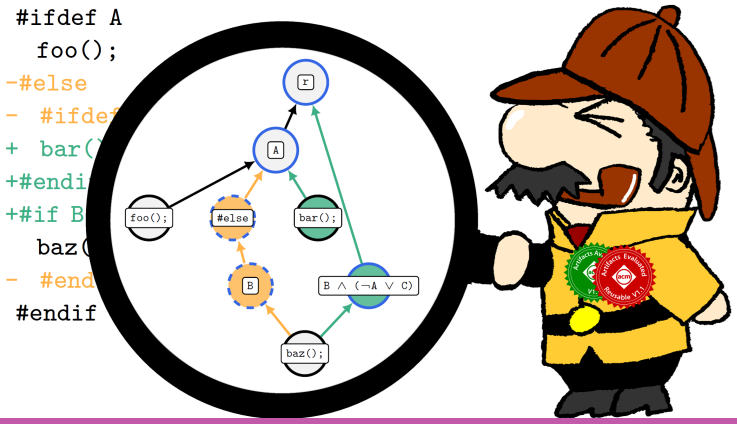


Variability-Aware Differencing with DiffDetective

Paul Bittner, Alexander Schultheiß, Benjamin Moosherr, Timo Kehrer, Thomas Thüm | July 17th, 2024 | FSE'24

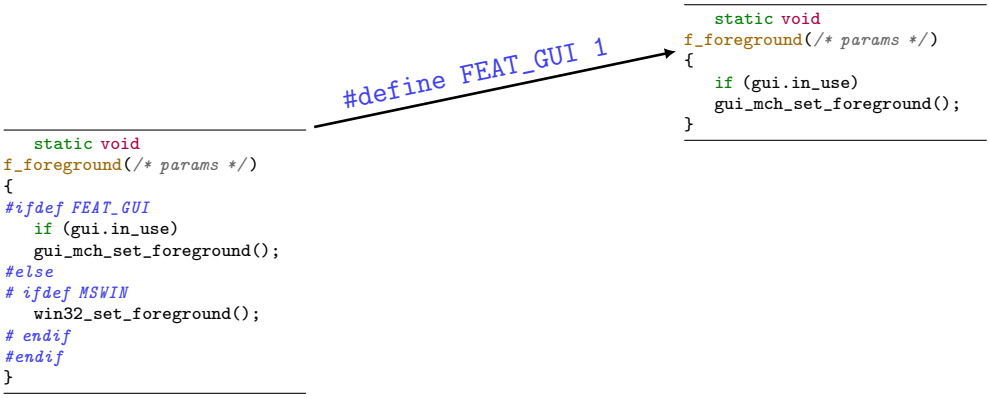
PADERBORN UNIVERSITY

universität ulm

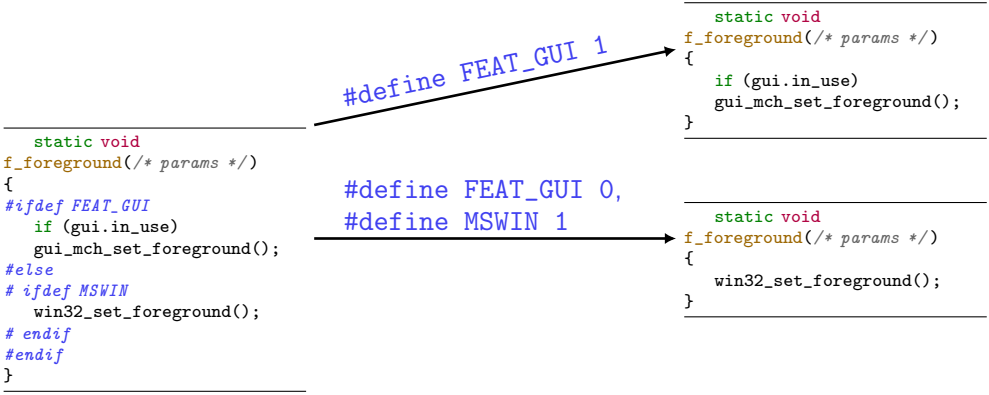
u^b

```
   static void
f_foreground(/* params */)
{
#ifdef FEAT_GUI
   if (gui.in_use)
   gui_mch_set_foreground();
#else
# ifdef MSWIN
   win32_set_foreground();
# endif
#endif
}
```

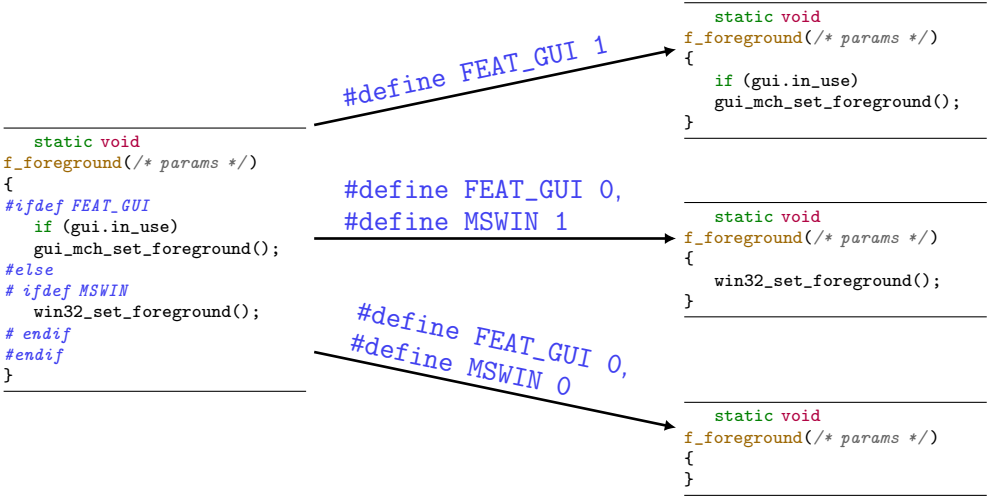
Vim Commit ab4cece.



Vim Commit ab4cece.



Vim Commit ab4cece.



Vim Commit ab4cece.


Vim


OPENVPN


GODOT
Game engine


Marlin


GCC

```
#ifdef A
 foo();
#else
 #ifdef B
 baz();
 #endif
#endif
```

```
#ifdef A
 foo();
 bar();
#endif
#if B && (!A || C)
 baz();
#endif
```

```
#ifdef A
 foo();
#else
 #ifdef B
 baz();
 #endif
#endif
```

diff

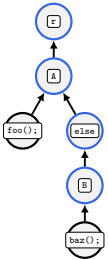
```
#ifdef A
 foo();
-#else
-  #ifdef B
+  bar();
+#endif
+#if B && (!A || C)
   baz();
-  #endif
 #endif
```

diff

```
#ifdef A
 foo();
 bar();
#endif
#if B && (!A || C)
 baz();
#endif
```

```
#ifdef A
 foo();
#else
 #ifdef B
 baz();
 #endif
#endif
```

parse



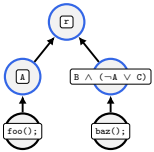
diff

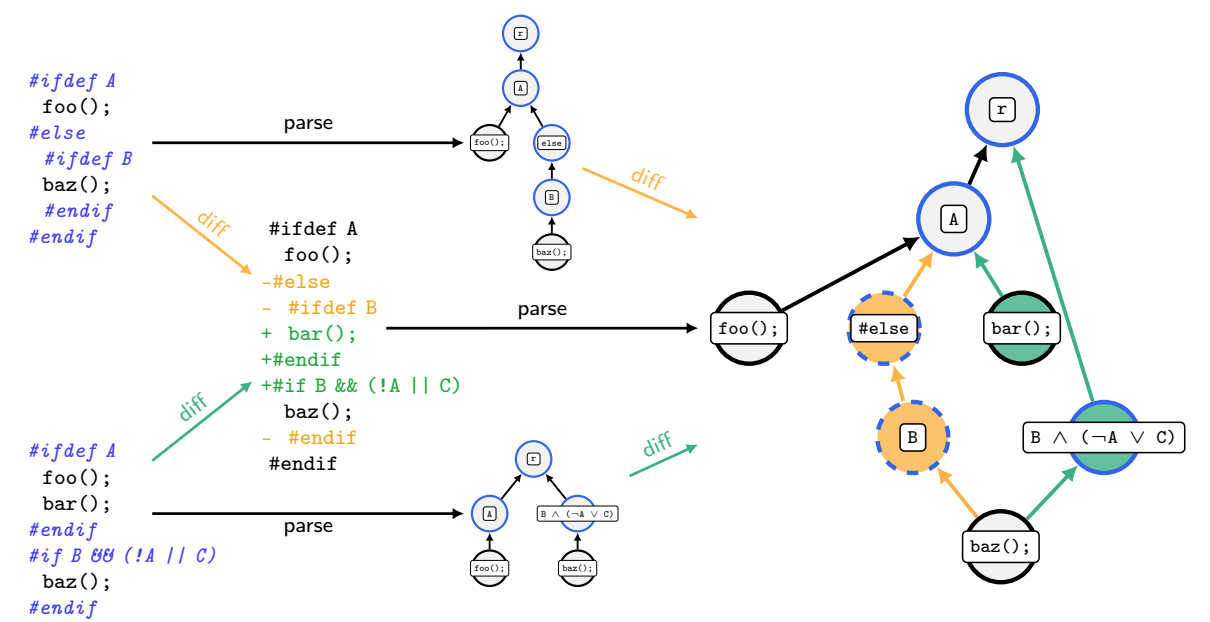
```
#ifdef A
 foo();
-#else
-  #ifdef B
+  bar();
+#endif
+#if B && (!A || C)
   baz();
-  #endif
   #endif
```

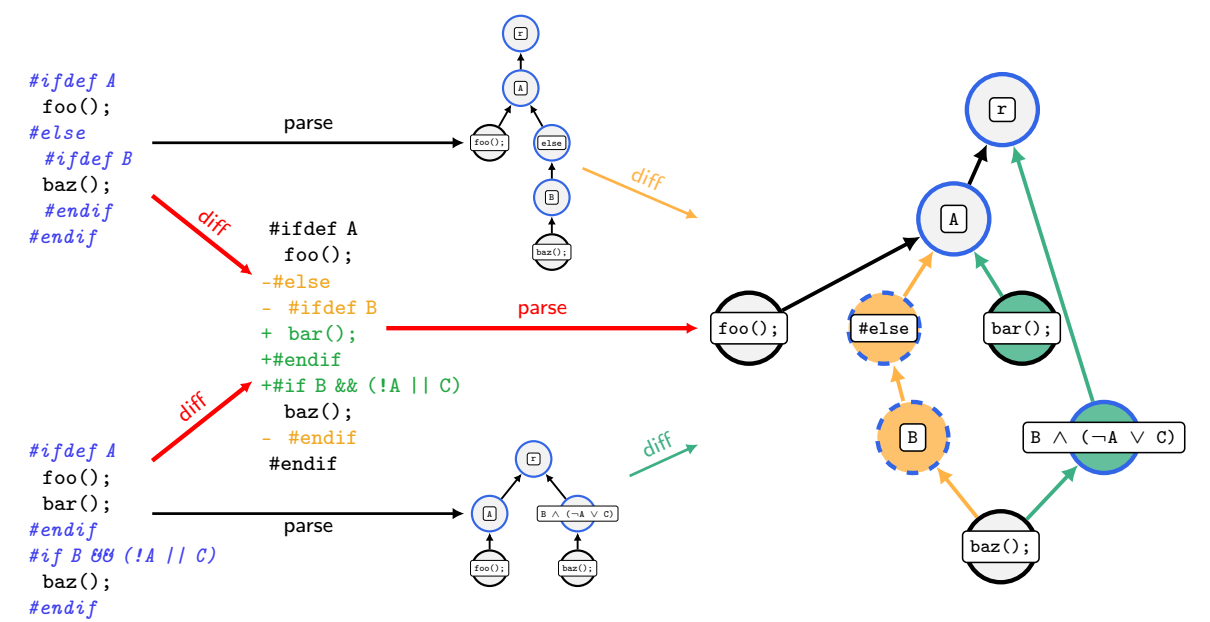
diff

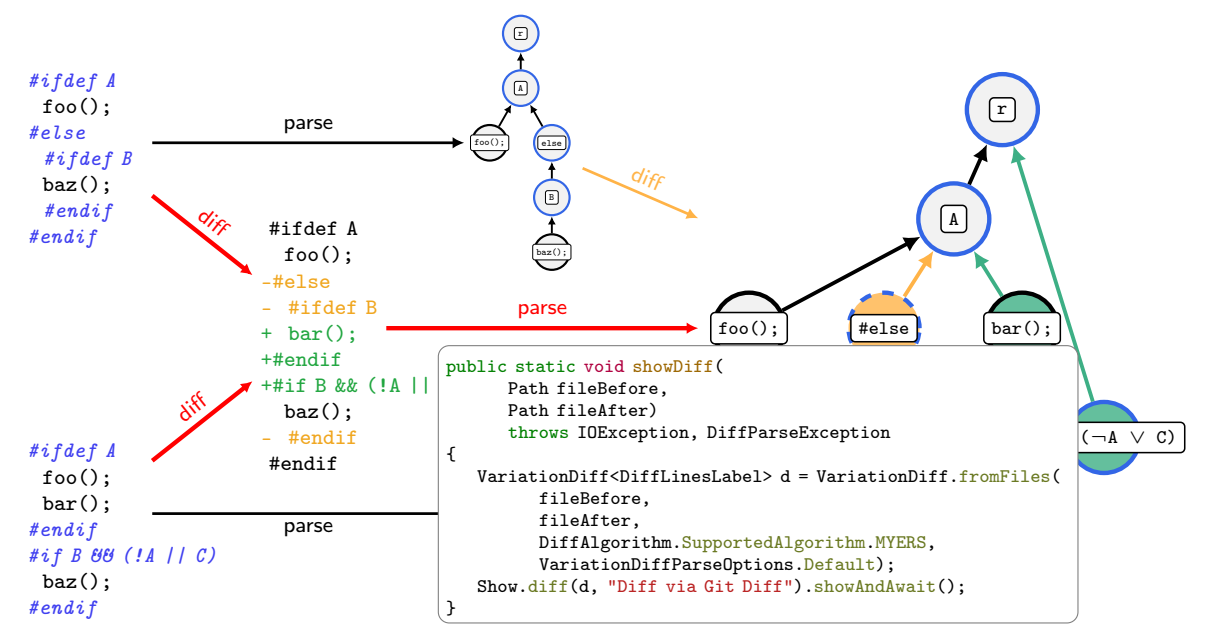
```
#ifdef A
 foo();
 bar();
#endif
#if B && (!A || C)
 baz();
#endif
```

parse

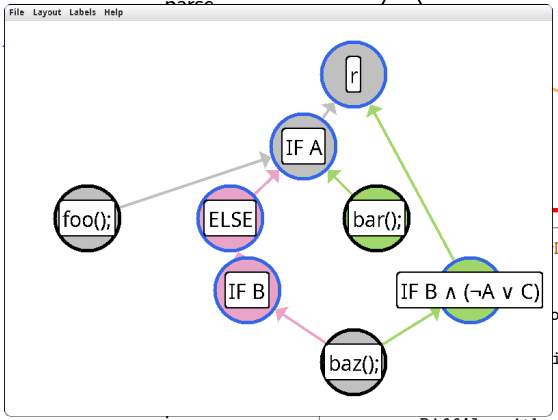


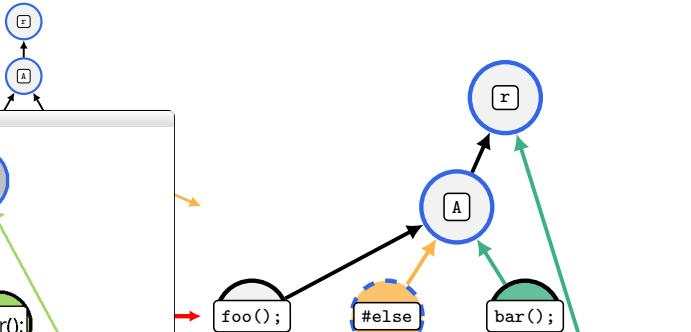

#ifdef A
foo();
#else
#ifdef B
baz();
#endif
#endif
parse
diff
#ifdef A
foo();
-#else
- #ifdef B
+ bar();
+#endif
+#if B && (!A || C)
baz();
- #endif
#endif
parse
diff
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
parse
diff
r
A
foo();
else
B
baz();
r
A
foo();
#else
bar();
B
B ∧ (¬A ∨ C)
baz();
r
A
B ∧ (¬A ∨ C)
foo();
baz();


#ifdef A
foo();
#else
#ifdef B
baz();
#endif
#endif
parse
diff
#ifdef A
foo();
-#else
- #ifdef B
+ bar();
+#endif
+#if B && (!A || C)
baz();
- #endif
#endif
parse
diff
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
parse
r
A
foo();
else
B
baz();
B ∧ (¬A ∨ C)
diff
diff
diff
r
A
foo();
#else
bar();
B
B ∧ (¬A ∨ C)
baz();


#ifdef A
foo();
#else
#ifdef B
baz();
#endif
#endif
parse
diff
#ifdef A
foo();
-#else
- #ifdef B
+ bar();
+#endif
+#if B && (!A ||
baz();
- #endif
#endif
parse
diff
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
parse
r
A
foo();
else
B
baz();
diff
r
A
foo();
#else
bar();
(¬A ∨ C)
public static void showDiff(
Path fileBefore,
Path fileAfter)
throws IOException, DiffParseException
{
VariationDiff<DiffLinesLabel> d = VariationDiff.fromFiles(
fileBefore,
fileAfter,
DiffAlgorithm.SupportedAlgorithm.MYERS,
VariationDiffParseOptions.Default);
Show.diff(d, "Diff via Git Diff").showAndAwait();
}

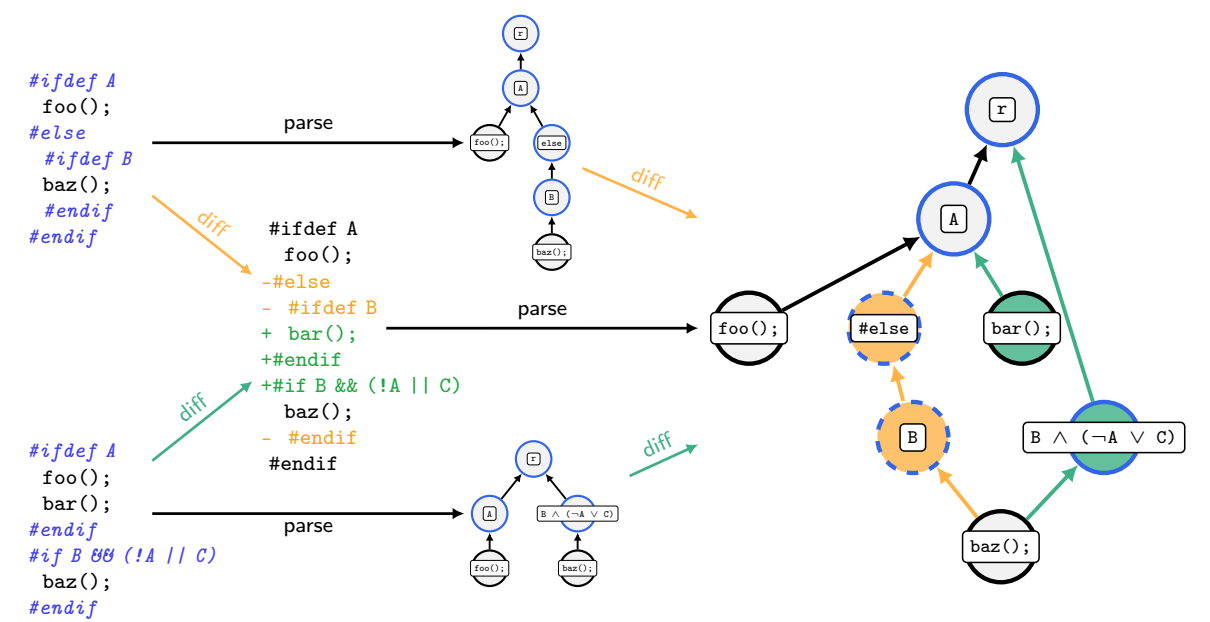
```
#ifdef A
  foo();
#else
  #ifdef
  baz();
  #endif
#endif
```

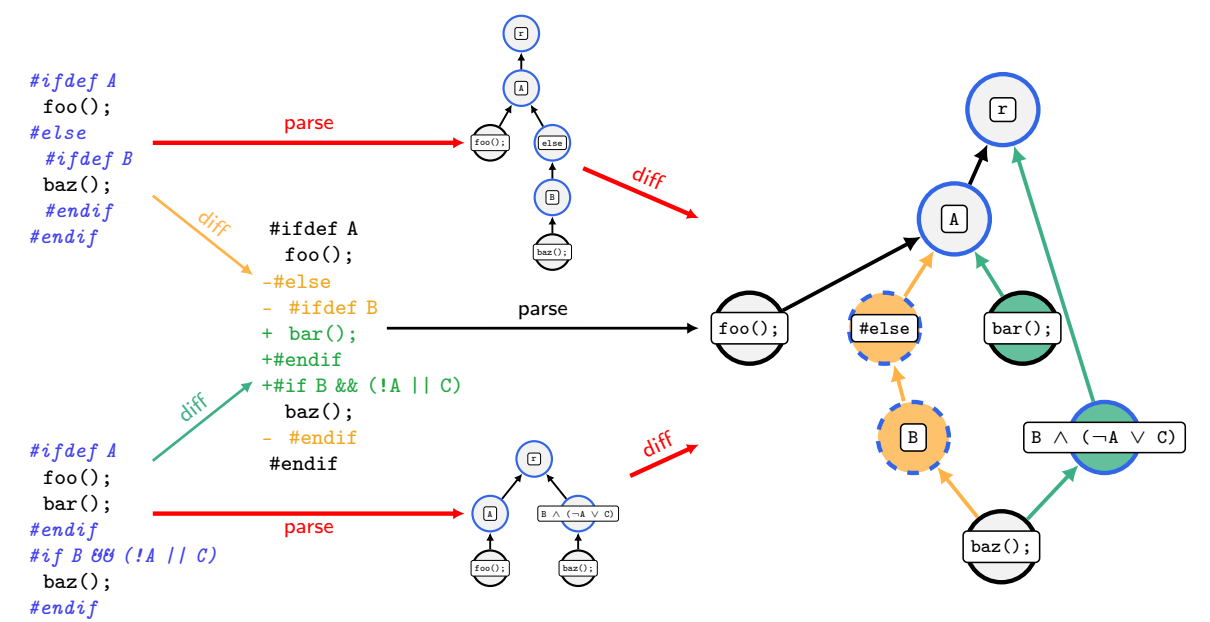


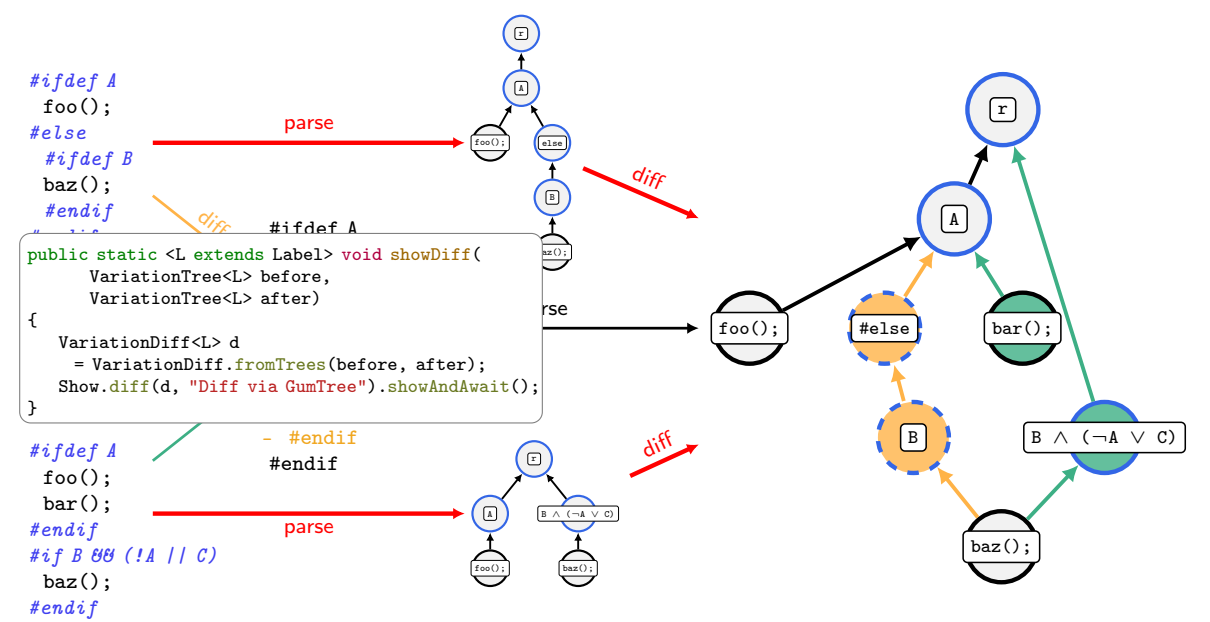


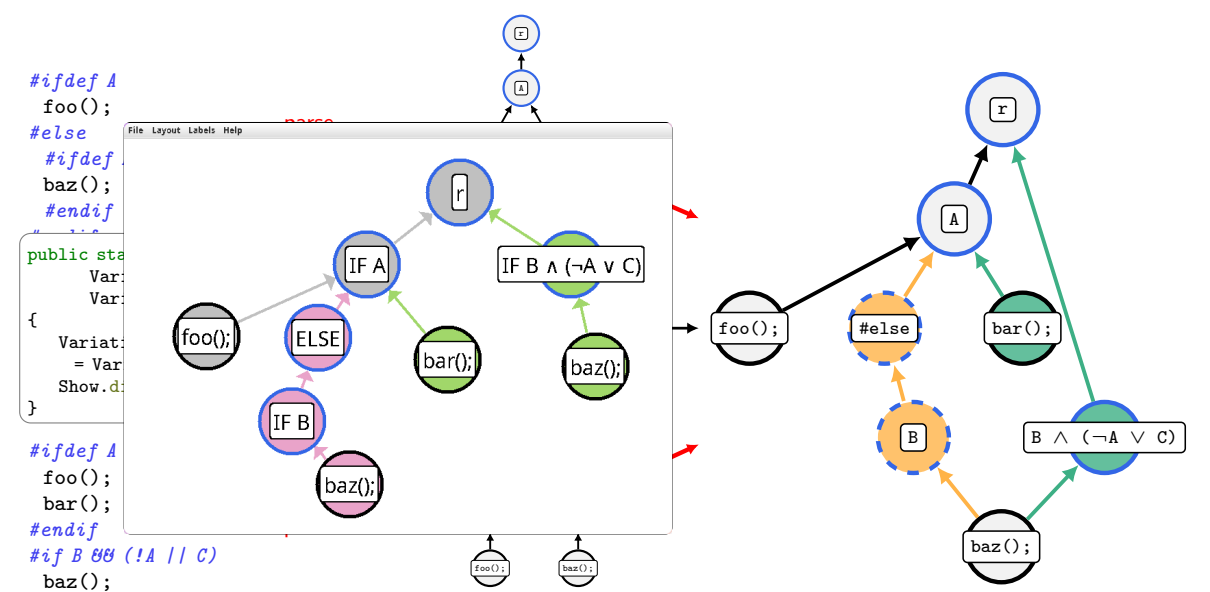

```
#ifdef A
  foo();
  bar();
#endif
#if B && (!A || C)
  baz();
#endif
```

```
Diff(

on, DiffParseException

inesLabel> d = VariationDiff.fromFiles(

        DiffAlgorithm.SupportedAlgorithm.MYERS,
        VariationDiffParseOptions.Default);
    Show.diff(d, "Diff via Git Diff").showAndAwait();
}
```


#ifdef A
foo();
#else
#ifdef B
baz();
#endif
#endif
parse
diff
#ifdef A
foo();
-#else
- #ifdef B
+ bar();
+#endif
+#if B && (!A || C)
baz();
- #endif
#endif
parse
diff
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
parse
diff
r
A
foo();
else
B
baz();
r
A
foo();
#else
bar();
B
B ∧ (¬A ∨ C)
baz();
r
A
B ∧ (¬A ∨ C)
foo();
baz();


#ifdef A
foo();
#else
#ifdef B
baz();
#endif
#endif
parse
diff
#ifdef A
foo();
-#else
- #ifdef B
+ bar();
+#endif
+#if B && (!A || C)
baz();
- #endif
#endif
parse
diff
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
parse
diff
diff
r
A
foo();
else
B
baz();
r
A
B ∧ (¬A ∨ C)
foo();
baz();
r
A
foo();
#else
bar();
B
B ∧ (¬A ∨ C)
baz();

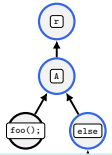

#ifdef A
foo();
#else
#ifdef B
baz();
#endif
parse
diff
#ifdef A
public static <L extends Label> void showDiff(
VariationTree<L> before,
VariationTree<L> after)
{
VariationDiff<L> d
= VariationDiff.fromTrees(before, after);
Show.diff(d, "Diff via GumTree").showAndAwait();
}
- #endif
#endif
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
parse
diff
r
A
foo();
else
B
baz();
B ∧ (¬A ∨ C)
foo();
#else
bar();
B
B ∧ (¬A ∨ C)
baz();

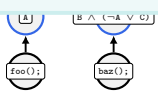

#ifdef A
foo();
#else
#ifdef
baz();
#endif
public sta
Var
Var
{
Variat
= Var
Show.d
}
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
File Layout Labels Help
r
IF A
IF B ∧ (¬A ∨ C)
foo();
ELSE
bar();
baz();
IF B
baz();
foo();
baz();
r
A
foo();
#else
bar();
B
B ∧ (¬A ∨ C)
baz();

```
#ifdef A
 foo();
#else
 #ifdef B
 l
 #e
 #i
 l
#endif
#if B && (!A || C)
 baz();
#endif
```

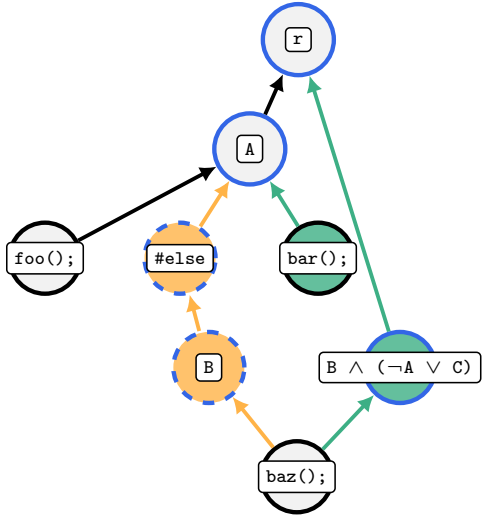
parse

parse



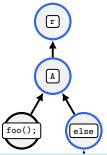


Use-cases for variability-aware differencing?

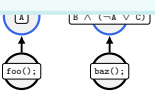


```
#ifdef A
 foo();
#else
 #ifdef B
 l
#e
#i
 l
#endif
#if B && (!A || C)
 baz();
#endif
```

parse



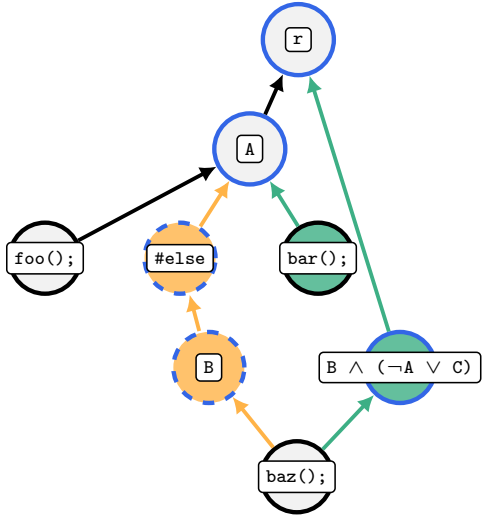
parse



Use-cases for variability-aware differencing?

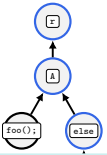
explaining edits *ESEC/FSE'22*

How did the set of variants of this line change?

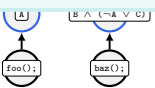


```
#ifdef A
 foo();
#else
 #ifdef B
#endif
#if B && (!A || C)
 baz();
#endif
```

parse



parse



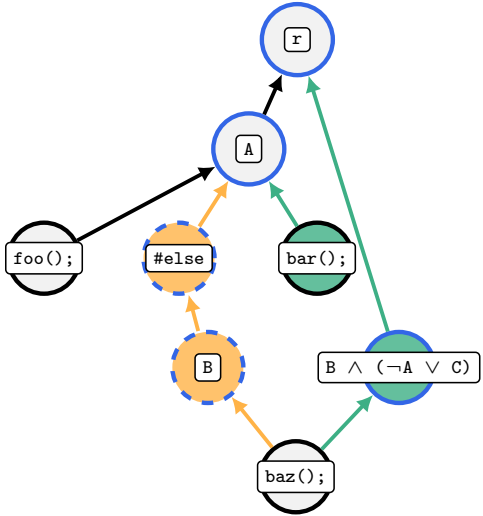
Use-cases for variability-aware differencing?

explaining edits *ESEC/FSE'22*

How did the set of variants of this line change?

feature-aware commit untangling *SPLC'23*

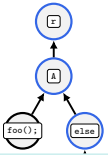
What's the diff for just feature B?

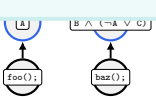


```
#ifdef A
 foo();
#else
 #ifdef B
 l
#e
#i
 t
 l
#endif
#if B && (!A || C)
 baz();
#endif
```

parse

parse





Use-cases for variability-aware differencing?

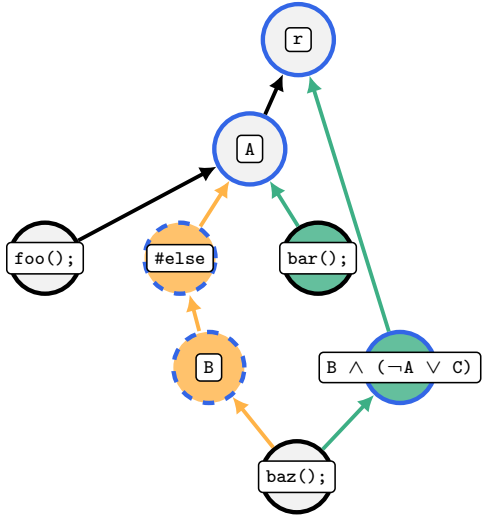
explaining edits *ESEC/FSE'22*

How did the set of variants of this line change?

feature-aware commit untangling *SPLC'23*

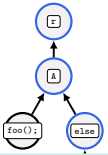
What's the diff for just feature B?

simulate clone-and-own evolution histories *EASE'22*



```
#ifdef A
 foo();
#else
 #ifdef B
```

parse



Use-cases for variability-aware differencing?

explaining edits — *ESEC/FSE'22*
How did the set of variants of this line change?

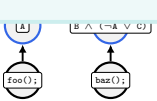
feature-aware commit untangling — *SPLC'23*
What's the diff for just feature B?

simulate clone-and-own evolution histories — *EASE'22*

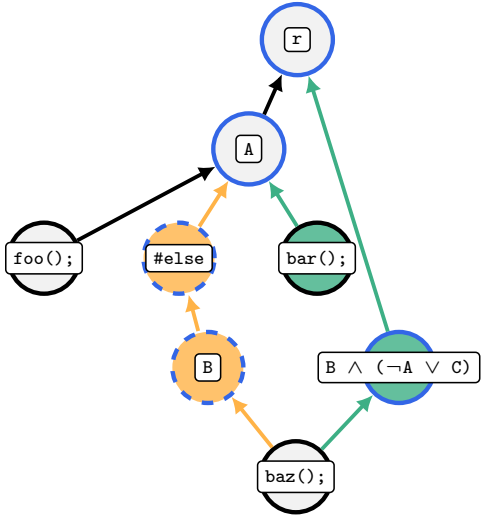
support reviewing merge requests

benchmarks for differencing algorithms

parse




```
#endif
#if B && (!A || C)
 baz();
#endif
```



Variability-Aware Differencing with DiffDetective

Features of DiffDetective

- framework for custom empirical studies of all changes in a git version history

Variability-Aware Differencing with DiffDetective

Features of DiffDetective

- framework for custom empirical studies of all changes in a git version history
- support for multiple differencing algorithms:
  - `git diff` (Myers/Histogram)
  - GumTree by Falleri et al.
  - TrueDiff by Erdweg et al.

Variability-Aware Differencing with DiffDetective

Features of DiffDetective

- framework for custom empirical studies of all changes in a git version history
- support for multiple differencing algorithms:
  - `git diff` (Myers/Histogram)
  - GumTree by Falleri et al.
  - TrueDiff by Erdweg et al.
- GUI for variability-aware diffs
- transformations + analyses on diffs
- demo + screencast
- dataset of 44 repository forks, including

Variability-Aware Differencing with DiffDetective

Features of DiffDetective

- framework for custom empirical studies of all changes in a git version history
- support for multiple differencing algorithms:
  - `git diff` (Myers/Histogram)
  - GumTree by Falleri et al.
  - TrueDiff by Erdweg et al.
- GUI for variability-aware diffs
- transformations + analyses on diffs
- demo + screencast
- dataset of 44 repository forks, including

Try it on

https://variantsync.github.io/DiffDetective

References

ESEC/FSE'22 *Classifying Edits to Variability in Source Code, ESEC/FSE'22*

SPLC'23 *Views on Edits to Variational Software, SPLC'23*

EASE'22 *Simulating the Evolution of Clone-and-Own Projects with VEVOS, EASE'22*