Dataflow analysis termination and correctness, and widening operators and collecting semantics

Jonathan Lam

09/25/21

1 Ch6: Dataflow analysis termination and correctness

1.1 Termination

- At each program point, the dataflow values over time represent an **ascending chain**: a sequence σ_k with the property $n \leq m \Leftrightarrow \sigma_n \sqsubseteq \sigma_m$
  - Ascending chain has finite height h if it contains $h+1$ elements
  - A lattice $(L, \sqsubseteq)$ has finite height h if there is an ascending chain in the lattice of height h (and none with larger height)
- Show that for a lattice of finite height, and monotonic flow functions, the lattice algorithm is guaranteed to terminate (Theorem 1: datalow analysis termination)
  - Need to show that the output information $\sigma_o^{'}$ is at least as high in the lattice as the old output information σ_o, which will be true if our flow functions are **monotonic**, i.e., iff $\sigma_1 \sqsubseteq \sigma_2 \Rightarrow f(\sigma_1) \sqsubseteq f(\sigma_2)$.
  - Consider the case $f_Z[x := 0](\sigma) = \sigma[x \mapsto Z]$; this can actually narrow $\sigma(x)$ from $\top$ to Z, but it is still monotonic from the definition

1.2 Correctness

- **Correctness**: the program analysis results correctly describe every actual execution of the program

- **Program trace**: a trace T of a program P is a potentially infinite sequence $[c_0, c_1, \dots]$ of program configurations, where $c_0 = E_0, 1$ is called the initial configuration, and for every $i \geq 0$ we have $P \vdash c_i \rightarrow c_{i+1}$
  - Plain English (PE): program trace is a sequence of program configurations that can be validly inferred
- **Dataflow analysis soundness**: The result $\{\sigma_n \mid n \in P\}$ of a program analysis running on program P is sound iff, for all traces T of P, for all i such that $0 \leq i < \mathit{length}(T)$, we have $\alpha(c_i) \sqsubseteq \sigma_{n_i}$.
  - PE: a dataflow analysis is sound if, for all program traces, any intermediate dataflow analysis never becomes more general than its end result (the end result is the most general bound of all instances of the program point in all possible program traces)
- **Local soundness**: A flow function f is locally found iff $P \vdash c_i \rightarrow c_{i+1}$ and $\alpha(c_i) \sqsubseteq \sigma_{n_i}$ and $f[P[n_i]](\sigma_{n_i}) = \sigma_{n_{i+1}}$ implies $\alpha(c_{i+1}) \sqsubseteq \sigma_{n_{i+1}}$.
  - PE: a flow function is sound if it always produces the correct analysis that produces a sound overall dataflow
- **Fixed point**: A dataflow analysis result $\{\sigma_i \mid i \in P\}$ is a fixed point iff $\sigma_0 \sqsubseteq \sigma_1$ where σ_0 is the initial analysis information and σ_1 is the information before the first instruction, and for each instruction i we have $\bigsqcup_{j \in \mathit{preds}(i)} f[P[j]](\sigma_j) \sqsubseteq \sigma_i$
  - PE: The fixed point represents a dataflow analysis that yields no changes on the next iteration
  - The worklist algorithm computes a fixed point when it terminates
- *Theorem 2* (a fixed point of a locally sound analysis is globally sound): If a dataflow analysis's flow function f is monotonic and locally sound, and for all traces T we have $\alpha(c_0) \sqsubseteq \sigma_0$ where σ_0 is the initial analysis information, then any fixed point $\{\sigma_n \mid n \in P\}$ of the analysis is sound.

2 Ch7: Widening operators and collecting semantics for dataflow analysis

2.1 Widening operators: dealing with infinite-height lattices

- **Interval analysis**: tracks the interval of values that each variable might hold
  - May be useful for arrays bounds checking
  - Infinite range, with positive and negative infinity sentinels
- Widening analysis considers the most recent two elements in a chain. If the second is higher than the first, the widening operator can choose to jump up in a chain. We can jump up to the upper limit.
  - Define the **widening operator** as follows:

$$W(\perp, l_{current}) = l_{current} \tag{1}$$

$$W([l_1, h_1], [l_2, h_2]) = \left[\min_W(l_1, l_2), \max_W(h_1, h_2)\right] \tag{2}$$

    where $\min_W(l_1, l_2) = l_1$ if $l_1 \leq l_2$ and $-\infty$ otherwise, and $\max_W(h_1, h_2) = h_1$ if $h_1 \geq h_2$ and ∞ otherwise
  - Require two properties of widening operators:
    * Must return the upper bound of its operands
    * When the widening operator is applied to an ascending chain l_i, the resulting ascending chain l_i^W must be of finite height. Define $l_0^W = l_0$ and $\forall i > 0. l_i^W = W(l_{i-1}^W, l_i)$
- Strategies for the widening operator:
  - Only apply the widening operator when needed, since we lose precision; can apply it only at the head of loops
  - Don't immediately jump to infinity, but jump to a finite number of steps (e.g., the constants in the program)
- Unrelated note: $\perp$ can be seen as a natural representation for dataflow values that propagate along a path that is infeasible

2.2 Collecting semantics (reaching definitions)

- **Collecting semantics**: a version of program semantics that has been augmented with additional information necessary for some particular analysis
  - Example: reaching definitions: collect information about where variables were declared as well as which variables definitions reach
  - Used for reasoning about correctness (??)