The Contributions of Alan Mycroft to Abstract Interpretation

Patrick Cousot

CIMS, CS, New York University

1 Introduction

Abstract interpretation started in the late 70's in Grenoble, France. At that time Rod Burstall was visiting the computer science department (IMAG), and Radhia and I have had numerous conversations (and steak-frites-salade lunches at the GUC (Grenoble University Club) restaurant) with Rod. I suspect that Alan Mycroft got news about abstract interpretation far north in Edinburgh, Scotland, thanks to Rod Burstall supervising his thesis together with Robin Milner. This led to his first publication [15] in 1980 and his thesis "Abstract Interpretation and Optimising Transformations for Applicative Programs [16] in 1982 (his second most cited paper on Google Scholar, not so common for a thesis). His work originated a lot of research on strictness analysis and more generally the static analysis of functional programs, a crucial contribution to the theory, diffusion, and application of abstract interpretation.

Alan Mycroft has had a very productive career with numerous applied and theoretical achievements in various domains of computer science and so my overview of his contributions will be restricted to those explicitly connected to abstract interpretation.

2 Strictness Analysis

The first problem Alan Mycroft solved was to determine statically when a lazy call by need (or call by name) in a side-effect free, higher-order, functional language (where the formal parameter is reevaluated in the context of the call whenever (or the first time) it is used in the function body) by a more efficient call by value (where the formal parameter is evaluated at call time and stored in the context/environment of evaluation of the function body). Because of the absence of side effects, the only difference is when the evaluation of the parameter does not terminate and it is never used in the function body/definition so that call be need/name will terminate while call by value will not, as in `let f x = f x and g y = if true then 0 else y in g (f 0)`. The two call methods are therefore equivalent when the function call evaluated with call by need/name does not terminates when the evaluation of the parameter does not terminate (assuming the "observations" of a functional program execution is its final value or non-termination). Explained in terms of denotational semantics, this is $f(\perp) = \perp$, where $\perp$ denotes nontermination, which is the infimum in Scott domain, meaning that f is strict, hence the term "strictness analysis".

2.1 Alan Mycroft's starting point

The state of the theory of abstract interpretation that Alan Mycroft relied on, was for reachability/invariant verification/analysis of transition systems $\langle \Sigma, \tau \rangle$ [1].

The objective is to infer an invariant $I \in \wp(\Sigma)$ over approximating the reachable states $\{\sigma' \mid \exists \sigma \in P \,.\, \langle \sigma, \sigma' \rangle \in \tau^*\} \subseteq I$ from initial states $P \in \wp(\Sigma)$ where $\tau^* \triangleq \bigcup_{n \in \mathbb{N}} \tau^n = \mathsf{lfp}^{\subseteq} T$ with $T \triangleq \boldsymbol{\lambda} X \bullet \tau^0 \cup \tau \circ X$ is the reflexive transitive closure of $\tau \in \wp(\Sigma \times \Sigma)$, τ^0 is the identity relation on the set of states Σ, and $\tau^{n+1} \triangleq \tau^n \circ \tau$ is the relation power. Define $\mathsf{post}(r)P \triangleq \{\sigma' \mid \exists \sigma \in P \,.\, \langle \sigma, \sigma' \rangle \in r\}$ to be the right-image of the states in P by the relation r on states. We look for an invariant I such that $\mathsf{post}(\mathsf{lfp}^{\subseteq} T) \subseteq I$. The method is explained in proposition 2 thereafter, proofs are relegated to the appendix.

Proposition 1. *Given $P \in \wp(\Sigma)$, we have the Galois connection $\langle \wp(\Sigma \times \Sigma), \subseteq \rangle \xrightleftharpoons[\boldsymbol{\lambda} r \bullet \mathit{post}(r)P]{\gamma_P} \langle \wp(\Sigma), \subseteq \rangle$*[2].

Proposition 2. *For all $P \in \wp(\Sigma)$, we have the following commutation property $\boldsymbol{\lambda} r \bullet \mathit{post}(r)P \circ T = F_P \circ \boldsymbol{\lambda} r \bullet \mathit{post}(r)P$ with $T(X) \triangleq \tau^0 \cup \tau \circ X$ and $F_P(X) \triangleq P \cup \mathit{post}(\tau)X$.*

By the fixpoint exact abstraction under this commutation condition, if follows that $\mathsf{post}(\mathsf{lfp}^{\subseteq} T) = \mathsf{lfp}^{\subseteq} F_P$, as shown by the following

Proposition 3. *if $\langle C, \preceq \rangle$ and $\langle A, \preccurlyeq \rangle$ are CPO's (every increasing chain has a lub, including the empty chain, so has an infimum), $f \in C \xrightarrow{c} C$ and $\bar{f} \in A \xrightarrow{c} A$ are continuous, $\langle C, \preceq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A, \preccurlyeq \rangle$ is a Galois connection, then the commutation condition $\alpha \circ f = \bar{f} \circ \alpha$ (respectively semi-commutation $\alpha \circ f \mathrel{\dot{\preccurlyeq}} \bar{f} \circ \alpha$, pointwise) implies that $\alpha(\mathsf{lfp}^{\preceq} f) = \mathsf{lfp}^{\preceq} \bar{f}$ (resp. $\alpha(\mathsf{lfp}^{\preceq} f) \preccurlyeq \mathsf{lfp}^{\preccurlyeq} \bar{f}$).*

The problem is thus to find an invariant I such that $\mathsf{lfp}^{\subseteq} F_P \subseteq I$. It is essential to remark that the computation ordering used for the fixpoint $\mathsf{lfp}^{\subseteq} F_P$ and the logical ordering in $\mathsf{lfp}^{\subseteq} F_P \subseteq I$ to over approximate the reachable states are the same so that the above fixpoint approximate abstraction under the semi-commutation condition is directly applicable

Therefore, by proposition 3, using a Galois connection $\langle \wp(\Sigma), \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A, \leqslant \rangle$, the problem can be reduced to the computation of an abstract invariant $\mathsf{lfp}^{\leqslant} \alpha \circ F_P \circ \gamma$ such that $\mathsf{lfp}^{\subseteq} F_P \subseteq \gamma(\mathsf{lfp}^{\leqslant} \alpha \circ F_P \circ \gamma)$, as desired.

Using a semi-commuting over approximation $\bar{F}_P$ such that $\alpha \circ \bar{F}_P \mathrel{\dot{\leqslant}} F_P \circ \alpha$ is also feasible since then $\mathsf{lfp}^{\subseteq} F_P \subseteq \gamma(\mathsf{lfp}^{\leqslant} \bar{F}_P)$, by proposition 3.

[1] Alan and his followers refer to "flowchart abstract interpretation" whereas in my thesis and POPL79, I had moved from flowcharts to transition systems for conciseness.

[2] $\langle C, \preceq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A, \preccurlyeq \rangle$ denotes the fact that $\langle C, \preceq \rangle$ and $\langle A, \preccurlyeq \rangle$ are posets, $\alpha \in C \longrightarrow A$, $\gamma \in A \longrightarrow C$, and $\forall x \in C, y \in A \,.\, \alpha(x) \preccurlyeq y \Leftrightarrow x \preceq \gamma(y)$.

2.2 Alan Mycroft's pioneer strictness analysis

Alan Mycroft formulates strictness analysis by first defining the denotational semantics of the (recursive) function as a fixpoint $\mathsf{lfp}^{\sqsubseteq} F \in \mathcal{D}_\perp \xrightarrow{c} \mathcal{D}_\perp$ where the domain $\mathcal{D}_\perp$ is a CPO $\langle \mathcal{D}_\perp,\ \sqsubseteq,\ \perp,\ \sqcup\rangle$ with ordering $\sqsubseteq$ and the functional $F \in (\mathcal{D}_\perp \xrightarrow{c} \mathcal{D}_\perp) \xrightarrow{c} (\mathcal{D}_\perp \xrightarrow{c} \mathcal{D}_\perp)$ is continuous. (F is defined on pages 38/39 of his thesis by structural induction on the functional programs he considers. As stated in [16, page 54], $\sqsubseteq$ is taken to be Scott ordering $\perp \sqsubseteq \perp \sqsubset x \sqsubseteq x, x \in D$ on the flat domain $\mathcal{D}_\perp = D \cup \{\perp\}$, $\perp \notin D$).

The collecting semantics of a function with denotational semantics $f \in \mathcal{D}_\perp \xrightarrow{c} \mathcal{D}_\perp$ is $\mathsf{post}(f)P \triangleq \{f(x) \mid x \in P\}$ where $P \in \wp(\mathcal{D}_\perp)$ is a set of possible values of the parameter (including $\perp$ in case of non termination) and $\mathsf{post}(f)P$ provides the possible results of the function for these parameters. Since the considered functions are total and determinitic, the image of a singleton is a singleton. Moreover, post preserves arbitrary joins so that the collecting semantics $\mathsf{post}(f)$ belongs to $\wp(\mathcal{D}_\perp) \xrightarrow{1\cup} \wp(\mathcal{D}_\perp)$ defined as

$$\begin{aligned}\wp(\mathcal{D}_\perp) \xrightarrow{1\cup} \wp(\mathcal{D}_\perp) \triangleq \{\phi \in \wp(\mathcal{D}_\perp) \longrightarrow \wp(\mathcal{D}_\perp) \mid \forall x \in \mathcal{D}_\perp \,.\, |\phi(\{x\})| = 1 \wedge{}\\ \forall X \in \wp(\wp(\mathcal{D}_\perp)) \,.\, \mathsf{post}(f) \textstyle\bigcup_{P\in X} P = \bigcup_{P \in X} \mathsf{post}(f)P\}\end{aligned}$$

as well as to $\mathsf{post}(f) \in \wp(\mathcal{D}_\perp) \setminus \{\emptyset\} \xrightarrow{1\cup} \wp(\mathcal{D}_\perp) \setminus \{\emptyset\}$ since $\mathsf{post}(f)X = \emptyset$ if and only if $X = \emptyset$.

Proposition 4. *We have the Galois connection* $\langle \mathcal{D}_\perp \longrightarrow \mathcal{D}_\perp,\ \dot{\sqsubseteq}\rangle \underset{\mathsf{post}}{\overset{\hat{\gamma}}{\leftrightarrows}} \langle \wp(\mathcal{D}_\perp) \setminus \{\emptyset\} \xrightarrow{1\cup} \wp(\mathcal{D}_\perp) \setminus \{\emptyset\},\ \dot{\hat{\sqsubseteq}}\rangle$ *where* $\hat{\gamma}(\phi) \triangleq \boldsymbol{\lambda} x \bullet \mathsf{let}\ \{y\} = \phi(\{x\})\ \mathsf{in}\ y$ *and* $\hat{\sqsubseteq}$ *is Egli-Milner ordering* $X \mathrel{\hat{\sqsubseteq}} Y \triangleq (\forall x \in X \,.\, \exists y \in Y \,.\, x \sqsubseteq y \wedge \forall y \in Y \,.\, \exists x \in X \,.\, x \sqsubseteq y)$ *and* $\dot{\hat{\sqsubseteq}}$ *is its pointwise extension.*

In order to characterize $\mathsf{post}(\mathsf{lfp}^{\sqsubseteq} F)$ as a fixpoint, we consider the commutation condition

Proposition 5. $\mathsf{post} \circ F = \hat{F} \circ \mathsf{post}$ *with* $\hat{F}(\phi)P \triangleq \mathsf{post}(F(\hat{\gamma}(\phi)))P$.

By propositions 4, 5, and 3, we have

$$\mathsf{post}(\mathsf{lfp}^{\dot{\sqsubseteq}} F) = \mathsf{lfp}^{\dot{\hat{\sqsubseteq}}} \hat{F} \qquad (1)$$

(where $\hat{F}$ is defined by the equations on top of page 42 of Alan's thesis).

On page 53, Mycroft defines $\alpha^\sharp(S) \triangleq (\!| S = \{\perp\}\ ¿\ 0 : 1 |\!)$ and $\alpha^\flat(S) \triangleq (\!| \perp \in S\ ¿\ 0 : 1 |\!)$ so that

Proposition 6. *Let* $\mathbb{B} \triangleq \{0, 1\}$ *and* $\leq$ *be logical implication. Then* $\langle \wp(D_\perp) \setminus \{\emptyset\}, \subseteq\rangle \underset{\alpha^\sharp}{\overset{\gamma^\sharp}{\leftrightarrows}} \langle \mathbb{B}, \leq\rangle$ *and* $\langle \wp(D_\perp), \subseteq\rangle \underset{\alpha^\flat}{\overset{\gamma^\flat}{\leftrightarrows}} \langle \mathbb{B}, \leq\rangle$ *with* $\gamma^\sharp(b) = (\!| b = 0\ ¿\ \{\perp\} : D_\perp |\!)$, $\gamma^\flat(b) = (\!| b = 1\ ¿\ D : D_\perp |\!)$, *and* $0 < 1$.

Observe that $\langle \wp(D_\perp) \setminus \{\hat{\sqsubseteq}\},\ \subseteq\rangle \xleftrightarrow[\alpha^\sharp]{\gamma^\sharp} \langle \mathbb{B},\ \leq\rangle$ does <u>not</u> hold since e.g. for Scott ordering, we have $\alpha^\sharp(S) \leq 1$ but not $S \mathrel{\hat{\sqsubseteq}} \gamma^\sharp(1) = D_\perp$ since for $S \subset D$ and $y \in D \setminus S \subseteq D_\perp$ we don't have $\exists x \in S \,.\, x \sqsubseteq y$ since for Scott ordering the only possibility is $x = y \notin S$. So proposition 3 is not applicable to express $\alpha^\sharp(\mathsf{post}(\mathsf{lfp}^{\dot{\sqsubseteq}} F) = \alpha^\sharp(\mathsf{lfp}^{\dot{\hat{\sqsubseteq}}} \hat{F}))$ as a fixpoint. This is the main difficulty Alan Mycroft had to solve with the theory of abstract interpretation, as available as the time. The required generalization of proposition 3 is the following

Proposition 7. *Let $\langle C,\ \perp,\ \sqsubseteq,\ \sqcup\rangle$ be a concrete CPO for the computational ordering $\sqsubseteq$ and $f \in C \xrightarrow{c} C$ be continuous. Let $\langle C,\ \leq\rangle$ be a poset for the approximation ordering $\leq$.*

Let $\langle A,\ \perp^\sharp,\ \sqsubseteq^\sharp,\ \sqcup^\sharp\rangle$ be an abstract CPO and $f^\sharp \in A \xrightarrow{c} A$ be continuous.

Let $\langle C,\ \leq\rangle \xleftrightarrow[\alpha]{\gamma} \langle A,\ \sqsubseteq^\sharp\rangle$ be an abstraction such that

$$\perp \leq \gamma(\perp^\sharp) \tag{2}$$

$$\forall x \in C, y \in A \,.\, (x \leq \gamma(y)) \Rightarrow (f(x) \leq \gamma(f^\sharp(y))) \tag{3}$$

for all increasing chains $\langle x_i,\ i \in \mathbb{N}\rangle$ for $\sqsubseteq$ and $\langle y_i,\ i \in \mathbb{N}\rangle$ for $\sqsubseteq^\sharp$.

$$(\forall i \in \mathbb{N} \,.\, x_i \leq \gamma(y_i)) \Rightarrow \bigsqcup_{i\in\mathbb{N}} x_i \leq \gamma(\bigsqcup_{j\in\mathbb{N}}^{\sharp} y_j) \tag{4}$$

Then $\mathsf{lfp}^{\sqsubseteq} f \leq \gamma(\mathsf{lfp}^{\sqsubseteq^\sharp} f^\sharp)$.

Notice that in proposition 7, the computational ordering $\sqsubseteq$ and the approximation ordering $\leq$ may differ, whereas in proposition 3 they must be the same. This solves Alan problem for strictness analysis. Define $\vec{\alpha}^\sharp(f) \triangleq \alpha^\sharp \circ f \circ \gamma^\sharp$ and $\vec{\gamma}^\sharp(f) \triangleq \gamma^\sharp \circ f \circ \alpha^\sharp$ so that

$$\langle \wp(\mathcal{D}_\perp) \xrightarrow{1\cup} \wp(\mathcal{D}_\perp),\ \dot{\subseteq}\rangle \xleftrightarrow[\vec{\alpha}^\sharp]{\vec{\gamma}^\sharp} \langle \mathbb{B} \xrightarrow{i} \mathbb{B},\ \dot{\leq}\rangle \tag{5}$$

where $\mathbb{B} \xrightarrow{i} \mathbb{B}$ is the set of $\leq$-increasing Boolean functions. Define $\hat{F}^\sharp \triangleq \vec{\alpha}^\sharp \circ \hat{F} \circ \vec{\gamma}^\sharp$ so that the hypotheses (2), (3), and (4) of proposition 7 are satisfied with $C = \wp(\mathcal{D}_\perp) \xrightarrow{1\cup} \wp(\mathcal{D}_\perp)$, $\sqsubseteq\, = \dot{\hat{\sqsubseteq}}$, $A = \mathbb{B} \xrightarrow{i} \mathbb{B}$, $\sqsubseteq^\sharp\, = \dot{\leq}$, $\leq\, = \dot{\subseteq}$, $\alpha = \vec{\alpha}^\sharp$, and $\gamma = \vec{\gamma}^\sharp$. Proposition 7 applies to $\hat{F}^\sharp$.

Proposition 8. $\mathsf{lfp}^{\dot{\hat{\sqsubseteq}}} \hat{F} \mathrel{\dot{\subseteq}} \vec{\gamma}^\sharp(\mathsf{lfp}^{\dot{\leq}} \hat{F}^\sharp)$.

Therefore by (1) and proposition 8, we have $\mathsf{post}(\mathsf{lfp}^{\dot{\sqsubseteq}} F) = \mathsf{lfp}^{\dot{\hat{\sqsubseteq}}} \hat{F} \mathrel{\dot{\subseteq}} \vec{\gamma}^\sharp(\mathsf{lfp}^{\dot{\leq}} \hat{F}^\sharp)$. Since $\hat{F}^\sharp$ operates on a finite domain, $f^\sharp = \mathsf{lfp}^{\dot{\leq}} \hat{F}^\sharp$ is computable for any functional program. Assume that $f^\sharp(0) = 0$. Then $\mathsf{post}(\mathsf{lfp}^{\dot{\sqsubseteq}} F)\{\perp\} \subseteq \gamma^\sharp(0) = \{\perp\}$, proving strictness $F(\perp) = \perp$. Mycroft's strictness analysis method is sound (and also incomplete by Rice's theorem).

Alan applies the same approach to the lower abstraction $\alpha^\flat$ but this is of limited applicability since function nontermination be can proved with this abstraction only when it does not depend upon the values of the parameters. However, it is anticipating the present-day interest in under approximation verification and static analysis!

For functions with multiple parameters, Mycroft uses a Cartesian abstraction in the collecting semantics [16, top of page 42] which is more efficient but less precise that a relational analysis.

The thesis goes on in chapter 4 to show that call-by-need can be replaced by call-by-value for strict functions, see also [15].

2.3 The large body of research on strictness analysis in the 1980/90s

Alan Mycroft strictness analysis originated an enormous amount of work on the subject in the 80's and early 90's, see e.g. [1]. Strictness analysis has routinely found its way in modern compilers for lazy purely functional languages such as Haskell[3].

Although purely Boolean, strictness analysis suffers combinatorial explosions at higher-orders, although widenings, which have not been much investigated in this context, might certainly have helped [11].

2.4 Connection between call-by-value and call-by-name

In his thesis, Alan proved that call-by-need strict functions without side effects can be replaced by call-by-value functions. Nearly 40 years later, he came back to the subject, establishing a Galois connection between call-by-name and call-by-value for functions with limited side effects [14,13,12]. The Galois connection is between pre-ordered programs, where programs can be understood as encoding their set-based semantics, which established the link with abstract interpretation.

3 Sharing Analysis

In chapter 5 of his thesis [16], Alan Mycroft considers Lisp-like lists and two versions of LISP, a pure applicative LISP-A (declarative without side effect) and destructive LISP-D (with data structure alteration rplaca, rplacd, nconc, expressed using an explicit deallocation by free). The objective is to analyse declarative programs and optimize them into destructive ones.

To go into more details of LISP-A (we don't consider LISP-D and the transformation of LISP-A into LISP-D described in [16, section 5.8] and proved correct in [16, section 5.9]). Let $\mathbb{A}$ be a set of atoms, $\mathbb{L}$ be a disjoint set of locations and $\mathbb{V} = \mathbb{A} \cup \mathbb{L}$, with $\mathbb{A} \cap \mathbb{L} = \emptyset$ be the set of values. Assume that the memory heap/store is represented by binary directed acyclic graphs (DAGs) $H \in \mathbb{H}$ which are sets of nodes/cells $\langle v,\ v_h,\ v_t,\ f\rangle \in H$ such that $v \in \mathbb{L}$ is the location of the node, $v_h, v_t \in \mathbb{V}$ are the respective left/car/head and right/cdr/tail

[3] e.g. in the open source compiler and interactive environment GHC wiki.haskell.org/Performance/Strictness or the strcitness analysis of The Helium Compiler for a subset of Haskell.

values of the node, and $f \in \mathbb{B}$ records whether the node has been explicitly freed. The locations uniquely identify nodes in that if $H \ni \langle v,\ v_h,\ v_t,\ f\rangle \neq \langle v',\ v'_h,\ v'_t,\ f'\rangle \in H$ then $v \neq v'$. If $v \in \mathbb{L}$, we define the head $\mathsf{h}(v) = v_h$ and the tail $\mathsf{t}(v) = v_t$, this is an error if $v_h, v_t \in \mathbb{A}$ or $f = \texttt{true}$. The locations allocated in the heap H are $\mathsf{L}(H) \triangleq \{v \mid \langle v,\ v_h,\ v_t,\ f\rangle \in H\}$. The roots of the $\mathsf{R}(H)$ graph H have no predecessors $\mathsf{R}(H) \triangleq \{v \in \mathsf{L}(H) \mid \forall v', v'', f \mathrel{.} \langle v',\ v,\ v'',\ f\rangle \notin H \wedge \langle v',\ v'',\ v,\ f\rangle \notin H\}$. The heap/graph has no cycles, meaning that if $v_1 \ldots v_n \in \mathsf{L}(H)^n$, $n \geqslant 0$ is any sequence of heap locations such that $v_{i+1} \in \{\mathsf{h}(v_i), \mathsf{t}(v_i)\}$, $i \in [1, n]$, then $\forall 0 \leqslant i < j \leqslant n \mathrel{.} v_i \neq v_j$. The construction operation c (cons) is $\mathsf{c}(v_h, v_t)H \triangleq \mathsf{let}\ v \notin \mathsf{L}(H)\ \mathsf{in}\ H \cup \{\langle v,\ v_h,\ v_t,\ \mathsf{false}\rangle\}$ where $v_h, v_t \in \mathbb{A} \cup \mathsf{L}(H)$ are atoms or locations of nodes allocated in H. It follows that $\mathsf{h}(\mathsf{c}(v_h, v_t)) = v_h$ and $\mathsf{t}(\mathsf{c}(v_h, v_t)) = v_t$. The considered language [16, page 154] is a functional language with a denotation for atoms, primitives including c, h, t, free, atom (checking for atomicity). The fixpoint denotational semantics of a function denotation [16, Section 5.12] for LISP-D is a function f taking the value $v \in \mathsf{A} \cup \mathsf{L}(H)$ of the actual parameter and a memory heap H as a parameter and returning the possibly modified heap H' and a returned value $v' \in \mathsf{A} \cup \mathsf{L}(H')$ or $\perp$ in case of non termination (so $f \in \mathbb{L} \times \mathbb{H} \rightarrow \mathbb{L} \cup \{\perp\} \times \mathbb{H}$).

Alan looks for “an approximation to the set of paths which will actually exist at run time, but as usual in abstract interpretation (see chapter 2) the paths we infer will be a superset of those which can occur at run time”. He claims [16, Section 5.7, page 134] that his abstraction was inspired and generalizes the isolation classes of Jacob T. Schwarz[4], to provide information on “how shared an object might be”.

Given a heap $H \in \mathbb{H}$, $v, v' \in \mathsf{L}(H)$, the paths $\mathit{\Pi}(H)\langle v,\ v'\rangle$ from v to v' are

$$\mathit{\Pi}(H)\langle v,\ v'\rangle \triangleq \{x_0 \ldots x_n \in \mathsf{L}(H)^n \mid x_0 = v \wedge \forall i \in [0, n[\mathrel{.} x_{i+1} \in \{\mathsf{h}(x_i), \mathsf{t}(x_i)\} \wedge x_n = v'\}$$

$\mathit{\Pi}(H)\langle v,\ v'\rangle$ is empty when v or v' is an atom (including erroneous h and t). Let $O(H)$ be the set of locations on the heap H reachable from the roots of H through heads and tails by one path only.

$$O(H) \triangleq \{v \in \mathbb{A} \cup \mathsf{L}(H) \cup \{\perp\} \mid (v \in \mathsf{L}(H)) \Rightarrow (\forall v' \in \mathsf{R}(H) \mathrel{.} |\mathit{\Pi}(H)\langle v',\ v\rangle| = 1)\}$$

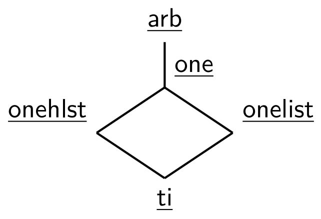
(where $|S|$ is the cardinality of a set S). $\mathit{\Delta}_{\mathsf{h}}(H)v$ (respectively $\mathit{\Delta}_{\mathsf{t}}(H)v$, $\mathit{\Delta}(H)v$) is the set of descendants of location $v \in \mathsf{L}(H)$ in the heap H going exclusively through heads (resp. through tails only, through heads or tails).

$$\begin{aligned}
\mathit{\Delta}_{\mathsf{h}}(H)v &\triangleq \{v' \mid \exists x_0 \ldots x_n \in \mathsf{L}(H)^n \mathrel{.} x_0 = v \wedge \forall i \in [0, n[\mathrel{.} x_{i+1} = \mathsf{h}(x_i) \wedge x_n = v'\} \\
\mathit{\Delta}_{\mathsf{t}}(H)v &\triangleq \{v' \mid \exists x_0 \ldots x_n \in \mathsf{L}(H)^n \mathrel{.} x_0 = v \wedge \forall i \in [0, n[\mathrel{.} x_{i+1} = \mathsf{t}(x_i) \wedge x_n = v'\} \\
\mathit{\Delta}(H)v &\triangleq \{v' \mid \mathit{\Pi}(H)\langle v,\ v'\rangle \neq \emptyset\}
\end{aligned}$$

[4] citing Schwarz, J. *Verifying the safe use of destructive operations in applicative programs*. Program Transformations - Proc. of the 3rd Int’l Symp. on Programming, Dunod Informatique, 1978, pp. 395–411, also DAI research report 55, Dept. of Artificial Intelligence, Edinburgh University, published while Jacob was visiting Edinburgh.

The abstract domain is the (complete) lattice

A =



The meaning of the abstract values (called “isolation classes”) is as follows.

- The supremum $\underline{\mathsf{arb}}$ can denote any atom (including error), element on the heap, or non termination. $\gamma(\underline{\mathsf{arb}}) \triangleq \{\langle v,\ H\rangle \mid H \in \mathbb{H} \wedge v \in \mathbb{A} \cup \mathsf{L}(H) \cup \{\bot\}\}$.
- The abstract value $\underline{\mathsf{one}}$ can denote any atom, non termination, or element on the heap accessible from the roots of the heap by one path only, so, “the object described [by $\underline{\text{one}}$] cannot be a shared CONS node” [16, page 138]. $\gamma(\underline{\mathsf{one}}) \triangleq \{\langle v,\ H\rangle \mid H \in \mathbb{H} \wedge v \in O(H)\}$.
- The abstract value $\underline{\mathsf{onehlst}}$ can denote any atom, non termination, or element on the heap accessible from the roots of the heap by one path only, and such that all its descendants by the head h are not accessible from the roots in any other way. $\gamma(\underline{\mathsf{onehlst}}) \triangleq \{\langle v,\ H\rangle \mid \forall v' \in \mathit{\Delta}_{\mathsf{h}}(H)v\ .\ v' \in O(H)\}$ (this includes $v' = v$).
- The abstract value $\underline{\mathsf{onelist}}$ is similar, but for tails only. So these nodes are uniquely accessible from the roots of H and so are all their descendants through tails only. $\gamma(\underline{\mathsf{onelist}}) \triangleq \{\langle v,\ H\rangle \mid \forall v' \in \mathit{\Delta}_{\mathsf{t}}(H)v\ .\ v' \in O(H)\}$
- The infimum $\underline{\mathsf{ti}}$ denotes any atom, nontermination, or location on the heap such that all its descendants are accessible form the roots by one path only (“objects totaly unshared from other objects” [16, page 135]). $\gamma(\underline{\mathsf{ti}}) \triangleq \{\langle v,\ H\rangle \mid \forall v' \in \mathit{\Delta}(H)v\ .\ v' \in O(H)\}$

Observe that $\gamma(\underline{\mathsf{onehlst}}) \sqcap \gamma(\underline{\mathsf{onelist}}) = \gamma(\underline{\mathsf{ti}})$ so we have a Galois connection with the abstraction of $P \in \wp(\mathbb{V} \times \mathbb{H})$ such that $\alpha(P) \triangleq \bigsqcap\{\underline{a}) \in A \mid P \subseteq \gamma(\underline{a})\}$. The abstraction is extended to functions of the collecting semantics $f \in \wp(\mathbb{V} \times \mathtt{H}) \xrightarrow{i} \wp(\mathbb{V} \times \mathtt{H})$ by $\alpha(f) \triangleq \alpha \circ f \circ \gamma$. This provides a fixpoint definition of the isolation class of a function in terms of the isolation classes of its parameters and its textual definition” [16, page 140], provided variables are handled correctly, as in [16, section 5.7.5], as roots of the DAG. Since the abstract domain is finite, the abstraction is computable for each subexpression appearing in a program.

3.1 Static analysis of shared data structures

Alan is one of the early users of abstract interpretation[5] for analyzing programs manipulating shared recursive data structures, a complex problem which, with parallelism, is still a hot research subject nowadays.

[5] following Cousot and Cousot, IFIP FDPC, 1978.

4 Main contributions of Alan Mycroft to Static Program Analysis

We have already underlined the pioneer work of Alan Mycroft on strictness analysis in Sect. 2.3 and linked data structure analysis in Sect. 3.1. In this section, we outline other important contributions to static program analysis (excluding dynamic analysis).

4.1 Denotational semantics based abstract interpretation

The work of Alan Mycroft [15,16] originated the used of denotational semantics as a standard semantics, as opposed to operational semantics, for abstract interpretation [20]. "The motivation is that abstract interpretation of denotational language definitions allows approximation of a wide class of programming language properties" [9, section 1.5]. The interest in denotational semantics, which is an abstract interpretation of operational semantics, later declined since it is not expressive enough (e.g. for trace or hyper properties) although basic principles like structural induction on the program syntax have perdured.

4.2 Static Analysis

Alan interest in static program analysis has persisted all along his scientific carrier, including for strictness analysis [3], analysis of procedures and functions [8,19], microcode [17], hardware [27,26], although his concerns in correctness proofs somewhat faded while concentrating on data flow analysis (in which programs are represented by graphs[6] and analysis algorithms are traditionally postulated rather than proved sound with respect to a semantics[7]) [7,6,4]. This informal approach is nevertheless with some exceptions, mainly without reference to a formal semantics [24,5,10].

4.3 Completeness in abstract interpretation

Alan Mycroft was one of the first [18] to develop completeness[8] in abstract interpretation, a subject that has since proliferated. His example of multiplication complete for sign analysis but not addition, is provided as introductory example in (almost) all papers on the subject! Moreover, [18] introduces predicate abstraction, somewhat before the standard reference [9]. Unfortunately fixpoints are postponed to later research. The fixpoint abstraction completeness problem is not yet solved satisfactorily.

[6] ironically close to decried "flowchart abstract interpretation".

[7] for example the classical liveness analysis in [6] is wrong since use is a syntactic over approximation of the semantic notion of use of a variable value, whereas, being negated, it should be an underapproximation, see Patrick Cousot: *Syntactic and Semantic Soundness of Structural Dataflow Analysis*. SAS 2019: 96-117

[8] Cousot and Cousot, POPL, 1979

[9] Susanne Graf, Hassen Saïdi: *Construction of Abstract State Graphs with PVS*. CAV 1997: 72-83

4.4 Types and Effects

Alan Mycroft most cited work is on types and type inference [21] and he has also maintained interest and important contributions on the subject [2,23], including effect systems [22,25]. Alan certainly understood the close link between types and abstract interpretation. An example is his thesis “There is an analogy between the system described here and the ”most general type” inference system used in a language such as ML [17]”[10] [16, page 67]. Another is “Control-flow operators also provide a link to abstract interpretation [6]. Primitive effectful operations in the concrete semantics are abstracted to effects in an effect algebra.” [22, page 18]. Finally in [18], he writes “Mycroft and Jones [10] manage to exhibit the Hindley-Milner type system as an abstract interpretation of the untyped -calculus but the result is much more unwieldly to use than the inference rule formulation.” (where [10] in this citation is our [19]).

He has certainly been too overwhelmed by new ideas to take time to explore this connection in more depth.

5 Conclusion

Starting from his PhD thesis [16], we have taken a rapid tour of Alan Mycroft’s publications related to abstract interpretation, where he played an outstanding pioneer role and went on to regularly introduce new inspiring ideas in the field. Personnaly, I would have hoped this he spend more time on the subject since he one of the few who, beyond theory, have a strong interest in applicability and practical applications as his other works in other domains do show.

References

1. Abramsky, S., Hankin, C. (eds.): Abstract Interpretation of Declarative Languages. Ellis Horwood (1987)
2. Dolan, S., Mycroft, A.: Polymorphism, subtyping, and type inference in mlsub. In: POPL. pp. 60–72. ACM (2017)
3. Ernoult, C., Mycroft, A.: Untyped strictness analysis. J. Funct. Program. **5**(1), 37–49 (1995)
4. Feigin, B., Mycroft, A.: Formally efficient program instrumentation. In: RV. Lecture Notes in Computer Science, vol. 6418, pp. 245–252. Springer (2010)
5. Gharat, P.M., Khedker, U.P., Mycroft, A.: Generalized points-to graphs: A precise and scalable abstraction for points-to analysis. ACM Trans. Program. Lang. Syst. **42**(2), 8:1–8:78 (2020)
6. Ivaskovic, A., Mycroft, A., Orchard, D.: Data-flow analyses as effects and graded monads. In: FSCD. LIPIcs, vol. 167, pp. 15:1–15:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)

[10] where [17] refers to Michael J. C. Gordon, Robin Milner, F. Lockwood Morris, Malcolm C. Newey, Christopher P. Wadsworth: *A Metalanguage for Interactive Proof in LCF*. POPL 1978: 119-130.

7. Jaiswal, S., Khedker, U.P., Mycroft, A.: A unified model for context-sensitive program analyses: : The blind men and the elephant. ACM Comput. Surv. **54**(6), 114:1–114:37 (2022)
8. Jones, N.D., Mycroft, A.: Data flow analysis of applicative programs using minimal function graphs. In: POPL. pp. 296–306. ACM Press (1986)
9. Jones, N.D., Nielson, F.: Abstract interpretation: a semantics-based tool for program analysis. In: Abramsky, S., Gabbay, D.M., Maibaum, T.S.E. (eds.) Handbook of logic in computer science. Volume 4. Semantic modelling, pp. 527–636. Clarendon Press (1995)
10. Khedker, U.P., Dhamdhere, D.M., Mycroft, A.: Bidirectional data flow analysis for type inferencing. Comput. Lang. Syst. Struct. **29**(1-2), 15–44 (2003)
11. Mauborgne, L.: Abstract interpretation using typed decision graphs. Sci. Comput. Program. **31**(1), 91–112 (1998)
12. McDermott, D., Mycroft, A.: Call-by-need effects via coeffects. Open Comput. Sci. **8**(1), 93–108 (2018)
13. McDermott, D., Mycroft, A.: Extended call-by-push-value: Reasoning about effectful programs and evaluation order. In: ESOP. Lecture Notes in Computer Science, vol. 11423, pp. 235–262. Springer (2019)
14. McDermott, D., Mycroft, A.: Galois connecting call-by-value and call-by-name. In: FSCD. LIPIcs, vol. 228, pp. 32:1–32:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
15. Mycroft, A.: The theory and practice of transforming call-by-need into call-by-value. In: Symposium on Programming. Lecture Notes in Computer Science, vol. 83, pp. 269–281. Springer (1980)
16. Mycroft, A.: Abstract interpretation and optimising transformations for applicative programs. Ph.D. thesis, University of Edinburgh, UK (1982)
17. Mycroft, A.: A study on abstract interpretation and 'validating microcode algebraically'. In: Abramsky and Hankin [1], pp. 199–218
18. Mycroft, A.: Completeness and predicate-based abstract interpretation. In: PEPM. pp. 179–185. ACM (1993)
19. Mycroft, A., Jones, N.D.: A relational framework for abstract interpretation. In: Programs as Data Objects. Lecture Notes in Computer Science, vol. 217, pp. 156–171. Springer (1985)
20. Mycroft, A., Nielson, F.: Strong abstract interpretation using power domains (extended abstract). In: ICALP. Lecture Notes in Computer Science, vol. 154, pp. 536–547. Springer (1983)
21. Mycroft, A., O'Keefe, R.A.: A polymorphic type system for prolog. In: Logic Programming Workshop. pp. 107–122. Núcleo de Intelligência Artificial, Universidade Nova De Lisboa, Portugal (1983)
22. Mycroft, A., Orchard, D.A., Petricek, T.: Effect systems revisited - control-flow algebra and semantics. In: Semantics, Logics, and Calculi. Lecture Notes in Computer Science, vol. 9560, pp. 1–32. Springer (2016)
23. Orchard, D.A., Mycroft, A.: Efficient and correct stencil computation via pattern matching and static typing. In: DSL. EPTCS, vol. 66, pp. 68–92 (2011)
24. Rodriguez-Prieto, O., Mycroft, A., Ortin, F.: An efficient and scalable platform for java source code analysis using overlaid graph representations. IEEE Access **8**, 72239–72260 (2020)
25. Schrijvers, T., Mycroft, A.: Strictness meets data flow. In: SAS. Lecture Notes in Computer Science, vol. 6337, pp. 439–454. Springer (2010)

26. Thompson, S., Mycroft, A.: Abstract interpretation of combinational asynchronous circuits. In: SAS. Lecture Notes in Computer Science, vol. 3148, pp. 181–196. Springer (2004)
27. Thompson, S., Mycroft, A.: Abstract interpretation of combinational asynchronous circuits. Sci. Comput. Program. **64**(1), 166–183 (2007)

A Proofs for Section 2 (Strictness Analysis)

Proof (proposition 1).

$$\begin{aligned}
&\quad \mathsf{post}(r)P \subseteq Q \\
&\Leftrightarrow \{\sigma' \mid \exists \sigma \,.\, \sigma \in P \wedge \langle \sigma,\, \sigma' \rangle \in r\} \subseteq Q && \wr\text{def. } \mathsf{post}\int \\
&\Leftrightarrow \forall \sigma' \,.\, (\exists \sigma \,.\, \sigma \in P \wedge \langle \sigma,\, \sigma' \rangle \in r) \Rightarrow \sigma' \in Q && \wr\text{def. } \subseteq\int \\
&\Leftrightarrow \forall \sigma' \,.\, \forall \sigma \,.\, (\sigma \in P \wedge \langle \sigma,\, \sigma' \rangle \in r) \Rightarrow \sigma' \in Q) && \wr\text{def. } \Rightarrow\int \\
&\Leftrightarrow \forall \sigma \,.\, \forall \sigma' \,.\, (\langle \sigma,\, \sigma' \rangle \in r) \Rightarrow (\sigma \in P \Rightarrow \sigma' \in Q) && \wr\text{def. } \forall \text{ and } \Rightarrow\int \\
&\Leftrightarrow r \subseteq \{\langle \sigma,\, \sigma' \rangle \mid \sigma \in P \Rightarrow \sigma' \in Q\} && \wr\text{def. } \subseteq\int \\
&\Leftrightarrow r \subseteq \gamma_P(Q) && \wr\text{by defining } \gamma_P(Q) = (P \times Q) \cup (\Sigma \setminus P) \times \Sigma\int \quad \square
\end{aligned}$$

Proof (proposition 2).

$$\begin{aligned}
&\quad \boldsymbol{\lambda} r \bullet \mathsf{post}(r)P \circ T(X) \\
&= \mathsf{post}(T(X))P && \wr\text{def. function composition } \circ\int \\
&= \mathsf{post}(\tau^0 \cup \tau \circ X)P && \wr\text{def. } T\int \\
&= \mathsf{post}(\tau^0)P \cup \mathsf{post}(\tau \circ X)P && \wr\mathsf{post}\text{ preserves arbitrary unions}\int \\
&= P \cup \mathsf{post}(\tau)(\mathsf{post}(X)P) && \wr\text{def. } \mathsf{post}\int \\
&= P \cup \mathsf{post}(\tau)(\boldsymbol{\lambda} r \bullet \mathsf{post}(r)(X)) && \wr\text{def. function application}\int \\
&= F_P(\boldsymbol{\lambda} r \bullet \mathsf{post}(r)(X)) && \wr \text{ with } F_P(X) \triangleq P \cup \mathsf{post}(\tau)X \int \\
&= F_P \circ \boldsymbol{\lambda} r \bullet \mathsf{post}(r)(X) && \wr\text{def. function composition } \circ\int \quad \square
\end{aligned}$$

Proof (proposition 3). Let f^n and $\bar{f}^n$, $\in \mathbb{N}$ be the iterates of f and $\bar{f}$ from the infima $\perp$ and $\bar{\perp}$. In a Galois connection, α preserves existing arbitrary joins in particular the infimum so $\alpha(f^0) = \alpha(\perp) = \bar{\perp} = \bar{f}^0$. Assume $\alpha(f^n) = \bar{f}^n$ by induction hypothesis. By the commutation condition, $\alpha(f^{n+1}) = \alpha(f(f^n)) = \bar{f}(\alpha(f^n)) = \bar{f}(\bar{f}^n) = \bar{f}^{n+1}$. By recurrence, $\forall n \in \mathbb{N} \,.\, \alpha(f^n) = \bar{f}^n$. Since f and $\bar{f}$ are continuous, they are increasing (isotone/monotone), so their iterates are increasing, and their limits do exist in the CPO. By continuity $\alpha(\mathsf{lfp}^{\preceq} f) = \alpha(\bigcup_{n \in \mathbb{N}} f^n) = \bigvee_{n \in \mathbb{N}} \alpha(f^n) = \bigvee_{n \in \mathbb{N}} \bar{f}^n = \mathsf{lfp}^{\bar{\preceq}} \bar{f}$. The proof is similar in the inequality case. $\square$

Proof (proposition 4).

$$\begin{aligned}
&\quad \mathsf{post}(f) \mathrel{\dot{\hat{\sqsubseteq}}} \phi \\
&\Leftrightarrow \forall P \in \wp(D_\perp) \,.\, \mathsf{post}(f)P \mathrel{\hat{\sqsubseteq}} \phi(P) && \wr\text{pointwise def. of } \dot{\hat{\sqsubseteq}}\int \\
&\Leftrightarrow \forall P \in \wp(D_\perp) \,.\, \{f(x) \mid x \in P\} \mathrel{\hat{\sqsubseteq}} \phi(P) && \wr\text{def. } \mathsf{post}\int \\
&\Leftrightarrow \forall P \in \wp(D_\perp) \,.\, (\forall x \in \{f(x) \mid x \in P\} \,.\, \exists y \in \phi(P) \,.\, x \sqsubseteq y \wedge \forall y \in \phi(P) \,.\, \exists x \in \\
&\qquad \{f(x) \mid x \in P\} \,.\, x \sqsubseteq y) && \wr\text{def. } \hat{\sqsubseteq}\int \\
&\Leftrightarrow \forall P \in \wp(D_\perp) \,.\, (\forall x \in P \,.\, \exists y \in \phi(P) \,.\, f(x) \sqsubseteq y \wedge \forall y \in \phi(P) \,.\, \exists x \in P \,.\, f(x) \sqsubseteq \\
&\qquad y) && \wr\text{def. } \in\int \\
&\Leftrightarrow \forall x \in D_\perp \,.\, f(x) \sqsubseteq \mathsf{let}\ \{y\} = \phi(\{x\})\ \mathsf{in}\ y
\end{aligned}$$

$\wr(\Rightarrow)$ Take $P = \{x\}$, $x \in D_\bot$, then $\exists y \in \phi(\{x\}) \;.\; f(x) \sqsubseteq y$. By $\phi \in \wp(\mathcal{D}_\bot) \xrightarrow{1\cup} \wp(\mathcal{D}_\bot)$, we have $\{y\} = \phi(\{x\})$ so $f(x) \sqsubseteq \mathsf{let}\ \{y\} = \phi(\{x\})\ \mathsf{in}\ y$.
$(\Leftarrow)$ Assume that $P \in \wp(D_\bot)$ and $x \in P$. Then $\{y\} = \phi(\{x\}) \subseteq \bigcup_{z\in P} \phi(\{z\}) = \phi(\bigcup_{z\in P}\{z\}) = \phi(P)$ proving $\exists y \in \phi(P) \;.\; f(x) \sqsubseteq y$.

Moreover, take any $y \in \phi(P) = \phi(\bigcup_{x\in P}\{x\}) = \bigcup_{x\in P} \phi(\{x\})$, there exists $x \in P$ such that $y \in \phi(\{x\})$, that is $\{y\} = \phi(\{x\})$ since $\phi(\{x\})$ is a singleton, proving $f(x) \sqsubseteq y$ by hypothesis.$\wr$

$\Leftrightarrow f \mathrel{\dot\sqsubseteq} \boldsymbol{\lambda} x \bullet \mathsf{let}\ \{y\} = \phi(\{x\})\ \mathsf{in}\ y$ $\wr$pointwise def. $\dot\sqsubseteq\wr$

$\Leftrightarrow f \mathrel{\dot\sqsubseteq} \hat\gamma(\phi)$ $\wr$def. $\hat\gamma\wr$

It follows that $\langle \mathcal{D}_\bot \longrightarrow \mathcal{D}_\bot,\ \dot\sqsubseteq\rangle \underset{\mathsf{post}}{\overset{\hat\gamma}{\leftrightarrows}} \langle \wp(\mathcal{D}_\bot) \xrightarrow{1\cup} \wp(\mathcal{D}_\bot),\ \dot{\hat\sqsubseteq}\rangle$ so that by restriction to the continuous functions, we also have $\langle \mathcal{D}_\bot \xrightarrow{c} \mathcal{D}_\bot,\ \dot\sqsubseteq\rangle \underset{\mathsf{post}}{\overset{\hat\gamma}{\leftrightarrows}} \langle \wp(\mathcal{D}_\bot) \xrightarrow{1\cup} \wp(\mathcal{D}_\bot),\ \dot{\hat\sqsubseteq}\rangle$. Moreover $\mathsf{post}(f)X = \emptyset$ if and only if $X = \emptyset$ so that $\langle \mathcal{D}_\bot \xrightarrow{c} \mathcal{D}_\bot,\ \dot\sqsubseteq\rangle \underset{\mathsf{post}}{\overset{\hat\gamma}{\leftrightarrows}} \langle \wp(\mathcal{D}_\bot \setminus \{\emptyset\}) \xrightarrow{1\cup} \wp(\mathcal{D}_\bot \setminus \{\emptyset\}),\ \dot{\hat\sqsubseteq}\rangle$. □

Proof (proposition 5). Given $F \in (\mathcal{D}_\bot \longrightarrow \mathcal{D}_\bot) \longrightarrow (\mathcal{D}_\bot \longrightarrow \mathcal{D}_\bot)$ and $f \in \mathcal{D}_\bot \longrightarrow \mathcal{D}_\bot$, we have

$\mathsf{post}(F(f))P$

$= \mathsf{post}(F(\boldsymbol{\lambda} x \bullet f(x)))P$ $\wr$def. $\boldsymbol{\lambda}$ notation$\wr$

$= \mathsf{post}(F(\boldsymbol{\lambda} x \bullet \mathsf{let}\ \{y\} = \{f(x)\}\ \mathsf{in}\ y))P$ $\wr$def. singleton equality$\wr$

$= \mathsf{post}(F(\boldsymbol{\lambda} x \bullet \mathsf{let}\ \{y\} = \{f(z) \mid z \in \{x\}\}\ \mathsf{in}\ y))P$ $\wr$def. $\in\wr$

$= \mathsf{post}(F(\boldsymbol{\lambda} x \bullet \mathsf{let}\ \{y\} = \boldsymbol{\lambda} X \bullet \{f(x) \mid x \in X\}(\{x\})\ \mathsf{in}\ y))P$ $\wr$def. application$\wr$

$= \mathsf{post}(F(\hat\gamma(\boldsymbol{\lambda} X \bullet \{f(x) \mid x \in X\})))P$ $\wr$def. $\hat\gamma\wr$

$= \mathsf{post}(F(\hat\gamma(\mathsf{post}(f))))P$ $\wr$def. $\mathsf{post}\wr$

$= \hat F(\mathsf{post}(f))P$ $\wr$def. $\hat F(\phi)P \triangleq \mathsf{post}(F(\hat\gamma(\phi)))P\wr$ □

Proof (proposition 6). Let $S \in \wp(D_\bot) \setminus \{\emptyset\}$ and $b \in \mathbb{B}$.

– $\alpha^\sharp(S) \le b$

$\Leftrightarrow (\!| S = \{\bot\}\ ¿\ 0 : 1 |\!) \le b$ $\wr$def. $\alpha^\sharp\wr$

$\Leftrightarrow (\!| S \subseteq \{\bot\}\ ¿\ 0 : 1 |\!) \le b$ $\wr$since $S \in \wp(D_\bot) \setminus \{\emptyset\}\wr$

$\Leftrightarrow (b = 0) \Rightarrow (S \subseteq \{\bot\})$ $\wr$by cases 0 or 1 for b with $0 \le 0$ and $1 \not\le 0\wr$

$\Leftrightarrow S \subseteq (\!| b = 0\ ¿\ \{\bot\} : D_\bot |\!)$ $\wr S \subseteq D_\bot\wr$

$\Leftrightarrow S \subseteq \gamma^\sharp(b)$ $\wr$def. $\gamma^\sharp\wr$

– $\alpha^\flat(S) \le b$

$\Leftrightarrow (\!| \bot \in S\ ¿\ 0 : 1 |\!) \le b$ $\wr$def. $\alpha^\flat\wr$

$\Leftrightarrow (b = 0) \Rightarrow (\bot \in S)$ $\wr$by cases 0 or 1 for b with $0 \le 0$ and $1 \not\le 0\wr$

$\Leftrightarrow (b = 1) \Rightarrow (\bot \notin S)$ $\quad$ $\wr$by cases $b = 0$ or $b = 1\wr$

$\Leftrightarrow S \subseteq (\!| b = 1 \;¿\; D : D_\bot |\!)$ $\quad$ $\wr$since $S \subseteq D_\bot\wr$

$\Leftrightarrow S \subseteq \gamma^\flat(b)$ $\quad$ $\wr$def. $\gamma^\flat\wr$ $\quad\square$

Proof (proposition 7). Let $\langle f^n,\ n \in \mathbb{N}\rangle$ be the increasing iterates of f from $\bot$ such that $\mathsf{lfp}^{\sqsubseteq} f = \bigsqcup_{n\in\mathbb{N}} f^n$. Let $\langle {f^\sharp}^n,\ n \in \mathbb{N}\rangle$ be the increasing iterates of $f^\sharp$ from $\bot^\sharp$ such that $\mathsf{lfp}^{\sqsubseteq^\sharp} f^\sharp = \bigsqcup^\sharp_{l\in\mathbb{N}} {f^\sharp}^n$. By (2), we have $f^0 = \bot \leq \gamma(\bot^\sharp) = \gamma({f^\sharp}^0)$. Assume by induction hypothesis that $f^n \leq \gamma({f^\sharp}^n)$. Then by (3), we have $f^{n+1} = f(f^n) \leq \gamma(f^\sharp({f^\sharp}^n)) = \gamma({f^\sharp}^{n+1})$, proving that $\forall n \in \mathbb{N} \,.\, f^n \leq \gamma({f^\sharp}^n)$. By (4), we have $\mathsf{lfp}^{\sqsubseteq} f = \bigsqcup_{n\in\mathbb{N}} f^n \leq \gamma(\bigsqcup^\sharp_{n\in\mathbb{N}} {f^\sharp}^n) = \gamma(\mathsf{lfp}^{\sqsubseteq^\sharp} f^\sharp)$.[11] $\quad\square$

Proof (of (5)).

$\vec{\alpha}^\sharp(f) \mathrel{\dot{\leq}} f^\sharp$

$\Leftrightarrow \forall b \in \mathbb{B} \,.\, \alpha^\sharp \circ f \circ \gamma^\sharp(b) \leq f^\sharp(b)$ $\quad$ $\wr$pointwise def. $\dot{\subseteq}$ and def. $\vec{\alpha}^\sharp\wr$

$\Leftrightarrow \forall b \in \mathbb{B} \,.\, f \circ \gamma^\sharp(b) \subseteq \gamma^\sharp \circ f^\sharp(b)$ $\quad$ $\wr$Galois connection in prop. (6)$\wr$

$\Leftrightarrow \forall X \in \wp(\mathcal{D}_\bot) \,.\, f(X) \subseteq \gamma^\sharp \circ f^\sharp \circ \alpha^\sharp(X)$

$\wr(\Leftarrow)$ Take $X = \gamma^\sharp(b)$ and $\gamma^\sharp \circ f^\sharp(b) \subseteq \gamma^\sharp \circ f^\sharp \circ \alpha^\sharp \circ \gamma^\sharp(b)$ since $\alpha^\sharp \circ \gamma^\sharp$ is reductive in $\langle \wp(D_\bot) \setminus \{\emptyset\},\ \subseteq\rangle \underset{\alpha^\sharp}{\overset{\gamma^\sharp}{\leftrightarrows}} \langle \mathbb{B},\ \leq\rangle$ of prop. 6, and $\gamma^\sharp$ and $f^\sharp(b)$ hence their composition is reductive.

$(\Rightarrow)$ $\gamma^\sharp \circ \alpha^\sharp$ is expansive in $\langle \wp(D_\bot) \setminus \{\emptyset\},\ \subseteq\rangle \underset{\alpha^\sharp}{\overset{\gamma^\sharp}{\leftrightarrows}} \langle \mathbb{B},\ \leq\rangle$ of prop. 6, f preserves joins so is increasing, so that taking $b = \alpha^\sharp(X)$ we have $f(X) \subseteq f \circ \gamma^\sharp \circ \alpha^\sharp(X) \subseteq \gamma^\sharp \circ f^\sharp(\alpha^\sharp(X)b)\wr$

$\Leftrightarrow \forall X \in \wp(\mathcal{D}_\bot) \,.\, f(X) \subseteq \vec{\gamma}^\sharp(f^\sharp)X$ $\quad$ $\wr$def. $\vec{\gamma}^\sharp\wr$

$\Leftrightarrow f \mathrel{\dot{\subseteq}} \vec{\gamma}^\sharp(f^\sharp)$ $\quad$ $\wr$pointwise def. $\dot{\subseteq}\wr$ $\quad\square$

Proof (of 8).

– $\bot \leq \vec{\gamma}^\sharp(\bot^\sharp)$

$\Leftrightarrow \boldsymbol{\lambda} X \bullet \{\bot\} \mathrel{\dot{\subseteq}} \vec{\gamma}^\sharp(\boldsymbol{\lambda} x \bullet 0)$ $\quad$ $\wr$def. $\bot$ and $\bot^\sharp\wr$

$\Leftrightarrow \forall X \in \wp(\mathcal{D}_\bot) \,.\, \{\bot\} \subseteq \gamma^\sharp \circ \boldsymbol{\lambda} x \bullet 0 \circ \alpha^\sharp(X)$ $\quad$ $\wr$pointwise def. of $\dot{\subseteq}$ and def. $\vec{\gamma}^\sharp\wr$

$\Leftrightarrow \{\bot\} \subseteq \gamma^\sharp(0)$ $\quad$ $\wr$def. function composition $\circ\wr$

$\Leftrightarrow \{\bot\} \subseteq \{\bot\}$ $\quad$ $\wr$def. $\gamma^\sharp$, which is true, proving (2)$\wr$

– Let $f \in \wp(\mathcal{D}_\bot) \xrightarrow{1\cup} \wp(\mathcal{D}_\bot)$, $f^\sharp \in \mathbb{B} \xrightarrow{i} \mathbb{B}$ be such that $f \leq \vec{\gamma}^\sharp(f^\sharp)$. Then

[11] We have not used the Galois connection hypothesis which is useful to rephrase the hypotheses using α. It is also possible to require equality, see Patrick Cousot, Radhia Cousot: *Galois Connection Based Abstract Interpretations for Strictness Analysis (Invited Paper).* Formal Methods in Programming and Their Applications 1993: 98-127.

$\hat{F}(f)$

$\dot{\subseteq}\ \hat{F}(\overrightarrow{\gamma}^\sharp(f^\sharp))$

⟅$f \leq \overrightarrow{\gamma}^\sharp(f^\sharp)$ and the composition $\hat{F}^\sharp$ of increasing functions is increasing⟆

$\dot{\subseteq}\ \overrightarrow{\gamma}^\sharp(\overrightarrow{\alpha}^\sharp \circ \hat{F} \circ \overrightarrow{\gamma}^\sharp(f^\sharp))$

⟅by the Galois connection $\langle\overrightarrow{\alpha}^\sharp,\ \overrightarrow{\gamma}^\sharp\rangle$, $\overrightarrow{\gamma}^\sharp \circ \overrightarrow{\alpha}^\sharp$ is extensive⟆

$=\ \overrightarrow{\gamma}^\sharp(\hat{F}^\sharp(f^\sharp))$ ⟅def. $\hat{F}^\sharp \triangleq \overrightarrow{\alpha}^\sharp \circ \hat{F} \circ \overrightarrow{\gamma}^\sharp$, proving hypothesis (3) of proposition 7⟆

– Let $\langle f_i,\ i \in \mathbb{N}\rangle$ be an increasing chain for $\dot{\hat{\sqsubseteq}}$ and $\langle f_i^\sharp,\ i \in \mathbb{N}\rangle$ be an increasing chain for $\dot{\leq}$ such that $\forall i \in \mathbb{N} \mathrel{.} f_i \mathrel{\dot{\subseteq}} \overrightarrow{\gamma}^\sharp(f_i^\sharp)$. We have

$\dot{\hat{\bigsqcup_{i\in\mathbb{N}}}} f_i \mathrel{\dot{\subseteq}} \overrightarrow{\gamma}^\sharp(\dot{\bigvee_{j\in\mathbb{N}}} f_j^\sharp)$

$\Leftrightarrow \forall X \in \wp(\mathcal{D}_\bot) \mathrel{.} \hat{\bigsqcup_{i\in\mathbb{N}}} f_i(X) \subseteq \overrightarrow{\gamma}^\sharp(\dot{\bigvee_{j\in\mathbb{N}}} f_j^\sharp)(X)$ ⟅pointwise def. $\dot{\hat{\sqsubseteq}}$ and $\dot{\leq}$⟆

⟅proving (4)⟆

$\Leftrightarrow \forall X \in \wp(\mathcal{D}_\bot) \mathrel{.} \hat{\bigsqcup_{i\in\mathbb{N}}} f_i(X) \subseteq \gamma^\sharp \circ (\dot{\bigvee_{j\in\mathbb{N}}} f_j^\sharp) \circ \alpha^\sharp(X)$ ⟅def. $\overrightarrow{\gamma}^\sharp(f) \triangleq \gamma^\sharp \circ f \circ \alpha^\sharp$⟆

$\Leftrightarrow \forall X \in \wp(\mathcal{D}_\bot) \mathrel{.} \hat{\bigsqcup_{i\in\mathbb{N}}} f_i(X) \subseteq \gamma^\sharp(\bigvee_{j\in\mathbb{N}} f_j^\sharp(\alpha^\sharp(X)))$ ⟅def. composition $\circ$⟆ (6)

Since $\langle f_j^\sharp,\ j \in \mathbb{N}\rangle$ is $\dot{\leq}$-increasing, $\langle f_j^\sharp(\alpha^\sharp(X)),\ j \in \mathbb{N}\rangle$ is $\leq$-increasing so is either a sequence of 0s or a sequence of 0s followed by 1s. There are two cases.

- If $\langle f_j^\sharp(\alpha^\sharp(X)),\ j \in \mathbb{N}\rangle$ is a sequence of 0s then $\bigvee_{j\in\mathbb{N}} f_j^\sharp(\alpha^\sharp(X)) = 0$. Moreover the hypothesis $\forall i \in \mathbb{N} \mathrel{.} f_i \mathrel{\dot{\subseteq}} \overrightarrow{\gamma}^\sharp(f_i^\sharp) = \gamma^\sharp \circ f_i^\sharp \circ \alpha^\sharp$ implies that $\forall i \in \mathbb{N} \mathrel{.} f_i(X) = \gamma^\sharp(f_i^\sharp(\alpha^\sharp(X))) = \gamma^\sharp(0) = \{\bot\}$. Therefore

⟅(6)⟆

$\Leftrightarrow \hat{\bigsqcup_{i\in\mathbb{N}}} f_i(X) \subseteq \gamma^\sharp(0)$ ⟅case $\langle f_j^\sharp(\alpha^\sharp(X)),\ j \in \mathbb{N}\rangle = \langle 0,\ j \in \mathbb{N}\rangle$⟆

$\Leftrightarrow \hat{\bigsqcup_{i\in\mathbb{N}}} \{\bot\} \subseteq \{\bot\}$ ⟅def. $\gamma^\sharp(b) = \llparenthesis b = 0 \mathrel{¿} \{\bot\} : D_\bot \rrparenthesis$⟆

$\Leftrightarrow \{\bot\} \subseteq \{\bot\}$

⟅by def. of the least upper bound $\hat{\bigsqcup}$ for $\hat{\sqsubseteq}$, proving (4) in this first case⟆

- Otherwise $\langle f_j^\sharp(\alpha^\sharp(X)), j \in \mathbb{N}\rangle$ is a sequence of 0s followed by 1s so $\bigvee_{j\in\mathbb{N}} f_j^\sharp(\alpha^\sharp(X))$ $= 1$ in (6) and therefore $\hat{\bigsqcup_{i\in\mathbb{N}}} f_i(X) \subseteq \gamma^\sharp(1) = D_\bot$ since $\hat{\bigsqcup_{i\in\mathbb{N}}} f_i(X) \in \wp(D_\bot)$ by def. of the lub in the poset $\langle \wp(D_\bot),\ \hat{\sqsubseteq}\rangle$. Again (4) holds in this second case. $\square$