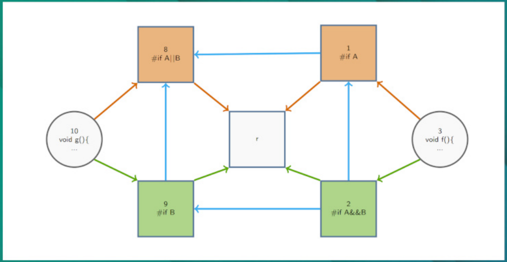
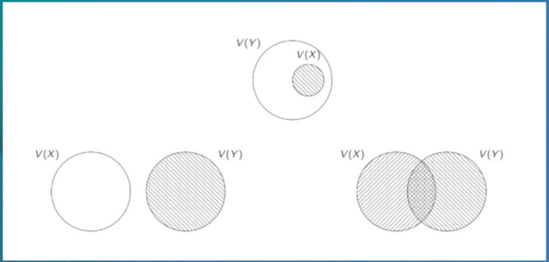
Karlsruher Institut für Technologie

Inspecting the Evolution of Feature Annotations in Configurable Software

FOSD 2023

Lukas Güthing, Paul Bittner, Christof Tinnes, Thomas Thüm, Timo Kehrer | March 27–31, 2023





Evolution of Feature Annotations in Configurable Software

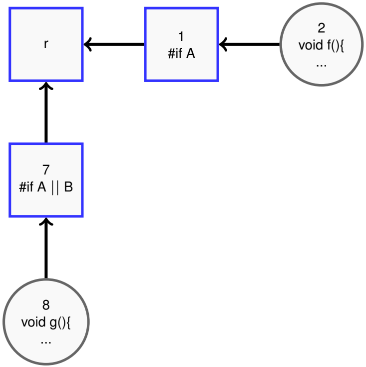
```
#ifdef A
void f(){
        ...
}
#endif

#ifdef A || B
void g(){
        ...
}
#endif
```

Evolution of Feature Annotations in Configurable Software

Karlsruher Institut für Technologie

```
#ifdef A
void f(){
        ...
}
#endif

#ifdef A || B
void g(){
        ...
}
#endif
```

Variation Trees [BTS$^+$22]



Evolution of Feature Annotations in Configurable Software

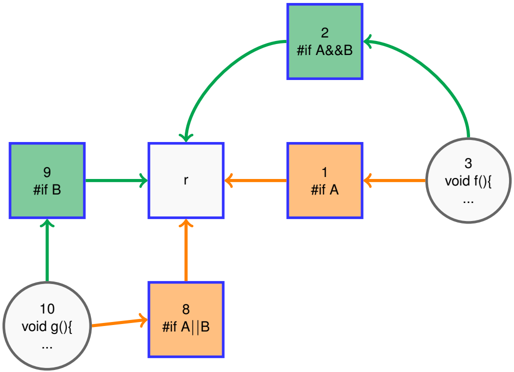
Karlsruher Institut für Technologie

```
- #ifdef A
+ #ifdef A && B
  void f(){
        ...
  }
  #endif

- #ifdef A || B
+ #ifdef B
  void g(){
        ...
  }
  #endif
```

Evolution of Feature Annotations in Configurable Software

```
- #ifdef A
+ #ifdef A && B
  void f(){
          ...
  }
  #endif

- #ifdef A || B
+ #ifdef B
  void g(){
          ...
  }
  #endif
```

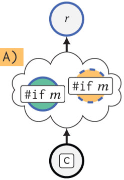
Variation Diffs [BTS+22]

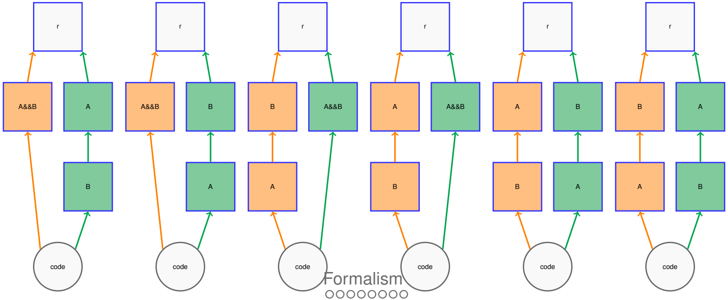


Edit Classes

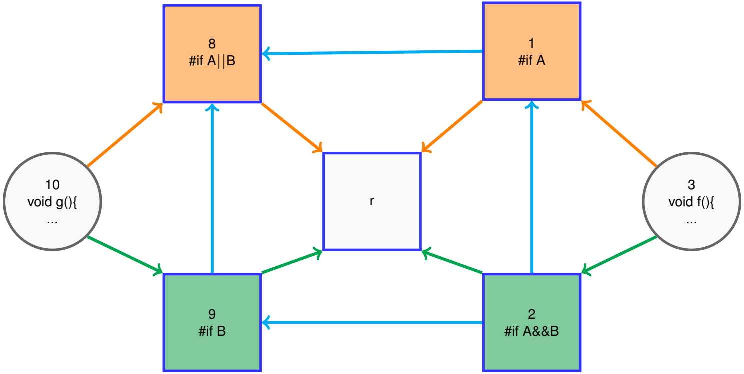
$$
\begin{aligned}
\mathit{Refactoring}(c) \coloneqq\ & \text{unchanged}(c) \\
& \wedge(\mathrm{PC}_{\mathrm{b}}(c) \models \mathrm{PC}_{\mathrm{a}}(c)) \\
& \wedge(\mathrm{PC}_{\mathrm{a}}(c) \models \mathrm{PC}_{\mathrm{b}}(c)) \\
& \wedge(\mathit{path}_{\mathrm{b}}(c) \neq \mathit{path}_{\mathrm{a}}(c))
\end{aligned}
$$

```
- #if A || (B && !A)
+ #if A || B
    c
  #endif
```





Edge-Typed Variation Diff



Edge-Typed Variation Diff

Karlsruher Institut für Technologie

$$EVD = (V, E, r, \tau, l, \Delta, \boxed{\pi})$$

$E : V \times V,$

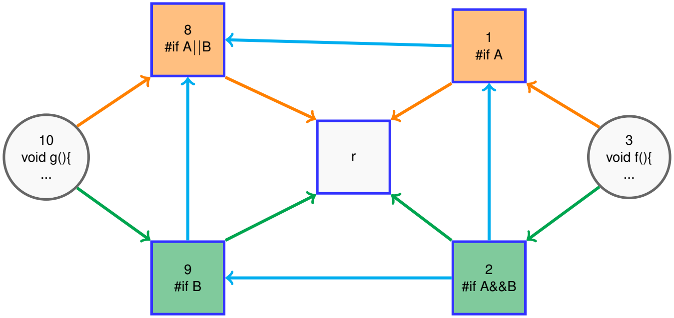
$$E = \{(x, y) | x, y \in V, xRy\}$$

for a relation R and an edge label function π.
Relations between feature mappings: $\tau(x) = \tau(y) = \texttt{mapping}$

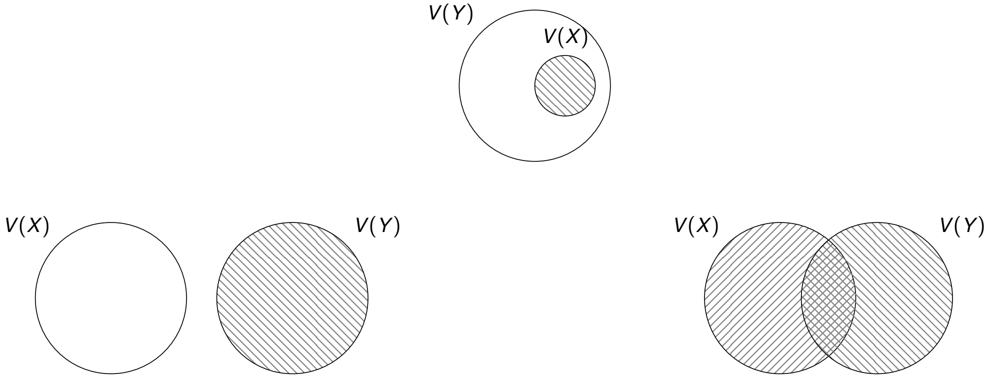
Implication Edges

$$E_n = \{(x, y) | x, y \in V, \tau(x) = \tau(y) = \texttt{mapping}, \boxed{xRy}\}$$

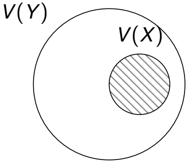
$$\boxed{xRy} := I(x) \models I(y)$$



Edge Relation Choice

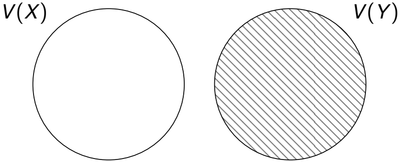


Subset/*Implication*



$$X \models Y$$
$$\Leftrightarrow TAUT(X \Rightarrow Y)$$
$$\Leftrightarrow \neg SAT(X \wedge \neg Y)$$

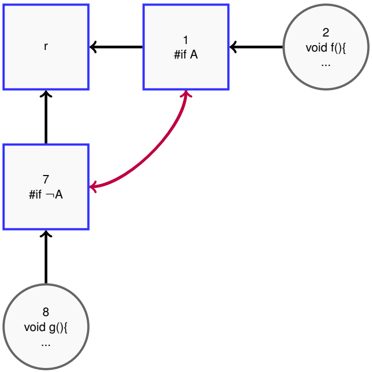
Disjoint/*Alternative*



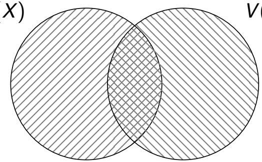
$$X \models \neg Y$$
$$\Leftrightarrow \neg SAT(X \wedge Y)$$
$$(\Leftrightarrow Y \models \neg X)$$

Alternative Edges

$$xRy := (x) \models \neg(y)$$

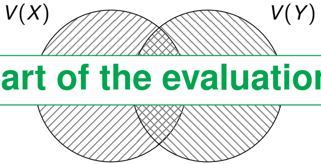


Intersected/*Common Intersection*



$$SAT(X \wedge Y)$$
$$\wedge SAT(\neg X \wedge Y)$$
$$\wedge SAT(X \wedge \neg Y)$$

Intersected/*Common Intersection*



$SAT(X \land Y)$

→ Not part of the evaluation! Still formally completes the set of relations.

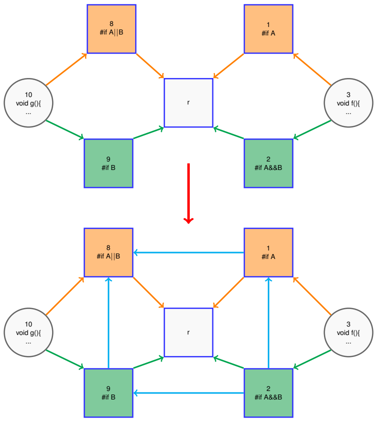
$\land SAT(X \land \neg Y)$

Evaluation

Karlsruher Institut für Technologie

Quantitative Analysis:

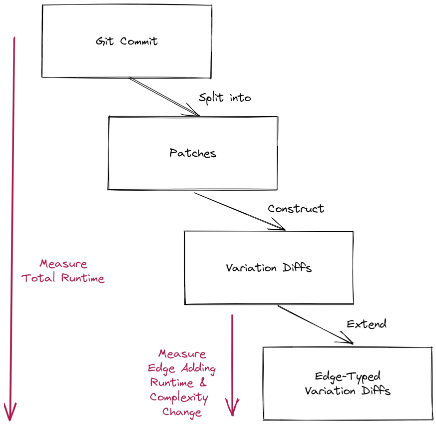
- Edge-Typed Variation Diffs vs Variation Diffs:
  - Construction performance
  - Complexity
- Compare potential optimizations for construction



Evaluation

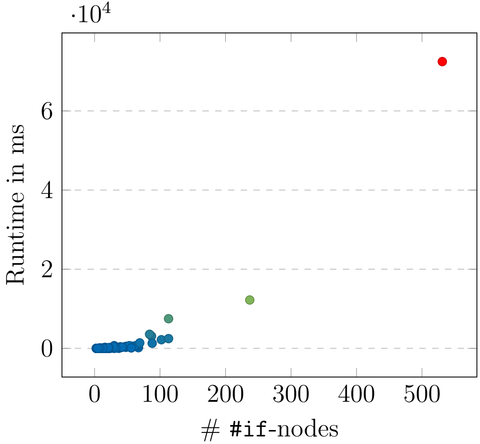
Quantitative Analysis:

- Edge-Typed Variation Diffs vs Variation Diffs:
  - Construction performance
  - Complexity
- Compare potential optimizations for construction

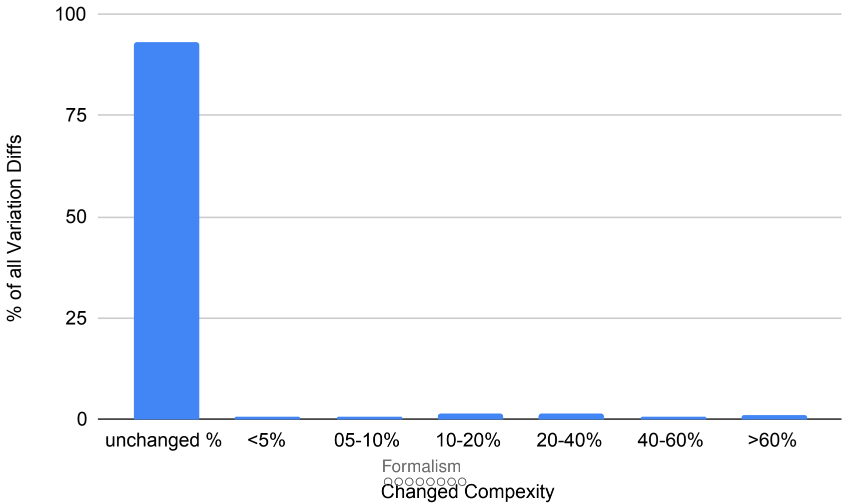


Repo	Additional Runtime in milliseconds			
	mean	min	max	median
apache-httpd	1.45	0	17295	0
berkeley-db-libdb	0.06	0	16	0
busybox	37.97	0	215733	0
cherokee-webserver	0.21	0	270	0
clamav	3.35	0	15116	0
cpython	36.93	0	329143	0
dia	0.13	0	125	0
emacs	4.96	0	72474	0
freebsd	8.66	0	549888	0
ghostscript	11.18	0	42650	0
gnumeric	3.16	0	71308	0
gnuplot	3.375	0	3462	0
irssi	0.03	0	17	0
libssh	0.33	0	519	0
mplayer-svn	1.39	0	2783	0
...	...	...	...	...

Runtime

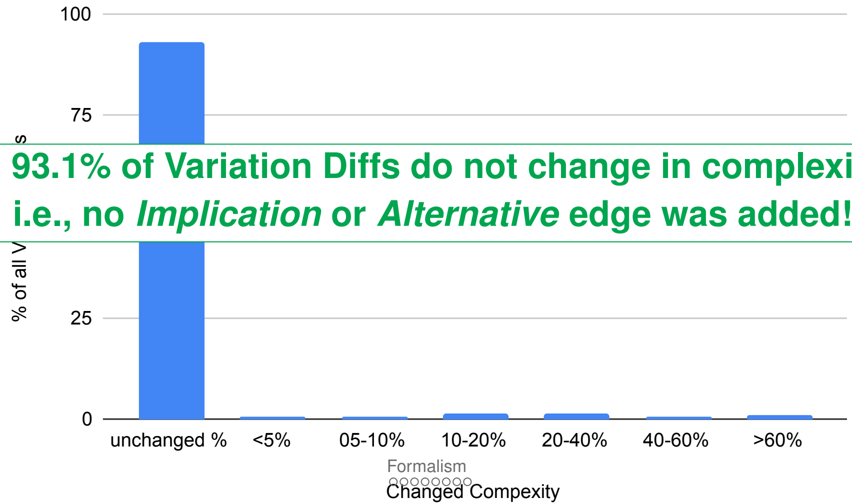


Complexity


100
75
50
25
0
% of all Variation Diffs
unchanged %
<5%
05-10%
10-20%
20-40%
40-60%
>60%
Changed Compexity

Complexity

Karlsruher Institut für Technologie

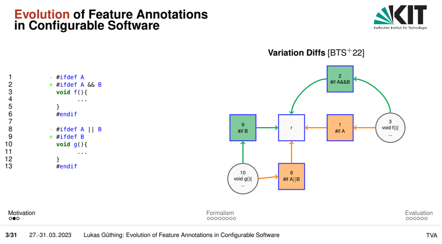


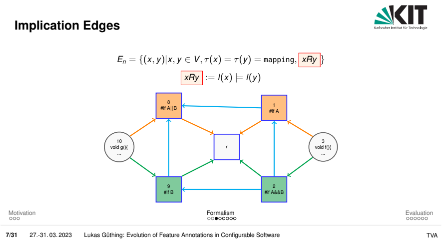
→ **93.1% of Variation Diffs do not change in complexity, i.e., no *Implication* or *Alternative* edge was added!**

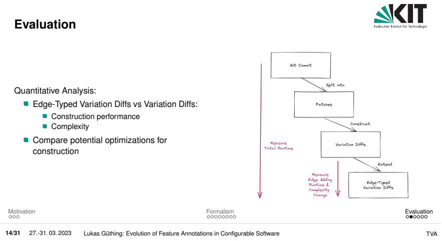
Conclusion

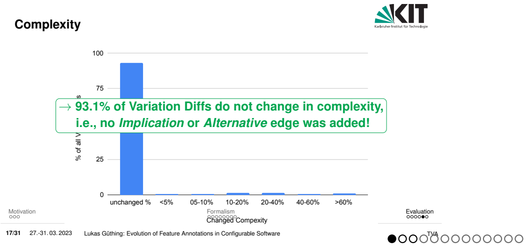
- Edge-typed variation diff construction runtime almost negligible
- Most edits do not change the hierarchy of code and feature mappings much
- Set of edge types is arbitrarily expandable

Thank you for your attention!









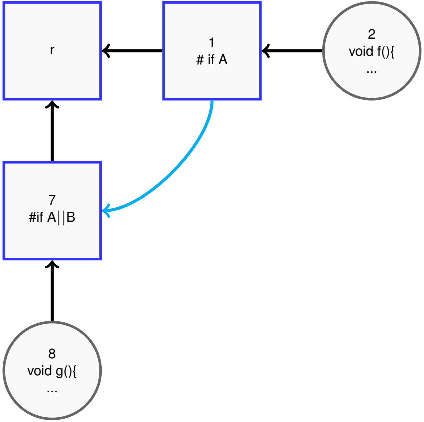
Literature

Paul Maximilian Bittner, Christof Tinnes, Alexander Schultheiß, Sören Viegener, Timo Kehrer, and Thomas Thüm.
Classifying Edits to Variability in Source Code.
pages 196–208, November 2022.

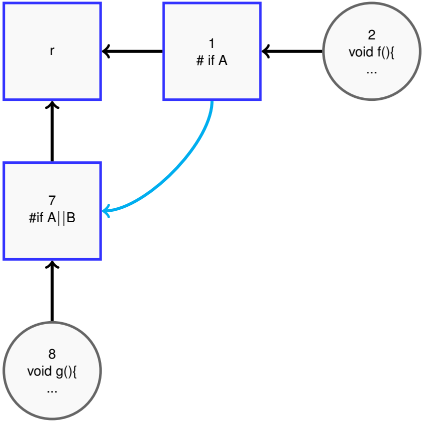
Future Work

- Pattern Finding
- Further Variation Diff Extensions
- Measured Variables and Metrics

Edge-Typed Variation Tree



Edge-Typed Variation Tree



$$EVT = (V, E, r, \tau, l, \boxed{\pi})$$

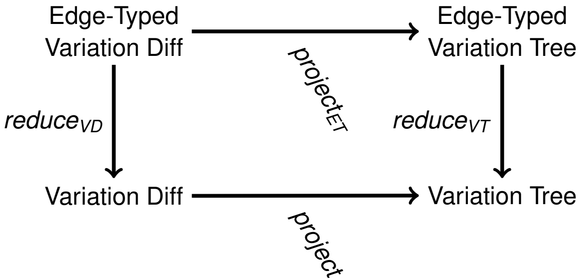
$E \subseteq V \times V$,

$$E = \{(x, y) | x, y \in V, \boxed{xRy}\}$$

for a relation R, an edge label function π and r, τ, l as defined for variation trees.

$\boxed{\tau(x) = \tau(y) = \texttt{mapping}}$ for new edges

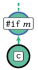
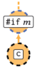
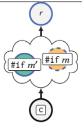
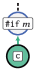
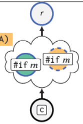
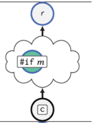
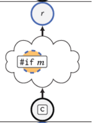
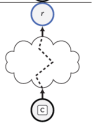
Repository	Domain	Number of Commits
apache-httpd	web server	32,927
berkeley-db-libdb	database system	7
busybox	embedded systems	17,447
cherokee-webserver	web server	5,853
clamav	antivirus program	10,656
cpython	program interpreter	112,096
dia	diagramming software	6,666
emacs	text editor	153,926
freebsd	operating system	271,937
ghostscript	postscript interpreter	22,137
gnumeric	spreadsheet application	24,134
gnuplot	plotting tool	11,748
irssi	IRC client	6,346
libssh	network	5,349
mplayer-svn	media player	37,992
MPSolve	mathematical software	1,773
openldap	LDAP directory service	23,928
opensolaris	operating system	11,422
parrot	virtual machine	49,989
...	...	...



$$reduce_{vd}((V, E_{ex}, r, \tau, l, \Delta, \pi)) := (V, \{e \in E_{ex} | \pi(e) = Nesting\}, r, \tau, l, \Delta).$$

$$
\begin{aligned}
\mathit{project}_{ET}((V, E_{ex}, r, \tau, l, \Delta, \pi), t) := \\
&(\{v \in V | \mathit{exists}(t, \Delta(v))\}, \\
&\{e \in E_{ex} | \pi(e) = \mathit{Nesting} \wedge \mathit{exists}(t, \Delta(e))\} \\
&\cup \{(x, y) \in E_{ex} | \pi(e) \neq \mathit{Nesting} \\
&\wedge \mathit{exists}(t, \Delta(x)) \wedge \mathit{exists}(t, \Delta(y))\}, \\
&r, \tau, l, \pi).
\end{aligned}
$$

Edit classes [BTS+22]

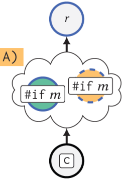
<table>
<tr><th>Edit Pattern Definition</th><th>Example Diff</th><th>Example Tree Diff</th></tr>
<tr><td>$\mathit{AddWithMapping}(c) := \text{added}(c) \land \text{added}(M_a(c))$</td><td><code>+ #if m</code>
<code>+ c</code>
<code>+ #endif</code></td><td></td></tr>
<tr><td>$\mathit{RemWithMapping}(c) := \text{removed}(c) \land \text{removed}(M_b(c))$</td><td><code>- #if m</code>
<code>- c</code>
<code>- #endif</code></td><td></td></tr>
<tr><td>$\mathit{Reconfiguration}(c) := \text{unchanged}(c) \land \neg(PC_b(c) \models PC_a(c)) \land \neg(PC_a(c) \models PC_b(c))$</td><td><code>- #if m</code>
<code>+ #if m'</code>
<code>c</code>
<code>#endif</code></td><td></td></tr>
<tr><td>$\mathit{AddToPC}(c) := \text{added}(c) \land \neg\,\text{added}(M_a(c))$</td><td><code>#if m</code>
<code>+ c</code>
<code>#endif</code></td><td></td></tr>
<tr><td>$\mathit{RemFromPC}(c) := \text{removed}(c) \land \neg\,\text{removed}(M_b(c))$</td><td><code>#if m</code>
<code>- c</code>
<code>#endif</code></td><td></td></tr>
<tr><td>$\mathit{Refactoring}(c) := \text{unchanged}(c) \land (PC_b(c) \models PC_a(c)) \land (PC_a(c) \models PC_b(c)) \land (\mathit{path}_b(c) \neq \mathit{path}_a(c))$</td><td><code>- #if A || (B && !A)</code>
<code>+ #if A || B</code>
<code>c</code>
<code>#endif</code></td><td></td></tr>
<tr><td>$\mathit{Specialization}(c) := \text{unchanged}(c) \land \neg(PC_b(c) \models PC_a(c)) \land (PC_a(c) \models PC_b(c))$</td><td><code>+ #if m</code>
<code>c</code>
<code>+ #endif</code></td><td></td></tr>
<tr><td>$\mathit{Generalization}(c) := \text{unchanged}(c) \land (PC_b(c) \models PC_a(c)) \land \neg(PC_a(c) \models PC_b(c))$</td><td><code>- #if m</code>
<code>c</code>
<code>- #endif</code></td><td></td></tr>
<tr><td>$\mathit{Untouched}(c) := \text{unchanged}(c) \land (PC_b(c) \models PC_a(c)) \land (PC_a(c) \models PC_b(c)) \land (\mathit{path}_b(c) = \mathit{path}_a(c))$</td><td></td><td></td></tr>
</table>

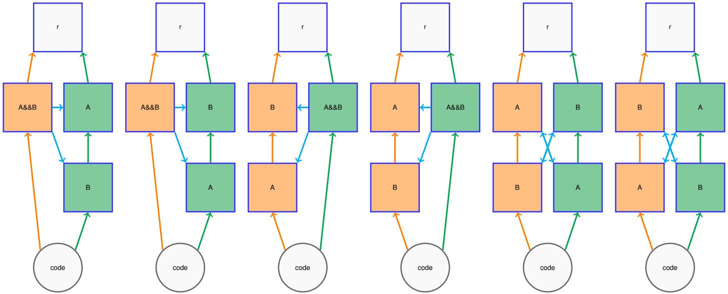
Karlsruher Institut für Technologie

Edit classes

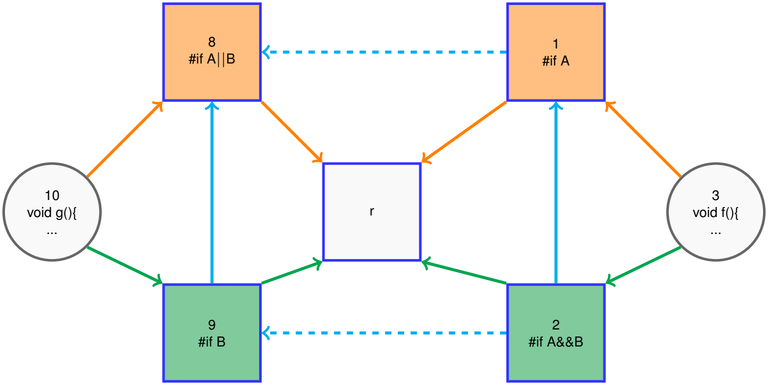
$$\begin{aligned} \mathit{Refactoring}(c) :=\ & \text{unchanged}(c) \\ & \wedge (\mathrm{PC}_b(c) \models \mathrm{PC}_a(c)) \\ & \wedge (\mathrm{PC}_a(c) \models \mathrm{PC}_b(c)) \\ & \wedge (\mathit{path}_b(c) \neq \mathit{path}_a(c)) \end{aligned}$$

```
- #if A || (B && !A)
+ #if A || B
    c
  #endif
```





Optimizations



Optimizations

General:

- Omit paper analysis tooling
- Filter *True* and *False*
- Only compare changed feature mappings
- Pre-analyse files in commits

Edge-Type Specific:

- Bidirectional edges
- Mutual exclusivity of different edge types
- Transitivity

Evaluation of the SAT calls			Relation type
$SAT(I(x) \wedge I(y))$	$SAT(I(x) \wedge \neg I(y))$	$SAT(\neg I(x) \wedge I(y))$	
1	1	1	*common-intersection*
1	1	0	*implication* (I(y) implies I(x))
1	0	1	*implication* (I(x) implies I(y))
1	0	0	*implication* (formula equivalency)
0	1	1	*alternative*
0	1	0	*implication* (I(y) is *False*)
0	0	1	*implication* (I(x) is *False*)
0	0	0	*implication/alternative* (I(x) and I(y) are *False*)

Tabelle: All potential cases the *commonintersection* predicate can succeed and fail for.