

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.

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. **Decreasing the need for 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 ^{e,}, Rui Abreu ^{f,}, Alexander Egged ^g



^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 Mondragón, Mondragón, 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 Argo4ML 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

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. *Decreasing the need for 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^{e,} Rui Abreu^{f,} Alexander Egged^g



^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 Mondragón, Mondragón, 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 Argo4ML 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

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. *Decreasing the need for 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^{e,} Rui Abreu^{f,} Alexander Egged^g

^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 Mondragón, Mondragón, Spain

^e FEUP and INESC-ID, Porto, Portugal



manual vs automation
accuracy and reliability
maintenance over time

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 Argo4ML 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

Feature Traceability

Feature Traceability

What is it and why should we care?

How can we do better?

Today: Large Evolving Software Projects

The screenshot shows the GitHub repository page for `nasa/fprime`. The repository is public and has 1.2k forks and 9.3k stars. It includes a navigation bar with links for Code, Issues (114), Pull requests (4), Discussions, Actions, Projects (8), and Wiki. The main content area is divided into two columns. The left column shows a list of files and folders, including `.github`, `Autocoders`, `CFDP`, `Drv`, `Fpp`, `Fw`, `Os`, `RPI`, `Ref`, `STest`, `Svc`, `Utils`, `ci`, `cmake`, `config`, `docs`, `.clang-tidy`, `.dockerignore`, and `.gitignore`. The right column shows the 'About' section, which includes a description of the project as a flight software and embedded systems framework, a list of tags such as `raspberry-pi`, `components`, `real-time`, `framework`, `embedded`, `cpp`, `nasa`, `embedded-systems`, `flight`, `spaceflight`, `object-oriented-programming`, `flight-software`, and `fprime`, and a list of releases, including the latest release `v3.1.1` from August 19, 2022. The contributors section shows 116 contributors.

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 - About

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

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

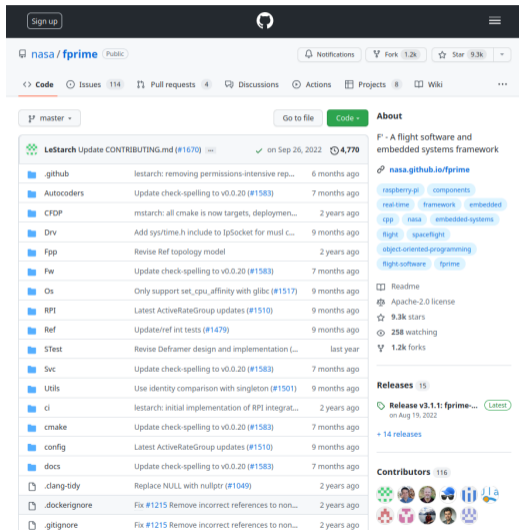
Readme
Apache-2.0 license
9.3k stars
258 watching
1.2k forks

Releases 15

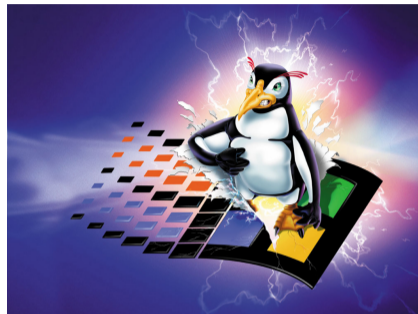
Release v3.1.1: fprime... (Latest)
on Aug 19, 2022
+ 14 releases

Contributors 116

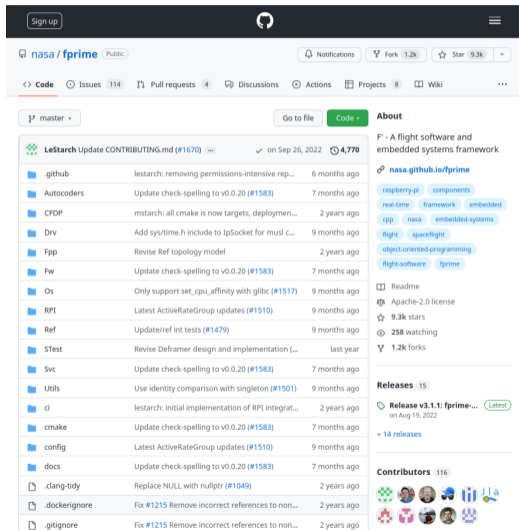
Today: Large Evolving Software Projects



The screenshot shows the GitHub repository page for `nasa/fprime`. The repository is public and has 1.2k forks and 9.3k stars. It is categorized under Code, Issues (114), Pull requests (4), Discussions, Actions, Projects (8), and Wiki. The repository is currently on the `master` branch. The main content area displays a list of recent updates, including contributions from `LeStarch` and various subdirectories like `.github`, `Autocoders`, `CFDP`, `Drv`, `Fpp`, `Fw`, `Os`, `RPI`, `Ref`, `STest`, `Svc`, `Utils`, `ci`, `cmake`, `config`, `docs`, `.clang-tidy`, `.dockerignore`, and `.gitignore`. The right sidebar provides an overview of the project, including a description: "F - A flight software and embedded systems framework", a list of tags such as `raspberry-pi`, `components`, `real-time`, `framework`, `embedded`, `cpp`, `nasa`, `embedded-systems`, `flight`, `spaceflight`, `object-oriented-programming`, `flight-software`, and `fprime`, and a list of releases, with the latest being `Release v3.1.1: fprime...` from August 19, 2022.



Today: Large Evolving Software Projects



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 - About

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

F - A flight software and embedded systems framework

nasa.github.io/fprime

tags: 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

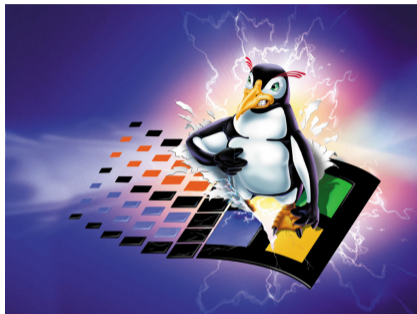
1.2k forks

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

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 - About

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

F - A flight software and embedded systems framework

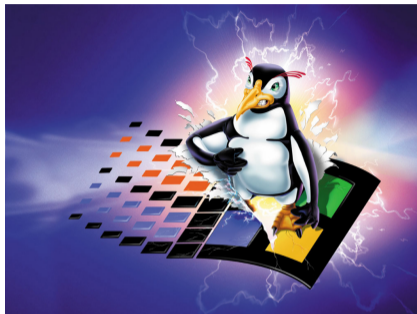
nasa.github.io/fprime

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

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 - About

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

Repository	Description	Time Ago
.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, deployem...	2 years ago
Drv	Add systime.h include to IpSocket for mud c...	9 months ago
Fpp	Revise Ref topology model	2 years ago
Fiw	Update check-spelling to v0.0.20 (#1583)	7 months ago
Os	Only support set_cpu_affinity with glibc (#1517)	9 months ago
RP1	Latest ActiveRateGroup updates (#1510)	9 months ago
Ref	Update/hel 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 RP1 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 (#1048)	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

F - A flight software and embedded systems framework

nasa.github.io/fprime

raspberrypi components
real-time framework embedded
c++ nasa embedded-systems
flight specific
object-oriented programming
flight software fprime

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

Releases 15

Release v2.1.1: fprime-... (Latest)
on Aug 19, 2022
+ 14 releases

Contributors 118

Problem Statement: Large Evolving Highly-Configurable Software Projects

FPrime

Sign up

nasa/fprime (Public)

Code Issues 114 Pull requests 4 Discussions 0 Actions 0 Projects 0 Wiki 0

master - Go to file Code - About

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

- lestartch: removing permissions-intensive rep... 6 months ago
- Autocoders Update check-spelling to v0.0.20 (#1583) 7 months ago
- CFDP mstarck: all cmake is now targets, deployment... 2 years ago
- Drv Add systime.h include to lpsocket for mud 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/hel 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
- Ulib Use identity comparison with singleton (#1501) 9 months ago
- ci lestartch: 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 (#1048) 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

Releases 15
Release v2.1.1: fprime... on Aug 19, 2022 + 14 releases

Contributors 118

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

```
4089 4087     }
4081 4068     if (not_cf.contig) {
4082 4069         *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
...
numpy/core/src/multiarray/textreading/rows.c
00 -297,12 +297,7 @@ read_rows(stream *s,
297 297     /*
298 298         data_array = (PyObject *)PyArray_SimpleNewFromDescr(
299 299             ndim, result_shape, out_descr);
300 300
301 300 - #ifndef NPY_RELAXED_STRIDES_DEBUG
302 301 - /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
303 302 -     if (result_shape[0] == 1) {
304 303 -         PyArray_STRIDES(data_array)[0] = row_size;
305 304 -     }
306 305 - #endif /* NPY_RELAXED_STRIDES_DEBUG */
...
306 306 +
307 306 +     if (data_array == NULL) {
308 307 +         goto error;
309 308 +     }
...
numpy/core/tests/test_multiarray.py
00 -3528,8 +3529,6 @@ def test_ravel(self):
3529 3528     # 1-element tidy strides test:
3530 3529     a = np.array([[1]])
3531 3528     a.strides = (123, 432)
3532 3529
3533 3529     # If the following stride is not 0, NPY_RELAXED_STRIDES_DEBUG is
3534 3529     # messing them up on purpose:
3535 3529     if np.ones(1).strides == (8,):
3536 3529         assert(np._may_share_memory(a.ravel('K'), a))
3537 3529         assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
...
00 -8032,9 +8030,7 @@ def test_export_record(self):
8032 8030     assert_equal(y.format, 'T{b:a:h:b1:c:1:d:q:dx:B:e@M:f:l:g:h:Q:l:Q:}
8033 8031     else:
8034 8032         assert_equal(y.format, 'T{b:a:h:b1:c:q:d:q:dx:B:e@M:f:l:g:h:Q:h:Q:}
8035 8035 -     # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
8036 8036 -     if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
8037 8037         assert_equal(y.strides, (sz,))
...
8033 8033 +     assert_equal(y.strides, (sz,))
```

Problem Statement: Large Evolving Highly-Configurable Software Projects

FPrime

Sign up

nasa / fprime (Public)

Code Issues 314 Pull requests 4 Discussions 0 Actions 0 Projects 0 Wiki 0

master -

Go to file Code - About

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, deployment... 2 years ago
- Drv Add systime.h include to IpSocket for mud c... 9 months ago
- Fpp Revise Ref topology model 2 years ago
- Fix Update check-spelling to v0.0.20 (#1583) 7 months ago
- Os Only support set_cpu_affinity with glibc (#1517) 9 months ago
- RPT Latest ActiveRateGroup updates (#1510) 9 months ago
- Ref UpdateRef int tests (#1479) 9 months ago
- Stest Revise Deframer design and implem... 9 months ago
- Svc Update check-spelling to v0.0.20 (#1510) 9 months ago
- Utils Use identity comparison with singles... 9 months ago
- ci lestarch: initial implementation of RP... 9 months ago
- cmake Update check-spelling to v0.0.20 (#1510) 9 months ago
- config Latest ActiveRateGroup updates (#1510) 9 months ago
- docs Update check-spelling to v0.0.20 (#1510) 9 months ago
- clang-tidy Replace NULL with nullptr (#1048) 9 months ago
- dockerignore Fix #1215 Remove incorrect referenc... 9 months ago
- gitignore Fix #1215 Remove incorrect referenc... 9 months ago

F - A flight software and embedded systems framework

nasa.github.io/fprime

- components
- framework
- embedded
- nasa
- embedded-systems
- flight
- specifics
- object-oriented-programming
- flight-software
- system

Readme

- Apache-2.0 license
- 9.3k stars
- 258 watching

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

```
4089 4087     }
4081 4088     if (not_cf.contig) {
4082 4089         *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
...
numpy/core/src/multiarray/textreading/rows.c
00 -297,12 +297,7 @@ read_rows(stream *s,
297 297     /*
298 298         data_array = (PyObject *)PyArray_SimpleNewFromDescr(
299 299         ndim, result_shape, out_descr);
300 300
301 300 - #ifndef NPY_RELAXED_STRIDES_DEBUG
302 301 - /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
303 302 -     if (result_shape[0] == 1) {
304 303 -         PyArray_STRIDES(data_array)[0] = row_size;
305 304 -     }
306 305 - #endif /* NPY_RELAXED_STRIDES_DEBUG */
...
numpy/core/tests/test_multiarray.py
529,6 @@ def test_ravel(self):
529,6 def test_ravel(self):
530 529     lenient tidy strides test:
531 530     p.array([[1]])
532 531     lides = (123, 432)
533 532     the following stride is not 0, NPY_RELAXED_STRIDES_DEBUG is
534 533     sing them up on purpose:
535 534     -.ones(1).strides == (8,):
536 535     sassert(np._may_share_memory(a.ravel('K'), a))
537 536     sassert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
...
5303,7 @@ def test_export_record(self):
5303,7 def test_export_record(self):
5304 5303     sassert_equal(y.format, 'T{b:a:h:b1:c:1:d:q:dx:8:e@M:f:l:g:Q:h:Q:f
...
5304 5303     sassert_equal(y.format, 'T{b:a:h:b1:c:1:d:q:dx:8:e@M:f:l:g:Q:h:Q:f
5305 5304     # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
5306 5305     if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
5307 5306         sassert_equal(y.strides, (sz,))
...
5303 5303 +     sassert_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

Sign up

nasa / fprime (Public)

Code Issues 114 Pull requests 4 Discussions 0 Actions 0 Projects 0 Wiki 0

master - Go to file Code - About

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

- 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 system.h include to IpSocket for mud c... 9 months ago
- Fpp Revise Ref topology model 2 years ago
- Fix Update check-spelling to v0.0.20 (#1583) 7 months ago
- Os Only support set_cpu_affinity with glibc (#1517) 9 months ago
- RPT Latest ActiveRateGroup updates (#1510) 9 months ago
- Ref UpdateRef int tests (#1479) 9 months ago
- Stest Revise Deframer design and implem... 9 months ago
- Svc Update check-spelling to v0.0.20 (#1510) 9 months ago
- Utils Use identity comparison with singles... 9 months ago
- ci lestarch: initial implementation of RP... 9 months ago
- cmake Update check-spelling to v0.0.20 (#1510) 9 months ago
- config Latest ActiveRateGroup updates (#1510) 9 months ago
- docs Update check-spelling to v0.0.20 (#1510) 9 months ago
- clang-tidy Replace NULL with nullptr (#1048) 9 months ago
- dockerignore Fix #1215 Remove incorrect referenc... 9 months ago
- gitignore Fix #1215 Remove incorrect referenc... 9 months ago

F - A flight software and embedded systems framework

github.com/nasa/fprime

flight software
system-oriented programming
flight software system

Readme
Apache-2.0 license
9.3k stars
258 watching

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
- pytesttester.py
- core
- config.h.in
- meson.build
- numeric.py

```
4889 4887     }
4881 4868     if (not_cf.contig) {
4882 4869         *objflags = ((*objflags)|NPY_ARRAY_C_CONTIGUOUS) &
...
numpy/core/src/multiarray/textreading/rows.c
00 -297,12 +297,7 @@ read_rows(stream *s,
297 297     /*
298 298         data_array = (PyObject **)PyObject_SimpleNewFromDescr(
299 299             ndia, result_shape, out_descr);
300
301     - #ifndef NPY_RELAXED_STRIDES_DEBUG
302     - /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
303     - if (result_shape[0] == 1) {
304     -     PyArray_STRIDES(data_array)[0] = row_size;
305     - }
306     - #endif /* NPY_RELAXED_STRIDES_DEBUG */
307
308 +
309 +     if (data_array == NULL) {
310 +         goto error;
311 +     }
...
numpy/core/tests/test_multiarray.py
529,6 @@ def test_ravel(self):
529,6 def test_ravel(self):
530,6     lement tidy strides test:
531,6     p_array([[1]])
532,6     ldes = (123, 432)
533,6     the following stride is not 0, NPY_RELAXED_STRIDES_DEBUG is
534,6     sing them up on purpose:
535,6     -ones(1).strides == (8,):
536,6     ssert(np._may_share_memory(a.ravel('K'), a))
537,6     ssert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
538,6 @@ def test_export_record(self):
539,6     ssert_equal(y.format, 'T{b:a;#b:1:c:1:d;q:dx:B:e@M:f:I:g:h:Q:
...
540,6     ssert_equal(y.format, 'T{b:a;#b:1:c:1:d;q:dx:B:e@M:f:I:g:h:Q:h:Q:
541,6
542,6
543,6     # Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
544,6     if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
545,6         ssert_equal(y.strides, (sz,))
546,6
547,6     ssert_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

```
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

```
1 // V1: {G, E, W}  
2 // FEAT: Graph  
3 interface Graph {  
4     List<Node> nodes();  
5     // FEAT: E  
6     List<Edge> edges();  
7     List<Node> nodes(double w);  
8 }
```

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

```
1 /// V1: {G, E, W}  
2 // FEAT: Graph  
3 interface Graph {  
4     List<Node> nodes();  
5     // FEAT: E  
6     List<Edge> edges();  
7     List<Node> nodes(double w);  
8 }
```

```
1 /// V2: {G, E, C}  
2 interface Graph {  
3     List<Node> nodes();  
4     // FEAT: C  
5     List<Node> nodes(Color c);  
6     List<Edge> edges();  
7     Graph subGraph(Color c);  
8 }
```

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

```
1 /// V1: {G, E, W}  
2 // FEAT: Graph  
3 interface Graph {  
4     List<Node> nodes();  
5     // FEAT: E  
6     List<Edge> edges();  
7     List<Node> nodes(double w);  
8 }
```

```
1 /// V2: {G, E, C}  
2 interface Graph {  
3     List<Node> nodes();  
4     // FEAT: C  
5     List<Node> nodes(Color c);  
6     List<Edge> edges();  
7     Graph subGraph(Color c);  
8 }
```

```
1 /// V3: {G, E, D}  
2 interface Graph {  
3     List<Node> nodes();  
4     List<Edge> edges();  
5     List<Edge> incomEdges(Node n);  
6     // FEAT: C  
7     Graph subGraph(Color c);  
8 }
```

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



through

commit messages

development on branches

```
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



through

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

```
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



through

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

```
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
- 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

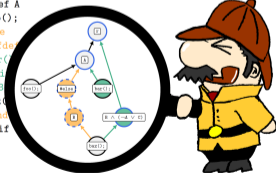
⇒ often neglected
⇒ missed opportunity

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
```

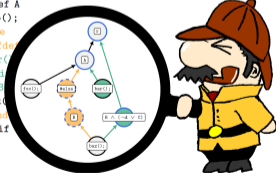
```
#ifdef A  
foo();  
-#else  
- #ifdef  
+ bar();  
+#endif  
+#if B  
baz();  
- #endif  
#endif
```



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  
- #ifdef  
+ bar();  
+#endif  
+#if B  
baz();  
- #endif  
#endif
```



through

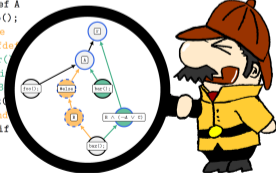
manual code inspection

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
```

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



through

manual code inspection

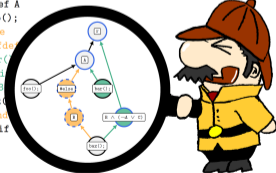
dynamic techniques (which compare executions)

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
```

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



through

manual code inspection

dynamic techniques (which compare executions)

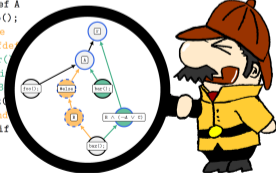
static techniques (e.g., slicing, clone detection, comparison of variants)

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
```

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



through

manual code inspection

dynamic techniques (which compare executions)

static techniques (e.g., slicing, clone detection, comparison of variants)

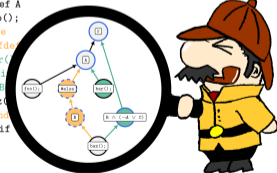
combinations thereof

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
```

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

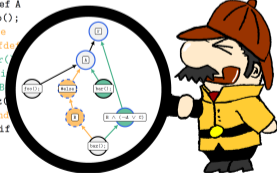
while automated, reliability?

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  
- #ifdef  
+ bar();  
+#endif  
+#if B  
baz();  
- #endif  
#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?

How can retroactive feature tracing benefit from proactive traces?

Results from experimenting with comparison-based feature location

Comparison-Based Feature Location

```
1 /// V1: {G, E, W}
2 // FEAT: Graph
3 interface Graph {
4   List<Node> nodes();
5   // FEAT: E
6   List<Edge> edges();
7   List<Node> nodes(double w);
8 }
```

```
1 /// V3: {G, E, D}
2 interface Graph {
3   List<Node> nodes();
4   List<Edge> edges();
5   List<Edge> incomEdges(Node n);
6   // FEAT: C
7   Graph subGraph(Color c);
8 }
```

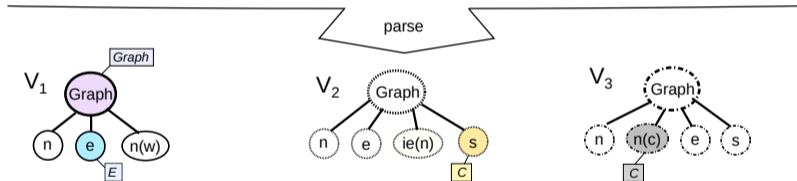
```
1 /// V2: {G, E, C}
2 interface Graph {
3   List<Node> nodes();
4   // FEAT: C
5   List<Node> nodes(Color c);
6   List<Edge> edges();
7   Graph subGraph(Color c);
8 }
```

Comparison-Based Feature Location

```
1 /// V1: {G, E, W}
2 // FEAT: Graph
3 interface Graph {
4   List<Node> nodes();
5   // FEAT: E
6   List<Edge> edges();
7   List<Node> nodes(double w);
8 }
```

```
1 /// V3: {G, E, D}
2 interface Graph {
3   List<Node> nodes();
4   List<Edge> edges();
5   List<Edge> incomEdges(Node n);
6   // FEAT: C
7   Graph subGraph(Color c);
8 }
```

```
1 /// V2: {G, E, C}
2 interface Graph {
3   List<Node> nodes();
4   // FEAT: C
5   List<Node> nodes(Color c);
6   List<Edge> edges();
7   Graph subGraph(Color c);
8 }
```

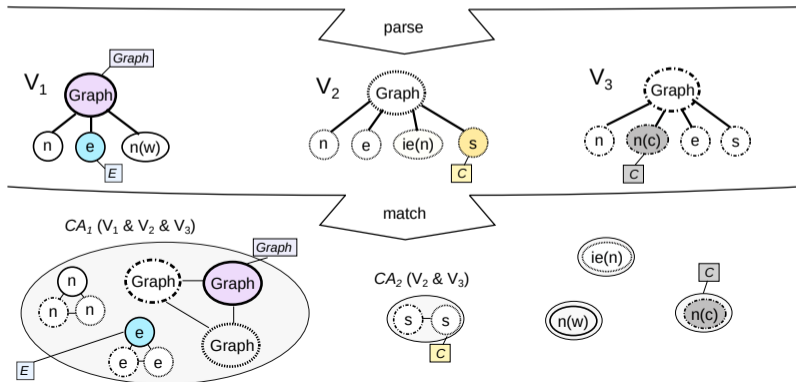


Comparison-Based Feature Location

```
1 /// V1: {G, E, W}
2 // FEAT: Graph
3 interface Graph {
4   List<Node> nodes();
5   // FEAT: E
6   List<Edge> edges();
7   List<Node> nodes(double w);
8 }
```

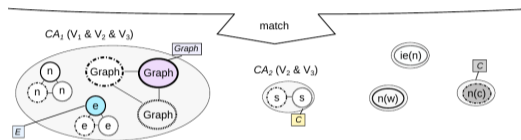
```
1 /// V3: {G, E, D}
2 interface Graph {
3   List<Node> nodes();
4   List<Edge> edges();
5   List<Edge> incomEdges(Node n);
6   // FEAT: C
7   Graph subGraph(Color c);
8 }
```

```
1 /// V2: {G, E, C}
2 interface Graph {
3   List<Node> nodes();
4   // FEAT: C
5   List<Node> nodes(Color c);
6   List<Edge> edges();
7   Graph subGraph(Color c);
8 }
```



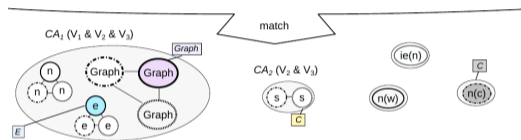
Comparison-Based Feature Location

annotation per node (based on features in configurations)



Comparison-Based Feature Location

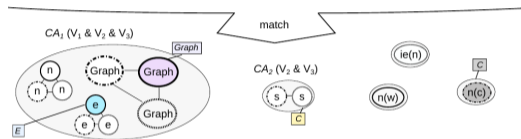
annotation per node (based on features in configurations)



Feature Trace

Comparison-Based Feature Location

annotation per node (based on features in configurations)



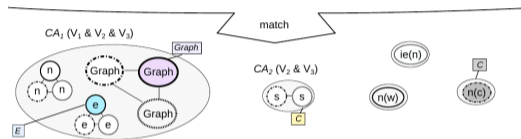
Feature Trace

Graph \wedge 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)

Controlled Experiment

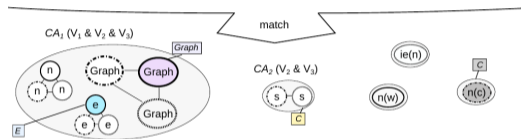


Feature Trace

Graph \wedge 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)



Controlled Experiment

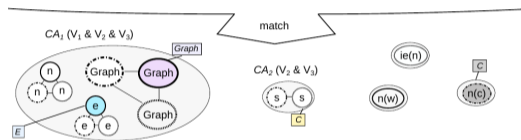
propagate 'reliable annotation'
whenever possible (not contradicting
among one set)
measure quantitatively

Feature Trace

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

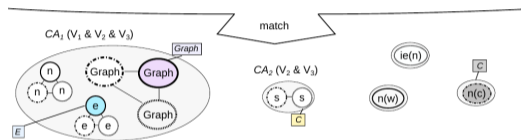
Controlled Experiment

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 \wedge 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?

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

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

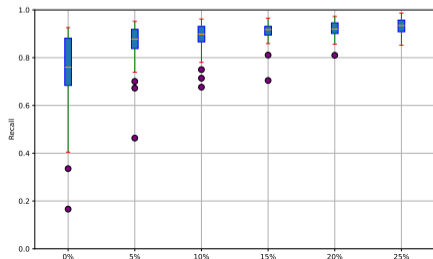
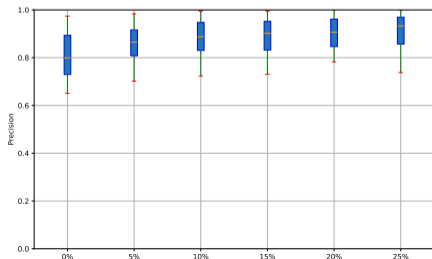
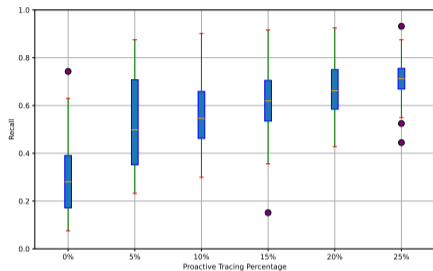
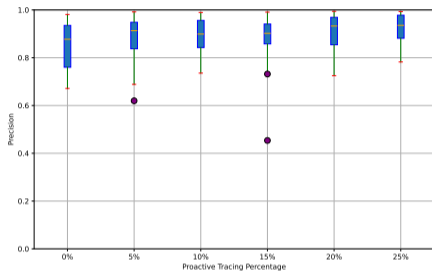
Measure:

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

Compute: precision, recall, F1-Score

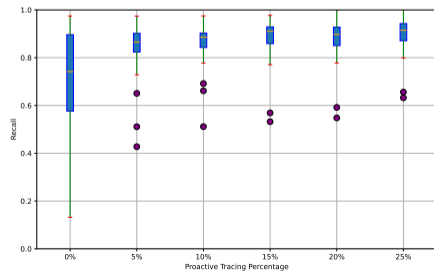
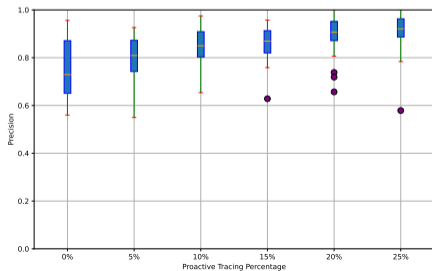
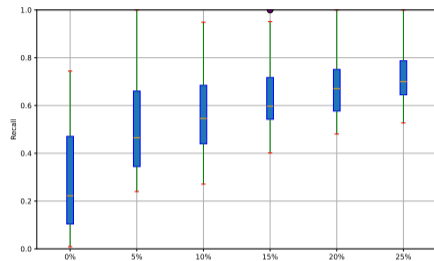
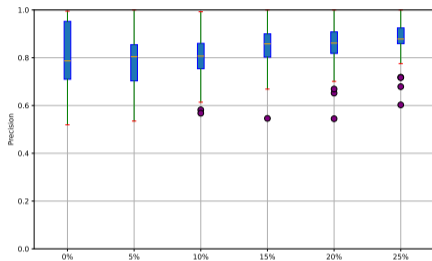
Results : Busybox (precision & recall _(row), 3 and 7 variants _(columns))

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
```

```
4080 4067      }
4081 4068      if (not_cf_contig) {
4082 4069          *objFlags = ((*objFlags)|NPY_ARRAY_C_CONTIGUOUS) &
...
numpy/core/src/multiarray/textreading/rows.c
297 297      /*
298 298      data_array = (PyObject *)PyArray_SimpleNewFromDescr(
299 299      HDM, result_shape, out_descr);
300 300      - #ifdef NPY_RELAXED_STRIDES_DEBUG
301 301      - /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
302 302      - if (result_shape[0] == 1) {
303 303      -     PyArray_STRIDES(data_array)[0] = row_size;
304 304      - }
305 305      - #endif /* NPY_RELAXED_STRIDES_DEBUG */
...
numpy/core/tests/test_multiarray.py
3520 3520      @ 1-element tiny strides test:
3521 3521      a = np.array([[1]])
3522 3522      a.strides = (123, 432)
3523 3523      - @ If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
3524 3524      - @ messing them up on purpose:
3525 3525      if np.ones(1).strides == (8,):
3526 3526      assert(np._may_share_memory(a.ravel('K'), a))
3527 3527      assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
...
numpy/core/tests/test_multiarray.py
8832 8830      assert_equal(y.format, "T(b:~h:~b:~i:~d:~d:~s:~e:~M:f:~i:~g:~h:~Q:~t)
8833 8831      else:
8834 8832      assert_equal(y.format, "T(b:~h:~b:~i:~d:~d:~s:~e:~M:f:~i:~g:~h:~Q:~t)
8835 8835      - @ Cannot test if NPY_RELAXED_STRIDES_DEBUG changes the strides
8836 8836      - if not (np.ones(1).strides[0] == np.infty[ep.intp].max):
8837 8837      -     assert_equal(y.strides, (sz,))
8838 8838      +     assert_equal(y.strides, (sz,))
```

Proactive Tracing



```
public class Graph {
    List<Node> getNodes(Color c) { ... }
    List<Edge> getIncoEdges(Node a) { ... }

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

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

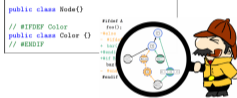
public class Node()
// #IFDEF Color
```

Retroactive Tracing

```
public class Graph {
    List<Node> getNodes(Color c){ ... }
    List<Edge> getIncoEdges(Node n) { ... }

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

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



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
```

```
4080 4067      }
4081 4068      if (not_cf_contig) {
4082 4069          *objFlags = ((*objFlags) | NPY_ARRAY_C_CONTIGUOUS) &
...
numpy/core/src/multiarray/textreading/rows.c
297 297      /*
298 298      data_array = (PyObject *)PyArray_SimpleNewFromDescr(
299 299      HERR, result_shape, out_descr);
300 300      - #ifdef NPY_RELAXED_STRIDES_DEBUG
301 301      - /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
302 302      - if (result_shape[0] == 1) {
303 303      -     PyArray_STRIDES(data_array)[0] = row_size;
304 304      - }
305 305      - #endif /* NPY_RELAXED_STRIDES_DEBUG */
...
numpy/core/tests/test_multiarray.py
3520 3520      @ 1-element tiny strides test:
3521 3521      a = np.array([[1]])
3522 3522      a.strides = (123, 432)
3523 3523      - @ If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
3524 3524      - @ messing them up on purpose:
3525 3525      if np.ones(1).strides == (8,):
3526 3526      assert(np.may_share_memory(a.ravel('K'), a))
3527 3527      assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
...
8832 8830      assert_equal(y.format, "T(b:a:hb:hb:ic:ld:qd:db:be:@#f:ic:ig:lh:Q?
8833 8831      else:
8834 8832      assert_equal(y.format, "T(b:a:hb:hb:ic:qd:dx:db:be:@#f:ic:ig:qh:Q?
8835 8835      - @ Cannot test IF NPY_RELAXED_STRIDES_DEBUG changes the strides
8836 8836      - if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
8837 8837      -     assert_equal(y.strides, (sz,))
8838 8838      +     assert_equal(y.strides, (sz,))
```

Proactive Tracing



```
public class Graph {
    List<Node> getNodes(Color c) { ... }
    List<Edge> getIncoEdges(Node a) { ... }

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

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

public class Node {
    // #IFDEF Color
    public class Color {
        // #IFDEF A
        #endif
        // #IFDEF B
        #endif
        // #IFDEF C
        #endif
        // #IFDEF D
        #endif
        // #IFDEF E
        #endif
        // #IFDEF F
        #endif
        // #IFDEF G
        #endif
        // #IFDEF H
        #endif
        // #IFDEF I
        #endif
        // #IFDEF J
        #endif
        // #IFDEF K
        #endif
        // #IFDEF L
        #endif
        // #IFDEF M
        #endif
        // #IFDEF N
        #endif
        // #IFDEF O
        #endif
        // #IFDEF P
        #endif
        // #IFDEF Q
        #endif
        // #IFDEF R
        #endif
        // #IFDEF S
        #endif
        // #IFDEF T
        #endif
        // #IFDEF U
        #endif
        // #IFDEF V
        #endif
        // #IFDEF W
        #endif
        // #IFDEF X
        #endif
        // #IFDEF Y
        #endif
        // #IFDEF Z
        #endif
    }
}

public class Node {
    // #IFDEF Color
```

feature traceability comes with plenty of techniques

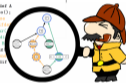
Retroactive Tracing

```
public class Graph {
    List<Node> getNodes(Color c){ ... }
    List<Edge> getIncoEdges(Node n) { ... }

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

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

public class Node {
    // #IFDEF Color
    public class Color {
        // #IFDEF A
        #endif
        // #IFDEF B
        #endif
        // #IFDEF C
        #endif
        // #IFDEF D
        #endif
        // #IFDEF E
        #endif
        // #IFDEF F
        #endif
        // #IFDEF G
        #endif
        // #IFDEF H
        #endif
        // #IFDEF I
        #endif
        // #IFDEF J
        #endif
        // #IFDEF K
        #endif
        // #IFDEF L
        #endif
        // #IFDEF M
        #endif
        // #IFDEF N
        #endif
        // #IFDEF O
        #endif
        // #IFDEF P
        #endif
        // #IFDEF Q
        #endif
        // #IFDEF R
        #endif
        // #IFDEF S
        #endif
        // #IFDEF T
        #endif
        // #IFDEF U
        #endif
        // #IFDEF V
        #endif
        // #IFDEF W
        #endif
        // #IFDEF X
        #endif
        // #IFDEF Y
        #endif
        // #IFDEF Z
        #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
```

```
4080 4067      }
4081 4068      if (not_cf_contig) {
4082 4069          *objFlags = ((*objFlags)|NPY_ARRAY_C_CONTIGUOUS) &
...
numpy/core/src/multiarray/textreading/rows.c
...
297 297      /*
298 298      data_array = (PyObject *)PyArray_SimpleNewFromDescr(
299 299      HEM, result_shape, out_descr);
300 300      - #ifdef NPY_RELAXED_STRIDES_DEBUG
301 301      /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
302 302      - if (result_shape[0] == 1) {
303 303      -     PyArray_STRIDES(data_array)[0] = row_size;
304 304      - }
305 305      - #endif /* NPY_RELAXED_STRIDES_DEBUG */
...
306 301      if (data_array == NULL) {
307 302          goto error;
308 303      }
...
numpy/core/tests/test_multiarray.py
...
3520 3520      @ 1-element tidy strides test:
3521 3521      a = np.array([[1]])
3522 3522      a.strides = (123, 432)
3523 3523      - @ If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
3524 3524      - @ messing them up on purpose:
3525 3523      if np.ones(1).strides == (8,):
3526 3524      assert(np._may_share_memory(a.ravel('K'), a))
3527 3525      assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
...
8832 8830      assert_equal(y.format, "T(b:~h:~i:c:~l:d:~x:~B:e:@H:f:~i:g:~h:~Q:t)
8833 8831      else:
8834 8832      assert_equal(y.format, "T(b:~h:~i:c:~l:d:~x:~B:e:@H:f:~i:g:~h:~Q:t)
8835 8835      - @ Cannot test IF NPY_RELAXED_STRIDES_DEBUG changes the strides
8836 8836      - if not (np.ones(1).strides[0] == np.iinfo(np.intp).max):
8837 8837      -     assert_equal(y.strides, (sz,))
8838 8833      +     assert_equal(y.strides, (sz,))
```

Proactive Tracing



```
public class Graph {
    List<Node> getNodes(Color c) { ... }
    List<Edge> getInEdges(Node n) { ... }

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

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

public class Node {
    List<Node> getNodes(Color c) { ... }
    List<Edge> getInEdges(Node n) { ... }

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

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

public class Node {
    // #IFDEF Color
```

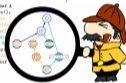
Retroactive Tracing

```
public class Graph {
    List<Node> getNodes(Color c) { ... }
    List<Edge> getInEdges(Node n) { ... }

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

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

public class Node {
    // #IFDEF Color
```



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

```
4080 4087      }
4081 4088      if (not_cf_contig) {
4082 4089          *objFlags = ((*objFlags)|NPY_ARRAY_C_CONTIGUOUS) &
...
numpy/core/src/multiarray/textreading/rows.c
...
297 297      /*
298 298          data_array = (PyObject *)PyArray_SimpleNewFromDescr(
299 299              HEM, result_shape, out_descr);
300 300          /* #ifndef NPY_RELAXED_STRIDES_DEBUG
301 301          /* Incompatible with NPY_RELAXED_STRIDES_DEBUG due to growing */
302 302          if (result_shape[0] == 1) {
303 303              PyArray_STRIDES(data_array)[0] = row_size;
304 304          }
305 305          #endif /* NPY_RELAXED_STRIDES_DEBUG */
...
numpy/core/tests/test_multiarray.py
...
3520 3520      # 1-element tidy strides test:
3521 3521      a = np.array([[1]])
3522 3522      a.strides = (123, 432)
3523 3523      # If the following stride is not 8, NPY_RELAXED_STRIDES_DEBUG is
3524 3524      # messing them up on purpose:
3525 3525      if np.ones(1).strides == (8,):
3526 3526          assert(np._may_share_memory(a.ravel('K'), a))
3527 3527          assert_equal(a.ravel('K').strides, (a.dtype.itemsize,))
...
8832 8832      assert_equal(y.format, "(b:ashb:ic:ld:q:dx:B:e:#f:ic:ig:lh:Q:t)
8833 8833      else:
8834 8834          assert_equal(y.format, "(b:ashb:ic:ld:q:dx:B:e:#f:ic:ig:lh:Q:t)
...
8835 8835      # Cannot test IF NPY_RELAXED_STRIDES_DEBUG changes the strides
8836 8836      if not (np.ones(1).strides[0] == np.intp[0].itemsize):
8837 8837          assert_equal(y.strides, (sz,))
8838 8838          assert_equal(y.strides, (sz,))
```

Proactive Tracing



```
public class Graph {
    List<Node> getNodes(Color c) { ... }
    List<Edge> getInEdges(Node n) { ... }
    // #IFDEF Weighted
    List<Edge> getEdges(double w)
    { ... }
    // #ENDIF
    Graph subGraph(double w) { ... }
}

public class Node {
    // #IFDEF Color
    Color c;
    // #ENDIF
}

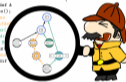
public class Color {
    // #IFDEF A
    int a;
    // #ENDIF
}

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

Retroactive Tracing

```
public class Graph {
    List<Node> getNodes(Color c) { ... }
    List<Edge> getInEdges(Node n) { ... }
    // #IFDEF Weighted
    List<Edge> getEdges(double w)
    { ... }
    // #ENDIF
    Graph subGraph(double w) { ... }
}

public class Node {
    // #IFDEF Color
    Color c;
    // #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