MSR course 2015

Clone and Similarity Management

Thomas Schmorleiz, May 12, 2015

Times Asia, Feb 2001

The problem

- Software is often developed as a **set of variants** to
  - meet **conflicting requirements** and
  - implement **diverse specifications**
- E.g.:
  - Legal frameworks
  - Use cases
  - Management and marketing strategies
  - Cultural specifics

Real-world example

- **Automotive Software**
- More
  - user features
  - sensors
  - electronic aids

  in higher-end models.

Running example

- **Human Resource Management System**
- **Features**
  - `Model a company` with departments and employees
  - `Total salaries` of all employees
  - `Cut salaries` of all employees
  - `Depth` of a company

Running example

- **3 target variants**
  - 1. “basic”
    - Company, Total, Cut

```haskell
-- companies as lists of departments
data Company = Company [Department]

-- departments with names, departments and employees
data Department = Department String [Department] [Employee]

-- employees with names and salaries
data Employee = Employee String Float
```

Running example

- **3 target variants**
  - 1. “basic”
    - Company, Total, Cut

```
totalSalaries :: Company -> Float
totalSalaries (Company depts) = sum$ map totalDept depts
    where
        totalDept (Department _ depts emps) =
            (sum $ map totalDept depts) + (sum $ map totalEmp emps)
        totalEmp (Employee _ s) = s
```

Running example

- **3 target variants**
  - 1. “basic”
    - Company, Total, <u>Cut</u>

```haskell
cutSalaries :: Company -> Company
cutSalaries (Company depts) = Company (map cutDept depts)
    where
        cutDept (Department n depts emps) =
            Department n (map cutDept depts) (map cutEmp emps)
        cutEmp (Employee n s) = Employee n (s / 2)
```

Running example

- **3 target variants**
  - 2. Implementation with “advanced cutting”
    - `Company`, `Total`, `Cut`
      - **Cloned** from “basic”

Running example

- **3 target variants**
  - 2. Implementation with “advanced cutting”
    - `Company, Total, Cut`
  - **Cloned** from “basic”, and modified

```
cutSalaries :: Company -> Float -> Company
cutSalaries (Company depts) factor = Company (map cutDept depts)
    where
        cutDept (Department n depts emps) =
            Department n (map cutDept depts) (map cutEmp emps)
        cutEmp (Employee n s) = Employee n (s / factor)
```

Running example

- **3 target variants**
  - 3. Implementation with “with depth”
    - `Company`, `Total`, `Cut`, `Depth`
      - **Cloned** from “basic”

Running example

- **3 target variants**
  - 2. Implementation with “with depth”
    - `Company, Total, Cut`
    - **Cloned** from “basic” (**not** “advanced cutting”), and modified

```
cutSalaries :: Company -> Float -> Company
cutSalaries (Company depts) x = Company (map cutDept depts)
    where
        cutDept (Department n depts emps) =
            Department n (map cutDept depts) (map cutEmp emps)
        cutEmp (Employee n s) = Employee n (s / x)
```

Running example

- **3 target variants**
  - 3. Implementation with “with cutting”
    - `Company, Total, Cut, Depth`

```haskell
depth :: Company -> Int
depth (Company depts) = 1 + (maximum $ map depthDept depts)
    where
        depthDept (Department _ depts emps) =
            1 + (maximum $ (map depthDept depts) ++ (map depthEmp emps))
        depthEmp e = 1
```

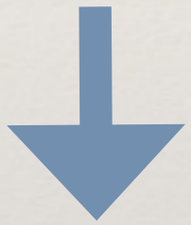
First issues

- **1.** For the Cut “advanced cutting” and “with depth” we should **establish an equality**

Evolution

- Say, the specification of the `Model` feature is updated

"Employees have a name and a salary"



"Employees have a name, a salary, and **an address**"

Evolution

- "Employees have a name, a salary, and **an address**"
- "basic" implementation co-evolves

```haskell
-- companies as lists of departments
data Company = Company [Department]

-- departments with names, departments and employees
data Department = Department String [Department] [Employee]

-- employees with names, salaries, and addresses
data Employee = Employee String Float String
```

Evolution

- “Employees have a name, a salary, and **an address**”
- “basic” implementation co-evolves

```
totalSalaries :: Company -> Float
totalSalaries (Company depts) = sum$ map totalDept depts
    where
        totalDept (Department _ depts emps) =
            (sum $ map totalDept depts) + (sum $ map totalEmp emps)
        totalEmp (Employee _ s _) = s
```

Evolution

- "Employees have a name, a salary, and **an address**"
- "basic" implementation co-evolves

```
37 cutSalaries :: Company -> Company
38 cutSalaries (Company depts) = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s a) = Employee n (s / 2) a
```

First issues

- **2.** Other implementations need to adjust too
  - For `Model` and `Total` we need to **restore equalities** to "basic"
  - For `Cut` we need to **increase similarities** to "basic"

First issues, observations

- Cloning may lead to inconsistencies and divergence
- Certain operations need to be performed to diminish such issues:
  - Establish equality
  - Restore equality
  - Increase similarity

Common approaches

- Two common approaches to develop a set of variants
  - 1. **“clone-and-own”**, copy assets to create a new variants from an existing one
    - Flexible; independent developers; non-disruptive development
    - No process; fragments diverge over time, have to synchronized manually

Common approaches

- Two common approaches to develop a set of variants
  - 2. **Product line engineering**, build a platform of shared assets
    - minimal redundancy; build-in change propagation
    - High-risk migration process; less independent developers; no proactively adoptable

Idea

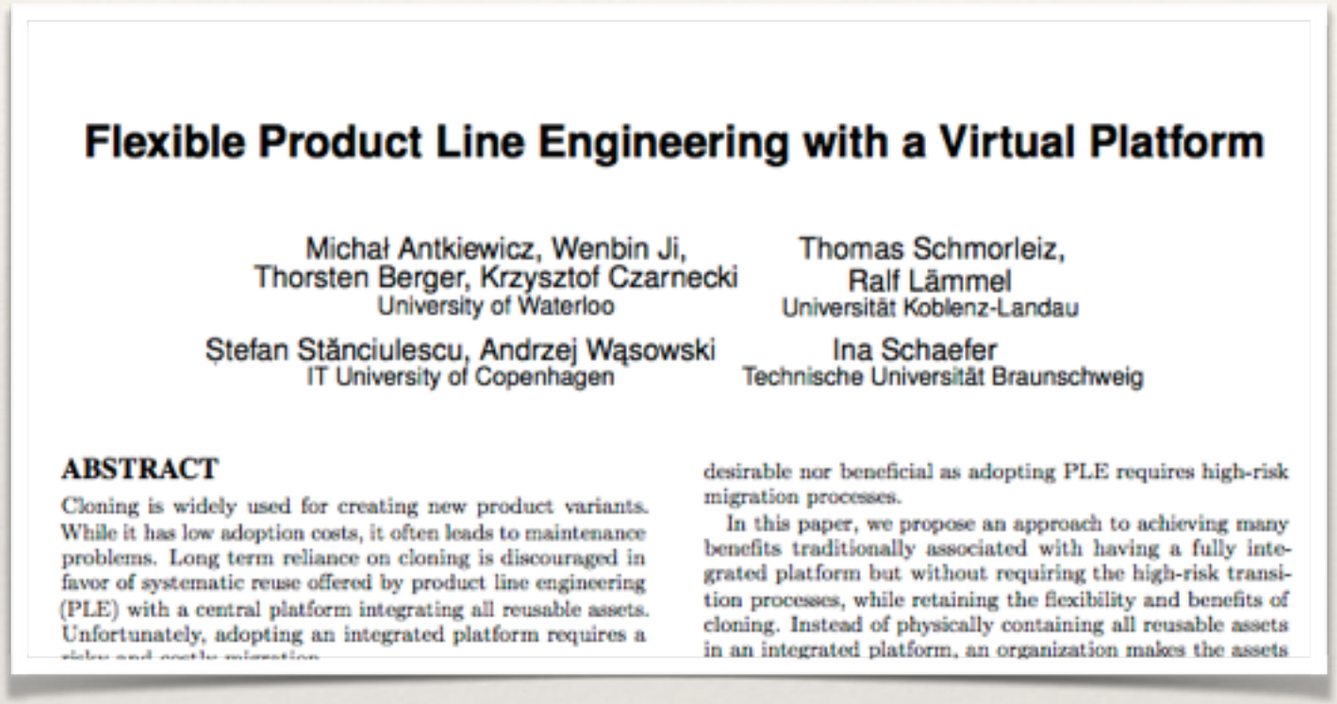
- Some middle ground is needed
- Should combine advantages of clone-and-own and PLE
- Should diminish their disadvantages
- Should allow cloning while providing means of variant, clone, <u>similarity management</u> and <u>monitoring</u>

Idea

- ***Analyze*** the similarities by extracting metadata throughout the history of a repository
- ***Annotate*** the similarities by expressing how they should be maintained further
- ***Maintain*** the similarities by automatic change propagation or manual actions

Previous work

- Virtual platform of metadata with cloning-related operator



Flexible Product Line Engineering with a Virtual Platform

Michał Antkiewicz, Wenbin Ji, Thorsten Berger, Krzysztof Czarnecki
University of Waterloo

Thomas Schmorleiz, Ralf Lämmel
Universität Koblenz-Landau

Ștefan Stănciulescu, Andrzej Wąsowski
IT University of Copenhagen

Ina Schaefer
Technische Universität Braunschweig

ABSTRACT

Cloning is widely used for creating new product variants. While it has low adoption costs, it often leads to maintenance problems. Long term reliance on cloning is discouraged in favor of systematic reuse offered by product line engineering (PLE) with a central platform integrating all reusable assets. Unfortunately, adopting an integrated platform requires a

desirable nor beneficial as adopting PLE requires high-risk migration processes.

In this paper, we propose an approach to achieving many benefits traditionally associated with having a fully integrated platform but without requiring the high-risk transition processes, while retaining the flexibility and benefits of cloning. Instead of physically containing all reusable assets in an integrated platform, an organization makes the assets

- In this context:
  - extracted metadata = virtual platform Change
  - propagation = implementation of propagate operator

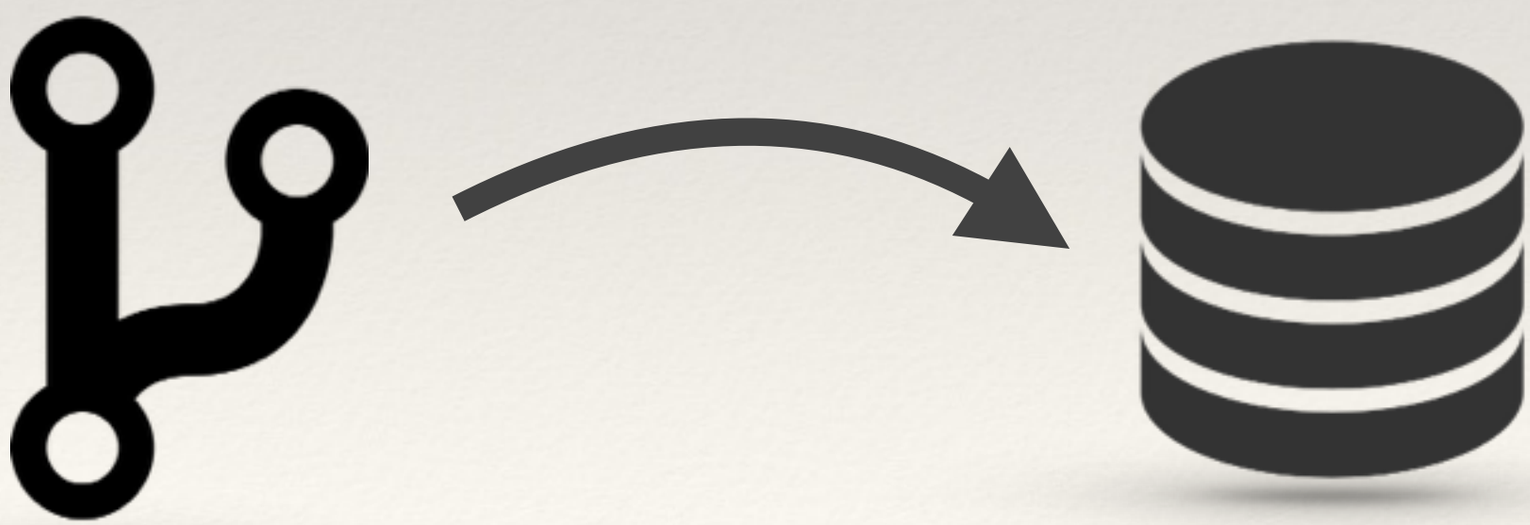
Agenda

1. Similarity analysis
2. Similarity annotations
3. Similarity management tool support
    1. Similarity exploration
    2. Annotation inference
    3. Automatic change propagation
    4. Recommending and updating annotations
    5. Manual maintenance
4. Git integration
5. Case study: *101haskell*

Similarity Analysis

Similarity analysis

- Automated process to extract metadata from the **history of a repository**
- History as **sequence of commits**
- Use **APIs** to access local repository
- Store extracted **metadata** in a **database** for later use



Extraction:

1. Variants

Variant extraction

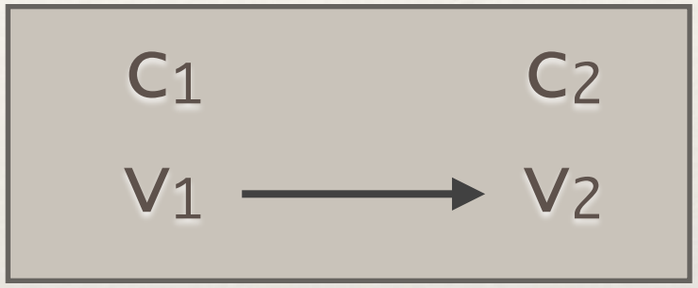
- Organizing variants
  - Dedicated repos
  - Dedicated branches in one repo
  - **Dedicated folders in one branch**

Variant extraction

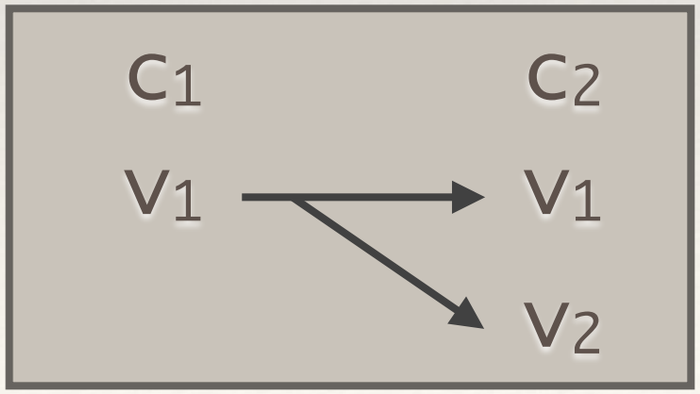
- Input: Set of possible root folders ps
- For every path p in ps, and every commit c:
  - check whether p exists at c
- For the first hit (regarding the order given by p):
  - extract names of subfolders as variant names

Variant extraction

- Can detect renaming: If
  - the majority of files in folder v_1 at c_1 where moved to v_2 at c_2
  - v_2 did not exist before, and
  - v_1 no longer exists
- Can detect splits:
  - Same as renaming but v_1 continues to exist



renaming



“split”

Implementation

- Language: Python, Git API: GitPython

```
6  from git import *
22 repo = Repo(repo_path)
```

- Iterate commits

```
37 commits = repo.iter_commits(branch_name)
```

- Extract variants

```
52 variant_extractor.extract(repo_path, variation_dirs, repo, db, commits)
```

Implementation

```
13 def extract(repo_path, variation_dirs, repo, db, commits):
```

```
104     last_commit = None
105     # extract variants from each commit
106     for commit in commits:
107         (current_renamings, current_variants) = from_commit(commit, last_commit)
108         all_variants[commit.hexsha] = current_variants
109         all_renamings[commit.hexsha] = current_renamings
110         last_commit = commit
111
112     document = {}
113     document["variants"] = all_variants
114     document["renamings"] = all_renamings
115     #print document
116     db.repos.update({"path": repo_path}, {"$set" : document})
```

Implementation

```
13 def extract(repo_path, variation_dirs, repo, db, commits):
```

```
104     last_commit = None
105     # extract variants from each commit
106     for commit in commits:
107         (current_renamings, current_variants) = from_commit(commit, last_commit)
108         all_variants[commit.hexsha] = current_variants
109         all_renamings[commit.hexsha] = current_renamings
110         last_commit = commit
111
112     document = {}
113     document["variants"] = all_variants
114     document["renamings"] = all_renamings
115     #print document
116     db.repos.update({"path": repo_path}, {"$set" : document})
```

Implementation

```
15    # find the current variation directory
16    def find_variation_dir():
17        for candidate in variation_dirs:
18            full_candidate = os.path.join(repo_path, candidate)
19            if os.path.isdir(full_candidate):
20                return candidate
21        return None
22
23    # extract all variants at commit
24    def from_commit(commit, last_commit):
25        print "> " + commit.hexsha
26        # first, checkout the commit
27        repo.git.checkout(commit.hexsha)
28        # find the current variation directory
29        variation_dir = find_variation_dir()
30        full_variation_dir = os.path.join(repo_path, variation_dir)
31        variants = filter(lambda x: x[0] != '.', os.listdir(full_variation_dir))
```

- Code for detecting renaming omitted

Implementation

```
15    # find the current variation directory
16    def find_variation_dir():
17        for candidate in variation_dirs:
18            full_candidate = os.path.join(repo_path, candidate)
19            if os.path.isdir(full_candidate):
20                return candidate
21        return None
22
23    # extract all variants at commit
24    def from_commit(commit, last_commit):
25        print "> " + commit.hexsha
26        # first, checkout the commit
27        repo.git.checkout(commit.hexsha)
28        # find the current variation directory
29        variation_dir = find_variation_dir()
30        full_variation_dir = os.path.join(repo_path, variation_dir)
31        variants = filter(lambda x: x[0] != '.', os.listdir(full_variation_dir))
```

- Code for detecting renaming omitted

Implementation

```
15    # find the current variation directory
16    def find_variation_dir():
17        for candidate in variation_dirs:
18            full_candidate = os.path.join(repo_path, candidate)
19            if os.path.isdir(full_candidate):
20                return candidate
21        return None
22
23    # extract all variants at commit
24    def from_commit(commit, last_commit):
25        print "> " + commit.hexsha
26        # first, checkout the commit
27        repo.git.checkout(commit.hexsha)
28        # find the current variation directory
29        variation_dir = find_variation_dir()
30        full_variation_dir = os.path.join(repo_path, variation_dir)
31        variants = filter(lambda x: x[0] != '.', os.listdir(full_variation_dir))
```

- Code for detecting renaming omitted

Implementation

```
15    # find the current variation directory
16    def find_variation_dir():
17        for candidate in variation_dirs:
18            full_candidate = os.path.join(repo_path, candidate)
19            if os.path.isdir(full_candidate):
20                return candidate
21        return None
22
23    # extract all variants at commit
24    def from_commit(commit, last_commit):
25        print "> " + commit.hexsha
26        # first, checkout the commit
27        repo.git.checkout(commit.hexsha)
28        # find the current variation directory
29        variation_dir = find_variation_dir()
30        full_variation_dir = os.path.join(repo_path, variation_dir)
31        variants = filter(lambda x: x[0] != '.', os.listdir(full_variation_dir))
```

- Code for detecting renaming omitted

Example

- Variants

```
{
  "7528896517eb0c42fbad6fb486e7a4866cae0cd8" : [
     "basic"
  ],
  "02fca68e0cb533e0663e3c8d0f3567e171b21116" : [
     "advanced-cutting",
     "basic"
  ],

  "f6dad913b0901f3eae767d556e44f32b8a9a723f" : [
     "advanced-cutting",
     "basic"
  ],
  "da6ad130fbf6cac3e2d21d96349fe60535b42ac5" : [
     "advanced-cutting",
     "basic",
     "with-depth"
  ]
}
```

- Renaming

```
{}
```

Extraction:

2. Fragment snapshots

Fragment snapshot extraction

- Fragment consist of **consecutive lines of code**
- It corresponds to a **hand-selected node** in the AST of the file
- Within in file, fragments are identified by a **classifier/name pair**
  - Classifiers:
    - Haskell: `function`, `data`, `type`
    - Java: `class`, `method`, `field`
  - Fact extractors developed for wide range of languages
- Fragment snapshots are bound to a commit point

Implementation

- Configuration file for extraction technology per language

```
1  {
2      "fragments-tech": {
3          "hs": {
4              "language": "Haskell",
5              "runner": "runhaskell",
6              "extractor": "technologies/HsFactExtractor/HsFactExtractor.hs",
7              "locator": "technologies/HsFragmentLocator/HsFragmentLocator.hs",
8              "line-pp": "technologies/HsTokenizer/TokenLinePrinter.hs",
9              "classifier_blacklist": []
10         },
44         …
45     }
46 }
```

Implementation

```
13 def extract(repo_path, tech_path, min_size, variation_dirs, repo, db, commits):
```

```
167     print 'Extracting from ' + str(len(commits)) +' commits.'
168
169     # lookup variations directories and variations
170     repo_data = db.repos.find({'path': repo_path})[0]
171     all_variants = repo_data['variants']
172
173     # extract fragment snapshots from each commit
177     for idx, commit in enumerate(commits):
178         from_commit(commit, variation_dirs, all_variants[commit.hexsha], db)
```

Implementation

```
99  def from_commit(commit, variation_dirs, variants, db):
100     global cache
101     result = []
102     print'> ' + commit.hexsha
103     # first, checkout the commit
104     repo.git.checkout(commit.hexsha)
105     # find the current variation directory
106     variation_dir = find_variation_dir(repo_path, variation_dirs)
107     full_variation_dir = os.path.join(repo_path, variation_dir)
108     # extract for all current variations
109     for variant in variants:
110         full_current_variation_dir = os.path.join(full_variation_dir, variant)
111         # extract for all known extensions
112         for extension in config:
113             # find all files with the current extension
114             found_files = list(find_files(
115                     path=full_current_variation_dir,
116                     match=Match(filetype='f',
117                      name='*.' + extension)
118             ))
```

Implementation

```
99  def from_commit(commit, variation_dirs, variants, db):
100     global cache
101     result = []
102     print'> ' + commit.hexsha
103     # first, checkout the commit
104     repo.git.checkout(commit.hexsha)
105     # find the current variation directory
106     variation_dir = find_variation_dir(repo_path, variation_dirs)
107     full_variation_dir = os.path.join(repo_path, variation_dir)
108     # extract for all current variations
109     for variant in variants:
110         full_current_variation_dir = os.path.join(full_variation_dir, variant)
111         # extract for all known extensions
112         for extension in config:
113             # find all files with the current extension
114             found_files = list(find_files(
115                     path=full_current_variation_dir,
116                     match=Match(filetype='f',
117                      name='*.' + extension)
118             ))
```

Implementation

```
120 for found_file in found_files:
121     rel_path = os.path.relpath(found_file, full_current_variation_dir)
122     # else extract
123     fragment_snapshots = get_fragment_snapshots(found_file, rel_path,
124         config[extension]['runner'],
125         config[extension]['extractor'],
126         config[extension]['classifier_blacklist'])
127     for idx, fragment_snapshot in enumerate(fragment_snapshots):
128         # extract line range and enrich fragment snapshot
129         line_range = get_line_range(found_file, rel_path, fragment_snapshot,
130             config[extension]['runner'],
131             config[extension]['locator'])
135         fragment_snapshots[idx]['language'] = config[extension]['language']
136         fragment_snapshots[idx]['from'] = line_range['from']
137         fragment_snapshots[idx]['to'] = line_range['to']
138         fragment_snapshots[idx]['repo_path'] = repo_path
```

Implementation

```
120 for found_file in found_files:
121     rel_path = os.path.relpath(found_file, full_current_variation_dir)
122     # else extract
123     fragment_snapshots = get_fragment_snapshots(found_file, rel_path,
124         config[extension]['runner'],
125         config[extension]['extractor'],
126         config[extension]['classifier_blacklist'])
127     for idx, fragment_snapshot in enumerate(fragment_snapshots):
128         # extract line range and enrich fragment snapshot
129         line_range = get_line_range(found_file, rel_path, fragment_snapshot,
130             config[extension]['runner'],
131             config[extension]['locator'])
135         fragment_snapshots[idx]['language'] = config[extension]['language']
136         fragment_snapshots[idx]['from'] = line_range['from']
137         fragment_snapshots[idx]['to'] = line_range['to']
138         fragment_snapshots[idx]['repo_path'] = repo_path
```

Implementation

```
120 for found_file in found_files:
121     rel_path = os.path.relpath(found_file, full_current_variation_dir)
122     # else extract
123     fragment_snapshots = get_fragment_snapshots(found_file, rel_path,
124         config[extension]['runner'],
125         config[extension]['extractor'],
126         config[extension]['classifier_blacklist'])
127     for idx, fragment_snapshot in enumerate(fragment_snapshots):
128         # extract line range and enrich fragment snapshot
129         line_range = get_line_range(found_file, rel_path, fragment_snapshot,
130             config[extension]['runner'],
131             config[extension]['locator'])
135         fragment_snapshots[idx]['language'] = config[extension]['language']
136         fragment_snapshots[idx]['from'] = line_range['from']
137         fragment_snapshots[idx]['to'] = line_range['to']
138         fragment_snapshots[idx]['repo_path'] = repo_path
```

Implementation

```
120 for found_file in found_files:
121     rel_path = os.path.relpath(found_file, full_current_variation_dir)
122     # else extract
123     fragment_snapshots = get_fragment_snapshots(found_file, rel_path,
124         config[extension]['runner'],
125         config[extension]['extractor'],
126         config[extension]['classifier_blacklist'])
127     for idx, fragment_snapshot in enumerate(fragment_snapshots):
128         # extract line range and enrich fragment snapshot
129         line_range = get_line_range(found_file, rel_path, fragment_snapshot,
130             config[extension]['runner'],
131             config[extension]['locator'])
135         fragment_snapshots[idx]['language'] = config[extension]['language']
136         fragment_snapshots[idx]['from'] = line_range['from']
137         fragment_snapshots[idx]['to'] = line_range['to']
138         fragment_snapshots[idx]['repo_path'] = repo_path
```

Implementation

```
50 def get_fragment_snapshots(abs_path, rel_path, runner, extractor_path, classifier_blacklist):
51     extractor_path = os.path.join(tp, extractor_path)
52     myinput = open(abs_path)
53     process_args = [runner, extractor_path]
54     result = Popen(process_args, stdin=myinput, stdout=PIPE).communicate()[0]
55     result = json.loads(result)
56     def collect_fragments(level):
57         if 'fragments' not in level:
58             return []
59         else:
60             fs = []
61             for f in level['fragments']:
62                 if f['classifier'] not in classifier_blacklist:
63                     f_doc = {
64                         'classifier': f['classifier'],
65                         'name': f['name']
66                     }
67                     fs.append(f_doc)
68                 fs.extend(collect_fragments(f))
69             return fs
70     return {'fragments': collect_fragments(result)}
```

Implementation

```
74 def get_line_range(abs_path, rel_path, fragment_snapshot, runner, locator_path):
75     locator_path = os.path.join(tp, locator_path)
76     fragment_snapshot_path = os.path.join(
77         fragment_snapshot['classifier'],
78         fragment_snapshot['name'])
79     myinput = open(abs_path)
80     result = Popen([runner, locator_path, fragment_snapshot_path],
81             stdin=myinput,
82             stdout=PIPE).communicate()[0]
83     return json.loads(result)
```

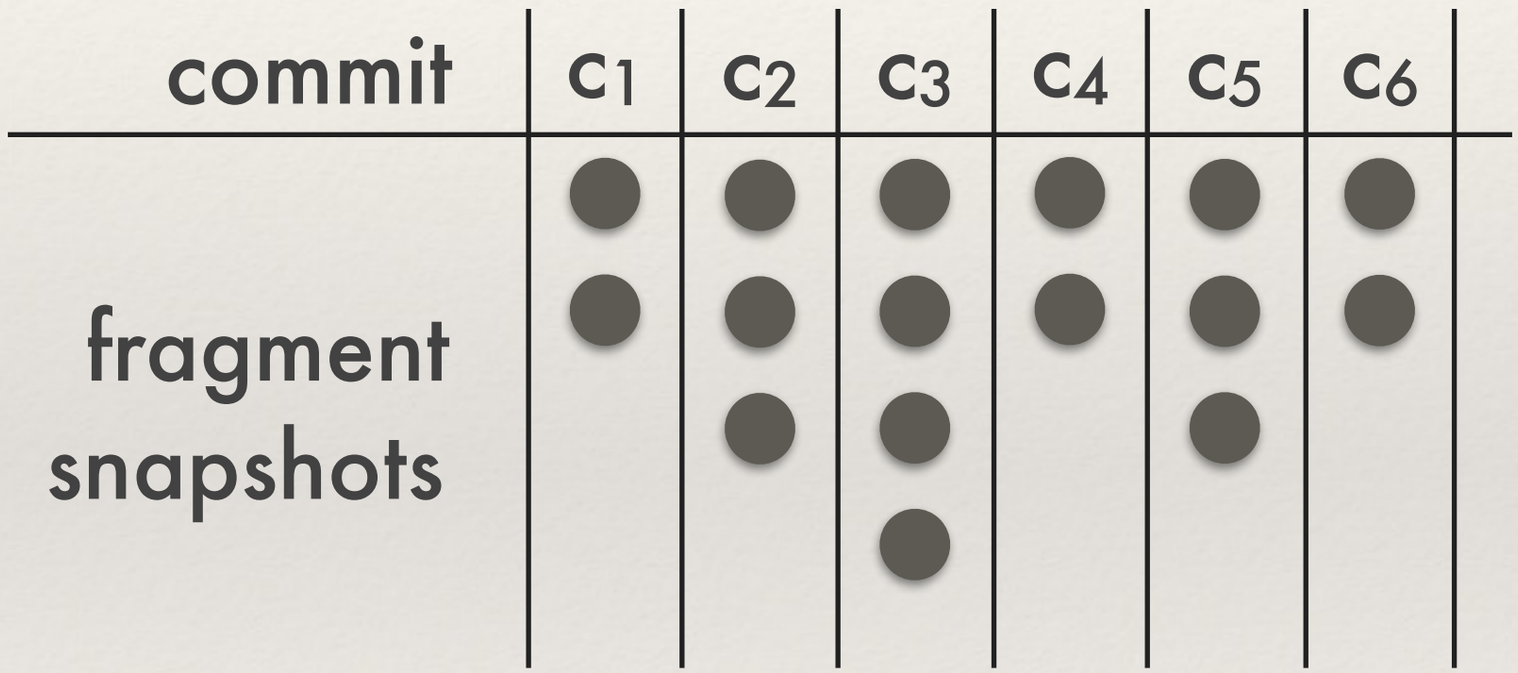
Example

- Fragment snapshot

```haskell
-- companies as lists of departments
data Company = Company [Department]
```


```
{
  "_id" : ObjectId(“554a5c6c353ec807909ba32c”),
  "variant" : “basic",
  "relative_path" : "Main.hs",
  "name" : "Company",
  "classifier" : "data",
  "from" : 1,
  "to" : 2,
  "language" : "Haskell",
  "sha" : "f6dad913b0901f3eae767d556e44f32b8a9a723f",
  "repo_path" : "/Users/tschmorleiz/Desktop/msr/implementations/"
}
```

Fragment snapshot extraction



Extraction:

3. Fragments

Fragment extraction

- Fragment snapshots are bound to commits
- A fragment consists of a **series of its snapshots**
- To extract a fragment we need to **link its snapshots**
- Need to be able to handle
  A. Renaming at variant, file, fragment level
  B. Content changes
  C. Interactions of renaming and content changes

Implementation, abstract

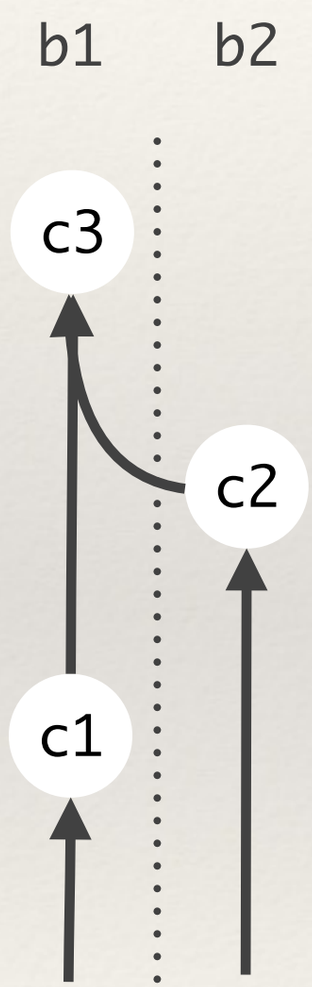
- For a fragment snapshot fs_i at commit point c_i
- Check each snapshot fs_{i-1} at c_{i-1} in the same file as fs_i:

1. No change: Link snapshots if snapshot can be found with exact same content
2. Pure renaming: Link snapshots if only the name was changed
3. Mass renaming: Link snapshots if contents match except for any name of fragments in the same file
4. Pure content editing: Link snapshots if classifier and name matches

Fragment extraction

- Order of cases is used when trying to find a linkable fragment and expresses priority
- With each case comes a set of flags for the linked snapshots
  - `is_new`
  - `is_changed`
  - `is_renamed`
- If no snapshot to link to can be found, extend search space to variant level
- If still no linkable snapshot can be found, snapshot marks the creation of a fragment

Commit traversal



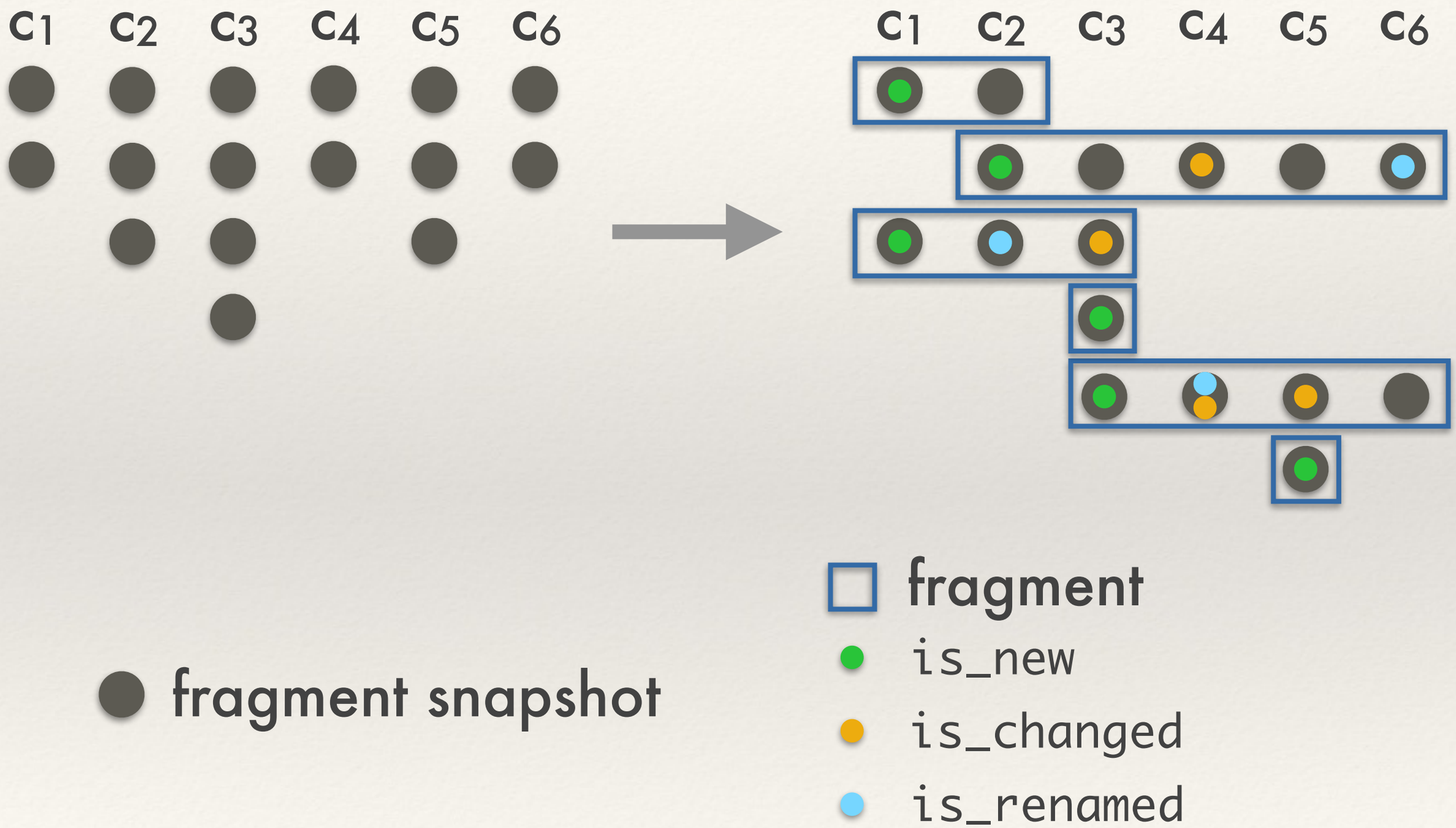
- We do not consider branches as a means of organizing variants
- However we consider them when traversing
- Scenario: fragment f was added at c1
- At c3 and fragment snapshot fs_{c3}:
  - Just looking at c2 (last commit via timestamp) fails
  - Look at both c1 and c2 for fragment snapshots to link to fs_{c3}!

Example

- Fragment

```
{
  "_id" : ObjectId("554a5bdf353ec805268f2ef1"),
  "fragment_snapshot_ids" : [
      ObjectId("554a5bd9353ec805268f2ec4"),
      ObjectId("554a5bdc353ec805268f2ed0"),
      ObjectId("554a5bdc353ec805268f2edc")
  ],
  "repo_path" : "/Users/tschmorleiz/Desktop/msr/implementations/"
}
```

Fragment extraction



Extraction:

4. Similarity Snapshots

Similarity snapshot extraction

- A similarity snapshot is a **relation** of **<u>two fragment snapshot</u>** at a **commit point**
- As a **similarity measure** we use "diff ratio" after pretty-printing tokens into many lines (proposed by Cordy et al.)
- We therefore disregard similarities solely based on whitespace
- To reduce number of relations stored, user provides a **similarity threshold**

Pretty printing

- We implemented a token pretty printer various languages

Input

```
-- companies as lists of departments
data Company = Company [Department]
```

PrettyPrinter

```
main = getContents >>= (putStrLn.ppTokens.lexerPass0)

ppTokens :: [PosToken] -> String
ppTokens pts = join "\n" $ map (snd.snd) $ filter ((/=Whitespace).fst) pts
```

Output

```
--
 companies as lists of departments

data
Company
=
Company
[
Department
]
```

Pretty Printing

- Configuration file for pretty printing technology per language

```
1  {
2      "fragments-tech": {
3          "hs": {
4              "language": "Haskell",
5              "runner": "runhaskell",
6              "extractor": "technologies/HsFactExtractor/HsFactExtractor.hs",
7              "locator": "technologies/HsFragmentLocator/HsFragmentLocator.hs",
8              "line-pp": "technologies/HsTokenizer/TokenLinePrinter.hs",
9              "classifier_blacklist": []
10         },
44         …
45     }
46 }
```

Implementation

```
152 def from_commit(commit, all_fragment_snapshots, db, fragmentTracks, track_lookup):
153     print '> ' + commit.hexsha
154     variation_dir = find_variation_dir(repo_path, variation_dirs)
155     # lookup current snapshots
156     fragment_snapshots = filter(lambda f: f['sha'] == commit.hexsha, all_fragment_snapshots)
157     print ' > ' + str(len(fragment_snapshots)) + ' fragments.'
158     # if either snapshot marks a change, compute diff ratio, otherwise copy
159     for fragment_snapshot in fragment_snapshots:
160         if fragment_snapshot['is_new'] or fragment_snapshot['is_changed']:
161             for other_fragment_snapshot in fragment_snapshots:
162                 from_pair(fragment_snapshot, other_fragment_snapshot, variation_dirs)
163         else:
164             for other_fragment_snapshot in fragment_snapshots:
165                 if other_fragment_snapshot['is_new'] or other_fragment_snapshot['is_changed']:
166                     from_pair(fragment_snapshot, other_fragment_snapshot, variation_dirs)
167                 else:
168                     copy_from_pair(...)
```

Implementation

```
152 def from_commit(commit, all_fragment_snapshots, db, fragmentTracks, track_lookup):
153     print '> ' + commit.hexsha
154     variation_dir = find_variation_dir(repo_path, variation_dirs)
155     # lookup current snapshots
156     fragment_snapshots = filter(lambda f: f['sha'] == commit.hexsha, all_fragment_snapshots)
157     print ' > ' + str(len(fragment_snapshots)) + ' fragments.'
158     # if either snapshot marks a change, compute diff ratio, otherwise copy
159     for fragment_snapshot in fragment_snapshots:
160         if fragment_snapshot['is_new'] or fragment_snapshot['is_changed']:
161             for other_fragment_snapshot in fragment_snapshots:
162                 from_pair(fragment_snapshot, other_fragment_snapshot, variation_dirs)
163         else:
164             for other_fragment_snapshot in fragment_snapshots:
165                 if other_fragment_snapshot['is_new'] or other_fragment_snapshot['is_changed']:
166                     from_pair(fragment_snapshot, other_fragment_snapshot, variation_dirs)
167                 else:
168                     copy_from_pair(...)
```

Implementation

```
152 def from_commit(commit, all_fragment_snapshots, db, fragmentTracks, track_lookup):
153     print '> ' + commit.hexsha
154     variation_dir = find_variation_dir(repo_path, variation_dirs)
155     # lookup current snapshots
156     fragment_snapshots = filter(lambda f: f['sha'] == commit.hexsha, all_fragment_snapshots)
157     print ' > ' + str(len(fragment_snapshots)) + ' fragments.'
158     # if either snapshot marks a change, compute diff ratio, otherwise copy
159     for fragment_snapshot in fragment_snapshots:
160         if fragment_snapshot['is_new'] or fragment_snapshot['is_changed']:
161             for other_fragment_snapshot in fragment_snapshots:
162                 from_pair(fragment_snapshot, other_fragment_snapshot, variation_dirs)
163         else:
164             for other_fragment_snapshot in fragment_snapshots:
165                 if other_fragment_snapshot['is_new'] or other_fragment_snapshot['is_changed']:
166                     from_pair(fragment_snapshot, other_fragment_snapshot, variation_dirs)
167                 else:
168                     copy_from_pair(...)
```

Implementation

```
113 def from_pair(fragment_snapshot, other_fragment_snapshot, variation_dirs):
114     if fragment_snapshot['_id'] == other_fragment_snapshot['_id']:
115         return None
116 
117     ratio = diff_ratio(fragment_snapshot, other_fragment_snapshot, variation_dirs)
118 
119     document = {}
120     document['sha'] = commit.hexsha
121     document['repo_path'] = rp
122     document['source_id'] = other_fragment_snapshot['_id']
123     document['source_variant'] = other_fragment_snapshot['variant']
124     document['target_id'] = fragment_snapshot['_id']
125     document['target_variant'] = fragment_snapshot['variant']
126 
127     db.similaritySnapshots.insert(document)
```

Implementation

```
113 def from_pair(fragment_snapshot, other_fragment_snapshot, variation_dirs):
114     if fragment_snapshot['_id'] == other_fragment_snapshot['_id']:
115         return None
116
117     ratio = diff_ratio(fragment_snapshot, other_fragment_snapshot, variation_dirs)
118
119     document = {}
120     document['sha'] = commit.hexsha
121     document['repo_path'] = rp
122     document['source_id'] = other_fragment_snapshot['_id']
123     document['source_variant'] = other_fragment_snapshot['variant']
124     document['target_id'] = fragment_snapshot['_id']
125     document['target_variant'] = fragment_snapshot['variant']
126
127     db.similaritySnapshots.insert(document)
```

Implementation

```
86 def diff_ratio(f1, f2, variation_dirs):
87     content_f1 = get_fragment_snapshot_content(f1, variation_dirs)
88     content_f2 = get_fragment_snapshot_content(f2, variation_dirs)
89
90     return difflib.SequenceMatcher(None, content_f1, content_f2).ratio()
```

Implementation

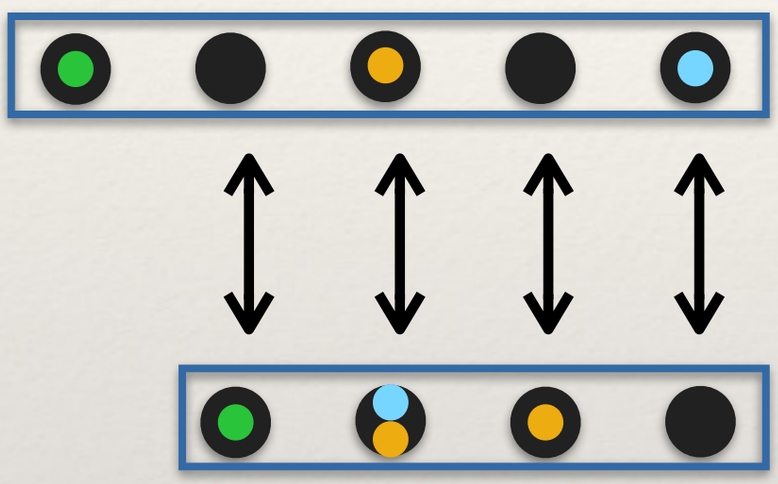
```
63 def get_fragment_snapshot_content(fragment, variation_dirs):
64     checkout(fragment['sha'])
65     variation_dir = find_variation_dir(rp, variation_dirs)
66     path = os.path.join(repo_path, variation_dir, fragment['variant'], fragment['relative_path'])
67     extension = os.path.splitext(fragment['relative_path'])[1][1:]
68     with open(path) as f:
69         content = ''.join(f.readlines()[(fragment['from'] - 1):fragment['to']])
70         if extension in config and 'line-pp' in config[extension]:
71             temp_f = tempfile.NamedTemporaryFile(delete=False)
72             temp_f.write(content)
73             temp_f.close()
74             line_pp_path = os.path.join(tp, config[extension]['line-pp'])
75             result = Popen([config[extension]['runner'], line_pp_path],
76                 stdin=open(temp_f.name), stdout=PIPE).communicate()[0]
77
78             content = result
79             os.unlink(temp_f.name)
80
81         return content
```

Example

- Similarity snapshot

```
{
  "_id" : ObjectId("554a5caf353ec807909ba36a"),
  "target_id" : ObjectId(“554a5c6c353ec807909ba331"),
  "source_id" : ObjectId(“554a5c6c353ec807909ba32b"),
  "diff_ratio" : 0.9425742574257425,
  "sha" : "f6dad913b0901f3eae767d556e44f32b8a9a723f",
  "repo_path" : "/Users/tschmorleiz/Desktop/msr/implementations/"
}
```

Similarity snapshot extraction



fragment

similarity snapshot

Extraction:

4. Similarity Evolutions

Similarity evolution extraction

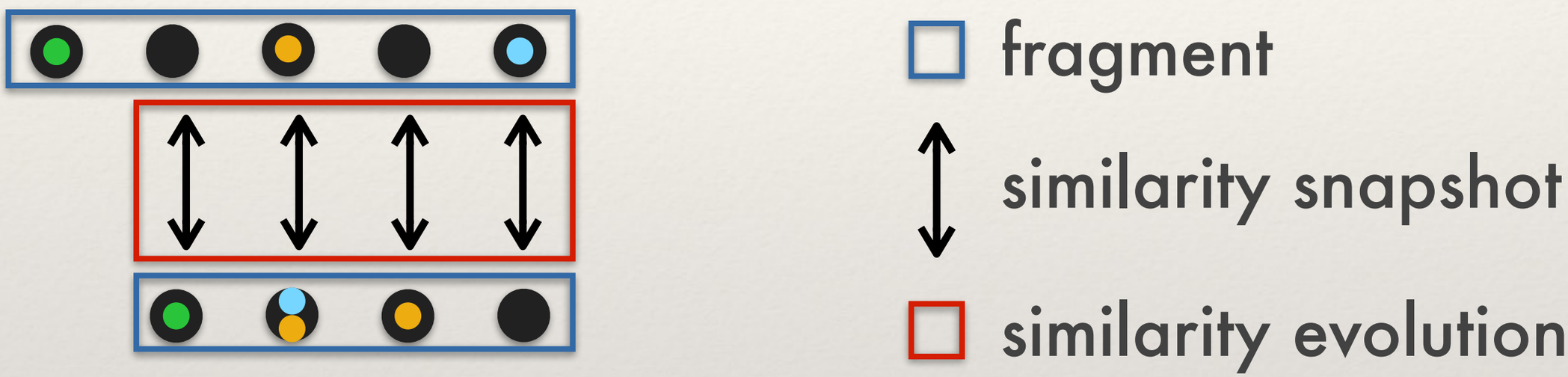
- Similarity evolutions are series of similarity snapshots
- To extract an evolution we extract similarity snapshots between respective fragment snapshots that all belong to the same fragment

Example

- Similarity evolution

```
{
  "_id" : ObjectId("554a5cb0353ec807909ba395"),
  "similarity_ids" : [
     ObjectId("554a5caf353ec807909ba35f"),
     ObjectId("554a5caf353ec807909ba36b"),
     ObjectId("554a5cb0353ec807909ba37d")
  ],
  "first_diff_ratio" : 1,
  "last_diff_ratio" : 0.9425742574257425,
  "min_diff_ratio" : 0.9425742574257425,
  "max_diff_ratio" : 1
}
```

Similarity evolution extraction



Similarity evolution categorization

- To later compute the applicability of annotations for similarity evolutions, we assign each evolution a category:
  - Always Equal holds similarity evolutions that have always been equalities, that is, the similarity value was always 1.
  - Converge to Equal holds similarity evolutions where the similarity value was once below 1 but then increased to 1 at the HEAD of the branch.
  - Diverge from Equal holds similarity evolutions where the similarity value was once 1 but then decreased to below 1 at the HEAD of the branch.
  - Always Non-equal holds similarity evolutions where the similarity value was constant or changing but always below 1.

Implementation

```python
def get_category(last_diff_ratio, min_diff_ratio, max_diff_ratio):
    if min_diff_ratio == 1 and max_diff_ratio == 1:
        return 'always_equal'
    if min_diff_ratio < 1 and last_diff_ratio == 1:
        return 'converge_to_equal'
    if max_diff_ratio == 1 and last_diff_ratio < 1:
        return 'diverge_from_equal'
    if max_diff_ratio < 1:
        return 'always_nonequal'
```

Extraction:

Updating metadata

Similarity evolution categorization

- For every new commit we have to update the metadata:
  - Extract new variant names
  - Extract new fragment snapshots
  - Create new or extend existing fragments
  - Extract new similarity snapshots
  - Create new or extend existing similarity evolutions

Similarity Annotations

Annotation categories

- We identify the following seven categories, each stating a **maintenance invariant** or **task** for the similarity
  - **Invariants**
    - *Maintain Equality* by automatic three-way merge when one or both fragments of the similarity evolution change and potentially by manual conflict resolution.

Maintain Equality

In “basic”:

```
4 -- departments with names, departments and employees
5 data Department = Department String [Department] [Employee]
```

Maintain Equality

In “advanced-cutting”:

```
4 -- departments with names, departments and employees
5 data Department = Department String [Department] [Employee]
```

Annotation categories

- **Invariants**
  - *Maintain Similarity* by manual actions when the similarity changes.

Maintain Similarity

In “basic”:

```
37 cutSalaries :: Company -> Company
38 cutSalaries (Company depts) = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s) = Employee n (s / 2)
```

Maintain Similarity

In “advanced-cutting”:

```
37 cutSalaries :: Company -> Float -> Company
38 cutSalaries (Company depts) factor = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s) = Employee n (s / factor)
```

Annotation categories

- **Invariants**
  - *Ignore Similarity* by not reporting the similarity anymore to the user.

Ignore Similarity

In “basic”:

```
30 totalSalaries :: Company -> Float
31 totalSalaries (Company depts) = sum $ map totalDept depts
32     where
33         totalDept (Department _ depts emps) =
34             (sum $ map totalDept depts) + (sum $ map totalEmp emps)
35         totalEmp (Employee _ s _) = s
```

Ignore Similarity

In “basic”:

```
37 cutSalaries :: Company -> Company
38 cutSalaries (Company depts) = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s a) = Employee n (s / 2) a
```

Annotation categories

- **Tasks**
  - *Restore Equality* by automatically propagating changes from one fragment to the other if a direction of propagation is defined, otherwise by automatic three-way merge and potentially by manual conflict resolution.

Restore Equality

In “basic” after adjusting to new spec:

```
7 -- employees with names, salaries, and addresses
8 data Employee = Employee String Float String
```

Restore Equality

In “advanced-cutting”:

```
7 -- employees with names and salaries
8 data Employee = Employee String Float
```

Annotation categories

- **Tasks**
  - *Establish Equality* by manual actions on fragments that have never been equal.

Establish Equality

In “advanced-cutting”:

```
37 cutSalaries :: Company -> Float -> Company
38 cutSalaries (Company depts) factor = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s) = Employee n (s / factor)
```

In “with-depth”:

Establish Equality

```
37 cutSalaries :: Company -> Float -> Company
38 cutSalaries (Company depts) x = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s) = Employee n (s / x)
```

Annotation categories

- **Tasks**
  - *Increase Similarity* by manual actions.

Maintain Similarity

In “basic” after adjusting to new spec:

```
37 cutSalaries :: Company -> Company
38 cutSalaries (Company depts) = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s a) = Employee n (s / 2) a
```

Increase Similarity

In “advanced-cutting”:

```
37 cutSalaries :: Company -> Float -> Company
38 cutSalaries (Company depts) factor = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s) = Employee n (s / factor)
```

Annotation categories

- **Tasks**
  - *Restore Similarity* by manual actions until a target similarity value is reached.

Maintain Similarity

In “basic”:

```
37 cutSalaries :: Company -> Company
38 cutSalaries (Company depts) = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s) = Employee n (s / 2)
```

Maintain Similarity

In “advanced-cutting”:

```
37 cutSalaries :: Company -> Float -> Company
38 cutSalaries (Company depts) factor = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s) = Employee n (s / factor)
```

Maintain Similarity

In “basic” after adjusting to new spec:

```
37 cutSalaries :: Company -> Company
38 cutSalaries (Company depts) = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s a) = Employee n (s / 2) a
```

Restore Similarity

In “advanced-cutting”:

```
37 cutSalaries :: Company -> Float -> Company
38 cutSalaries (Company depts) factor = Company (map cutDept depts)
39     where
40         cutDept (Department n depts emps) =
41             Department n (map cutDept depts) (map cutEmp emps)
42         cutEmp (Employee n s) = Employee n (s / factor)
```

Annotation categories

- **Tasks**
  - *Remove Equality* by manually making fragments unequal.

Annotation applicabilities

	MAINTAIN EQUALITY	RESTORE EQUALITY	ESTABLISH EQUALITY	MAINTAIN SIMILARITY	RESTORE SIMILARITY	INCREASE SIMILARITY	IGNORE
ALWAYS EQUAL	AUTO	X	X	X	X	X	AUTO
CONVERGE TO EQUAL	AUTO	X	X	X	X	X	AUTO
DIVERGE FROM EQUAL	X	AUTO	X	MANUAL	MANUAL	MANUAL	AUTO
ALWAYS NON-EQUAL	X	X	MANUAL	MANUAL	MANUAL	MANUAL	AUTO

Annotation structure

- *similarity*: reference to the annotated similarity
- *category:* Name of the category assigned by the user or by updating the category automatically
- *intent:* User comment about the annotation, can be about who should perform manual tasks, what to change manually, etc.
- *direction:* Only for category `Restore Equality`, specifies which fragments changes should be overwritten by automatic change propagation

Similarity Management Tool Support

Similarity Exploration

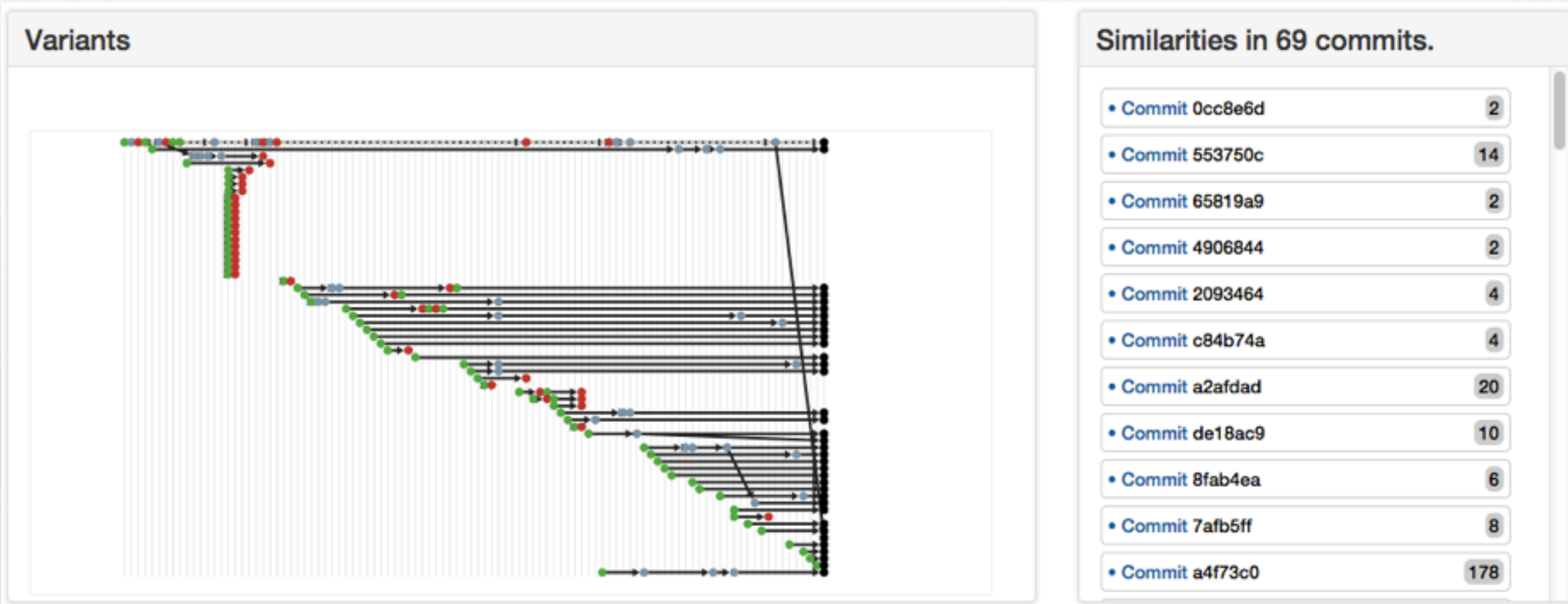
Annotators

- To provide two different annotators
- Usage depends on how the user want to explore the similarities

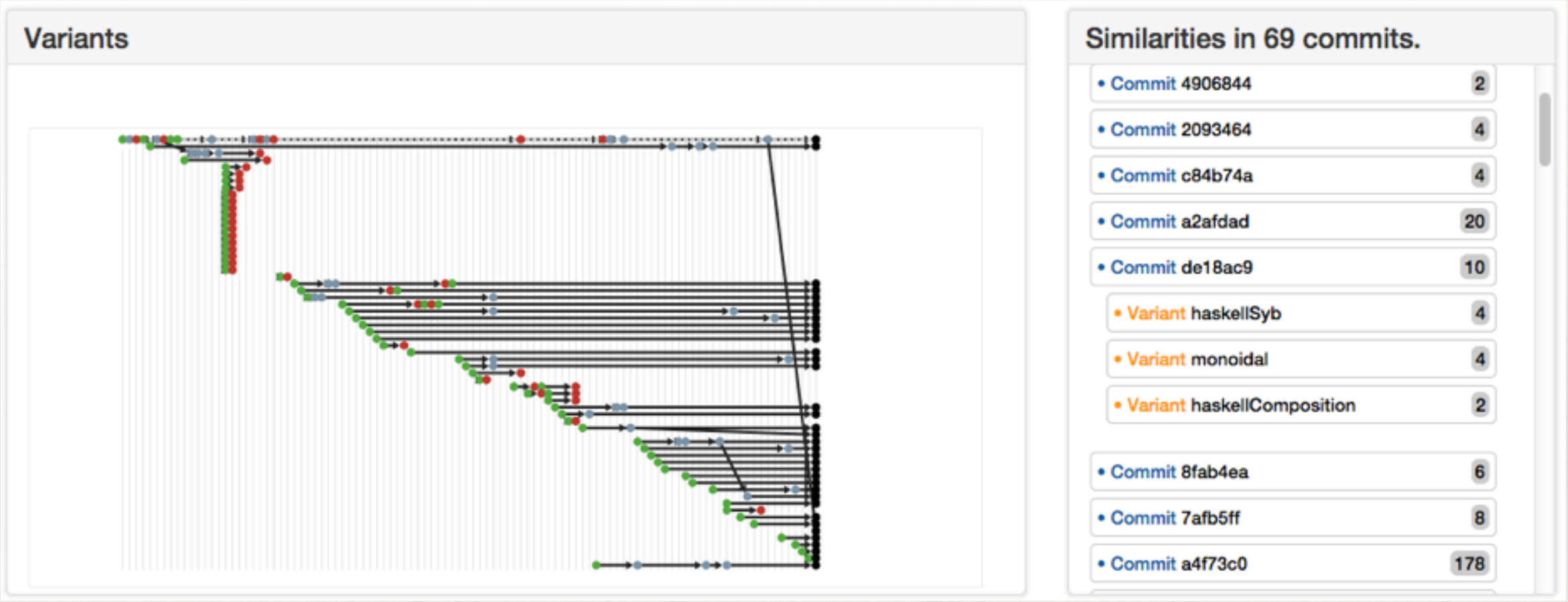
Commit-centric annotator

- Tailored for the use case where commits should be inspected one by one

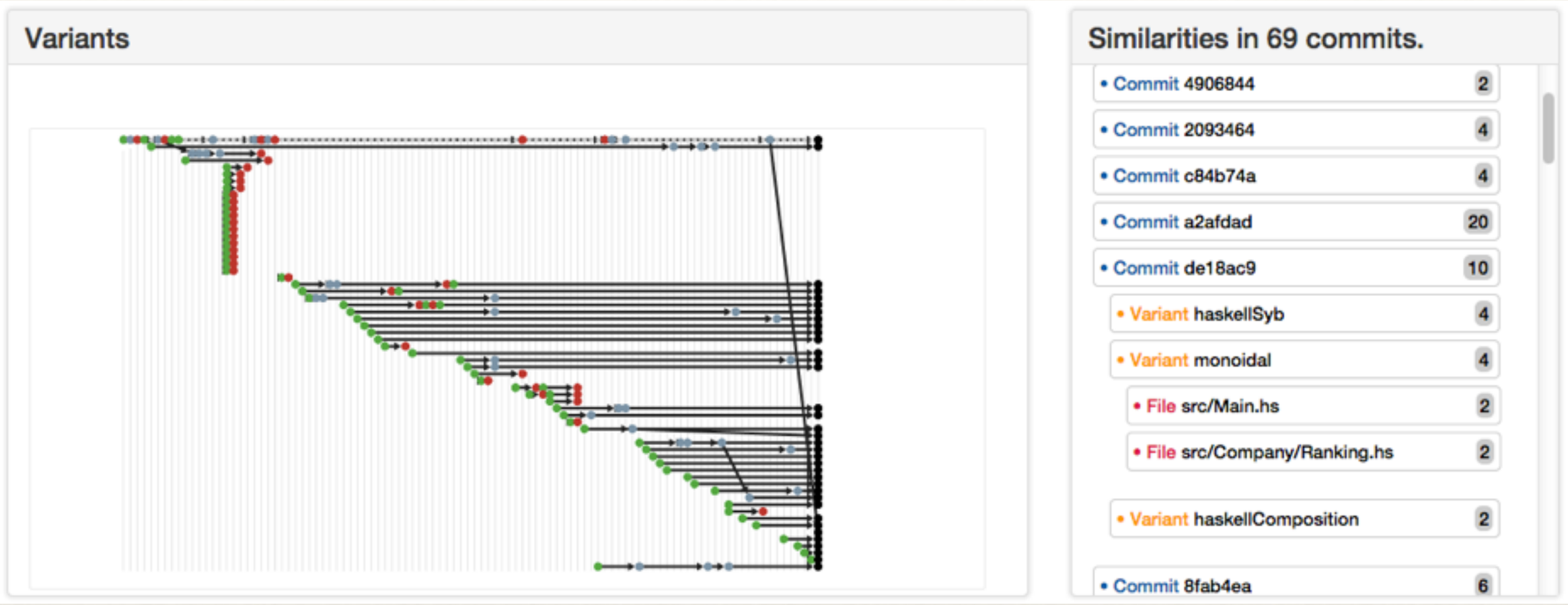
Commit-centric annotator



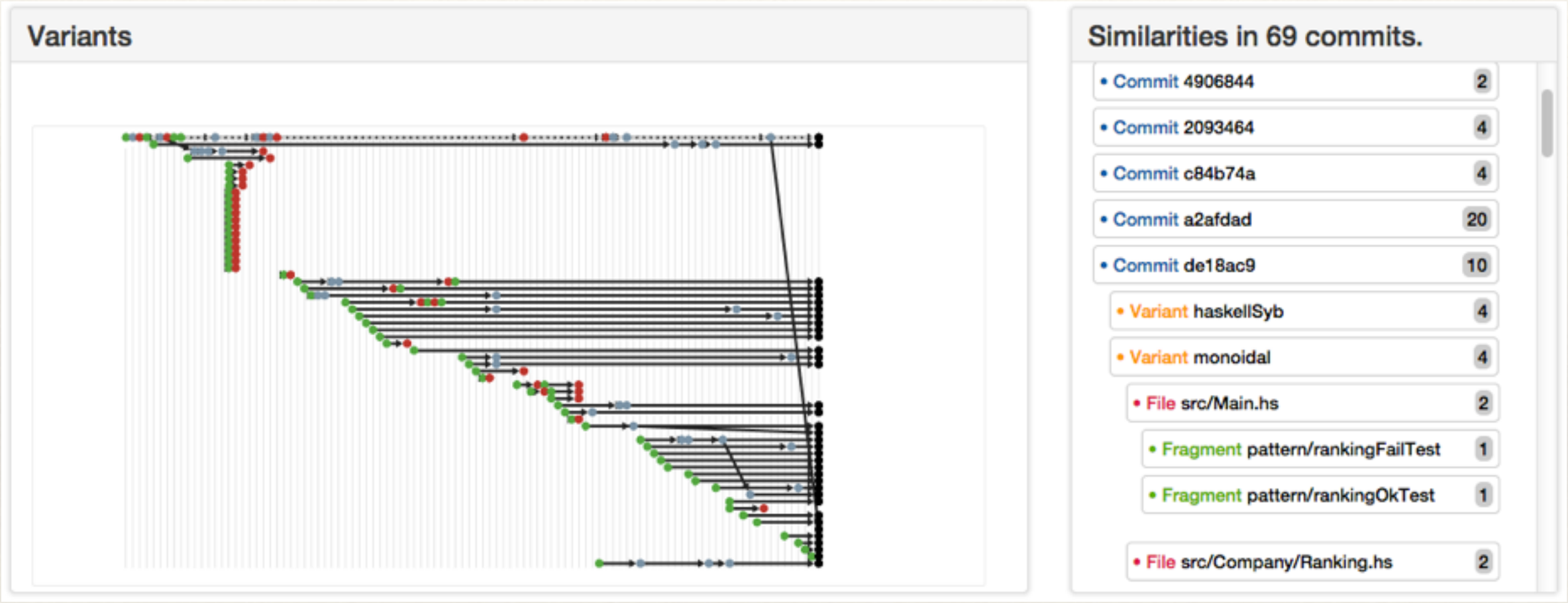
Commit-centric annotator



Commit-centric annotator



Commit-centric annotator

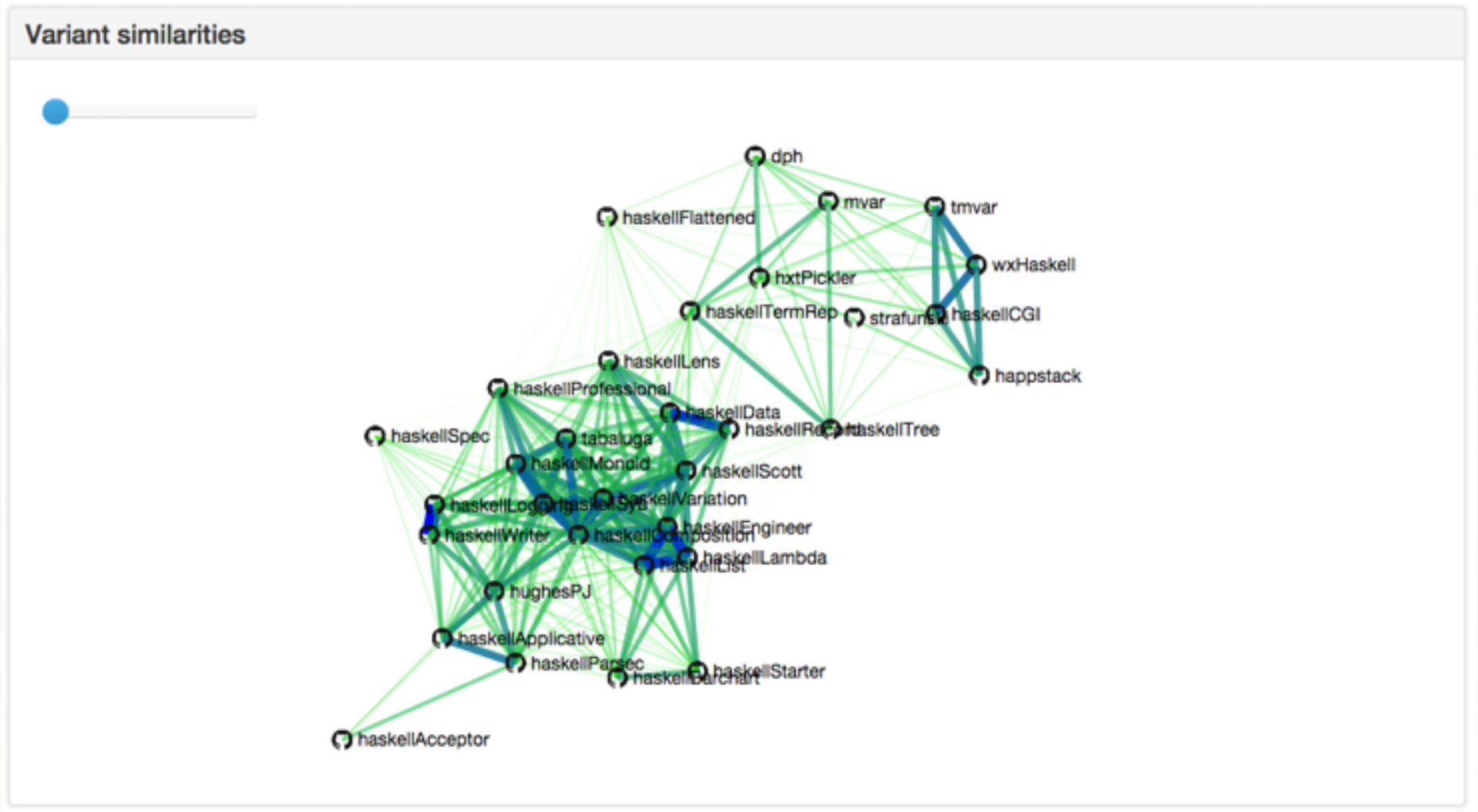


Commit-centric annotator

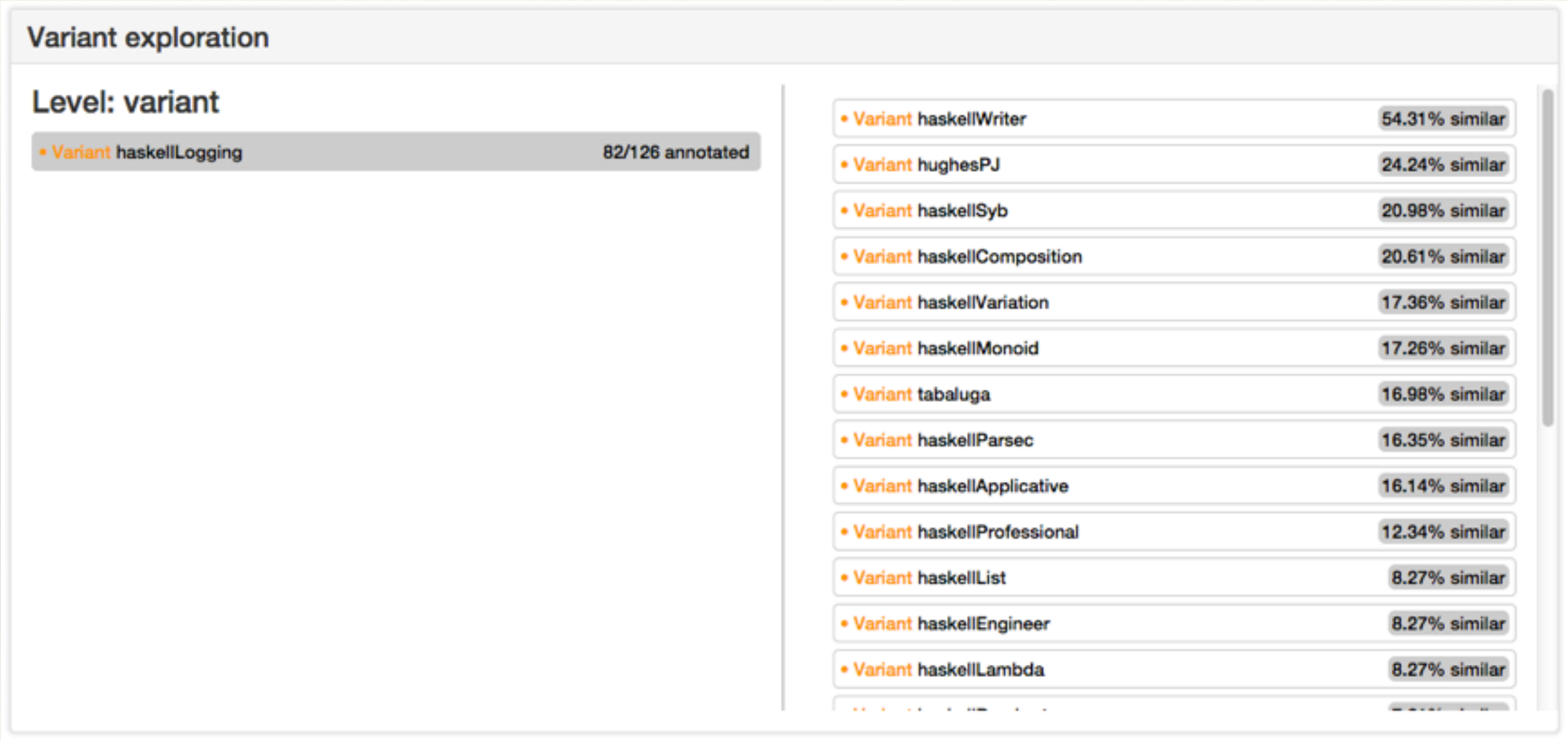
Variant-centric annotator

- Tailored for the use case where variants should be excepted at HEAD

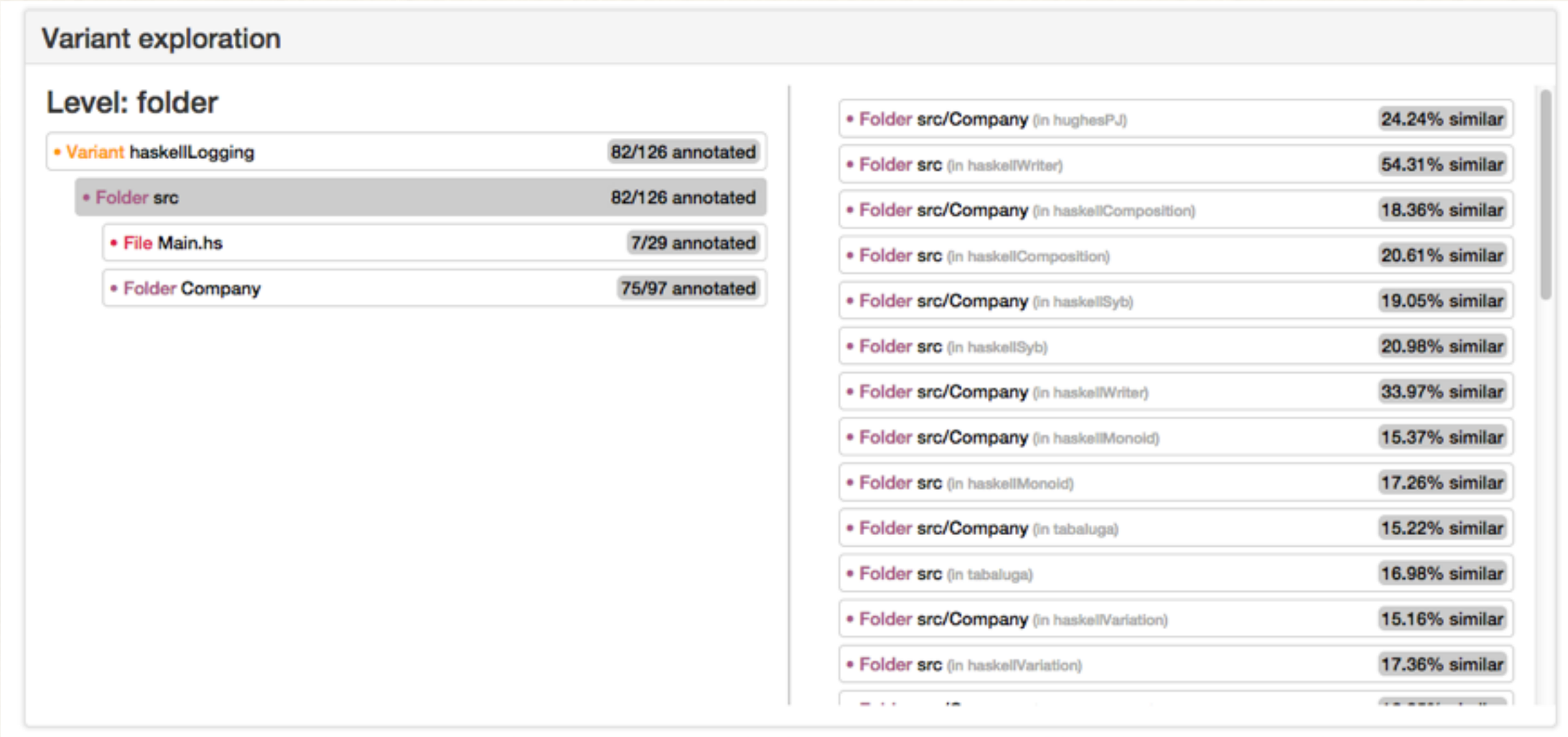
Variant-centric annotator



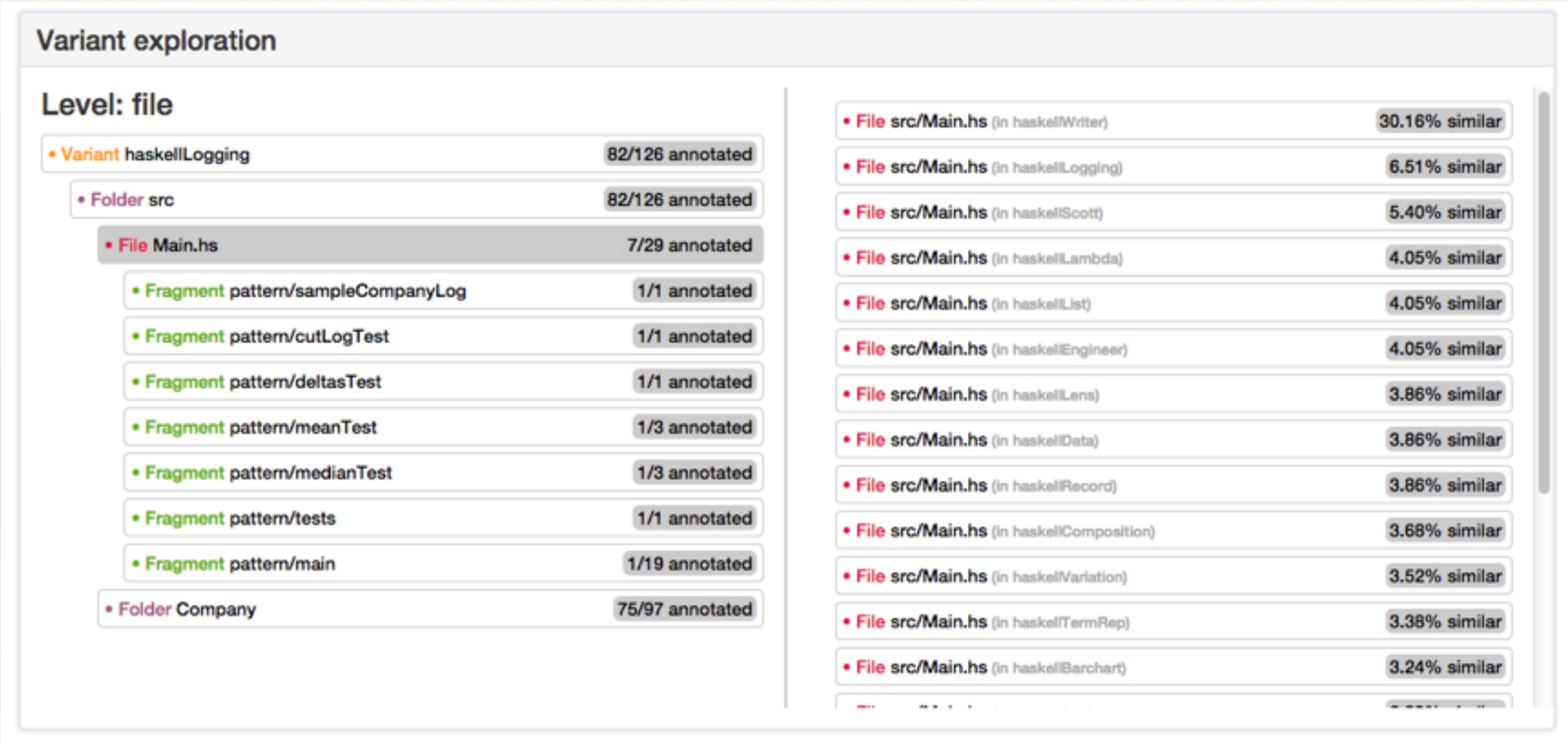
Variant-centric annotator



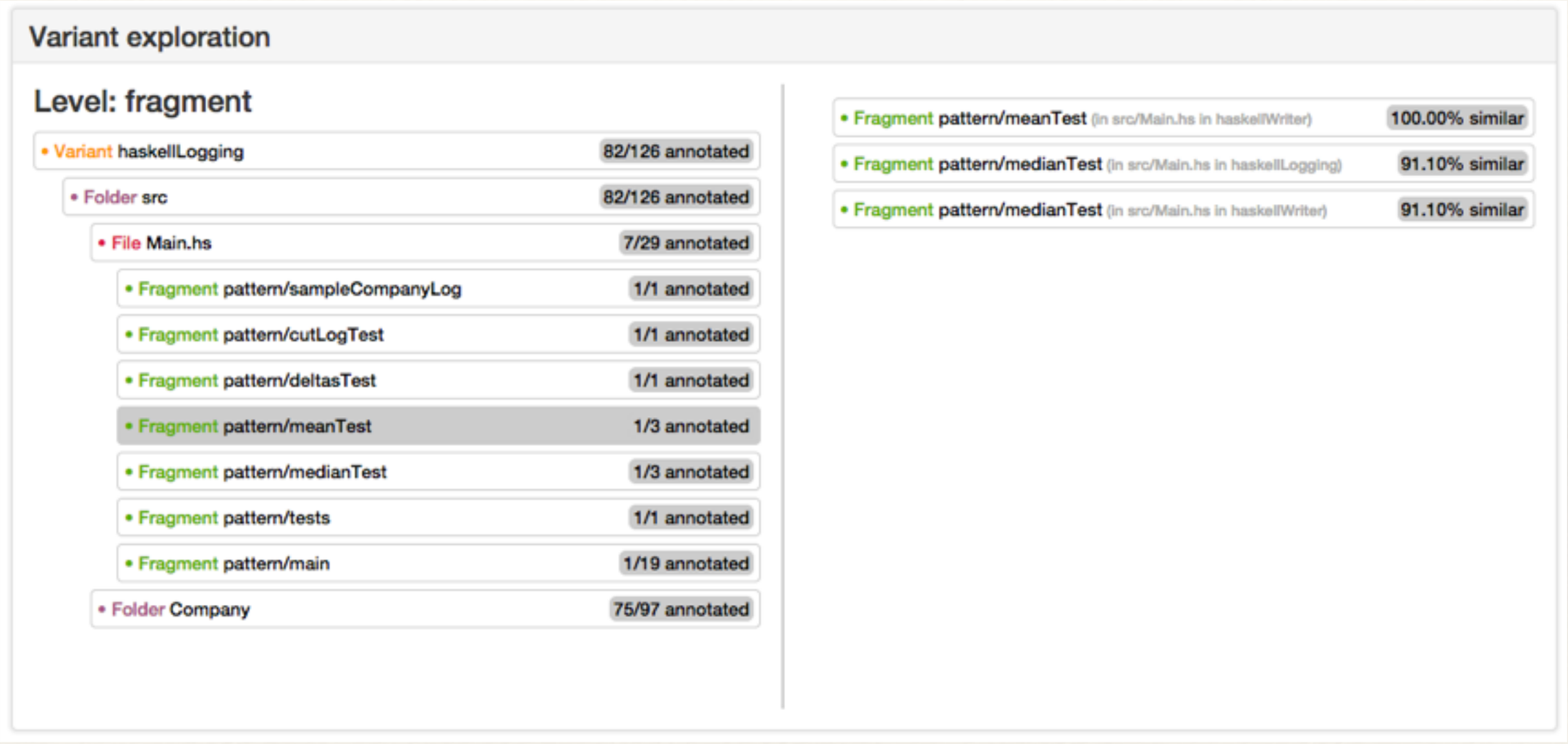
Variant-centric annotator



Variant-centric annotator



Variant-centric annotator



Variant-centric annotator

Annotations

Similar fragments

Similarity: 0.90 ~> 0.91

Annotate

pattern/medianTest in
Variant: nonmonadic
Path: src/Main.hs

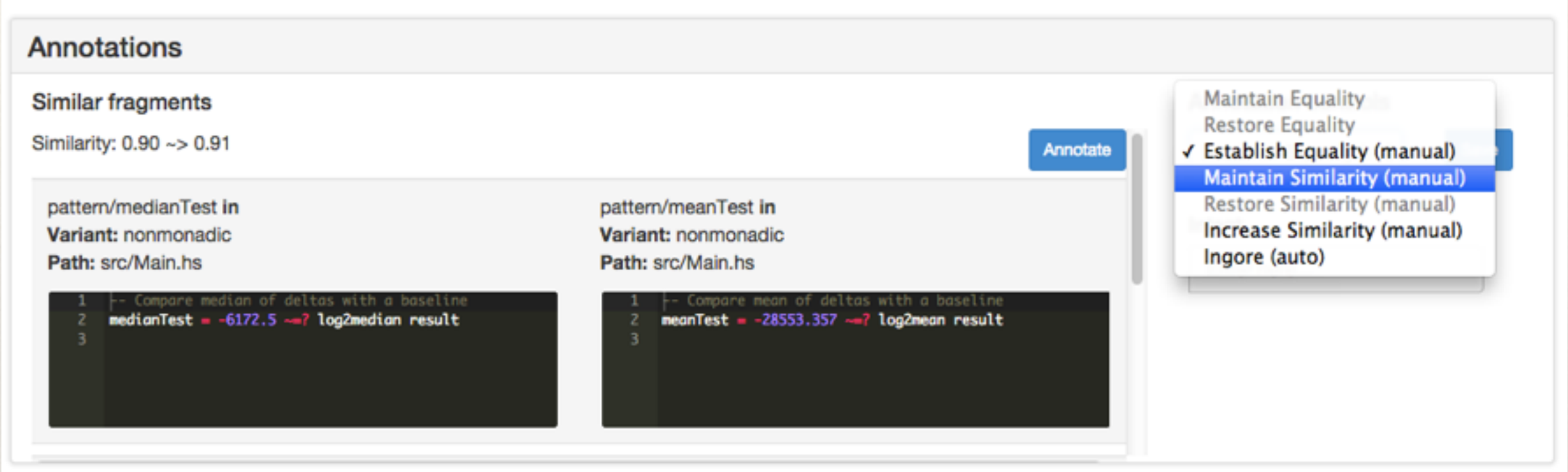
```
1 |- Compare median of deltas with a baseline
2 medianTest = -6172.5 ~=? log2median result
3
```

pattern/meanTest in
Variant: nonmonadic
Path: src/Main.hs

```
1 |- Compare mean of deltas with a baseline
2 meanTest = -28553.357 ~=? log2mean result
3
```

Annotation controls

Variant-centric annotator



Annotation Inference

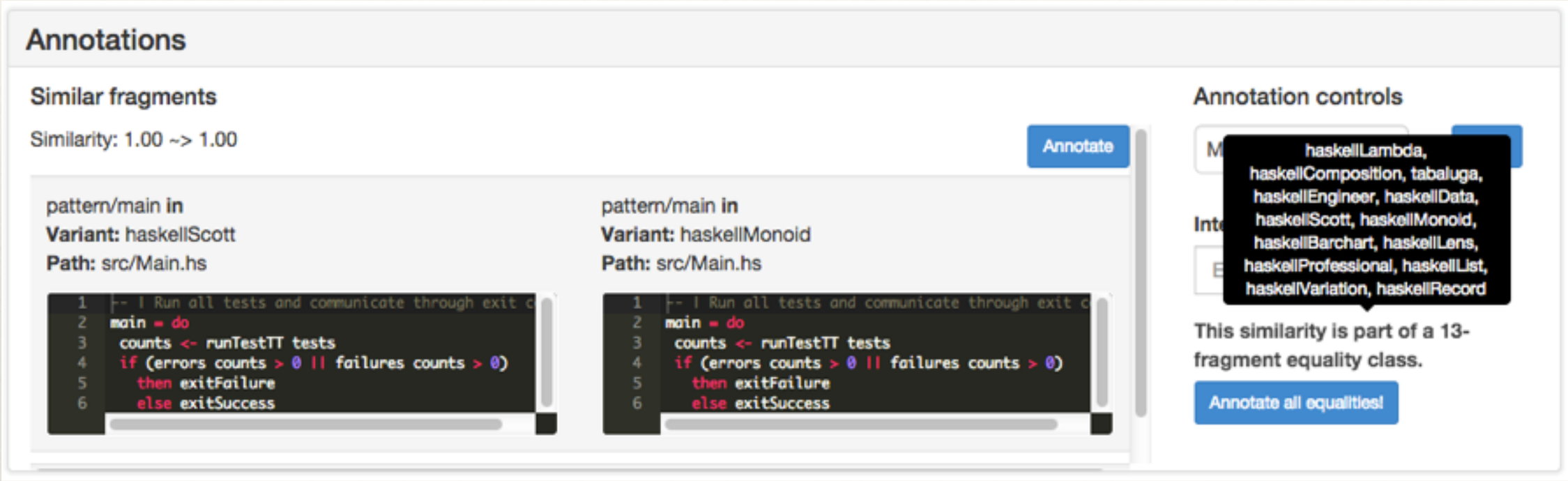
Similarity Graph

- We can think of the extracted similarities as a graph
  - Fragments are nodes
  - Similarities are weighted edges connecting fragments
- We can use some properties of the graph to infer annotations
  - Mass-annotate equality classes
  - Infer annotations via rules

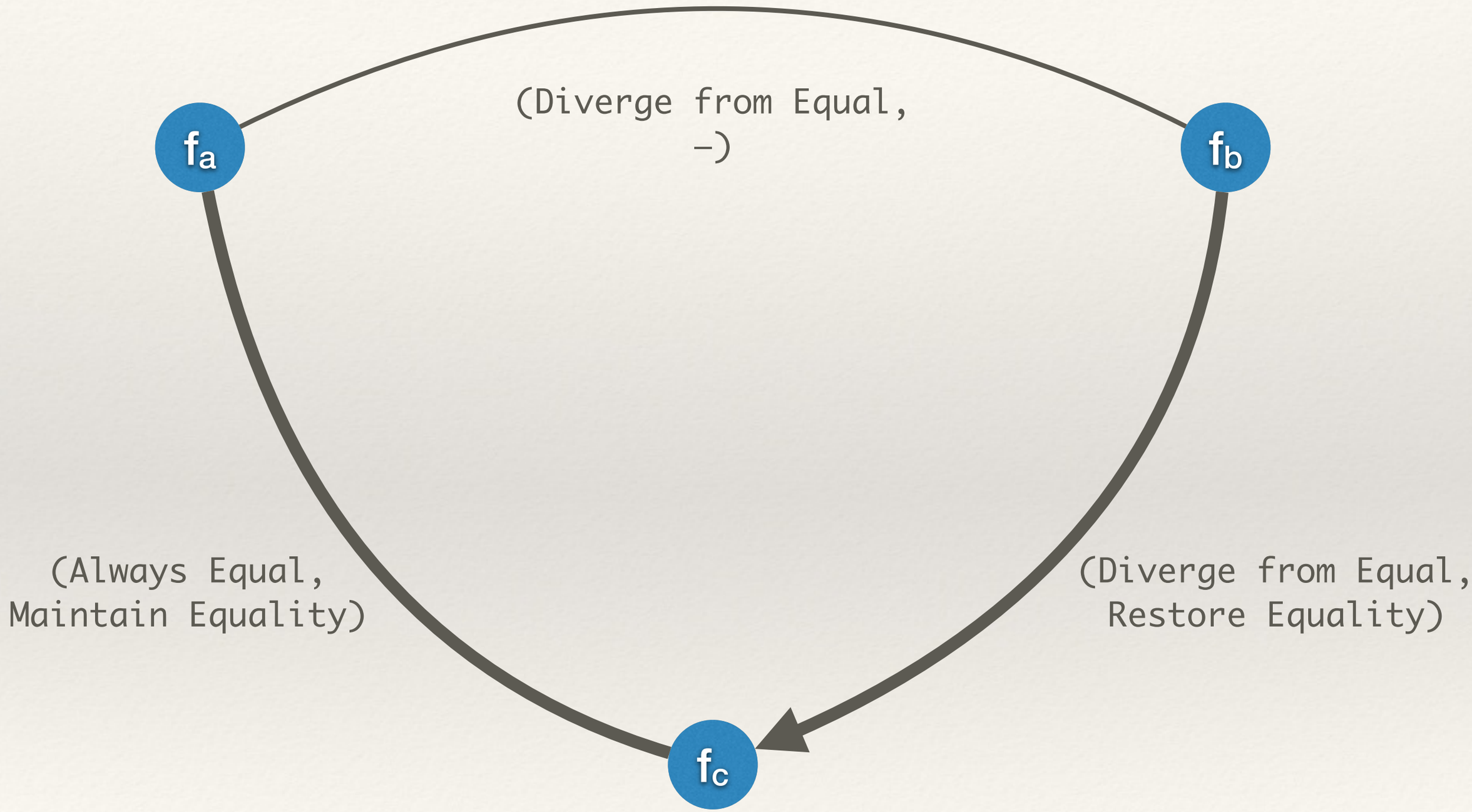
Equality classes

- An equality class is a set of pairwise equal fragments ("cloning class")
- In the similarity graph an equality class:
  - Complete subgraph with all edges weighted 1.0
- For n fragments there are $\Sigma_{i=0}^{n-1} i$ equalities in the class
- Let the user annotate all equalities with one action!

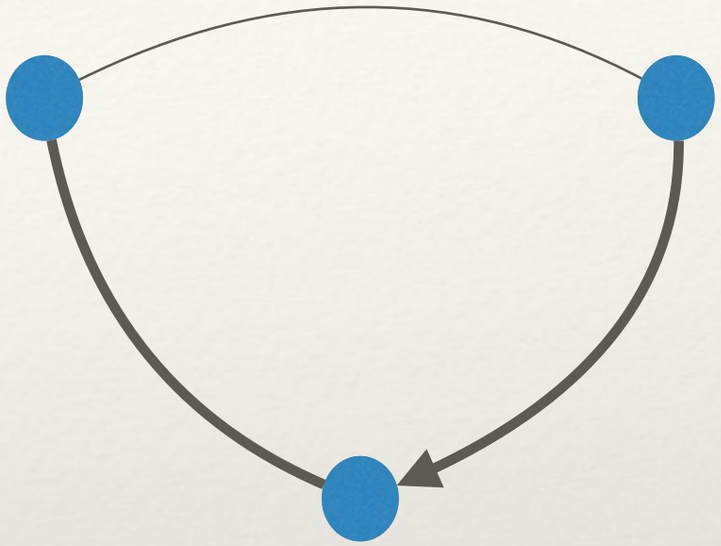
Equality classes



Annotation inference



Annotation inference

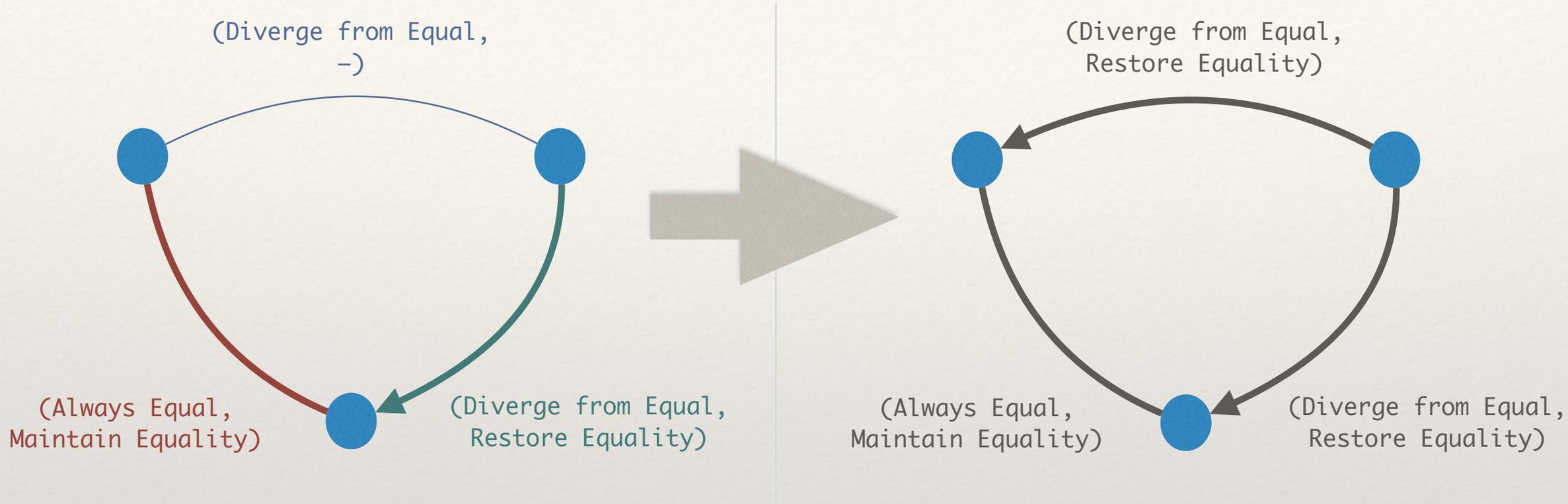


- We can infer the third annotation
- We can generally write down inference rules
- E.g.:
  - `A({AlwaysEqual,ConvergeToEqual}, MaintainEquality, None),`

    `A({DivergeFromEqual}, RestoreEquality, Left)`

    → `A({DivergeFromEqual}, RestoreEquality, Right)`

Rule application



- A({AlwaysEqual,ConvergeToEqual}, MaintainEquality, None),

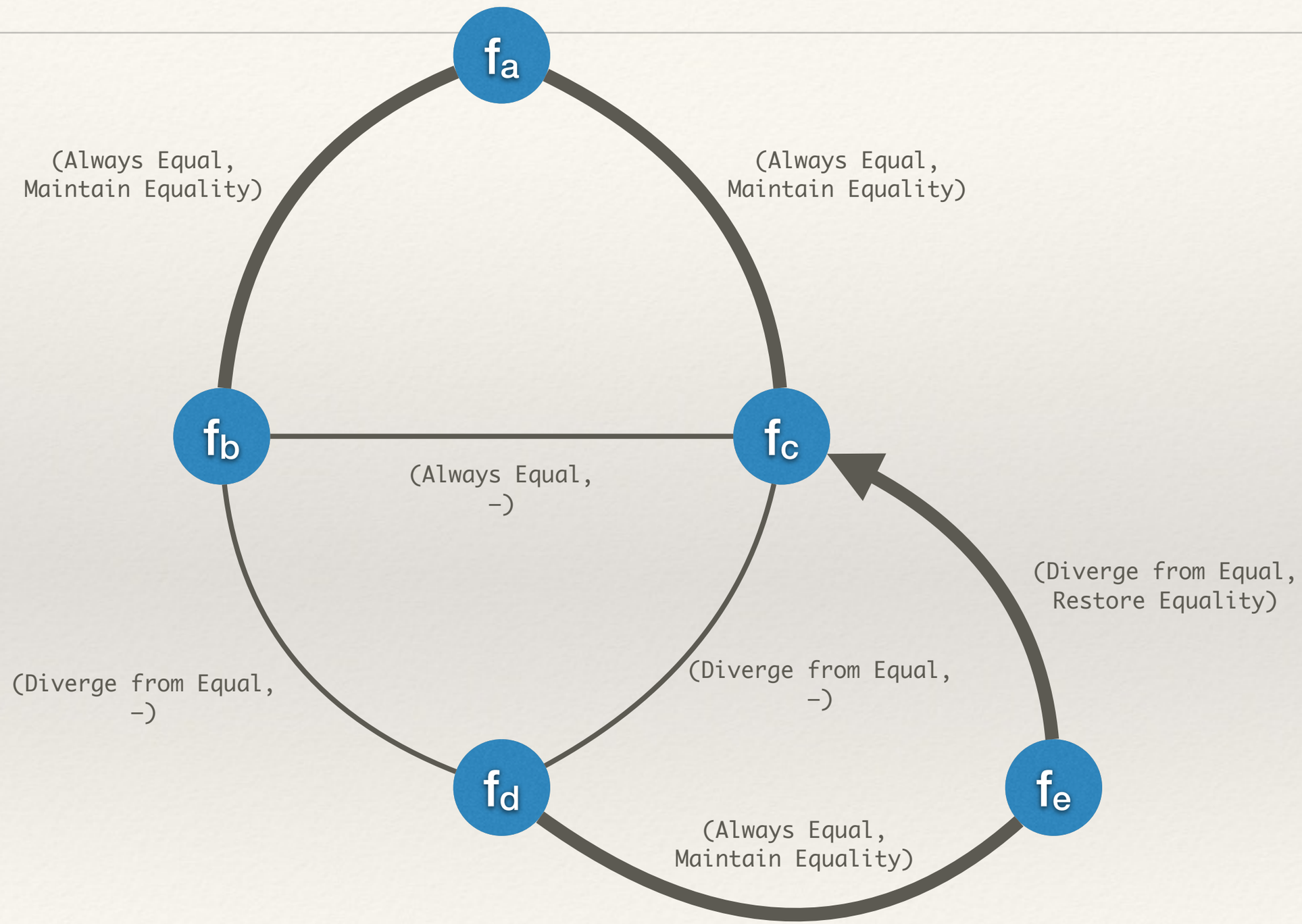
  A({DivergeFromEqual}, RestoreEquality, Left)

  → A({DivergeFromEqual}, RestoreEquality, Right)

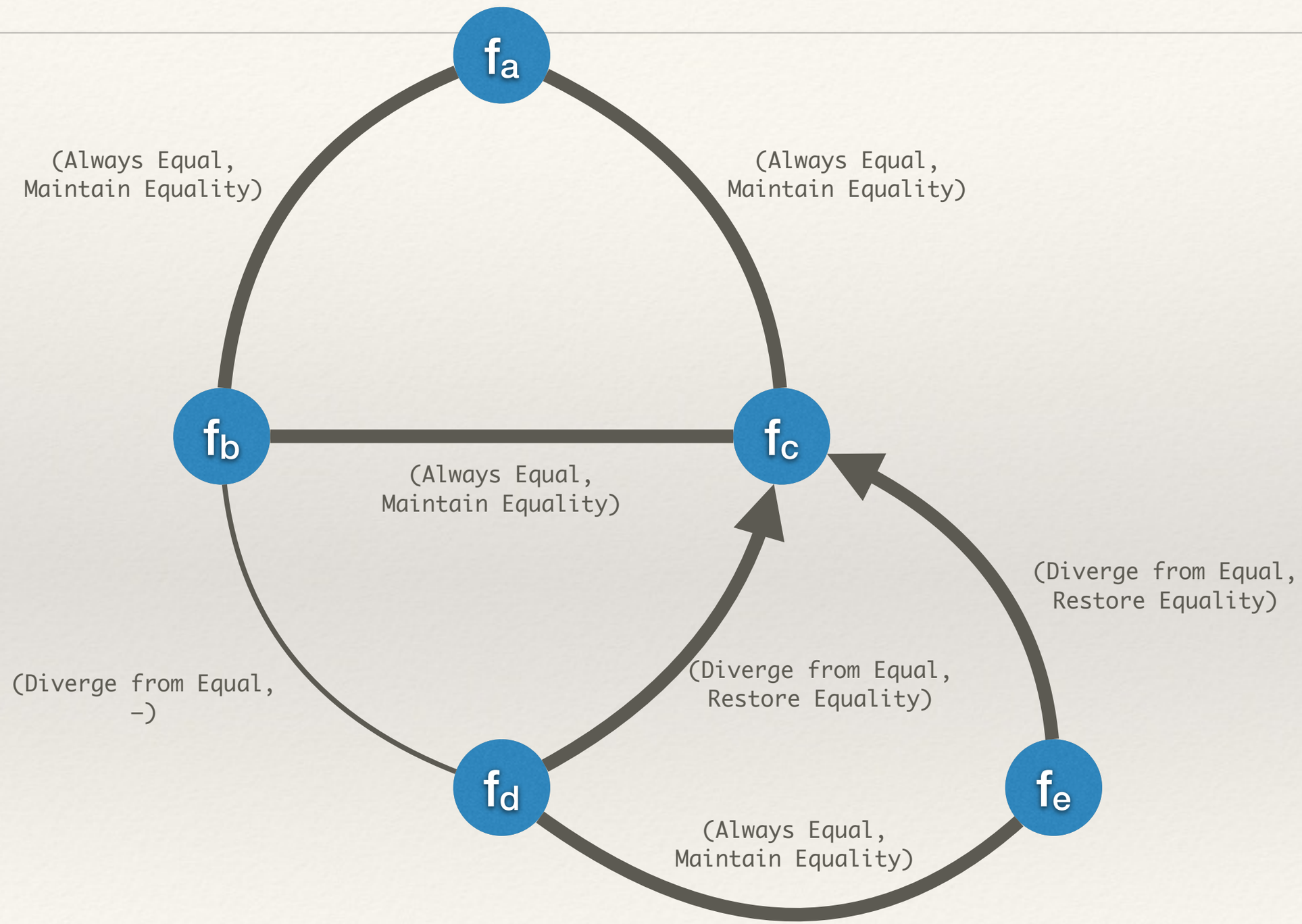
Inference rules

- A({AlwaysEqual,ConvergeToEqual}, MaintainEquality, None),
  A({AlwaysEqual,ConvergeToEqual}, MaintainEquality, None)
  → A({AlwaysEqual,ConvergeToEqual}, MaintainEquality, None)
- A({AlwaysEqual,ConvergeToEqual}, MaintainEquality, None),
  A({DivergeFromEqual}, RestoreEquality, Right)
  → A({DivergeFromEqual}, RestoreEquality, Left)

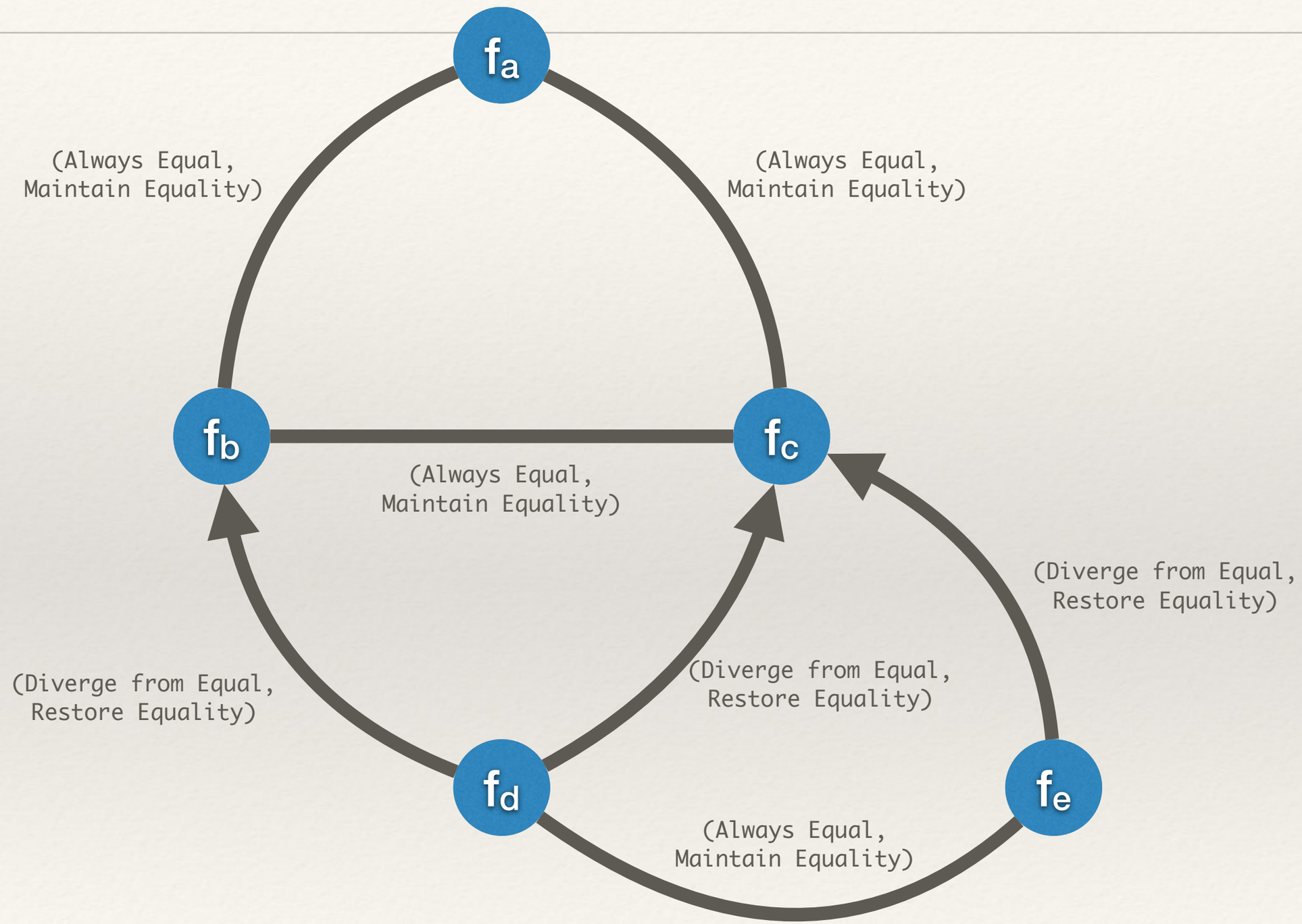
Cascading rule application



Cascading rule application



Cascading rule application



Automatic Change Propagation

Automatic change propagation

- For annotations of category `Maintain Equality` we can easily detect an invariant violation and possibly turn the annotation into `Restore Equality`
- For all annotations for category `Restore Equality` we can perform the expressed maintenance task automatically via change propagation

Automatic change propagation

- For each **similarity** annotated with `Restore Equality`
  - If the annotation has a **direction**, overwrite the **other fragment**
  - Else compute a **three-way-merge** of both fragments
  - If a **conflict** occurs, fail
  - Otherwise, if only **one fragment** differs from the merge content, overwrite other with the merge content
  - Otherwise ask the **user** whether **both fragments** should be changed

Automatic change propagation

- Changes may propagate over many similarity edges
- Repeat process for all **adjacent similarity edges** until no change was made in an iteration

Updating annotations

- Automatic execution can fail for two reasons
  - Three-way-merge has conflicts
  - User has declined changing both fragments
- Depending on successful or failed executing we need to update the annotation

Automatic change propagation

- <u>If execution failed</u>
  - Set manual flag
- <u>If execution succeeded</u>
  - Update to category `Maintain Equality`
  - Remove direction

Recommending and Updating Annotations

Updating & recommending annotations

- Extract possibly new similarities from new commits

Updating & recommending annotations

- **Recommend** invariants for unannotated similarities
  - Maintain Equality, if fragments are currently equal
  - Maintain Similarity otherwise

Updating & recommending annotations

- **Turn** invariants into tasks
  - Maintain Equality > Restore Equality, if inequality at HEAD
  - Maintain Similarity > Restore Similarity, if similarity value below value when invariant was created

Updating & recommending annotations

- **Turn** tasks into invariants
  - Restore Equality > Maintain Equality
  - Establish Equality > Maintain Equality
  - Increase Similarity > Maintain Equality,
    if equality at HEAD
  - Restore Similarity > Maintain Similarity,
    if diff ratio at least as high as when invariant was created

Updating & recommending annotations

- **Recommend** invariants for similarities annotated with tasks
  - Increase Similarity > Maintain Similarity
  - Restore Equality > Maintain Similarity
    if similarity value is above the one when task was added
  - Remove Equality > Maintain Similarity,
    if inequality at HEAD

Manual Maintenance

User tasks

- A user might have to act on an annotation if
  - Automatic change propagation failed or
  - The annotation expresses as user task by default
- Ann collects these user tasks in a TODO list

TODO list

Todo List

Name	Auto?	Via rule?	Via eq-class?	Intent	Fragment 1	Fragment 2	Similarity
Increase Similarity	Manual	No	No		data/Employee **in** **Variant:** wxHaskell **Path:** src/Company/Company.hs	data/Employee **in** **Variant** happstack **Path:** Company.hs	1 → 0.98
Restore Equality	Failed!	No	No		data/Department **in** **Variant:** happstack **Path:** Company.hs	data/Department **in** **Variant** tmvar **Path:** Company.hs	1 → 0.99
Establish Equality	Manual	No	No		pattern/company **in** **Variant:** happstack **Path:** SampleCompany.hs	pattern/company **in** **Variant** tmvar **Path:** SampleCompany.hs	0.95 → 0.95

TODO list

- For each annotation the developer has several options
  - Edit fragments to complete manual task
  - Accept/reject a recommendation
  - Remove the annotation
  - Edit the annotation
  - Add annotation to newly extracted similarity

Git Integration

Git integration

- In general many version control systems can be supported
- We have realized the integration with Git
  - Use Git API for metadata extraction
  - Extend set of Git command with new commands

New Git commands

- `git ann init`
  - **Functionality**:
    - Registers the repo with our system
    - Triggers initial metadata extraction
  - **Integration points**:
    - Any point were repo is in a clean state

New Git commands

- `git ann update`
  - **Functionality**:
    - Updates metadata for new commits
    - Updates annotations based on user actions
  - **Integration points**:
    - After new commits are pulled or created locally

New Git commands

- `git ann propagate`
  - **Functionality**:
    - Calls `git ann update` if needed
    - Trigger automatic change propagation
  - **Integration points**:
    - Before changes are pushed

Case Study: *101haskell*

101companies & 101haskell

- *101companies* documents software languages, technologies and concepts
- *101companies* is a software chrestomathy
  - A collection of small software systems useful for teaching software engineering
- 101companies is a set of variants all implementing a common feature model
  - Variance by
    - (a) implementing different features
    - (b) common features differently to demonstrate languages, techs, concepts

101companies & 101haskell

- *101haskell* is a subproject of *101companies*
- *101haskell* focuses on demonstrating
  - concepts in Haskell
  - technologies for Haskell
- 36 variants ("contributions")
  - *haskellStarter*: Contribution with small language footprint
  - *haskellComposition*: Use of recursive data types
  - *haskellVariation*: Use of multiple constructors per type
  - *wxHaskell*: GUI programming in Haskell with wxHaskell

Research question

How to measure the <u>similarity improvements</u> achieved by applying our approach and tools?

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

1. Automatically restore equalities:
   - Annotate all **equalities**
   - Annotate all **divergences from equalities**
   - Execute **automatic change propagation**

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

1. Automatically restore equalities, results:

   - **Fragments**

		Δ0,1
total	925	+/-0
unique	617	-15
variants sharing (median)	1	+/-0
variants sharing (average)	1.51	+0.05

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

1. Automatically restore equalities, results:
   - **Similarities**

		Δ0,1
median	1.0	+1.163%
average	0.96495	+2.452%

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

1. Automatically restore equalities, results:

   - **Variants, uniqueness (ratio of non-shared fragments)**

		Δ0,1
median	49.00%	-3.47%
average	51.39%	-1.95%

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

  2. Manually establish equalities:

     - Annotate all **remaining similarities**
     - Manually establish equalities by **editing fragments**

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

2. Manually establish equalities, results:

  - **Fragments**

		Δ1,2
total	925	+/-0
unique	595	-22
variants sharing (median)	1	+/-0
variants sharing (average)	1.55	+0.04

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

  2. Manually establish equalities, results:

    - **Similarities**

		Δ1,2
median	1.0	+/-0.0%
average	0.96795	+0.310%

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

2. Manually establish equalities, results:

  - **Variants, uniqueness**

		Δ1,2
median	47.34	-1.66
average	50.00	-1.39

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

  3. Increase similarities:

     - **Increase** similarities by editing

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

  3. Manually establish equalities, results:

     - **Fragments**

no change

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

  3. Manually establish equalities, results:

     - **Similarities**

		Δ2,3
median	1.0	+/-0.0%
average	0.96802	+0.007%

Step-wise case study

- Four steps and tactics to improve quality of *101haskell*:

  3. Manually establish equalities, results:

     - **Variants, uniqueness**

no change

Results

- **Similarity (median/average)**
  - Median: 98.85% to 100.00%
  - Average: 94.18% to 96.80%

Results

- **Sharing**
  - #unique fragments
    - 632 to 595 (-5.85%)
  - #variants a fragment is shared in (median/average)
    - 1/1.46 to 1/1.55

Results

- **Non-trivial equality classes**
  - #: 95 to 85
  - max: 13 to 20

Results

- **Variant uniqueness**
  - Median: 52.47% to 47.34% (-5.13%)
  - Average: 53.34% to 50.00% (-3.34%)

Results

- **Similarity evolutions**
  - Initially:
    - Always Equal (38.1%)
    - Always Similar (35.0%)
    - Converge to Equal (8.6%)
    - Diverge from Equal (18.2%)
  - After restoring/establishing eqs.:
    - Converge to Equal (32.5%)

Results

- **Annotations**
  - 324 `Restore Equality` / unintentional divergences identified
  - 0 `Remove Equality`
  - 130 `Establish Equality`
  - 421 `Ignore Similarity` with 0.8 similarity threshold
    - Most interesting similarities detected

Conclusion

Summary

- We have proposed an approach to manage similarities and cloning across variants
- We have realized a system to
  - analyze,
  - annotate, and
  - maintain similarities.
- We have presented a case study on *101haskell* where the approach helped to improve the projects quality

Future work

- Support for more operators
  - `clone variant`, `clone feature`
  - May require additional metadata
  - May create annotations automatically

Future work

- Guided annotation process
  - Guide user which similarity to annotate next
  - Maximize number of inferred annotations
  - Minimize number of manual annotations

Thanks! Questions?