Advanced Static Analysis Techniques

20190065 김기환 (kimkihwan@kaist.ac.kr)

* Abstract domains =

1. Non-relational

2. Relational

3. Product domain

Use two or more different abstract domains together

3-1. Reduced Product

Each abstraction refines other abstraction to be more precise

4. Disjunctive Completion

Use multiple abstraction with ‘exact abstract join’

In practice, it cost much and sometimes it’s not possible to construct ‘e.a.j.’

5. Cardinal Power

Abstraction is consisted with conjunction of implications

Because of left-hand-side of implication, it looks like partition under

5-1. State Partitioning

Cardinal power abstract domain for memory

flow-sensitive = Compute on each program point

set as a label of program

context-sensitive = Compute on each context of procedure calls

set as an info. context of procedure call

5-2. dynamic partitioning

Choose an abstract domain that fit the target program

5-3. Trace Partitioning

Cardinal power abstract domain for program point

* Abstract domain = set of logical properties + abstract operators
* Abstract operators =

1. abstract lattice operation =

2. transfer function =

3. abstraction relation =

* More precise abstract iteration skills =

1. Loop unrolling

Start abstract iteration after analyze few iteration

2-1. Delay widening

Before start widening, first few iterations use abstract union instead

2-2. Threshold widening

If value is inside the threshold, widening stops at pre-defined threshold

3. Abstract approximation (narrowing)

Do more iteration from fixpoint to gain more precision

* Sparse analysis = Efficient analysis without losing any precision
* Spatial sparsity = Store unnecessary info. of memory to every program part

Abstract garbage collection (a.k.a. Frame rule)

* Temporal sparsity = Read the memory after few times ago from the definition

Def-Use chain

*note. second property helps to recover over-approximated*

* Modular analysis = Analyze separately and combine them together

*note. it can save resources when user do analysis again after fixing some error*

1. Parameterization

Use parameterization to express pre-state (a.k.a. context)

2. Summary-based

Analyze based on each summary (= description of behavior of procedure)

3. Scalability

Do not need to analyze again for unchanged procedure

* Backward analysis = Calculate pre-condition from post-condition of given program

1. non-invertible = ‘E’ does not contain ‘x’

sol. Filtering with ‘x := E’ and erase all constraint about ‘x’

2. invertible = ‘E’ contains ‘x’

sol. Derive new constraint of ‘x’ with ‘E’

* Application of backward analysis =

1. Find necessary condition to reach given post-condition

2. Find sufficient condition to not reach given post-condition

3. Refine precision of forward analysis

Doing backward analysis with the result of forward analysis can give more precision than before