Théorie des Langages

Cours 9 : Analyse LL(1)

Lionel Rieg

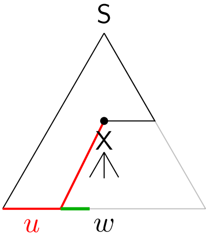
Grenoble INP - Ensimag, 1re année

Analyse LL(k)

Cadre : grammaire HC non ambiguë
$\rightsquigarrow$ un seul arbre possible, comment le trouver ?

Left-to-right reading : on lit de gauche à droite
Leftmost derivation : on construit une dérivation gauche
k look-ahead : on regarde les k prochains terminaux

Analyse descendante : on passe de S à w



$u \in V_T^*$
Quelle règle utiliser pour transformer X ?
On décide à partir des k prochains terminaux.
Condition : chaque règle depuis un non-terminal X correspond à un ensemble de k terminaux disjoints des autres

Regardons plus particulièrement le cas k = 1 : analyse LL(1)

Analyse LL(1) (cas facile)

Condition : pour un non-terminal donné X, chaque règle depuis X doit correspondre à des terminaux différents.

Cas facile : les règles depuis X commencent toutes par un terminal différent

Exemple (Expressions arithmétiques préfixées)

$V_T = \{\mathbf{num}, +, -, *, /\}$ et $V_N = \{\mathsf{exp}\}$

$$\mathsf{exp} \rightarrow + \,\mathsf{exp}\,\mathsf{exp} \mid - \,\mathsf{exp}\,\mathsf{exp} \mid * \,\mathsf{exp}\,\mathsf{exp} \mid / \,\mathsf{exp}\,\mathsf{exp} \mid \mathbf{num}$$

- Si le prochain caractère est **num**, utiliser la règle $\mathsf{exp} \rightarrow \mathbf{num}$
- Si le prochain caractère est $+$, utiliser la règle $\mathsf{exp} \rightarrow + \,\mathsf{exp}\,\mathsf{exp}$
- ...

OK si même terminal pour deux règles depuis des non-terminaux différents

Construction de l'analyseur (cas facile)

Une fonction d'analyse pour chaque non-terminal

```
def parse_exp():
  if current == num:
    consume_token(num)
    return
  if current == +:
    consume_token(+)
    parse_exp()
    parse_exp()
    return
  if current == *:
    consume_token(*)
    ...
```

Cas général : pour chaque non-terminal X

- une fonction d'analyse `parse_X`
- un cas pour chaque règle
  - un test sur le prochain terminal
  - des actions pour la partie droite
    - pour chaque terminal a, `consume_token(`a`)`
    - pour chaque non-terminal Y, `parse_Y()`

Le prochain terminal se trouve dans la variable globale `current`.
La fonction `parse_X` consomme tout le sous-arbre enraciné en X.
La fonction `consume_token(`a`)` vérifie que le prochain terminal est bien a : si oui, passe au terminal suivant ; si non, lève une erreur.

Exemple d'analyse

Exemple :

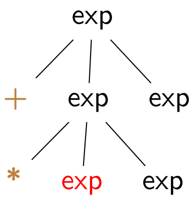
$$\text{exp} \;\rightarrow\; +\,\text{exp exp} \;\mid\; -\,\text{exp exp} \;\mid\; *\,\text{exp exp} \;\mid\; /\,\text{exp exp} \;\mid\; \textbf{num}$$

Mot à analyser : + * **num - num num num**

Appels imbriqués

```
parse_exp()
  consume_token(+)
  parse_exp()
    consume_token(*)
    parse_exp()
```

Arbre



Exemple d'analyse

Exemple :

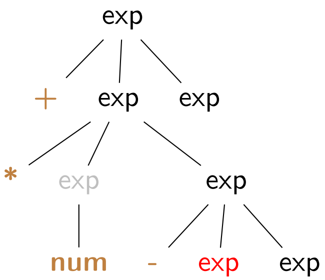
$$\text{exp} \;\rightarrow\; +\,\text{exp}\,\text{exp} \;\mid\; -\,\text{exp}\,\text{exp} \;\mid\; *\,\text{exp}\,\text{exp} \;\mid\; /\,\text{exp}\,\text{exp} \;\mid\; \textbf{num}$$

Mot à analyser : + * **num** - **num** **num** **num**

Appels imbriqués

```
parse_exp()
  consume_token(+)
  parse_exp()
    consume_token(*)
    parse_exp()
      consume_token(num)
    parse_exp()
      consume_token(-)
      parse_exp()
```

Arbre



Exemple d'analyse

Exemple :

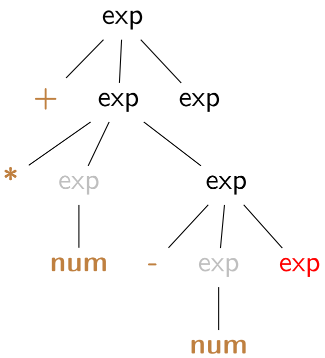
$$\mathsf{exp} \;\rightarrow\; +\,\mathsf{exp}\,\mathsf{exp} \;\mid\; -\,\mathsf{exp}\,\mathsf{exp} \;\mid\; *\,\mathsf{exp}\,\mathsf{exp} \;\mid\; /\,\mathsf{exp}\,\mathsf{exp} \;\mid\; \mathbf{num}$$

Mot à analyser : + * **num - num num num**

Appels imbriqués

```
parse_exp()
  consume_token(+)
  parse_exp()
    consume_token(*)
    parse_exp()
      consume_token(num)
    parse_exp()
      consume_token(-)
      parse_exp()
        consume_token(num)
      parse_exp()
```

Arbre



Exemple d'analyse

Exemple :

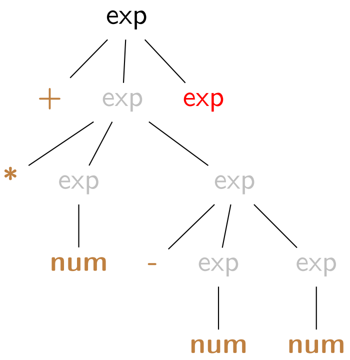
$$\mathsf{exp} \;\rightarrow\; +\,\mathsf{exp}\,\mathsf{exp} \;\mid\; -\,\mathsf{exp}\,\mathsf{exp} \;\mid\; *\,\mathsf{exp}\,\mathsf{exp} \;\mid\; /\,\mathsf{exp}\,\mathsf{exp} \;\mid\; \mathbf{num}$$

Mot à analyser : + * **num** - **num num num**

Appels imbriqués

```
parse_exp()
  consume_token(+)
  parse_exp()
    consume_token(*)
    parse_exp()
      consume_token(num)
    parse_exp()
      consume_token(-)
      parse_exp()
        consume_token(num)
      parse_exp()
        consume_token(num)
  parse_exp()
```

Arbre



Exemple d'analyse

Exemple :

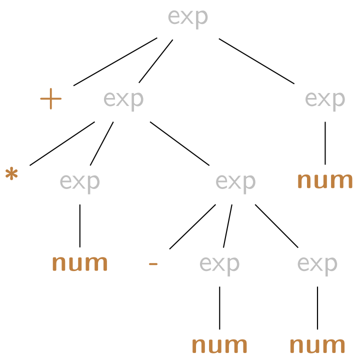
$$\text{exp} \rightarrow + \,\text{exp}\,\text{exp} \;\mid\; - \,\text{exp}\,\text{exp} \;\mid\; * \,\text{exp}\,\text{exp} \;\mid\; / \,\text{exp}\,\text{exp} \;\mid\; \mathbf{num}$$

Mot à analyser : **+ * num - num num num**

Appels imbriqués

```
parse_exp()
  consume_token(+)
  parse_exp()
    consume_token(*)
    parse_exp()
      consume_token(num)
    parse_exp()
      consume_token(-)
      parse_exp()
        consume_token(num)
      parse_exp()
        consume_token(num)
  parse_exp()
    consume_token(num)
```

Arbre



Analyse LL(1) (cas général)

Si le premier symbole d'une règle X $\rightarrow$ α n'est pas un terminal, comment choisir la règle ? Où se trouve le prochain symbole terminal ?
Dans le sous-arbre enraciné en X.
On recherche le premier symbole des mots engendrés par la règle X $\rightarrow$ α.

$$\mathrm{Prem}(\alpha) \stackrel{\mathsf{def}}{=} \{x \in V_T \mid \exists w \in V^*, \alpha \Longrightarrow^* xw\}$$

Cas facile : si α commence par un terminal $\mathbf{a}$, alors $\mathrm{Prem}(\alpha) = \{\mathbf{a}\}$

Comment calculer $\mathrm{Prem}(\alpha)$ en général ?

Calcul de $\mathrm{Prem}(\alpha)$

cf. exercice 7 du recueil de TD

Une grammaire HC vérifie un système d'équations avec $\cup$ et . :

$$\begin{array}{lcl} \mathsf{X} \rightarrow u_1 \mid \ldots \mid u_k & & \mathcal{L}(\mathsf{X}) = \mathcal{L}(u_1) \cup \cdots \cup \mathcal{L}(u_k) \\ \vdots & \rightsquigarrow & \vdots \\ \mathsf{Y} \rightarrow v_1 \mid \ldots \mid v_n & & \mathcal{L}(\mathsf{Y}) = \mathcal{L}(v_1) \cup \cdots \cup \mathcal{L}(v_n) \end{array}$$

donc son langage est un plus petit point fixe.

Pour calculer Prem sur ces langages, lemme de commutation

Point fixe et lemme de commutation

Intuition des points fixes : Pour une fonction croissante f,
on cherche $\mu(f)$ le plus petit ensemble X tel que $f(X) = X$.

- On commence avec $\emptyset$: si $f(\emptyset) = \emptyset$, c'est gagné.
- Sinon, $\emptyset \subsetneq f(\emptyset)$ et on réessaie avec $f(\emptyset)$, puis $f^2(\emptyset)$, puis $f^3(\emptyset)$, etc.
- Si on n'atteint jamais l'égalité (donc $\emptyset \subsetneq f(\emptyset) \subsetneq f^2(\emptyset) \subsetneq \dots$), alors $\bigcup_{i \in \mathbb{N}} f^i(\emptyset)$ convient. (mais il faut la continuité de f)

Lemme (Lemme de commutation, rappel)

Pour $k \in \{1, 2\}$, soit f_k applications continues de $\mathcal{P}(E_k) \to \mathcal{P}(E_k)$
Soit g application continue de $\mathcal{P}(E_1) \to \mathcal{P}(E_2)$ avec

$$g(\emptyset) = \emptyset \qquad \text{et} \qquad g \circ f_1 = f_2 \circ g$$

Alors, on a :

$$\mu(f_2) = \bigcup_{i \in \mathbb{N}} f_2^i(\emptyset) = g(\bigcup_{i \in \mathbb{N}} f_1^i(\emptyset)) = g(\mu(f_1))$$

Lemme de commutation pour le calcul de Prem

Lemme (Lemme de commutation, rappel)

Pour $k \in \{1,2\}$, soit f_k applications continues de $\mathcal{P}(E_k) \to \mathcal{P}(E_k)$
Soit g application continue de $\mathcal{P}(E_1) \to \mathcal{P}(E_2)$ avec

$$g(\emptyset) = \emptyset \qquad \textit{et} \qquad g \circ f_1 = f_2 \circ g$$

Alors, on a :

$$\mu(f_2) = \bigcup_{i \in \mathbb{N}} f_2^i(\emptyset) = g(\bigcup_{i \in \mathbb{N}} f_1^i(\emptyset)) = g(\mu(f_1))$$

Ici, on a :

- f_1 équations du système
$$\begin{aligned} \mathcal{L}(\mathsf{X}) &= \mathcal{L}(u_1) \cup \cdots \cup \mathcal{L}(u_k) \\ &\vdots \\ \mathcal{L}(\mathsf{Y}) &= \mathcal{L}(v_1) \cup \cdots \cup \mathcal{L}(v_n) \end{aligned}$$
- $g(X, \ldots, Y) = (\mathrm{Prem}(X), \ldots, \mathrm{Prem}(Y))$
- f_2 exprime $\mathrm{Prem}(u_1 \mid \ldots \mid u_k), \ldots, \mathrm{Prem}(v_1 \mid \ldots \mid v_n)$ en fonction des $\mathrm{Prem}(X), \ldots, \mathrm{Prem}(Y)$

Calcul de $\mathrm{Prem}(\alpha)$

cf. exercice 7 du recueil de TD

Une grammaire HC vérifie un système d'équations avec $\cup$ et . :

$$\begin{array}{lll} \mathsf{X} \to u_1 \mid \ldots \mid u_k & & \mathcal{L}(\mathsf{X}) = \mathcal{L}(u_1) \cup \cdots \cup \mathcal{L}(u_k) \\ \vdots & \rightsquigarrow & \vdots \\ \mathsf{Y} \to v_1 \mid \ldots \mid v_n & & \mathcal{L}(\mathsf{Y}) = \mathcal{L}(v_1) \cup \cdots \cup \mathcal{L}(v_n) \end{array}$$

donc son langage est un plus petit point fixe.

Pour calculer Prem sur ces langages, lemme de commutation

Équations simplifiantes :

$$\begin{array}{ll} \mathrm{Prem}(\mathbf{a}) & = \{\mathbf{a}\} \\ \mathrm{Prem}(\varepsilon) & = \emptyset \\ \mathrm{Prem}(\alpha \mid \beta) & = \mathrm{Prem}(\alpha) \cup \mathrm{Prem}(\beta) \\ \mathrm{Prem}(\alpha.\beta) & = \begin{cases} \mathrm{Prem}(\alpha) & \text{si } \varepsilon \notin \mathcal{L}(\alpha) \\ \mathrm{Prem}(\alpha) \cup \mathrm{Prem}(\beta) & \text{si } \varepsilon \in \mathcal{L}(\alpha) \end{cases} \\ & = \mathrm{Prem}(\alpha) \cup \varepsilon(\alpha).\mathrm{Prem}(\beta) \end{array}$$

Calcul de $\varepsilon(\alpha)$

$$\varepsilon(\alpha) \stackrel{\mathsf{def}}{=} \mathcal{L}(\alpha) \cap \{\varepsilon\}$$

Même méthode : lemme de commutation

Équations simplifiantes :

$$\begin{array}{ll} \varepsilon(\mathbf{a}) & = \emptyset \\ \varepsilon(\varepsilon) & = \{\varepsilon\} \\ \varepsilon(\alpha \mid \beta) & = \varepsilon(\alpha) \cup \varepsilon(\beta) \\ \varepsilon(\alpha.\beta) & = \varepsilon(\alpha) \cap \varepsilon(\beta) \end{array}$$

Analyse LL(1) (cas général)

Si le premier symbole d'une règle X $\rightarrow$ α n'est pas un terminal, comment choisir la règle ? Où se trouve le prochain symbole terminal ?
Dans le sous-arbre enraciné en X.
On recherche le premier symbole des mots engendrés par la règle X $\rightarrow$ α.

$$\mathrm{Prem}(\alpha) \stackrel{\text{def}}{=} \{x \in V_T \mid \exists w \in V^*, \alpha \Longrightarrow^* xw\}$$

Cas facile : si α commence par un terminal $\mathbf{a}$, alors $\mathrm{Prem}(\alpha) = \{\mathbf{a}\}$

Comment calculer $\mathrm{Prem}(\alpha)$ en général ?
On calcule ε puis Prem avec le lemme de commutation.

Exemple

Calculer ε et Prem pour la grammaire HC suivante :

$$
\begin{array}{lcl}
\mathsf{S} & \rightarrow & \mathsf{K}\,\mathsf{P}\,\mathsf{L} \\
\mathsf{K} & \rightarrow & \mathsf{P}\,\mathsf{L} \mid \mathsf{P}\,\mathbf{a}\,\mathsf{L} \mid \mathsf{K}\,\mathsf{L}\,\mathsf{P} \\
\mathsf{P} & \rightarrow & \mathsf{L}\,\mathbf{b} \mid \mathsf{K}\,\mathsf{L} \\
\mathsf{L} & \rightarrow & \mathsf{L}\,\mathbf{c}\,\mathsf{P}\,\mathsf{K} \mid \varepsilon
\end{array}
$$

Calcul de ε

$$
\begin{array}{l}
\varepsilon(\mathsf{S}) = \varepsilon(\mathsf{K}) \cap \varepsilon(\mathsf{P}) \cap \varepsilon(\mathsf{L}) \\
\varepsilon(\mathsf{K}) = \varepsilon(\mathsf{P}) \cap \varepsilon(\mathsf{L}) \quad \cup \quad \varepsilon(\mathsf{P}) \cap \varepsilon(\mathbf{a}) \cap \varepsilon(\mathsf{L}) \quad \cup \quad \varepsilon(\mathsf{K}) \cap \varepsilon(\mathsf{L}) \cap \varepsilon(\mathsf{P}) \\
\varepsilon(\mathsf{P}) = \varepsilon(\mathsf{L}) \cap \varepsilon(\mathbf{b}) \quad \cup \quad \varepsilon(\mathsf{K}) \cap \varepsilon(\mathsf{L}) \\
\varepsilon(\mathsf{L}) = \varepsilon(\mathsf{L}) \cap \varepsilon(\mathbf{c}) \cap \varepsilon(\mathsf{P}) \cap \varepsilon(\mathsf{K}) \quad \cup \quad \varepsilon(\varepsilon)
\end{array}
$$

Exemple

Calculer ε et Prem pour la grammaire HC suivante :

$$\begin{array}{lcl}
\mathsf{S} & \rightarrow & \mathsf{K\,P\,L} \\
\mathsf{K} & \rightarrow & \mathsf{P\,L} \mid \mathsf{P}\,\mathbf{a}\,\mathsf{L} \mid \mathsf{K\,L\,P} \\
\mathsf{P} & \rightarrow & \mathsf{L}\,\mathbf{b} \mid \mathsf{K\,L} \\
\mathsf{L} & \rightarrow & \mathsf{L}\,\mathbf{c}\,\mathsf{P\,K} \mid \varepsilon
\end{array}
\qquad
\begin{array}{l}
\varepsilon(\mathsf{S}) = \emptyset \\
\varepsilon(\mathsf{K}) = \emptyset \\
\varepsilon(\mathsf{P}) = \emptyset \\
\varepsilon(\mathsf{L}) = \{\varepsilon\}
\end{array}$$

Calcul de ε

$$\begin{array}{ll}
\varepsilon(\mathsf{S}) = \varepsilon(\mathsf{K}) \cap \varepsilon(\mathsf{P}) \cap \varepsilon(\mathsf{L}) & \\
\varepsilon(\mathsf{K}) = \varepsilon(\mathsf{P}) \cap \varepsilon(\mathsf{L}) & \cup \quad \varepsilon(\mathsf{K}) \cap \varepsilon(\mathsf{L}) \cap \varepsilon(\mathsf{P}) \\
\varepsilon(\mathsf{P}) = \qquad \varepsilon(\mathsf{K}) \cap \varepsilon(\mathsf{L}) & \\
\varepsilon(\mathsf{L}) = \qquad \{\varepsilon\} &
\end{array}$$

Itérations :

	$\varepsilon(\mathsf{S})$	$\varepsilon(\mathsf{K})$	$\varepsilon(\mathsf{P})$	$\varepsilon(\mathsf{L})$
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\emptyset$	$\{\varepsilon\}$
2	$\emptyset$	$\emptyset$	$\emptyset$	$\{\varepsilon\}$

Exemple

Calculer ε et Prem pour la grammaire HC suivante :

$\mathsf{S} \rightarrow \mathsf{K\,P\,L}$	$\varepsilon(\mathsf{S}) = \emptyset$	$\mathrm{Prem}(\mathsf{S}) = \{\mathbf{b}, \mathbf{c}\}$
$\mathsf{K} \rightarrow \mathsf{P\,L} \mid \mathsf{P}\,\mathbf{a}\,\mathsf{L} \mid \mathsf{K\,L\,P}$	$\varepsilon(\mathsf{K}) = \emptyset$	$\mathrm{Prem}(\mathsf{K}) = \{\mathbf{b}, \mathbf{c}\}$
$\mathsf{P} \rightarrow \mathsf{L}\,\mathbf{b} \mid \mathsf{K\,L}$	$\varepsilon(\mathsf{P}) = \emptyset$	$\mathrm{Prem}(\mathsf{P}) = \{\mathbf{b}, \mathbf{c}\}$
$\mathsf{L} \rightarrow \mathsf{L}\,\mathbf{c}\,\mathsf{P\,K} \mid \varepsilon$	$\varepsilon(\mathsf{L}) = \{\varepsilon\}$	$\mathrm{Prem}(\mathsf{L}) = \{\mathbf{c}\}$

Calcul de Prem

$$\mathrm{Prem}(\mathsf{S}) = \mathrm{Prem}(\mathsf{K}) \cup \varepsilon(\mathsf{K}).\mathrm{Prem}(\mathsf{P\,L}) = \mathrm{Prem}(\mathsf{K})$$
$$\mathrm{Prem}(\mathsf{K}) = \mathrm{Prem}(\mathsf{P}) \cup \mathrm{Prem}(\mathsf{K})$$
$$\mathrm{Prem}(\mathsf{P}) = \mathrm{Prem}(\mathsf{L}) \cup \{\mathbf{b}\} \cup \mathrm{Prem}(\mathsf{K})$$
$$\mathrm{Prem}(\mathsf{L}) = \mathrm{Prem}(\mathsf{L}) \cup \mathrm{Prem}(\mathbf{c}\,\mathsf{P\,K}) \cup \mathrm{Prem}(\varepsilon) = \mathrm{Prem}(\mathsf{L}) \cup \{\mathbf{c}\}$$

Itérations :

	$\mathrm{Prem}(\mathsf{S})$	$\mathrm{Prem}(\mathsf{K})$	$\mathrm{Prem}(\mathsf{P})$	$\mathrm{Prem}(\mathsf{L})$
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\{\mathbf{b}\}$	$\{\mathbf{c}\}$
2	$\emptyset$	$\{\mathbf{b}\}$	$\{\mathbf{b}, \mathbf{c}\}$	$\{\mathbf{c}\}$
3	$\{\mathbf{b}\}$	$\{\mathbf{b}, \mathbf{c}\}$	$\{\mathbf{b}, \mathbf{c}\}$	$\{\mathbf{c}\}$
5, 4	$\{\mathbf{b}, \mathbf{c}\}$	$\{\mathbf{b}, \mathbf{c}\}$	$\{\mathbf{b}, \mathbf{c}\}$	$\{\mathbf{c}\}$

Exemple

Calculer ε et Prem pour la grammaire HC suivante :

$$
\begin{array}{lcl}
\mathsf{S} & \rightarrow & \mathsf{K\,P\,L} \\
\mathsf{K} & \rightarrow & \mathsf{P\,L} \mid \mathsf{P}\,\mathbf{a}\,\mathsf{L} \mid \mathsf{K\,L\,P} \\
\mathsf{P} & \rightarrow & \mathsf{L}\,\mathbf{b} \mid \mathsf{K\,L} \\
\mathsf{L} & \rightarrow & \mathsf{L}\,\mathbf{c}\,\mathsf{P\,K} \mid \varepsilon
\end{array}
\qquad
\begin{array}{l}
\varepsilon(\mathsf{S}) = \emptyset \\
\varepsilon(\mathsf{K}) = \emptyset \\
\varepsilon(\mathsf{P}) = \emptyset \\
\varepsilon(\mathsf{L}) = \{\varepsilon\}
\end{array}
\qquad
\begin{array}{l}
\mathrm{Prem}(\mathsf{S}) = \{\mathbf{b}, \mathbf{c}\} \\
\mathrm{Prem}(\mathsf{K}) = \{\mathbf{b}, \mathbf{c}\} \\
\mathrm{Prem}(\mathsf{P}) = \{\mathbf{b}, \mathbf{c}\} \\
\mathrm{Prem}(\mathsf{L}) = \{\mathbf{c}\}
\end{array}
$$

Code de l'analyseur

```python
def parse_K():
  if current in ['b','c']:
    parse_P() # K -> PL
    parse_L()
    return
  if current in ['b','c']:
    parse_P() # K -> PaL
    consume_token('a')
    parse_L()
  ...
```

$$\mathrm{Prem}(\mathsf{P\,L}) = \mathrm{Prem}(\mathsf{P}) = \{\mathbf{b}, \mathbf{c}\}$$
$$\mathrm{Prem}(\mathsf{P}\,\mathbf{a}\,\mathsf{L}) = \mathrm{Prem}(\mathsf{P}) = \{\mathbf{b}, \mathbf{c}\}$$
$$\mathrm{Prem}(\mathsf{K\,L\,P}) = \mathrm{Prem}(\mathsf{K}) = \{\mathbf{b}, \mathbf{c}\}$$

Problème pour choisir entre

- $\mathsf{K} \rightarrow \mathsf{P\,L}$
- $\mathsf{K} \rightarrow \mathsf{P}\,\mathbf{a}\,\mathsf{L}$
- $\mathsf{K} \rightarrow \mathsf{K\,L\,P}$

Cette grammaire n'est pas LL(1) !

Exemple qui fonctionne

$$\mathsf{S} \rightarrow \mathbf{aSb} \mid \mathbf{c}$$

$\varepsilon(\mathsf{S}) = \big(\varepsilon(\mathbf{a}) \cap \varepsilon(\mathsf{S}) \cap \varepsilon(\mathbf{b})\big) \cup \varepsilon(\mathbf{c}) \; = \; \emptyset$
$\mathrm{Prem}(\mathbf{aSb}) = \{\mathbf{a}\}$
$\mathrm{Prem}(\mathbf{c}) = \{\mathbf{c}\}$

```
def parse_S():
  if current in ['a']: # S -> aSb
    consume_token('a')
    parse_S()
    consume_token('b')
    return
  if current in ['c']: # S -> c
    consume_token('c')
    return
```

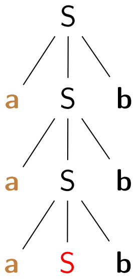
✓

Exemple qui fonctionne

$$\mathsf{S} \;\rightarrow\; \mathbf{aSb} \;\mid\; \varepsilon$$

$\varepsilon(\mathsf{S}) = \big(\varepsilon(\mathbf{a}) \cap \varepsilon(\mathsf{S}) \cap \varepsilon(\mathbf{b})\big) \cup \varepsilon(\varepsilon) \;=\; \{\varepsilon\}$
$\mathrm{Prem}(\mathbf{aSb}) = \{\mathbf{a}\}$
$\mathrm{Prem}(\varepsilon) = \emptyset$

```
def parse_S():
  if current in ['a']: # S -> aSb
    consume_token('a')
    parse_S()
    consume_token('b')
    return
  if current in []: # S -> eps
    return
```



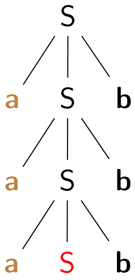
✗

Exemple qui fonctionne

$$S \rightarrow \mathbf{aSb} \mid \varepsilon$$

$\varepsilon(S) = \big(\varepsilon(\mathbf{a}) \cap \varepsilon(S) \cap \varepsilon(\mathbf{b})\big) \cup \varepsilon(\varepsilon) \;=\; \{\varepsilon\}$
$\mathrm{Prem}(\mathbf{aSb}) = \{\mathbf{a}\}$
$\mathrm{Prem}(\varepsilon) = \emptyset$

```
def parse_S():
  if current in ['a']: # S -> aSb
    consume_token('a')
    parse_S()
    consume_token('b')
    return
  if current in ['b']: # S -> eps
    return
```



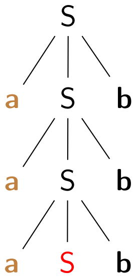
✓ **b** suit la règle $S \rightarrow \varepsilon$

Exemple qui fonctionne

$$S \rightarrow \mathbf{aSb} \mid \varepsilon$$

$\varepsilon(S) = \big(\varepsilon(\mathbf{a}) \cap \varepsilon(S) \cap \varepsilon(\mathbf{b})\big) \cup \varepsilon(\varepsilon) \; = \; \{\varepsilon\}$
$\mathrm{Prem}(\mathbf{aSb}) = \{\mathbf{a}\}$
$\mathrm{Prem}(\varepsilon) = \emptyset$

```
def parse_S():
  if current in ['a']: # S -> aSb
    consume_token('a')
    parse_S()
    consume_token('b')
    return
  if current in ['b', $]: # S -> eps
    return
```



S

✓

b et $ suivent la règle $S \rightarrow \varepsilon$

Analyse LL(1) (cas général)

Si le premier symbole d'une règle X $\rightarrow$ α n'est pas un terminal, comment choisir la règle ? Où se trouve le prochain symbole terminal ?
Dans le sous-arbre enraciné en X.
On recherche le premier symbole des mots engendrés par la règle X $\rightarrow$ α.

$$\mathrm{Prem}(\alpha) \stackrel{\mathsf{def}}{=} \{x \in V_T \mid \exists w \in V^*, \alpha \Longrightarrow^* xw\}$$

Cas facile : si α commence par un terminal $\mathbf{a}$, alors $\mathrm{Prem}(\alpha) = \{\mathbf{a}\}$

Comment calculer $\mathrm{Prem}(\alpha)$ en général ?
On calcule ε puis Prem avec le lemme de commutation.

Si $\alpha \Longrightarrow^* \varepsilon$, il faut aussi regarder ce qui suit la règle X $\rightarrow$ α.
$\rightsquigarrow$ Calcul de $\mathrm{Suiv}(\mathsf{X})$

Expression de Suiv(X)

$$\mathrm{Suiv}(\mathsf{X}) \stackrel{\text{def}}{=} \{\text{le premier terminal qui suit un sous-arbre enraciné en X}\}$$
$$\stackrel{\text{def}}{=} \{\mathrm{Prem}(\beta) \mid \mathsf{S}.\$ \Longrightarrow^* \alpha\mathsf{X}\beta\}$$

$\$$ = sentinelle de fin d'entrée (nouveau terminal)

- Si $\mathsf{S}.\$ \Longrightarrow^0 \alpha\mathsf{X}\beta$, alors on a $\alpha = \varepsilon$, $\mathsf{X} = \mathsf{S}$ et $\beta = \$$.
  Si X est l'axiome, $\mathrm{Suiv}(\mathsf{X})$ contient $\mathrm{Prem}(\$) = \{\$\}$.
- Sinon, X est dans la partie droite d'une règle $\mathsf{Y} \rightarrow u\mathsf{X}v$.
  Alors $\mathsf{S}.\$ \Longrightarrow^* \alpha Y\beta \Longrightarrow \alpha uXv\beta$
  donc $\mathrm{Suiv}(\mathsf{X})$ contient $\mathrm{Prem}(v\beta) = \mathrm{Prem}(v) \cup \varepsilon(v).\mathrm{Suiv}(\mathsf{Y})$
  Pour toute règle $\mathsf{Y} \rightarrow u\mathsf{X}v$, $\mathrm{Suiv}(\mathsf{X})$ contient $\mathrm{Prem}(v) \cup \varepsilon(v).\mathrm{Suiv}(\mathsf{Y})$

Expression de Suiv

$$\mathrm{Suiv}(\mathsf{X}) = \bigcup_{\mathsf{Y}\rightarrow u\mathsf{X}v} \mathrm{Prem}(v) \cup \varepsilon(v).\mathrm{Suiv}(\mathsf{Y})$$
$$\cup \{\$\} \quad \text{si X est l'axiome}$$

⚠ Chaque occurrence de X compte (Exemple : $\mathsf{Y} \rightarrow \mathbf{a}\mathsf{X}\mathbf{a}\mathsf{X}$ compte 2 fois)

Cela donne un système d'équations $\rightsquigarrow$ résolution par itération de point fixe

Analyse LL(1) (cas général)

Si le premier symbole d'une règle X $\rightarrow$ α n'est pas un terminal, comment choisir la règle ? Où se trouve le prochain symbole terminal ?
Dans le sous-arbre enraciné en X.
On recherche le premier symbole des mots engendrés par la règle X $\rightarrow$ α.

$$\mathrm{Prem}(\alpha) \stackrel{\text{def}}{=} \{x \in V_T \mid \exists w \in V^*, \alpha \Longrightarrow^* xw\}$$

Cas facile : si α commence par un terminal $\mathbf{a}$, alors $\mathrm{Prem}(\alpha) = \{\mathbf{a}\}$

Comment calculer $\mathrm{Prem}(\alpha)$ en général ?
On calcule ε puis Prem avec le lemme de commutation.

Si $\alpha \Longrightarrow^* \varepsilon$, il faut aussi regarder ce qui suit la règle X $\rightarrow$ α.
$\rightsquigarrow$ Calcul de $\mathrm{Suiv}(\mathsf{X})$ par itérations de point fixe

Cas général : le premier symbole est dans le directeur de X $\rightarrow$ α :

$$\mathrm{Dir}(\mathsf{X} \rightarrow \alpha) \stackrel{\text{def}}{=} \mathrm{Prem}(\alpha) \cup \varepsilon(\alpha).\mathrm{Suiv}(\mathsf{X})$$

Calcul des directeurs en analyse LL(1)

Pour chaque non-terminal X :

1. Calcul de $\varepsilon(\mathsf{X})$

$$\varepsilon(\mathbf{a}) = \emptyset \qquad \varepsilon(\alpha\,.\,\beta) = \varepsilon(\alpha) \cap \varepsilon(\beta)$$
$$\varepsilon(\varepsilon) = \{\varepsilon\} \qquad \varepsilon(\alpha \mid \beta) = \varepsilon(\alpha) \cup \varepsilon(\beta)$$

2. Calcul de $\mathrm{Prem}(\mathsf{X})$

$$\mathrm{Prem}(\mathbf{a}) = \{\mathbf{a}\} \qquad \mathrm{Prem}(\alpha.\beta) = \mathrm{Prem}(\alpha) \cup \varepsilon(\alpha).\mathrm{Prem}(\beta)$$
$$\mathrm{Prem}(\varepsilon) = \emptyset \qquad \mathrm{Prem}(\alpha \mid \beta) = \mathrm{Prem}(\alpha) \cup \mathrm{Prem}(\beta)$$

3. Calcul de $\mathrm{Suiv}(\mathsf{X})$

$$\mathrm{Suiv}(\mathsf{X}) = \bigcup_{\mathsf{Y} \to u\mathsf{X}v} \mathrm{Prem}(v) \cup \varepsilon(v).\mathrm{Suiv}(\mathsf{Y})$$
$$\cup\, \{\$\} \quad \text{si X est l'axiome}$$

Les trois par itération de point fixe

4. Calcul de $\mathrm{Dir}(\mathsf{X} \ \to \ \alpha)$ pour chaque règle $\mathsf{X} \ \to \ \alpha$

$$\mathrm{Dir}(\mathsf{X} \ \to \ \alpha) \stackrel{\text{def}}{=} \mathrm{Prem}(\alpha) \cup \varepsilon(\alpha).\mathrm{Suiv}(\mathsf{X})$$

Exercice : Parmi $\varepsilon(\mathsf{X})$, $\mathrm{Prem}(\mathsf{X})$, $\mathrm{Suiv}(\mathsf{X})$, $\mathrm{Dir}(\mathsf{X} \to \alpha)$, lesquels peuvent être vides ?

Exercice

Calculer les directeurs de la grammaire suivante.

$$\begin{aligned} \mathsf{S} &\to \mathsf{X} \mid \mathsf{Y}\mathbf{c} \\ \mathsf{X} &\to \mathbf{a}\mathsf{X}\mathbf{b} \mid \varepsilon \\ \mathsf{Y} &\to \mathbf{b}\mathsf{Y} \mid \varepsilon \end{aligned}$$

Calcul de ε

$$\begin{aligned} \varepsilon(\mathsf{S}) &= \varepsilon(\mathsf{X}) \cup \emptyset = \{\varepsilon\} \\ \varepsilon(\mathsf{X}) &= \emptyset \cup \{\varepsilon\} = \{\varepsilon\} \\ \varepsilon(\mathsf{Y}) &= \emptyset \cup \{\varepsilon\} = \{\varepsilon\} \end{aligned}$$

Calcul de Prem

$$\begin{aligned} \mathrm{Prem}(\mathsf{S}) &= \mathrm{Prem}(\mathsf{X}) \cup \mathrm{Prem}(\mathsf{Y}) \cup \{\mathbf{c}\} \\ &= \{\mathbf{a}, \mathbf{b}, \mathbf{c}\} \\ \mathrm{Prem}(\mathsf{X}) &= \{\mathbf{a}\} \cup \emptyset = \{\mathbf{a}\} \\ \mathrm{Prem}(\mathsf{Y}) &= \{\mathbf{b}\} \cup \emptyset = \{\mathbf{b}\} \end{aligned}$$

Calcul de Suiv

$$\begin{aligned} \mathrm{Suiv}(\mathsf{S}) &= \emptyset \cup \{\$\} &&= \{\$\} \\ \mathrm{Suiv}(\mathsf{X}) &= \mathrm{Suiv}(\mathsf{S}) \cup \{\mathbf{b}\} \cup \emptyset &&= \{\mathbf{b}, \$\} \\ \mathrm{Suiv}(\mathsf{Y}) &= \{\mathbf{c}\} \cup \mathrm{Suiv}(\mathsf{Y}) \cup \emptyset &&= \{\mathbf{c}\} \end{aligned}$$

Calcul de Dir

$$\begin{aligned} \mathrm{Dir}(\mathsf{S} \to \mathsf{X}) &= \{\mathbf{a}\} \cup \{\$\} \\ \mathrm{Dir}(\mathsf{S} \to \mathsf{Y}\mathbf{c}) &= \{\mathbf{b}\} \cup \{\mathbf{c}\} \\ \mathrm{Dir}(\mathsf{X} \to \mathbf{a}\mathsf{X}\mathbf{b}) &= \{\mathbf{a}\} \\ \mathrm{Dir}(\mathsf{X} \to \varepsilon) &= \{\mathbf{b}, \$\} \\ \mathrm{Dir}(\mathsf{Y} \to \mathbf{b}\mathsf{Y}) &= \{\mathbf{b}\} \\ \mathrm{Dir}(\mathsf{Y} \to \varepsilon) &= \{\mathbf{c}\} \end{aligned}$$

Grammaire LL(1)

Définition (Grammaire LL(1))

Une grammaire est dite LL(1) si
tous les directeurs des règles depuis un même non terminal sont disjoints

$$\text{Pour tout } X \to \alpha, X \to \beta \text{ avec } \alpha \neq \beta,\ \mathrm{Dir}(X \to \alpha) \cap \mathrm{Dir}(X \to \beta) = \emptyset$$

Idée : Pour chaque X, aucun terminal ne correspond à plusieurs règles.

Construction de l'analyseur : comme dans le cas facile en utilisant Dir

```
def parse_X():
  if current in Dir(X → α):
    parse_Y()
    consume_token(...)
    ⋮
```

Exercice

Calculer les directeurs de la grammaire suivante. Est-elle LL(1) ?

$$\begin{aligned} \mathsf{S} &\to \mathsf{X} \mid \mathsf{Y}\mathbf{c} \\ \mathsf{X} &\to \mathbf{a}\mathsf{X}\mathbf{b} \mid \varepsilon \\ \mathsf{Y} &\to \mathbf{b}\mathsf{Y} \mid \varepsilon \end{aligned}$$

Calcul de ε

$$\begin{aligned} \varepsilon(\mathsf{S}) &= \varepsilon(\mathsf{X}) \cup \emptyset = \{\varepsilon\} \\ \varepsilon(\mathsf{X}) &= \emptyset \cup \{\varepsilon\} = \{\varepsilon\} \\ \varepsilon(\mathsf{Y}) &= \emptyset \cup \{\varepsilon\} = \{\varepsilon\} \end{aligned}$$

Calcul de Prem

$$\begin{aligned} \mathrm{Prem}(\mathsf{S}) &= \mathrm{Prem}(\mathsf{X}) \cup \mathrm{Prem}(\mathsf{Y}) \cup \{\mathbf{c}\} \\ &= \{\mathbf{a}, \mathbf{b}, \mathbf{c}\} \\ \mathrm{Prem}(\mathsf{X}) &= \{\mathbf{a}\} \cup \emptyset = \{\mathbf{a}\} \\ \mathrm{Prem}(\mathsf{Y}) &= \{\mathbf{b}\} \cup \emptyset = \{\mathbf{b}\} \end{aligned}$$

La grammaire est bien LL(1).

Calcul de Suiv

$$\begin{aligned} \mathrm{Suiv}(\mathsf{S}) &= \emptyset \cup \{\$\} &&= \{\$\} \\ \mathrm{Suiv}(\mathsf{X}) &= \mathrm{Suiv}(\mathsf{S}) \cup \{\mathbf{b}\} \cup \emptyset &&= \{\mathbf{b}, \$\} \\ \mathrm{Suiv}(\mathsf{Y}) &= \{\mathbf{c}\} \cup \mathrm{Suiv}(\mathsf{Y}) \cup \emptyset &&= \{\mathbf{c}\} \end{aligned}$$

Calcul de Dir

$$\left.\begin{aligned} \mathrm{Dir}(\mathsf{S} \to \mathsf{X}) &= \{\mathbf{a}\} \cup \{\$\} \\ \mathrm{Dir}(\mathsf{S} \to \mathsf{Y}\mathbf{c}) &= \{\mathbf{b}\} \cup \{\mathbf{c}\} \end{aligned}\right\} \checkmark$$

$$\left.\begin{aligned} \mathrm{Dir}(\mathsf{X} \to \mathbf{a}\mathsf{X}\mathbf{b}) &= \{\mathbf{a}\} \\ \mathrm{Dir}(\mathsf{X} \to \varepsilon) &= \{\mathbf{b}, \$\} \end{aligned}\right\} \checkmark$$

$$\left.\begin{aligned} \mathrm{Dir}(\mathsf{Y} \to \mathbf{b}\mathsf{Y}) &= \{\mathbf{b}\} \\ \mathrm{Dir}(\mathsf{Y} \to \varepsilon) &= \{\mathbf{c}\} \end{aligned}\right\} \checkmark$$

Quel message d'erreur ?

Pour la grammaire $S \rightarrow \mathbf{a}\,S\,\mathbf{b} \mid \varepsilon$,
on a $\mathrm{Dir}(S \rightarrow \mathbf{a}\,S\,\mathbf{b}) = \{\mathbf{a}\}$ et $\mathrm{Dir}(S \rightarrow \varepsilon) = \{\mathbf{b}, \$\}$.

Trois choix d'implémentation corrects :

```
def parse_S():
  if current in [a]:
    consume_token(a)
    parse_S()
    consume_token(b)
    return
  else:
    return
```

```
def parse_S():
  if current in [b,$]:
    return
  else:
    consume_token(a)
    parse_S()
    consume_token(b)
    return
```

```
def parse_S():
  if current in [a]:
    consume_token(a)
    parse_S()
    consume_token(b)
    return
  elif current in [b,$]:
    return
  else:
    raise Error("a b $")
```

Messages d'erreur sur **ac** :

Attend un **b** par `consume_token(b)`

Attend un **a** par `consume_token(a)`

Attend un **a**, **b** ou $ par le **else**

Ceci explique que parfois l'erreur n'est pas indiquée au bon endroit.

Gestion des erreurs en analyse LL(1)

Actuellement, si erreur, toute l'analyse échoue.
Comment faire du rattrapage d'erreurs ?

Cause d'erreurs : oubli/ajout d'un terminal (remplacement = les 2)

Principes :

- Ne pas remettre en cause ce qui a déjà été analysé
- Essayer de valider le reste malgré tout
- Ignorer jusqu'à un « mot rattrapant » puis reprendre l'analyse
  $\leadsto$ mots rattrapants à définir, contiennent au moins $\mathrm{Suiv}(X)$

Algorithme :

```
def gestion_erreur(X):
  while current not in mots-rattrapant(X):
    consume_token(current)
  return
```

On reprend ensuite l'analyse normalement.

Mise en forme LL(1)

Mise en forme LL(1)

Comment transformer une grammaire non LL(1) en grammaire LL(1) ?

Pas toujours possible, on utilise des heuristiques

Pré-requis : grammaire non-ambiguë

$\leadsto$ Au besoin, niveaux de priorité pour lever les ambiguïtés

Heuristiques :

- Préfixe commun $\mathsf{X} \rightarrow uv \mid uw$

  Problème : on fait le choix entre les règles trop tôt

  Solution : on factorise pour repousser le choix à plus tard

  $$\mathsf{X} \rightarrow u\mathsf{X}' \qquad \mathsf{X}' \rightarrow v \mid w$$

- Récursivité à gauche $\mathsf{X} \rightarrow \mathsf{X}u \mid v$

  Problème : on doit savoir combien de fois utiliser la règle avant de lire

  Solution : on modifie complètement les règles de X

  On a : $\mathcal{L}(\mathsf{X}) = \mathcal{L}(\mathsf{X}).\mathcal{L}(u) \cup \mathcal{L}(v)$

  Solution de l'équation ? $\mathcal{L}(\mathsf{X}) = \mathcal{L}(v).\mathcal{L}(u)^*$ (lemme d'Arden)

  $$\mathsf{X} \rightarrow v\mathsf{X}' \qquad \mathsf{X}' \rightarrow u\mathsf{X}' \mid \varepsilon$$

  Cela peut arriver en plusieurs étapes : $\mathsf{X} \Longrightarrow \mathsf{Y}\beta \Longrightarrow^* \mathsf{X}\alpha\beta$

Exemple de transformation

Exemple (Expressions arithmétiques)

$$\mathsf{exp} \;\to\; \mathsf{exp} * \mathsf{exp} \;\mid\; \mathsf{exp} \,/\, \mathsf{exp} \;\mid\; \mathsf{exp} ** \mathsf{exp} \;\mid\; (\,\mathsf{exp}\,) \;\mid\; \mathbf{nb}$$

Niveaux de priorité

$$\begin{array}{lcl}
\mathsf{exp}_2 & \to & \mathsf{exp}_2 * \mathsf{exp}_1 \mid \mathsf{exp}_2 \,/\, \mathsf{exp}_1 \mid \mathsf{exp}_1 \\
\mathsf{exp}_1 & \to & \mathsf{exp}_0 ** \mathsf{exp}_1 \mid \mathsf{exp}_0 \\
\mathsf{exp}_0 & \to & (\,\mathsf{exp}_2\,) \mid \mathbf{nb}
\end{array}$$

Factorisation

$$\begin{array}{lcl}
\mathsf{exp}_2 & \to & \mathsf{exp}_2\, \mathsf{exp}'_2 \mid \mathsf{exp}_1 \\
\mathsf{exp}'_2 & \to & * \,\mathsf{exp}_1 \mid /\, \mathsf{exp}_1 \\
\mathsf{exp}_1 & \to & \mathsf{exp}_0\, \mathsf{exp}'_1 \\
\mathsf{exp}'_1 & \to & ** \,\mathsf{exp}_1 \mid \varepsilon \\
\mathsf{exp}_0 & \to & (\,\mathsf{exp}_2\,) \mid \mathbf{nb}
\end{array}$$

Élimination de la récursion à gauche

$\mathcal{L}(\mathsf{exp}_2) = \mathcal{L}(\mathsf{exp}_2).\mathcal{L}(\mathsf{exp}'_2) \cup \mathcal{L}(\mathsf{exp}_1)$

donc $\mathcal{L}(\mathsf{exp}_2) = \mathcal{L}(\mathsf{exp}_1).\mathcal{L}(\mathsf{exp}'_2)^*$

D'où

$$\begin{array}{lcl}
\mathsf{exp}_2 & \to & \mathsf{exp}_1\, \mathsf{Y} \\
\mathsf{Y} & \to & \mathsf{exp}'_2\, \mathsf{Y} \mid \varepsilon \\
\mathsf{exp}'_2 & \to & * \,\mathsf{exp} \mid /\, \mathsf{exp} \\
\mathsf{exp}_1 & \to & \mathsf{exp}_0\, \mathsf{exp}'_1 \\
\mathsf{exp}'_1 & \to & ** \,\mathsf{exp}_1 \mid \varepsilon \\
\mathsf{exp}_0 & \to & (\,\mathsf{exp}_2\,) \mid \mathbf{nb}
\end{array}$$

On calcule ensuite les directeurs LL(1) et on écrit l'analyseur.