Finding API Usages with Pullbacks and Constraint Satisfaction

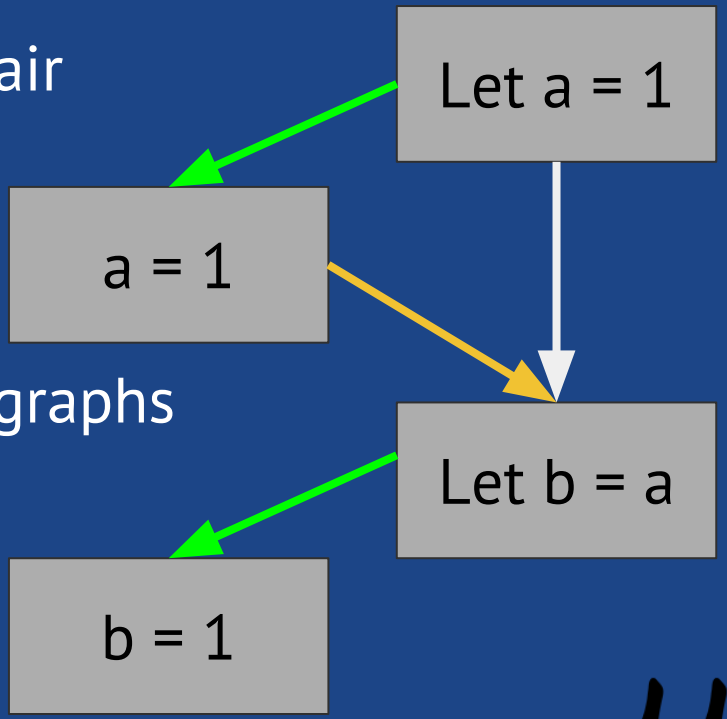
Rhys Olsen

Charter Communications
Boulder CUPLV

Finding common API usages in large codebases finds application in:

- Guided, low-cost automated repair synthesis
- Smarter code completion
- API usage recommendation

Idea: use *groums* (abstract program graphs that capture control flow and data dependency) to find API usages.



CUPLV

Define the following inductive types:

Command types $K = \{k_1, \ldots, k_i\}$

Data types $O = \{o_1, \ldots, o_j\}$.

Among our data types are function types

$M = \{$

$m_1 = (\mathrm{o}m_1 : O)(\overrightarrow{a_{m_1} : O}) : (\mathrm{r}m_1 : O),$

$\ldots,$

$m_k = (\mathrm{o}m_k : O)(\overrightarrow{a_{m_k} : O}) : (\mathrm{r}m_k : O)$

$\}$

CUPLV

An API usage tableau Ξ is as follows:

$$
\begin{aligned}
G = (\\
&V = ((S = \{s_i : O\}) \sqcup (C = \{c_i : K\})),\\
&E = (\\
&\quad (F = \{(f_i \in C, f_j \in C) \quad \forall i, j \in [C]\}) \sqcup\\
&\quad (D = \{(c_i \in K, o_j \in O) \; \forall i \in [K], j \in [O]\}) \sqcup\\
&\quad (U = \{(o_i \in O, c_j \in K) \; \forall i \in [O], j \in [K]\}))\\
)
\end{aligned}
$$

It describes all possible control-flow and data dependency matches.

CUPLV

A groum is a graph $G = (V, E)$ as follows:
$G = (V = (S' \subseteq S, C' \subseteq C), E = (F' \subseteq F, D' \subseteq D, U' \subseteq U))$

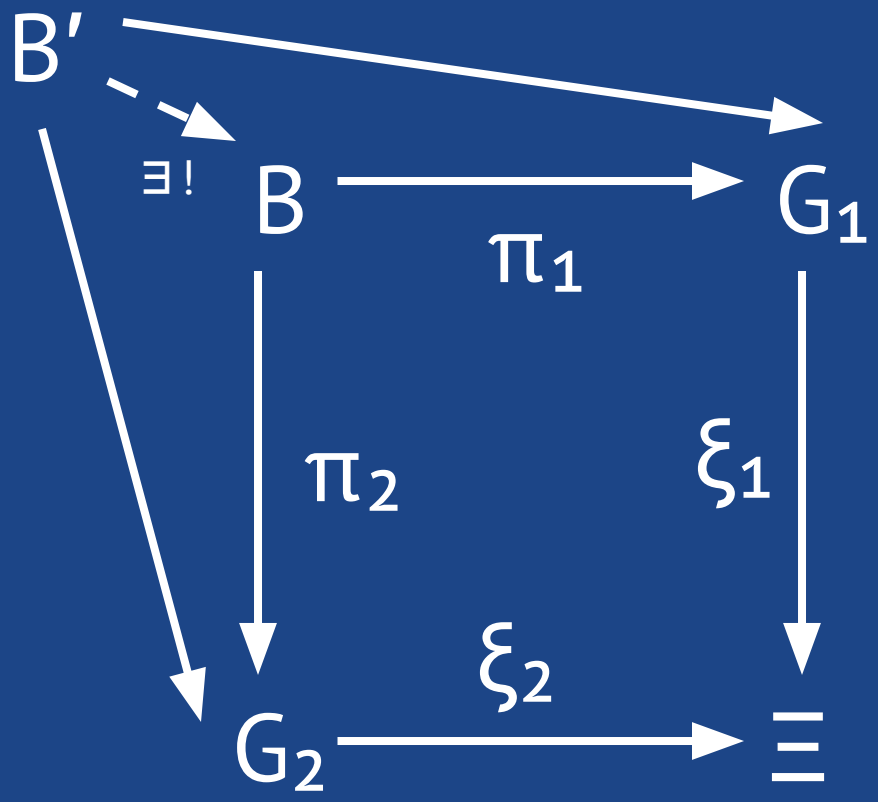
For each edge of kind t, we define compatibility propositions $P_{t,i,j}$. $P_{F,i,j}$ is command constructor equality and $P_{D,i,j}$ and $P_{U,i,j}$ are type equality between object i and returnee j and vice-versa, respectively.

Tl;dr: G must have sensible control flow and data dependency. In practice, G is safely derived from a checked and compiled program. It embeds into the tableau through ξ.

CUPLV

Lastly, we define the feasible set *B* of matches between any two groums G_1 and G_2 and their partial projections π_1 and π_2 into *B*.

CUPLV



CUPLV

For any two groums G_1 and G_2, we wish to find their maximal mutual embedding ϵ in B. This problem is NP-Complete, but tractable for instances of program graph size. We therefore pose it as a MAX-SAT problem.

For each edge of kind t, we define compatibility propositions $P_{t,i,j}$. $P_{F,i,j}$ is command constructor equality and $P_{D,i,j}$ and $P_{U,i,j}$ are type equality between object i and returnee j and vice-versa, respectively.

CUPLV

For two nodes or edges between different graphs, we define the relation $R_{x,y}$ on them if they've equal propositional forms.

$$R_{\text{AtLeastOne(x)}} = \bigwedge_{\forall i} R_{x,i}$$

$$R_{\text{AtMostOne(x)}} = \bigvee_{\forall i \neq j} \sim(R_{x,i} \wedge R_{x,j})$$

$$R_{\text{ExactlyOne(x)}} = R_{\text{AtLeastOne(x)}} \; R_{\text{AtMostOne(x)}}$$

Our maximal embedding is then given by:

$$\max_{\forall x} = w_x R_x \quad \text{s.t.} \quad R_{\text{ExactlyOne(x)}}$$

CUPLV

That's Cute, But Who Cares?

Finding API usages with pullbacks and constraint satisfaction is:

- **General**: can be *enhanced* under rich types and specs by adding new edge constraints, but the approach also functions without any.
- **Scalable and Precise**: 435% as fast as previous state of the art with better measured precision and recall in correct/incorrect API usage experiments.
- **Practical**: We mined a quarter million real Androids repos and derived correct API usages by statistical means, which can be used to patch real bugs in the wild. Talks are on to incorporate our work into a well-known code repository service.

Thank you!
Questions?

Sergio Mover, Sriram Sankaranarayanan, Rhys Olsen, and Bor-Yuh Evan Chang, Mining framework usage graphs from app corpora In International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 277-289 (2018).

Rhys Olsen, When Popular is Beautiful: Network Relationships as a Predictor of Correctness in Graph-Based Program Models. BS Thesis, University of Colorado at Boulder.

This material is based on research spon- sored in part by DARPA under agreement number FA8750-14- 2-0263. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

CUPLV