Parametric Shape Analysis via 3-Valued Logic

Mooly Sagiv, Thomas Reps,
Reinhard Wilhelm

Motivation

- Many shape analysis algorithms developed
  - Different abstractions
  - Hard to compare
- Parametric Framework
  - yacc for shape analysis?

Overview

- Use logic structures to represent stores
- By choosing different predicates, the framework is instantiated into different shape analysis algorithms.
- Previous approach:
  - Define abstraction, give transfer function, prove, implement
- With the framework:
  - Choose predicate, define update formula for instrumentation predicates, prove correctness of the formulae
  - The rest is automatically done by the system

Representation

- Logical Structures:
  - S=<U, ι>
    - U: individuals
    - ι: maps $p(u_1, \ldots u_k)$ to 0, 1 or 1/2
- Predicates:
  - Constituents of shape invariants that can be used to characterize a data structure
  - Core Predicates:
    - Tracking Pointer Variables and Pointer-valued fields
    - Common to all the shape analysis
    - Eg: x(v), n(v1, v2), sm(v)

Representation

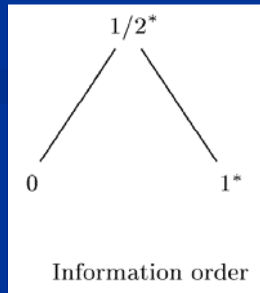
- Predicates
  - Instrumentation predicates:
    - Properties derived from core semantics, not explicitly part of the semantics of pointers in a language,
    - Different algorithms use different sets of instrumentation
    - Eg: is(v) (sharing), $r_x(v)$ (reachability)
    - Defining formulae:

$$\varphi_{is}(v) \stackrel{\text{def}}{=} \exists v_1, v_2 : n(v_1, v) \wedge n(v_2, v) \wedge v_1 \neq v_2$$

$$\varphi_{r_x}(v) \stackrel{\text{def}}{=} x(v) \vee \exists v_1 : x(v_1) \wedge n^+(v_1, v)$$

Representation

- Property-Extraction Principle
  - Concrete Store: 2-Valued Logic
    - Questions about properties of stores can be answered by evaluating formulae: 1=>hold, 0=>doesn't hold
  - Abstract store: 3-Valued Logic
    - A formulae can evaluate to 1, 0, or ½.
    - 1=>hold
    - 0=>doesn't hold
    - ½ => don't know



∧	0	1	1/2
0	0	0	0
1	0	1	1/2
1/2	0	1/2	1/2

∨	0	1	1/2
0	0	1	1/2
1	1	1	1
1/2	1/2	1	1/2

¬	
0	1
1	0
1/2	1/2

Representation

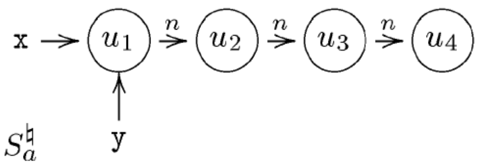
- Examples

unary preds.

indiv.	x	y	t	e
u_1	1	1	0	0
u_2	0	0	0	0
u_3	0	0	0	0
u_4	0	0	0	0

binary preds.

n	u_1	u_2	u_3	u_4
u_1	0	1	0	0
u_2	0	0	1	0
u_3	0	0	0	1
u_4	0	0	0	0



$S_a^\natural$

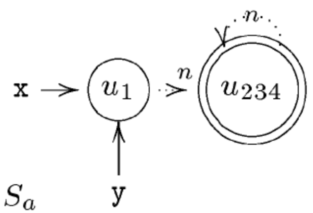
abstracts to →

unary preds.

indiv.	x	y	t	e	sm
u_1	1	1	0	0	0
u_{234}	0	0	0	0	1/2

binary preds.

n	u_1	u_{234}
u_1	0	1/2
u_{234}	0	1/2



S_a

Bounded Structures

- Bounded Structures:
  - A logical structure where no two individuals evaluates to the same value for all predicates
- Upper bound on the size of bounded structures:

$$|U^S| \leq 3^{|\mathcal{A}|}$$

- Canonical Abstraction:

$$t\_embed_c(u) = u_{\{p \in \mathcal{A} | \iota^S(p)(u)=1\},\{p \in \mathcal{A} | \iota^S(p)(u)=0\}}$$

Embedding Theorem

- Embedding:
  - A way to relate 2-valued and 3-valued structures
  - S can be embedded in S':
    - Surjective function f: $U^S \rightarrow U^{S'}$
    - $\iota^S(p)(u_1, \ldots, u_k) \sqsubseteq \iota^{S'}(p)(f(u_1), \ldots, f(u_k))$
- Embedding Theorem:
  - If S can be embedded in S', every piece of information extracted from S' via a formula is a conservative approximation of the information extracted from S.

Predicate-update formula

- Expressing semantics using logic
  - Predicate-update formulae φ_p^{st} : Define the new value of p for every statement st
  - Transfer function:

$$[\![st]\!](S) = \left\langle \begin{array}{l} U^S, \\ \lambda p.\lambda u_1, \ldots, u_k.[\![\varphi_p^{st}]\!]_3^S([v_1 \mapsto u_1, \ldots, v_k \mapsto u_k]) \end{array} \right\rangle$$

Predicate-update formula

- Core Predicates: the predicate-update formulae is exactly the same for 3-valued logic and 2-valued logic
- Instrumentation Predicate:
  - Trivial update formula: usually unsatisfactory
  - User supplied formula: need to prove it maintains correct instrumentation.

Predicate-update formula

- Core Predicates:

st	φ_p^{st}
`x = NULL`	$\varphi_x^{st}(v) \stackrel{\text{def}}{=} \mathbf{0}$
`x = t`	$\varphi_x^{st}(v) \stackrel{\text{def}}{=} t(v)$
`x = t->sel`	$\varphi_x^{st}(v) \stackrel{\text{def}}{=} \exists v_1 : t(v_1) \wedge sel(v_1, v)$
`x->sel = NULL`	$\varphi_{sel}^{st}(v_1, v_2) \stackrel{\text{def}}{=} sel(v_1, v_2) \wedge \neg x(v_1)$
`x->sel = t` (assuming that `x->sel == NULL`)	$\varphi_{sel}^{st}(v_1, v_2) \stackrel{\text{def}}{=} sel(v_1, v_2) \vee (x(v_1) \wedge t(v_2))$
`x = malloc()`	$\varphi_x^{st}(v) \stackrel{\text{def}}{=} isNew(v)$ $\varphi_z^{st}(v) \stackrel{\text{def}}{=} z(v) \wedge \neg isNew(v)$, for each `z` $\in (PVar - \{$`x`$\})$ $\varphi_{sel}^{st}(v_1, v_2) \stackrel{\text{def}}{=} sel(v_1, v_2) \wedge \neg isNew(v_1) \wedge \neg isNew(v_2)$ for each `sel` $\in PSel$

Predicate-update formula

- Instrumentation predicate

st	φ_{is}^{st}
`x->n = NULL`	$\varphi_{is}^{st}(v) \stackrel{\text{def}}{=} \begin{cases} is(v) \wedge \varphi_{is}[n \mapsto \varphi_n^{st}] & \text{if } \exists v' : x(v') \wedge n(v', v) \\ is(v) & \text{otherwise} \end{cases}$
`x->n = t` (assuming that `x->n == NULL`)	$\varphi_{is}^{st}(v) \stackrel{\text{def}}{=} \begin{cases} is(v) \vee \varphi_{is}[n \mapsto \varphi_n^{st}] & \text{if } \exists v_1 : t(v) \wedge n(v_1, v) \\ is(v) & \text{otherwise} \end{cases}$
`x = malloc()`	$\varphi_{is}^{st}(v) \stackrel{\text{def}}{=} is(v) \wedge \neg new(v)$

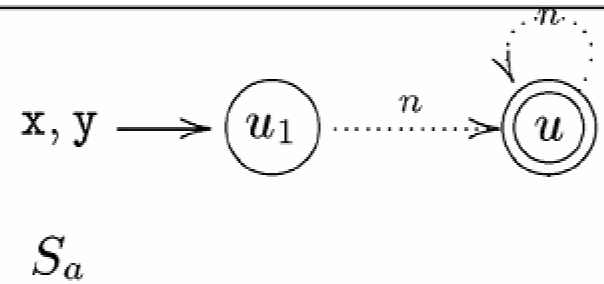
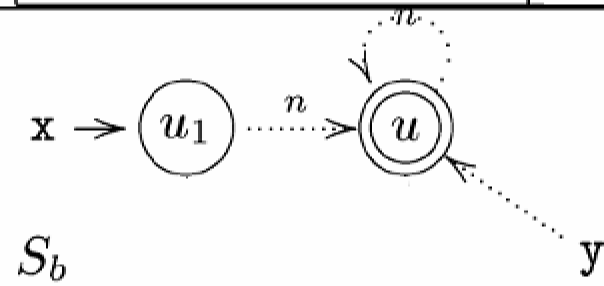
The Shape Analysis Algorithm

$$StructSet[v] = \begin{cases} \bigcup_{w \rightarrow v \in G} \{t\_embed_c[\![st(w)]\!](S) \mid S \in StructSet[w]\} & \text{if } v \neq start \\ \{\langle \emptyset, \lambda p.\lambda u_1, \ldots, u_k.1/2 \rangle\} & \text{if } v = start \end{cases}$$

- When analyzing a single procedure, allow an arbitrary set of 3-valued structures to hold at the entry of the procedure

The Shape Analysis Algorithm

- Example:

input structure	
update formulae	$\varphi_y^{st_0}(v)$ $\exists v_1 : y(v_1) \wedge n(v_1, v)$
output structure	

A More Precise Abstract Semantics

- Overview
  - Focus
  - Apply transfer function
  - coerce

A More Precise Abstract Semantics

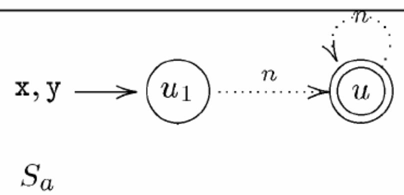
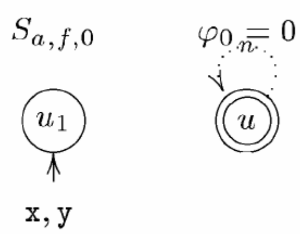
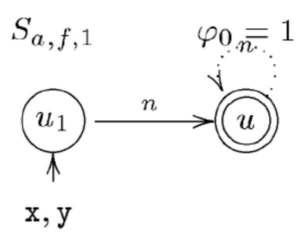
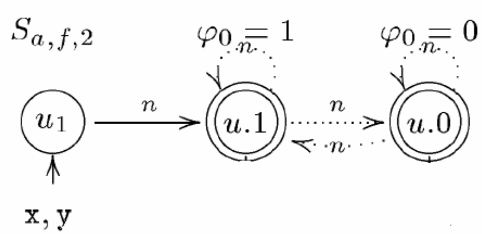
- Focus: forces a given formula to a definite value

$$maximal(XS) \stackrel{\text{def}}{=} XS - \{X \in XS \mid \exists X' \in XS : X \sqsubseteq X' \ \mathit{and}\ X' \not\sqsubseteq X\}$$

$$focus_{\varphi}(S) = maximal\left(\left\{ S' \;\middle|\; \begin{array}{l} S' \in \textit{3-STRUCT}[\mathcal{P}] \\ S' \sqsubseteq S \\ \mathit{for\ all}\ Z : [\![\varphi]\!]_3^{S'}(Z) \neq 1/2 \end{array} \right\}\right)$$

A More Precise Abstract Semantics

- Focus Example:

input structure	
focus formulae	$\{\varphi_0(v)\}$, where $\varphi_0(v) \stackrel{\text{def}}{=} \exists v_1 : y(v_1) \wedge n(v_1, v)$
focused structures	  

A More Precise Abstract Semantics

- Coerce

A *compatibility constraint* is a term of the form $\varphi_1 \rhd \varphi_2$, where φ_1 is an arbitrary 3-valued formula, and φ_2 is either an atomic formula or the negation of an atomic formula over distinct logical variables.

  - Sharpen a structure according to Compatibility Constraints
  - Compatibility Constraints from Instrumentation Predicates
  - Compatibility Constraints from Hygience Conditions

A More Precise Abstract Semantics

- An algorithm to generate compatibility constraints
  - Definition Formula:

$$\forall v : (\exists v_1, v_2 : n(v_1, v) \wedge n(v_2, v) \wedge v_1 \neq v_2) \Rightarrow is(v)$$

  - Extended Horn Clause:

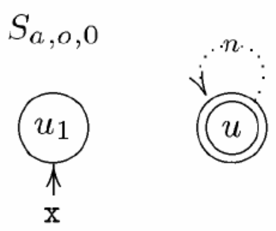
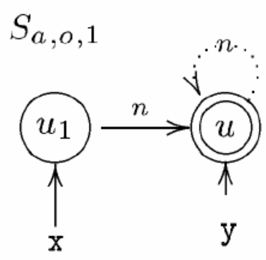
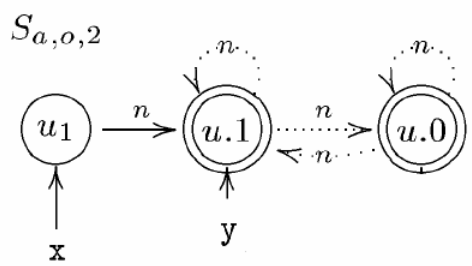
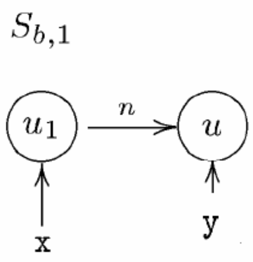
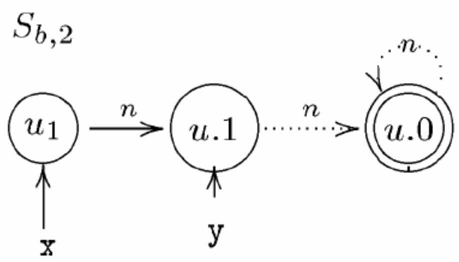
$$\forall v, v_1, v_2 : \neg n(v_1, v) \vee \neg n(v_2, v) \vee v_1 = v_2 \vee is(v)$$

  - Compatibility constraints:

$$(\exists v_1, v_2 : n(v_1, v) \wedge n(v_2, v) \wedge v_1 \neq v_2) \triangleright is(v)$$
$$(\exists v_1 : n(v_1, v) \wedge v_1 \neq v_2 \wedge \neg is(v)) \triangleright \neg n(v_2, v)$$
$$(\exists v_2 : n(v_2, v) \wedge v_1 \neq v_2 \wedge \neg is(v)) \triangleright \neg n(v_1, v)$$
$$(\exists v : n(v_1, v) \wedge n(v_2, v) \wedge \neg is(v)) \triangleright v_1 = v_2.$$

A More Precise Abstract Semantics

- Coerce Example:

update formulae		$\varphi_y^{st_0}(v)$: $\exists v_1 : y(v_1) \wedge n(v_1, v)$	$\varphi_{r_{n,y}}^{st_0}(v)$: $r_{y,n}(v) \wedge (c_n(v) \vee \neg y(v))$
output structures			
coerced structures			

$$(\exists v_1 : n(v_1, v) \wedge v_1 \neq v_2 \wedge \neg is(v)) \triangleright \neg n(v_2, v)$$

Related work

- K-limiting
  - Use instrumentation predicates "reachable-from-x-via-access-path-α", for $|\alpha| <= k$
- Storage Shape Graphs [CWZ'90]
  - Use core predicates that record the allocation sites of heap cells
- Doubly-linked list
  - Use Instrument Predicate $c_{f.b}(v)$ and $c_{b.f}(v)$

Related Work

- Biased versus unbiased static program analysis
  - Conventional analysis has one-sided bias:
  - May Analysis:
    - false => false
    - true => may be true/ may be false
  - Must Analysis:
    - true => true
    - false => may be true/ may be false
  - 3-Valued Logic:
    - unbiased

Summary

- A parametric framework
- Easy to experiment with new algorithms
- For core predicates, abstract semantics falls out from the concrete semantics
- No need for a proof for a particular instantiation

Limitations

- Size potentially exponential
- Efficiency
- Usually need to provide predicate-update formulae for instrumentation predicates and to prove that these formulae maintains the correct instrumentation. Is it more or less burdensome?