Access-Based Localization for Octagons

Eva Beckschulze (RWTH Aachen),
Jörg Brauer (Verified Systems International GmbH, Bremen),
Stefan Kowalewski (RWTH Aachen)
Motivation

• Inefficiency in fixed-point computations:
  propagating large abstract states
  → requires memory
  → operations take a lot of time

• Idea of access-based localization:
  propagate only those parts of the abstract state that are accessed

• “Access Analysis-based Tight Localization of Abstract Memories”
  – written by Oh et al. (2011)
Localization for Intervals

main()
{
    ... 
    assume(x ≤ 4);
    assume(y ≤ 0);
    
    g();
    ... 
    
}

g()
{
    x++;
}

only x is accessed in g!
What about octagonal constraints?

- we deal with octagonal constraints of the form $\pm x \pm y \leq c$
- which relational constraints should be propagated to a procedure?
1st Approach: access-based

- relational constraint is not transferred to $g$
- update of $x + y \leq 2$ is lost
- delete relational constraint to be sound

$\Rightarrow$ loss of information
2nd Approach: dependency-based

- Transfer relational constraint, too
- relational constraint is updated correctly
- though, the reduction of the size of input state is smaller

© 2012 Eva Beckschulze, Embedded Software Laboratory, RWTH Aachen
3rd Approach: anchoring

• Introduce a slack variable \( x_{\text{slack}} = x \)
• use \( x_{\text{slack}} \) to keep track of all changes to \( x \)
• combine constraints to obtain relational constraints \( x + y_i \leq c_i + 1 \)
Implicit Constraints

• implicit constraints are those derived from other constraints
• e.g. \( x \leq 4 \) and \( y \leq 0 \) implies \( x + y \leq 4 \)
• octagonal transfer functions require that all implicit constraints have been derived, i.e. that the octagon is closed (normal form)
• the dependency-based approach suffers from implicit constraints (all variables depend on each other)
Significant Constraints

We call a constraint **significant** if it contains more information than interval bounds

→ Transfer only significant constraints (dep-based approach)
→ Introduce slack variables only for those variables that are part of a significant constraint (anchoring approach)
Localization Formally

\[ o_{out} = \text{Unaffected}_f(o) \cup f^\#(\text{ Relevant}_f(o)) \]

- Constraints that are redundant or get outdated
- Constraints that might be updated

Discard constraints \( \xrightarrow{\text{Discard}_f(o)} \) call \( \xrightarrow{\varepsilon} \) Relevant constraints \( \xrightarrow{\text{ Relevant}_f(o)} \) return \( \xrightarrow{f^\#(\text{ Relevant}_f(o))} \)

© 2012 Eva Beckschulze, Embedded Software Laboratory, RWTH Aachen
Experiments

• experiments based on a prototype implementation in JAVA
• flow and context sensitive value range analysis based on octagons
• each analysis starts in `main` with an octagon relating all accessed variables to each other and reduce the octagon at call sites
• we tested a few simple C programs
Experiments - Results

• with localization we can analyze more programs
  → analysis requires less memory
• the dependency-based approach still propagates many constraints
• localization decreases the number of necessary re-analyses of procedures
  – this effect is distinct for both the access-based and the anchoring approach
Related Work

- Octagon Packing (Miné 2006)
  - relate only few variables to each other
  - pre-analysis determines fixed packing
- Work on localization by Oh et al.
  - Access Analysis-based Tight Localization of Abstract Memories (2011)
  - Access-based Localization with Bypassing (2011)
  - Design and Implementation of Sparse Global Analyses for C-like Languages (2012)
Conclusion

- Localization for a relational domain is more complicated than for intervals.
- Determination of significant constraints is important for access-based localization for octagons.
Thank you very much!