Monotone Data Flow Analysis Frameworks

John B. Kam and Jeffrey D. Ullman, March 24th 1975

Fengyun Liu, Ólafur Páll Geirsson

March 31, 2015

Today's agenda

1. Introduction
2. Background
3. Monotone Data Flow Analysis Frameworks
4. Approaches to solving monotone frameworks
5. A Variant of Kildall's Algorithm
6. Undecidability of MOP Problem for monotone frameworks

Overview

2 Background

3 Monotone Data Flow Analysis Frameworks

4 Approaches to solving monotone frameworks

5 A Variant of Kildall's Algorithm

6 Undecidability of MOP Problem for monotone frameworks

Overview

2 Background
- Flow graph
- Semilattice
- Semilattice: ordering
- Semilattice: 0 and 1
- Semilattice: bounded chains

3 Monotone Data Flow Analysis Frameworks

4 Approaches to solving monotone frameworks

5 A Variant of Kildall's Algorithm

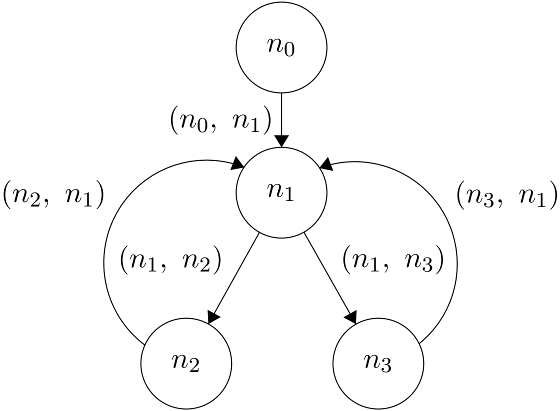
6 Undecidability of MOP Problem for monotone frameworks

Flow graph

Definition

A flow graph is a triple $G = (N, E, n_0)$.

Example:
$N = \{n_0, n_1, n_2, n_3\}$
$E = \{(n_0, n_1),$
$(n_1, n_2), (n_1, n_3),$
$(n_2, n_1), (n_3, n_1)\}$



Semilattice

Definition

A semilattice is a set L with *meet* operation $\wedge$ where the meet operation satisfies the following properties:

$$
\begin{aligned}
a \wedge a &= a && (\textit{idempotent}) \\
a \wedge b &= b \wedge a && (\textit{commutative}) \\
a \wedge (b \wedge c) &= (a \wedge b) \wedge c && (\textit{associative})
\end{aligned}
$$

Semilattice: ordering

Definition

The meet $\wedge$ defines a partial order on L

$$a \geqq b \qquad \text{iff } a \wedge b = b$$
$$a > b = b \wedge a \qquad \text{iff } a \wedge b = b \text{ and } a \neq b$$

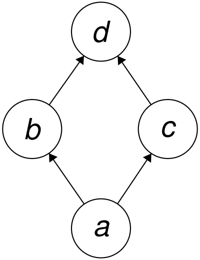


Figure: $d > b, c > a$. However, $b \not> c$ and $c \not> b$.

Definition

Element $e \in L$ is called zero, labeled 0, if

$$e \wedge x = e \qquad \forall x \in L$$

Analogous to $\perp$ from last lecture.

Definition

Element $e \in L$ is called one, labeled 1, if

$$e \wedge x = x \qquad \forall x \in L$$

Analogous to $\top$ from last lecture.

Example

$L = \{\{1,2\}, \{1\}, \{2\}, \{\}\}$
$A \wedge B = A \cap B$
Example 0: $\{\}$
Example 1: $\{1,2\}$

Definition

Element $e \in L$ is called zero, labeled 0, if

$$e \wedge x = e \qquad \forall x \in L$$

Analogous to $\perp$ from last lecture.

Definition

Element $e \in L$ is called one, labeled 1, if

$$e \wedge x = x \qquad \forall x \in L$$

Analogous to $\top$ from last lecture.

Example

$L = \{\{1,2\},\{1\},\{2\},\{\}\}$
$A \wedge B = A \cap B$
Example 0: $\{\}$
Example 1: $\{1,2\}$

Definition

Element $e \in L$ is called zero, labeled 0, if

$$e \wedge x = e \qquad \forall x \in L$$

Analogous to $\perp$ from last lecture.

Definition

Element $e \in L$ is called one, labeled 1, if

$$e \wedge x = x \qquad \forall x \in L$$

Analogous to $\top$ from last lecture.

Example

$L = \{\{1,2\},\{1\},\{2\},\{\}\}$
$A \wedge B = A \cap B$
Example 0: $\{\}$
Example 1: $\{1,2\}$

Definition

A sequence $x_1, x_2, \ldots, x_n$ forms a chain if $x_i > x_{i+1}$ for $1 \leqq i < n$.

Definition

The set L is said to be bounded if for each $x \in L$ there is a constant b_x such that each chain beginning with x has length at most b_x.

Example

$L = \{\{1, 2\}, \{1\}, \{2\}, \{\}\}$
$A \wedge B = A \cap B$
Example chain: $\{1, 2\}, \{1\}, \{\}$
Any finite set is bounded. Can infinite sets be bounded?

Definition

A sequence $x_1, x_2, \ldots, x_n$ forms a chain if $x_i > x_{i+1}$ for $1 \leqq i < n$.

Definition

The set L is said to be bounded if for each $x \in L$ there is a constant b_x such that each chain beginning with x has length at most b_x.

Example

$L = \{\{1, 2\}, \{1\}, \{2\}, \{\}\}$
$A \wedge B = A \cap B$
Example chain: $\{1, 2\}, \{1\}, \{\}$
Any finite set is bounded. Can infinite sets be bounded?

Definition

A sequence $x_1, x_2, \ldots, x_n$ forms a chain if $x_i > x_{i+1}$ for $1 \leqq i < n$.

Definition

The set L is said to be bounded if for each $x \in L$ there is a constant b_x such that each chain beginning with x has length at most b_x.

Example

$L = \{\{1,2\}, \{1\}, \{2\}, \{\}\}$
$A \wedge B = A \cap B$
Example chain: $\{1,2\}, \{1\}, \{\}$
Any finite set is bounded. Can infinite sets be bounded?

Overview

2 Background

3 Monotone Data Flow Analysis Frameworks
- Monotone function space
- Monotone data flow analysis framework
- Constant propagation

4 Approaches to solving monotone frameworks

5 A Variant of Kildall's Algorithm

6 Undecidability of MOP Problem for monotone frameworks

Monotone function space

Definition

Given a bounded semilattice L, a set of functions F on L is said to be a *monotone function space* associated with L if the following conditions are satisfied:

M1 Each $f \in F$ satisfies the monotonicity condition if for all $x, y \in L$,

$$f(x \wedge y) \leqq f(x) \wedge f(y)$$

M2 There exists an identity function i in F.

M3 F is closed under composition.

M4 L is equal to the closure of $\{0\}$ under the meet operation and application of functions in F (more on next slide).

Monotone function space (M4)

Definition

M4 L is equal to the closure of $\{0\}$ under the meet operation and application of functions in F.

- Read: given F, $\{0\}$ and the meet operation, we can generate all elements of L.
- Alternatively: Any element in L can be expressed as a sequence of applications of functions in F and the meet operation with $\{0\}$.

Monotone data flow analysis framework

Definition

A Monotone data flow analysis framework (from here on *framework*) is a triple $D = (L, \wedge, F)$, where

(1) L is a bounded semilattice with meet $\wedge$

(2) F is a monotone function space associated with L

Framework instance

Definition

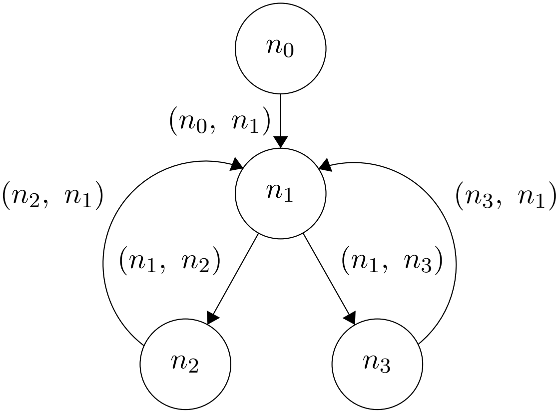
A particular instance of a framework is a pair $I = (G, M)$, where

(1) $G = (N, E, n_0)$ is a flow graph.

(2) $M : N \rightarrow F$ is a function which maps each node in N to a function in F.

Putting it together

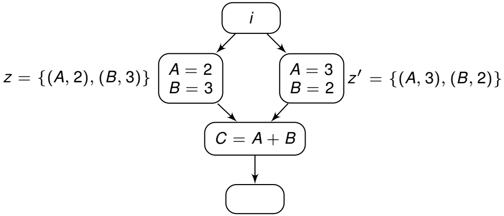
- Instance $I = (G, M)$
- Framework $D = (L, \wedge, F)$
- $M : N \rightarrow F$
- $\forall f \in F, f : L \rightarrow L$
- $\wedge : L \times L \rightarrow L$



Constant propagation

Constant propagation can be formalized as a monotone data flow analysis framework where

$$\begin{aligned} CONST &= (L, \wedge, F) && \text{framework} \\ V &= \{A_1, A_2, \ldots\} && \text{infinite set of variables} \\ L &\subset 2^{V \times \mathbb{R}} && \text{possible variable assignments} \\ \wedge &= \text{set intersection} \end{aligned}$$



Constant propagation is not distributive

Definition

For all $x, y \in L$ and $f \in F$, distributivity is satisfied when

$$f(x \wedge y) = f(x) \wedge f(y)$$

- $f(x \wedge y) = \{(A,2),(B,3)\} \cap \{(A,3),(B,2)\} = \emptyset$
- $f(x) \wedge f(y) = \{(A,2),(B,3),(C,5)\} \cap \{(A,3),(B,2),(C,5)\} = \{(C,5)\}$

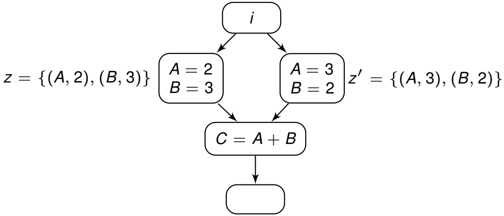


Figure: Counter example to distributivity of CONST

Overview

2 Background

3 Monotone Data Flow Analysis Frameworks

4 Approaches to solving monotone frameworks
- Algorithm 1 (Kildall's Algorithm)
- Properties of Algorithm 1

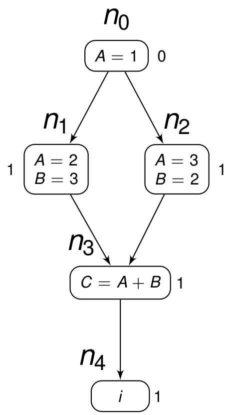
5 A Variant of Kildall's Algorithm

6 Undecidability of MOP Problem for monotone frameworks

Initialization

$$A[n] = \begin{cases} 0 & \text{if } n = n_0 \\ 1 & \textit{otherwise} \end{cases}$$

- 0 - zero element of the lattice
- 1 - one element of the lattice
- i - identity function
- $L \subset 2^{V \times \mathbb{R}}$ where $V = \{A, B, C\}$



Iteration Step

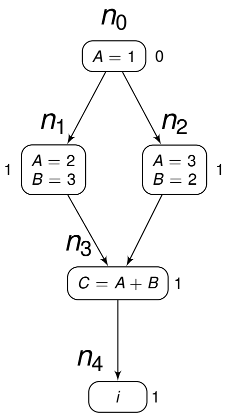
Visit nodes other than n_0 in order $v_1, v_2, ...$, for each visited node n, set

$$A[n] = \bigwedge_{p \in PRED(n)} f_p(A[p])$$

Conditions of the sequence$(v_1, v_2, ...)$:

- If a node doesn't satisfy the equation above, it should be visited again.
- If all nodes satisfy the equation, the sequence eventually end.

Example: n_2, n_1, n_3, n_2, n_3



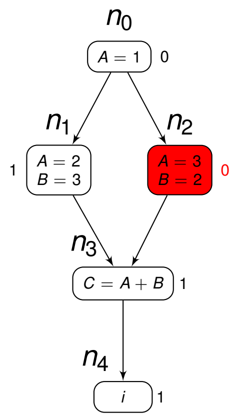
Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, ...$, for each visited node n, set

$$A[n] = \bigwedge_{p \in PRED(n)} f_p(A[p])$$

Conditions of the sequence$(v_1, v_2, ...)$:

- If a node doesn't satisfy the equation above, it should be visited again.
- If all nodes satisfy the equation, the sequence eventually end.



n_2, n_1, n_3, n_2, n_4

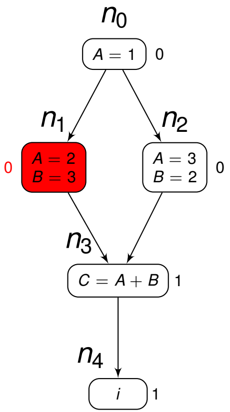
Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, ...$, for each visited node n, set

$$A[n] = \bigwedge_{p \in PRED(n)} f_p(A[p])$$

Conditions of the sequence$(v_1, v_2, ...)$:

- If a node doesn't satisfy the equation above, it should be visited again.
- If all nodes satisfy the equation, the sequence eventually end.



n_2, n_1, n_3, n_2, n_4

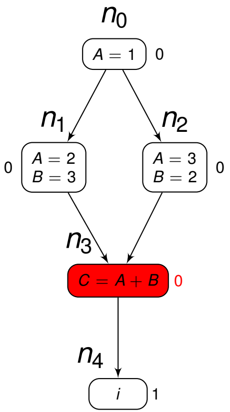
Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, ...$, for each visited node n, set

$$A[n] = \bigwedge_{p \in PRED(n)} f_p(A[p])$$

Conditions of the sequence$(v_1, v_2, ...)$:

- If a node doesn't satisfy the equation above, it should be visited again.
- If all nodes satisfy the equation, the sequence eventually end.



n_2, n_1, n_3, n_2, n_4

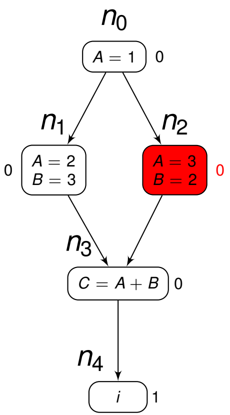
Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, ...$, for each visited node n, set

$$A[n] = \bigwedge_{p \in PRED(n)} f_p(A[p])$$

Conditions of the sequence$(v_1, v_2, ...)$:

- If a node doesn't satisfy the equation above, it should be visited again.
- If all nodes satisfy the equation, the sequence eventually end.



n_2, n_1, n_3, n_2, n_4

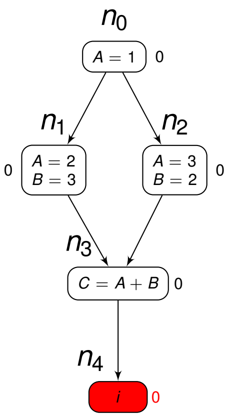
Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, ...$, for each visited node n, set

$$A[n] = \bigwedge_{p \in PRED(n)} f_p(A[p])$$

Conditions of the sequence($v_1, v_2, ...$):

- If a node doesn't satisfy the equation above, it should be visited again.
- If all nodes satisfy the equation, the sequence eventually end.



n_2, n_1, n_3, n_2, n_4

Properties of Algorithm 1

Theorem

Given an instance $I = (G, M)$ of a framework $D = (L, \wedge, F)$ as input to Algorithm 1:

(i) *Algorithm 1 will eventually halt.* *proof: A[n] decrease and L is bounded.*

(ii) $(\forall n \in N) A[n] \leq \bigwedge_{P \in PATH(n)} f_P(0)$. *proof: Induction on the length of P.*

(iii) *The result we get is unique independent of the order in which the nodes are visited.* *proof: A[n] is the maximal fix point of the set of equations.*

(iv) *There're cases where A[n] is strictly less than MOP when the data flow graph is monotone.* *proof: illustrated in the example.*

Properties of Algorithm 1

Theorem

Given an instance $I = (G, M)$ of a framework $D = (L, \wedge, F)$ as input to Algorithm 1:

(i) *Algorithm 1 will eventually halt.* proof: A[n] decrease and L is bounded.

(ii) $(\forall n \in N) A[n] \leqq \bigwedge_{P \in PATH(n)} f_P(0)$. proof: Induction on the length of P.

(iii) *The result we get is unique independent of the order in which the nodes are visited.* proof: A[n] is the maximal fix point of the set of equations.

(iv) *There're cases where A[n] is strictly less than MOP when the data flow graph is monotone.* proof: illustrated in the example.

Properties of Algorithm 1

Theorem

Given an instance $I = (G, M)$ of a framework $D = (L, \wedge, F)$ as input to Algorithm 1:

(i) *Algorithm 1 will eventually halt.* *proof: A[n] decrease and L is bounded.*

(ii) $(\forall n \in N) A[n] \leq \bigwedge_{P \in PATH(n)} f_P(0)$. *proof: Induction on the length of P.*

(iii) *The result we get is unique independent of the order in which the nodes are visited.* *proof: A[n] is the maximal fix point of the set of equations.*

(iv) *There're cases where A[n] is strictly less than MOP when the data flow graph is monotone.* *proof: illustrated in the example.*

Properties of Algorithm 1

Theorem

Given an instance $I = (G, M)$ of a framework $D = (L, \wedge, F)$ as input to Algorithm 1:

(i) *Algorithm 1 will eventually halt.* proof: A[n] decrease and L is bounded.

(ii) $(\forall n \in N) A[n] \leq \bigwedge_{P \in PATH(n)} f_P(0)$. proof: Induction on the length of P.

(iii) *The result we get is unique independent of the order in which the nodes are visited.* proof: A[n] is the maximal fix point of the set of equations.

(iv) *There're cases where A[n] is strictly less than MOP when the data flow graph is monotone.* proof: illustrated in the example.

Overview

2 Background

3 Monotone Data Flow Analysis Frameworks

4 Approaches to solving monotone frameworks

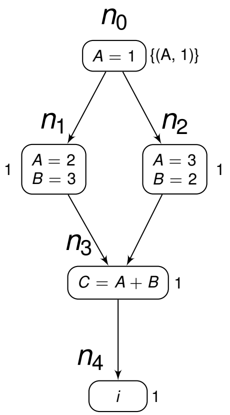
5 A Variant of Kildall's Algorithm
- Algorithm 2
- Properties of Algorithm 2
- Algorithm 2 is not Universal

6 Undecidability of MOP Problem for monotone frameworks

Initialization

$$B[n] = \begin{cases} f_{n_0}(0) & \text{if } n = n_0 \\ 1 & \textit{otherwise} \end{cases}$$

- 0 - zero element of the lattice
- 1 - one element of the lattice
- i - identity function
- f_{n_0} - the function that n_0 corresponds to
- $L \subset 2^{V \times \mathbb{R}}$ where $V = \{A, B, C\}$

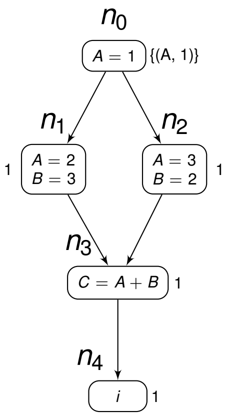


Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, ...$, for each visited node n, set

$$B[n] = \bigwedge_{p \in PRED(n)} f_n(B[p])$$

Same conditions for the sequence as in Algorithm 1.

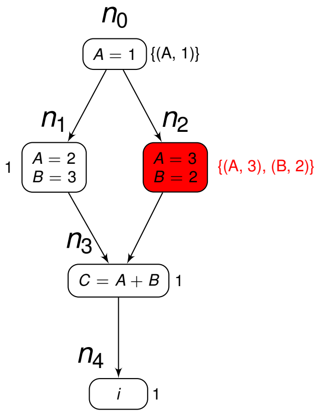


Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, \ldots$, for each visited node n, set

$$B[n] = \bigwedge_{p \in PRED(n)} f_n(B[p])$$

Same conditions for the sequence as in Algorithm 1.



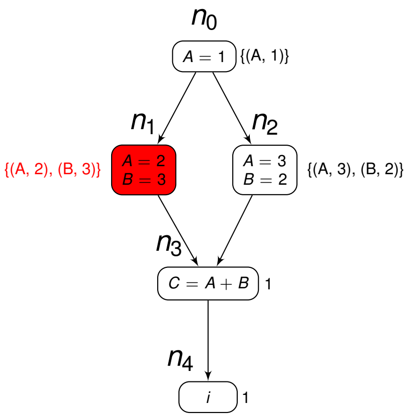
n_2, n_1, n_3, n_2, n_4

Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, ...,$ for each visited node n, set

$$B[n] = \bigwedge_{p \in PRED(n)} f_n(B[p])$$

Same conditions for the sequence as in Algorithm 1.



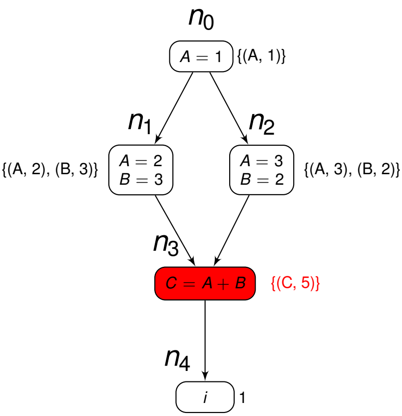
n_2, n_1, n_3, n_2, n_4

Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, ...,$ for each visited node n, set

$$B[n] = \bigwedge_{p \in PRED(n)} f_n(B[p])$$

Same conditions for the sequence as in Algorithm 1.



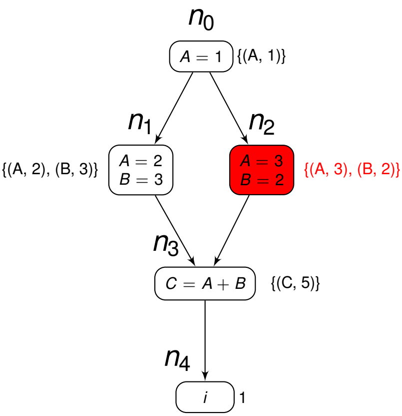
n_2, n_1, n_3, n_2, n_4

Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, ...$, for each visited node n, set

$$B[n] = \bigwedge_{p \in PRED(n)} f_n(B[p])$$

Same conditions for the sequence as in Algorithm 1.



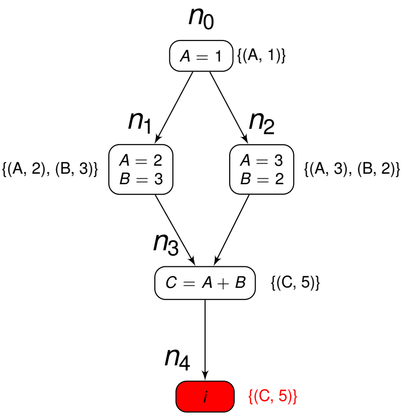
n_2, n_1, n_3, n_2, n_4

Iteration Step

Visit nodes other than n_0 in order $v_1, v_2, \ldots$, for each visited node n, set

$$B[n] = \bigwedge_{p \in PRED(n)} f_{\color{red}{n}}(B[p])$$

Same conditions for the sequence as in Algorithm 1.

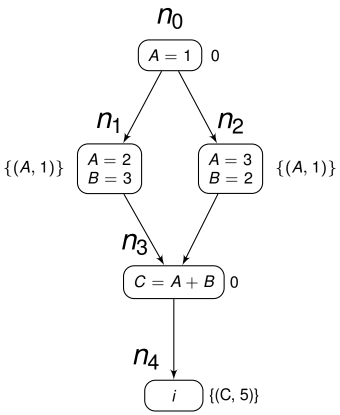


$n_2, n_1, n_3, n_2, {\color{red}n_4}$

Final Step

For each node, set

$$H[n] = \begin{cases} 0 & \text{if } n = n_0 \\ \bigwedge_{p \in PRED(n)} B[p] & \textit{otherwise} \end{cases}$$



Properties of Algorithm 2

Theorem

Given an instance $I = (G, M)$ of a framework $D = (L, \wedge, F)$ as input to Algorithm 2:

(i) *Algorithm 2 will eventually halt.* *proof: B[n] decrease and L is bounded.*

(ii) *The result we get is unique independent of the order in which the nodes are visited.* *proof: B[n] is the maximal fix point of the set of equations.*

(iii) $(\forall n \in N) H[n] \leq \bigwedge_{P \in PATH(n)} f_P(0)$. *proof: Induction on the length of P.*

(iv) *If A[n] is the result of applying Algorithm 1 to $I = (G, M)$, then $A[n] \leq H[n]$.*

Properties of Algorithm 2

Theorem

Given an instance $I = (G, M)$ of a framework $D = (L, \wedge, F)$ as input to Algorithm 2:

(i) *Algorithm 2 will eventually halt.* proof: B[n] decrease and L is bounded.

(ii) *The result we get is unique independent of the order in which the nodes are visited.* proof: B[n] is the maximal fix point of the set of equations.

(iii) $(\forall n \in N) H[n] \leq \bigwedge_{P \in PATH(n)} f_P(0)$. proof: Induction on the length of P.

(iv) *If A[n] is the result of applying Algorithm 1 to $I = (G, M)$, then $A[n] \leq H[n]$.*

Properties of Algorithm 2

Theorem

Given an instance $I = (G, M)$ of a framework $D = (L, \wedge, F)$ as input to Algorithm 2:

(i) *Algorithm 2 will eventually halt.* proof: B[n] decrease and L is bounded.

(ii) *The result we get is unique independent of the order in which the nodes are visited.* proof: B[n] is the maximal fix point of the set of equations.

(iii) $(\forall n \in N) H[n] \leqq \bigwedge_{P \in PATH(n)} f_P(0)$. proof: Induction on the length of P.

(iv) *If A[n] is the result of applying Algorithm 1 to $I = (G, M)$, then $A[n] \leqq H[n]$.*

Properties of Algorithm 2

Theorem

Given an instance $I = (G, M)$ of a framework $D = (L, \wedge, F)$ as input to Algorithm 2:

(i) *Algorithm 2 will eventually halt.* *proof: B[n] decrease and L is bounded.*

(ii) *The result we get is unique independent of the order in which the nodes are visited.* *proof: B[n] is the maximal fix point of the set of equations.*

(iii) $(\forall n \in N) H[n] \leqq \bigwedge_{P \in PATH(n)} f_P(0)$. *proof: Induction on the length of P.*

(iv) *If A[n] is the result of applying Algorithm 1 to $I = (G, M)$, then $A[n] \leqq H[n]$.*

Properties of Algorithm 2

Theorem

Given an instance $I = (G, M)$ of a framework $D = (L, \wedge, F)$ as input to Algorithm 2:

(i) *Algorithm 2 will eventually halt.* *proof: B[n] decrease and L is bounded.*

(ii) *The result we get is unique independent of the order in which the nodes are visited.* *proof: B[n] is the maximal fix point of the set of equations.*

(iii) $(\forall n \in N) H[n] \leqq \bigwedge_{P \in PATH(n)} f_P(0)$. *proof: Induction on the length of P.*

(iv) *If A[n] is the result of applying Algorithm 1 to $I = (G, M)$, then $A[n] \leqq H[n]$.* *Let's prove this!*

Property (iv) of Algorithm 2 - Proof

Proposition (*iv*)

If A[n] is the result of applying Algorithm 1 to $I = (G, M)$, then $A[n] \leqq H[n]$.

Proof by induction on steps of algorithm 2

For node n_0, the property trivially holds because $A[n_0] = H[n_0] = 0$.
For all other node $n \in N$, $A[n] \leqq H[n]$ is equivalent to: $f_n(A[n]) \leqq B[n]$ because

$$A[n] = \bigwedge_{p \in PRED(n)} f_p(A[p])$$

$$H[n] = \bigwedge_{p \in PRED(n)} B[p]$$

Base step ($m = 0$). $B^0[n] = 1$ and by definition $f_n(A[n]) \leqq 1$

Proof by induction on steps of algorithm 2 (cont.)

Induction step ($m > 0$). Our induction hypothesis (IH) is:

$$B^{m-1}[n] \geqq f_n(A[n])$$

Then

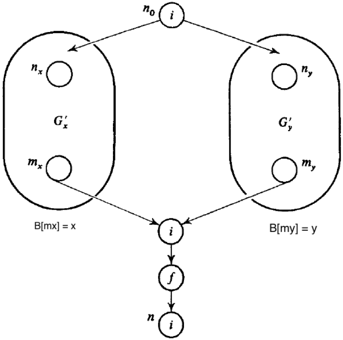
$$\begin{aligned} B^m[n] &= \bigwedge_{p \in PRED(n)} f_n(B^{m-1}[p]) \\ &\geqq \bigwedge_{p \in PRED(n)} f_n(f_p(A[p])) && \text{(IH)} \\ &\geqq f_n\Big(\bigwedge_{p \in PRED(n)} f_p(A[p])\Big) && \text{(monotonicity)} \\ &= f_n(A[n]) \end{aligned}$$

$\square$

Algorithm 2 is not Universal

An infinite number of counterexamples exist. Consider the following constructed instance of a non-distributive monotone framework:

$$MOP(n) = \bigwedge_{P \in PATH(n)} f_P(0) \geqq f(x) \wedge f(y) > f(x \wedge y) = H[n].$$



Overview

2. Background
3. Monotone Data Flow Analysis Frameworks
4. Approaches to solving monotone frameworks
5. A Variant of Kildall's Algorithm
6. Undecidability of MOP Problem for monotone frameworks
   - Previous result
   - Strengthened result

Previous result

Computing the MOP solution for a monotone framework is NP-hard.

- Dana Charmian Angluin[1]

[1] Missing reference

Strengthened result

Theorem

Given arbitrary instance $I = (G, M)$ of an arbitrary monotone framework $D = (L, \wedge, F)$, there does not exist an algorithm which computes $\bigwedge_{P \in PATH(n)} f_P(0)$ for all nodes $n \in G$.

Intuition

The *Modified Post Correspondence Problem* (MPCP), which is well known to be undecidable, can be reduced to the *MOP problem*. This is accomplished by modeling the MPCP problem as a monotone data flow framework.

MPCP example

Definition (Modified Post Correspondence Problem)

Given arbitrary lists A and B, of k strings each in $\{0,1\}^+$, say

$$A = w_1, w_2, \ldots, w_k \quad B = z_1, z_2, \ldots, z_k$$

does there exist a sequence of integers $i_1, i_2, \ldots, i_r$ such at

$$w_1 w_{i_1} w_{i_2} \ldots w_{i_r} = z_1 z_{i_1} z_{i_2} \ldots z_{i_r}$$

This example has a solution: 3, 2, 2, 4.

$$w_1 w_3 w_2 w_2 w_4 = 11{,}0111{,}1{,}1{,}10 = 1101111110$$
$$x_1 x_3 x_2 x_2 x_4 = 1{,}10{,}111{,}111{,}0 = 1101111110$$

i	w_i	x_i
1	11	1
2	1	111
3	0111	10
4	10	0

Reducing MPCP to MOP

Define

- L = 0, \$, and all strings of integers beginning with 1
- $x \wedge y = 0$ if $x \neq y$
- $h(x) = $ "1"
- $f_i(0) = 0, f_i(\$) = \$,$ $f_i(x) = concat(x, i)$
- $g(0) = 0, g(\$) = \$$
- $g(x) = 0$ if x is a solution
- $g(x) = \$$ otherwise

We have $MOP(*) = 0$ if there's a solution, otherwise $MOP(*) = \$$.

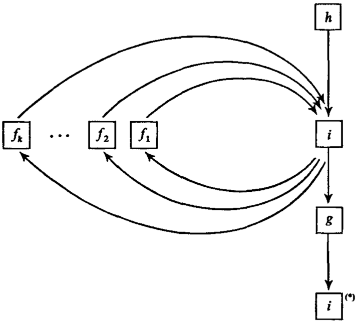


Figure: The solution to a MPCP would be one path from n_0 to n