Typage et analyse statique

week 4: Algorithmic approach

2019-2020
Giovanni Bernardi, gioXYZirif.fr
http://www.irif.fr/~gio/index.xhtml
Université Paris Diderot
Recap

- the WHILE language
- an operational semantics for it
- four static analysis defined over statements
- monotone framework / constant propagation
The overall approach

Given a statement $S$ we extract the data about $S$

\[ \text{labels}(S), \text{flow}(S), \text{FV}(S), \text{AExp}, \ldots \]

that we use to define a monotone framework (MF)

\[ (L, \mathcal{F}, F, E, \iota, f) \]

▶ the entities in MF depend on the code $S$,

\[
\begin{array}{c}
\left\langle P(\text{AExp}), \supseteq \right\rangle \\
\end{array}
\]

▶ MF give rise to a system of equations (two equations for each label in $S$)
General form of the equations

The following definitions depend only on \((L, \mathcal{F}, F, E, \iota, f)\),

\[
\text{Analysis}_\circ(\ell) = \bigsqcup \{ \text{Analysis}_\bullet(\ell') \mid (\ell', \ell) \in F \} \sqcup \iota_E^\ell
\]

where \(\iota_E^\ell = \begin{cases} 
\iota & \text{if } \ell \in E \\
\bot & \text{otherwise}
\end{cases}\)

\[
\text{Analysis}_\bullet(\ell) = f(\ell)(\text{Analysis}_\circ(\ell))
\]
Mutual recursion and never-ending computations

Recall the program

\[
[x := 5]_1; [y := 1]_2; \text{while } [x > 1]_3 \text{ do } [y := x \times y]_4; [x := x - 1]_5
\]

We have that

\[
\text{RD}_{entry}(4) = \text{RD}_{exit}(3) = \text{RD}_{entry}(3)
\]
\[
= \text{RD}_{exit}(2) \cup \text{RD}_{exit}(5)
\]
\[
= (\text{RD}_{entry}(2) \setminus \{(y,?), (y, 2), (y, 4)\} \cup \{(y, 2)\})
\]
\[
\cup (\text{RD}_{entry}(5) \setminus \{(x,?), (x, 1), (x, 5)\} \cup \{(x, 5)\})
\]
\[
= \text{RD}_{exit}(1) \setminus \{(x,?), (x, 1), (x, 5)\} \setminus \{(y,?), (y, 2), (y, 4)\}
\]
\[
\cup \text{RD}_{exit}(4) \setminus \{(x,?), (x, 1), (x, 5)\} \cup \{(x, 1), (y, 2), (x, 5)\}
\]
\[
= \text{RD}_{entry}(4) \setminus \{(x,?), (x, 1), (x, 5)\} \cup \{(x, 1), (y, 2), (x, 5), (y, 4)\}
\]
Mutual recursion and never-ending computations

Recall the program

\[
\begin{align*}
[x := 5] \quad [y := 1] ; \\
\text{while } [x > 1] \quad \text{do } [y := x * y] ; [x := x - 1]
\end{align*}
\]

We have that

\[
\begin{align*}
\text{RD}_{\text{entry}}(4) & = \text{RD}_{\text{exit}}(3) = \text{RD}_{\text{entry}}(3) \\
& = \text{RD}_{\text{exit}}(2) \cup \text{RD}_{\text{exit}}(5) \\
& = (f(2)(\text{RD}_{\text{entry}}(2))) \\
& \quad \cup f(5)(\text{RD}_{\text{entry}}(5)) \\
& = \text{RD}_{\text{exit}}(1) \setminus \{(x,?), (x,1), (x,5)\} \setminus \{(y,?), (y,2), (y,4)\} \\
& \quad \cup \text{RD}_{\text{exit}}(4) \setminus \{(x,?), (x,1), (x,5)\} \cup \{(x,1), (y,2), (x,5)\} \\
& = \text{RD}_{\text{entry}}(4) \setminus \{(x,?), (x,1), (x,5)\} \cup \{(x,1), (y,2), (x,5), (y,4)\}
\end{align*}
\]
Mutual recursion and never-ending computations

Recall the program

\[
\begin{align*}
&x := 5; \\
y := 1; \\
\text{while } x > 1 \text{ do } [y := x \ast y] \\
&[x := x - 1]
\end{align*}
\]

We have that

\[
\begin{align*}
\text{RD}_{\text{entry}}(4) &= \text{RD}_{\text{exit}}(3) = \text{RD}_{\text{entry}}(3) \\
&= \text{RD}_{\text{exit}}(2) \cup \text{RD}_{\text{exit}}(5) \\
&= (f(2)(\text{RD}_{\text{entry}}(2))) \\
&\quad \cup f(5)(\text{RD}_{\text{entry}}(5)) \\
&= f(2)(f(1)(\text{RD}_{\text{entry}}(1))) \\
&\quad \cup f(5)(f(4)(\text{RD}_{\text{entry}}(4))) \\
&= \text{RD}_{\text{entry}}(4) \setminus \{(x,?), (x,1), (x,5)\} \cup \{(x,1), (y,2), (x,5), (y,4)\}
\end{align*}
\]
Mutual recursion and never-ending computations

Recall the program

\[ x := 5;\ y := 1;\ \text{while}\ [x > 1] \ do\ [y := x \times y;\ [x := x - 1] \]

We have that

\[
\begin{align*}
\text{RD}_{\text{entry}}(4) & = \text{RD}_{\text{exit}}(3) = \text{RD}_{\text{entry}}(3) \\
& = \text{RD}_{\text{exit}}(2) \cup \text{RD}_{\text{exit}}(5) \\
& = (f(2)(\text{RD}_{\text{entry}}(2))) \\
& \quad \cup f(5)(\text{RD}_{\text{entry}}(5)) \\
& = f(2)(f(1)(\iota)) \\
& \quad \cup f(5)(f(4)(\text{RD}_{\text{entry}}(4))) \\
& = \text{RD}_{\text{entry}}(4) \setminus\{(x,?), (x,1), (x,5)\} \cup \{(x,1), (y,2), (x,5), (y,4)\}
\end{align*}
\]
Mutual recursion and never-ending computations

Recall the program

\[ \begin{align*}
  x &:= 5^1; y := 1^2; \text{while } x > 1^3 \text{ do } [y := x \times y]^4; [x := x - 1]^5
\end{align*} \]

We have that

\[
\begin{align*}
  \text{RD}_{\text{entry}}(4) &= \text{RD}_{\text{exit}}(3) = \text{RD}_{\text{entry}}(3) \\
  &= \text{RD}_{\text{exit}}(2) \cup \text{RD}_{\text{exit}}(5) \\
  &= (f(2)(\text{RD}_{\text{entry}}(2))) \\
  &\quad \cup f(5)(\text{RD}_{\text{entry}}(5)) \\
  &= f(2)(f(1)(\iota)) \\
  &\quad \cup f(5)(f(4)(\text{RD}_{\text{entry}}(4)))
\end{align*}
\]

\[
\begin{align*}
  \text{Analysis}_\circ(4) &= (f(5) \circ f(4))(\text{Analysis}_\circ(4))) \cup \{(x,1),(y,2),(x,5),(y,4)\}
\end{align*}
\]

in general due to mutual recursion there exist \(\ell\)'s and \(H\) s.t.

\[
\text{Analysis}_\circ(\ell) = H((f(\ell_n) \circ \cdots \circ f(\ell_1))(\text{Analysis}_\circ(\ell)))
\]
\[
x_1 = (f(\ell_{1n}) \circ \cdots \circ f(\ell_{11}))(x_1)
\]
\[
\vdots
\]
\[
x_m = (f(\ell_{mn}) \circ \cdots \circ f(\ell_{1m}))(x_m)
\]

We have to deal with circularity in the definitions

\[\downarrow\]

We have to solve equations!
Paths and their functions

Paths up to $\ell$ but not including $\ell$:

$$\text{path}_\circ(\ell) = \{ \ell_1 \ell_2 \ldots \ell_{n-1} \mid n \geq 1 \text{ and } \forall i < n. (\ell_i, \ell_{i+1}) \in F \text{ and } \ell_n = \ell \text{ and } \ell_1 \in E \}$$

Paths up to $\ell$ and including $\ell$:

$$\text{path}_\bullet(\ell) = \{ \ell_1 \ell_2 \ldots \ell_n \mid n \geq 1 \text{ and } \forall i < n. (\ell_i, \ell_{i+1}) \in F \text{ and } \ell_n = \ell \text{ and } \ell_1 \in E \}$$

Every block $B^\ell$ is associated to a function $f_\ell$, thus given a path $\vec{\ell} = \ell_1 \ell_2 \ldots \ell_n$ we let

$$f_{\vec{\ell}} = f(\ell_n) \circ \cdots \circ f(\ell_1) \circ id$$

note that $f_\varepsilon = id$. 
Meet over all paths solution (MOP)

misleading name: we use the join (lub), not the meet (glb).

\[
MOP_\circ(\ell) = \bigsqcup \{ f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in \text{path}_\circ(\ell) \}
\]

\[
MOP_\bullet(\ell) = \bigsqcup \{ f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in \text{path}_\bullet(\ell) \}
\]

Great! No circularity in definitions.
Meet over all paths solution (MOP)

misleading name: we use the join (lub), not the meet (glb).

\[
MOP_\circ(\ell) = \bigcup \{ f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in path_\circ(\ell) \}
\]

\[
MOP_\bullet(\ell) = \bigcup \{ f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in path_\bullet(\ell) \}
\]

Great! No circularity in definitions.

Lemma (2.31 (NNH))

The MOP solution for Constant Propagation is undecidable.

Why? There exists a program in WHILE such that \(MOP_\bullet(\ell)\) maps a given variable to 1 if and only if the Modified Post Correspondence Problem has no solution. Since the MPCP is undecidable so is the MOP solution.
Let’s change approach

By definition mutually recursive functions

\[
\begin{align*}
\text{Analysis}_\circ (\ell) &= \bigsqcup \{ \text{Analysis}_\bullet (\ell') \mid (\ell', \ell) \in F \} \sqcup \iota_E \\
\text{Analysis}_\bullet (\ell) &= f(\ell)(\text{Analysis}_\circ (\ell))
\end{align*}
\]
Let's change approach

By definition recursive functions

\[
\text{Analysis}_\circ (\ell) = \bigsqcup \{ f(\ell')(\text{Analysis}_\circ (\ell')) \mid (\ell', \ell) \in F \} \sqcup \iota^\ell_E
\]

\[
\text{Analysis}_\bullet (\ell) = f(\ell)(\bigsqcup \{ \text{Analysis}_\bullet (\ell') \mid (\ell', \ell) \in F \} \sqcup \iota^\ell_E)
\]
Let’s change approach

By definition recursive functions

\[
\text{Analysis}_\circ(\ell) = \bigsqcup \{ f(\ell')(\text{Analysis}_\circ(\ell')) \mid (\ell', \ell) \in F \} \sqcup \iota^\ell_E
\]

\[
\text{Analysis}_\bullet(\ell) = f(\ell)(\bigsqcup \{ \text{Analysis}_\bullet(\ell') \mid (\ell', \ell) \in F \} \sqcup \iota^\ell_E)
\]

We abstract away and investigate the poset \( \langle \text{labels}(S) \to L, \leq \rangle \),

- \( \text{labels}(S) \to L \) set of functions from \( \text{labels}(S) \) to \( L \)
- \( \leq \) is the usual point-wise order of functions

Actually \( \langle \text{labels}(S) \to L, \leq \rangle \) is a complete lattice.

Consider the following equations,

\[
X(\ell) = \bigsqcup \{ f(\ell')(X(\ell')) \mid (\ell', \ell) \in F \} \sqcup \iota^\ell_E
\]

\[
Y(\ell) = f(\ell)(\bigsqcup \{ Y(\ell') \mid (\ell', \ell) \in F \} \sqcup \iota^\ell_E)
\]
Solutions of equations / fixed points

Can we find the least solutions of the following equations?

\[
X(\ell) = \bigcup \{ f(\ell')(X(\ell')) \mid (\ell', \ell) \in F \} \sqcup i^\ell_E
\]

\[
Y(\ell) = f(\ell)(\bigcup \{ Y(\ell') \mid (\ell', \ell) \in F \} \sqcup i^\ell_E)
\]

=  

Can we find the least fixed points of the following functions?

\[
H(X) = \lambda \ell. \bigcup \{ f(\ell')(X(\ell')) \mid (\ell', \ell) \in F \} \sqcup i^\ell_E
\]

\[
G(Y) = \lambda \ell. f(\ell)(\bigcup \{ Y(\ell') \mid (\ell', \ell) \in F \} \sqcup i^\ell_E)
\]

Note that \( H, G : (\text{labels}(S) \rightarrow L) \rightarrow (\text{labels}(S) \rightarrow L) \) and that both functions are monotone because \( f(\ell) \) is monotone, so by Knaster-Tarski their least fixed points exist.
Maximal Fixed Point solution (MFP)

Misleading name: the computed fixed point is the minimum.

In the following

- We use the following representation of a monotone framework

\[(L, \mathcal{F}, F, E, \nu, f)\]

\[\downarrow\]

\[latt, \_ , flow, ext, i, f\]

- analysis, mfpIn, mfpOut are mutable arrays of values in \(L\) indexed by the labels

```haskell
let solveMF latt ext i flow f =
  let labels = ext \(\cup\) \{\(\ell\) | \((\ell, \_ )\) \(\in\) flow \(\cup\) flow\(^{-1}\}\) in
  initAnalysis latt ext i flow;
  updateInfo latt flow f analysis;
  result labels analysis;;
```
Maximal Fixed Point solution (MFP)

let rec initAnalysis labels latt ext i =
    match labels with
    | [] -> ()
    | l :: ls -> let v = if l ∈ ext then i else ⊥ latt in
       analysis[l] := v;
       initAnalysis ls latt ext i

let rec result labels f analysis =
    match labels with
    | [] -> (mfpIn, mfpOut)
    | l :: ls -> mfpIn[l] := analysis[l];
              mfpOut[l] := f(l)(analysis[l]);
    result ls f analysis;;
Maximal Fixed Point solution (MFP)

1 (* w contains pairs of labels to analyse *)
2 let rec updateInfo latt w flow f analysis =
3 match w with
4 | [] -> analysis
5 | (x,y) :: tail ->
6   if f(x)(analysis[x]) ⊑ analysis[y]
7     then analysis[y] := analysis[y] ⊔ f(x)(analysis[x]);
8     let add = filter (fun (fst,_) = fst = y) flow in
9     in updateInfo latt (add @ tail) flow f analysis
10   else updateInfo latt tail flow f analysis

mind

the code above is not ocaml, it is pseudo code!
Maximal Fixed Point solution (MFP)

**Lemma**

*If in MF the lattice enjoys ACC, then solveMF MF terminates.*

- The functions initAnalysis and result scan the list labels, whose size decreases at each recursive call, and thus they terminate.

- Why does the function updateInfo terminate?

  By hypothesis *L* enjoys ACC, and hence, thanks to line 6, the condition in line 5 can be true only a finite number of times. It follows that the number of recursive calls on a list `add @ tail` is finite (the size of `add @ tail` is also finite). All the other recursive calls are on the list `tail`, whose size is smaller than the size of `w`, thus the function must terminate.

---

1 Argument similar to proof of Lemma 1.8 [NNH]
Maximal Fixed Point solution (MFP)

Let \( \overline{\text{analysis}_o} \) and \( \overline{\text{analysis}_\bullet} \) be the least solution to the MF given as input to \( \text{solveMF} \), that is to the equations

\[
X(\ell) = \bigcup \{ f(\ell')(X(\ell')) \mid (\ell', \ell) \in F \} \sqcup \iota_E
\]

\[
Y(\ell) = f(\ell)(\bigcup \{ Y(\ell') \mid (\ell', \ell) \in F \} \sqcup \iota_E)
\]

That is \( \overline{\text{analysis}_o} = \mu H \) and \( \overline{\text{analysis}_\bullet} = \mu G \).

Let \( (\text{mfpIn}, \text{mfpOut}) = \text{solveMF} \) MF.
Correctness

Lemma

For every $\ell \in \text{labels}(S)$. $\text{mfpIn}[\ell] = \overline{\text{analysis}}(\ell)$.

Proof.

(a) After initAnalysis and every call to updateInfo invariant

$$\forall \ell \in \text{labels}(S). \text{analysis}[\ell] \sqsubseteq \overline{\text{analysis}}(\ell) \quad (1)$$

(b) After updateInfo terminates, for every $\ell \in \text{labels}(S)$

$$\bigsqcup \{ f(\ell')(\text{analysis}[\ell']) \mid (\ell', \ell) \in F \} \sqcup \nu^\ell_E \sqsubseteq \text{analysis}[\ell]$$

$$H(\text{analysis})(\ell) \sqsubseteq$$

that is $H(\text{analysis}) \leq \text{analysis}$.

After result, mfpIn = analysis, thus $H(\text{mfpIn}) \leq \text{mfpIn}$, and hence by Knaster-Tarski $\mu H \leq \text{mfpIn}$, but by definition $\overline{\text{analysis}} = \mu H$, thus $\overline{\text{analysis}} \leq \text{mfpIn}$. Thanks to antisimmetry and to (1) we conclude $\forall \ell. \text{analysis}(\ell) = \text{mfpIn}[\ell]$. \qed
Correctness

Proof of (a)

After $\text{initAnalysis}$ $\forall \ell \in \text{labels}(S). \text{analysis}[\ell] \in \{\bot_L, \iota\}$ thus $\forall \ell \in \text{labels}(S). \text{analysis}[\ell] \sqsubseteq \text{analysis}_\circ(\ell)$.

By definition

$$\text{Analysis}_\circ(\ell) = \bigsqcup\{ \text{Analysis}_\bullet(\ell') \mid (\ell', \ell) \in F\} \sqcup \iota_E^\ell$$

$$= \bigsqcup\{ f(\ell')(\text{Analysis}_\circ(\ell')) \mid (\ell', \ell) \in F\} \sqcup \iota_E^\ell$$

To reason on $\text{updateInfo}$ consider the value $\text{analysis}[\ell]$ before and after a call, denoted

- $\text{anBefore}[\ell]$
- $\text{anAfter}[\ell]$

and observe that $w$ contains only pairs in $\text{flow}$. 

24
Correctness
Proof of (a)

Fix a $y \in labels(S)$. By the invariant

$$\text{anBefore}[y] \subseteq \text{analysis}_\circ(y)$$

(2)

Either $\text{anAfter}[y] = \text{anBefore}[y]$ and we are done thanks to (2), or due to line 6 for some $x$ s.t. $(x, y) \in F$ we have

$$\text{anAfter}[y] = \text{anBefore}[y] \sqcup f(x)(\text{anBefore}[x])$$

Again by the invariant $\text{anBefore}[x] \subseteq \text{analysis}_\circ(x)$. As $f(x)$ is monotone, this implies

$$f(x)(\text{anBefore}[x]) \subseteq f(x)(\text{analysis}_\circ(x))$$

(3)

and thus $\text{anAfter}[y] \subseteq \text{analysis}_\circ(y) \sqcup f(x)(\text{analysis}_\circ(x))$. 
Correctness

Proof of (a)

By assumption \( \text{analysis}_\circ = \mu H \), thus

\[ \text{analysis}_\circ(y) = \bigcup \{ f(z)(\text{analysis}_\circ(z)) \mid (z, y) \in F \} \sqcup \iota_E. \]

Since \((x, y) \in F\) we have in particular that

\[ \text{analysis}_\circ(y) = f(x)(\text{analysis}_\circ(x)) \sqcup \bigcup \{ f(z)(\text{analysis}_\circ(z)) \mid (z, y) \in F \} \sqcup \iota_E. \]

It follows that

\[ f(x)(\text{analysis}_\circ(x)) \sqsubseteq \text{analysis}_\circ(y) \quad (4) \]

Now we obtain

\[ \text{anAfter}[y] = \text{anBefore}[y] \sqcup f(x)(\text{anBefore}[x]) \]

\[ \sqsubseteq \text{analysis}_\circ(y) \sqcup f(x)(\text{anBefore}[x]) \quad \text{By (2)} \]

\[ \sqsubseteq \text{analysis}_\circ(y) \sqcup f(x)(\text{analysis}_\circ(x)) \quad \text{By (3)} \]

\[ \sqsubseteq \text{analysis}_\circ(y) \quad \text{By (4)} \]
Lattices

The algebraic approach

For the project you will implement the lattice \texttt{latt}.

Why ?

What are the operations of a lattice ?

Theorem (2.9 (DP))

Let $L$ be a lattice. Then $\lor$ and $\land$ satisfy for all $a, b, c \in L$

- \[(L1) \quad (a \lor b) \lor c = a \lor (b \lor c)\] \hspace{1cm} \text{associative laws}
- \[(L1)^{\delta} \quad (a \land b) \land c = a \land (b \land c)\]
- \[(L2) \quad a \lor b = b \lor a\] \hspace{1cm} \text{commutative laws}
- \[(L2)^{\delta} \quad a \land b = b \land a\]
- \[(L3) \quad a \lor a = a\] \hspace{1cm} \text{idempotency laws}
- \[(L3)^{\delta} \quad a \land a = a\]
- \[(L4) \quad a \lor (a \land b) = a\] \hspace{1cm} \text{absorption laws}
- \[(L4)^{\delta} \quad a \land (a \lor b) = a\]
Lattices
The algebraic approach

For the project you will implement the lattice \texttt{latt}.

\textbf{Why ?}

\textbf{What are the operations of a lattice ?}

\textbf{Theorem (2.10 (DP))}

Let \( \langle L, \lor, \land \rangle \) be a non-empty set equipped with two binary operations which satisfy (L1)-(L4) and (L1)\( ^\delta \)-(L4)\( ^\delta \).

(i) For all \( a, b \in L \), we have \( a \lor b = b \iff a \land b = a \).

(ii) Define \( \leq \) on \( L \) by letting \( a \leq b \) if \( a \lor b = b \). Then \( \leq \) is a partial order.

(iii) With \( \leq \) as in (ii), \( \langle L, \leq \rangle \) is a lattice in which for all \( a, b \in L \),

\[ a \lor b = \text{sup}\{a, b\}, \quad a \land b = \text{inf}\{a, b\}. \]

\textbf{You know the properties the operations must satisfy}
Project

You have

- all the theoretical ingredients,
- the pseudo-code of the main algorithm.

TO DO:

1. Implement the function that generates all the equations of the reaching definition and the available expressions analysis for any statement $S$ No procedure declarations / calls

2. Implement the function `solveMF` and use it to run the reaching definitions and the available expression analysis over any statement $S$ No procedure declarations / calls

All the code must be in gaufre

GitLab

https://gaufre.informatique.univ-paris-diderot.fr/
Material: Section 2.4 [NNH]