Context-, Flow- and Field-Sensitive Data-Flow Analysis using Synchronized Pushdown Systems

Johannes Späth, Karim Ali, Eric Bodden

30. 10. 2020

Abstract

- precise static analyses are context-, flow-, and field-sensitive
- *k*-limited access path, access graph do not scale well on recursive data structures
- context- and field- sensitivity both expressible as CFL reachability problems
- introducing synchronized push-down system (SPDS)
  - field-sensitive
  - flow-sensitive
  - context-sensitive

Outline

- data-flow analysis and its properties
- pushdown systems
- call-PDS
- field-PDS
- SPDS

Data-flow analysis

- static analysis
- for each program point: set of computed values
- *flow* functions give semantics of instructions
- *join* function combines values from multiple predecessors

Data-flow analysis: a simple approach

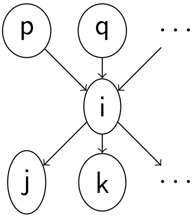

Figure: Control flow graph fragment

Data-flow analysis: a simple approach

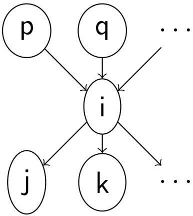

Figure: Control flow graph fragment

$$out_i = flow_i(join(out_p, out_q, \cdots))$$

Properties of data-flow analysis

Flow-sensitivity
control flow paths

Context-sensitivity
call contexts

Field-sensitivity
fields of the same object

Push-down systems

A push-down system is a triple $\mathcal{P} = (P, \Gamma, \Delta)$ where:

- P: finite set called control locations
- Γ: finite set called stack alphabet
- Δ: finite set of rules

Configuration is a pair $\langle\langle p, w \rangle\rangle$ where $p \in P$, $w \in \Gamma^*$

Rules

Δ is a set of rules in form: $\langle\langle p, \gamma \rangle\rangle \rightarrow \langle\langle p', w \rangle\rangle$

- $p, p' \in P$
- $\gamma \in \Gamma$
- $w \in \Gamma^*$

Rules

Δ is a set of rules in form: $\langle\!\langle p, \gamma \rangle\!\rangle \to \langle\!\langle p', w \rangle\!\rangle$

Types of rules

- $|w| = 0$ pop rules
- $|w| = 1$ normal rules
- $|w| = 2$ push rules

Rules with $|w| > 2$ can be subdivided into multiple push rules.

call-PDS

- flow-sensitive
- context-sensitive
- field-insensitive

Intuition

- push rule $\approx$ call
- pop rule $\approx$ return
- normal rule $\approx$ assignment

Construction of call-PDS

- $\mathcal{P} = (\mathbb{V}, \mathbb{S}, \Delta_{\mathbb{S}})$
- $\mathbb{V}$: variables
- $\mathbb{S}$: statements
- $\Delta_{\mathbb{S}}$
  - normal rules – intra-procedural data-flows
  - push,pop rules – inter-procedural data-flows

Analysing call-PDS

- set of configurations reachable from given configuration
  - where does the value flow?
- $\mathcal{P}$-automaton $\mathcal{A}_{\mathbb{S}}$ accepts configuration of PDS
- start with trivial automaton
- post*-saturate the trivial automaton
  - IA159 Formal verification methods

field-PDS

- flow-sensitive
- context-insensitive
- field-sensitive

Intuition

- normal rule ≈ no modification, arguments, return
- push rule ≈ store into field
- pop rule ≈ load from field

Construction of field-PDS

- $\mathcal{P} = (\mathbb{V} \times \mathbb{S}, \mathbb{F}, \Delta_{\mathbb{F}})$
- $\mathbb{V}$: variables
- $\mathbb{S}$: statements
- $\mathbb{F}$: fields
- configuration $\langle\!\langle x@s, f_0 \cdot f_1 \cdots \rangle\!\rangle$

Construction of field-PDS

Example rules

Push

- statement 36 | `v.f = u`
- rule $\langle\!\langle u@35, * \rangle\!\rangle \rightarrow \langle\!\langle v@36, f \cdot * \rangle\!\rangle$

Pop

- statement 37 | `x = w.f`
- rule $\langle\!\langle w@36, f \rangle\!\rangle \rightarrow \langle\!\langle x@37, \varepsilon \rangle\!\rangle$

SPDS
Intuition

- we can compute context-sensitive dataflow
- we can compute field-sensitive dataflow
- combination: precise dataflow holds only if it holds in both cases

SPDS

A little formally

- call-PDS configuration: $\langle\!\langle x, s_0 \cdot s_1 \cdots \rangle\!\rangle$
- field-PDS configuration: $\langle\!\langle v@s, f_0 \cdot f_1 \cdots \rangle\!\rangle$
- SPDS configuration: $\langle\!\langle v \cdot f_0 \cdot f_1 \cdots @ s_0^{s_1 \cdot s_2 \cdots} \rangle\!\rangle$

- $\mathcal{P}$-automatons $\mathcal{A}_{\mathbb{S}}$ (call-PDS) and $\mathcal{A}_{\mathbb{F}}$ (field-PDS)
- let $\mathcal{A}_{\mathbb{S}}^{\mathbb{F}} = (\mathcal{A}_{\mathbb{S}}, \mathcal{A}_{\mathbb{F}})$
- $\mathcal{A}_{\mathbb{S}}^{\mathbb{F}}$ accepts iff both accept

Undecidability

- context-sensitive data-dependence analysis is generally undecidable :(
- SPDS over-approximates the fully precise solution

Undecidability

- context-sensitive data-dependence analysis is generally undecidable :(
- SPDS over-approximates the fully precise solution
- what if the configuration is reachable via different CFG paths?
- let's look at an example!