Program Analysis (70020)

Control Flow Analysis

Herbert Wiklicky

Department of Computing
Imperial College London

herbert@doc.ic.ac.uk
h.wiklicky@imperial.ac.uk

Autumn 2023

Control Flow Analysis

- Flow information is essential for the specification of Data Flow Analyses. In the case of the Monotone Framework, flow information is represented by the flow function **F**.
- WHILE language: flow information can be extracted directly from the program text. Procedure calls are performed by explictly mentioning the name of a procedure.
- Not so trivial for more general languages e.g imperative languages with procedures as parameters, functional languages or object-oriented languages.
- A special analysis is required: Control Flow Analysis

The λ-Calculus

$$\begin{array}{lcll} N & \in & \textbf{Term} & \lambda\text{-terms} \\ x & \in & \textbf{Var} & \text{variables} \end{array}$$

$$N \quad ::= \quad x \mid (\lambda x.N) \mid (N_1 N_2)$$

Substitution: $(\lambda x.M)N \longrightarrow_\beta M[x/N]$

$$\begin{array}{rcl} (\lambda x.x)z & \longrightarrow_\beta & z \\ (\lambda x.x)(\lambda y.y) & \longrightarrow_\beta & (\lambda y.y) \end{array}$$

Syntax of `Fun`

$$\begin{array}{lcll} e & \in & \textbf{Exp} & \text{expressions (or labelled terms)} \\ t & \in & \textbf{Term} & \text{terms (or unlabelled expressions)} \end{array}$$

$$\begin{array}{lcl} e & ::= & t^\ell \\ t & ::= & c \mid x \mid \texttt{fn}\ x\ \texttt{=>}\ e_0 \mid e_1\ e_2 \\ & & \mid \texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \mid e_1\ op\ e_2 \\ & & \mid \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \end{array}$$

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

An Example

$$
\begin{array}{ll}
\texttt{let} & f = \texttt{fn}\ x \texttt{ => } x\ 1\texttt{;} \\
 & g = \texttt{fn}\ y \texttt{ => } y + 2\texttt{;} \\
 & h = \texttt{fn}\ z \texttt{ => } z + 3 \\
\texttt{in} & (f\ g) + (f\ h)
\end{array}
$$

$$
\begin{array}{lll}
(f\ g) + (f\ h) & \longrightarrow & ((\texttt{fn}\ x \texttt{ => } x\ 1)\ g) + ((\texttt{fn}\ x \texttt{ => } x\ 1)\ h) \\
 & \longrightarrow & (g\ 1) + (h\ 1) \\
 & \longrightarrow & ((\texttt{fn}\ y \texttt{ => } y + 2)\ 1) + (\texttt{fn}\ z \texttt{ => } z + 3)\ 1) \\
 & \longrightarrow & (1 + 2) + (1 + 3) \\
 & \longrightarrow & 7
\end{array}
$$

Evaluating Fun

$\rho \in$ **Env**	=	**Var** $\mapsto$ **Value**	Environments
$v \in$ **Value**	=	**Constant** $\cup$ **Closure**	Values
Closure	::=	$[(\texttt{fn}\ x \texttt{ => } e_0), \rho]$	Closures

$$\boxed{\mathrm{eval}(\rho, e) = v}$$

iff "e evaluates to v in ρ"

- $\mathrm{eval}(\rho, e) = v$ can also be read as an specification for building an interpreter for the Fun language.
- We will use this specification just as a aid to help us understand the Control Flow Analysis.

Environment Rules I [Provided in Exam]

$$\mathrm{eval}(\rho, c^{\ell}) = c$$

$$\mathrm{eval}(\rho, x^{\ell}) = \rho(x)$$

$$\mathrm{eval}(\rho, (t_1^{\ell_1}\, op\, t_2^{\ell_2})^{\ell}) = \mathrm{eval}(\rho, t_1^{\ell_1})\; op\; \mathrm{eval}(\rho, t_2^{\ell_2})$$

$$\mathrm{eval}(\rho, (\texttt{if } t_0^{\ell_0} \texttt{ then } t_1^{\ell_1} \texttt{ else } t_2^{\ell_2})^{\ell}) = v$$

$$\textit{where}\;\; v = \begin{cases} \mathrm{eval}(\rho, t_1^{\ell_1}) \text{ for } \mathrm{eval}(\rho, t_0^{\ell_0}) = \texttt{true} \\ \mathrm{eval}(\rho, t_2^{\ell_2}) \text{ for } \mathrm{eval}(\rho, t_0^{\ell_0}) = \texttt{false} \end{cases}$$

Environment Rules II [Provided in Exam]

$$\mathrm{eval}(\rho, (\texttt{fn } x \texttt{ => } e_0)^{\ell}) = [(\texttt{fn } x \texttt{ => } e_0), \rho]$$ closure creation

$$\mathrm{eval}(\rho, (\texttt{let } x = t_1^{\ell_1} \texttt{ in } t_2^{\ell_2})^{\ell}) = \mathrm{eval}(\rho[x \mapsto v_1], t_2^{\ell_2})$$

$$\textit{where}\;\; v_1 = \mathrm{eval}(\rho, t_1^{\ell_1})$$

$$\mathrm{eval}(\rho, (t_1^{\ell_1}\;\, t_2^{\ell_2})^{\ell}) = \mathrm{eval}(\rho_0[x \mapsto v_2], e_0)$$ function application

$$\textit{where}\quad \begin{array}{l} \mathrm{eval}(\rho, t_1^{\ell_1}) = [(\texttt{fn } x \texttt{ => } e_0), \rho_0] \wedge \\ \mathrm{eval}(\rho, t_2^{\ell_2}) = v_2 \end{array}$$

Control Flow Analysis (CFA)

As we allow variables/names to be bound/associated to/with values as well as functions (closures) any function application only makes sense in an environment ρ or context:

$$\ldots (f\ 3) \ldots \quad \text{or better} \quad \ldots (f^{\ell_1}\ 3^{\ell_2})^{\ell_3} \ldots$$

It might be that $f \mapsto 3^{\ell'}$ (constant) or $f \mapsto (\texttt{fn}\ x \texttt{ => } x^{\ell'})^{\ell''}$ (identity) or $f \mapsto (\texttt{fn}\ x \texttt{ => } (x^{\ell'}\ x^{\ell''}))^{\ell'''}$ (doubling).

In our imperative setting WHILE we might also allow variables to point to programs, e.g. $\ldots \mid [p := S]^{\ell} \mid p \mid \ldots$ Then, e.g.

$$\textbf{if } b \textbf{ then } [p := S_1]^1 \textbf{ else } [p := S_2]^2;\ p$$

leads to the the question whether $(1, init(S_1))$ and/or $(1, init(S_2))$ should be in the control *flow*.

CFA and Functional Programs

Consider the following `Fun` program:

$$\begin{array}{ll} \texttt{let} & f = \texttt{fn}\ x \texttt{ => } x\ 1\texttt{;} \\ & g = \texttt{fn}\ y \texttt{ => } y + 2\texttt{;} \\ & h = \texttt{fn}\ z \texttt{ => } z + 3 \\ \texttt{in} & (f\ g) + (f\ h) \end{array}$$

The aim of Control Flow Analysis is:

For each function application, which functions may be applied

Overview

- Control Flow Analysis
  - Abstract Domains and Specification
  - Contraint Generation
  - Constraint Solving Algorithm
- Control and Data Flow Analysis
- Context-Sensitive Analysis Concepts

0-CFA Analysis

We will define a 0-CFA Analysis; the presentation requires two components:

- Abstract Domains
- Specification of the Analysis

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

- $\widehat{\mathsf{C}}$ is the abstract cache associating abstract values with each labelled program point.
- $\widehat{\rho}$ is the abstract environment associating abstract values with each variable.

Abstract Domains

An abstract value $\widehat{v}$ is a set of **terms** of the form: `fn` x `=>` e_0

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

Compare this with the Concrete Domain (see before):

$$
\begin{array}{llcll}
\rho & \in & \mathbf{Env} & = & \mathbf{Var} \to \mathbf{Val} & \text{environments} \\
v & \in & \mathbf{Val} & = & \mathbf{Z} \cup \mathbf{Closure} & \text{values} \\
 & & \mathbf{Closure} & ::= & [\texttt{fn}\ x\ \texttt{=>}\ e_0, \rho] & \text{closures}
\end{array}
$$

Acceptable CFA

For the formulation of the 0-CFA analysis we shall write

$$(\widehat{C}, \widehat{\rho}) \models e$$

for when $(\widehat{C}, \widehat{\rho})$ is an acceptable Control Flow Analysis of the expression e. Thus the relation "$\models$" has functionality

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

Our Goal therefore is:

If a sub-expression t^ℓ evaluates to a function (closure), then the function must be "predicted" by $\widehat{C}(\ell)$

CFA: Example

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

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

Specification: Rules I

$$(\widehat{C}, \widehat{\rho}) \models_s c^{\ell} \text{ always}$$

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

$$\begin{aligned}
&(\widehat{C}, \widehat{\rho}) \models_s (\texttt{if}\ t_0^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^{\ell} \\
&\quad \text{iff} \quad (\widehat{C}, \widehat{\rho}) \models_s t_0^{\ell_0} \ \wedge \\
&\qquad\qquad (\widehat{C}, \widehat{\rho}) \models_s t_1^{\ell_1} \ \wedge\ (\widehat{C}, \widehat{\rho}) \models_s t_2^{\ell_2} \ \wedge \\
&\qquad\qquad \widehat{C}(\ell_1) \subseteq \widehat{C}(\ell) \ \wedge\ \widehat{C}(\ell_2) \subseteq \widehat{C}(\ell)
\end{aligned}$$

$$\begin{aligned}
&(\widehat{C}, \widehat{\rho}) \models_s (\texttt{let}\ x = t_1^{\ell_1}\ \texttt{in}\ t_2^{\ell_2})^{\ell} \\
&\quad \text{iff} \quad (\widehat{C}, \widehat{\rho}) \models_s t_1^{\ell_1} \ \wedge\ (\widehat{C}, \widehat{\rho}) \models_s t_2^{\ell_2} \ \wedge \\
&\qquad\qquad \widehat{C}(\ell_1) \subseteq \widehat{\rho}(x) \ \wedge\ \widehat{C}(\ell_2) \subseteq \widehat{C}(\ell)
\end{aligned}$$

Specification: Rules II

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

$$(\widehat{\mathsf{C}}, \widehat{\rho}) \models_s (\texttt{fn}\ x \texttt{ => } e_0)^{\ell}$$
$$\text{iff} \quad \{\texttt{fn}\ x \texttt{ => } e_0\} \subseteq \widehat{\mathsf{C}}(\ell) \;\wedge (\widehat{\mathsf{C}}, \widehat{\rho}) \models_s e_0$$

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

Constraint Generation

To implement the specification, we must generate a set of constraints from a given program. $\mathcal{C}_\star[\![e_\star]\!]$ is a set of constraints and conditional constraints of the form

$$lhs \subseteq rhs$$

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

where *rhs* is of the form $\mathsf{C}(\ell)$ or $\mathsf{r}(x)$, and *lhs* is of the form $\mathsf{C}(\ell), \mathsf{r}(x)$, or $\{t\}$, and all occurrences of t are of the form $\texttt{fn}\ x \texttt{ => } e_0$.

Constraint-Based CFA I

$$(\widehat{\mathsf{C}}, \widehat{\rho}) \models_s (\texttt{fn}\ x \texttt{ => } e_0)^\ell$$
$$\text{iff}\quad \{\texttt{fn}\ x \texttt{ => } e_0\} \subseteq \widehat{\mathsf{C}}(\ell) \ \wedge (\widehat{\mathsf{C}}, \widehat{\rho}) \models_s e_0$$

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

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

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

Constraint-Based CFA II

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

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

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

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

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

Contraint Generation: Example I

$$\mathcal{C}_\star[\![((\texttt{fn}\ x \texttt{ => } x^1)^2\ (\texttt{fn}\ y \texttt{ => } y^3)^4)^5]\!] =$$

$$\begin{aligned} &\mathcal{C}_\star[\![(\texttt{fn}\ x \texttt{ => } x^1)^2]\!] \cup \mathcal{C}_\star[\![(\texttt{fn}\ y \texttt{ => } y^3)^4]\!] \\ \cup &\{\{t\} \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(x) \mid t = (\texttt{fn}\ x \texttt{ => } t_0^{\ell_0}) \in \mathbf{Term}_\star\} \\ \cup &\{\{t\} \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(\ell_0) \subseteq \mathsf{C}(5) \mid t = (\texttt{fn}\ x \texttt{ => } t_0^{\ell_0}) \in \mathbf{Term}_\star\} \end{aligned}$$

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

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

Contraint Generation: Example II

$$\{\{t\} \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(x) \mid t = (\texttt{fn}\ x \texttt{ => } t_0^{\ell_0}) \in \mathbf{Term}_\star\}$$

$$\begin{aligned} = \{\ &\texttt{fn}\ x \texttt{ => } x^1 \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(\texttt{x}), \\ &\texttt{fn}\ y \texttt{ => } y^3 \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{r}(\texttt{y})\ \} \end{aligned}$$

$$\{\{t\} \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(\ell_0) \subseteq \mathsf{C}(5) \mid t = (\texttt{fn}\ x \texttt{ => } t_0^{\ell_0}) \in \mathbf{Term}_\star\}$$

$$\begin{aligned} = \{\ &\texttt{fn}\ x \texttt{ => } x^1 \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(1) \subseteq \mathsf{C}(5), \\ &\texttt{fn}\ y \texttt{ => } y^3 \subseteq \mathsf{C}(2) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(5)\ \} \end{aligned}$$

Contraint Generation: Example III

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

Constraint Solving

To solve the constraints, we use a graph-based formulation.
The algorithm uses the following main data structures:

- a worklist W, i.e. a list of nodes whose outgoing edges should be traversed;
- a data array D that for each node gives an element of $\widehat{\mathbf{Val}}_\star$; and
- an edge array E that for each node gives a list of constraints from which a list of the successor nodes can be computed.

Constraints Graph

The graph will have nodes $\mathsf{C}(\ell)$ and $\mathsf{r}(x)$ for $\ell \in \mathbf{Lab}_\star$ and $x \in \mathbf{Var}_\star$. Associated with each node p we have a data field D[p] that initially is given by:

$$\mathrm{D[p]} = \{t \mid (\{t\} \subseteq p) \in \mathcal{C}_\star[\![e_\star]\!]\}$$

The graph will have edges for a subset of the constraints in $\mathcal{C}_\star[\![e_\star]\!]$; each edge will be decorated with the constraint that gives rise to it:

- a constraint $p_1 \subseteq p_2$ gives rise to an edge from p_1 to p_2, and
- a constraint $\{t\} \subseteq p \Rightarrow p_1 \subseteq p_2$ gives rise to an edge from p_1 to p_2 *and* an edge from p to p_2.

Algorithm I

INPUT: $\mathcal{C}_\star[\![e_\star]\!]$

OUTPUT: $(\widehat{\mathsf{C}}, \widehat{\rho})$

METHOD: Step 1: Initialisation

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

Algorithm II

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]);

Algorithm III

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]);

Algorithm IV

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)];

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

Example I

p	D[p]	E[p]
C(1)	$\emptyset$	[id$_x$⊆C(2) ⇒ C(1)⊆C(5)]
C(2)	id$_x$	[id$_y$⊆C(2) ⇒ C(3)⊆C(5), id$_y$⊆C(2) ⇒ C(4)⊆r(y), id$_x$⊆C(2) ⇒ C(1)⊆C(5), id$_x$⊆C(2) ⇒ C(4)⊆r(x)]
C(3)	$\emptyset$	[id$_y$⊆C(2) ⇒ C(3)⊆C(5)]
C(4)	id$_y$	[id$_y$⊆C(2) ⇒ C(4)⊆r(y), id$_x$⊆C(2) ⇒ C(4)⊆r(x)]
C(5)	$\emptyset$	[ ]
r(x)	$\emptyset$	[r(x)⊆C(1)]
r(y)	$\emptyset$	[r(y)⊆C(3)]

Example II

W	[C(4),C(2)]	[r(x),C(2)]	[C(1),C(2)]	[C(5),C(2)]	[C(2)]	[ ]
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)	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Control Flow + Data Flow

Let **Data** be a set of *abstract data values* (i.e. abstract properties of booleans and arithmetic constants)

$$\widehat{v} \in \widehat{\mathbf{Val}}_d = \mathcal{P}(\mathbf{Term} \cup \mathbf{Data}) \quad \text{abstract values}$$

For each constant $c \in \mathbf{Const}$ we need an element $d_c \in \mathbf{Data}$
Similarly, for each operator $op \in \mathbf{Op}$ we need a total function

$$\widehat{\mathrm{op}} : \widehat{\mathbf{Val}}_d \times \widehat{\mathbf{Val}}_d \to \widehat{\mathbf{Val}}_d$$

Typically, $\widehat{\mathrm{op}}$ will have a definition of the form:

$$\widehat{v}_1 \ \widehat{\mathrm{op}} \ \widehat{v}_2 = \bigcup \{ d_{op}(d_1, d_2) \mid d_1 \in \widehat{v}_1 \cap \mathbf{Data}, d_2 \in \widehat{v}_2 \cap \mathbf{Data} \}$$

for some function $d_{op} : \mathbf{Data} \times \mathbf{Data} \to \mathcal{P}(\mathbf{Data})$

Detection of Sign

$$\mathbf{Data}_{\mathsf{sign}} = \{\mathsf{tt}, \mathsf{ff}, -, 0, +\}$$

$$d_{\texttt{true}} = \mathsf{tt} \qquad d_7 = +$$

$\widehat{+}$ is defined from:

d_+	tt	ff	$-$	0	$+$
tt	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
ff	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$-$	$\emptyset$	$\emptyset$	$\{-\}$	$\{-\}$	$\{-, 0, +\}$
0	$\emptyset$	$\emptyset$	$\{-\}$	$\{0\}$	$\{+\}$
$+$	$\emptyset$	$\emptyset$	$\{-, 0, +\}$	$\{+\}$	$\{+\}$

Abstract Values I

$$(\widehat{\mathsf{C}}, \widehat{\rho}) \models_d (\texttt{fn}\ x \texttt{ => } e_0)^\ell \text{ iff } \{\texttt{fn}\ x \texttt{ => } e_0\} \subseteq \widehat{\mathsf{C}}(\ell) \wedge (\widehat{\mathsf{C}}, \widehat{\rho}) \models_d e_0$$

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

$$\begin{array}{l} (\widehat{\mathsf{C}}, \widehat{\rho}) \models_d (\texttt{if}\ t_0^{\ell_0}\ \texttt{then}\ t_1^{\ell_1}\ \texttt{else}\ t_2^{\ell_2})^\ell \\ \quad \text{iff} \quad (\widehat{\mathsf{C}}, \widehat{\rho}) \models_d t_0^{\ell_0} \wedge \\ \qquad\quad (d_{\texttt{true}} \in \widehat{\mathsf{C}}(\ell_0) \Rightarrow ((\widehat{\mathsf{C}}, \widehat{\rho}) \models_d t_1^{\ell_1} \ \wedge\ \widehat{\mathsf{C}}(\ell_1) \subseteq \widehat{\mathsf{C}}(\ell))) \wedge \\ \qquad\quad (d_{\texttt{false}} \in \widehat{\mathsf{C}}(\ell_0) \Rightarrow ((\widehat{\mathsf{C}}, \widehat{\rho}) \models_d t_2^{\ell_2} \ \wedge\ \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\mathsf{C}}(\ell))) \end{array}$$

Abstract Values II

$$(\widehat{\mathsf{C}}, \widehat{\rho}) \models_d c^\ell \text{ iff } \{d_c\} \subseteq \widehat{\mathsf{C}}(\ell)$$

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

$$\begin{aligned}(\widehat{\mathsf{C}}, \widehat{\rho}) \models_d (\texttt{let } x = t_1^{\ell_1} \texttt{ in } t_2^{\ell_2})^\ell & \\ \text{iff} \quad & (\widehat{\mathsf{C}}, \widehat{\rho}) \models_d t_1^{\ell_1} \ \wedge \ (\widehat{\mathsf{C}}, \widehat{\rho}) \models_d t_2^{\ell_2} \ \wedge \\ & \widehat{\mathsf{C}}(\ell_1) \subseteq \widehat{\rho}(x) \ \wedge \ \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\mathsf{C}}(\ell)\end{aligned}$$

$$\begin{aligned}(\widehat{\mathsf{C}}, \widehat{\rho}) \models_d (t_1^{\ell_1} \ op \ t_2^{\ell_2})^\ell & \\ \text{iff} \quad & (\widehat{\mathsf{C}}, \widehat{\rho}) \models_d t_1^{\ell_1} \ \wedge \ (\widehat{\mathsf{C}}, \widehat{\rho}) \models_d t_2^{\ell_2} \ \wedge \\ & \widehat{\mathsf{C}}(\ell_1) \ \widehat{\mathsf{op}} \ \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\mathsf{C}}(\ell)\end{aligned}$$

Example: Sign Detection

$$\begin{aligned}&\texttt{let} \quad f = (\texttt{fn } x \texttt{ =>} \quad (\texttt{if } (x^1 > 0^2)^3 \texttt{ then } (\texttt{fn } y \texttt{ => } y^4)^5 \\ &\qquad\qquad\qquad\qquad\quad \texttt{else } (\texttt{fn } z \texttt{ => } 25^6)^7)^8)^9 \\ &\texttt{in} \quad ((f^{10} 3^{11})^{12} 0^{13})^{14})^{15}\end{aligned}$$

C(1)	$\emptyset$
C(2)	$\emptyset$
C(3)	$\emptyset$
C(4)	$\emptyset$
C(5)	id_y
C(6)	$\emptyset$
C(7)	c_{25}

C(8)	$\{\mathsf{id}_y, \mathsf{c}_{25}\}$
C(9)	$\{\texttt{fn x ...)}^8\}$
C(10)	$\{\texttt{fn x ...)}^8\}$
C(11)	$\emptyset$
C(12)	$\{\mathsf{id}_y, \mathsf{c}_{25}\}$
C(13)	$\emptyset$

C(14)	$\emptyset$
C(15)	$\emptyset$
r(f)	$\{\texttt{fn x ...)}^8\}$
r(x)	$\emptyset$
r(y)	$\emptyset$
r(z)	$\emptyset$

C(1)	$\{+\}$
C(2)	$\{0\}$
C(3)	$\{\texttt{tt}\}$
C(4)	$\{0\}$
C(5)	id_y
C(6)	$\emptyset$
C(7)	c_{25}

C(8)	$\{\mathsf{id}_y\}$
C(9)	$\{\texttt{fn x ...)}^8\}$
C(10)	$\{\texttt{fn x ...)}^8\}$
C(11)	$\{+\}$
C(12)	$\{\mathsf{id}_y\}$
C(13)	$\{0\}$

C(14)	$\{0\}$
C(15)	$\{0\}$
r(f)	$\{\texttt{fn x ...)}^8\}$
r(x)	$\{+\}$
r(y)	$\{0\}$
r(z)	$\emptyset$

A pure 0-CFA analysis will not be able to discover that the

Context-Sensitive CFA

The Control Flow Analyses presented so far are imprecise in that they cannot distinguish the various instances of function calls from one another. In the terminology of Data Flow Analysis the 0-CFA analysis is context-insensitive and in the terminology of Control Flow Analysis it is monovariant.

To get a more precise analysis it is useful to introduce a mechanism that distinguishes different dynamic instances of variables and labels from one another. This results in a context-sensitive analysis and in the terminology of Control Flow Analysis the term polyvariant is used.

Example: Context

Consider the expression:

$$\begin{array}{ll}(\texttt{let} & f = (\texttt{fn}\ x \Rightarrow x^1)^2 \\ \texttt{in} & ((f^3\ f^4)^5\ (\texttt{fn}\ y \Rightarrow y^6)^7)^8)^9\end{array}$$

The least 0-CFA analysis is given by $(\widehat{\mathsf{C}}_{\mathsf{id}}, \widehat{\rho}_{\mathsf{id}})$:

0-CFA Solutions

$\widehat{C}_{id}(1) = \{\texttt{fn}\ x\texttt{=>}\ x^1, \texttt{fn}\ y\ \texttt{=>}\ y^6\}$ $\widehat{C}_{id}(2) = \{\texttt{fn}\ x\texttt{=>}\ x^1\}$
$\widehat{C}_{id}(3) = \{\texttt{fn}\ x\texttt{=>}\ x^1\}$ $\widehat{C}_{id}(4) = \{\texttt{fn}\ x\texttt{=>}\ x^1\}$
$\widehat{C}_{id}(5) = \{\texttt{fn}\ x\texttt{=>}\ x^1, \texttt{fn}\ y\ \texttt{=>}\ y^6\}$ $\widehat{C}_{id}(6) = \{\texttt{fn}\ y\ \texttt{=>}\ y^6\}$
$\widehat{C}_{id}(7) = \{\texttt{fn}\ y\ \texttt{=>}\ y^6\}$
$\widehat{C}_{id}(8) = \{\texttt{fn}\ x\texttt{=>}\ x^1, \texttt{fn}\ y\ \texttt{=>}\ y^6\}$
$\widehat{C}_{id}(9) = \{\texttt{fn}\ x\texttt{=>}\ x^1, \texttt{fn}\ y\ \texttt{=>}\ y^6\}$

$\widehat{\rho}_{id}(\texttt{f}) = \{\texttt{fn}\ x\texttt{=>}\ x^1\}$
$\widehat{\rho}_{id}(\texttt{x}) = \{\texttt{fn}\ x\texttt{=>}\ x^1, \texttt{fn}\ y\ \texttt{=>}\ y^6\}$
$\widehat{\rho}_{id}(\texttt{y}) = \{\texttt{fn}\ y\ \texttt{=>}\ y^6\}$

Expansion

Expand the program into

$$\begin{array}{ll} \texttt{let} & f_1 = (\texttt{fn}\ x_1\ \texttt{=>}\ x_1) \\ \texttt{in} & \texttt{let}\ f_2 = (\texttt{fn}\ x_2\ \texttt{=>}\ x_2) \\ & \quad \texttt{in}\ (f_1\ f_2)\ (\texttt{fn}\ y\ \texttt{=>}\ y) \end{array}$$

and then analyse the expanded expression: the 0-CFA analysis is now able to deduce that x_1 can only be bound to $\texttt{fn}\ x_2\ \texttt{=>}\ x_2$ and that x_2 can only be bound to $\texttt{fn}\ y\ \texttt{=>}\ y$ so the overall expression will evaluate to $\texttt{fn}\ y\ \texttt{=>}\ y$ only.

Further CFA Analyses

A more satisfactory solution to the problem is to extend the analysis with context information allowing it to distinguish between the various instances of variables and program points and still analyse the original expression.

Examples of such analyses include k-CFA analyses, uniform k-CFA analyses, polynomial k-CFA analyses (mainly of interest for $k > 0$) and the Cartesian Product Algorithm.