Yoyak

An "Interactive" Abstract Interpreter

Motivation & Plan

Jaeho Shin

2010-07-09
ROPAS Show&Tell

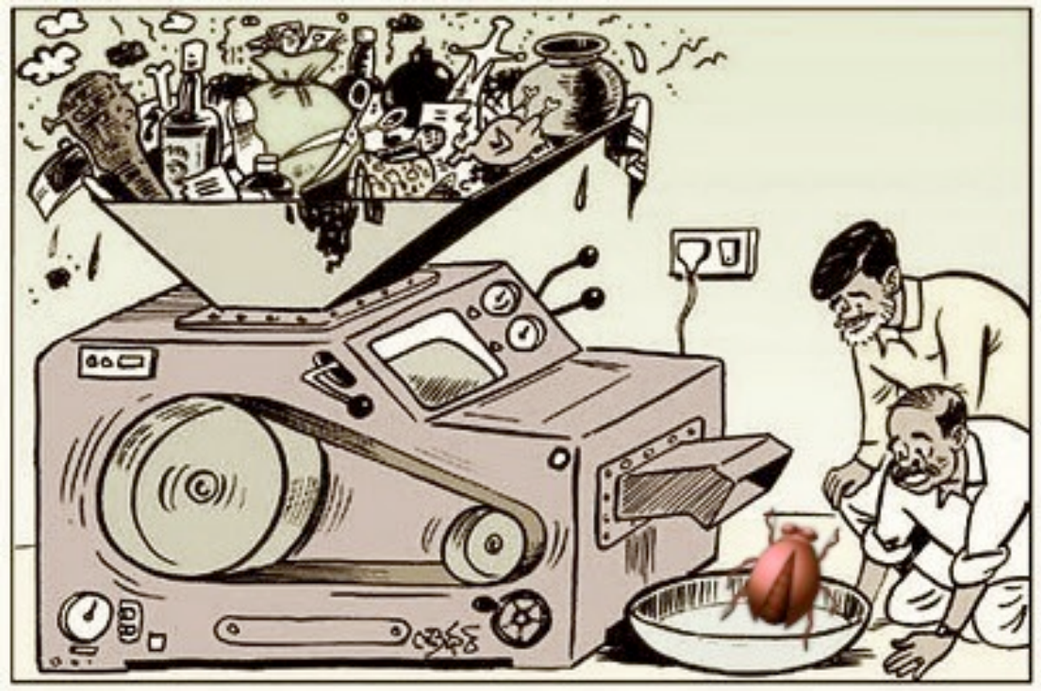
Contents

Motivation

for “Interactive”ness

Past experiences with our fully automatic analyzers

Bad User Experiences



Images borrowed from sriks6711.wordpress.com, www.cartoonbank.com.

What we have done was the

"automation for Analyzer"

What we wanted perhaps was an

“automation for Humans”

Automation for Humans

Images borrowed from internet20.org, last.fm.

Plan

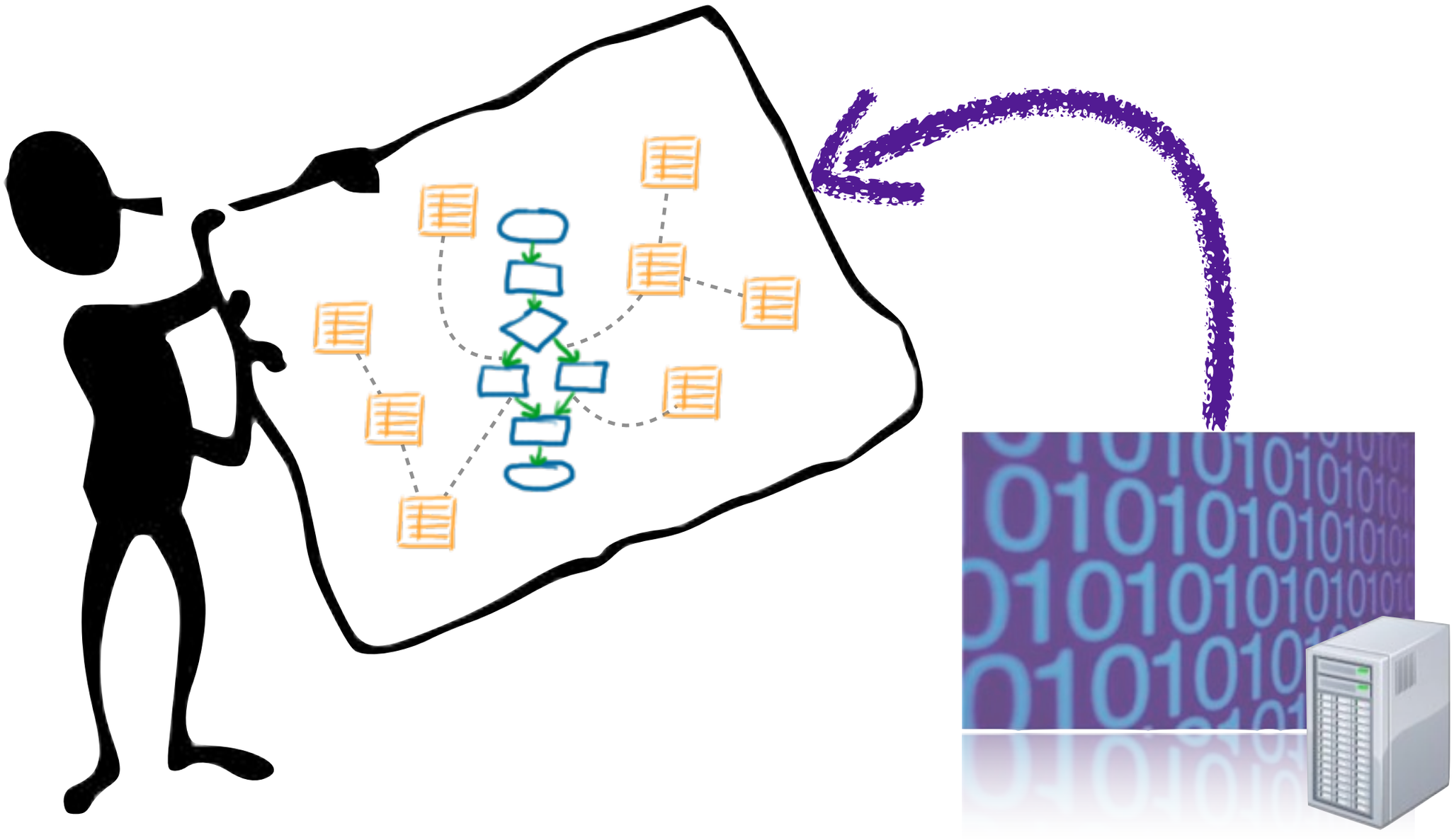
for “Interactive”ness

Plan

1. Let's make things visible to the user!
   - ✳ Analysis Browser

2. User would be happy to play with them!
   - ✳ Feedback System

Analysis Browser



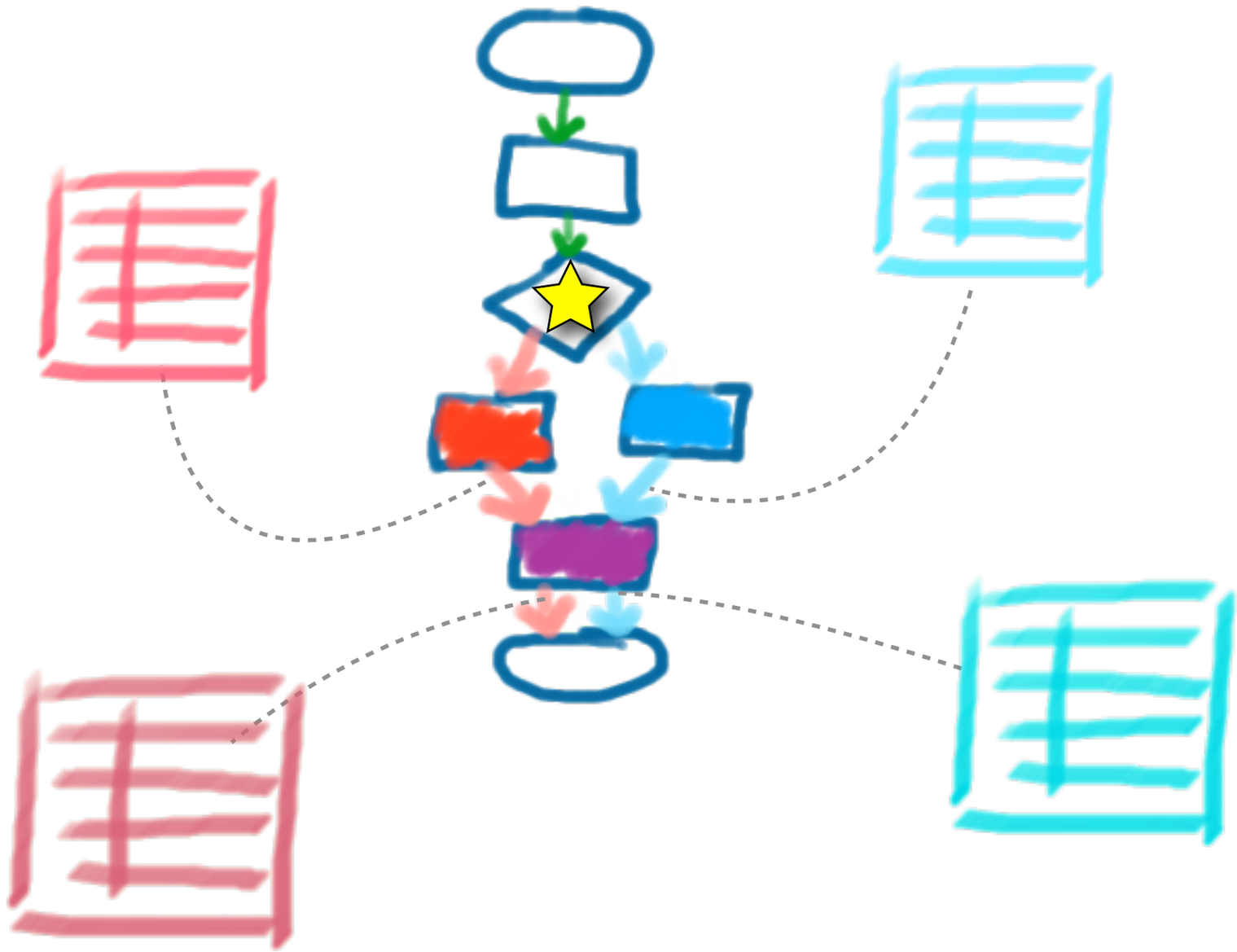
Feedback System

- Control Abstraction Directives
  - Branch Discrimination
  - Loop Unrolling
- Data Abstraction Directives
  - Assertions
  - Domain Selection

Branch Discrimination

Branch Discrimination

Branch Discrimination



Other Thoughts

- Leveraging Web Technologies for Interactive Features
  - HTML5 & Javascript for analysis browser and feedback system
- Considering a re-implementation of Airac5
  - Modular & Composite Design with simple independent parts, instead of the complicated monolithic one
  - Java & Scala for worklist algorithm computing fixed-points
  - C (APRON) for core operations on abstract states and values
- Rules of Transparency & Representation
  - Designing for visibility to make inspection and debugging easier.
  - Folding knowledge into data so program logic can be stupid and robust.

Schedule

Jul 2010	Phase 1	How control flow graphs of program (G form) and abstract values and states should be serialized are defined, probably in JSON format.
	Phase 2	Set of Java/Scala libraries are ready which will help us write/read G form and abstract states to/from serialized forms.
	Phase 3	A basic analysis browser lets us navigate through C code or G form and display corresponding abstract states.
Aug 2010	Phase 4	Basic implementation of operations on abstract values and states are ready in Java/Scala.
	Phase 5	Basic implementation of work-list algorithm that computes fixed-points is ready in Java/Scala.
	Phase 6	A front-end that converts C code into G form is ready and can initiate a full abstract interpretation.
Sep 2010	Phase 7	Annotations for the feedback system are defined and their semantics are implemented in the analyzer. The analysis browser now lets user to add/remove annotations while navigating through the program.
	Phase 8	Memory error checkers, e.g. array index range and memory leak checkers, are implemented to compare with Airac5.
...	Phase 9	Implementations of operations on abstract values and states are improved for better precision.
	Phase 10	Analysis browser shows control flow graph diagrams with nice layout. It also visualizes the progress of on-line analyses.
	Phase 11	Many parts are optimized for performance and scalability so that large programs can be analyzed, e.g. operations re-implemented in C/JNI, journaling implementation for abstract states, etc.
	...	...

References

- Jaeho Shin. *Master's Thesis: An Abstract Interpretation with the Interval Domain for C-like Programs*. School of Computer Science and Engineering, College of Engineering, Seoul National University. Aug 2006.

- Jaeho Shin. *Lessons of Airac5*. ROPAS Show & Tell. Mar 2006.

- Laurent Mauborgne and Xavier Rival. *Trace partitioning in abstract interpretation based static analyzers*. In M. Sagiv, editor, European Symposium on Programming (ESOP'05), Lecture Notes in Computer Science, pages 5-20. Springer-Verlag, 2005.

- Eric S. Raymond. *The Art of Unix Programming*. Addison-Wesley, October 2003.