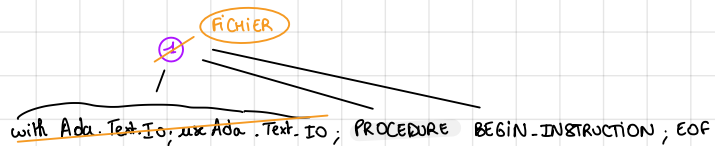
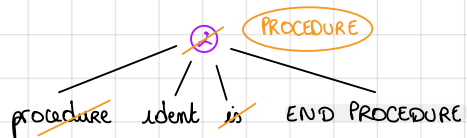


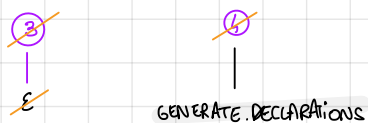
FICHIER



PROCEDURE



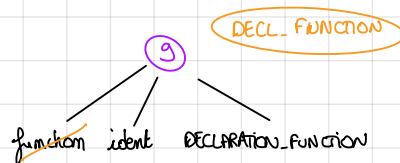
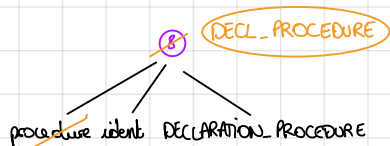
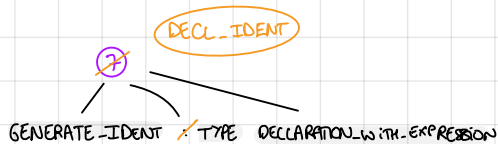
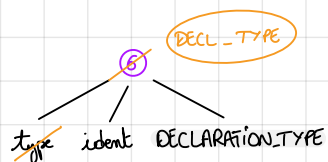
END PROCEDURE



GENERATE-DECLARATIONS



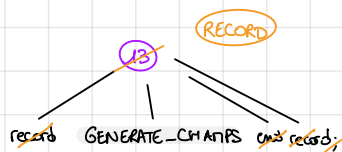
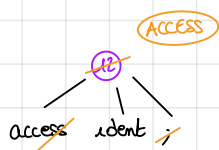
DECLARATION



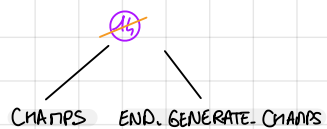
DECLARATION-TYPE



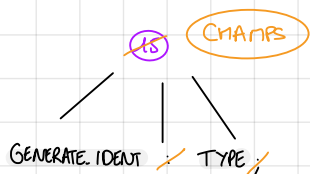
ACCESS-RECORD



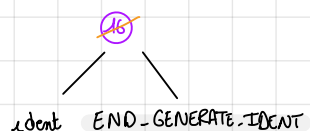
GENERATE-CHAMPS



CHAMPS



GENERATE-IDENT



The diagram shows a non-terminal 'ID' in a purple circle with a diagonal slash. It has two children: a terminal 'i' in a purple circle with a diagonal slash, and the text 'GENERATE_IDENT'.

Diagram illustrating the mapping of node IDs to their parent node types:

- Node 19 (crossed out) is labeled **TYPE-IDENT**. It is connected to the label **ident**.
- Node 20 (crossed out) is labeled **TYPE-ACCESS**. It is connected to the labels **access** and **ident**.

DECL - WITH - EXPRESSION

UNARY EXPRESSION

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 parse tree for the expression `2 + 3 * 4`. The root node is `EXPRESSION.CHAR_VAL` (circled in orange). It branches into six children: `Character_val` (circled in orange), `UNARY_EXPRESSION`, `EXPRESSION_OR`, and three other unnamed nodes. The unnamed nodes further branch into `2`, `+`, `3`, `*`, and `4`.

```

graph TD
    A[UNARY EXPRESSION] --> B[NOT]
    A --> C[UNARY EXPRESSION]
    C --> D["3 + 4 * 5"]
    C --> E[UNARY EXPRESSION]
    E --> F[5]
    
```

32
|
EXPRESSION-3

```

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

~~37~~

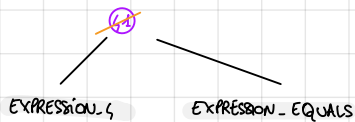
Diagram illustrating the OR-ELSE operator precedence. The root node is a purple circle containing '36' with a diagonal slash. It branches into four children: 'else', 'UNARY', 'EXPRESSION_1', and 'EXPRESSION_OR'. An orange oval labeled 'OR-ELSE' is positioned above the 'EXPRESSION_OR' branch.

Diagram illustrating the evaluation of the expression "AND 37". The word "AND" is circled in orange. The value "37" is circled in purple. A diagonal line is drawn through the "37", indicating it is the operand of the AND operation.

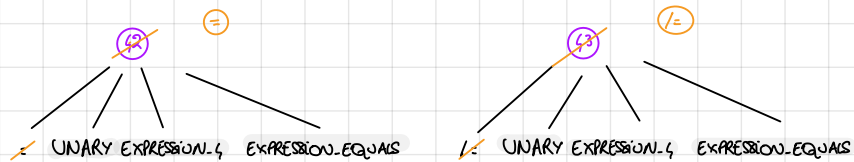
~~3~~
|
~~£~~

then UNARY EXPRESSION NOT EXPRESSION AND

EXPRESSION_3



EXPRESSION_EQUALS



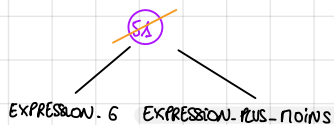
EXPRESSION_4



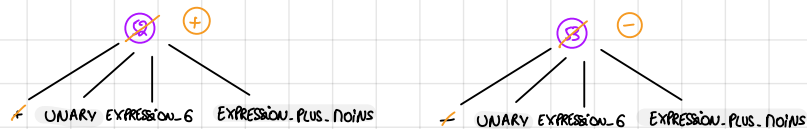
EXPRESSION_COMPARISON



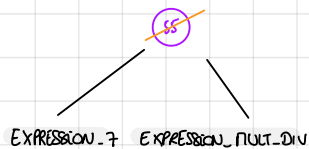
EXPRESSION_5



EXPRESSION_PLUS_MINUS



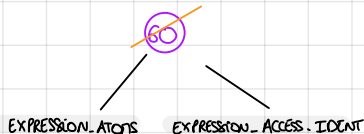
EXPRESSION_6



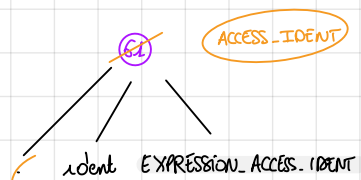
EXPRESSION_MULT_DIV



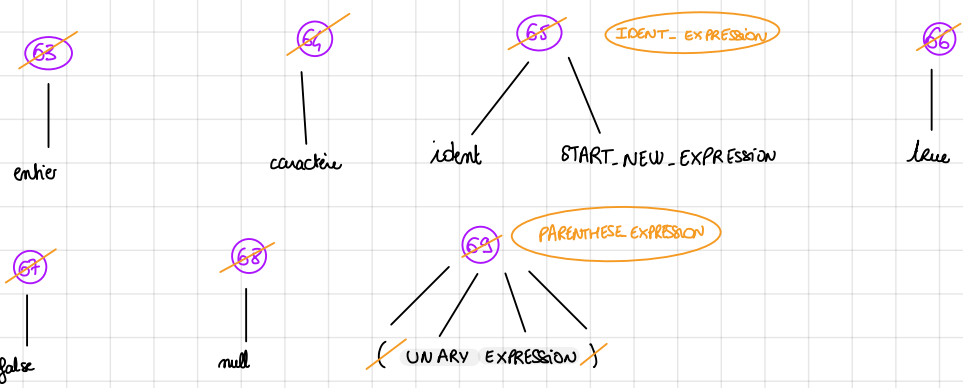
EXPRESSION_7



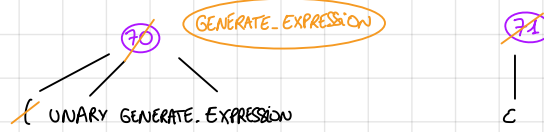
EXPRESSION_ACCESS_IDENT



EXPRESSION_ATOMS



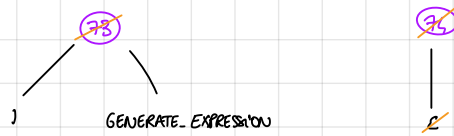
START_NEW_EXPRESSION



GENERATE_EXPRESSION



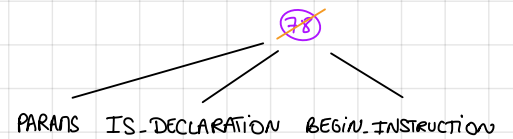
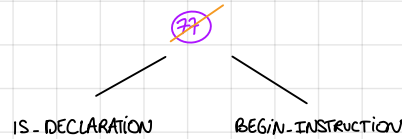
END_GENERATE_EXPRESSION



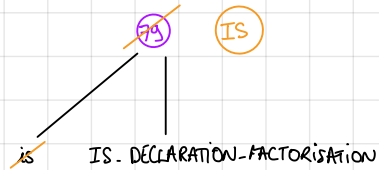
GENERATE_DECLARATIONS_FACTORISATIONS



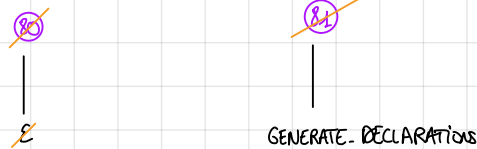
DECLARATION_PROCEDURE



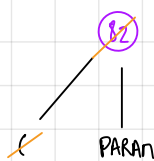
IS_DECLARATION



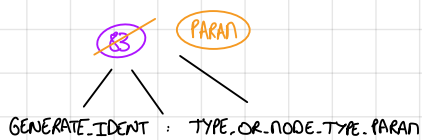
IS_DECLARATION_FACTORISATION



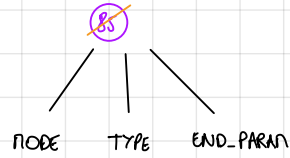
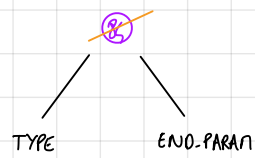
PARAMS



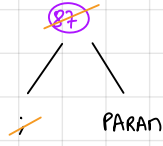
PARAM



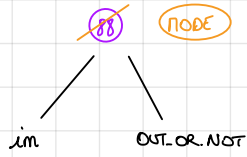
TYPE_OR_NODE_TYPE_PARAMS



END-PARAN



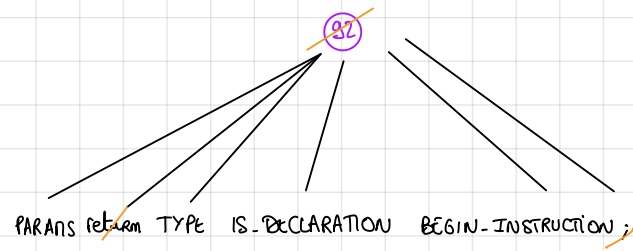
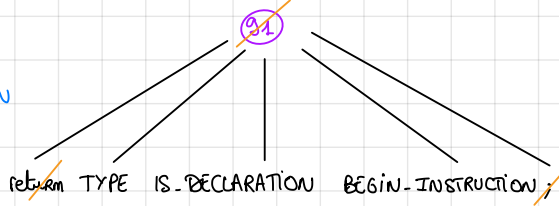
NODE



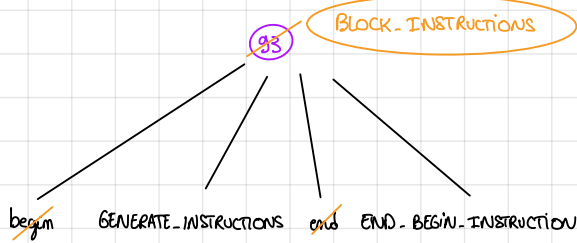
OUT-OR-NOT



DECLARATIONS-FUNCTION



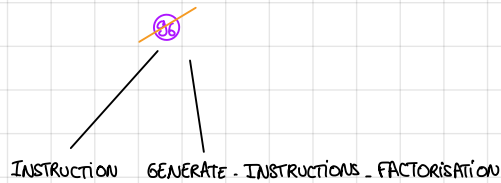
BEGIN-INSTRUCTION



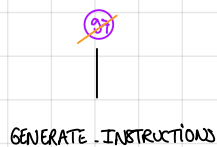
END-BEGIN-INSTRUCTION



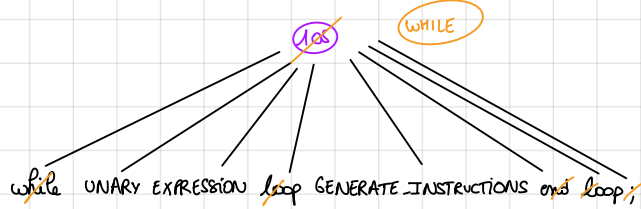
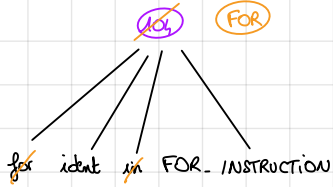
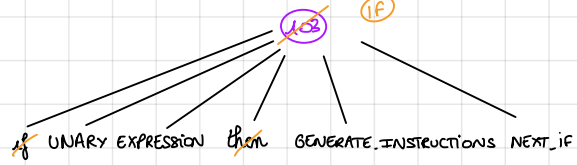
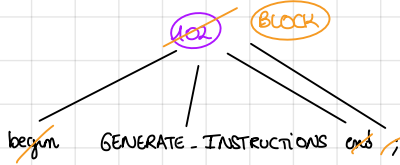
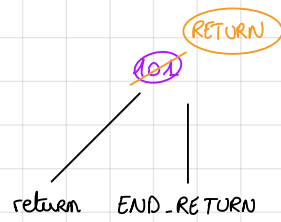
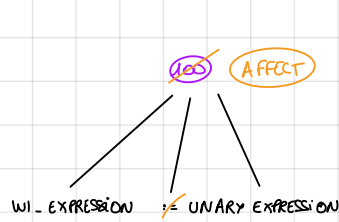
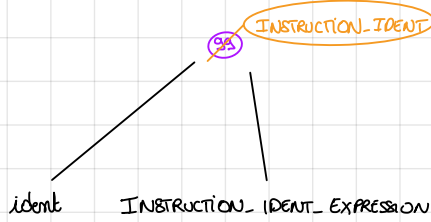
GENERATE-INSTRUCTIONS



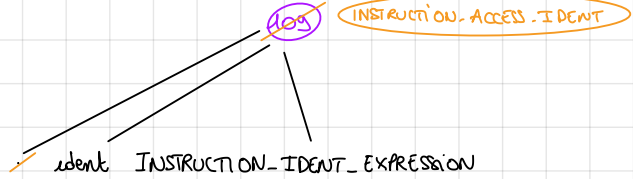
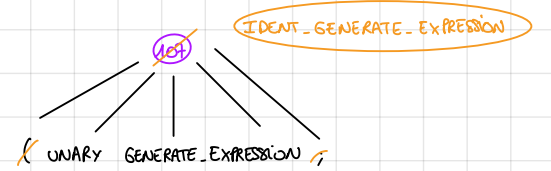
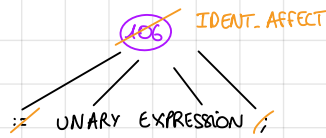
GENERATE-INSTRUCTIONS-FACTORISATION



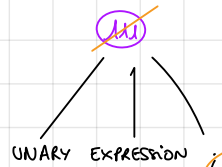
INSTRUCTION



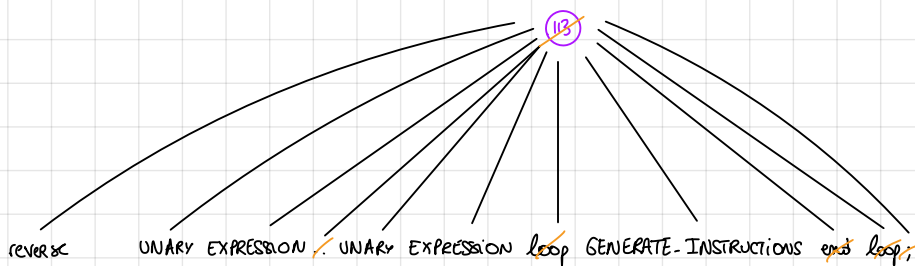
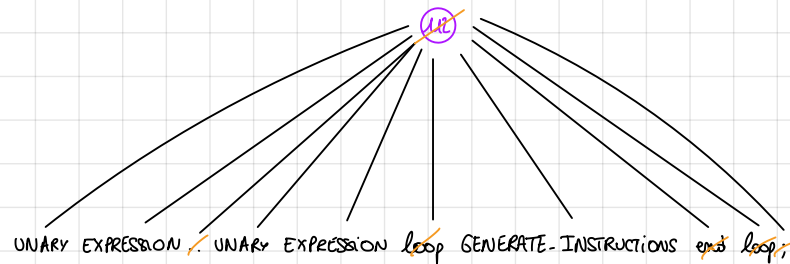
INSTRUCTION_IDENT_EXPRESSION



END_RETURN



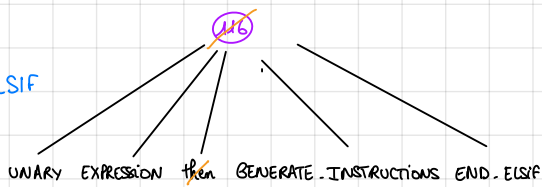
FOR_INSTRUCTION



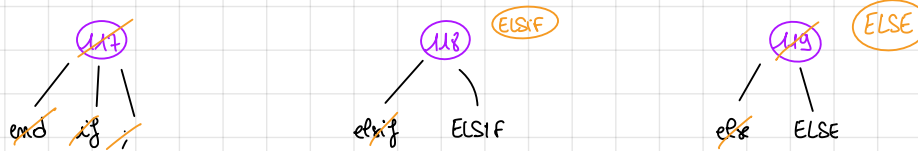
NEXT_IF



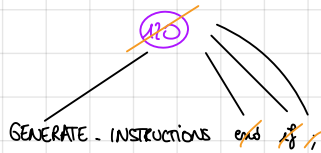
ELSIF



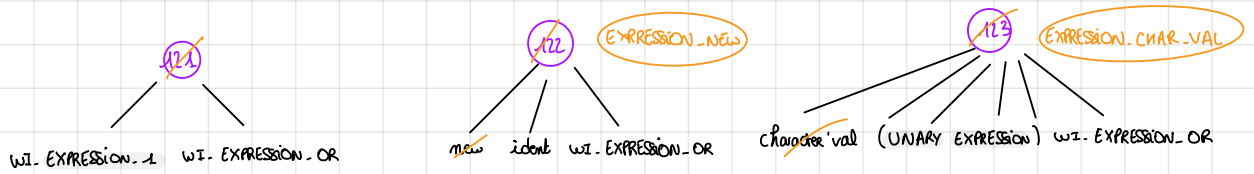
END_ELSIF



ELSE



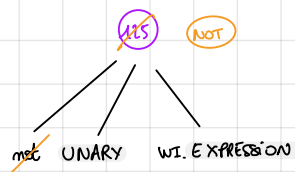
WI_EXPRESSION



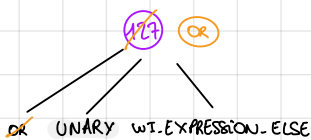
WI_EXPRESSION_1



WI_EXPRESSION_NOT



WI_EXPRESSION_OR



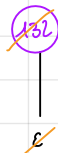
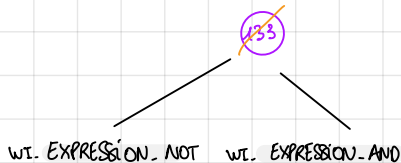
WI_EXPRESSION_ELSE



WI_EXPRESSION_AND



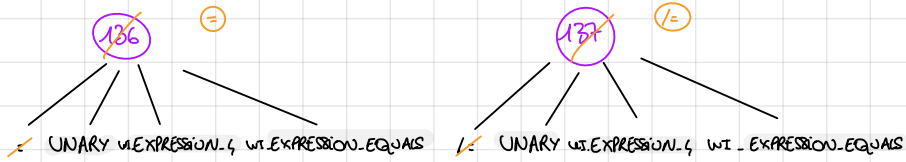
WI_EXPRESSION_THEN



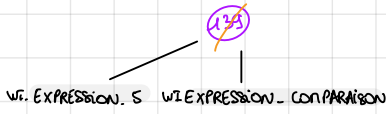
WI-EXPRESSION-3



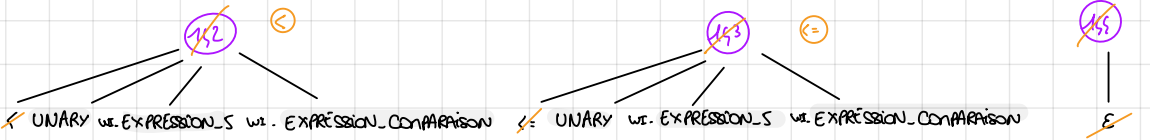
WI-EXPRESSION-EQUALS



WI-EXPRESSION-4



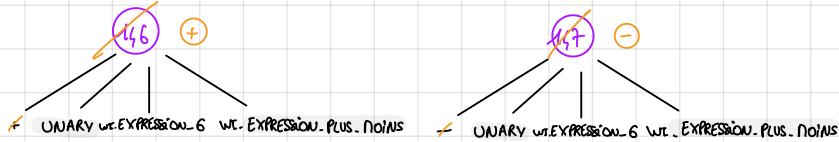
WI-EXPRESSION-COMPARISON



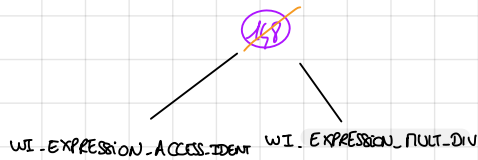
WI-EXPRESSION-5



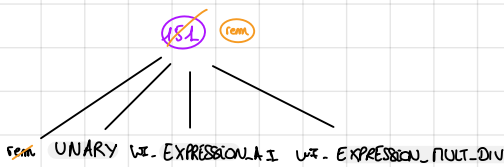
WI-EXPRESSION-PLUS-MINUS



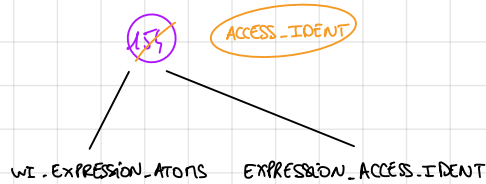
WI-EXPRESSION-6



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