Bachelor Thesis:
Interfacing TVLA and Sample

Raphael Fuchs

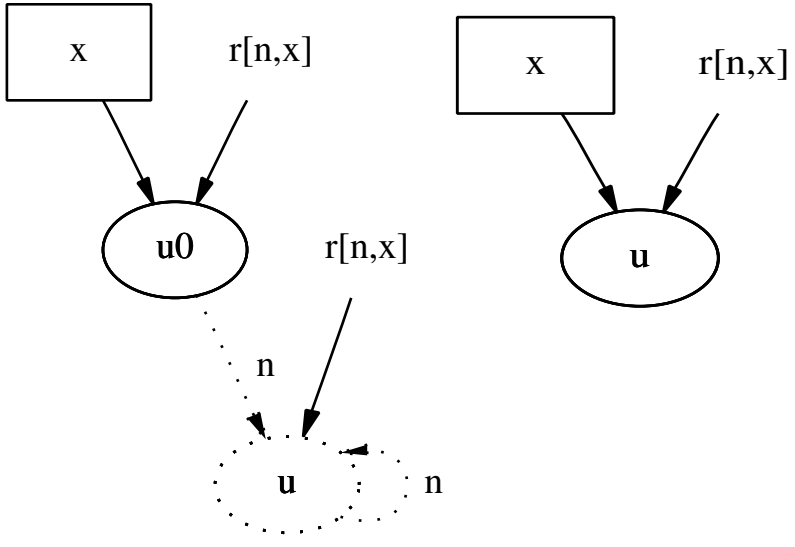
April, 2011

Outline

TVLA

Three-**V**alued-**L**ogic **A**nalyzer

- ten years of track record in performing heap analyses
- proved to be quite powerful
- Heap structures and semantics of statements encoded with logical formulas (Kleene Logic)
- approximates heap safely as a bounded set of structures
- performs summarization and materialization of heap nodes, guided by user-specified instrumentation predicates



Sample

- Generic static analyzer
- Addresses multiple (object-oriented) languages
- Tracks different abstract domains
  - Numerical information: Signs, intervals, octagons, polyhedra,...
  - Other Analyses: access permissions, type information,...
  - Heap domains: Heap identifiers pointed to by program variables.

Sample

- Generic static analyzer
- Addresses multiple (object-oriented) languages
- Tracks different abstract domains
  - Numerical information: Signs, intervals, octagons, polyhedra,...
  - Other Analyses: access permissions, type information,...
  - Heap domains: Heap identifiers pointed to by program variables.

Current Heap Domains

- Top domain: All references approximated by one heap identifier.
- Program-point bounded: For every program-point, keep at most one heap identifier.

Goal

Sample needs a better heap domain!

This involves coming up with

- a representation of the heap (environment, heap identifiers)
- implementations of operations such as object creation, assignment, field access.

Goal

Sample needs a better heap domain!

This involves coming up with

- a representation of the heap (environment, heap identifiers)
- implementations of operations such as object creation, assignment, field access.

Approach: For every statement that modifies the heap

- encode the current heap as TVS (three-valued structures)
- let TVLA execute an action
- parse TVLA's output
- update representation in Sample

Challenges

- TVLA model and Sample model are quite different.
  - TVLA: unary/binary predicates
  - Sample: we simply have heap identifiers and operations createObject, assign, fieldAccess etc.

Challenges

- TVLA model and Sample model are quite different.
  - TVLA: unary/binary predicates
  - Sample: we simply have heap identifiers and operations createObject, assign, fieldAccess etc.
- Describe the semantics of Sample statements with TVLA update formulae

Challenges

- TVLA model and Sample model are quite different.
  - TVLA: unary/binary predicates
  - Sample: we simply have heap identifiers and operations createObject, assign, fieldAccess etc.
- Describe the semantics of Sample statements with TVLA update formulae
- Matching of input and output of TVLA:
  - We somehow have to give names to nodes
  - Determine which heap nodes were summarized or materialized.

Challenges

- TVLA model and Sample model are quite different.
  - TVLA: unary/binary predicates
  - Sample: we simply have heap identifiers and operations createObject, assign, fieldAccess etc.
- Describe the semantics of Sample statements with TVLA update formulae
- Matching of input and output of TVLA:
  - We somehow have to give names to nodes
  - Determine which heap nodes were summarized or materialized.
- Optimize the way TVLA is invoked (efficiency)

Possible Extensions

Supporting and Choosing Instrumentation Predicates

- Instrumentation predicates guide abstraction of the heap
- right choice is crucial to obtain precise results
- evaluation necessary

Possible Extensions

Supporting and Choosing Instrumentation Predicates

- Instrumentation predicates guide abstraction of the heap
- right choice is crucial to obtain precise results
- evaluation necessary

Benchmarking

- Apply analysis to a wide set of benchmarks

Possible Extensions

Supporting and Choosing Instrumentation Predicates

- Instrumentation predicates guide abstraction of the heap
- right choice is crucial to obtain precise results
- evaluation necessary

Benchmarking

- Apply analysis to a wide set of benchmarks

Predicates for Data Structures

- Develop specific predicates for data structures like lists and trees.
- May be useful to show that e.g “list-ness” is preserved by destructive updates.

Questions?