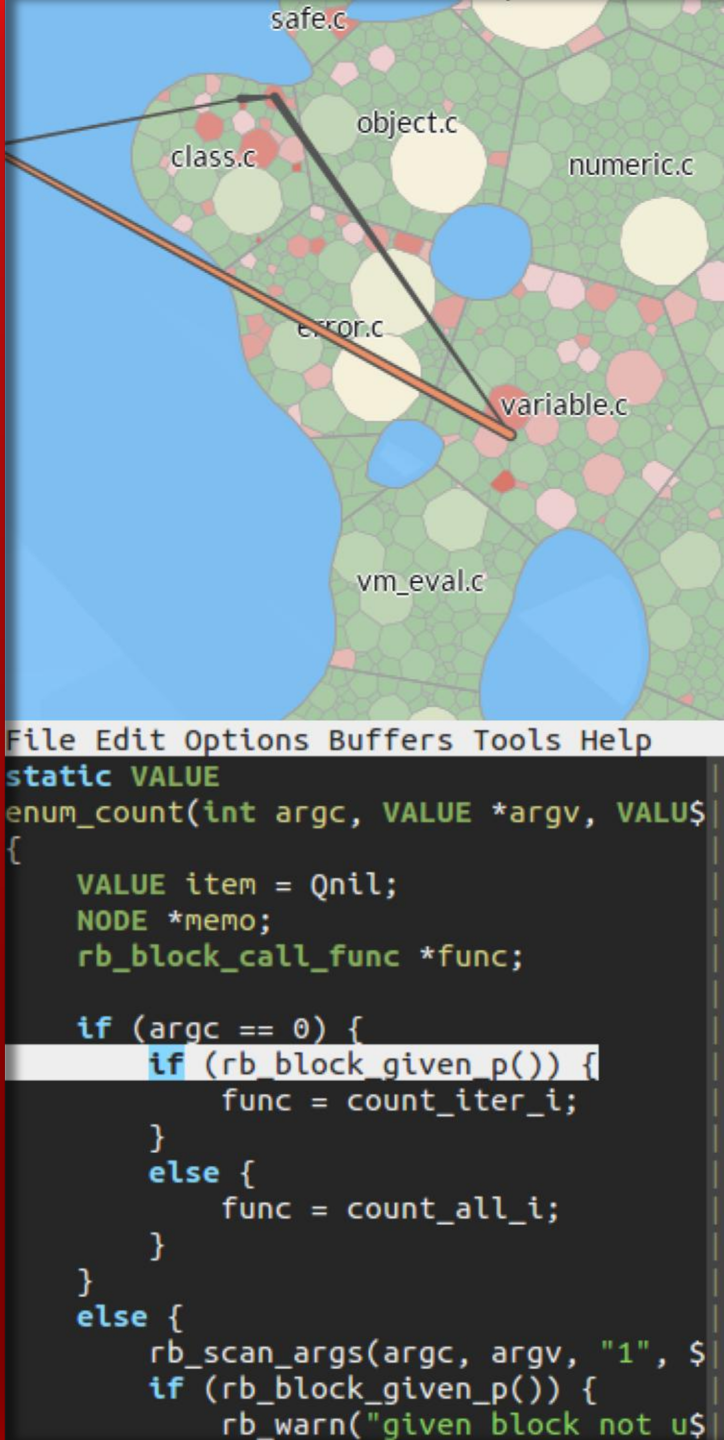
ORACLE®

ORACLE

Frappé: A Code Comprehension Tool for Large Codebases

Ben Barham and Nathan Hawes

April 2014



Disclaimer

The following is intended to provide some insight into a line of research in Oracle Labs.

It is intended for information purposes only, and may not be incorporated into any contract.

It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle.

Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.

ORACLE

Program Agenda

- Background
- The Frappé Tool
- Demonstration
- Future Work

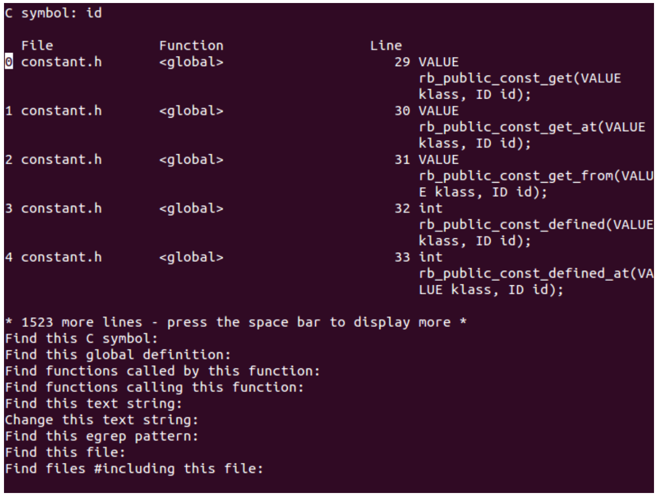
Code Comprehension

- Code comprehension is an integral part of a programmer's every day tasks
- Modern IDEs provide many features to facilitate this
  - Go to definition/implementation, find usages, class overview, etc.
- Unfortunately, using IDEs is often impractical for many large C/C++ codebases
  - Difficult to integrate with custom build systems
  - Often use custom parsers that lag behind real compilers and can be imprecise
  - Excessive startup time and memory usage

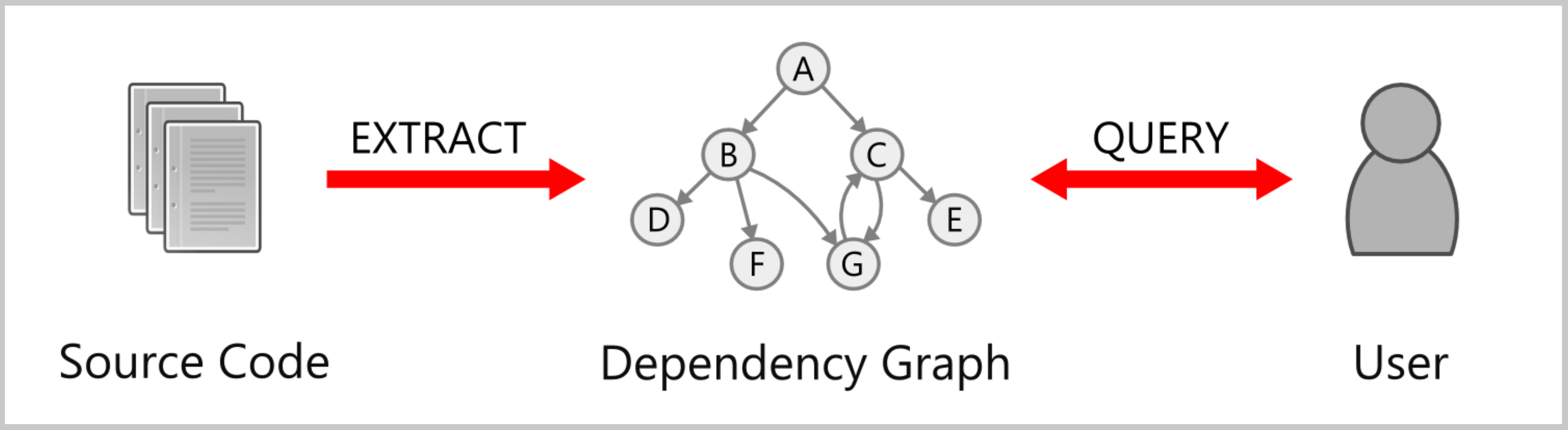
ORACLE

Current Tools

- Lightweight text editors
  - Regex based tools for lookup (Grep, CScope, etc.)
- Fast and easy to integrate but imprecise:
  - Largely unaware of symbol types, scopes, linking information, and the preprocessor
  - Deal only with direct dependencies
- Our goal: address these limitations and maintain comparable scalability and ease of integration



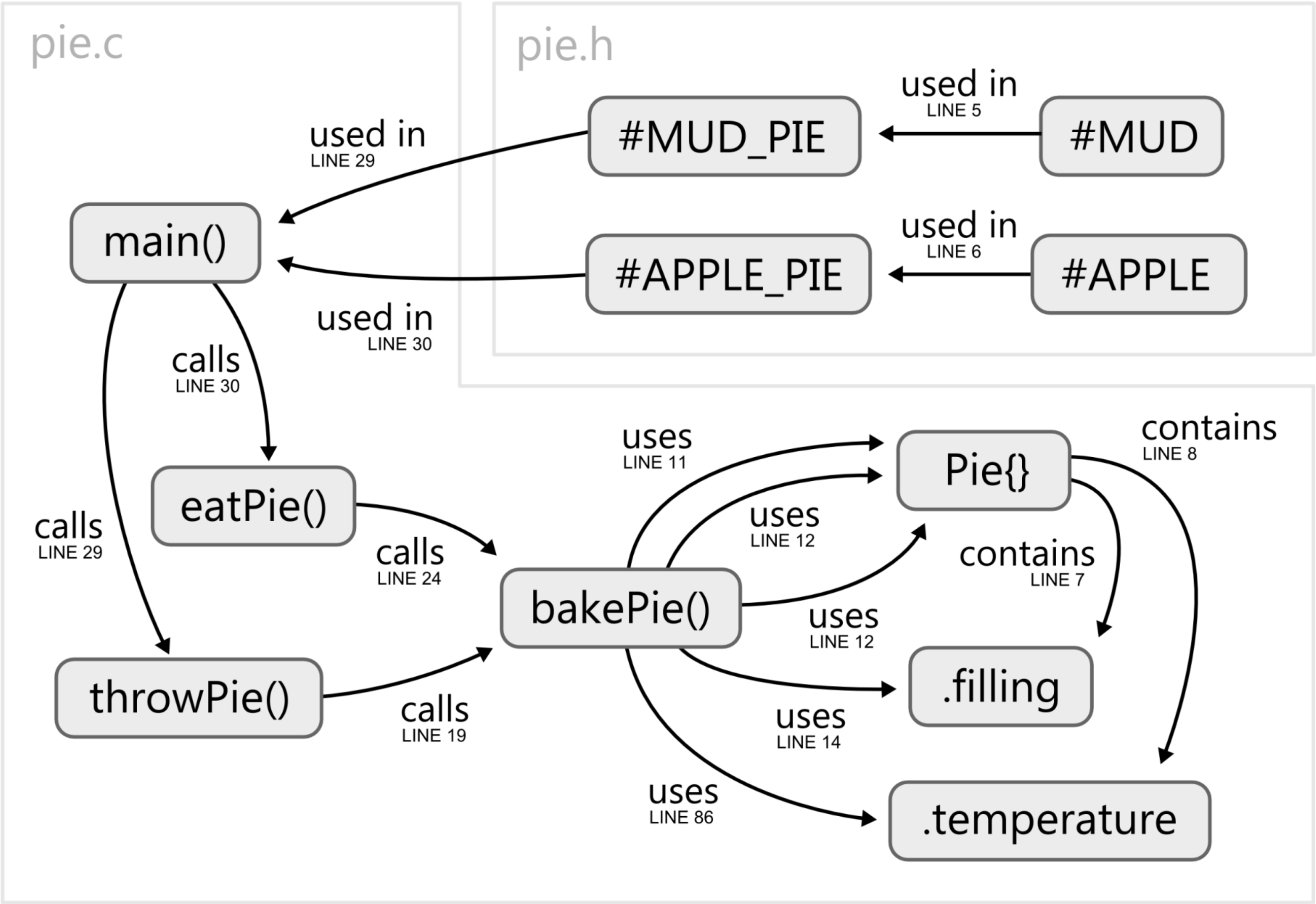
Our Approach



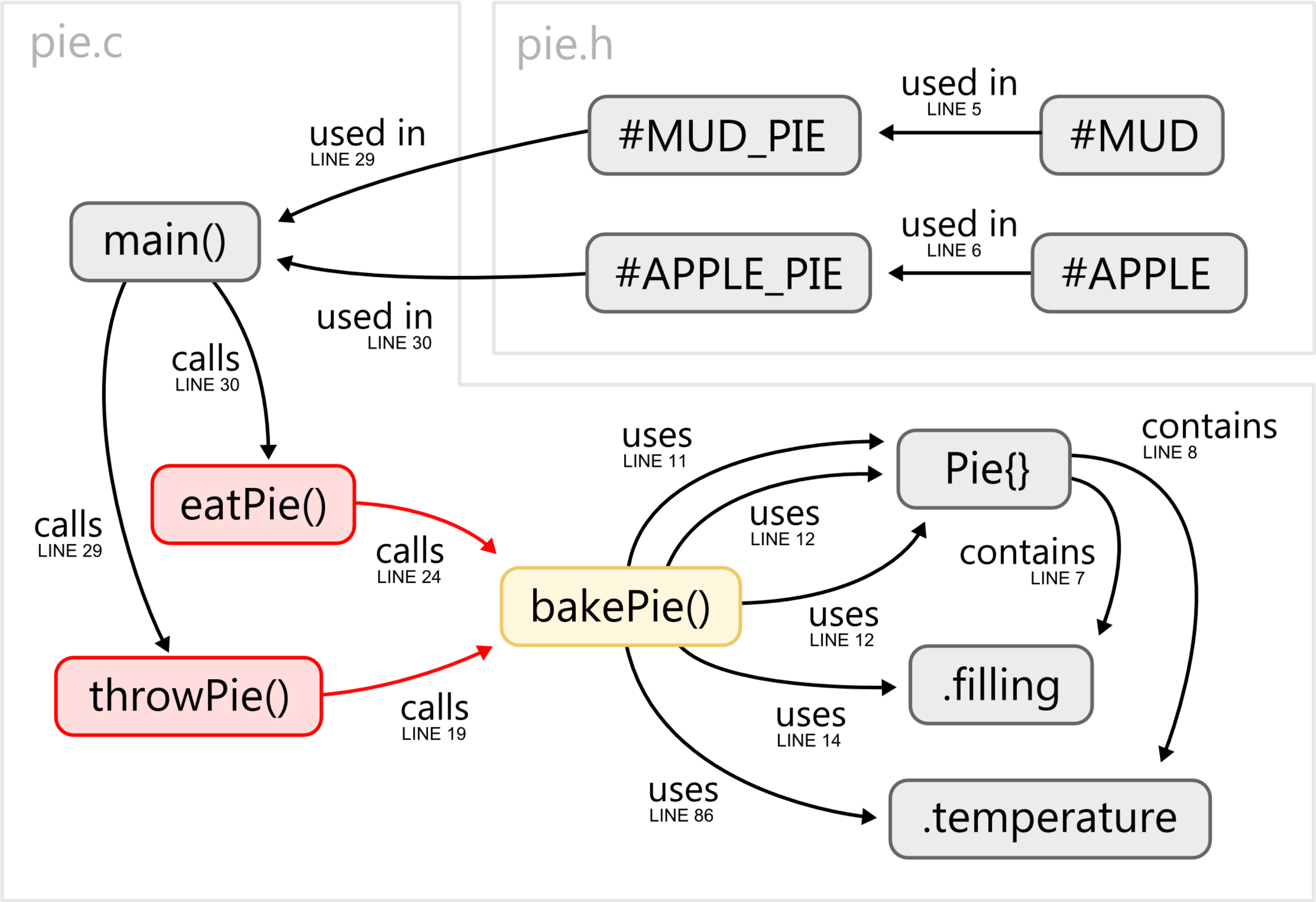
- Extract dependency graph of codebase
  - Nodes are entities within the code (functions, macros, etc.)
  - Edges are dependencies between those entities (calls, uses, etc.)

- Can express code comprehension queries as graph matches

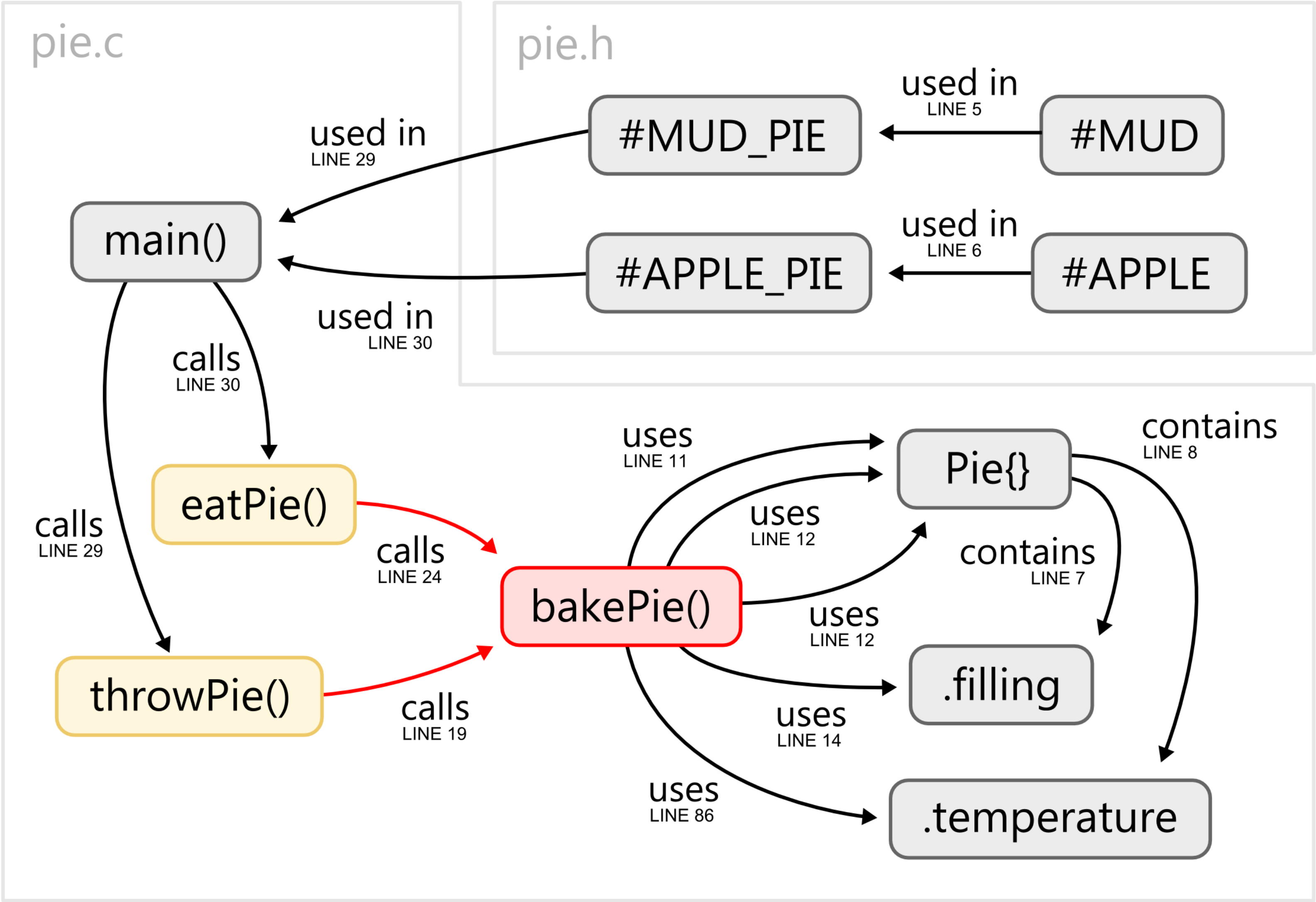
Graph Matching



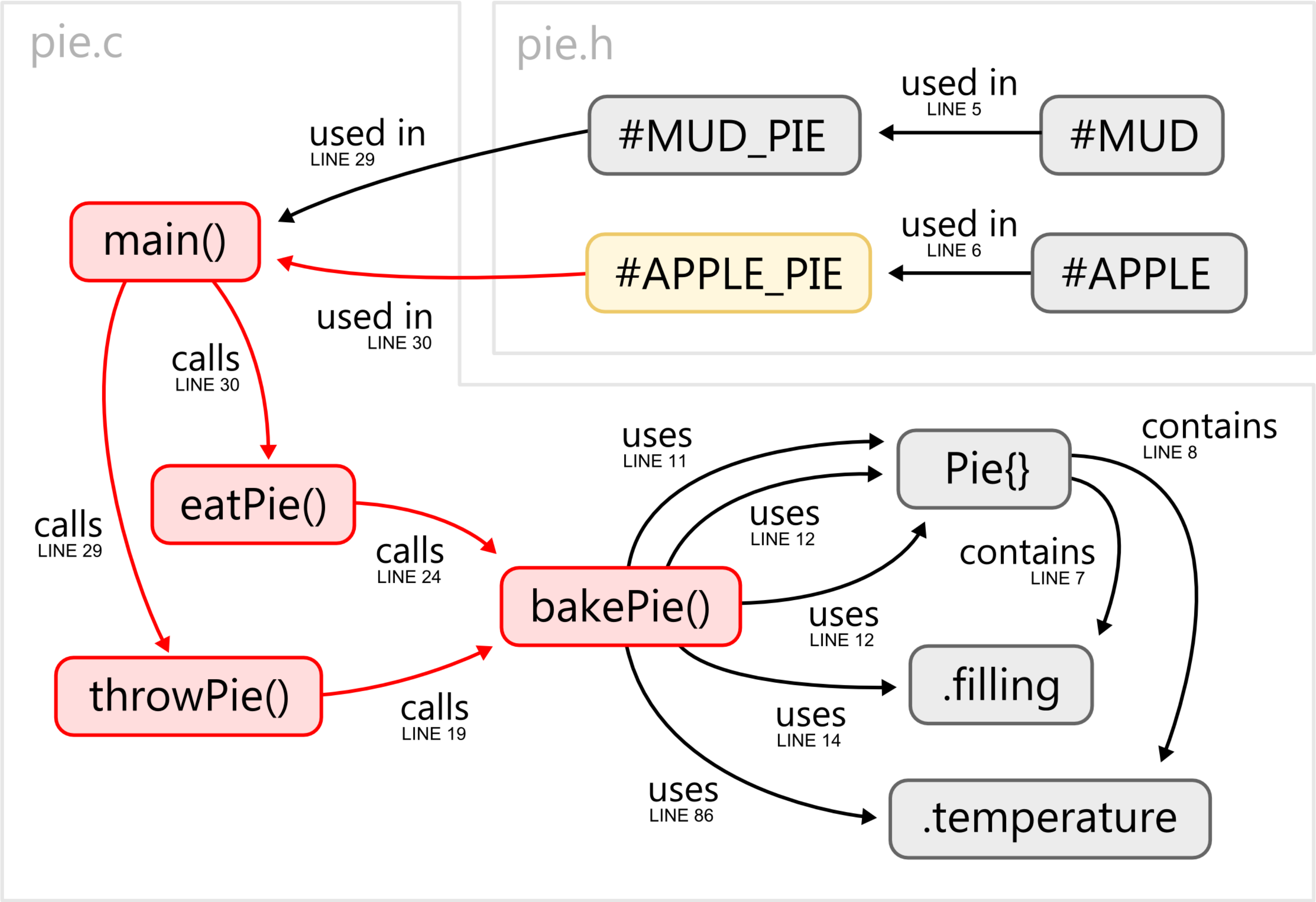
Find References



Go to Definition

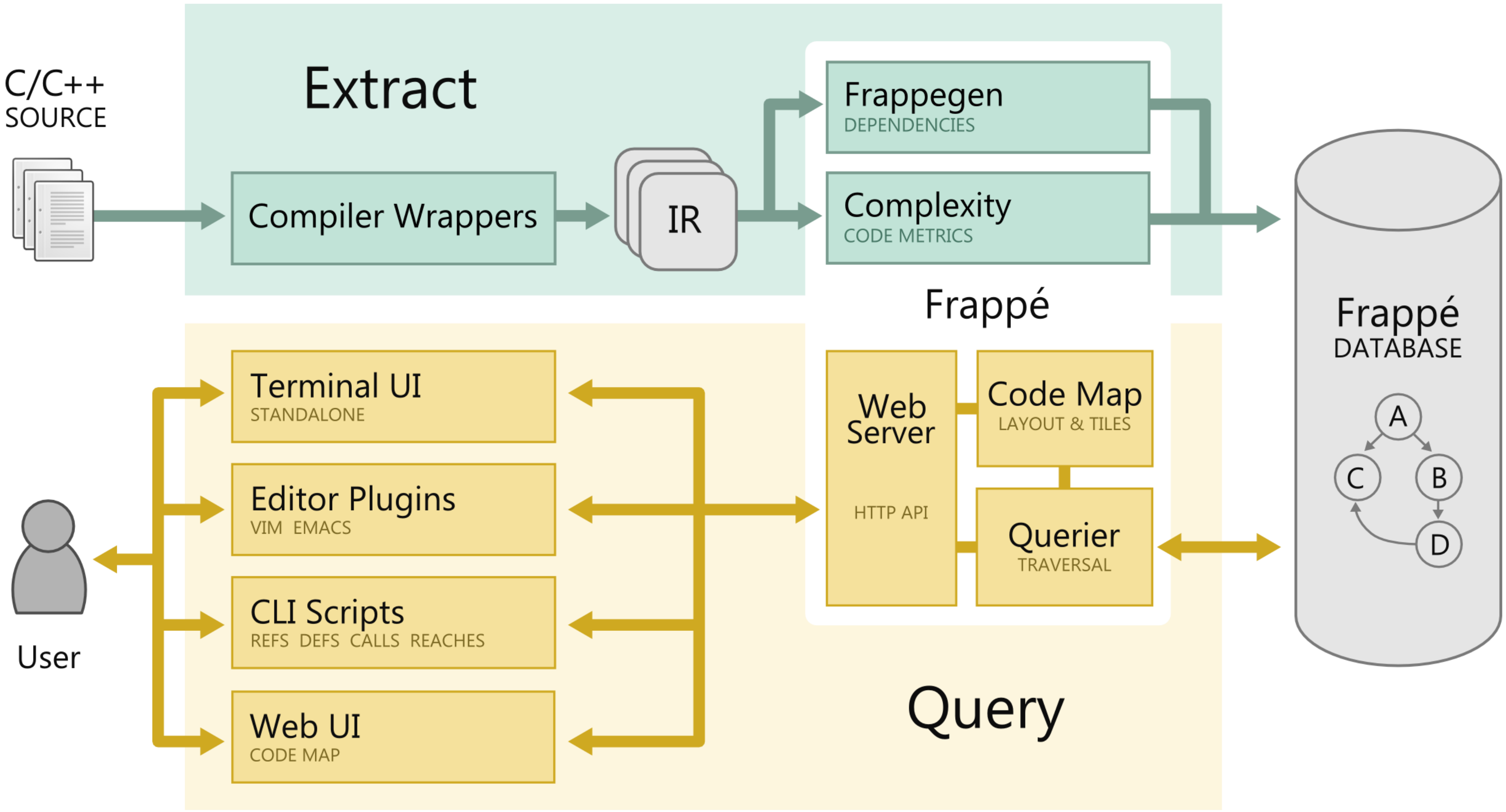


Impact Estimation



Frappé

A Code Comprehension Tool for Large Codebases



Current Performance

C Codebase - About 10 MLOC

Step	Time	Peak Memory
Extract dependencies (4 jobs)	2.5 hours	1 GB
Generate code map	1.5 hours	15GB
Search for specific function	<1s	< 1GB
Compute impact set (15 jumps)	<1s	< 1GB
Compute reachability (15 jumps)	<1s	< 1GB

Improving Extraction Time

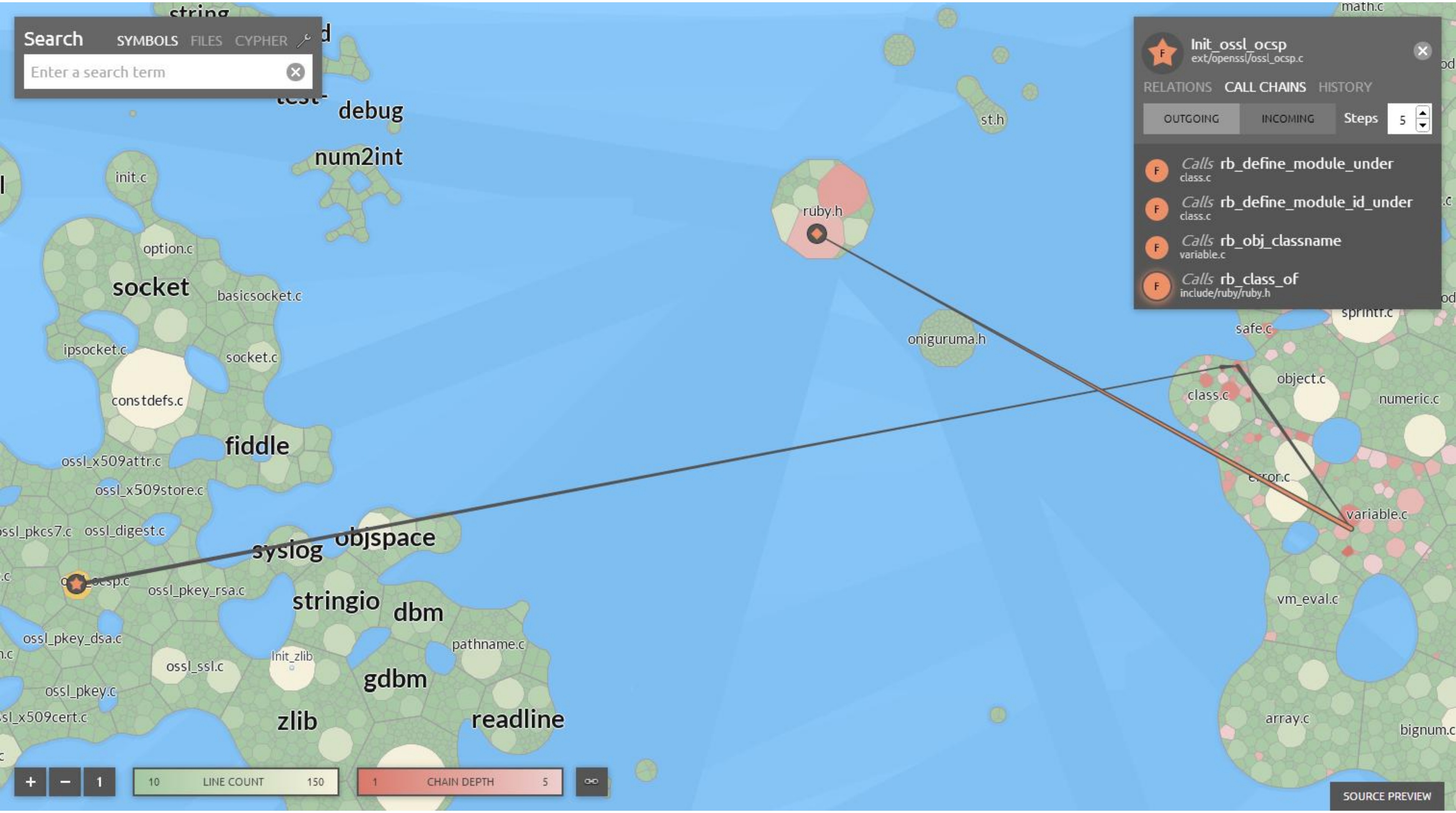
Solutions being investigated

- Build complete graph in nightly, then either:
- Pull down graph database and maintain locally
  - Database size has to be reasonable
- Create local graph of only modified files
  - Database size doesn't matter
  - Queries potentially slower (network latency)
  - Division between central and local databases makes transitive queries difficult
- Or, don't worry about local changes

ORACLE

Demonstration

Emacs, CLI Scripts, and Web UI



Going Forward

- Reduce initial graph extraction time
- Produce more precise graph
  - Pointer analysis (calls via function pointers, virtual calls, etc)
  - Template handling
- Filter on dynamic information
  - Show the commonly used paths
- Java support
  - Cross language (JNI)

ORACLE

Hardware and Software

ORACLE

Engineered to Work Together

ORACLE®