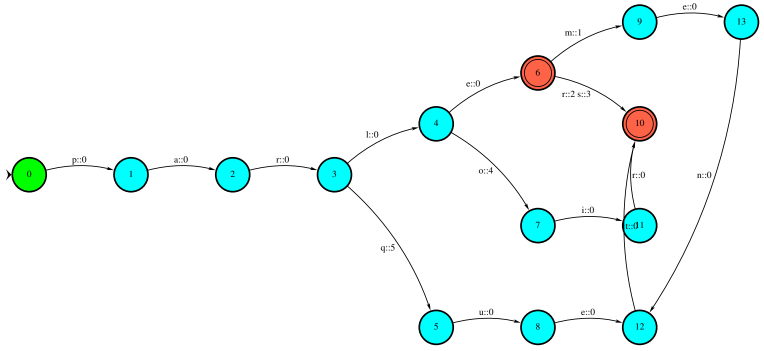
Flemm `http://www.univ-nancy2.fr/pers/namer/Telecharger_Flemm.htm` est un analyseur morphologique pour le français écrit en Perl et pouvant gérer les mots inconnus.

Automates et transducers à états finis

Technologie très efficace pour

- lexiques
- morphologie
- détection de frontières et de constituants non récursifs

```
parle
parlement
parler
parles
parloir
parquet
```



Extension des FST

Boite à outils XEROX (voir aussi AT&T Bell Labs et INTEX)

Opérateurs traditionnels : disjonction |, concatenation, optionel ?, répétition * et +,

Extensions1 : Complément, intersection, soustraction $(A - B)$, produits cartésien, composition

Extension2 :

- **$A** les chaînes contenant une sous-chaîne vérifiant A.
- **A => L_R** A avec un context gauche L et droit R
  Dans le cas de transducteurs, variantes pour tester le contexte avant ou après transduction
- **A −>B** remplacement de A par B
- **A −> B ... C** Balisage de A par B et C, donne BAC
- Variantes @−> de plus long remplacement ou balisage.

FST : exemple

Day = Monday | Tuesday | .... | Sunday
Month = January | .... | December
Date = 1To (| [!|2] 0 To9 | 3 [0|1]
Year = 1To9 (0To9 (0To9 (0To9)))
AllDates = Day | (Day ",") Month " " Date (", " Year)
AllDates @−> "<date>" ... "</ date>"

Today is <date>Tuesday, July 25, 2000</ date> beauce yesterday was <date>Moday</date> and it was <date>July 24</ date>

Chunking

Abney (1991) Parsing by chunks

[I begin] [with an intuition] : [when I read] [a sentence], [I read it] [a chunk] [at a time]

Un **Chunk** comprends un mot plein entouré par des mots creux (ou fonctionnels), et vérifie un motif.

Ce motif exprimable comme :
- une expression régulière (avec action de balisage)
- une condition sur le mot courant et les mots dans un contexte

Par example, ouverture d'un chunk sur certains mots creux :
- <det>=>[<SN>
- <prep>=>[<SP>

Analyse syntaxique « profonde »

Le but de l'analyse syntaxique est de
1. vérifier la bonne formation grammaticale d'une phrase (**reconnaisseur**).
2. extraire les analyses grammaticales possibles sous forme d'arbres de syntaxe (**analyseur**).

Quelques grands principes guident le développement d'un analyseur/reconnaisseur syntaxique pour des traitements linguistiques.

Ces principes se retrouvent pour différents formalismes linguistiques.
(même si certains formalismes privilégient certains types d'analyseurs)

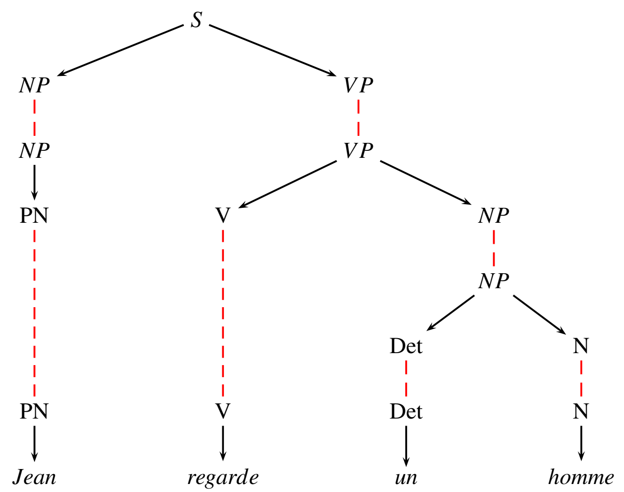
Mise en avant de l'approche **déclarative** dissociant la grammaire (sous forme de règles) et l'analyseur (algorithme).

Stratégie d'analyse

Une stratégie d'analyse décrit les étapes possibles d'une analyse.
- Stratégies **descendantes** (analyse guidée par les requêtes)
- Stratégies **ascendantes** (analyse guidée par les sous-analyses)
- Stratégies hybrides (Stratégie Earley)
- Stratégies avec utilisation de tables (compilées) de décision
  (Left Corner, Head Corner, LR, . . .)

Stratégie d'analyse et collage d'arbres

```
s  --> np vp
np --> pn
np --> det n
np --> np pp
vp --> v np
vp --> vp pp
pp --> prep np
```



Stratégie de contrôle

La stratégie de contrôle specifie l'ordre de traitement des étapes de l'analyse.

Ordonnencement Dans quel ordre effectuer les différentes étapes d'analyse ?
- profondeur d'abord,
- largeur d'abord,
- parallele, concurrent, ...

Ambiguité Comment gérer les alternatives ?
- desambiguisation (probabilités, heuristiques, regard en avant),
- retour-arriere,
- tabulation, ...

Lecture Dans quel sens lire la phrase ?
- gauche vers droite (le plus courant),
- bidirectionnel, ...

Le concept de tabulation

La notion de **Tabulation** largement utilisé en analyse syntaxique, mais aussi dans d'autres domaines (programmation dynamique, mémo-fonctions).

Tabuler consiste à stocker dans une table des traces de sous-calculs pour :
- détecter des boucles pour terminer
- partager des calculs
- avoir une trace des calculs
- choisir l'ordonnancement des calculs
- agréger des calculs

Détection de Boucles

Problèmes de boucles et terminaison en présence de récursions,
dont les **récursion gauches** en analyse syntaxique

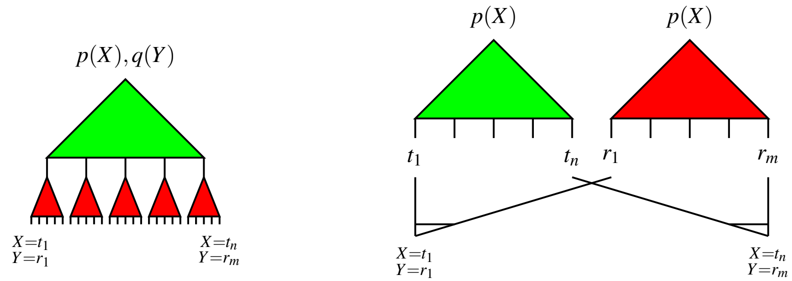
(a) directe	(b) indirecte	(c) cachée
NP $\longleftarrow$ NP Prep NP	$A \longleftarrow B\ c \quad B \longleftarrow A$	$A \longleftarrow B\ A\ c \quad B \longleftarrow \varepsilon \vert b$

Remèdes :

1. Transformer les programmes, mais
   - pas toujours possible facilement
   - change la sémantique
2. Comparer le calcul courant aux traces tabulées de ses ancêtres

Partage de calcul et Non déterminisme

La gestion du non-déterminisme par retour-arrière à la Prolog (backtracking) entraîne de nombreux recalculs.



Sans partage, les GN sont recalculés 5 fois.

Jean regarde un homme sur la colline avec un télescope.

Le nombre de recalculs augmente exponentiellement avec le nombre ambiguïtés.

Trace des calcul – Forêts

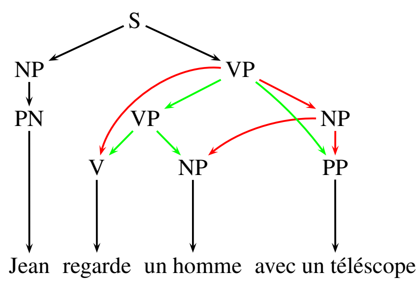
Classiquement, pour obtenir un arbre d'analyse :

$$\mathrm{S} \leftarrow \mathrm{NP\ VP} \text{ transformée en } \mathrm{S}(s(NP,V)) \leftarrow \mathrm{NP}(NP)\ \mathrm{VP}(VP)$$

Mais construction des arbres même pour les échecs

La tabulation permet d'extraire les arbres à partir des traces tabulées pour les analyses réussies.

De plus, permet en fait d'extraire une **forêt** d'analyse compacte partageant les points ambiguïtés



Tabulation et CFG

Une longue histoire avec de nombreux algorithmes :

- Cocke-Kasami-Younger [CKY]
- Algorithme d'Earley – Analyseurs à charte [Kay]
- Algorithme de Tomita (LR)
- Automate à piles / Programmation Dynamique [Lang]

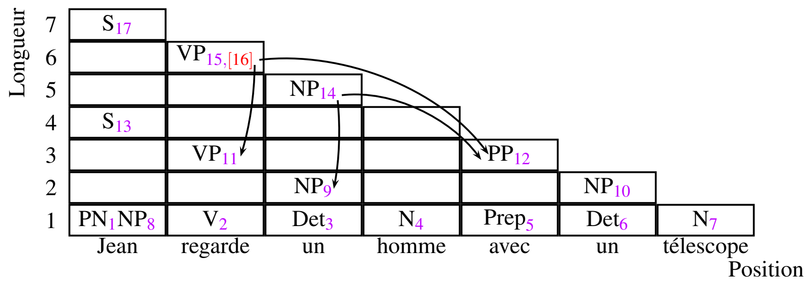
Mais important de comprendre la distinction entre

- Stratégie d'analyse
- Stratégie de contrôle

Algorithme Cocke-Kasami-Younger [CKY]

Algorithme par Programmation Dynamique (1965)
Stratégie d'analyse ascendante avec tabulation des constituants.

« Si il existe une production $A_0 \leftarrow A_1 \dots A_n$ avec, pour tout $i > 0$, A_i présent dans la case (x_i, l_i) et $x_{i+1} = x_i + l_i$, alors tabuler le non terminal A_0 dans la case $(x_1, x_n + l_n)$ (si non déjà tabulé). »



Algorithme et Complexité

initialiser la table avec les terminaux
pour toutes position x et longueur l
pour toute règle $A_0 \leftarrow A_1 \dots A_v$
pour toutes longueurs $l_1, \dots, l_{v-1}$ avec $\Sigma_{k=1..v-1} l_k < l$
$l_v = l - \Sigma_{k=1..v-1} l_k$
$x_j = x + l_1 + \dots + l_{j-1}$
si $A_j \in T[x_j, l_j]$ pour tout $j > 1$
alors ajouter (si non present) A_0 dans $T[x, l]$

Complexité en temps pour le pire des cas provient des itérations de x, l et l_j $(1 \leq j < v)$ sur la longueur n de la chaîne.
$\Rightarrow O(n^{v+1})$ où v est la longueur de la plus grande production.

Pour un reconnaisseur, complexité en place donnée par le mombre de cases de la table et nombre d'éléments par case
$\Rightarrow O(n^2)$

Forme normale de Chomsky (binarisation)

La complexité en $O(n^{v+1})$ réduite à $O(n^3)$ par mise sous **forme normale de Chomsky** (**binarisation**).

La règle ternaire `VP --> V, NP, NP` donne une complexité en $O(n^4)$
mais peut être remplacée par les règles binaires :

```
VP --> V, VP_ARGS.
VP_ARGS --> NP, NP.
```

Mais implique transformation de grammaire
plus élégant de manipuler des **règles pointées**.

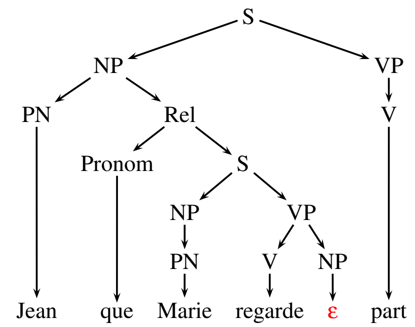
Complexités temps $O(n^3)$ et place $O(n^2)$ meilleures possibles pour CFG
Mais CKY non efficace !

Problèmes de CKY : calculs inutiles

Constituants inutiles

Jean qui regarde $[_S$ Marie part $]$

Hypothèses sur les traces



Analyseurs à Charte Chart Parsing

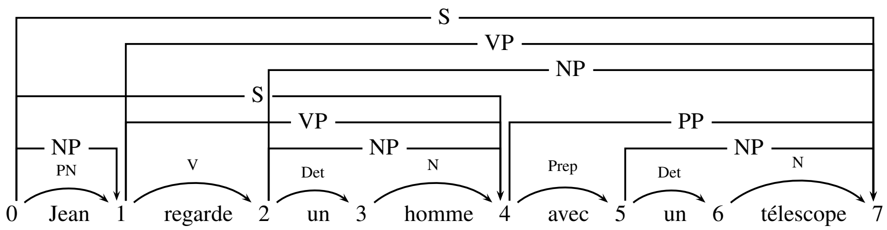
Historiquement, volonté de

- conserver la tabulation (pour réutiliser les sous-calculs)
- conserver une complexité temps en $O(n^3)$ pour les CFG
- introduire de la prédiction (descendante)

$\Rightarrow$ développement de techniques générales : les **chartes**.

CKY comme analyseur à charte passive

Les entrées de la table CKY représentées visuellement par des arcs et stockées par des **items** $\langle i, j, Cat\rangle$.



Complexité temps reste en $O(n^{v+1})$

Charte active et Productions pointées

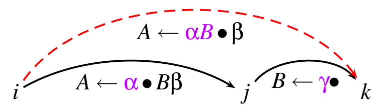
Une charte active n'est plus seulement utilisée pour stocker les constituants reconnus mais aussi les calculs en cours.

Utilisation

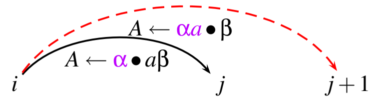
- de **productions pointées** (dotted rules) $\boxed{A_0 \leftarrow A_1 \ldots A_i \bullet A_{i+1} \ldots A_n}$
- d'arcs étiquetés par des productions pointées (items $\equiv \langle i, j, A \leftarrow \alpha \bullet \beta \rangle$)
- d'un système déductif indiquant comment construire les items

Earley algorithm [1970] & Active Chart Parsing [Kay]

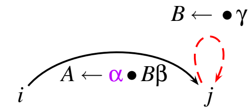
(Complete) $\dfrac{\langle i,j,A \leftarrow \alpha \bullet B\beta\rangle \quad \langle j,k,B \leftarrow \gamma \bullet\rangle}{\langle i,k,A \leftarrow \alpha B \bullet \beta\rangle}$



(Scan) $\dfrac{\langle i,j,A \leftarrow \alpha \bullet a\beta\rangle}{\langle i,j+1,A \leftarrow \alpha a \bullet \beta\rangle}$ $\quad a = a_{j+1}$



(Pred) $\dfrac{\langle i,j,A \leftarrow \alpha \bullet B\beta\rangle}{\langle j,j,B \leftarrow \bullet\gamma\rangle}$ $\quad \exists B \leftarrow \gamma$

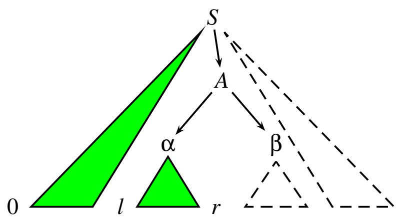


Dotted rules gives time complexity $O(n^3)$
Still space complexity $O(n^2)$ without backpointers and $O(n^3)$ otherwise
Rule (Pred) gives top-down prediction.

Invariant et Complexité

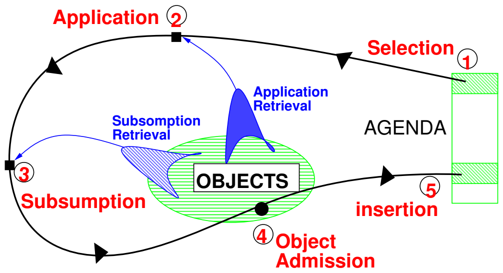
Un item $\langle l, r, A \leftarrow \alpha \bullet \beta\rangle$ vérifie deux invariants :

1. Reconnaissance de α entre l et r (comme CKY)
2. **Validité du préfixe** $0, l : \exists \gamma \in (T \cup N)^\star,\ S \rightarrow^\star a_1 \ldots a_l A \gamma$



Complexité en pire de cas reste $O(n^3)$
Mais en pratique, la prédiction réduit grandement la complexité.

Une architecture uniforme pour les chartes



Variations :

- Structure de la table table : monolithique ou non
- Sélection dans l'agenda ...
- Récupération mémoire

Ce modèle s'applique à l'ensemble des analyseurs syntaxiques tabulaires.

Charte : Mise en oeuvre

Un algorithme à charte comprend :

- une **table** où sont stockés les items, sans doublons.
- un **agenda** où sont stockés les items non traités.

Un pas de l'algorithme consiste à

1. Prendre un item *I* dans l'agenda
2. Construire de nouveaux items en combinant *I* avec des items de la table.
3. Insérer les nouveaux items dans la table et l'agenda, si non doublon.

Dans le cas CFG, l'ordre de sélection dans l'agenda n'a pas d'importance (univers fini) : l'algorithme termine et est complet.

En général, $\langle i, j, A\rangle$ sélectionné avant $\langle k, l, B\rangle$ si $j < l$:
$\Rightarrow$ lecture synchronisée gauche droite.

Schémas d'analyse

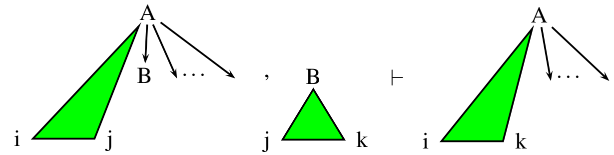
Description de stratégies d'analyse en terme (de classes) d'arbres partiels d'analyse

> [Sikkel] « These intermediate results are not necessarily partial trees, but they must be objects that denote relevant properties of those partial parses. »

Permet d'avoir

- la forme des items
- les invariants

Très proche des algorithmes à charte.



Stratégies de Résolution avec "tables de décisions"

Principe : collecter statiquement des informations sur la grammaire pour mieux guider l'analyse.

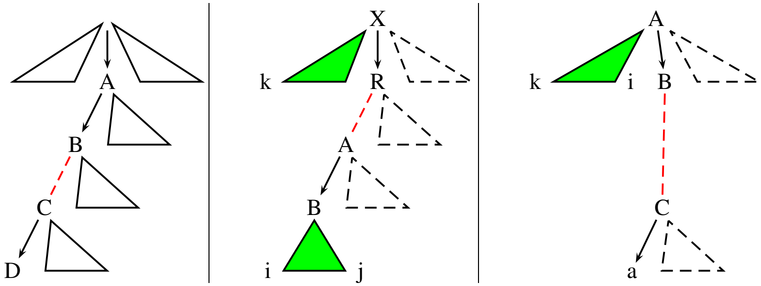
1. Stratégies dirigées par les **Coins Gauches**
2. Stratégies dirigées par les **têtes** (Head-driven strategies)
3. Stratégies **LR** (ou **Décalage/Réduction**, ou **Shift/Reduce**)
   $\Rightarrow$ nouvelle forme d'analyse tabulaire : **piles cactus**

Stratégies par Coin Gauche

(CompleteCG) $\dfrac{\langle i, j, B \leftarrow \beta\bullet\rangle \quad \langle k, i, X \leftarrow \nu \bullet R\mu\rangle}{\langle i, j, A \leftarrow B \bullet \alpha\rangle}$ $\exists A \leftarrow B\alpha$ avec $A \angle R$

(PredCG) $\dfrac{\langle k, i, A \leftarrow \alpha \bullet B\beta\rangle}{\langle i, i+1, C \leftarrow a \bullet \gamma\rangle}$ $\exists C \leftarrow a\gamma$ avec $a_{i+1} = a \angle B$

B, C, D coins gauches de A, noté $D \angle A$



Partage des Préfixes Communs

$\langle l, r, \alpha \rangle$ indique la reconnaissance d'un représentant de $\{\langle l, r, A \leftarrow \alpha \bullet \beta \rangle \mid A \leftarrow \alpha\beta\}$.

(Complete1) $\qquad \dfrac{\langle i, j, \alpha \rangle \quad \langle j, k, \gamma \rangle}{\langle i, k, \alpha B \rangle} \qquad \exists A \leftarrow \alpha B \beta$ et $B \leftarrow \gamma$

(Complete2) $\qquad \dfrac{\langle i, j, \alpha \rangle \quad \langle j, k, \gamma \rangle}{\langle i, k, C \rangle} \qquad \exists A \leftarrow \alpha B \beta$ et $C \leftarrow \gamma$ et $C \angle B$

(PredCG) $\qquad \dfrac{\langle k, i, \alpha \rangle}{\langle i, i+1, a \rangle} \qquad \exists A \leftarrow \alpha B \beta$ et $C \leftarrow a\gamma$ avec $a_{i+1} = a \angle B$

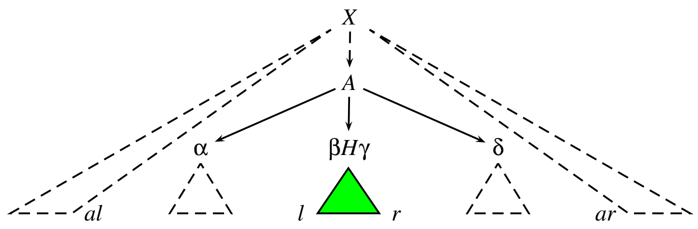
En combinant Coin Gauche et Partage Préfixes $\Rightarrow$ variantes de LR [Nederhof]

Lecture bidirectionnelle : Analyse guidée par les têtes

Analyseurs à charte non limités à des lectures gauche droite.

Lecture bidirectionnelle possible,
par exemple pour des stratégies dirigées par les têtes syntaxiques.

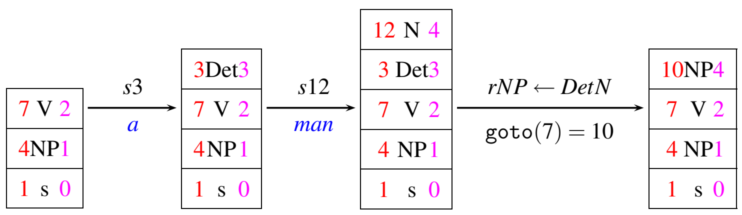
- similaire aux stratégies coin-gauche, sauf que la tête n'est pas le mot suivant.
- stratégie mixte descendante et ascendante
- items à 2 ou 4 points $\langle l, r, A \leftarrow \alpha \bullet \beta H \gamma \bullet \delta \rangle \qquad \langle l, r, A \leftarrow \alpha \bullet \beta H \gamma \bullet \delta, al, ar \rangle$



Strategie LR

The LR table describes a FSA which guides the transitions of a PDA :

- Stack elements are triples (state, symbol, position) ;
- A shift action pushes a new element ;
- A reduce action for a production $P_u : A \leftarrow A_1 \ldots A_n$ pops n elements et, for a top state s pushes state given by "$goto(s, A)$".

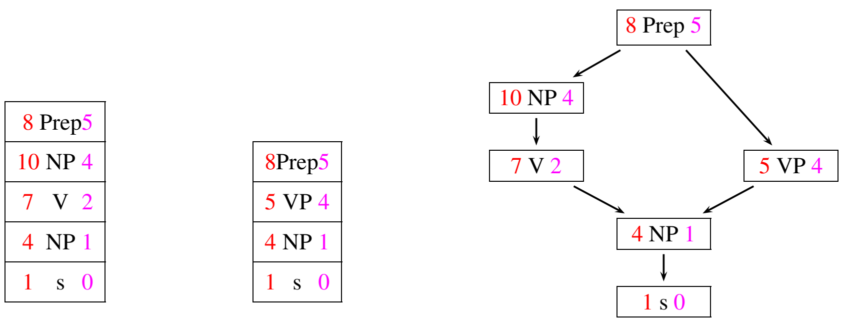


Graph Structured Stack [Tomita'85]

Original LR-like algorithms designed for deterministic execution and for restricted CFG.

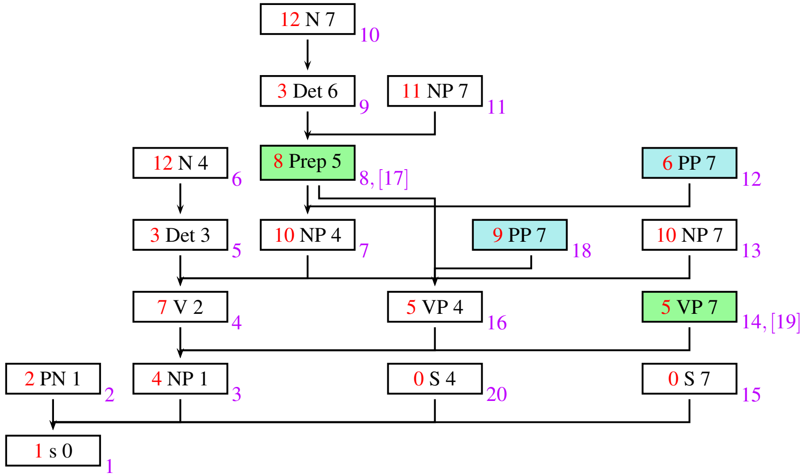
Extension to all CFGs $\Rightarrow$ requires way to efficiently handle conflicts coming from non-determism (in a better way than by backtracking)

Elegant solution with Graph Structured Stacks :



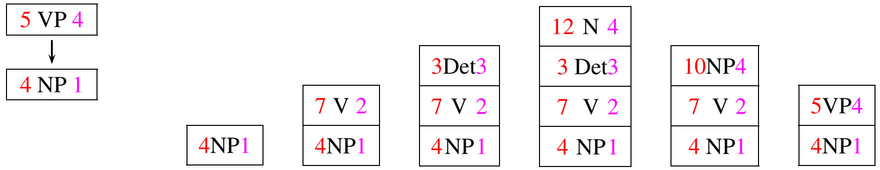
Graph Structured Stack : a full example

Jean watches a man with a telescope



Derivation-based Invariants

- Edges in a graph-structured stack denote Derivation-based Invariants :



What is below 4 NP 1 and the steps followed by the derivation are not relevant ⇒ potential computation sharing

- worst-case complexities are time $O(n^3)$ and space $O(n^2)$.
- with some care, one may handle cyclic grammars
- may be adapted for many parsing strategies
  ⇒ very close from Push-Down Automata and Dynamic Programming [lang]

Traitement des formalismes logiques

Le méchanisme de charte et en particulier l'algorithme d'Earley s'adapte sans problème pour les formalismes linguistiques logiques (exemple DCG).

Dans les règles, utilisation de l'**unification** pour combiner les items.

$$[\text{Fondamentale}]\ \frac{\langle i,j,A\leftarrow\alpha\bullet B\beta\rangle\quad\langle j,k,C\leftarrow\gamma\bullet\rangle}{\langle i,k,(A\leftarrow\alpha B\bullet\beta)\sigma\rangle}\qquad \sigma=mgu(B,C)$$

$$[\text{Down}]\ \frac{\langle i,j,A\leftarrow\alpha\bullet B\beta\rangle\quad C\leftarrow\gamma}{\langle j,j,(C\leftarrow\bullet\gamma)\sigma\rangle}\qquad \sigma=mgu(B,C)$$

Renommage des variables des items avant utilisation dans les règles.

Pour le stockage des items, utilisation de la **subsumption** pour éliminer les items redondants.

$\langle i,j,A\leftarrow\alpha\bullet\beta\rangle$ généralise $\langle i,j,A'\leftarrow\alpha'\bullet\beta'\rangle$ si $\exists\sigma,\ A'\leftarrow\alpha'\bullet\beta'=(A\leftarrow\alpha\bullet\beta)\sigma$

Extension aux formalismes logiques (2)

Terminaison et complétude ne sont plus assurées
(infinité de termes comme $p(f^n(a))$)

- Utilisation d'agenda **équitable** pour assurer la complétude.
  Tout item dans l'agenda sera un jour sélectionné
  Ainsi, "Premier Entré / Premier Sorti" [FIFO] est équitable, LIFO non !

- Utilisation de **restriction des prédictions** [Shieber] pour améliorer la terminaison.

$$[\text{DownR}]\frac{\langle i,j,A\leftarrow\alpha\bullet B\beta\rangle\quad C\leftarrow\gamma}{\langle j,j,(C\leftarrow\bullet\gamma)\sigma\rangle}\qquad \sigma=mgu(\Phi B,C)$$

avec ΦB généralisation de B (exemple : $\Phi p(f^n(t))=p(f(X))$ pour $n>1$).

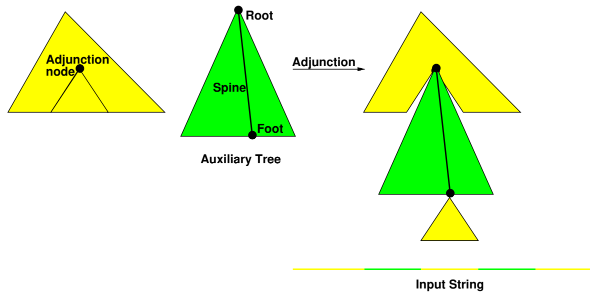
Extension aux formalismes logiques (3)

Egalement des problèmes technologiques.

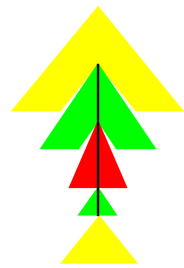
- **Indexation** d'une table d'items complexes pour des accès rapides modulo unification et subsumption
- **Stockage** d'items complexes devant être utilisés à renommage de variable près
  $\Rightarrow$ copie de termes ou partage de structure.
- Définition de bonnes stratégies de sélection pour l'agenda.

Traitement des TAGs

TAGs [Joshi] construisent des arbres dérivés à partir d'arbres initiaux et auxilaires en utilisant la substitution et l'adjonction.



(a) Adjonction



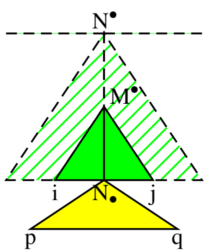
(b) Adjonctions enchassées

CKY pour TAG

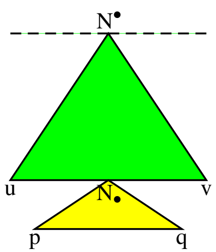
Algorithme CKY pour TAGs [Vijay-Shanker & Joshi 85]

Utilisation :

- Arbres pointés $\nu^\bullet$ et $\nu_\bullet$ où N est un noeud d'un arbre élémentaire.
- Items $\langle \nu^\bullet, i, p, q, j\rangle$ et $\langle \nu_\bullet, i, p, q, j\rangle$ avec p, q couvrant un pied éventuel.



$\langle M_\bullet, i, p, q, j\rangle$



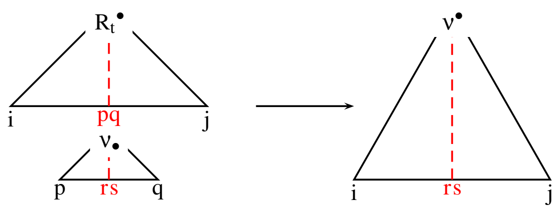
Sans adjonction : $\langle \nu_\bullet, p, -, -, q\rangle$

Avec adjonction : $\langle \nu^\bullet, u, -, -, v\rangle$

Règle (Adjoin)

« Recolle un sous-arbre au pied d'un arbre auxiliaire. »

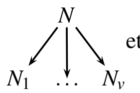
(Adjoin) $$\frac{\langle \nu_\bullet, p, r, s, q\rangle \ \langle R_t^\bullet, i, p, q, j\rangle}{\langle \nu^\bullet, i, r, s, j\rangle} \qquad \text{label}(N) = \text{label}(R_t)$$



Règle (Complète)

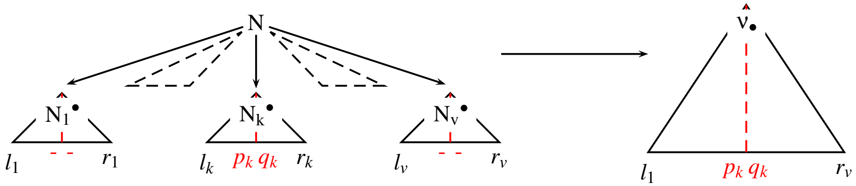
« Recolle l'ensemble des fils d'un noeud. »

(Complete) $\dfrac{\langle \mathrm{N_i}^{\bullet}, l_i, p_i, q_i, r_i \rangle}{\langle \mathrm{v}_{\bullet}, l_1, \cup p_i, \cup q_i, r_v \rangle}$



et $l_{i+1} = r_i$

Note : Au plus un fils (k) couvre un pied et $(\cup p_i, \cup q_i) = (p_k, q_k)$



Complexité

D'autres règles nécessaires pour

1. substitution
2. lecture des terminaux
3. noeuds sans adjonction

Complexité temps $O(n^{\max(6,1+v+2)})$ avec

- v : nombre maximum de fils pour un noeud
- 2 : nombre d'indices pour couvrir l'unique pied des fils

Normalisation avec arbres binaires ($v = 2$) $\Rightarrow$ complexité $O(n^6)$

4 indices par items $\Rightarrow$ Complexité place en $O(n^4)$

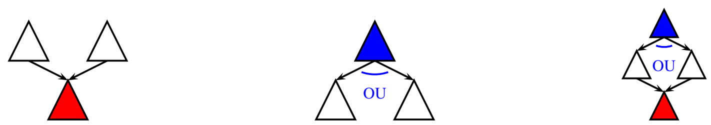
Donne les meilleurs complexités (pires des cas),
mais en pratique, encore moins efficace que CKY pour CFG

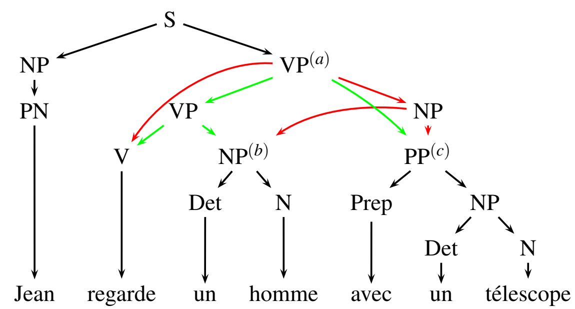
Forets partagées (analyse et dérivation)

Une forêt partagée d'analyse (pour CFG)

- résume un ensemble (même infini) d'arbres d'analyse
- représentable par des graphes ET-OU
- représentable par une grammaire hors contexte (choix DyALog)

Graphe ET-OU





Forêt CFG ≡ grammaire CFG

La forêt est une grammaire G' spécialisant G par intersection avec une chaîne vue comme langage régulier [Lang].

0 Jean 1 regarde 2 un 3 homme 4 avec 5 un 6 télescope 7

```
s  --> np vp
np --> pn
np --> det n
np --> np pp
vp --> v np
vp --> vp pp
pp --> prep np
```

```
s07   --> np01 vp17          pn01 --> Jean
np01  --> pn01                v12 --> regarde
vp17  --> v12 np27          det23 --> un
vp17  --> vp14 pp47           n34 --> homme
np27  --> np24 pp47        prep45 --> avec
n37   --> n34 pp47          det56 --> un
np24  --> det23 n34           n67 --> télescope
pp47  --> prep45 np57
np57  --> det56 n67
vp14  --> v12 np24
```

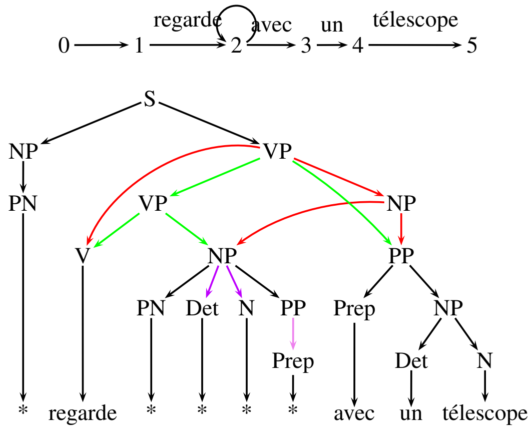
Certains non-terminaux (vp17) ont plusieurs définitions (ambiguité).

Certains non-terminaux (v12,np24,pp47) sont utilisés plusieurs fois (partage).

Chaîne vs Langages Réguliers

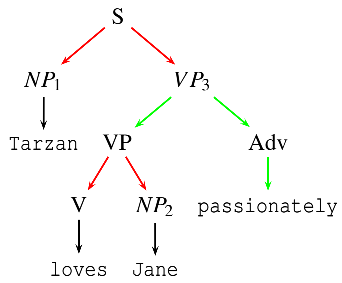
Analyse et affichage des forêts sont possibles pour tout langage régulier

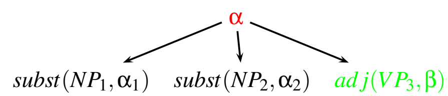
« [mot illisible] regarde [mots illisibles] avec un télescope »



Forêts de dérivation (TAG)

Pour les TAGs, arbres dérivés et arbres de dérivation non isomorphes

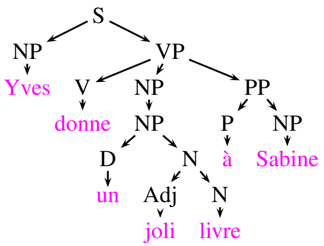


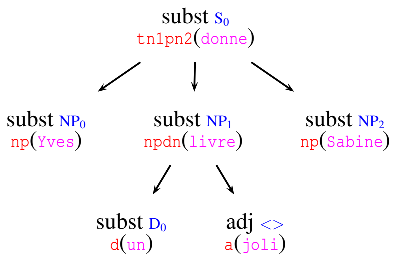


Possibilité d'extraire la forêt d'analyse ou la forêt de dérivation.

Forêt de dérivation sous forme CFG ou LIG.

Forêt de dérivation





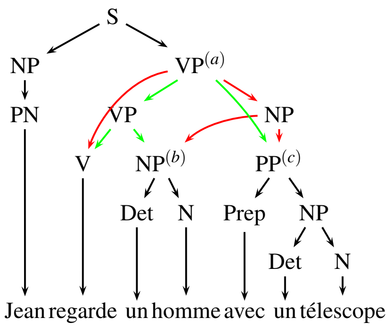
Applications des forêts

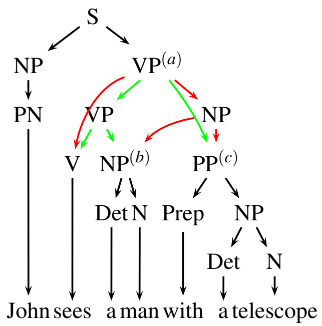
Une forêt partagée représente de manière très compacte un ensemble éventuellement exponentiel ou infini d'arbre
⇒ Structure d'échange intéressante pour les traitements suivants.

- Stockage dans des banques d'arbres (tree banks) pour des corpus
- Analyse Syntaxique **multi-passes guidée** par les forêts
  Pour une grammaire TAG avec traits :
  - **Passe 1** Analyse sans les traits et production d'une forêt F.
  - **Passe 2** Utilisation des productions de F pour guider l'analyse avec traits.
- Calcul de formes sémantiques (partagées)
- Conservation des ambiguïtés lors des traductions

Conserver les ambiguïtés

Conservation possible des ambiguïtés lors de traductions entre langues proches sur certains énnoncés :





Analyse robuste

Lexicalisation

L'évolution actuelle des formalismes linguistiques accorde de plus en plus d'importance au **lexique** associant toute sorte d'informations aux **mots**, **lemmes** ou **concepts**.

Les informations s'échelonnent entre phonologie et sémantiques.

- Phonologie
- Morphologie dérivationnelle
- Syntaxe (nombre, genre, catégorisation, structure de sélection)
- Sémantique lexicale
- Sémantique (réseau sémantiques, ontologies)

Pb : Il existe beaucoup de mots (éventuellement infini)
⇒ identifier classes de mots se comportant similairement
⇒ comprendre les règles de dérivations (nouveaux mots, nouveaux sens)

Information morphologique

Informations permettant de construire les formes dérivées d'un lemme.

- Morphologie flexionnelle
- Morphologie dérivationnelle
  - modifications : connu donne inconnu
  - changement de catégorie syntaxique : V vers N (demander ↦ demande)
  - concepts dérivés : employer ↦ employeur et employé
    (note : demander ↦ demandeur mais pas demandé [N])

  Identification de classes de mots se comportant similairement pour des ensembles de dérivation.

Information syntaxique

Information gouvernant l'emploi syntaxique d'un mot dans la phrase :

- sous-catégorisation des verbes : intransitif, transitif, ditransitif
- transformations possibles pour les verbes : passif

Grammaires Lexicalisées

Chaque production de la grammaire consomme au moins un littéral (ancre, tête).

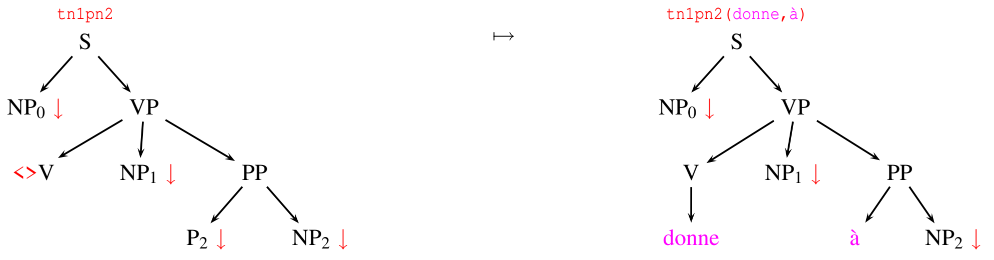
1. Simplification des algorithmes d'analyse : pas de boucles.
2. Possibilité de pré-filtrer les productions nécessaires à partir des mots de la phrase
   ⇒ réduction dynamique de la taille de la grammaire.
3. Utilisation d'algorithme de « Suppertagging » (super-étiqueteur) pour filtrer la production la plus probable pour un mot (extension de tagging qui associe la catégorie lexicale).

TAG Lexicalisées

Arbres ancrées (lexicalisés) par des lemmes

donner quelque chose à quelqu'un

```
donne: \DONNER\,V
       mode=ind,num=sing
\DONNER\,V: tn1pn2[p_2=à]
       NP_0.top:restr=+hum,
       NP_2.top:restr=+hum
```



Sémantique lexicale

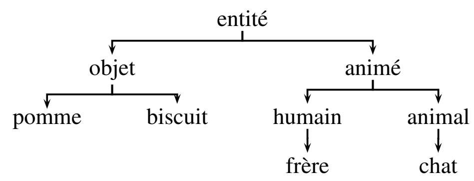
Information « sémantique » pouvant guider l'emploi lexical et syntaxique des mots.

- Rôles thématiques
- Classes sémantiques et Restrictions de sélection

Classes sémantiques et taxinomie

L'utilisation de classes sémantiques permet de mieux cibler les arguments possibles d'un verbe (mais aussi d'un adjectif ou d'un verbe).

Ces classes sont souvent organisées hiérarchiquement dans une **taxinomie**



Les taxinomies offrent aussi la possibilite de mesurer des distances entre classes.

Restrictions de sélection des verbes

Completent et précisent les informations de sous-catégorisation à l'aide de classes sémantiques.

sous-catégorisation donner : donner NP à NP
Accepte "Paul donne un biscuit à son frère"
Ambiguité "Paul donne un biscuit à la pomme à Pierre"

restriction de sélection donner : [humain] donner [objet] à [humain]
accepte "Paul donne un biscuit à son frère"
désambiguise "Paul donne un biscuit à la pomme à Pierre" en
"Paul donne [un biscuit à la pomme] à [Pierre]"

Lexical Conceptual Structures [LCS] [Jackendoff]

Structures établissant des relations prédicatives (dénotant des états, des évenements, …) entre des arguments typés.

Par composition des LCS des mots, on peut construire la forme prédicate d'une phrase.

John runs into the store $[_{\text{event}}\mathrm{GO}_{+\text{loc}}([_{\text{thing}}John],[_{\text{path}}\mathrm{TO}_{+\text{loc}}([_{\text{place}}\mathrm{IN}_{+\text{loc}}([_{\text{thing}}store])])])]$

run $[_{\text{event}}\mathrm{GO}_{+\text{loc}}([_{\text{thing}}],[_{\text{path}}])]$

into $[_{\text{path}}\mathrm{TO}_{+\text{loc}}([_{\text{place}}\mathrm{IN}_{+\text{loc}}([_{\text{thing}}])])]$

Extension : fragment de « acheter »

$$[_{\text{event}}\mathrm{GO}_{+\text{pos}}([_{\text{thing}}money],[_{\text{path}}\mathrm{FROM}_{+\text{pos}}([_{\text{human}}I]),\mathrm{TO}_{+\text{pos}}([_{\text{human}}K])])]$$

Rôles thématiques

Agent : L'acteur d'une action. Sujets de tuer, manger, frapper

Patient : Celui qui subit l'action. Objet de tuer, manger, frapper mais pas de entendre, voir, aimer.

Experiencer : Celui qui perçoit quelque chose. Exemples : sujet d'aimer, objet de ennuyer.

Theme : Celui qui modifie sa condition ou sa position. Exemples : objets de donner ; Sujets de marcher, mourir.

Goal : Object cible d'un mouvement. Exemples : sujet de recevoir, acheter, objets indirects de dire, donner.

donner (agent(goal(theme)))

craindre (experiencer(theme))

effrayer (theme(experiencer))

Les noms et les adjectifs peuvent aussi avoir des rôles (venant en particulier des verbes dont ils dérivent).

De même les prépositions peuvent avoir des rôles.

à (goal)

Alternation

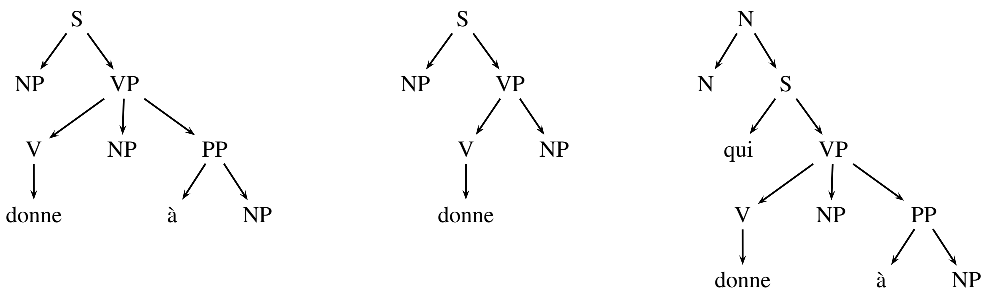
Les réalisations syntaxiques possibles d'un verbe (alternations) peuvent souvent être reliées à sa sémantique et aux rôles thématiques attendus
⇒ classification des verbes.

La classe {abandonner, accorder, acheter, adjuger, allouer, assigner, attribuer, céder, choisir, concéder} sous classe de verbe de Transfert de possession admet les alternations :

- NP Verb NP PP[prep=a]
- NP Verb PP[prep=a] NP
- NP Verb NP
- NP Verb[+passive] NP PP[prep=par]
- NP[+agent] Verb[+refl] NP : il s'accorde une cigarette
- NP[+theme] Verb[+refl] NP : l'essence s'achète au prix fort

Alternations : mise en oeuvre

Notion de familles en XTAG regroupant les arbres associés à diverses alternations.



Ressources linguistiques

- dictionnaires electroniques : liste de mots avec des informations morphologiques et lexicales, éventuellement avec une définition.
- thesaurus : liste de mots hiérarchisée
- lexiques informatiques : mots reliées par des relations lexicales (WordNet)
- Bases de Connaissance Lexicales [LKB] mots complétés par des informations de sémantique lexicale (ACQUILEX).
- ontologies (SENSUS, ONTOSORUS, PENMAN) et réseaux sémantiques (MEMODATA, MINDNET) : concepts et relations entre concepts

Distinctions floues entre les divers types de ressources.

WordNet [Miller]

Information sur les liens entre les mots :

synonymie sens proches

antinomie sens oposés

hyperonymie relation hiérachique « est une sorte de » (kind-of).

méronymie relation hiérachique « est une partie de » (part-of)

1. car#1 auto#1 automobile#1 machine#4 motorcar#1 :
   4-wheeled motor vehicle ; usually propelled by an internal combustion engine ; "he needs a car to get to work".
2. car#2 railcar#1 railway car#1 railroad car#1 :
   a wheeled vehicle adapted to the rails of railroad ; "three cars had jumped the rails".
3. car#3 gondola#3 :
   car suspended from an airship and carrying personnel and cargo and power plant.
4. car#4 elevator car#1 :
   where passengers ride up and down ; "the car was on the top floor".
5. cable car#1 car#5 :
   a conveyance for passengers or freight on a cable railway ; "they took a cable car to the top of the mountain".

WordNet : « car » est une sorte de . . .

1. car#1 auto#1 automobile#1 machine#4 motorcar#1

$\Longrightarrow$ motor vehicle#1 automotive vehicle#1

$\Longrightarrow$ vehicle#1

$\Longrightarrow$ conveyance#3 transport#1

$\Longrightarrow$ instrumentality#3 instrumentation#1

$\Longrightarrow$ artifact#1 artefact#1

$\Longrightarrow$ object#1 physical object#1

$\Longrightarrow$ entity#1 something#1

Représentations

Les lexiques représentent des ressources couteuses à obtenir et de taille importantes ⇒ effort sur les représentations :

Format d'échange en XML suivant une DTD expliquant bien le contenu du lexique.

Format d'accès dans une base de données pour des accès rapides aux ressources impliquées lors d'un traitement.

Format de traitement au sein d'une application (par exemple, analyseur syntaxique). Représentation par des structures de traits typées (à la Carpenter).

Organisation

Un lexique représente beaucoup d'information à indiquer et à maintenir
⇒ organisation intelligente

- défault : la donnée d'une propriéte induit un certain nombre de défaults
- héritage : les entrées des lexiques sont typés et héritent des propriétés des super-types
- macros (« templates ») : noms donnés à un paquet de propriétés pour abréger l'écriture

$$\texttt{2sg} \equiv \texttt{pers} = sg \wedge \texttt{nbr} = 2$$

- lexiques multi-couches : plusieurs niveaux de lexiques, une entrée au niveau $n+1$ pouvant faire référence à des entrées au niveau n

Par exemple XTAG est multi-couches (formes fléchies ; lemmes ; familles ; arbres) et avec « templates ».

Echange : Architecture XTAG en XML

```
<lemma cat="v" name="*DONNER*">
 <anchor tree_id="family[@name=tn1pn2]">
     <coanchor node_id="p_2"> <lex>à</lex> </coanchor>
     <equation node_id="np_0" type="top">
         <fs> <f name="restr"><val>plushum</val></f> </fs>
     </equation>
     <equation node_id="np_2" type="top">
         <fs> <f name="restr"><val>plushum</val></f> </fs>
     </equation>
 </anchor>
</lemma>
```