Source Code Analyzer Based on Clang

高晚秋

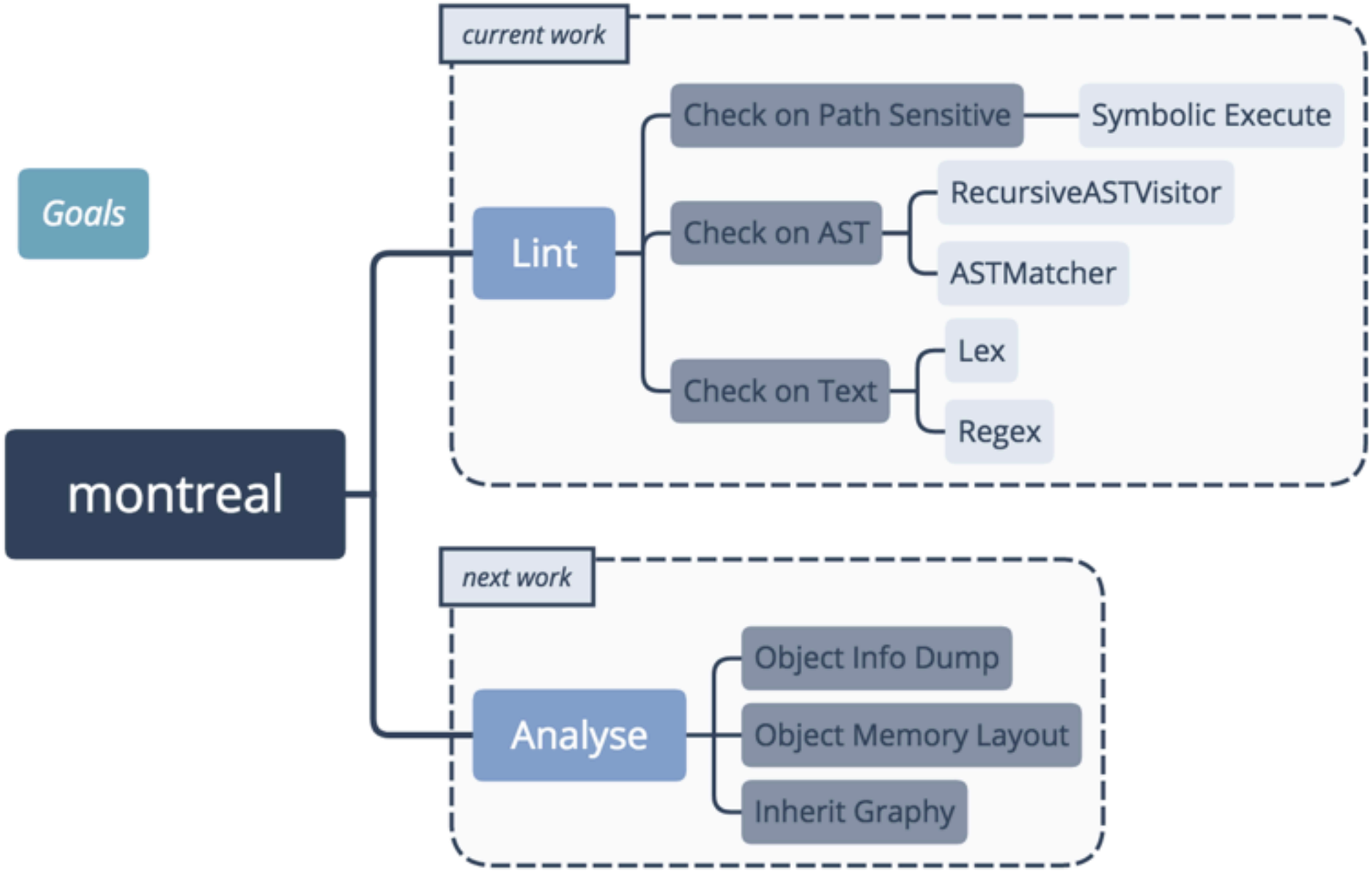
montreal

High customized and flexible static analyzer for ObjC 🎉

Why we need this?

1. Checkers in clang static analyzer are designed for generic bugs/problems.
2. We need a deep analyzer to help new employee to understand project code quickly.
3. Drive myself
4. Just for fun 😜

Use it for:


Goals
montreal
current work
Lint
Check on Path Sensitive
Symbolic Execute
Check on AST
RecursiveASTVisitor
ASTMatcher
Check on Text
Lex
Regex
next work
Analyse
Object Info Dump
Object Memory Layout
Inherit Graphy

Choose Proper Interface

1. LibClang
2. LibTooling
3. ClangPlugin

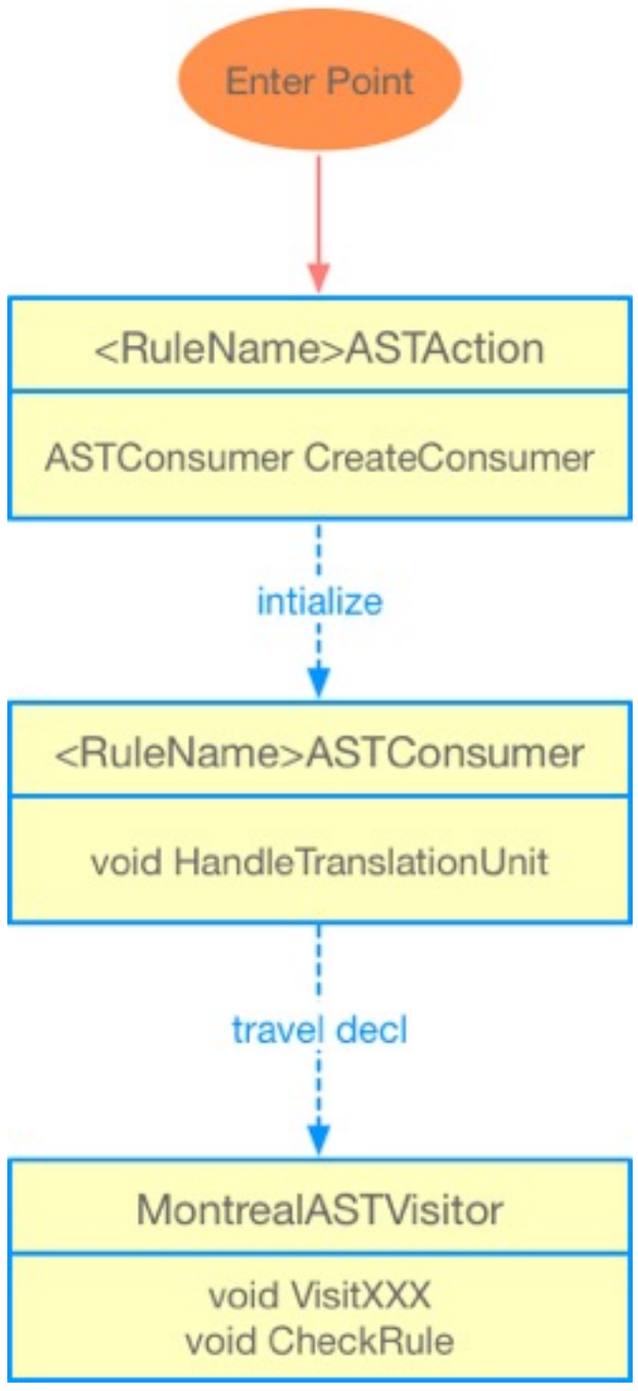
Choose Proper Interface

1. LibClang
2. LibTooling
3. ClangPlugin 

Implementation

1. Plugin based on AST
2. Plugin based on static checker

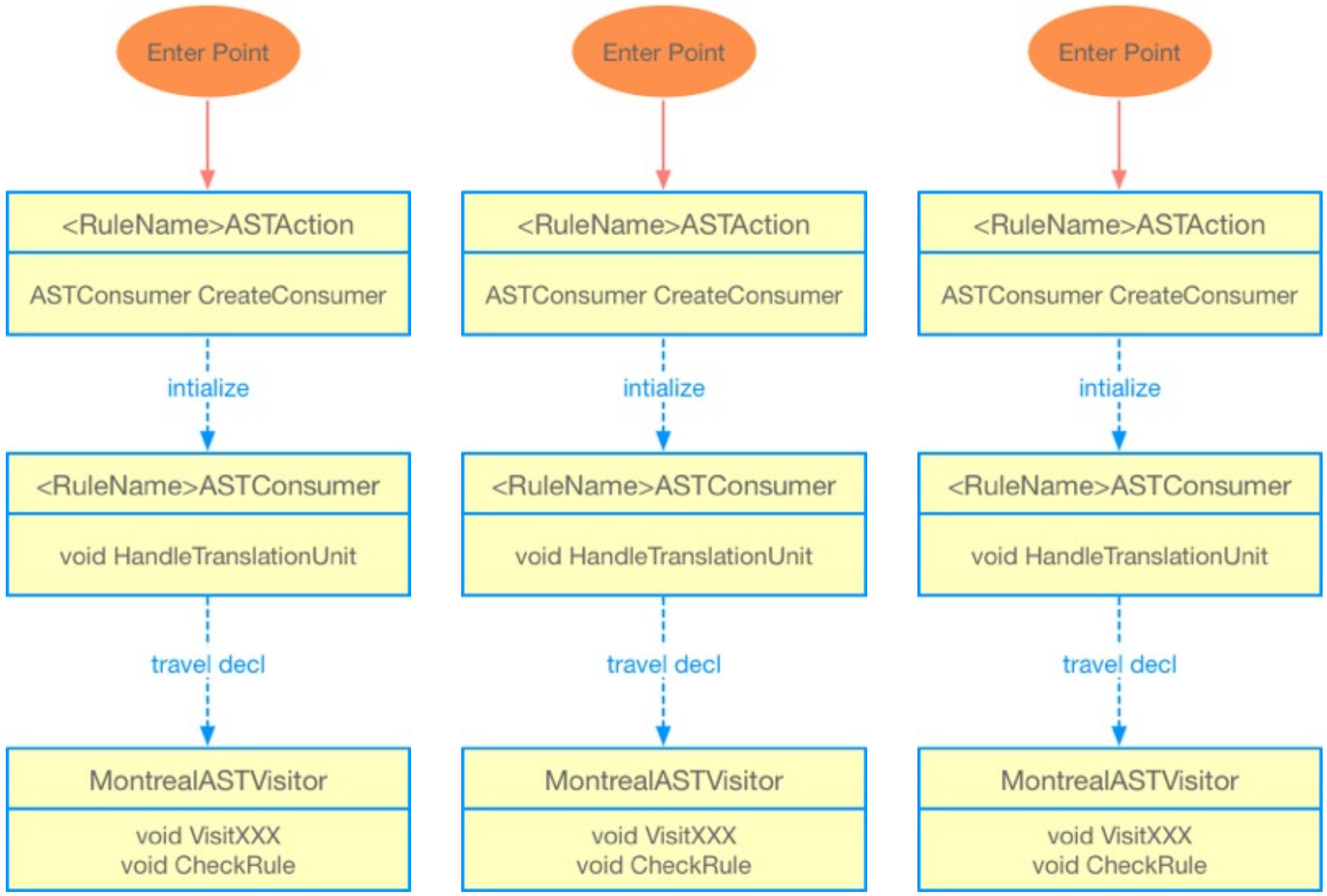
ASTPlugin Arch #1



one rule

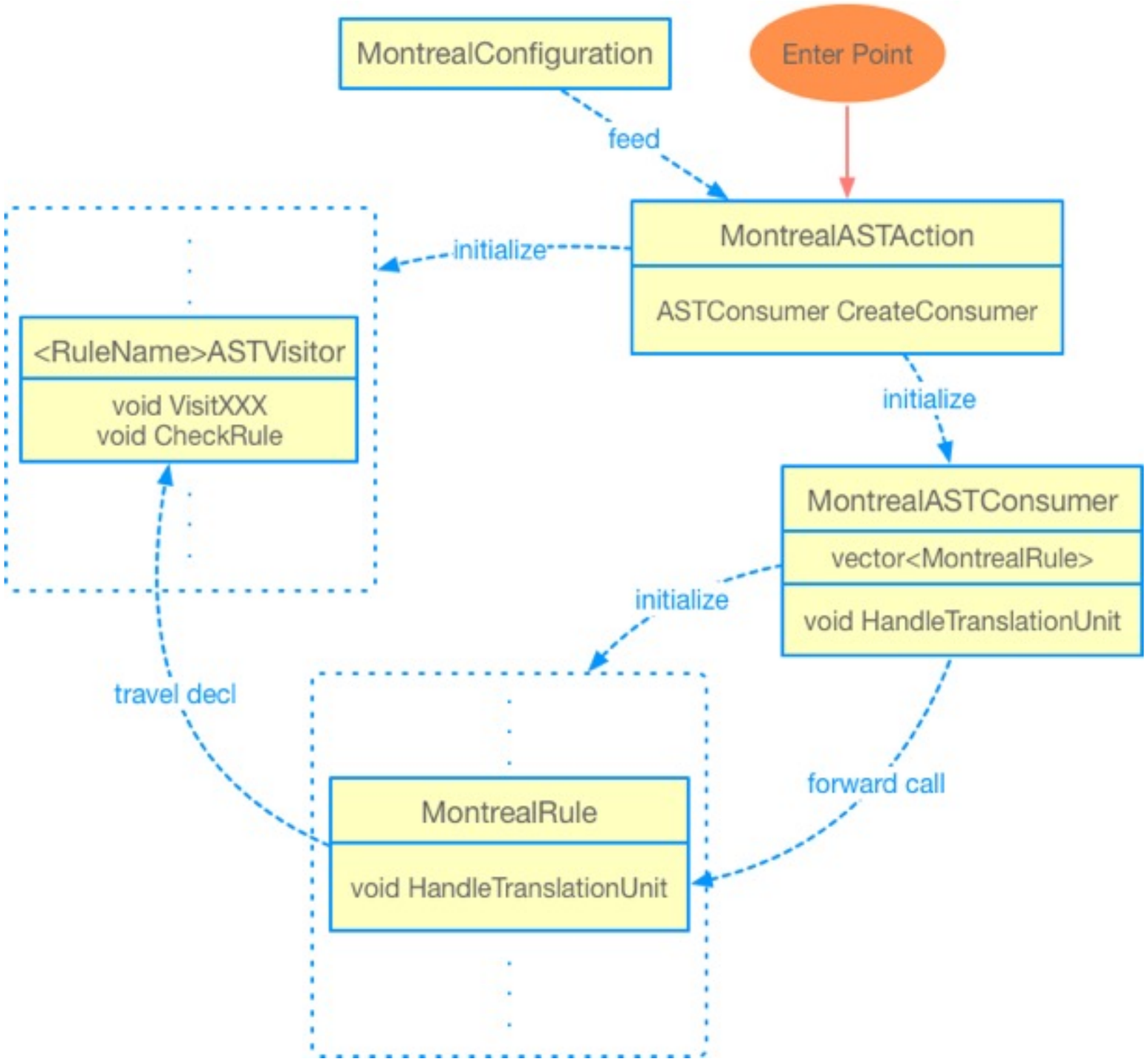
ASTPlugin Arch #1

multiple rules



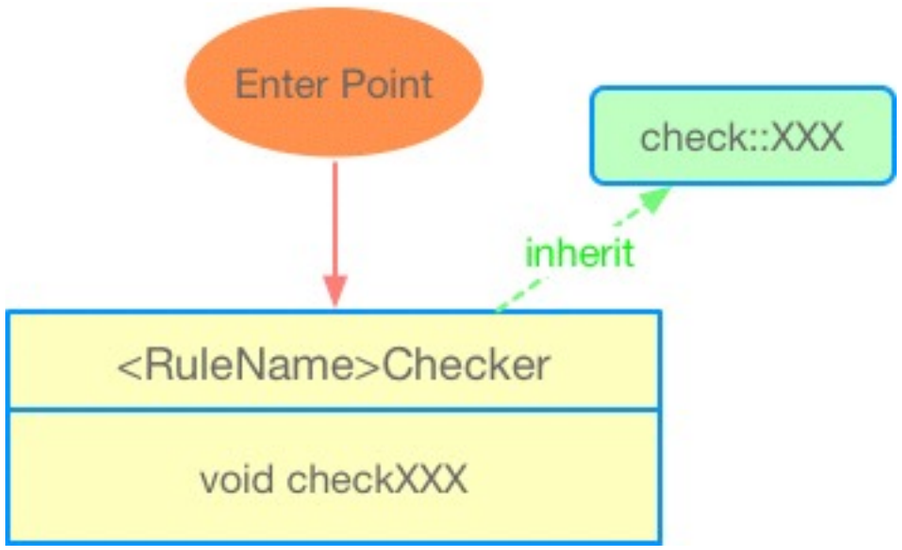
...

ASTPlugin Arch #2



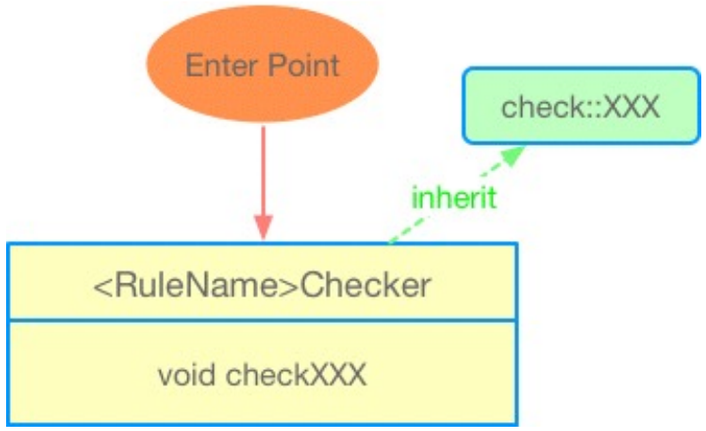
CheckerPlugin Arch #1

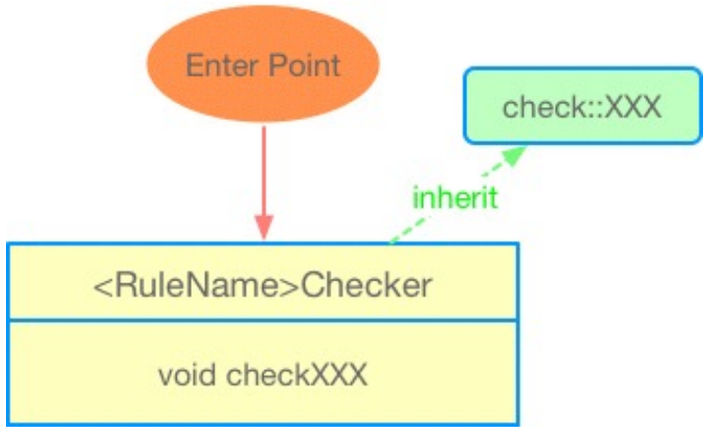
one rule



CheckerPlugin Arch #1

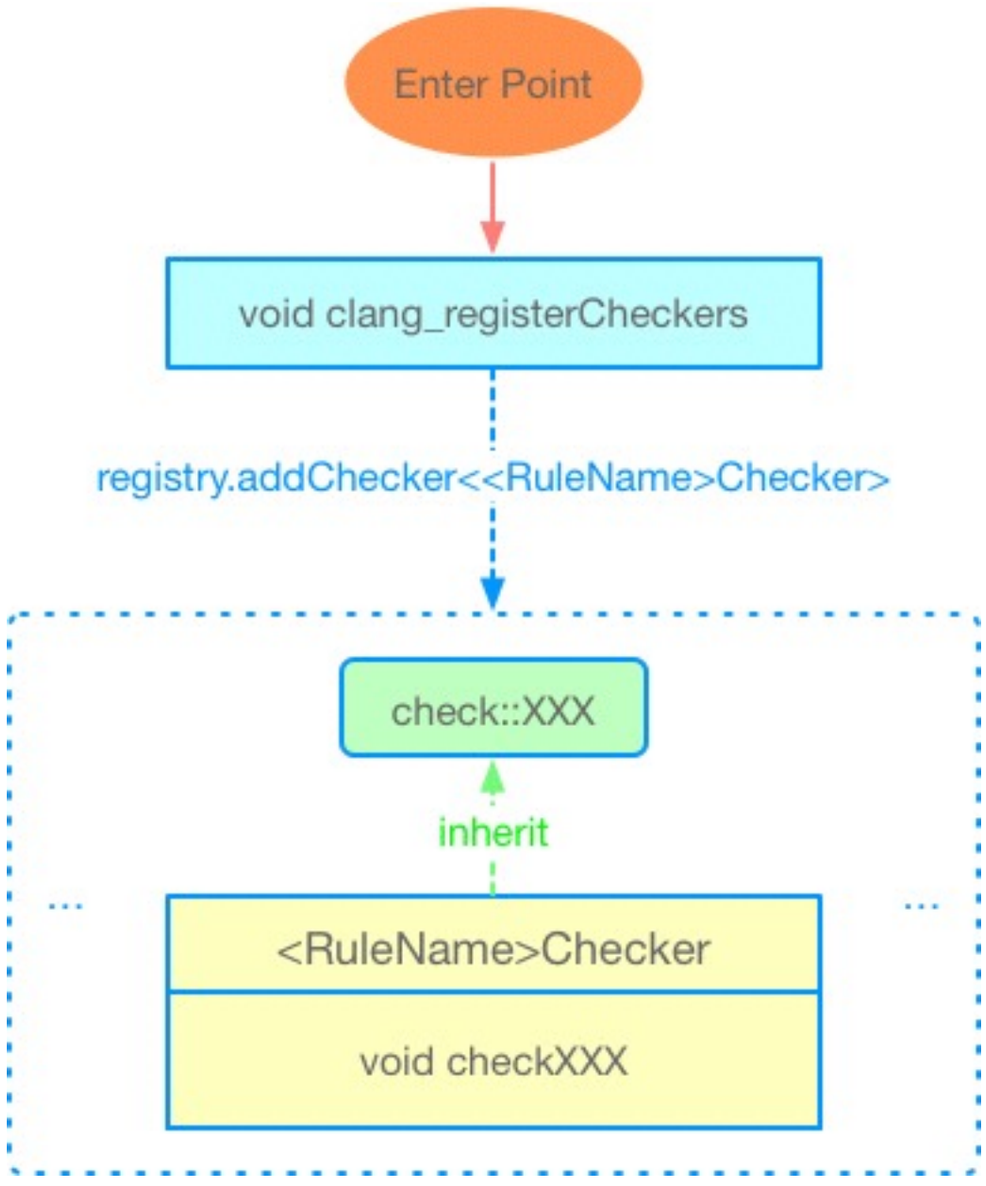
multiple rules





...

CheckerPlugin Arch #2



checker based on AST

```
# gaoge at gaoge.local in ~/Development/montreal/test on git:test ✖ [21:49:20]
→ clang -Xclang -load -Xclang ../build/libcheck_objc_property_attribute.dylib -Xclang -plugin -Xclang check-objc-property-attribute -fsyntax-onl
y ./objc_property_attribute_test.m -fobjc-arc
Analysing file: ./objc_property_attribute_test.m
In file included from ./objc_property_attribute_test.m:1:
./objc_property_attribute_test.h:10:1: error: property with 'copy' attribute must be of object type
@property (nonatomic, copy) CGRect testCStruct;
^
./objc_property_attribute_test.h:6:1: warning: property 'testBlock' is a block type, should set copy
@property (nonatomic, strong) void(^testBlock)(void);
^
./objc_property_attribute_test.h:7:1: error: property 'testString2' is a object type, must not set assign
@property (nonatomic, assign) NSString *testString2;
^
./objc_property_attribute_test.h:8:1: warning: property 'testArray' is a container type, should set copy
@property (nonatomic, assign) NSArray<NSObject *> *testArray;
^
./objc_property_attribute_test.h:9:1: error: property 'mutableSet' is a mutable container type, must not set copy
@property (nonatomic, copy) NSMutableSet<NSObject *> *mutableSet;
^
./objc_property_attribute_test.h:10:1: error: property 'testCStruct' is a primary type, must set assign
@property (nonatomic, copy) CGRect testCStruct;
^
2 warnings and 4 errors generated.
```

checker based on AST

```
# gaoge at gaoge.local in ~/Development/montreal/test on git:master ● [21:54:15]
→ cat objc_block_must_call_test.m
#import <Foundation/Foundation.h>

@interface TestBlock : NSObject
/// montreal:must-call
@property (nonatomic, copy) void(^testBlock)(void);

@end

@implementation TestBlock

- (instancetype)init {
  if (self = [super init]) {
    self.testBlock();
  }
  return self;
}

+ (instancetype)sharedInstance {
  return [[self alloc] init];
}

@end
```

```
# gaoge at gaoge.local in ~/Development/montreal/test on git:test ● [21:48:08]
→ clang -Xclang -load -Xclang ../build/libcheck_objc_block_must_call.dylib -Xclang -plugin -Xclang check-objc-block-must-call -fsyntax-only ./objc_block_must_call_test.m -fobjc-arc
Analysing file: ./objc_block_must_call_test.m
error: 'testBlock' is an must-call block, need be called at least once in same TU
1 error generated.
```

checker based on symbolic execution

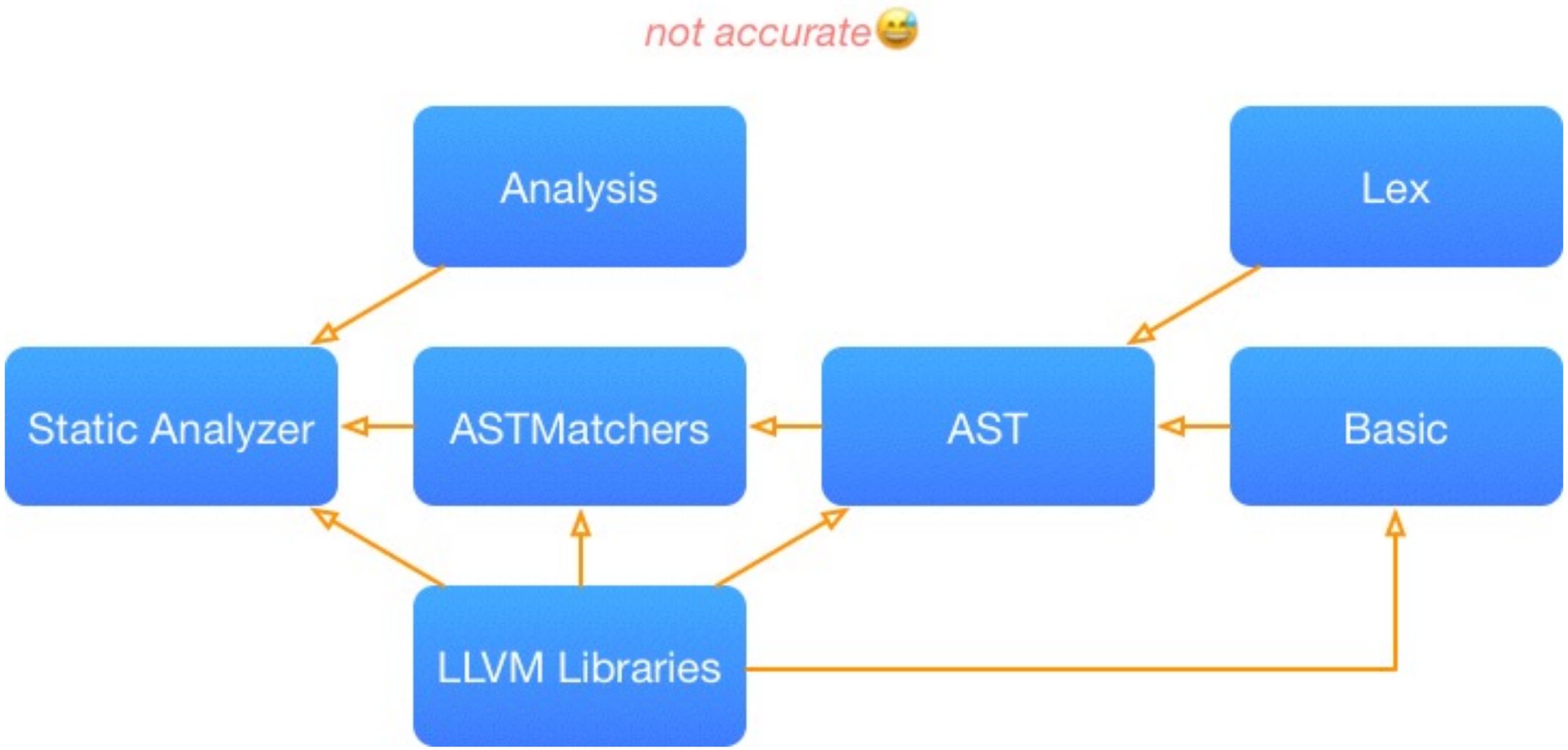
```
# gaoge at gaoge.local in ~/Development/clang-tools/test on git:master ● [21:52:52]
→ cat checker_test.c
int SCRAM();
int turnReactorOn();

void test_loop(int wrongTemperature, int restart) {
  turnReactorOn();
  if (wrongTemperature) {
    SCRAM();
  }
  if (restart) {
    SCRAM();
  }
  turnReactorOn();
  SCRAM();
}
```

```
# gaoge at gaoge.local in ~/Development/clang-tools on git:master ● [18:48:54]
→ clang -cc1 -load ./build/libchecker.dylib -analyze -analyzer-checker="example.reactorChecker" -fcolor-diagnostics ./test/checker_test.c
./test/checker_test.c:10:5: warning: Called a SCRAM procedure twice
    SCRAM();
    ^~~~~~~
./test/checker_test.c:12:3: warning: Turn on the reactor two times
  turnReactorOn();
  ^~~~~~~~~~~~~~
2 warnings generated.
```

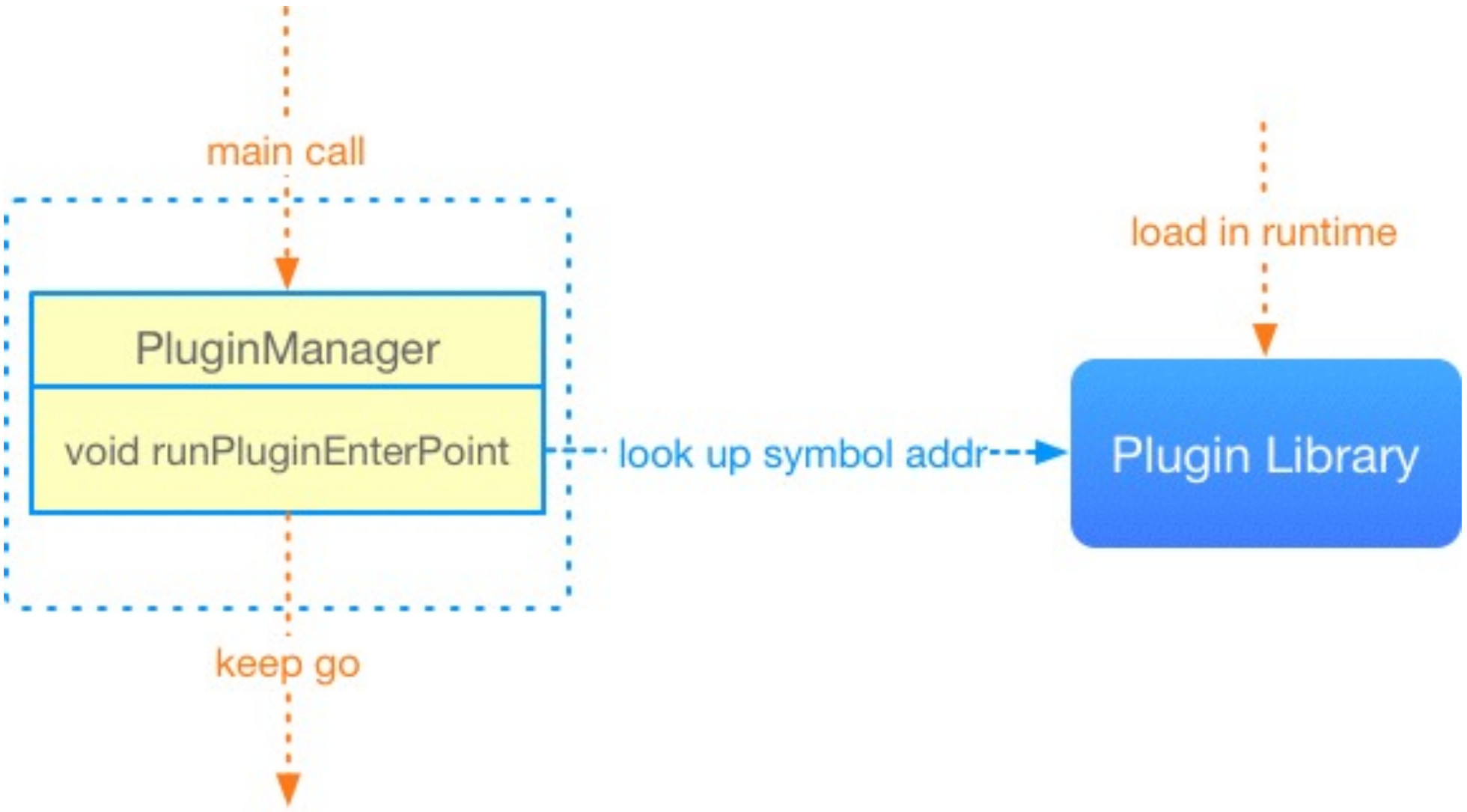
So, how it works?

#1 How plugin works

Clang is a set of libraries



load plugin library in runtime

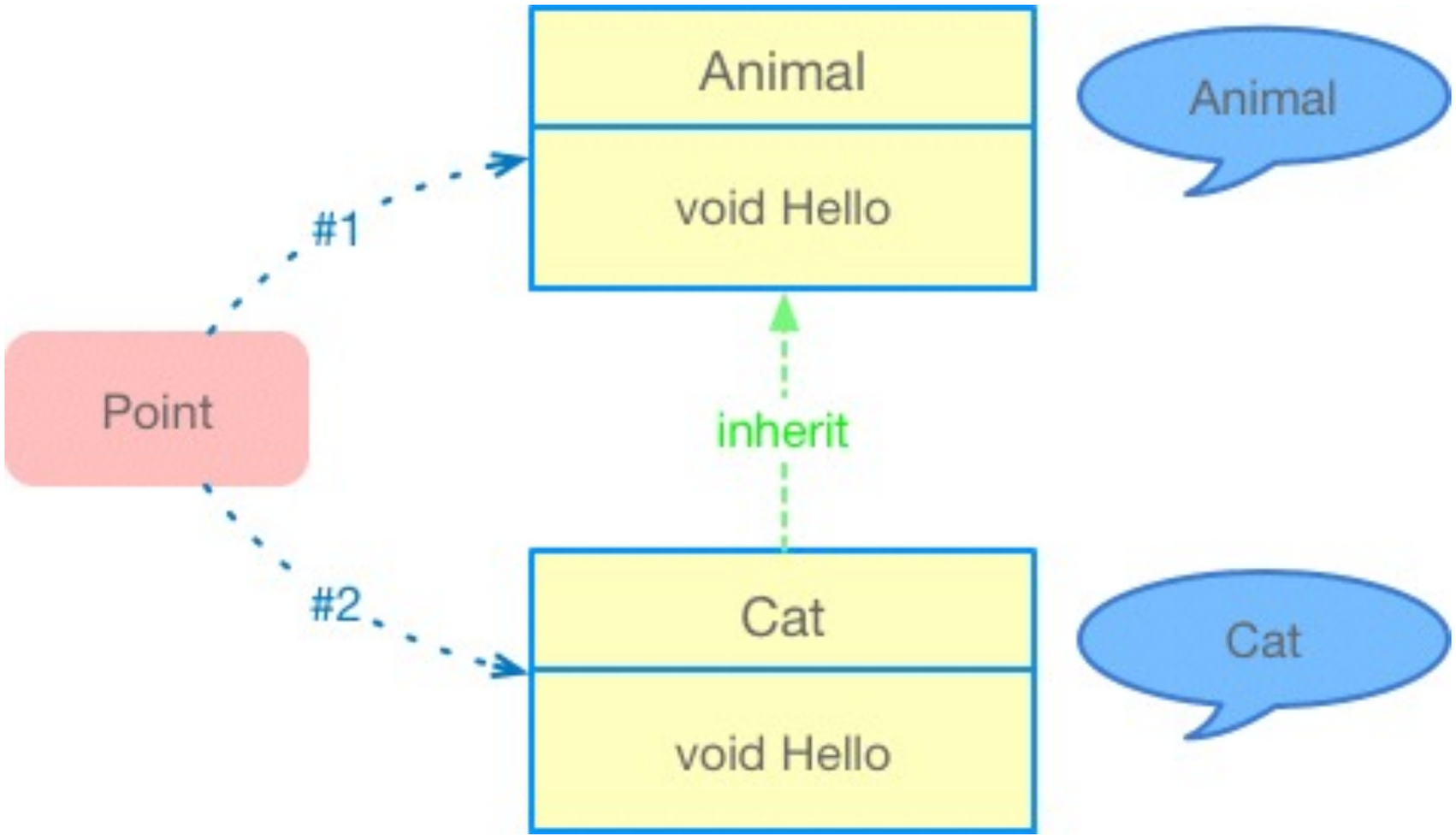


#2 How RecursiveASTVisitor works

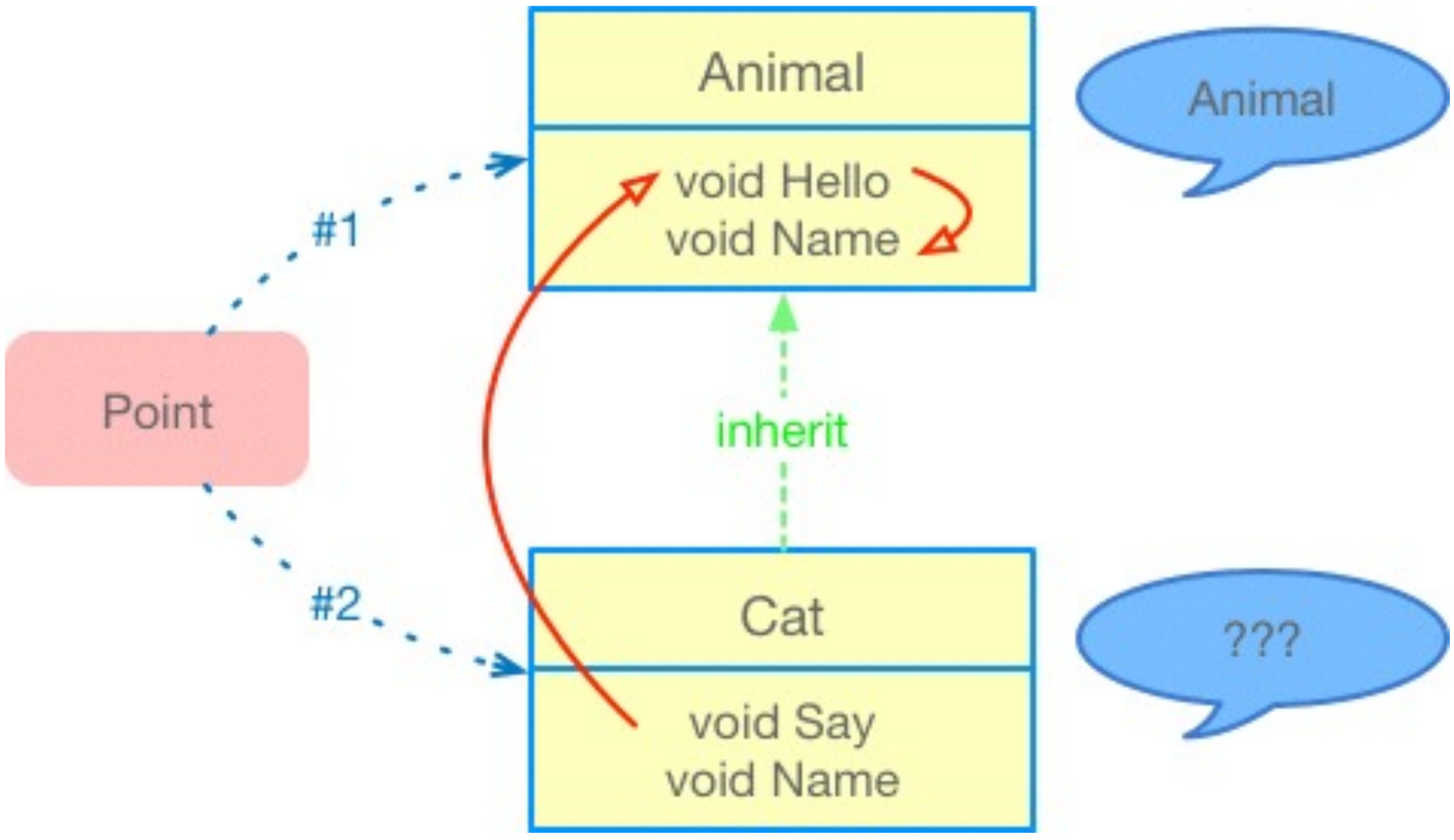
polymorphism in OOP

1. function address resolved in compiler time
2. function address resolved in runtime

liskov substitution principle



but … for this case?🤔



curiously recurring template pattern (CRTP)

```
class CheckObjCBlockMustCallVisitor
    : public RecursiveASTVisitor<CheckObjCBlockMustCallVisitor> {
private:
  ASTContext *Context;
  std::list<std::string> blocksMustCall;

  void CheckObjCBlockMustCall(ObjCPropertyDecl *Decl) {
    auto Type = Decl->getType();

    if (Type->getAs<BlockPointerType>()) { // property is a block type
      auto Comment = Context->getRawCommentForDeclNoCache(Decl);
      auto RawCommentString = Comment->getRawText(Context->getSourceManager());
      if (RawCommentString.contains(
              "montreal:must-call")) { // current block must be call
        blocksMustCall.push_back(Decl->getNameAsString());
      }
    }
  }
```

Bonus

```
3  class CARS<T> {
4    func test() -> String {
5      return self.implement()
6    }
7
8    func implement() -> String {
9      return "CARS"
10   }
11 }
12
13 class BARS : CARS<BARS> {
14   override func implement() -> String {
15     return "BARS"
16   }
17 }
18
19 //let bars = BARS() // won't work
```

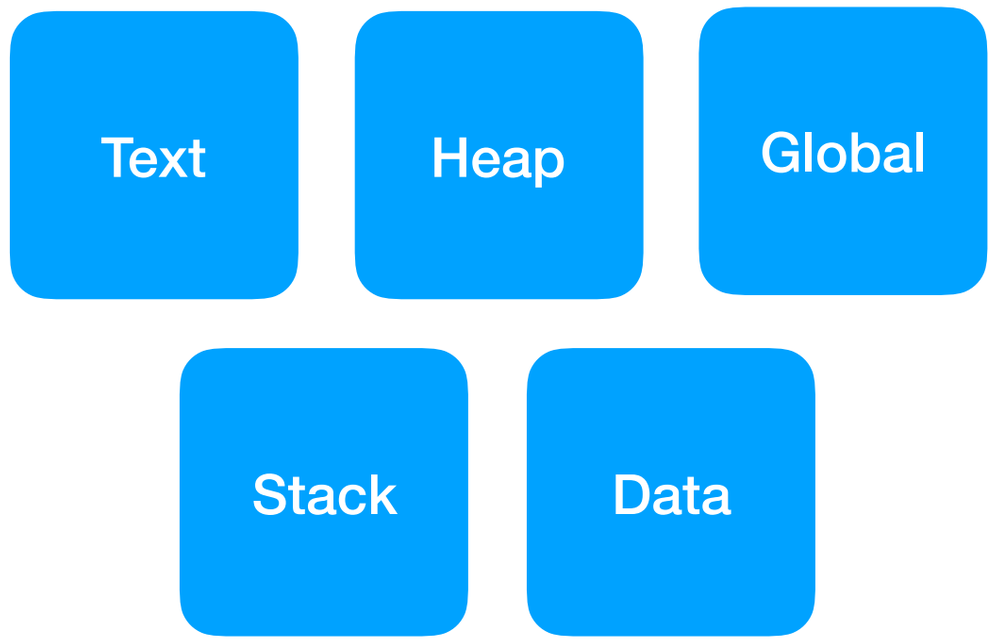
CRTP won’t work in Swift, it is a bug actually

Bonus

```
21  class Base {
22    func hello() -> String {
        return name()
24    }
25
26    func name() -> String {
        return "Base"
28    }
29  }
30
    let base = Base()
    base.hello() // print "Base"
33
34  class Sub : Base {
35    override func hello() -> String {
        return name()
37    }
38
39    override func name() -> String {
        return "Sub"
41    }
42  }
43
    let sub = Sub()
    sub.hello() // print "Sub"
46  //let cast_sub = unsafeDowncast(sub, to: Base.self)
47  //let cast_sub = unsafeBitCast(sub, to: Base.self)
    let cast_sub = sub as Base
    cast_sub.hello() // print "Sub"
```

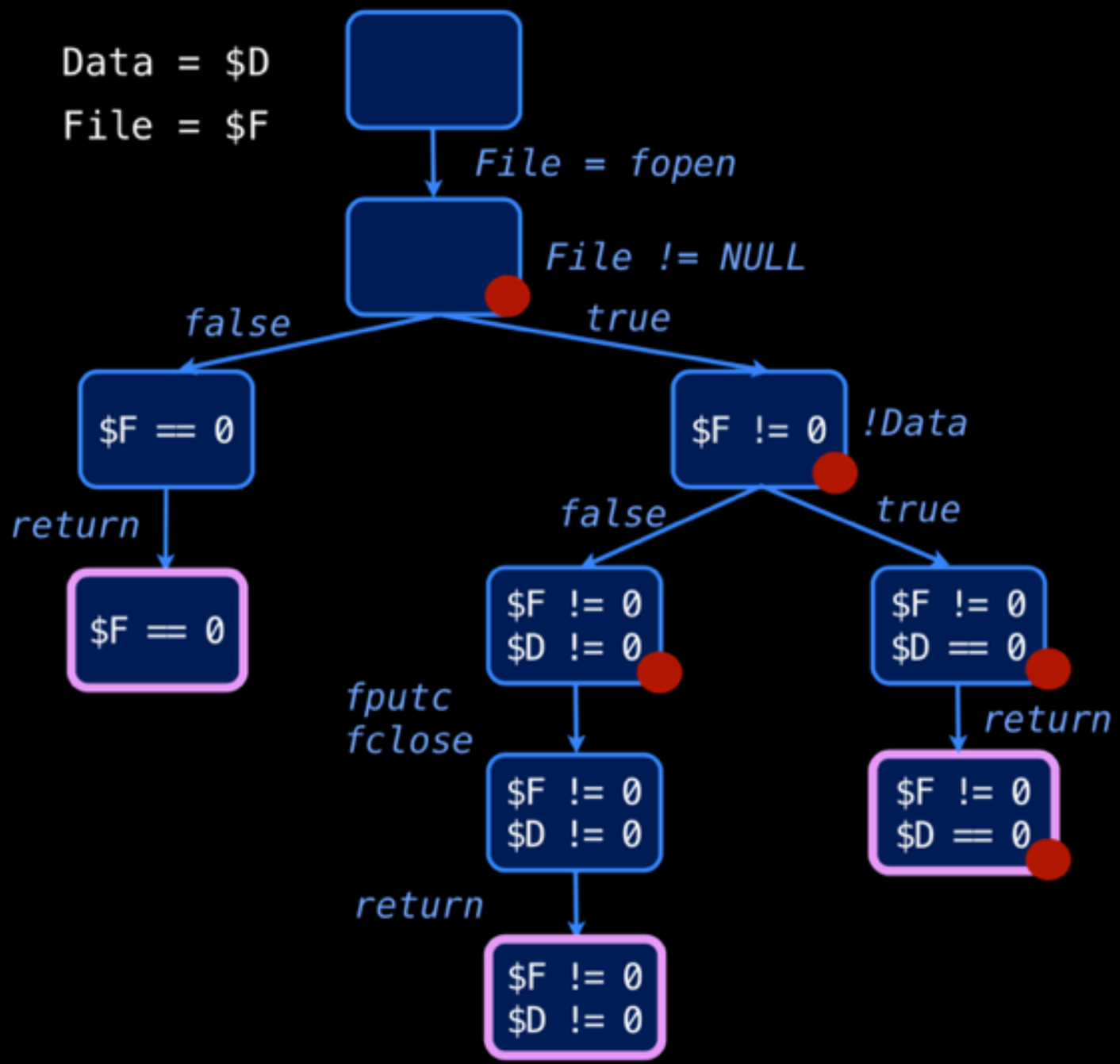
```
21  class Base {
22    func hello() -> String {
        return name()
24    }
25
26    func name() -> String {
        return "Base"
28    }
29  }
30
    let base = Base()
    base.hello() // print "Base"
33
34  class Sub : Base {
35    override func name() -> String {
        return "Sub"
37    }
38
39    func say() -> String {
        return hello()
41    }
42  }
43
    let sub = Sub()
    sub.say() // print "Sub"
```

#3 How static analyzer works



memory region

```
void writeCharToLog(char *Data) {
    FILE *File = fopen("mylog.txt", "w");

    if (File != NULL) {

        if (!Data)
            return;

        fputc(*Data, File);
        fclose(File);
    }

    return;
}
```



● Denotes that the file is open

capture from llvm-dev talk

Thank you 😎

#swordlinker