

## Priorite des operateurs

- $S \rightarrow S + S$
- $S \rightarrow S * S$
- $S \rightarrow n$

On peut decire un peu plus:

- $S \rightarrow S_+ | S_* | S_n$
- $S_+ \rightarrow S + S$
- $S_* \rightarrow S * S$
- $S_n \rightarrow n$

Il faut respecter l'associativite et la priorite.

Voici la liste des configurations possibles lorsque l'on rencontre  $S_+$ :

- $S_+$  et  $S_+$  NOPE
- $S_+$  et  $S_*$  OK
- $S_+$  et  $S_n$  OK
- $S_*$  et  $S_+$  NOPE
- $S_*$  et  $S_*$  OK
- $S_*$  et  $S_n$  OK
- $S_n$  et  $S_+$  NOPE
- $S_n$  et  $S_*$  OK
- $S_n$  et  $S_n$  OK

Voici la liste des configurations possibles lorsque l'on rencontre  $S_*$ :

- $S_+$  et  $S_+$  NOPE
- $S_+$  et  $S_*$  NOPE
- $S_+$  et  $S_n$  NOPE
- $S_*$  et  $S_+$  NOPE
- $S_*$  et  $S_*$  NOPE
- $S_*$  et  $S_n$  OK
- $S_n$  et  $S_+$  NOPE
- $S_n$  et  $S_*$  NOPE
- $S_n$  et  $S_n$  OK

Du coup on a une nouvelle grammaire

- $S \rightarrow S_+ | S_* | S_n$
- $S_+ \rightarrow S + S_* | S + S_n$
- $S_* \rightarrow S_* * S_n | S_n * S_n$
- $S_n \rightarrow n$

On peut reduire cette grammaire:

- $S \rightarrow S + T | T$
- $T \rightarrow T * S_n | S_n$
- $S_n \rightarrow n$

## Ajoutons des features

On va ajouter des parentheses, le moins unaire, sqrt...

- $S \rightarrow S "+" T | T$
- $T \rightarrow T "*" S_n | S_n$
- $S_n \rightarrow "n" | "(" S ")" | "-" S | "sqrt(" S ")"$

## Reconnaissance des grammaire

Il y a deux strategies:

1. Partir du mot et arriver a l'axiome
2. Partir de l'axiome et arriver au mot.

## Exemple parser LL

1.  $inst \rightarrow \text{"if" exp "then" inst "fi"}$
2.  $inst \rightarrow \text{"print foo"}$
3.  $exp \rightarrow cond$
4.  $cond \rightarrow \text{"true"} | \text{"false"}$

$First(inst) = \{ 'if', 'print' \}$

$First(exp) = First(cond) = \{ 'true', 'false' \}$

L'utilisation de ce parser a des cas penibles. Par exemple lorsque deux regles ont le meme first. Il faut aussi gerer les recursions gauches.