Abstract Interpretation with Alien Expressions and Heap Structures

Bar-Yehuda Chang  
University of California, Berkeley

K. Rustan M. Leino  
Microsoft Research

January 18, 2005

VMCAI 2005  
Paris, France

Problem and Motivation

- Standard abstract interpretation infer properties following a domain specific-schema of relations among (program) variables

\[ 0 \leq x \leq y \]
\[ z := 2 \cdot y - 2 \cdot x; \]
\[ 0 \leq z \]

- e.g., can infer this with Polyhedra [CH78]

Outline

- Overview
- Handling Alien Expressions
- Handling Heap Updates
- Concluding Remarks
Overview of Contributions

- To extend base domains to work with alien expressions
  - use a general abstract domain parameterized by base domains that hide alien expressions as fresh variables (cf. Nelson-Oppen)
  - congruence-closure abstract domain
- To deal with heap updates
  - track successive heaps as a separate base domain
  - heap succession abstract domain

Fooling the Base Domains

```
Constrain( sel(H, o, f) ≥ 8 )
```

Understandable to the Base Domain

```
2 \cdot x + sel(H, o, f) ≤ Absy - z
```

Understandable to the Base Domain

```
2 \cdot x + α ≤ Absy - z
```

Congruence-Closure Domain

- Store mappings in an equivalence graph (e-graph)
  - give the same symbolic value for equivalent expressions
- Tracks equalities of uninterpreted functions
  - an e-graph with abstract domain operations
  - symbolic values “name” equivalence classes of expressions
  - implements congruence closure
E-Graph

- $w = f(x) \land g(x, y) = f(y) \land w = h(w)$
- A set of mappings:
  
  $w \mapsto \alpha$
  $x \mapsto \beta$
  $f(i) \mapsto \epsilon$
  $y \mapsto \gamma$
  $g(\delta, \gamma) \mapsto \delta$
  $f(y) \mapsto \delta$
  $h(a) \mapsto \alpha$

- Always congruence-closed

Join

- Roughly, join the e-graphs, then join the base domains

Join of E-Graphs

- Think of the lattice over conjunctions of equalities (including infinite ones)
- Let $G = \text{Join}(G_0, G_1)$

Widen

- Widen the e-graphs, then widen the base domains

So Far We Have ...

- Reasoning for uninterpreted functions
- Base domains that work with alien expressions transparently
- What we need for field reads
  - `set` is alien to all base domains
Outline

- Overview
- Handling Alien Expressions
- Handling Heap Updates
- Concluding Remarks

Heap Updates

Java/C#

```java
if (p.g == 8) { o.f = x; }
```

Guarded

```java
assume H[p,g] == 8;
```

Commands

```java
H := H' where
sel(H', o, f) = x and
H' \equiv_{o,f} H
```

Abstract Interpreter

```java
Constrain( sel(H, p, g) = 8 )
Constrain( sel(H', o, f) = x )
Constrain( H' \equiv_{o,f} H )
Eliminate( H )
Rename( H', H )
```

Tracked by a new base domain: Heap Succession

Heap Update Example

Heap Succession

```java
H' \equiv_{o,f} H
```

E-Graph

```java
sel(H, p, g) \mapsto \alpha
8 \mapsto \alpha
sel(H', o, f) \mapsto \beta
x \mapsto \beta
p \mapsto p
H' \mapsto H'
g \mapsto g
o \mapsto o
f \mapsto f
```

Garbage values remain

- Only removes mapping
- “Lazy quantifier elimination”

Heap Update Example

Heap Succession

```java
H' \equiv_{o,f} H
```

E-Graph

```java
sel(H, p, g) \mapsto \alpha
8 \mapsto \alpha
sel(H', o, f) \mapsto \beta
x \mapsto \beta
p \mapsto p
H' \mapsto H'
g \mapsto g
o \mapsto o
f \mapsto f
```

Garbage values remain
Heap Update Example

Heap Succession

H' \equiv_{\alpha, f} H

E-Graph

\text{Can you give me an equivalent expression without } H\text{?}

1. Do Eliminate (H)
   - \text{EquivalentExp}r
     - \text{QueryExp} \times \text{Exp} \times \text{Var}
     - \text{Expr option}

2. To query other abstract domains (e.g., $d \equiv p$)
3. Conjoin Equalities

Conclusion and Future Work

- Extended the power of abstract domains to work with alien expressions using the congruence-closure domain
- Added reasoning about heap updates with the heap succession domain
- Close to having “cooperating abstract interpreters”?
  - missing propagating back equalities inferred by base domains
- Implementation and experiments in progress

Related Work

- Join for Uninterpreted Functions [Gulwani, Tiwari, Necula 2004]
  - same as our join for e-graphs
- Shape Analysis [many] and
  TVLA [Sagiv, Reps, Wilhelm, ...]
  - they abstract heap nodes into summary nodes
  - they use special “instrumentation predicates” whereas we use “off-the-shelf” abstract domains
  - could use shape analysis as base domain?

Thank you!

Questions? Comments?