Traitement Automatique des Langues

Jean-Yves Antoine

Université François Rabelais de Tours

www.info.univ-tours.fr/~antoine

UNIVERSITÉ FRANÇOIS - RABELAIS TOURS

Traitement Automatique des Langues

GRAMMAIRES FORMELLES ET SYSTEMES DE REECRITURE

Grammaires formelles

Définition [Chomsky 1956]

Grammaire G : quadruplet $<V_t,V_n,S,P>$ où :

- V_t ensemble des **symboles terminaux** encore appelé **vocabulaire** (on parle alors des mots), représente un ensemble non vide de symboles.
- V_n ensemble des symboles non terminaux (ou **catégories syntaxiques**) représente un ensemble non vide de symboles tel que l'intersection entre V_t et V_n est vide.
- S **symbole initial** (ou **axiome**) est un symbole de Vn et sert de point de départ.
- P ensemble des **règles de production** de la forme $\alpha \rightarrow \beta$ avec α (tête) et β (corps) toute séquence de symboles construite sur $V = V_t \cup V_n$ (β pouvant être éventuellement vide). Ces productions sont encore appelées **règles de réécritures** car elles précisent que la séquence de symbole α peut être remplacée par la séquence de symboles β.

⇨ **s'applique aussi bien au traitement du langage naturel que des langages artificiels (compilateurs langage informatique)**

Grammaires formelles

Notations

- **Terminaux** chaîne de caractères minuscules

  exemples : *Paul, le, chien*

- **Non terminaux** chaîne de caractères entre chevron

  exemples : *<S>, <GN>, <GV>*

Langage généré par une grammaire

- **Définition :** L(G) langage généré par la grammaire G = ensemble des séquences de symboles obtenues en partant du symbole initial S et en appliquant les productions de G par réécritures successives jusqu'à ce que la chaîne résultante ne présente plus que des symboles terminaux.

- **Notation** L(G) ou L<S>

Grammaires formelles

Exemple

V_t = {*aime,beaucoup,passionnément,un_peu,il,elle,m,t*}

V_n = { <S>,<SN>,<SV>,<GV>,<GADV>,<pronom>,<verbe>,<clitique>,<adv> }

P = <S> → <SN><SV>
<SN> → <pronom>
<SV> → <GV>
<SV> → <GV><GADV>
<GV> → <verbe>
<GV> → <clitique><verbe>
<GADV> → <adv>
<verbe> → *aime*
<pronom> → *il* | *elle*
<clitique> → *m* | *t*
<adv> → *un_peu* | *beaucoup* | *passionnément*

Hiérarchie de Chomsky

Grammaire d'ordre 0

– aucune restriction sur les règles de production

Ordre 1 : grammaire contextuelle (context-sensitive grammar)

γ <X> $\alpha \rightarrow \gamma\ \beta\ \alpha$	avec α, β et γ séquences (éventuellement nulle) de symboles (terminaux ou non) et <X> non terminal
contextuelle	X se réécrit en β dans le contexte ($\gamma\ \alpha$)

Ordre 2 : grammaire hors contexte (CFG : context free grammar)

<X> $\rightarrow \beta$	avec <X> symbole non terminal et β séquence de symboles quelconques (terminaux ou non).
exemples	<verbe> $\rightarrow$ *aimer* <GN> $\rightarrow$ <det><nom>

Ordre 3 : grammaire régulière (regular grammar)

<X> $\rightarrow \beta$ <Y>	avec <X> et <Y> symboles non terminaux unique (ou nul pour <Y>) et β symbole terminal unique.
exemples	<verbe> $\rightarrow$ *aimer* <GV> $\rightarrow$ se <Vreflex>

Hiérarchie de Chomsky

Pouvoir de génération [Chomsky 1956, Pullum 1984, Joshi *et al*. 1991]

- **langage généré** — **régulière < hors-contexte < contextuelle < 0**
- **langage naturel** — langage **«modérément» sensible au contexte** :

hors-contexte < langage naturel < contextuel

Combinatoire d'analyse

- **combinatoire** — régulière < hors-contexte < contextuelle < ordre 0
- **régulière** — traduction en automatique d'états finis — **o(n)**
- **hors-contexte** — traduction en automate à pile : — **$o(n^{\alpha})$**
- **contextuelle** — pas d'algorithme acceptable — **$o(e^{n})$**
- **ordre 0** — non décidable

TALN : Grammaires hors contexte et régulières

Dérivation

Appartenance à un langage

Une séquence W donnée appartient à un langage L(G) généré par la grammaire G = <Vt, Vn, S, P> ssi il existe une suite de production de P permettant de dériver W à partir du symbole initial S.

Dérivation

Application successive des règles de production mises en jeu depuis <S> jusqu'à la séquence W.

Remarque : il peut y avoir plusieurs dérivations pour une séquence W

Arbre de dérivation

Application successive des règles de production mises en jeu depuis S jusqu'à la séquence W

Exemple : **arbre de Dérivation** de l'énoncé W = *elle m'aime un peu* par la grammaire précédente :

<S> ⇨ (1) ⇨ (2) ⇨ (9) ⇨ (4) ⇨ (6) ⇨ (10) ⇨ (8) ⇨ (7) ⇨ (11) ⇨ W

Déterminisme

Stratégie d'analyse

- Pour une grammaire donnée, plusieurs dérivations sont en général possibles pour dériver une séquence donnée

  exemple : *elle m 'aime un peu*

  <S> ⇨ (1) ⇨ (2) ⇨ (9) ⇨ (4) ⇨ (6) ⇨ (10) ⇨ (8) ⇨ (7) ⇨ (11) ⇨ W

  ↘ (4) ⇨ ...

  ↘ (7)⇨ (8) ⇨ (11) ⇨ W

  ↘ (11) ⇨ (8) ⇨ W

- Le choix de l'ordre d'application des règles dépend de la stratégie d 'analyse mise en œuvre par l'analyseur (cf. infra)

Grammaire déterministe

Une grammaire G est dite déterministe si, à tout moment de l'analyse de toute séquence W de L(G), une seule règle de production de G peut être appliquée.

Déterminisme

Grammaires régulières et déterminisme

Toute grammaire régulière peut se ramener à une grammaire déterministe

Dérivation gauche (ou droite)

– On appelle dérivation gauche (resp. droite) d'une séquence W la dérivation privilégiant systématiquement la dérivation du symbole non-terminal le plus à gauche (resp. droite) dans la séquence

– Heuristique s'affranchissant sans frais d'une partie du non-déterminisme de la grammaire, mais qui n'assure pas d'un déterminisme total.

Exemple : *elle m 'aime un peu*

<S> ⇨g (1) ⇨g (2) ⇨g (9) ⇨g (4) ⇨g (6) ⇨g (10) ⇨g (8) ⇨g (7) ⇨g (11) ⇨g W

Grammaire hors-contexte et déterminisme

– Stratégies de recherche avec regard en avant [Marcus 1980, Charniak 1983]

– non étudiées dans ce cours

Analyse

Analyse

Construction de la structure de la séquence (ou rejet si elle n'appartient pas au langage) telle qu'établie indirectement au cours de la dérivation.

Arbre d'analyse (arbre syntaxique)

Représente la structure de W en illustrant comment l'axiome de la grammaire se dérive successivement en non-terminaux et terminaux pour former W.

Rq: un arbre d 'analyse peut correspondre à plusieurs arbres de dérivation

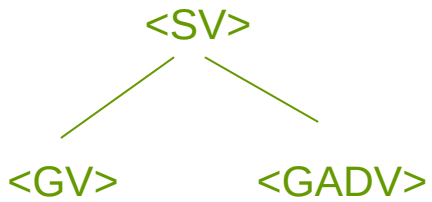
Principe de construction

A chaque dérivation, on adjoint l'arbre élémentaire correspondant à la règle de production utilisée. Noeud initial : <S>

Exemples

<pronom> →*elle*

<SV> → <GV><GADV>



Pouvoir de génération

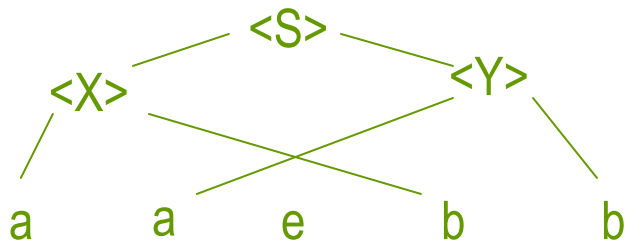
Pouvoir de génération faible

faible capacité du formalisme à engendrer une classe de langage donnée

fort capacité du formalisme d'affecter en outre certaines structures aux énoncés du langage concerné .

Exemple : contrainte de planarité

les CFG engendrent des langages du type an e bn mais ne peuvent leur attribuer un structure non planaire (liens croisés) telle que :



or, certaines structures linguistiques ne sont pas planaires (exemple : syntagmes non continus en russe)...

Équivalence forte / faible entre grammaires

[Joshi *et al*. 1991]

Ambiguïté

Grammaire ambiguë

Une grammaire G est dite ambiguë s'il existe une chaîne du langage L(G) pour laquelle il existe au moins deux arbres d ’analyse différents

Ambiguïté et langage naturel

Langage naturel ambigu, à la différence des langages informatiques

Exemple

- Vt = { *la*,*porte*,*belle*}
- Vn = { <GN>, <ART>, <ADJ>, <NOM>, <VBE>, <PH>}
- P =
  - <S> → <SN> <SV>
  - <S> → <SN>
  - <SV> → <VBE>
  - <SN> → <ART> <NOM>
  - <SN> → <ART> <ADJ> <NOM>
  - <ART> → la
  - <ADJ> → belle
  - <NOM> → belle | porte
  - <VBE> → porte

Arbre d’analyse de : *la belle porte* …

Algorithmes d'analyse

Algorithmes spécifiques à chaque type de grammaire

- Algorithmes optimisés à chaque niveau de la hiérarchie de Chomsky

Grammaires régulières

- non étudiées dans ce cours
- toute grammaire régulière peut être ramenée à un automate à état finis
- temps d 'analyse linéaire
- utilisation en TAL ou en compilation : analyse lexicale, morphologie

Grammaires hors-contexte

- base de l'analyse syntaxique (+ éléments contextuels)
- analyse ascendante ou descendante : temps d 'analyse cubique au pire cas

  ⇨ **cours** : étude détaillée de la stratégie d'analyse descendante

Grammaires contextuelles

- pas d 'algorithme à la combinatoire satisfaisante

Analyse descendante

Principes de base

création de l'arbre en partant de la racine (axiome <S>) et en dérivant par décompositions successives les nœuds inférieurs jusqu 'aux terminaux correspondant à la séquence de mots W.

- **dérivation gauche** langage naturel...
- **non-déterminisme** priorité à l'application de la première des règles de production applicables
- **stratégie de recherche** profondeur d'abord ou largeur d'abord
- **profondeur d'abord** **régime par tentative** : on développe en priorité la 1ère règle applicable. Mémorisation à chaque pas des productions alternatives pour **retour-arrière** en cas d'échec
- **largeur d'abord** développement parallèle de toutes les analyses.

Mise en œuvre informatique

- mémorisation par une pile des états déjà parcourus
- type de pile influant sur la stratégie (profondeur ou largeur d'abord)

Grammaires formelles

Exemple (rappel)

V_t = {*aime,beaucoup,passionnément,un_peu,il,elle,m,t*}

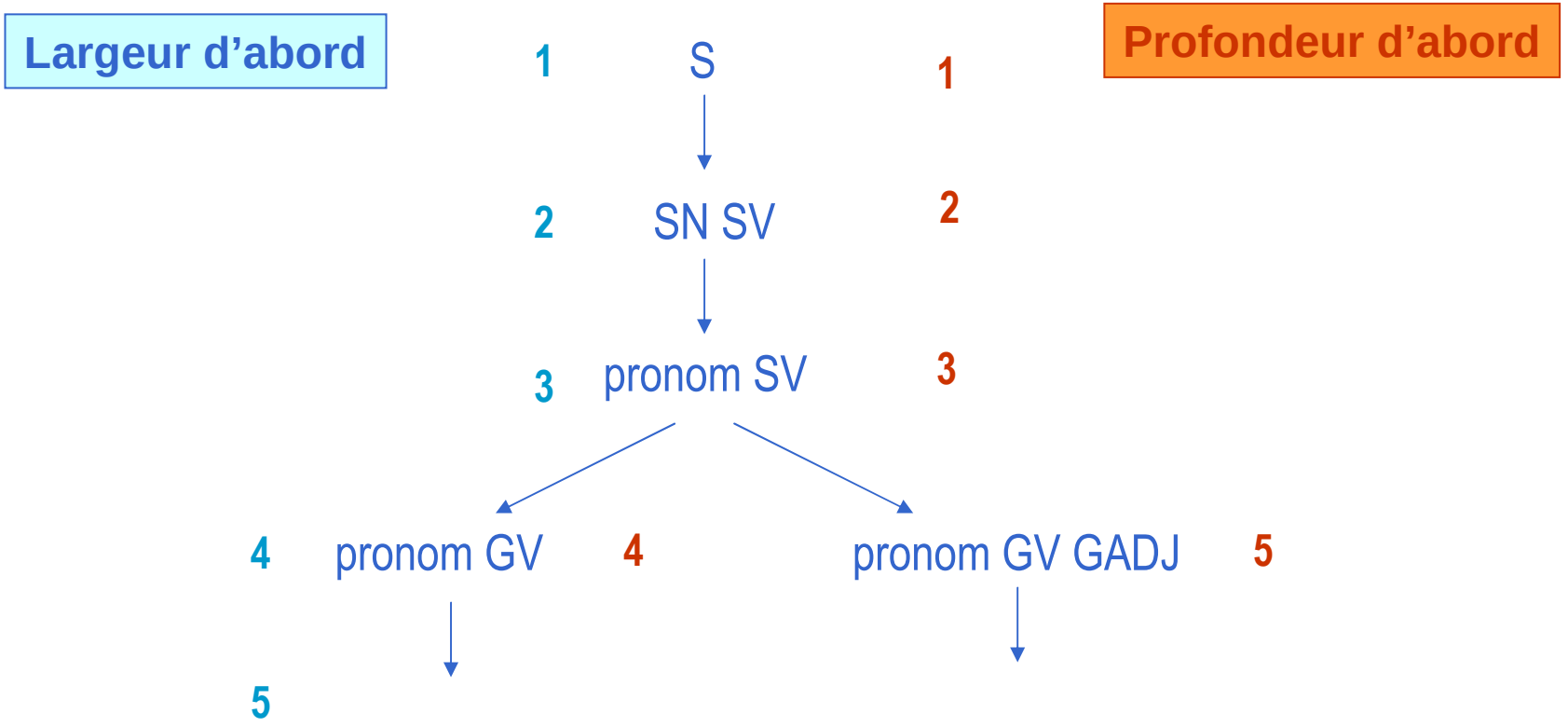
V_n = { <S>,<SN>,<SV>,<GV>,<GADV>,<pronom>,<verbe>,<clitique>,<adv> }

P =
<S> → <SN><SV>
<SN> → <pronom>
<SV> → <GV>
<SV> → <GV><GADV>
<GV> → <verbe>
<GV> → <clitique><verbe>
<GADV> → <adv>
<verbe> → *aime*
<pronom> → *il* | *elle*
<clitique> → *m* | *t*
<adv> → *un_peu* | *beaucoup* | *passionnément*

Analyse descendante

Stratégie de recherche

Exemple : *il m'aime un peu*



Analyse descendante

Algorithme de base [Aho et Ulmann 1972, Allen 1995]

données	piles de listes correspondant à 2 types d'états
état courant	liste des symboles non terminaux à trouver + position dans la séquence du mot courant
état de retour	pile des états alternatifs donnant, pour chaque position, les états que l'on peut obtenir par application les autres règles non utilisées
Etat initial	état courant : ((S) 1)) état de retour : ()
Récursion	on essaie à chaque pas de décomposer le 1er symbole de l'état courant par application d'une règle de même tête. Si c'est possible, on remplace le symbole par le corps de la règle et on empile les états alternatifs dans les états de retour. Sinon, retour-arrière avec dépilage d'un état alternatif.
Arrêt	succès si état courant vide en fin de phrase, échec sinon.

Analyse descendante

Exemple

V_t = {*aime,beaucoup,passionnément,un_peu,il,elle,m,t*}

V_n = { <S>,<SN>,<SV>,<GV>,<GADV>,<pronom>,<verbe>,<clitique>,<adv> }

P = <S> → <SN><SV>
<SN> → <pronom>
<SV> → <GV>
<SV> → <GV><GADV>
<GV> → <verbe>
<GV> → <clitique><verbe>
<GADV> → <adv>
<verbe> → *aime*
<pronom> → *il* | *elle*
<clitique> → *m* | *t*
<adv> → *un_peu* | *beaucoup* | *passionnément*

Analyse descendante

Stratégie de recherche

- **profondeur d'abord** pile LIFO pour l 'état courant (pile)
- **largeur d'abord** pile FIFO pour l 'état courant (queue)

Exemple (profondeur d'abord) *il m'aime un peu*

pas	état courant	état de retour	
1	((S) 1)	()	
2	((SN SV) 1)	()	
3	((Pronom SV) 1)	()	
4	((SV 2))	()	on a traité *il*
5	((GV) 2)	((GV GADV) 2)	première alternative
6	((V) 2)	((clitique V) 2),((GV GADV) 2)	
7	*à vous de jouer...*		

Analyse descendante

Exemple (largeur d'abord) *il m'aime un peu*

pas	état courant	état alternatif	
1	((S) 1)	()	
2	((SN SV) 1)	()	
3	((Pronom SV) 1)	()	
4	((SV 2))	()	on a traité *il*
5	((GV) 2)	((GV GADV) 2)	première alternative
6	((GV GADV) 2)	((V) 2)((clitique V) 2)	pile FIFO
7	*à vous de jouer...*		

Analyse descendante

Construction de l'arbre d'analyse

- chaque nouvelle transition d 'état courant correspond à l'ajout du sous-arbre élémentaire correspondant à la règle appliquée
- **retour-arrière** : de même qu'il y a un état de retour, il faut prévoir un arbre d'analyse de retour (ou pile des états à chaque pas + pointeur de retour)

Exemple *elle m 'aime un peu*

pas	état courant	état de retour	arbre courant	arbre de retour
1	((S) 1)	()	$[?]_{S}$	
2	((SN SV) 1)	()	$[[?]_{SN}\ [?]_{SV}]_{S}$	
3	((Pronom SV) 1)	()	$[[[?]_{pronom}]_{SN}\ [?]_{SV}]_{S}$	
4	((SV 2))	()	$[[[il]_{pronom}]_{SN}\ [?]_{SV}]_{S}$	
5	((GV) 2)	((GV GADJ) 2)	$[[[il]_{pronom}]_{SN}\ [[?]_{GV}]_{SV}]_{S}$	*à vous de jouer*

Analyse ascendante

Principes de base

création de l'arbre en partant de la séquence W que l'on essaie de réduire progressivement jusqu'à l'axiome <S> par applications successives des productions « à l'envers » : corps de règle qui satisfait une partie de la séquence

- **dérivation gauche** traitement prioritaire de la partie la plus à gauche de la pseudo-phrase pouvant être réduite
- **non-déterminisme** priorité à l'application de la première des règles de productions applicables,
- **régime par tentative** mémorisation à chaque pas des règles de production alternatives pour dépilage éventuel. **Retour-arrière** en cas d'échec partiel.

Mise en œuvre informatique

- Mémorisation par une pile des états déjà parcourus

Analyse asecendante

Exemple

V_t = {*aime,beaucoup,passionnément,un_peu,il,elle,m,t*}

V_n = { <S>,<SN>,<SV>,<GV>,<GADV>,<pronom>,<verbe>,<clitique>,<adv> }

P = <S> → <SN><SV>
<SN> → <pronom>
<SV> → <GV>
<SV> → <GV><GADV>
<GV> → <verbe>
<GV> → <clitique><verbe>
<GADV> → <adv>
<verbe> → *aime*
<pronom> → *il* | *elle*
<clitique> → *m* | *t*
<adv> → *un_peu* | *beaucoup* | *passionnément*

Analyse ascendante

- **Algorithme de base**
  - **données** — **piles de listes correspondant à 2 types d'états**
    - ***état courant*** — liste des symboles non terminaux ou terminaux à réduire (pseudo-phrase courante)
    - ***état de retour*** — pile des états alternatifs correspondant aux états que l'on peut obtenir par application des autres règles de production non utilisées prioritairement mais qui matchent la séquence.
  - **état initial** — état courant : séquence W à traiter état de retour : ()
  - **récursion** — on essaie à chaque pas de réduire une chaîne de symboles la plus à gauche possible de l'état courant (mais les autres réductions sont des alternatives possibles). Si c'est possible, on remplace la chaîne réduite par le corps de la règle utilisée et on empile les états de retour correspondant aux réductions alternatives. Sinon, retour-arrière avec dépilage d'un état alternatif.
  - **Arrêt** — succès si remontée à <S> (état courant : (S)), échec sinon

Analyse ascendante

- **Exemple** *elle m'aime un peu*

pas	état courant	état de retour
1	(il m aime un_peu)	()
2	(PRONOM m aime un peu)	((il CLITIQUE aime un peu), (il m VERBE un peu), (il m aime ADV))
3	(SN m aime un peu)	((PRONOM CLITIQUE aime un peu) (PRONOM m VERBE un peu), (PRONOM m aime ADV), (il CLITIQUE aime un peu), (il m VERBE un peu), (il m aime ADV))

- **Explosion combinatoire**

  on recherche de multiple fois à faire les mêmes réductions

Analyse ascendante tabulaire

- **Intérêt**

  analyse guidée par la séquence à traiter

- **Limitation**

  grammaire réelle nombreuses alternatives donc retours en arrière multiples en dépit du fait que l'analyse soit guidée par les données à traiter

  ⇨ algorithme de base trop coûteux en temps de calcul en pratique

- **Solution : analyse tabulaire (*chart parser)***

  idée mémoriser les sous-structures (syntagmes ou autres) analysés à chaque étape pour réutiliser cette information le cas échéant : structure de ***chart*** d'analyse permettant de factoriser les calculs

  chart formalisé par M. Kay dès 1973. [Kay,1980]
  équivalent au ***left-corner parser*** de [Aho & Ullman 1972]

  complexité cubique

Analyse ascendante tabulaire

- **Structure de chart**
  - liste des mots déjà traités avec leur position
  - **arcs actifs** — liste des règles utilisées **partiellement** à un moment donné avec la position de début du corps de la règle et le point où s'est arrêtée l'analyse

    exemple — (SN → DET ° ADJ NC, 1, 2)
  - **agenda** — liste des constituants analysés avec leur position (début - fin). L 'agenda mémorise donc le résultat des règles déjà utilisées en intégralité.

Analyse ascendante tabulaire

- **Algorithme : opérations de base sur le chart**
  - **ajout d'un arc actif** — dès qu'on a un nouveau élément de catégorie C, ajout à la liste des arcs actifs toutes les règles dont le corps commence par C.

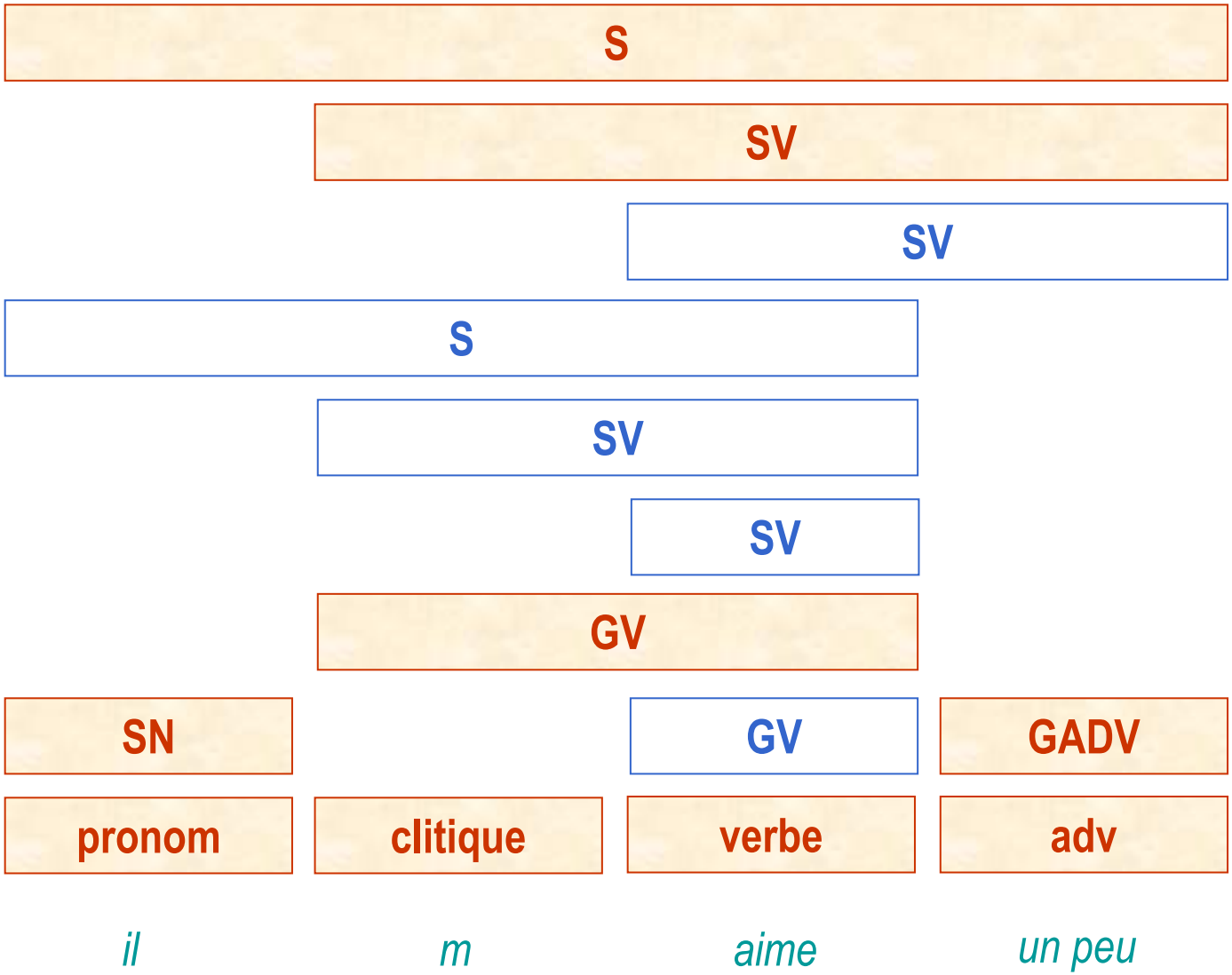
    Exemple *le* : DET NP → **DET °** ADJ NC
  - **extension d'un arc** — si arc actif de la forme X → X1 ... Xn **°** C Y1 ... Yn et que le mot courant est de type C, on ajoute à la liste des arcs actifs : X → X1 ... Xn C **°** Y1 ... Yn
  - **mise à jour agenda** — si une règle est appliquée intégralement à la fin d 'une extension, ajout du constituant (tête de la règle) à l'agenda.
- **Algorithme : analyse**
  - avance gauche-droite dans l'énoncé en actualisant le *chart* (m.a.j. arcs et agenda)**.**
  - En cas de mise à jour de l'agenda, on regarde si le nouveau constituant ne peut pas déclencher de nouvelles règles avant d'avancer dans l'énoncé : **coin gauche**
  - **arrêt** — succès si on arrive à une analyse **complète** avec le constituant <S>

Analyse ascendante tabulaire

- **Exemple** *il m'aime un peu*

pas	position	arcs actifs	agenda (constituants rajoutés)
1	1		([*il*]$_{pronom}$ 1, 1)
2	1		([[*il*]$_{pronom}$]$_{SN}$ 1, 1)
3	1	S → SN ° SV,1,2	
4	2		([*m*]$_{clitik}$ 2, 2)
5	2	GV → clitik ° V,2,3	
6	3		([*aime*]$_{V}$ 3, 3)
7	3		**([[*aime*]$_{V}$]$_{GV}$ 3, 3)**
8	3		([[*m*]$_{clitik}$ [*aime*]$_{V}$]$_{GV}$ 2, 3)
9	3		**([[[*aime*]$_{V}$]$_{GV}$]$_{SV}$ 3, 3)** ; ([[[*m*]$_{clitik}$ [*aime*]$_{V}$]$_{GV}$]$_{SV}$ 2, 3)
10	3	**SV → GV ° GADV,3,4**	
12	3	SV → GV ° GADV,2,4	
13	3		**(([[[*il*]$_{pronom}$]$_{SN}$ [[[*m*]$_{clitik}$ [*aime*]$_{V}$]$_{GV}$]$_{SV}$]$_{S}$ 1, 3)**
14	4		([*un peu*]$_{adv}$ 4, 4)
15	4		([[*un peu*]$_{adv}$]$_{GADV}$ 4, 4)
16	4		**([[[*aime*]$_{V}$]$_{GV}$ [[*un peu*]$_{adv}$]$_{GADV}$]$_{SV}$ 3, 4) ;** [[[*m*]$_{clitik}$ [*aime*]$_{V}$]$_{GV}$ [[*un peu*]$_{adv}$]$_{GADV}$]$_{SV}$ 2, 4)
17	4		([[*il*]$_{pronom}$]$_{SN}$ [[[*m*]$_{clitik}$[*aime*]$_{V}$]$_{GV}$ [[*un peu*]$_{adv}$]$_{GADV}$]$_{SV}$]$_{S}$ 1, 4)

Analyse ascendante tabulaire



Analyse descendante tabulaire

- **Analyse ascendante et descendante**
  - analyse descendante intéressante car on traite les terminaux en contexte : certaines ambiguïtés artificielles ne sont pas étudiées
  - analyse ascendante intéressante car guidée par la séquence à traiter
  - chaque analyse manque par contre d'efficacité (répétition des mêmes sous-analyses) dans des situations différentes

- **Analyse descendante par chart : principes** (non étudiée dans ce cours)
  - combinaison des avantages d'une analyse ascendante et descendante en utilisant ici aussi une structure de *chart* pour factoriser les calculs.
  - Algorithme de **Earley** (1970) *top-down chart parser*
  - algorithme le plus efficace pour les CFG : **complexité cubique**

Conclusion

- **Algorithme de Earley et autres *chart parsers***
  - Techniques efficaces de base pour l'analyse des grammaires hors-contexte
  - **Améliorations** : techniques de réduction des effets du non déterminisme par regard en avant (*lookahead*) : analyse **prédictive**
    - algorithmes **LR(n)** et autres ***shift-reduce*** *parsers* [Aho *et al*. 1986, Tomita 1986]
  - analyse descendante prédictive — **LL(n**) — réduite aux grammaires non récursive à gauche

- **Application au TALN**
  - Pouvoir des grammaires hors-contexte légèrement insuffisant
  - Grammaires hors-contexte peu pratiques telles quelles pour le TALN
  - Cependant, tous les formalismes plus évolués développés pour le TALN resteront basés sur un substrat algorithmique hors-contexte : algorithme toujours d'actualité.

Bibliographie

Articles ou ouvrages d'entrée

- **Allen J.** (1995) Natural Language Understanding. Benjamin / Cummings Publ. Comp. Redwood City, CA.
- **Grosz B., Jones K., Webber B.** (Eds.) (1986) Readings in Natural Language Processing. Morgan Kaufmann Publishers Inc.
- **Aho A., Sethi R., Ullman J.D.** (1986) Compilers : principles, techniques and tools. Addison-Wesley, Radling, MA.

Articles ou ouvrages de référence

- **Aho A., Ullman J.D.** (1972) The theory of parsing, translation and compiling. Prentice-Hall.
- **Chomsky N**. (1956) Aspects of the theory of syntax. MIT Press.
- **Earley J.** (1970) An efficient context-free parsing algorithm. *Communication of the ACM*, 13(2), 94-102. (repris dans Grosz *et al*. 1986)
- **Kay M.** (1980) Algorithm shemata and data structures in syntactic processing. Rapport CSL-80-12. Xerox Corporation. (repris dans Grosz *et al*. 1986)
- **Tomita M.** (1986) Efficient parsing for natural language. Kluwer Acad. Publ., Boston

Bibliographie (2)

Travaux cités

- **Charniak E.** (1983) A parser with something for everyone. *In* King M. (Ed.) *Parsing natural language*. Academic Press, New-York, NJ.
- **Joshi A., Weir D., Vijay-Shanker K..** (1991) The convergence of mildly context-sensitive formalisms. In Sells P., Shieber S., Wasow T. (Eds.) *Foundational issues in NLP*. MIT Press, Cambridge, MA.
- **Marcus M.** (1980) A theory of syntactic recognition for natural language. MIT Press, Cambridge, MA.
- **Pullum G.K.** (1984) On two recent attempts to show that English is not a CFL. *American Journal of Computational Linguistics*, 10(3-4). 182-186.