Rekursywny parser zstępujący

- na wejściu mamy ciąg terminali;
- każdemu nieterminalowi $\langle A \rangle$ gramatyki odpowiada funkcja rekursywna f_A, której wywołanie powoduje
  - **albo** przeczytanie z wejścia najdłuższego ciągu w terminali, takiego że $\langle A \rangle \Rightarrow^* w$, oraz skonstruowanie drzewa wywodu słowa w z nieterminalu $\langle A \rangle$,
  - **albo** sygnalizację błędu;
- funkcja f_A jest definiowana według produkcji gramatycznych z nieterminalu $\langle A \rangle$
  - każdemu terminalowi występującemu w produkcji odpowiada sprawdzenie, czy na wejściu pojawia się ten właśnie terminal, i przeczytanie go z wejścia,
  - każdemu nieterminalowi $\langle B \rangle$ występującemu w produkcji odpowiada wywołanie funkcji f_B,
  - dojściu do końca produkcji odpowiada skonstruowanie drzewa z mniejszych drzew dostarczonych przez te wszystkie wywołania,
  - niezgodności terminalu w produkcji i na wejściu oraz błędowi przy wywołaniu funkcji f_B odpowiada próba przejścia do następnej produkcji z tego nieterminalu — przy tym nie można cofnąć już wczytanych terminali z wejścia.

Przekształcenie gramatyki dla parsera zstępującego

Zmiana kolejności produkcji i „wyłączenie przed nawias":

$$
\begin{aligned}
\langle W \rangle &::= \langle S \rangle \mid \langle S \rangle + \langle W \rangle \mid \langle S \rangle - \langle W \rangle \\
\langle S \rangle &::= \langle C \rangle \mid \langle C \rangle * \langle S \rangle \mid \langle C \rangle / \langle S \rangle \\
\langle C \rangle &::= \langle L \rangle \mid (\langle W \rangle) \\
\langle L \rangle &::= 0 \mid 1 \mid 0 \langle L \rangle \mid 1 \langle L \rangle
\end{aligned}
$$

$$
\begin{aligned}
\langle W \rangle &::= \langle S \rangle \left\{ + \langle W \rangle \mid - \langle W \rangle \mid \lambda \right\} \\
\langle S \rangle &::= \langle C \rangle \left\{ * \langle S \rangle \mid / \langle S \rangle \mid \lambda \right\} \\
\langle C \rangle &::= (\langle W \rangle) \mid \langle L \rangle \\
\langle L \rangle &::= 0 \left\{ \langle L \rangle \mid \lambda \right\} \mid 1 \left\{ \langle L \rangle \mid \lambda \right\}
\end{aligned}
$$

Programowanie parsera zstępującego

$\langle W \rangle \ ::= \ \langle S \rangle \ \{ + \langle W \rangle \mid - \langle W \rangle \mid \lambda \}$

```
drzewo f_W() {
  t_1 = f_S();
  if (wejście == '+') {
    nast_leks(); t_2 = f_W();
    return (drzewo zrobione z t_1, +, t_2)
  } else
  if (wejście == '-') {
    nast_leks(); t_2 = f_W();
    return (drzewo zrobione z t_1, −, t_2)
  }
  else return (drzewo zrobione z t_1)
}
```

Uwaga: To jeszcze nie jest dokładnie to, co trzeba. Patrz dalej.

Parser zstępujący — problemy

Lewostronna rekursja w gramatyce powoduje nieskończone obliczenie

Przykład: inna gramatyka —
$\langle W \rangle \ ::= \ \langle S \rangle \mid \langle W \rangle + \langle S \rangle$

```
void f_W(Boolean* ok, drzewo* t) {
  drzewo  t_1, t_2;
  f_S(ok, &t_1);
  if (*ok)  *t = &(drzewo zrobione z t_1);
  else {
    f_W(ok, &t_1);
    if (*ok) {
      if (wejście == '+') {
        nast_leks(); f_S(ok, &t_2);
        if (*ok)  *t = &(drzewo zrobione z t_1, +, t_2);
        else  blad();
      }
      else  *ok = FALSE;
    }
  }
}
```

Leczenie lewostronnej rekursji

Problem: produkcje gramatyczne postaci

$$\langle \mathrm{N} \rangle \quad ::= \quad \ldots \mid \langle \mathrm{N} \rangle \ldots \mid \ldots$$

(czyli *lewostronnie rekursywne*) prowadzą do zapętlenia parsera zstępującego. Takie produkcje potrzebne są dla często spotykanej w składni języków programowania konstrukcji *„ciąg . . . grupowany do lewej"*.

Przykład:

$$\langle \text{liczba} \rangle \quad ::= \quad \langle \text{cyfra} \rangle \mid \langle \text{cyfra} \rangle \langle \text{liczba} \rangle$$

— ciąg cyfr grupowany do prawej (nie ma problemu)

$345 = 3\lfloor 45 \rfloor = 3 \cdot 10^2 + 45$

$$\langle \text{liczba} \rangle \quad ::= \quad \langle \text{cyfra} \rangle \mid \langle \text{liczba} \rangle \langle \text{cyfra} \rangle$$

— ciąg cyfr grupowany do lewej (jest problem)

$345 = \lfloor 34 \rfloor 5 = 34 \cdot 10 + 5$

Leczenie lewostronnej rekursji

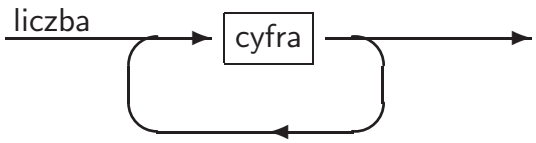
Wyjście: do gramatyk wprowadzić specjalną konstrukcję oznaczającą *„ciąg . . . grupowany do lewej"*:

Przykład:
Żeby wyrazić „liczba to niepusty ciąg cyfr grupowany do lewej" — zamiast

$$\langle \text{liczba} \rangle \quad ::= \quad \langle \text{cyfra} \rangle \mid \langle \text{liczba} \rangle \langle \text{cyfra} \rangle$$

piszemy

$$\langle \text{liczba} \rangle \quad ::= \quad \langle \text{cyfra} \rangle \langle \text{cyfra} \rangle^*$$



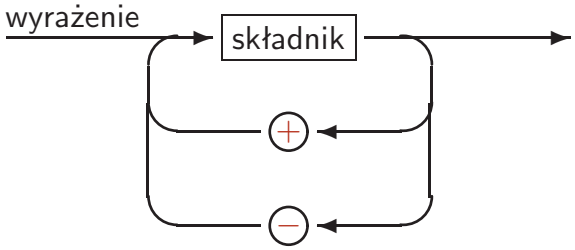
Leczenie lewostronnej rekursji

Wyjście: do gramatyk wprowadzić specjalną konstrukcję oznaczającą *„ciąg . . . grupowany do lewej”*:

Przykład:
Żeby wyrazić „wyrażenie to niepusty ciąg składników porozdzielanych plusami i minusami grupowany do lewej” —

zamiast $\langle \text{wyrażenie} \rangle ::= \langle \text{składnik} \rangle \mid \langle \text{wyrażenie} \rangle + \langle \text{składnik} \rangle \mid \langle \text{wyrażenie} \rangle - \langle \text{składnik} \rangle$

piszemy $\langle \text{wyrażenie} \rangle ::= \langle \text{składnik} \rangle \left\{ \left\{ + \mid - \right\} \langle \text{składnik} \rangle \right\}^*$



Leczenie lewostronnej rekursji

Gramatyka oryginalna z lewostronną rekursją:

$\langle W \rangle ::= \langle S \rangle \mid \langle W \rangle + \langle S \rangle \mid \langle W \rangle - \langle S \rangle$

$\langle S \rangle ::= \langle C \rangle \mid \langle S \rangle * \langle C \rangle \mid \langle S \rangle / \langle C \rangle$

$\langle C \rangle ::= \langle L \rangle \mid (\langle W \rangle)$

$\langle L \rangle ::= 0 \mid 1 \mid \langle L \rangle 0 \mid \langle L \rangle 1$

Gramatyka wyleczona z iteracją:

$\langle W \rangle ::= \langle S \rangle \left\{ \left\{ + \mid - \right\} \langle S \rangle \right\}^*$

$\langle S \rangle ::= \langle C \rangle \left\{ \left\{ * \mid / \right\} \langle C \rangle \right\}^*$

$\langle C \rangle ::= (\langle W \rangle) \mid \langle L \rangle$

$\langle L \rangle ::= \left\{ 0 \mid 1 \right\} \left\{ 0 \mid 1 \right\}^*$