ParallelGumtree

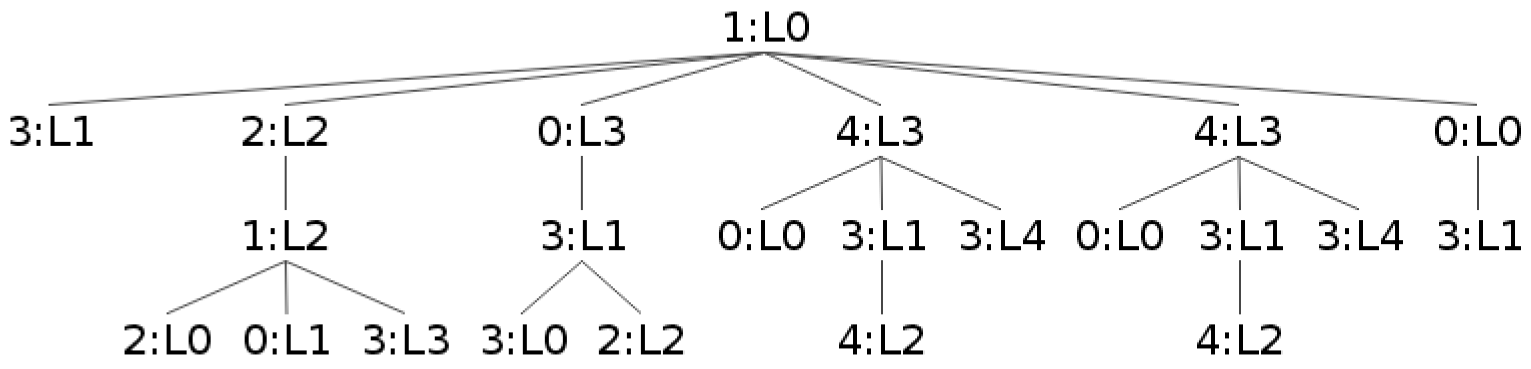
DPHPC project of

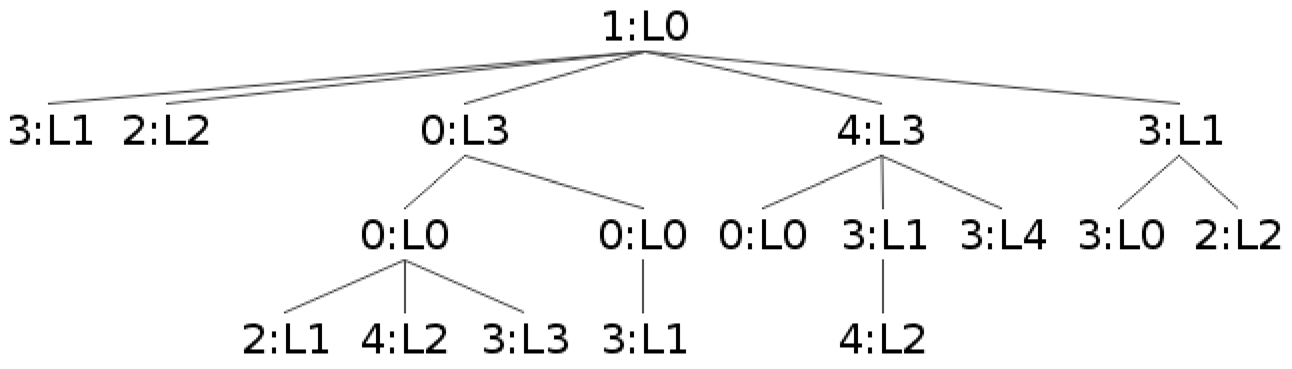
Amirreza Bahreini
Christopher Signer
Balz Guenat

Problem: Matching Trees

- Given two similar rooted ordered trees, find a matching of nodes.
- Find edit operations transforming one tree into the other.
- Applications:
  - Different versions of source code ASTs
  - Compare XML documents

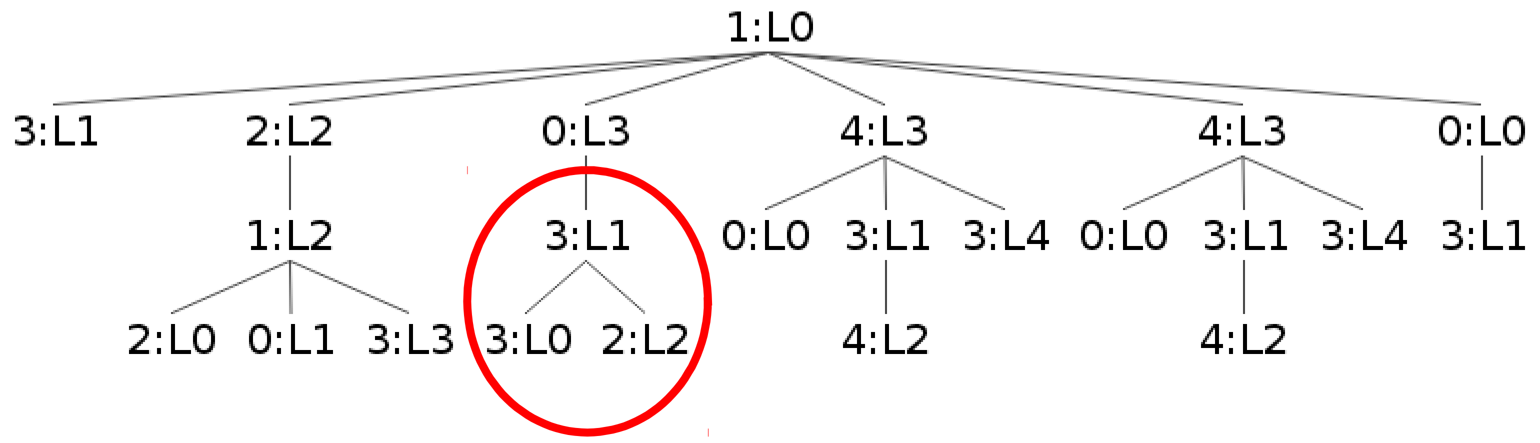
Trees

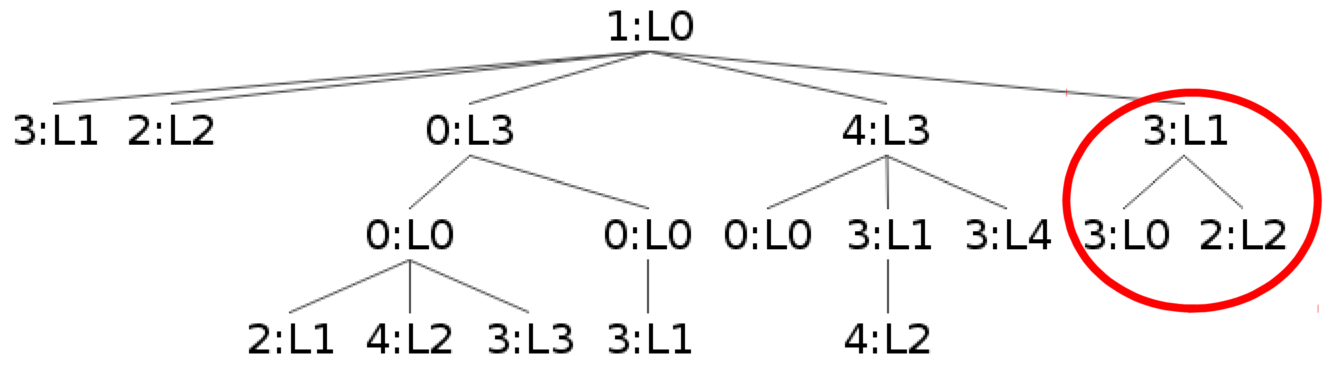

1:L0
3:L1
2:L2
0:L3
4:L3
4:L3
0:L0
1:L2
3:L1
0:L0
3:L1
3:L4
0:L0
3:L1
3:L4
3:L1
2:L0
0:L1
3:L3
3:L0
2:L2
4:L2
4:L2


1:L0
3:L1
2:L2
0:L3
4:L3
3:L1
0:L0
0:L0
0:L0
3:L1
3:L4
3:L0
2:L2
2:L1
4:L2
3:L3
3:L1
4:L2

Trees – Top down

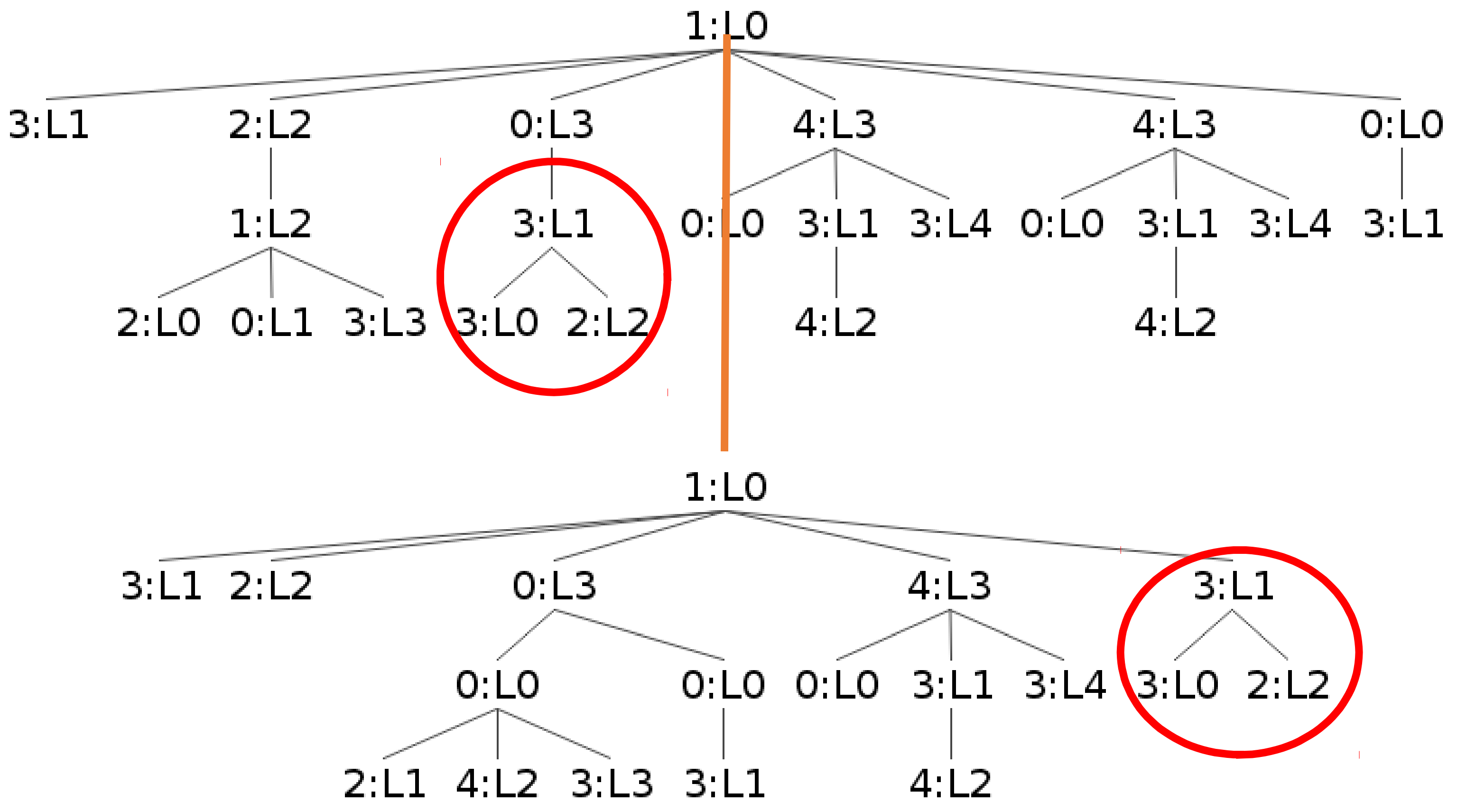
- Match identical subtrees





Trees – Bottom up

- Match by similarity



Approach

- Port from single thread Java to C++
- Tree parsing, writing
- Generation of pairs of similar trees
- Add parallelizing (OpenMP)
- Write test tool
- Optimizing and evaluating

Challenges

- Start off with a large amount of code
- Hard to get realistic tree generation
  - Should be similar to source code edits
  - Hard to validate
- gperftools caused freezes :(

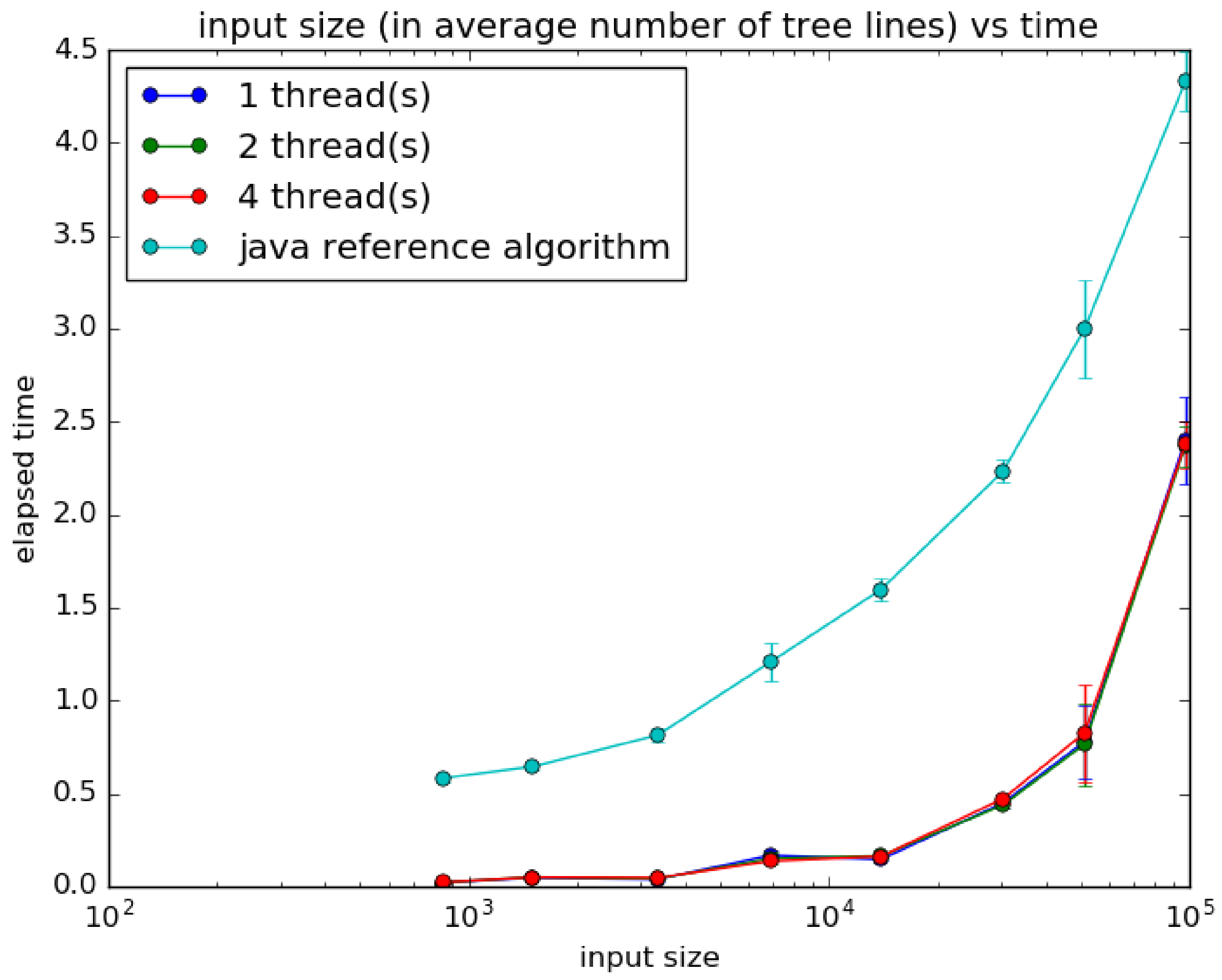
Results

Good news:

- Much faster than Java
- Lower memory footprint

Bad news:

- Basically no speedup



Compiled with GCC 5.2.1, Ubuntu 15.10, Intel Core i7-3520M

The problem:

- Size of parallelizable parts depends on problem instance
- Found to be generally small
- Inherent in algorithm

```
#pragma omp parallel for reduction(pair_max:maxPair)
for (unsigned i = 0; i < candidatesSize; ++i) {
     auto c = candidates[i];
     double sim = jaccardSimilarity(t, c);
     if (sim > maxPair.first && sim >= SIM_THRESHOLD) {
          maxPair = make_pair(sim, c);
     }
}
```

Ideas for improvement

- Use additional cores for precomputation
  - All-in-all more work but
  - Chance that needed values are already available when needed

```
auto candidates = getDstCandidates(t);

#pragma omp parallel for reduction(pair_max:maxPair)
for (unsigned i = 0; i < candidatesSize; ++i) {
     auto c = candidates[i];
     double sim = jaccardSimilarity(t, c);
     if (sim > maxPair.first && sim >= SIM_THRESHOLD) {
          maxPair = make_pair(sim, c);
     }
}
```

Ideas for improvement

- Use additional cores for precomputation
  - All-in-all more work but
  - Chance that needed values are already available when needed

```
auto candidates = getAllUnmatchedNodes(t);

#pragma omp parallel for reduction(pair_max:maxPair)
for (unsigned i = 0; i < candidatesSize; ++i) {
     auto c = candidates[i];
     double sim = jaccardSimilarity(t, c);
     if (sim > maxPair.first && sim >= SIM_THRESHOLD) {
          maxPair = make_pair(sim, c);
     }
}
```