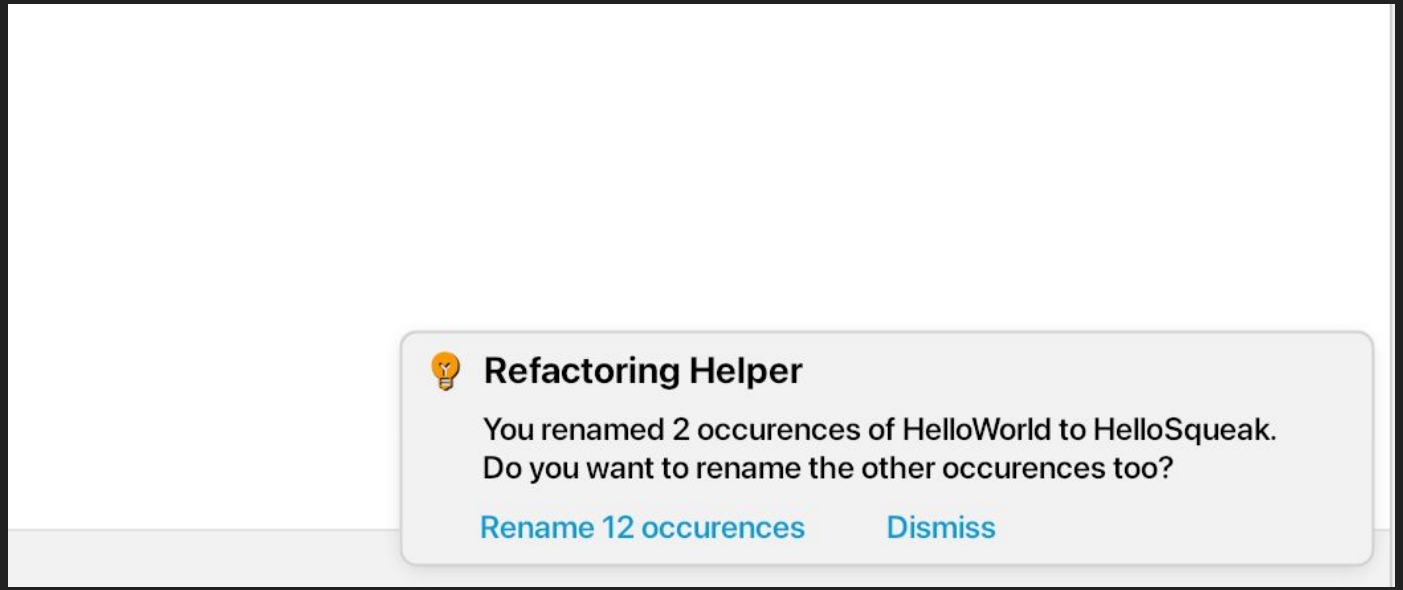
Detecting Edit Actions

Reverse Engineering SoSe 2022

Paul Methfessel, Antonius Naumann
Betreuer: Tom Beckmann

Motivation

Example 1: Action aware refactoring tooling in editor



1. Tool Mock: Unser tool erlaubt es refactorings zu erkennen. Darauf aufbauend könnte wie im Bild ein Toll gebaut werden, dass angefangene Actions (hier: rename) erkennt und vorschlägt diese zu vervollständigen

Example 2: Semantic grouping of editing actions for exploring changes in PR

Connected Methods (get_player...)

18 players/model_player.py — Viewed

```
 8  11
 9  12  def get_player(name_or_path: str, **kwargs):
10  13      player_class = globals().get(name_or_path)
11  14      if player_class:
12  15          return player_class(**kwargs)
    16  +   elif path.isdir(name_or_path):
    17  +       return MultiModelPlayer(name_or_path, **kwargs)
13  18      else:
14  19          return ModelPlayer(name_or_path, **kwargs)
```

65 players/multi_model_player.py — Viewed

```
@@ -0,0 +1,65 @@
16 + class MultiModelPlayer(Player, Observer):
17 +     def __init__(self, folder: str, device="auto", *args, **kwargs):
18 +         """
19 +         Creates a player from the latest versions of self-trained models
20 +         :param path: The path to the folder containing sub-folders for each position, relative to the root folder.
21 +         This also determines the config.
22 +         """
23 +         super().__init__(*args, **kwargs)
24 +         subfolders = os.listdir(folder)
25 +         for pos in PlayerPosition:
```

2. Tool Mock: Änderungen die semantisch zusammenhängen, werden auch zusammen in einem PR angezeigt

Types of Edit Actions

categorize(diff(codeAtTime(t1), codeAtTime(t2)))

- Single character added / removed
- Whole commit / PR changes
- Semantic AST changes: Add statement, rename identifier
- Advanced semantic changes: Extract method, Replace loop with stream

Demo Time

Challenge

Categorize edit actions to provide advanced tooling with better code insights.

Requirements

- Language Agnostic / Easily adaptable
- Completeness
- Detail on Demand
- Online

Related Work

GumTree

+ Robust
+ Configurable for additional language support
- Low level AST diff

RefMiner

+ High level abstraction
+ High precision
+ Complex Edit Actions
- Java-Only
- Other implementations exist for a very limited set of languages

Our tool should detect changes while editing, taking editing history into account.

Gumtree

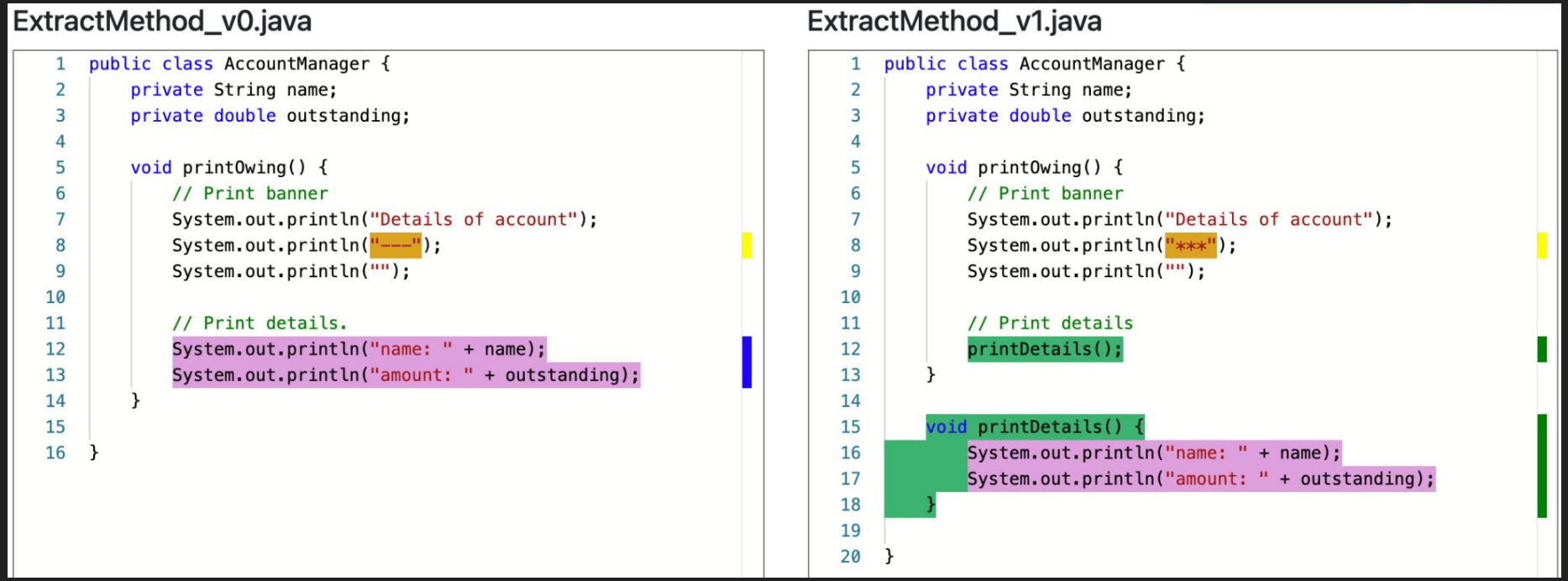
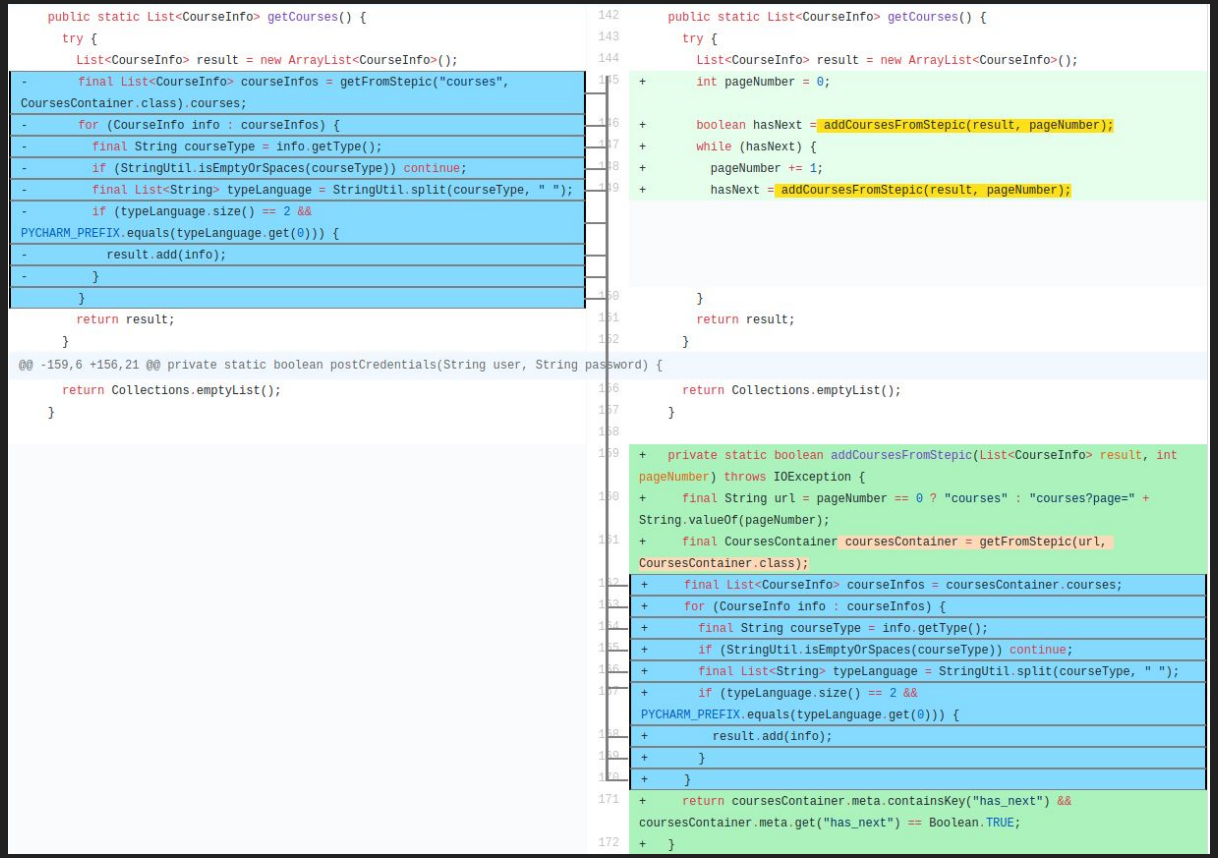


Image: https://github.com/GumTreeDiff/gumtree

RefactoringMiner

Image: https://github.com/tsantalis/RefactoringMiner

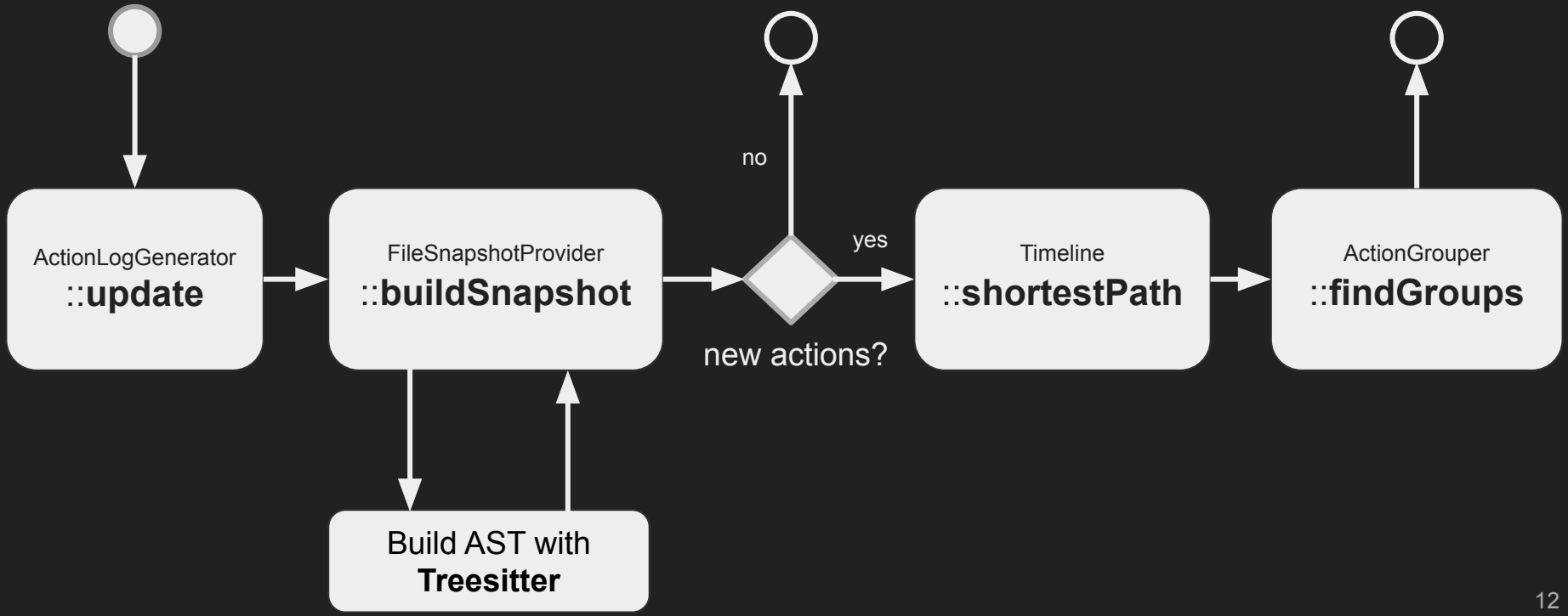


```
- public static List<CourseInfo> getCourses() {
-   try {
-     List<CourseInfo> result = new ArrayList<CourseInfo>();
-     final List<CourseInfo> courseInfos = getFromStepic("courses", CoursesContainer.class).courses;
-     for (CourseInfo info : courseInfos) {
-       final String courseType = info.getType();
-       if (StringUtil.isEmptyOrSpaces(courseType)) continue;
-       final List<String> typeLanguage = StringUtil.split(courseType, " ");
-       if (typeLanguage.size() == 2 && PYCHARM_PREFIX.equals(typeLanguage.get(0))) {
-         result.add(info);
-       }
-     }
-     return result;
-   }
@@ -159,6 +156,21 @@ private static boolean postCredentials(String user, String password) {
-   return Collections.emptyList();
- }
```

```
+ public static List<CourseInfo> getCourses() {
+   try {
+     List<CourseInfo> result = new ArrayList<CourseInfo>();
+     int pageNumber = 0;
+     boolean hasNext = addCoursesFromStepic(result, pageNumber);
+     while (hasNext) {
+       pageNumber += 1;
+       hasNext = addCoursesFromStepic(result, pageNumber);
+     }
+     return result;
+   }
+   return Collections.emptyList();
+ }
+ private static boolean addCoursesFromStepic(List<CourseInfo> result, int pageNumber) throws IOException {
+   final String url = pageNumber == 0 ? "courses" : "courses?page=" + String.valueOf(pageNumber);
+   final CoursesContainer coursesContainer = getFromStepic(url, CoursesContainer.class);
+   final List<CourseInfo> courseInfos = coursesContainer.courses;
+   for (CourseInfo info : courseInfos) {
+     final String courseType = info.getType();
+     if (StringUtil.isEmptyOrSpaces(courseType)) continue;
+     final List<String> typeLanguage = StringUtil.split(courseType, " ");
+     if (typeLanguage.size() == 2 && PYCHARM_PREFIX.equals(typeLanguage.get(0))) {
+       result.add(info);
+     }
+   }
+   return coursesContainer.meta.containsKey("has_next") && coursesContainer.meta.get("has_next") == Boolean.TRUE;
+ }
```

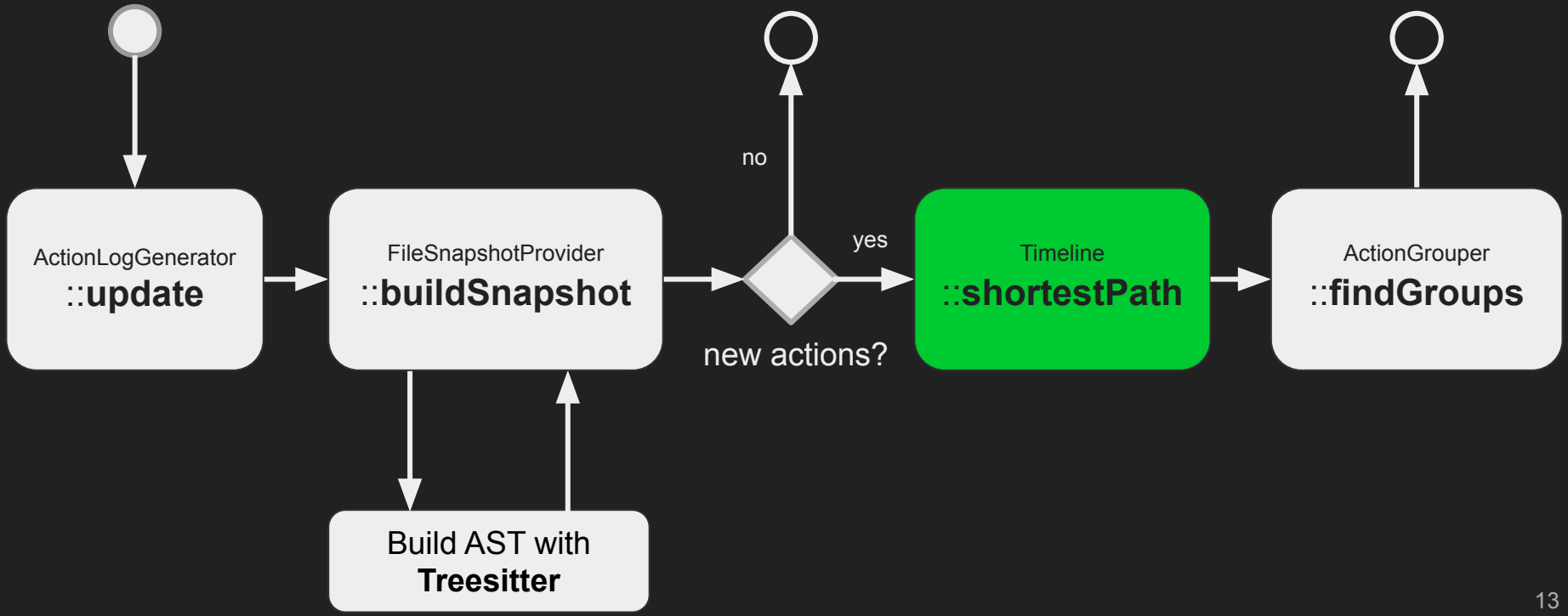
Our System

file opened | key pressed



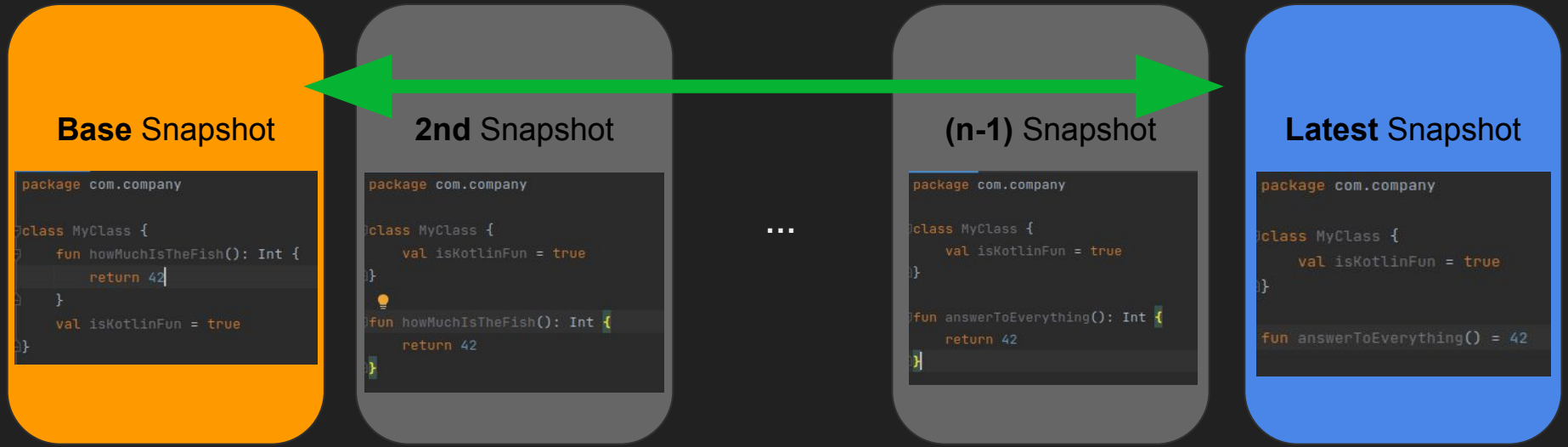
Our System

file opened | key pressed



Approach I: Latest Snapshot

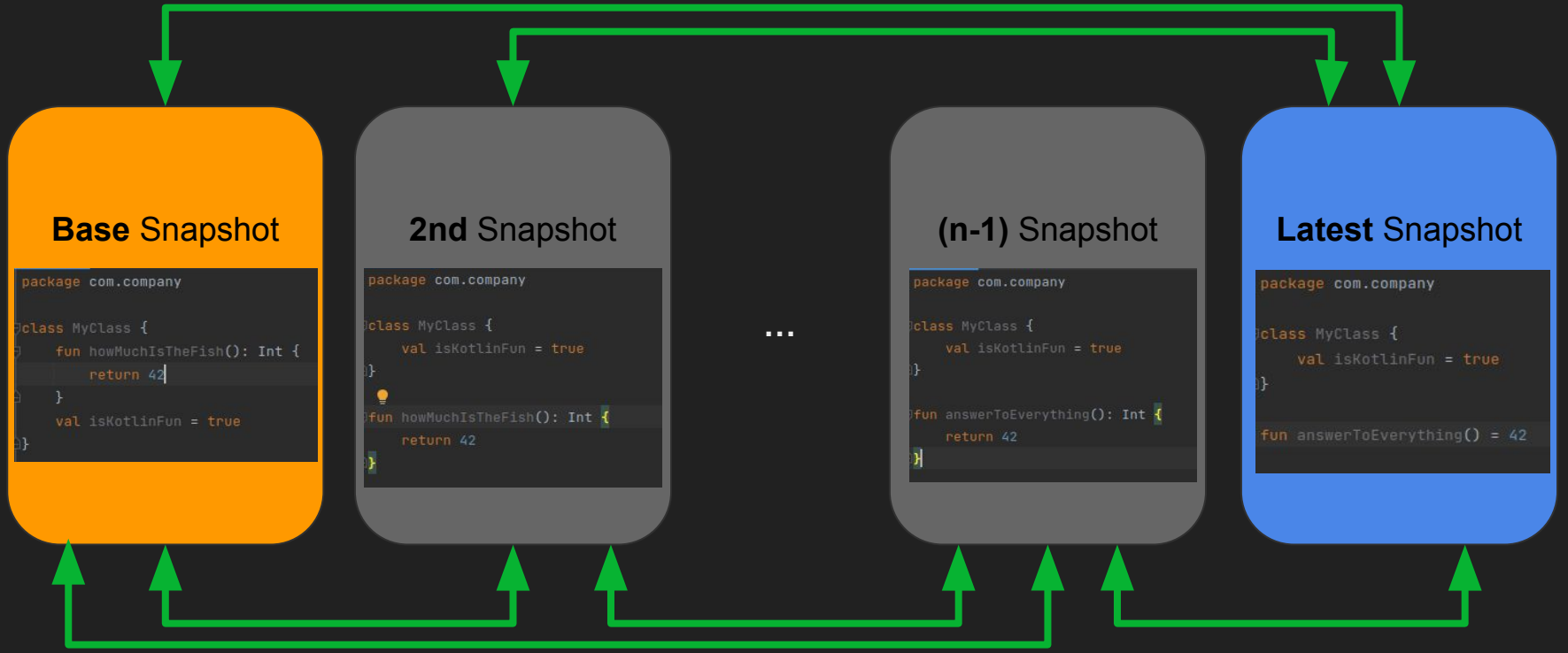
Only compare **base** snapshot with **latest**



Offline Change Detection. This is what Refactoring Miner does

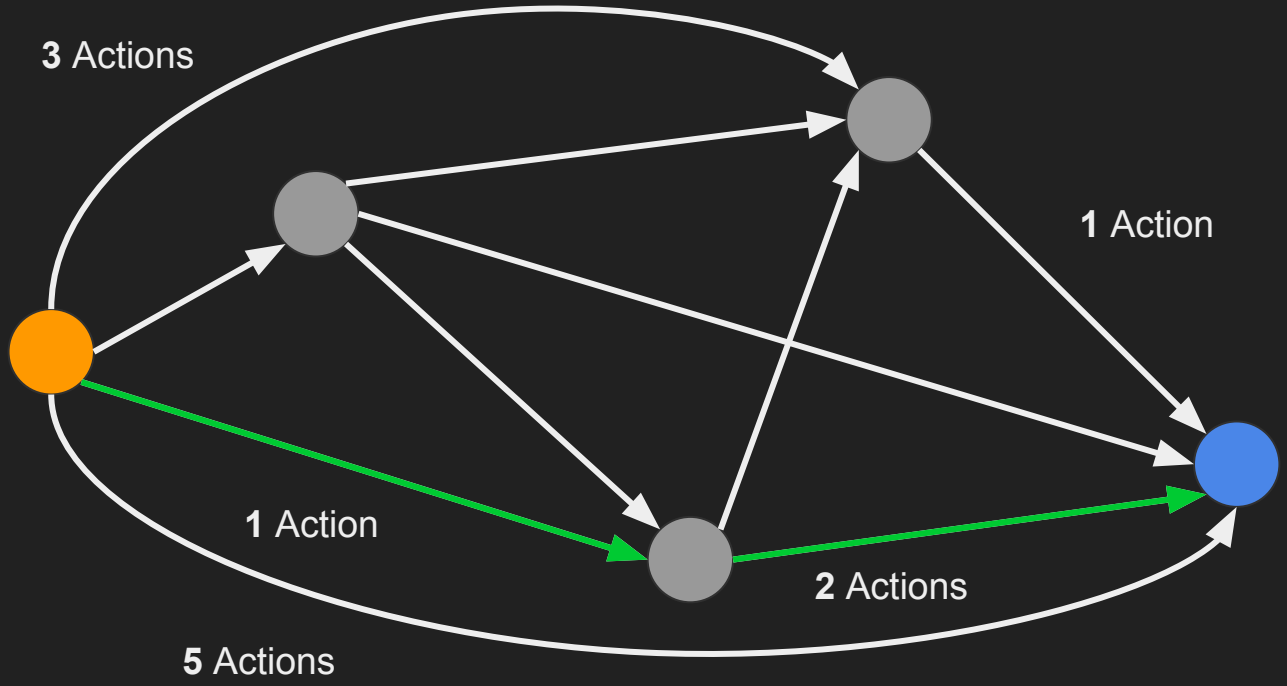
Problem: Detail Loss due to low-level AST diff. No history considered.

Approach II: Compare All Snapshots



How to choose **which** actions to display?

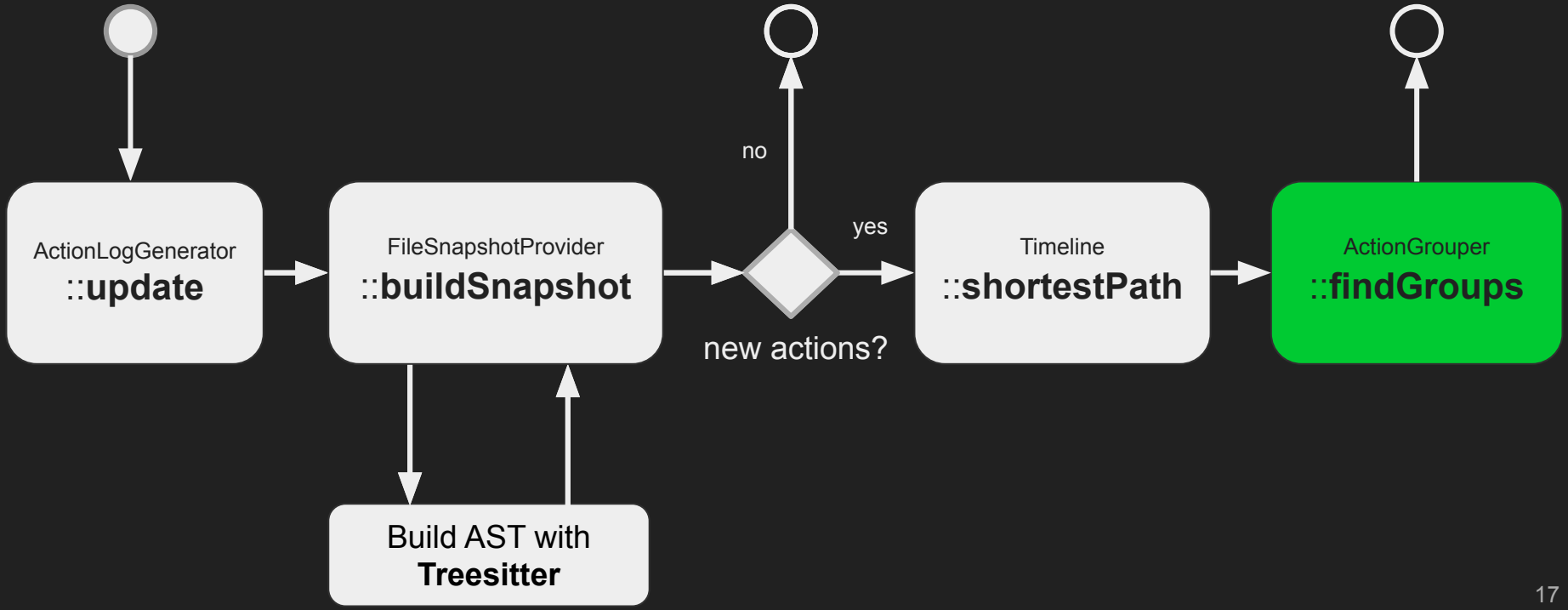
Approach II: Compare All Snapshots



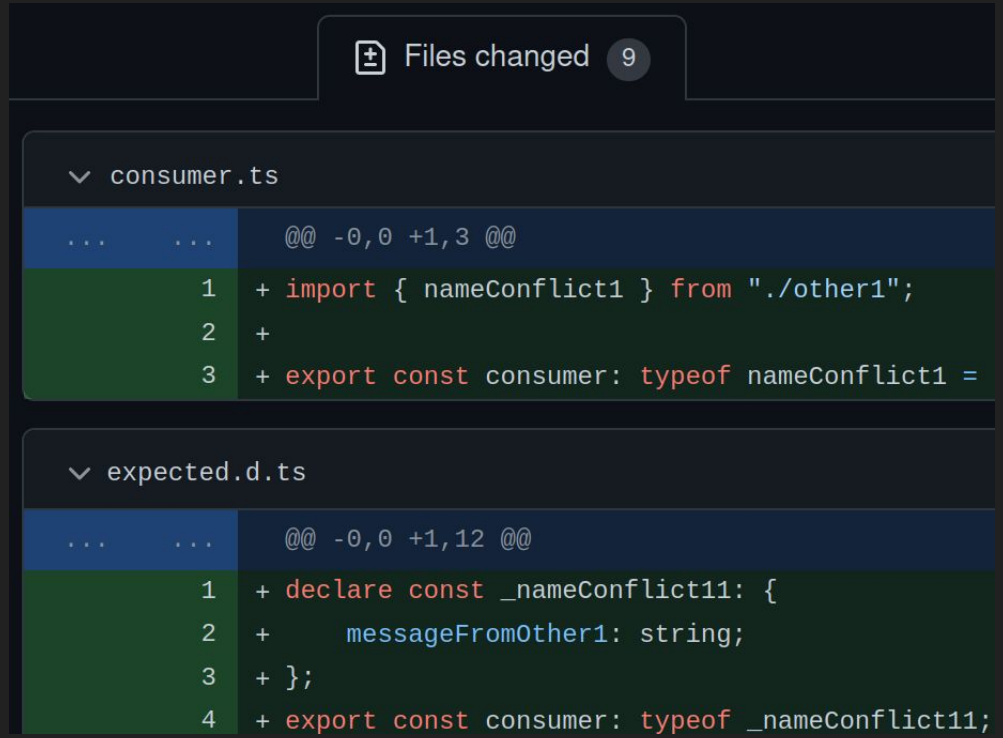
Find shortest path of **aggregated actions** through timeline graph

Our System

file opened | key pressed



Grouping (by files)

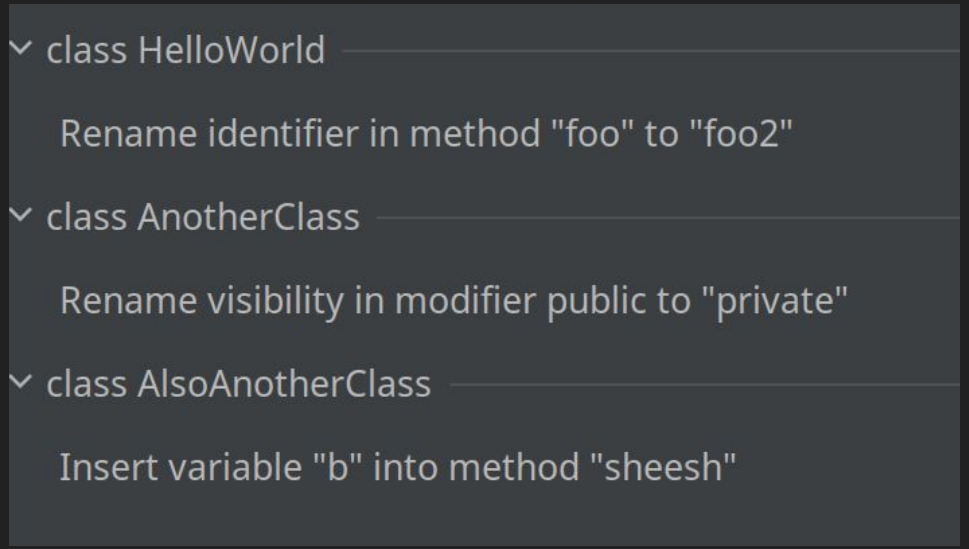


this is what github does

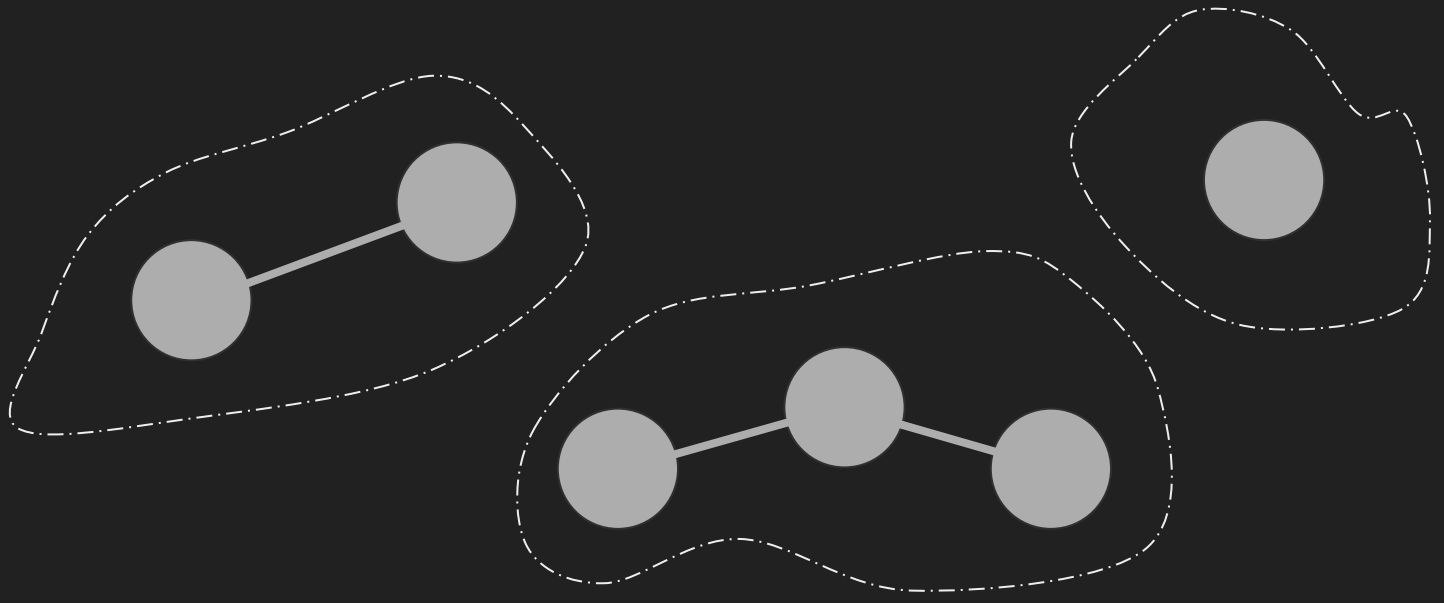
Spatial Grouping

group by:

- File
- **Class**
- Method
- Scope



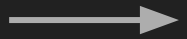
Semantic Grouping



changes grouped by semantic connection

Semantic Grouping

```
void bar() {
    foo( a: 42);
}
int foo(int a) {
    return 20 + a;
}
```



Connected Methods (foo, ...)

foo

Rename decimal integer literal in binary expression to "20"

bar

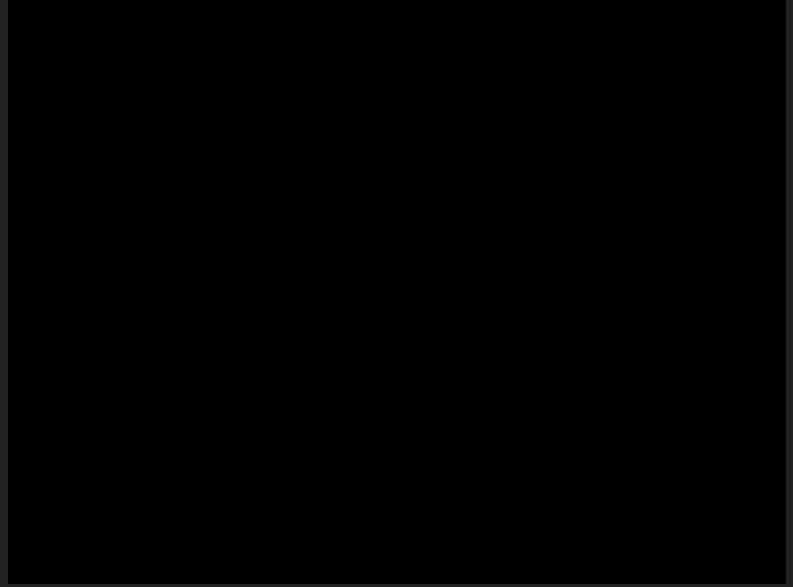
Rename decimal integer literal in argument list to "42"

heuristic: changes that are connected via a method call are in same group

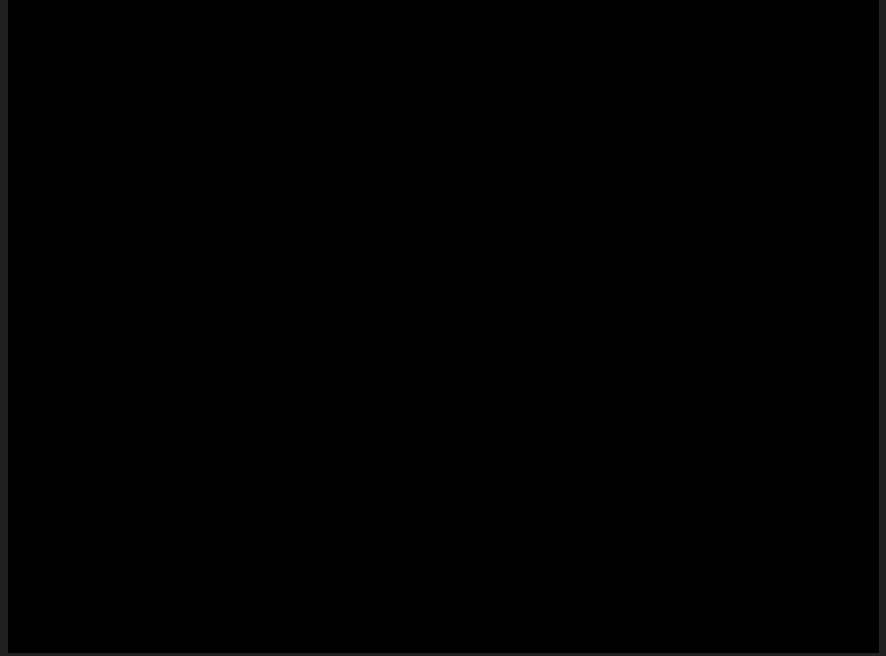
Performance

Create Snapshot (run tree-sitter):	279 ms
Find last action:	13 ms
Find all actions:	71 ms
Find shortest path:	1 ms
Group:	1 ms

30 lines of code, 15 changes



https://drive.google.com/file/d/19kSmb8ZutXcan7Rm9DgJvBxRwIrdXpOz/view?usp=sharing



https://drive.google.com/file/d/1K9E49UDsrgZnmbogShpwMR3ibXK7SipK/view?usp=sharing

Conclusion

How well did we fulfill the requirements?

- Language Agnostic / Easily adaptable
  - To add language support, just provide a treesitter config!
- Completeness
  - Based on primitive AST-operations
- Detail on Demand
  - (Expandable) Grouping
- Online
  - New snapshot at every AST change

Future Work

- Minimal additional language config to correctly name scopes
- Semantic Grouping Tool for PRs
  - Keep a log that stores information obtained via online observation
- Aggregate Actions to higher-level semantic actions e.g. “Inline Variable, Extract Method, etc…”

Conclusion

How well did we fulfill the requirements?

- Language Agnostic / Easily adaptable
  - To add language support, just provide a treesitter config!
- Completeness
  - Based on primitive AST-operations
- Detail on Demand
  - (Expandable) Grouping
- Online
  - New snapshot at every AST change