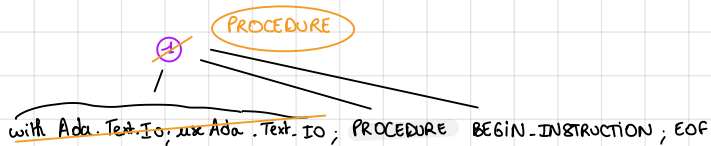


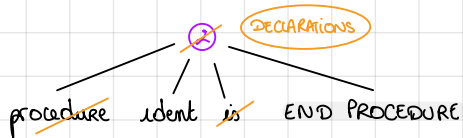
FICHIER



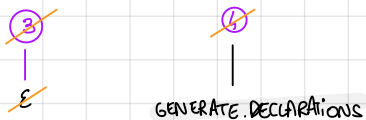
FICHIER

ROOT

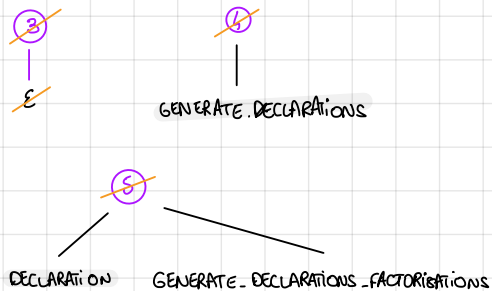
PROCEDURE



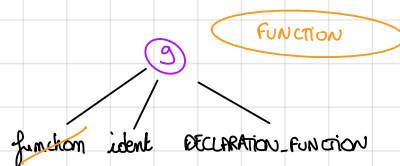
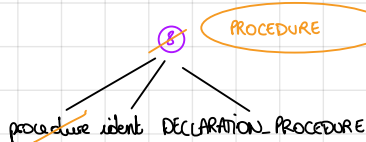
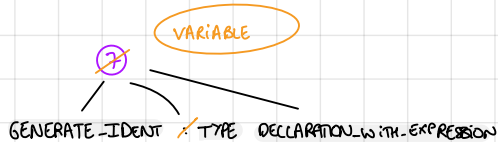
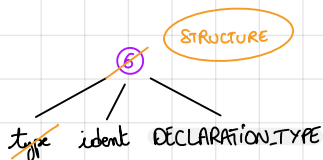
END PROCEDURE



GENERATE-DECLARATIONS



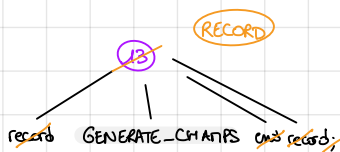
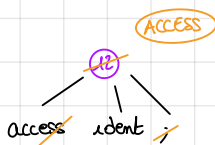
DECLARATION



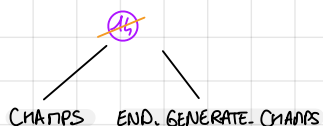
DECLARATION-TYPE



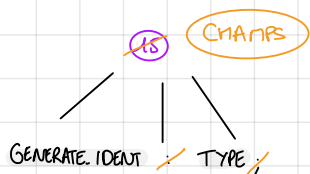
ACCESS-RECORD



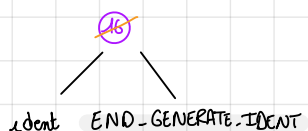
GENERATE-CHAMPS



CHAMPS



GENERATE-IDENT



The diagram shows a non-terminal 'ID' in a purple circle with a diagonal slash. Two arrows point down from 'ID' to a terminal 'i' (also with a slash) and the text 'GENERATE_IDENT'.

Diagram illustrating the structure of the tree:

- Node 19 (labeled ~~19~~) is associated with **TYPE-IDENT**. It has a single child labeled **ident**.
- Node 20 (labeled ~~20~~) is associated with **TYPE-ACCESS**. It has two children: **access** and **ident**.

DECL - WITH - EXPRESSION

DECL

IDENT

UNARY EXPRESSION

SEMICOLON

Diagram illustrating a choice between two expressions, `EXPRESSION_1` and `EXPRESSION_OR`, with a red circle and a diagonal line indicating a selection or conflict.

A parse tree diagram for the expression "new ident EXPRESSION-OR". The root node is a purple circle containing "28", which is crossed out with a red diagonal line. To the right of the root is an orange oval containing the text "EXPRESSION-NEW". The root has three children: "new" (with a red diagonal line), "ident", and "EXPRESSION-OR".

Diagram illustrating the structure of the expression `2 + 3 * 4` using the `EXPRESSION.CHAR_VAL` type.

- The root node is `EXPRESSION.CHAR_VAL` (circled in orange).
- It branches into six children:
 - `Character_val` (circled in orange)
 - `UNARY_EXPRESSION`
 - `EXPRESSION_OR`
 - Three unlabeled nodes.
- The three unlabeled nodes further branch into:
 - `2` (circled in orange)
 - `+`
 - `3`
 - `*`
 - `4`
 - One unlabeled node.

A tree diagram for the expression "NOT 3+4". The root node is "UNARY EXPRESSION", which branches into "NOT" and "3+4". The "3+4" node branches into "3" and "4". The "3" node is marked with a red "not" and a red "x".

32
|
EXPRESSION-3

```

graph TD
    OR1((OR)) --- OR2((OR))
    OR1 --- UNARY1[UNARY]
    OR1 --- EXPRESSION1[EXPRESSION-ELSE]
    OR2 --- OR3((OR))
    OR2 --- UNARY2[UNARY]
    OR2 --- EXPRESSION2[EXPRESSION-ELSE]
  
```

~~37~~

Diagram illustrating the OR-ELSE operator precedence. The root node is a purple circle containing '36' with a diagonal slash. It branches into four children: 'else', 'UNARY', 'EXPRESSION_1', and 'EXPRESSION_OR'. An orange oval labeled 'OR-ELSE' is positioned above the 'EXPRESSION_OR' node.

37 AND

AND UNARY EXPRESSION-THEN

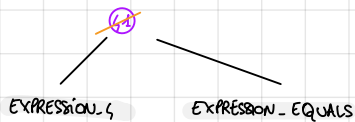
~~3~~

[illegible]

then UNARY EXPRESSION NOT EXPRESSION. AND

AND.THEN

EXPRESSION_3



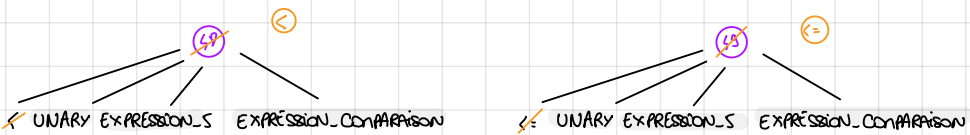
EXPRESSION_EQUALS



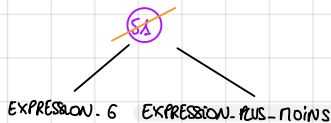
EXPRESSION_4



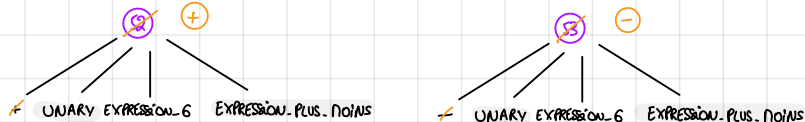
EXPRESSION_COMPARISON



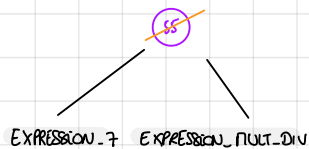
EXPRESSION_5



EXPRESSION_PLUS_MINUS



EXPRESSION_6



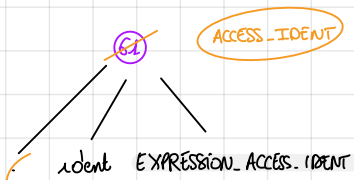
EXPRESSION_MULT_DIV



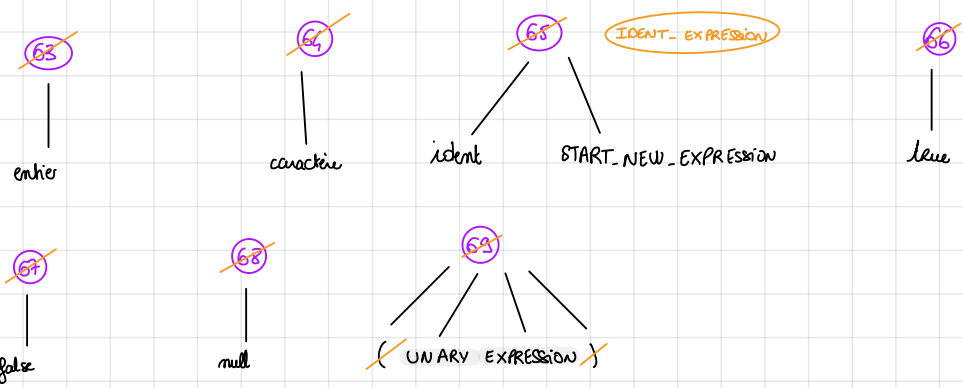
EXPRESSION_7



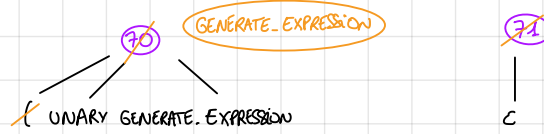
EXPRESSION_ACCESS_IDENT



EXPRESSION_ATOMS



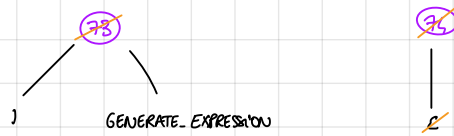
START_NEW_EXPRESSION



GENERATE_EXPRESSION



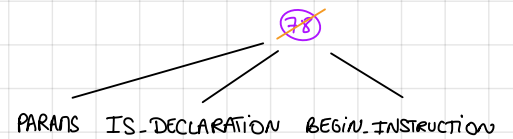
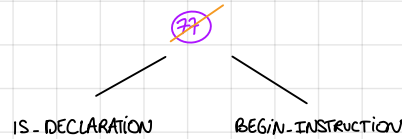
END_GENERATE_EXPRESSION



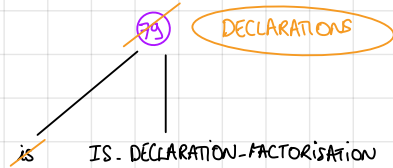
GENERATE_DECLARATIONS_FACTORISATIONS



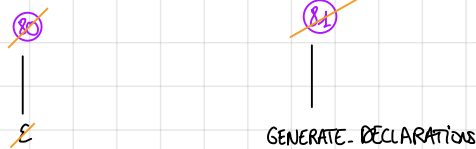
DECLARATION_PROCEDURE



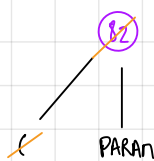
IS_DECLARATION



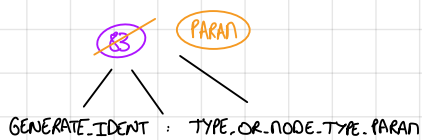
IS_DECLARATION_FACTORISATION



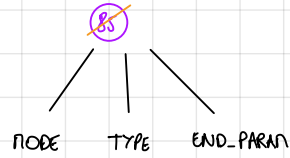
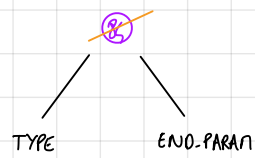
PARANS



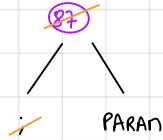
PARAM



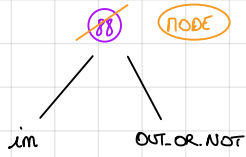
TYPE_OR_NODE_TYPE_PARAMS



END-PARAN



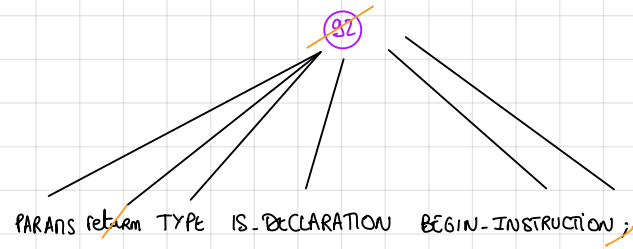
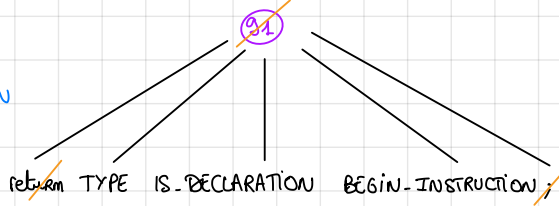
NODE



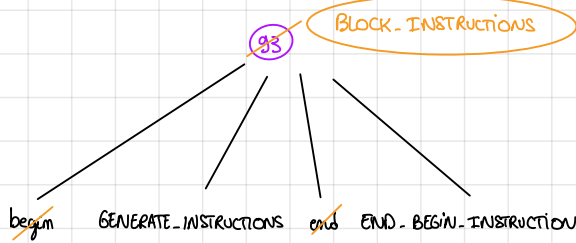
OUT-OR-NOT



DECLARATIONS-FUNCTION



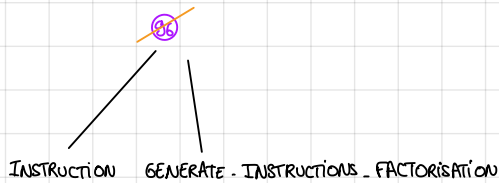
BEGIN-INSTRUCTION



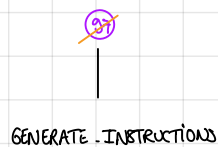
END-BEGIN-INSTRUCTION



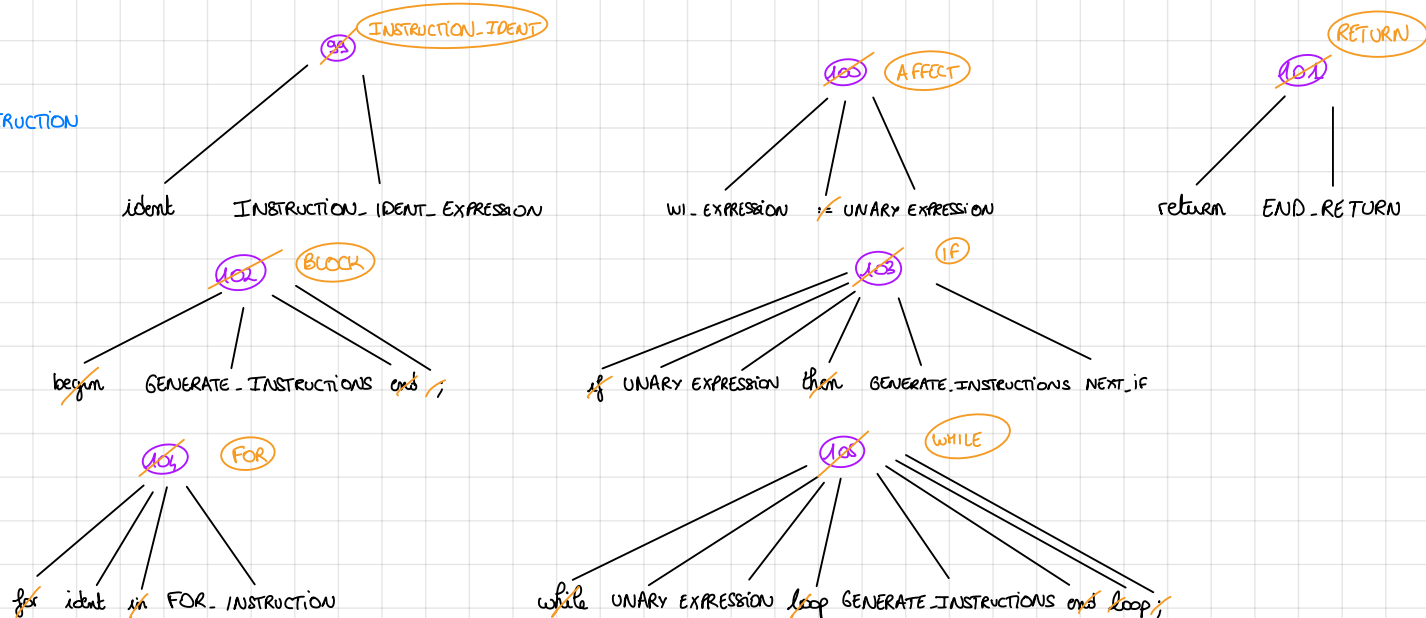
GENERATE-INSTRUCTIONS



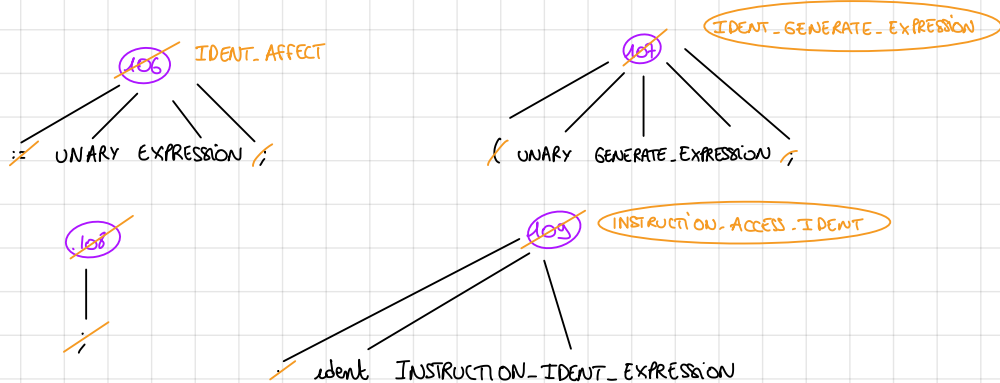
GENERATE-INSTRUCTIONS-FACTORISATION



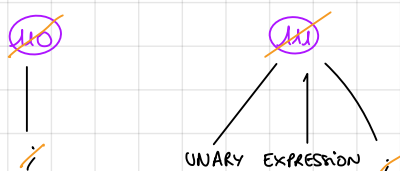
INSTRUCTION



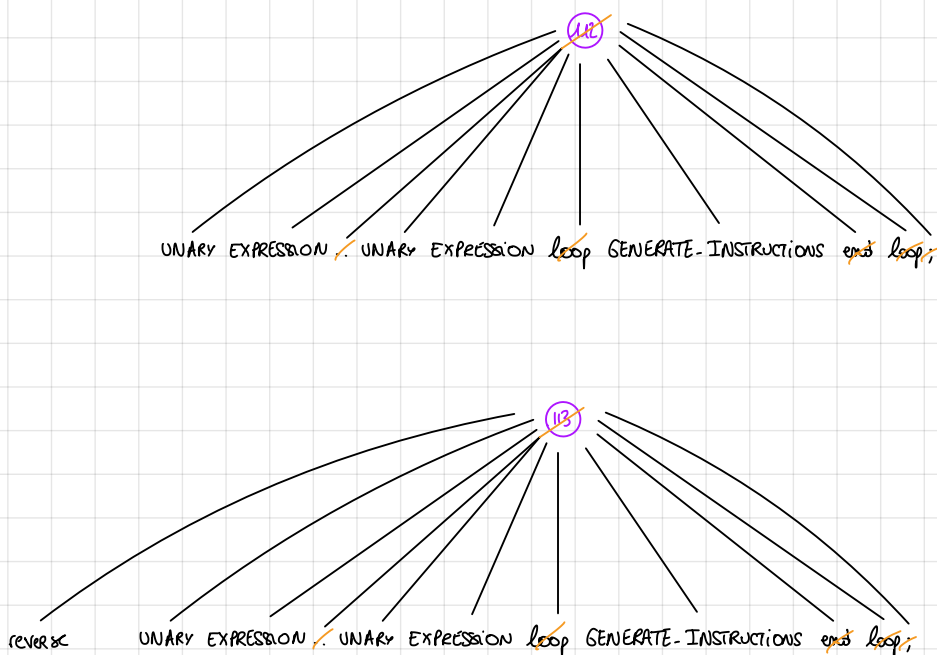
INSTRUCTION-IDENT-EXPRESSION



END-RETURN



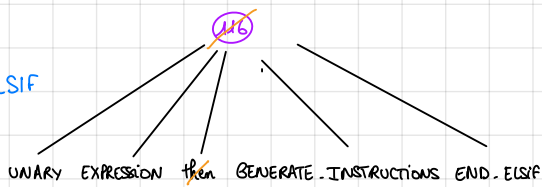
FOR-INSTRUCTION



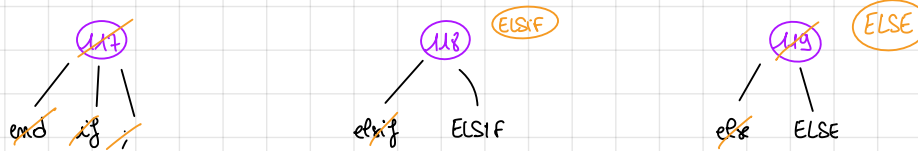
NEXT_IF



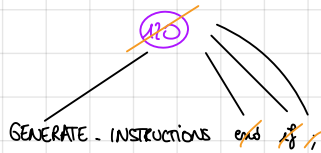
ELSIF



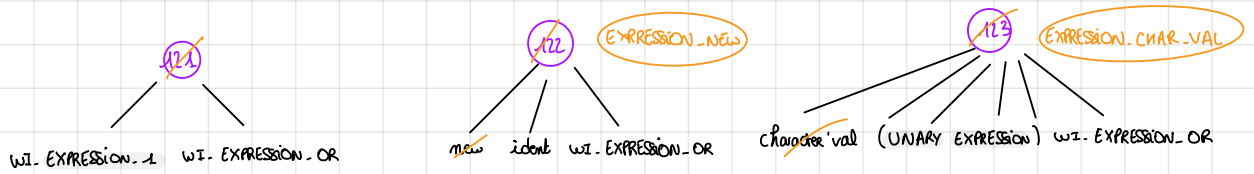
END_ELSIF



ELSE



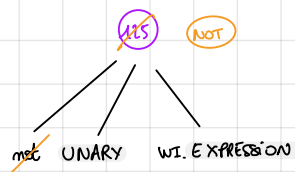
WI_EXPRESSION



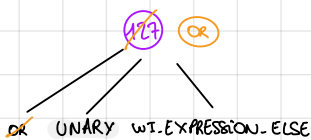
WI_EXPRESSION_1



WI_EXPRESSION_NOT



WI_EXPRESSION_OR



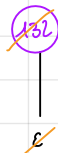
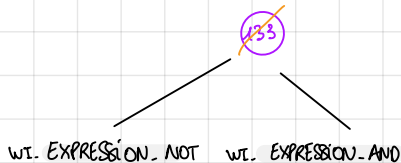
WI_EXPRESSION_ELSE



WI_EXPRESSION_AND



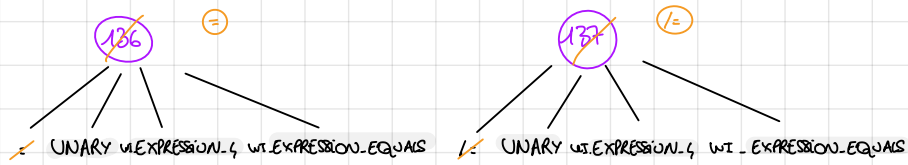
WI_EXPRESSION_THEN



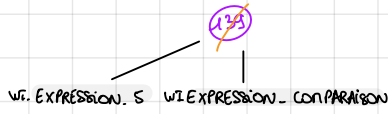
WI-EXPRESSION-3



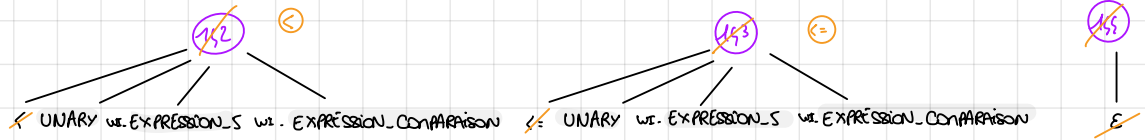
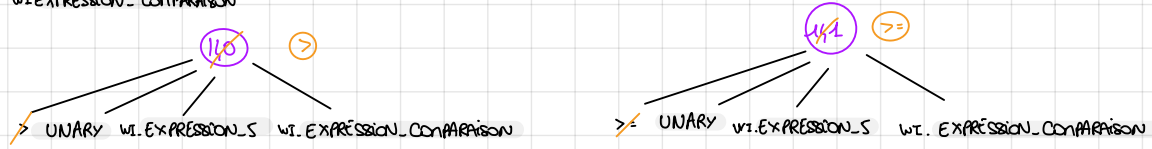
WI-EXPRESSION-EQUALS



WI-EXPRESSION-4



WI-EXPRESSION-COMPARISON



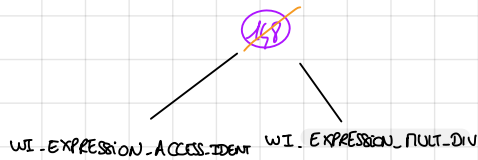
WI-EXPRESSION-5



WI-EXPRESSION-PLUS-MINUS



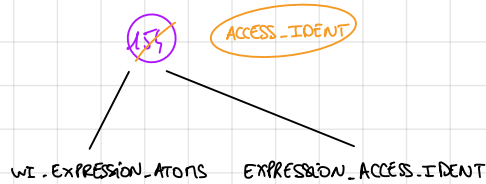
WI-EXPRESSION-6



WI-EXPRESSION-MULT-DIV



WI-EXPRESSION-ACCESS-IDENT



EXPRESSION-ATONS

~~156~~

enter

~~156~~

character

~~157~~

(UNARY GENERATE-EXPRESSION

IDENT-EXPRESSION

~~158~~

leave

~~159~~

false

~~160~~

null