Grammaires algébriques

2005/2006

Motivation

Définition

Comment définir ce qu'est un langage ? Pour décrire les différents sens prêtés à ce terme, le *Trésors de la Langue Française* propose les catégories suivantes :

Langage, [lɑ̃gaːʒ], n. m. :

1. Faculté ou système
   (a) Faculté d'expression et de communication
   (b) Système de signes vocaux et/ou graphiques
      i. Naturel
      ii. Artificiel
2. Moyen d'expression, usage

L'analyse syntaxique est consacrée au traitement de textes issus de tels systèmes de signes.

Syntaxe valide... ou non

Exemple 0.1.
- Cet artiste peint la nuit.
- *Cet artiste la nuit peint.
- La nuit, cet artiste peint.

Formaliser permet de reconnaître exactement ce qui est valide.

Cependant, l'analyse syntaxique n'est pas une fin en soi, mais une étape vers l'analyse sémantique et la compréhension du texte. La manière dont est décrit le langage doit se prêter à une analyse sémantique, ce qui suppose qu'elle soit révélatrice de la structure syntaxique du texte.

Structure syntaxique

Exhiber une structure donne une interprétation sémantique. Les linguistes décomposeraient la phrase « Cet artiste peint la nuit. » comme illustré dans la Figure 1 à l'aide de *syntagmes*, de groupes syntaxiques :

Ph représente une phrase,

SN représente un syntagme nominal,

SV représente un syntagme verbal,

SP représente un syntagme propositionnel.

Cycle de développement

Le cycle de développement de la Figure 2 montre la vie (un peu idéalisée) d'un programme depuis sa spécification jusqu'aux tests vérifiant que l'implémentation vérifie bien les détails de la spécification. Ce cycle se prête à des erreurs en particulier lors de la phase d'implémentation.

Un formalisme bien étudié utilisé comme spécification permet d'automatiser la tâche de dériver une implémentation : celle-ci est alors assurée de respecter la spécification.

C'est un objectif de plus pour la formalisation de la syntaxe : pouvoir automatiquement dériver un analyseur syntaxique correct.

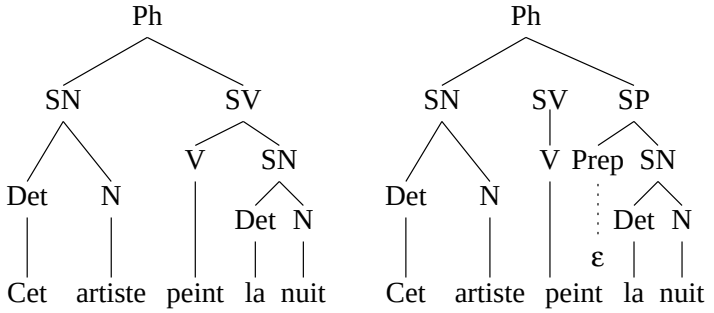


FIG. 1 – Deux structures possibles pour la phrase de l'Exemple 0.1.

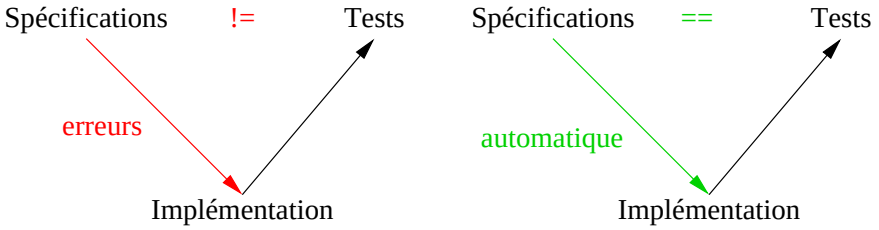


FIG. 2 – Le cycle de développement logiciel en V.

En bref
- Formaliser
- Exhiber la structure
- Permettre de dériver un analyseur

Table des matières

1 Langages

Comment comprendre un langage d'un point de vue mathématique ? C'est un ensemble de séquences de symboles, tous issus d'un alphabet. En algèbre, on définit des monoïdes pour représenter cette structure.

1.1 Monoïdes

Définition 1.1. Un triplet $\langle M, \cdot, \varepsilon \rangle$ est un *monoïde* si

– M est un ensemble,
– $\cdot$ est une opération binaire *associative*, vérifiant donc pour tout x, y et z de M

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z, \tag{1}$$

– et ε est l'*identité* pour $\cdot$, c'est-à-dire que pour tout x de M

$$\varepsilon \cdot x = x \cdot \varepsilon = x. \tag{2}$$

Exemple 1.2. Le singleton $\{x\}$ est un monoïde trivial pour $x \cdot x = x$.

Étant donné un ensemble A, l'ensemble des fonctions de A dans A est un monoïde utilisant la composition de fonctions ; la fonction identité sert simplement d'identité.

L'ensemble des listes bâties sur un ensemble est un monoïde : la concaténation sert d'opération binaire, et la liste vide d'identité.

Le couple $\langle M, \cdot \rangle$ est un *semigroupe*, et on appelle donc aussi un monoïde un semigroupe avec identité. Un semigroupe ne peut avoir qu'une seule identité.

L'opération de concaténation $\cdot$ est souvent plus simplement dénotée par la juxtaposition, c'est-à-dire que $x \cdot y$ s'écrit plus simplement xy.

On définit ensuite sur la base de $\cdot$ la puissance n d'un élément x de M par

$$x^0 = \varepsilon \text{ et} \tag{3}$$

$$x^n = xx^{n-1} \text{ pour } n > 1. \tag{4}$$

On étend souvent les opérations sur des éléments de M à des opérations sur des sous-ensembles de M, dans le cas de $\cdot$, cela donne étant donnés deux sous-ensembles A et B de M

$$A \cdot B = \{x \cdot y \mid x \in A \text{ et } y \in B\}. \tag{5}$$

Le triplet $\langle 2^M, \cdot, \{\varepsilon\} \rangle$ est alors lui aussi un monoïde.

Définition 1.3. Un sous-ensemble A d'un monoïde M est *fermé* si pour tout nombre naturel n

$$x_1, \ldots, x_n \in A \text{ implique } x_1 \cdots x_n \in A. \tag{6}$$

Si A est un sous-ensemble fermé de M, alors $\langle A, \cdot, \varepsilon \rangle$ est aussi un monoïde.

Définition 1.4. La *fermeture* d'un sous-ensemble A d'un monoïde M est

$$A^* = \bigcup_{n=0}^{\infty} A^n. \tag{7}$$

Lemme 1.5. *Soit A un sous-ensemble de M. Alors A^* est le plus petit sous-ensemble fermé de M contenant A.*

A génère M si $A^* = M$, et le génère *librement* s'il existe pour chaque élément x de M une unique décomposition $x_1 \cdots x_n$ en éléments de A. Le triplet $\langle A^*, \cdot, \varepsilon \rangle$ est donc trivialement un monoïde libre généré par A.

Lemme 1.6. *Si M est un monoïde libre, alors il est* simplifiable *à gauche et à droite, c'est-à-dire que pour tout x, y et z de M,*

$$zx = zy \text{ implique } x = y \text{ et} \tag{8}$$

$$xz = yz \text{ implique } x = y. \tag{9}$$

1.2 Langages formels

Un ensemble Σ est un *alphabet* ou *vocabulaire* s'il est fini et non vide.

Définition 1.7. Un *langage formel* sur un *alphabet* Σ est un sous-ensemble du *monoïde libre* $\langle \Sigma^*, \cdot, \varepsilon \rangle$ généré par Σ.

Exemple 1.8. Soit $\Sigma = \{a, b\}$; $\{a^n b^n \mid n \geq 0\}$ et $\{ww \mid w \in \Sigma^*\}$ sont des langages.

Exercice 1.9. Donner un exemple de semigroupe qui n'est pas un monoïde.

Exercice 1.10. Prouver qu'un semigroupe ne peut avoir qu'une unique identité.

Exercice 1.11. Montrer que pour tout ensemble A, $\langle 2^A, \cap, A \rangle$ et $\langle 2^A, \cup, \emptyset \rangle$ sont des monoïdes.

Le français comme un ensemble

Exemple 1.12.

$$\Sigma = \{\text{Adj}, \text{Adv}, \text{Det}, \text{N}, \text{Prep}, \text{V}\},$$

$$\begin{aligned}\mathcal{L}_{\text{français}} = \{&\text{Det N V},\\ &\text{Det Adj N V},\\ &\text{Det N V Adj},\\ &\text{Det Adj N V Adj},\\ &\text{Det N V Det N},\\ &\dots\}.\end{aligned}$$

... pas très pratique... et aucune structure !

1.3 Opérations sur les langages

La définition précédente n'est pas suffisante si l'on souhaite

1. se représenter un langage : les langages non bornés sont souvent difficiles à représenter sous forme d'ensembles ;
2. associer une structure au langage : que ce soit dans le cadre des langages de programmation ou dans celui des langues naturelles, l'analyse du texte a pour objectif d'expliciter la structure du texte et de permettre l'analyse sémantique.

Opérations

- opérations ensemblistes (union, intersection, complément) : $\cup, \cap, -$,
- concaténation (ou produit cartésien) : $\mathcal{L}_1\mathcal{L}_2 = \{xy \mid x \in \mathcal{L}_1, y \in \mathcal{L}_2\}$, étendue par $\mathcal{L}^0 = \{\varepsilon\}$ et, pour $i \geq 0$, $\mathcal{L}^{i+1} = \mathcal{L}^i\mathcal{L}$,
- étoile de Kleene : $\mathcal{L}^* = \cup_{i\geq 0}\mathcal{L}^i$,
- plus de Kleene : $\mathcal{L}^+ = \cup_{i\geq 1}\mathcal{L}^i$,
- quotient gauche : $\mathcal{L}_2\backslash\mathcal{L}_1 = \{w \mid \exists x \in \mathcal{L}_2, xw \in \mathcal{L}_1\}$,
- quotient droit : $\mathcal{L}_1/\mathcal{L}_2 = \{w \mid \exists x \in \mathcal{L}_2, wx \in \mathcal{L}_1\}$,
- ...

[1]

Diviser pour régner

Exemple 1.13.

$$\begin{aligned}\mathcal{L}_{\text{français}} =& \mathcal{L}_{\text{Ph}}^+\\ \mathcal{L}_{\text{Ph}} =& \mathcal{L}_{\text{SN}}\mathcal{L}_{\text{SV}}\\ &\cup \mathcal{L}_{\text{SN}}\mathcal{L}_{\text{SV}}\mathcal{L}_{\text{SP}}\\ &\cup \dots\\ \mathcal{L}_{\text{SN}} =& \{\text{Det}\}\mathcal{L}_{\text{SA}}\{\text{N}\}\mathcal{L}_{\text{SA}}\\ \mathcal{L}_{\text{SV}} =& \{\text{V}\}\\ &\cup \{\text{V}\}\mathcal{L}_{\text{SA}}\\ &\cup \{\text{V}\}\mathcal{L}_{\text{SN}}\\ &\cup \dots\\ \mathcal{L}_{\text{SA}} =& \{\text{Adj}\}^*\\ &\dots\end{aligned}$$

[1] L'ensemble Σ^* avec les opérations d'union $\cup$ et de concaténation $\cdot$ et la fonction d'étoile de Kleene * forme une *algèbre de Kleene*. Les expressions finies dans cette algèbre sont appelées *expressions rationnelles*, et définissent les langages rationnels. Les structures algébriques sont encore un autre moyen de décrire des classes de langages formels !

En subdivisant le langage $\mathcal{L}_{\text{français}}$ en ses constituants, sa structure devient apparente. Nos deux objectifs de formalisation et de structuration de la syntaxe semblent maintenant remplis. En revanche, la dérivation d'un analyseur syntaxique depuis cette formalisation n'est pas encore très claire. De plus, la formulation est un peu lourde.

Les deux formulations suivantes sont plus lisibles. Elles ont été développées indépendamment par Chomsky dans [Cho56] en vue de formaliser la syntaxe de la langue anglaise, et par Backus dans [Bac59] pour la syntaxe d'ALGOL 60 [BBG$^+$60]. On devra à Ginsburg le rapprochement des communautés de linguistes et d'informaticiens [GR62].

Grammaire syntagmatique [Cho56]

Exemple 1.14.

français→français Ph
français→Ph
Ph→SN SV
Ph→SN SV SP
Ph→...
SN→Det SA N SA
SV→V
SV→V SA
SV→V SN
SV→...
SA→SA Adj
SA→ε

Forme de Backus-Naur (BNF) [Bac59]

Exemple 1.15.

$<francais>::=<francais><Ph>$
$<francais>::=<Ph>$
$<Ph>::=<SN><SV>$
$<Ph>::=<SN><SV><SP>$
$<Ph>::=\ldots$
$<SN>::=$Det $<SA>$ N $<SA>$
$<SV>::=$V
$<SV>::=$V $<SA>$
$<SV>::=$V $<SN>$
$<SV>::=\ldots$
$<SA>::=<SA>$ Adj
$<SA>::=\varepsilon$

Nous allons vérifier que ces formulations sont bien équivalentes à la décomposition en sous-langages de l'Exemple 1.13.

2 Systèmes de réécriture

Les formalisations précédentes sont en fait des systèmes de réécriture.

2.1 Définitions

Systèmes de réécriture

Définition 2.1. $\langle V, P\rangle$ est un *système de réécriture*
- V est un vocabulaire et
- P un sous-ensemble de $V^* \times V^*$.

Les éléments $\alpha{\rightarrow}\beta$ de P sont appelés des *règles de réécriture*.

Dérivations

Définition 2.2. Soit $\langle V, P\rangle$ un système de réécriture.

La relation de *dérivation* $\overset{r}{\Rightarrow}$ sur V^* est définie pour tout φ et σ de V^* par

$$\varphi\alpha\sigma \overset{r}{\Rightarrow} \varphi\beta\sigma \text{ ssi } r = \alpha{\rightarrow}\beta \in P. \tag{10}$$

Cette relation est étendue aux séquences de règles dans le monoïde libre P^* par

$$\overset{\varepsilon}{\Rightarrow} = \mathrm{id}_{V^*}$$
$$\overset{r\pi}{\Rightarrow} = \overset{r}{\Rightarrow}\overset{\pi}{\Rightarrow}.$$

Enfin,

$$\Rightarrow = \bigcup_{r\in P} \overset{r}{\Rightarrow} \text{ et}$$
$$\Rightarrow^* = \bigcup_{\pi\in P^*} \overset{\pi}{\Rightarrow}.$$

Langage décrit par un système de réécriture

On distingue souvent une chaîne de caractères ρ de V^* ; on peut alors définir un langage par
- génération depuis l'axiome ρ comme

$$\mathcal{L}(\langle V, P\rangle) = \{\omega \mid \rho{\Rightarrow}^*\omega\}, \tag{11}$$

ou bien par
- reconnaissance dans l'état d'acceptation ρ comme

$$\mathcal{L}(\langle V, P\rangle) = \{\omega \mid \omega \models^* \rho\}. \tag{12}$$

(Dans ce dernier cas, on substitue les notations $\vdash$ et $\models$ à $\rightarrow$ et $\Rightarrow$.)

Deux systèmes $\langle V_1, P_1\rangle$ et $\langle V_2, P_2\rangle$ sont équivalents si $\mathcal{L}(\langle V_1, P_1\rangle) = \mathcal{L}(\langle V_2, P_2\rangle)$.

2.2 Grammaires

Grammaires syntagmatiques [Cho59]

Définition 2.3. Un quadruplet $\langle \Sigma, N, S, P\rangle$ est une *grammaire syntagmatique* $\mathcal{G}$ (en anglais *phrase structure grammar*) :
- Σ est l'alphabet terminal,
- N est l'alphabet non terminal,
- $S \in N$ est l'axiome,
- P est un ensemble de règles de la forme $\varphi A\sigma{\rightarrow}\varphi\alpha\sigma$ où $A \in N$.

Soit $V = \Sigma \cup N$ le vocabulaire, $\langle V, P\rangle$ est bien un système de réécriture génératif.

Conventions de notation
- $a, b, c, d, \ldots$ sont des éléments de Σ ;
- $A, B, C, D, \ldots$ sont des éléments de N ;
- X, Y, Z sont des éléments de V ;
- u, v, w, x, y, z sont des éléments de Σ^* ;
- $\alpha, \beta, \gamma, \delta, \ldots$ sont des éléments de V^*.

Hiérarchie de Chomsky [Cho59]

Restrictions sur la forme des règles de P :

0. $\alpha \to \beta$ avec α dans V^+ et β dans V^* : grammaires contextuelles avec effacement ; ne sont pas des grammaires syntagmatiques ;
1. $\varphi A \sigma \to \varphi \alpha \sigma$ avec φ et σ dans V^*, A dans N et α dans V^+ : grammaires contextuelles ou monotones ;
2. $A \to \alpha$ avec A dans N et α dans V^* : grammaires non contextuelles ou algébriques ;
3. $A \to \alpha$ avec A dans N et α dans Σ^* ou dans $\Sigma^* \cdot N$: grammaires linéaires à droite.

Machines et automates

La hiérarchie de Chomsky trouve un écho dans les systèmes de réécriture recognitifs :

0. Langages récursivement énumérables, reconnaissables par une Machine de Turing (TM) ;
1. Langages contextuels, reconnaissables par une Machine de Turing dont le ruban a une longueur linéaire de la taille des données traitées (LBTM) ;
2. Langages non contextuels ou algébriques, reconnaissables par un automate à pile (*Push-Down Automaton*, PDA) ;
3. Langages rationnels, reconnaissables par un automate d'états finis (*Finite-State Automaton*, FSA).

Nous voici en possession d'une équivalence entre langages générés par certains systèmes génératifs et langages reconnus par certains systèmes recognitifs. Nous ne sommes plus si éloignés d'une dérivation automatique d'un analyseur syntaxique.

Exercice 2.4. Donner un algorithme de transformation d'une grammaire linéaire à droite en grammaire linéaire à gauche.

Exercice 2.5. Donner un algorithme de construction d'un automate d'états finis équivalent à une grammaire linéaire à droite. On en déduit par le théorème de Kleene [Kle56] que les expressions rationnelles sont une autre description finie des grammaires linéaires.

2.3 Grammaires algébriques

Les grammaires algébriques modélisent les notations vues dans les Exemples 1.14 et 1.15.

Langages algébriques

L'ensemble des langages algébriques est fermé par
- union
- concaténation
- intersection avec un langage rationnel
- homomorphisme
- ...

Les deux premiers points montrent que les notations de Chomsky et de Backus-Naur des Exemples 1.14 et 1.15 sont bien équivalentes aux équations de langages de l'Exemple 1.13.

Dérivations droites et gauches

Définition 2.6. Une *dérivation droite* $\underset{\text{rm}}{\Rightarrow}$ est définie par

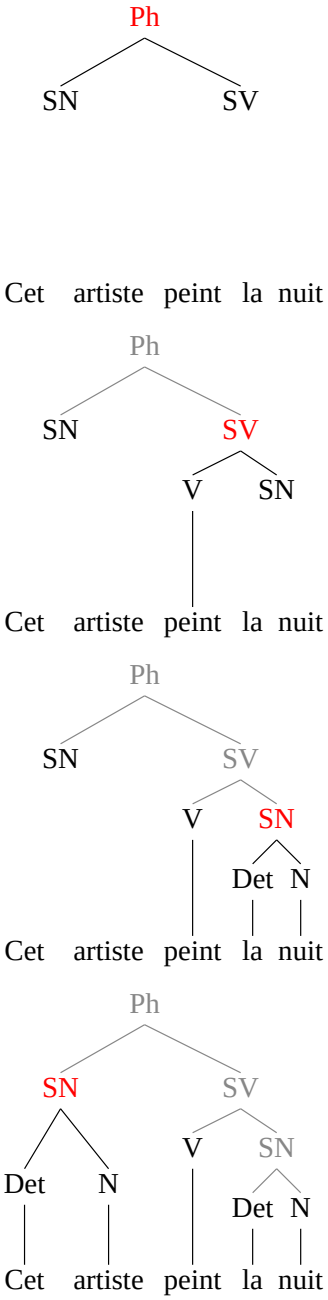
$$\varphi A x \underset{\text{rm}}{\Rightarrow} \varphi \alpha x \text{ ssi } A \to \alpha \in P \tag{13}$$

Définition 2.7. Une *dérivation gauche* $\underset{\text{lm}}{\Rightarrow}$ est définie par

$$y A \sigma \underset{\text{lm}}{\Rightarrow} y \alpha \sigma \text{ ssi } A \to \alpha \in P \tag{14}$$

Parcours d'un arbre syntaxique

Exemple 2.8. Lors d'une dérivation droite :



Une seule dérivation droite (ou gauche) par arbre de dérivation.

Ambiguïté

L'existence de deux arbres syntaxiques différents marque une ambiguïté.

Théorème 2.9. *Soit $\mathcal{G}$ une grammaire algébrique ; elle est ambiguë si et seulement si une phrase de $\mathcal{L}_\mathcal{G}$ possède plus d'une dérivation droite (ou gauche).*

Il existe des langages algébriques intrinsèquement ambigus.

Exemple 2.10 ([Par66]). Le langage $\{a^i b^j c^k \mid i = j \text{ ou } j = k\}$ est intrinsèquement ambigu.

Théorème 2.11 ([Can62, CS63])**.** *Le problème de l'ambiguïté d'une grammaire algébrique n'est pas décidable.*

2.4 Analyseurs

Automate à pile

Définition 2.12. Un *automate à pile* $\mathcal{A} = \langle Q, \Sigma, R, \varphi_s, F, \$, \| \rangle$ est un système de réécriture $\langle Q \cup \Sigma \cup \{\$, \|\}, R \rangle$ recognitif où

- Q est l'alphabet de pile,
- Σ est l'alphabet d'entrée,
- $R \subseteq \{\$\} \cdot Q^* \cdot \{\|\} \cdot \Sigma^* \cdot \{\$\}$ est un ensemble de règles

$$\varphi \| xy \vdash \psi \| y, \tag{15}$$

- $\varphi_s \in Q^*$ est le contenu initial de la pile,
- $F \subseteq Q^*$ est l'ensemble des contenus finaux de la pile,
- $\$$ est le marqueur de fin, et
- $\|$ est le délimiteur de sommet de pile.

$$\mathcal{L}_{\mathcal{A}} = \{w \mid \$\varphi_s \| w\$ \models^* \$\varphi_f \|\$ \text{ avec } \varphi_f \in F\}. \tag{16}$$

Analyseur descendant

Définition 2.13. L'*analyseur récursif descendant* pour une grammaire algébrique $\mathcal{G} = \langle N, \Sigma, P, S \rangle$ est un automate à pile $\mathcal{A}_{\text{desc}} = \langle Q, \Sigma, R, \varphi_s, F, \$, \| \rangle$ où

- Q est l'ensemble des production pointées de P', notées

$$[A \to \alpha \bullet \alpha'] \text{ si } A \to \alpha\alpha' \in P', \tag{17}$$

- R est l'ensemble des règles

$$[A \to \alpha \bullet B\alpha'] \| \underset{\text{predict}}{\vdash} [A \to \alpha B \bullet \alpha'][B \to \bullet \beta] \|,$$

$$[A \to \alpha \bullet a\alpha'] \| a \underset{\text{match}}{\vdash} [A \to \alpha a \bullet \alpha'] \|,$$

$$[A \to \alpha \bullet] \| \vdash \|,$$

- $\varphi_s = [S' \to \bullet S\$]$,
- $F = \{[S' \to S \bullet \$]\}$.

Analyse descendante

Exemple 2.14.

$$
\begin{aligned}
& \$[S'\to\bullet \text{Ph } \$]\|\text{Det N V Det N\$} \\
\vdash_{\text{predict}} \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\bullet \text{SN SV}]\|\text{Det N V Det N\$} \\
\vdash_{\text{predict}} \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN}\bullet\text{SV}][\text{SN}\to\bullet \text{Det N}]\|\text{Det N V Det N\$} \\
\vdash_{\text{match}} \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN}\bullet\text{SV}][\text{SN}\to\text{Det}\bullet\text{N}]\|\text{N V Det N\$} \\
\vdash_{\text{match}} \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN}\bullet\text{SV}][\text{SN}\to\text{Det N}\bullet]\|\text{V Det N\$} \\
\vdash \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN}\bullet\text{SV}]\|\text{V Det N\$} \\
\vdash_{\text{predict}} \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN SV}\bullet][\text{SV}\to\bullet \text{V SN}]\|\text{V Det N\$} \\
\vdash_{\text{match}} \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN SV}\bullet][\text{SV}\to\text{V}\bullet\text{SN}]\|\text{Det N\$} \\
\vdash_{\text{predict}} \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN SV}\bullet][\text{SV}\to\text{V SN}\bullet][\text{SN}\to\bullet \text{Det N}]\|\text{Det N\$} \\
\vdash_{\text{match}} \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN SV}\bullet][\text{SV}\to\text{V SN}\bullet][\text{SN}\to\text{Det}\bullet\text{N}]\|\text{N\$} \\
\vdash_{\text{match}} \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN SV}\bullet][\text{SV}\to\text{V SN}\bullet][\text{SN}\to\text{Det N}\bullet]\|\$ \\
\vdash \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN SV}\bullet][\text{SV}\to\text{V SN}\bullet]\|\$ \\
\vdash \; & \$[S'\to\text{Ph}\bullet\$][\text{Ph}\to\text{SN SV}\bullet]\|\$ \\
\vdash \; & \$[S'\to\text{Ph}\bullet\$]\|\$
\end{aligned}
$$

Transducteur à pile

L'analyse précédente ne dit que si une phrase appartient au langage de $\mathcal{G}$ ou non.

Définition 2.15. Un *transducteur à pile* $\langle \mathcal{A}, \tau \rangle$ pour $\mathcal{G}$ est constitué d'un automate à pile $\mathcal{A}$ pour $\mathcal{G}$ et d'un homomorphisme τ de R^* dans P^* défini par

$$\tau([A\to\alpha\bullet B\alpha']\| \vdash_{\text{predict}} [A\to\alpha B\bullet\alpha'][B\to\bullet\beta]\|) = B\to\beta,$$

$$\tau([A\to\alpha\bullet a\alpha']\|a \vdash_{\text{match}} [A\to\alpha a\bullet\alpha']) = \varepsilon, et$$

$$\tau([A\to\alpha\bullet]\| \vdash \|) = \varepsilon.$$

L'exemple précédent donnait les productions dans l'ordre d'une dérivation gauche.

Faiblesses de l'analyse descendante

1. L'automate à pile est non déterministe ; il nécessite un *backtrack* coûteux ;
2. l'ensemble de règles R permet des règles de la forme $[A\to\bullet A\alpha]\| \vdash_{\text{predict}} [A\to A\bullet\alpha][A\to\bullet A\alpha]\|$ si la grammaire est récursive gauche.

La suite du cours sera consacrée à des constructions de machines déterministes plus puissantes.

3 Conclusion

En résumé

Les grammaires algébriques permettent

1. de générer des langages formels de la classe des langages algébriques ;
2. de structurer la syntaxe sous forme d'arbres de dérivation ;
3. une analyse à l'aide d'automates à pile.

Sur la dérivation d'un analyseur syntaxique généraliste, lire [Ear70] en préparation de la séance numéro 4.

Questions ?

3.1 Exercices

Exercices

Exercice 3.1. Soit la grammaire

$$E \to E + T \,|\, T,\ T \to T * F \,|\, F,\ F \to a \,|\, (E). \qquad (\mathcal{G}_1)$$

Donner les arbres de dérivation correspondant aux phrases

1. $a + a * a + a$ et
2. $a * (a + a + a)$.

Donnez les étapes de l'analyse de ces phrases par un automate à pile récursif descendant.

Exercice 3.2. Le langage $\{a^i b^j c^k \mid i = j \text{ ou } j = k\}$ est intrinsèquement ambigu. Donnez une grammaire algébrique générant ce langage. Montrez l'ambiguïté de cette grammaire.

References

[Bac59] John W. Backus. The syntax and semantics of the proposed international algebraic language of the Zürich ACM-GAMM Conference. In *IFIP Congress*, pages 125–131, 1959.

[BBG+60] John W. Backus, Friedrich L. Bauer, Julien Green, C. Katz, John McCarthy, Alan J. Perlis, Heinz Rutishauser, Klaus Samelson, Bernard Vauquois, Joseph Henry Wegstein, Adriaan van Wijngaarden, and Michael Woodger. Report on the algorithmic language ALGOL 60. *Communications of the ACM*, 3(5):299–314, 1960.

[Can62] David G. Cantor. On the ambiguity problem of Backus systems. *Journal of the ACM*, 9(4):477–479, 1962.

[Cho56] Noam Chomsky. Three models for the description of language. *IEEE Transactions on Information Theory*, 2:113–124, 1956.

[Cho59] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959.

[CS63] Noam Chomsky and Marcel Paul Schützenberger. *The Algebraic Theory of Context-free Languages*. Computer Programming and Formal Systems. North-Holland Publishing Co., Amsterdam, 1963.

[Ear70] Jay Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, 1970.

[GR62] Seymour Ginsburg and H. Gordon Rice. Two families of languages related to ALGOL. *Journal of the ACM*, 9(3):350–371, 1962.

[Kle56] Stephen C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–40. Princeton University Press, Princeton, NJ, 1956.

[Par66] Rohit J. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.