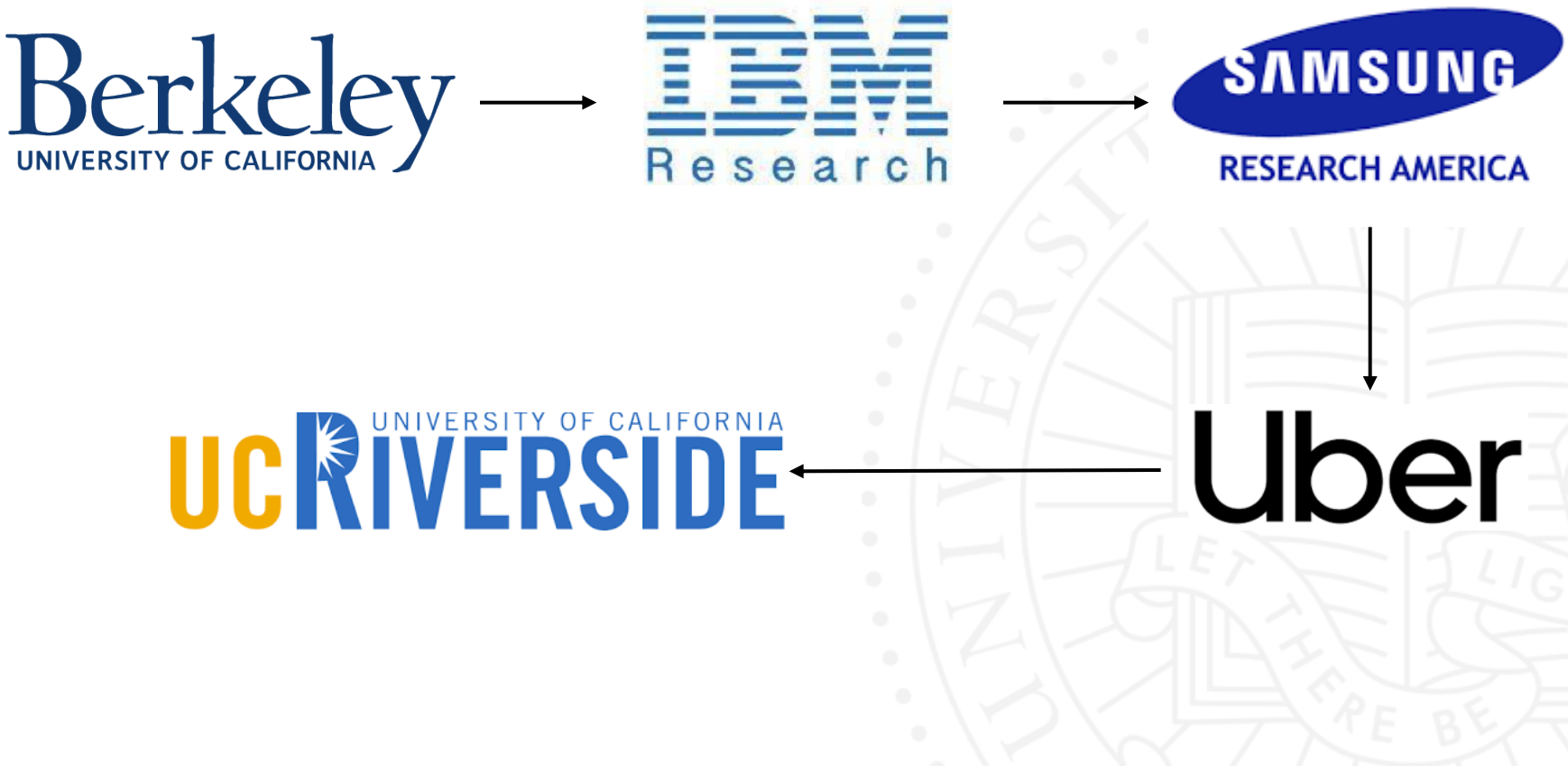
Program Analysis, in Industry and Academia

Manu Sridharan

UCR
UNIVERSITY OF CALIFORNIA, RIVERSIDE

My Journey


Berkeley
UNIVERSITY OF CALIFORNIA
IBM Research
SAMSUNG
RESEARCH AMERICA
Uber
UC RIVERSIDE UNIVERSITY OF CALIFORNIA

This talk

- My personal journey in industrial research / industry / academia
  - For a great overview of CS PhD jobs, see Kathleen Fisher's PLMW@PLDI'19 talk: http://bit.ly/careerOptions

- **My focus**: program analysis at scale
  - Static analysis
  - Type systems and type inference

Grad School

2002-2007

Ras Bodik
(my advisor)

Problem Space: Pointer Analysis

- Finds values for pointer variables
- Crucial building block
  - "What gets called by `x.m()`?"
  - "What code uses `untrustedInput()`?"
- Others formulated C pointer analysis as a CFL-reachability problem
- Ras's suggestion (~January 2003): Java pointer analysis using CFL-reachability

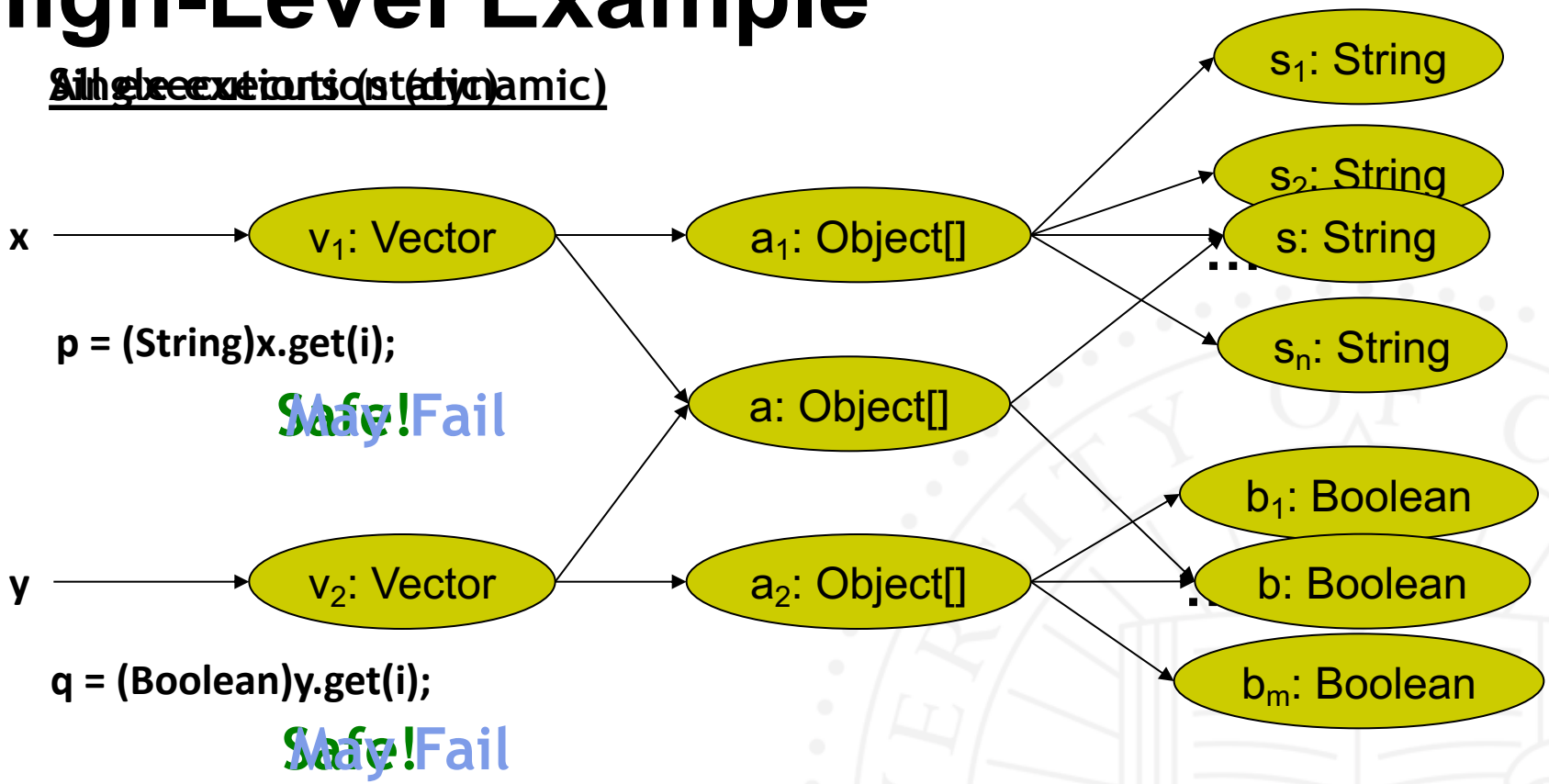
Target programs: Benchmarks

- Standard suites used in performance papers
  - SpecJVM98
  - Dacapo
- Tried to use "real-world" clients
  - E.g., proving safety of downcasts
  - Atypical for pointer analysis papers at the time
- "Large" programs
  - Hundreds of thousands of lines (with libraries)
  - I learned later they weren't that big…

Making progress

- Took a while (> 1 year) to ramp up
  - Understanding the (extensive) literature
  - Learning the engineering "tricks"
- A few false starts, paper rejections
  - Spent multiple months on an idea that didn't work
- **Insight**: Java analysis has a "balanced parentheses" structure
- Led to a refinement-based technique with improved scalability and precision

High-Level Example



- Must merge some objects (decidability)
- Too much merging: precision loss
- Too little merging: space explosion (typically exponential)
- Our technique: unmerge through refinement

Branching out

- Idea: balanced parentheses for slicing
- Implemented with Steve Fink from IBM
  - Ras had long collaborated with IBM
  - Used newly open-source WALA framework
- Wrote a fun paper ("Thin Slicing")
- Realized how much I enjoy close collaboration...

IBM Research

2008-2013

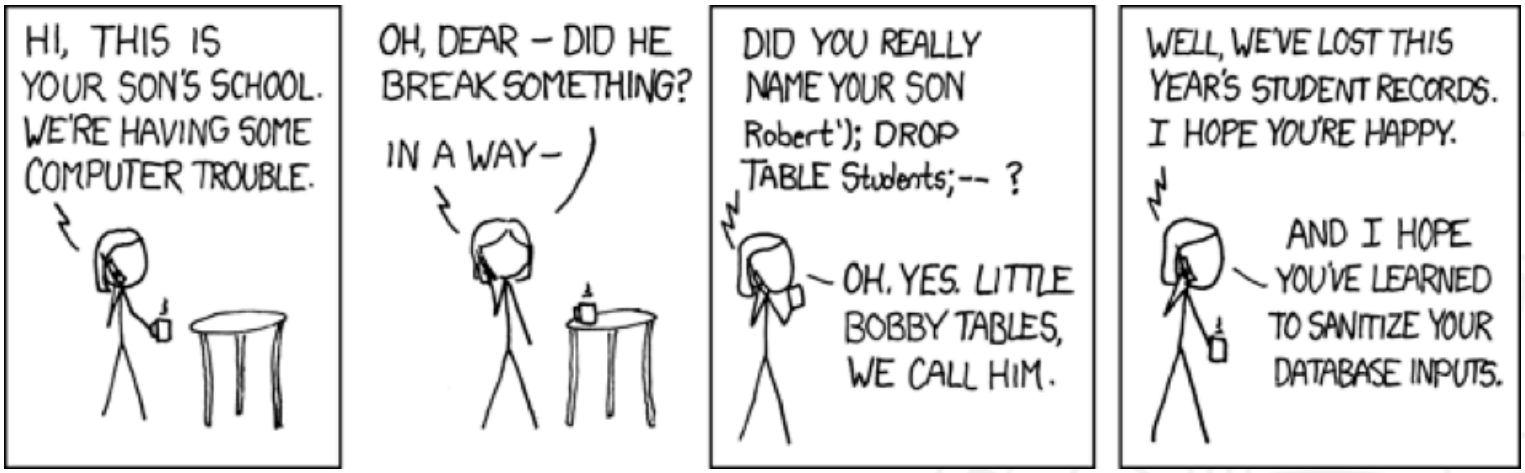
Balance in Industrial Research

- Some work should "pay the bills"
  - Applying research results to a product
  - Lower risk (but still publishable!)
  - Tangible impact
- Other work should be forward looking
  - Speculative, but with a story for eventual impact
  - Sometimes turns into "pay the bills" work
- I really enjoyed this balance!
  - Get to have a portfolio of impact

Information flow vulnerabilities

- Untrusted data used in sensitive operation
  - E.g., SQL injection
- Leaking of confidential data
  - E.g., showing exception stack traces

(paying the bills)

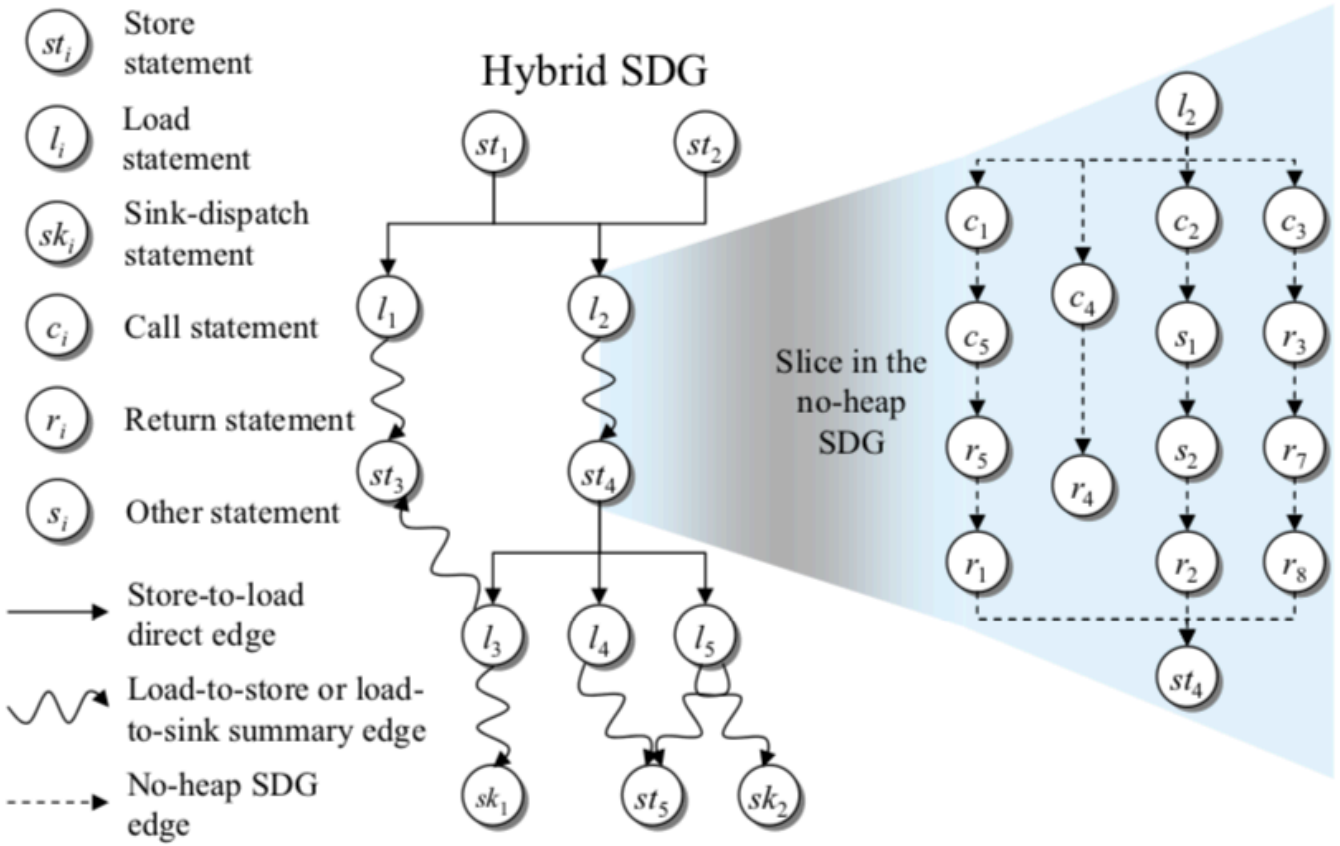


https://www.xkcd.com/327/

Target programs: Customer code

- Much different than what I saw in grad school
- Enormous!
  - Dozens of library jars in classpath
  - Even building class hierarchy had to be optimized
- Often based on complex frameworks
  - No `main()` method in application
  - Framework calls into app based on XML config
  - Hard to analyze!

Technique: hybrid thin slicing



The real world

- Many heuristics needed for huge programs
  - Cutoffs, timeouts, etc.
  - Hard to build robustly
- Frameworks led to many missed issues
- Tools sell by doing well in "proof-of-concept"
  - Sales engineer tries tool on customer code
  - Find some bugs, don't miss obvious ones
  - "Verifying" the code not relevant
- So, design analysis to work well in PoCs

Eventual approach

- No whole-program pointer analysis!
- Use simple class hierarchy + local tracking of data flow and aliasing
- Built a sophisticated tool F4F to make modeling of frameworks easier
- Successful product (still sold today)

Shifting gears: JavaScript

- ~2011, heard from Julian Dolby about challenges in analyzing JavaScript web apps
- We started studying challenges in core static analysis for JavaScript
- Forward looking: no particular customer

Reflective code

Java: `x.f = y`

If no `f` field, compile error

JavaScript: `x[e] = y`

If field doesn't exist, create it!

Disaster for pointer analysis

Used in real world 😐

```
var e = “blur,focus,load”.split(“,”);
// e is [“blur”,”focus”,”load”]
for (var i=0;i<e.length;i++) {
  var o = e[i];
  jQuery.fn[o] = function() { … };
  jQuery.fn[“un”+o] = function() { … };
  jQuery.fn[“one”+o] = function() { … };
}
```

Techniques

- Improvements to traditional pointer analysis
  - Correlation tracking
  - Dynamic determinacy analysis
- Scalable, approximate (unsound) call graphs
  - Used to pay the bills! AppScan JS analysis

Samsung Research

2013-2016

Shifting gears: Performance

TIZEN™

- A push for running JS / web apps on devices
- Performance was lagging behind
  - Dynamic features hard to optimize
  - Hard to run JIT, GC with resource constraints
- Goal: ahead-of-time compilation for JS

Target programs: web apps

- Often, not much client code
  - Many had < 10,000 LoC
- But, relied on complex frameworks
  - And web browser itself is complex!
- When optimizing, **can't be unsound**

Technique: type inference

- Defined a type system for a rich subset of JS
  - Enabled efficient code generation
- Defined a (global) type inference algorithm, to handle extant code
- Built a full compiler backend (compiled to C)

$$\boxed{X_R, \Gamma \vdash e : X \mid C}$$

$$\textsc{C-Int}\ \frac{\mathsf{fresh}\ X}{X_R, \Gamma \vdash n : X \mid X^{\mathsf{r}} \equiv \mathsf{int}} \qquad \textsc{C-Var}\ \frac{\Gamma(x) = X}{X_R, \Gamma \vdash x : X \mid \emptyset} \qquad \textsc{C-This}\ \frac{}{X_R, \Gamma \vdash \mathsf{this} : X_R \mid \emptyset}$$

$$\textsc{C-VarDecl}\ \frac{X_R, \Gamma\,[x \mapsto X_1] \vdash e_1 : Y_1 \mid C_1 \qquad \mathsf{fresh}\ X_1 \qquad X_R, \Gamma\,[x \mapsto X_1] \vdash e_2 : X \mid C_2}{X_R, \Gamma \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : X \mid C_1 \wedge C_2 \wedge Y_1^{\mathsf{r}} <: X_1^{\mathsf{r}} \wedge Y_1^{\mathsf{w}} <: X_1^{\mathsf{w}}}$$

$$\textsc{C-VarUpd}\ \frac{x : X_1 \in \Gamma \qquad X_R, \Gamma \vdash e_1 : X \mid C_1}{X_R, \Gamma \vdash x := e_1 : X \mid C_1 \wedge X^{\mathsf{r}} <: X_1^{\mathsf{r}} \wedge X^{\mathsf{w}} <: X_1^{\mathsf{w}}} \qquad \textsc{C-Null}\ \frac{\mathsf{fresh}\ X}{X_R, \Gamma \vdash \mathsf{null} : X \mid X^{\mathsf{w}} <: \langle\,\rangle}$$

$$\textsc{C-MethDecl}\ \frac{\mathsf{fresh}\ Y_R, Y_1, X \quad \mathsf{has\_this}(e) \quad Y_R, \Gamma\,[x \mapsto Y_1] \vdash e : Y_2 \mid C}{X_R, \Gamma \vdash \mathsf{function}\ (x)\ \{e\} : X \mid C \wedge Y_R^{\mathsf{w}} <: \langle\,\rangle \wedge \mathsf{concrete}(Y_R) \wedge \mathsf{notproto}(Y_R) \wedge X^{\mathsf{r}} \equiv ([Y_R]\ Y_1 \Rightarrow Y_2)}$$

$$\textsc{C-MethApp}\ \frac{\mathsf{fresh}\ X_M, Y_R, X_3, X \qquad X_R, \Gamma \vdash e_1 : X_1 \mid C_1 \qquad X_R, \Gamma \vdash e_2 : X_2 \mid C_2}{\begin{array}{l} X_R, \Gamma \vdash e_1.a\,(e_2) : X \mid \quad C_1 \wedge C_2 \wedge X_1^{\mathsf{r}} <: \langle a : X_M \rangle \wedge X_M^{\mathsf{r}} \equiv ([Y_R]\ X_3 \Rightarrow X) \wedge \mathsf{strip}(X_M) \wedge \mathsf{concrete}(X_1) \\ \qquad\qquad \wedge\ \mathsf{concrete}(Y_R) \wedge Y_R^{\mathsf{w}} <: \langle\,\rangle \wedge \mathsf{notproto}(Y_R) \wedge X_2^{\mathsf{r}} <: X_3^{\mathsf{r}} \wedge X_2^{\mathsf{w}} <: X_3^{\mathsf{w}} \end{array}}$$

$$\textsc{C-ObjEmp}\ \frac{}{X_R, \Gamma \vdash \{\cdot\} : X \mid \mathsf{proto}(X) \wedge X^{\mathsf{r}} \equiv \langle\,\rangle \wedge X^{\mathsf{mr}} \equiv \langle\,\rangle} \qquad \textsc{C-Attr}\ \frac{\mathsf{fresh}\ X \quad X_R, \Gamma \vdash e : X_1 \mid C}{X_R, \Gamma \vdash e.a : X \mid C \wedge X_1^{\mathsf{r}} <: \langle a : X \rangle \wedge \mathsf{notmethod}(X)}$$

$$\textsc{C-AttrUpd}\ \frac{\mathsf{fresh}\ X_f \quad X_R, \Gamma \vdash e_1 : X_b \mid C_1 \quad X_R, \Gamma \vdash e : X_v \mid C_2}{X_R, \Gamma \vdash e_1.a := e : X_v \mid C_1 \wedge C_2 \wedge X_b^{\mathsf{w}} <: \langle a : X_f \rangle \wedge X_v^{\mathsf{r}} <: X_f^{\mathsf{r}} \wedge X_v^{\mathsf{w}} <: X_f^{\mathsf{w}} \wedge \mathsf{attach}(X_b, X_f, X_v)}$$

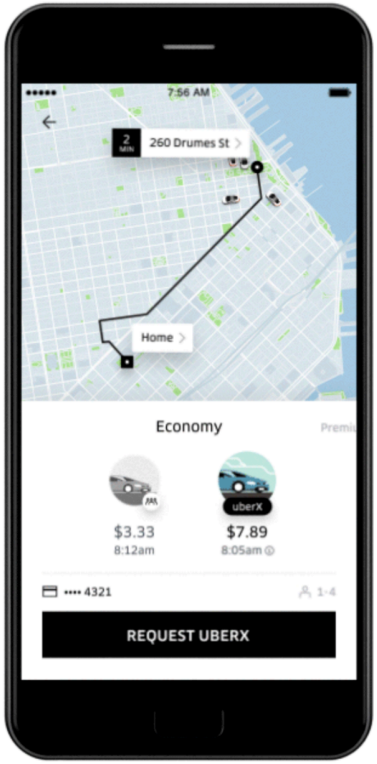
$$\textsc{C-ObjLit}\ \frac{\mathsf{fresh}\ X \qquad \forall i \in 1..n.\ \mathsf{fresh}\ X_i \qquad \forall i \in 1..n.\ X_R, \Gamma \vdash e_i : Y_i \mid C_i \quad X_R, \Gamma \vdash e_p : X_p \mid C_p}{\begin{array}{l} X_R, \Gamma \vdash \{a_1 : e_1, \ldots, a_n : e_n\}\ \mathsf{proto}\ e_p : X \mid \quad C_p \wedge \bigwedge_i (C_i \wedge Y_i^{\mathsf{r}} <: X_i^{\mathsf{r}} \wedge Y_i^{\mathsf{w}} <: X_i^{\mathsf{w}} \wedge \mathsf{attach}(X, X_i, Y_i)) \\ \qquad\qquad \wedge\ X^{\mathsf{w}} \equiv \langle a_1 : X_1, \ldots, a_n : X_n \rangle \wedge X^{\mathsf{r}} <: X_p^{\mathsf{r}} \wedge X_p^{\mathsf{r}} <: X^{\mathsf{r}} \setminus \{a_1, \ldots, a_n\} \\ \qquad\qquad \wedge\ \mathsf{proto}(X) \wedge \mathsf{proto}(X_p) \wedge X^{\mathsf{mr}} <: X_p^{\mathsf{mr}} \wedge X^{\mathsf{mw}} <: X_p^{\mathsf{mw}} \end{array}}$$

Figure 8: Constraint generation.

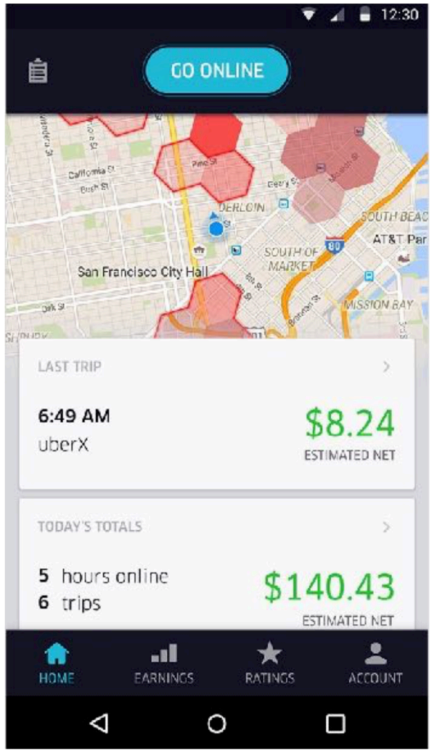
Uber

2017-2018

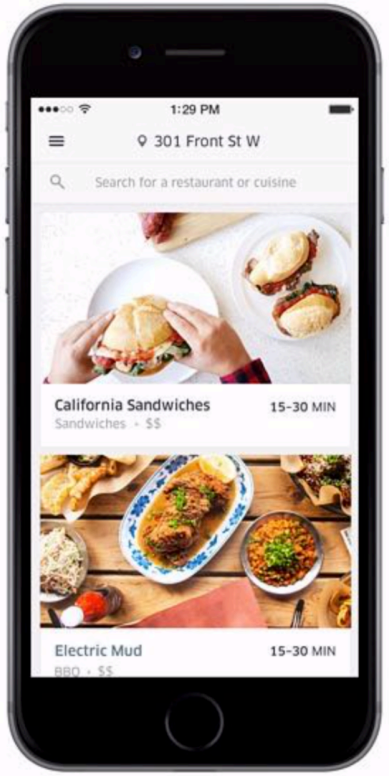
Uber Apps



Rider



Driver



Eats

iOS and Android

App reliability: crucial

Rider app crash: can't get home

Driver app crash: can't earn

Using apps involves payments

Apps take significant time to patch

Target programs: our own code

- We can rewrite code to work well with tool!
- Goal: integrate with development process
  - Fast, modular analysis
  - Understandable error messages
- Can require reasonable annotation burden

Technique: Pluggable Types

- Idea: add extra type checking to compiler
  - Leveraging source code annotations
- Pioneered by Checker Framework
- **NullAway**: fast, practical NPE prevention
  - Engineered for speed
  - Soundness tradeoffs to reduce annotations
  - Runs on every Android build at Uber
  - https://github.com/uber/NullAway, https://arxiv.org/abs/1907.02127

Example: Nullability

```
static void log(Object x) {
  System.out.println(x.toString());
}
static void foo() {
  log(null); Error: cannot pass null to @NonNull
}              parameter x
```

Example: Nullability

```
static void log(@Nullable Object x) {
  System.out.println(x.toString());
}
static void foo() {
  log(null);
}
```

Error: de-referencing x may yield NPE

Example: Nullability

```
static void log(@Nullable Object x) {
  if (x == null) return;
  System.out.println(x.toString());
}
static void foo() {
  log(null);
}
```

Other Uber projects

- Type-based detection of multithreading bugs
  - Specialized to stream APIs
  - Also running on every Android build
- Optimization of Swift protocols
  - 12% speedup in app startup
  - Upstreamed to Apple
- Auto-deletion of stale feature flags
  - Stale flags hurt reliability, code readability
  - Hundreds of flags removed

UC Riverside

2019-present

What's next?

- I still like to pay the bills!
  - Make time for open-source hacking
  - Polish tools, within reason
- Invest in teaching / advising
- Research: greater risk / time horizon
- Find good collaborators!
  - Students, faculty (Riverside / elsewhere), industry

Tips

- Be aware of “adjacent” areas
  - Read broadly (papers, Hacker News, …)
  - Make time for it
- Networking and visibility matters!
  - Talk to people
  - Do open source, give talks (with video)
- Value in different perspectives on a problem
  - The “right”, academic solution
  - Dealing with existing code as-is
  - Working with developers

Thanks!