GALOIS CONNECTION BASED ABSTRACT INTERPRETATIONS FOR STRICTNESS ANALYSIS

Patrick COUSOT
École Normale Supérieure

and

Radhia COUSOT
École Polytechnique

Abstract Interpretation

Abstract Interpretation [CC77, CC79] is method for constructing conservative approximations of the semantics of programming languages.

Abstract Interpretation is used to:

- Specify hierarchies of semantics of programming languages at different levels of abstraction;
- Design program proof methods;
- Specify automatic program analyzers (by interpretation of programs in abstract domains);
- Etc.

Strictness analysis

Strictness analysis [Myc80] is an abstract interpretation, due to Alan Mycroft, for determining statically which *call-by-need* parameters of lazy functional programs can be replaced by *call-by-value*.

Traditional example (addition):

$$\texttt{f(x, y)} \equiv \texttt{((x = 0)} \rightarrow \texttt{y, (1 + f(x - 1, y)))}$$

- `x` is always evaluated on first call, hence `x` can be passed by-value;
- `y` is evaluated on final call or `f` does not terminate, hence `y` can be passed by-value.

Strictness analysis by abstract interpretation

The traditional abstract interpretation framework using:

- An operational-based collecting semantics;
- Fixpoints of monotone operators on complete lattices;
- Galois connections;

was considered difficult to apply to strictness analysis because one had to use denotational semantics to take non-termination into account [MN83, Nie88].

Critique of the denotational theory of abstract interpretation

The simplicity of the original abstract interpretation is lost:

- CPOs/powerdomains are more complicated than powersets/-complete lattices;
- Analysis inversion is lost: denotational semantics is well-suited for forward analyses but present difficulties for backward analyses;
- Logical relations are weaker than Galois connections: the constructive aspect of the original abstract interpretation framework is lost (only safeness verification remains);

Objectives

- Objective of the paper:

Show that the Galois connection-based abstract interpretation framework is applicable to strictness analysis.

- Next objectives:

  Use this abstract interpretation framework to compare the strictness analysis algorithms known in the literature with Mycroft's method:

  - Forward and backward analyses are isomorphic;
  - Projection analysis is a very simple variant.

Plan

1. Relational semantics;
2. The Galois connection-based abstract interpretation framework;
3. Application to Mycroft's strictness analysis algorithm;
4. Principle of Johnson's algorithm;
5. Using widening operators as a compromise between the precision of Mycroft's algorithm and the efficiency of Johnson's algorithm.

Relational semantics

- Represent a computation by a relation between initial and final states (Ω for run-time errors, $\bot$ for non-termination);
- Rule-based presentation using "iterated well-founded systems of bi-inductive definitions" [CC92d];
- Equivalent presentation based upon fixpoints of monotonic operators on complete lattices.

A lazy first-order functional language

Syntax of expressions

e	$::=$	k	constant
	$\mid$	v	variable (formal parameter)
	$\mid$	$b\vec{e}$	basic operation
	$\mid$	$f\vec{e}$	function call
	$\mid$	$(e_1 \rightarrow e_2,\ e_3)$	conditional
$\vec{v}$	$::=$	$(v_1, \ldots, v_n)$	tuple of formal parameters
$\vec{e}$	$::=$	$(e_1, \ldots, e_n)$	tuple of actual arguments

Syntax of programs

$$\prod_{f \epsilon \vec{f}} f\vec{v} \equiv \vec{F}[f]$$

is a shorthand for:

$$\begin{cases} f_1(v_1, \ldots, v_{n_1}) \equiv e_1 \\ \cdots \\ f_k(v_1, \ldots, v_{n_k}) \equiv e_k \end{cases}$$

where the body $\vec{F}[f_i] = e_i$ of function f_i depends on the parameters $\vec{v} = (v_1, \ldots, v_{i_1})$ and may call other functions $f_j, j = 1, \ldots, k$.

Relational semantics

- The semantics $f^{\Re}$ of a function f is a relation between the values of its actual parameters and the corresponding result;
- These values and results may include run-time errors Ω and non-termination $\bot$;
- Fonctions may be non-deterministic (for example

  $$\texttt{f}() \ \equiv \ ?();$$

  returns a random natural number);

Semantic domains

Υ			values of variables
Υ^{Ω}	$\stackrel{\text{def}}{=}$	$\Upsilon \cup \{\Omega\}$	values or errors
$\Upsilon_{\bot}$	$\stackrel{\text{def}}{=}$	$\Upsilon \cup \{\bot\}$	values or non-termination
$\Upsilon_{\bot}^{\Omega}$	$\stackrel{\text{def}}{=}$	$\Upsilon \cup \{\Omega, \bot\}$	values, errors or non-termination
$\mathcal{D}^{\Re}$	$\stackrel{\text{def}}{=}$	$\Upsilon_{\bot}^{\Omega}$	values of expressions
$\vec{\mathcal{D}}^{\Re}$	$\stackrel{\text{def}}{=}$	$\prod_{v \epsilon \vec{v}} \mathcal{D}^{\Re}$	values of tuples of expressions
$\mathcal{F}^{\Re}$	$\stackrel{\text{def}}{=}$	$\wp(\vec{\mathcal{D}}^{\Re} \times \mathcal{D}^{\Re})$	values of functions

(The semantics $f^{\Re}$ of a function f is a relation between the values $\vec{\nu}$ of its actual parameters and the corresponding result $f(\vec{\nu})$.)

Fixpoint presentation of the relational semantics

The relational semantics $\vec{f}^{\Re}$ of the program:

$$\prod_{f \epsilon \vec{f}} f\vec{v} \equiv \vec{F}[f]$$

is the least fixpoint:

$$\vec{f}^{\Re} \stackrel{\text{def}}{=} \mathrm{lfp}^{\vec{\sqsubseteq}^*}_{\vec{\perp}^*} \vec{F}^{\Re}$$

of a monotonic operator:

$$\vec{F}^{\Re} \in \vec{\mathcal{F}}^{\Re} \stackrel{m}{\longmapsto} \vec{\mathcal{F}}^{\Re} \qquad \vec{F}^{\Re} \stackrel{\text{def}}{=} \lambda\vec{\varphi} \cdot \prod_{f \epsilon \vec{f}} \vec{F}[f]^{\Re}[\![\vec{\varphi}]\!]$$

on a complete lattice:

$$\vec{\mathcal{F}}^{\Re}(\vec{\sqsubseteq}^{\Re}, \vec{\perp}^{\Re}, \vec{\top}^{\Re}, \vec{\sqcup}^{\Re}, \vec{\sqcap}^{\Re})$$

Example of fixpoint presentation of the relational semantics

For the program:

```
f(x) ≡ (x = 0 → 0, (x < 0 → f(?()), f(x - 1)))
```

the fixpoint equation is:

$$\vec{\varphi} = \vec{F}^{\Re}[f][\![\vec{\varphi}]\!]$$

where:

$$\begin{aligned} \vec{F}^{\Re}[f][\![\vec{\varphi}]\!] = \; & \{\langle \bot, \bot\rangle, \langle \Omega, \Omega\rangle, \langle 0, 0\rangle\} \\ & \cup \{\langle x, y\rangle \mid x < 0 \wedge \exists n \geq 0 : \langle n, y\rangle \in \vec{\varphi}[f]\} \\ & \cup \{\langle x, y\rangle \mid x > 0 \wedge \langle x-1, y\rangle \in \vec{\varphi}[f]\} \end{aligned}$$

The transfinite iterates $\varphi^{\lambda} = \vec{F}^{\Re}[f]^{\lambda}(\bot^{\Re})$ are:

$$\begin{aligned} \varphi^{0} &= \bot^{\Re} = \mathbb{Z}^{\Omega}_{\bot} \times \{\bot\} \\ \varphi^{1} &= \{\langle \bot, \bot\rangle, \langle \Omega, \Omega\rangle, \langle 0, 0\rangle\} \cup \{\langle x, \bot\rangle \mid x \neq 0\} \\ \varphi^{2} &= \{\langle \bot, \bot\rangle, \langle \Omega, \Omega\rangle\} \cup \{\langle x, 0\rangle \mid x < 2\} \cup \\ &\qquad \{\langle x, \bot\rangle \mid x < 0 \vee x \geq 2\} \\ &\cdots \\ \varphi^{n} &= \{\langle \bot, \bot\rangle, \langle \Omega, \Omega\rangle\} \cup \{\langle x, 0\rangle \mid x < n\} \cup \\ &\qquad \{\langle x, \bot\rangle \mid x < 0 \vee x \geq n\} \\ &\cdots \\ \varphi^{\omega} &= \{\langle \bot, \bot\rangle, \langle \Omega, \Omega\rangle\} \cup \{\langle x, 0\rangle \mid x \in \mathbb{Z}\} \cup \{\langle x, \bot\rangle \mid x < 0\} \\ \varphi^{\omega+1} &= \{\langle \bot, \bot\rangle, \langle \Omega, \Omega\rangle\} \cup \{\langle x, 0\rangle \mid x \in \mathbb{Z}\} \\ \varphi^{\omega+2} &= \varphi^{\omega+1} \end{aligned}$$

proving that the program returns 0 for all integer parameters.

Computational ordering

- Initially, non termination is assumed for all actual parameters:
$$\bot^{\Re} \stackrel{\text{def}}{=} \vec{\Upsilon}^{\Omega}_{\bot} \times \{\bot\}$$
- Terminating functions are a subset of:
$$\top^{\Re} \stackrel{\text{def}}{=} \vec{\Upsilon}^{\Omega}_{\bot} \times \Upsilon^{\Omega}$$
- Each iterate introduces new possible finite behaviors and eliminates previous infinite behaviors now shown to be impossible:
$$\varphi \sqsubseteq^{\Re} \varphi' \stackrel{\text{def}}{=} (\varphi \cap \top^{\Re}) \subseteq (\varphi' \cap \top^{\Re}) \;\wedge\; (\varphi \cap \bot^{\Re}) \supseteq (\varphi' \cap \bot^{\Re})$$
- Passing to the limits collects the possible finite behaviors and the infinite behaviors which are not impossible:
$$\bigsqcup^{\Re}_{i \in \Delta} \varphi_i \stackrel{\text{def}}{=} \bigcup_{i \in \Delta} (\varphi_i \cap \top^{\Re}) \cup \bigcap_{i \in \Delta} (\varphi_i \cap \bot^{\Re})$$

Fixpoint operator

The fixpoint operator:

$$\vec{F}^{\Re}$$

associated with a program:

$$\prod_{f \epsilon \vec{f}} f\vec{v} \equiv \vec{F}[f]$$

is defined componentwise for each function $f \epsilon \vec{f}$ of the program, by induction on the syntax of the body of this function f:

$$e = \vec{F}[f]$$

Relational semantics of an expression

- The relational semantics $e^{\Re}$ of an expression e:
  - is a set of pairs $\langle \vec{\nu},\ \nu \rangle$ specifying the possible values ν of expression e given the values $\vec{\nu}[v]$ of the variables v which are free within e;
  - it depends on the relational semantics $\vec{\varphi}[f]$ of the functions f of the program called within e;
- $\langle \vec{\nu},\ \bot \rangle \in e^{\Re}$ means that non-termination is possible for values $\vec{\nu}$ of the free variables;
- $\langle \vec{\nu},\ \Omega \rangle \in e^{\Re}$ means that a run-time error is possible for values $\vec{\nu}$ of the free variables;

Constant

- The evaluation of a constant k in a function body always terminates and returns its value $\underline{k}$;
- The relational semantics $k^{\Re}$ of the constant k is therefore a relation which holds between the vector of values $\vec{\nu}$ of the parameters $\vec{v}$ and the value $\underline{k}$ of this constant k:

$$k^{\Re}[\![\vec{\varphi}]\!] \stackrel{\text{def}}{=} \{\langle \vec{\nu},\ \underline{k} \rangle \mid \vec{\nu} \in \vec{\mathcal{D}}^{\Re}\}$$

Formal parameter

- The evaluation of a formal parameter v in a function body returns the value $\vec{\nu}[v]$ of the corresponding actual parameter;
- The relational semantics $v^{\Re}$ of the variable v is therefore a relation which holds between the vector of values $\vec{\nu}$ of the parameters and the actual value $\vec{\nu}[v]$ of this variable v:

$$v^{\Re}[\![\vec{\varphi}]\!] \stackrel{\text{def}}{=} \{\langle \vec{\nu},\ \vec{\nu}[v] \rangle \mid \vec{\nu} \in \vec{\mathcal{D}}^{\Re}\}$$

- In particular this value $\vec{\nu}[v]$ may be $\bot$ if the evaluation of the actual parameter does not terminate, and Ω if it is erroneous.

Actual parameters

- The evaluation of a list of actual parameters $\vec{e}$ consists in evaluating each parameter $\vec{e}\,[v], v \in \vec{e}$ in the list;
- The relational semantics $\vec{e}^{\,\Re}$ of the vector of expressions $\vec{e}$ is therefore a relation which holds between the vector of values $\vec{\nu}$ of the parameters used in these expressions and the actual values $\vec{\nu}'[v]$ of these expressions $\vec{e}\,[v]$, $v \in \vec{e}$:

$$\vec{e}^{\,\Re}[\![\vec{\varphi}]\!] \stackrel{\text{def}}{=} \{\langle \vec{\nu},\ \vec{\nu}' \rangle \mid \forall v \in \vec{e} : \langle \vec{\nu},\ \vec{\nu}'[v] \rangle \in \vec{e}\,[v]^{\Re}[\![\vec{\varphi}]\!]\}$$

- This evaluation:
  - may terminate: $\vec{\nu}'[v] \in \Upsilon$;
  - may be erroneous: $\vec{\nu}'[v] = \Omega$;
  - may not terminate: $\vec{\nu}'[v] = \bot$;

Basic operation

The relational semantics of a basic operation b is specified by a total relation $b^{\Re} \in \mathbb{P}(\vec{\Upsilon}^{\Omega}_{\bot} \times \Upsilon^{\Omega}_{\bot})$:

$$b\vec{e}^{\,\Re}[\![\vec{\varphi}]\!] \stackrel{\text{def}}{=} \vec{e}^{\,\Re}[\![\vec{\varphi}]\!] \circ b^{\Re}$$

Example (left-to-right addition):

$$\begin{aligned} +^{\Re} \stackrel{\text{def}}{=}\ & \{\langle\langle \bot,\ \nu' \rangle,\ \bot \rangle \mid \nu' \in \Upsilon^{\Omega}_{\bot}\} \cup \\ & \{\langle\langle \nu,\ \nu' \rangle,\ \Omega \rangle \mid \nu \in \Upsilon^{\Omega} - \mathbb{Z} \wedge \nu' \in \Upsilon^{\Omega}_{\bot}\} \cup \\ & \{\langle\langle \nu,\ \bot \rangle,\ \bot \rangle \mid \nu \in \mathbb{Z}\} \cup \\ & \{\langle\langle \nu,\ \nu' \rangle,\ \Omega \rangle \mid \nu \in \Upsilon \wedge \nu' \in \Upsilon^{\Omega} - \mathbb{Z}\} \cup \\ & \{\langle\langle \nu,\ \nu' \rangle,\ \nu + \nu' \rangle \mid \nu \in \mathbb{Z} \wedge \nu' \in \mathbb{Z}\} \end{aligned}$$

Conditional

$$\begin{aligned} (e_1 \rightarrow e_2,\ e_3)^{\Re}[\![\vec{\varphi}]\!] \stackrel{\text{def}}{=}\ & \\ & \{\langle \vec{\nu},\ \bot \rangle \mid \langle \vec{\nu},\ \bot \rangle \in e_1^{\Re}[\![\vec{\varphi}]\!]\} \\ & \cup \{\langle \vec{\nu},\ \Omega \rangle \mid \exists \nu \in \Upsilon^{\Omega} - \{\mathrm{tt},\ \mathrm{ff}\} : \langle \vec{\nu},\ \nu \rangle \in e_1^{\Re}[\![\vec{\varphi}]\!]\} \\ & \cup \{\langle \vec{\nu},\ \nu \rangle \mid \langle \vec{\nu},\ \mathrm{tt} \rangle \in e_1^{\Re}[\![\vec{\varphi}]\!] \wedge \langle \vec{\nu},\ \nu \rangle \in e_2^{\Re}[\![\vec{\varphi}]\!]\} \\ & \cup \{\langle \vec{\nu},\ \nu \rangle \mid \langle \vec{\nu},\ \mathrm{ff} \rangle \in e_1^{\Re}[\![\vec{\varphi}]\!] \wedge \langle \vec{\nu},\ \nu \rangle \in e_3^{\Re}[\![\vec{\varphi}]\!]\} \end{aligned}$$

Evaluation of the conditional $(e_1 \rightarrow e_2,\ e_3)$:

- does not terminate, if evaluation of e_1 does not terminate;
- is erroneous, if evaluation of e_1 does not terminate;
- is the value of $e_2^{\Re}$, if evaluation of e_1 is true;
- is the value of $e_3^{\Re}$, if evaluation of e_1 is false.

Function call

The semantics of a function call:

$$f\vec{e}^{\,\Re}[\![\vec{\varphi}]\!] \stackrel{\text{def}}{=} \vec{e}^{\,\Re}[\![\vec{\varphi}]\!] \circ \vec{\varphi}\,[f]$$

is obtained by composition of the semantics $\vec{\varphi}\,[f]$ of function f and the semantics $\vec{e}^{\,\Re}$ of the actual arguments $\vec{e}$.

Call-by-need versus call-by-name: example

Call-by-value or call-by-need `f(x) ≡ x + x`:

$$f^{\Re} \stackrel{\text{def}}{=} \{\langle \bot, \bot\rangle, \langle \Omega, \Omega\rangle\} \cup \{\langle x, 2x\rangle \mid x \in \mathbb{Z}\}$$

Non-deterministic choice 1□2 with free variable `x`:

$$1\square 2^{\Re} \stackrel{\text{def}}{=} \{\langle x, 1\rangle \mid x \in \mathbb{Z}^{\Omega}_{\bot}\} \cup \{\langle x, 2\rangle \mid x \in \mathbb{Z}^{\Omega}_{\bot}\}$$

Call-by-need `g(x) ≡ f(1□2)`:

$$\begin{aligned} g^{\Re} &\stackrel{\text{def}}{=} 1\square 2^{\Re} \circ f^{\Re} \\ &= \{\langle x, z\rangle \mid \exists y : \langle x, y\rangle \in 1\square 2^{\Re} \wedge \langle y, z\rangle \in f^{\Re}\} \\ &= \{\langle x, 2\rangle \mid x \in \mathbb{Z}^{\Omega}_{\bot}\} \cup \{\langle x, 4\rangle \mid x \in \mathbb{Z}^{\Omega}_{\bot}\} \end{aligned}$$

Call-by-name `g(x) ≡ f(1□2)` would be:

$$\{\langle x, 2\rangle \mid x \in \mathbb{Z}^{\Omega}_{\bot}\} \cup \{\langle x, 3\rangle \mid x \in \mathbb{Z}^{\Omega}_{\bot}\} \cup \{\langle x, 4\rangle \mid x \in \mathbb{Z}^{\Omega}_{\bot}\}$$

The Galois connection-based abstract interpretation framework

1. Define a concrete collecting semantics of programs (as a fixpoint of a monotonic operator on a complete lattice) and an approximation relation (to deal with undecidability);
2. Choose an approximation of concrete properties by abstract properties (defined by a Galois connection/surjection);
3. Constructively derive the abstract semantics (specifying the abstract interpreter) from the concrete fixpoint semantics.

Principle of the fixpoint approximation

The intuitive idea is to mimic the iterative computation of the collecting semantics:

$$\mathrm{lfp}^{\sqsubseteq}_{\bot} F \;=\; \bigsqcup_{\lambda \in \mathbb{O}} F^{\lambda}(\bot)$$

in a concrete domain $\mathcal{F}(\sqsubseteq, \bot, \sqcup)$ by the abstract iterative computation:

$$\bigsqcup^{\sharp}_{\lambda \in \mathbb{O}} F^{\sharp\lambda}(\bot^{\sharp})$$

in an abstract domain $\mathcal{F}^{\sharp}(\bot^{\sharp}, F^{\sharp}, \sqcup^{\sharp})$ such that:

$$\mathrm{lfp}^{\sqsubseteq}_{\bot} F \leq \gamma\Big(\bigsqcup^{\sharp}_{\lambda \in \mathbb{O}} F^{\sharp\lambda}\big(\bot^{\sharp}\big)\Big)$$

i.e. the concretization γ of the abstract iteration is a safe approximation of the collecting semantics.

Complete lattice of concrete properties

The set of concrete properties of a program:

$$\mathcal{F}(\sqsubseteq, \bot, \top, \sqcup, \sqcap)$$

is a complete lattice for the concrete *computational ordering* $\sqsubseteq$.

Fixpoint definition of the concrete properties of a program

The concrete properties of a program are defined by the collecting semantics as the least fixpoint:

$$\mathrm{lfp}_{\bot}^{\sqsubseteq} F$$

of a monotonic operator:

$$F \in \mathcal{F} \stackrel{\mathrm{m}}{\longmapsto}{}^{\sqsubseteq} \mathcal{F}$$

on the complete lattice $\mathcal{F}(\sqsubseteq, \bot, \sqcup)$.

Constructive version of Tarski's fixpoint theorem

The least fixpoint of F greater than or equal to $\bot$ for the computational ordering $\sqsubseteq$:

$$\mathrm{lfp}_{\bot}^{\sqsubseteq} F = \bigsqcup_{\lambda \in \mathbb{O}} F^{\lambda}(\bot)$$

is obtained by transfinite iterates:

$$\begin{aligned} F^{0}(X) &= X \\ F^{\lambda+1}(X) &= F(F^{\lambda}(X)) \qquad \text{for successor ordinals } \lambda+1 \\ F^{\lambda}(X) &= \bigsqcup_{\beta<\lambda} F^{\beta}(X) \qquad \text{for limit ordinals } \lambda \end{aligned}$$

Concrete approximation relation $\leq$

- Since the collecting semantics is not effectively computable and sometimes not even computer representable, approximations must be considered;
- The concrete approximation relation:

  $$\leq \text{ is a partial order on } \mathcal{F}$$

  $\varphi \leq \psi$ means that property ψ safely approximates φ;
- The concrete semantic function preserves approximations:

  $$F \in \mathcal{F} \stackrel{\mathrm{m}}{\longmapsto}{}^{\leq} \mathcal{F}$$

Abstract approximation ordering $\leq^{\sharp}$

- We can consider an abstract version $\leq^{\sharp}$ on $\mathcal{F}^{\sharp}$ of the concrete approximation ordering $\leq$ on $\mathcal{F}$;
- The abstract approximation relation:

  $$\leq^{\sharp} \text{ is a partial order on } \mathcal{F}$$

Galois connection

The correspondence between the concrete properties $\mathcal{F}$ and the abstract properties $\mathcal{F}^\sharp$ is given by a *Galois connection*, written:

$$\mathcal{F}(\leq) \ \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \ \mathcal{F}^\sharp(\leq^\sharp)$$

$$\overset{\text{def}}{=}$$

$$\forall \varphi \in \mathcal{F} : \forall \psi^\sharp \in \mathcal{F}^\sharp : \varphi \leq \gamma(\psi^\sharp) \ \Leftrightarrow \ \alpha(\varphi) \leq^\sharp \psi^\sharp$$

Intuition behind this Galois connection

- The *concretization function* γ gives the concrete meaning $\gamma(\psi^\sharp)$ of abstract properties $\psi^\sharp$;
- The *abstraction function* α gives the best abstract approximation $\alpha(\varphi)$ of a concrete property φ:
  - $\alpha(\varphi)$ is an approximation that correctly describes φ, so:
  $$\forall \varphi \in \mathcal{F} : \varphi \leq \gamma\big(\alpha(\varphi)\big)$$
  - $\alpha(\varphi)$ is the most precise approximation that correctly describes φ, so:
  $$\forall \varphi \in \mathcal{F} : \forall \psi^\sharp \in \mathcal{F}^\sharp : \varphi \leq \gamma(\psi^\sharp) \ \Rightarrow \ \alpha(\varphi) \leq^\sharp \psi^\sharp$$

Approximation of the infimum

The abstract infimum $\bot^\sharp$ is a safe approximation of the concrete infimum $\bot$:

$$\bot^\sharp \geq^\sharp \alpha(\bot)$$

Approximation of the computational join

The abstract computational join $\bigsqcup^\sharp$ is a safe approximation of the concrete join $\bigsqcup$ of chains increasing for the computational ordering $\sqsubseteq$:

$$\Big(\forall \beta \leq \beta' < \lambda : \varphi_\beta \sqsubseteq \varphi_{\beta'} \wedge \alpha(\varphi_\beta) \leq^\sharp \psi_\beta^\sharp\Big)$$

$$\Rightarrow$$

$$\alpha\Big(\bigsqcup_{\beta<\lambda} \varphi_\beta\Big) \leq^\sharp \bigsqcup_{\beta<\lambda}^\sharp \psi_\beta^\sharp$$

Approximation of the semantic function

- Abstraction of a concrete function:

$$\begin{array}{ccc} \mathcal{F}^\sharp & \xrightarrow{\quad\alpha \circ \varphi \circ \gamma\quad} & \mathcal{F}^\sharp \\ {\scriptstyle\gamma}\downarrow & & \uparrow{\scriptstyle\alpha} \\ \mathcal{F} & \xrightarrow[\quad\varphi\quad]{} & \mathcal{F} \end{array}$$

- The abstract semantic function $F^\sharp$ upper approximates the concrete semantic function F. We have:

$$\alpha \circ F \circ \gamma \;\leq^\sharp\; F^\sharp$$

for the pointwise ordering $\varphi \leq^\sharp \psi \stackrel{\text{def}}{=} \forall x : \varphi(x) \leq^\sharp \psi(x)$

Fixpoint approximation

- The abstract approximation of the concrete collecting semantics is safe:

$$\mathrm{lfp}^{\sqsubseteq}_{\perp} F \leq \gamma\Big(\bigsqcup^\sharp_{\lambda \in \mathbb{O}} {F^\sharp}^\lambda\big(\perp^\sharp\big)\Big)$$

Galois surjection

- When no abstract property is useless, that is the abstraction function α is surjective;
- In a Galois connection α is surjective iff γ is injective iff $\alpha \circ \gamma$ is the identity;
- A "Galois surjection" is a Galois connection with α surjective, written:

$$\begin{array}{rl} & \mathcal{F}(\leq) \;\underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\; \mathcal{F}^\sharp(\leq^\sharp) \\ \stackrel{\text{def}}{=} & \\ & \mathcal{F}(\leq) \;\underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\; \mathcal{F}^\sharp(\leq^\sharp) \;\wedge\; \forall \psi \in \mathcal{F}^\sharp : \alpha \circ \gamma(\psi) = \psi \end{array}$$

Coincidence of the abstract approximation and computational ordering

If the abstract ordering is an abstraction of both concrete computational and approximation orderings:

$$\mathcal{F}(\leq) \;\underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\; \mathcal{F}^\sharp(\leq^\sharp) \qquad \mathcal{F}(\sqsubseteq) \;\underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\; \mathcal{F}^\sharp(\leq^\sharp)$$

then $\mathcal{F}^\sharp(\leq^\sharp, \perp^\sharp, \top^\sharp, \sqcup^\sharp, \sqcap^\sharp)$ is a complete lattice such that:

$$\begin{array}{c} \perp^\sharp = \alpha(\perp) \\ \bigsqcup^\sharp_{\lambda \in \mathbb{O}} \varphi^\sharp_\lambda = \alpha\Big(\bigsqcup_{\lambda \in \mathbb{O}} \gamma\big(\varphi^\sharp_\lambda\big)\Big) \end{array}$$

Application to strictness analysis

Definition of strictness

f is *strict* in its parameters $v \in I$ if and only if for all $\vec{\nu} \in \vec{\mathcal{D}}^{\Re}$ and $\nu \in \mathcal{D}^{\Re}$:

$$(\forall v \in I : \vec{\nu}[v] \in \{\bot, \Omega\} \wedge \langle \vec{\nu}, \nu \rangle \in f^{\Re}) \Rightarrow \nu \in \{\bot, \Omega\}$$

Taking errors into account

- Left-to-right addition $+$ is strict in its <u>first</u> parameter:

$$\bot + \nu = \bot \qquad \forall \nu \in \Upsilon_{\bot}^{\Omega}$$

- Left-to-right addition $+$ is strict in its <u>second</u> parameter:

$$\nu + \bot = \bot \qquad \forall \nu \in \Upsilon_{\bot}$$

only if errors are included in the definition of strictness:

$$\Omega + \bot = \Omega$$

Concrete approximation ordering

The concrete approximation ordering is $\subseteq$

Proposition 1 *For all φ, $\varphi' \in \mathcal{F}^{\Re}$, if $\varphi \subseteq \varphi'$ and φ' is strict in its parameters $v \in I$ then φ is strict in its parameters $v \in I$.*

Proof: If $\forall v \in I : \vec{\nu}[v] \in \{\bot, \Omega\}$ and $\langle \vec{\nu}, \nu \rangle \in \varphi^{\Re}$ then $\langle \vec{\nu}, \nu \rangle \in \varphi$ whence $\langle \vec{\nu}, \nu \rangle \in \varphi'$ since $\varphi \subseteq \varphi'$ that is $\langle \vec{\nu}, \nu \rangle \in \varphi'^{\Re}$ so that $\nu \in \{\bot, \Omega\}$ since φ' is strict in its parameters $v \in I$ proving that φ is strict in $v \in I$. □

Construction of Mycroft's algorithm

Mycroft's algorithm can be derived from the relational semantics using the approximation formalized by the Galois surjection:

$$\wp(\mathcal{D}^{\Re})(\subseteq) \underset{\alpha^{\sharp}_{\mathcal{D}}}{\overset{\gamma^{\sharp}_{\mathcal{D}}}{\leftrightarrows}} \mathcal{D}^{\sharp}(\leq)$$

- $\mathcal{D}^{\sharp} = \{0, 1\}$ with $0 \leq 0 < 1 \leq 1$;
- $\gamma^{\sharp}_{\mathcal{D}}(0) \stackrel{\text{def}}{=} \{\bot, \Omega\}$ and $\gamma^{\sharp}_{\mathcal{D}}(1) \stackrel{\text{def}}{=} \Upsilon_{\bot}^{\Omega}$;
- $\alpha^{\sharp}_{\mathcal{D}}(V) \stackrel{\text{def}}{=} (V \subseteq \{\bot, \Omega\} \rightarrow 0, 1)$.

Abstraction of sets of arguments

A set of vectors in $\vec{\mathcal{D}}^{\Re} = \prod_{v \epsilon \vec{v}} \mathcal{D}^{\Re}$ is approximated componentwise:

$$\wp(\vec{\mathcal{D}}^{\Re})(\subseteq) \quad \underset{\alpha^{\sharp}_{\vec{\mathcal{D}}}}{\overset{\gamma^{\sharp}_{\vec{\mathcal{D}}}}{\leftrightarrows}} \quad \vec{\mathcal{D}}^{\sharp}(\vec{\leq})$$

- $\vec{\mathcal{D}}^{\sharp} \stackrel{\text{def}}{=} \prod_{v \epsilon \vec{v}} \mathcal{D}^{\sharp}$ is a complete lattice for the componentwise ordering:
$$\vec{\nu} \vec{\leq} \vec{\nu}\,' \stackrel{\text{def}}{=} (\forall v \in \vec{v} : \vec{\nu}[v] \leq \vec{\nu}\,'[v])$$
- $\alpha^{\sharp}_{\vec{\mathcal{D}}}(\vec{V}) \stackrel{\text{def}}{=} \prod_{v \epsilon \vec{v}} \alpha^{\sharp}_{\mathcal{D}}(\{\vec{\nu}[v] \mid \vec{\nu} \in \vec{V}\})$
- $\gamma^{\sharp}_{\vec{\mathcal{D}}}(\vec{\nu}) \stackrel{\text{def}}{=} \{\vec{\nu}\,' \in \vec{\mathcal{D}}^{\Re} \mid \forall v \in \vec{v} : \vec{\nu}\,'[v] \in \gamma^{\sharp}_{\mathcal{D}}(\vec{\nu}[v])\}$

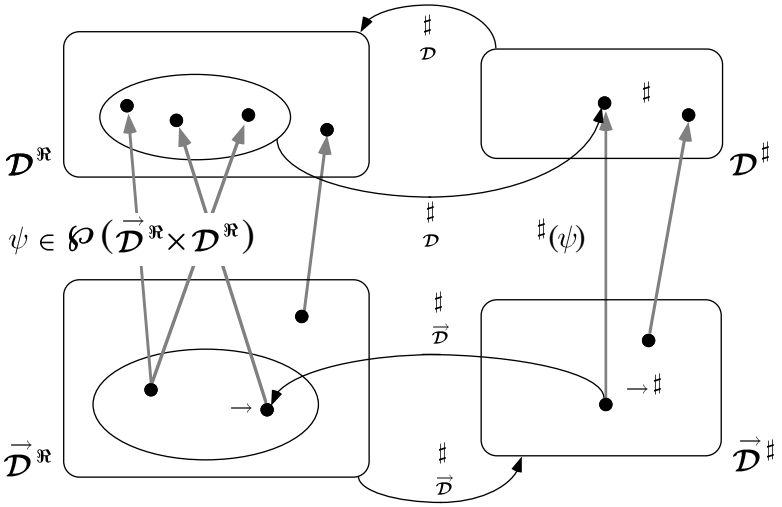
Abstraction of arguments-result relations

An arguments-result relation in $\mathcal{F}^{\Re} = \wp(\vec{\mathcal{D}}^{\Re} \times \mathcal{D}^{\Re})$ is approximated by:

$$\mathcal{F}^{\Re}(\subseteq) \quad \underset{\alpha^{\sharp}}{\overset{\gamma^{\sharp}}{\leftrightarrows}} \quad \mathcal{F}^{\sharp}(\leq)$$

- $\mathcal{F}^{\sharp} \stackrel{\text{def}}{=} \vec{\mathcal{D}}^{\sharp} \stackrel{\vec{\leq}}{\longmapsto}_{\text{m}} \mathcal{D}^{\sharp}$ is a complete lattice for the pointwise ordering:
$$\varphi \leq \varphi' \stackrel{\text{def}}{=} (\forall \vec{\nu} \in \vec{\mathcal{D}}^{\sharp} : \varphi(\vec{\nu}) \leq \varphi'(\vec{\nu})$$
- $\alpha^{\sharp}(\psi) \stackrel{\text{def}}{=} \lambda \vec{\nu}^{\sharp} . \alpha^{\sharp}_{\mathcal{D}}(\{\nu \mid \exists \vec{\nu} \in \gamma^{\sharp}_{\vec{\mathcal{D}}}(\vec{\nu}^{\sharp}) : \langle \vec{\nu},\, \nu \rangle \in \psi\})$
- $\gamma^{\sharp}(\varphi) \stackrel{\text{def}}{=} \{\langle \vec{\nu},\, \nu \rangle \mid \nu \in \gamma^{\sharp}_{\mathcal{D}} \circ \varphi \circ \alpha^{\sharp}_{\vec{\mathcal{D}}}(\{\vec{\nu}\})\}$

Abstraction of a relation by a function



Abstraction of vectors of relations

A vector of relations in $\vec{\mathcal{F}}^{\Re} = \prod_{f \epsilon \vec{f}} \mathcal{F}^{\Re}$ is approximated componentwise by:

$$\vec{\mathcal{F}}^{\Re}(\vec{\subseteq}) \quad \underset{\vec{\alpha}^{\sharp}}{\overset{\vec{\gamma}^{\sharp}}{\leftrightarrows}} \quad \vec{\mathcal{F}}^{\sharp}(\vec{\leq})$$

- $\vec{\mathcal{F}}^{\sharp} \stackrel{\text{def}}{=} \prod_{f \epsilon \vec{f}} \mathcal{F}^{\sharp}$ is a complete lattice for the componentwise ordering:
$$\vec{\varphi} \vec{\leq} \vec{\varphi}\,' \stackrel{\text{def}}{=} (\forall f \in \vec{f} : \vec{\varphi}\,[f] \leq \vec{\varphi}\,'[f]$$
- $\vec{\alpha}^{\sharp}(\vec{\psi}) \stackrel{\text{def}}{=} \prod_{f \epsilon \vec{f}} \alpha^{\sharp}(\vec{\psi}[f])$
- $\vec{\gamma}^{\sharp}(\vec{\varphi}) \stackrel{\text{def}}{=} \prod_{f \epsilon \vec{f}} \gamma^{\sharp}(\vec{\varphi}\,[f])$

Properties of this abstraction

The abstract ordering $\vec{\leq}$ on vectors of abstract functions in $\vec{\mathcal{F}}^\sharp$ is an abstraction both of the concrete computational ordering:

$$\vec{\mathcal{F}}^\Re(\vec{\sqsubseteq}^\Re) \underset{\vec{\alpha}^\sharp}{\overset{\vec{\gamma}^\sharp}{\leftrightarrows}} \vec{\mathcal{F}}^\sharp(\vec{\leq})$$

and of the concrete approximation ordering:

$$\vec{\mathcal{F}}^\Re(\vec{\subseteq}) \underset{\vec{\alpha}^\sharp}{\overset{\vec{\gamma}^\sharp}{\leftrightarrows}} \vec{\mathcal{F}}^\sharp(\vec{\leq})$$

Construction of Mycroft's algorithm

- $\underset{i\in\Delta}{\vec{\sqcup}^\sharp}\ \vec{\varphi}_i = \vec{\alpha}^\sharp(\underset{i\in\Delta}{\vec{\sqcup}^\Re}\ \vec{\gamma}^\sharp(\vec{\varphi}_i))$ — definition of $\vec{\sqcup}$
  $= \underset{i\in\Delta}{\vec{\vee}^\sharp}\ \vec{\alpha}^\sharp(\vec{\gamma}^\sharp(\vec{\varphi}_i))$ — $\vec{\alpha}^\sharp$ is a complete $\vec{\sqcup}^\Re$-morphism
  $= \underset{i\in\Delta}{\vec{\vee}^\sharp}\ \vec{\varphi}_i$ — since $\vec{\alpha}^\sharp \circ \vec{\gamma}^\sharp$ is the identity
- $\vec{\bot}^\sharp = \vec{\alpha}^\sharp(\vec{\bot}^\Re) = \ldots = \prod_{f\in\vec{f}} \lambda\vec{\nu}^\sharp.0$
- $\vec{\alpha}^\sharp \circ \vec{F}^\Re \circ \vec{\gamma}^\sharp = \ldots \vec{\leq} \vec{F}^\sharp$

After three pages of hand-computation ... proceeding by induction on the syntax of the programs and consisting in expanding these definitions and then in simplifying them, we have constructively derived Mycroft's algorithm from the above specification.

Mycroft's strictness semantics of $\prod_{f\in\vec{f}} f\vec{v} \equiv \vec{F}[f]$

$\nu : \mathcal{D}^\sharp \stackrel{\text{def}}{=} \{0,1\}$ $\vec{\nu} : \vec{\mathcal{D}}^\sharp \stackrel{\text{def}}{=} \prod_{v\in\vec{v}} \mathcal{D}^\sharp$ $\varphi, f^\sharp : \mathcal{F}^\sharp \stackrel{\text{def}}{=} \vec{\mathcal{D}}^\sharp \overset{\text{m}\vec{\leq}}{\longmapsto} \mathcal{D}^\sharp$ $\vec{\varphi}, \vec{f}^\sharp : \vec{\mathcal{F}}^\sharp \stackrel{\text{def}}{=} \prod_{f\in\vec{f}} \mathcal{F}^\sharp$	$e^\sharp : \mathcal{E}^\sharp \stackrel{\text{def}}{=} \vec{\mathcal{F}}^\sharp \overset{\text{m}\vec{\leq}}{\longmapsto} \vec{\mathcal{D}}^\sharp \overset{\text{m}\vec{\leq}}{\longmapsto} \mathcal{D}^\sharp$ $\vec{e}^\sharp : \vec{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} \vec{\mathcal{F}}^\sharp \overset{\text{m}\vec{\leq}}{\longmapsto} \vec{\mathcal{D}}^\sharp \overset{\text{m}\vec{\leq}}{\longmapsto} \vec{\mathcal{D}}^\sharp$ $\vec{F}^\sharp : \vec{\mathcal{F}}^\sharp \overset{\text{m}\vec{\leq}}{\longmapsto} \vec{\mathcal{F}}^\sharp$ $\stackrel{\text{def}}{=} \lambda\vec{\varphi}\cdot\prod_{f\in\vec{f}} \lambda\vec{\nu}\cdot\vec{F}[f]^\sharp[\![\vec{\varphi}]\!]\vec{\nu}$
$k^\sharp[\![\vec{\varphi}]\!]\vec{\nu} \stackrel{\text{def}}{=} 1$ $v^\sharp[\![\vec{\varphi}]\!]\vec{\nu} \stackrel{\text{def}}{=} \vec{\nu}[v]$ $b\vec{e}^\sharp[\![\vec{\varphi}]\!]\vec{\nu} \stackrel{\text{def}}{=} b^\sharp(\vec{e}^\sharp[\![\vec{\varphi}]\!]\vec{\nu})$ where $b^\sharp \geq^\sharp \alpha^\sharp(b^\Re)$ $(e_1 \to e_2, e_3)^\sharp[\![\vec{\varphi}]\!]\vec{\nu} \stackrel{\text{def}}{=} e_1^\sharp[\![\vec{\varphi}]\!]\vec{\nu} \wedge (e_2^\sharp[\![\vec{\varphi}]\!]\vec{\nu} \vee e_3^\sharp[\![\vec{\varphi}]\!]\vec{\nu})$ $f\vec{e}^\sharp[\![\vec{\varphi}]\!]\vec{\nu} \stackrel{\text{def}}{=} \vec{\varphi}[f](\vec{e}^\sharp[\![\vec{\varphi}]\!]\vec{\nu})$ $\vec{e}^\sharp[\![\vec{\varphi}]\!]\vec{\nu} \stackrel{\text{def}}{=} \prod_{v\in\vec{e}} \vec{e}[v]^\sharp[\![\vec{\varphi}]\!]\vec{\nu}$	
$\vec{\bot}^\sharp \stackrel{\text{def}}{=} \prod_{f\in\vec{f}} \lambda\vec{\nu}.0$ $\vec{f}^\sharp \stackrel{\text{def}}{=} \text{lfp}^{\vec{\leq}}_{\vec{\bot}^\sharp} \vec{F}^\sharp = \vec{\bigvee}_{n\in\mathbb{N}} \vec{F}^{\sharp n}(\vec{\bot}^\sharp)$	

Safeness of Mycroft's algorithm

By construction we have:

Proposition 2 *The strictness semantics is an abstraction of the relational semantics:*

$$\vec{\alpha}^\sharp(\vec{f}^{\,\Re}) = \vec{\alpha}^\sharp(\text{lfp}^{\vec{\sqsubseteq}^*}_{\vec{\bot}^*} \vec{F}^\Re) \vec{\leq} \vec{f}^{\,\sharp} = \text{lfp}^{\vec{\leq}}_{\vec{\bot}^\sharp} \vec{F}^\sharp$$

so that Mycroft's algorithm is safe:

Proposition 3 *If* $\vec{f}^{\,\sharp}[f](\vec{1}[v \leftarrow 0, v \in I]) = 0$ *then* $\vec{f}^{\,\Re}[f]$ *is strict in its parameters* I.

Example

- Program: `f(x, y)` $\equiv$ `((x = 0)` $\rightarrow$ `y, (1 + f(x - 1, y)))`
- Equation: $f^\sharp(x,y) = (x \wedge 1) \wedge (y \vee (1 \wedge f^\sharp(x \wedge 1, y)))$
- Strictness of first parameter:

$$\begin{aligned} f^{\sharp 0}(0,1) &= 0 \\ f^{\sharp 1}(0,1) &= (0 \wedge 1) \wedge (1 \vee (1 \wedge f^{\sharp 0}(0 \wedge 1, 1))) \\ &= 0 \end{aligned}$$

- Strictness of second parameter:

$$\begin{aligned} f^{\sharp 0}(1,0) &= 0 \\ f^{\sharp 1}(1,0) &= (1 \wedge 1) \wedge (0 \vee (1 \wedge f^{\sharp 0}(1 \wedge 1, 0))) \\ &= 0 \end{aligned}$$

Johnson's algorithm

- Johnson's algorithm is obtained by a further approximation:

$$f^\sharp(x,y)$$

is approximated by a pair a functions:

$$f^{\divideontimes}{}_x(x) \stackrel{\text{def}}{=} f^\sharp(x,1) \qquad f^{\divideontimes}{}_y(y) \stackrel{\text{def}}{=} f^\sharp(1,y)$$

- Johnson's algorithm is constructed formally in the paper;
- The formal construction lead to an algorithm better than those given in the literature [Joh81, Hug88]

Comparison of Mycroft's and Johnson's algorithms

- Mycroft's algorithm may be exponential since in the worst case we have to compute $f^\sharp(x_1, \ldots, x_n)$ for all $x_i \in \{0,1\}$, $i = 1, \ldots, n$ that is 2^n possibilities for each function f of the program.
- Johnson's algorithm is linear since in the worst case we have to compute $f^{\divideontimes}{}_{x_i}(x_i)$ for all $x_i \in \{0,1\}$, that is $2n$ possibilities for each function f of the program.

- Johnson's algorithm is less precise since one cannot express join strictness in several parameters:
  - Program:

$$\texttt{f(x, y, z)} \equiv (\texttt{x} \rightarrow \texttt{y}, \texttt{z})$$

  - Mycroft's equation:

$$f^\sharp(x,y,z) = (x \wedge (y \vee z))$$

  f is jointly strict in y and z since $f^\sharp(1,0,0) = 0$
  - Johnson's equation:

$$\begin{aligned} f^{\divideontimes}{}_x(x) &= x \\ f^{\divideontimes}{}_y(y) &= 1 \\ f^{\divideontimes}{}_z(z) &= 1 \end{aligned}$$

  f is not jointly strict in y and z since $f^{\divideontimes}{}_y(0) \wedge f^{\divideontimes}{}_z(0) = 1$

- We can limit the dependencies to a given κ (for example $\kappa = 3$);
- $f^\sharp(x_1, \ldots, x_n)$ with less than κ 0-valued parameters x_i is evaluated normally;
- $f^\sharp(x_1, \ldots, x_n)$ with more than κ 0-valued parameters x_i is upper approximated by:

$$\bigwedge_{\substack{i=1 \\ x_i=0}}^{n} f^\sharp(1, \ldots, x_i, \ldots, 1)$$

- Johnsson's algorithm corresponds to $\kappa = 1$.

Conclusion

- Relational semantics seems to be more convenient than denotational semantics for abstract interpretation;
- Constructive derivation of the abstract interpreter specification is preferable to empirical design with a posteriori safeness verification;
- Abstract interpretation is not, contrary to a common believe, intrinsically exponential;
- Well-chosen widening operators often offer a good compromise between precision and cost of the analysis.

[CC77] P. Cousot & R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4^{th} POPL*, pp. 238–252, Los Angeles, California, 1977. ACM Press.

[CC79] P. Cousot & R. Cousot. Systematic design of program analysis frameworks. In *6^{th} POPL*, pp. 269–282, San Antonio, Texas, 1979. ACM Press.

[CC92a] P. Cousot & R. Cousot. Abstract interpretation and application to logic programs. *J. Logic Prog.*, 13(2–3):103–179, 1992.

[CC92b] P. Cousot & R. Cousot. Abstract interpretation frameworks. *J. Logic and Comp.*, 2(4):511–547, Aug. 1992.

[CC92c] P. Cousot & R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In M. Bruynooghe & M. Wirsing, eds., *Programming Language Implementation and Logic Programming, Proceedings of the Fourth International Symposium, PLILP'92*, Leuven, Belgium, 13–17 Aug. 1992, LNCS 631, pp. 269–295. Springer-Verlag, 1992.

[CC92d] P. Cousot & R. Cousot. Inductive definitions, semantics and abstract interpretation. In *19^{th} POPL*, pp. 83–94, Albuquerque, New Mexico, 1992. ACM Press.

[Hug88] R. J. M. Hughes. Backwards analysis of functional programs. In A. P. Bjørner D., Ershov & N. D. Jones, eds., *Partial Evaluation and Mixed Computation*, Proceedings IFIP TC2 Workshop, Gammel Avernæs, Denmark, pp. 187–208. Elsevier, Oct. 1988.

[Joh81] T. Johnsson. Detecting when call-by-value can be used instead of call-by-need. Research Report LPM MEMO 14, Laboratory for Programming Methodology, Department of Computer Science, Chalmers University of Technology, S-412 96 Göteborg, Sweden, Oct. 1981.

[MN83] A. Mycroft & F. Nielson. Strong abstract interpretation using power domains. In J. Diaz, ed., *Tenth ICALP*, LNCS 154, pp. 536–547. Springer-Verlag, 1983.

[Myc80] A. Mycroft. The theory and practice of transforming call-by-need into call-by-value. In B. Robinet, ed., *Proc. Fourth International Symposium on Programming*, Paris, France, 22-24 Apr. 1980, LNCS 83, pp. 270–281. Springer-Verlag, 1980.

[Nie88] F. Nielson. Strictness analysis and denotational abstract interpretation. *Inf. & Comp.*, 76(1):29–92, 1988.