

Optimizing Compilers

Data Flow Analysis Frameworks and Algorithms

Jens Knoop, Markus Schordan, Gergő Barany

Institut für Computersprachen
Technische Universität Wien

Towards a General Framework

- The analyses operate over a **property space** representing the analysis information
 - for bit vector frameworks: $\mathcal{P}(D)$ for finite set D
 - more generally: complete lattice (L, \sqsubseteq)
- The analyses of programs are defined in terms of **transfer functions**
 - for bit vector frameworks: $f_\ell(X) = (X \setminus \text{kill}_\ell) \cup \text{gen}_\ell$
 - more generally: monotone functions $f_\ell : L \rightarrow L$

Property Space

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

- L is a complete lattice meaning that it is a partially ordered set, (L, \sqsubseteq) , such that each subset, Y , has a least upper bound, $\sqcup Y$.
- L satisfies the Ascending Chain Condition meaning that each ascending chain eventually stabilizes: if (l_n) is such that $l_1 \sqsubseteq l_2 \sqsubseteq l_3 \sqsubseteq \dots$, then there exists n such that $l_n = l_{n+1} = \dots$.

Complete Lattice

Let Y be a subset of L . Then

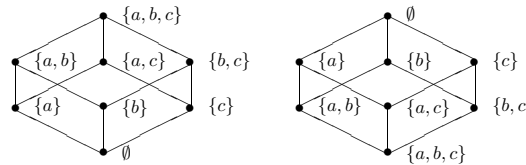
- l is an upper bound of Y if $\forall l' \in Y : l' \sqsubseteq l$ and
- l is a lower bound of Y if $\forall l' \in Y : l \sqsubseteq l'$.
- l is a least upper bound of Y if it is an upper bound of Y that satisfies $l \sqsubseteq l_0$ whenever l_0 is another upper bound of Y .
- l is a greatest lower bound of Y if it is a lower bound of Y that satisfies $l_0 \sqsubseteq l$ whenever l_0 is another lower bound of Y .

A **complete lattice** $L = (L, \sqsubseteq)$ is a partially ordered set (L, \sqsubseteq) such that all subsets have least upper bounds as well as greatest lower bounds.

Notation: $\top = \bigsqcup \emptyset = \bigsqcup L$ is the greatest element of L

$\perp = \bigsqcap \emptyset = \bigsqcap L$ is the least element of L

Example



lattice	$(\mathcal{P}(\{a, b, c\}), \sqsubseteq)$	$(\mathcal{P}(\{a, b, c\}), \supseteq)$
\sqcup	\cup	\cap
\perp	\emptyset	$\{a, b, c\}$

Chain

A subset $Y \subseteq L$ of a partially ordered set $L = (L, \sqsubseteq)$ is a chain if

$$\forall l_1, l_2 \in Y : (l_1 \sqsubseteq l_2) \vee (l_2 \sqsubseteq l_1)$$

It is a finite chain if it is a finite subset of L .

A sequence $(l_n)_n = (l_n)_{n \in \mathbb{N}}$ of elements in L is an

- ascending chain if $n \leq m \rightarrow l_n \sqsubseteq l_m$
- descending chain if $n \leq m \rightarrow l_m \sqsubseteq l_n$

We shall say that a sequence $(l_n)_n$ eventually stabilizes if and only if

$$\exists n_0 \in \mathbb{N} : \forall n \in \mathbb{N} : n \geq n_0 \rightarrow l_n = l_{n_0}$$

Ascending and Descending Chain Conditions

A partially ordered set $L = (L, \sqsubseteq)$ has finite height if and only if all chains are finite.

The partially ordered set L satisfies the

- **Ascending Chain Condition** if and only if all ascending chains eventually stabilize.
- **Descending Chain Condition** if and only if all descending chains eventually stabilize.

Lemma: A partially ordered set $L = (L, \sqsubseteq)$ has finite height if and only if it satisfies both the Ascending and Descending Chain Conditions.

A lattice $L = (L, \sqsubseteq)$ satisfies the ascending chain condition if all ascending chains eventually stabilize; it satisfies the descending chain condition if all descending chains eventually stabilize.

Transfer Functions

The set of transfer functions, \mathcal{F} , is a set of **monotone functions** over $L = (L, \sqsubseteq)$, meaning that

$$l \sqsubseteq l' \rightarrow f_\ell(l) \sqsubseteq f_\ell(l')$$

for all $l, l' \in L$ and furthermore they fulfill the following conditions

- \mathcal{F} contains all the transfer functions $f_\ell : L \rightarrow L$ in question (for $\ell \in \text{Lab}_*$)
- \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 \sqcup 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 of \mathcal{F} are required to be distributive:

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

A **Bit Vector Framework** is a Monotone Framework where additionally is a powerset of a finite set and all functions f of \mathcal{F} have the form

$$f(l) = (l \setminus \text{kill}) \sqcup \text{gen}$$

Instances of a Framework

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 $\text{flow}(S_*)$ or $\text{flow}^R(S_*)$)
- a finite set of extremal labels, E (typically $\{\text{init}(S_*)\}$ or $\{\text{final}(S_*)\}$)
- an extremal value, $\iota \in L$, for the extremal labels
- a mapping, f_\cdot , from the labels Lab_* to transfer functions in \mathcal{F} .

Equations of the Instance

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

$$\text{where } \iota_E^\ell = \begin{cases} \iota & : \text{ if } \ell \in E \\ \perp & : \text{ if } \ell \notin E \end{cases}$$

$$\text{Analysis}_\bullet(\ell) = f_\ell(\text{Analysis}_\circ(\ell))$$

On Bit Vector Frameworks (1)

A **Bit Vector Framework** is a Monotone Framework

- $\mathcal{P}(D)$ is a complete lattice satisfying the Ascending Chain Condition (because D is finite)
- the transfer functions $f_\ell(l) = (l \setminus \text{kill}_\ell) \sqcup \text{gen}_\ell$
 - are monotone: $l_1 \subseteq l_2 \rightarrow l_1 \setminus \text{kill}_\ell \subseteq l_2 \setminus \text{kill}_\ell$
 $\rightarrow (l_1 \setminus \text{kill}_\ell) \sqcup \text{gen}_\ell \subseteq (l_2 \setminus \text{kill}_\ell) \sqcup \text{gen}_\ell$
 $\rightarrow f_\ell(l_1) \subseteq f_\ell(l_2)$
 - contain the identity function: $\text{id}(l) = (l \setminus \emptyset) \sqcup \emptyset$
 - are closed under function composition:

$$\begin{aligned} f_2 \circ f_1 &= f_2(f_1(l)) = ((l \setminus \text{kill}_1^1) \sqcup \text{gen}_1^1) \setminus \text{kill}_2^2 \sqcup \text{gen}_2^2 \\ &= (l \setminus (\text{kill}_1^1 \cup \text{kill}_2^2)) \sqcup ((\text{gen}_1^1 \setminus \text{kill}_2^2) \sqcup \text{gen}_2^2) \end{aligned}$$

On Bit Vector Frameworks (2)

A Bit Vector Framework is a Distributive Framework

- a Bit Vector Framework is a Monotone Framework
- the transfer functions of a Bit Vector Framework are distributive

$$\begin{aligned}
 f(l_1 \sqcup l_2) &= f(l_1 \cup l_2) \\
 &= ((l_1 \cup l_2) \setminus \text{kill}_l) \cup \text{gen}_l \\
 &= ((l_1 \setminus \text{kill}_l) \cup (l_2 \setminus \text{kill}_l)) \cup \text{gen}_l \\
 &= ((l_1 \setminus \text{kill}_l) \cup \text{gen}_l) \cup ((l_2 \setminus \text{kill}_l) \cup \text{gen}_l) \\
 &= f(l_1) \cup f(l_2) = f_l(l_1) \sqcup f_l(l_2)
 \end{aligned}$$

Analogous for the case with \sqcap being \cap .

Note, a Bit Vector Framework is (a special case of) a Distributive Framework. And a Distributive Framework is (a special case of) a Monotone Framework.

Minimal Fixed Point Algorithm (MFP)

Input: an instance $(L, \mathcal{F}, F, E, \iota, f)$ of a Monotone Framework

Output: the MFP Solution: $\text{MFP}_\circ, \text{MFP}_\bullet$.

$$\text{MFP}_\circ(\ell) := A(\ell)$$

$$\text{MFP}_\bullet(\ell) := f_\ell(A(\ell))$$

Data Structures: to represent a work list and the analysis result

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

Lemma: The worklist algorithm always terminates and computes the least (or MFP_\circ) solution to the instance given as input.

^ofor historical reasons MFP is also called maximal fixed point in the literature

Generic Worklist Algorithm

```

W:=nil;
foreach  $(\ell, \ell') \in F$  do W := cons(( $\ell, \ell'$ ), W); od;
foreach  $\ell \in E \cup \{\ell, \ell' \mid (\ell, \ell') \in F\}$  do
  if  $\ell \in E$  then
    A[ $\ell$ ] :=  $\perp_L$ 
  else
    A[ $\ell$ ] :=  $\perp_L$ 
  fi
fi
od
while W  $\neq$  nil do
  ( $\ell, \ell'$ ) := head(W);
  W := tail(W);
  if  $f_\ell(A[\ell]) \not\sqsubseteq A[\ell']$  then
    A[ $\ell'$ ] := A[ $\ell$ ]  $\sqcup$   $f_\ell(A[\ell])$ ;
    foreach  $\ell''$  with  $(\ell', \ell'')$  in F do
      W := cons(( $\ell', \ell''$ ), W);
    od
  fi
od

```

Complexity

Assume that

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

Count as basic operations the application of f_ℓ , applications of \sqcup , or updates of A .

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

Meet Over All Paths Solution (MOP)

Idea: Propagate analysis information along paths to determine the information available at the different program points.

- The paths up to but not including ℓ :
 $path_\circ(\ell) = \{[\ell_1, \dots, \ell_{n-1}] \mid n \geq 1 \wedge \forall i < n : (\ell, \ell') \in F \wedge \ell_1 \in E \wedge \ell_n = \ell\}$
- The paths up to and including ℓ :
 $path_\bullet(\ell) = \{[\ell_1, \dots, \ell_n] \mid n \geq 1 \wedge \forall i < n : (\ell, \ell') \in F \wedge \ell_1 \in E \wedge \ell_n = \ell\}$

With each path $\vec{\ell} = [\ell_1, \dots, \ell_n]$ we associate a transfer function:

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

MOP Solution

- The solution up to but not including ℓ :

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

- The solution up to and including ℓ :

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

MOP vs MFP Solution

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).

Decidability of MOP and MFP solution

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.

– See "Principles of Program Analysis" for more details.

References

- Material for this 4th lecture
www.complang.tuwien.ac.at/knoop/oue185187_ws0809.html
- Book
Flemming Nielson, Hanne Riis Nielson, Chris Hankin:
Principles of Program Analysis.
Springer, (450 pages, ISBN 3-540-65410-0), 1999.
 - Chapter 2 (Data Flow Analysis)
 - and transparencies available at www.imm.dtu.dk/~riis/ppa.htm