Is the Feature Traceability Problem Already Solved?

Sandra Greiner **Timo Kehrer**

Software Engineering Group, University of Bern, CH
ACP Section, University of Southern Denmark, DK

WSRE, April 2024, Bad Honnef

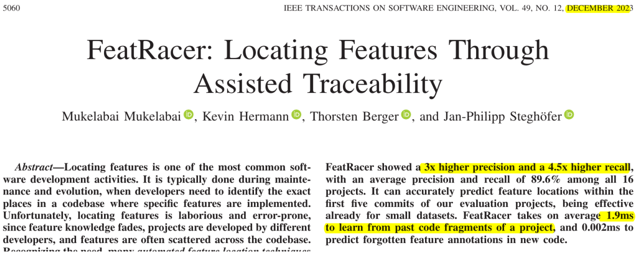
First thought ...

Yes,

First thought ...

Yes,

but ...

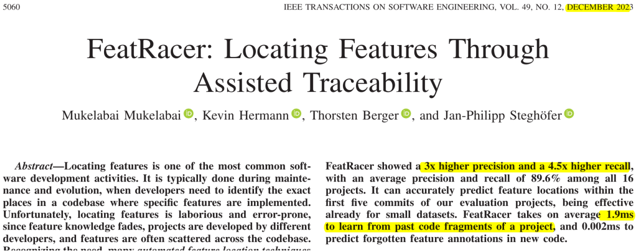


FeatRacer: Locating Features Through Assisted Traceability

Mukelabai Mukelabai, Kevin Hermann, Thorsten Berger, and Jan-Philipp Steghöfer

Abstract*—Locating features is one of the most common software development activities. It is typically done during maintenance and evolution, when developers need to identify the exact places in a codebase where specific features are implemented. Unfortunately, locating features is laborious and error-prone, since feature knowledge fades, projects are developed by different developers, and features are often scattered across the codebase. Recognizing the need, many *automated feature location techniques

FeatRacer showed a 3x higher precision and a 4.5x higher recall, with an average precision and recall of 89.6% among all 16 projects. It can accurately predict feature locations within the first five commits of our evaluation projects, being effective already for small datasets. FeatRacer takes on average 1.9ms to learn from past code fragments of a project, and 0.002ms to predict forgotten feature annotations in new code.



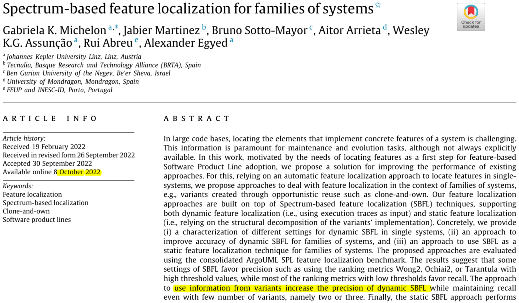
5060 IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 49, NO. 12, DECEMBER 2023

FeatRacer: Locating Features Through Assisted Traceability

Mukelabai Mukelabai, Kevin Hermann, Thorsten Berger, and Jan-Philipp Steghöfer

Abstract—Locating features is one of the most common software development activities. It is typically done during maintenance and evolution, when developers need to identify the exact places in a codebase where specific features are implemented. Unfortunately, locating features is laborious and error-prone, since feature knowledge fades, projects are developed by different developers, and features are often scattered across the codebase. Recognizing the need, many _automated feature location techniques_

FeatRacer showed a 3x higher precision and a 4.5x higher recall, with an average precision and recall of 89.6% among all 16 projects. It can accurately predict feature locations within the first five commits of our evaluation projects, being effective already for small datasets. FeatRacer takes on average 1.9ms to learn from past code fragments of a project, and 0.002ms to predict forgotten feature annotations in new code.



Spectrum-based feature localization for families of systems☆

Gabriela K. Michelon[a,*], Jabier Martinez[b], Bruno Sotto-Mayor[c], Aitor Arrieta[d], Wesley K.G. Assunção[a], Rui Abreu[e], Alexander Egyed[a]

[a] Johannes Kepler University Linz, Linz, Austria
[b] Tecnalia, Basque Research and Technology Alliance (BRTA), Spain
[c] Ben Gurion University of the Negev, Be'er Sheva, Israel
[d] University of Mondragon, Mondragon, Spain
[e] FEUP and INESC-ID, Porto, Portugal

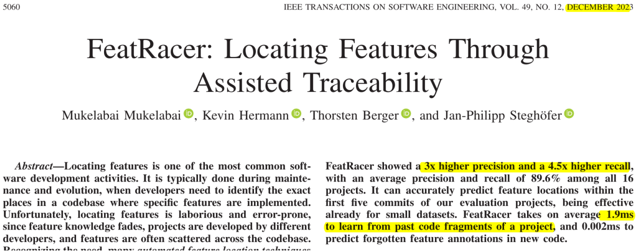
ARTICLE INFO

Article history:
Received 19 February 2022
Received in revised form 26 September 2022
Accepted 30 September 2022
Available online 8 October 2022

Keywords:
Feature localization
Spectrum-based localization
Clone-and-own
Software product lines

ABSTRACT

In large code bases, locating the elements that implement concrete features of a system is challenging. This information is paramount for maintenance and evolution tasks, although not always explicitly available. In this work, motivated by the needs of locating features as a first step for feature-based Software Product Line adoption, we propose a solution for improving the performance of existing approaches. For this, relying on an automatic feature localization approach to locate features in single-systems, we propose approaches to deal with feature localization in the context of families of systems, e.g., variants created through opportunistic reuse such as clone-and-own. Our feature localization approaches are built on top of Spectrum-based feature localization (SBFL) techniques, supporting both dynamic feature localization (i.e., using execution traces as input) and static feature localization (i.e., relying on the structural decomposition of the variants' implementation). Concretely, we provide (i) a characterization of different settings for dynamic SBFL in single systems, (ii) an approach to improve accuracy of dynamic SBFL for families of systems, and (iii) an approach to use SBFL as a static feature localization technique for families of systems. The proposed approaches are evaluated using the consolidated ArgoUML SPL feature localization benchmark. The results suggest that some settings of SBFL favor precision such as using the ranking metrics Wong2, Ochiai2, or Tarantula with high threshold values, while most of the ranking metrics with low thresholds favor recall. The approach to use information from variants increase the precision of dynamic SBFL while maintaining recall even with few number of variants, namely two or three. Finally, the static SBFL approach performs



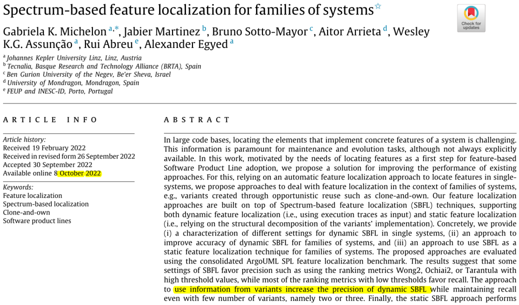
5060 IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 49, NO. 12, DECEMBER 2023

FeatRacer: Locating Features Through Assisted Traceability

Mukelabai Mukelabai, Kevin Hermann, Thorsten Berger, and Jan-Philipp Steghöfer

Abstract*—Locating features is one of the most common software development activities. It is typically done during maintenance and evolution, when developers need to identify the exact places in a codebase where specific features are implemented. Unfortunately, locating features is laborious and error-prone, since feature knowledge fades, projects are developed by different developers, and features are often scattered across the codebase. Recognizing the need, many *automated feature location techniques

FeatRacer showed a 3x higher precision and a 4.5x higher recall, with an average precision and recall of 89.6% among all 16 projects. It can accurately predict feature locations within the first five commits of our evaluation projects, being effective already for small datasets. FeatRacer takes on average 1.9ms to learn from past code fragments of a project, and 0.002ms to predict forgotten feature annotations in new code.



Spectrum-based feature localization for families of systems☆

Gabriela K. Michelon[a,*], Jabier Martinez[b], Bruno Sotto-Mayor[c], Aitor Arrieta[d], Wesley K.G. Assunção[a], Rui Abreu[e], Alexander Egyed[a]

[a] *Johannes Kepler University Linz, Linz, Austria*
[b] *Tecnalia, Basque Research and Technology Alliance (BRTA), Spain*
[c] *Ben Gurion University of the Negev, Be'er Sheva, Israel*
[d] *University of Mondragon, Mondragon, Spain*
[e] *FEUP and INESC-ID, Porto, Portugal*

ARTICLE INFO

Article history:
Received 19 February 2022
Received in revised form 26 September 2022
Accepted 30 September 2022
Available online 8 October 2022

Keywords:
Feature localization
Spectrum-based localization
Clone-and-own
Software product lines

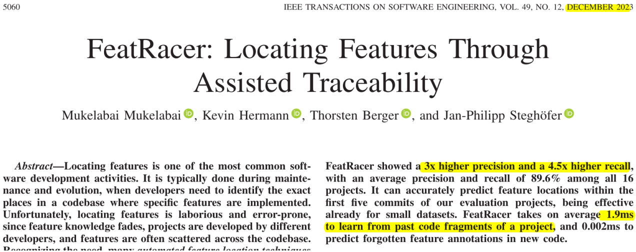
ABSTRACT

In large code bases, locating the elements that implement concrete features of a system is challenging. This information is paramount for maintenance and evolution tasks, although not always explicitly available. In this work, motivated by the needs of locating features as a first step for feature-based Software Product Line adoption, we propose a solution for improving the performance of existing approaches. For this, relying on an automatic feature localization approach to locate features in single-systems, we propose approaches to deal with feature localization in the context of families of systems, e.g., variants created through opportunistic reuse such as clone-and-own. Our feature localization approaches are built on top of Spectrum-based feature localization (SBFL) techniques, supporting both dynamic feature localization (i.e., using execution traces as input) and static feature localization (i.e., relying on the structural decomposition of the variants' implementation). Concretely, we provide (i) a characterization of different settings for dynamic SBFL in single systems, (ii) an approach to improve accuracy of dynamic SBFL for families of systems, and (iii) an approach to use SBFL as a static feature localization technique for families of systems. The proposed approaches are evaluated using the consolidated ArgoUML SPL feature localization benchmark. The results suggest that some settings of SBFL favor precision such as using the ranking metrics Wong2, Ochiai2, or Tarantula with high threshold values, while most of the ranking metrics with low thresholds favor recall. The approach to use information from variants increase the precision of dynamic SBFL while maintaining recall even with few number of variants, namely two or three. Finally, the static SBFL approach performs

manual vs automation

accuracy and reliability

maintenance over time



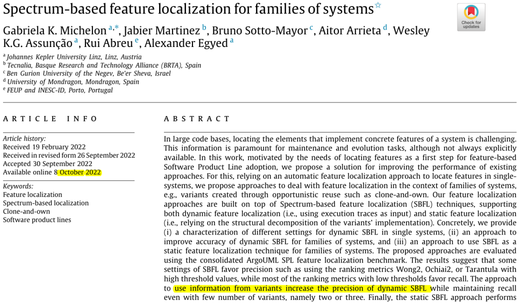
5060 IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 49, NO. 12, DECEMBER 2023

FeatRacer: Locating Features Through Assisted Traceability

Mukelabai Mukelabai, Kevin Hermann, Thorsten Berger, and Jan-Philipp Steghöfer

Abstract*—Locating features is one of the most common software development activities. It is typically done during maintenance and evolution, when developers need to identify the exact places in a codebase where specific features are implemented. Unfortunately, locating features is laborious and error-prone, since feature knowledge fades, projects are developed by different developers, and features are often scattered across the codebase. Recognizing the need, many *automated feature location techniques

FeatRacer showed a 3x higher precision and a 4.5x higher recall, with an average precision and recall of 89.6% among all 16 projects. It can accurately predict feature locations within the first five commits of our evaluation projects, being effective already for small datasets. FeatRacer takes on average 1.9ms to learn from past code fragments of a project, and 0.002ms to predict forgotten feature annotations in new code.



Spectrum-based feature localization for families of systems☆

Gabriela K. Michelon[a,*], Jabier Martinez[b], Bruno Sotto-Mayor[c], Aitor Arrieta[d], Wesley K.G. Assunção[a], Rui Abreu[e], Alexander Egyed[a]

[a] Johannes Kepler University Linz, Linz, Austria
[b] Tecnalia, Basque Research and Technology Alliance (BRTA), Spain
[c] Ben Gurion University of the Negev, Be'er Sheva, Israel
[d] University of Mondragon, Mondragon, Spain
[e] FEUP and INESC-ID, Porto, Portugal

ARTICLE INFO

Article history:
Received 19 February 2022
Received in revised form 26 September 2022
Accepted 30 September 2022
Available online 8 October 2022

Keywords:
Feature localization
Spectrum-based localization
Clone-and-own
Software product lines

ABSTRACT

In large code bases, locating the elements that implement concrete features of a system is challenging. This information is paramount for maintenance and evolution tasks, although not always explicitly available. In this work, motivated by the needs of locating features as a first step for feature-based Software Product Line adoption, we propose a solution for improving the performance of existing approaches. For this, relying on an automatic feature localization approach to locate features in single-systems, we propose approaches to deal with feature localization in the context of families of systems, e.g., variants created through opportunistic reuse such as clone-and-own. Our feature localization approaches are built on top of Spectrum-based feature localization (SBFL) techniques, supporting both dynamic feature localization (i.e., using execution traces as input) and static feature localization (i.e., relying on the structural decomposition of the variants' implementation). Concretely, we provide (i) a characterization of different settings for dynamic SBFL in single systems, (ii) an approach to improve accuracy of dynamic SBFL for families of systems, and (iii) an approach to use SBFL as a static feature localization technique for families of systems. The proposed approaches are evaluated using the consolidated ArgoUML SPL feature localization benchmark. The results suggest that some settings of SBFL favor precision such as using the ranking metrics Wong2, Ochiai2, or Tarantula with high threshold values, while most of the ranking metrics with low thresholds favor recall. The approach to use information from variants increase the precision of dynamic SBFL while maintaining recall even with few number of variants, namely two or three. Finally, the static SBFL approach performs

manual vs automation
accuracy and reliability
maintenance over time

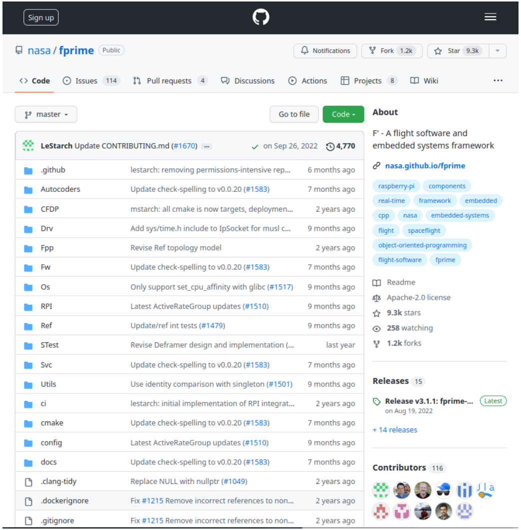
Feature Traceability

Feature Traceability

What is it and why should we care?

How can we do better?

Today: Large Evolving Software Projects



LeStarch Update CONTRIBUTING.md (#1670) ... ✓ on Sep 26, 2022 4,770

.github	lestarch: removing permissions-intensive rep...	6 months ago
Autocoders	Update check-spelling to v0.0.20 (#1583)	7 months ago
CFDP	mstarch: all cmake is now targets, deploymen...	2 years ago
Drv	Add sys/time.h include to IpSocket for musl c...	9 months ago
Fpp	Revise Ref topology model	2 years ago
Fw	Update check-spelling to v0.0.20 (#1583)	7 months ago
Os	Only support set_cpu_affinity with glibc (#1517)	9 months ago
RPI	Latest ActiveRateGroup updates (#1510)	9 months ago
Ref	Update/ref int tests (#1479)	9 months ago
STest	Revise Deframer design and implementation (...	last year
Svc	Update check-spelling to v0.0.20 (#1583)	7 months ago
Utils	Use identity comparison with singleton (#1501)	9 months ago
ci	lestarch: initial implementation of RPI integrat...	2 years ago
cmake	Update check-spelling to v0.0.20 (#1583)	7 months ago
config	Latest ActiveRateGroup updates (#1510)	9 months ago
docs	Update check-spelling to v0.0.20 (#1583)	7 months ago
.clang-tidy	Replace NULL with nullptr (#1049)	2 years ago
.dockerignore	Fix #1215 Remove incorrect references to non...	2 years ago
.gitignore	Fix #1215 Remove incorrect references to non...	2 years ago

About

F' - A flight software and embedded systems framework

nasa.github.io/fprime

raspberry-pi components real-time framework embedded cpp nasa embedded-systems flight spaceflight object-oriented-programming flight-software fprime

Readme
Apache-2.0 license
9.3k stars
258 watching
1.2k forks

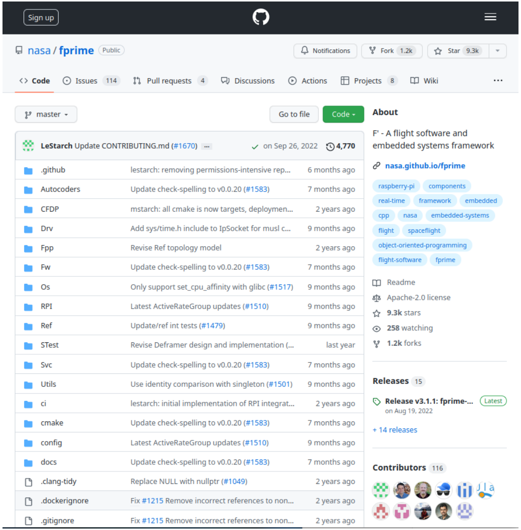
Releases 15

Release v3.1.1: fprime-... Latest
on Aug 19, 2022

+ 14 releases

Contributors 116

Today: Large Evolving Software Projects



LeStarch Update CONTRIBUTING.md (#1670) ✓ on Sep 26, 2022 4,770

.github	lestarch: removing permissions-intensive rep...	6 months ago
Autocoders	Update check-spelling to v0.0.20 (#1583)	7 months ago
CFDP	mstarch: all cmake is now targets, deploymen...	2 years ago
Drv	Add sys/time.h include to IpSocket for musl c...	9 months ago
Fpp	Revise Ref topology model	2 years ago
Fw	Update check-spelling to v0.0.20 (#1583)	7 months ago
Os	Only support set_cpu_affinity with glibc (#1517)	9 months ago
RPI	Latest ActiveRateGroup updates (#1510)	9 months ago
Ref	Update/ref int tests (#1479)	9 months ago
STest	Revise Deframer design and implementation (...	last year
Svc	Update check-spelling to v0.0.20 (#1583)	7 months ago
Utils	Use identity comparison with singleton (#1501)	9 months ago
ci	lestarch: initial implementation of RPI integrat...	2 years ago
cmake	Update check-spelling to v0.0.20 (#1583)	7 months ago
config	Latest ActiveRateGroup updates (#1510)	9 months ago
docs	Update check-spelling to v0.0.20 (#1583)	7 months ago
.clang-tidy	Replace NULL with nullptr (#1049)	2 years ago
.dockerignore	Fix #1215 Remove incorrect references to non...	2 years ago
.gitignore	Fix #1215 Remove incorrect references to non...	2 years ago

About

F' - A flight software and embedded systems framework

nasa.github.io/fprime

raspberry-pi, components, real-time, framework, embedded, cpp, nasa, embedded-systems, flight, spaceflight, object-oriented-programming, flight-software, fprime

Readme

Apache-2.0 license

9.3k stars

258 watching

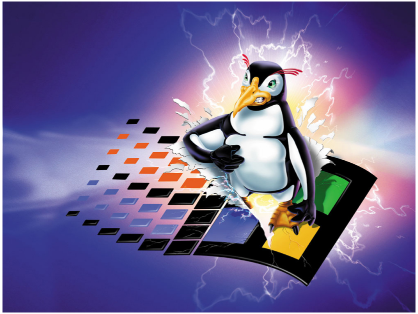
1.2k forks

Releases 15

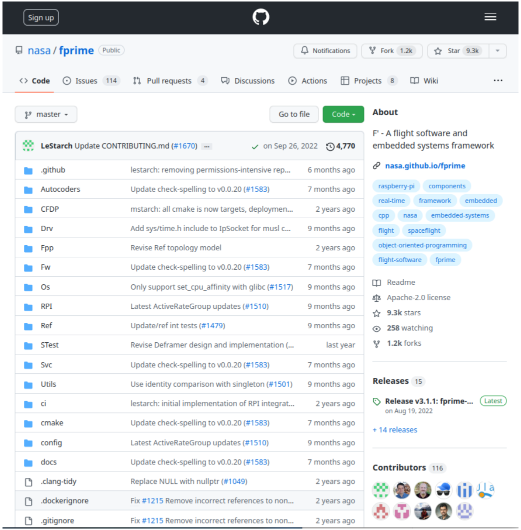
Release v3.1.1: fprime-... Latest
on Aug 19, 2022

+ 14 releases

Contributors 116



Today: Large Evolving Software Projects



nasa / fprime Public

LeStarch Update CONTRIBUTING.md (#1670) ... ✓ on Sep 26, 2022 4,770

.github	lestarch: removing permissions-intensive rep...	6 months ago
Autocoders	Update check-spelling to v0.0.20 (#1583)	7 months ago
CFDP	mstarch: all cmake is now targets, deploymen...	2 years ago
Drv	Add sys/time.h include to IpSocket for musl c...	9 months ago
Fpp	Revise Ref topology model	2 years ago
Fw	Update check-spelling to v0.0.20 (#1583)	7 months ago
Os	Only support set_cpu_affinity with glibc (#1517)	9 months ago
RPI	Latest ActiveRateGroup updates (#1510)	9 months ago
Ref	Update/ref int tests (#1479)	9 months ago
STest	Revise Deframer design and implementation (...	last year
Svc	Update check-spelling to v0.0.20 (#1583)	7 months ago
Utils	Use identity comparison with singleton (#1501)	9 months ago
ci	lestarch: initial implementation of RPI integrat...	2 years ago
cmake	Update check-spelling to v0.0.20 (#1583)	7 months ago
config	Latest ActiveRateGroup updates (#1510)	9 months ago
docs	Update check-spelling to v0.0.20 (#1583)	7 months ago
.clang-tidy	Replace NULL with nullptr (#1049)	2 years ago
.dockerignore	Fix #1215 Remove incorrect references to non...	2 years ago
.gitignore	Fix #1215 Remove incorrect references to non...	2 years ago

About

F' - A flight software and embedded systems framework

nasa.github.io/fprime

raspberry-pi components real-time framework embedded cpp nasa embedded-systems flight spaceflight object-oriented-programming flight-software fprime

Readme

Apache-2.0 license

9.3k stars

258 watching

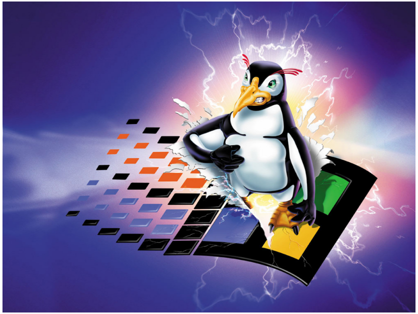
1.2k forks

Releases 15

Release v3.1.1: fprime-... Latest
on Aug 19, 2022

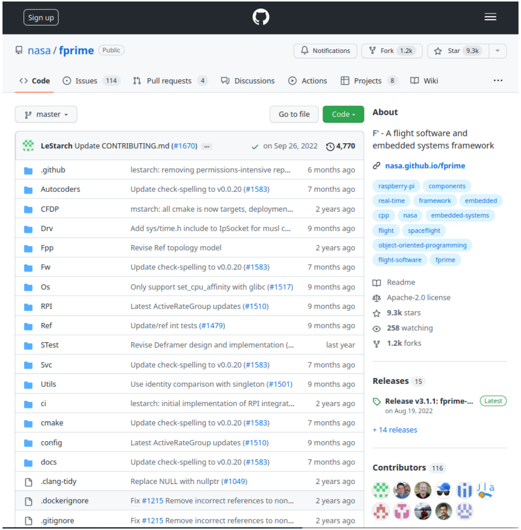
+ 14 releases

Contributors 116



hardly possible to comprehend, analyze, and modernize

Today: Large Evolving Software Projects



LeStarch Update CONTRIBUTING.md (#1670) ... ✓ on Sep 26, 2022 4,770

.github	lestarch: removing permissions-intensive rep...	6 months ago
Autocoders	Update check-spelling to v0.0.20 (#1583)	7 months ago
CFDP	mstarch: all cmake is now targets, deploymen...	2 years ago
Drv	Add sys/time.h include to IpSocket for musl c...	9 months ago
Fpp	Revise Ref topology model	2 years ago
Fw	Update check-spelling to v0.0.20 (#1583)	7 months ago
Os	Only support set_cpu_affinity with glibc (#1517)	9 months ago
RPI	Latest ActiveRateGroup updates (#1510)	9 months ago
Ref	Update/ref int tests (#1479)	9 months ago
STest	Revise Deframer design and implementation (...	last year
Svc	Update check-spelling to v0.0.20 (#1583)	7 months ago
Utils	Use identity comparison with singleton (#1501)	9 months ago
ci	lestarch: initial implementation of RPI integrat...	2 years ago
cmake	Update check-spelling to v0.0.20 (#1583)	7 months ago
config	Latest ActiveRateGroup updates (#1510)	9 months ago
docs	Update check-spelling to v0.0.20 (#1583)	7 months ago
.clang-tidy	Replace NULL with nullptr (#1049)	2 years ago
.dockerignore	Fix #1215 Remove incorrect references to non...	2 years ago
.gitignore	Fix #1215 Remove incorrect references to non...	2 years ago

About

F' - A flight software and embedded systems framework

nasa.github.io/fprime

raspberry-pi, components, real-time, framework, embedded, cpp, nasa, embedded-systems, flight, spaceflight, object-oriented-programming, flight-software, fprime

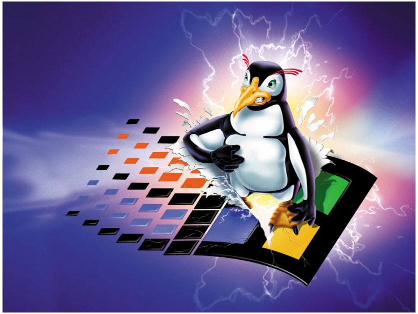
Readme
Apache-2.0 license
9.3k stars
258 watching
1.2k forks

Releases 15

Release v3.1.1: fprime-... Latest
on Aug 19, 2022

+ 14 releases

Contributors 116

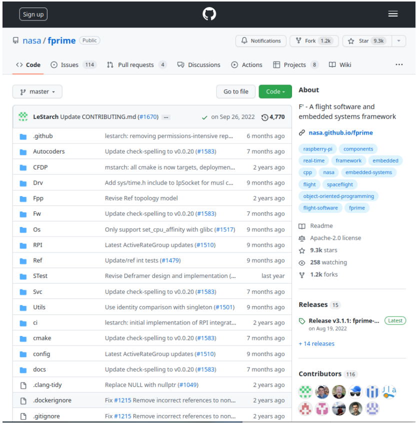


hardly possible to comprehend, analyze, and modernize

particularly, if highly configurable

Problem Statement: Large Evolving Highly-Configurable Software Projects

FPrime



nasa / fprime Public

LeStarch Update CONTRIBUTING.md (#1670) ... ✓ on Sep 26, 2022 4,770

.github	lestarch: removing permissions-intensive rep...	6 months ago
Autocoders	Update check-spelling to v0.0.20 (#1583)	7 months ago
CFDP	mstarch: all cmake is now targets, deploymen...	2 years ago
Drv	Add sys/time.h include to IpSocket for musl c...	9 months ago
Fpp	Revise Ref topology model	2 years ago
Fw	Update check-spelling to v0.0.20 (#1583)	7 months ago
Os	Only support set_cpu_affinity with glibc (#1517)	9 months ago
RPI	Latest ActiveRateGroup updates (#1510)	9 months ago
Ref	Update/ref int tests (#1479)	9 months ago
STest	Revise Deframer design and implementation (...	last year
Svc	Update check-spelling to v0.0.20 (#1583)	7 months ago
Utils	Use identity comparison with singleton (#1501)	9 months ago
ci	lestarch: initial implementation of RPI integrat...	2 years ago
cmake	Update check-spelling to v0.0.20 (#1583)	7 months ago
config	Latest ActiveRateGroup updates (#1510)	9 months ago
docs	Update check-spelling to v0.0.20 (#1583)	7 months ago
.clang-tidy	Replace NULL with nullptr (#1049)	2 years ago
.dockerignore	Fix #1215 Remove incorrect references to non...	2 years ago
.gitignore	Fix #1215 Remove incorrect references to non...	2 years ago

About

F' - A flight software and embedded systems framework

nasa.github.io/fprime

raspberry-pi components real-time framework embedded cpp nasa embedded-systems flight spaceflight object-oriented-programming flight-software fprime

Readme
Apache-2.0 license
9.3k stars
258 watching
1.2k forks

Releases 15

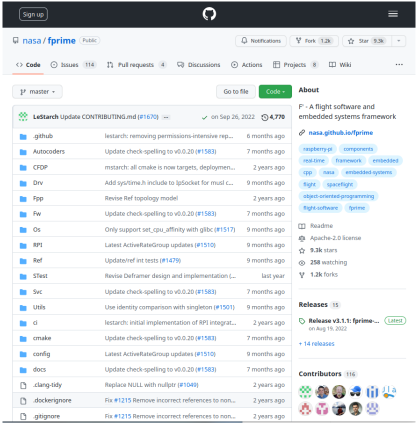
Release v3.1.1: fprime-... Latest
on Aug 19, 2022

+ 14 releases

Contributors 116

Problem Statement: Large Evolving Highly-Configurable Software Projects

FPrime



Sign up

nasa / **fprime** Public

Notifications | Fork 1.2k | Star 9.3k

Code | Issues 114 | Pull requests 4 | Discussions | Actions | Projects 8 | Wiki | ···

master ▾ | Go to file | Code ▾

LeStarch Update CONTRIBUTING.md (#1670) ··· ✓ on Sep 26, 2022 **4,770**

.github	lestarch: removing permissions-intensive rep...	6 months ago
Autocoders	Update check-spelling to v0.0.20 (#1583)	7 months ago
CFDP	mstarch: all cmake is now targets, deploymen...	2 years ago
Drv	Add sys/time.h include to IpSocket for musl c...	9 months ago
Fpp	Revise Ref topology model	2 years ago
Fw	Update check-spelling to v0.0.20 (#1583)	7 months ago
Os	Only support set_cpu_affinity with glibc (#1517)	9 months ago
RPI	Latest ActiveRateGroup updates (#1510)	9 months ago
Ref	Update/ref int tests (#1479)	9 months ago
STest	Revise Deframer design and implementation (...	last year
Svc	Update check-spelling to v0.0.20 (#1583)	7 months ago
Utils	Use identity comparison with singleton (#1501)	9 months ago
ci	lestarch: initial implementation of RPI integrat...	2 years ago
cmake	Update check-spelling to v0.0.20 (#1583)	7 months ago
config	Latest ActiveRateGroup updates (#1510)	9 months ago
docs	Update check-spelling to v0.0.20 (#1583)	7 months ago
.clang-tidy	Replace NULL with nullptr (#1049)	2 years ago
.dockerignore	Fix #1215 Remove incorrect references to non...	2 years ago
.gitignore	Fix #1215 Remove incorrect references to non...	2 years ago

About

F' - A flight software and embedded systems framework

nasa.github.io/fprime

raspberry-pi | components | real-time | framework | embedded | cpp | nasa | embedded-systems | flight | spaceflight | object-oriented-programming | flight-software | fprime

Readme

Apache-2.0 license

9.3k stars

258 watching

1.2k forks

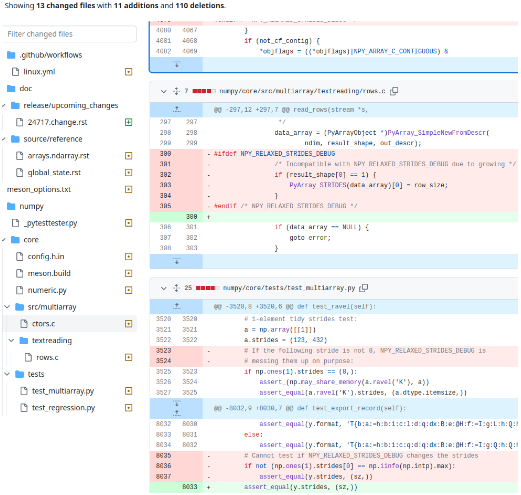
Releases 15

Release v3.1.1: fprime-... Latest
on Aug 19, 2022

+ 14 releases

Contributors 116

numpy



Showing **13 changed files** with **11 additions** and **110 deletions**.

Filter changed files

- .github/workflows
  - linux.yml
- doc
  - release/upcoming_changes
    - 24717.change.rst
  - source/reference
    - arrays.ndarray.rst
    - global_state.rst
- meson_options.txt
- numpy
  - _pytesttester.py
  - core
    - config.h.in
    - meson.build
    - numeric.py
    - src/multiarray
      - ctors.c
      - textreading
        - rows.c
    - tests
      - test_multiarray.py
      - test_regression.py

```
                }
                if (not_cf_contig) {
                    *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
```

7 numpy/core/src/multiarray/textreading/rows.c

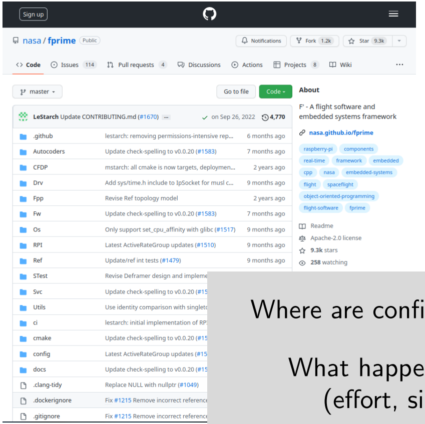
```
@@ -297,12 +297,7 @@ read_rows(stream *s,
                         */
                        data_array = (PyArrayObject *)PyArray_SimpleNewFromDescr(
                                ndim, result_shape, out_descr);
- #ifdef NPY_RELAXED_STRIDES_DEBUG
-                       /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
-                       if (result_shape[0] == 1) {
-                           PyArray_STRIDES(data_array)[0] = row_size;
-                       }
- #endif /* NPY_RELAXED_STRIDES_DEBUG */
+
                        if (data_array == NULL) {
                            goto error;
                        }
```

25 numpy/core/tests/test_multiarray.py

```
@@ -3520,8 +3520,6 @@ def test_ravel(self):
                # 1-element tidy strides test:
                a = np.array([[1]])
                a.strides = (123, 432)
-               # If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
-               # messing them up on purpose:
                if np.ones(1).strides == (8,):
                    assert_(np.may_share_memory(a.ravel('K'), a))
                    assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
@@ -8032,9 +8030,7 @@ def test_export_record(self):
                    assert_equal(y.format, 'T{b:a:=h:b:i:c:l:d:q:dx:B:e:@H:f:=I:g:L:h:Q:h
                else:
                    assert_equal(y.format, 'T{b:a:=h:b:i:c:q:d:q:dx:B:e:@H:f:=I:g:Q:h:Q:h
-               # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
-               if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
-                   assert_equal(y.strides, (sz,))
+               assert_equal(y.strides, (sz,))
```

Problem Statement: Large Evolving Highly-Configurable Software Projects

FPrime



LeStarch Update CONTRIBUTING.md (#1670) ... ✓ on Sep 26, 2022 4,770

.github	lestarch: removing permissions-intensive rep...	6 months ago
Autocoders	Update check-spelling to v0.0.20 (#1583)	7 months ago
CFDP	mstarch: all cmake is now targets, deploymen...	2 years ago
Drv	Add sys/time.h include to IpSocket for musl c...	9 months ago
Fpp	Revise Ref topology model	2 years ago
Fw	Update check-spelling to v0.0.20 (#1583)	7 months ago
Os	Only support set_cpu_affinity with glibc (#1517)	9 months ago
RPI	Latest ActiveRateGroup updates (#1510)	9 months ago
Ref	Update/ref int tests (#1479)	9 months ago
STest	Revise Deframer design and impleme	
Svc	Update check-spelling to v0.0.20 (#15	
Utils	Use identity comparison with singlet	
ci	lestarch: initial implementation of RP	
cmake	Update check-spelling to v0.0.20 (#15	
config	Latest ActiveRateGroup updates (#15	
docs	Update check-spelling to v0.0.20 (#15	
.clang-tidy	Replace NULL with nullptr (#1049)	
.dockerignore	Fix #1215 Remove incorrect reference	
.gitignore	Fix #1215 Remove incorrect reference	

About

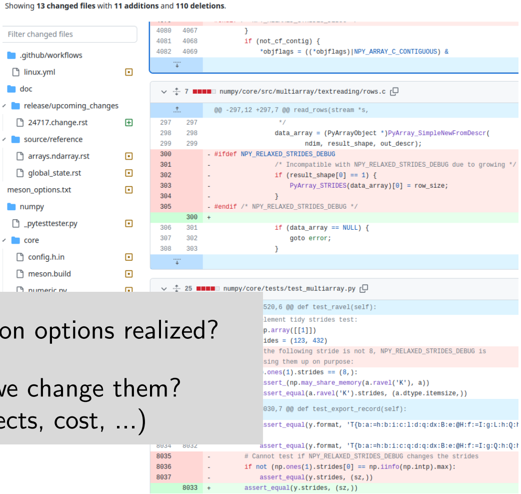
F' - A flight software and embedded systems framework

nasa.github.io/fprime

raspberry-pi components real-time framework embedded cpp nasa embedded-systems flight spaceflight object-oriented-programming flight-software fprime

Readme
Apache-2.0 license
9.3k stars
258 watching

numpy



Showing **13 changed files** with **11 additions** and **110 deletions**.

```
            }
            if (not_cf_contig) {
                *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
```

7 numpy/core/src/multiarray/textreading/rows.c

```
@@ -297,12 +297,7 @@ read_rows(stream *s,
                     */
                    data_array = (PyArrayObject *)PyArray_SimpleNewFromDescr(
                            ndim, result_shape, out_descr);
- #ifdef NPY_RELAXED_STRIDES_DEBUG
-                   /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
-                   if (result_shape[0] == 1) {
-                       PyArray_STRIDES(data_array)[0] = row_size;
-                   }
- #endif /* NPY_RELAXED_STRIDES_DEBUG */
+
                    if (data_array == NULL) {
                        goto error;
                    }
```

25 numpy/core/tests/test_multiarray.py

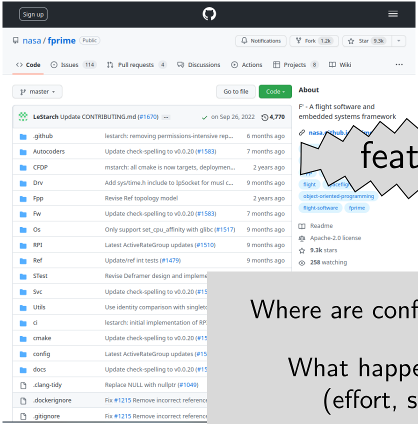
```
520,6 @@ def test_ravel(self):
lement tidy strides test:
p.array([[1]])
ides = (123, 432)
the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
sing them up on purpose:
.ones(1).strides == (8,):
ssert_(np.may_share_memory(a.ravel('K'), a))
ssert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
030,7 @@ def test_export_record(self):
ssert_equal(y.format, 'T{b:a:=h:b:i:c:l:d:q:dx:B:e:@H:f:=I:g:L:h:Q:h

        assert_equal(y.format, 'T{b:a:=h:b:i:c:q:d:q:dx:B:e:@H:f:=I:g:Q:h:Q:h
-       # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
-       if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
-           assert_equal(y.strides, (sz,))
+       assert_equal(y.strides, (sz,))
```

Where are configuration options realized?

What happens if we change them?
(effort, side effects, cost, ...)

Problem Statement: Large Evolving Highly-Configurable Software Projects

FPrime



LeStarch Update CONTRIBUTING.md (#1670) ... ✓ on Sep 26, 2022 4,770

.github	lestarch: removing permissions-intensive rep...	6 months ago
Autocoders	Update check-spelling to v0.0.20 (#1583)	7 months ago
CFDP	mstarch: all cmake is now targets, deploymen...	2 years ago
Drv	Add sys/time.h include to IpSocket for musl c...	9 months ago
Fpp	Revise Ref topology model	2 years ago
Fw	Update check-spelling to v0.0.20 (#1583)	7 months ago
Os	Only support set_cpu_affinity with glibc (#1517)	9 months ago
RPI	Latest ActiveRateGroup updates (#1510)	9 months ago
Ref	Update/ref int tests (#1479)	9 months ago
STest	Revise Deframer design and impleme	
Svc	Update check-spelling to v0.0.20 (#15	
Utils	Use identity comparison with singlet	
ci	lestarch: initial implementation of RP	
cmake	Update check-spelling to v0.0.20 (#15	
config	Latest ActiveRateGroup updates (#15	
docs	Update check-spelling to v0.0.20 (#15	
.clang-tidy	Replace NULL with nullptr (#1049)	
.dockerignore	Fix #1215 Remove incorrect reference	
.gitignore	Fix #1215 Remove incorrect reference	

About

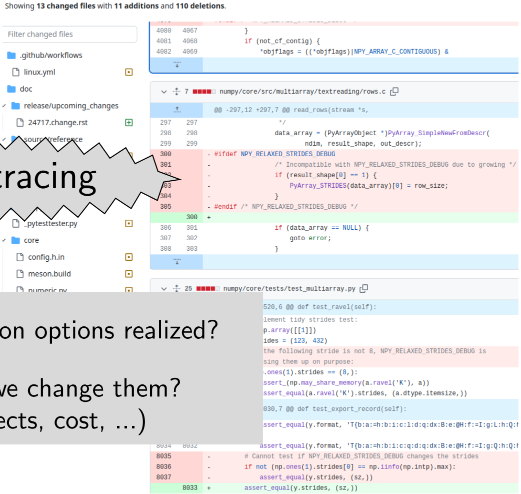
F' - A flight software and embedded systems framework

nasa.github.i

flight spacefligh object-oriented-programming flight-software fprime

Readme
Apache-2.0 license
9.3k stars
258 watching

numpy



Showing **13 changed files** with **11 additions** and **110 deletions**.

```
            }
            if (not_cf_contig) {
                *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
```

7 numpy/core/src/multiarray/textreading/rows.c

```
@@ -297,12 +297,7 @@ read_rows(stream *s,
                  */
                 data_array = (PyArrayObject *)PyArray_SimpleNewFromDescr(
                         ndim, result_shape, out_descr);
- #ifdef NPY_RELAXED_STRIDES_DEBUG
-                /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
-                if (result_shape[0] == 1) {
-                    PyArray_STRIDES(data_array)[0] = row_size;
-                }
- #endif /* NPY_RELAXED_STRIDES_DEBUG */
+
                 if (data_array == NULL) {
                     goto error;
                 }
```

25 numpy/core/tests/test_multiarray.py

```
520,6 @@ def test_ravel(self):
lement tidy strides test:
p.array([[1]])
ides = (123, 432)
the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
sing them up on purpose:
.ones(1).strides == (8,):
ssert_(np.may_share_memory(a.ravel('K'), a))
ssert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
030,7 @@ def test_export_record(self):
ssert_equal(y.format, 'T{b:a:=h:b:i:c:l:d:q:dx:B:e:@H:f:=I:g:L:h:Q:h
        assert_equal(y.format, 'T{b:a:=h:b:i:c:q:d:q:dx:B:e:@H:f:=I:g:Q:h:Q:h
-       # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
-       if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
-           assert_equal(y.strides, (sz,))
+       assert_equal(y.strides, (sz,))
```

feature tracing

Where are configuration options realized?

What happens if we change them?
(effort, side effects, cost, ...)

Background

How does feature tracing work?

Example: Configurable (?) Graph Implementations

Example: Configurable (?) Graph Implementations

single system

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

multiple variants

Example: Configurable (?) Graph Implementations

single system

```java
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

multiple variants

```java
/// V1: {G, E, W}
// FEAT: Graph
interface Graph {
  List<Node> nodes();
  // FEAT: E
  List<Edge> edges();
  List<Node> nodes(double w);
}
```

Example: Configurable (?) Graph Implementations

single system

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

multiple variants

```
/// V1: {G, E, W}
// FEAT: Graph
interface Graph {
  List<Node> nodes();
  // FEAT: E
  List<Edge> edges();
  List<Node> nodes(double w);
}
```

```
/// V2: {G, E, C}
interface Graph {
  List<Node> nodes();
  // FEAT: C
  List<Node> nodes(Color c);
  List<Edge> edges();
  Graph subGraph(Color c);
}
```

Example: Configurable (?) Graph Implementations

single system

```java
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

multiple variants

```java
/// V1: {G, E, W}
// FEAT: Graph
interface Graph {
  List<Node> nodes();
  // FEAT: E
  List<Edge> edges();
  List<Node> nodes(double w);
}
```

```java
/// V2: {G, E, C}
interface Graph {
  List<Node> nodes();
  // FEAT: C
  List<Node> nodes(Color c);
  List<Edge> edges();
  Graph subGraph(Color c);
}
```

```java
/// V3: {G, E, D}
interface Graph {
  List<Node> nodes();
  List<Edge> edges();
  List<Edge> incomEdges(Node n);
  // FEAT: C
  Graph subGraph(Color c);
}
```

Proactive Tracing

through

commit messages

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

Proactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

through

- commit messages
- development on branches

Proactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

through

- commit messages
- development on branches
- semi-automated feature trace recording

Proactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

through

commit messages
development on branches
semi-automated feature trace recording
semi-automated nudging (based on reinforcement learning)

Proactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

through

commit messages
development on branches
semi-automated feature trace recording
semi-automated nudging (based on reinforcement learning)

requires

discipline
development practices and guidelines
remains a **manual** task

Proactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

through

- commit messages
- development on branches
- semi-automated feature trace recording
- semi-automated nudging (based on reinforcement learning)

requires

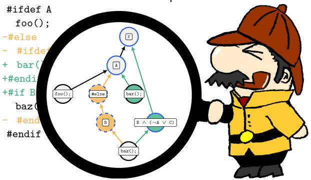
- discipline
- development practices and guidelines
- remains a **manual** task

⇒ often neglected

⇒ missed opportunity

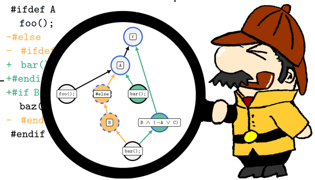
Retroactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```



Retroactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```



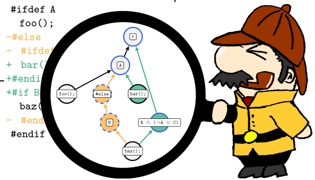
```
 #ifdef A
   foo();
-#else
-  #ifde
+  bar(
+#endi
+#if B
   baz(
-  #end
 #endif
```

through

manual code inspection

Retroactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

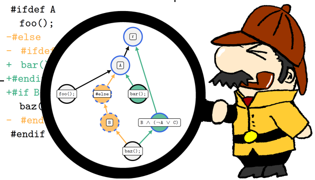


through

- manual code inspection
- dynamic techniques (which compare executions)

Retroactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

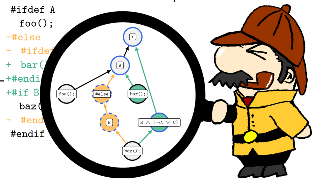


through

- manual code inspection
- dynamic techniques (which compare executions)
- static techniques (e.g., slicing, clone detection, comparison of variants)

Retroactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

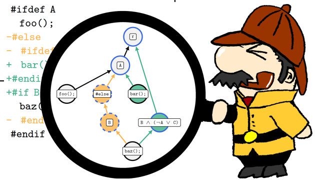


through

- manual code inspection
- dynamic techniques (which compare executions)
- static techniques (e.g., slicing, clone detection, comparison of variants)
- combinations thereof

Retroactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```



through

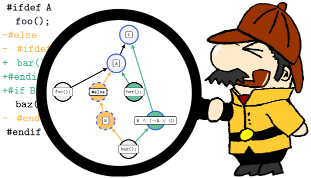
- manual code inspection
- dynamic techniques (which compare executions)
- static techniques (e.g., slicing, clone detection, comparison of variants)
- combinations thereof

requires

- heuristics
- execution effort and data
- while automated, reliability?

Retroactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```



```
 #ifdef A
   foo();
-#else
-  #ifde
+  bar(
+#endi
+#if B
   baz(
-  #end
 #endif
```

through

- manual code inspection
- dynamic techniques (which compare executions)
- static techniques (e.g., slicing, clone detection, comparison of variants)
- combinations thereof

requires

- heuristics
- execution effort and data
- while automated, reliability?
- → **manual** task

Feature Tracing – Summary

proactive

reliable*

pin-pointed

Feature Tracing – Summary

proactive

reliable*

pin-pointed

without immediate benefit

additional documentation burden

(*exclude uncertainty, feature interactions...)

Feature Tracing – Summary

proactive

reliable*
pin-pointed
without immediate benefit
additional documentation burden

(*exclude uncertainty, feature interactions...)

retroactive

manual vs. automated

Feature Tracing – Summary

proactive

- reliable*
- pin-pointed
- without immediate benefit
- additional documentation burden

(*exclude uncertainty, feature interactions...)

retroactive

- manual vs. automated
- automated: static or dynamic or hybrid

Feature Tracing – Summary

proactive

reliable*
pin-pointed
without immediate benefit
additional documentation burden

(*exclude uncertainty, feature interactions...)

retroactive

manual vs. automated
automated: static or dynamic or hybrid
based on heuristics

Feature Tracing – Summary

proactive

reliable*

pin-pointed

without immediate benefit

additional documentation burden

(*exclude uncertainty, feature interactions...)

retroactive

manual vs. automated

automated: static or dynamic or hybrid

based on heuristics

⇒ less reliable

⇒ many techniques, but not optimal

Feature Traceability

Feature Traceability

What is it and why should we care?

How can we do better?

Proactive + Retroactive Feature Tracing

How can retroactive feature tracing benefit from proactive traces?

Proactive + Retroactive Feature Tracing

How can retroactive feature tracing benefit from proactive traces?

Results from experimenting with comparison-based feature location

Comparison-Based Feature Location

```
/// V1: {G, E, W}
// FEAT: Graph
interface Graph {
  List<Node> nodes();
  // FEAT: E
  List<Edge> edges();
  List<Node> nodes(double w);
}
```

```
/// V3: {G, E, D}
interface Graph {
  List<Node> nodes();
  List<Edge> edges();
  List<Edge> incomEdges(Node n);
  // FEAT: C
  Graph subGraph(Color c);
}
```

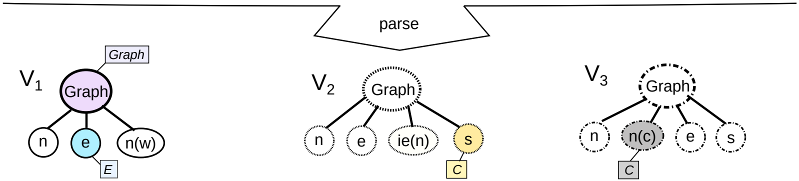
```
/// V2: {G, E, C}
interface Graph {
  List<Node> nodes();
  // FEAT: C
  List<Node> nodes(Color c);
  List<Edge> edges();
  Graph subGraph(Color c);
}
```

Comparison-Based Feature Location

```
/// V1: {G, E, W}
// FEAT: Graph
interface Graph {
  List<Node> nodes();
  // FEAT: E
  List<Edge> edges();
  List<Node> nodes(double w);
}
```

```
/// V3: {G, E, D}
interface Graph {
  List<Node> nodes();
  List<Edge> edges();
  List<Edge> incomEdges(Node n);
  // FEAT: C
  Graph subGraph(Color c);
}
```

```
/// V2: {G, E, C}
interface Graph {
  List<Node> nodes();
  // FEAT: C
  List<Node> nodes(Color c);
  List<Edge> edges();
  Graph subGraph(Color c);
}
```

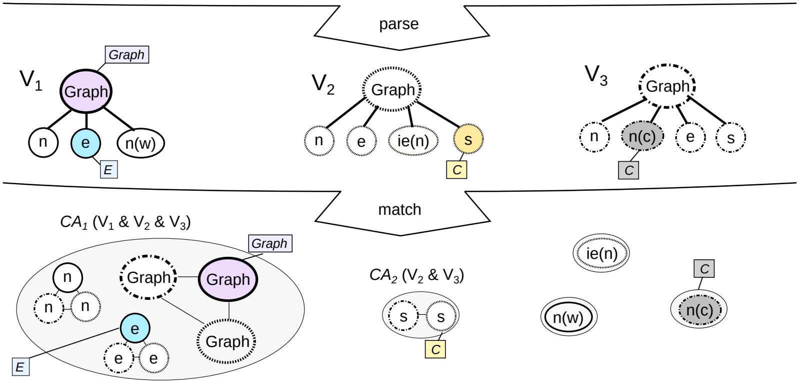


Comparison-Based Feature Location

```
/// V1: {G, E, W}
// FEAT: Graph
interface Graph {
  List<Node> nodes();
  // FEAT: E
  List<Edge> edges();
  List<Node> nodes(double w);
}
```

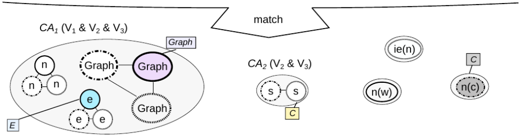
```
/// V3: {G, E, D}
interface Graph {
  List<Node> nodes();
  List<Edge> edges();
  List<Edge> incomEdges(Node n);
  // FEAT: C
  Graph subGraph(Color c);
}
```

```
/// V2: {G, E, C}
interface Graph {
  List<Node> nodes();
  // FEAT: C
  List<Node> nodes(Color c);
  List<Edge> edges();
  Graph subGraph(Color c);
}
```



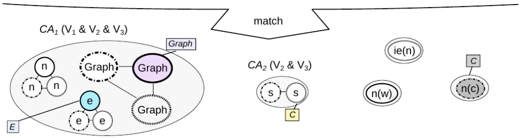
Comparison-Based Feature Location

annotation per node (based on features in configurations)



Comparison-Based Feature Location

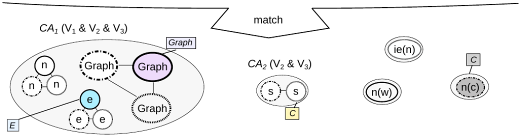
annotation per node (based on features in configurations)



Feature Trace

Comparison-Based Feature Location

annotation per node (based on features in configurations)

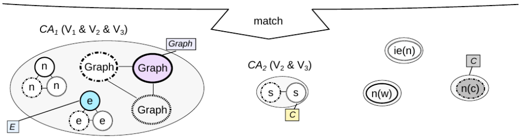


Feature Trace

Graph ∧ Edge	interface Graph, nodes(), edges(), subGraph()
Weighted	nodes (double w)
Colored	nodes(Color c)
Directed	incomingEdges(Edge e)

Comparison-Based Feature Location

annotation per node (based on features in configurations)



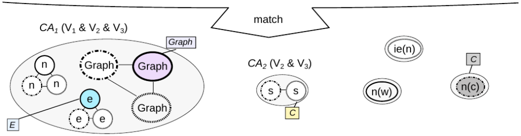
Feature Trace

Graph ∧ Edge	interface Graph, nodes(), edges(), subGraph()
Weighted	nodes (double w)
Colored	nodes(Color c)
Directed	incomingEdges(Edge e)

Controlled Experiment

Comparison-Based Feature Location

annotation per node (based on features in configurations)



Feature Trace

`Graph ∧ Edge`	`interface Graph`, `nodes()`, `edges()`, `subGraph()`
`Weighted`	`nodes (double w)`
`Colored`	nodes(Color c)
`Directed`	`incomingEdges(Edge e)`

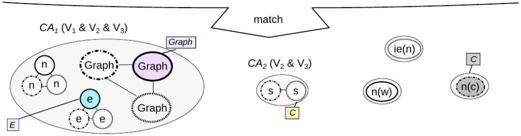
Controlled Experiment

propagate ‘reliable annotation’ whenever possible (not contradicting among one set)

measure quantitatively

Comparison-Based Feature Location

annotation per node (based on features in configurations)



Feature Trace

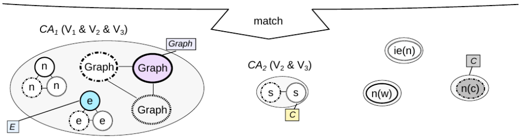
Graph ∧ Edge	interface Graph, nodes(), edges(), subGraph()
Weighted	nodes (double w)
Colored	nodes(Color c)
Directed	incomingEdges(Edge e)

Controlled Experiment

- propagate ‘reliable annotation’ whenever possible (not contradicting among one set)
- measure quantitatively
    - effect of adding proactive mappings per variant
    - effect of number of compared variants

Comparison-Based Feature Location

annotation per node (based on features in configurations)



Feature Trace

`Graph ∧ Edge`	`interface Graph, nodes(), edges(), subGraph()`
`Weighted`	`nodes (double w)`
`Colored`	nodes(Color c)
`Directed`	`incomingEdges(Edge e)`

Controlled Experiment

propagate ‘reliable annotation’ whenever possible (not contradicting among one set)

measure quantitatively

effect of adding proactive mappings per variant

effect of number of compared variants

⇒ increase in accuracy?

⇒ invest in exploiting proactive knowledge?

Setup

Subject Systems: Marlin, ArgoUML, VIM; OpenVPN, BusyBox

Setup

Subject Systems: Marlin, ArgoUML, VIM; OpenVPN, BusyBox

Groundtruth: VEVOS, Benchmark generator for highly configurable software

Setup

Subject Systems: Marlin, ArgoUML, VIM; OpenVPN, BusyBox

Groundtruth: VEVOS, Benchmark generator for highly configurable software

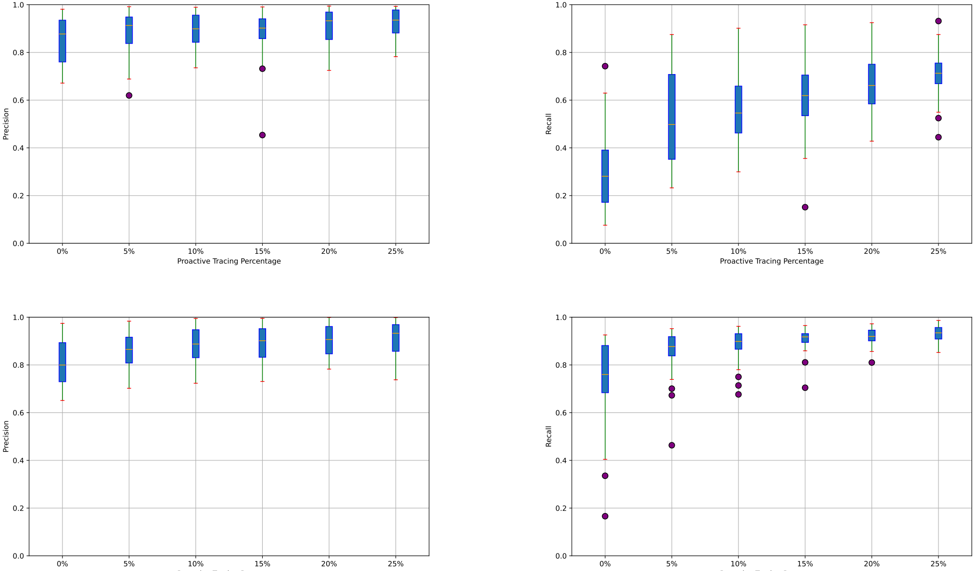
Measure:

Agreement between ground truth and computed mapping to keep a line of code in variant

Compute: precision, recall, F1-Score

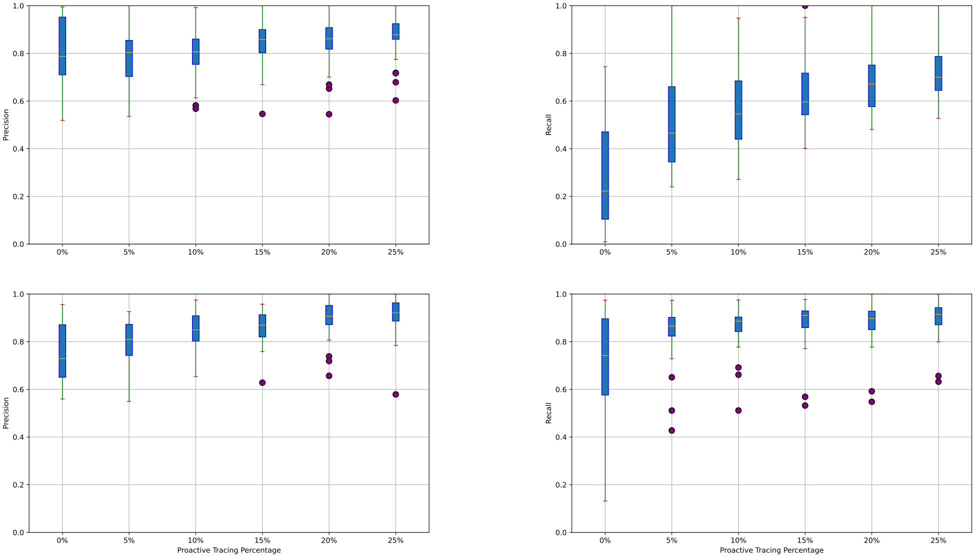
Results : Busybox (precision & recall (row), 3 and 7 variants (columns))

Results : Busybox (precision & recall (row), 3 and 7 variants (columns))



Results : VIM (precision & recall (row), 3 and 7 variants (columns))

Results : VIM (precision & recall (row), 3 and 7 variants (columns))



Results: Summary

only 5% of proactive trace raise the overall accuracy by 10-20%

Results: Summary

only 5% of proactive trace raise the overall accuracy by 10-20%

difference between precision and recall → many false positives
(→ include more than necessary in source code)

Results: Summary

only 5% of proactive trace raise the overall accuracy by 10-20%

difference between precision and recall → many false positives
(→ include more than necessary in source code)

the more variants, the higher the accuracy (with and without added traces)

Results: Summary

only 5% of proactive trace raise the overall accuracy by 10-20%

difference between precision and recall → many false positives
(→ include more than necessary in source code)

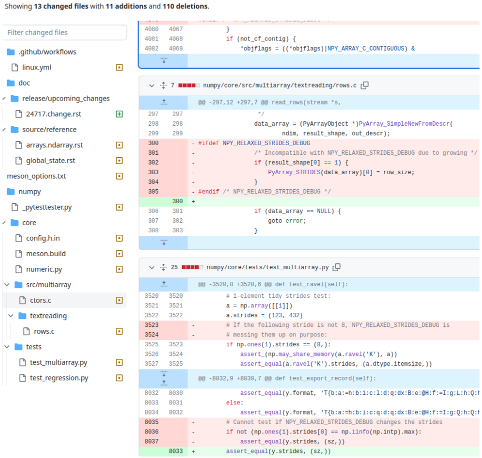
the more variants, the higher the accuracy (with and without added traces)

⇒ potential to exploit proactive traces further

⇒ examine the effect in further techniques, also qualitatively

⇒ optimize retroactive techniques but also inform semi-automated tracing (e.g. for machine learning)

Summary



Showing **13 changed files** with **11 additions** and **110 deletions**.

Filter changed files

- .github/workflows
  - linux.yml
- doc
  - release/upcoming_changes
    - 24717.change.rst
  - source/reference
    - arrays.ndarray.rst
    - global_state.rst
- meson_options.txt
- numpy
  - _pytesttester.py
  - core
    - config.h.in
    - meson.build
    - numeric.py
    - src/multiarray
      - ctors.c
      - textreading
        - rows.c
    - tests
      - test_multiarray.py
      - test_regression.py

```
            }
            if (not_cf_contig) {
                *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
```

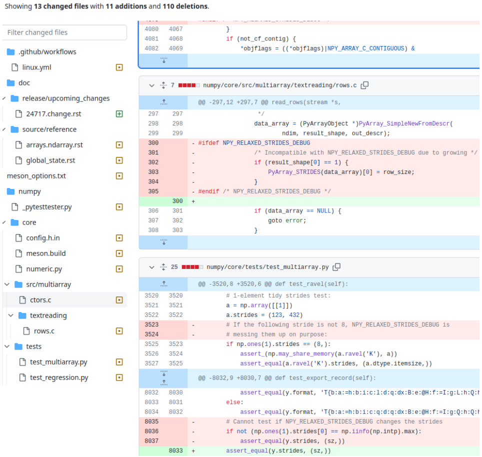
7 numpy/core/src/multiarray/textreading/rows.c

```
@@ -297,12 +297,7 @@ read_rows(stream *s,
                     */
                    data_array = (PyArrayObject *)PyArray_SimpleNewFromDescr(
                            ndim, result_shape, out_descr);
- #ifdef NPY_RELAXED_STRIDES_DEBUG
-                   /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
-                   if (result_shape[0] == 1) {
-                       PyArray_STRIDES(data_array)[0] = row_size;
-                   }
- #endif /* NPY_RELAXED_STRIDES_DEBUG */
+
                    if (data_array == NULL) {
                        goto error;
                    }
```

25 numpy/core/tests/test_multiarray.py

```
@@ -3520,8 +3520,6 @@ def test_ravel(self):
            # 1-element tidy strides test:
            a = np.array([[1]])
            a.strides = (123, 432)
-           # If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
-           # messing them up on purpose:
            if np.ones(1).strides == (8,):
                assert_(np.may_share_memory(a.ravel('K'), a))
                assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
@@ -8032,9 +8030,7 @@ def test_export_record(self):
                assert_equal(y.format, 'T{b:a:=h:b:i:c:l:d:q:dx:B:e:@H:f:=I:g:L:h:Q:
            else:
                assert_equal(y.format, 'T{b:a:=h:b:i:c:q:d:q:dx:B:e:@H:f:=I:g:Q:h:Q:
-           # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
-           if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
-               assert_equal(y.strides, (sz,))
+           assert_equal(y.strides, (sz,))
```

Summary



Showing **13 changed files** with **11 additions** and **110 deletions**.

Filter changed files

- .github/workflows
  - linux.yml
- doc
  - release/upcoming_changes
    - 24717.change.rst
  - source/reference
    - arrays.ndarray.rst
    - global_state.rst
- meson_options.txt
- numpy
  - _pytesttester.py
  - core
    - config.h.in
    - meson.build
    - numeric.py
    - src/multiarray
      - ctors.c
      - textreading
        - rows.c
    - tests
      - test_multiarray.py
      - test_regression.py

```
            }
            if (not_cf_contig) {
                *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
```

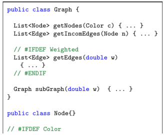
7 numpy/core/src/multiarray/textreading/rows.c

```
@@ -297,12 +297,7 @@ read_rows(stream *s,
                          */
                        data_array = (PyArrayObject *)PyArray_SimpleNewFromDescr(
                                ndim, result_shape, out_descr);
- #ifdef NPY_RELAXED_STRIDES_DEBUG
-                       /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
-                       if (result_shape[0] == 1) {
-                           PyArray_STRIDES(data_array)[0] = row_size;
-                       }
- #endif /* NPY_RELAXED_STRIDES_DEBUG */
+
                        if (data_array == NULL) {
                            goto error;
                        }
```

25 numpy/core/tests/test_multiarray.py

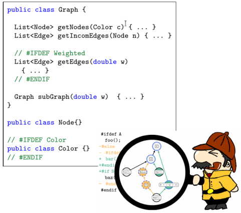
```
@@ -3520,8 +3520,6 @@ def test_ravel(self):
            # 1-element tidy strides test:
            a = np.array([[1]])
            a.strides = (123, 432)
-           # If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
-           # messing them up on purpose:
            if np.ones(1).strides == (8,):
                assert_(np.may_share_memory(a.ravel('K'), a))
                assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
@@ -8032,9 +8030,7 @@ def test_export_record(self):
                assert_equal(y.format, 'T{b:a:=h:b:i:c:l:d:q:dx:B:e:@H:f:=I:g:L:h:Q:h
            else:
                assert_equal(y.format, 'T{b:a:=h:b:i:c:q:d:q:dx:B:e:@H:f:=I:g:Q:h:Q:h
-           # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
-           if not (np.ones(1).strides[0] == np.intp).max):
-               assert_equal(y.strides, (sz,))
+           assert_equal(y.strides, (sz,))
```

Proactive Tracing



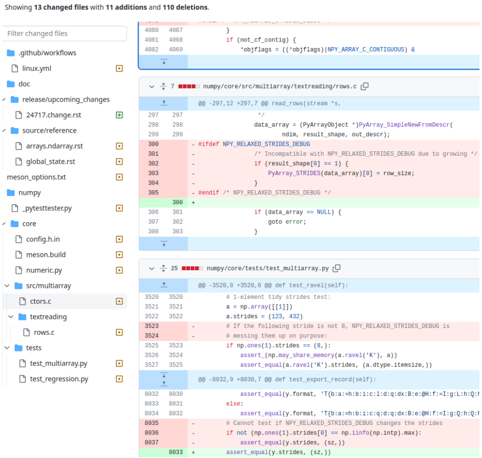
```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
```

Retroactive Tracing



```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

Summary



Showing **13 changed files** with **11 additions** and **110 deletions**.

Filter changed files

- .github/workflows
  - linux.yml
- doc
  - release/upcoming_changes
    - 24717.change.rst
  - source/reference
    - arrays.ndarray.rst
    - global_state.rst
- meson_options.txt
- numpy
  - _pytesttester.py
  - core
    - config.h.in
    - meson.build
    - numeric.py
    - src/multiarray
      - ctors.c
      - textreading
        - rows.c
    - tests
      - test_multiarray.py
      - test_regression.py

```
              }
              if (not_cf_contig) {
                  *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
```

7 numpy/core/src/multiarray/textreading/rows.c

```
@@ -297,12 +297,7 @@ read_rows(stream *s,
                       */
                      data_array = (PyArrayObject *)PyArray_SimpleNewFromDescr(
                              ndim, result_shape, out_descr);
- #ifdef NPY_RELAXED_STRIDES_DEBUG
-                     /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
-                     if (result_shape[0] == 1) {
-                         PyArray_STRIDES(data_array)[0] = row_size;
-                     }
- #endif /* NPY_RELAXED_STRIDES_DEBUG */
+
                      if (data_array == NULL) {
                          goto error;
                      }
```

25 numpy/core/tests/test_multiarray.py

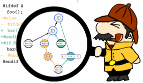
```
@@ -3520,8 +3520,6 @@ def test_ravel(self):
          # 1-element tidy strides test:
          a = np.array([[1]])
          a.strides = (123, 432)
-         # If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
-         # messing them up on purpose:
          if np.ones(1).strides == (8,):
              assert_(np.may_share_memory(a.ravel('K'), a))
              assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
@@ -8032,9 +8030,7 @@ def test_export_record(self):
              assert_equal(y.format, 'T{b:a:=h:b:i:c:l:d:q:dx:B:e:@H:f:=I:g:L:h:Q:h
          else:
              assert_equal(y.format, 'T{b:a:=h:b:i:c:q:d:q:dx:B:e:@H:f:=I:g:Q:h:Q:h
-         # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
-         if not (np.ones(1).strides[0] == np.intp).max):
-             assert_equal(y.strides, (sz,))
+         assert_equal(y.strides, (sz,))
```

Proactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
```

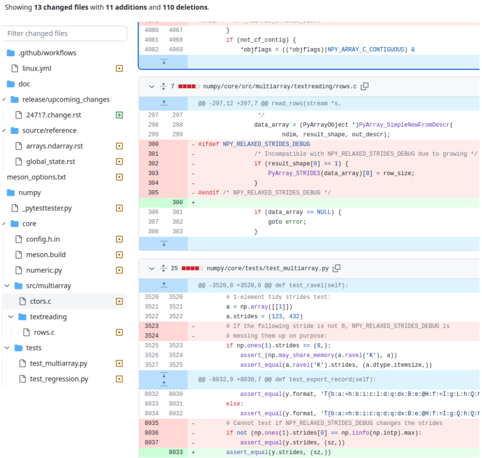
Retroactive Tracing

```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```



feature traceability comes with plenty of techniques

Summary



Showing **13 changed files** with **11 additions** and **110 deletions**.

Filter changed files

- .github/workflows
  - linux.yml
- doc
  - release/upcoming_changes
    - 24717.change.rst
  - source/reference
    - arrays.ndarray.rst
    - global_state.rst
- meson_options.txt
- numpy
  - _pytesttester.py
  - core
    - config.h.in
    - meson.build
    - numeric.py
    - src/multiarray
      - ctors.c
      - textreading
        - rows.c
    - tests
      - test_multiarray.py
      - test_regression.py

```
            }
            if (not_cf_contig) {
                *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
```

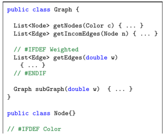
7 numpy/core/src/multiarray/textreading/rows.c

```
@@ -297,12 +297,7 @@ read_rows(stream *s,
                     */
                    data_array = (PyArrayObject *)PyArray_SimpleNewFromDescr(
                            ndim, result_shape, out_descr);
- #ifdef NPY_RELAXED_STRIDES_DEBUG
-                   /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
-                   if (result_shape[0] == 1) {
-                       PyArray_STRIDES(data_array)[0] = row_size;
-                   }
- #endif /* NPY_RELAXED_STRIDES_DEBUG */
+
                    if (data_array == NULL) {
                        goto error;
                    }
```

25 numpy/core/tests/test_multiarray.py

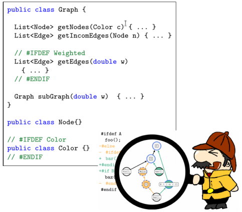
```
@@ -3520,8 +3520,6 @@ def test_ravel(self):
        # 1-element tidy strides test:
        a = np.array([[1]])
        a.strides = (123, 432)
-       # If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
-       # messing them up on purpose:
        if np.ones(1).strides == (8,):
            assert_(np.may_share_memory(a.ravel('K'), a))
            assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
@@ -8032,9 +8030,7 @@ def test_export_record(self):
            assert_equal(y.format, 'T{b:a:=h:b:i:c:l:d:q:dx:B:e:@H:f:=I:g:L:h:Q:h
        else:
            assert_equal(y.format, 'T{b:a:=h:b:i:c:q:d:q:dx:B:e:@H:f:=I:g:Q:h:Q:h
-       # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
-       if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
-           assert_equal(y.strides, (sz,))
+       assert_equal(y.strides, (sz,))
```

Proactive Tracing



```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
```

Retroactive Tracing

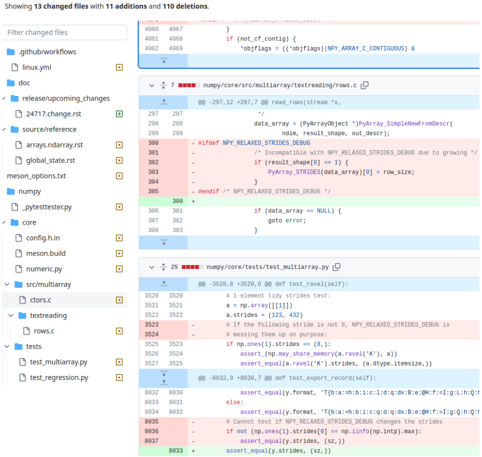


```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

feature traceability comes with plenty of techniques

no optimal solution (we examined proactive potentials)

Summary



Showing **13 changed files** with **11 additions** and **110 deletions**.

Filter changed files

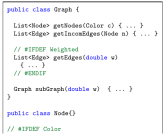
- .github/workflows
  - linux.yml
- doc
  - release/upcoming_changes
    - 24717.change.rst
  - source/reference
    - arrays.ndarray.rst
    - global_state.rst
- meson_options.txt
- numpy
  - _pytesttester.py
  - core
    - config.h.in
    - meson.build
    - numeric.py
    - src/multiarray
      - ctors.c
      - textreading
        - rows.c
    - tests
      - test_multiarray.py
      - test_regression.py

```
            }
            if (not_cf_contig) {
                *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
```

```
7 numpy/core/src/multiarray/textreading/rows.c
@@ -297,12 +297,7 @@ read_rows(stream *s,
                         */
                        data_array = (PyArrayObject *)PyArray_SimpleNewFromDescr(
                                ndim, result_shape, out_descr);
- #ifdef NPY_RELAXED_STRIDES_DEBUG
-                       /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
-                       if (result_shape[0] == 1) {
-                           PyArray_STRIDES(data_array)[0] = row_size;
-                       }
- #endif /* NPY_RELAXED_STRIDES_DEBUG */
+
                        if (data_array == NULL) {
                            goto error;
                        }
```

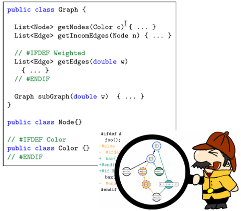
```
25 numpy/core/tests/test_multiarray.py
@@ -3520,8 +3520,6 @@ def test_ravel(self):
        # 1-element tidy strides test:
        a = np.array([[1]])
        a.strides = (123, 432)
-       # If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
-       # messing them up on purpose:
        if np.ones(1).strides == (8,):
            assert_(np.may_share_memory(a.ravel('K'), a))
            assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
@@ -8032,9 +8030,7 @@ def test_export_record(self):
            assert_equal(y.format, 'T{b:a:=h:b:i:c:l:d:q:dx:B:e:@H:f:=I:g:L:h:Q:h
        else:
            assert_equal(y.format, 'T{b:a:=h:b:i:c:q:d:q:dx:B:e:@H:f:=I:g:Q:h:Q:h
-       # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
-       if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
-           assert_equal(y.strides, (sz,))
+       assert_equal(y.strides, (sz,))
```

Proactive Tracing



```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
```

Retroactive Tracing



```
public class Graph {

  List<Node> getNodes(Color c) { ... }
  List<Edge> getIncomEdges(Node n) { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF

  Graph subGraph(double w)  { ... }
}

public class Node{}

// #IFDEF Color
public class Color {}
// #ENDIF
```

feature traceability comes with plenty of techniques

no optimal solution (we examined proactive potentials)

potentials in machine learning, necessity for maintenance over time and different artifacts

Thanks!

Feedback, Questions, ...?

greiner@imada.sdu.dk