



Pavol Cerny



Bor-Yuh Evan Chang



Sriram Sankaranarayanan

Programming Languages and Verification at the University of Colorado Boulder



PLV research, right?



PLV research at CU has *breadth!*

PLV research at CU has *breadth*!

How do we **assist reasoning**
about programs?

program analysis, developer tools



PLV research at CU has *breadth*!

How do we **assist reasoning**
about programs?

program analysis, developer tools



How do we get **reliable,**
secure software?

verification, model checking



PLV research at CU has *breadth*!

How do we let computers
code for us?
synthesis



How do we **assist reasoning**
about programs?
program analysis, developer tools

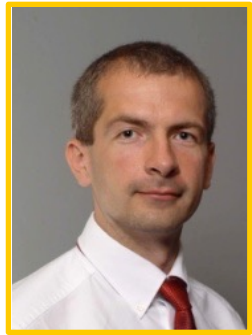


How do we get **reliable,**
secure software?
verification, model checking



PLV research at CU has *breadth*!

How do we let computers
code for us?
synthesis



How do we **assist reasoning**
about programs?
program analysis, developer tools



How do we get **reliable,**
secure software?
verification, model checking



PLV researchers at CU *collaborate!*

synthesis



program analysis



verification

PLV researchers at CU *collaborate!*

synthesis



Concurrency Synthesis



program analysis



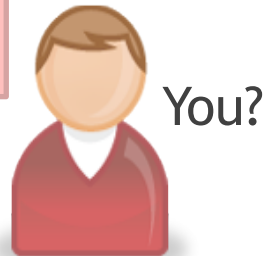
verification

PLV researchers at CU *collaborate!*

synthesis



Concurrency Synthesis



program analysis

Mobile Security

You?



verification

PLV researchers at CU *collaborate!*

synthesis



Shawn Meier



Concurrency Synthesis



You?

You?



Transferring Bug Fixes



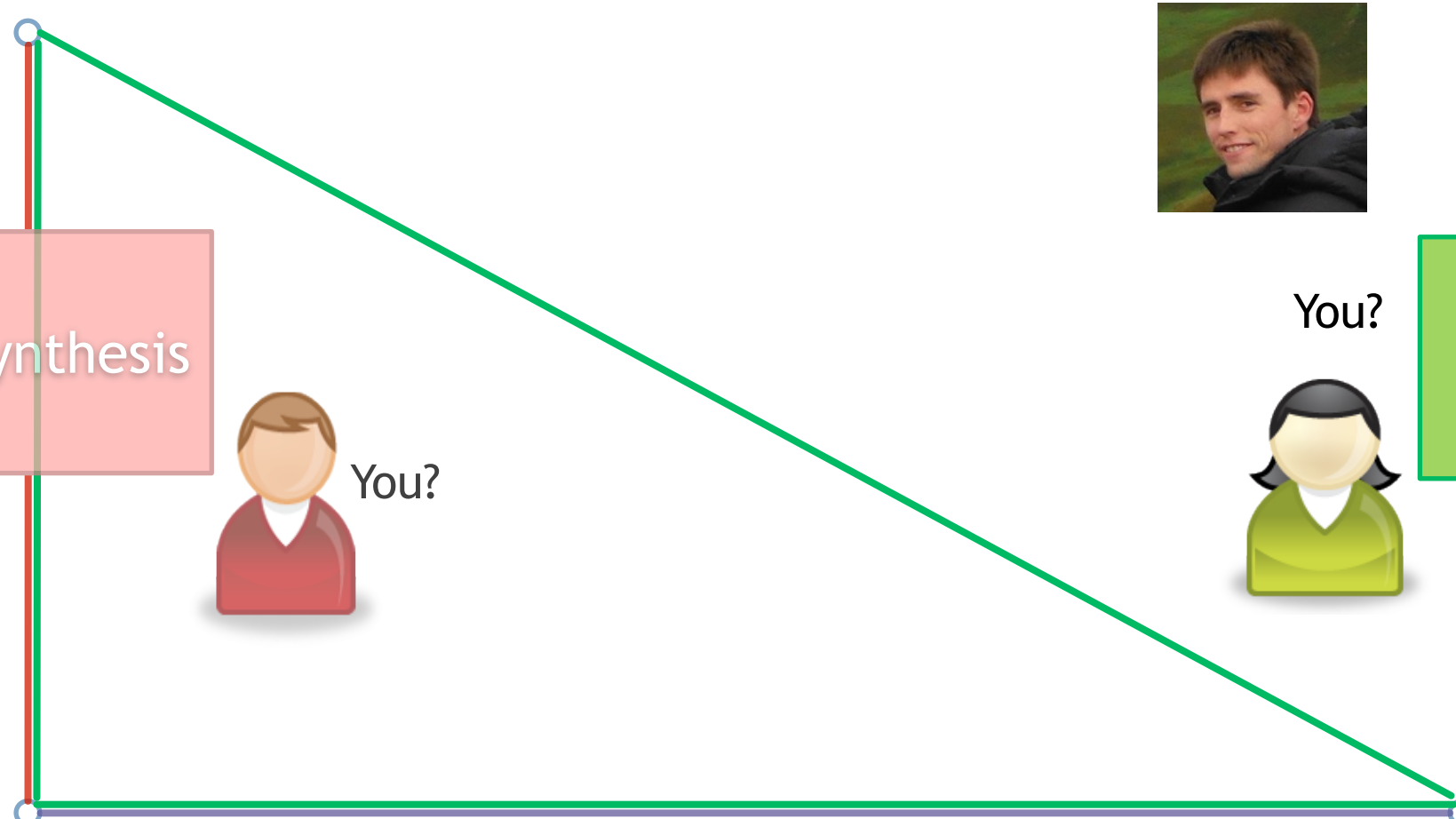
program analysis

Mobile Security

You?



verification





JORGE CHAM © 2005

www.phdcomics.com



PLV is about compassion!

Do you Android?



Do you Android?



Do you Android?



Do you Android?



I'm not making this up ...



I'm not making this up ...



stackoverflow


Questions

Tags

Tour

Users

Android: Crash on rotation, horizontal to vertical

Crash is detected after rotating phone in Gmail Sync now view 

[phonegap](#) >

[important bug]cordova 1.9 crash on rotation android

5 posts by 2 authors   +1



stackoverflow

Questions

Tags

Tour

Users

App crashes when rotating Samsung phone



androidterm

Android Terminal Emulator

[Project Home](#)

[Downloads](#)

[Wiki](#)

Issues

[Source](#)

[New issue](#)

Search

Open issues



for

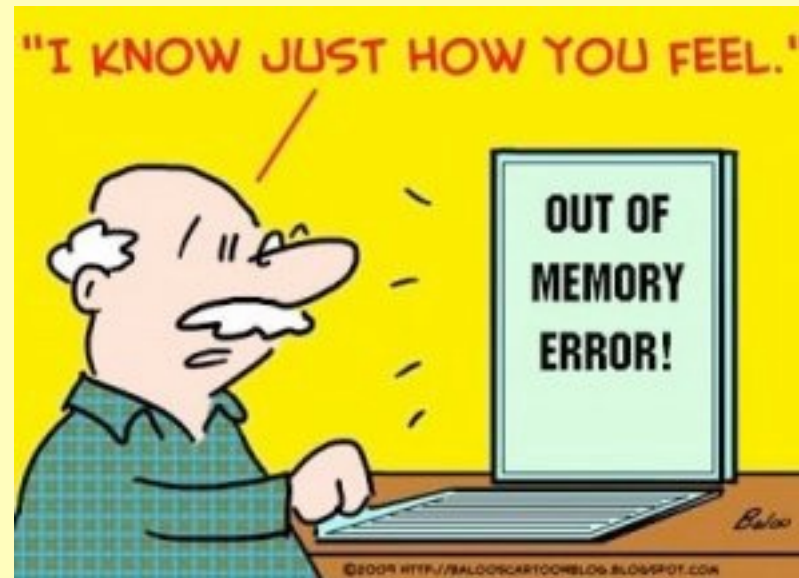
★ **Issue 20: Crashes when rotating phone horizontally**

1 person starred this issue and may be notified of changes.

I'm not making this up ...



Reason:



Gmail Sync now view

phonegap >
[important bug]com
5 posts by 2 authors

StackOverflow

Tour

Users

App crashes when rotating Samsung phone



androidterm

Android Terminal Emulator

[Project Home](#) [Downloads](#) [Wiki](#) [Issues](#) [Source](#)

[New issue](#) Search for

★ **Issue 20: Crashes when rotating phone horizontally**
1 person starred this issue and may be notified of changes.



State of practice for debugging leaks ...



The screenshot shows a web browser window displaying the Android Developers Blog. The address bar shows the URL: `android-developers.blogspot.dk/2011/03/memory-analysis-for-android.html`. The page features a dark blue header with the Android logo and the text "Android Developers Blog". Below the header, the article title "Memory Analysis for Android Applications" is displayed, dated "24 MARCH 2011". The author is identified as Patrick Dubroy. The article text discusses memory management in the Dalvik runtime and mentions tools like the Allocation Tracker in DDMS. A sidebar on the left contains a search bar and an archive menu with months from 2012 to May 2011.

Developers

24 MARCH 2011

Memory Analysis for Android Applications

[This post is by Patrick Dubroy, an Android engineer who writes about programming, usability, and interaction on his personal blog. — Tim Bray]

The Dalvik runtime may be garbage-collected, but that doesn't mean you can ignore memory management. You should be especially mindful of memory usage on mobile devices, where memory is more constrained. In this article, we're going to take a look at some of the memory profiling tools in the Android SDK that can help you trim your application's memory usage.

Some memory usage problems are obvious. For example, if your app leaks memory every time the user touches the screen, it will probably trigger an `OutOfMemoryError` eventually and crash your app. Other problems are more subtle, and may just degrade the performance of both your app (as garbage collections are more frequent and take longer) and the entire system.

Tools of the trade

The Android SDK provides two main ways of profiling the memory usage of an app: the *Allocation Tracker* tab in DDMS, and heap dumps. The Allocation Tracker is useful when you want to get a sense of what kinds of allocation are happening over a

State of practice for debugging leaks ...



I. Run the app

State of practice for debugging leaks ...



The screenshot shows the Dalvik Debug Monitor (DDMS) interface, specifically the VM Heap tab. The interface displays heap usage statistics for a client. The "Cause GC" button is visible. The "Heap updates will happen after every GC for this client" message is shown. The "Cause GC" button is highlighted with a red circle.

ID	Heap Size	Allocated	Free	% Used	# Objects
1	8.570 MB	8.452 MB	121.320 KB	98.62%	59,281

Display: Stats

Type	Count	Total Size	Smallest	Largest	Median	Average
free	1,772	107.312 KB	16 B	48.297 KB	24 B	62 B
data object	40,528	1.229 MB	16 B	1.047 KB	32 B	31 B
class object	2,187	637.234 KB	168 B	34.125 KB	168 B	298 B
1-byte array (byte[], boolean[])	2,247	5.654 MB	24 B	1.500 MB	48 B	2.576 KB
2-byte array (short[], char[])	10,373	677.352 KB	24 B	28.023 KB	48 B	66 B
4-byte array (object[], int[], float[])	3,663	276.812 KB	24 B	16.023 KB	40 B	77 B
8-byte array (long[], double[])	283	14.875 KB	24 B	4.000 KB	32 B	53 B
non-Java object	92	14.219 KB	16 B	8.023 KB	32 B	158 B

1. Run the app
2. Watch the heap usage

State of practice for debugging leaks ...



1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and **finally** find the culprit!

The screenshot displays two windows from an Android development environment. The top window is the Dalvik Debug Monitor, showing heap statistics. The bottom window is the Eclipse Memory Analyzer, showing a detailed view of a memory leak.

Dalvik Debug Monitor - VM Heap

ID	Heap Size	Allocated	Free	% Used	# Objects
1	8.570 MB	8.452 MB	127.320 KB	98.62%	59,281

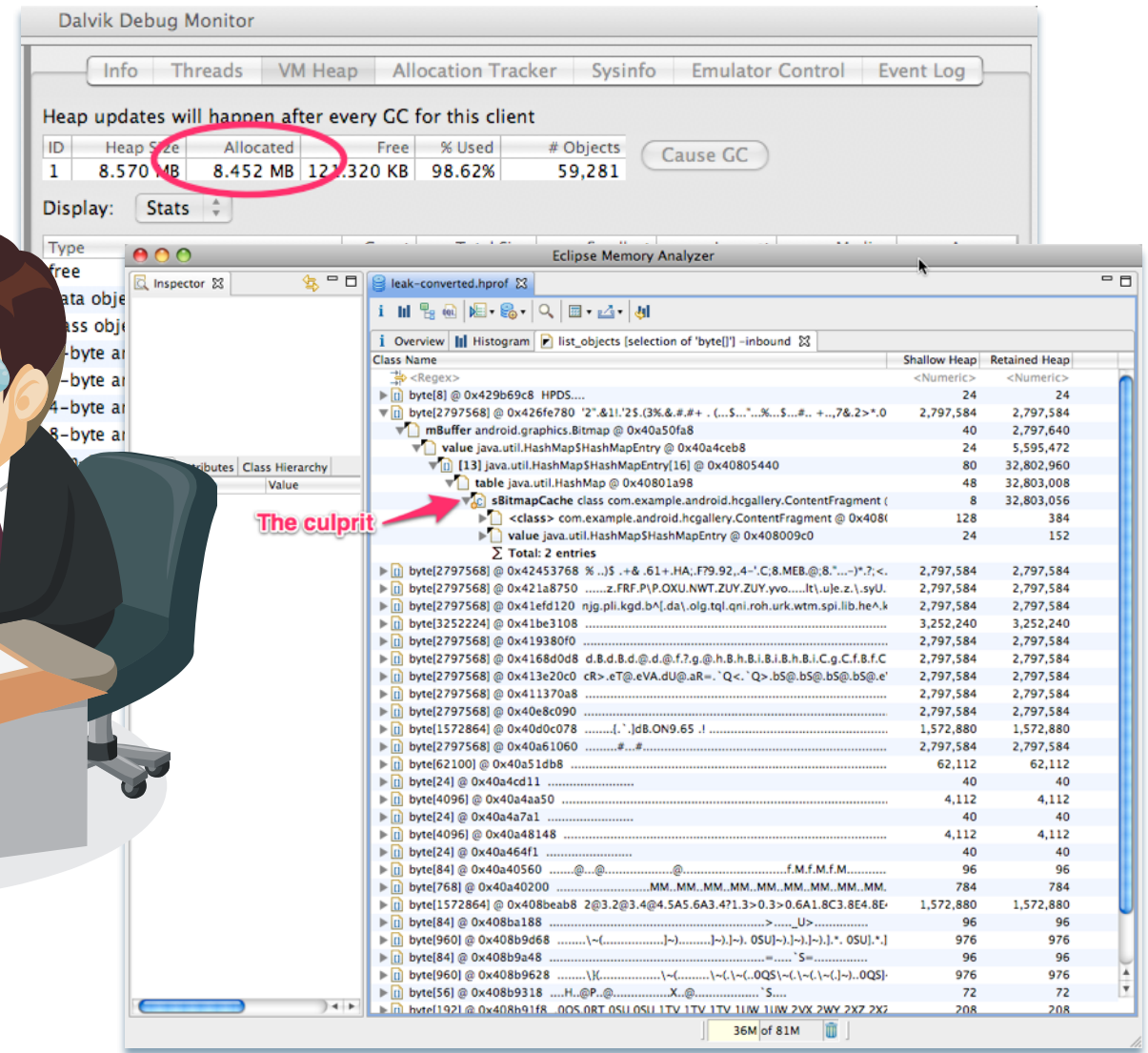
Eclipse Memory Analyzer - Overview

Class Name	Shallow Heap	Retained Heap
<Regex>	<Numeric>	<Numeric>
byte[8] @ 0x429b69c8	24	24
byte[2797568] @ 0x426fe780	2,797,584	2,797,584
mBuffer android.graphics.Bitmap @ 0x40a50fa8	40	2,797,640
value java.util.HashMap\$HashMapEntry @ 0x40a4ceb8	24	5,595,472
[13] java.util.HashMap\$HashMapEntry[16] @ 0x40805440	80	32,802,960
table java.util.HashMap @ 0x40801a98	48	32,803,008
sBitmapCache class com.example.android.hcgallery.ContentFragment (8	32,803,056
<class> com.example.android.hcgallery.ContentFragment @ 0x40801a98	128	384
value java.util.HashMap\$HashMapEntry @ 0x408009c0	24	152

Total: 2 entries

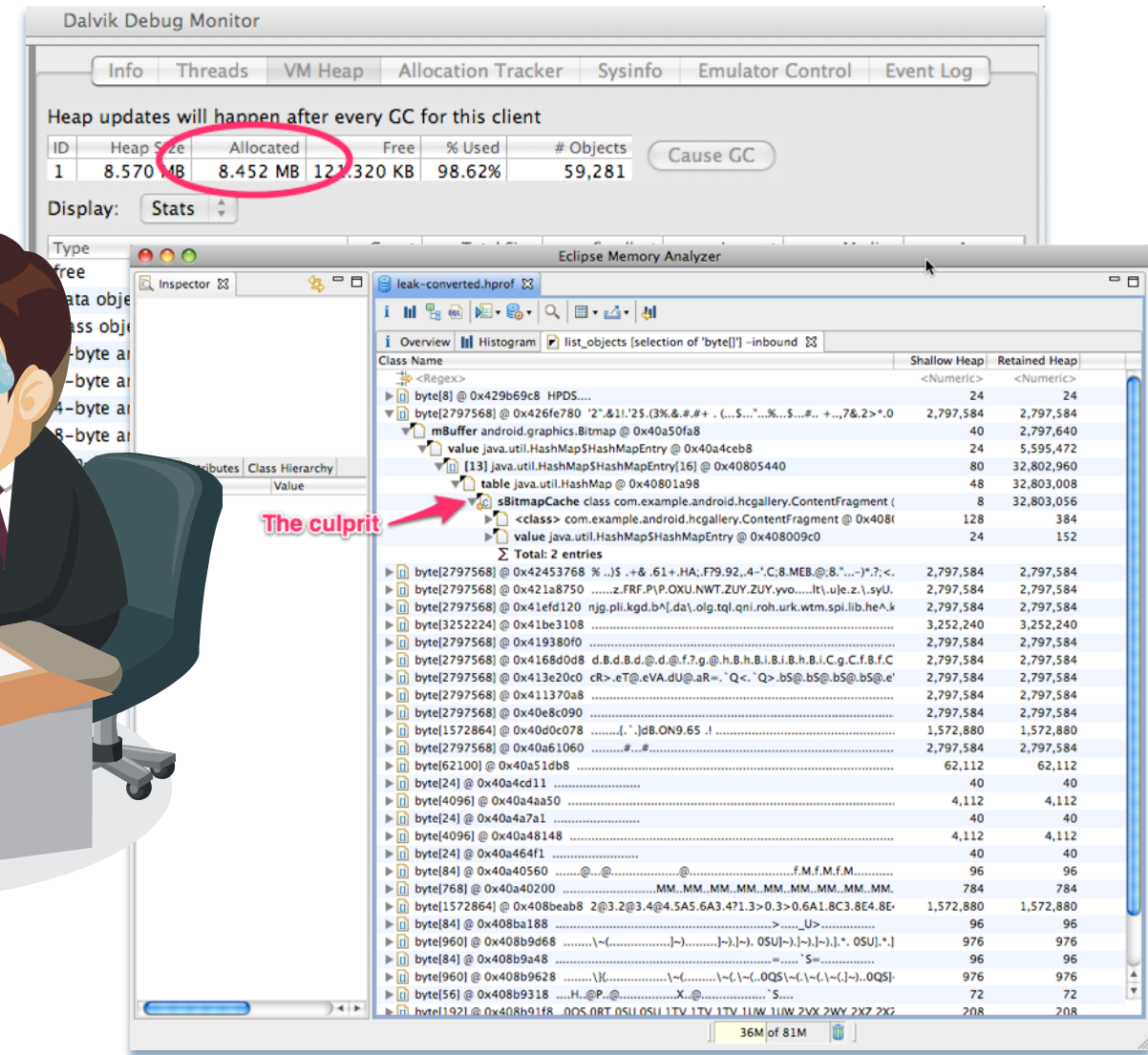
The culprit is highlighted with a red arrow pointing to the `sBitmapCache` class.

State of practice for debugging leaks ...



1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and **finally** find the culprit!

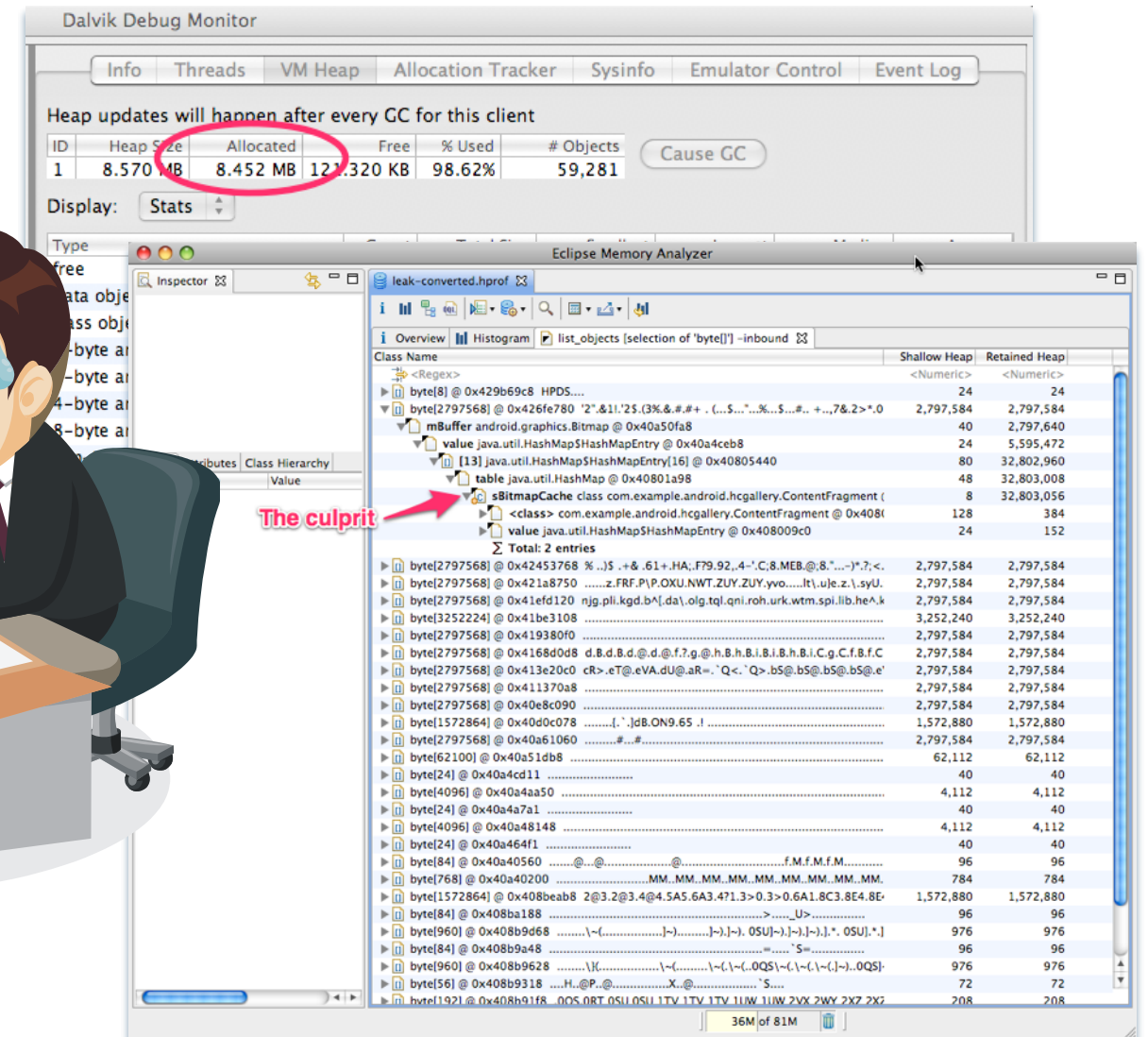
State of practice for debugging leaks ...



1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and **finally** find the culprit!
4. **Commit** a **bugfix**



State of practice for debugging leaks ...

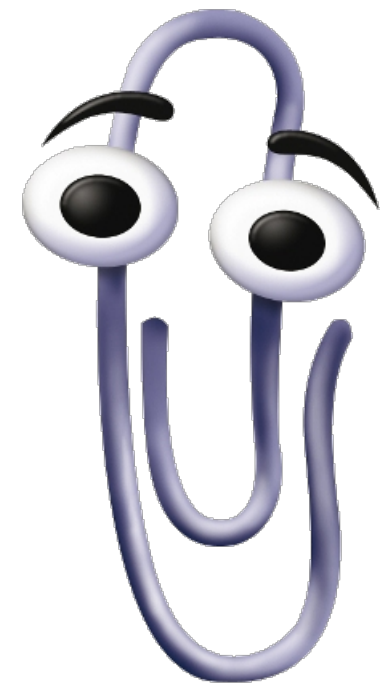


1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and **finally** find the culprit!
4. **Commit** a **bugfix**
5. **Bugfix** is picked up by our tool **Fixr**





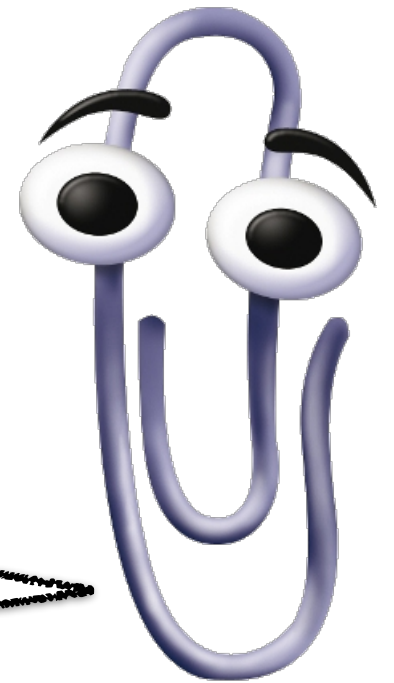
A **Fixr**-enabled IDE responds ...

A Fixr-enabled IDE responds ...




A Fixr-enabled IDE responds ...

It looks like you've created a memory leak like  and 100,000 others. Would you like to apply  ?

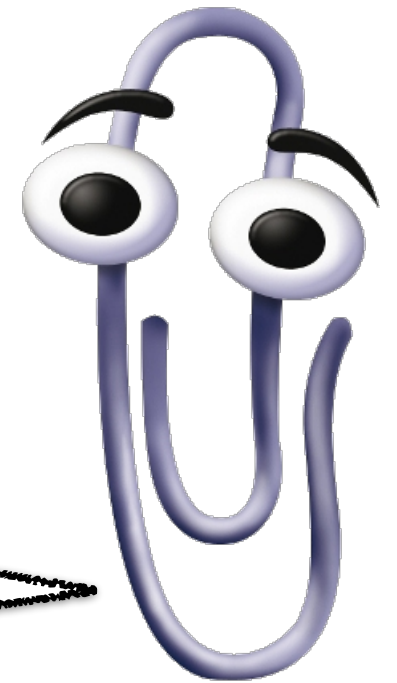


A Fixr-enabled IDE responds ...

It looks like you've created a memory leak like  and 100,000 others. Would you like to apply



?



the **bugfix** is “transferred”

Program synthesis for network updates

Program synthesis for network updates

Program synthesis



Program synthesis for network updates

Program synthesis



Network Updates



Program synthesis for network updates

Program synthesis



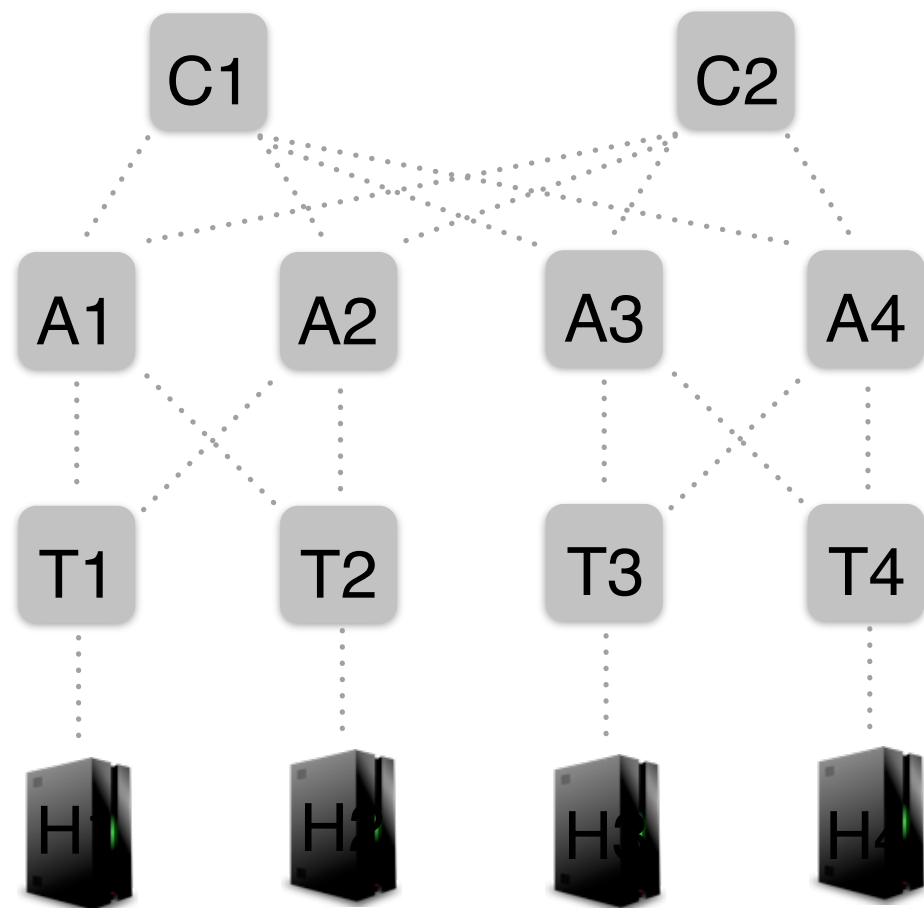
Network Updates



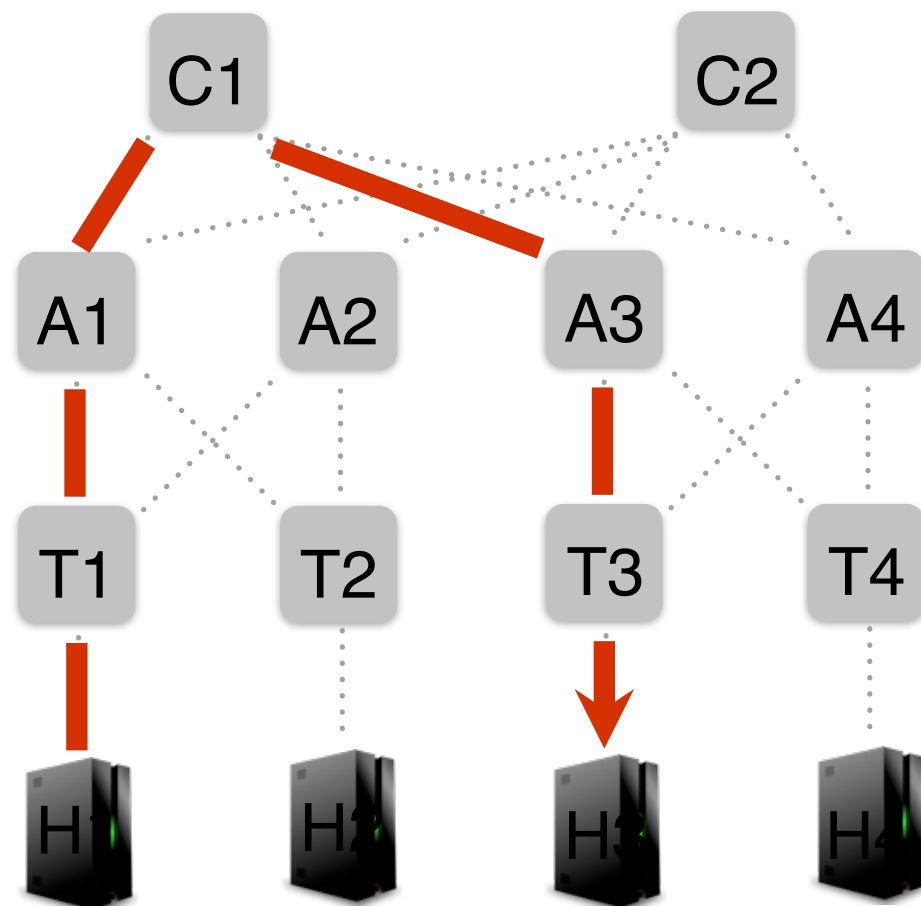
Work to appear at PLDI 2015
First author: Jed McClurg
(second-year PhD student)



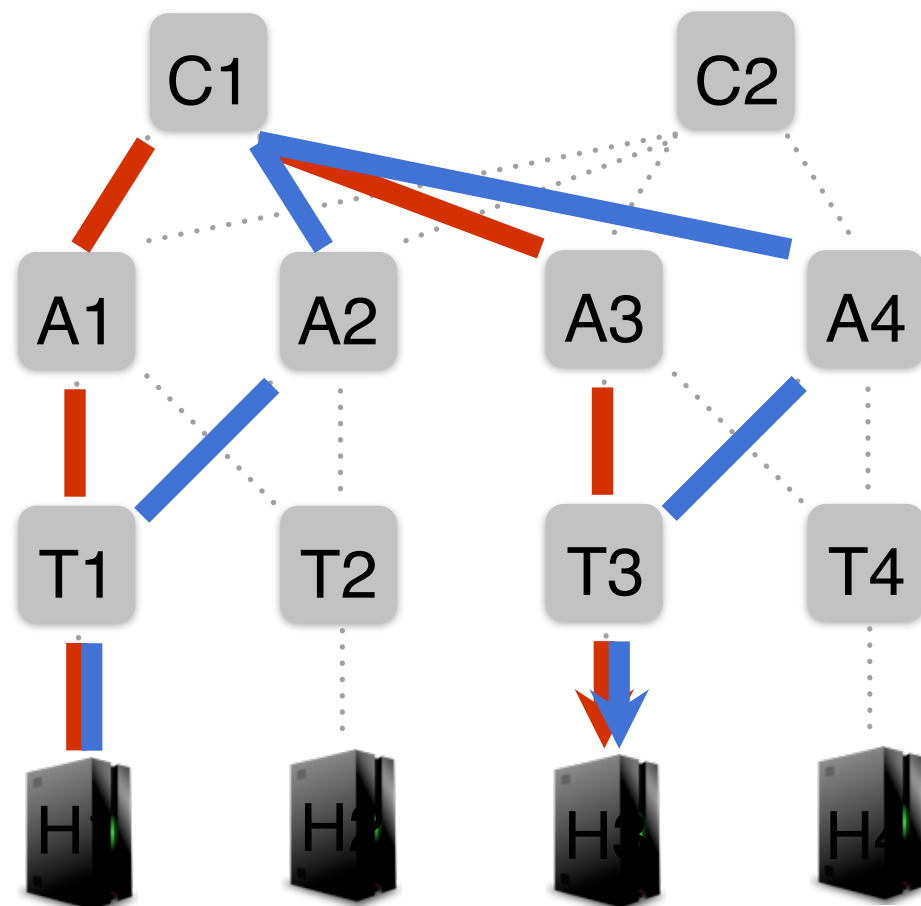
Order Update Example



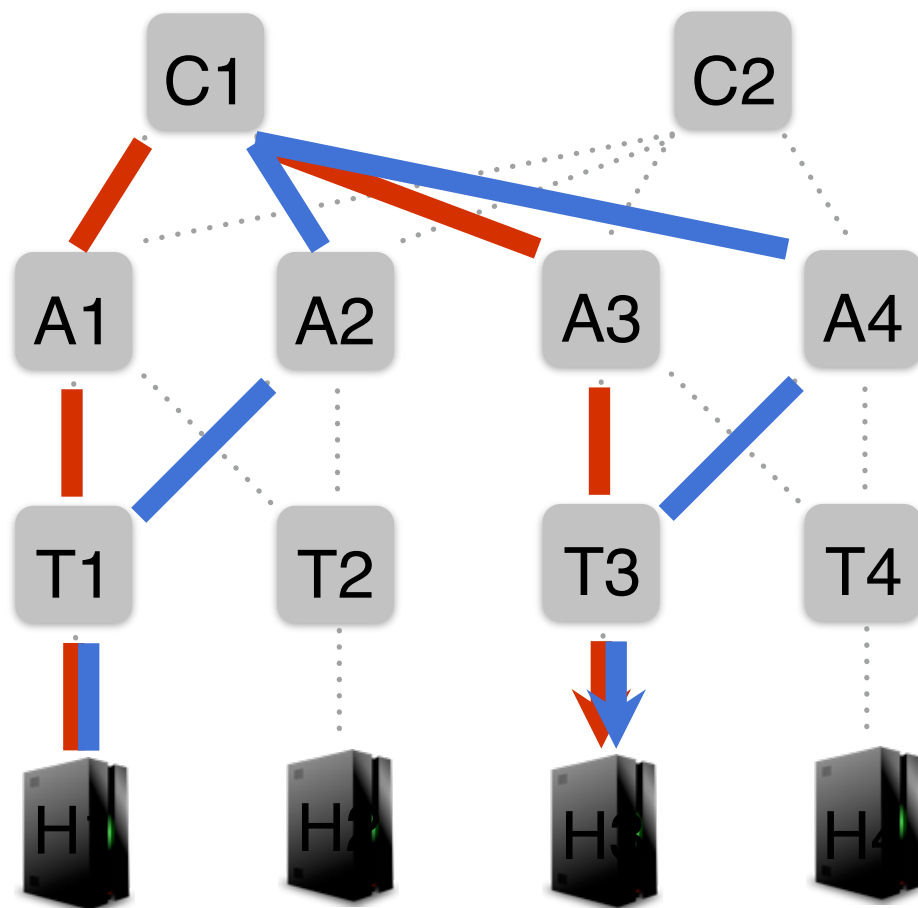
Order Update Example



Order Update Example

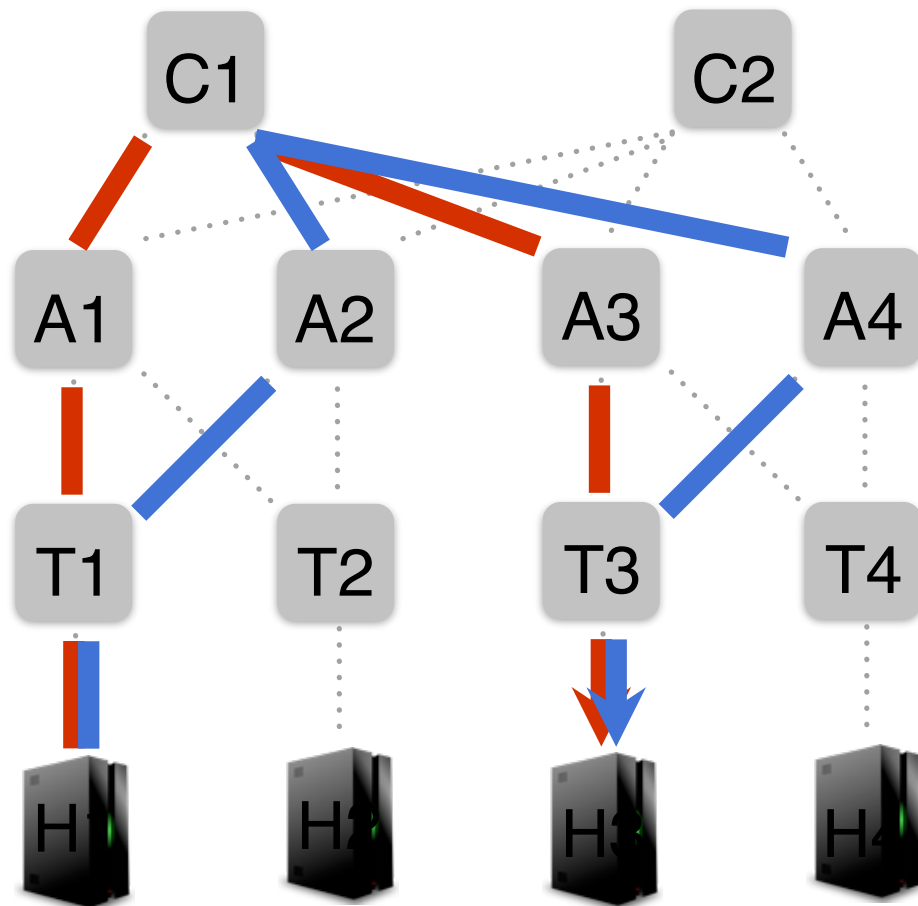


Order Update Example



Property: at all times, maintain H1-H3 connectivity and either traverse A2 or A3

Order Update Example

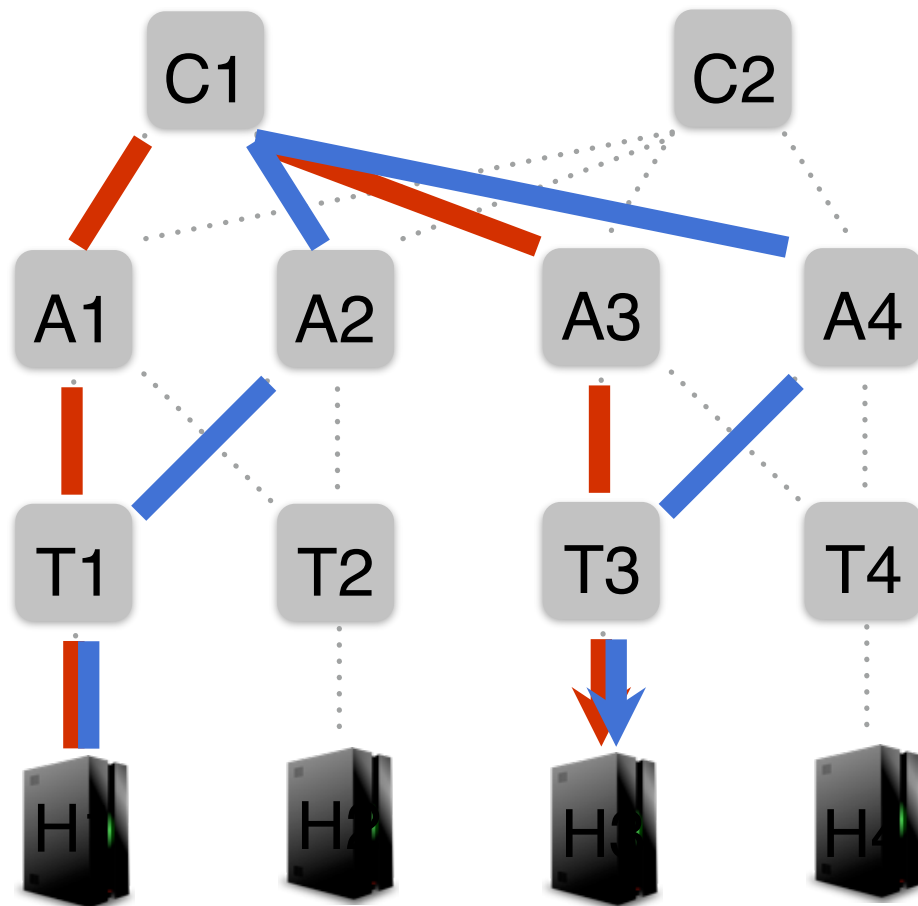


A2-A4-C1 (not good)

A2-A4-T1-C1 ?

Property: at all times, maintain H1-H3 connectivity and either traverse A2 or A3

Order Update Example



A2-A4-C1 (not good)

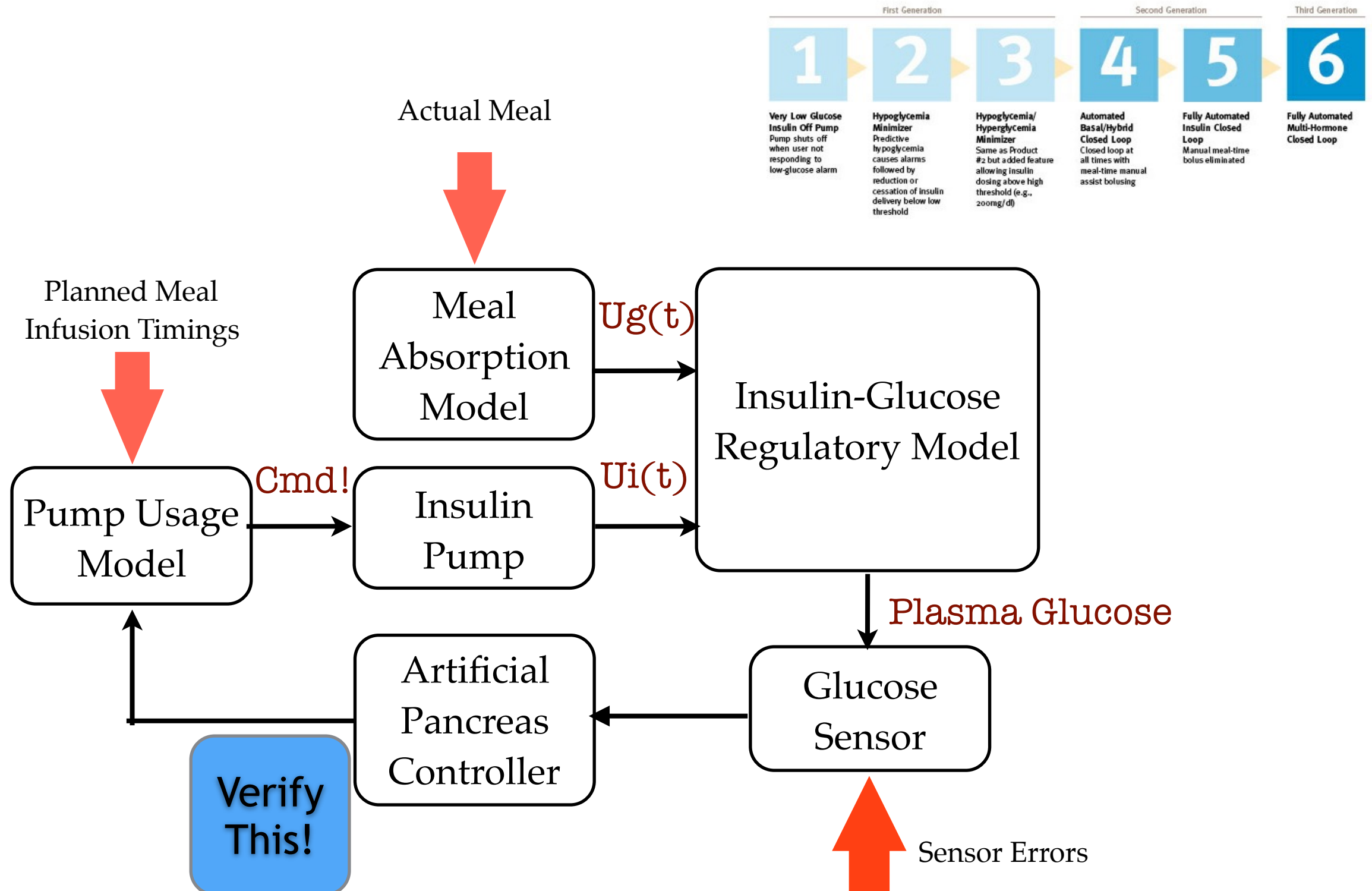
A2-A4-T1-C1 ?

A2-A4-T1-wait-C1

Property: at all times, maintain H1-H3 connectivity and either traverse A2 or A3

Artificial Pancreas Verification Project

Collaboration with UC Denver Medical School and UT El Paso.



PLV research at CU is *successful!*



PLDI 2015: Portland, OR

McClurg, Hojjat, Cerny, Foster. *Efficient Synthesis of Network Updates.*

PLDI 2014: Edinburgh, UK

Logozzo, Fahndrich, Lahiri, Blackshear. *Verification Modulo Versions: Towards Usable Verification.*

POPL 2014 (2): San Diego, CA

Coughlin, Chang. *Fissile Type Analysis: Modular Checking of Almost Everywhere Invariants.*

Jeannet, Schrammel, and Sankaranarayanan. *Abstract Acceleration of General Linear Loops.*

CAV 2014: Vienna, Austria

Cox, Chang, Sankaranarayanan. *QUICr: A Reusable Library for Parametric Abstraction of Sets and Numbers.*

ESOP 2015 (2): London, UK

Cox, Chang, Rival. *Desynchronized Multi-State Abstractions for Open Programs in Dynamic Languages.*

Cerny, Henzinger, Kovacs, Radhakrishna, Zwirchmayr. *Segment Abstraction for Worst-Case Execution Time Analysis.*

EMSOFT 2014 (2): New Delhi, India

Ravanbakhsh, Sankaranarayanan. *Infinite Horizon Safety Controller Synthesis through Disjunctive Polyhedral Abstract Interpretation.*

Zutshi, Sankaranarayanan, Deshmukh, Kapinski. *Multiple Shooting, CEGAR-based Falsification for Hybrid Systems.*

PLV research at CU is *successful!*



PLDI 2015: Portland, OR

McClurg, Hojjat, Cerny, Foster. *Efficient Synthesis of Network Updates.*

PLDI 2014: Edinburgh, UK

Logozzo, Fahndrich, Lahiri, Blackshear. *Verification Modulo Versions: Towards Usable Verification.*

POPL 2014 (2): San Diego, CA

Coughlin, Chang. *Fissile Type Analysis: Modular Checking of Almost Everywhere Invariants.*

Jeannet, Schrammel, and Sankaranarayanan. *Abstract Acceleration of General Linear Loops.*

CAV 2014: Vienna, Austria

Cox, Chang, Sankaranarayanan. *QUICK: A Domain-Specific Abstraction of Sets and Numbers.*

ESOP 2015 (2): London

Fun Destinations!

Cox, Chang, Rival. *Desynchronized Multi-State Abstractions for Open Programs in Dynamic Languages.*

Cerny, Henzinger, Kovacs, Radhakrishna, Zwirchmayr. *Segment Abstraction for Worst-Case Execution Time Analysis.*

EMSOFT 2014 (2): New Delhi, India

Ravanbakhsh, Sankaranarayanan. *Infinite Horizon Safety Controller Synthesis through Disjunctive Polyhedral Abstract Interpretation.*

Zutshi, Sankaranarayanan, Deshmukh, Kapinski. *Multiple Shooting, CEGAR-based Falsification for Hybrid Systems.*

Top Venues!

Research at CU is *successful!*



PLDI 2015: Portland, OR

McClurg, Hojjat, Cerny, Foster. *Efficient Synthesis of Network Updates.*

PLDI 2014: Edinburgh, UK

Logozzo, Fahndrich, Lahiri, Blackshear. *Verification Modulo Versions: Towards Usable Verification.*

POPL 2014 (2): San Diego, CA

Coughlin, Chang. *Fissile Type Analysis: Modular Checking of Almost Everywhere Invariants.*

Jeannet, Schrammel, and Sankaranarayanan. *Abstract Acceleration of General Linear Loops.*

CAV 2014: Vienna, Austria

Cox, Chang, Sankaranarayanan. *QUICK: A Domain-Specific Abstraction of Sets and Numbers.*

ESOP 2015 (2): London

Fun Destinations!

Cox, Chang, Rival. *Desynchronized Multi-State Abstractions for Open Programs in Dynamic Languages.*

Cerny, Henzinger, Kovacs, Radhakrishna, Zwirchmayr. *Segment Abstraction for Worst-Case Execution Time Analysis.*

EMSOFT 2014 (2): New Delhi, India

Ravanbakhsh, Sankaranarayanan. *Infinite Horizon Safety Controller Synthesis through Disjunctive Polyhedral Abstract Interpretation.*

Zutshi, Sankaranarayanan, Deshmukh, Kapinski. *Multiple Shooting, CEGAR-based Falsification for Hybrid Systems.*

Top Venues!

Research at CU is *successful!*



PLDI 2015: Portland, OR

McClurg, Hojjat, Cerny, Foster. *Efficient Synthesis of Network Updates.*

PLDI 2014: Edinburgh, UK

Logozzo, Fahndrich, Lahiri, Blackshear. *Verification Modulo Versions: Towards Usable Verification.*

POPL 2014 (2): San Diego, CA

Coughlin, Chang. *Fissile Type Analysis: Modular Checking of Almost Everywhere Invariants.*

Jeannet, Schrammel, and Sankaranarayanan. *Abstract Acceleration of General Linear Loops.*

CAV 2014: Vienna, Austria

Cox, Chang, Sankaranarayanan. *QUICK: A Framework for Parametric Abstraction of Sets and Numbers.*

ESOP 2015 (2): London

Fun Destinations!

Cox, Chang, Rival. *Desynchronized Multi-State Abstractions for Open Programs in Dynamic Languages.*

Cerny, Henzinger, Kovacs, Radhakrishna, Zwirchmayr. *Segment Abstraction for Worst-Case Execution Time Analysis.*

EMSOFT 2014 (2): New Delhi, India

Ravanbakhsh, Sankaranarayanan. *Infinite Horizon Safety Controller Synthesis through Disjunctive Polyhedral Abstract Interpretation.*

Zutshi, Sankaranarayanan, Deshmukh, Kapinski. *Multiple Shooting, CEGAR-based Falsification for Hybrid Systems.*

Top Venues!

Research at CU is *successful!*



PLDI 2015: Portland, OR

McClurg, Hojjat, Cerny, Foster. *Efficient Synthesis of Network Updates.*

PLDI 2014: Edinburgh, UK

Logozzo, Fahndrich, Lahiri, Blackshear. *Verification Modulo Versions: Towards Usable Verification.*

POPL 2014 (2): San Diego, CA

Coughlin, Chang. *Fissile Type Analysis: Modular Checking of Almost Everywhere Invariants.*

Jeannet, Schrammel, and Sankaranarayanan. *Abstract Acceleration of General Linear Loops.*

CAV 2014: Vienna, Austria

Cox, Chang, Sankaranarayanan. *QUICK: A Domain-Specific Abstraction of Sets and Numbers.*

ESOP 2015 (2): London

Fun Destinations!

Cox, Chang, Rival. *Desynchronized Multi-State Abstractions for Open Programs in Dynamic Languages.*

Cerny, Henzinger, Kovacs, Radhakrishna, Zwirchmayr. *Segment Abstraction for Worst-Case Execution Time Analysis.*

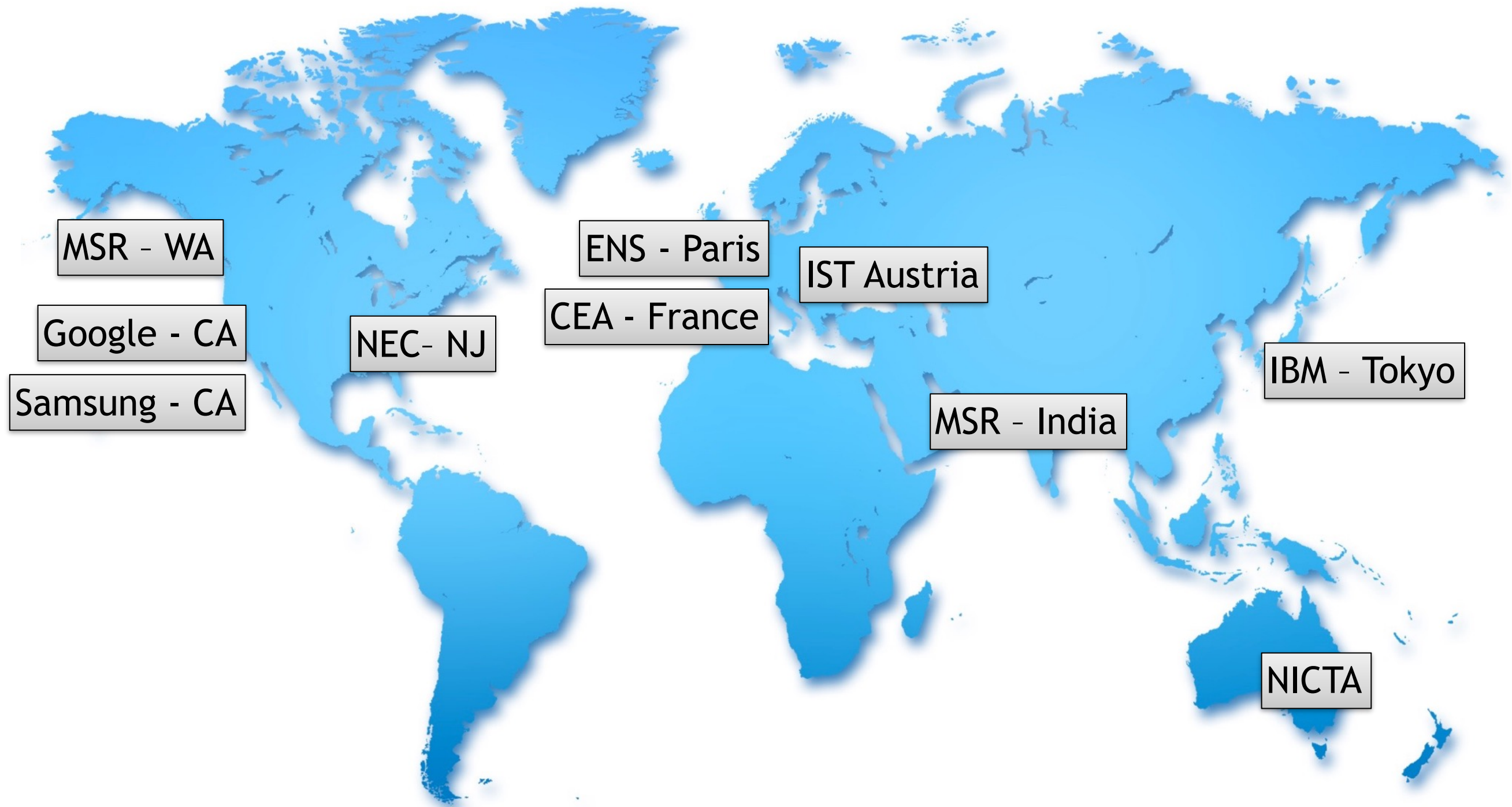
EMSOFT 2014 (2): New Delhi, India

Ravanbakhsh, Sankaranarayanan. *Infinite Horizon Safety Controller Synthesis through Disjunctive Polyhedral Abstract Interpretation.*

Zutshi, Sankaranarayanan, Deshmukh, Kapinski. *Multiple Shooting, CEGAR-based Falsification for Hybrid Systems.*

CUPLV Students!

PLV research at CU has *world-wide collaborations!*



PLV research at CU has *world-wide collaborations!*



PLV students have *interned* at ...

Microsoft®
Research

Redmond, Washington
Cambridge, UK
Bangalore, India

Google™



facebook

NEC

parc®
Palo Alto Research Center



 **TOYOTA**
TOYOTA TECHNICAL CENTER

CRAY®
THE SUPERCOMPUTER COMPANY

The PLV group has *fun* together!

The PLV group has *fun* together!

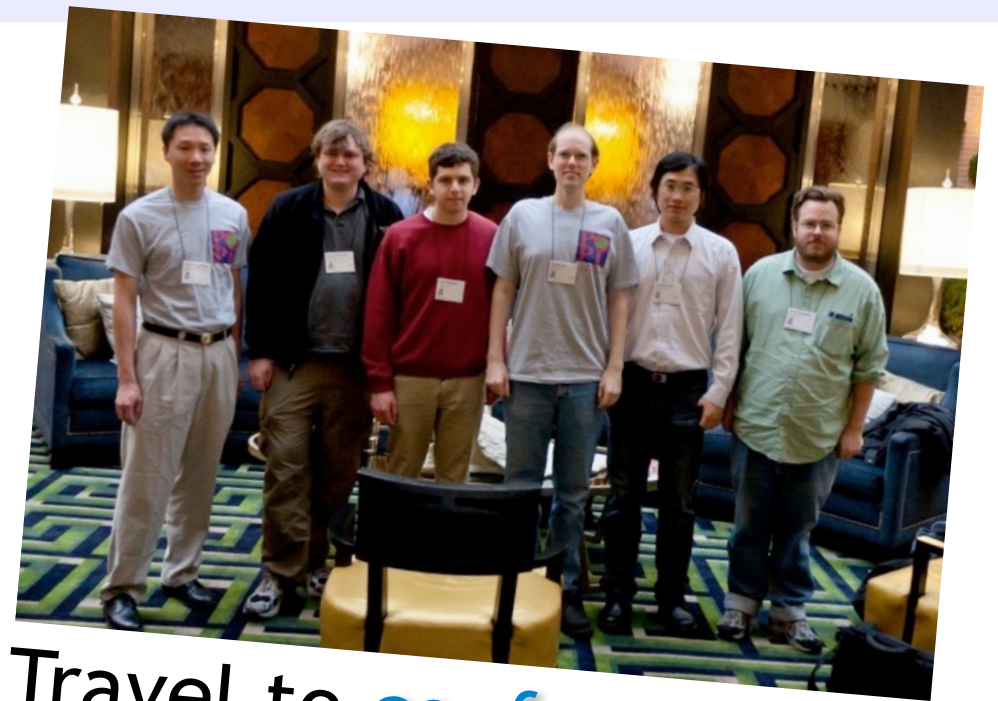


Group meetings at the
Boulder Tea House twice
a semester

The PLV group has *fun* together!



Group meetings at the **Boulder Tea House** twice a semester



Travel to **conferences** (POPL 2012)

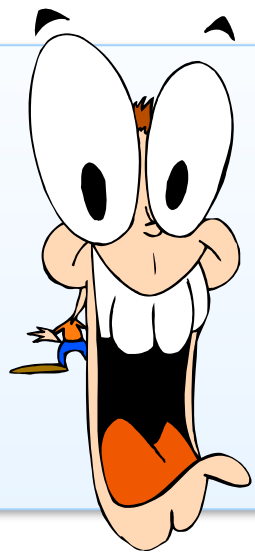
The PLV group has *fun* together!



Group meetings at the **Boulder Tea House** twice a semester



Travel to **conferences** (POPL 2012)



Our mentoring: Guide you to research that *excites* you!

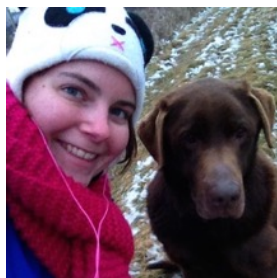
Our group



Shawn

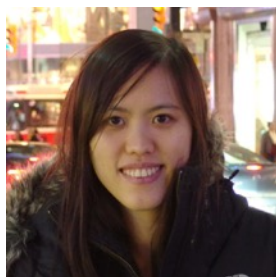


Jed

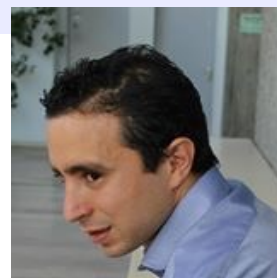


Alex

Post-Doc



Vris

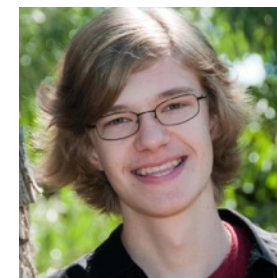


Amin

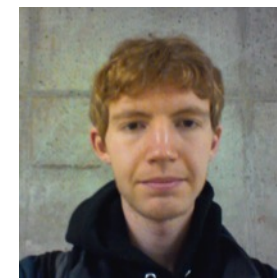
BS



Evan

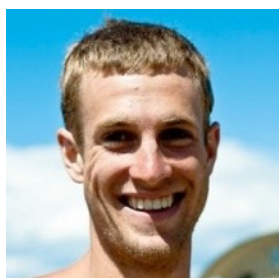


Kyle

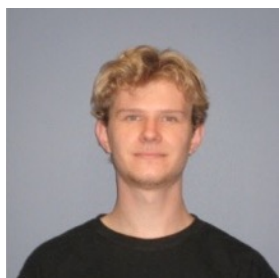


Max

PhD



Sam



Aleks



Hadi



Aditya



Fabio



Sriram

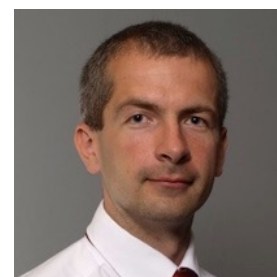


You?



You?

Faculty



Pavol



Evan



Dirk

Some of our other research projects

- False alarm triage analysis
 - Modular invariant checking
 - Analysis of dynamic languages
 - Mobile app malware detection
 - Incremental verification-validation
 - Analysis of medical devices
 - Health care process analysis
 - Cyber-physical systems verification
 - Program synthesis
 - Synthesis for software-defined networking
- **And soon projects created by you!**

Welcome

News

People

Papers

<http://pl.cs.colorado.edu/>

CUPLV

PROGRAMMING LANGUAGES AND VERIFICATION AT THE UNIVERSITY OF COLORADO BOULDER

Expressivity, Performance, Dependability, and Understanding of Computational Systems