Mutation Testing

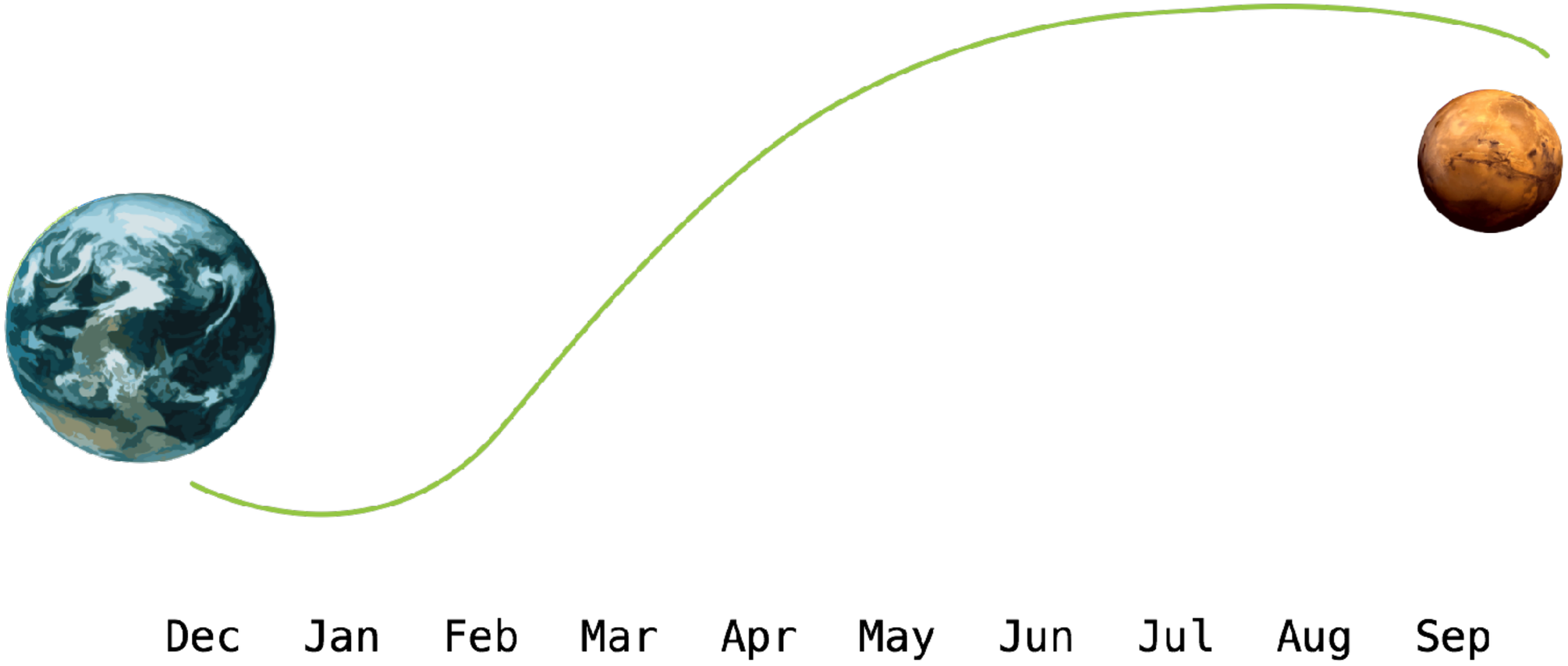
How good your tests are

2017

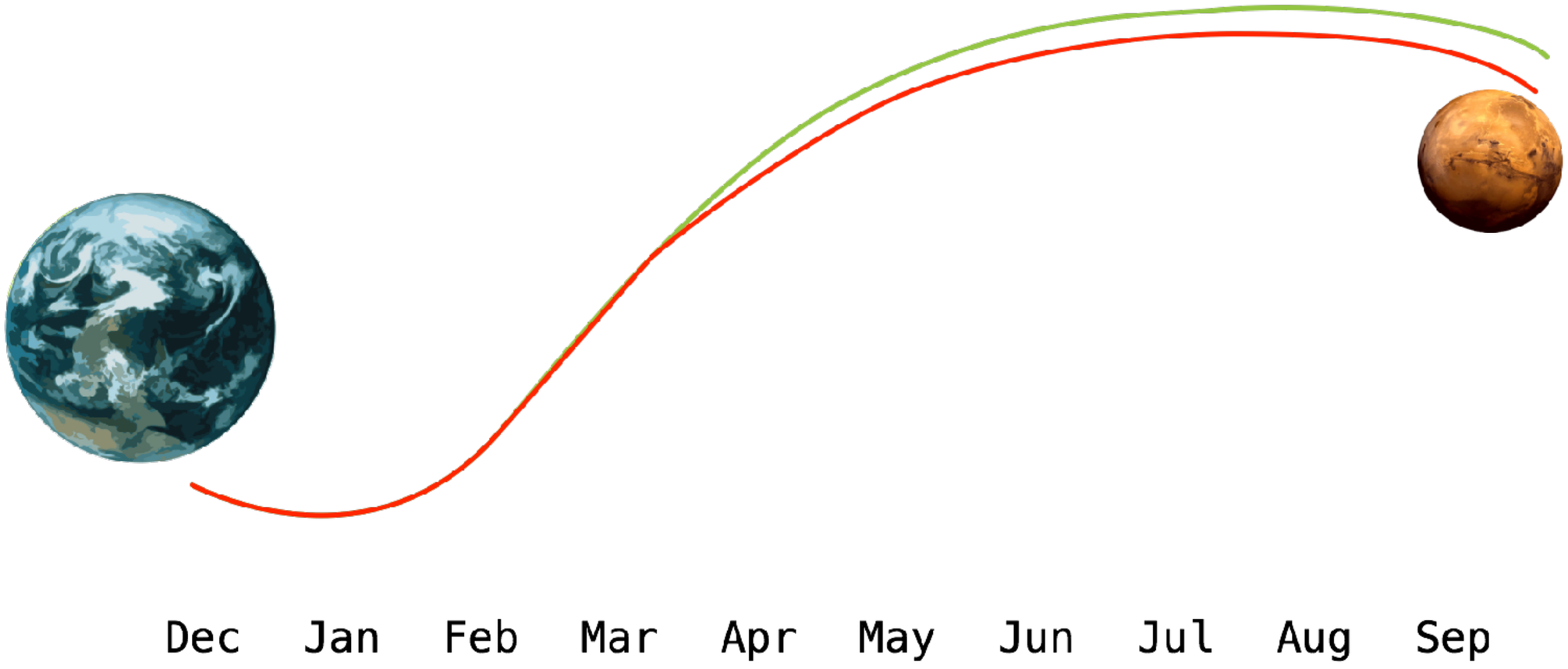
whoami

- iOS Developer by day
- compiler hacker by night
- https://twitter.com/1101_debian
- https://lowlevelbits.org
- https://systemundertest.org

Mars Climate Orbiter



Mars Climate Orbiter

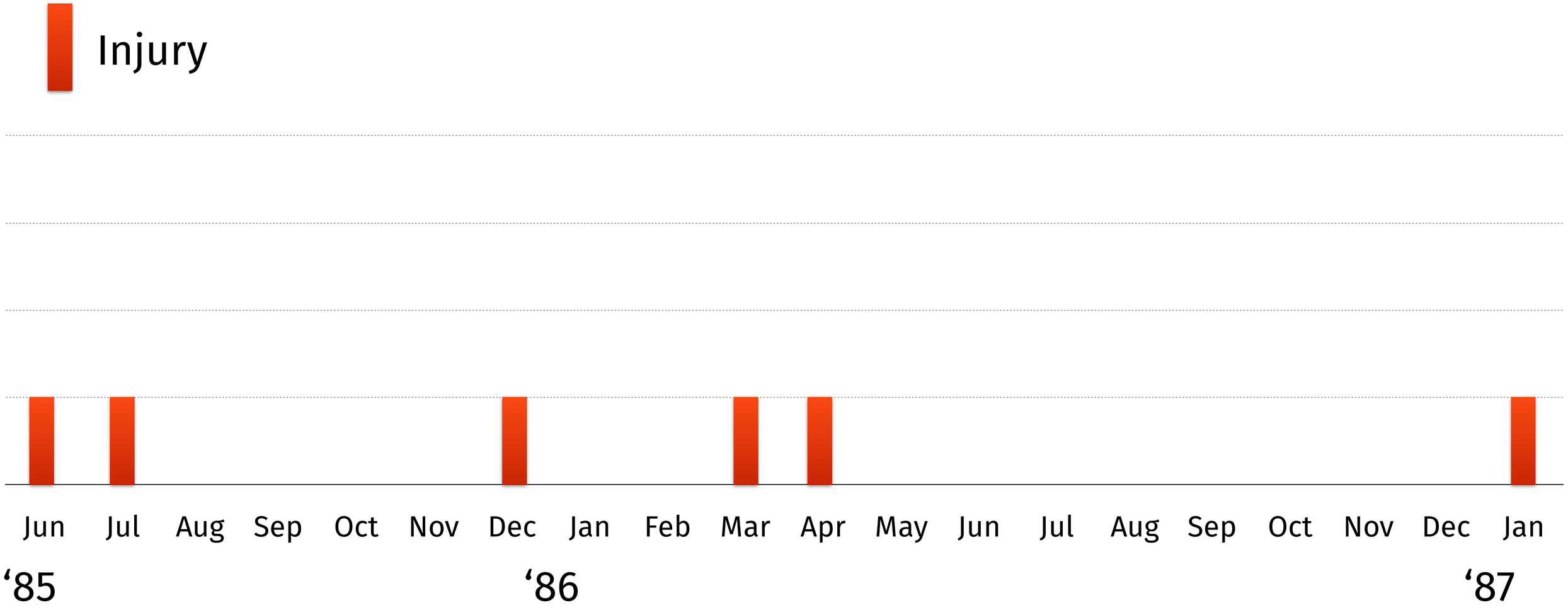


Therac-25

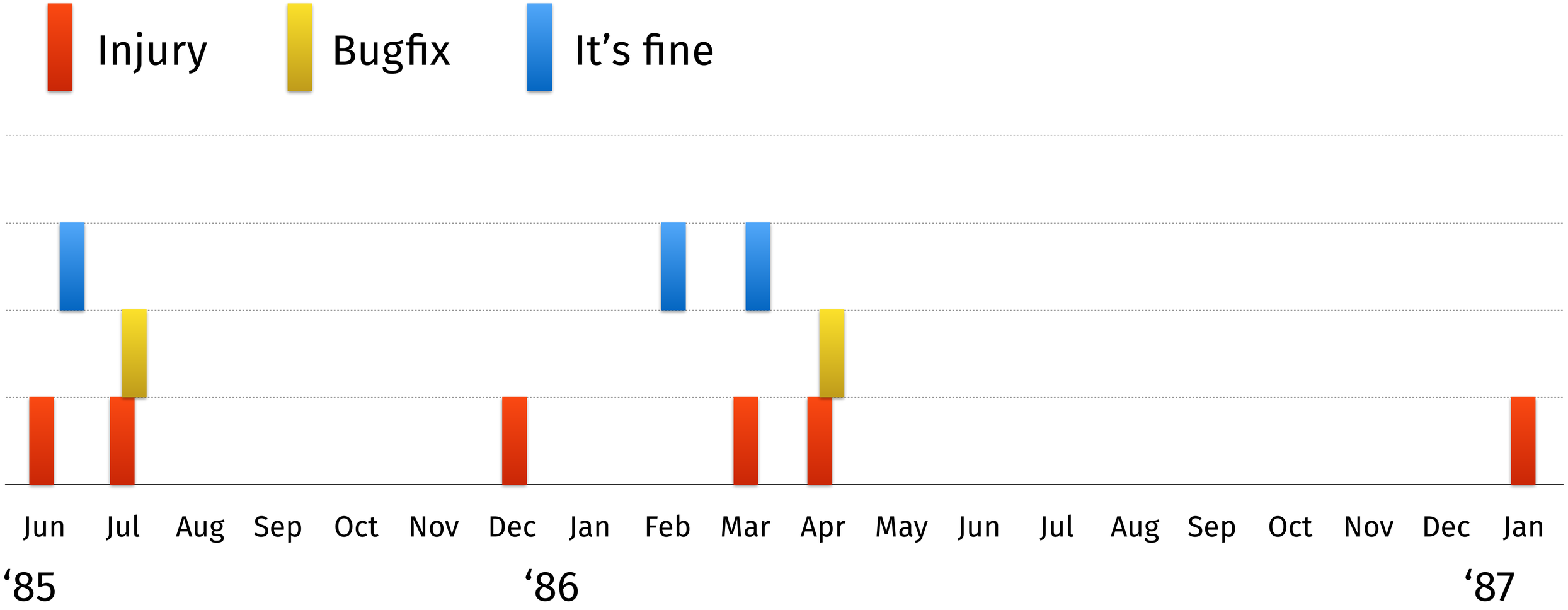
Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan

‘85 ‘86 ‘87

Therac-25


Injury
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
‘85
‘86
‘87

Therac-25


Injury
Bugfix
It’s fine
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
‘85
‘86
‘87

More Incidents

https://github.com/stanislaw/awesome-safety-critical#incidents

Quality of Software

Quality of Software

- Formal Verification

Quality of Software

- Formal Verification
- Fuzz Testing

Quality of Software

- Formal Verification
- Fuzz Testing
- Unit Testing + Code Coverage

Quality of Software

- Formal Verification
- Fuzz Testing
- **Unit Testing + Code Coverage**

Unit Testing

Unit Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}
```

Failed Tests: 0

Passed Tests: 1

Code Coverage: 100%

Mutation Testing

Mutation Testing

```
run_test(program, test)
```

Mutation Testing

```
run_test(program, test)
mutant = mutate(program)
```

Mutation Testing

```
run_test(program, test)
mutant = mutate(program)
result = run_test(mutant, test)
```

Mutation Testing

```
run_test(program, test)
mutant = mutate(program)
result = run_test(mutant, test)
if (result == Failed)
  report_killed_mutant(mutant, test)
```

Mutation Testing

```
run_test(program, test)
mutant = mutate(program)
result = run_test(mutant, test)
if (result == Failed)
  report_killed_mutant(mutant, test)
else
  report_survived_mutant(mutant, test)
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}

int sum'(int a, int b) {
  return a * b;
}
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}

int sum'(int a, int b) {
  return a * b;
}

int sum''(int a, int b) {
  return a - b;
}
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
test passed ->
  mutant survived
```

```
int sum(int a, int b) {
  return a + b;
}
```

```
int sum'(int a, int b) {
  return a * b;
}
```

```
int sum''(int a, int b) {
  return a - b;
}
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

test **passed** ->
  mutant **survived**

test **failed** ->
  mutant **killed**

```
int sum(int a, int b) {
  return a + b;
}
```

```
int sum'(int a, int b) {
  return a * b;
}
```

```
int sum''(int a, int b) {
  return a - b;
}
```

Mutation Testing

Total Mutants: 2

Killed Mutants: 1

Survived Mutants: 1

Mutation Score = killed / total * 100%

Mutation Score: **50%**

Mutation Testing

- First proposed by Richard Lipton in **1971**

Mutation Testing

- First proposed by Richard Lipton in **1971**
- First implemented by Timothy Budd in **1980**

Mutation Testing

- First proposed by Richard Lipton in **1971**
- First implemented by Timothy Budd in **1980**
- Studies say that MT was able to detect **70%-90%** of real faults

Mutation Testing

- Generates lots of data

Mutation Testing

- Generates lots of data
- Time consuming

Mutation Testing

- Generates lots of data
- Time consuming
- Languages are not mutation-testing-friendly

Mutation Testing

- Generates lots of data
- Time consuming
- Languages are not mutation-testing-friendly
- Problem of a Human Test Oracle

Mutation Testing

- Generates lots of data
- Time consuming
- Languages are not mutation-testing-friendly
- Problem of a Human Test Oracle
- "Excuse me, but I write good tests"

Mull

Mull

- Smart mutant selection

Mull

- Smart mutant selection
- Control over data generation

Mull

- Smart mutant selection
- Control over data generation
- Runtime compilation

Mull

- Smart mutant selection
- Control over data generation
- Runtime compilation
- Operates on LLVM IR level

Mull

- Smart mutant selection
- Control over data generation
- Runtime compilation
- Operates on LLVM IR level
- Language agnostic*

Language agnostic

Frontend

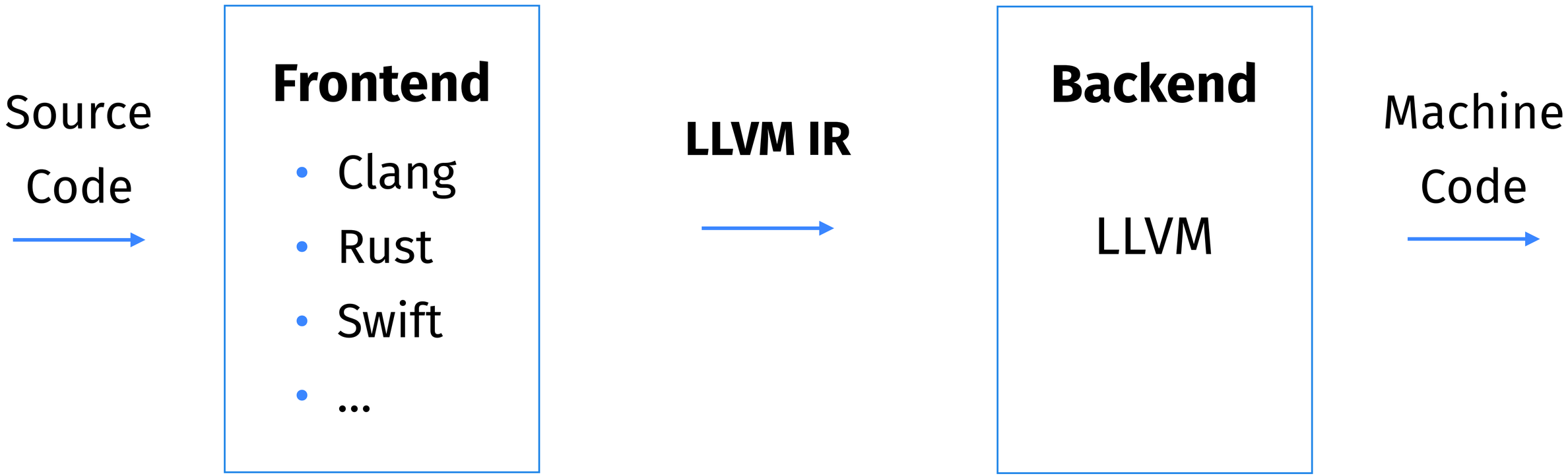
- Clang
- Rust
- Swift
- …

Backend

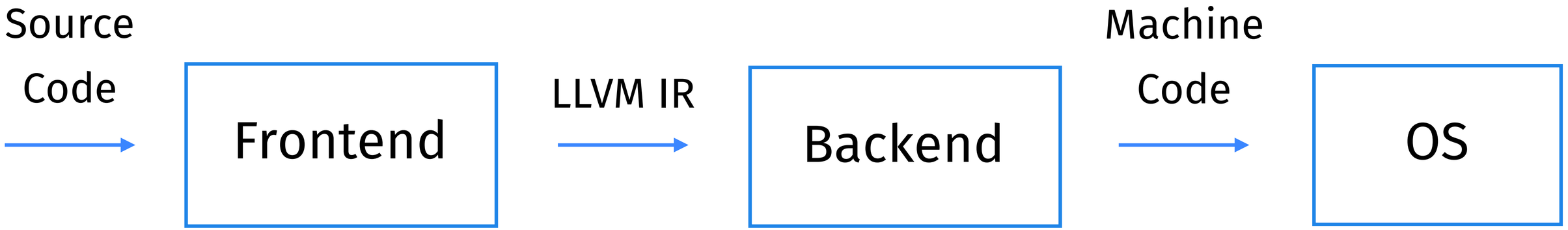
LLVM

Language agnostic

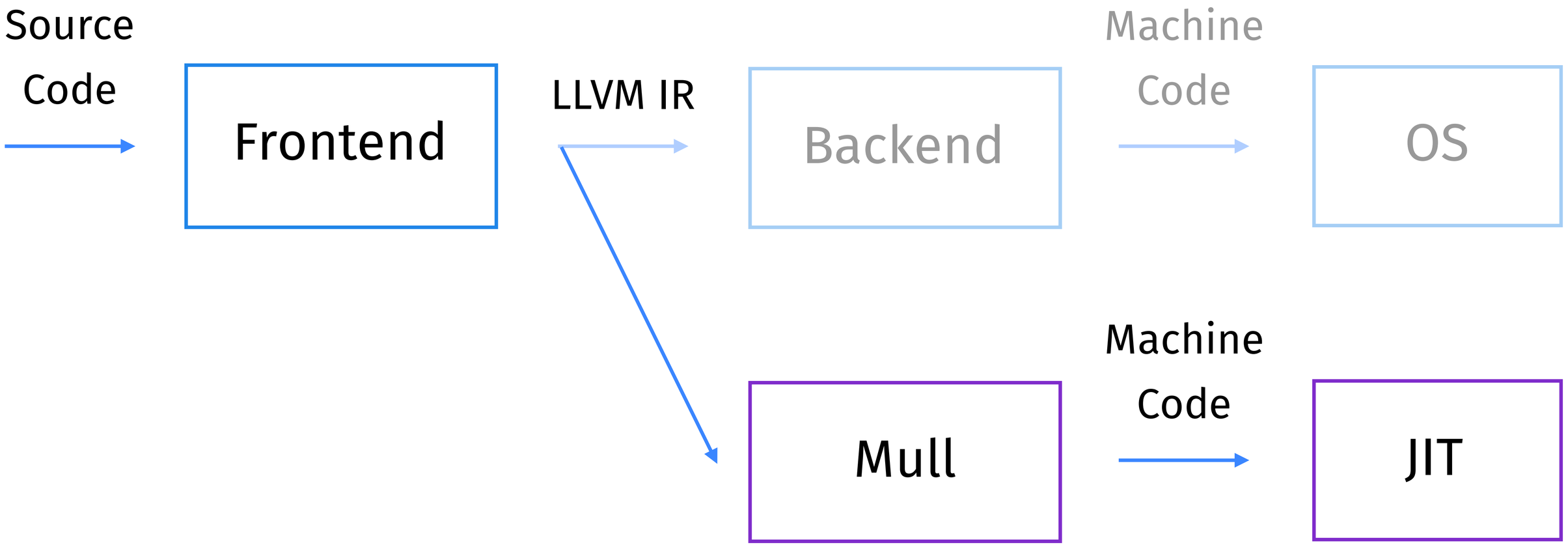
Language agnostic



Language agnostic



Language agnostic



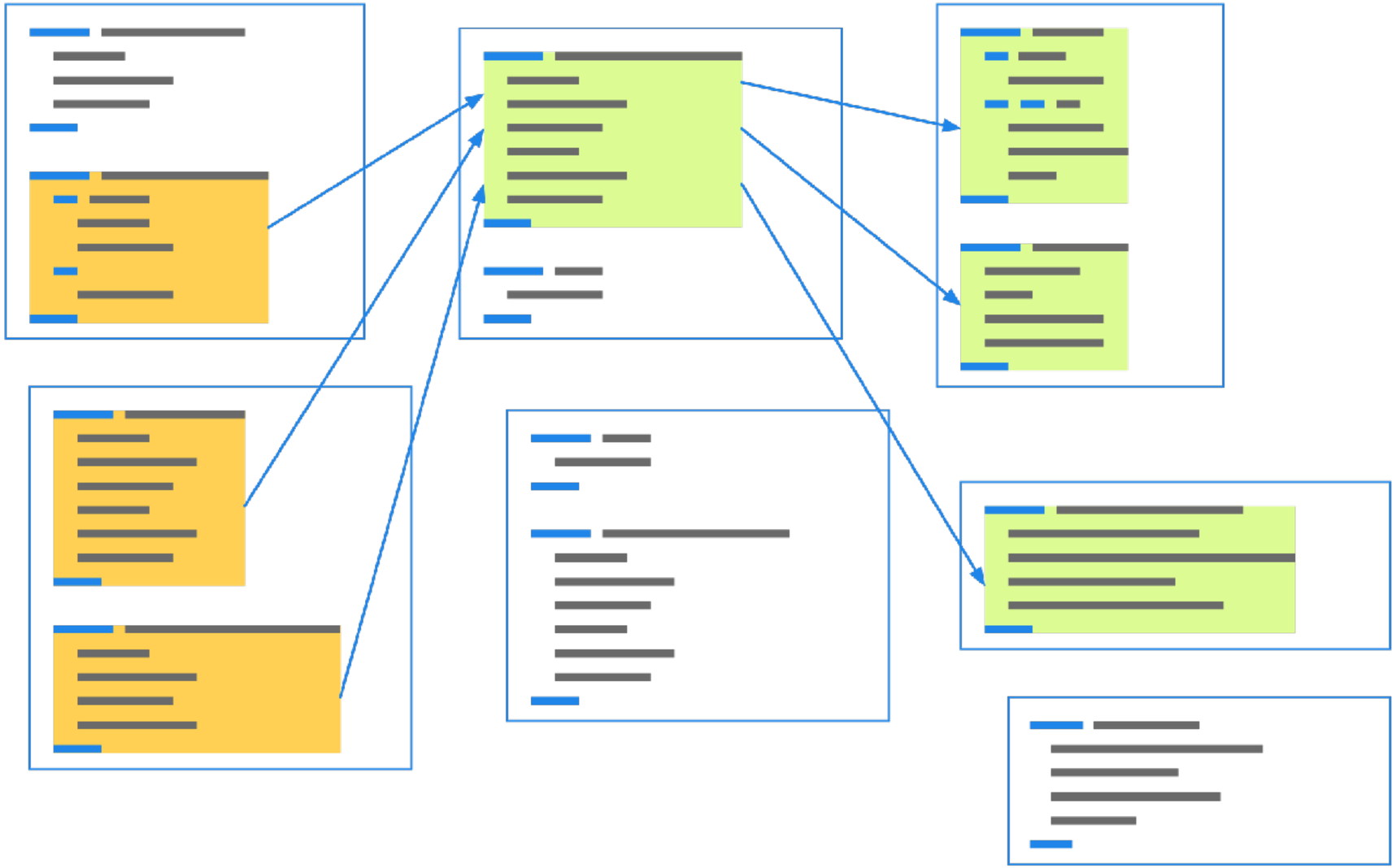
Mutant Selection

Mutant Selection

Mutant Selection

Mutant Selection

Mutant Selection



Mutant Selection

IRTests*: 238 tests

Before:

391 modules

85 minutes

** Part of LLVM's test suite*

IRTests*: 238 tests

Before:	After:
391 modules	124 modules
85 minutes	48 minutes

** Part of LLVM's test suite*

Mutation Control

Mutation Control

IRTests*: 238 tests

Distance: 2

Number of mutants: ~1.5k

Real execution time: ~1 hour

** Part of LLVM's test suite*

IRTests*: 238 tests

Distance: 2

Number of mutants: ~1.5k

Real execution time: ~1 hour

Distance: 29

Number of mutants: ~18k

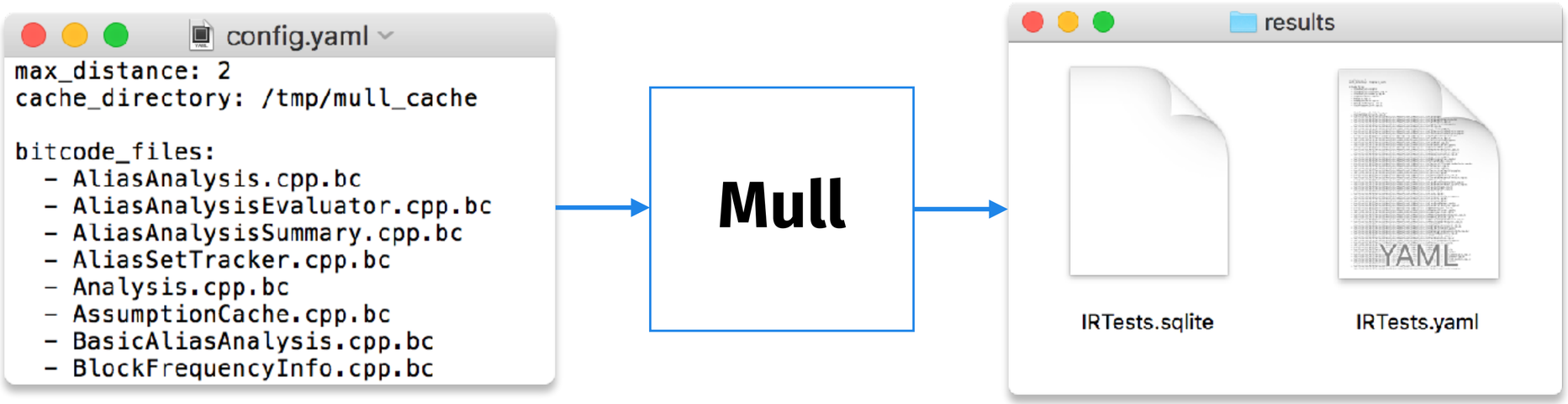
Approximate execution time: ~11 days

** Part of LLVM's test suite*

Mutation Control

System Design

System Design



System Design

Core

- Driver
- Reporter
- Mutation Operators

System Design

Core

- Driver
- Reporter
- Mutation Operators

Toolchain

- JIT Compiler
- Object Cache

System Design

Core

- Driver
- Reporter
- Mutation Operators

Toolchain

- JIT Compiler
- Object Cache

Test Framework

- Test Finder
- Test Runner

System Design

Core

- Driver
- Reporter
- Mutation Operators

Toolchain

- JIT Compiler
- Object Cache

Test Framework

Google Test

- Test Finder
- Test Runner

System Design

Core

- Driver
- Reporter
- Mutation Operators

Toolchain

- JIT Compiler
- Object Cache

Test Framework

Google Test

XCTest

- Test Finder
- Test Runner

Showcase

- LLVM - compilers and dev tools, C++.
- fmt - formatting library, C++.
- Nom - parser combinators library, Rust.
- CryptoSwift - collection of cryptographic algorithms, Swift.

Example #1

```
template<typename T>
unsigned ComputeEditDistance(ArrayRef<T> FromArray, ArrayRef<T> ToArray,
                             bool AllowReplacements = true,
                             unsigned MaxEditDistance = 0) {

  typename ArrayRef<T>::size_type m = FromArray.size();
  typename ArrayRef<T>::size_type n = ToArray.size();

  const unsigned SmallBufferSize = 64;
  unsigned SmallBuffer[SmallBufferSize];
  std::unique_ptr<unsigned[]> Allocated;
  unsigned *Row = SmallBuffer;
  if (n + 1 > SmallBufferSize) {
    Row = new unsigned[n + 1];
    Allocated.reset(Row);
  }

  for (unsigned i = 1; i <= n; ++i)
    Row[i] = i;

  for (typename ArrayRef<T>::size_type y = 1; y <= m; ++y) {
    Row[0] = y;
    unsigned BestThisRow = Row[0];

    unsigned Previous = y - 1;
    for (typename ArrayRef<T>::size_type x = 1; x <= n; ++x) {
      int OldRow = Row[x];
      if (AllowReplacements) {
        Row[x] = std::min(
            Previous + (FromArray[y-1] == ToArray[x-1] ? 0u : 1u),
            std::min(Row[x-1], Row[x])+1);
      }
      else {
        if (FromArray[y-1] == ToArray[x-1]) Row[x] = Previous;
        else Row[x] = std::min(Row[x-1], Row[x]) + 1;
      }
      Previous = OldRow;
      BestThisRow = std::min(BestThisRow, Row[x]);
    }

    if (MaxEditDistance && BestThisRow > MaxEditDistance)
      return MaxEditDistance + 1;
  }

  unsigned Result = Row[n];
  return Result;
}
```

```
template<typename T>
unsigned ComputeEditDistance(ArrayRef<T> FromArray, ArrayRef<T> ToArray,
                             bool AllowReplacements = true,
                             unsigned MaxEditDistance = 0) {

  typename ArrayRef<T>::size_type m = FromArray.size();
  typename ArrayRef<T>::size_type n = ToArray.size();

  const unsigned SmallBufferSize = 64;
  unsigned SmallBuffer[SmallBufferSize];
  std::unique_ptr<unsigned[]> Allocated;
  unsigned *Row = SmallBuffer;
  if (n + 1 > SmallBufferSize) {
    Row = new unsigned[n + 1];
    Allocated.reset(Row);
  }

  for (unsigned i = 1; i <= n; ++i)
    Row[i] = i;

  for (typename ArrayRef<T>::size_type y = 1; y <= m; ++y) {
    Row[0] = y;
    unsigned BestThisRow = Row[0];

    unsigned Previous = y - 1;
    for (typename ArrayRef<T>::size_type x = 1; x <= n; ++x) {
      int OldRow = Row[x];
      if (AllowReplacements) {
        Row[x] = std::min(
            Previous + (FromArray[y-1] == ToArray[x-1] ? 0u : 1u),
            std::min(Row[x-1], Row[x])+1);
      }
      else {
        if (FromArray[y-1] == ToArray[x-1]) Row[x] = Previous;
        else Row[x] = std::min(Row[x-1], Row[x]) + 1;
      }
      Previous = OldRow;
      BestThisRow = std::min(BestThisRow, Row[x]);
    }

    if (MaxEditDistance && BestThisRow > MaxEditDistance)
      return MaxEditDistance + 1;
  }

  unsigned Result = Row[n];
  return Result;
}
```

```
template<typename T>
unsigned ComputeEditDistance(ArrayRef<T> FromArray, ArrayRef<T> ToArray,
                             bool AllowReplacements = true,
                             unsigned MaxEditDistance = 0) {

  typename ArrayRef<T>::size_type m = FromArray.size();
  typename ArrayRef<T>::size_type n = ToArray.size();

  const unsigned SmallBufferSize = 64;
  unsigned SmallBuffer[SmallBufferSize];
  std::unique_ptr<unsigned[]> Allocated;
  unsigned *Row = SmallBuffer;

  for (typename ArrayRef<T>::size_type y = 1; y <= m; ++y) {
    Row[0] = y;
    unsigned BestThisRow = Row[0];

    unsigned Previous = y - 1;
    for (typename ArrayRef<T>::size_type x = 1; x <= n; ++x) {
      int OldRow = Row[x];
      if (AllowReplacements) {
        Row[x] = std::min(
            Previous + (FromArray[y-1] != ToArray[x-1] ? 0u : 1u),
            std::min(Row[x-1], Row[x])+1);
      }
      Previous = OldRow;
      BestThisRow = std::min(BestThisRow, Row[x]);
    }
  }

  unsigned Result = Row[n];
  return Result;
}
```

```
TEST(StringRefTest, EditDistance) {
  StringRef Str("hello");
  EXPECT_EQ(2U, Str.edit_distance("hill"));
}
```

Fix: r300312

```
TEST(StringRefTest, EditDistance) {
  StringRef Hello("hello");
  EXPECT_EQ(2U, Hello.edit_distance("hill"));

  StringRef Soylent("soylent green is people");
  StringRef People("people soiled our green");
  EXPECT_EQ(19U, Soylent.edit_distance(People));
  EXPECT_EQ(26U, Soylent.edit_distance(People, false));
  EXPECT_EQ(9U, Soylent.edit_distance(People, true, 8));
}
```

Example #2

```
Triple T = Triple("");
T.setObjectFormat(Triple::ELF);
EXPECT_EQ(Triple::ELF, T.getObjectFormat());
```

```
Triple T = Triple("");
// T.setObjectFormat(Triple::ELF);
EXPECT_EQ(Triple::ELF, T.getObjectFormat());
```

Fix: r294104

```
Triple T = Triple("");
T.setObjectFormat(Triple::ELF);
EXPECT_EQ(Triple::ELF, T.getObjectFormat());

T.setObjectFormat(Triple::MachO);
EXPECT_EQ(Triple::MachO, T.getObjectFormat());
```

Example #3

```
T.setArch(Triple::mips64);
EXPECT_EQ(Triple::mips64el,
          T.getLittleEndianArchVariant().getArch());
```

Fix: r294095, r294096

```
T.setArch(Triple::mips64);
EXPECT_EQ(Triple::mips64el,
          T.getLittleEndianArchVariant().getArch());

T.setArch(Triple::tce);
EXPECT_EQ(Triple::tcele,
          T.getLittleEndianArchVariant().getArch());
```

Triple_82_4_0_remove_void_function_mutation_operator 0/1

Affected Tests:

```
TripleTest.EndianArchVariants
```

/usr/local/LLVM/llvm/lib/Support/Triple.cpp:1413

```
  case Triple::tce:         T.setArch(Triple::tcele);     break;
                              ^
```

Survived

Distance: 1

Duration: 366ms

Caller path:

```
/usr/local/LLVM/llvm/unittests/ADT/TripleTest.cpp:693
  /usr/local/LLVM/llvm/lib/Support/Triple.cpp:1413
```

Caller path (source code):

Triple_82_4_0_remove_void_function_mutation_operator 0/1

Affected Tests:

```
TripleTest.EndianArchVariants
```

/usr/local/LLVM/llvm/lib/Support/Triple.cpp:1413

```
  case Triple::tce:         T.setArch(Triple::tcele);     break;
                              ^
```

Survived

Distance: 1

Duration: 366ms

Caller path:

```
/usr/local/LLVM/llvm/unittests/ADT/TripleTest.cpp:693
  /usr/local/LLVM/llvm/lib/Support/Triple.cpp:1413
```

Caller path (source code):

Demo

Results

- https://lowlevelbits.org/ADTTests/
- https://lowlevelbits.org/IRTests/
- https://github.com/krzyzanowskim/CryptoSwift/issues/417

Project: https://github.com/mull-project/mull

Contact: alex@lowlevelbits.org

Updates: https://twitter.com/1101_debian

Questions?

Project: https://github.com/mull-project/mull

Contact: alex@lowlevelbits.org

Updates: https://twitter.com/1101_debian