Logic-Flow Analysis of Higher-Order Programs

Matt Might

http://matt.might.net/

POPL 2007
Why?

Tim Sweeney, POPL 2006
Static array-bounds checking.

Example

... a[i] ...

Will 0 \leq i < a.length always hold?
Why?

Tim Sweeney, POPL 2006
Static array-bounds checking.

Example

... a[i] ...

Will $0 \leq i < a\.length$ always hold?

Wait...
Hasn’t this been done already? (Range analysis, etc.)
Why?

Tim Sweeney, POPL 2006
Static array-bounds checking.

Example

... a[i] ...

Will $0 \leq i < a.length$ always hold?

Wait...
Hasn’t this been done already? (Range analysis, etc.)
Yeah, but not for permutation/vertex array code.
Why?

Tim Sweeney, POPL 2006
Static array-bounds checking.

Example

... a[i] ...

Will $0 \leq i < a.length$ always hold?

Wait...

Hasn’t this been done already? (Range analysis, etc.)
Yeah, but not for permutation/vertex array code.

Not the big idea

LFA is not about array-bounds checking.
The Idea

Theorem Proving   Flow Analysis
The Idea

Theorem Proving → Flow Analysis
The Idea

Theorem Proving → Flow Analysis
How?

Abstract interpretation

Mechanical (flow): \[ \hat{\varsigma} \rightarrow \hat{\varsigma}' \rightarrow \hat{\varsigma}'' \rightarrow \cdots \]

Propositional (logic): \[ \Pi \rightarrow \Pi' \rightarrow \Pi'' \rightarrow \cdots \]
How?

Abstract interpretation

Mechanical (flow):

Propositional (logic):
Higher-order flow analysis fails

Example

... a[i] ...

Will $0 \leq i < a.length$ always hold?
Higher-order flow analysis fails

Example

... a[i] ...

Will $0 \leq i < a$.length always hold?

Flow analysis results

- a is an array.
Higher-order flow analysis fails

Example

... a[i] ...

Will $0 \leq i < a.length$ always hold?

Flow analysis results

- a is an array from line 10 or line 30.
Higher-order flow analysis fails

Example

... a[i] ...

Will $0 \leq i < a.length$ always hold?

Flow analysis results

- a is an array from line 10 or line 30.
- i is an integer.
Higher-order flow analysis fails

Example

... a[i] ... 

Will $0 \leq i < a.length$ always hold?

Flow analysis results

- a is an array from line 10 or line 30.
- i is a non-negative integer.
Higher-order flow analysis fails

Example

... a[i] ...

Will $0 \leq i < a.length$ always hold?

Flow analysis results

- a is an array from line 10 or line 30.
- i is a non-negative integer.
- a.length is a positive integer.
Higher-order flow analysis fails

Example

... a[i] ...

Will $0 \leq i < a\.length$ always hold?

Flow analysis results

- $a$ is an array from line 10 or line 30.
- $i$ is a non-negative integer.
- $a\.length$ is a positive integer.

Insufficiently rich information

Frequently can’t show $i < a\.length$. 
First attempt: Enrich flow values with relations

Example

... a[i] ...

Will $0 \leq i < a.length$ always hold?
First attempt: Enrich flow values with relations

Example

\[ \ldots a[i] \ldots \]

Will \( 0 \leq i < a.length \) always hold?

Hypothetical results

- \( a \) is an array from line 10 or line 30.
- \( i \) is a non-negative integer.
- \( a.length \) is a positive integer.
First attempt: Enrich flow values with relations

Example

... a[i] ...

Will $0 \leq i < a.length$ always hold?

Hypothetical results

- $a$ is an array from line 10 or line 30.
- $i$ is in $\{x: 0 \leq x < a.length\}$.
- $a.length$ is a positive integer.
First attempt: Enrich flow values with relations

Example

\[
\ldots \ a[i] \ \ldots \\
\]

Will \( 0 \leq i < a.\text{length} \) always hold?

Hypothetical results

- \( a \) is an array from line 10 or line 30.
- \( i \) is in \( \{x : 0 \leq x < a.\text{length}\} \).
- \( a.\text{length} \) is a positive integer.

Problems

1. What does \( a.\text{length} \) mean where \( a \) is out of scope?
2. How did this set get there in the first place?
First attempt: Enrich flow values with relations

Example

... a[i] ...

Will \(0 \leq i < a.length\) always hold?

Hypothetical results

- a is an array from line 10 or line 30.
- i is in \(\{x: 0 \leq x < a.length\}\).
- a.length is a positive integer.

Problems

1. What does a.length mean where a is out of scope?
2. How did this set get there in the first place?
Problem 1: Meaning of variables

Ambiguity problem

- Need environment-independent identities for values.
Problem 1: Meaning of variables

Ambiguity problem

► Need environment-independent identities for values.

Solutions

► Constants. *E.g.* 5 means 5 anywhere.
Problem 1: Meaning of variables

Ambiguity problem

- Need environment-independent \textbf{identities} for values.

Solutions

- Constants. \textit{E.g.} 5 means 5 anywhere.
- Heap locations. \textit{E.g.} Heap location 10 offset 3.
Problem 1: Meaning of variables

Ambiguity problem

- Need environment-independent identities for values.

Solutions

- Constants. *E.g.* 5 means 5 anywhere.
- Heap locations. *E.g.* Heap location 10 offset 3.
- Bindings. [Shivers 1988]
Bindins

Definition
A **binding** is a variable-time pairing, e.g. \((x, 3)\).
Bindings

Definition
A binding is a variable-time pairing, e.g. \((x, 3)\).

Example (Variable v. binding)
- Value of variable \(x\) depends on environment.
- Value of \(x\) bound at time 3: Same in any environment/state.
Definitions

A **binding** is a variable-time pairing, *e.g.* \((x, 3)\).

Example (Variable v. binding)
- Value of variable \(x\) depends on environment.
- Value of \(x\) **bound at time 3**: Same in any environment/state.

Example (Abstract bindings)
- 0CFA: All values of \(x\) bound at any time.
- 1CFA: All values of \(x\) bound at a time while calling \(foo\).
Build binding-sensitive flow analysis.
Strategy

- Build binding-sensitive flow analysis.
- Build binding-sensitive logic.
Strategy

- Build binding-sensitive flow analysis.
- Build binding-sensitive logic.
- Build binding-sensitive propositional abstract interpretation.
Strategy

- Build binding-sensitive flow analysis.
- Build binding-sensitive logic.
- Build binding-sensitive propositional abstract interpretation.
- Weave.
Tool 1: Continuation-passing style (CPS)

Contract

- Calls never return.
- Continuations are passed to receive the result.

Example

Direct-style identity function:

\[
\text{(define (id x) x)}
\]

Example

CPS identity function:

\[
\text{(define (id x return) (return x))}
\]
CPS simplifies

In CPS,
  fun call,
  fun return,
  conditional branch,
  sequencing,
  iteration,
  exception throw,
  coroutine switch,
  continuation invocation...
CPS simplifies

In CPS,
fun call,
fun return,
conditional branch,
sequencing,
iteration,
exception throw,
coroutine switch,
continuation invocation...

...all become call to \( \lambda \)
CPS simplifies

In CPS,
  fun call,
  fun return,
  conditional branch,
  sequencing,
  iteration,
  exception throw,
  coroutine switch,
  continuation invocation...

...all become call to $\lambda$

Machine state without CPS

$$\varsigma \in State = CALL \times Env \times Store \times Stack \times Time$$
CPS simplifies

In CPS,
fun call,
fun return,
conditional branch,
sequencing,
itration,
exception throw,
coroutine switch,
continuation invocation...

...all become call to $\lambda$

Machine state without CPS

$$\varsigma \in \text{State} = \text{CALL} \times \text{Env} \times \text{Store} \times \text{Stack} \times \text{Time}$$
CPS simplifies

In CPS,
  fun call,
  fun return,
  conditional branch,
  sequencing,
  iteration,
  exception throw,
  coroutine switch,
  continuation invocation...

...all become call to $\lambda$

Machine state with CPS

$$\varsigma \in State = CALL \times Env \times Store \times Time$$
Tool 2: Machine-based abstract interpretation

Idea
Abstract machine states component-wise.

Machine state

- A call site.
- An environment for variable lookup.
- A heap.
- A time.

More formally

\[ \varsigma \in State = \text{CALL} \times \text{Env} \times \text{Store} \times \text{Time} \]
Tool 2: Machine-based abstract interpretation

Idea
Abstract machine states component-wise.

Abstract machine state
- An abstract call site.
- An abstract environment for variable lookup.
- An abstract heap.
- An abstract time.

More formally

\[ \hat{\varsigma} \in \widehat{\text{State}} = \widehat{\text{CALL}} \times \widehat{\text{Env}} \times \widehat{\text{Store}} \times \widehat{\text{Time}} \]
Tool 2: Machine-based abstract interpretation

Idea
Abstract machine states component-wise.

Abstract machine state
- An abstract call site.
- An abstract environment for variable lookup.
- An abstract heap.
- An abstract time.

More formally

$$\hat{\varsigma} \in \hat{\text{State}} = \hat{\text{CALL}} \times \hat{\text{Env}} \times \hat{\text{Store}} \times \hat{\text{Time}}$$
Binding-factored environment

Definition
An environment maps variables to values.
Definition
An environment maps variables to values.

Definition
A binding-factored environment \((\beta, ve)\) [Shivers 1988] maps:

- variables to their binding times. \((\beta)\)
- and then, variables plus times (bindings) to values. \((ve)\)
Tool 3: Restricted first-order logic for states

Features

- Propositions are facts about concrete machine states.
- Ground terms are identities (bindings, locations, constants).
Tool 3: Restricted first-order logic for states

Features

- Propositions are facts about concrete machine states.
- Ground terms are identities (bindings, locations, constants).

Restrictions

- No existential quantifiers.
- Only outer-level universal quantifiers.
- Quantifiers range over abstract identities.
Example: Proposition

Example

“Every value of \( x \) bound while calling \( \text{foo} \) is less than the length of every array bound to \( a \).”
Example: Proposition

Example

“Every value of $x$ bound while calling $\text{foo}$ is less than the length of every array bound to $a$.”

In longhand:

$$(\forall x : (x, \hat{t}_\text{foo}) \quad (\forall a : (a, \top) \quad (< x (\text{alen} a)))$$
Example: Proposition

Example

“Every value of $x$ bound while calling $\text{foo}$ is less than the length of every array bound to $a$.”

In longhand:

\[
(\forall x : (x, \hat{t}_\text{foo}) \\
(\forall a : (a, \top) \\
\quad (< x (\text{alen } a))))
\]

Or, in convenient (but incomplete) shorthand:

\[
(< (x, \hat{t}_\text{foo}) (\text{alen } (a, \top)))
\]
Logic syntax

Features

- S-Expressions.
- Just or, not.
- Relations encoded as functions.
Logic semantics

Question
How do we know when proposition $\phi$ is true for state $\varsigma$?
Logic semantics

Question
How do we know when proposition $\phi$ is true for state $\varsigma$?

Answer
When $\varsigma \models \phi$ holds.
Logic semantics

Question
How do we know when proposition $\phi$ is true for state $\varsigma$?

Answer
When $\varsigma \models \phi$ holds.

Means exactly what you think it means.
Filtered concretization

Set of concrete states \((\text{State})\)
Filtered concretization

Set of concrete states \((\text{State})\)

\[ \{ \varsigma : |\varsigma| \sqsubseteq \widehat{\varsigma} \} \]
Filtered concretization

Set of concrete states \((State)\) \(\{\varsigma : \varsigma \models \Pi\}\)
Filtered concretization

Set of concrete states \((\text{State})\)

\[\{\varsigma : |\varsigma| \subseteq \hat{\varsigma} \text{ and } \varsigma \models \Pi\}\]
Deriving new propositions

Example
If $\varsigma \models (= x y)$
and $\varsigma \models (= y z)$,
does $\varsigma \models (= x z)$ hold?
Deriving new propositions

Example
If $\varsigma \vdash (= x y)$
and $\varsigma \vdash (= y z)$,
does $\varsigma \vdash (= x z)$ hold?

Answer
Yes, if $\{ (= x y), (= y z) \} \vdash (= x z)$ holds.
Deriving new propositions

Example

If $\varsigma \models (\equiv x y)$
and $\varsigma \models (\equiv y z)$,
does $\varsigma \models (\equiv x z)$ hold?

Answer

Yes, if $\{ (\equiv x y), (\equiv y z) \} \vdash (\equiv x z)$ holds.

\[ \begin{array}{l}
(\text{Assm}) \quad \frac{\psi \in \Pi}{\Pi \vdash \psi} \\
(\lor \text{Ant}) \quad \frac{\Pi \vdash \phi \quad \Pi \subseteq \Pi'}{\Pi' \vdash \phi} \\
(\text{Ant}) \quad \frac{\Pi \vdash \phi}{\Pi \vdash \phi} \\
(\lor \text{Cases}) \quad \frac{\Pi \cup \{ \phi_1 \} \vdash \phi \quad \Pi \cup \{ (\lor \phi_1 \phi_2) \} \vdash \phi_3}{\Pi \cup \{ \phi_2 \} \vdash \phi_3} \\
(\lor \text{Cons}) \quad \frac{\Pi \vdash \phi_1}{\Pi \vdash (\lor \phi_1 \phi_2), (\lor \phi_2 \phi_1)} \\
(\forall \text{Intro}) \quad \frac{\Pi \vdash \psi \quad x \not\in \text{free}(\psi)}{\Pi \vdash (\forall x : i \psi)} \\
(\forall \text{Swap}) \quad \frac{\Pi \vdash (\forall x_1, x_2 : i_1 \psi)}{\Pi \vdash (\forall x_2, x_1 : i_2 \psi)} \\
\end{array} \]
T rusting the theorem prover

Summary

- $\models$: What a proposition means.
- $\vdash$: What a proposition implies.
T rusting the theorem prover

Summary

◮ $\models$: What a proposition means.
◮ $\vdash$: What a proposition implies.

Question

How can we trust an external theorem prover?
Trusting the theorem prover

Summary

- $\models$ : What a proposition means.
- $\vdash$ : What a proposition implies.

Question

How can we trust an external theorem prover?

Theorem (Syntactic soundness)

If $\Pi \vdash \phi$ holds, then $\Pi \models \phi$ holds.
All together now
Woven state

Example (Machine, \( \hat{\varsigma} \))

- call site \((f \ x \ k)\)

- local env
  - \(f \mapsto \hat{t}_{\text{foo}}\)
  - \(k \mapsto \hat{t}_{\text{foo}}\)
  - \(x \mapsto \hat{t}_{\text{foo}}\)

- global env
  - \((f, \hat{t}_{\text{foo}}) \mapsto \ldots\)
  - \((k, \hat{t}_{\text{foo}}) \mapsto \ldots\)
  - \((x, \hat{t}_{\text{foo}}) \mapsto \text{positive}\)
  - \((z, \hat{t}_{\text{bar}}) \mapsto \text{positive}\)

- time \(\hat{t}_f\)

Example (Assumptions, \( \Pi \))

- \((\forall x : (x, \hat{t}_{\text{foo}})\)
- \((\forall z : (z, \hat{t}_{\text{bar}})\)
- \((< x \ z)\))
Woven state

Example (Machine, $\zeta$)

- call site: $(f \ x \ k)$

  - local env:
    - $f \mapsto \hat{t}_{foo}$
    - $k \mapsto \hat{t}_{foo}$
    - $x \mapsto \hat{t}_{foo}$

  - global env:
    - $(f, \hat{t}_{foo}) \mapsto \cdots$
    - $(k, \hat{t}_{foo}) \mapsto \cdots$
    - $(x, \hat{t}_{foo}) \mapsto positive$
    - $(z, \hat{t}_{bar}) \mapsto positive$

- time: $\hat{t}_f$

Example (Assumptions, $\Pi$)

- $(\forall x: (x, \hat{t}_{foo})$ (forall $z: (z, \hat{t}_{bar})$ ($< x \ z))$)
Woven transition relation

\[(\widehat{\varsigma}, \Pi) \Rightarrow (\widehat{\varsigma}', \Pi')\]
Example: Transition

Example
call site \((f \times k)\)

time \(\hat{t}_f\)
Example: Transition

Example

call site (f x k)

local env f \mapsto \hat{t}_{foo}

time \hat{t}_f
Example: Transition

Example

call site \((f \times k)\)

local env \(f \mapsto \hat{t}_{\text{foo}}\)

global env \((f, \hat{t}_{\text{foo}}) \mapsto \text{a closure over } (\lambda (a \ q) \ldots)\)

time \(\hat{t}_{f}\)
Example: Transition

Example

call site \( (f \ x \ k) \)

local env 
\( f \mapsto \hat{t}_{\text{foo}} \)
\( x \mapsto \hat{t}_{\text{foo}} \)

global env 
\( (f, \hat{t}_{\text{foo}}) \mapsto \text{a closure over } (\lambda (a \ q) \ldots) \)

time \( \hat{t}_{f} \)
Example: Transition

Example

call site \((f \ x \ k)\)

local env \(f \mapsto \hat{t}_{\text{foo}}\)
\(x \mapsto \hat{t}_{\text{foo}}\)

global env \((f, \hat{t}_{\text{foo}}) \mapsto \) a closure over \((\lambda (a \ q) \ldots)\)

time \(\hat{t}_f\)

New fact?

\[(\forall \langle x, a \rangle : \langle (x, \hat{t}_{\text{foo}}), (a, \hat{t}_f) \rangle \ (\equiv x \ a))\]
It depends.
Chaining equal values

Candidate for $\Pi'$

$$\phi = (\forall \langle x, a \rangle : \langle (x, \hat{t}_{\text{foo}}), (a, \hat{t}_{\text{f}}) \rangle (= x a))$$

Prerequisites
Can add it if $\Pi \vdash \phi$.

Chicken and egg
How can $\phi$ be in there already?
ΓCFA: Abstract counting

Idea
Keep count of concrete counterparts to abstract identities.
ΓCFA: Abstract counting

Idea
Keep count of concrete counterparts to abstract identities.

Mechanism
► Add counter to every abstract machine state.
► Counter maps each binding to times allocated.
► Stop counting after 1.
ΓCFA: Abstract counting

Idea
Keep count of concrete counterparts to abstract identities.

Mechanism
- Add counter to every abstract machine state.
- Counter maps each binding to times allocated.
- Stop counting after 1.

Theorem
If \( \{\text{binding}_1\} = \{\text{binding}_2\} \),
then \( \text{binding}_1 = \text{binding}_2 \).
Chaining equal values

Candidate for $\Pi'$

$$\phi = (\text{forall } \langle x, a \rangle: \langle (x, \hat{t}_{\text{foo}}), (a, \hat{t}_f) \rangle (= x a))$$

Prerequisites

- Can add it if $\Pi \vdash \phi$.
- Or, if count of $(x, \hat{t}_{\text{foo}})$ is 1 and count of $(a, \hat{t}_f)$ is 0.
CFA: Abstract garbage collection

Idea
Discard unreachable bindings.
Idea
Discard unreachable bindings.

Mechanism
- Start with bindings touched by current state.
- Take transitive closure.
- Can reset unreachable bindings’ counts to 0.
Chaining equal values

Candidate for $\Pi'$

$$\phi = (\forall \langle x, a \rangle : \langle (x, \hat{t}_{\text{foo}}), (a, \hat{t}_f) \rangle (= x \ a))$$

Prerequisites

- Can add it if $\Pi \vdash \phi$.
- Or, if count of $(x, \hat{t}_{\text{foo}})$ is 1 and count of $(a, \hat{t}_f)$ is 0.
- Or, if count of $(x, \hat{t}_{\text{foo}})$ is 1 and $(a, \hat{t}_f)$ is unreachable.
- (More in paper.)
Example: Invertible rebinding

Example

call site \((f (+ x 1) k)\)

local env \(f \mapsto \hat{t}_{f\text{oo}}\)
\(x \mapsto \hat{t}_f\)

global env \((f, \hat{t}_{f\text{oo}}) \mapsto \text{a closure over } (\lambda (x \ q) \ldots)\)

time \(\hat{t}_f\)

Updating assumption base
Can replace \((x, \hat{t}_f)\) with \((- (x, \hat{t}_f) \ 1)\) in \(\Pi\)?
Example: Invertible rebinding

Example
call site  \((f \ (+ \ x \ 1) \ k)\)

local env  \(f \mapsto \hat{t}_{f_{oo}}\)
\(x \mapsto \hat{t}_f\)

global env  \((f, \hat{t}_{f_{oo}}) \mapsto \text{a closure over } (\lambda (x \ q) \ldots)\)

time  \(\hat{t}_f\)

Updating assumption base
Can replace \((x, \hat{t}_f)\) with \((- (x, \hat{t}_f) 1)\) in \(\Pi\)?

Yes, if \((x, \hat{t}_f)\) is unreachable and its count is 1.

\((E.g. \text{ tail recursion, for loops.})\)

\((More \text{ on this in the paper.})\)
Example: Conditional

Example

call site  (if (< i (alen a)) ...  ... )
Example: Conditional

Example

call site (if (< i (alen a)) ... ...)

Case 1
Π can (dis)prove (< i (alen a)). Branch one way.

\[ \hat{\zeta}_{true} \]

\[ \hat{\zeta} \]

\[ \hat{\zeta}_{false} \]
Example: Conditional

Example

call site  \((\text{if } (< \ i \ \text{alen} \ a)) \ \ldots \ \ldots)\)

Case 2

\((< \ i \ \text{alen} \ a))\) has one counterpart. Branch both ways & assert.
Example: Conditional

Example
 call site (if (< i (alen a)) ... ...)

Case 3
None of the above. Branch both ways. Don’t touch $\Pi'$. 

\[
\hat{\varsigma} \quad \hat{\varsigma}_{true} \\
\hat{\varsigma}_{false}
\]
Walkthrough: Simple for loop

Example

```javascript
for (i = 0; i < a.length; i++)
    print(a[i]) ;
```

Example (CPS)

```lisp
(letrec ((loop (λ (i)
    (if (< i (alen a))
        (print (aget a i) (λ ()
            (loop (+ i 1))))
        ...))))
   (loop 0))
```

Parameters

- 0CFA contour set. (Bindings = Variables.)
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
    (if (< i (alen a))
        (print (aget a i) (λ ()
            (loop (+ i 1))))
        ...))))
    (loop 0))

Assumption base, Π
(< 0 (alen a))
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
    (if (< i (alen a))
        (print (aget a i) (λ ()
            (loop (+ i 1)))))
    ...))))

(loop 0)

Assumption base, Π

(< 0 (alen a))
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
  (if (< i (alen a))
     (print (aget a i) (λ ()
       (loop (+ i 1))))
     ...))))
(loop 0))

Assumption base, Π
(< 0 (alen a)), (= 0 i)
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
          (if (< i (alen a))
            (print (aget a i) (λ ()
               (loop (+ i 1))))
            ...)))))

(loop 0))

Assumption base, Π

(< 0 (alen a)), (= 0 i)
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
    (if (< i (alen a))
        (print (aget a i) (λ ()
            (loop (+ i 1)))))
        ...))))
(loop 0))

Assumption base, Π
(< 0 (alen a)), (= 0 i)

Safe
0 ≤ i < (alen a) holds!
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
    (if (< i (alen a))
        (print (aget a i) (λ ()
            (loop (+ i 1)))))
    ...))))

(loop 0))

Assumption base, \(\Pi\)

(< 0 (alen a)), (= 0 i)
Walkthrough: Simple for loop

(letrec ((loop (\ (i)
  (if (< i (alen a))
   (print (aget a i) (\ ()
     (loop (+ i 1)))
   ...
)))))
(loop 0))

Assumption base, \( \Pi \)
(< 0 (alen a)), (= 0 i)
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
    (if (< i (alen a))
        (print (aget a i) (λ ()
            (loop (+ i 1))))
        ...)))
    (loop 0))

Assumption base, \( \Pi \)

\(< 0 (alen a)), (= 0 (− i 1))\)
(letrec ((loop (\(i)  
    (if (< i (alen a))  
      (print (aget a i) (\()  
        (loop (+ i 1))))  
    ...)))  
  (loop 0)))

Assumption base, \(\Pi\)

(< 0 (alen a)), (\(\leq\) 0 i)
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
    (if (< i (alen a))
        (print (aget a i) (λ ()
            (loop (+ i 1)))))
    ...)))

(loop 0))

Assumption base, Π
(< 0 (alen a)), (≤ 0 i), (< i (alen a))

Safe
0 ≤ i < (alen a) holds!
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
    (if (< i (alen a))
        (print (aget a i) (λ ()
            (loop (+ i 1))))
        ...
    ))))

(loop 0))

Assumption base, Π
(< 0 (alen a)), (≤ 0 i), (< i (alen a))
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
            (if (< i (alen a))
                (print (aget a i) (λ ()
                                (loop (+ i 1)))))
            ...)))))

(loop 0)

Assumption base, Π

(< 0 (alen a)), (≤ 0 i), (< i (alen a))
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
  (if (< i (alen a))
    (print (aget a i) (λ ()
      (loop (+ i 1))))
    ...))))

(loop 0))

Assumption base, Π
(< 0 (alen a)), (≤ 0 (- i 1)), (< (- i 1) (alen a))
Walkthrough: Simple for loop

(letrec ((loop (λ (i)
    (if (< i (alen a))
        (print (aget a i) (λ ()
            (loop (+ i 1)))))
    ...)))

(loop 0))

Assumption base, \( \Pi \)

\(< 0 (alen a)), (\leq 0 i), (\leq (- i 1) (alen a)), (\less i (alen a))\)

Safe

\(0 \leq i < (alen a)\) holds!
Walkthrough: Simple for loop

(letrec ((loop (λ (i)

  (if (< i (alen a))
    (print (aget a i) (λ ()
    (loop (+ i 1)))
    ...
    )))

  (loop 0)))

Assumption base, Π
(< 0 (alen a)), (≤ 0 i), (< i (alen a))

Finished
State already visited.
More in the paper

- Formal treatment.
- Flow analysis as oracle inference rules.
- More rules for assumption base management.
- Rules for handling arrays.
- Three-page worked example for vertex arrays.
Related & inspiring work

- Cousot & Cousot. (Abstract interpretation)
Related & inspiring work

- Cousot & Cousot. (Abstract interpretation)
- Cousot & Cousot. (Invariant synthesis)
Related & inspiring work

- Cousot & Cousot. (Abstract interpretation)
- Cousot & Cousot. (Invariant synthesis)
- Cousot & Cousot. (Widening & narrowing)
Related & inspiring work

- Cousot & Cousot. (Abstract interpretation)
- Cousot & Cousot. (Invariant synthesis)
- Cousot & Cousot. (Widening & narrowing)
- Cousot & Cousot. (Relational abstract domains)
Related & inspiring work

- Cousot & Cousot. (Abstract interpretation)
- Cousot & Cousot. (Invariant synthesis)
- Cousot & Cousot. (Widening & narrowing)
- Cousot & Cousot. (Relational abstract domains)
- Shivers. (Control-flow analysis w/ factored environment)
Related & inspiring work

- Cousot & Cousot. (Abstract interpretation)
- Cousot & Cousot. (Invariant synthesis)
- Cousot & Cousot. (Widening & narrowing)
- Cousot & Cousot. (Relational abstract domains)
- Shivers. (Control-flow analysis w/ factored environment)
- Hudak. (Abstract reference counting)
Related & inspiring work

- Cousot & Cousot. (Abstract interpretation)
- Cousot & Cousot. (Invariant synthesis)
- Cousot & Cousot. (Widening & narrowing)
- Cousot & Cousot. (Relational abstract domains)
- Shivers. (Control-flow analysis w/ factored environment)
- Hudak. (Abstract reference counting)
- Nanevski & Morrisett. (Soundness of theorem prover interaction)
Related & inspiring work

- Cousot & Cousot. (Abstract interpretation)
- Cousot & Cousot. (Invariant synthesis)
- Cousot & Cousot. (Widening & narrowing)
- Cousot & Cousot. (Relational abstract domains)
- Shivers. (Control-flow analysis w/ factored environment)
- Hudak. (Abstract reference counting)
- Nanevski & Morrisett. (Soundness of theorem prover interaction)
- Ball. (Predicate abstraction)
Related & inspiring work

- Cousot & Cousot. (Abstract interpretation)
- Cousot & Cousot. (Invariant synthesis)
- Cousot & Cousot. (Widening & narrowing)
- Cousot & Cousot. (Relational abstract domains)
- Shivers. (Control-flow analysis w/ factored environment)
- Hudak. (Abstract reference counting)
- Nanevski & Morrisett. (Soundness of theorem prover interaction)
- Ball. (Predicate abstraction)
- ...
Related & future work

\[ \nabla \text{CFA} \quad \downarrow \quad \nabla \text{LFA} \\
\Gamma \text{CFA} \quad \nabla \text{LFA} \\
\Gamma \text{CFA} \quad \nabla \text{LFA} \\
\]
Related & future work

\[ \Delta CFA \]

\[ \Gamma CFA \]

\[ LFA \]
Related & future work
Conclusion

$\text{HOFA} + \text{FOL} = \text{LFA}$
Conclusion

HOFA + FOL = LFA
Merci
Question
What are some other applications of LFA?

Answer

- Improving flow precision.
- Static checks of `assert` statements.
- Pre- and post-condition checks.
Question
Do you have an implementation?

Answer
Half of one:
  - Modified ACL/2 for theorem prover.
  - Modified $\Gamma$CFA.
Goal: Meet in the middle.
Question
Does a backward version exist?

Answer
- Not yet.
- Start by widening into constraint-solving form.
Question
What is the complexity of LFA?

Answer
Exponential in theory.
Usually much friendlier in practice.
It depends on...
...the contour set.
...the degree of widening to assumption base & abstract heap.
Question
Do you support floats?

Answer
Patrick, not yet.