Constraint-based Analysis

Harry Xu

CS 253/INF 212

Spring 2013

Acknowledgements

Many slides in this file were taken from Prof. Crista Lope's slides on functional programming as well as slides provided by the authors of the book *Principles of Program Analysis* available at

http://www2.imm.dtu.dk/~hrni/PPA/ppasup2004.html

Lambda Calculus

Lambda calculus is a formal system for expressing computation by way of variable binding and substitution

λ

Syntax

$M ::= x$	(variable)
$\quad \mid \lambda x.M$	(abstraction)
$\quad \mid MM$	(application)

Nothing else!

- No numbers
- No arithmetic operations
- No loops
- No etc.

Symbolic computation

Syntax reminder

anonymous functions

λx.M ➔ function(x) { M }

LM, e.g. $\underbrace{\lambda x.N}_{L}\ \underbrace{y}_{M}$ ➔ apply L to M

Terminology – bound variables

λx.M

The *binding operator* λ **binds** *the variable x in the* λ*-term x.M*

- M is called the *scope of x*
- *x* is said to be a *bound variable*

Terminology – free variables

Free variables are all symbols that aren’t bound (duh)

$$FV(x) = \{x\}$$
$$FV(MN) = FV(M) \cup FV(N)$$
$$FV(x.M) = FV(M) - x$$

Renaming of bound variables

$\lambda x.M = \lambda y.(M[y/x])$ if y not in FV(M)

i.e. you can replace x with y
aka “renaming”

α-conversion

Operational Semantics

- Evaluating function application: ($\lambda x.e_1) \, e_2$
  - Replace every x *in* e_1 *with* e_2
  - Evaluate the resulting term
  - Return the result of the evaluation
- Formally: "β-reduction" (aka "substitution")
  - $(\lambda\, x.e_1) \, e_2 \rightarrow_{\beta} e_1[e_2/x]$
  - A term that can be β-reduced is a *redex (reducible expression)*
  - We omit β when obvious

Note again

- Computation = pure symbolic manipulation
  - Replace some symbols with other symbols

Scoping etc.

- Scope of λ extends as far to the right as possible
  - $\lambda x.\lambda y.xy$ *is* $\lambda x.(\lambda y.(x\ y))$
- Function application is left-associative
  - xyz *means* $(xy)z$

Multiple arguments

- $\lambda(x,y).e$???
  - Doesn't exist
- Solution: $\lambda x.\lambda y.e$ *[remember, $(\lambda x.(\lambda y.e))$]*
  - A function that takes *x* and returns another function that takes *y* and returns *e*
  - $(\lambda x.\lambda y.e)\ a\ b \rightarrow (\lambda y.e[a/x])\ b \rightarrow e[a/x][b/y]$
  - "Currying" after Curry: transformation of multi-arg functions into higher-order functions

- Multiple argument functions are nothing but syntactic sugar

Boolean Values and Conditionals

- True = $\lambda x.\lambda y.x$
- False = $\lambda x.\lambda y.y$
- If-then-else = $\lambda a.\lambda b.\lambda c.\ a\ b\ c$

Boolean Values and Conditionals

- If True M N = $\underbrace{(\lambda a.\lambda b.\lambda c.abc)}_{\text{If}}$ True M N

$\rightarrow$ $(\lambda b.\lambda c.\mathit{True}\ b\ c)$ M N
$\rightarrow$ $(\lambda c.\mathit{True}\ M\ c)$ N
$\rightarrow$ $\mathit{True}\ M\ N$
= $(\lambda x.\lambda y.x)\ M\ N$
$\rightarrow$ $(\lambda y.M)\ N$
$\rightarrow$ M

Numbers

- Numbers are counts of things, any things. Like function applications!
  - $0 = \lambda f.\ \lambda x.\ x$
  - $1 = \lambda f.\ \lambda x.\ (f\ x)$
  - $2 = \lambda f.\ \lambda x.\ (f\ (f\ x))$
  - $3 = \lambda f.\ \lambda x.\ (f\ (f\ (f\ x)))$
  - …
  - $N = \lambda f.\ \lambda x.\ (f^N\ x)$

Church numerals

Successor

- succ = $\lambda n.\ \lambda f.\ \lambda x.\ f\ (n\ f\ x)$
  - Want to try it on succ(1)?
  - $\lambda n.\ \lambda f.\ \lambda x.\ f\ (n\ f\ x)\ \underbrace{(\lambda f.\ \lambda x.\ (f\ x))}_{1}$

$\rightarrow \lambda f.\ \lambda x.\ f\ ((\lambda f.\ \lambda x.\ (f\ x))\ f\ x)$

$\rightarrow \underbrace{\lambda f.\ \lambda x.\ f\ (f\ x)}_{2\ !}$

Closures

- Function with free variables that are bound to values in the enclosing environment

```
(lambda (x)
   (lambda (y)
     x+y))
```

closure

Function Execution by Substitution

plus x y = x + y

1. **plus 2 3 → 2 + 3 → 5**
2. **plus (2*3) (plus 4 5)**

→ **plus 6 (4+5)**	→ **(2*3) +(plus 4 5)**
→ **plus 6 9**	→ **6 + (4+5)**
→ **6 + 9**	→ **6 + 9**
→ **15**	→ **15**

The final answer did not depend upon the order in which reductions were performed

Blocks

```
let
      x =  a * a
      y =  b * b
in
      (x - y)/(x + y)
```

- a variable can have at most one definition in a block
- ordering of bindings does not matter

Layout Convention in Haskell

This convention allows us to omit many delimiters

```
let
      x =  a * a
      y =  b * b
in
      (x - y)/(x + y)
```

is the same as

```
let
      { x =  a * a ;
        y =  b * b ;}
in
      (x - y)/(x + y)
```

α-renaming

let y = 2 * 2 x = 3 + 4 z = ***let*** x = 5 * 5 w = x + y * x ***in*** w ***in*** x + y + z	$\equiv$	***let*** y = 2 * 2 x = 3 + 4 z = ***let*** x' = 5 * 5 w = x' + y * x' ***in*** w ***in*** x + y + z

Lexical Scoping

```
let
   y = 2 * 2
   x = 3 + 4
   z = let
          x = 5 * 5
          w = x + y * x
       in
          w
in
   x + y + z
```

Lexically closest definition of a variable prevails.

Dynamic Dispatch Problem

```
⋮
[call p(p1,1,v)]
⋮
[call p(p2,2,v)]
⋮
```

$[\texttt{call p(p1,1,v)}]^{\ell_c^1}_{\ell_r^1}$

$[\texttt{call p(p2,2,v)}]^{\ell_c^2}_{\ell_r^2}$

$\texttt{proc p(procval q, val x, res y) is}^{\ell_n}$

$\vdots$

$[\texttt{call q (x,y)}]^{\ell_c^p}_{\ell_r^p}$

$\vdots$

$\texttt{end}^{\ell_x}$

which procedure is called?

These problems arise for:

- imperative languages with procedures as parameters
- object oriented languages
- functional languages

Example

```
let f = fn x => x 1;
    g = fn y => y+2;
    h = fn z => z+3
in (f g) + (f h)
```

The aim of Control Flow Analysis:

For each function application, which functions may be applied?

Control Flow Analysis computes the interprocedural flow relation used when formulating interprocedural Data Flow Analysis.

A Simple Functional Language

Syntactic categories:

$$\begin{array}{rcll} e & \in & \mathbf{Exp} & \text{expressions (or labelled terms)} \\ t & \in & \mathbf{Term} & \text{terms (or unlabelled expressions)} \\[4pt] f, x & \in & \mathbf{Var} & \text{variables} \\ c & \in & \mathbf{Const} & \text{constants} \\ op & \in & \mathbf{Op} & \text{binary operators} \\ \ell & \in & \mathbf{Lab} & \text{labels} \end{array}$$

Syntax:

$$e ::= t^{\ell}$$

$$\begin{array}{rcl} t & ::= & c \mid x \mid \texttt{fn}\ x\ \texttt{=>}\ e_0 \mid \texttt{fun}\ f\ x\ \texttt{=>}\ e_0 \mid e_1\ e_2 \\ & \mid & \texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \mid \texttt{let}\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2 \mid e_1\ op\ e_2 \end{array}$$

(Labels correspond to program points or nodes in the parse tree.)

Examples

- $((\texttt{fn x => x}^1)^2\ (\texttt{fn y => y}^3)^4)^5$

- $(\texttt{let f = }(\texttt{fn x => }(\texttt{x}^1\ \texttt{1}^2)^3)^4;$
  $\texttt{in }(\texttt{let g = }(\texttt{fn y => y}^5)^6;$
  $\texttt{in }(\texttt{let h = }(\texttt{fn z => z}^7)^8$
  $\texttt{in }((\texttt{f}^9\ \texttt{g}^{10})^{11} + (\texttt{f}^{12}\ \texttt{h}^{13})^{14})^{15})^{16})^{17})^{18}$

- $(\texttt{let g = }(\texttt{fun f x => }(\texttt{f}^1\ (\texttt{fn y => y}^2)^3)^4)^5$
  $\texttt{in }(\texttt{g}^6\ (\texttt{fn z => z}^7)^8)^9)^{10}$

0-CFA Analysis

- Abstract domains (i.e., maps)
- Specification of the analysis

Abstract Domains

The *result* of a 0-CFA analysis is a pair $(\widehat{\mathsf{C}}, \widehat{\rho})$:

- $\widehat{\mathsf{C}}$ is the *abstract cache* associating abstract values with each labelled program point

- $\widehat{\rho}$ is the *abstract environment* associating abstract values with each variable

Example

$((\texttt{fn x => x}^1)^2\ (\texttt{fn y => y}^3)^4)^5$

Three guesses of a 0-CFA analysis result:

	$(\widehat{C}_e, \widehat{\rho}_e)$	$(\widehat{C}'_e, \widehat{\rho}'_e)$	$(\widehat{C}''_e, \widehat{\rho}''_e)$
1	$\{\texttt{fn y => y}^3\}$	$\{\texttt{fn y => y}^3\}$	$\{\texttt{fn x => x}^1, \texttt{fn y => y}^3\}$
2	$\{\texttt{fn x => x}^1\}$	$\{\texttt{fn x => x}^1\}$	$\{\texttt{fn x => x}^1, \texttt{fn y => y}^3\}$
3	$\emptyset$	$\emptyset$	$\{\texttt{fn x => x}^1, \texttt{fn y => y}^3\}$
4	$\{\texttt{fn y => y}^3\}$	$\{\texttt{fn y => y}^3\}$	$\{\texttt{fn x => x}^1, \texttt{fn y => y}^3\}$
5	$\{\texttt{fn y => y}^3\}$	$\{\texttt{fn y => y}^3\}$	$\{\texttt{fn x => x}^1, \texttt{fn y => y}^3\}$
x	$\{\texttt{fn y => y}^3\}$	$\emptyset$	$\{\texttt{fn x => x}^1, \texttt{fn y => y}^3\}$
y	$\emptyset$	$\emptyset$	$\{\texttt{fn x => x}^1, \texttt{fn y => y}^3\}$

A More Complicated Example

```
(let g = (fun f x => (f^1 (fn y => y^2)^3)^4)^5
 in (g^6 (fn z => z^7)^8)^9)^10
```

Abbreviations:

$$
\begin{array}{rcl}
\mathsf{f} & = & \texttt{fun f x => (f}^1\ \texttt{(fn y => y}^2\texttt{)}^3\texttt{)}^4 \\
\mathsf{id}_y & = & \texttt{fn y => y}^2 \\
\mathsf{id}_z & = & \texttt{fn z => z}^7
\end{array}
$$

One guess of a 0-CFA analysis result:

$$
\begin{array}{rclrclrcl}
\widehat{\mathsf{C}}_{\mathsf{lp}}(1) & = & \{\mathsf{f}\} & \widehat{\mathsf{C}}_{\mathsf{lp}}(6) & = & \{\mathsf{f}\} & \widehat{\rho}_{\mathsf{lp}}(\texttt{f}) & = & \{\mathsf{f}\} \\
\widehat{\mathsf{C}}_{\mathsf{lp}}(2) & = & \emptyset & \widehat{\mathsf{C}}_{\mathsf{lp}}(7) & = & \emptyset & \widehat{\rho}_{\mathsf{lp}}(\texttt{g}) & = & \{\mathsf{f}\} \\
\widehat{\mathsf{C}}_{\mathsf{lp}}(3) & = & \{\mathsf{id}_y\} & \widehat{\mathsf{C}}_{\mathsf{lp}}(8) & = & \{\mathsf{id}_z\} & \widehat{\rho}_{\mathsf{lp}}(\texttt{x}) & = & \{\mathsf{id}_y, \mathsf{id}_z\} \\
\widehat{\mathsf{C}}_{\mathsf{lp}}(4) & = & \emptyset & \widehat{\mathsf{C}}_{\mathsf{lp}}(9) & = & \emptyset & \widehat{\rho}_{\mathsf{lp}}(\texttt{y}) & = & \emptyset \\
\widehat{\mathsf{C}}_{\mathsf{lp}}(5) & = & \{\mathsf{f}\} & \widehat{\mathsf{C}}_{\mathsf{lp}}(10) & = & \emptyset & \widehat{\rho}_{\mathsf{lp}}(\texttt{z}) & = & \emptyset
\end{array}
$$

Abstract Domains

Formally:

$$\begin{array}{llllll} \widehat{v} & \in & \widehat{\mathbf{Val}} & = & \mathcal{P}(\mathbf{Term}) & \textit{abstract values} \\ \widehat{\rho} & \in & \widehat{\mathbf{Env}} & = & \mathbf{Var} \to \widehat{\mathbf{Val}} & \textit{abstract environments} \\ \widehat{\mathsf{C}} & \in & \widehat{\mathbf{Cache}} & = & \mathbf{Lab} \to \widehat{\mathbf{Val}} & \textit{abstract caches} \end{array}$$

An abstract value $\widehat{v}$ is a set of terms of the forms

- $\texttt{fn}\ x\ \texttt{=>}\ e_0$
- $\texttt{fun}\ f\ x\ \texttt{=>}\ e_0$

When is a proposed guess $(\widehat{\mathsf{C}}, \widehat{\rho})$ of an analysis results an *acceptable 0-CFA analysis* for the program?

Specification of 0-CFA

$(\widehat{\mathsf{C}}, \widehat{\rho}) \models e$ means that $(\widehat{\mathsf{C}}, \widehat{\rho})$ is an *acceptable Control Flow Analysis* of the expression e

The relation $\models$ has functionality:

$$\models \; : (\widehat{\mathbf{Cache}} \times \widehat{\mathbf{Env}} \times \mathbf{Exp}) \to \{\mathit{true}, \mathit{false}\}$$

Clauses for 0-CFA (1)

$(\widehat{\mathsf{C}}, \widehat{\rho}) \models c^\ell$ always

$(\widehat{\mathsf{C}}, \widehat{\rho}) \models x^\ell$ $\underline{\text{iff}}$ $\widehat{\rho}(x) \subseteq \widehat{\mathsf{C}}(\ell)$

$$
\begin{array}{ll}
(\widehat{\mathsf{C}}, \widehat{\rho}) \models (\mathtt{let}\ x = t_1^{\ell_1}\ \mathtt{in}\ t_2^{\ell_2})^\ell & \\
\quad \underline{\text{iff}} \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2} \ \wedge & \\
\qquad\qquad \widehat{\mathsf{C}}(\ell_1) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\mathsf{C}}(\ell) &
\end{array}
$$

Clauses for 0-CFA (2)

$$(\widehat{C}, \widehat{\rho}) \models (\texttt{if } t_0^{\ell_0} \texttt{ then } t_1^{\ell_1} \texttt{ else } t_2^{\ell_2})^{\ell}$$
$$\underline{\text{iff}} \quad (\widehat{C}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge$$
$$(\widehat{C}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge \ (\widehat{C}, \widehat{\rho}) \models t_2^{\ell_2} \wedge$$
$$\widehat{C}(\ell_1) \subseteq \widehat{C}(\ell) \ \wedge \ \widehat{C}(\ell_2) \subseteq \widehat{C}(\ell)$$

$$(\widehat{C}, \widehat{\rho}) \models (t_1^{\ell_1} \ op \ t_2^{\ell_2})^{\ell}$$
$$\underline{\text{iff}} \quad (\widehat{C}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge \ (\widehat{C}, \widehat{\rho}) \models t_2^{\ell_2}$$

Clauses for 0-CFA (3)

$(\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{fn}\ x\ \texttt{=>}\ t_0^{\ell_0})^\ell$ iff $\{\texttt{fn}\ x\ \texttt{=>}\ t_0^{\ell_0}\} \subseteq \widehat{\mathsf{C}}(\ell)$

$$
\begin{array}{lll}
(\widehat{\mathsf{C}}, \widehat{\rho}) \models (t_1^{\ell_1}\ t_2^{\ell_2})^\ell & & \\
\quad \underline{\text{iff}} & (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge \ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2} \ \wedge & \\
& (\forall (\texttt{fn}\ x\ \texttt{=>}\ t_0^{\ell_0}) \in \widehat{\mathsf{C}}(\ell_1) : & (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge \\
& & \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x) \ \wedge \ \widehat{\mathsf{C}}(\ell_0) \subseteq \widehat{\mathsf{C}}(\ell))
\end{array}
$$

Clauses for 0-CFA (4)

$(\widehat{\mathsf{C}}, \widehat{\rho}) \models (\texttt{fun}\ f\ x\ \texttt{=>}\ e_0)^\ell$ iff $\{\texttt{fun}\ f\ x\ \texttt{=>}\ e_0\} \subseteq \widehat{\mathsf{C}}(\ell)$

$$
\begin{array}{lll}
\multicolumn{3}{l}{(\widehat{\mathsf{C}}, \widehat{\rho}) \models (t_1^{\ell_1}\ t_2^{\ell_2})^\ell} \\
\underline{\text{iff}} & \multicolumn{2}{l}{(\widehat{\mathsf{C}}, \widehat{\rho}) \models t_1^{\ell_1} \ \wedge\ (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_2^{\ell_2} \ \wedge} \\
 & (\forall (\texttt{fn}\ x\ \texttt{=>}\ t_0^{\ell_0}) \in \widehat{\mathsf{C}}(\ell_1) : & (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge \\
 & & \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{\mathsf{C}}(\ell_0) \subseteq \widehat{\mathsf{C}}(\ell)) \ \wedge \\
 & (\forall (\texttt{fun}\ f\ x\ \texttt{=>}\ t_0^{\ell_0}) \in \widehat{\mathsf{C}}(\ell_1) : & (\widehat{\mathsf{C}}, \widehat{\rho}) \models t_0^{\ell_0} \ \wedge \\
 & & \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{\mathsf{C}}(\ell_0) \subseteq \widehat{\mathsf{C}}(\ell) \ \wedge \\
 & & \{\texttt{fun}\ f\ x\ \texttt{=>}\ t_0^{\ell_0}\} \subseteq \widehat{\rho}(f))
\end{array}
$$

Constraint-based 0-CFA (1)

$\mathcal{C}_\star[\![e_\star]\!]$ is a set of constraints of the form

$$\mathit{lhs} \subseteq \mathit{rhs}$$

$$\{t\} \subseteq \mathit{rhs}' \Rightarrow \mathit{lhs} \subseteq \mathit{rhs}$$

where

$$\mathit{rhs} ::= \mathsf{C}(\ell) \mid \mathsf{r}(x)$$

$$\mathit{lhs} ::= \mathsf{C}(\ell) \mid \mathsf{r}(x) \mid \{t\}$$

and all occurrences of t are of the form $\texttt{fn}\ x\ \texttt{=>}\ e_0$ or $\texttt{fun}\ f\ x\ \texttt{=>}\ e_0$

Constraint-based 0-CFA (2)

$$\mathcal{C}_\star[\![(\texttt{fn}\ x\ \texttt{=>}\ e_0)^\ell]\!] = \{\ \{\texttt{fn}\ x\ \texttt{=>}\ e_0\} \subseteq \mathsf{C}(\ell)\ \} \cup \mathcal{C}_\star[\![e_0]\!]$$

$$\begin{aligned}\mathcal{C}_\star[\![(\texttt{fun}\ f\ x\ \texttt{=>}\ e_0)^\ell]\!] &= \{\ \{\texttt{fun}\ f\ x\ \texttt{=>}\ e_0\} \subseteq \mathsf{C}(\ell)\ \} \cup \mathcal{C}_\star[\![e_0]\!] \\ &\cup \{\ \{\texttt{fun}\ f\ x\ \texttt{=>}\ e_0\} \subseteq \mathsf{r}(f)\ \}\end{aligned}$$

$$\begin{aligned}\mathcal{C}_\star[\![(t_1^{\ell_1}\ t_2^{\ell_2})^\ell]\!] = \mathcal{C}_\star[\![t_1^{\ell_1}]\!] &\cup \mathcal{C}_\star[\![t_2^{\ell_2}]\!] \\ &\cup \{\ \{t\} \subseteq \mathsf{C}(\ell_1) \Rightarrow \mathsf{C}(\ell_2) \subseteq \mathsf{r}(x) \mid t = (\texttt{fn}\ x\ \texttt{=>}\ t_0^{\ell_0}) \in \mathbf{Term}_\star\} \\ &\cup \{\ \{t\} \subseteq \mathsf{C}(\ell_1) \Rightarrow \mathsf{C}(\ell_0) \subseteq \mathsf{C}(\ell) \mid t = (\texttt{fn}\ x\ \texttt{=>}\ t_0^{\ell_0}) \in \mathbf{Term}_\star\} \\ &\cup \{\ \{t\} \subseteq \mathsf{C}(\ell_1) \Rightarrow \mathsf{C}(\ell_2) \subseteq \mathsf{r}(x) \mid t = (\texttt{fun}\ f\ x\ \texttt{=>}\ t_0^{\ell_0}) \in \mathbf{Term}_\star\} \\ &\cup \{\ \{t\} \subseteq \mathsf{C}(\ell_1) \Rightarrow \mathsf{C}(\ell_0) \subseteq \mathsf{C}(\ell) \mid t = (\texttt{fun}\ f\ x\ \texttt{=>}\ t_0^{\ell_0}) \in \mathbf{Term}_\star\}\end{aligned}$$

Constraint-based 0-CFA (3)

$$\mathcal{C}_\star[\![c^\ell]\!] = \emptyset$$

$$\mathcal{C}_\star[\![x^\ell]\!] = \{\ \mathsf{r}(x) \subseteq \mathsf{C}(\ell)\ \}$$

$$\begin{aligned}\mathcal{C}_\star[\![(\texttt{if}\ t_0^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^\ell]\!] &= \mathcal{C}_\star[\![t_0^{\ell_0}]\!] \cup \mathcal{C}_\star[\![t_1^{\ell_1}]\!] \cup \mathcal{C}_\star[\![t_2^{\ell_2}]\!] \\ &\cup \{\ \mathsf{C}(\ell_1) \subseteq \mathsf{C}(\ell)\ \} \\ &\cup \{\ \mathsf{C}(\ell_2) \subseteq \mathsf{C}(\ell)\ \}\end{aligned}$$

$$\begin{aligned}\mathcal{C}_\star[\![(\texttt{let}\ x\ \texttt{=}\ t_1^{\ell_1}\ \texttt{in}\ t_2^{\ell_2})^\ell]\!] &= \mathcal{C}_\star[\![t_1^{\ell_1}]\!] \cup \mathcal{C}_\star[\![t_2^{\ell_2}]\!] \\ &\cup \{\ \mathsf{C}(\ell_1) \subseteq \mathsf{r}(x)\ \} \cup \{\ \mathsf{C}(\ell_2) \subseteq \mathsf{C}(\ell)\ \}\end{aligned}$$

$$\mathcal{C}_\star[\![(t_1^{\ell_1}\ op\ t_2^{\ell_2})^\ell]\!] = \mathcal{C}_\star[\![t_1^{\ell_1}]\!] \cup \mathcal{C}_\star[\![t_2^{\ell_2}]\!]$$

Constraint-based 0-CFA (4)

$$
\begin{aligned}
&\mathcal{C}_\star[\![((\texttt{fn x => x}^1)^2\ (\texttt{fn y => y}^3)^4)^5]\!] = \\
&\quad \{\ \{\texttt{fn x => x}^1\} \subseteq \mathsf{C}(2), \\
&\quad\ \ \mathsf{r}(\texttt{x}) \subseteq \mathsf{C}(1), \\
&\quad\ \ \{\texttt{fn y => y}^3\} \subseteq \mathsf{C}(4), \\
&\quad\ \ \mathsf{r}(\texttt{y}) \subseteq \mathsf{C}(3), \\
&\quad\ \ \{\texttt{fn x => x}^1\} \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(\texttt{x}), \\
&\quad\ \ \{\texttt{fn x => x}^1\} \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(1) \subseteq \mathsf{C}(5), \\
&\quad\ \ \{\texttt{fn y => y}^3\} \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(\texttt{y}), \\
&\quad\ \ \{\texttt{fn y => y}^3\} \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(5)\ \}
\end{aligned}
$$

Solving the Constraints (1)

Input: a set of constraints $\mathcal{C}_\star[\![e_\star]\!]$

Output: the least solution $(\widehat{\mathsf{C}}, \widehat{\rho})$ to the constraints

Data structures: a graph with one node for each $\mathsf{C}(\ell)$ and $\mathsf{r}(x)$ (where $\ell \in \mathbf{Lab}_\star$ and $x \in \mathbf{Var}_\star$) and zero, one or two edges for each constraint in $\mathcal{C}_\star[\![e_\star]\!]$

- W: the worklist of the nodes whose outgoing edges should be traversed
- D: an array that for each node gives an element of $\widehat{\mathbf{Val}}_\star$
- E: an array that for each node gives a list of constraints influenced (and outgoing edges)

Auxiliary procedure:

procedure add(q,d) is if $\neg$ ($d \subseteq$ D[q]) then D[q] := D[q] $\cup$ d;
W := cons(q,W);

Solving the Constraints (2)

Step 1 Initialisation

W := nil;

for q in Nodes do D[q] := $\emptyset$; E[q] := nil;

Step 2 Building the graph

for cc in $\mathcal{C}_\star[\![e_\star]\!]$ do

case cc of $\{t\} \subseteq p$: add(p,$\{t\}$);

$p_1 \subseteq p_2$: E[p_1] := cons(cc,E[p_1]);

$\{t\} \subseteq p \Rightarrow p_1 \subseteq p_2$: E[$p_1$] := cons($cc$,E[$p_1$]);

E[p] := cons(cc,E[p]);

Step 3 Iteration

while W $\neq$ nil do

q := head(W); W := tail(W);

for cc in E[q] do

case cc of $p_1 \subseteq p_2$: add(p_2, D[p_1]);

$\{t\} \subseteq p \Rightarrow p_1 \subseteq p_2$: if $t \in$ D[p] then add(p_2, D[p_1]);

Step 4 Recording the solution

for ℓ in $\mathbf{Lab}_\star$ do $\widehat{\mathsf{C}}(\ell)$:= D[C(ℓ)]; for x in $\mathbf{Var}_\star$ do $\widehat{\rho}(x)$:= D[r(x)];

Example

Initialisation of data structures

p	$\mathsf{D}[p]$	$\mathsf{E}[p]$
C(1)	$\emptyset$	$[\mathsf{id}_x \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(1) \subseteq \mathsf{C}(5)]$
C(2)	id_x	$[\mathsf{id}_y \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(5),\ \ \mathsf{id}_y \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(\mathrm{y}),$ $\mathsf{id}_x \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(1) \subseteq \mathsf{C}(5),\ \ \mathsf{id}_x \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(\mathrm{x})]$
C(3)	$\emptyset$	$[\mathsf{id}_y \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(5)]$
C(4)	id_y	$[\mathsf{id}_y \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(\mathrm{y}),\ \ \mathsf{id}_x \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(\mathrm{x})]$
C(5)	$\emptyset$	$[\ ]$
r(x)	$\emptyset$	$[\mathsf{r}(\mathrm{x}) \subseteq \mathsf{C}(1)]$
r(y)	$\emptyset$	$[\mathsf{r}(\mathrm{y}) \subseteq \mathsf{C}(3)]$

Iteration Steps

Iteration steps

W	[C(4),C(2)]	[r(x),C(2)]	[C(1),C(2)]	[C(5),C(2)]	[C(2)]	[ ]
p	D[p]	D[p]	D[p]	D[p]	D[p]	D[p]
C(1)	$\emptyset$	$\emptyset$	id_y	id_y	id_y	id_y
C(2)	id_x	id_x	id_x	id_x	id_x	id_x
C(3)	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
C(4)	id_y	id_y	id_y	id_y	id_y	id_y
C(5)	$\emptyset$	$\emptyset$	$\emptyset$	id_y	id_y	id_y
r(x)	$\emptyset$	id_y	id_y	id_y	id_y	id_y
r(y)	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

K-CFA

- An abstract value in K-CFA is a calling context that records the last k dynamic call points (i.e., call sites)
- Contexts are sequences of labels of length at most k and they will be updated whenever a function application is analyzed

K-CFA for Imperative Languages

- A calling context is a sequence of call sites
- Compute a solution for a function under each such calling context
- Scalability is the biggest challenge