

As shown in the table below, we identified seven merging rules for type-based graph simplification:

TABLE I: Merging rules in type-based graph simplification. “PType” and “CType” indicate the types of the parent and child nodes of a directed edge, respectively. The * represents any node type, and the PType “Augment” in Rule 6 represents the parameters of function call statements.

Rule	PType	CType
1	ExpressionStatement	AssignmentExpression UnaryExpression CallExpression PostIncDecOperationExpression
2	IdentifierDeclStatement	IdentifierDecl
3	Condition	*
4	ForInit	*
5	CallExpression	ArgumentList
6	Argument	*
7	Callee	Identifier

For each rule in the table (except rule 2, which has an example in our paper), we give a detailed example. In the following graphs, each node has two attributes: the first line of text in the node means the value and the second line of text in the node means the type.

Rule 1

(ExpressionStatement, AssignmentExpression)

Figure 1.1 is the AST of statement “commandLength = strlen (cat) + strlen (argv [1]) + 1”. According to rule 1, node 2 is merged into node 1. The value of node 2 is repetitive with the value of node 1, and the value information of node 2 is also reflected in its descendant nodes.

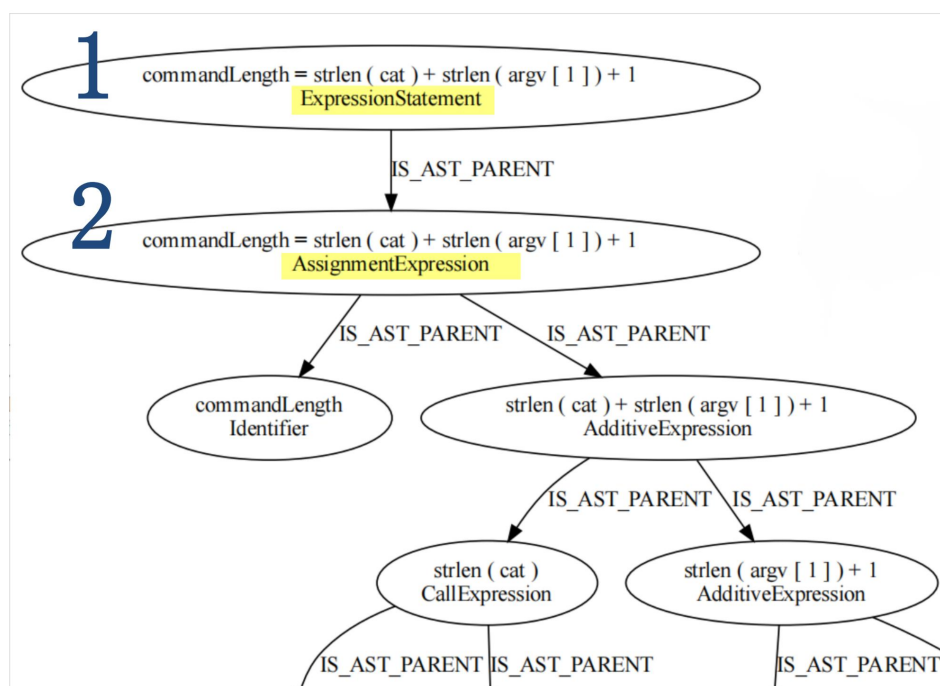


Figure 1.1. Rule 1: (ExpressionStatement, AssignmentExpression)

(ExpressionStatement, UnaryExpression)

Figure 1.2 is the AST of statement “++i;”. Node 2 is merged into node 1.

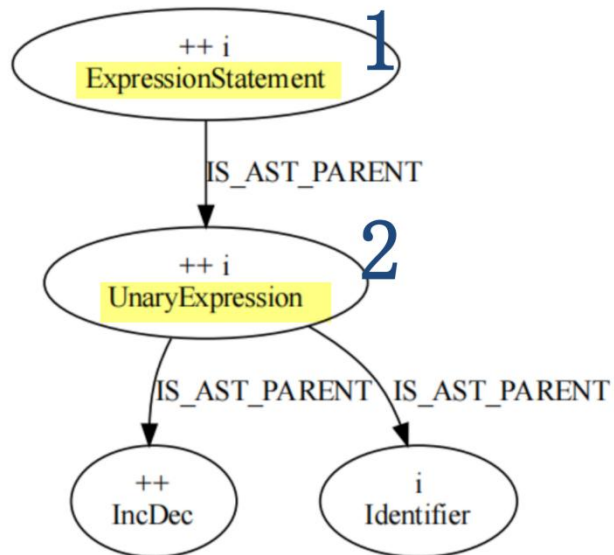


Figure 1.2. Rule 1: (ExpressionStatement, UnaryExpression)

(ExpressionStatement, CallExpression)

Figure 1.3 is the AST of statement “strncat(command, argv[1], (commandLength - strlen(cat)))”. According to rule 1, node 2 is merged into node 1.

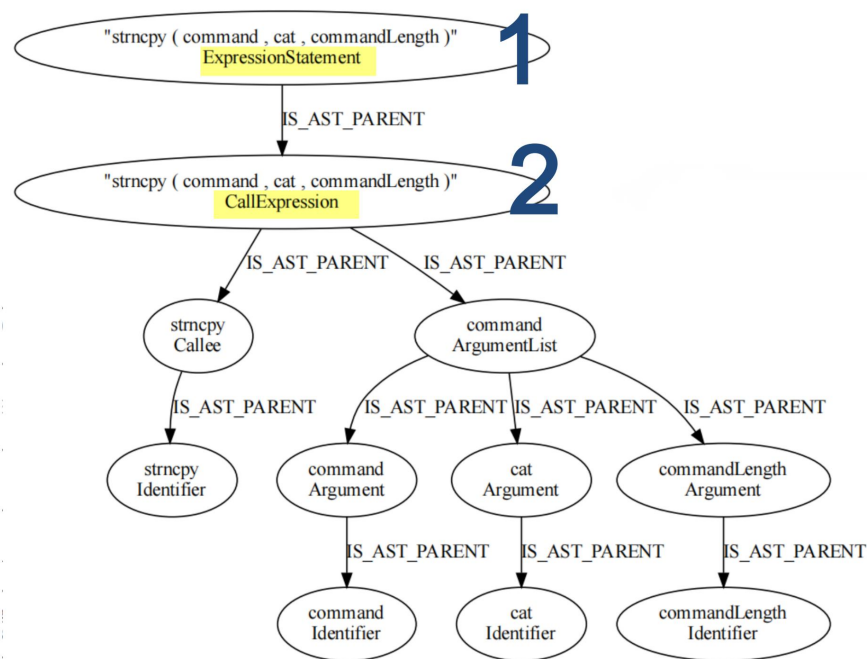


Figure 1.3. Rule 1: (ExpressionStatement, CallExpression)

(ExpressionStatement, PostIncDecOperationExpression)

Figure 1.4 is the AST of statement “i++;”. Node 2 is merged into node 1.

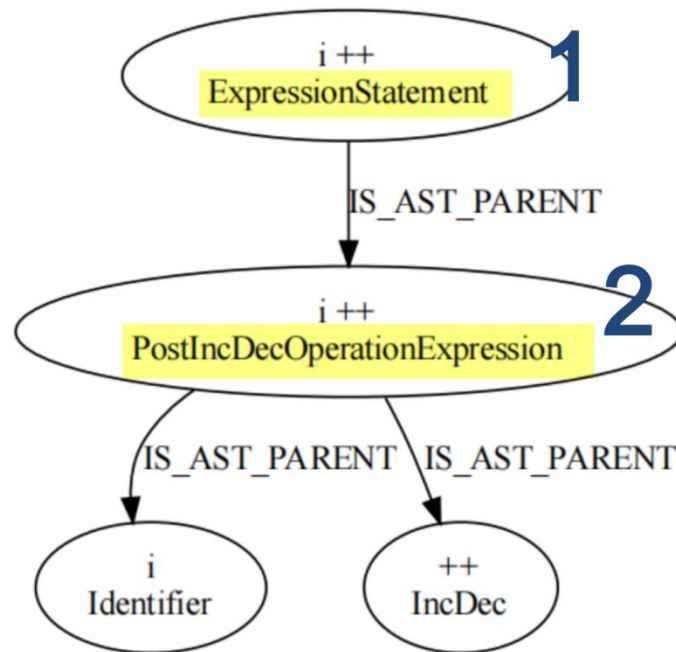


Figure 1.4. Rule 1: (ExpressionStatement, PostIncDecOperationExpression)

Rule 3 (Condition, RelationalExpression)

Figure 3 is the AST of statement “if(i < strlen(user_supplied_string))”, and node 2 is merged into node 1. The value of node 2 is repetitive with the value of node 1, and the value information of node 2 is also reflected in its descendant nodes.

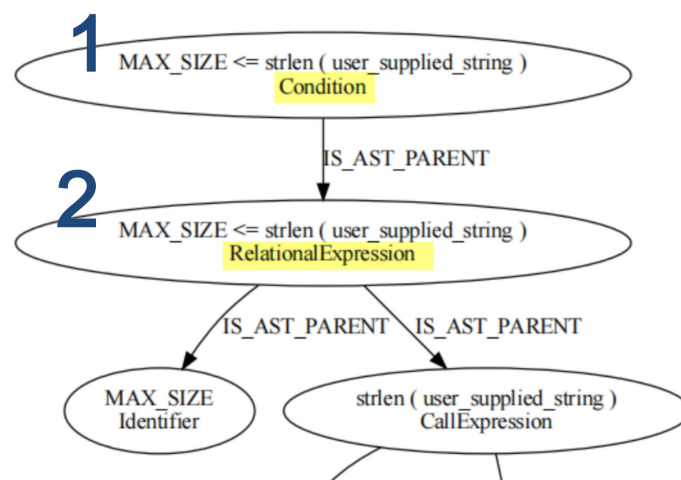


Figure 3. Rule 3: (Condition, RelationalExpression)

Rule 4 (ForInit, *)

As shown in Figure 4, the AST in the red line is about the statement “i=0” which is contained in the for-loop statement “for (i = 0; i < 10; i++){}” or “for(i=0;i<10;){}” or “for(i=0;;){}”. Node 2 is merged into node 1.

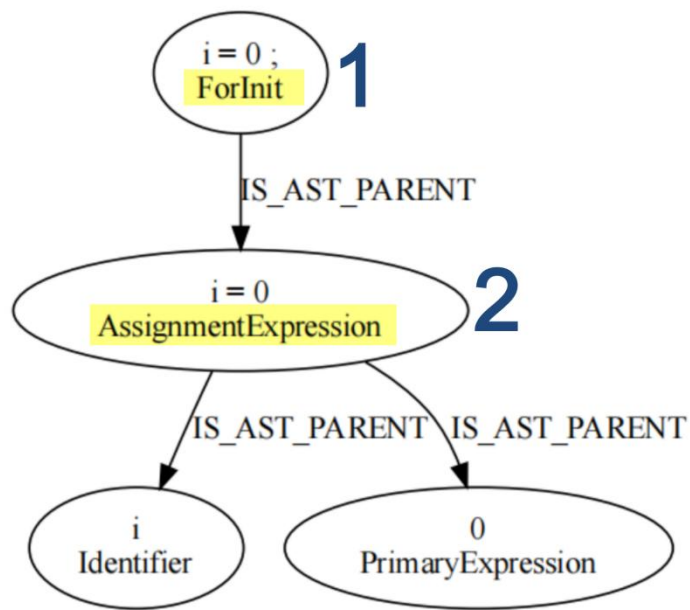


Figure 4. Rule 4: (ForInit, *)

Rule 5 (CallExpression, ArgumentList)

As shown in Figure 5, node 2 is merged into node 1. Although the type of node 2 is “ArgumentList” that means the parameter list, the value of node 2 only contains the first parameter of the function “strcpy”. And the three child nodes of node 2: node 3-5 represents the three parameters of the function “strcpy” respectively. So we merge the node 2 into node 1.

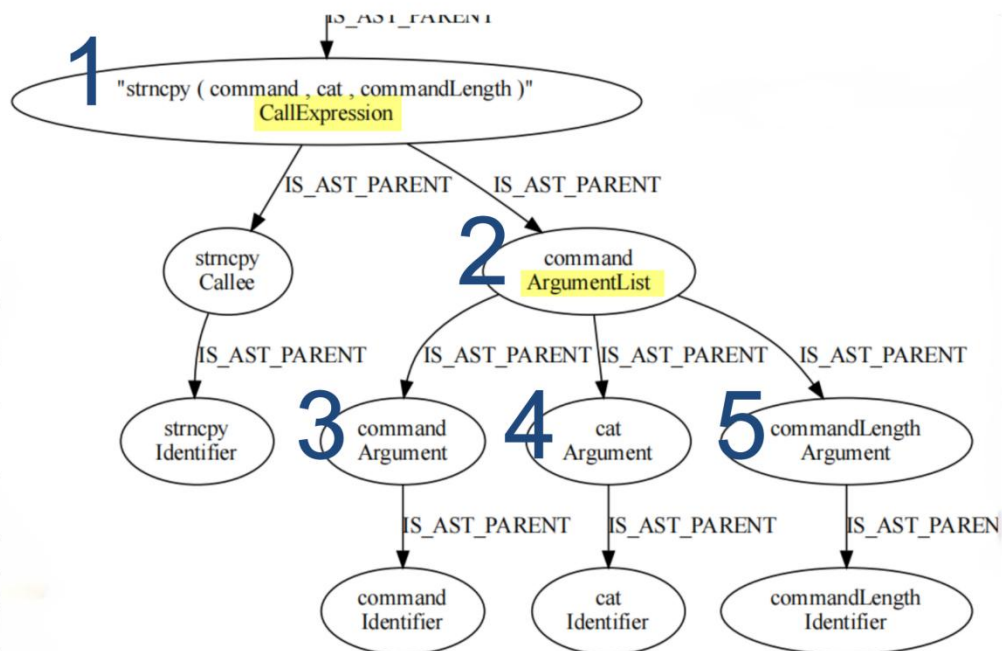


Figure 5. Rule 5: (CallExpression, ArgumentList)

Rule 6 (Argument, *)

As shown in Figure 6, nodes 1,3,5 are merged into nodes 2,4,6 respectively.

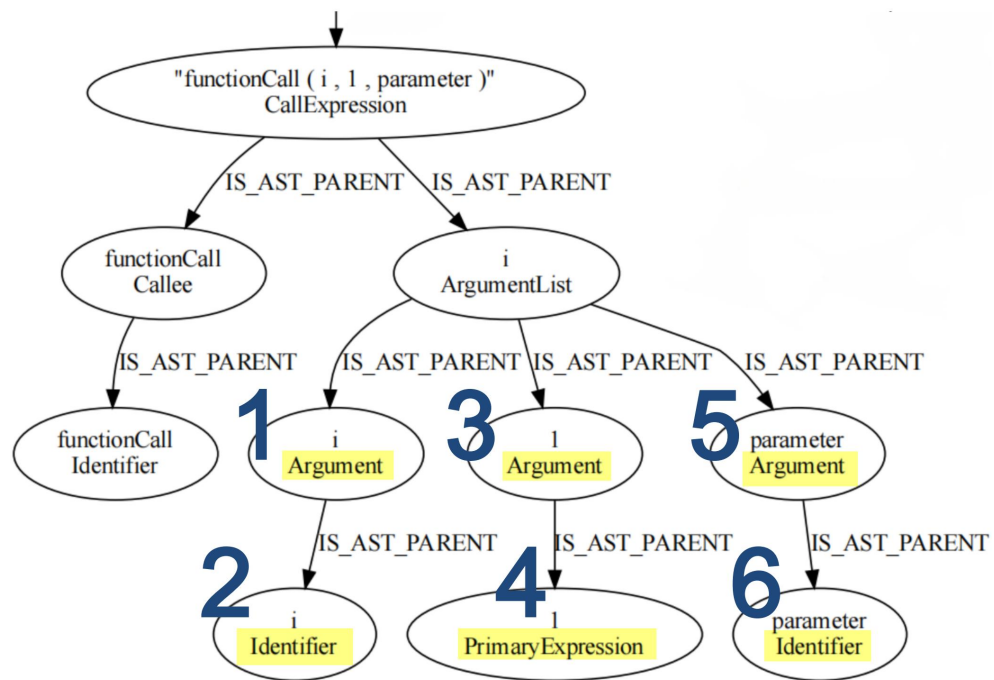


Figure 6. Rule 6: (Argument, *)

Rule 7 (Callee, Identifier)

As shown in Figure 7, node 2 is merged into node 1.

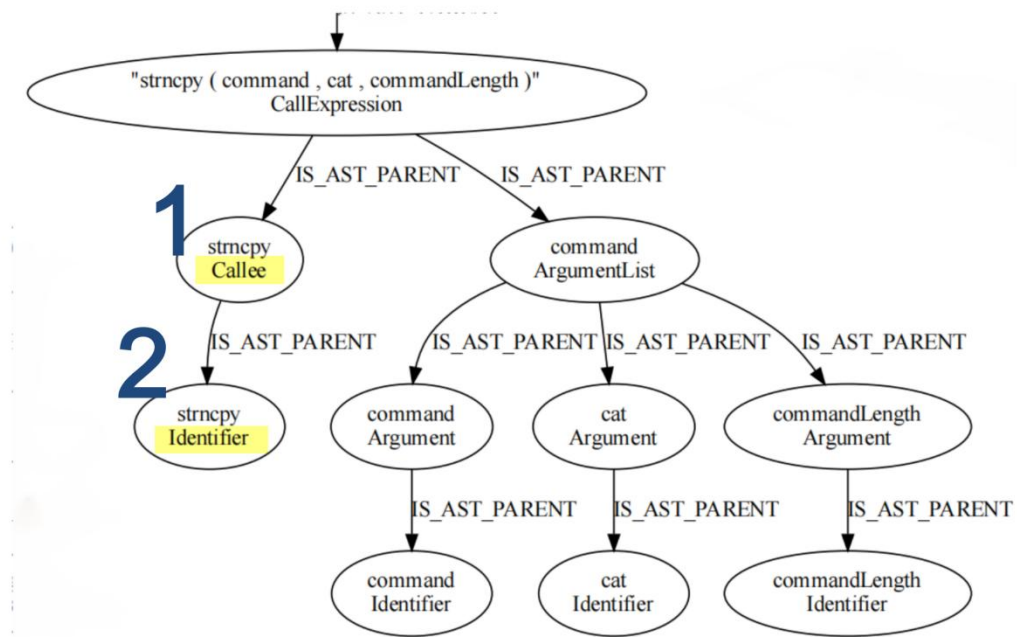


Figure 7