# 3.3. Recoursive parsing

## LL(1)-grammar

- Let us observe the following grammar $G$:

$$\begin{array}{rcl} E & \to & T + E \mid T - E \mid T \\ T & \to & a \mid (E) \end{array}$$

- Modify the productions of $E$ to get an equivalent grammar $G'$:

$$\begin{array}{rcl} E & \to & TE' \\ E' & \to & +E \mid -E \mid \epsilon \\ T & \to & a \mid (E) \end{array}$$

- The leftmost derivations of the strings of $G'$ can easily formed from the start symbol $E$: In every phase of parsing the *next* character of the string uniquely determines the production which has to be selected.

- Grammar with this property is called *LL(1)-grammar* ("left to right scan, producing left parse with 1 symbol lookahead")

## Recoursive parser for LL(1)-grammar

- The parser (parsing program) consists of recoursive functions that correspond the nonterminal variables.
- The parser for $G'$:

```
function E
begin
  do something;
  T;
  E';
end

function E'
begin
  if (next == '+') then
  begin
      do something;
      next = getnext;
      E;
  end
  else
      begin
          if (next == '-') then
              begin
                  do something;
                  next = getnext;
                  E;
              end
          else do something;
      end
end

function T
begin
  if (next == 'a') then
```

```
        begin
            do something;
            next = getnext;
            return;
        end
    else
        begin
            if (next == '(') then
                begin
                    do something;
                    next = getnext;
                    E;
                    if (next ≠ ')')
                            ERROR;
                    next = getnext;
                end
            else ERROR;
        end
end
```

In the main program:
$next = getnext$;
$E$;

- The parse of string `a-(a+a)` when the command "do something"prints the used production:

  ```
  E → TE'
  T → a
  E' → -E
  E → TE'
  T → (E)
  E → TE'
  T → a
  E' → +E
  E → TE'
  T → a
  E' → ϵ
  E' → ϵ.
  ```

  The above corresponds the derivation:

$$
\begin{aligned}
E &\Rightarrow TE' \Rightarrow aE' \Rightarrow a - E \Rightarrow a - TE' \\
&\Rightarrow a - (E)E' \Rightarrow a - (TE')E' \\
&\Rightarrow a - (aE')E' \Rightarrow a - (a + E)E' \\
&\Rightarrow a - (a + TE')E' \Rightarrow a - (a + aE')E' \\
&\Rightarrow a - (a + a)E' \Rightarrow a - (a + a)
\end{aligned}
$$

## The general form of a LL(1) grammar

- In LL(1) grammars we also allow in restricted form
  - Productions which has the right side starting with a product
  - Products $A$ for which $A \Rightarrow^* \epsilon$
- The grammar producing language $a^*b \cup c^*d$:

$$\begin{array}{rcl} S & \rightarrow & Ab \mid Cd \\ A & \rightarrow & aA \mid \epsilon \\ C & \rightarrow & cC \mid \epsilon. \end{array}$$

  Grammar is LL(1), despite that the firstly used product can not be decided directly looking the products of $S$

## Modifying a grammar to LL(1) form

- LL(1) rather an limited subclass of context free languages
- Whith the following transformations some "almost"LL(1) grammars can be modified to LL(1) form

## 1. LEFT FACTORING

- Grammar with products

$$A \to \alpha\beta_1 \mid \alpha\beta_2, \qquad \alpha \neq \epsilon,\ \beta_1 \neq \beta_2$$

can not be LL(1)

- Modification: let us take a new variable $A'$ and replace the above products with the following

$$\begin{aligned} A &\to \alpha A' \\ A' &\to \beta_1 \mid \beta_2, \end{aligned}$$

where $\alpha$ is the longest common prefix of $\alpha\beta_1$ and $\alpha\beta_2$

2. REMOVAL IMMEDIATE LEFT RECURSION

- Grammar is *left recoursive*, if for some variable $A$ and string $\gamma$
$$A \Rightarrow^+ A\gamma,$$
where $\alpha \Rightarrow^+ \beta$ means that $\beta$ can be derived from $\alpha$ with derivatin having lenght at least one
- Left recoursive grammar can not be LL(1)
- *Immediate* left recursion, i.e., the derivations $A \Rightarrow A\gamma$, can be eliminated by replacing products
$$A \to A\beta \mid \alpha, \qquad \beta \neq \epsilon,$$
with
$$\begin{aligned} A &\to \alpha A' \\ A' &\to \beta A' \mid \epsilon \end{aligned}$$
- Products of the form $A \to A$ can be removed