Monotone Frameworks

- Monotone and Distributive Frameworks
- Instances of Frameworks
- Constant Propagation Analysis

The Overall Pattern

Each of the four classical analyses take the form

$$\begin{array}{rcl} \mathit{Analysis}_\circ(\ell) & = & \begin{cases} \iota & \text{if } \ell \in E \\ \bigsqcup\{\mathit{Analysis}_\bullet(\ell') \mid (\ell', \ell) \in F\} & \text{otherwise} \end{cases} \\ \mathit{Analysis}_\bullet(\ell) & = & f_\ell(\mathit{Analysis}_\circ(\ell)) \end{array}$$

where

- $\bigsqcup$ is $\bigcap$ or $\bigcup$ (and $\sqcup$ is $\cup$ or $\cap$),
- F is either $\mathit{flow}(S_\star)$ or $\mathit{flow}^R(S_\star)$,
- E is $\{\mathit{init}(S_\star)\}$ or $\mathit{final}(S_\star)$,
- ι specifies the initial or final analysis information, and
- f_ℓ is the transfer function associated with $B^\ell \in \mathit{blocks}(S_\star)$.

The Principle: forward versus backward

- The *forward analyses* have F to be *flow*$(S_\star)$ and then *Analysis*$_\circ$ concerns entry conditions and *Analysis*$_\bullet$ concerns exit conditions; the equation system presupposes that $S_\star$ has isolated entries.

- The *backward analyses* have F to be *flow*$^R(S_\star)$ and then *Analysis*$_\circ$ concerns exit conditions and *Analysis*$_\bullet$ concerns entry conditions; the equation system presupposes that $S_\star$ has isolated exits.

The Principle: union versus intersecton

- When $\bigsqcup$ is $\bigcap$ we require the greatest sets that solve the equations and we are able to detect properties satisfied by *all execution paths* reaching (or leaving) the entry (or exit) of a label; the analysis is called a must-analysis.

- When $\bigsqcup$ is $\bigcup$ we require the smallest sets that solve the equations and we are able to detect properties satisfied by *at least one execution path* to (or from) the entry (or exit) of a label; the analysis is called a may-analysis.

Property Spaces

The *property space*, L, is used to represent the data flow information, and the *combination operator*, $\bigsqcup$: $\mathcal{P}(L) \rightarrow L$, is used to combine information from different paths.

- L is a *complete lattice*, that is, a partially ordered set, $(L, \sqsubseteq)$, such that each subset, Y, has a least upper bound, $\bigsqcup Y$.

- L satisfies the *Ascending Chain Condition*; that is, each ascending chain eventually stabilises (meaning that if $(l_n)_n$ is such that $l_1 \sqsubseteq l_2 \sqsubseteq l_3 \sqsubseteq \cdots$,then there exists n such that $l_n = l_{n+1} = \cdots$).

Example: Reaching Definitions

- $L = \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$ is partially ordered by subset inclusion so $\sqsubseteq$ is $\subseteq$
- the least upper bound operation $\bigsqcup$ is $\bigcup$ and the least element $\bot$ is $\emptyset$
- L satisfies the Ascending Chain Condition because $\mathbf{Var}_\star \times \mathbf{Lab}_\star$ is finite (unlike $\mathbf{Var} \times \mathbf{Lab}$)

Example: Available Expressions

- $L = \mathcal{P}(\mathbf{AExp}_\star)$ is partially ordered by superset inclusion so $\sqsubseteq$ is $\supseteq$

- the least upper bound operation $\bigsqcup$ is $\bigcap$ and the least element $\perp$ is $\mathbf{AExp}_\star$

- L satisfies the Ascending Chain Condition because $\mathbf{AExp}_\star$ is finite (unlike $\mathbf{AExp}$)

Transfer Functions

The set of transfer functions, $\mathcal{F}$, is a set of monotone functions over L, meaning that

$$l \sqsubseteq l' \text{ implies } f_\ell(l) \sqsubseteq f_\ell(l')$$

and furthermore they fulfil the following conditions:

- $\mathcal{F}$ contains *all* the transfer functions $f_\ell : L \to L$ in question (for $\ell \in \mathbf{Lab}_\star$)

- $\mathcal{F}$ contains the *identity function*

- $\mathcal{F}$ is *closed under composition* of functions

Frameworks

A *Monotone Framework* consists of:

- a complete lattice, L, that satisfies the Ascending Chain Condition; we write $\bigsqcup$ for the least upper bound operator

- a set $\mathcal{F}$ of monotone functions from L to L that contains the identity function and that is closed under function composition

A *Distributive Framework* is a Monotone Framework where additionally all functions f in $\mathcal{F}$ are required to be distributive:

$$f(l_1 \sqcup l_2) = f(l_1) \sqcup f(l_2)$$

Instances

An *instance* of a Framework consists of:

– the complete lattice, L, of the framework

– the space of functions, $\mathcal{F}$, of the framework

– a finite flow, F (typically *flow*$(S_\star)$ or *flow*$^R(S_\star)$)

– a finite set of *extremal labels*, E (typically $\{$*init*$(S_\star)\}$ or *final*$(S_\star)$)

– an *extremal value*, $\iota \in L$, for the extremal labels

– a mapping, $f_\cdot$, from the labels $\mathbf{Lab}_\star$ to transfer functions in $\mathcal{F}$

The Examples Revisited

	Available Expressions	Reaching Definitions	Very Busy Expressions	Live Variables
L	$\mathcal{P}(\mathbf{AExp}_\star)$	$\mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$	$\mathcal{P}(\mathbf{AExp}_\star)$	$\mathcal{P}(\mathbf{Var}_\star)$
$\sqsubseteq$	$\supseteq$	$\subseteq$	$\supseteq$	$\subseteq$
$\bigsqcup$	$\bigcap$	$\bigcup$	$\bigcap$	$\bigcup$
$\perp$	$\mathbf{AExp}_\star$	$\emptyset$	$\mathbf{AExp}_\star$	$\emptyset$
ι	$\emptyset$	$\{(x,?) \mid x \in \mathit{FV}(S_\star)\}$	$\emptyset$	$\emptyset$
E	$\{\mathit{init}(S_\star)\}$	$\{\mathit{init}(S_\star)\}$	$\mathit{final}(S_\star)$	$\mathit{final}(S_\star)$
F	$\mathit{flow}(S_\star)$	$\mathit{flow}(S_\star)$	$\mathit{flow}^R(S_\star)$	$\mathit{flow}^R(S_\star)$
$\mathcal{F}$	$\{f : L \to L \mid \exists l_k, l_g : f(l) = (l \setminus l_k) \cup l_g\}$			
f_ℓ	$f_\ell(l) = (l \setminus \mathit{kill}(B^\ell)) \cup \mathit{gen}(B^\ell)$ where $B^\ell \in \mathit{blocks}(S_\star)$			

Bit Vector Frameworks

A *Bit Vector Framework* has

- $L = \mathcal{P}(D)$ for D finite
- $\mathcal{F} = \{f \mid \exists l_k, l_g : f(l) = (l \setminus l_k) \cup l_g\}$

Examples:

- Available Expressions
- Live Variables
- Reaching Definitions
- Very Busy Expressions

Lemma: Bit Vector Frameworks are always Distributive Frameworks

Proof

$$
\begin{array}{rclcl}
f(l_1 \sqcup l_2) & = & \left\{\begin{array}{l} f(l_1 \cup l_2) \\ f(l_1 \cap l_2) \end{array}\right. & = & \left\{\begin{array}{l} ((l_1 \cup l_2) \setminus l_k) \cup l_g \\ ((l_1 \cap l_2) \setminus l_k) \cup l_g \end{array}\right. \\
 & = & \left\{\begin{array}{l} ((l_1 \setminus l_k) \cup (l_2 \setminus l_k)) \cup l_g \\ ((l_1 \setminus l_k) \cap (l_2 \setminus l_k)) \cup l_g \end{array}\right. & = & \left\{\begin{array}{l} ((l_1 \setminus l_k) \cup l_g) \cup ((l_2 \setminus l_k) \cup l_g) \\ ((l_1 \setminus l_k) \cup l_g) \cap ((l_2 \setminus l_k) \cup l_g) \end{array}\right. \\
 & = & \left\{\begin{array}{l} f(l_1) \cup f(l_2) \\ f(l_1) \cap f(l_2) \end{array}\right. & = & f(l_1) \sqcup f(l_2)
\end{array}
$$

- $id(l) = (l \setminus \emptyset) \cup \emptyset$
- $f_2(f_1(l)) = (((l \setminus l_k^1) \cup l_g^1) \setminus l_k^2) \cup l_g^2 = (l \setminus (l_k^1 \cup l_k^2)) \cup ((l_g^1 \setminus l_k^2) \cup l_g^2)$
- monotonicity follows from distributivity
- $\mathcal{P}(D)$ satisfies the Ascending Chain Condition because D is finite

The Constant Propagation Framework

An example of a Monotone Framework that is not a Distributive Framework

The aim of the *Constant Propagation Analysis* is to determine

> For each program point, whether or not a variable has a constant value whenever execution reaches that point.

Example:

$$[\mathtt{x}{:=}6]^1; [\mathtt{y}{:=}3]^2; \mathtt{while}\ [\mathtt{x} > \mathtt{y}]^3\ \mathtt{do}\ ([\mathtt{x}{:=}\mathtt{x}-1]^4; [\mathtt{z}{:=}\mathtt{y}*\mathtt{y}]^6)$$

The analysis enables a transformation into

$$[\mathtt{x}{:=}6]^1; [\mathtt{y}{:=}3]^2; \mathtt{while}\ [\mathtt{x} > 3]^3\ \mathtt{do}\ ([\mathtt{x}{:=}\mathtt{x}-1]^4; [\mathtt{z}{:=}9]^6)$$

Elements of L

$$\widehat{\mathbf{State}}_{\mathsf{CP}} = ((\mathbf{Var}_\star \to \mathbf{Z}^\top)_\perp, \sqsubseteq)$$

Idea:

- $\perp$ is the least element: no information is available
- $\widehat{\sigma} \in \mathbf{Var}_\star \to \mathbf{Z}^\top$ specifies for each variable whether it is constant:
  - $\widehat{\sigma}(x) \in \mathbf{Z}$: x is constant and the value is $\widehat{\sigma}(x)$
  - $\widehat{\sigma}(x) = \top$: x might not be constant

Partial Ordering on L

The partial ordering $\sqsubseteq$ on $(\mathbf{Var}_\star \to \mathbf{Z}^\top)_\bot$ is defined by

$$\forall \widehat{\sigma} \in (\mathbf{Var}_\star \to \mathbf{Z}^\top)_\bot : \quad \bot \sqsubseteq \widehat{\sigma}$$

$$\forall \widehat{\sigma}_1, \widehat{\sigma}_2 \in \mathbf{Var}_\star \to \mathbf{Z}^\top : \; \widehat{\sigma}_1 \sqsubseteq \widehat{\sigma}_2 \quad \underline{\text{iff}} \quad \forall x : \widehat{\sigma}_1(x) \sqsubseteq \widehat{\sigma}_2(x)$$

where $\mathbf{Z}^\top = \mathbf{Z} \cup \{\top\}$ is partially ordered as follows:

$$\forall z \in \mathbf{Z}^\top : z \sqsubseteq \top$$

$$\forall z_1, z_2 \in \mathbf{Z} : (z_1 \sqsubseteq z_2) \Leftrightarrow (z_1 = z_2)$$

Transfer Functions in $\mathcal{F}$

$$\mathcal{F}_{\mathsf{CP}} = \{f \mid f \text{ is a monotone function on } \widehat{\mathbf{State}}_{\mathsf{CP}}\}$$

Lemma

Constant Propagation as defined by $\widehat{\mathbf{State}}_{\mathsf{CP}}$ and $\mathcal{F}_{\mathsf{CP}}$ is a Monotone Framework

Instances

Constant Propagation is a forward analysis, so for the program $S_\star$:

- the flow, F, is *flow*$(S_\star)$,

- the extremal labels, E, is $\{$*init*$(S_\star)\}$,

- the extremal value, ι_{CP}, is $\lambda x.\top$, and

- the mapping, $f_\cdot^{\mathsf{CP}}$, of labels to transfer functions is as shown next

Constant Propagation Analysis

$$\begin{array}{c} \mathcal{A}_{\mathsf{CP}} \;:\; \mathbf{AExp} \to (\widehat{\mathbf{State}}_{\mathsf{CP}} \to \mathbf{Z}^{\top}_{\perp}) \\ \hline \end{array}$$

$$\begin{aligned}
\mathcal{A}_{\mathsf{CP}}[\![x]\!]\widehat{\sigma} &= \begin{cases} \perp & \text{if } \widehat{\sigma} = \perp \\ \widehat{\sigma}(x) & \text{otherwise} \end{cases} \\
\mathcal{A}_{\mathsf{CP}}[\![n]\!]\widehat{\sigma} &= \begin{cases} \perp & \text{if } \widehat{\sigma} = \perp \\ n & \text{otherwise} \end{cases} \\
\mathcal{A}_{\mathsf{CP}}[\![a_1 \; op_a \; a_2]\!]\widehat{\sigma} &= \mathcal{A}_{\mathsf{CP}}[\![a_1]\!]\widehat{\sigma} \; \widehat{\mathsf{op}}_a \; \mathcal{A}_{\mathsf{CP}}[\![a_2]\!]\widehat{\sigma}
\end{aligned}$$

$$\begin{array}{c} \text{transfer functions: } f^{\mathsf{CP}}_{\ell} \\ \hline \end{array}$$

$$\begin{array}{lrcl}
[x := a]^{\ell} : & f^{\mathsf{CP}}_{\ell}(\widehat{\sigma}) &=& \begin{cases} \perp & \text{if } \widehat{\sigma} = \perp \\ \widehat{\sigma}[x \mapsto \mathcal{A}_{\mathsf{CP}}[\![a]\!]\widehat{\sigma}] & \text{otherwise} \end{cases} \\
[\mathtt{skip}]^{\ell} : & f^{\mathsf{CP}}_{\ell}(\widehat{\sigma}) &=& \widehat{\sigma} \\
[b]^{\ell} : & f^{\mathsf{CP}}_{\ell}(\widehat{\sigma}) &=& \widehat{\sigma}
\end{array}$$

Lemma

Constant Propagation is not a Distributive Framework

Proof

Consider the transfer function f_ℓ^{CP} for $[\mathtt{y}:=\mathtt{x}*\mathtt{x}]^\ell$

Let $\widehat{\sigma}_1$ and $\widehat{\sigma}_2$ be such that $\widehat{\sigma}_1(\mathtt{x}) = 1$ and $\widehat{\sigma}_2(\mathtt{x}) = -1$

Then $\widehat{\sigma}_1 \sqcup \widehat{\sigma}_2$ maps $\mathtt{x}$ to $\top$ — $f_\ell^{\mathsf{CP}}(\widehat{\sigma}_1 \sqcup \widehat{\sigma}_2)$ maps $\mathtt{y}$ to $\top$

Both $f_\ell^{\mathsf{CP}}(\widehat{\sigma}_1)$ and $f_\ell^{\mathsf{CP}}(\widehat{\sigma}_2)$ map $\mathtt{y}$ to 1 — $f_\ell^{\mathsf{CP}}(\widehat{\sigma}_1) \sqcup f_\ell^{\mathsf{CP}}(\widehat{\sigma}_2)$ maps $\mathtt{y}$ to 1

Equation Solving

- The MFP solution — "Maximum" (actually least) Fixed Point
  - Worklist algorithm for Monotone Frameworks

- The MOP solution — "Meet" (actually join) Over all Paths

The MFP Solution

– Idea: iterate until stabilisation.

Worklist Algorithm

Input: An instance $(L, \mathcal{F}, F, E, \iota, f_\cdot)$ of a Monotone Framework

Output: The MFP Solution: $\mathit{MFP}_\circ, \mathit{MFP}_\bullet$

Data structures:

- Analysis: the current analysis result for block entries (or exits)
- The worklist W: a list of pairs (ℓ, ℓ') indicating that the current analysis result has changed at the entry (or exit) to the block ℓ and hence the entry (or exit) information must be recomputed for ℓ'

Worklist Algorithm

Step 1 Initialisation (of W and Analysis)

W := nil;
for all (ℓ, ℓ') in F do W := cons((ℓ, ℓ'),W);
for all ℓ in F or E do
  if $\ell \in E$ then Analysis[ℓ] := ι else Analysis[ℓ] := $\bot_L$;

Step 2 Iteration (updating W and Analysis)

while W $\neq$ nil do
  ℓ := fst(head(W)); ℓ' = snd(head(W)); W := tail(W);
  if f_ℓ(Analysis[ℓ]) $\not\sqsubseteq$ Analysis[ℓ'] then
    Analysis[ℓ'] := Analysis[ℓ'] $\sqcup$ f_ℓ(Analysis[ℓ]);
    for all ℓ'' with (ℓ', ℓ'') in F do W := cons((ℓ', ℓ''),W);

Step 3 Presenting the result (*MFP*$_\circ$ and *MFP*$_\bullet$)

for all ℓ in F or E do
  MFP$_\circ(\ell)$:= Analysis[ℓ];
  MFP$_\bullet(\ell)$:= f_ℓ(Analysis[ℓ])

Correctness

The worklist algorithm always terminates and it computes the least (or MFP) solution to the instance given as input.

Complexity

Suppose that E and F contain at most $b \geq 1$ distinct labels, that F contains at most $e \geq b$ pairs, and that L has finite height at most $h \geq 1$.

Count as basic operations the applications of f_ℓ, applications of $\sqcup$, or updates of Analysis.

Then there will be at most $O(e \cdot h)$ basic operations.

Example: Reaching Definitions (assuming unique labels):

$O(b^2)$ where b is size of program: $O(h) = O(b)$ and $O(e) = O(b)$.

The MOP Solution

– Idea: propagate analysis information along paths.

Paths

The paths up to but not including ℓ:

$$path_\circ(\ell) = \{[\ell_1, \cdots, \ell_{n-1}] \mid n \geq 1 \wedge \forall i < n : (\ell_i, \ell_{i+1}) \in F \wedge \ell_n = \ell \wedge \ell_1 \in E\}$$

The paths up to and including ℓ:

$$path_\bullet(\ell) = \{[\ell_1, \cdots, \ell_n] \mid n \geq 1 \wedge \forall i < n : (\ell_i, \ell_{i+1}) \in F \wedge \ell_n = \ell \wedge \ell_1 \in E\}$$

Transfer functions for a path $\vec{\ell} = [\ell_1, \cdots, \ell_n]$:

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

The MOP Solution

The solution up to but not including ℓ:

$$MOP_\circ(\ell) = \bigsqcup\{f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in path_\circ(\ell)\}$$

The solution up to and including ℓ:

$$MOP_\bullet(\ell) = \bigsqcup\{f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in path_\bullet(\ell)\}$$

Precision of the MOP versus MFP solutions

The MFP solution safely approximates the MOP solution: $MFP \sqsupseteq MOP$ ("because" $f(x \sqcup y) \sqsupseteq f(x) \sqcup f(y)$ when f is monotone).

For Distributive Frameworks the MFP and MOP solutions are equal: $MFP = MOP$ ("because" $f(x \sqcup y) = f(x) \sqcup f(y)$ when f is distributive).

Lemma

Consider the MFP and MOP solutions to an instance $(L, \mathcal{F},\ F, B, \iota, f_\cdot)$ of a Monotone Framework; then:

$$MFP_\circ \sqsupseteq MOP_\circ \text{ and } MFP_\bullet \sqsupseteq MOP_\bullet$$

If the framework is distributive and if $path_\circ(\ell) \neq \emptyset$ for all ℓ in E and F then:

$$MFP_\circ = MOP_\circ \text{ and } MFP_\bullet = MOP_\bullet$$

Decidability of MOP and MFP

The MFP solution is always computable (meaning that it is decidable) because of the Ascending Chain Condition.

The MOP solution is often uncomputable (meaning that it is undecidable): the existence of a general algorithm for the MOP solution would imply the decidability of the *Modified Post Correspondence Problem*, which is known to be undecidable.

Lemma

The MOP solution for Constant Propagation is undecidable.

Proof: Let $u_1, \cdots, u_n$ and $v_1, \cdots, v_n$ be strings over the alphabet $\{1,\cdots,9\}$; let $|\ u\ |$ denote the length of u; let $[\![u]\!]$ be the natural number denoted.

The Modified Post Correspondence Problem is to determine whether or not $u_{i_1} \cdots u_{i_m} = v_{i_1} \cdots v_{i_n}$ for some sequence $i_1, \cdots, i_m$ with $i_1 = 1$.

$\texttt{x:=}[\![u_1]\!]\texttt{; y:=}[\![v_1]\!]\texttt{;}$
$\texttt{while } [\cdots] \texttt{ do}$
$\quad \texttt{(if } [\cdots] \texttt{ then x:=x * 10}^{|u_1|} \texttt{ + } [\![u_1]\!]\texttt{; y:=y * 10}^{|v_1|} \texttt{ + } [\![v_1]\!] \texttt{ else}$
$\quad \vdots$
$\quad \texttt{if } [\cdots] \texttt{ then x:=x * 10}^{|u_n|} \texttt{ + } [\![u_n]\!]\texttt{; y:=y * 10}^{|v_n|} \texttt{ + } [\![v_n]\!] \texttt{ else skip)}$
$\texttt{[z:=abs((x-y)*(x-y))]}^{\ell}$

Then $\mathit{MOP}_{\bullet}(\ell)$ will map z to 1 if and only if the Modified Post Correspondence Problem has no solution. This is undecidable.