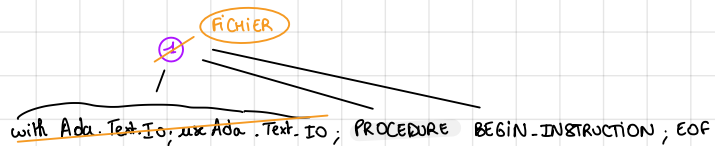
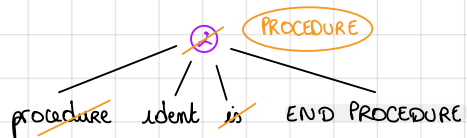


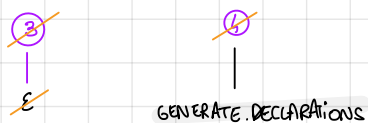
FICHIER



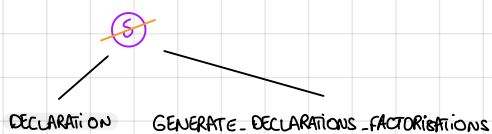
PROCEDURE



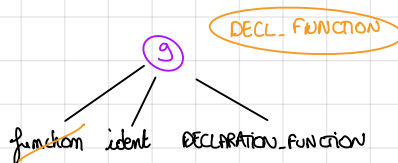
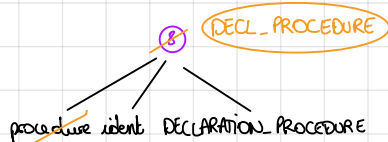
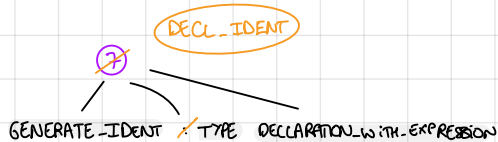
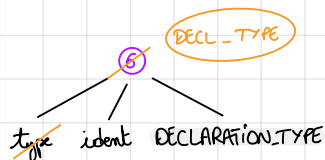
END PROCEDURE



GENERATE-DECLARATIONS



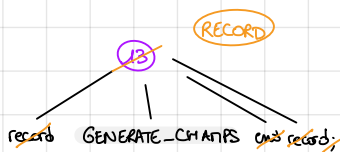
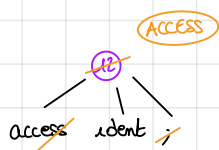
DECLARATION



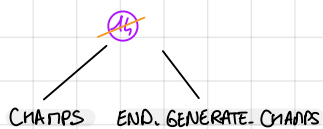
DECLARATION-TYPE



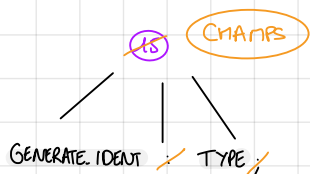
ACCESS-RECORD



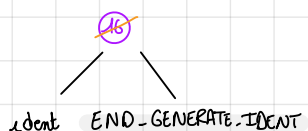
GENERATE-CHAMPS



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.

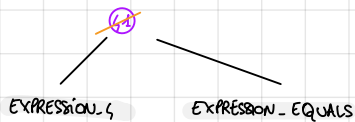
Diagram illustrating the evaluation of the expression "AND 37". The word "AND" is circled in orange. The value "37" is circled in purple. A diagonal line is drawn through the "37".

~~3~~  
|  
~~£~~

[illegible]

then UNARY EXPRESSION NOT EXPRESSION AND

EXPRESSION\_3



EXPRESSION\_EQUALS



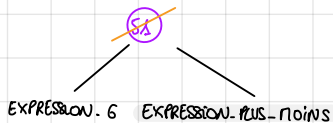
EXPRESSION\_4



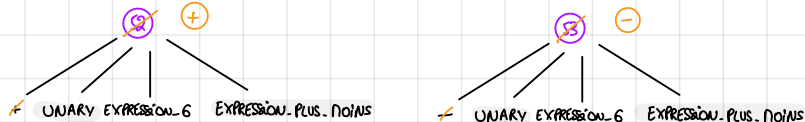
EXPRESSION\_COMPARISON



EXPRESSION\_5



EXPRESSION\_PLUS\_MINUS



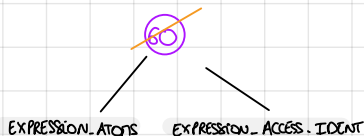
EXPRESSION\_6



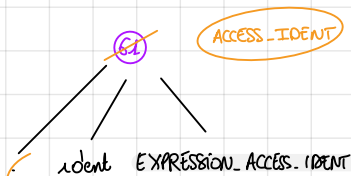
EXPRESSION\_MULT\_DIV



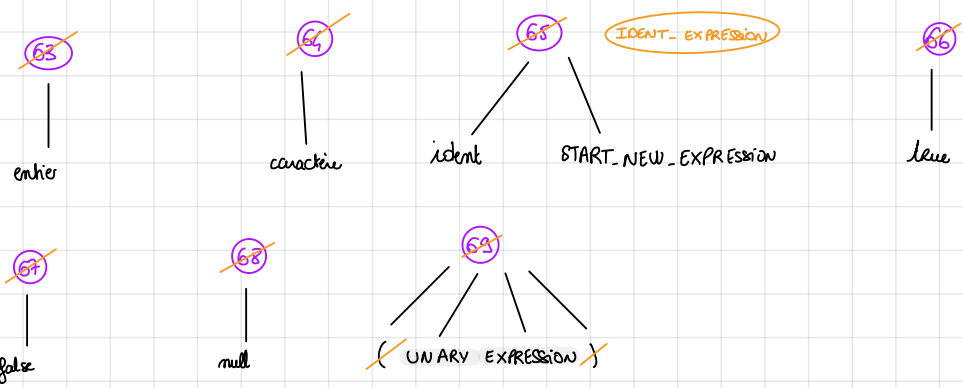
EXPRESSION\_7



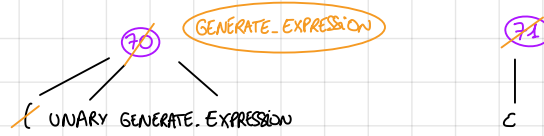
EXPRESSION\_ACCESS\_IDENT



## EXPRESSION\_ATOMS



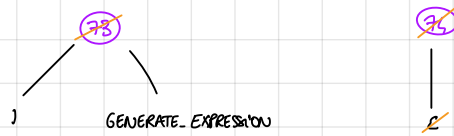
## START\_NEW\_EXPRESSION



## GENERATE\_EXPRESSION



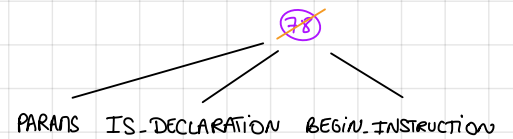
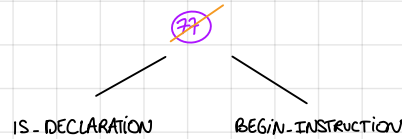
## END\_GENERATE\_EXPRESSION



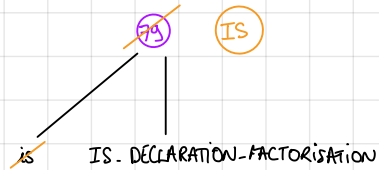
## GENERATE\_DECLARATIONS\_FACTORISATIONS



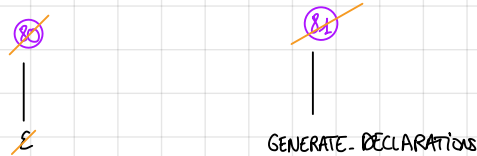
## DECLARATION\_PROCEDURE



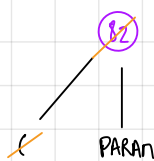
## IS\_DECLARATION



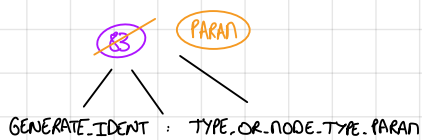
## IS\_DECLARATION\_FACTORISATION



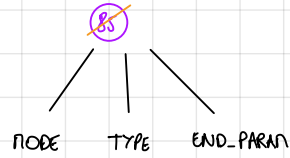
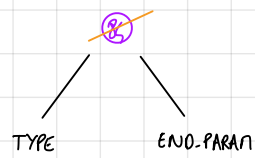
## PARAMS



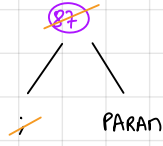
## PARAM



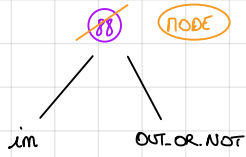
TYPE\_OR\_NODE\_TYPE\_PARAMS



END-PARAN



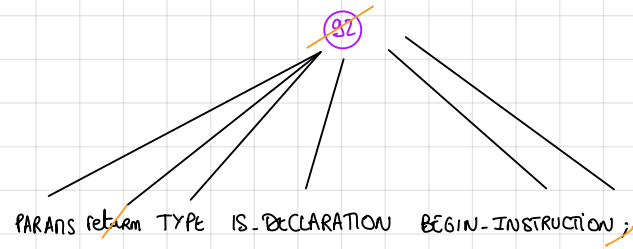
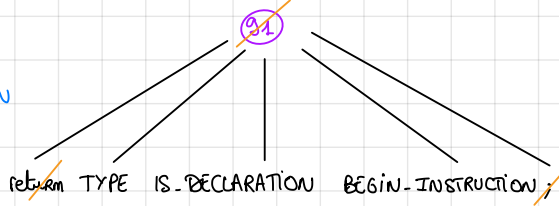
NODE



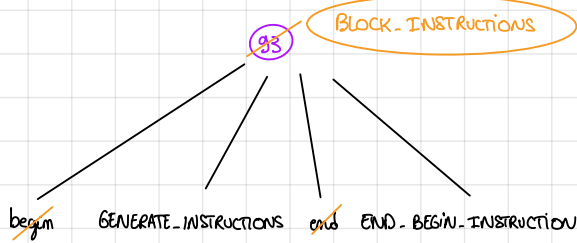
OUT-OR-NOT



DECLARATIONS-FUNCTION



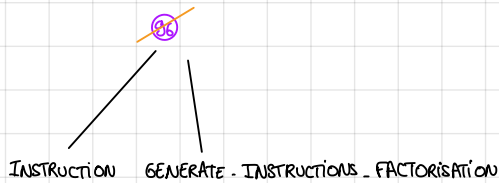
BEGIN-INSTRUCTION



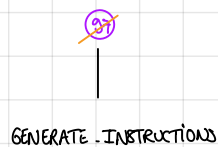
END-BEGIN-INSTRUCTION



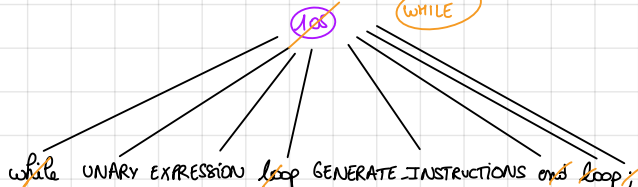
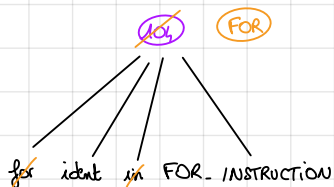
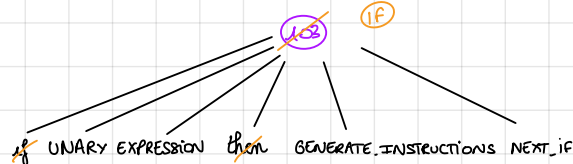
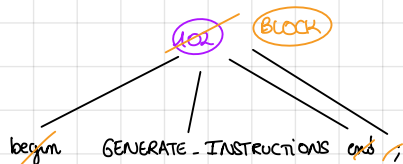
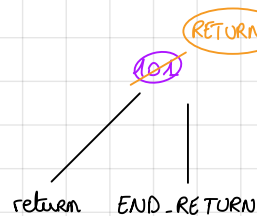
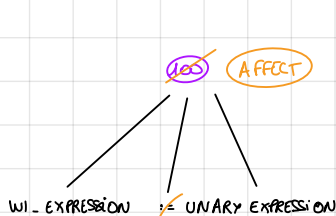
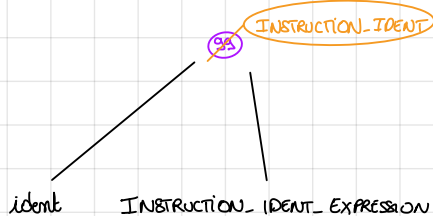
GENERATE-INSTRUCTIONS



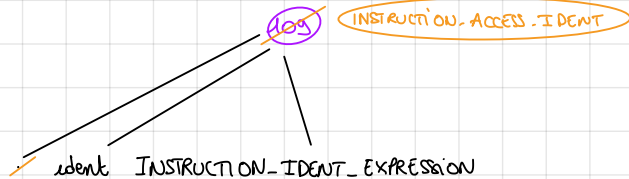
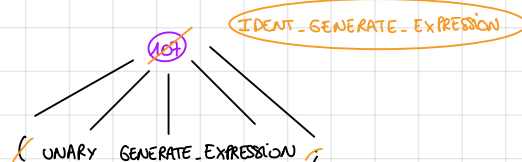
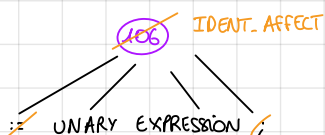
GENERATE-INSTRUCTIONS-FACTORISATION



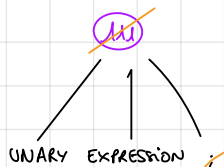
INSTRUCTION



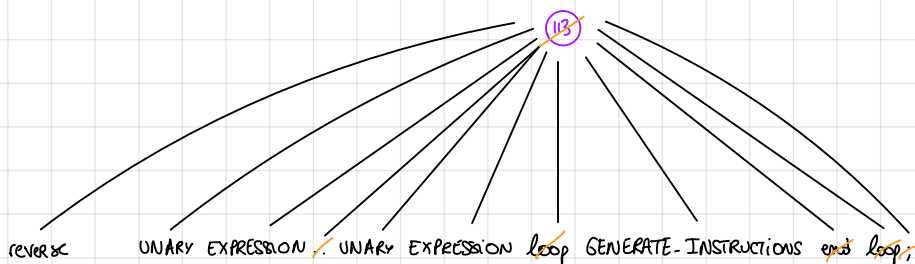
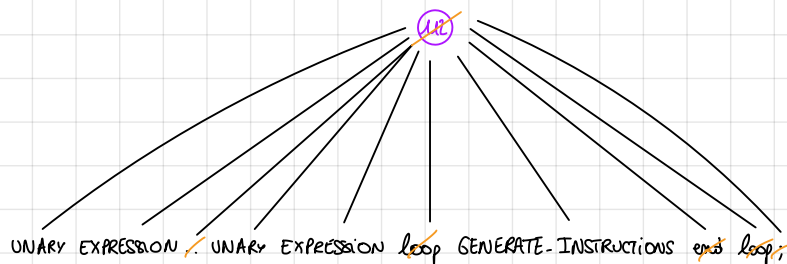
INSTRUCTION\_IDENT\_EXPRESSION



END\_RETURN



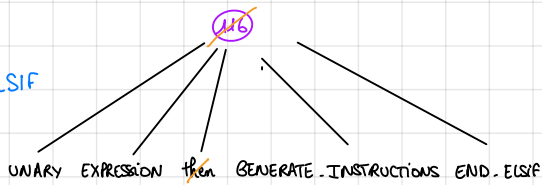
FOR\_INSTRUCTION



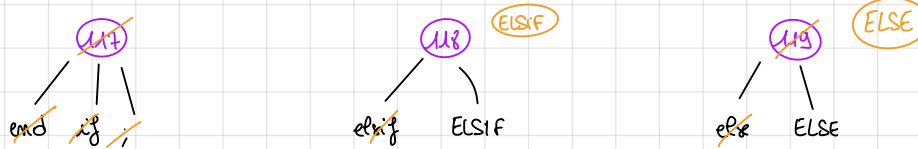
NEXT\_IF



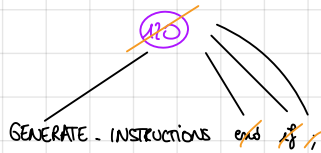
ELSIF



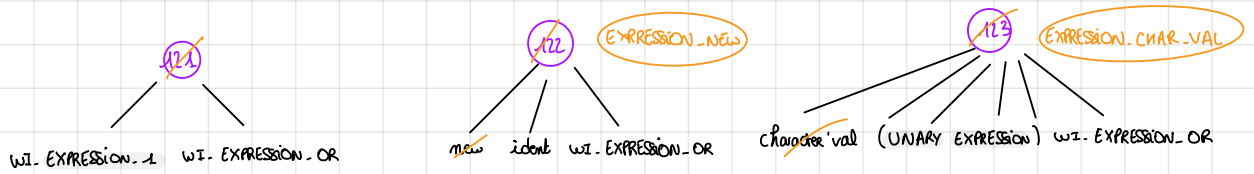
END\_ELSIF



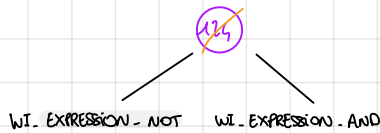
ELSE



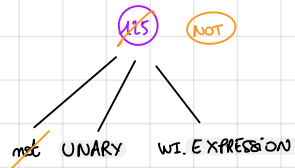
WI\_EXPRESSION



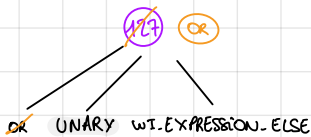
WI\_EXPRESSION\_1



WI\_EXPRESSION\_NOT



WI\_EXPRESSION\_OR



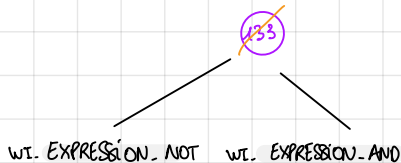
WI\_EXPRESSION\_ELSE



WI\_EXPRESSION\_AND



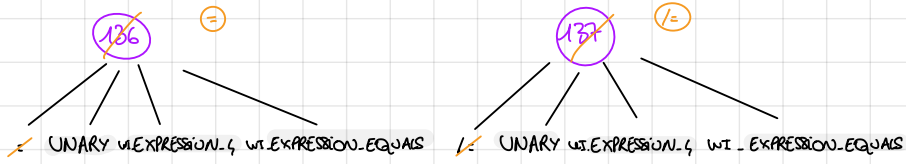
WI\_EXPRESSION\_THEN



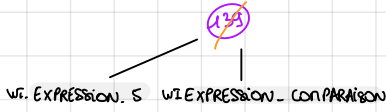
WI-EXPRESSION-3



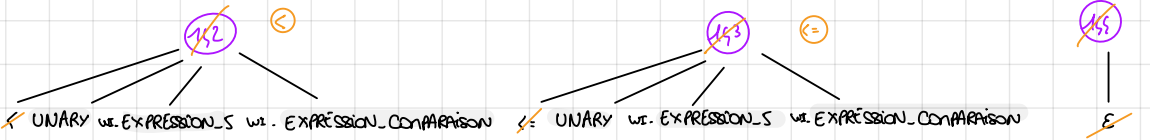
WI-EXPRESSION-EQUALS



WI-EXPRESSION-4



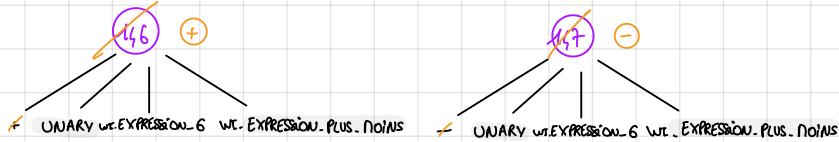
WI-EXPRESSION-COMPARISON



WI-EXPRESSION-5



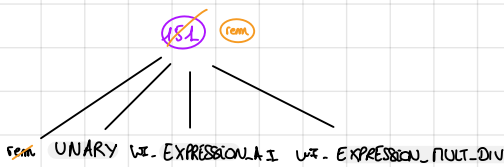
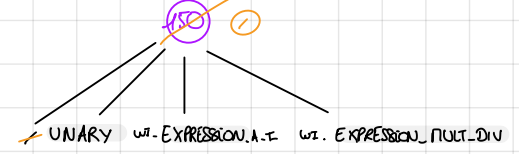
WI-EXPRESSION-PLUS-MINUS



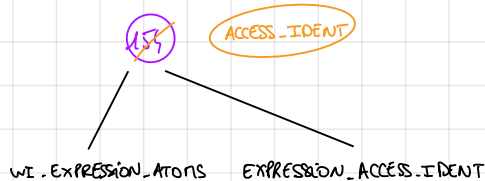
WI-EXPRESSION-6



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