Program Analysis

Data Flow Analysis (Part 4)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2020/2021

Outline

- First example: Available expressions
- Basic principles
- More examples
- Solving data flow problems
- Inter-procedural analysis
- Sensitivities

Data Flow Equations

- **Transfer functions yield data flow equations for each statement**
  - At entry, e.g., $AE_{entry}(2) = ...$
  - At exit, e.g., $AE_{exit}(3) = ...$
- **How to solve these equations?**
  - Goal: Fix point, i.e., nothing changes anymore

Data Flow Equations

- Transfer functions yield **data flow equations** for **each statement**
  - At entry, e.g., $AE_{entry}(2) = ...$
  - At exit, e.g., $AE_{exit}(3) = ...$

May depend on each other

- **How to solve these equations?**
  - Goal: Fix point, i.e., nothing changes anymore

Naive Algorithm

Round-robin, iterative algorithm

- For each statement s
  - Initialize entry and exit set of s
- While sets are still changing
  - For each statement s
    - Update entry set of s by applying meet operator to exit sets of incoming statements
    - Compute exit set of s based on its entry set

Algorithms assume forward analysis (analogous for backward a.)

Naive Algorithm

Round-robin, iterative algorithm

- For each statement s
  - Initialize entry and exit set of s
- While sets are still changing
  - For each statement s
    - Update entry set of s by applying meet operator to exit sets of incoming statements
    - Compute exit set of s based on its entry set

Repeatedly computes each set, even if the input hasn't changed

Algorithms assume forward analysis (analogous for backward a.)

Work List Algorithm

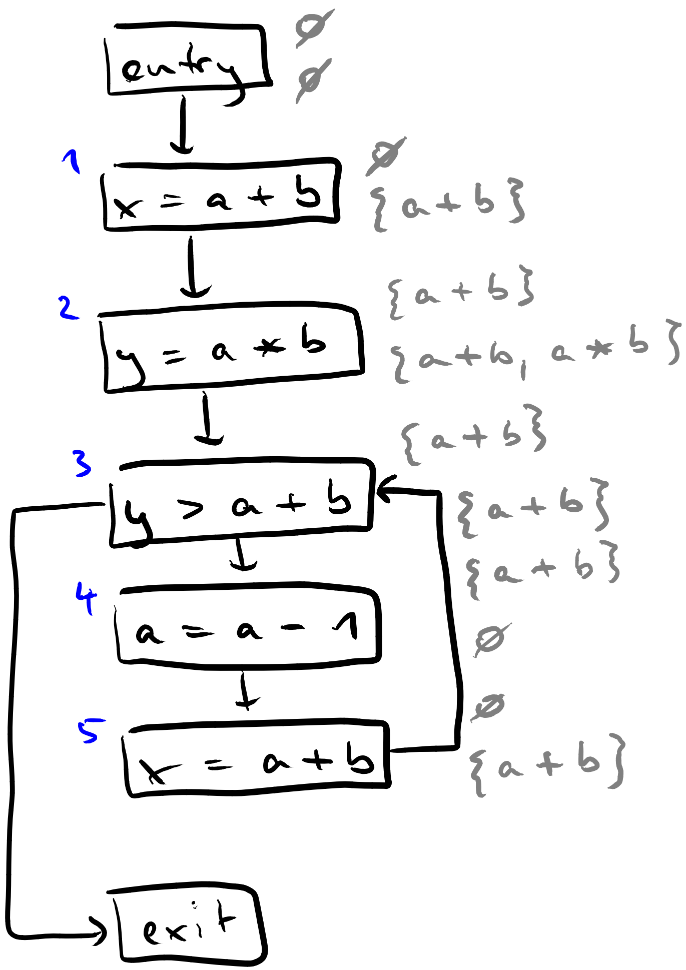
- For each statement s: Initialize entry and exit set
- Initialize W with initial/final node (for forward/backward analysis)
- While W not empty
  - Remove a statement s from W
  - Update entry set of s by applying meet operator to exit sets of incoming statements
  - Compute exit set of s based on its entry set
  - If exit set has changed (or statement visited for the first time): Add successors of s to W

Work List Algorithm

- For each statement s: Initialize entry and exit set
- Initialize $\boxed{W}$ with initial/final node (for forward/backward analysis)
- While $\boxed{W}$ not empty
  - Remove a statement s from W
  - Update entry set of s by applying meet operator to exit sets of incoming statements
  - Compute exit set of s based on its entry set
  - If exit set has changed (or statement visited for the first time): Add successors of s to W

Work list: Statements that need to be processed

Work List Algorithm : Example (Avail. Expr.)



W :

Convergence

Will it always terminate?

- In principle, work list algorithms may run forever
- Impose constraints to ensure termination
  - Domain of analysis: Partial order with finite height
    - No infinite ascending chains $X1 < X2 < ...$
  - Transfer function and meet operator:

    Monotonic w.r.t. partial order
    - Sets stay the same or grow larger

Convergence

Will it always terminate?

- In principle, work list algorithms may run forever
- Impose constraints to ensure termination
  - Domain of analysis: Partial order with finite height
    - No infinite ascending chains $X1 < X2 < \ldots$
  - Transfer function and meet operator:

    Monotonic w.r.t. partial order
    - Sets stay the same or grow larger

Monotone framework