"Firsts" : terminals that may eventually start a non-terminal

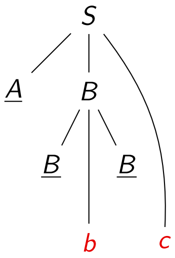
Prior to "follows", let's look at easier "firsts"

Definition of $\texttt{First}_G(N) \subseteq \mathcal{T} \cup \{\varepsilon\}$

$c \in \texttt{First}_G(N) \iff$ { there exists a syntactic tree G rooted by N which leftmost terminal leaf is c.

Example with

$$
\begin{aligned}
S &::= ABc \\
A &::= \epsilon \\
  &\quad \mid aA \\
B &::= \epsilon \\
  &\quad \mid BbB
\end{aligned}
$$



Thus
$b \in \texttt{First}(S)$

Firsts without Empty Rules

Firsts: Terminals that can start a word of non-terminals and terminals w

$$\texttt{First}(w \in (\mathcal{N} \cup \mathcal{T})^*) \subseteq \mathcal{T}$$

$\texttt{First}$ is defined as the smallest set such that:

- if $t \in \mathcal{T}$, $\texttt{First}(t) = \{t\}$,
- for any rule $N \mapsto w$, $\texttt{First}(N) \supseteq \texttt{First}(w)$,
- $\texttt{First}(w_1 w_2) = \texttt{First}(w_1)$.

Example of *Firsts* without Empty Rules By Fixed Point

Grammar

$\langle exp \rangle$ ::= $\langle exp \rangle$-$\langle term \rangle$
 | $\langle term \rangle$
$\langle term \rangle$ ::= $\langle term \rangle$*$\langle fac \rangle$
 | $\langle fac \rangle$
$\langle fac \rangle$ ::= ($\langle exp \rangle$)
 | -$\langle fac \rangle$
 | $\langle NB \rangle$

Firsts

$\texttt{First}(\langle exp \rangle) \supseteq \{\}$

$\texttt{First}(\langle term \rangle) \supseteq \{\}$

$\texttt{First}(\langle fac \rangle) \supseteq \{\}$

Example of *Firsts* without Empty Rules By Fixed Point

Grammar

$\langle \texttt{exp} \rangle ::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle$
$\quad | \ \langle \texttt{term} \rangle$
$\langle \texttt{term} \rangle ::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle$
$\quad | \ \langle \texttt{fac} \rangle$
$\langle \texttt{fac} \rangle ::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)}$
$\quad | \ \texttt{-} \langle \texttt{fac} \rangle$
$\quad | \ \langle \texttt{NB} \rangle$

Firsts

$\texttt{First}(\langle \texttt{exp} \rangle) \supseteq \{\}$
$\texttt{First}(\langle \texttt{term} \rangle) \supseteq \{\}$
$\texttt{First}(\langle \texttt{fac} \rangle) \supseteq \{\texttt{(}\}$

Example of *Firsts* without Empty Rules By Fixed Point

Grammar

$$\langle \texttt{exp} \rangle ::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle \mid \langle \texttt{term} \rangle$$

$$\langle \texttt{term} \rangle ::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle \mid \langle \texttt{fac} \rangle$$

$$\langle \texttt{fac} \rangle ::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)} \mid \texttt{-} \langle \texttt{fac} \rangle \mid \langle \texttt{NB} \rangle$$

Firsts

$\texttt{First}(\langle \texttt{exp} \rangle) \supseteq \{\}$

$\texttt{First}(\langle \texttt{term} \rangle) \supseteq \{\}$

$\texttt{First}(\langle \texttt{fac} \rangle) \supseteq \{\texttt{(}, \texttt{-}\}$

Example of *Firsts* without Empty Rules By Fixed Point

Grammar

$$
\begin{aligned}
\langle \texttt{exp} \rangle &::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle \\
&\mid \langle \texttt{term} \rangle \\
\langle \texttt{term} \rangle &::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle \\
&\mid \langle \texttt{fac} \rangle \\
\langle \texttt{fac} \rangle &::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)} \\
&\mid \texttt{-} \langle \texttt{fac} \rangle \\
&\mid \langle \texttt{NB} \rangle
\end{aligned}
$$

Firsts

$\texttt{First}(\langle \texttt{exp} \rangle) \supseteq \{\}$

$\texttt{First}(\langle \texttt{term} \rangle) \supseteq \{\}$

$\texttt{First}(\langle \texttt{fac} \rangle) \supseteq \{\texttt{(}, \texttt{-}, \langle \texttt{NB} \rangle\}$

Example of *Firsts* without Empty Rules By Fixed Point

Grammar

$$\langle \texttt{exp} \rangle ::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle \mid \langle \texttt{term} \rangle$$

$$\langle \texttt{term} \rangle ::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle \mid \langle \texttt{fac} \rangle$$

$$\langle \texttt{fac} \rangle ::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)} \mid \texttt{-} \langle \texttt{fac} \rangle \mid \langle \texttt{NB} \rangle$$

Firsts

$\texttt{First}(\langle \texttt{exp} \rangle) \supseteq \{\}$

$\texttt{First}(\langle \texttt{term} \rangle) \supseteq \{\texttt{(}, \texttt{-}, \langle \texttt{NB} \rangle\}$

$\texttt{First}(\langle \texttt{fac} \rangle) \supseteq \{\texttt{(}, \texttt{-}, \langle \texttt{NB} \rangle\}$

Example of *Firsts* without Empty Rules By Fixed Point

Grammar

$$
\begin{aligned}
\langle \texttt{exp} \rangle &::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle \\
&\mid \langle \texttt{term} \rangle \\
\langle \texttt{term} \rangle &::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle \\
&\mid \langle \texttt{fac} \rangle \\
\langle \texttt{fac} \rangle &::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)} \\
&\mid \texttt{-} \langle \texttt{fac} \rangle \\
&\mid \langle \texttt{NB} \rangle
\end{aligned}
$$

Firsts

$\texttt{First}(\langle \texttt{exp} \rangle) \supseteq \{\texttt{(}, \texttt{-}, \langle \texttt{NB} \rangle\}$

$\texttt{First}(\langle \texttt{term} \rangle) \supseteq \{\texttt{(}, \texttt{-}, \langle \texttt{NB} \rangle\}$

$\texttt{First}(\langle \texttt{fac} \rangle) \supseteq \{\texttt{(}, \texttt{-}, \langle \texttt{NB} \rangle\}$

Example of *Firsts* without Empty Rules By Fixed Point

Grammar

$$\langle \texttt{exp} \rangle ::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle \mid \langle \texttt{term} \rangle$$
$$\langle \texttt{term} \rangle ::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle \mid \langle \texttt{fac} \rangle$$
$$\langle \texttt{fac} \rangle ::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)} \mid \texttt{-} \langle \texttt{fac} \rangle \mid \langle \texttt{NB} \rangle$$

Firsts

$\texttt{First}(\langle \texttt{exp} \rangle) = \{\texttt{(}, \texttt{-}, \langle \texttt{NB} \rangle\}$

$\texttt{First}(\langle \texttt{term} \rangle) = \{\texttt{(}, \texttt{-}, \langle \texttt{NB} \rangle\}$

$\texttt{First}(\langle \texttt{fac} \rangle) = \{\texttt{(}, \texttt{-}, \langle \texttt{NB} \rangle\}$

Example of *Firsts* with Empty Rules

Grammar

$$S ::= T\,;S \mid \texttt{!}$$
$$T ::= T+T \mid \texttt{a} \mid \epsilon$$

Firsts ??

$\texttt{First}(S) = \{\texttt{a}, \texttt{!}\}$
$\texttt{First}(T) = \{\texttt{a}\}$

False: ';' must be in $\texttt{First}(S)$

Firsts

Firsts: Terminals that can start a word of non-terminals and terminals w

Attention, we use ε as an additional symbol to signify that w can recognize the empty word:

$$\texttt{First}(w \in (\mathcal{N} \cup \mathcal{T})^*) \subseteq \mathcal{T} \cup \{\varepsilon\}$$

$\texttt{First}$ is defined as the smallest set such that:

- if $t \in \mathcal{T}$, $\texttt{First}(t) = \{t\}$,
- for any rule $N \mapsto w$, $\texttt{First}(N) \supseteq \texttt{First}(w)$,
- $\texttt{First}(\epsilon) = \{\varepsilon\}$,
- $\texttt{First}(w_1 w_2) = \texttt{First}(w_1) \uplus_{\varepsilon} \texttt{First}(w_2)$.

Notation: Let $A, B \subseteq \mathcal{T} \cup \{\varepsilon\}$, then we denote

$$A \uplus_{\varepsilon} B := \begin{cases} A & \text{if } \varepsilon \notin A \\ (A - \{\varepsilon\}) \cup B & \text{if } \varepsilon \in A \end{cases}$$

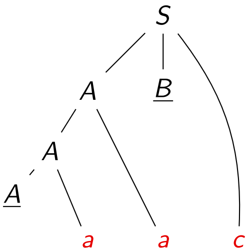
"Follows" : terminals that may eventually follow a non-terminal.

Definitions of $\texttt{Follow}_G(N) \subseteq \sqcup \cup \{\$\}$

$c \in \texttt{Follow}_G(N) \Longleftrightarrow$ { There exists a ST G with an internal node N which leftmost terminal leas RIGHT OF N is c.

Example ins

$$S ::= ABc$$
$$A ::= \epsilon \mid aA$$
$$B ::= \epsilon \mid BbB$$



Thus
$a \in \texttt{Follow}(A)$
$c \in \texttt{Follow}(A)$

Follows

Follows : the terminals that can follow a non-terminal N

The symbol \$ is used to denote the fact that N can be at the end of a file

$$\texttt{Follow}(N \in \mathcal{N}) \subseteq \mathcal{T} \cup \{\$\}$$

`Follow` is defined by the smallest set such that :

- if S is the main NT,

$$\texttt{Follow}(S) \ni \$ \ ,$$

- for each rule $N' \mapsto w_1 N w_2$,

$$\texttt{Follow}(N) \supseteq \texttt{First}(w_2) \cup_{\epsilon} \texttt{Follow}(N')$$

- for each rule $N' \mapsto w_1 N$,

$$\texttt{Follow}(N) \supseteq \texttt{Follow}(N')$$

Example of *Follow* (without ϵ rule)

Grammar

$$\langle \texttt{exp} \rangle ::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle \mid \langle \texttt{term} \rangle$$

$$\langle \texttt{term} \rangle ::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle \mid \langle \texttt{fac} \rangle$$

$$\langle \texttt{fac} \rangle ::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)} \mid \texttt{-} \langle \texttt{fac} \rangle \mid \langle \texttt{NB} \rangle$$

Follows

$\texttt{Follow}(\langle \texttt{exp} \rangle) \supseteq \{\$\}$

$\texttt{Follow}(\langle \texttt{term} \rangle) \supseteq \{\}$

$\texttt{Follow}(\langle \texttt{fac} \rangle) \supseteq \{\}$

Example of *Follow* (without ϵ rule)

Grammar

$\langle \texttt{exp} \rangle ::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle$
$\quad | \ \langle \texttt{term} \rangle$
$\langle \texttt{term} \rangle ::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle$
$\quad | \ \langle \texttt{fac} \rangle$
$\langle \texttt{fac} \rangle ::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)}$
$\quad | \ \texttt{-} \langle \texttt{fac} \rangle$
$\quad | \ \langle \texttt{NB} \rangle$

Follows

$\texttt{Follow}(\langle \texttt{exp} \rangle) \supseteq \{\$\}$
$\texttt{Follow}(\langle \texttt{term} \rangle) \supseteq \{\}$
$\texttt{Follow}(\langle \texttt{fac} \rangle) \supseteq \{\}$

Example of *Follow* (without ϵ rule)

Grammar

$$\langle \texttt{exp} \rangle ::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle \mid \langle \texttt{term} \rangle$$
$$\langle \texttt{term} \rangle ::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle \mid \langle \texttt{fac} \rangle$$
$$\langle \texttt{fac} \rangle ::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)} \mid \texttt{-} \langle \texttt{fac} \rangle \mid \langle \texttt{NB} \rangle$$

Follows

$\texttt{Follow}(\langle \texttt{exp} \rangle) \supseteq \{\$, \texttt{-}\}$

$\texttt{Follow}(\langle \texttt{term} \rangle) \supseteq \{\}$

$\texttt{Follow}(\langle \texttt{fac} \rangle) \supseteq \{\}$

Example of *Follow* (without ϵ rule)

Grammar

⟨exp⟩ ::= ⟨exp⟩-⟨term⟩
| ⟨term⟩
⟨term⟩ ::= ⟨term⟩*⟨fac⟩
| ⟨fac⟩
⟨fac⟩ ::= (⟨exp⟩)
| -⟨fac⟩
| ⟨NB⟩

Follows

Follow(⟨exp⟩) $\supseteq$ {$, -}
Follow(⟨term⟩) $\supseteq$ {$, -}
Follow(⟨fac⟩) $\supseteq$ {}

Example of *Follow* (without ϵ rule)

Grammar

$$\langle \texttt{exp} \rangle ::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle \mid \langle \texttt{term} \rangle$$

$$\langle \texttt{term} \rangle ::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle \mid \langle \texttt{fac} \rangle$$

$$\langle \texttt{fac} \rangle ::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)} \mid \texttt{-} \langle \texttt{fac} \rangle \mid \langle \texttt{NB} \rangle$$

Follows

$\texttt{Follow}(\langle \texttt{exp} \rangle) \supseteq \{\$, \texttt{-}\}$

$\texttt{Follow}(\langle \texttt{term} \rangle) \supseteq \{\$, \texttt{-}, \texttt{*}\}$

$\texttt{Follow}(\langle \texttt{fac} \rangle) \supseteq \{\}$

Example of *Follow* (without ϵ rule)

Grammar

$$
\begin{aligned}
\langle \texttt{exp} \rangle &::= \langle \texttt{exp} \rangle \texttt{-} \langle \texttt{term} \rangle \\
&\mid \langle \texttt{term} \rangle \\
\langle \texttt{term} \rangle &::= \langle \texttt{term} \rangle \texttt{*} \langle \texttt{fac} \rangle \\
&\mid \langle \texttt{fac} \rangle \\
\langle \texttt{fac} \rangle &::= \texttt{(} \langle \texttt{exp} \rangle \texttt{)} \\
&\mid \texttt{-} \langle \texttt{fac} \rangle \\
&\mid \langle \texttt{NB} \rangle
\end{aligned}
$$

Follows

$\texttt{Follow}(\langle \texttt{exp} \rangle) \supseteq \{\$, \texttt{-}\}$

$\texttt{Follow}(\langle \texttt{term} \rangle) \supseteq \{\$, \texttt{-}, \texttt{*}\}$

$\texttt{Follow}(\langle \texttt{fac} \rangle) \supseteq \{\$, \texttt{-}, \texttt{*}\}$

Example of *Follow* (without ϵ rule)

Grammar

⟨exp⟩ ::= ⟨exp⟩-⟨term⟩
| ⟨term⟩
⟨term⟩ ::= ⟨term⟩*⟨fac⟩
| ⟨fac⟩
⟨fac⟩ ::= (⟨exp⟩)
| -⟨fac⟩
| ⟨NB⟩

Follows

Follow(⟨exp⟩) ⊇ {$, -,)}
Follow(⟨term⟩) ⊇ {$, -, *}
Follow(⟨fac⟩) ⊇ {$, -, *}

Example of *Follow* (without ϵ rule)

Grammar

$\langle exp\rangle ::= \langle exp\rangle - \langle term\rangle$
$\quad | \ \langle term\rangle$
$\langle term\rangle ::= \langle term\rangle * \langle fac\rangle$
$\quad | \ \langle fac\rangle$
$\langle fac\rangle ::= (\langle exp\rangle)$
$\quad | \ -\langle fac\rangle$
$\quad | \ \langle NB\rangle$

Follows

$Follow(\langle exp\rangle) \supseteq \{\$, -,)\}$
$Follow(\langle term\rangle) \supseteq \{\$, -, *,)\}$
$Follow(\langle fac\rangle) \supseteq \{\$, -, *\}$

Example of *Follow* (without ϵ rule)

Grammar

$$\langle\texttt{exp}\rangle ::= \langle\texttt{exp}\rangle\texttt{-}\langle\texttt{term}\rangle \mid \langle\texttt{term}\rangle$$

$$\langle\texttt{term}\rangle ::= \langle\texttt{term}\rangle\texttt{*}\langle\texttt{fac}\rangle \mid \langle\texttt{fac}\rangle$$

$$\langle\texttt{fac}\rangle ::= \texttt{(}\langle\texttt{exp}\rangle\texttt{)} \mid \texttt{-}\langle\texttt{fac}\rangle \mid \langle\texttt{NB}\rangle$$

Follows

$\texttt{Follow}(\langle\texttt{exp}\rangle) \supseteq \{\$, \texttt{-}, \texttt{)}\}$

$\texttt{Follow}(\langle\texttt{term}\rangle) \supseteq \{\$, \texttt{-}, \texttt{*}, \texttt{)}\}$

$\texttt{Follow}(\langle\texttt{fac}\rangle) \supseteq \{\$, \texttt{-}, \texttt{*}, \texttt{)}\}$

Example of *Follow* (without ϵ rule)

Grammar

$$\langle\texttt{exp}\rangle ::= \langle\texttt{exp}\rangle\texttt{-}\langle\texttt{term}\rangle \mid \langle\texttt{term}\rangle$$

$$\langle\texttt{term}\rangle ::= \langle\texttt{term}\rangle\texttt{*}\langle\texttt{fac}\rangle \mid \langle\texttt{fac}\rangle$$

$$\langle\texttt{fac}\rangle ::= \texttt{(}\langle\texttt{exp}\rangle\texttt{)} \mid \texttt{-}\langle\texttt{fac}\rangle \mid \langle\texttt{NB}\rangle$$

Follows

$\texttt{Follow}(\langle\texttt{exp}\rangle) = \{\$, \texttt{-}, \texttt{)}\}$

$\texttt{Follow}(\langle\texttt{term}\rangle) = \{\$, \texttt{-}, \texttt{*}, \texttt{)}\}$

$\texttt{Follow}(\langle\texttt{fac}\rangle) = \{\$, \texttt{-}, \texttt{*}, \texttt{)}\}$

Example of complexe *Follow* (with ϵ)

Grammaire

$$S ::= TS; \mid \epsilon$$

$$T ::= T{+}T \mid \mathtt{a} \mid \epsilon$$

Follows

$\mathtt{Follow}(S) \supseteq \{\$\}$

$\mathtt{Follow}(T) \supseteq \{\}$

Knowing that:

$\mathtt{First}(S) = \{\varepsilon, \mathtt{a}, \mathtt{+}, \mathtt{;}\}$

$\mathtt{First}(T) = \{\varepsilon, \mathtt{a}, \mathtt{+}\}$

Example of complexe *Follow* (with ϵ)

Grammaire

$$S ::= T S ; \mid \epsilon$$

$$T ::= T + T \mid \mathtt{a} \mid \epsilon$$

Follows

$\texttt{Follow}(S) \supseteq \{\$\}$

$\texttt{Follow}(T) \supseteq \{\mathtt{a}, \mathtt{+}, ;\}$

Knowing that:

$\texttt{First}(S) = \{\varepsilon, \mathtt{a}, \mathtt{+}, ;\}$

$\texttt{First}(T) = \{\varepsilon, \mathtt{a}, \mathtt{+}\}$

Example of complexe *Follow* (with ϵ)

Grammaire

$$S ::= T S ;$$
$$\mid \epsilon$$
$$T ::= T + T$$
$$\mid \texttt{a}$$
$$\mid \epsilon$$

Follows

$\texttt{Follow}(S) \supseteq \{\$, ;\}$

$\texttt{Follow}(T) \supseteq \{\texttt{a}, +, ;\}$

Knowing that:

$\texttt{First}(S) = \{\varepsilon, \texttt{a}, +, ;\}$

$\texttt{First}(T) = \{\varepsilon, \texttt{a}, +\}$

Example of complexe *Follow* (with ϵ)

Grammaire

$$S ::= T S ;$$
$$\mid \epsilon$$
$$T ::= T + T$$
$$\mid a$$
$$\mid \epsilon$$

Follows

$\texttt{Follow}(S) = \{\$, ;\}$

$\texttt{Follow}(T) = \{a, +, ;\}$

Knowing that:
$\texttt{First}(S) = \{\varepsilon, a, +, ;\}$
$\texttt{First}(T) = \{\varepsilon, a, +\}$

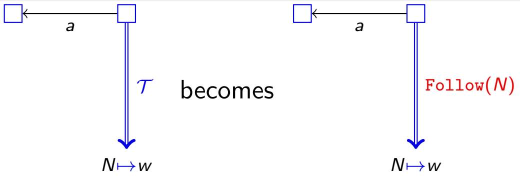
SLR Parser: narrowing conditions of actions

If a would-be NT can't be followed by peeked at terminal there is no reason to take the action

This is why we compute **"follows"** :
 the set of terminals that may eventually follow a given non-terminal.

Idea : narrowing reductions paths to *Follows* only

Same construction as LR_0 except that reduction action $N \mapsto w$ are are narrowed to `Follow`(N) (before or after determinisation).



LL : Guessing using Firsts and Follows

Reminder : in Java, a non-terminal is mapped to a regexp of terminals+NT

Cases disjunction $r \mid s$

These cases correspond to Firsts of each pattern
ex : $\texttt{First}(+\langle\texttt{term}\rangle) = \{+\}$

Kleene star r^* (or recursive non-terminal)

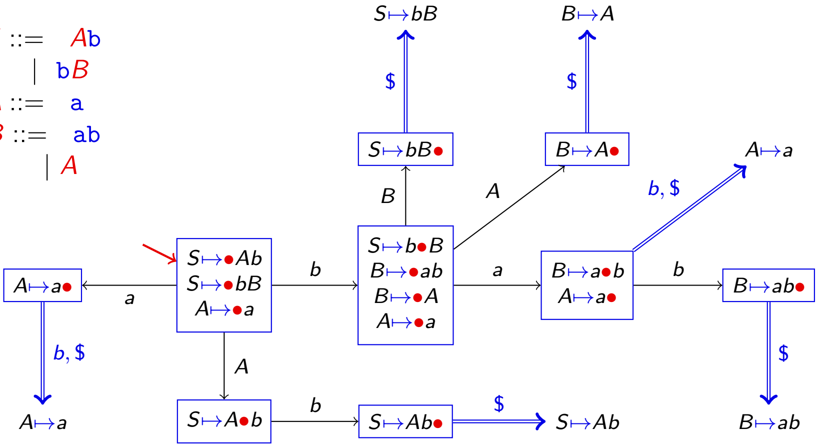
Idem plus an additional case to exit the loop using a Follow.
ex : $\texttt{Follow}(\langle\texttt{term}\rangle) = \{\backslash n, +, -,)\}$

Conflict

If these are two conflicting “cases” in a switch then the grammar is not considered LL_1

Some artificial conflicts remaining

$S ::= Ab \mid bB$
$A ::= a$
$B ::= ab \mid A$



LR_1 Automaton: a distinct non-terminal for each follow

The grammar is refined by duplicating non-terminals

- For each $N \in \mathcal{T}$ and each $t \in \texttt{Follow}(N)$ we create N_t,
- Rules of N_t are those of N, but split for each encountered non-terminal
- Compute the SLR automaton on resulting grammar,
- Remove annotations.

$$\begin{aligned} S &::= A\texttt{b} \\ &\mid \texttt{b}B \\ A &::= \texttt{a} \\ B &::= \texttt{ab} \\ &\mid A \end{aligned}$$

becomes

$$\begin{aligned} S_\$ &::= A_b\texttt{b} \\ &\mid \texttt{b}B_\$ \\ A_\$ &::= \texttt{a} \\ A_b &::= \texttt{a} \\ B_\$ &::= \texttt{ab} \\ &\mid A_\$ \end{aligned}$$

Computing intermediate grammar for LR_1 algorithm

Reminder: $\mathcal{G} = (\mathcal{T}, \mathcal{N}, \mathcal{R})$

$\mathcal{T}$: terminlks $\mathcal{N}$: non terminals
$\mathcal{R} \subseteq \mathcal{N} \times (\mathcal{N}+\mathcal{T})^*$ rules of the form $N \mapsto w$.

$\mathtt{LR1}(\mathcal{G}) = (\mathcal{T}, \mathcal{N}', \mathcal{R}'$

$$\mathcal{N}' = \{N_t \mid N \in \mathcal{N}, t \in \mathtt{Follow}(N)\}$$

$$\mathcal{R}' = \{N_t \mapsto w' \mid (N \mapsto w) \in \mathcal{R},\ w' \in \langle w \mid t \rangle\}$$

$$\langle wt' \mid t \rangle = \langle w \mid t' \rangle t'$$

$$\langle wN \mid t \rangle = \bigcup_{t' \in \mathtt{First}(Nt)} \langle w \mid t' \rangle N_t$$

LR_1 parser are huge

$$
\begin{aligned}
S &::= A\mathtt{b}B \\
&\mid \mathtt{b}AB \\
A &::= B\mathtt{a} \\
B &::= \mathtt{a}S \\
&\mid \mathtt{b}A
\end{aligned}
$$

devient

$$
\begin{aligned}
S_\$ &::= A_b\mathtt{b} & S_a &::= A_b\mathtt{b} \\
&\mid \mathtt{b}A_aB_\$ & &\mid \mathtt{b}A_aB_a \\
&\mid \mathtt{b}A_bB_\$ & &\mid \mathtt{b}A_bB_a \\
A_\$ &::= B_a\mathtt{a} & A_b &::= B_a\mathtt{a} \\
A_a &::= B_a\mathtt{a} & & \\
B_\$ &::= \mathtt{ab} & B_a &::= \mathtt{ab} \\
&\mid bA_\$ & &\mid bA_a
\end{aligned}
$$

The resulting grammar has its size multiplied by T^p

where T it the number of terminals and p the maximum number of terminals appearing in a single rule.
... and this is before determinisation which is potentially exponential.

Used in modern generators

Minimized version

Minimisation can be peformed on the fly, resulting in re more reasonably sized parser.

Modern LR parser generators use LR_1.
They are not seen in TP because they are more dificult to install.

LR_k

It is possible to perform a LR algorithms splitting the grammar using more "look-ahead" information (similar to a follow but looking for sub-words of size k rather than letters), the resulting algorithm is called LR_k.

The resulting grammar becomes of size

$$T^{p*k}$$

which soon becomes unrealistic, for little gain.

Universality of LR_1

As we have seen, any grammair LR_k can be modified into an "equivalent" *SLR* grammar.
In fact any non-ambiguous grammar can be modified into a *SLR* grammar, but there is no generic algorithm and there can't be any (the proof is fundamentally non-constructive).

LALR : A SLR-sized parser nearly as expressive as LR_1

Principle

- Create a psedo-deterministic LR_1 parser
- errase annotation,
- fuse states with the same names

The fusion will not create new shift-action conflict

Because shifts are forced by names.

Howeover, there could be new action-action conflicts (corresponding to the creation of the same non-terminals from diferent follows).

Size of SLR

Since the names are set of the the non-deterministic LR_0

Used by “old” parser generators such as Bison or Yacc.

Why not using a unic algorithm

Ambiguity is not decidable

It is only recursivelly enumerable, which mean that it is alway possible to have a proof of ambiguity, but not necessarlily the other way around.

Prouvable via a reduction to "Post correspondance" problem.

Consequence: to accept any non-ambiguous grammar, we have to accept any (context-free) grammar.

Not interesting for programming language: we need to be sure that there is no ambuguity in order not to confuse the programmer.
In addition those are longuer to parse.

Chart parsers : GLR, Earley, CYK, Packrat...

Those are “universal” parser generators that accept any (context-free) grammar, but which parsers can produce several trees from the same input.

Generated parsers work in time $o(n^{2+\epsilon})$

When the grammar in non-ambiguous, they are linear, but with arbitrarily large constant.

Same coplexity as the matrix multiplication

Used for natural language processing

Human language are all ambiguous.

Potatoes view of parser generators