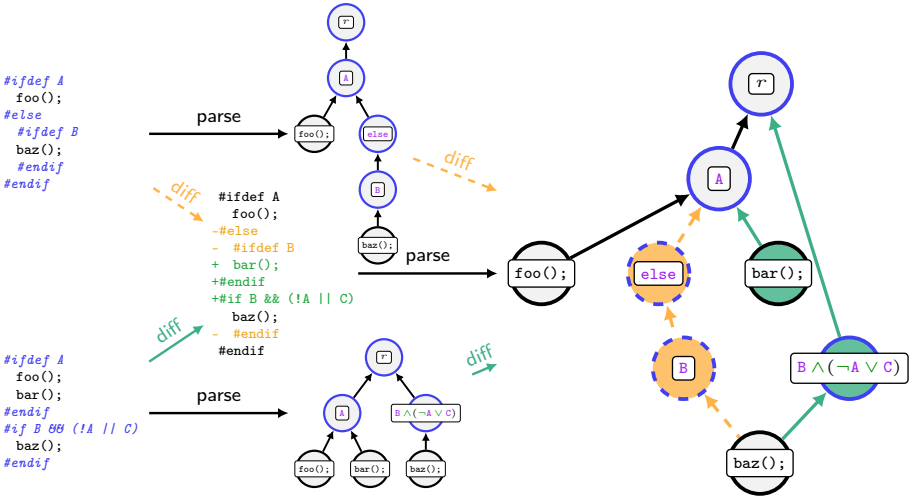


Analyzing Edits to Static Variability

Paul Bittner | PhD Defense

Software Engineering
Programming Languages

universität ulm

Commit ab4cece in 

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

Commit ab4cece in 

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

#define FEAT_GUI

```
  static void
f_foreground(/* params */)
{
  if (gui.in_use)
  gui_mch_set_foreground();
}
```

Commit ab4cece in 

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

`#define FEAT_GUI`

```
  static void
f_foreground(/* params */)
{
  if (gui.in_use)
  gui_mch_set_foreground();
}
```

`#undef FEAT_GUI`
`#define MSWIN`

```
  static void
f_foreground(/* params */)
{
  win32_set_foreground();
}
```

Commit ab4cece in 

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

#define FEAT_GUI

```
  static void
f_foreground(/* params */)
{
  if (gui.in_use)
  gui_mch_set_foreground();
}
```

#undef FEAT_GUI
#define MSWIN

```
  static void
f_foreground(/* params */)
{
  win32_set_foreground();
}
```

#undef FEAT_GUI
#undef MSWIN

```
  static void
f_foreground(/* params */)
{
}
```

Commit ab4cece in 

```
   static void
f_foreground(/* params */)
{
#ifdef FEAT_GUI
   if (gui
   gui_mch
#else
# ifdef MS
   win32_s
# endif
#endif
}
```

#define FEAT_GUI

#ifdef MSWIN

Let's improve code reuse in case `vim.dll` is around!

```
   static void
f_foreground(/* params */)
{
   if (gui.in_use)
   gui_mch_set_foreground();
}
```

```
   static void
f_foreground(/* params */)
{
   win32_set_foreground();
}
```

```
   static void
f_foreground(/* params */)
{
}
```

Hey Bob, I just refactored  to reuse more code! Can you review my change?

Hey Bob, I just refactored  to reuse more code! Can you review my change?

Awesome, of course!

```
 #ifdef FEAT_GUI
     if (gui.in_use)
+    {
         gui_mch_set_foreground();
-#else
-# ifdef MSWIN
+        return;
+    }
+#endif
+#if defined(MSWIN) && (!defined(FEAT_GUI) || defined(VIMDLL))
     win32_set_foreground();
-# endif
 #endif
```

Commit afde13b in 

```
 #ifdef FEAT_GUI
     if (gui.in_use)
+    {
         gui_mch_set_foreground();
-#else
-# ifdef MSWIN
+        return;
+    }
+#endif
+#if defined(MSWIN) && (!defined(FEAT_GUI) || defined(VIMDLL))
     win32_set_foreground();
-# endif
 #endif
```

How does this change improve code reuse?

Commit afde13b in 

```
 #ifdef FEAT_GUI
     if (gui.in_use)
+    {
         gui_mch_set_foreground();
-#else
-# ifdef MSWIN
+        return;
+    }
+#endif
+#if defined(MSWIN) && (!defined(FEAT_GUI) || defined(VIMDLL))
     win32_set_foreground();
-# endif
 #endif
```

Commit afde13b in 

How does this change improve code reuse?

I have to reason on the syntax of three languages simultaneously!

```
>>= vim --version
VIM - Vi IMproved 9.1 (2024 Jan 02, compiled Jan 01 1980 00:00:00)
Included patches: 1-1122
Compiled by nixbld
Huge version without GUI.  Features included (+) or not (-):
+acl               +file_in_path      +mouse_urxvt       -tag_any_white
+arabic            +find_in_path      +mouse_xterm       -tcl
+autocmd           +float             +multi_byte        +termguicolors
+autochdir         +folding           +multi_lang        +terminal
-autoservername    -footer            -mzscheme          +terminfo
-balloon_eval      +fork()            +netbeans_intg     +termresponse
+balloon_eval_term +gettext           +num64             +textobjects
-browse            -hangul_input      +packages          +textprop
++builtin_terms    +iconv             +path_extra        +timers
+byte_offset       +insert_expand     -perl              +title
+channel           +ipv6              +persistent_undo   -toolbar
+cindent           +job               +popupwin          +user_commands
-clientserver      +jumplist          +postscript        +vartabs
-clipboard         +keymap            +printer           +vertsplit
+cmdline_compl     +lambda            +profile           +vim9script
+cmdline_hist      +langmap           -python            +viminfo
+cmdline_info      +libcall           -python3           +virtualedit
+comments          +linebreak         +quickfix          +visual
+conceal           +lispindent        +reltime           +visualextra
+cryptv            +listcmds          +rightleft         +vreplace
+cscope            +localmap          -ruby              +wildignore
+cursorbind        -lua               +scrollbind        +wildmenu
+cursorshape       +menu              +signs             +windows
+dialog_con        +mksession         +smartindent       +writebackup
+diff              +modify_fname      -sodium            -X11
+digraphs          +mouse             -sound             +xattr
-dnd               -mouseshape        +spell             -xfontset
-ebcdic            +mouse_dec         +startuptime       -xim
+emacs_tags        -mouse_gpm         +statusline        -xpm
+eval              -mouse_jsbterm     -sun_workshop      -xsmp
+ex_extra          +mouse_netterm     +syntax            -xterm_clipboard
+extra_search      +mouse_sgr         +tag_binary        -xterm_save
-farsi             -mouse_sysmouse    -tag_old_static
```

```
>>= vim --version
VIM - Vi IMproved 9.1 (2024 Jan 02, compiled Jan 01 1980 00:00:00)
Included patches: 1-1122
Compiled by nixbld
Huge version without GUI.  Features included (+) or not (-):
+acl               +file_in_path      +mouse_urxvt       -tag_any_white
+arabic            +find_in_path      +mouse_xterm       -tcl
+autocmd           +float             +multi_byte        +termguicolors
+autochdir         +folding           +multi_lang        +terminal
-autoservername    -footer            -mzscheme          +terminfo
-balloon_eval      +fork()            +netbeans_intg     +termresponse
+balloon_eval_term +gettext           +num64             +textobjects
-browse            -hangul_input      +packages          +textprop
++builtin_terms    +iconv             +path_extra        +timers
+byte_offset       +insert_expand     -perl              +title
+channel           +ipv6              +persistent_undo   -toolbar
+cindent           +job               +popupwin          +user_commands
-clientserver      +jumplist          +postscript        +vartabs
-clipboard         +keymap            +printer           +vertsplit
+cmdline_compl     +lambda            +profile           +vim9script
+cmdline_hist      +langmap           -python            +viminfo
+cmdline_info      +libcall           -python3           +virtualedit
+comments          +linebreak         +quickfix          +visual
+conceal           +lispindent        +reltime           +visualextra
+cryptv            +listcmds          +rightleft         +vreplace
+cscope            +localmap          -ruby              +wildignore
+cursorbind        -lua               +scrollbind        +wildmenu
+cursorshape       +menu              +signs             +windows
+dialog_con        +mksession         +smartindent       +writebackup
+diff              +modify_fname      -sodium            -X11
+digraphs          +mouse             -sound             +xattr
-dnd               -mouseshape        +spell             -xfontset
-ebcdic            +mouse_dec         +startuptime       -xim
+emacs_tags        -mouse_gpm         +statusline        -xpm
+eval              -mouse_jsbterm     -sun_workshop      -xsmp
+ex_extra          +mouse_netterm     +syntax            -xterm_clipboard
+extra_search      +mouse_sgr         +tag_binary        -xterm_save
-farsi             -mouse_sysmouse    -tag_old_static
```

- $\geq$ 50% of the code is affected by features [Liebig et al. 2010](v7.1)
- $\geq$ 21,000 commits, multiple commits per day

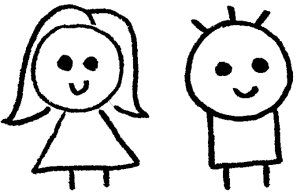
GPL. https://commons.wikimedia.org/wiki/File:The_GIMP_icon_-_gnome.svg

CC BY 4.0. Andrea Calabró. https://github.com/godotengine/godot/blob/master/LOGO_LICENSE.txt

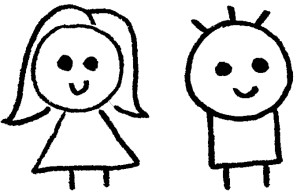
GPL3. Free Software Foundation Inc.. https://commons.wikimedia.org/wiki/File:Emacs-Icon.svg

CC BY-SA. Free Software Foundation Inc.. https://de.wikipedia.org/wiki/GNU_Compiler_Collection#/media/Datei:GNU_Compiler_Collection_logo.svg

Can you *explain* our change?

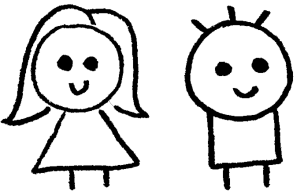


Can you *explain* our change?

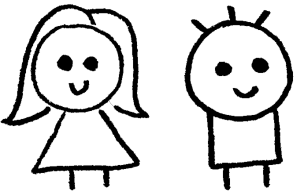


No, you should have asked before doing a change!
[Seidl et al. 2012]

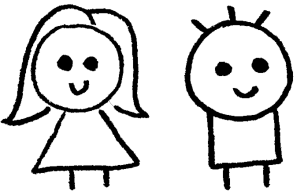
Can you *explain* our change?



Can you *explain* our change?



Can you *explain* our change?



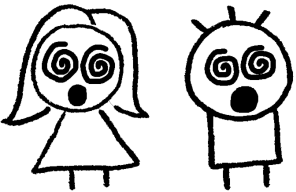
Maybe. We can explain some but not all changes.
[Stănciulescu et al. 2016; Ji et al. 2015; Neves et al. 2015; Schulze et al. 2013; Borba et al. 2012; Schulze et al. 2012; Seidl et al. 2012]

No, you should have asked before doing a change!
[Seidl et al. 2012]

Yes, but you have to apply our classification by hand.
[Passos et al. 2016; Ji et al. 2015; Borba et al. 2012]

Yes, but only if you give us your entire version history.
[Kröher et al. 2023; Dintzner et al. 2018; Fischer et al. 2003]

Can you *explain* our change?



Maybe. We can explain some but not all changes.
[Stănciulescu et al. 2016; Ji et al. 2015; Neves et al. 2015; Schulze et al. 2013; Borba et al. 2012; Schulze et al. 2012; Seidl et al. 2012]

No, you should have asked before doing a change!
[Seidl et al. 2012]

Yes, but you have to apply our classification by hand.
[Passos et al. 2016; Ji et al. 2015; Borba et al. 2012]

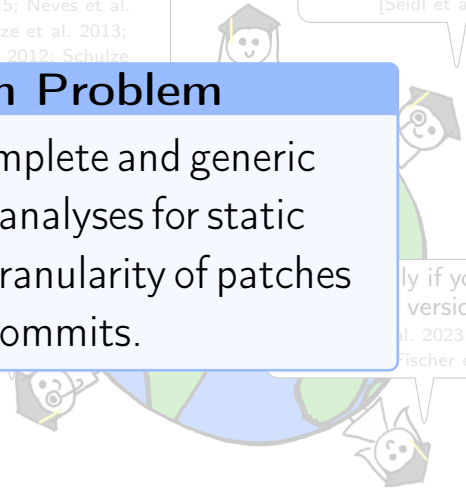
Yes, but only if you give us your entire version history.
[Kröher et al. 2023; Dintzner et al. 2018; Fischer et al. 2003]

Maybe. We can explain some but not all changes. [Stănciulescu et al. 2016; Ji et al. 2015; Neves et al. 2015; Schulze et al. 2013; Borba et al. 2012; Schulze

No, you should have asked before doing a change! [Seidl et al. 2012]

Can you e
cha

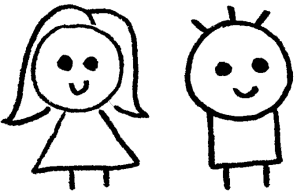
ly if you give us version history.
l. 2023; Dintzner
Fischer et al. 2003]



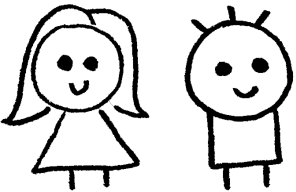
Research Problem

There are no complete and generic change impact analyses for static variability at the granularity of patches and commits.

Do you have a *theory for edits to variability* so that we can develop analyses ourselves?

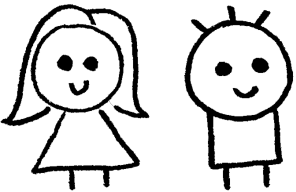


Do you have a *theory for edits to variability* so that we can develop analyses ourselves?



Sorry, but none of our numerous differencing algorithms is *variability-aware*! [Erdweg et al. 2021; Nugroho et al. 2020; Dotzler and Philippsen 2016; Falleri et al. 2014; Asaduzzaman et al. 2013; Canfora et al. 2009; Fluri et al. 2007; Apiwattanapong et al. 2004]

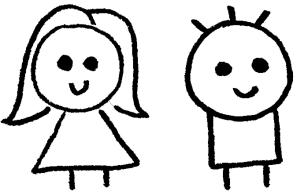
Do you have a *theory for edits to variability* so that we can develop analyses ourselves?



Sorry, but none of our numerous differencing algorithms is *variability-aware*! [Erdweg et al. 2021; Nugroho et al. 2020; Dotzler and Philippsen 2016; Falleri et al. 2014; Asaduzzaman et al. 2013; Canfora et al. 2009; Fluri et al. 2007; Apiwattanapong et al. 2004]

Sorry, but our theories have no claim on completeness. [Ananieva et al. 2022; Stănculescu et al. 2016; Neves et al. 2015; Schulze et al. 2013; Borba et al. 2012; Schulze et al. 2012]

Do you have a *theory for edits to variability* so that we can develop analyses ourselves?



Sorry, but none of our numerous differencing algorithms is *variability-aware*! [Erdweg et al. 2021; Nugroho et al. 2020; Dotzler and Philippsen 2016; Falleri et al. 2014; Asaduzzaman et al. 2013; Canfora et al. 2009; Fluri et al. 2007; Apiwattanapong et al. 2004]

Sorry, but our theories have no claim on completeness. [Ananieva et al. 2022; Stănciulescu et al. 2016; Neves et al. 2015; Schulze et al. 2013; Borba et al. 2012; Schulze et al. 2012]

Sorry, but our tools and models are not generic. [Kröher et al. 2023; Dintzner et al. 2018; Al-Hajjaji et al. 2016; Fischer et al. 2003]

Sorry, but our theories have no claim on completeness. [Ananieva et al. 2022; Stănciulescu et al. 2016; Neves et al. 2015; Schulze et al. 2013; Borba et al. 2012; Schulze et al. 2012]

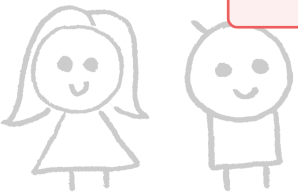
Do you have a *theory for edits to variability* so that we can develop analyses

Sorry, but none of our numerous differencing algo-

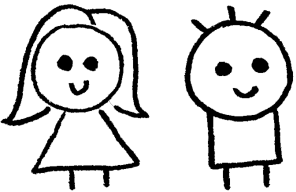
Research Problem

There is no formal and generic theory for changes to static variability.

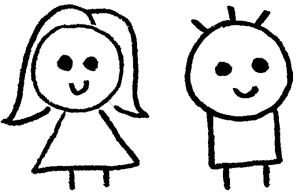
our tools and re not generic. l. 2023; Dintzner et al. 2018; Al-Hajjaji et al. 2016; Fischer et al. 2003]



To develop a theory for edits to variability, we need a *theory of variability* first, right?

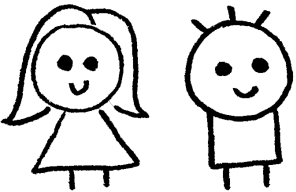


To develop a theory for edits to variability, we need a *theory of variability* first, right?



Yes, all you need is the *choice calculus*! After all, it is the λ-calculus of software variability.
[Walkingshaw 2013; Erwig and Walkingshaw 2011]

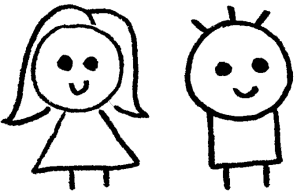
To develop a theory for edits to variability, we need a *theory of variability* first, right?



Yes, all you need is the *choice calculus*! After all, it is the λ-calculus of software variability.
[Walkingshaw 2013; Erwig and Walkingshaw 2011]

We prefer *algebraic decision diagrams*, much simpler!
[Castro et al. 2021]

To develop a theory for edits to variability, we need a *theory of variability* first, right?

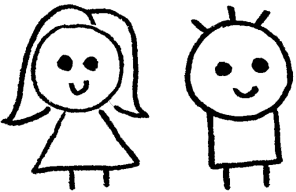


Yes, all you need is the *choice calculus*! After all, it is the λ-calculus of software variability. [Walkingshaw 2013; Erwig and Walkingshaw 2011]

We prefer *algebraic decision diagrams*, much simpler! [Castro et al. 2021]

All you need is my dissertation! [Gruler 2010]

To develop a theory for edits to variability, we need a *theory of variability* first, right?



Yes, all you need is the *choice calculus*! After all, it is the λ-calculus of software variability. [Walkingshaw 2013; Erwig and Walkingshaw 2011]

We prefer *algebraic decision diagrams*, much simpler! [Castro et al. 2021]

All you need is my dissertation! [Gruler 2010]

Don't forget our feature algebra! [Apel et al. 2013; Apel et al. 2010]

To develop a theory for edits to variability, we need a [illegible] *variability*

Yes, all you need is the *choice calculus*! After all, it is the λ-calculus of software variability. [Walkingshaw 2013; Erwig and Walkingshaw 2011]

We prefer *algebraic decision diagrams*, much simpler! [Castro et al. 2021]

Research Problem

It is unknown how existing formal languages for static variability relate and whether there is a common, rigorous model for static variability.

[illegible] forget our [illegible]e algebra! [illegible]et al. 2013; [illegible]et al. 2010]

Analyzing Edits to Static Variability

Research Problem 3:
There are no complete and generic change impact analyses for static variability at the granularity of patches and commits.

Research Problem 2:
There is no formal and generic theory for changes to static variability.

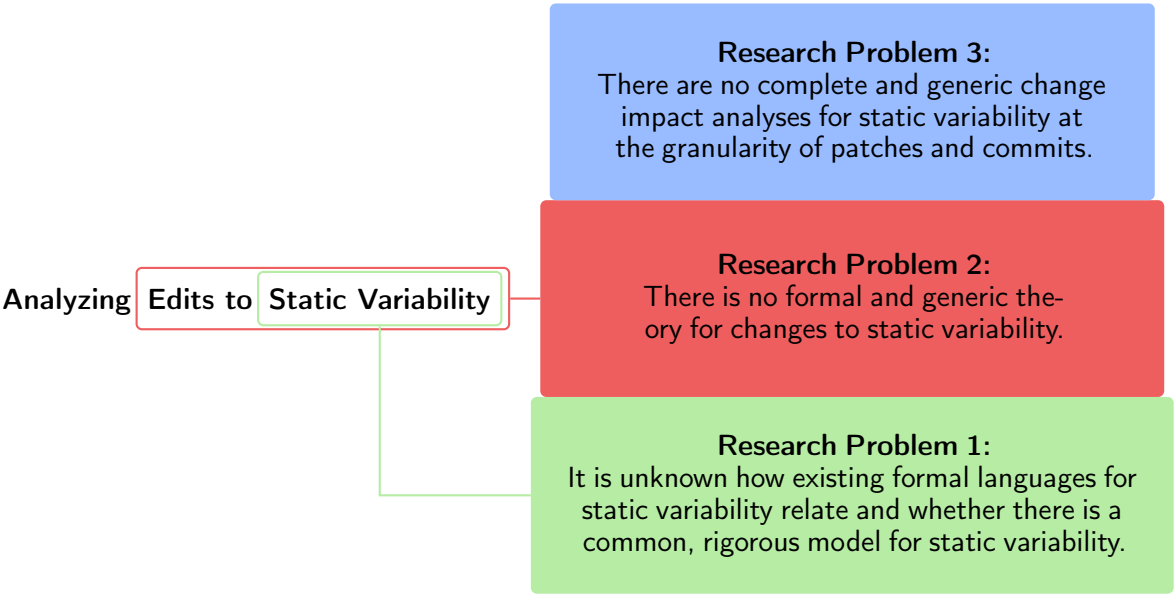
Research Problem 1:
It is unknown how existing formal languages for static variability relate and whether there is a common, rigorous model for static variability.

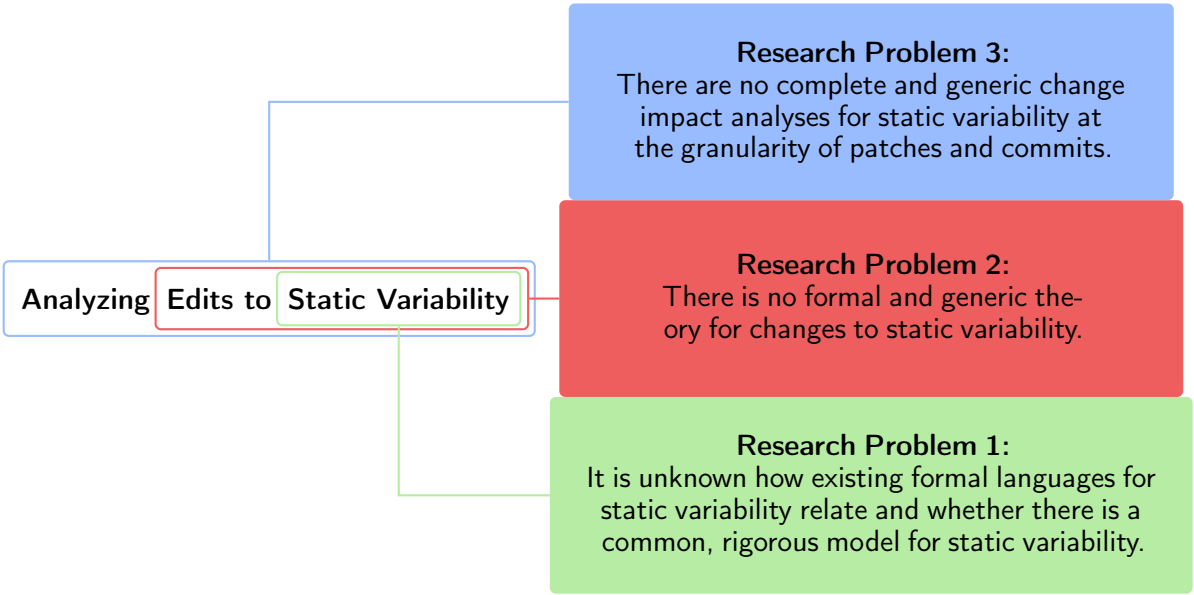
Analyzing Edits to Static Variability

Research Problem 3:
There are no complete and generic change impact analyses for static variability at the granularity of patches and commits.

Research Problem 2:
There is no formal and generic theory for changes to static variability.

Research Problem 1:
It is unknown how existing formal languages for static variability relate and whether there is a common, rigorous model for static variability.


Analyzing Edits to Static Variability
Research Problem 3:
There are no complete and generic change impact analyses for static variability at the granularity of patches and commits.
Research Problem 2:
There is no formal and generic theory for changes to static variability.
Research Problem 1:
It is unknown how existing formal languages for static variability relate and whether there is a common, rigorous model for static variability.


Analyzing Edits to Static Variability
Research Problem 3:
There are no complete and generic change impact analyses for static variability at the granularity of patches and commits.
Research Problem 2:
There is no formal and generic theory for changes to static variability.
Research Problem 1:
It is unknown how existing formal languages for static variability relate and whether there is a common, rigorous model for static variability.

On the Expressive Power of Languages for Static Variability
Bittner, Schultheiß, Moosherr, Young, Teixeira, Walkingshaw, Ataei, Thüm
OOPSLA'24,  , Distinguished Artifact 

Research Problem 1
Static Variability


DINER


VEGETARIAN
WHICH WICH WOULD YOU LIKE?
TRIPLE CHEESE MELT
ELVIS WICH (Honey & Bananas)
TOMATO & AVOCADO
BLACK BEAN PATTY
HUMMUS & BELL PEPPERS
CHOOSE YOUR BREAD
WHEAT
WHITE
CHOOSE YOUR CHEESE (Optional)
AMERICAN
SWISS
PROVOLONE
CHEDDAR
PEPPER JACK
MOZZARELLA
How Would You Like Your WICH Worked?
MUSTARDS
Yellow
Dijon
Honey
Deli
MAYOS
Regular
Lite
Horseradish
Spicy
SPREADS & SAUCES
BBQ
Buffalo
Marinara
1000 Island
Ranch
ONIONS
Red
Grilled
Crispy Strings
VEGGIES
Lettuce
Tomato
Pickles
Jalapenos
Olive Salad
Mushrooms
Sauerkraut
Coleslaw
Bell Peppers
OILS & SPICES
Oil
Vinegar
Oregano
Parmesan

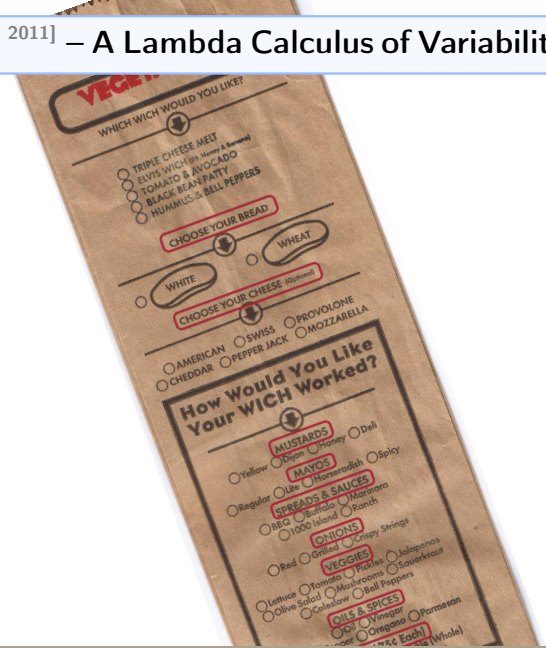
Does this relate to our problems?

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

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

The Choice Calculus [Erwig and Walkingshaw 2011] [Walkingshaw 2013] – A Lambda Calculus of Variability?

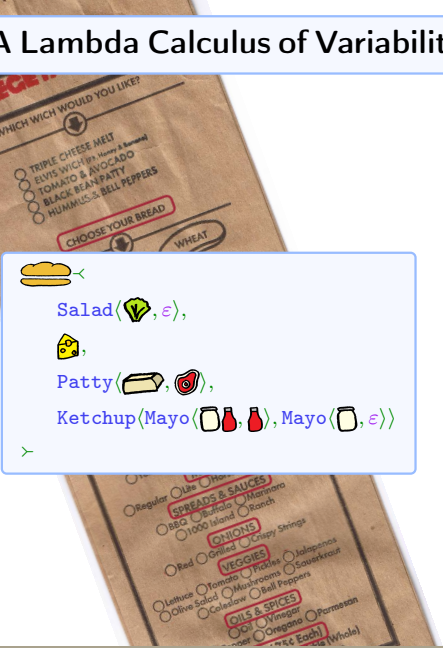
```
f_foreground(/* params */)
{
#ifdef FEAT_GUI
  if (gui.in_use)
  gui_mch_set_foreground();
  #else
# ifdef MSWIN
  win32_set_foreground();
# endif
#endif
}
```



The Choice Calculus [Erwig and Walkingshaw 2011] [Walkingshaw 2013] – A Lambda Calculus of Variability?

```
f_foreground(/* params */)
{
```

```
block≺
    "static void",
    "f_foreground(/* params */)",
    "{",
    FEAT_GUI⟨
        block≺
            "if (gui.in_use)",
            "gui_mch_set_foreground();"
        ≻,
        MSWIN⟨
            block≺"win32_set_foreground();"≻,
            block≺""≻
        ⟩
    ⟩
≻
```

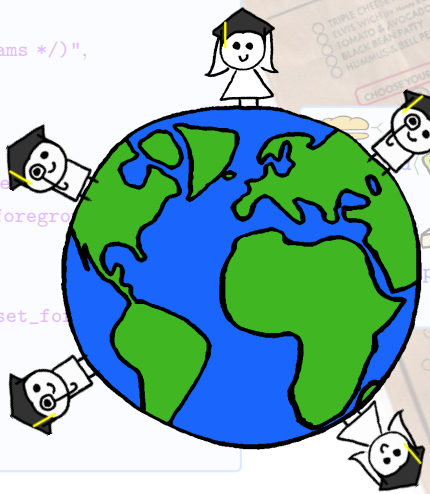



```
≺
    Salad⟨, ε⟩,
    ,
    Patty⟨, ⟩,
    Ketchup⟨Mayo⟨, ⟩, Mayo⟨, ε⟩⟩
≻
```

The Choice Calculus [Erwig and Walkingshaw 2011] [Walkingshaw 2013] – A Lambda Calculus of Variability?

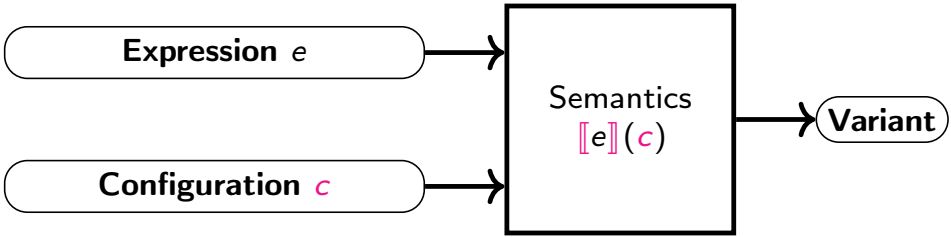
```
f_foreground(/* params */)
{
```

```
block≺
    "static void",
    "f_foreground(/* params */)",
    "{",
    FEAT_GUI⟨
        block≺
            "if (gui.in_use",
            "gui_mch_set_foregro
        ≻,
        MSWIN⟨
            block≺"win32_set_fo
            block≺""≻
        ⟩
    ⟩
≻
```



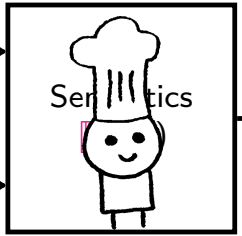
The Choice Calculus [Erwig and W[illegible] 2011] [Walkingsha[illegible]] da Calculus of Variability?

Language and Origin	Syntax e	Configuration Language C	Variant Language $\mathcal{V}$	Semantics $[\![\_]\!] : e \to C \to \mathcal{V}$
Core Choice Calculus (CCC) [Erwig and Walkingshaw 2011] [Walkingshaw 2013]	$e ::= a\prec e^*\succ \mid D\langle e^+\rangle$	$\mathbb{F} \to \mathbb{N}$	Trees $\mathbb{T}$	$[\![a\prec l\succ]\!](c) := a\prec \mathrm{map}([\![\_]\!](c), l)\succ$ $[\![D\langle l\rangle]\!](c) := [\![l\downarrow_{c(D)}]\!](c)$
Binary Choice Calculus (2CC) [many]	$e ::= a\prec e^*\succ \mid D\langle e, e\rangle$	$\mathbb{F} \to \mathbb{B}$	Trees $\mathbb{T}$	$[\![a\prec l\succ]\!](c) := a\prec \mathrm{map}([\![\_]\!](c), l)\succ$ $[\![D\langle l, r\rangle]\!](c) := \begin{cases} [\![l]\!](c) & c(D) = \mathit{true} \\ [\![r]\!](c) & \text{else} \end{cases}$
Algebraic Decision Trees (ADT) [Bahar et al. 1993] [Castro et al. 2021]	$e ::= \mathrm{leaf}\ a \mid D\langle e, e\rangle$	$\mathbb{F} \to \mathbb{B}$	Atoms $\mathbb{A}$	$[\![\mathrm{leaf}\ v]\!](c) := v$ $[\![D\langle l, r\rangle]\!](c) := \begin{cases} [\![l]\!](c) & c(D) = \mathit{true} \\ [\![r]\!](c) & \text{else} \end{cases}$
Gruler's Language (GL) [Gruler 2010]	$e ::= \mathrm{ntrl} \mid \mathrm{asset}\ a \mid e \parallel e \mid e \oplus_{n\in\mathbb{N}} e$	$\mathbb{N} \to \mathbb{B}$	Lists $\mathbb{L}$	$[\![\mathrm{ntrl}]\!](c) := [\,]$ $[\![\mathrm{asset}\ a]\!](c) := a :: [\,]$ $[\![l \parallel r]\!](c) := \mathrm{append}([\![l]\!](c), [\![r]\!](c))$ $[\![l \oplus_n r]\!](c) := \begin{cases} [\![l]\!](c) & c(n) = \mathit{true} \\ [\![r]\!](c) & \text{else} \end{cases}$
Option Calculus (OC) **[new]**	$e ::= a\prec t^*\succ$ $t ::= e \mid O\langle t\rangle$	$\mathbb{F} \to \mathbb{B}$	Trees $\mathbb{T}$ with a fixed root	$[\![a\prec l\succ]\!](c) := a\prec \kappa(\mathrm{map}([\![\_]\!](c), l))\succ$ $[\![O\langle e\rangle]\!](c) := \begin{cases} [\![e]\!](c) & c(O) = \mathit{true} \\ \varepsilon & \text{else} \end{cases}$
Feature Structure Trees (FST) [Apel et al. 2010] [Apel et al. 2013]	$e ::= a\blacktriangleleft f^*\blacktriangleright$ $f ::= F : \underline{e}^*$	$\mathbb{F} \to \mathbb{B}$	Trees $\mathbb{T}$ with a fixed root but without duplicate neighbors	$\mathrm{select}(c) := \mathrm{map}(\lambda(F : l) \to l) \circ \mathrm{filter}(\lambda(F : l) \to c(F))$ $[\![a\blacktriangleleft l\blacktriangleright]\!](c) := a\prec \mathrm{fold}^r(\_ \oplus \_, [\,], \mathrm{select}(c)(l))\succ$


Expression e
Configuration c
Semantics
$[\![e]\!](c)$
Variant

Expression *e*

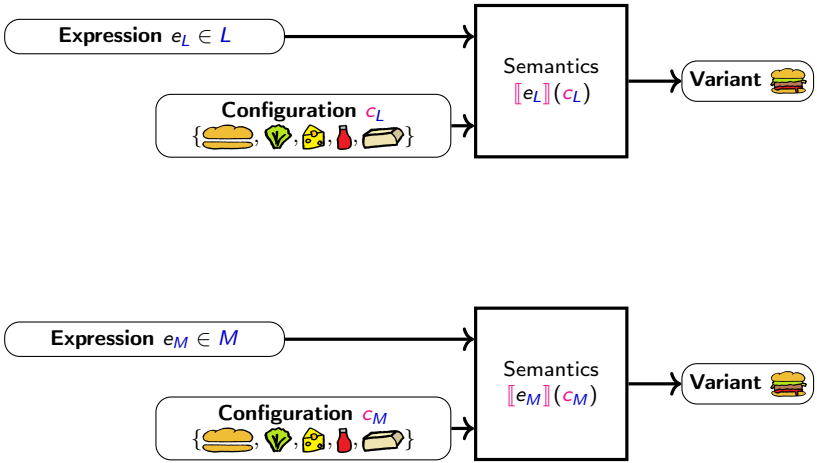
Configuration *c*



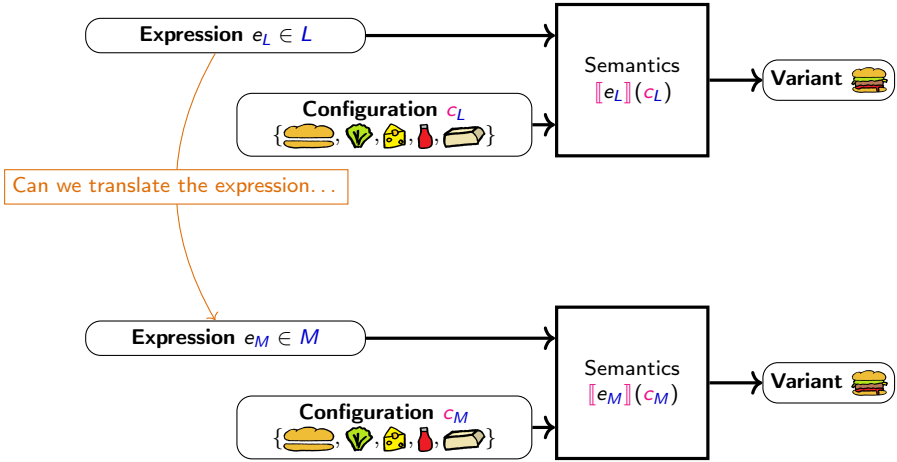
Variant

How to Compare Variability Languages L and M semantically?

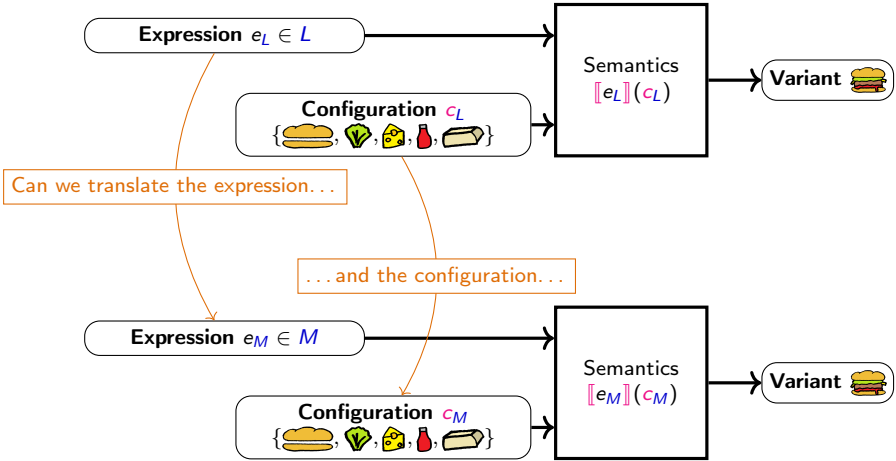
How to Compare Variability Languages L and M semantically?



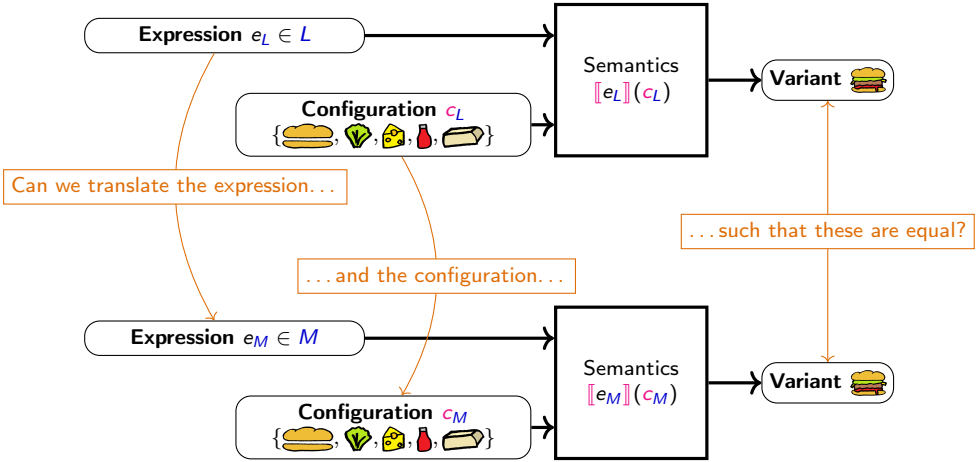
How to Compare Variability Languages L and M semantically?



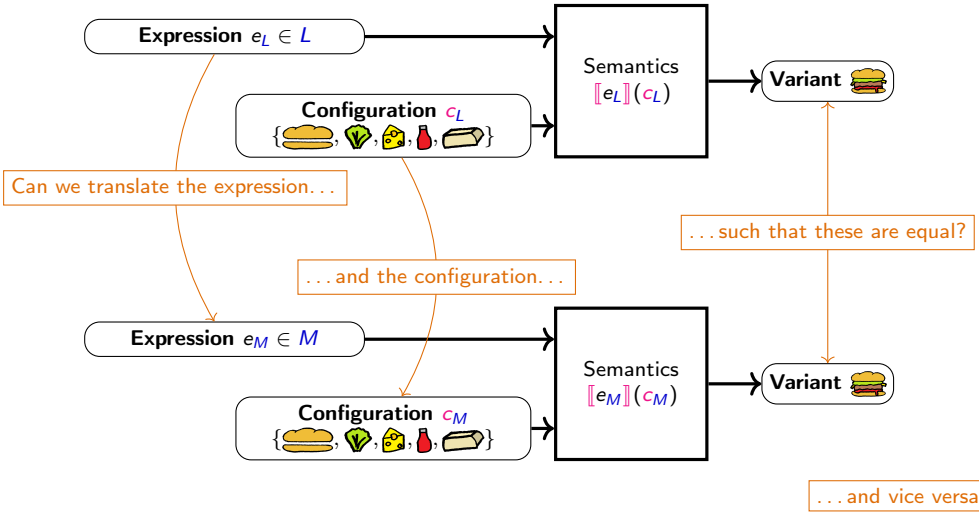
How to Compare Variability Languages L and M semantically?



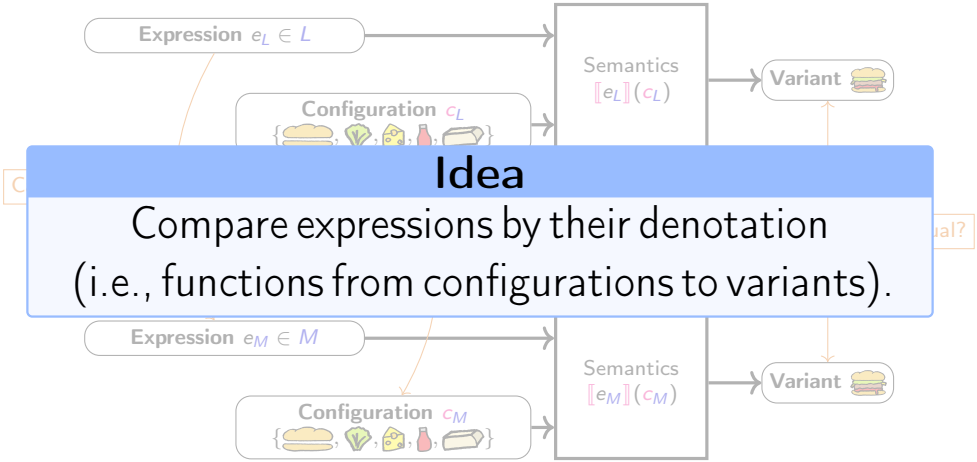
How to Compare Variability Languages L and M semantically?

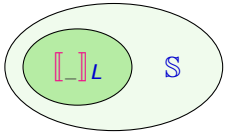


How to Compare Variability Languages L and M semantically?

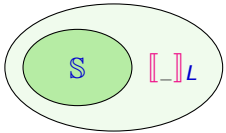


How to Compare Variability Languages L and M semantically?

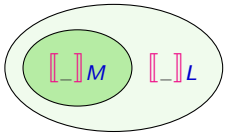


$\text{Sound}(\mathbb{S}, L) \quad := \quad$ 

(Upper bound for expressiveness)

$\text{Complete}(\mathbb{S}, L) \quad := \quad$ 

(Lower bound for expressiveness)

$L \succeq M \quad := \quad$ 

(Relative expressiveness)

Binary Choice Calculus (2CC)

Algebraic Decision Trees (ADT)

Core Choice Calculus (CCC)

Feature Structure Trees (FST)

New language to cover essence of optional variability
(i.e., *"Pick*  *or don't"*).

Binary Choice Calculus (2CC)

Option Calculus (OC)[new]

Algebraic Decision Trees (ADT)

Core Choice Calculus (CCC)

Feature Structure Trees (FST)

Formalize more languages to simplify proofs.

n-ary Choice Calculus (*n*-CC)

Binary Choice Calculus (2CC)

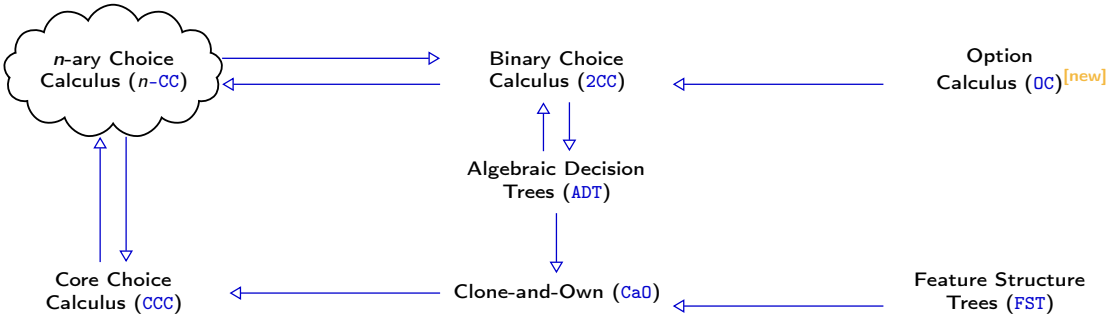
Option Calculus (OC)[new]

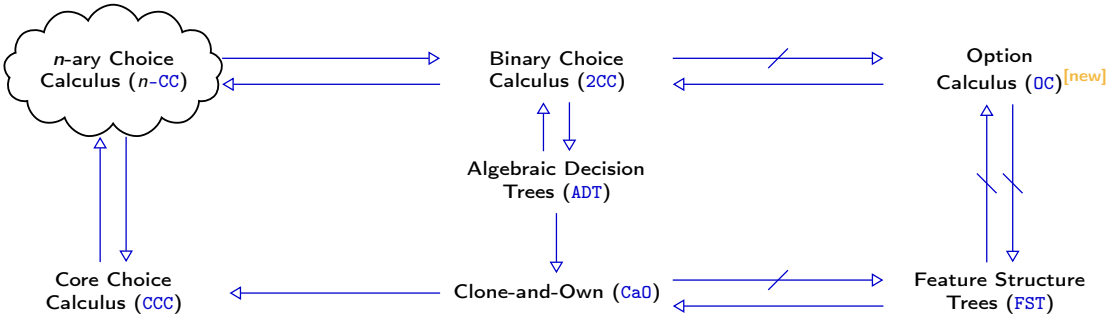
Algebraic Decision Trees (ADT)

Core Choice Calculus (CCC)

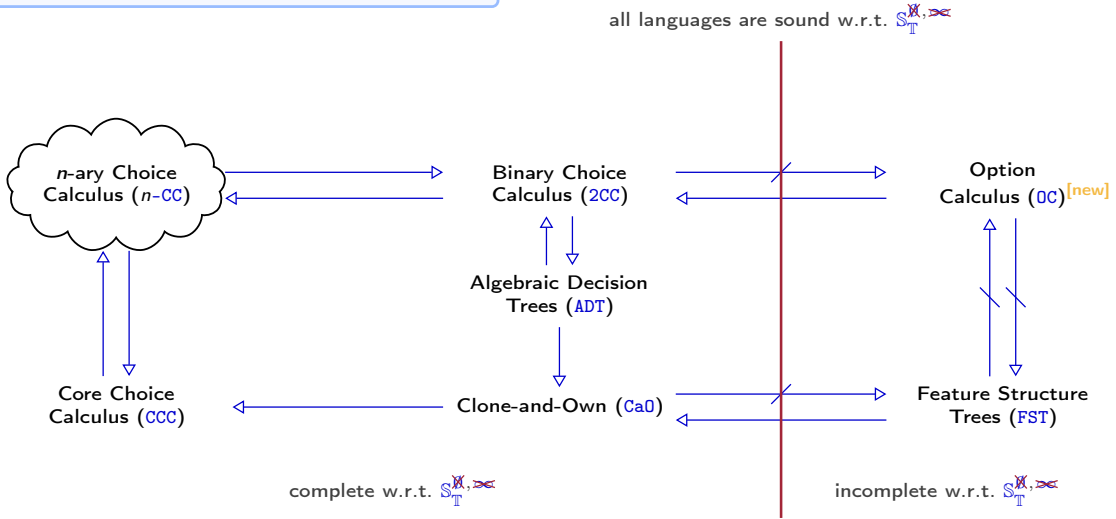
Clone-and-Own (CaO)

Feature Structure Trees (FST)


n-ary Choice Calculus (*n*-CC)
Binary Choice Calculus (2CC)
Option Calculus (OC)[new]
Algebraic Decision Trees (ADT)
Core Choice Calculus (CCC)
Clone-and-Own (CaO)
Feature Structure Trees (FST)


n-ary Choice Calculus (*n*-CC)
Binary Choice Calculus (2CC)
Option Calculus (OC)
[new]
Algebraic Decision Trees (ADT)
Core Choice Calculus (CCC)
Clone-and-Own (CaO)
Feature Structure Trees (FST)

$\mathbb{S}_{\mathrm{T}}^{\not\Delta,\not\asymp}$:= non-empty, finite sets of variants



Bittner et al., OOPSLA'24

Everything is formalized in Agda

Everything is formalized in Agda

Proofs are programs:

Bittner et al., OOPSLA'24

Everything is formalized in Agda

Proofs are programs:

A **soundness** proof enumerates all variants:

Everything is formalized in Agda

Proofs are programs:

A **soundness** proof enumerates all variants:

A **completeness** proof creates an expression from variants:

Bittner et al., OOPSLA'24

Everything is formalized in Agda

Proofs are programs:

A **soundness** proof enumerates all variants:

A **completeness** proof creates an expression from variants:

An **expressiveness** proof is a compiler:

Research Problem 3: Analyzing Edits to Variability

Research Problem 2: Edits to Static Variability

Research Problem 1: Static Variability

Research Problem 3: Analyzing Edits to Variability

Research Problem 2: Edits to Static Variability

Research Problem 1: Static Variability

Foundations and meta-theory

Unified syntax and semantics of languages

Landscape of languages:
· choice calculus is a λ-calculus of variability
· multiple levels of expressiveness

Bittner et al., OOPSLA'24, 
Distinguished Artifact
Vatras – Agda Formalization

Now we understand the foundations of static variability.

Now we understand the foundations of static variability.

So now we can start thinking about a theory for *edits* to variability.

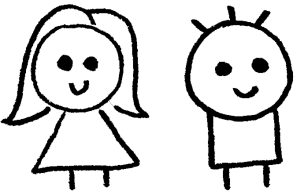
Classifying Edits to Variability in Source Code
Bittner, Tinnes, Schultheiß, Viegener, Kehrer, Thüm
ESEC/FSE'22,

Views on Edits to Variational Software
Bittner, Schultheiß, Greiner, Moosherr, Krieter, Tinnes, Kehrer, Thüm
SPLC'23,

Variability-Aware Differencing with DiffDetective
Bittner, Schultheiß, Moosherr, Kehrer, Thüm
FSE Demonstrations'24, , Best Demonstrations Paper

Research Problem 2
Edits to Static Variability

Do you have a *theory for edits to variability* so that we can develop change impact analyses?



Sorry, but our theories have no claim on completeness. [Ananieva et al. 2022; Stănciulescu et al. 2016; Neves et al. 2015; Schulze et al. 2013; Borba et al. 2012; Schulze et al. 2012]

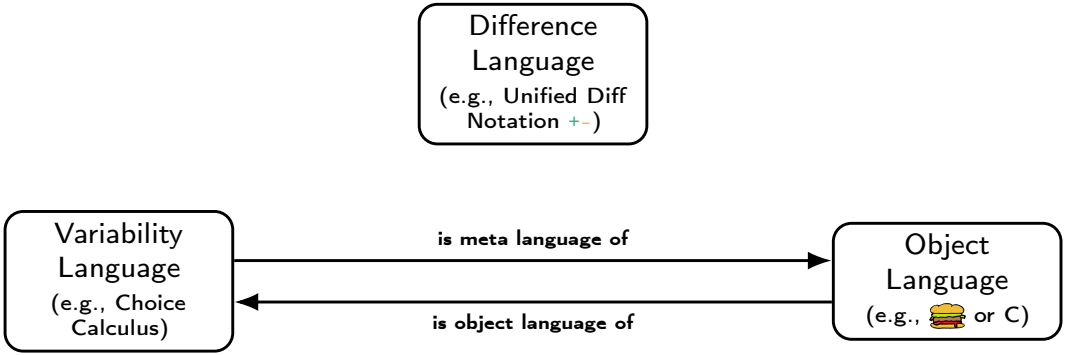
Sorry, but none of our numerous differencing algorithms is *variability-aware*! [Erdweg et al. 2021; Nugroho et al. 2020; Dotzler and Philippsen 2016; Falleri et al. 2014; Asaduzzaman et al. 2013; Canfora et al. 2009; Fluri et al. 2007; Apiwattanapong et al. 2004]

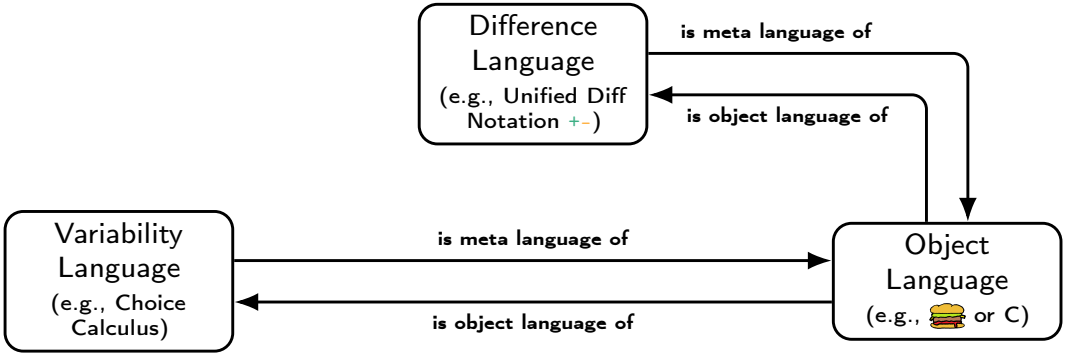
Sorry, but our tools and models are not generic. [Kröher et al. 2023; Dintzner et al. 2018; Al-Hajjaji et al. 2016; Fischer et al. 2003]

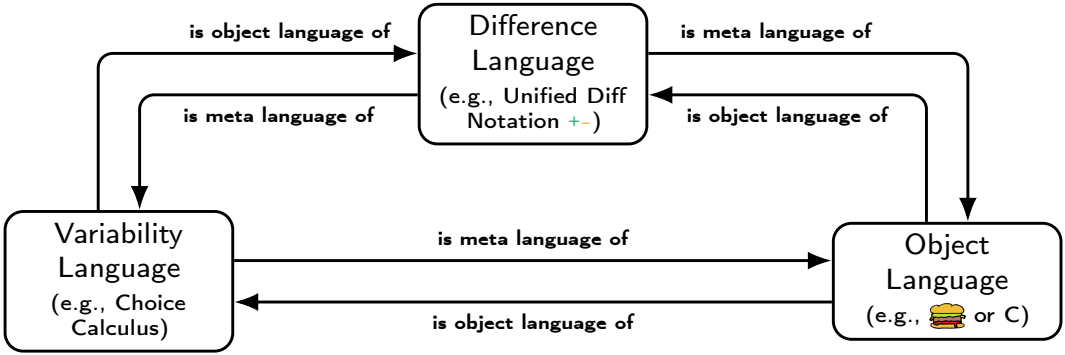
Difference Language
(e.g., Unified Diff Notation +-)

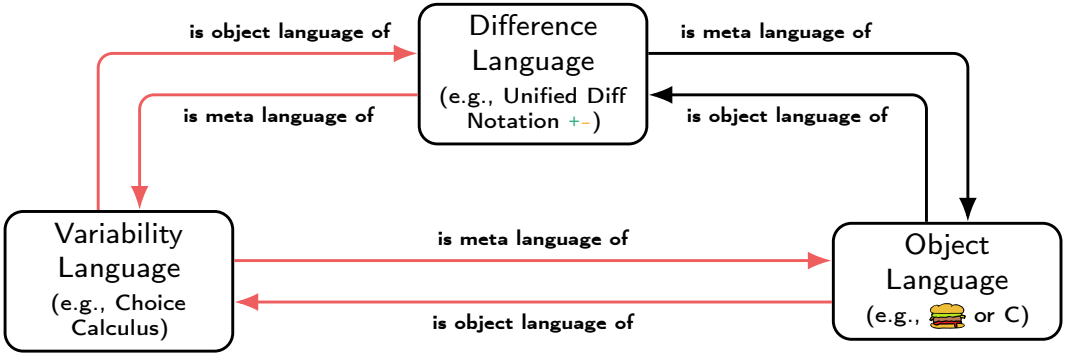
Variability Language
(e.g., Choice Calculus)

Object Language
(e.g.,  or C)


Difference Language
(e.g., Unified Diff Notation +-)
Variability Language
(e.g., Choice Calculus)
is meta language of
is object language of
Object Language
(e.g., or C)


Difference Language
(e.g., Unified Diff Notation +-)
is meta language of
is object language of
Variability Language
(e.g., Choice Calculus)
is meta language of
is object language of
Object Language
(e.g., or C)


Difference Language
(e.g., Unified Diff Notation +-)
is object language of
is meta language of
is meta language of
is object language of
Variability Language
(e.g., Choice Calculus)
is meta language of
is object language of
Object Language
(e.g., or C)



Goal: Make the variability language explicit.

Bittner et al., FSE Demonstrations'24

Bittner et al., ESEC/FSE'22

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

diff

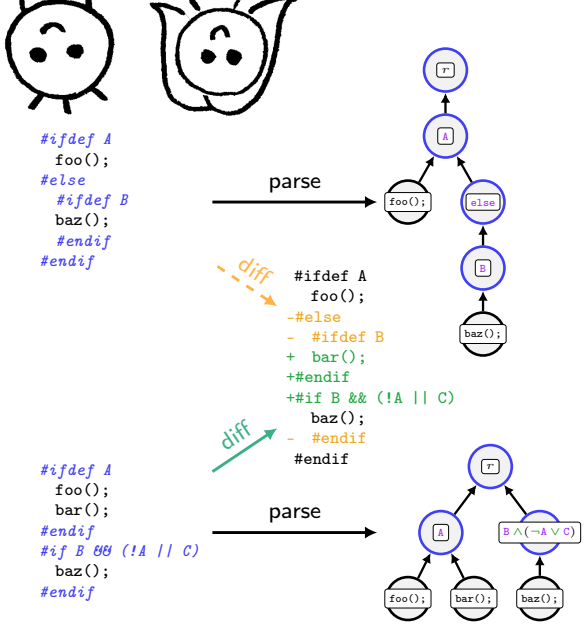
```
 #ifdef A
   foo();
-#else
-  #ifdef B
+  bar();
+#endif
+#if B && (!A || C)
   baz();
-  #endif
 #endif
```

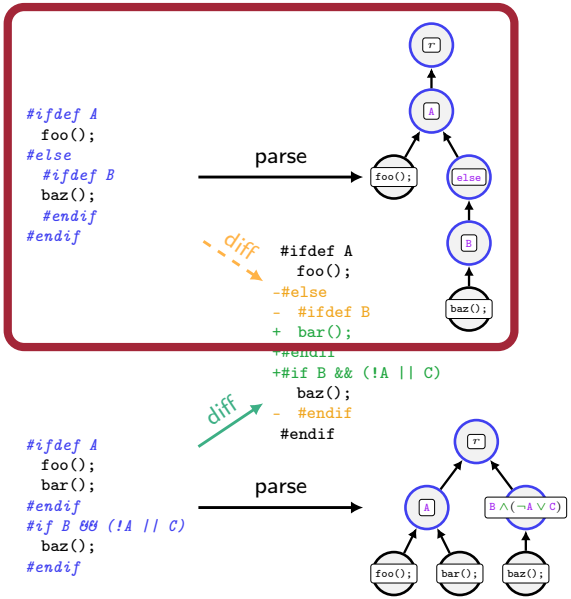
diff

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

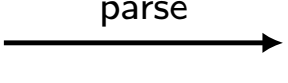
Bittner et al., FSE Demonstrations'24

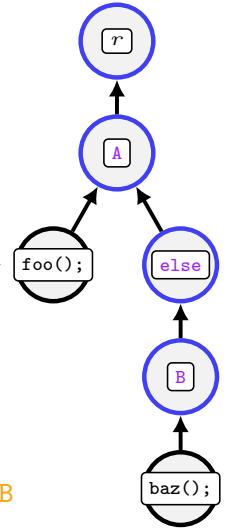
Bittner et al., ESEC/FSE'22




#ifdef A
foo();
#else
#ifdef B
baz();
#endif
#endif
parse
r
A
foo();
else
B
baz();
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
r
A
B ∧(¬A ∨ C)
foo();
bar();
baz();

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

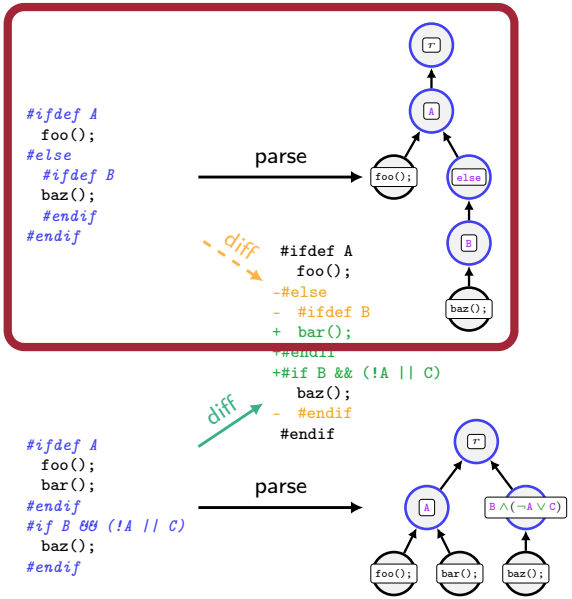


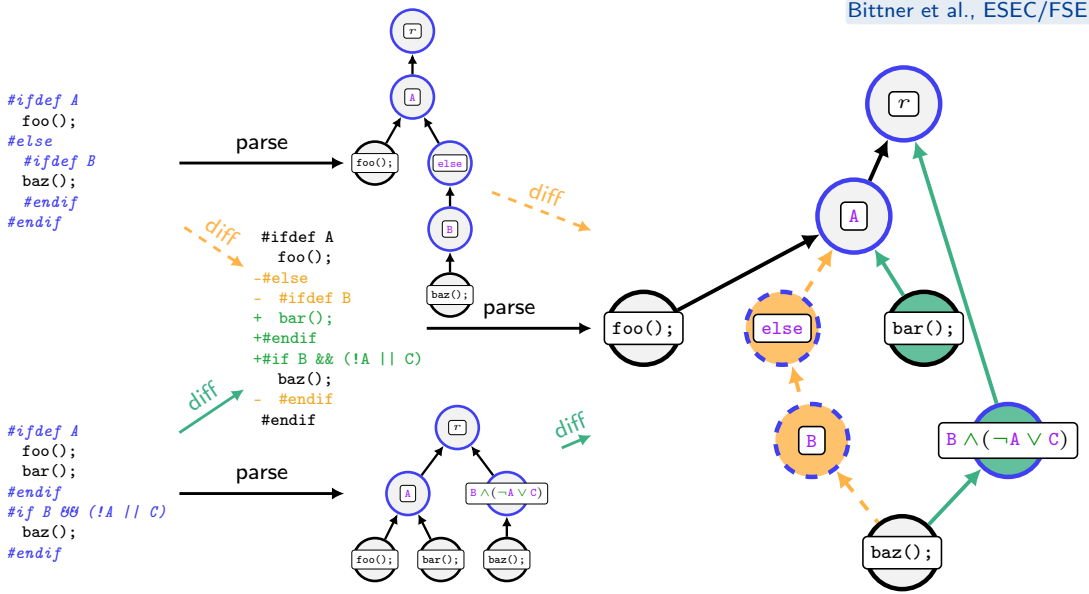


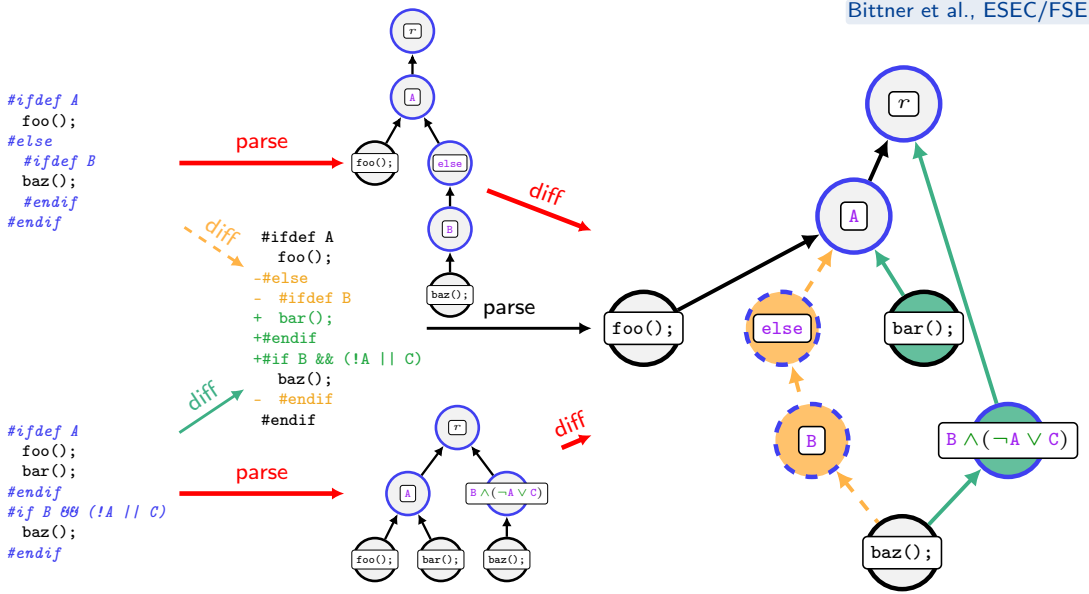

```
 #ifdef A
   foo();
-#else
-  #ifdef B
+  bar();
+#endif
+#if B && (!A || C)
```

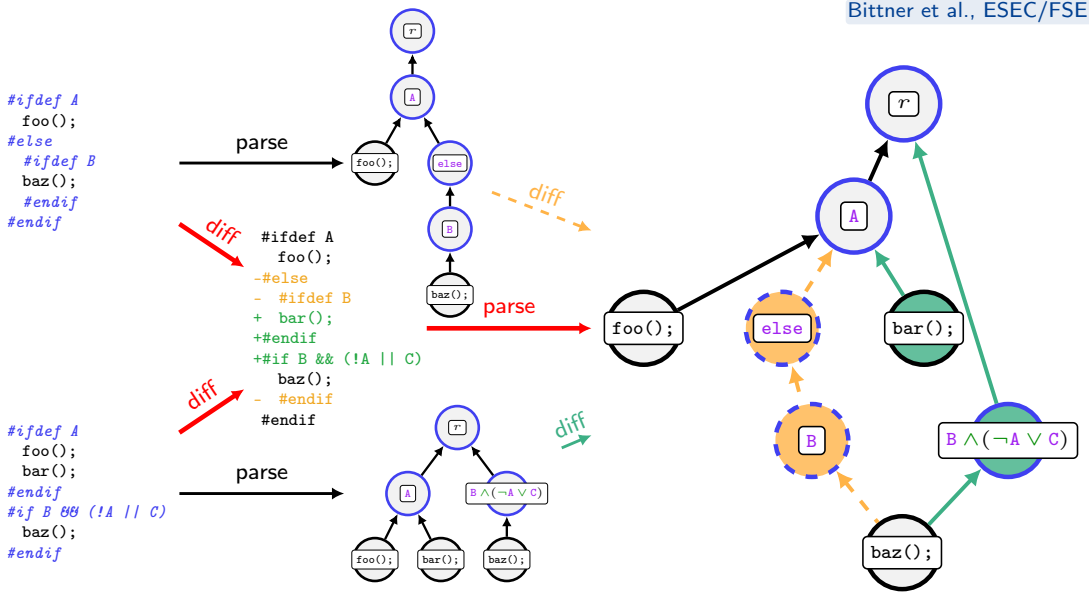
Bittner et al., FSE Demonstrations'24

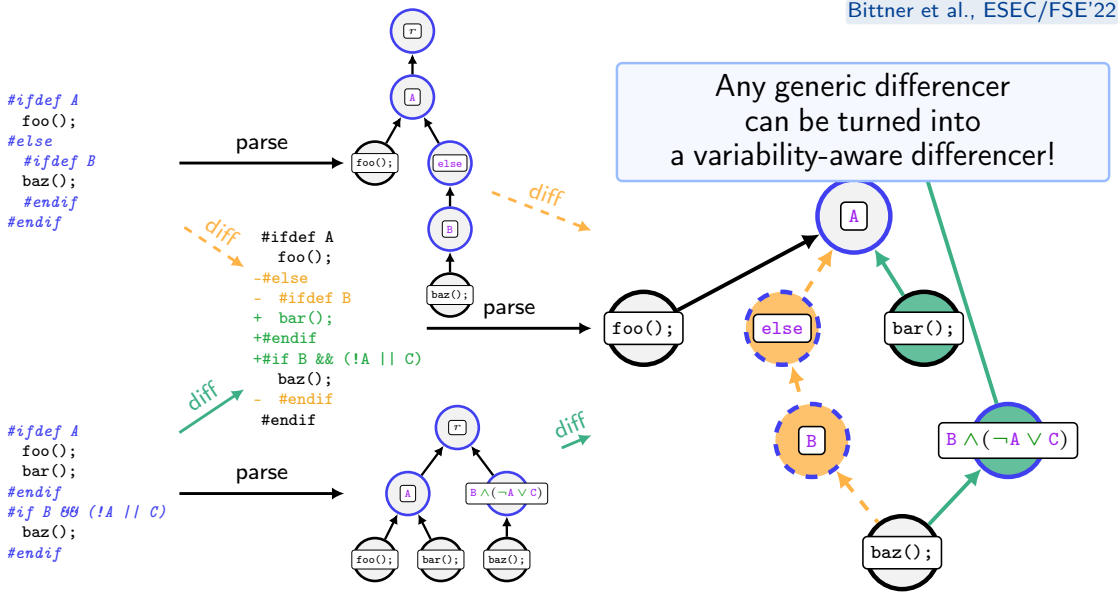
Bittner et al., ESEC/FSE'22



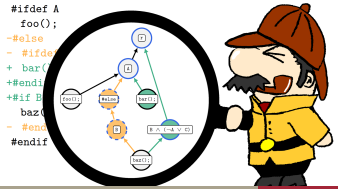

#ifdef A
foo();
#else
#ifdef B
baz();
#endif
#endif
parse
r
A
foo();
else
B
baz();
diff
#ifdef A
foo();
-#else
- #ifdef B
+ bar();
+#endif
+#if B && (!A || C)
baz();
- #endif
#endif
parse
foo();
else
bar();
B
baz();
A
r
B ∧ (¬A ∨ C)
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
parse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


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

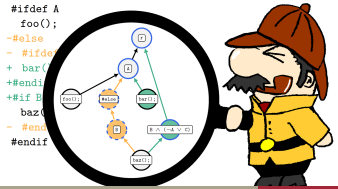

#ifdef A
foo();
#else
#ifdef B
baz();
#endif
#endif
parse
r
A
foo();
else
B
baz();
diff
#ifdef A
foo();
-#else
- #ifdef B
+ bar();
+#endif
+#if B && (!A || C)
baz();
- #endif
#endif
parse
diff
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
parse
r
A
B ∧ (¬A ∨ C)
foo();
bar();
baz();
diff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


#ifdef A
foo();
#else
#ifdef B
baz();
#endif
#endif
parse
r
A
foo();
else
B
baz();
diff
#ifdef A
foo();
-#else
- #ifdef B
+ bar();
+#endif
+#if B && (!A || C)
baz();
- #endif
#endif
parse
#ifdef A
foo();
bar();
#endif
#if B && (!A || C)
baz();
#endif
B ∧ (¬A ∨ C)
bar();
Any generic differencer
can be turned into
a variability-aware differencer!

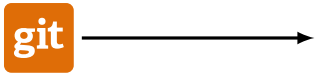
DiffDetective
A Framework for Variability-Aware Differencing

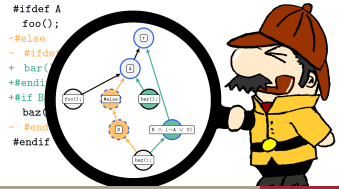


DiffDetective
A Framework for Variability-Aware Differencing

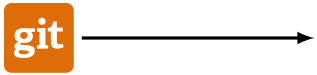


DiffDetective
A Framework for Variability-Aware Differencing

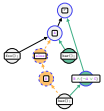




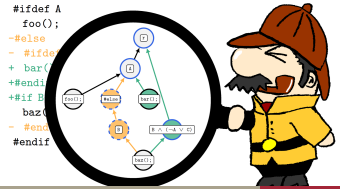
DiffDetective
A Framework for Variability-Aware Differencing



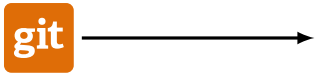
Parse all Patches to



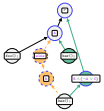
Your Empirical Analysis



DiffDetective
A Framework for Variability-Aware Differencing



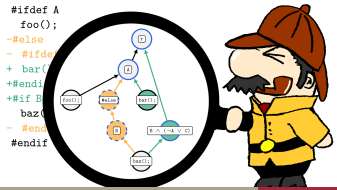
Parse all Patches to



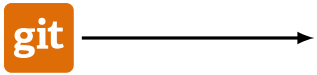
Your Empirical Analysis

Datasets

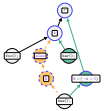
44 open-source system histories (incl. GODOT, Marlin, OPENVPN)



DiffDetective
A Framework for Variability-Aware Differencing



Parse all Patches to

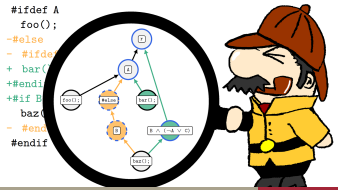


Your Empirical Analysis

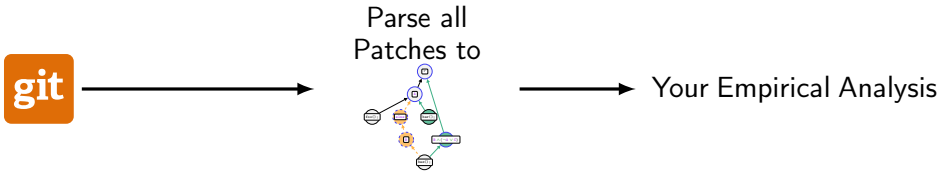
Datasets

44 open-source system histories (incl. GODOT, Marlin, OPENVPN)

1.7 million commits
45 million changed blocks



DiffDetective
A Framework for Variability-Aware Differencing



Datasets

44 open-source system histories (incl. GODOT, Marlin, OPENVPN)

1.7 million commits
45 million changed blocks

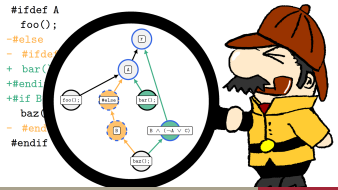
Features

integration of generic diff tools:
- Git diff (Myers/Histogram),
- GumTree [Falleri et al. 2014],
- truediff [Erdweg et al. 2021]

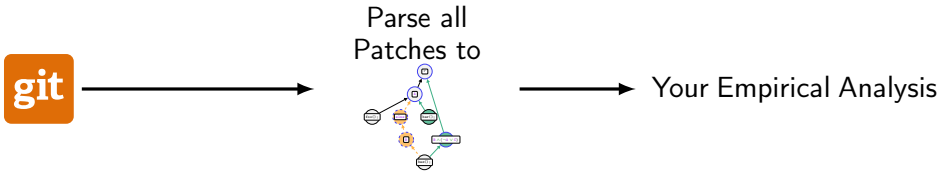
GUI for variability-aware diffs

transformations + analyses of diffs

demo + screencast



DiffDetective
A Framework for Variability-Aware Differencing



Datasets

44 open-source system histories (incl. GODOT, Marlin, OPENVPN)

1.7 million commits
45 million changed blocks

Features

integration of generic diff tools:
- Git diff (Myers/Histogram),
- GumTree [Falleri et al. 2014],
- truediff [Erdweg et al. 2021]

GUI for variability-aware diffs

transformations + analyses of diffs

demo + screencast

Usages

Bittner et al., ESEC/FSE'22,
Bittner et al., SPLC'23,
Greiner et al., SPLC'24,
Güthing et al., VaMoS'24,

+ 6 bachelor/master theses

Research Problem 3: Analyzing Edits to Variability

Research Problem 2: Edits to Static Variability

Research Problem 1: Static Variability

Foundations and meta-theory

Unified syntax and semantics of languages

Landscape of languages:
· choice calculus is a λ-calculus of variability
· multiple levels of expressiveness

Bittner et al., OOPSLA'24, 
Distinguished Artifact
Vatras – Agda Formalization

Research Problem 3: Analyzing Edits to Variability

Research Problem 2: Edits to Static Variability

Variability-aware differencing:
· Denotational semantics
· Language for diffs of annotative variability
· Make any generic differ variability-aware
· Differencing in milliseconds

Bittner et al., ESEC/FSE'22,
Bittner et al., SPLC'23,
Bittner et al., FSE Demonstrations'24,
Best Demonstrations Paper
DiffDetective – Empirical Evaluation Framework

Research Problem 1: Static Variability

Foundations and meta-theory

Unified syntax and semantics of languages

Landscape of languages:
· choice calculus is a λ-calculus of variability
· multiple levels of expressiveness

Bittner et al., OOPSLA'24,
Distinguished Artifact
Vatras – Agda Formalization

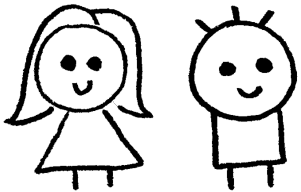
Classifying Edits to Variability in Source Code
Bittner, Tinnes, Schultheiß, Viegener, Kehrer, Thüm
ESEC/FSE'22,

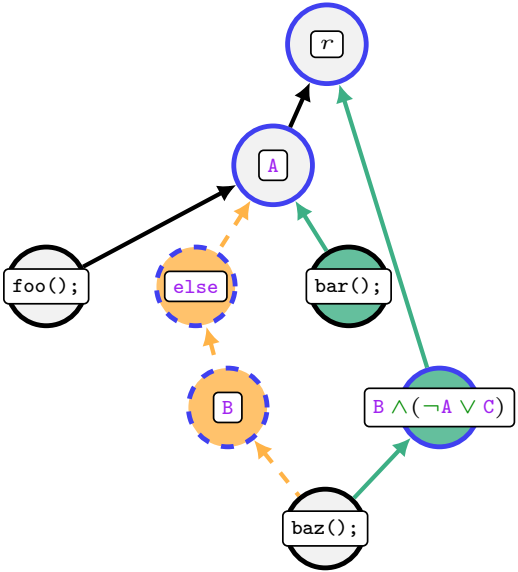
Views on Edits to Variational Software
Bittner, Schultheiß, Greiner, Moosherr, Krieter, Tinnes, Kehrer, Thüm
SPLC'23,

Feature Trace Recording
Bittner, Schultheiß, Thüm, Kehrer, Young, Linsbauer
ESEC/FSE'21, , Best Artifact

Research Problem 3
Analyzing Edits to Variability

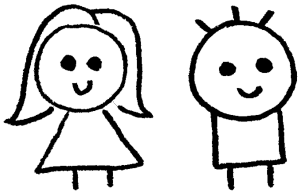
Ok, so now we have a variability-aware diff of our change.

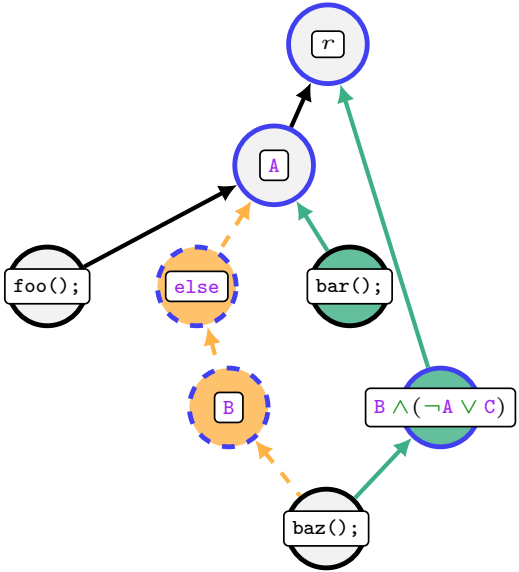




Ok, so now we have a variability-aware diff of our change.

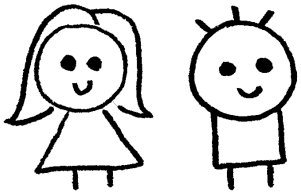
How does it help us though?





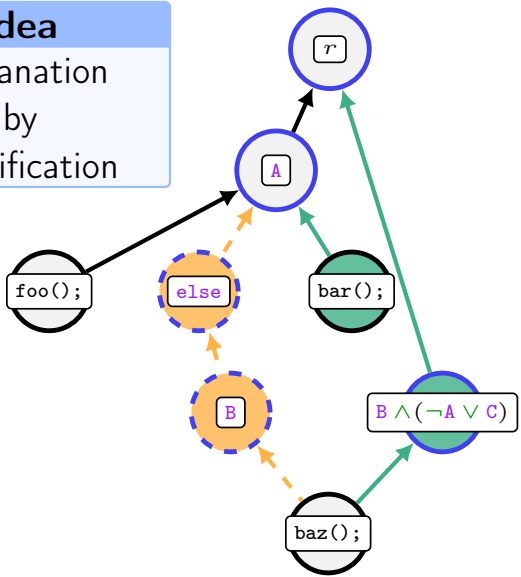
Ok, so now we have a variability-aware diff of our change.

How does it help us though?



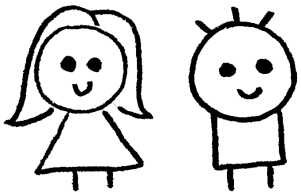
Idea

Explanation by Classification



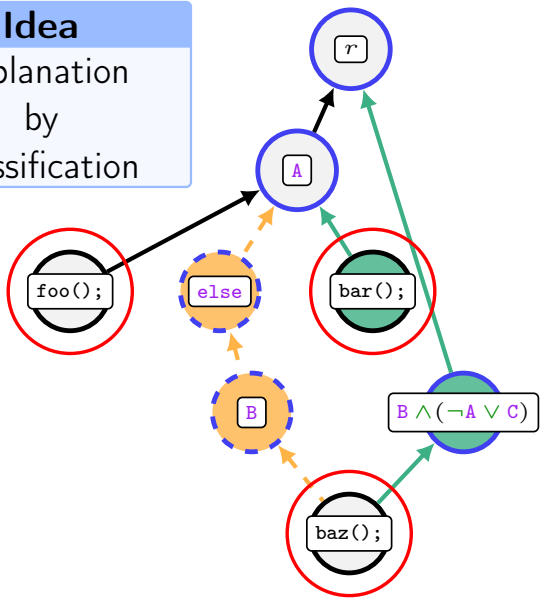
Ok, so now we have a variability-aware diff of our change.

How does it help us though?



Idea

Explanation by Classification



AddWithMapping(c) :=
added(c) ∧ added(M_a(c))

```
+ #if m
+ c
+ #endif
```

AddToPC(c) :=
added(c) ∧ ¬ added(M_a(c))

```
#if m
+ c
#endif
```

Specialization(c) := unchanged(c)
∧¬(PC$_b$(c) ⊨ PC$_a$(c))
∧ (PC$_a$(c) ⊨ PC$_b$(c))

```
+ #if m
  c
+ #endif
```

RemWithMapping(c) :=
removed(c) ∧ removed(M_b(c))

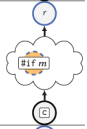
```
- #if m
- c
- #endif
```

RemFromPC(c) :=
removed(c) ∧ ¬ removed(M_b(c))

```
#if m
- c
#endif
```

Generalization(c) := unchanged(c)
∧ (PC$_b$(c) ⊨ PC$_a$(c))
∧¬(PC$_a$(c) ⊨ PC$_b$(c))

```
- #if m
  c
- #endif
```



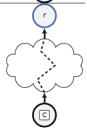
Reconfiguration(c) := unchanged(c)
∧¬(PC$_b$(c) ⊨ PC$_a$(c))
∧¬(PC$_a$(c) ⊨ PC$_b$(c))

```
- #if m
+ #if m'
  c
#endif
```

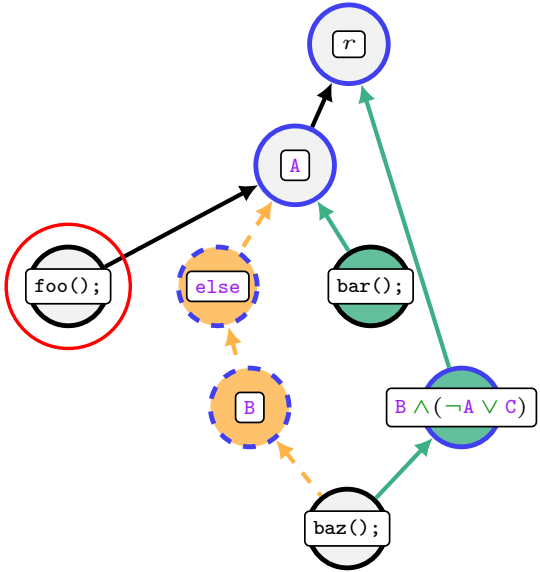
Refactoring(c) := unchanged(c)
∧(PC$_b$(c) ⊨ PC$_a$(c))
∧(PC$_a$(c) ⊨ PC$_b$(c))
∧($path_b$(c) ≠ $path_a$(c))

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

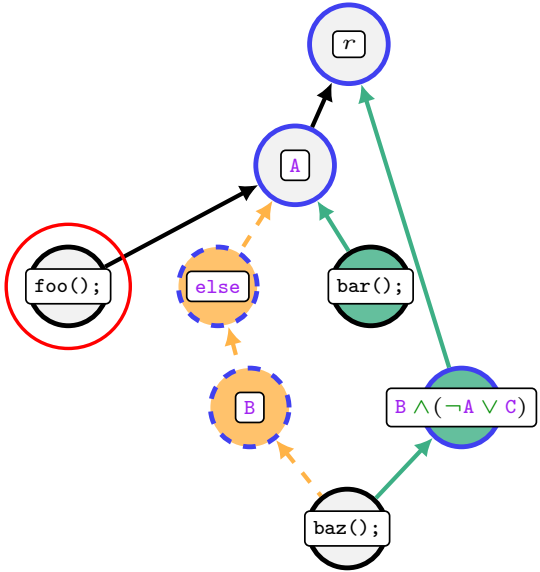
Untouched(c) := unchanged(c)
∧(PC$_b$(c) ⊨ PC$_a$(c))
∧(PC$_a$(c) ⊨ PC$_b$(c))
∧($path_b$(c) = $path_a$(c))



is unchanged

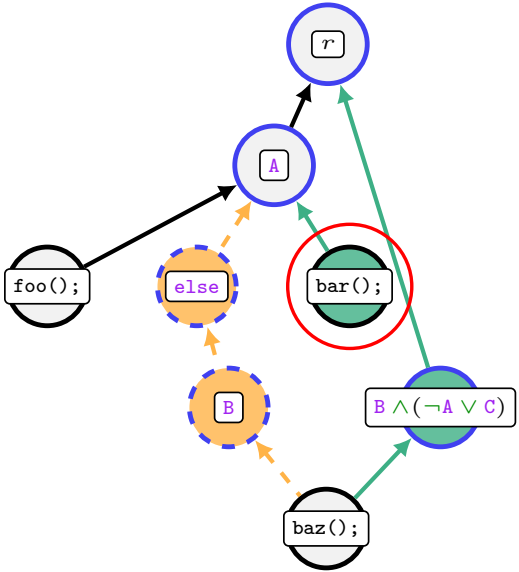


Untouched(foo();) = *true*



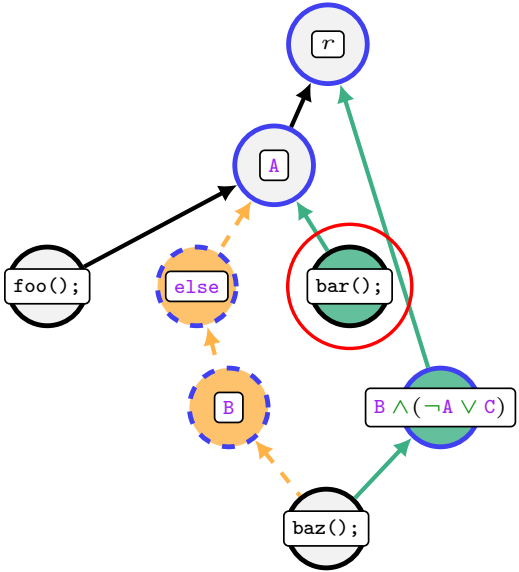
Untouched(`foo();`) = *true*

`bar();` is added to feature A



Untouched(foo();) = *true*

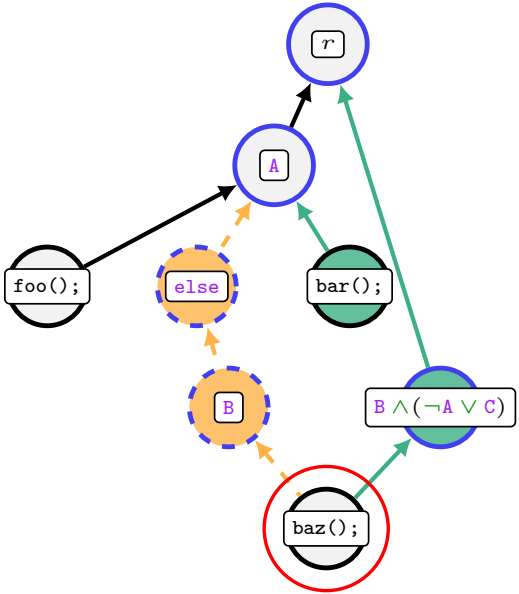
AddToPC(bar();) = *true*



Untouched(foo();) = *true*

AddToPC(bar();) = *true*

baz(); is moved
from B ∧ ¬A
to B ∧ (¬A ∨ C)

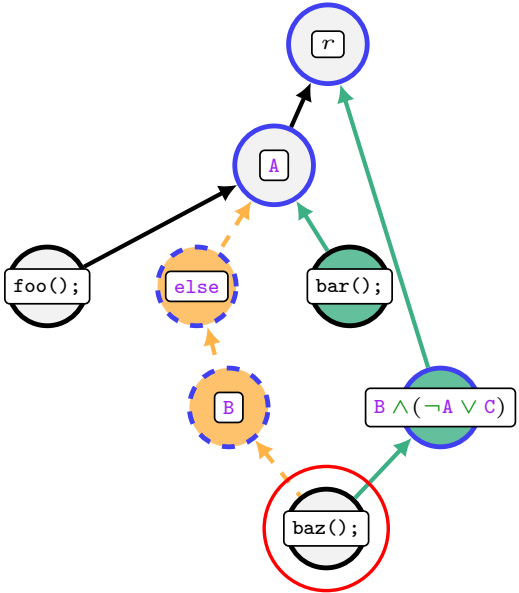


Untouched(foo();) = *true*

AddToPC(bar();) = *true*

Generalization(baz();) = *true*

because $(B \wedge \neg A) \models (B \wedge (\neg A \vee C))$



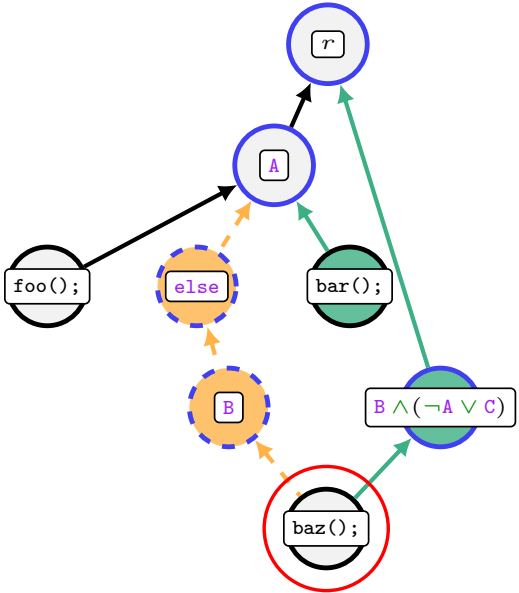
Untouched(foo();) = *true*

AddToPC(bar();) = *true*

Generalization(baz();) = *true*

because $(B \wedge \neg A) \models (B \wedge (\neg A \vee C))$

$\Rightarrow$ baz(); is used in more variants now



Untouched(`foo();`) = *true*

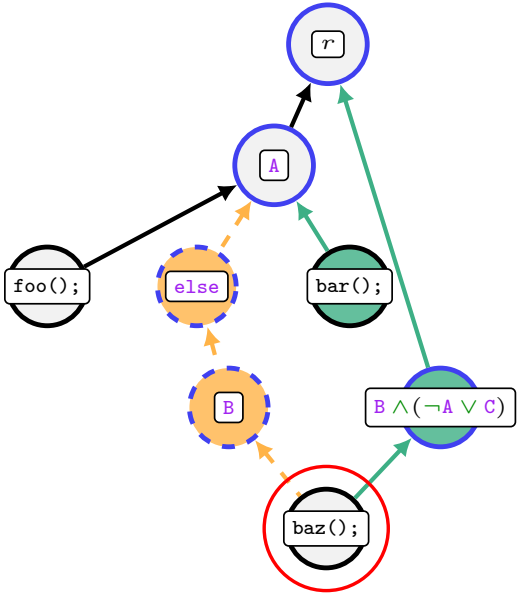
AddToPC(`bar();`) = *true*

Generalization(`baz();`) = *true*

because $(B \wedge \neg A) \models (B \wedge (\neg A \vee C))$

$\Rightarrow$ `baz();` is used in more variants now

$\Rightarrow$ `baz();` is reused



Untouched(foo();) = *true*

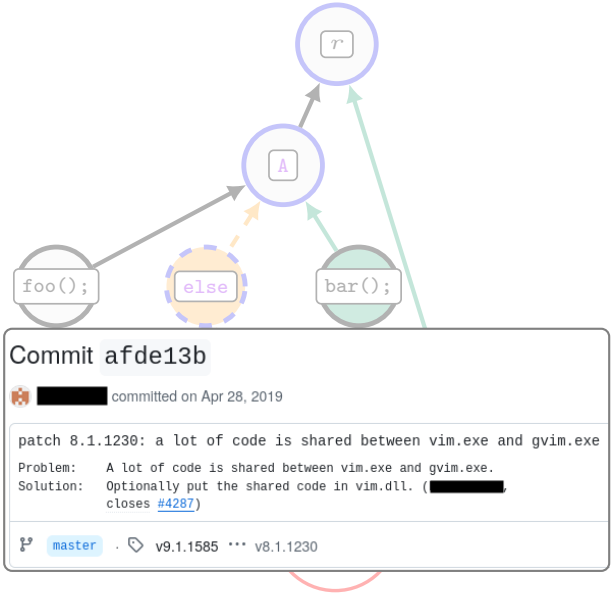
AddToPC(bar();) = *true*

Generalization(baz();) = *true*

because $(B \wedge \neg A) \models (B \wedge (\neg A \vee C))$

$\Rightarrow$ baz(); is used in more variants now

$\Rightarrow$ baz(); is reused



Untouched(foo();) = *true*

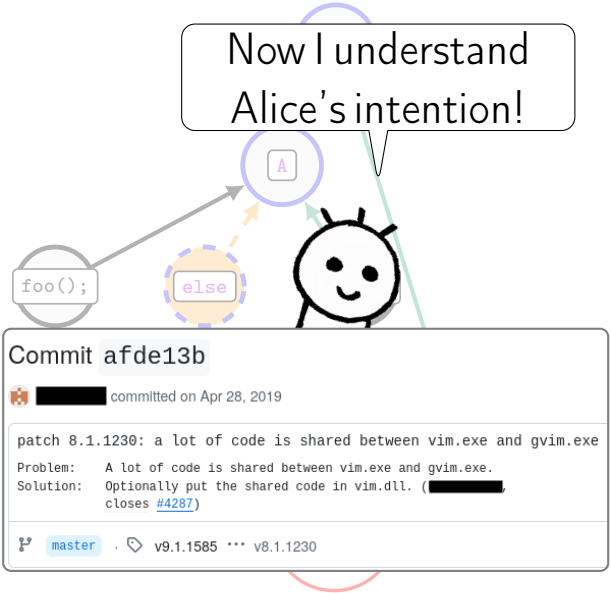
AddToPC(bar();) = *true*

Generalization(baz();) = *true*

because $(B \wedge \neg A) \models (B \wedge (\neg A \vee C))$

$\Rightarrow$ baz(); is used in more variants now

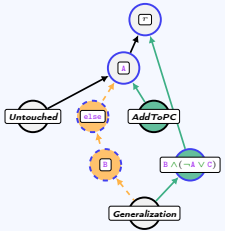
$\Rightarrow$ baz(); is reused



Reactive Analysis

Classifying Edits

Bittner et al., ESEC/FSE'22



How did the variability of this line of code change?

Does this code appear in more or less variants after the change?

Proactive Analysis

Slicing Edits

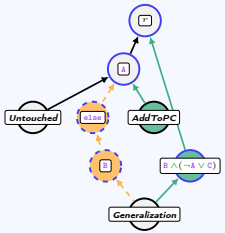
Interactive Analysis

Recording Edits

Reactive Analysis

Classifying Edits

Bittner et al., ESEC/FSE'22



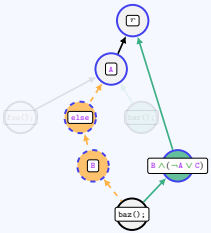
How did the variability of this line of code change?

Does this code appear in more or less variants after the change?

Proactive Analysis

Slicing Edits

Bittner et al., SPLC'23



What is the effect to

- *a particular feature?*
- *a particular variant?*
- *a subset of variants?*

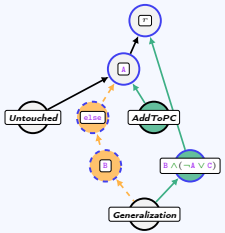
Interactive Analysis

Recording Edits

Reactive Analysis

Classifying Edits

Bittner et al., ESEC/FSE'22



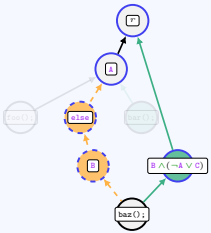
How did the variability of this line of code change?

Does this code appear in more or less variants after the change?

Proactive Analysis

Slicing Edits

Bittner et al., SPLC'23



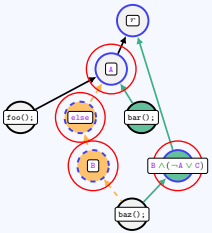
What is the effect to

- *a particular feature?*
- *a particular variant?*
- *a subset of variants?*

Interactive Analysis

Recording Edits

Bittner et al., ESEC/FSE'21



When editing a particular feature, how should the code be annotated afterwards?

Research Problem 3: Analyzing Edits to Variability

Research Problem 2: Edits to Static Variability

Variability-aware differencing:
· Denotational semantics
· Language for diffs of annotative variability
· Make any generic differ variability-aware
· Differencing in milliseconds

Bittner et al., ESEC/FSE'22,
Bittner et al., SPLC'23,
Bittner et al., FSE Demonstrations'24,
Best Demonstrations Paper
DiffDetective – Empirical Evaluation Framework

Research Problem 1: Static Variability

Foundations and meta-theory

Unified syntax and semantics of languages

Landscape of languages:
· choice calculus is a λ-calculus of variability
· multiple levels of expressiveness

Bittner et al., OOPSLA'24,
Distinguished Artifact
Vatras – Agda Formalization

Research Problem 3: Analyzing Edits to Variability

Change Impact Analyses:
· Classifying Edits
· Slicing Edits
· Recording Edits

Results in milliseconds

Bittner et al., ESEC/FSE'21,
Bittner et al., ESEC/FSE'22,
Bittner et al., SPLC'23,
Best Artifact
DiffDetective
Feature Trace Recording in Haskell

Research Problem 2: Edits to Static Variability

Variability-aware differencing:
· Denotational semantics
· Language for diffs of annotative variability
· Make any generic differ variability-aware
· Differencing in milliseconds

Bittner et al., ESEC/FSE'22,
Bittner et al., SPLC'23,
Bittner et al., FSE Demonstrations'24,
Best Demonstrations Paper
DiffDetective – Empirical Evaluation Framework

Research Problem 1: Static Variability

Foundations and meta-theory

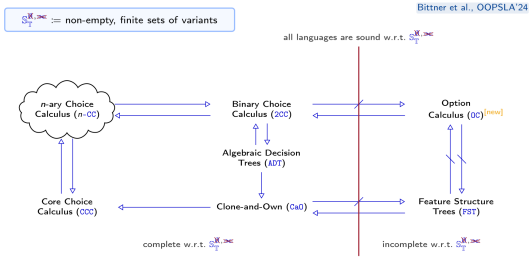
Unified syntax and semantics of languages

Landscape of languages:
· choice calculus is a λ-calculus of variability
· multiple levels of expressiveness

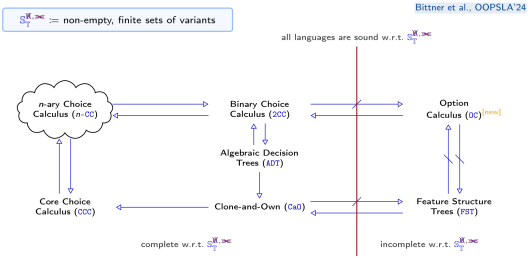
Bittner et al., OOPSLA'24,
Distinguished Artifact
Vatras – Agda Formalization

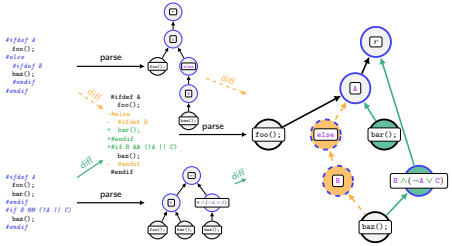

Can you *explain* our change?
Yes, but we cannot explain all your changes.
[Berba et al. 2012; Ji et al. 2015; Neves et al. 2015; Schulze et al. 2013; Schulze et al. 2012; Seidl et al. 2012; Stănciulescu et al. 2016]
Yes, but only if you would have asked before doing a change!
[Seidl et al. 2012]
Yes, but you have to apply our classification by hand.
[Berba et al. 2012; Ji et al. 2015; Passos et al. 2016]
Yes, but only if you give us your entire version history.
[Dintzner et al. 2018; Fischer et al. 2003; Kröher et al. 2023]

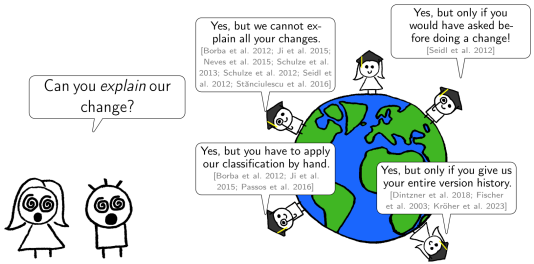

Can you *explain* our change?
Yes, but we cannot explain all your changes.
[Berba et al. 2012; Ji et al. 2015; Neves et al. 2015; Schulze et al. 2013; Schulze et al. 2012; Seidl et al. 2012; Stănciulescu et al. 2016]
Yes, but only if you would have asked before doing a change!
[Seidl et al. 2012]
Yes, but you have to apply our classification by hand.
[Berba et al. 2012; Ji et al. 2015; Passos et al. 2016]
Yes, but only if you give us your entire version history.
[Dintzner et al. 2018; Fischer et al. 2003; Kröher et al. 2023]

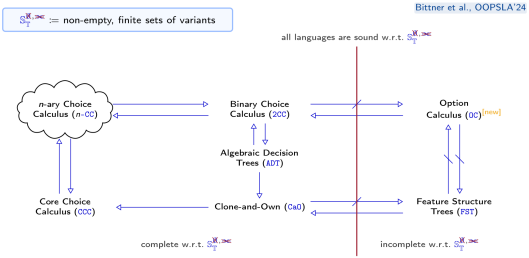

Bittner et al., OOPSLA'24
$\mathbb{S}^{\mathbb{M},\mathbb{INC}}_{\mathbb{V}}$:= non-empty, finite sets of variants
all languages are sound w.r.t. $\mathbb{S}^{\mathbb{M},\mathbb{INC}}_{\mathbb{V}}$
n-ary Choice Calculus (*n*-CC)
Binary Choice Calculus (2CC)
Option Calculus (OC)[new]
Algebraic Decision Trees (ADT)
Core Choice Calculus (CCC)
Clone-and-Own (CaO)
Feature Structure Trees (FST)
complete w.r.t. $\mathbb{S}^{\mathbb{M},\mathbb{INC}}_{\mathbb{V}}$
incomplete w.r.t. $\mathbb{S}^{\mathbb{M},\mathbb{INC}}_{\mathbb{V}}$

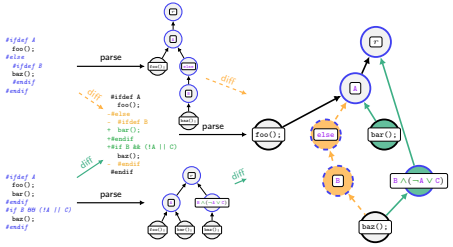

Can you *explain* our change?
Yes, but we cannot explain all your changes.
[Berba et al. 2012; Ji et al. 2015; Neves et al. 2015; Schulze et al. 2013; Schulze et al. 2012; Seidl et al. 2012; Stănciulescu et al. 2016]
Yes, but only if you would have asked before doing a change!
[Seidl et al. 2012]
Yes, but you have to apply our classification by hand.
[Berba et al. 2012; Ji et al. 2015; Passos et al. 2016]
Yes, but only if you give us your entire version history.
[Dintzner et al. 2018; Fischer et al. 2003; Kröher et al. 2023]


Bittner et al., OOPSLA'24
$\mathbb{S}^{\mathbb{M},\mathbb{INC}}_{\mathbb{Z}}$:= non-empty, finite sets of variants
all languages are sound w.r.t. $\mathbb{S}^{\mathbb{M},\mathbb{INC}}_{\mathbb{Z}}$
n-ary Choice Calculus (nCC)
Binary Choice Calculus (2CC)
Option Calculus (OC)[new]
Algebraic Decision Trees (ADT)
Core Choice Calculus (CCC)
Clone-and-Own (CaO)
Feature Structure Trees (FST)
complete w.r.t. $\mathbb{S}^{\mathbb{M},\mathbb{INC}}_{\mathbb{Z}}$
incomplete w.r.t. $\mathbb{S}^{\mathbb{M},\mathbb{INC}}_{\mathbb{Z}}$


parse
diff
foo();
else
bar();
A
B
B ∧(¬A ∨ C)
baz();







Reactive Analysis
Classifying Edits
Bittner et al., ESEC/FSE'22

How did the variability of this line of code change?

Does this code appear in more or less variants after the change?

Proactive Analysis
Slicing Edits
Bittner et al., SPLC'23

What is the effect to
- *a particular feature?*
- *a particular variant?*
- *a subset of variants?*

Interactive Analysis
Recording Edits
Bittner et al., ESEC/FSE'21

When editing a particular feature, how should the code be annotated afterwards?

Publications

Primary Publications I

On the Expressive Power of Languages for Static Variability
Paul Maximilian Bittner and Alexander Schultheiß and Benjamin Moosherr and Jeffrey M. Young and Leopoldo Teixeira and Eric Walkingshaw and Parisa Ataei and Thomas Thüm,
Proceedings of the ACM on Programming Languages (PACMPL), Object-Oriented Programming, Systems, Languages & Applications (OOPSLA), October 2024, ACM, New York, NY, USA
DOI: 10.1145/3689747,  , Distinguished Artifact

Variability-Aware Differencing with DiffDetective
Paul Maximilian Bittner and Alexander Schultheiß and Benjamin Moosherr and Timo Kehrer and Thomas Thüm,
Companion Proc. Int'l Conference on the Foundations of Software Engineering (FSE Companion), pages 632–636, Porto de Galinhas, Brazil, July 2024, ACM, New York, NY, USA
DOI: 10.1145/3663529.3663813,  , Best Demonstrations Paper

Views on Edits to Variational Software
Paul Maximilian Bittner and Alexander Schultheiß and Sandra Greiner and Benjamin Moosherr and Sebastian Krieter and Christof Tinnes and Timo Kehrer and Thomas Thüm,
Proc. Int'l Systems and Software Product Line Conf. (SPLC), pages 141–152, Tokyo, Japan, August 2023, ACM, New York, NY, USA
DOI: 10.1145/3579027.3608985, 

Primary Publications II

Classifying Edits to Variability in Source Code
Paul Maximilian Bittner and Christof Tinnes and Alexander Schultheiß and Sören Viegener and Timo Kehrer and Thomas Thüm,
Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE), pages 196–208, Singapore, November 2022, ACM, New York, NY, USA
DOI: 10.1145/3540250.3549108, 

Feature Trace Recording
Paul Maximilian Bittner and Alexander Schultheiß and Thomas Thüm and Timo Kehrer and Jeffrey M. Young and Lukas Linsbauer,
Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE), pages 1007–1020, Athens, Greece, August 2021, ACM, New York, NY, USA
DOI: 10.1145/3468264.3468531,  , Best Artifact

Other Publications Partly Contributing to This Thesis I

Decades of GNU Patch and Git Cherry-Pick: Can We Do Better?
Alexander Schultheiß and Alexander Boll and Paul Maximilian Bittner and Sandra Greiner and Thomas Thüm and Timo Kehrer,
Proc. Int'l Conf. on Software Engineering (ICSE), Rio de Janeiro, Brazil, 2026

Give an Inch and Take a Mile? Effects of Adding Reliable Knowledge to Heuristic Feature Tracing
Sandra Greiner and Alexander Schultheiß and Paul Maximilian Bittner and Thomas Thüm and Timo Kehrer,
Proc. Int'l Systems and Software Product Line Conf. (SPLC), pages 84–95, Dommeldange, Luxembourg, September 2024, ACM, New York, NY, USA
DOI: 10.1145/3646548.3672593, Best Research Paper

RaQuN: A Generic and Scalable N-Way Model Matching Algorithm
Alexander Schultheiß and Paul Maximilian Bittner and Alexander Boll and Lars Grunske and Thomas Thüm and Timo Kehrer,
Software and Systems Modeling (SoSyM), pages 1495–1517, October 2023, Springer
DOI: 10.1007/s10270-022-01062-5

Scalable N-Way Model Matching Using Multi-Dimensional Search Trees
Alexander Schultheiß and Paul Maximilian Bittner and Lars Grunske and Thomas Thüm and Timo Kehrer,
Proc. Int'l Conf. on Model Driven Engineering Languages and Systems (MODELS), pages 1–12, Virtual Event, Fukuoka, Japan, October 2021, IEEE, Washington, DC, USA
DOI: 10.1109/MODELS50736.2021.00010,

Other Publications Partly Contributing to This Thesis II

Explaining Edits to Variability Annotations in Evolving Software Product Lines
Lukas Güthing and Paul Maximilian Bittner and Ina Schaefer and Thomas Thüm,
Proc. Int'l Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS), pages 93–102, Bern, Switzerland, February 2024, ACM, New York, NY, USA
DOI: 10.1145/3634713.3634725,

Benchmark Generation With VEVOS: A Coverage Analysis of Evolution Scenarios in Variant-Rich Systems
Alexander Schultheiß and Paul Maximilian Bittner and Sandra Greiner and Timo Kehrer,
Proc. Int'l Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS), pages 13–22, Odense, Denmark, January 2023, ACM, New York, NY, USA
DOI: 10.1145/3571788.3571793

Evaluating State-of-the-Art #SAT Solvers on Industrial Configuration Spaces
Chico Sundermann and Tobias Heß and Michael Nieke and Paul Maximilian Bittner and Jeffrey M. Young and Thomas Thüm and Ina Schaefer,
Empirical Software Engineering (EMSE), 29, pages 38, January 2023, Springer
DOI: 10.1007/s10664-022-10265-9

Simulating the Evolution of Clone-and-Own Projects With VEVOS
Alexander Schultheiß and Paul Maximilian Bittner and Sascha El-Sharkawy and Thomas Thüm and Timo Kehrer,
Proc. Int'l Conf. on Evaluation Assessment in Software Engineering (EASE), pages 231–236, Gothenburg, Sweden, June 2022, ACM, New York, NY, USA
DOI: 10.1145/3530019.3534084

Other Publications Partly Contributing to This Thesis III

Variational Satisfiability Solving: Efficiently Solving Lots of Related SAT Problems
Jeffrey M. Young and Paul Maximilian Bittner and Eric Walkingshaw and Thomas Thüm,
Empirical Software Engineering (EMSE), pages 53, November 2022, Springer
DOI: 10.1007/s10664-022-10217-3

Quantifying the Potential to Automate the Synchronization of Variants in Clone-and-Own
Alexander Schultheiß and Paul Maximilian Bittner and Thomas Thüm and Timo Kehrer,
Proc. Int'l Conf. on Software Maintenance and Evolution (ICSME), pages 269–280, Limassol, Cyprus, October 2022, IEEE, Piscataway, NJ, USA
DOI: 10.1109/ICSME55016.2022.00032,

Derivation of Subset Product Lines in FeatureIDE
Lukas Linsbauer and Paul Westphal and Paul Maximilian Bittner and Sebastian Krieter and Thomas Thüm and Ina Schaefer,
Proc. Int'l Systems and Software Product Line Conf. (SPLC), pages 38–41, Graz, Austria, September 2022, ACM, New York, NY, USA
DOI: 10.1145/3503229.3547033, Best Demo/Tools Paper

Bridging the Gap Between Clone-and-Own and Software Product Lines
Timo Kehrer and Thomas Thüm and Alexander Schultheiß and Paul Maximilian Bittner,
Proc. Int'l Conf. on Software Engineering (ICSE), pages 21–25, May 2021, IEEE, Piscataway, NJ, USA
DOI: 10.1109/ICSE-NIER52604.2021.00013

Other Publications Partly Contributing to This Thesis IV

Applications of #SAT Solvers on Feature Models
Chico Sundermann and Michael Nieke and Paul Maximilian Bittner and Tobias Heß and Thomas Thüm and Ina Schaefer,
Proc. Int'l Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS), Krems, Austria, February 2021, ACM, New York, NY, USA
DOI: 10.1145/3442391.3442404

On the Use of Product-Line Variants as Experimental Subjects for Clone-and-Own Research: A Case Study
Alexander Schultheiß and Paul Maximilian Bittner and Timo Kehrer and Thomas Thüm,
Proc. Int'l Systems and Software Product Line Conf. (SPLC), Montreal, QC, Canada, October 2020, ACM, New York, NY, USA
DOI: 10.1145/3382025.3414972,

SAT Encodings of the At-Most-k Constraint – A Case Study on Configuring University Courses
Paul Maximilian Bittner and Thomas Thüm and Ina Schaefer,
Proc. Int'l Conf. on Software Engineering and Formal Methods (SEFM), pages 127–144, Oslo, Norway, September 2019, Springer, Berlin, Heidelberg
DOI: 10.1007/978-3-030-30446-1_7

Other Publications Partly Contributing to This Thesis V

Extended Abstracts:

Evaluating State-of-the-Art #SAT Solvers on Industrial Configuration Spaces
Chico Sundermann and Tobias Heß and Michael Nieke and Paul Maximilian Bittner and Jeffrey M. Young and Thomas Thüm and Ina Schaefer,
Proc. Software Engineering (SE), pages 67–68, February 2024, Gesellschaft für Informatik, Bonn, Germany
DOI: 10.18420/sw2024_18

Classifying Edits to Variability in Source Code – Summary
Paul Maximilian Bittner and Christof Tinnes and Alexander Schultheiß and Sören Viegener and Timo Kehrer and Thomas Thüm,
Proc. Software Engineering (SE), pages 39–40, Paderborn, Germany, February 2023, Gesellschaft für Informatik, Bonn, Germany

Quantifying the Potential to Automate the Synchronization of Variants in Clone-and-Own – Summary
Alexander Schultheiß and Paul Maximilian Bittner and Thomas Thüm and Timo Kehrer,
Proc. Software Engineering (SE), pages 109–110, Paderborn, Germany, February 2023, Gesellschaft für Informatik, Bonn, Germany

Variational Satisfiability Solving: Efficiently Solving Lots of Related SAT Problems – Summary
Jeffrey M. Young and Paul Maximilian Bittner and Eric Walkingshaw and Thomas Thüm,
Proc. Software Engineering (SE), pages 129–130, Paderborn, Germany, February 2023, Gesellschaft für Informatik, Bonn, Germany

Other Publications Partly Contributing to This Thesis VI

Feature Trace Recording – Summary
Paul Maximilian Bittner and Alexander Schultheiß and Thomas Thüm and Timo Kehrer and Jeffrey M. Young and Lukas Linsbauer,
Proc. Software Engineering (SE), pages 19–20, Berlin, Germany, February 2022, Gesellschaft für Informatik, Bonn, Germany
DOI: 10.18420/se2022-ws-002

Scalable N-Way Model Matching Using Multi-Dimensional Search Trees – Summary
Alexander Schultheiß and Paul Maximilian Bittner and Thomas Thüm and Timo Kehrer,
Proc. Software Engineering (SE), pages 83–84, Berlin, Germany, February 2022, Gesellschaft für Informatik, Bonn, Germany
DOI: 10.18420/se2022-ws-028

Moreover, the contributions of our ESEC/FSE'21 publication [Bittner et al. 2021], which is one of the main publications of this thesis, are partially based on the results of my master's thesis:

Semi-Automated Inference of Feature Traceability During Software Development
Paul Maximilian Bittner, Master's Thesis, TU Braunschweig, February 2020
DOI: 10.24355/dbbs.084-202002271120-0

Supervised Theses Relevant to This Thesis I

Variability-Aware Patching of Software Product-Line Variants
Pia Meier, Master's Thesis, TU Braunschweig, 2025
In progress

On the Succinctness of Languages for Static Variability
Benjamin Moosherr, Master's Thesis, University of Ulm, 2025
In progress

Understanding Variant Drift via Cherry-Picking
Manfred Küppers, Bachelor's Thesis, TU Braunschweig, 2025
In progress

Unparsing von Datenstrukturen zur Analyse von C-Präprozessor-Variabilität
Eugen Shulimov, Bachelor's Thesis, University of Paderborn, January 2025
DOI: 10.17619/UNIPB/1-2306

Constructing Variation Diffs Using Tree Diffing Algorithms
Benjamin Moosherr, Bachelor's Thesis, University of Ulm, April 2023
DOI: 10.18725/OPARU-50108

Inspecting the Evolution of Feature Annotations in Configurable Software
Lukas Güthing, Master's Thesis, University of Ulm, January 2023

Supervised Theses Relevant to This Thesis II

Reverse Engineering Feature-Aware Commits From Software Product-Line Repositories
Lukas Bormann, Bachelor's Thesis, University of Ulm, October 2022
DOI: 10.18725/OPARU-47892

Type-Checking Variability in Clone-and-Own Variants With Product-Line Tooling
Kevin Jedelhauser, Master's Thesis, University of Ulm, September 2022

Empirical Evaluation of Feature Trace Recording on the Edit History of Marlin
Sören Viegener, Bachelor's Thesis, University of Ulm, April 2021
DOI: 10.18725/OPARU-38603

Additional Publications I

Temporal Consistent Motion Parallax for Omnidirectional Stereo Panorama Video
Moritz Mühlhausen and Moritz Kappel and Marc Kassubeck and Paul Maximilian Bittner and Susana Castillo and Marcus Magnor,
Proc. ACM Symposium on Virtual Reality Software and Technology (VRST), November 2020, ACM, New York, NY, USA
DOI: 10.1145/3385956.3418965

Depth Augmented Omnidirectional Stereo for 6-DoF VR Photography
Tobias Bertel and Moritz Mühlhausen and Moritz Kappel and Paul Maximilian Bittner and Christian Richardt and Marcus Magnor,
Proc. IEEE Virtual Reality Workshop (VR), pages 660–661, May 2020, IEEE, Piscataway, NJ, USA
DOI: 10.1109/VRW50115.2020.00181

Gaze and Motion-Aware Real-Time Dome Projection System
Steve Grogorick and Matthias Überheide and Jan-Philipp Tauscher and Paul Maximilian Bittner and Marcus Magnor,
Proc. IEEE Virtual Reality Workshop (VR), pages 1780–1783, Osaka, Japan, March 2019, IEEE, Piscataway, NJ, USA
DOI: 10.1109/VR.2019.8797902

Immersive EEG: Evaluating Electroencephalography in Virtual Reality
Jan-Philipp Tauscher and Fabian Wolf Schottky and Steve Grogorick and Paul Maximilian Bittner and Maryam Mustafa and Marcus Magnor,
Proc. IEEE Virtual Reality Workshop (VR), pages 1794–1800, Osaka, Japan, March 2019, IEEE, Piscataway, NJ, USA
DOI: 10.1109/VR.2019.8797858

Additional Publications II

Evaluation of Optimised Centres of Rotation Skinning
Paul Maximilian Bittner and Jan-Philipp Tauscher and Steve Grogorick and Marcus Magnor, *Poster at International Conference on Computational Visual Media*, April 2019
URL: https://graphics.tu-bs.de/publications/bittner2019evaluation

Research Artifacts and Tools I

Vatras – Agda library of verified compilers for variability languages.

Related Publications	*Related Theses*
Bittner et al., OOPSLA'24,	Moosherr, Master's Thesis'25

DiffDetective – Java library and framework for variability-aware differencing and empirical evaluations on Git histories.

Related Publications	*Related Theses*
Bittner et al., FSE Demonstrations'24, Bittner et al., SPLC'23, Bittner et al., ESEC/FSE'22, Greiner et al., SPLC'24, Güthing et al., VaMoS'24,	Meier, Master's Thesis'25 Shulimov, Bachelor's Thesis'25 Moosherr, Bachelor's Thesis'23 Güthing, Master's Thesis'23 Bormann, Bachelor's Thesis'22 Viegener, Bachelor's Thesis'21

Research Artifacts and Tools II

Feature Trace Recording – Haskell Formalization

Related Publications	*Related Theses*
Bittner et al., ESEC/FSE'21,	Bittner, Master's Thesis'20

Bibliography

Meier, Pia (2025). "Variability-Aware Patching of Software Product-Line Variants". Supervised by Paul Maximilian Bittner, Rahel Sundermann, Thomas Thüm, and Timo Kehrer. Master's Thesis. Germany: TU Braunschweig (cit. on p. 139).

Moosherr, Benjamin (2025). "On the Succinctness of Languages for Static Variability". Supervised by Paul Maximilian Bittner, Matthias Tichy, and Thomas Thüm. Master's Thesis. Germany: University of Ulm (cit. on p. 139).

Shulimov, Eugen (2025). "Unparsing von Datenstrukturen zur Analyse von C-Präprozessor-Variabilität". Bachelor's Thesis. University of Paderborn. doi: 10.17619/UNIPB/1-2306 (cit. on p. 139).

Bittner, Paul Maximilian, Alexander Schultheiß, Benjamin Moosherr, Timo Kehrer, and Thomas Thüm (2024a). "Variability-Aware Differencing with DiffDetective". In: *Companion Proc. Int'l Conference on the Foundations of Software Engineering (FSE Companion)*. ACM, pp. 632–636. doi: 10.1145/3663529.3663813 (cit. on pp. 82–98, 100, 120, 121, 139).

Bittner, Paul Maximilian, Alexander Schultheiß, Benjamin Moosherr, Jeffrey M. Young, Leopoldo Teixeira, Eric Walkingshaw, Parisa Ataei, and Thomas Thüm (2024b). "On the Expressive Power of Languages for Static Variability". In: *Proceedings of the ACM on Programming Languages (PACMPL)* 8.Object-Oriented Programming, Systems, Languages & Applications (OOPSLA). doi: 10.1145/3689747 (cit. on pp. 49–70, 72, 99, 100, 120, 121, 139).

Greiner, Sandra, Alexander Schultheiß, Paul Maximilian Bittner, Thomas Thüm, and Timo Kehrer (2024). "Give an Inch and Take a Mile? Effects of Adding Reliable Knowledge to Heuristic Feature Tracing". In: *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, pp. 84–95. doi: 10.1145/3646548.3672593 (cit. on pp. 91–98, 139).

Güthing, Lukas, Paul Maximilian Bittner, Ina Schaefer, and Thomas Thüm (2024). "Explaining Edits to Variability Annotations in Evolving Software Product Lines". In: *Proc. Int'l Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM, pp. 93–102. doi: 10.1145/3634713.3634725 (cit. on pp. 91–98, 139).

Bittner, Paul Maximilian, Alexander Schultheiß, Sandra Greiner, Benjamin Moosherr, Sebastian Krieter, Christof Tinnes, Timo Kehrer, and Thomas Thüm (2023). "Views on Edits to Variational Software". In: *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, pp. 141–152. doi: 10.1145/3579027.3608985 (cit. on pp. 91–98, 100, 117–121, 139).

Güthing, Lukas (2023). "Inspecting the Evolution of Feature Annotations in Configurable Software". Master's Thesis. University of Ulm (cit. on p. 139).

Kröher, Christian, Lea Gerling, and Klaus Schmid (2023). "Comparing the Intensity of Variability Changes in Software Product Line Evolution". In: *J. Systems and Software (JSS)* 203, p. 111737. doi: 10.1016/j.jss.2023.111737 (cit. on pp. 17–34, 76).

Moosherr, Benjamin (2023). "Constructing Variation Diffs Using Tree Diffing Algorithms". Bachelor's Thesis. University of Ulm. doi: 10.18725/OPARU-50108 (cit. on p. 139).

Ananieva, Sofia, Sandra Greiner, Jacob Krueger, Lukas Linsbauer, Sten Gruener, Timo Kehrer, Thomas Kuehn, Christoph Seidl, and Ralf Reussner (2022). "Unified Operations for Variability in Space and Time". In: *Proc. Int'l Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM. doi: 10.1145/3510466.3510483 (cit. on pp. 17–34, 76).

Bittner, Paul Maximilian, Christof Tinnes, Alexander Schultheiß, Sören Viegener, Timo Kehrer, and Thomas Thüm (2022). "Classifying Edits to Variability in Source Code". In: *Proc. Europ. Software Engineering*

Conf./Foundations of Software Engineering (ESEC/FSE). ACM, pp. 196–208. doi: 10.1145/3540250.3549108 (cit. on pp. 82–98, 100, 102–121, 139).


Bormann, Lukas (2022). "Reverse Engineering Feature-Aware Commits From Software Product-Line Repositories". Bachelor's Thesis. University of Ulm. doi: 10.18725/OPARU-47892 (cit. on p. 139).


Bittner, Paul Maximilian, Alexander Schultheiß, Thomas Thüm, Timo Kehrer, Jeffrey M. Young, and Lukas Linsbauer (2021). "Feature Trace Recording". In: *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM, pp. 1007–1020. doi: 10.1145/3468264.3468531 (cit. on pp. 117–119, 121, 134, 140).


Castro, Thiago, Leopoldo Teixeira, Vander Alves, Sven Apel, Maxime Cordy, and Rohit Gheyi (2021). "A Formal Framework of Software Product Line Analyses". In: *Trans. on Software Engineering and Methodology (TOSEM)* 30.3. doi: 10.1145/3442389 (cit. on pp. 17–34, 40–49).


Erdweg, Sebastian, Tamás Szabó, and André Pacak (2021). "Concise, Type-Safe, and Efficient Structural Diffing". In: *Proc. ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI)*. ACM, pp. 406–419. doi: 10.1145/3453483.3454052 (cit. on pp. 17–34, 76, 91–98).


Viegener, Sören (2021). "Empirical Evaluation of Feature Trace Recording on the Edit History of Marlin". Bachelor's Thesis. University of Ulm. doi: 10.18725/OPARU-38603 (cit. on p. 139).


Bittner, Paul Maximilian (2020). "Semi-Automated Inference of Feature Traceability During Software Development". Master's Thesis. TU Braunschweig. doi: 10.24355/dbbs.084-202002271120-0 (cit. on p. 140).


Nugroho, Yusuf Sulistyo, Hideaki Hata, and Kenichi Matsumoto (2020). "How Different are Different Diff Algorithms in Git?" In: *Empirical Software Engineering (EMSE)* 25.1, pp. 790–823. doi: 10.1007/s10664-019-09772-z (cit. on pp. 17–34, 76).

Dintzner, Nicolas, Arie van Deursen, and Martin Pinzger (2018). "FEVER: An Approach to Analyze Feature-Oriented Changes and Artefact Co-Evolution in Highly Configurable Systems". In: *Empirical Software Engineering (EMSE)* 23.2, pp. 905–952. doi: 10.1007/s10664-017-9557-6 (cit. on pp. 17–34, 76).

Linsbauer, Lukas, Roberto Erick Lopez-Herrejon, and Alexander Egyed (2017). "Variability Extraction and Modeling for Product Variants". In: *Software and Systems Modeling (SoSyM)* 16.4, pp. 1179–1199. doi: 10.1007/s10270-015-0512-y (cit. on pp. 40–48).

Dotzler, Georg and Michael Philippsen (2016). "Move-Optimized Source Code Tree Differencing". In: *Proc. Int'l Conf. on Automated Software Engineering (ASE)*. ACM, pp. 660–671 (cit. on pp. 17–34, 76).

Al-Hajjaji, Mustafa, Fabian Benduhn, Thomas Thüm, Thomas Leich, and Gunter Saake (2016). "Mutation Operators for Preprocessor-Based Variability". In: *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM, pp. 81–88. doi: 10.1145/2866614.2866626 (cit. on pp. 17–34, 76).

Passos, Leonardo, Leopoldo Teixeira, Nicolas Dintzner, Sven Apel, Andrzej Wąsowski, Krzysztof Czarnecki, Paulo Borba, and Jianmei Guo (2016). "Coevolution of Variability Models and Related Software Artifacts". In: *Empirical Software Engineering (EMSE)* 21.4. doi: 10.1007/s10664-015-9364-x (cit. on pp. 17–34).

Stănciulescu, Stefan, Thorsten Berger, Eric Walkingshaw, and Andrzej Wąsowski (2016). "Concepts, Operations, and Feasibility of a Projection-Based Variation Control System". In: *Proc. Int'l Conf. on Software Maintenance and Evolution (ICSME)*. IEEE, pp. 323–333. doi: 10.1109/ICSME.2016.88 (cit. on pp. 17–34, 76).

Ji, Wenbin, Thorsten Berger, Michal Antkiewicz, and Krzysztof Czarnecki (2015). "Maintaining Feature Traceability With Embedded Annotations". In: *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, pp. 61–70. doi: 10.1145/2791060.2791107 (cit. on pp. 17–34).

Neves, Laís, Paulo Borba, Vander Alves, Lucinéia Turnes, Leopoldo Teixeira, Demóstenes Sena, and Uirá Kulesza (2015). "Safe Evolution Templates for Software Product Lines". In: *J. Systems and Software (JSS)* 106, pp. 42–58. doi: 10.1016/j.jss.2015.04.024 (cit. on pp. 17–34, 76).

Falleri, Jean-Rémy, Floréal Morandat, Xavier Blanc, Matias Martinez, and Martin Monperrus (2014). "Fine-Grained and Accurate Source Code Differencing". In: *Proc. Int'l Conf. on Automated Software Engineering (ASE)*, pp. 313–324. doi: 10.1145/2642937.2642982 (cit. on pp. 17–34, 76, 91–98).

Apel, Sven, Christian Kästner, and Christian Lengauer (2013). "Language-Independent and Automated Software Composition: The FeatureHouse Experience". In: *IEEE Trans. on Software Engineering (TSE)* 39.1, pp. 63–79. doi: 10.1109/TSE.2011.120 (cit. on pp. 17–34, 40–49).

Asaduzzaman, Muhammad, Chanchal K. Roy, Kevin A. Schneider, and Massimiliano Di Penta (2013). "LHDiff: A Language-Independent Hybrid Approach for Tracking Source Code Lines". In: *Proc. Int'l Conf. on Software Maintenance (ICSM)*. IEEE, pp. 230–239. doi: 10.1109/ICSM.2013.34 (cit. on pp. 17–34, 76).

Schulze, Sandro, Oliver Richers, and Ina Schaefer (2013). "Refactoring Delta-Oriented Software Product Lines". In: *Proc. Int'l Conf. on Aspect-Oriented Software Development (AOSD)*. ACM, pp. 73–84. doi: 10.1145/2451436.2451446 (cit. on pp. 17–34, 76).

Walkingshaw, Eric (2013). "The Choice Calculus: A Formal Language of Variation". PhD thesis. Oregon State University (cit. on pp. 17–34, 40–49).

Borba, Paulo, Leopoldo Teixeira, and Rohit Gheyi (2012). "A Theory of Software Product Line Refinement". In: *Theoretical Computer Science* 455.0, pp. 2–30. doi: 10.1016/j.tcs.2012.01.031 (cit. on pp. 17–34, 76).

Schulze, Sandro, Thomas Thüm, Martin Kuhlemann, and Gunter Saake (2012). "Variant-Preserving Refactoring in Feature-Oriented Software Product Lines". In: *Proc. Int'l Workshop on Variability Modelling of*

Software-Intensive Systems (VaMoS). ACM, pp. 73–81. doi: 10.1145/2110147.2110156 (cit. on pp. 17–34, 76).


Seidl, Christoph, Florian Heidenreich, and Uwe Aßmann (2012). "Co-Evolution of Models and Feature Mapping in Software Product Lines". In: *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, pp. 76–85. doi: 10.1145/2362536.2362550 (cit. on pp. 17–34).


Erwig, Martin and Eric Walkingshaw (2011). "The Choice Calculus: A Representation for Software Variation". In: *Trans. on Software Engineering and Methodology (TOSEM)* 21.1, 6:1–6:27. doi: 10.1145/2063239.2063245 (cit. on pp. 17–34, 40–49).


Apel, Sven, Christian Lengauer, Bernhard Möller, and Christian Kästner (2010). "An Algebraic Foundation for Automatic Feature-Based Program Synthesis". In: *Science of Computer Programming (SCP)* 75.11, pp. 1022–1047 (cit. on pp. 17–34, 40–49).


Gruler, Alexander (2010). "A Formal Approach to Software Product Families". PhD thesis. TU München, p. 304 (cit. on pp. 17–34, 40–49).


Liebig, Jörg, Sven Apel, Christian Lengauer, Christian Kästner, and Michael Schulze (2010). "An Analysis of the Variability in Forty Preprocessor-Based Software Product Lines". In: *Proc. Int'l Conf. on Software Engineering (ICSE)*. IEEE, pp. 105–114. doi: 10.1145/1806799.1806819 (cit. on pp. 14, 15).


Canfora, Gerardo, Luigi Cerulo, and Massimiliano Di Penta (2009). "Ldiff: An Enhanced Line Differencing Tool". In: *Proc. Int'l Conf. on Software Engineering (ICSE)*. IEEE, pp. 595–598. doi: 10.1109/ICSE.2009.5070564 (cit. on pp. 17–34, 76).


Kästner, Christian, Sven Apel, and Martin Kuhlemann (2008). "Granularity in Software Product Lines". In: *Proc. Int'l Conf. on Software Engineering (ICSE)*. ACM, pp. 311–320. doi: 10.1145/1368088.1368131 (cit. on pp. 40–48).

Fluri, Beat, Michael Wuersch, Martin Pinzger, and Harald Gall (2007). "Change Distilling: Tree Differencing for Fine-Grained Source Code Change Extraction". In: *IEEE Trans. on Software Engineering (TSE)* 33.11, pp. 725–743. doi: 10.1109/TSE.2007.70731 (cit. on pp. 17–34, 76).

Apiwattanapong, Taweesup, Alessandro Orso, and Mary Jean Harrold (2004). "A Differencing Algorithm for Object-Oriented Programs". In: *Proc. Int'l Conf. on Automated Software Engineering (ASE)*. IEEE, pp. 2–13. doi: 10.1109/ASE.2004.10015 (cit. on pp. 17–34, 76).

Fischer, Michael, Martin Pinzger, and Harald C. Gall (2003). "Populating a Release History Database from Version Control and Bug Tracking Systems". In: *Proc. Int'l Conf. on Software Maintenance (ICSM)*. IEEE, p. 23. doi: 10.1109/ICSM.2003.1235403 (cit. on pp. 17–34, 76).

Bahar, R. Iris, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi (1993). "Algebraic Decision Diagrams and Their Applications". In: *Proc. Int'l Conf. on Computer-Aided Design (ICCAD)*. IEEE, pp. 188–191 (cit. on pp. 40–49).