Première partie I

Grammaires algébriques

Syntaxe valide

Exemple

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

Syntaxe valide... ou non

Exemple

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

Syntaxe valide... ou non

Exemple

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

Syntaxe valide... ou non

Exemple

- Cet artiste peint la nuit.
- *La nuit cet artiste peint.
- La nuit, cet artiste peint.
- La nuit peint cet artiste.
- La nuit nuit à cet artiste.

Syntaxe valide... ou non

Exemple

- Cet artiste peint la nuit.
- *La nuit cet artiste peint.
- La nuit, cet artiste peint.
- La nuit peint cet artiste.
- La nuit nuit à cet artiste.

Un langage formel, pour un vocabulaire donné, est un ensemble de phrases syntaxiquement correctes utilisant ce vocabulaire.

Compétence et performance linguistique

Chomsky

Exemple

- *La nuit cet artiste peint.

Et si Yoda est l'élocuteur ?

Compétence et performance linguistique

Chomsky

Exemple

- *La nuit cet artiste peint.
- La nuit cet artiste peint.

Et si Yoda est l'élocuteur ?

Compétence et performance linguistique

Chomsky

Exemple

- *La nuit cet artiste peint.
- La nuit cet artiste peint.

Et si Yoda est l'élocuteur ?

Compétence et performance linguistique

Chomsky

Exemple

- *La nuit cet artiste peint.
- Cet artiste peint la nuit.

Et si Yoda est l'élocuteur ?

Compétence et performance linguistique

CHOMSKY

Exemple

- *La nuit cet artiste peint.
- Cet artiste peint la nuit.

Performance la phrase dite ou écrite

Compétence et performance linguistique

CHOMSKY

Exemple

- *La nuit cet artiste peint.
- Cet artiste peint la nuit.

Performance la phrase dite ou écrite

Compétence comment construire et comprendre une phrase correcte

Compétence et performance linguistique

CHOMSKY

Exemple

- *La nuit cet artiste peint.
- Cet artiste peint la nuit.

Performance la phrase dite ou écrite

Compétence comment construire et comprendre une phrase correcte
$\Rightarrow$ modèle (fini) du langage

Compétence linguistique

Exemple

- Cet artiste peint la nuit.

Compétence linguistique

Exemple

- Cet artiste peint la nuit.
- Cet artiste original peint la nuit.

Compétence linguistique

Exemple

- Cet artiste peint la nuit.
- Cet artiste original peint la nuit.
- Cet artiste original et méconnu peint la nuit.

Compétence linguistique

Exemple

- Cet artiste peint la nuit.
- Cet artiste original peint la nuit.
- Cet artiste original et méconnu peint la nuit.
- Cet artiste original, talentueux et méconnu peint la nuit.

Compétence linguistique

Exemple

- Cet artiste peint la nuit.
- Cet artiste original peint la nuit.
- Cet artiste original et méconnu peint la nuit.
- Cet artiste original, talentueux et méconnu peint la nuit.
- Cet artiste original, talentueux, illuminé et méconnu peint la nuit.

Compétence linguistique

Exemple

- Cet artiste peint la nuit.
- Cet artiste original peint la nuit.
- Cet artiste original et méconnu peint la nuit.
- Cet artiste original, talentueux et méconnu peint la nuit.
- Cet artiste original, talentueux, illuminé et méconnu peint la nuit.
- ...

Compétence linguistique

Exemple

- Cet artiste peint la nuit.
- Cet artiste original peint la nuit.
- Cet artiste original et méconnu peint la nuit.
- Cet artiste original, talentueux et méconnu peint la nuit.
- Cet artiste original, talentueux, illuminé et méconnu peint la nuit.
- ...

- phrases correctes

Compétence linguistique

Exemple

- Cet artiste peint la nuit.
- Cet artiste original peint la nuit.
- Cet artiste original et méconnu peint la nuit.
- Cet artiste original, talentueux et méconnu peint la nuit.
- Cet artiste original, talentueux, illuminé et méconnu peint la nuit.
- ...

- phrases correctes
- règles de formation simples et en petit nombre : ajout d'adjectifs, de ponctuation et de conjonctions

Compétence linguistique

Exemple

- Cet artiste peint la nuit.
- Cet artiste original peint la nuit.
- Cet artiste original et méconnu peint la nuit.
- Cet artiste original, talentueux et méconnu peint la nuit.
- Cet artiste original, talentueux, illuminé et méconnu peint la nuit.
- ...

- phrases correctes
- règles de formation simples et en petit nombre : ajout d'adjectifs, de ponctuation et de conjonctions
- la compétence permet des phrases de longueur infinie : limite pragmatique

Règles de réécriture
Thue

Exemple

$$n\ v \longrightarrow n\ adj\ v$$

Cet artiste peint la nuit.

Règles de réécriture
Thue

Exemple

$$n\ v \longrightarrow n\ adj\ v$$

Cet artiste peint la nuit.

Règles de réécriture
Thue

Exemple

$$n\ v \longrightarrow n\ adj\ v$$

Cet artiste original peint la nuit.

Règles de réécriture

Thue

Exemple

$$n\ v \longrightarrow n\ adj\ v$$

Cet artiste original peint la nuit.

Règles de réécriture

Thue

Exemple

$$n\ v \longrightarrow n\ adj\ v$$

Cet artiste original peint la nuit.

Exemple

$$n\ adj\ v \longrightarrow n\ adj \text{ et } adj\ v$$

Cet artiste original et méconnu peint la nuit.

Règles de réécriture
Thue

Exemple

$$n\ v \longrightarrow n\ adj\ v$$

Cet artiste original peint la nuit.

Exemple

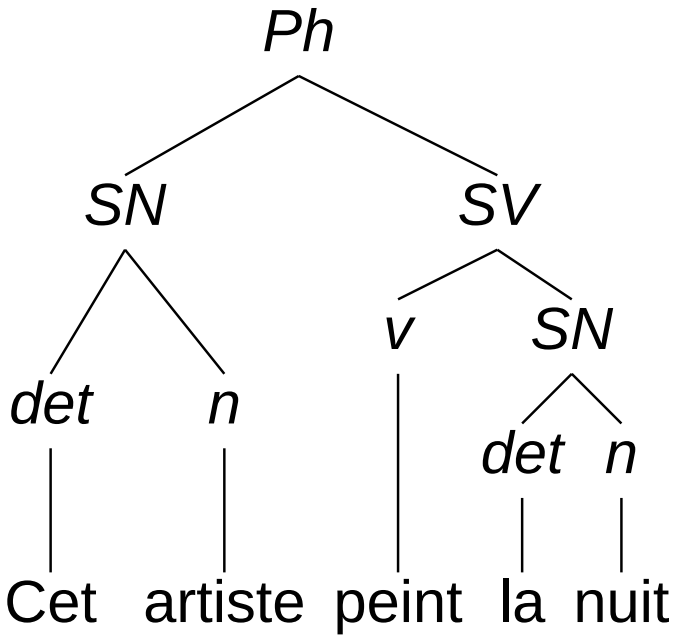
$$n\ adj\ v \longrightarrow n\ adj \text{ et } adj\ v$$

Cet artiste original et méconnu peint la nuit.

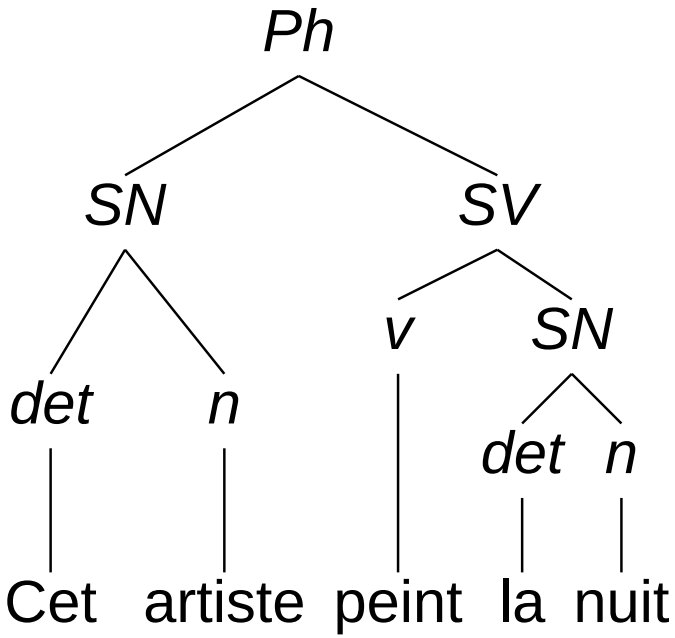
Exercice

Quelle règle de réécriture pour les ajouts d'articles dans « Cet artiste original, talentueux et méconnu peint la nuit. » et dans « Cet artiste original, talentueux, illuminé et méconnu peint la nuit. » ?

Grammaires syntagmatiques



Grammaires syntagmatiques

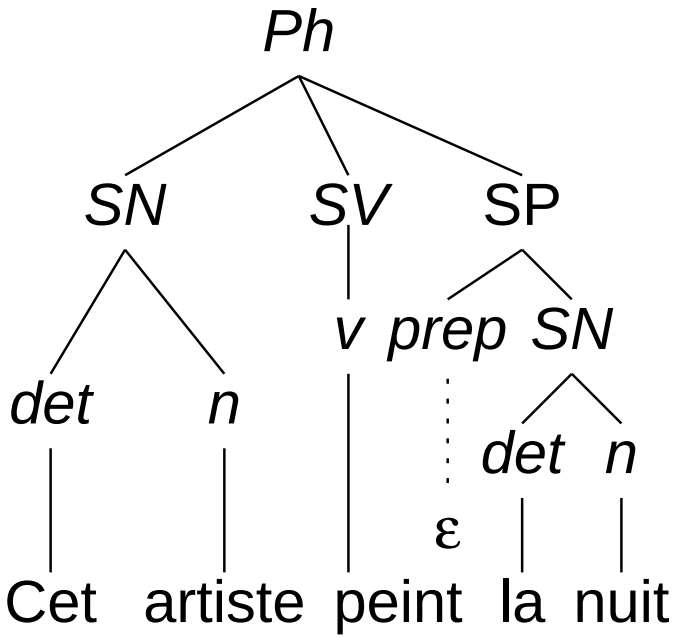


Faire intervenir les catégories SN, SV, etc. dans les règles de réécriture.

Exemple

$$Ph \rightarrow SN\ SV$$

Ambiguïté

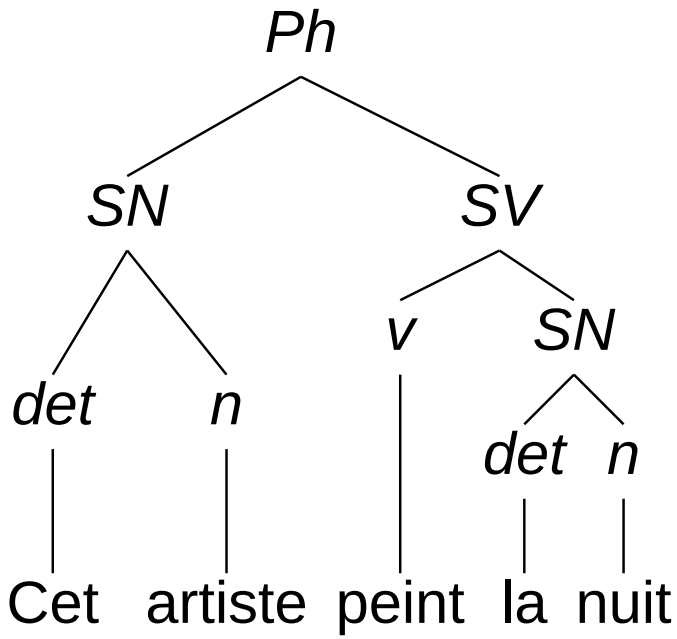


Analyse syntaxique

- la grammaire (ensemble de règles de réécriture $Ph \longrightarrow SN\ SV$)
- la phrase à analyser (« Cet artiste peint la nuit. »)

Analyse syntaxique

- la grammaire (ensemble de règles de réécriture $Ph \longrightarrow SN\ SV$)
- la phrase à analyser (« Cet artiste peint la nuit. »)
- la (les) structure(s) syntaxique(s)



En bref

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

Langages formels

Définition

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

Langages formels

Définition

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

- Σ est un ensemble fini non vide

Langages formels

Définition

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

- Σ est un ensemble fini non vide
- $\cdot$ est une opération de concaténation associative $: \cdot : \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$ vérifie

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

Langages formels

Définition

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

- Σ est un ensemble fini non vide
- $\cdot$ est une opération de concaténation associative
- ε est son identité

$$x \in \Sigma^* \text{ implique } \varepsilon \cdot x = x \cdot \varepsilon = x$$

Langages formels

Définition

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

- Σ est un ensemble fini non vide
- $\cdot$ est une opération de concaténation associative
- ε est son identité
- Σ^* est la fermeture de Σ

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

Langages formels

Définition

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

- Σ est un ensemble fini non vide
- $\cdot$ est une opération de concaténation associative
- ε est son identité
- Σ^* est la fermeture de Σ

Exemple

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

Opérations sur les langages

- 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 : $\mathcal{L}_1/\mathcal{L}_2 = \{w \mid \exists x \in \mathcal{L}_2, wx \in \mathcal{L}_1\}$,
- ...

Opérations sur les langages

- 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 : $\mathcal{L}_1 / \mathcal{L}_2 = \{w \mid \exists x \in \mathcal{L}_2, wx \in \mathcal{L}_1\}$,
- ...

Opérations sur les langages

- 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 : $\mathcal{L}_1/\mathcal{L}_2 = \{w \mid \exists x \in \mathcal{L}_2, wx \in \mathcal{L}_1\}$,
- ...

Opérations sur les langages

- 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 : $\mathcal{L}_1/\mathcal{L}_2 = \{w \mid \exists x \in \mathcal{L}_2, wx \in \mathcal{L}_1\}$,
- ...

Opérations sur les langages

- 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 : $\mathcal{L}_1 / \mathcal{L}_2 = \{w \mid \exists x \in \mathcal{L}_2, wx \in \mathcal{L}_1\}$,
- ...

Opérations sur les langages

- 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 : $\mathcal{L}_1 / \mathcal{L}_2 = \{w \mid \exists x \in \mathcal{L}_2, wx \in \mathcal{L}_1\}$,
- ...

Opérations sur les langages

- 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 : $\mathcal{L}_1/\mathcal{L}_2 = \{w \mid \exists x \in \mathcal{L}_2, wx \in \mathcal{L}_1\}$,
- ...

Le français comme un ensemble

Exemple

$$\Sigma = \{adj, adv, det, n, prep, v, \text{et}\},$$

$$\begin{aligned}\mathcal{L}_{\text{français}} = \{&det\ n\ v,\\ &det\ n\ adj\ v,\\ &det\ n\ adj\ \text{et}\ adj\ v,\\ &\dots\}.\end{aligned}$$

Le français comme un ensemble

Exemple

$$\Sigma = \{adj, adv, det, n, prep, v, \text{et}\},$$

$$\begin{aligned}\mathcal{L}_{\text{français}} = \{&det\ n\ v,\\ &det\ n\ adj\ v,\\ &det\ n\ adj\ \text{et}\ adj\ v,\\ &\dots\}.\end{aligned}$$

Le français comme un ensemble

Exemple

$$\Sigma = \{adj, adv, det, n, prep, v, \text{et}\},$$

$$\begin{aligned}\mathcal{L}_{\text{français}} = \{&det\ n\ v,\\ &det\ n\ adj\ v,\\ &det\ n\ adj\ \text{et}\ adj\ v,\\ &\dots\}.\end{aligned}$$

Le français comme un ensemble

Exemple

$$\Sigma = \{adj, adv, det, n, prep, v, \text{et}\},$$

$$\begin{aligned}\mathcal{L}_{\text{français}} = \{&det\ n\ v,\\ &det\ n\ adj\ v,\\ &det\ n\ adj\ \text{et}\ adj\ v,\\ &\dots\}.\end{aligned}$$

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

Systèmes de réécriture

Définition

$\langle V, P \rangle$ est un système de réécriture

- V est un vocabulaire et
- P un sous-ensemble de $V^* \times V^*$.

Systèmes de réécriture

Définition

$\langle V, P \rangle$ est un système de réécriture

- V est un vocabulaire et
- P un sous-ensemble de $V^* \times V^*$.

Systèmes de réécriture

Définition

$\langle V, P \rangle$ est un système de réécriture

- V est un vocabulaire et
- P un sous-ensemble de $V^* \times V^*$.

Systèmes de réécriture

Définition

$\langle V, P \rangle$ 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.

Systèmes de réécriture

Définition

$\langle V, P \rangle$ 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.

Exemple

$$n\ v \longrightarrow n\ adj\ v$$

Dérivations

Définition

La relation de dérivation $\overset{r}{\Rightarrow}$ sur V^* est définie par

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

Dérivations

Définition

La relation de dérivation $\overset{r}{\Rightarrow}$ sur V^* est définie par

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

Dérivations

Définition

La relation de dérivation $\overset{r}{\Rightarrow}$ sur V^* est définie par

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

Dérivations

Définition

La relation de dérivation $\overset{r}{\Rightarrow}$ sur V^* est définie par

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

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

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

Dérivations

Définition

La relation de dérivation $\overset{r}{\Rightarrow}$ sur V^* est définie par

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

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

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

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

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

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

- reconnaissance dans l'état d'acceptation ρ comme

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

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

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

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

- reconnaissance dans l'état d'acceptation ρ comme

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

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

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

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

- reconnaissance dans l'état d'acceptation ρ comme

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

(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 $\mathcal{L}(\langle V_1, P_1\rangle) = \mathcal{L}(\langle V_2, P_2\rangle)$.

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

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

- reconnaissance dans l'état d'acceptation ρ comme

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

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

Grammaires syntagmatiques [Cho59]

Définition

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 $A \in N$.

Grammaires syntagmatiques [Cho59]

Définition

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 $A \in N$.

Grammaires syntagmatiques [Cho59]

Définition

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 $A \in N$.

Grammaires syntagmatiques [Cho59]

Définition

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 $A \in N$.

Grammaires syntagmatiques [Cho59]

Définition

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 $A \in N$.

Grammaires syntagmatiques [Cho59]

Définition

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^*.

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^*.

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^*.

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^*.

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^*.

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

Hiérarchie de Chomsky [Cho59]

Restrictions sur la forme des règles de P :

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

Hiérarchie de Chomsky [Cho59]

Restrictions sur la forme des règles de P :

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

Hiérarchie de Chomsky [Cho59]

Restrictions sur la forme des règles de P :

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

Hiérarchie de Chomsky [Cho59]

Restrictions sur la forme des règles de P :

0. $\alpha \rightarrow \beta$ avec α dans V^+ et β dans V^* : grammaires contextuelles avec effacement ;
1. $\varphi A \sigma \rightarrow \varphi \alpha \sigma$ avec φ et σ dans V^*, A dans N et α dans V^+ : grammaires contextuelles ou monotones ;
2. $A \rightarrow \alpha$ avec A dans N et α dans V^* : grammaires non contextuelles ou algébriques ;
3. $A \rightarrow \alpha$ 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).

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).

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).

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).

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).

Grammaire algébrique (CFG) [Cho56]

Exemple

français $\longrightarrow$ français Ph

français $\longrightarrow Ph$

$Ph \longrightarrow SN\ SV$

$Ph \longrightarrow SN\ SV\ SP$

$Ph \longrightarrow \ldots$

$SN \longrightarrow det\ SA\ n\ SA$

$SV \longrightarrow v$

$SV \longrightarrow v\ SA$

$SV \longrightarrow v\ SN$

$SV \longrightarrow \ldots$

$SA \longrightarrow SA\ adj$

$SA \longrightarrow \varepsilon$

Forme de Backus-Naur (BNF) [Bac59]

Exemple

$$<francais> ::= <francais> \ <Ph>$$

$$<francais> ::= <Ph>$$

$$<Ph> ::= <SN> \ <SV>$$

$$<Ph> ::= <SN> \ <SV> \ <SP>$$

$$<Ph> ::= \ldots$$

$$<SN> ::= det \ <SA> \ n \ <SA>$$

$$<SV> ::= v$$

$$<SV> ::= v \ <SA>$$

$$<SV> ::= v \ <SN>$$

$$<SV> ::= \ldots$$

$$<SA> ::= <SA> \ adj$$

$$<SA> ::= \varepsilon$$

Systèmes d'équations [GR62]

Exemple

$$
\begin{aligned}
\mathcal{L}_{\text{français}} &= \mathcal{L}_{Ph}^{+} \\
\mathcal{L}_{Ph} &= \mathcal{L}_{SN}\mathcal{L}_{SV} \\
&\quad \cup \mathcal{L}_{SN}\mathcal{L}_{SV}\mathcal{L}_{SP} \\
&\quad \cup \dots \\
\mathcal{L}_{SN} &= \{det\}\mathcal{L}_{SA}\{n\}\mathcal{L}_{SA} \\
\mathcal{L}_{SV} &= \{v\} \\
&\quad \cup \{v\}\mathcal{L}_{SA} \\
&\quad \cup \{v\}\mathcal{L}_{SN} \\
&\quad \cup \dots \\
\mathcal{L}_{SA} &= \{adj\}^{*} \\
&\quad \dots
\end{aligned}
$$

Langages algébriques

L'ensemble des langages algébriques est fermé par

- union
- concaténation

Langages algébriques

L'ensemble des langages algébriques est fermé par

- union
- concaténation

Langages algébriques

L'ensemble des langages algébriques est fermé par

- union
- concaténation

Langages algébriques

L'ensemble des langages algébriques est fermé par

- union
- concaténation
- intersection avec un langage rationnel

Langages algébriques

L'ensemble des langages algébriques est fermé par

- union
- concaténation

- homomorphisme

Langages algébriques

L'ensemble des langages algébriques est fermé par

- union
- concaténation

- ...

Dérivations droites et gauches

Définition

Une dérivation droite $\underset{\text{rm}}{\Rightarrow}$ est définie par

$$\varphi A x \underset{\text{rm}}{\Rightarrow} \varphi \alpha x \text{ ssi } A \rightarrow \alpha \in P$$

Définition

Une dérivation gauche $\underset{\text{lm}}{\Rightarrow}$ est définie par

$$y A \sigma \underset{\text{lm}}{\Rightarrow} y \alpha \sigma \text{ ssi } A \rightarrow \alpha \in P$$

Dérivations droites et gauches

Définition

Une dérivation droite $\underset{\text{rm}}{\Rightarrow}$ est définie par

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

Définition

Une dérivation gauche $\underset{\text{lm}}{\Rightarrow}$ est définie par

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

Dérivations droites et gauches

Définition

Une dérivation droite $\underset{\text{rm}}{\Rightarrow}$ est définie par

$$\varphi Ax \underset{\text{rm}}{\Rightarrow} \varphi \alpha x \text{ ssi } A \rightarrow \alpha \in P$$

Définition

Une dérivation gauche $\underset{\text{lm}}{\Rightarrow}$ est définie par

$$yA\sigma \underset{\text{lm}}{\Rightarrow} y\alpha\sigma \text{ ssi } A \rightarrow \alpha \in P$$

Dérivations droites et gauches

Définition

Une dérivation droite $\underset{\text{rm}}{\Rightarrow}$ est définie par

$$\varphi A x \underset{\text{rm}}{\Rightarrow} \varphi \alpha x \text{ ssi } A \rightarrow \alpha \in P$$

Définition

Une dérivation gauche $\underset{\text{lm}}{\Rightarrow}$ est définie par

$$y A \sigma \underset{\text{lm}}{\Rightarrow} y \alpha \sigma \text{ ssi } A \rightarrow \alpha \in P$$

Dérivations droites et gauches

Définition

Une dérivation droite $\underset{\text{rm}}{\Rightarrow}$ est définie par

$$\varphi A x \underset{\text{rm}}{\Rightarrow} \varphi \alpha x \text{ ssi } A \rightarrow \alpha \in P$$

Définition

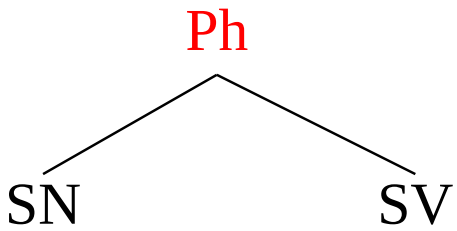
Une dérivation gauche $\underset{\text{lm}}{\Rightarrow}$ est définie par

$$y A \sigma \underset{\text{lm}}{\Rightarrow} y \alpha \sigma \text{ ssi } A \rightarrow \alpha \in P$$

Parcours d'un arbre syntaxique

Exemple

Lors d'une dérivation droite :

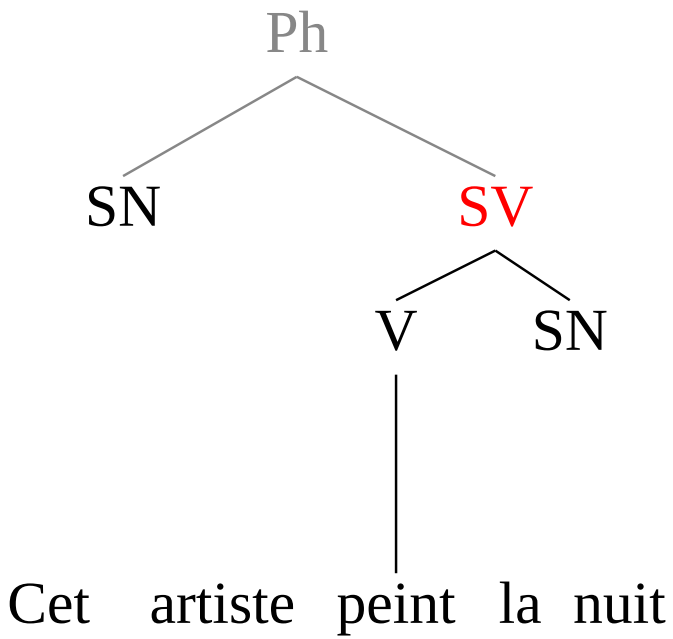


Cet artiste peint la nuit

Parcours d'un arbre syntaxique

Exemple

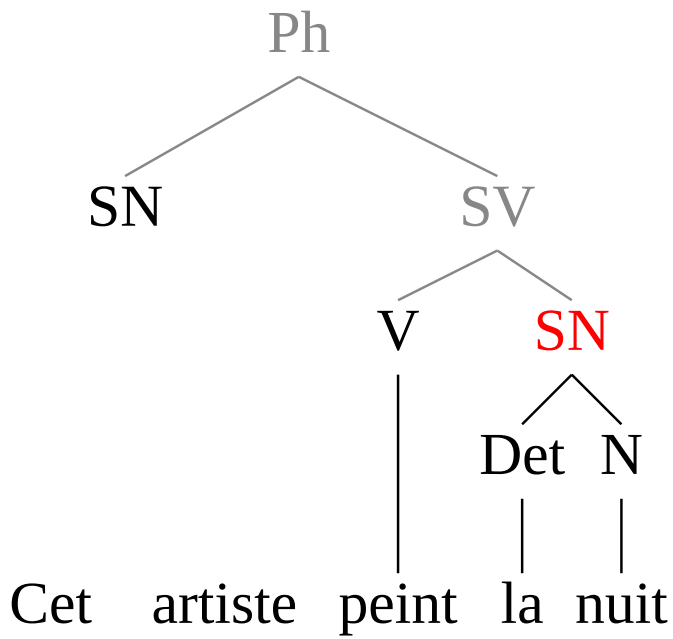
Lors d'une dérivation droite :



Parcours d'un arbre syntaxique

Exemple

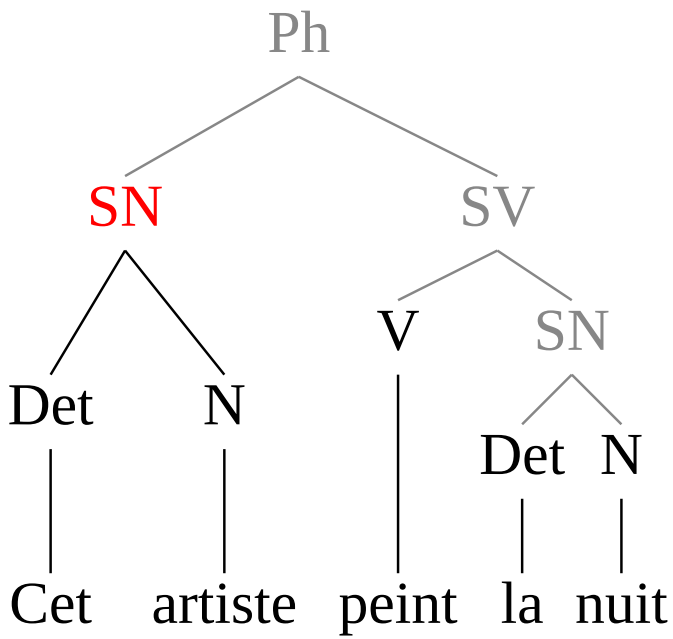
Lors d'une dérivation droite :



Parcours d'un arbre syntaxique

Exemple

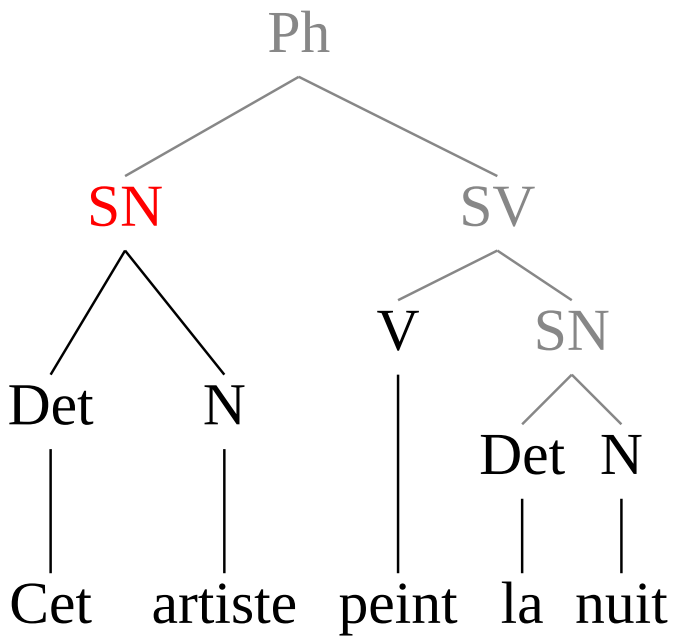
Lors d'une dérivation droite :



Parcours d'un arbre syntaxique

Exemple

Lors d'une dérivation droite :



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

Ambiguité

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

Théorème

Soit $\mathcal{G}$ une grammaire algébrique ; elle est ambigüe 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 ([Par66])

Le langage $\{a^i b^j c^k \mid i = j \text{ ou } j = k\}$ est intrinsèquement ambigu.

Théorème ([Can62, CS63])

Le problème de l'ambiguité d'une grammaire algébrique n'est pas décidable.

Génératif vers recognitif

- Syntaxe formalisée par une grammaire algébrique $\mathcal{G} = \langle N, \Sigma, P, S \rangle$.
- Construction automatique d'un automate à pile $\mathcal{A} = \langle Q, \Sigma, R, \varphi_s, F, \$, \| \rangle$ pour $\mathcal{G}$.

Génératif vers recognitif

- Syntaxe formalisée par une grammaire algébrique $\mathcal{G} = \langle N, \Sigma, P, S \rangle$.
- Construction automatique d'un automate à pile $\mathcal{A} = \langle Q, \Sigma, R, \varphi_s, F, \$, \| \rangle$ pour $\mathcal{G}$.

Génératif vers recognitif

- Syntaxe formalisée par une grammaire algébrique $\mathcal{G} = \langle N, \Sigma, P, S \rangle$.
- Construction automatique d'un automate à pile $\mathcal{A} = \langle Q, \Sigma, R, \varphi_s, F, \$, \| \rangle$ pour $\mathcal{G}$.

Automate à pile (PDA)

Définition

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

$$\varphi \| xy \vdash \psi \| y,$$

- $\varphi_s \in Q^*$ 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.

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

Automate à pile (PDA)

Définition

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

$$\varphi \| xy \vdash \psi \| y,$$

- $\varphi_s \in Q^*$ 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.

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

Automate à pile (PDA)

Définition

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

$$\varphi \| xy \vdash \psi \| y,$$

- $\varphi_s \in Q^*$ 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.

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

Automate à pile (PDA)

Définition

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

$$\varphi \| xy \vdash \psi \| y,$$

- $\varphi_s \in Q^*$ 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.

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

Automate à pile (PDA)

Définition

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

$$\varphi \| xy \vdash \psi \| y,$$

- $\varphi_s \in Q^*$ 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.

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

Automate à pile (PDA)

Définition

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

$$\varphi \| xy \vdash \psi \| y,$$

- $\varphi_s \in Q^*$ 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.

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

Automate à pile (PDA)

Définition

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

$$\varphi \| xy \vdash \psi \| y,$$

- $\varphi_s \in Q^*$ 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.

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

Automate à pile (PDA)

Définition

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

$$\varphi \| xy \vdash \psi \| y,$$

- $\varphi_s \in Q^*$ 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.

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

Analyseur descendant

Définition

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
$$[A \to \alpha \bullet \alpha'] \text{ si } A \to \alpha\alpha' \in P',$$
- R est l'ensembles des règles
$$[A \to \alpha \bullet B\alpha'] \| \vdash_{\text{predict}} [A \to \alpha B \bullet \alpha'][B \to \bullet \beta] \|,$$
$$[A \to \alpha \bullet a\alpha'] \| a \vdash_{\text{match}} [A \to \alpha a \bullet \alpha'] \|,$$
$$[A \to \alpha \bullet] \| \vdash \|,$$
- $\varphi_s = [S' \to \bullet S\$]$,
- $F = \{[S' \to S \bullet \$]\}$.

Analyseur descendant

Définition

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
$$[A \to \alpha \bullet \alpha'] \text{ si } A \to \alpha\alpha' \in P',$$
- R est l'ensembles des règles
$$[A \to \alpha \bullet B\alpha'] \| \vdash_{\text{predict}} [A \to \alpha B \bullet \alpha'][B \to \bullet \beta] \|,$$
$$[A \to \alpha \bullet a\alpha'] \| a \vdash_{\text{match}} [A \to \alpha a \bullet \alpha'] \|,$$
$$[A \to \alpha \bullet] \| \vdash \|,$$
- $\varphi_s = [S' \to \bullet S\$]$,
- $F = \{[S' \to S \bullet \$]\}$.

Analyseur descendant

Définition

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
$$[A \to \alpha \bullet \alpha'] \text{ si } A \to \alpha\alpha' \in P',$$
- R est l'ensembles des règles
$$[A \to \alpha \bullet B\alpha'] \| \vdash_{\text{predict}} [A \to \alpha B \bullet \alpha'][B \to \bullet \beta] \|,$$
$$[A \to \alpha \bullet a\alpha'] \| a \vdash_{\text{match}} [A \to \alpha a \bullet \alpha'] \|,$$
$$[A \to \alpha \bullet] \| \vdash \|,$$
- $\varphi_s = [S' \to \bullet S\$]$,
- $F = \{[S' \to S \bullet \$]\}$.

Analyseur descendant

Définition

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
$$[A \to \alpha \bullet \alpha'] \text{ si } A \to \alpha\alpha' \in P',$$
- R est l'ensembles des règles
$$[A \to \alpha \bullet B\alpha'] \| \vdash_{\text{predict}} [A \to \alpha B \bullet \alpha'][B \to \bullet \beta] \|,$$
$$[A \to \alpha \bullet a\alpha'] \| a \vdash_{\text{match}} [A \to \alpha a \bullet \alpha'] \|,$$
$$[A \to \alpha \bullet] \| \vdash \|,$$
- $\varphi_s = [S' \to \bullet S\$]$,
- $F = \{[S' \to S \bullet \$]\}$.

Analyseur descendant

Définition

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
$$[A \to \alpha \bullet \alpha'] \text{ si } A \to \alpha\alpha' \in P',$$
- R est l'ensembles des règles
$$[A \to \alpha \bullet B\alpha'] \| \vdash_{\text{predict}} [A \to \alpha B \bullet \alpha'][B \to \bullet \beta] \|,$$
$$[A \to \alpha \bullet a\alpha'] \| a \vdash_{\text{match}} [A \to \alpha a \bullet \alpha'] \|,$$
$$[A \to \alpha \bullet] \| \vdash \|,$$
- $\varphi_s = [S' \to \bullet S\$]$,
- $F = \{[S' \to S \bullet \$]\}$.

Analyse descendante

Exemple

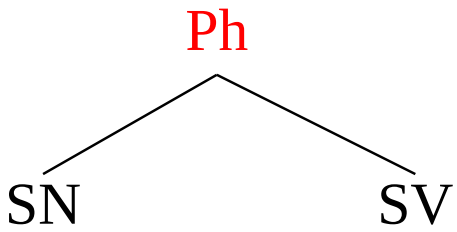
Ph

Cet artiste peint la nuit

$[S' \rightarrow \bullet \text{Ph } \$] \| \text{det N V det N\$}$

Analyse descendante

Exemple

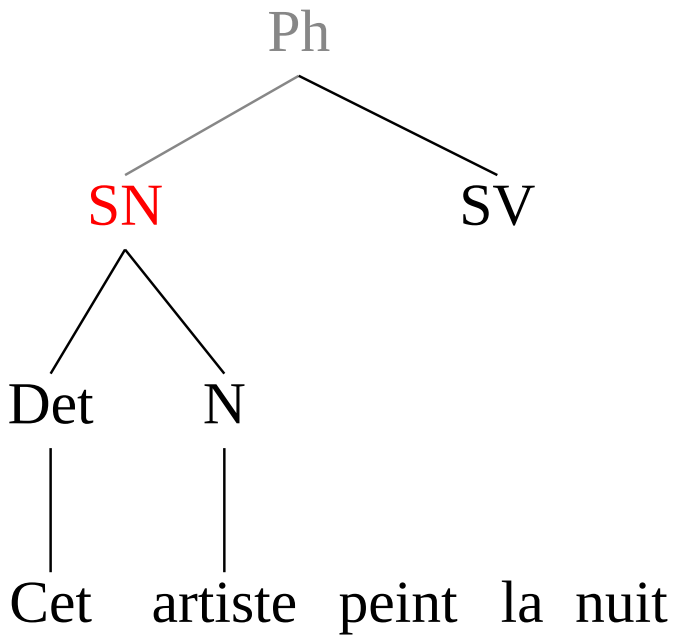


Cet artiste peint la nuit

$$\models_{\text{predict}} \$[S' \rightarrow \text{Ph} \bullet \$][\text{Ph} \rightarrow \bullet \text{SN SV}] \| \text{det N V det N\$}$$

Analyse descendante

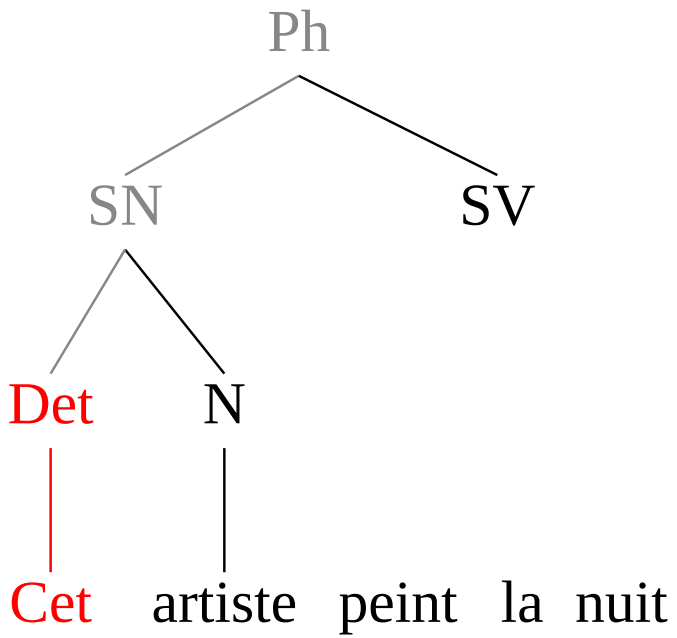
Exemple



$$\models_{\text{predict}} \$[S' \rightarrow \text{Ph} \bullet \$][\text{Ph} \rightarrow \text{SN} \bullet \text{SV}][\text{SN} \rightarrow \bullet \text{det N}] \| \text{det N V det N\$}$$

Analyse descendante

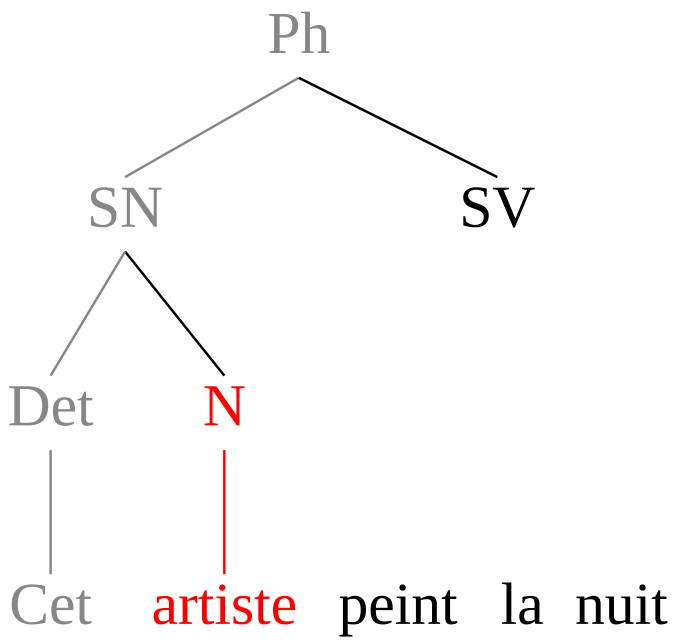
Exemple



$$\models_{\text{match}} \$[S' \rightarrow \text{Ph} \bullet \$][\text{Ph} \rightarrow \text{SN} \bullet \text{SV}][\text{SN} \rightarrow \text{det} \bullet \text{N}] \| \text{N V det N\$}$$

Analyse descendante

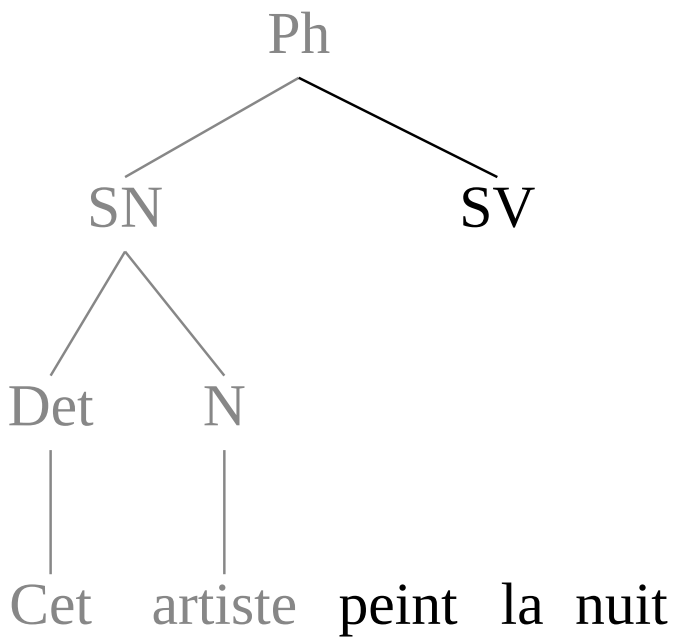
Exemple



$$\models_{\text{match}} \$[S' \rightarrow \text{Ph} \bullet \$][\text{Ph} \rightarrow \text{SN} \bullet \text{SV}][\text{SN} \rightarrow \text{det N} \bullet] \| \text{V det N\$}$$

Analyse descendante

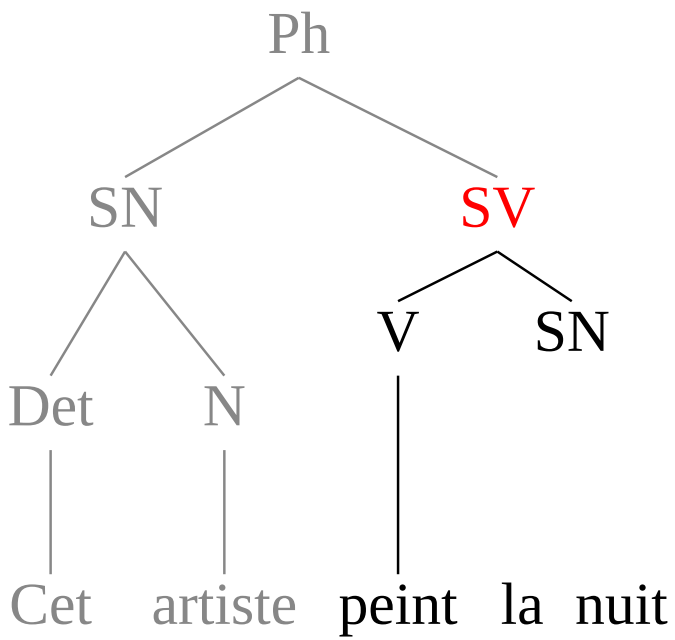
Exemple



$$\models \$[S' \rightarrow \mathrm{Ph} \bullet \$][\mathrm{Ph} \rightarrow \mathrm{SN} \bullet \mathrm{SV}] \| \mathrm{V\ det\ N\$}$$

Analyse descendante

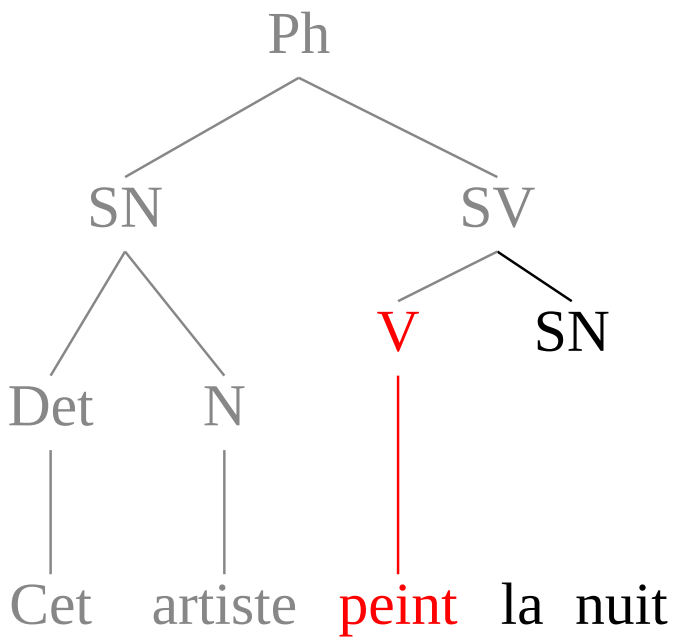
Exemple



$$\models_{\text{predict}} \$[S' \rightarrow \text{Ph} \bullet \$][\text{Ph} \rightarrow \text{SN SV} \bullet][\text{SV} \rightarrow \bullet \text{V SN}] \| \text{V det N\$}$$

Analyse descendante

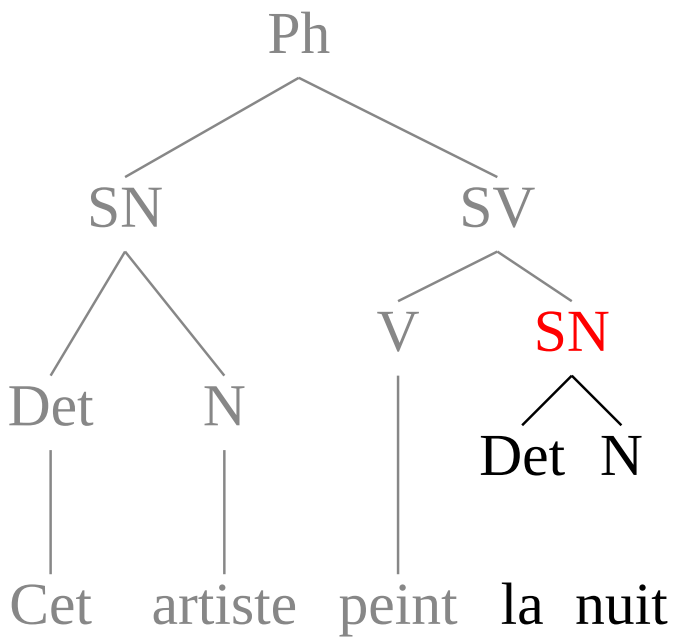
Exemple



$$\models_{\text{match}} \$[S' \rightarrow \text{Ph} \bullet \$][\text{Ph} \rightarrow \text{SN SV} \bullet][\text{SV} \rightarrow \text{V} \bullet \text{SN}] \| \text{det N\$}$$

Analyse descendante

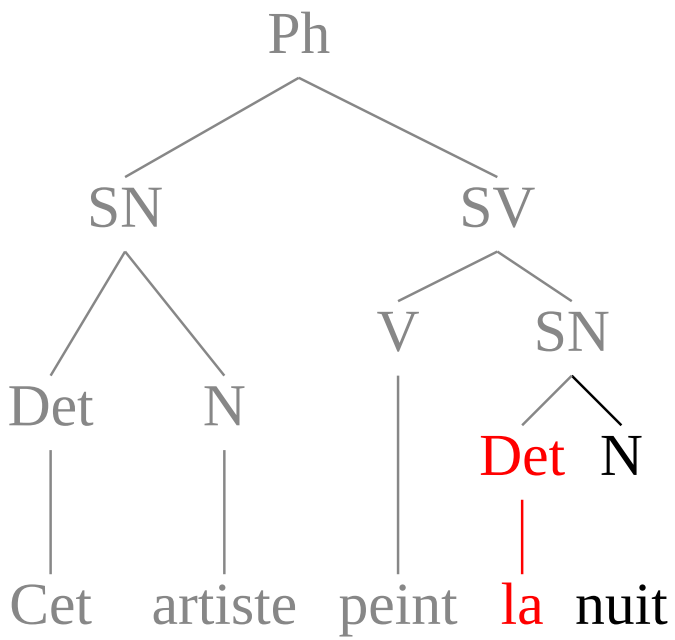
Exemple



$$\models_{\text{predict}} \$[S' \rightarrow \text{Ph} \bullet \$][\text{Ph} \rightarrow \text{SN SV} \bullet][\text{SV} \rightarrow \text{V SN} \bullet][\text{SN} \rightarrow \bullet \text{det N}] \| \text{det N}\$$$

Analyse descendante

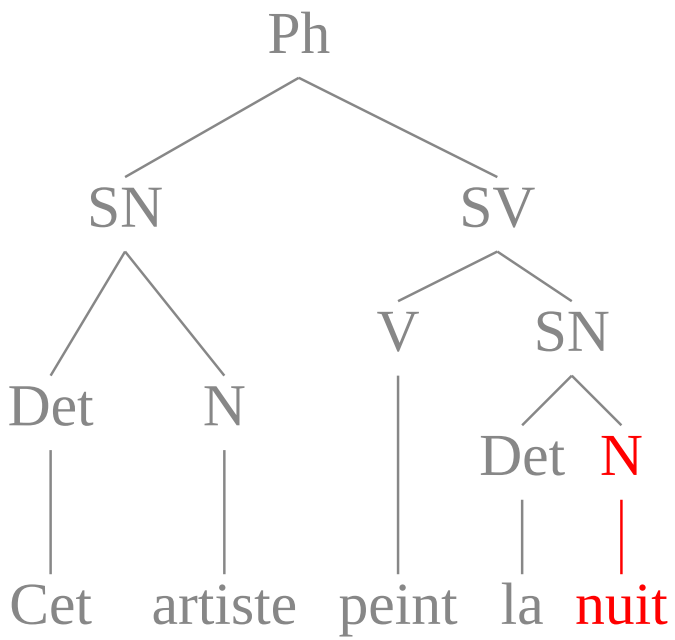
Exemple



$$\models_{\text{match}} \$[S' \rightarrow \text{Ph} \bullet \$][\text{Ph} \rightarrow \text{SN SV} \bullet][\text{SV} \rightarrow \text{V SN} \bullet][\text{SN} \rightarrow \text{det} \bullet \text{N}] \| \text{N}\$$$

Analyse descendante

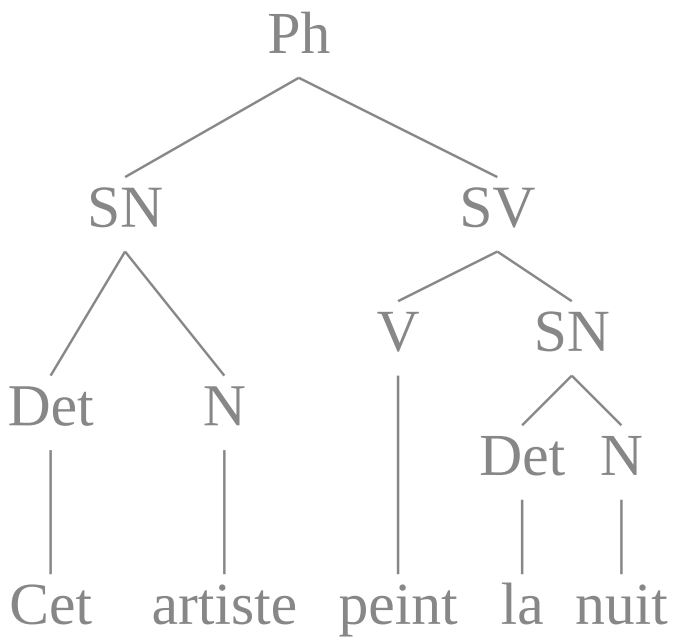
Exemple



$$\models_{\text{match}} \$[S' \rightarrow \text{Ph} \bullet \$][\text{Ph} \rightarrow \text{SN SV} \bullet][\text{SV} \rightarrow \text{V SN} \bullet][\text{SN} \rightarrow \text{det N} \bullet] \| \$$$

Analyse descendante

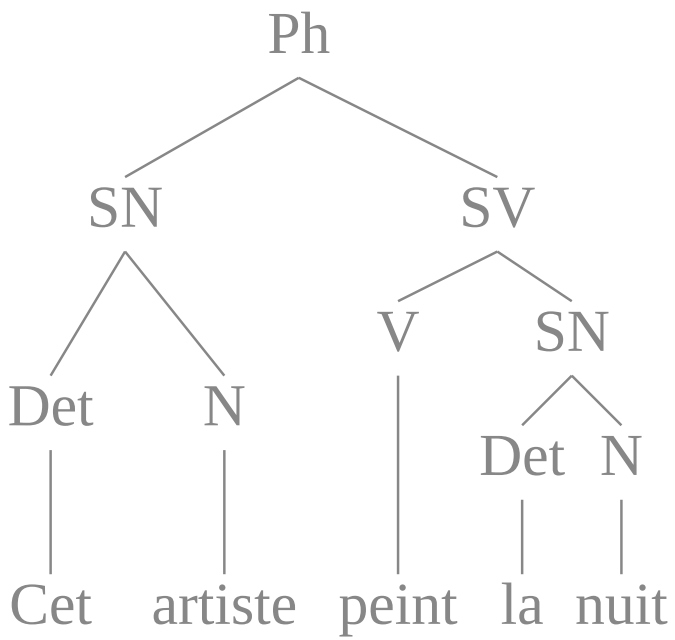
Exemple



$$\models \$[S' \rightarrow \mathrm{Ph} \bullet \$][\mathrm{Ph} \rightarrow \mathrm{SN}\ \mathrm{SV} \bullet][\mathrm{SV} \rightarrow \mathrm{V}\ \mathrm{SN} \bullet] \| \$$$

Analyse descendante

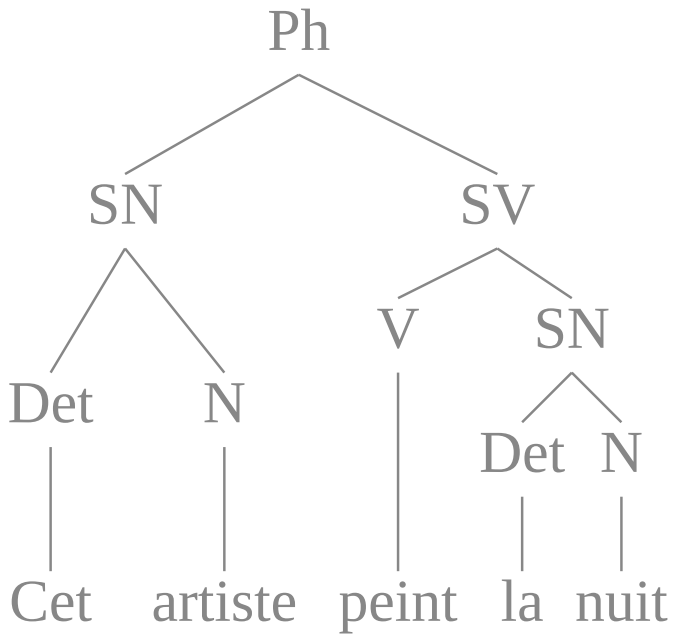
Exemple



$$\models \$[S' \rightarrow \mathrm{Ph} \bullet \$][\mathrm{Ph} \rightarrow \mathrm{SN}\ \mathrm{SV} \bullet] \| \$$$

Analyse descendante

Exemple



$$\models \$[S' \rightarrow \mathsf{Ph} \bullet \$] \| \$$$

Transducteur à pile

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

Définition

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 homomophisme τ de R^* dans P^* défini par

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

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

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

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 $[A \rightarrow \bullet A\alpha] \| \vdash_{\text{predict}} [A \rightarrow A \bullet \alpha][A \rightarrow \bullet A\alpha] \|$ si la grammaire est récursive gauche.

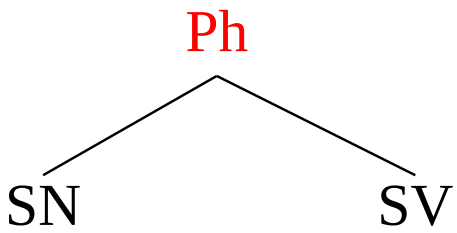
Faiblesses de l'analyse descendante

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 $[A \rightarrow \bullet A\alpha] \| \vdash_{\text{predict}} [A \rightarrow A \bullet \alpha][A \rightarrow \bullet A\alpha] \|$ si la grammaire est récursive gauche.

Dérivations droites

Exemple

Lors d'une dérivation droite :

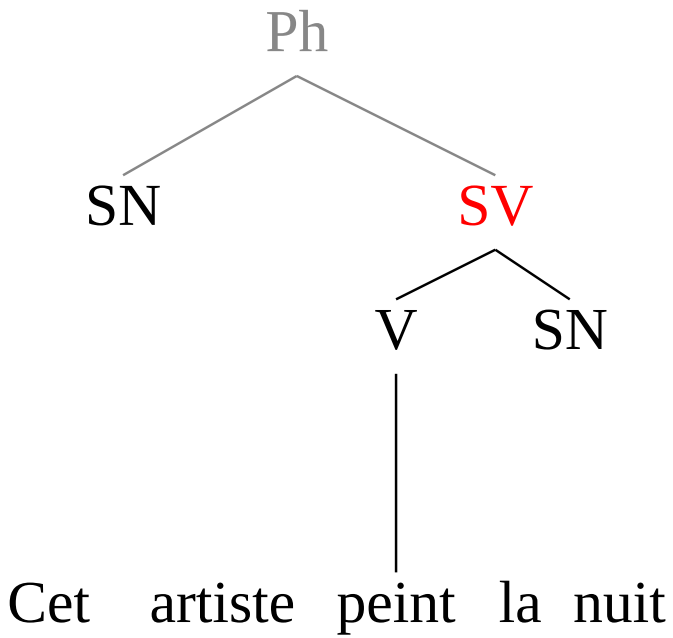


Cet artiste peint la nuit

Dérivations droites

Exemple

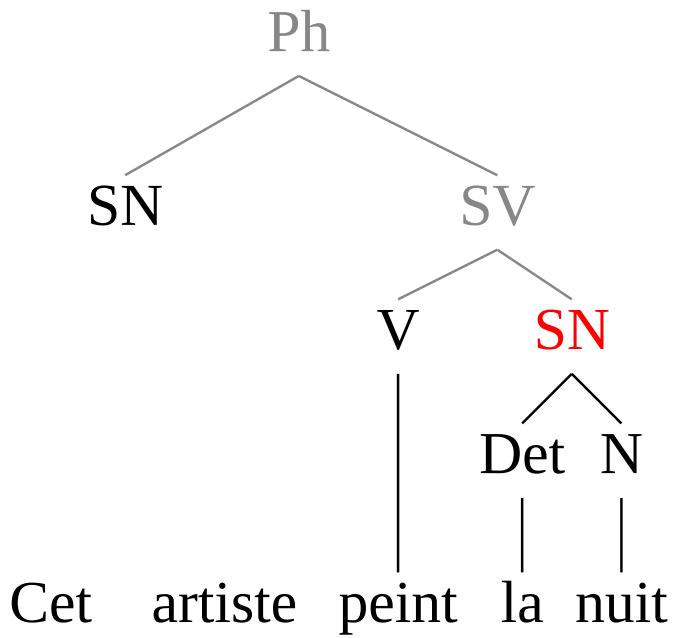
Lors d'une dérivation droite :



Dérivations droites

Exemple

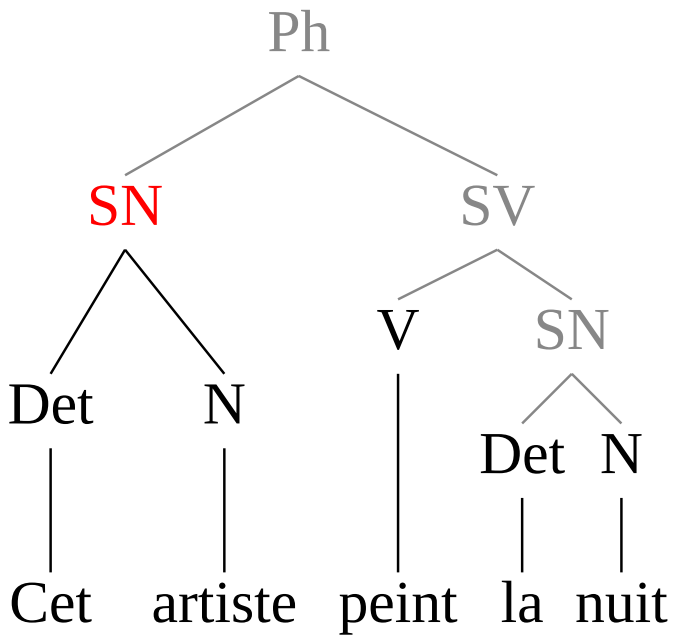
Lors d'une dérivation droite :



Dérivations droites

Exemple

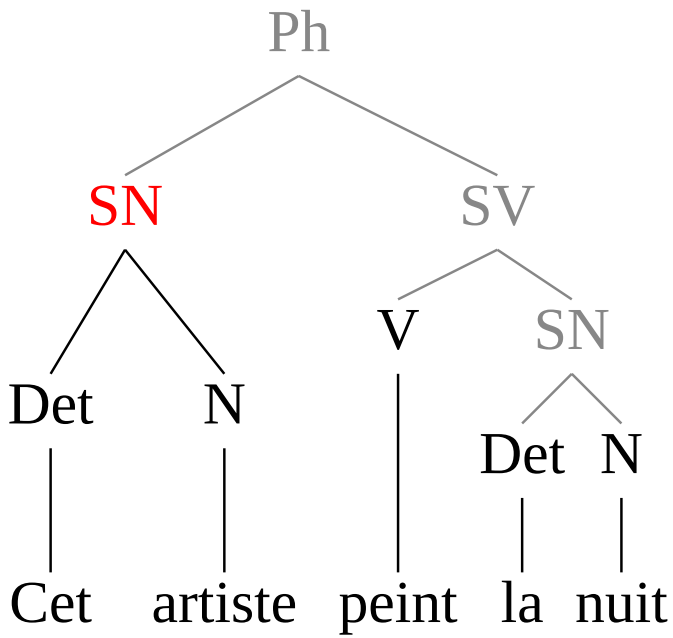
Lors d'une dérivation droite :



Dérivations droites

Exemple

Lors d'une dérivation droite :



Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Dérivations droites

Exemple

Cet artiste peint la nuit

Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Dérivations droites

Exemple

Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

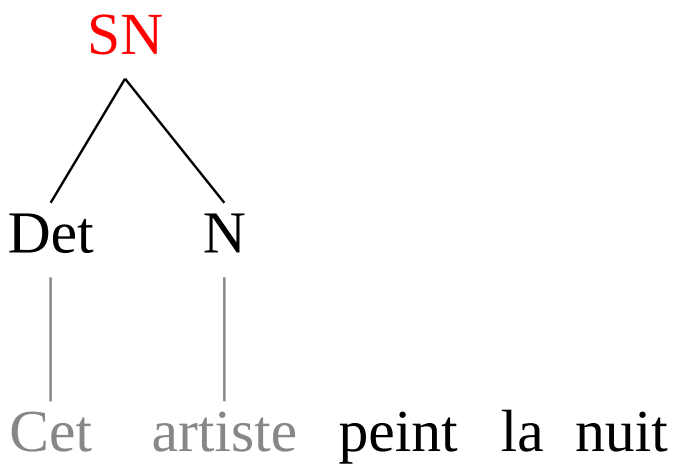
Dérivations droites

Exemple

Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Dérivations droites

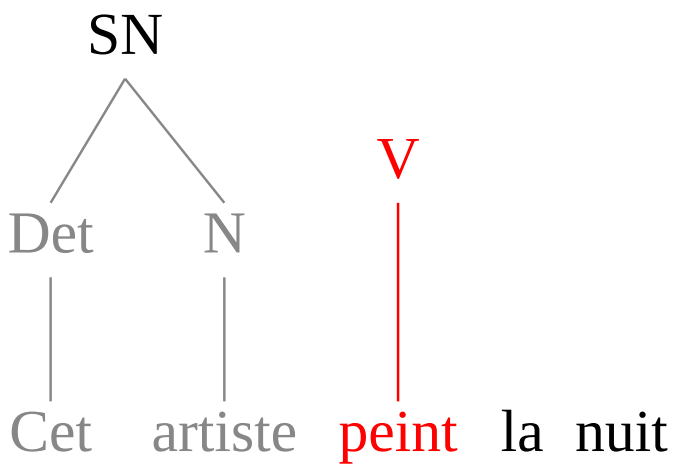
Exemple



Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Dérivations droites

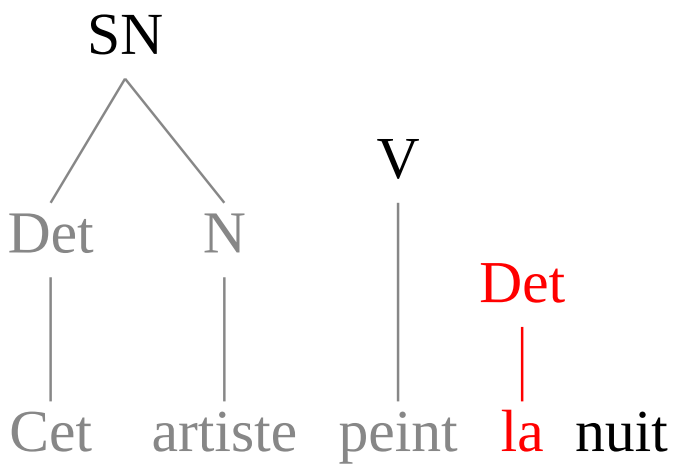
Exemple



Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Dérivations droites

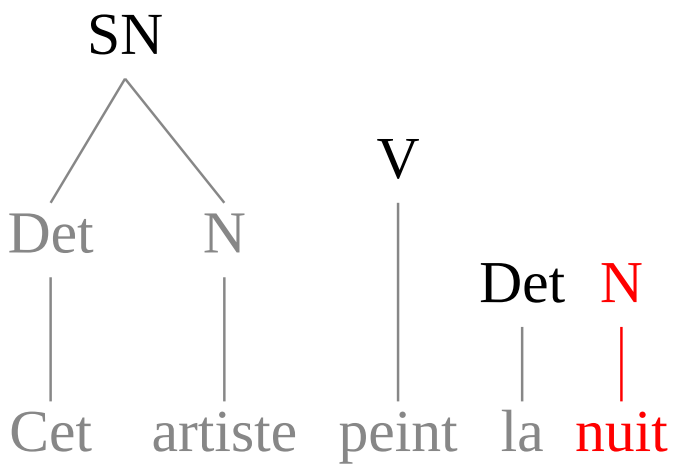
Exemple



Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Dérivations droites

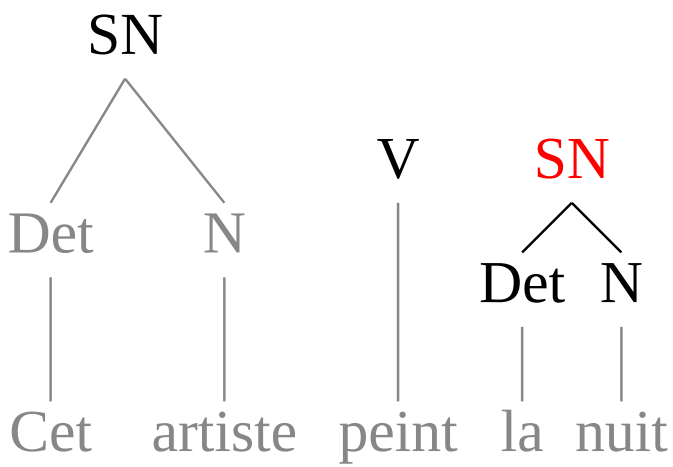
Exemple



Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Dérivations droites

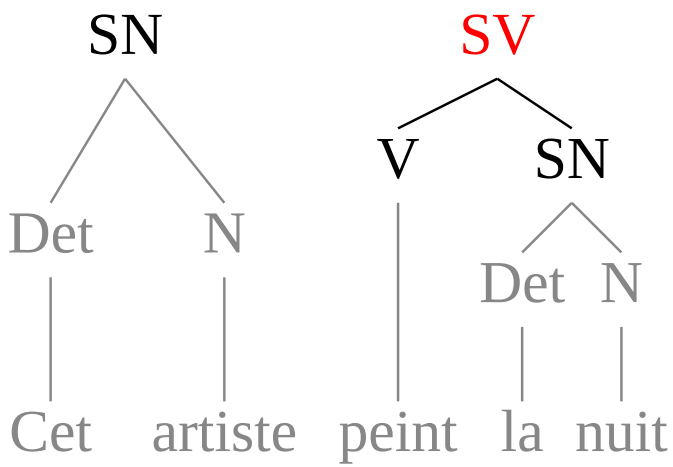
Exemple



Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Dérivations droites

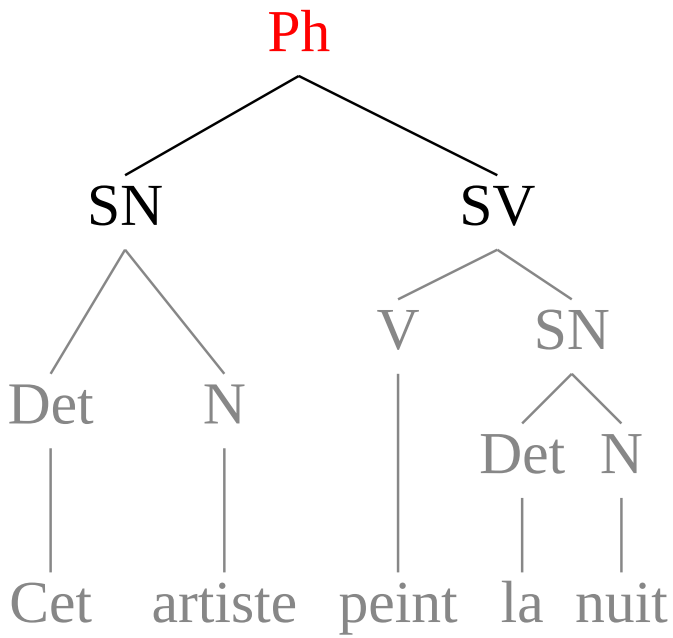
Exemple



Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Dérivations droites

Exemple



Et si on inverse la relation : si on évalue $\underset{\text{rm}}{\Rightarrow}^{-1}$ depuis la chaîne terminale ?

Analyseur ascendant

Définition

L'analyseur ascendant pour une grammaire algébrique $\mathcal{G} = \langle N, \Sigma, P, S \rangle$ est un automate à pile $\mathcal{A}_{\text{asc}} = \langle V, \Sigma, R, \varphi_s, F, \$, \| \rangle$ où

- R est l'ensembles des règles

$$\alpha \| \vdash_{A \to \alpha} A \|,$$
$$\| a \vdash_{\text{shift}} a \|,$$

- $\varphi_s = \varepsilon$,
- $F = \{S\}$.

Analyseur ascendant

Définition

L'analyseur ascendant pour une grammaire algébrique $\mathcal{G} = \langle N, \Sigma, P, S \rangle$ est un automate à pile $\mathcal{A}_{\text{asc}} = \langle V, \Sigma, R, \varphi_s, F, \$, \| \rangle$ où

- R est l'ensembles des règles

$$\alpha \| \vdash_{A \to \alpha} A \|,$$
$$\| a \vdash_{\text{shift}} a \|,$$

- $\varphi_s = \varepsilon$,
- $F = \{S\}$.

Analyseur ascendant

Définition

L'analyseur ascendant pour une grammaire algébrique $\mathcal{G} = \langle N, \Sigma, P, S \rangle$ est un automate à pile $\mathcal{A}_{\text{asc}} = \langle V, \Sigma, R, \varphi_s, F, \$, \| \rangle$ où

- R est l'ensembles des règles

$$\alpha \| \vdash_{A \to \alpha} A \|,$$
$$\| a \vdash_{\text{shift}} a \|,$$

- $\varphi_s = \varepsilon$,
- $F = \{S\}$.

Analyseur ascendant

Définition

L'analyseur ascendant pour une grammaire algébrique $\mathcal{G} = \langle N, \Sigma, P, S \rangle$ est un automate à pile $\mathcal{A}_{\text{asc}} = \langle V, \Sigma, R, \varphi_s, F, \$, \| \rangle$ où

- R est l'ensembles des règles

$$\alpha \| \vdash_{A \to \alpha} A \|,$$
$$\| a \vdash_{\text{shift}} a \|,$$

- $\varphi_s = \varepsilon$,
- $F = \{S\}$.

Analyseur ascendant

Définition

L'analyseur ascendant pour une grammaire algébrique $\mathcal{G} = \langle N, \Sigma, P, S \rangle$ est un automate à pile $\mathcal{A}_{\text{asc}} = \langle V, \Sigma, R, \varphi_s, F, \$, \| \rangle$ où

- R est l'ensembles des règles

$$\alpha \| \vdash_{A \to \alpha} A \|,$$
$$\| a \vdash_{\text{shift}} a \|,$$

- $\varphi_s = \varepsilon$,
- $F = \{S\}$.

Définition

Le transducteur ascendant $\langle \mathcal{A}, \tau \rangle$ est défini par

$$\tau(\alpha \| \vdash_{A \to \alpha} A \|) = A \to \alpha,$$
$$\tau(\| a \vdash_{\text{shift}} a \|) = \varepsilon.$$

Analyse ascendante

Exemple

Cet artiste peint la nuit

$||Det N V Det N$

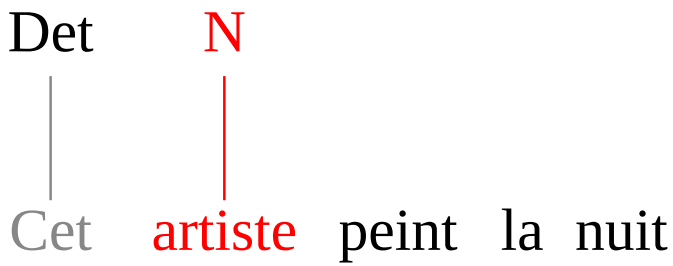
Analyse ascendante

Exemple

$\vdash_{\text{shift}}$ \$Det||N V Det N\$

Analyse ascendante

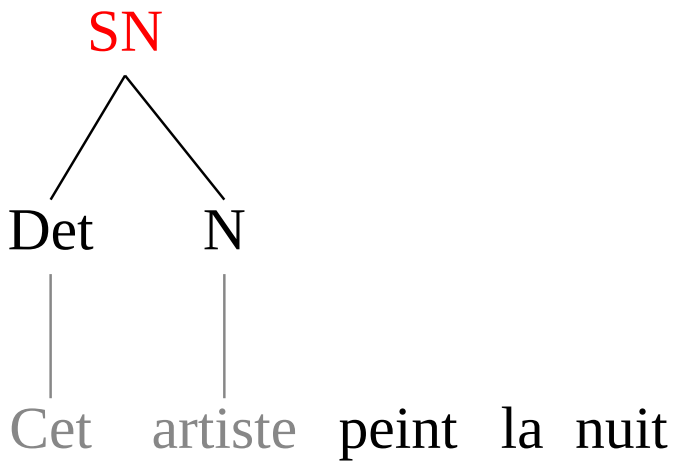
Exemple



$$\vdash_{\text{shift}} \$\text{Det N}\|\text{V Det N}\$$$

Analyse ascendante

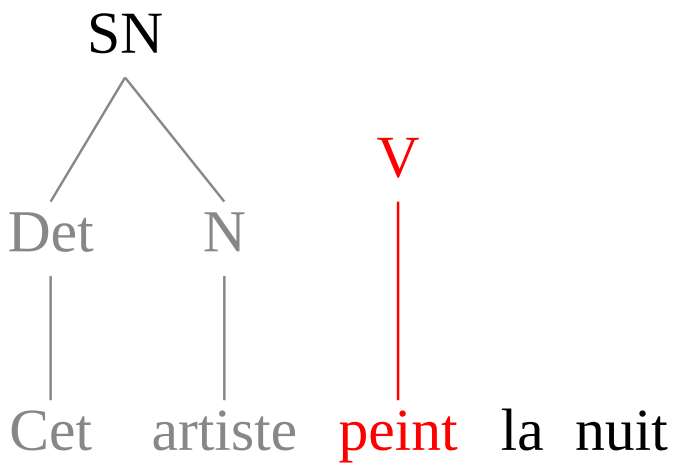
Exemple



$$\vdash_{\overline{SN} \rightarrow Det\ N} \$SN \| V\ Det\ N\$$$

Analyse ascendante

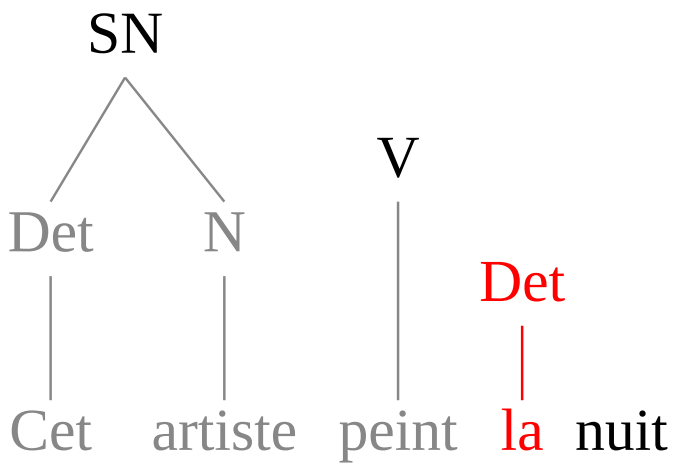
Exemple



$$\models_{\text{shift}} \$SN\ V \| Det\ N\$$$

Analyse ascendante

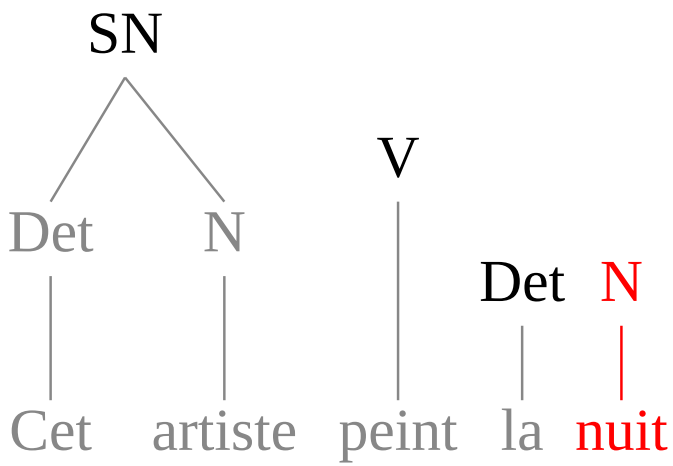
Exemple



$$\models_{\text{shift}} \$SN\ V\ Det \| N\$$$

Analyse ascendante

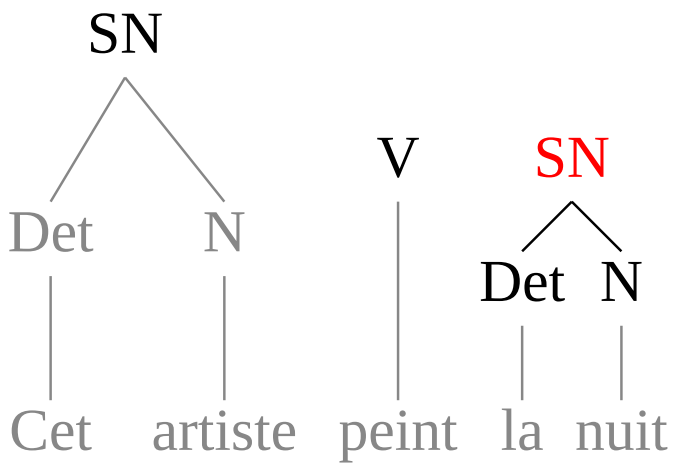
Exemple



$\models_{\text{shift}}$ \$SN V Det N||\$

Analyse ascendante

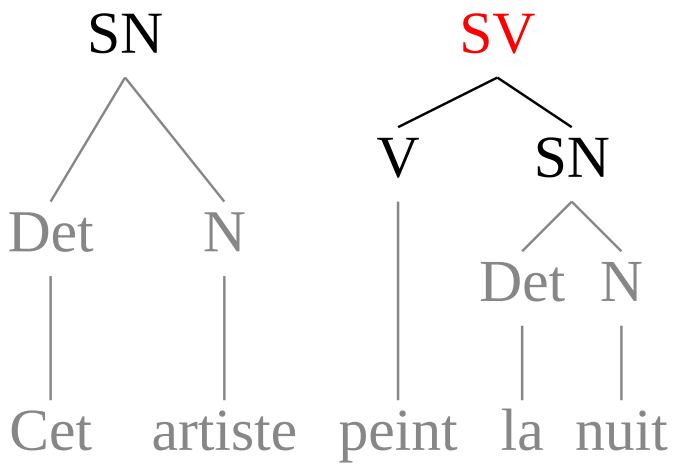
Exemple



$$\models_{\overline{SN} \to Det\ N} \$SN\ V\ SN \| \$$$

Analyse ascendante

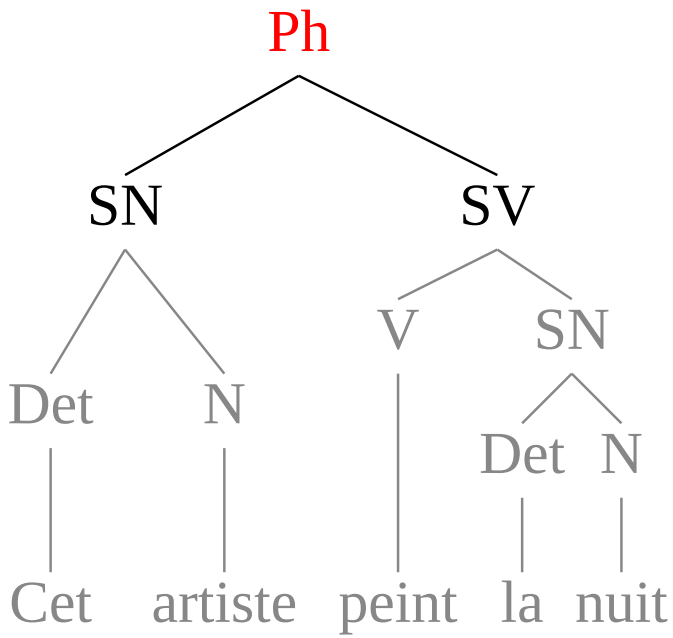
Exemple



$$\models_{\overline{SV} \to V\ SN} \$SN\ SV \| \$$$

Analyse ascendante

Exemple



$$\vdash_{\overline{\mathsf{Ph}} \to \mathsf{SN}\ \mathsf{SV}} \$\mathsf{Ph}\|\$$$

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.

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.

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.

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.


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.


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


Noam Chomsky.
Three models for the description of language.
IEEE Transactions on Information Theory, 2(3):113–124, 1956.


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


Noam Chomsky and Marcel Paul Schützenberger.
The algebraic theory of context-free languages.
In P. Braffort and D. Hirshberg, editors, *Computer Programming and Formal Systems*, Studies in Logic, pages 118–161. North Holland