The State Delta Verification System (SDVS)

Steve Crocker, Shinkuro, Inc.

steve@shinkuro.com

Presented at VTSA16, Liège, 2 September 2016

A Long Time Ago

Time period: 1975 – 1990. PhD thesis 1977.

Prove functional properties of real programs

"Real" = machine code in running systems

Designed new logic and verification system

Intended for assembly language programs. Mainly used for microcode verification of instruction sets.

That's where the $$ were

Found some subtle bugs

Summary of Principal Ideas

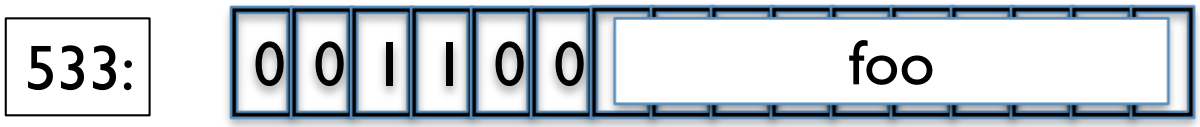
- A temporal logic syllogism
  - $A \rightsquigarrow B$ and $B \rightsquigarrow C$ implies $A \rightsquigarrow C$
- Characterize the action of each machine language instruction.
  - Particularly what changes
- Maintain a large set of statements about the state
- Update the state based on changes
  - Anything not touched isn't changed
  - Need a fairly rich model of memory

Example: IRS foo

- IRS is "increment memory and skip if zero."
  - Used primarily for loop control on small, old machines. (Honeywell 316, 516, etc.)
- `Loop: IRS foo`

Example: IRS foo

- IRS is "increment memory and skip if zero."
- `Loop: IRS foo`



Loop = 533

Example: IRS foo

- IRS is "increment memory and skip if zero."
  - Used primarily for loop control on small, old machines. (Honeywell 316, 516, etc.)
- `Loop: IRS foo`

- Approximately:
  - .pc = loop ↝
    - #foo = .foo + 1 and
    - (if #foo != 0, #pc=.pc +1 else #pc=.pc+2)
- BUT…

BUT… Say what changed

- Almost right:
  - .pc = loop ⇝ {pc, foo}
    - #foo = .foo + 1 and
    - (if #foo != 0, then #pc=.pc +1 else #pc=.pc+2)

- The places – a slight generalization of locations – pc and foo have changed. Places that are known to be disjoint have not changed.

Places

- Memory locations and registers are places.
- Changing the contents of one of these does not change another.
- More complex places can be created, e.g. arrays and linked lists.

Motivation for the Places model

- In general, storage is allocated into disjoint pieces.
- However when there are dynamic structures, e.g. lists, stack, etc., it's necessary to be able to introduce a decompositon of a portion of the space without being specific about how it relates to the underlying structure.

Places as "support" for formulas

- A formula like .x > .y depends on the contents of places x and y. If the formula is true at one point in time, it will remain true as long as neither x nor y is changed.

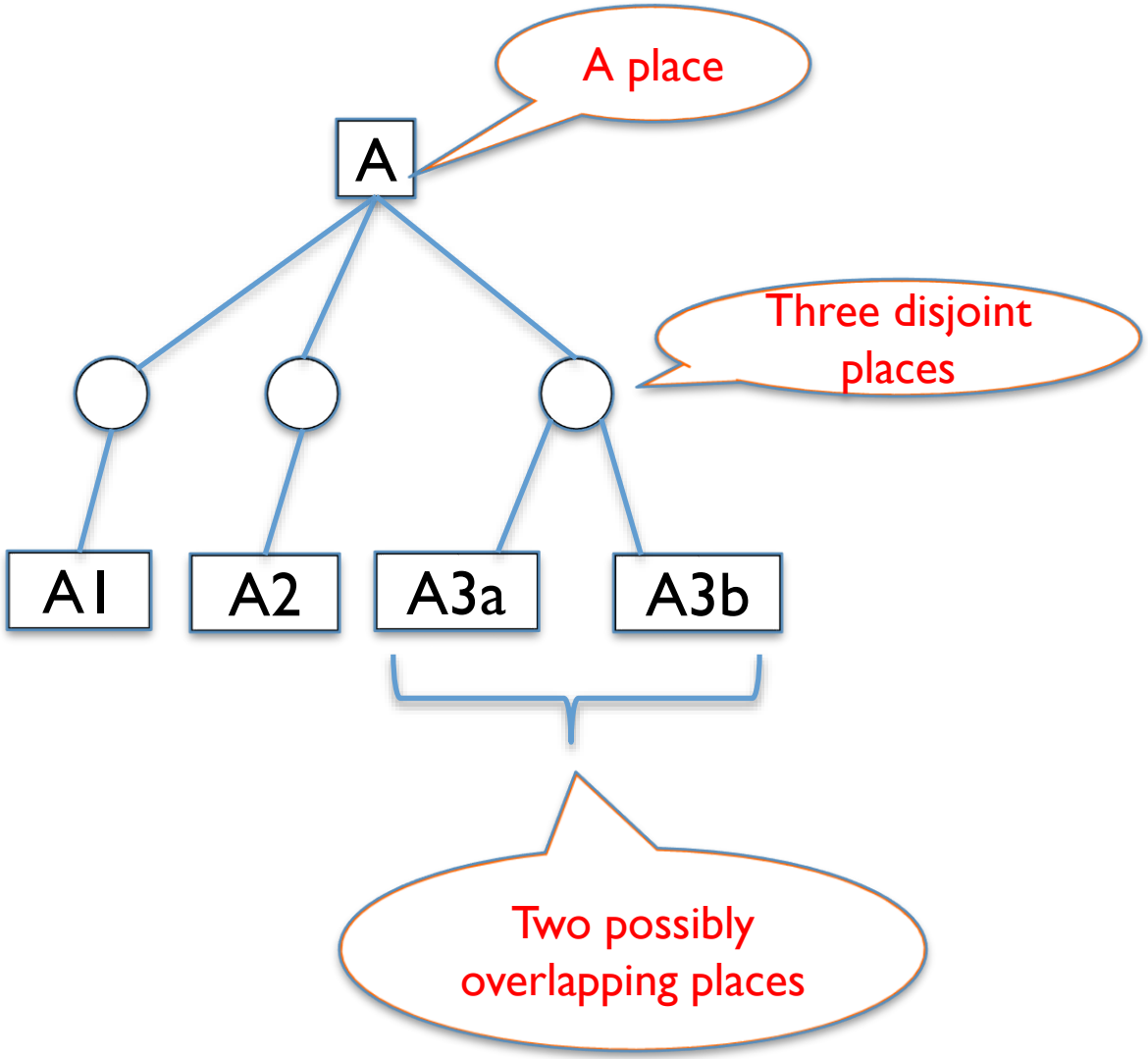
- {x,y} is the support for .x>.y

Relationships among places

- *(x, y, z) means x, y and z all disjoint from each other. Changing the contents of x will not affect the contents of y, etc.

- x $\triangleleft$ y means x is a subplace of y. If y is not affected by a set of changes, x is also not affected.

A Conservative Model of Places



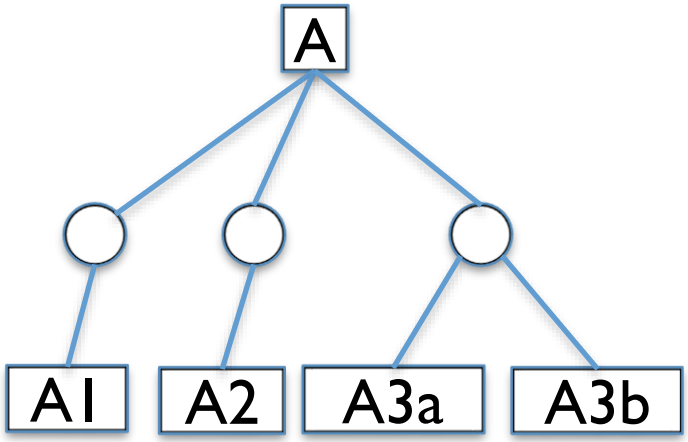
Changes…

- If A1 changes, so does A.
  - A2 and A3 do not change.
- If A3a changes, A changes and A3b might change too.
  - A1 and A2 do not change.

A Solver for Support

This graph is developed during the proof process with assertions and is searched during application of a state delta to determine which formulas to remove from the database.

When changes are made within a proof context, they are removed when the proof is closed.



State Deltas

- A State Delta is a temporal logic statement of the form: $_{E}[P \rightsquigarrow _{C} Q]$ where C and E are lists of places, P is a formula over the contents of places at the present time and Q is a formula over the contents of places both at both the present time and a future time.

State Deltas – Prose definition

- If the system is in a state where P is true, it will reach a state in which Q is true, and it will do so modifying no more than places not known to be disjoint from the list C.

- P and Q include the control as well as the usual values of variables.

What is the role of "E"?

- State deltas are part the overall state of the system. They are formulas about the state and can be treated the same as any other formula about the state of the system.

- E is the support for the state delta itself. Hence, if a state delta is part of the state but another state delta is applied which modifies any place not known to be disjoint from E, the state delta is removed from the state of the system.

Reasoning with State Deltas

- Reasoning with state deltas is patterned along the lines of symbolic execution.
- At any given time, there is:
  - A set of places,
  - A set of formulas over the contents of the places,
  - A set of relations among the places, and
  - A set of state deltas that characterize the behavior of the system.

Applying a state delta

- The primary inference rule is application of a state delta to the current state.
- A state delta that is contained within the current state is applicable iff its precondition is true in the current state.
- The result of applying a state delta is a new state that is the same as the old state except:
  - All formulas whose support is not known to be disjoint from the list of places that may have been changed by application of the state delta are removed, and
  - The post-condition formula is instantiated and added to the state.

Inference Rule: Sequencing*

$$\frac{{}_{E1}[P1 \sim_{C1} Q1],\ {}_{E2}[P2 \sim_{C2} Q2],\ Q1 \Rightarrow P2}{{}_{E1 \cup E2}[P1 \sim_{C1 \cup C2} Q2]}$$

* Two caveats: E2 must be known to be disjoint from C1 and there may be other formulas hidden from view that also imply P2.

Inference Rule: Disjunction

$$\frac{{}_{E1}[P1 \sim_{C1} Q],\ {}_{E2}[P2 \sim_{C2} Q]}{{}_{E1 \cup E2}[P1 \vee P2 \sim_{C1 \cup C2} Q]}$$

A Small Example

BEFORE

- .pc = 972, .x=7, .y = 2, .x > .y, F(.p,.q, …)
- *(pc,e,x,w,p,q,), y $\triangleleft$ w, z $\triangleleft$ w
- $_{\{e\}}[.pc = 972 \rightsquigarrow_{\{x,z,pc\}} \#pc=.pc+1, \#x = .x+1]$

AFTER

- .pc = 973, .x=8, ~~.y = 2, .x > .y~~, F(.p,.q, …)
- *(pc,e,x,w,p,q,), y $\triangleleft$ w, z $\triangleleft$ w
- $_{\{e\}}[.pc = 972 \rightsquigarrow_{\{x,z,pc\}} \#pc=.pc+1, \#x = .x+1]$

The State Delta Verification System

- Symbolic execution oriented verification
- Initialize with large set of state deltas and initial conditions.
- New state deltas are proven by
  - Application of state deltas
  - Case analysis
  - Induction
- SAT/SMT was added using Oppen-Nelson system.
  - New SMT solvers for arrays and other theories were added.

Questions?