

# Coccinelle Explorer: Learning Semantic Patching Interactively

Michele MARTONE

Leibniz Supercomputing Centre, Garching near Munich, Germany

Kernel devroom at FOSDEM'25  
Bruxelles, February 02, 2025



## Coccinelle: a program matching and transformation engine

- ▶ own C/C++ parser, own AST
- ▶ matches language entities
- ▶ matches control-flow
- ▶ can patch what is matched

All together is better!!!



COCCINELLE  
*inside*

## make cocccheck – static code analysis


```
scripts/  
├── coccinelle/  
│   ├── api/  
│   │   └── ...  
│   ├── free/  
│   │   ├── clk_put.cocci  
│   │   ├── devm_free.cocci  
│   │   ├── ifnulldev_put.cocci  
│   │   ├── ifnullfree.cocci  
│   │   ├── iounmap.cocci  
│   │   ├── kfree.cocci  
│   │   └── ...  
│   └── api/  
│       └── ...
```

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/>


Develop HPC-oriented changes

- ▶ advanced `#pragma` manipulation  
(`OPENMP`, `OPENACC`, ...)
- ▶ for GPU
- ▶ C++-related
- ▶ removal of *bloat* and *cruff*

Popularization/trainings, helping HPC users

C++ source #1 A ▾  + ▾     C++ ▾


```
1 // Type your code here, or load an example
2 int square(int num) {
3     return num * num;
4 }
```

x86-64 gcc 14.2 (Editor #1)  Xx86-64 gcc 14.2 ▾   Compiler opti ▾A ▾  ▾   + ▾ 

```
1 square(int):
2     push    rbp
3     mov     rbp, rsp
4     mov     DWORD PTR [rbp-4], edi
5     mov     eax, DWORD PTR [rbp-4]
6     imul   eax, eax
7     pop     rbp
8     ret
```

  Output (0/0) x86-64 gcc 14.2  - cached (2811B) ~170 lines filtered 

Compiler License

Rust source #1 


 A ▾  + ▾ 



Rust ▾

```

1 // Type your code here, or load an example
2
3 // As of Rust 1.75, small functions are
4 // marked as `#[inline]` so they will not
5 // the output when compiling with optimis
6 // `#[no_mangle]` or `#[inline(never)]`
7 // this issue.
8 // See https://github.com/compiler-explor
9 #[no_mangle]
10 pub fn square(num: i32) -> i32 {
11     num * num
12 }
13
14 // If you use `main()`, declare it as `pub
15 // pub fn main() { ... }
16

```

 rustc 1.84.0 (Editor #1) 

 rustc 1.84.0 ▾  




Compiler opti ▾

 A ▾     + ▾ 

```

1 square:
2     push    rax
3     mov     dword ptr [rsp + 4], edi
4     imul   edi, edi
5     mov     dword ptr [rsp], edi
6     seto   al
7     jo     .LBB0_2
8     mov     eax, dword ptr [rsp]
9     pop     rcx
10    ret
11 .LBB0_2:
12     lea    rdi, [rip + .L__unnamed_
13     mov     rax, qword ptr [rip + co
14     call   rax
15
16 .L__unnamed_2:

```

  Output (0/0) rustc 1.84.0  - cached (3324B) ~180 lines filtered 

Compiler License

## Compiler Explorer

- ▶ <https://godbolt.org/>, online resource
- ▶ mostly TypeScript
- ▶ useful learning tool
- ▶ open to new *compilers* and *languages*



## Compiler Explorer

- ▶ <https://godbolt.org/>, online resource
- ▶ mostly TypeScript
- ▶ useful learning tool
- ▶ open to new *compilers* and *languages*
- ▶ so... why not Coccinelle?

C with Coccinelle source #1

A ▾ [Icons] + ▾ v

C with Coccinelle ▾

```

1 // Write your C program in this section:
2 int main(){ return 0; }
3
4 #ifdef COCCINELLE // this keyword should
5 // Write your Coccinelle 🐞 rules in this
6 @patch@
7 @@
8 - 0
9 + 1
10 #endif /* COCCINELLE */
11

```

Spatch (Editor #1)

Spatch

Compiler opti ▾

A ▾ [Icons] + ▾

```

1 // Write your C program in this section:
2 int main(){ return 1; }

```

```

init_defs_builtins: /usr/local/lib/coccinelle/standard.h
HANDLING: <source>
diff =
--- <source>
+++ /tmp/cocci-output-2385750-2fc964-example.c
@@ -1,3 +1,3 @@
// Write your C program in this section:
-int main(){ return 0; }
+int main(){ return 1; }

```

Output (10/0) Spatch i - cached (66B)

C with Coccinelle source #1

A ▾ [Icons]

C with Coccinelle ▾

```

1 int main(){ return 0; }
2
3 #ifdef COCCINELLE
4 @patch@
5 constant c;
6 @@
7 - c
8 + 'c'
9 #endif /* COCCINELLE */

```

Spatch (Editor #1)

Spatch

Compiler opti ▾

A ▾ [Icons]

```

1 int main(){ return 'c'; }

```

```

init_defs_builtins: /usr/local/lib/coccinelle/standard.h
HANDLING: <source>
diff =
--- <source>
+++ /tmp/cocci-output-2422211-302d6f-example.c
@@ -1,2 +1,2 @@
-int main(){ return 0; }
+int main(){ return 'c'; }

```

Output (9/0)

Spatch i -30ms (27B)

C with Coccinelle source #1

A ▾ [lock] + ▾ [undo]

C with Coccinelle ▾

```

1 int main(){ int a=0; return a; }
2
3 #ifdef COCCINELLE
4 @patch identifier@
5 @@
6 - a
7 + b
8 #endif /* COCCINELLE */

```

Spatch (Editor #1)

Spatch



Compiler opti ▾

A ▾ [gear] ▾ [filter] [list] [undo] [redo]

```

1 int main(){ int b=0; return b; }

```

```

init_defs_builtins: /usr/local/lib/coccinelle/standard.h
warning: line 3: should a be a metavariable?
HANDLING: <source>
diff =
--- <source>
+++ /tmp/cocci-output-2430745-129382-example.c
@@ -1,2 +1,2 @@
-int main(){ int a=0; return a; }
+int main(){ int b=0; return b; }

```

Output (10/0) Spatch 1 - 154ms (34B)

C with Coccinelle source #1

C with Coccinelle ▾

```

1  int main(){ return 0 + 1 + 2 - 3; }
2
3  #ifdef COCCINELLE
4  @patch@
5  expression e;
6  @@
7  | return
8  - e
9  + 0
10 | ;
11 #endif /* COCCINELLE */

```

(1)

Spatch (Editor #1)


Spatch

Compiler opti


A ▾ ⚙ ▾ ▾ ▾ + ▾

```
1  int main(){ return 0; }
```

Output (9/0) Spatch i - 78ms (25B)

C++ with Coccinelle source #1 A ▾  + ▾  C++ with Coccinelle ▾




```
1 int main(){
2     a -= b = --a +1;
3 }
4
5 #ifdef COCCINELLE
6 @@
7 @@
8     -     -= b = --a
9 #endif /* COCCINELLE */
```

Spatch (Editor #1)  XSpatch ▾  

Compiler options.. ▾

A ▾  ▾    + ▾ 

```
1 {Compilation failed>
2
3 # For more information see the output window
4 # To open the output window, click or drag th
```

  Output (0/6) Spatch i - 102ms 

C with Spatch (Editor #1)

 A ▾  
 + ▾

Spatch

C with Coccinelle ▾

options...

```

1  ce_1
2  int
3
4  }
5
6  #ifc
7  @@
8  @@
9  -
10 #enc
  
```

A ▾ ⚙ ▾ ▾ ⌨ + ▾

```

1  <Compilation failed>
2
3  # For more information
4  # To open the output wi
  
```

Output of Spatch (Compiler #1)

 A ▾  Wrap lines ☰ Select all

```

init_defs_builtins: /usr/local/lib/coccinelle/standard.h
minus: parse error:
  File "patch.cocci", line 3, column 6, charpos = 12
  around = '-=',
  whole content = -      -= b = --a

Compiler returned: 255
  
```

```

init_defs_builtins: /usr/local/lib/coccinelle/standard.h
minus: parse error:
  File "patch.cocci", line 3, column 6, charpos = 12
  around = '-=',
  whole content = -      -= b = --a
  
```

C++ with Coccinelle source #1

A ▾ + ▾

C++ with Coccinelle ▾

```

1  int main(){
2      a -= b = --a +1;
3  }
4
5  #ifdef COCCINELLE
6  @@
7  @@
8  -   a -= b = --a +1;
9  +   a -=      +1;
10 #endif /* COCCINELLE */

```

Spatch (Editor #1)

Spatch ▾

Compiler options.. ▾

A ▾ ▾ + ▾

```

1  int main(){
2      a -= +1;
3  }

```

 Output (13/0) Spatch **i** - 103ms (29B)



C++ with Coccinelle source #1

Spatch (Editor #1)

A ▾ C++ with Coccinelle ▾

Spatch Compiler options... ▾

```

1 struct S { int operator [] (int
2 int main()
3 {
4     ..... int a[1][1][1];
5     ..... int i=0,j=0,k=0;
6     ..... return a[i][j][k];
7 }
8
9 #ifdef COCCINELLE
10 @@ @@
11 .. int main() {
12 - int a[1][1][1];
13 + S a;
14     ...
15 .. }
16 @@
17 symbol a,i,j,k;
18 @@
19 - a[i][j][k]
20 + a[i, j, k]
21 #endif /* COCCINELLE */

```

A ▾

```

1 struct S { int operator [] (int x,int y,int z) {ret
2 int main()
3 {
4     S a;
5     int i=0,j=0,k=0;
6     return a[i, j, k];
7 }

```

Output (18/0) Spatch i - 83ms (147B)

## Load and save editor text



Examples

Browser-local storage

Browser-local history

File system

## Load/save to your system

Load from a local file

Save to file

Close

```
15  .. }
16  @@
17  symbol a, i, j, k;
18  @@
19  - a[i][j][k]
20  + a[i, j, k]
21  #endif /* COCCINELLE */
```

localhost:10240/#load-examples

Output (18/0) Spatch i - 83ms (147B)

# How to run Coccinelle Explorer

1. Install Coccinelle locally  
(<https://coccinelle.gitlabpages.inria.fr/website/>)
2. Check  
<https://github.com/compiler-explorer/compiler-explorer/pull/7001>  
and get the source code
3. As of now, Compiler Explorer requires specific version of Node.js; e.g. in  
`path-to-your/node-v20.18.2-linux-x64`
4. `EXTRA_ARGS=--language coccinelle_for_cpp,coccinelle_for_c`  
`NODE_DIR=path-to-your/node-v20.18.2-linux-x64 make dev`
5. Point browser at <http://localhost:10240/>

## Perspectives

- ▶ proof of concept

- ▶ under review

`https://github.com/compiler-explorer/  
compiler-explorer/pull/7001`

- ▶ not yet there:

- separate windows for C/C++ and SmPL
- specific material
- which default options?

## Perspectives

- ▶ proof of concept

- ▶ under review

<https://github.com/compiler-explorer/compiler-explorer/pull/7001>

- ▶ not yet there:

- separate windows for C/C++ and SmPL

- specific material

- which default options?

- ▶ feedback welcome!

## Perspectives

- ▶ proof of concept

- ▶ under review

<https://github.com/compiler-explorer/compiler-explorer/pull/7001>

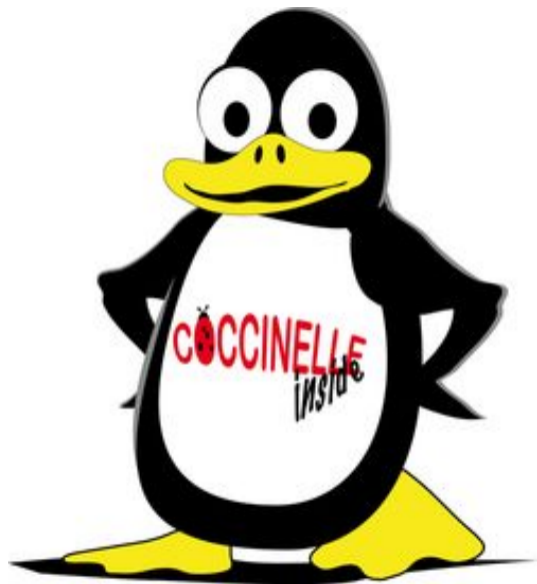
- ▶ not yet there:

- separate windows for C/C++ and SmPL

- specific material

- which default options?

- ▶ feedback welcome (also about learning Coccinelle)!




## New: full-day Coccinelle training

- ▶ held at LRZ in Garching, Germany on 22.01.2025
- ▶ elements of grammar
- ▶ examples appendix, rather HPC-related
- ▶ <https://doi.org/10.5281/zenodo.14728519> ↓ lecture notes





# New: cheat sheet

GCCINELLE Cheat Sheet: basics of invocation and elements of SMPPL (SEMANTIC PATCH LANGUAGE)																																																																																																																																																													
<pre> Invocation: spatch ... ... --parse-cocci a.cocci ... --parse-c a.c ... --parse-c++ a.cpp ... a.cocci directory ... --sp-file a.cocci a.c ... --sp-file a.cocci a.cpp ... --test a ... --test a           </pre>	<pre> description do a semantic patch parse check do a C source file parse check do a C++ source file parse check patch a directory get C patch of a.c get C++ patch of a.cpp a.cocci begins with #spatch --c+ get C patch of a.c a.cocci begins with #spatch --c++           </pre>	<pre> requires           </pre>																																																																																																																																																											
<pre> #spatch command line options @rulename ...S metavariablename v; other metadecclarations @S rule body provides context and references metavariables - minus code context usually optional + plus code context usually optional ** multiple insertion           </pre>	<pre> // a semantic patch file @rule1@ may match or not match @rule2 depends on rule1@ matches if rule1 matches @rule2 depends on !rule1@ matches if rule1 does not @rule2@ metavariablename rule1.v; @S v is inherited and usable           </pre>	<pre> @S statement S; // or other @S ... when != S // constraint ... when any // greedy dots &lt;&lt;... required match ...&gt; ... optional match ...&gt;           </pre>																																																																																																																																																											
<pre> @initialize:python@ @S python code executed once @r1@ metavariablename x;           </pre>	<pre> @S @S first pattern to match &amp; // conjunction of patterns   // disjunction of patterns * // max one insert per construct } // changes only within (...)           </pre>	<pre> #include "file.cocci" @S // comment (ignored) @S + 0 // after // are comments - 0 // need rule even if // #include'ing rules           </pre>	<pre> SMPPL code referencing a @script: python @S a &lt;&lt; r1.r // inherit a from r1 @S python code referencing a @initialize:python@ @S python code executed once           </pre>																																																																																																																																																										
<table border="1"> <thead> <tr> <th>metavariablename v</th> <th>no v identifier, but token v(a,...) or v(a,...)</th> <th>v==pppp</th> <th>list for {a,...} v</th> <th>single reasonable (-)</th> <th>in top-level</th> <th>notes</th> </tr> </thead> <tbody> <tr> <td>attribute name</td> <td>✓</td> <td></td> <td></td> <td></td> <td></td> <td>implicit semantic: patch-wide scope; on types, identifiers and function headers;</td> </tr> <tr> <td>declaration</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> <td>given identifier I and declaration D, usable as ID</td> </tr> <tr> <td>expression</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> <td>given expression E and statement S, usable as ES</td> </tr> <tr> <td>field</td> <td></td> <td></td> <td>✓</td> <td></td> <td></td> <td></td> </tr> <tr> <td>format</td> <td></td> <td></td> <td>✓</td> <td></td> <td></td> <td></td> </tr> <tr> <td>function</td> <td>✓</td> <td></td> <td>✓</td> <td></td> <td>✓</td> <td>declarations only</td> </tr> <tr> <td>identifier</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> <td>also: fresh identifier "v##"new", identifier list in macro definitions</td> </tr> <tr> <td>local idexpression</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>e.g. local idexpression type i</td> </tr> <tr> <td>global idexpression</td> <td></td> <td></td> <td></td> <td></td> <td>✓</td> <td>e.g. global idexpression i</td> </tr> <tr> <td>assignment operator</td> <td></td> <td>✓</td> <td></td> <td></td> <td></td> <td>replacing operator token with metavariable may unbind metamorphisms</td> </tr> <tr> <td>binary operator</td> <td></td> <td>✓</td> <td></td> <td></td> <td></td> <td>can produce expression list EL from parameter list PL, e.g. PL.EL</td> </tr> <tr> <td>parameter</td> <td></td> <td></td> <td>✓</td> <td></td> <td></td> <td>in rules as @p after any token</td> </tr> <tr> <td>pragma/info</td> <td></td> <td>✓</td> <td>✓</td> <td></td> <td></td> <td>only regeop match</td> </tr> <tr> <td>statement</td> <td></td> <td></td> <td>✓</td> <td>✓</td> <td></td> <td></td> </tr> <tr> <td>symbol</td> <td>✓</td> <td></td> <td></td> <td></td> <td></td> <td>implicit semantic: patch-wide scope</td> </tr> <tr> <td>type</td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> <td></td> <td></td> </tr> <tr> <td>typedef</td> <td>✓</td> <td></td> <td></td> <td></td> <td></td> <td>implicit semantic: patch-wide scope</td> </tr> </tbody> </table>	metavariablename v	no v identifier, but token v(a,...) or v(a,...)	v==pppp	list for {a,...} v	single reasonable (-)	in top-level	notes	attribute name	✓					implicit semantic: patch-wide scope; on types, identifiers and function headers;	declaration	✓	✓	✓	✓		given identifier I and declaration D, usable as ID	expression	✓	✓	✓	✓		given expression E and statement S, usable as ES	field			✓				format			✓				function	✓		✓		✓	declarations only	identifier	✓	✓	✓	✓		also: fresh identifier "v##"new", identifier list in macro definitions	local idexpression						e.g. local idexpression type i	global idexpression					✓	e.g. global idexpression i	assignment operator		✓				replacing operator token with metavariable may unbind metamorphisms	binary operator		✓				can produce expression list EL from parameter list PL, e.g. PL.EL	parameter			✓			in rules as @p after any token	pragma/info		✓	✓			only regeop match	statement			✓	✓			symbol	✓					implicit semantic: patch-wide scope	type	✓	✓	✓				typedef	✓					implicit semantic: patch-wide scope	<table border="1"> <thead> <tr> <th>S rulekind@</th> <th>implicit</th> <th>match requirement</th> <th>error</th> <th>missing</th> </tr> </thead> <tbody> <tr> <td>expression</td> <td>✓</td> <td>tokens must also be expression</td> <td>invalid token</td> <td>not simply removable (need context)</td> </tr> <tr> <td>identifier</td> <td>✗</td> <td>any token (also in declaration)</td> <td>already tagged token</td> <td>conflicting changes overlap</td> </tr> <tr> <td>forall in ...</td> <td></td> <td>between dots, all paths must match</td> <td>parse error</td> <td>trying matching more top-level things?</td> </tr> <tr> <td>return in ...</td> <td></td> <td>some, but no patch</td> <td></td> <td>does the source fully parse?</td> </tr> <tr> <td>attach all</td> <td>✗</td> <td>no metamorphisms (see --LSP-LSB11 O)</td> <td>no available token to attach</td> <td>* just outside a @ or ! construct</td> </tr> </tbody> </table>	S rulekind@	implicit	match requirement	error	missing	expression	✓	tokens must also be expression	invalid token	not simply removable (need context)	identifier	✗	any token (also in declaration)	already tagged token	conflicting changes overlap	forall in ...		between dots, all paths must match	parse error	trying matching more top-level things?	return in ...		some, but no patch		does the source fully parse?	attach all	✗	no metamorphisms (see --LSP-LSB11 O)	no available token to attach	* just outside a @ or ! construct
metavariablename v	no v identifier, but token v(a,...) or v(a,...)	v==pppp	list for {a,...} v	single reasonable (-)	in top-level	notes																																																																																																																																																							
attribute name	✓					implicit semantic: patch-wide scope; on types, identifiers and function headers;																																																																																																																																																							
declaration	✓	✓	✓	✓		given identifier I and declaration D, usable as ID																																																																																																																																																							
expression	✓	✓	✓	✓		given expression E and statement S, usable as ES																																																																																																																																																							
field			✓																																																																																																																																																										
format			✓																																																																																																																																																										
function	✓		✓		✓	declarations only																																																																																																																																																							
identifier	✓	✓	✓	✓		also: fresh identifier "v##"new", identifier list in macro definitions																																																																																																																																																							
local idexpression						e.g. local idexpression type i																																																																																																																																																							
global idexpression					✓	e.g. global idexpression i																																																																																																																																																							
assignment operator		✓				replacing operator token with metavariable may unbind metamorphisms																																																																																																																																																							
binary operator		✓				can produce expression list EL from parameter list PL, e.g. PL.EL																																																																																																																																																							
parameter			✓			in rules as @p after any token																																																																																																																																																							
pragma/info		✓	✓			only regeop match																																																																																																																																																							
statement			✓	✓																																																																																																																																																									
symbol	✓					implicit semantic: patch-wide scope																																																																																																																																																							
type	✓	✓	✓																																																																																																																																																										
typedef	✓					implicit semantic: patch-wide scope																																																																																																																																																							
S rulekind@	implicit	match requirement	error	missing																																																																																																																																																									
expression	✓	tokens must also be expression	invalid token	not simply removable (need context)																																																																																																																																																									
identifier	✗	any token (also in declaration)	already tagged token	conflicting changes overlap																																																																																																																																																									
forall in ...		between dots, all paths must match	parse error	trying matching more top-level things?																																																																																																																																																									
return in ...		some, but no patch		does the source fully parse?																																																																																																																																																									
attach all	✗	no metamorphisms (see --LSP-LSB11 O)	no available token to attach	* just outside a @ or ! construct																																																																																																																																																									

- ▶ selected essentials
- ▶ 1 page!
- ▶ <https://doi.org/10.5281/zenodo.14728558> ↓



## Stay tuned

- ▶ We'll offer a Coccinelle training this year (not scheduled yet)
- ▶ LRZ trainings 2025 calendar: ↓



<https://tiny.badw.de/zc5D9e>

Work made possible by



BayFrance23 travel grant



Thanks to Julia LAWALL for the collaboration!