Types as Abstract Interpretations

(invited paper)

Patrick Cousot
LIENS, École Normale Supérieure
45, rue d'Ulm
75230 Paris cedex 05 (France)
cousot@dmi.ens.fr, http://www.ens.fr/~cousot

Abstract

Starting from a denotational semantics of the eager untyped lambda-calculus with explicit runtime errors, the standard collecting semantics is defined as specifying the strongest program properties. By a first abstraction, a new sound type collecting semantics is derived in compositional fixpoint form. Then by successive (semi-dual) Galois connection based abstractions, type systems and/or type inference algorithms are designed as abstract semantics or abstract interpreters approximating the type collecting semantics. This leads to a hierarchy of type systems, which is part of the lattice of abstract interpretations of the untyped lambda-calculus. This hierarchy includes two new à la Church/Curry polytype systems. Abstractions of this polytype semantics lead to classical Milner/Mycroft and Damas/-Milner polymorphic type schemes, Church/Curry monotypes and Hindley principal typing algorithm. This shows that types are abstract interpretations.

1 Introduction

The leading idea of abstract interpretation [6, 7, 9, 12] is that program semantics, proof and static analysis methods have common structures which can be exhibited by abstraction of the structure of run-time computations. This leads to an organization of the more or less approximate or refined semantics into a lattice of abstract interpretations. This unifying point of view allows for a synthetic understanding of a wide range of works from theoretical semantical specifications to practical static analysis algorithms.

It will be shown that this point of view can be applied to type theory, in particular to type soundness and à la Curry type inference which, following [17, 29], have been dominating research themes in programming languages during the last two decades, at least for functional programming languages [1, 19, 31]. Traditionally the design of a type system *"involves defining the notion of type error for a given language, formalizing the type system by a set of type rules, and verifying that program execution of well-typed programs cannot produce type errors. This process, if successful, guarantees the type-soundness of a language as a whole. Type-checking algorithms can then be developed as a separate concern, and their correctness can be verified with respect to a given type system; this process guarantees that type checkers satisfy the language definition."* [2]. Abstract interpretation allows viewing all these different aspects in the more unifying framework of semantic approximation. Formalization of program analysis and type systems within the same abstract interpretation framework should lead to a better understanding of the relationship between these seemingly different approaches to program correctness and optimization.

2 Syntax

The syntax of the untyped eager lambda calculus is:

$$\begin{aligned} \mathtt{x}, \mathtt{f}, \ldots \in \mathbb{X} &\quad : \quad \text{program variables} \\ e \in \mathbb{E} &\quad : \quad \text{program expressions} \\ e &::= \mathtt{x} \mid \boldsymbol{\lambda}\mathtt{x} \cdot e \mid e_1(e_2) \mid \boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e \mid \\ &\quad\ \mathbf{1} \mid e_1 - e_2 \mid (e_1 \mathbin{\mathbf{?}} e_2 \mathbin{\mathbf{:}} e_3) \end{aligned}$$

$\boldsymbol{\lambda}\mathtt{x} \cdot e$ is the lambda abstraction and $e_1(e_2)$ the application. In $\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e$, the function $\mathtt{f}$ with formal parameter $\mathtt{x}$ is defined recursively. $(e_1 \mathbin{\mathbf{?}} e_2 \mathbin{\mathbf{:}} e_3)$ is the test for zero.

3 Denotational Semantics

The semantic domain $\mathbb{S}$ is defined by the following equations [20]:

$$\begin{aligned} \mathbb{W} &\triangleq \{\omega\} && \text{wrong} \\ \mathsf{z} \in \mathbb{Z} & && \text{integers} \\ \mathsf{u}, \mathsf{f}, \varphi \in \mathbb{U} &\cong \mathbb{W}_\perp \oplus \mathbb{Z}_\perp \oplus [\mathbb{U} \mapsto \mathbb{U}]_\perp && \text{values} \\ \mathsf{R} \in \mathbb{R} &\triangleq \mathbb{X} \mapsto \mathbb{U} && \text{environments} \\ \phi \in \mathbb{S} &\triangleq \mathbb{R} \mapsto \mathbb{U} && \text{semantic domain} \end{aligned}$$

where ω is the wrong value, $\perp$ denotes non-termination, $D_\perp$ is the lift of domain D (with up injection $\uparrow(\bullet) \in D \mapsto D_\perp$ and partial down injection $\downarrow(\bullet) \in D_\perp \not\mapsto D$), $D_1 \oplus D_2$ is the coalesced sum of domains D_1 and D_2 (with left and right injections $\bullet :: D_1 \in D_1 \mapsto D_1 \oplus D_2$ and $\bullet :: D_2 \in D_2 \mapsto D_1 \oplus D_2$), $\Omega \triangleq \uparrow(\omega) :: \mathbb{W}_\perp$ and $[D_1 \mapsto D_2]$ is the domain of continuous, $\perp$-strict, Ω-strict functions from D_1 into D_2. $\sqsubseteq$ is the computational ordering on $\mathbb{U}$ and $\sqcup$ is the least upper bound (lub) of increasing chains.

In the metalanguage for defining the denotational semantics below, $\Lambda x \cdot \ldots$ or $\Lambda x \in S \cdot \ldots$ is the lambda abstraction.

POPL 97, Paris, France

$(\dots\ ?\ \dots \mid \dots\ ?\ \dots \mid \dots)$ is the conditional expression. The assignment $\mathsf{R}[\mathtt{x}\leftarrow\mathsf{u}]$ is such that $\mathsf{R}[\mathtt{x}\leftarrow\mathsf{u}](\mathtt{x}) = \mathsf{u}$ and $\mathsf{R}[\mathtt{x}\leftarrow\mathsf{u}](\mathtt{y}) = \mathsf{R}(\mathtt{y})$ when $\mathtt{x} \neq \mathtt{y}$. $\mathbf{lfp}^{\sqsubseteq}_{\perp}\ \varphi$ is the $\sqsubseteq$-least fixpoint of the monotone operator $\varphi \in L \mapsto L$ on a complete partial order (cpo) $\langle L, \sqsubseteq\rangle$, which is greater than or equal to $\perp$ [8, 20].

The denotational semantics [19, 29]

$$\mathbf{S}[\![\bullet]\!] \in \mathbb{E} \mapsto \mathbb{S}$$

defines a call-by-value evaluation with run-time type check and propagation of errors:

$$\begin{aligned}
\mathbf{S}[\![\mathtt{x}]\!] &\triangleq \Lambda\mathsf{R}\cdot \mathsf{R}(\mathtt{x})\\
\mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!] &\triangleq \Lambda\mathsf{R}\cdot \uparrow\!\big(\Lambda\mathsf{u}\cdot(\mathsf{u} = \perp \vee \mathsf{u} = \Omega\ ?\ \mathsf{u} \mid\\
&\qquad \mathbf{S}[\![e]\!]\mathsf{R}[\mathtt{x}\leftarrow\mathsf{u}])\big) :: [\mathbb{U} \mapsto \mathbb{U}]_{\perp}\\
\mathbf{S}[\![e_1(e_2)]\!] &\triangleq \Lambda\mathsf{R}\cdot(\mathbf{S}[\![e_1]\!]\mathsf{R} = \perp \vee \mathbf{S}[\![e_2]\!]\mathsf{R} = \perp\ ?\ \perp \mid\\
&\qquad \mathbf{S}[\![e_1]\!]\mathsf{R} = \mathsf{f} :: [\mathbb{U} \mapsto \mathbb{U}]_{\perp}\ ?\ \downarrow\!(\mathsf{f})\big(\mathbf{S}[\![e_2]\!]\mathsf{R}\big) \mid\\
&\qquad \Omega)\\
\mathbf{S}[\![\boldsymbol{\mu}\mathtt{f}\cdot\boldsymbol{\lambda}\mathtt{x}\cdot e]\!] &\triangleq \Lambda\mathsf{R}\cdot \mathbf{lfp}^{\sqsubseteq}_{\uparrow(\Lambda\mathsf{u}\cdot\perp)::[\mathbb{U}\mapsto\mathbb{U}]_{\perp}}\ \Lambda\varphi\cdot \mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\mathsf{R}[\mathtt{f}\leftarrow\varphi]\\
\mathbf{S}[\![\mathbf{1}]\!] &\triangleq \Lambda\mathsf{R}\cdot \uparrow\!(1) :: \mathbb{Z}_{\perp}\\
\mathbf{S}[\![e_1 - e_2]\!] &\triangleq \Lambda\mathsf{R}\cdot(\mathbf{S}[\![e_1]\!]\mathsf{R} = \perp \vee \mathbf{S}[\![e_2]\!]\mathsf{R} = \perp\ ?\ \perp \mid\\
&\qquad \mathbf{S}[\![e_1]\!]\mathsf{R} = \mathsf{z}_1 :: \mathbb{Z}_{\perp} \wedge \mathbf{S}[\![e_2]\!]\mathsf{R} = \mathsf{z}_2 :: \mathbb{Z}_{\perp}\ ?\\
&\qquad \uparrow\!(\downarrow\!(\mathsf{z}_1) - \downarrow\!(\mathsf{z}_2)) :: \mathbb{Z}_{\perp} \mid \Omega)\\
\mathbf{S}[\![(e_1\ \mathbf{?}\ e_2\ \mathbf{:}\ e_3)]\!] &\triangleq \Lambda\mathsf{R}\cdot(\mathbf{S}[\![e_1]\!]\mathsf{R} = \perp\ ?\ \perp \mid \mathbf{S}[\![e_1]\!]\mathsf{R} =\\
&\qquad \mathsf{z} :: \mathbb{Z}_{\perp}\ ?\ (\downarrow\!(\mathsf{z}) = 0\ ?\ \mathbf{S}[\![e_2]\!]\mathsf{R} \mid \mathbf{S}[\![e_3]\!]\mathsf{R}) \mid\\
&\qquad \Omega)
\end{aligned}$$

The choice of a denotational semantics instead of the usual operational semantics follows [29] and corresponds to an initial abstraction [12] where the notion of computation step is lost. The relational semantics of [30] was initially chosen but finally abandoned in favor of a semantic model where the use of fixpoints is explicit both in the standard and type semantics. The denotational semantics introduces an initial approximation which prevents discussing notions linked to computation steps such as subject reduction [31]. It illustrates the direct application of the Galois connection based abstract interpretation framework [6, 7, 9] initially conceived for transition systems, that is small-step operational semantics, to denotational semantics.

4 Standard Collecting Semantics

A semantic property is a set of possible semantics of a program. The set of semantic properties:

$$P \in \mathbb{P} \triangleq \wp(\mathbb{S})$$

is a complete boolean lattice $\langle\mathbb{P}, \subseteq, \emptyset, \mathbb{S}, \cup, \cap, \neg\rangle$ for subset inclusion $\subseteq$, that is logical implication.

The standard collecting semantics:

$$\mathbf{C}[\![\bullet]\!] \in \mathbb{E} \mapsto \mathbb{P}$$
$$\mathbf{C}[\![e]\!] \triangleq \{\mathbf{S}[\![e]\!]\}$$

is the strongest program property.

Type systems are understood as abstract compositional semantics of the lambda calculus which approximate the standard collecting semantics. The à la Church/Curry simple type system [1] is first considered.

5 Church/Curry Monotype Semantics

The type of a program e is a set of typings $\langle H, m\rangle$ stating that the standard evaluation of the program e in an environment R where global variables $\mathtt{x}$ have type $H(\mathtt{x})$ given by type environment H returns a value $\mathbf{S}[\![e]\!]\mathsf{R}$ with monotype m. For example, program $\boldsymbol{\lambda}\mathtt{x}\cdot\mathtt{x}$ has type $\{\langle H, m \to m\rangle \mid H \in \mathbb{H}^{\mathrm{C}} \wedge m \in \mathbb{M}^{\mathrm{C}}\}$. The syntax of types and type semantic domains are:

$m \in \mathbb{M}^{\mathrm{C}},\quad m ::= \mathtt{int} \mid m_1 \to m_2$	monotype
$H \in \mathbb{H}^{\mathrm{C}} \triangleq \mathbb{X} \mapsto \mathbb{M}^{\mathrm{C}}$	type environment
$\theta \in \mathbb{I}^{\mathrm{C}} \triangleq \mathbb{H}^{\mathrm{C}} \times \mathbb{M}^{\mathrm{C}}$	typing
$T \in \mathbb{T}^{\mathrm{C}} \triangleq \wp(\mathbb{I}^{\mathrm{C}})$	program type

The meaning of types is defined by the *concretization function* γ^{C} as follows:

$$\begin{aligned}
\gamma_1^{\mathrm{C}} &\in \mathbb{M}^{\mathrm{C}} \mapsto \wp(\mathbb{U})\\
\gamma_1^{\mathrm{C}}(\mathtt{int}) &\triangleq \{\uparrow\!(\mathsf{z}) :: \mathbb{Z}_{\perp} \mid \mathsf{z} \in \mathbb{Z}\} \cup \{\perp\}\\
\gamma_1^{\mathrm{C}}(m_1 \to m_2) &\triangleq \{\uparrow\!(\varphi) :: [\mathbb{U} \mapsto \mathbb{U}]_{\perp} \mid \varphi \in [\mathbb{U} \mapsto \mathbb{U}] \wedge\\
&\qquad \forall \mathsf{u} \in \gamma_1^{\mathrm{C}}(m_1) : \varphi(\mathsf{u}) \in \gamma_1^{\mathrm{C}}(m_2)\} \cup \{\perp\}\\
\gamma_2^{\mathrm{C}} &\in \mathbb{H}^{\mathrm{C}} \mapsto \wp(\mathbb{R})\\
\gamma_2^{\mathrm{C}}(H) &\triangleq \{\mathsf{R} \in \mathbb{R} \mid \forall \mathtt{x} \in \mathbb{X} : \mathsf{R}(\mathtt{x}) \in \gamma_1^{\mathrm{C}}(H(\mathtt{x}))\}\\
\gamma_3^{\mathrm{C}} &\in \mathbb{I}^{\mathrm{C}} \mapsto \mathbb{P}\\
\gamma_3^{\mathrm{C}}(\langle H, m\rangle) &\triangleq \{\phi \in \mathbb{S} \mid \forall \mathsf{R} \in \gamma_2^{\mathrm{C}}(H) : \phi(\mathsf{R}) \in \gamma_1^{\mathrm{C}}(m)\}\\
\gamma^{\mathrm{C}} &\in \mathbb{T}^{\mathrm{C}} \mapsto \mathbb{P}\\
\gamma^{\mathrm{C}}(T) &\triangleq \bigcap_{\theta \in T} \gamma_3^{\mathrm{C}}(\theta), \qquad \gamma^{\mathrm{C}}(\emptyset) \triangleq \mathbb{S}
\end{aligned}$$

One has:

$$\gamma^{\mathrm{C}}\big(\bigcup_{i \in \Delta} T_i\big) = \bigcap_{i \in \Delta} \gamma^{\mathrm{C}}(T_i)$$

so that there exists a unique upper adjoint $\alpha^{\mathrm{C}} \in \mathbb{P} \mapsto \mathbb{T}^{\mathrm{C}}$ such that $\langle\alpha^{\mathrm{C}}, \gamma^{\mathrm{C}}\rangle$ is a *Galois connection* [9], a fact that is denoted:

$$\langle\mathbb{P}, \subseteq\rangle \underset{\alpha^{\mathrm{C}}}{\overset{\gamma^{\mathrm{C}}}{\leftrightarrows}} \langle\mathbb{T}^{\mathrm{C}}, \supseteq\rangle$$

The notation $\langle L, \leq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \preceq\rangle$ for semi-dual Galois connections (that are called Galois connection for short) means that $\langle L, \leq\rangle$ and $\langle M, \preceq\rangle$ are posets such that the pair of *abstraction function* $\alpha \in L \mapsto M$ and *concretization function* $\gamma \in M \mapsto L$ satisfy, for all $x \in L$ and $y \in M$:

$$\alpha(x) \preceq y \iff x \leq \gamma(y)$$

The abstraction α preserves existing lubs while the concretization γ preserves existing glbs. Reciprocally, if α preserves existing lubs or γ preserves existing glbs then there is a unique adjoint such that $\langle L, \leq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \preceq\rangle$ where $\alpha(x) = \mathrm{glb}\{y \mid x \leq \gamma(y)\}$ and $\gamma(y) = \mathrm{lub}\{x \mid \alpha(x) \preceq y\}$.

The intuition is that $\alpha^{\mathrm{C}}(P)$ is the best possible (i.e. most precise) type of programs with property P. This is in contrast with the use of *logical* or *algebraic relations* [24, 30, 31, 38] for which no such notion exists [1]. The critique

[1] The left image $\gamma(y) \triangleq \{x \mid \mathcal{L}(x, y)\}$ of the logical relation $\mathcal{L}$ is the usual concretization towards the standard collecting semantics so the use of a logical relation amounts to the use of a concretization only, the implicit corresponding abstraction being unused.

of the use of Galois connections by [38] relies, as often, on the use of a collecting semantics of type $\mathbf{C}'[\![e]\!] \in \wp(\mathbb{R}) \mapsto \wp(\mathbb{U})$, which is inadequate anyway because it is too abstract for types, a phenomenon already observed by [13] in the context of comportment analysis. The above Galois connection clearly shows that logical implication of program properties $\subseteq$ is abstracted by superset inclusion $\supseteq$ of typings, hence the use of co-induction in [30]. More importantly, the abstraction function is essential in the design of the abstract type semantics, as observed in [11] and illustrated below.

The Church/Curry monotype type semantics:

$$\mathbf{T}^{\mathrm{C}}[\![\bullet]\!] \in \mathbb{E} \mapsto \mathbb{T}^{\mathrm{C}}$$

defines the type $\mathbf{T}^{\mathrm{C}}[\![e]\!]$ of program e. The type semantics $\mathbf{T}^{\mathrm{C}}[\![\bullet]\!]$ is said to be *sound* in that for all programs $e \in \mathbb{E}$, one has:

$$\begin{array}{rrcl} & \alpha^{\mathrm{C}}(\mathbf{C}[\![e]\!]) & \supseteq & \mathbf{T}^{\mathrm{C}}[\![e]\!] \\ \Longleftrightarrow & \mathbf{C}[\![e]\!] & \subseteq & \gamma^{\mathrm{C}}(\mathbf{T}^{\mathrm{C}}[\![e]\!]) \\ \Longleftrightarrow & \mathbf{S}[\![e]\!] & \in & \gamma^{\mathrm{C}}(\mathbf{T}^{\mathrm{C}}[\![e]\!]) \end{array}$$

This correctness condition for the abstract Church/Curry type semantics $\mathbf{T}^{\mathrm{C}}[\![\bullet]\!]$ is the classical $\subseteq$-upper *concrete* approximation of the collecting semantics $\mathbf{C}[\![e]\!]$. The corresponding *abstract* approximation is $\supseteq$-upper (hence $\subseteq$-lower, which should not be confusing). It follows that any other abstract semantics $\mathbf{T}'^{\mathrm{C}}[\![e]\!] \subseteq \mathbf{T}^{\mathrm{C}}[\![e]\!]$ is also sound.

A program e is *typable* if and only if $\mathbf{T}^{\mathrm{C}}[\![e]\!] \neq \emptyset$. Soundness implies that *typable programs cannot go wrong* [29] in that:

$$\langle H,\ m\rangle \in \mathbf{T}^{\mathrm{C}}[\![e]\!] \wedge \mathsf{R} \in \gamma_2^{\mathrm{C}}(H) \wedge \mathbf{S}[\![e]\!]\mathsf{R} \neq \bot \Longrightarrow \mathbf{S}[\![e]\!]\mathsf{R} \neq \Omega$$

(since, by definition of γ^{C}, $\mathbf{S}[\![e]\!]\mathsf{R} \in \gamma_1^{\mathrm{C}}(m)$ and $\Omega \notin \gamma_1^{\mathrm{C}}(m)$).

For clarity of the presentation, the design of the Church/-Curry monotype semantics $\mathbf{T}^{\mathrm{C}}[\![\bullet]\!]$ by abstract interpretation of the collecting semantics $\mathbf{C}[\![\bullet]\!]$ is postponed to **Sec. 7**. Anyway the result is well-known:

$$\begin{aligned}
\mathbf{T}^{\mathrm{C}}[\![\mathtt{x}]\!] &\triangleq \{\langle H,\ H(\mathtt{x})\rangle \mid H \in \mathbb{H}^{\mathrm{C}}\} \\
\mathbf{T}^{\mathrm{C}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \{\langle H,\ m_1 \to m_2\rangle \mid \\ &\qquad\qquad \langle H[\mathtt{x} \leftarrow m_1],\ m_2\rangle \in \mathbf{T}^{\mathrm{C}}[\![e]\!]\} \\
\mathbf{T}^{\mathrm{C}}[\![e_1(e_2)]\!] &\triangleq \{\langle H,\ m_2\rangle \mid \langle H,\ m_1 \to m_2\rangle \in \\ &\qquad\qquad \mathbf{T}^{\mathrm{C}}[\![e_1]\!] \wedge \langle H,\ m_1\rangle \in \mathbf{T}^{\mathrm{C}}[\![e_2]\!]\} \\
\mathbf{T}^{\mathrm{C}}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \{\langle H,\ m\rangle \mid \langle H[\mathtt{f} \leftarrow m],\ m\rangle \in \mathbf{T}^{\mathrm{C}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\} \\
\mathbf{T}^{\mathrm{C}}[\![\mathtt{1}]\!] &\triangleq \{\langle H,\ \mathtt{int}\rangle \mid H \in \mathbb{H}^{\mathrm{C}}\} \\
\mathbf{T}^{\mathrm{C}}[\![e_1 - e_2]\!] &\triangleq \{\langle H,\ \mathtt{int}\rangle \mid \\ &\qquad\qquad \langle H,\ \mathtt{int}\rangle \in \mathbf{T}^{\mathrm{C}}[\![e_1]\!] \cap \mathbf{T}^{\mathrm{C}}[\![e_2]\!]\} \\
\mathbf{T}^{\mathrm{C}}[\![(e_1 \ \mathtt{?}\ e_2 \ \mathtt{:}\ e_3)]\!] &\triangleq \{\langle H,\ m\rangle \mid \langle H,\ \mathtt{int}\rangle \in \mathbf{T}^{\mathrm{C}}[\![e_1]\!] \wedge \\ &\qquad\qquad \langle H,\ m\rangle \in \mathbf{T}^{\mathrm{C}}[\![e_2]\!] \cap \mathbf{T}^{\mathrm{C}}[\![e_3]\!]\}
\end{aligned}$$

Like any inductive definition on a poset, the Church/-Curry monotype semantics can be presented in equivalent rule based form [15]. For that purpose, a set of rules of the form:

$$\frac{\theta_1 \in \mathbf{T}^{\mathrm{C}}[\![e_1]\!], \ldots, \theta_n \in \mathbf{T}^{\mathrm{C}}[\![e_n]\!], \theta \in \mathbf{T}^{\mathrm{C}}[\![e(e_1, \ldots, e_n)]\!]}{\Psi_e^i(\theta_1, \ldots, \theta_n, \theta) \in \mathbf{T}^{\mathrm{C}}[\![e(e_1, \ldots, e_n)]\!]}\ i \in \Delta$$

for a program expression $e(e_1, \ldots, e_n)$ with immediate components $e_1, \ldots, e_n$ is interpreted as:

$$\begin{aligned}\mathbf{T}^{\mathrm{C}}[\![e(e_1, \ldots, e_n)]\!] \triangleq \mathbf{lfp}_{\emptyset}^{\subseteq}\ \Lambda X \cdot \{\Psi_e^i(\theta_1, \ldots, \theta_n, \theta) \mid i \in \Delta \wedge \\ \theta_1 \in \mathbf{T}^{\mathrm{C}}[\![e_1]\!] \wedge \ldots \wedge \theta_n \in \mathbf{T}^{\mathrm{C}}[\![e_n]\!] \wedge \theta \in X\}\end{aligned}$$

where, by structural induction, the $\mathbf{T}^{\mathrm{C}}[\![e_i]\!]$, $i = 1, \ldots, n$ are already defined while $\mathbf{T}^{\mathrm{C}}[\![e(e_1, \ldots, e_n)]\!]$ is defined inductively. At least one of the rules should not be recursive (else $\mathbf{T}^{\mathrm{C}}[\![e(e_1, \ldots, e_n)]\!] = \emptyset$):

$$\frac{\theta_1 \in \mathbf{T}^{\mathrm{C}}[\![e_1]\!], \ldots, \theta_n \in \mathbf{T}^{\mathrm{C}}[\![e_n]\!]}{\Psi_e^j(\theta_1, \ldots, \theta_n) \in \mathbf{T}^{\mathrm{C}}[\![e(e_1, \ldots, e_n)]\!]}$$

If all rules are of that form, the fixpoint of a constant function reduces to:

$$\begin{aligned}\mathbf{T}^{\mathrm{C}}[\![e(e_1, \ldots, e_n)]\!] \triangleq \{\Psi_e^i(\theta_1, \ldots, \theta_n) \mid i \in \Delta \wedge \\ \theta_1 \in \mathbf{T}^{\mathrm{C}}[\![e_1]\!] \wedge \ldots \wedge \theta_n \in \mathbf{T}^{\mathrm{C}}[\![e_n]\!]\}\end{aligned}$$

This is the case for Church/Curry monotype semantics, which, by defining judgments as:

$$H \overset{\mathrm{C}}{\vdash} e \Rightarrow m \triangleq \langle H,\ m\rangle \in \mathbf{T}^{\mathrm{C}}[\![e]\!]$$

can be presented in the equivalent rule based form:

$$H \overset{\mathrm{C}}{\vdash} \mathtt{x} \Rightarrow H(\mathtt{x}) \qquad \frac{H \overset{\mathrm{C}}{\vdash} e_1 \Rightarrow m_1 \to m_2,\ H \overset{\mathrm{C}}{\vdash} e_2 \Rightarrow m_1}{H \overset{\mathrm{C}}{\vdash} e_1(e_2) \Rightarrow m_2}$$

$$\frac{H[\mathtt{x} \leftarrow m_1] \overset{\mathrm{C}}{\vdash} e \Rightarrow m_2}{H \overset{\mathrm{C}}{\vdash} \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow m_1 \to m_2} \qquad \frac{H[\mathtt{f} \leftarrow m] \overset{\mathrm{C}}{\vdash} \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow m}{H \overset{\mathrm{C}}{\vdash} \boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow m}$$

$$H \overset{\mathrm{C}}{\vdash} \mathtt{1} \Rightarrow \mathtt{int} \qquad \frac{H \overset{\mathrm{C}}{\vdash} e_1 \Rightarrow \mathtt{int},\ H \overset{\mathrm{C}}{\vdash} e_2 \Rightarrow \mathtt{int}}{H \overset{\mathrm{C}}{\vdash} e_1 - e_2 \Rightarrow \mathtt{int}}$$

$$\frac{H \overset{\mathrm{C}}{\vdash} e_1 \Rightarrow \mathtt{int},\ H \overset{\mathrm{C}}{\vdash} e_2 \Rightarrow m,\ H \overset{\mathrm{C}}{\vdash} e_3 \Rightarrow m}{H \overset{\mathrm{C}}{\vdash} (e_1 \ \mathtt{?}\ e_2 \ \mathtt{:}\ e_3) \Rightarrow m}$$

The functional versus the rule-based presentation of the abstract semantics is only a superfluous difference between abstract interpretation and type theory. The main difference seems to be that in type theory the above rule based assignment of Church/Curry monotypes to program expressions is taken for granted [1] whereas in abstract interpretation it is constructively derived from the collecting semantics and the $\langle \alpha,\ \gamma\rangle$ abstraction-concretization pair (as shown thereafter). In particular changing the standard semantics $\mathbf{S}[\![\bullet]\!]$ from an eager call-by-value to a lazy call-by-name would change the considered type systems (as e.g. in [34]).

6 À la Church/Curry Polytype Semantics

The use of abstract interpretation to (re-)design the well-known Church/Curry monotype system is comforting but may seem unconvincing because the type system was known in advance. A new refinement is now introduced allowing for recursive calls of f in a fixpoint definition $\mathbf{T}^{\mathrm{C}}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!]$ to be assigned different monotypes (which is the essence of Milner's polymorphic type schemes [29], although it is restricted to the $\mathtt{let}\,\mathtt{x} = e\ \mathtt{in}\, e'$ construct). This leads to a new polytype system which is simple enough to illustrate application of abstract interpretation to type theory.

Polytypes are defined as follows:

$$\begin{array}{lll}
m \in \mathbb{M}^{\mathrm{PC}}, \quad m ::= \mathtt{int} \mid m_1 \to m_2 & \text{monotype} & \\
p \in \mathbb{P}^{\mathrm{PC}} \triangleq \wp(\mathbb{M}^{\mathrm{PC}}) & \text{polytype} & \\
H \in \mathbb{H}^{\mathrm{PC}} \triangleq \mathbb{X} \mapsto \mathbb{P}^{\mathrm{PC}} & \text{type environment} & \\
\theta \in \mathbb{I}^{\mathrm{PC}} \triangleq \mathbb{H}^{\mathrm{PC}} \times \mathbb{M}^{\mathrm{PC}} & \text{typing} & \\
T \in \mathbb{T}^{\mathrm{PC}} \triangleq \wp(\mathbb{I}^{\mathrm{PC}}) & \text{program type} & (1)
\end{array}$$

The innovation with respect to Church/Curry monotype system is to associate *sets of monotypes understood conjunctively* to global program variables in type environments (instead of a single simple type).

The concretization function γ^{PC} is:

$$\begin{aligned}
\gamma_1^{\text{PC}} &\in \mathbb{M}^{\text{PC}} \mapsto \wp(\mathbb{U})\\
\gamma_1^{\text{PC}}(\texttt{int}) &\triangleq \{\uparrow(\mathsf{z}) :: \mathbb{Z}_\perp \mid \mathsf{z} \in \mathbb{Z}\} \cup \{\perp\}\\
\gamma_1^{\text{PC}}(m_1 \rightarrow m_2) &\triangleq \{\uparrow(\varphi) :: [\mathbb{U} \mapsto \mathbb{U}]_\perp \mid \varphi \in [\mathbb{U} \mapsto \mathbb{U}] \wedge\\
&\qquad \forall \mathsf{u} \in \gamma_1^{\text{PC}}(m_1) : \varphi(\mathsf{u}) \in \gamma_1^{\text{PC}}(m_2)\} \cup \{\perp\}\\
\gamma_2^{\text{PC}} &\in \mathbb{P}^{\text{PC}} \mapsto \wp(\mathbb{U})\\
\gamma_2^{\text{PC}}(p) &\triangleq \bigcap_{m \in p} \gamma_1^{\text{PC}}(m), \qquad \gamma_2^{\text{PC}}(\emptyset) \triangleq \mathbb{U}\\
\gamma_3^{\text{PC}} &\in \mathbb{H}^{\text{PC}} \mapsto \wp(\mathbb{R})\\
\gamma_3^{\text{PC}}(H) &\triangleq \{\mathsf{R} \in \mathbb{R} \mid \forall \mathsf{x} \in \mathbb{X} : \mathsf{R}(\mathsf{x}) \in \gamma_2^{\text{PC}}(H(\mathsf{x}))\}\\
\gamma_4^{\text{PC}} &\in \mathbb{I}^{\text{PC}} \mapsto \mathbb{P}\\
\gamma_4^{\text{PC}}(\langle H,\ m\rangle) &\triangleq \{\phi \in \mathbb{S} \mid \forall \mathsf{R} \in \gamma_3^{\text{PC}}(H) : \phi(\mathsf{R}) \in \gamma_1^{\text{PC}}(m)\}\\
\gamma^{\text{PC}} &\in \mathbb{T}^{\text{PC}} \mapsto \mathbb{P}\\
\gamma^{\text{PC}}(T) &\triangleq \bigcap\{\gamma_4^{\text{PC}}(\theta) \mid \theta \in T\}, \qquad \gamma^{\text{PC}}(\emptyset) \triangleq \mathbb{S}
\end{aligned}$$

Since γ^{PC} preserves greatest lower bounds:

$$\gamma^{\text{PC}}(\bigcup_{i\in\Delta} T_i) = \bigcap_{i\in\Delta} \gamma^{\text{PC}}(T_i)$$

The Galois connection:

$$\langle \mathbb{P},\ \subseteq\rangle \underset{\alpha^{\text{PC}}}{\overset{\gamma^{\text{PC}}}{\leftrightarrows}} \langle \mathbb{T}^{\text{PC}},\ \supseteq\rangle$$

is such that $\alpha^{\text{PC}}(P)$ is the best possible (i.e. most precise) type of programs with property P. The Church/Curry polytype semantics:

$$\mathbf{T}^{\text{PC}}[\![\bullet]\!] \in \mathbb{E} \mapsto \mathbb{T}^{\text{PC}}$$

is designed to be sound. For all programs $e \in \mathbb{E}$, $\alpha^{\text{PC}}(\mathbf{C}[\![e]\!]) \supseteq \mathbf{T}^{\text{PC}}[\![e]\!] \iff \mathbf{C}[\![e]\!] \subseteq \gamma^{\text{PC}}(\mathbf{T}^{\text{PC}}[\![e]\!]) \iff \mathbf{S}[\![e]\!] \in \gamma^{\text{PC}}(\mathbf{T}^{\text{PC}}[\![e]\!])$ so that typable programs e cannot go wrong since $\langle H,\ m\rangle \in \mathbf{T}^{\text{PC}}[\![e]\!] \wedge \mathsf{R} \in \gamma_3^{\text{PC}}(H) \wedge \mathbf{S}[\![e]\!]\mathsf{R} \neq \perp \Longrightarrow \mathbf{S}[\![e]\!]\mathsf{R} \neq \Omega$.

Again for simplicity of the presentation, the design of the Church/Curry polytype semantics $\mathbf{T}^{\text{PC}}[\![\bullet]\!]$ by abstract interpretation of the collecting semantics $\mathbf{C}[\![\bullet]\!]$ is postponed until **Sec. 16**. The `let` construct is defined such that:

$$\mathbf{S}[\![\texttt{let x = } e_1 \texttt{ in } e_2]\!] \triangleq \mathbf{S}[\![(\boldsymbol{\lambda}\mathtt{x} \cdot e_2)(e_1)]\!]$$

The Church/Curry polytype semantics is as follows (in the metalanguage, $\mathbf{gfp}_{\top}^{\sqsubseteq}\ \phi$ is the $\sqsubseteq$-greatest fixpoint of the monotone operator $\phi \in L \mapsto L$ on a cpo $\langle L,\ \sqsubseteq\rangle$ which is less than or equal to $\top$. The polytype $\mathbb{M}^{\text{PC}} \rightarrow \mathbb{M}^{\text{PC}}$ is defined to be a shorthand for $\{m_1 \rightarrow m_2 \mid m_1, m_2 \in \mathbb{M}^{\text{PC}}\}$):

$$\begin{aligned}
\mathbf{T}^{\text{PC}}[\![\mathtt{x}]\!] &\triangleq \{\langle H,\ m\rangle \mid m \in H(\mathtt{x})\}\\
\mathbf{T}^{\text{PC}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \{\langle H,\ m_1 \rightarrow m_2\rangle \mid\\
&\qquad \langle H[\mathtt{x} \leftarrow \{m_1\}],\ m_2\rangle \in \mathbf{T}^{\text{PC}}[\![e]\!]\}\\
\mathbf{T}^{\text{PC}}[\![e_1(e_2)]\!] &\triangleq \{\langle H,\ m_2\rangle \mid \langle H,\ m_1 \rightarrow m_2\rangle \in\\
&\qquad \mathbf{T}^{\text{PC}}[\![e_1]\!] \wedge \langle H,\ m_1\rangle \in \mathbf{T}^{\text{PC}}[\![e_2]\!]\}\\
\mathbf{T}^{\text{PC}}[\![\texttt{let x = } e_1 \texttt{ in } e_2]\!] &\triangleq \{\langle H,\ m_2\rangle \mid \exists p_1 \neq \emptyset : \forall m_1 \in p_1 :\\
&\qquad \langle H,\ m_1\rangle \in \mathbf{T}^{\text{PC}}[\![e_1]\!] \wedge\\
&\qquad \langle H[\mathtt{x} \leftarrow p_1],\ m_2\rangle \in \mathbf{T}^{\text{PC}}[\![e_2]\!]\}\\
\mathbf{T}^{\text{PC}}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \{\langle H,\ m\rangle \mid m \in \mathbf{gfp}^{\subseteq}_{\mathbb{M}^{\text{PC}} \rightarrow \mathbb{M}^{\text{PC}}}\ \Psi\}\\
&\text{where } \Psi \triangleq \Lambda p \cdot \{m' \mid \langle H[\mathtt{f} \leftarrow p],\ m'\rangle \in \mathbf{T}^{\text{PC}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\}\\
&\qquad = \{\langle H,\ m\rangle \mid \exists p \subseteq \mathbb{M}^{\text{PC}} \rightarrow \mathbb{M}^{\text{PC}} : m \in\\
&\qquad p \wedge \forall m' \in p : \langle H[\mathtt{f} \leftarrow p],\ m'\rangle \in\\
&\qquad \mathbf{T}^{\text{PC}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\}\\
\mathbf{T}^{\text{PC}}[\![\mathbf{1}]\!] &\triangleq \{\langle H,\ \texttt{int}\rangle \mid H \in \mathbb{H}^{\text{PC}}\}\\
\mathbf{T}^{\text{PC}}[\![e_1 - e_2]\!] &\triangleq \{\langle H,\ \texttt{int}\rangle \mid \langle H,\ \texttt{int}\rangle \in\\
&\qquad \mathbf{T}^{\text{PC}}[\![e_1]\!] \cap \mathbf{T}^{\text{PC}}[\![e_2]\!]\}\\
\mathbf{T}^{\text{PC}}[\![(e_1\ \texttt{?}\ e_2 : e_3)]\!] &\triangleq \{\langle H,\ m\rangle \mid \langle H,\ \texttt{int}\rangle \in \mathbf{T}^{\text{PC}}[\![e_1]\!] \wedge\\
&\qquad \langle H,\ m\rangle \in \mathbf{T}^{\text{PC}}[\![e_2]\!] \cap \mathbf{T}^{\text{PC}}[\![e_3]\!]\}
\end{aligned}$$

Notice in the definition of $\mathbf{T}^{\text{PC}}[\![\texttt{let x = } e_1 \texttt{ in } e_2]\!]$ that $p_1 \neq \emptyset$. Otherwise $\texttt{let x = } e_1 \texttt{ in } \mathbf{1}$ would be always be typable with $p_1 = \emptyset$ because `x` does not appear in $\mathbf{1}$. This is not sound since if $\mathbf{S}[\![e_1]\!]\mathsf{R} = \Omega$ then by definition of the eager denotational semantics $\mathbf{S}[\![\texttt{let x = } e_1 \texttt{ in } \mathbf{1}]\!]\mathsf{R} = \mathbf{S}[\![(\boldsymbol{\lambda}\mathtt{x} \cdot \mathbf{1})(e_1)]\!]\mathsf{R} = \Omega$. However this would be sound with a lazy semantics where e_1 is not evaluated.

By defining judgments as:

$$H \overset{\text{PC}}{\vdash} e \Rightarrow m \triangleq \langle H,\ m\rangle \in \mathbf{T}^{\text{PC}}[\![e]\!]$$

the Church/Curry polytype semantics can be presented in the equivalent rule based form:

$$\frac{m \in H(\mathtt{x})}{H \overset{\text{PC}}{\vdash} \mathtt{x} \Rightarrow m} \qquad \frac{H[\mathtt{x} \leftarrow \{m_1\}] \overset{\text{PC}}{\vdash} e \Rightarrow m_2}{H \overset{\text{PC}}{\vdash} \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow m_1 \rightarrow m_2}$$

$$\frac{H \overset{\text{PC}}{\vdash} e_1 \Rightarrow m_1 \rightarrow m_2,\ H \overset{\text{PC}}{\vdash} e_2 \Rightarrow m_1}{H \overset{\text{PC}}{\vdash} e_1(e_2) \Rightarrow m_2}$$

$$\frac{p_1 \neq \emptyset,\ \forall m_1 \in p_1 : H \overset{\text{PC}}{\vdash} e_1 \Rightarrow m_1,\ H[\mathtt{x} \leftarrow p_1] \overset{\text{PC}}{\vdash} e_2 \Rightarrow m_2}{H \overset{\text{PC}}{\vdash} \texttt{let x = } e_1 \texttt{ in } e_2 \Rightarrow m_2}$$

$$\frac{\forall m_1 \in p_1 : H[\mathtt{f} \leftarrow p_1] \overset{\text{PC}}{\vdash} \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow m_1,\ m \in p_1}{H \overset{\text{PC}}{\vdash} \boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow m} \tag{2}$$

$$H \overset{\text{PC}}{\vdash} \mathbf{1} \Rightarrow \texttt{int} \qquad \frac{H \overset{\text{PC}}{\vdash} e_1 \Rightarrow \texttt{int},\ H \overset{\text{PC}}{\vdash} e_2 \Rightarrow \texttt{int}}{H \overset{\text{PC}}{\vdash} e_1 - e_2 \Rightarrow \texttt{int}}$$

$$\frac{H \overset{\text{PC}}{\vdash} e_1 \Rightarrow \texttt{int},\ H \overset{\text{PC}}{\vdash} e_2 \Rightarrow m,\ H \overset{\text{PC}}{\vdash} e_3 \Rightarrow m}{H \overset{\text{PC}}{\vdash} (e_1\ \texttt{?}\ e_2 : e_3) \Rightarrow m}$$

Example 1 The following ML program (such that `F` f g n $x = g(f^n(x))$) is not typable by Damas-Milner's polymorphic type system [17, 29]. To show this, it is sufficient to submit it to an ML compiler[2]:

```
>       Caml Light version 0.71/mac

#let rec F f g n x =
   if n = 0 then g(x)
   else F(f)(fun x -> (fun h -> g(h(x))))(n-1)(x)(f);;

Toplevel input:
>.......F f g n x =
>   if n = 0 then g(x)
>   else F(f)(fun x -> (fun h -> g(h(x))))(n-1)(x)(f)..
This expression has type
 'a -> ('b -> 'c) -> int -> 'b -> 'c,
but is used with type
 'a -> ('b -> ('b -> 'b) -> 'c) -> int -> 'b -> 'a -> 'c.
```

[2]In practice, ML programmers would rewrite `F` as `let rec F f g n x = if n = 0 then g(x) else F f (fun x -> g(f x)) (n-1) x;;` with type `('a -> 'a) -> ('a -> 'b) -> int -> 'a -> 'b`.

Using the above à la Church/Curry polytype system, the type of this program is formally defined as:

$$\{\langle H,\ m\rangle \mid m \in \mathbf{gfp}^{\subseteq}_{\mathbb{M}^{\mathrm{PC}} \to \mathbb{M}^{\mathrm{PC}}}\ \Psi\}$$

where:

$$\Psi(p) \ = \ \{m_1 \to ((m_2 \to m_3) \to (\mathtt{int} \to (m_2 \to m_3))) \mid$$
$$m_1 \to ((m_4 \to ((m_4 \to m_2) \to m_3)) \to (\mathtt{int} \to (m_2 \to (m_1 \to m_3)))) \in p\}$$

that is, by hand-computation of the greatest fixpoint:

$$\{\langle H,\ (m_1 \to m_1) \to ((m_1 \to m_2) \to (\mathtt{int} \to (m_1 \to m_2)))\rangle \mid H \in \mathbb{H}^{\mathrm{PC}} \wedge m_1, m_2 \in \mathbb{M}^{\mathrm{PC}}\} \qquad \square$$

Prop. 26 will show that Damas-Milner-Mycroft polymorphic type semantics $\mathbf{T}^{\mathrm{DM}^2}[\![\bullet]\!]$ is an abstraction of $\mathbf{T}^{\mathrm{PC}}[\![\bullet]\!]$. This example is given to show that this abstraction is strict[3].

The next refinement is to consider polytypes p to the left of the arrow $p \to m$ to get a simple form of infinite conjunctive types [3, 5, 23] with rules:

$$\frac{H[\mathtt{x} \leftarrow p_1] \vdash^{\wedge} e \Rightarrow m_2}{H \vdash^{\wedge} \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow p_1 \to m_2}$$

$$\frac{H \vdash^{\wedge} e_1 \Rightarrow p_1 \to m_2,\ p_1 \neq \emptyset,\ \forall m_1 \in p_1 : H \vdash^{\wedge} e_2 \Rightarrow m_1}{H \vdash^{\wedge} e_1(e_2) \Rightarrow m_2}$$

so that the $\mathtt{let}\,\mathtt{x} = e_1\ \mathtt{in}\,e_2$ rule no longer appears as a special case. For lack of space, this type system (otherwise similar to $\vdash^{\mathrm{PC}}$) will not be developed, and consider instead a further refinement in the form of the type collecting semantics $\mathbf{T}^{\mathrm{C}\circ}[\![\bullet]\!]$.

7 Polytype to Monotype Abstraction

In order to illustrate the constructive design of an abstract semantics by abstract interpretation of a concrete semantics on a very simple example, $\mathbf{T}^{\mathrm{C}}[\![e]\!]$ is now derived from $\mathbf{T}^{\mathrm{PC}}[\![e]\!]$. The correspondence is given by the Galois insertion[4]:

$$\langle \mathbb{T}^{\mathrm{PC}},\ \supseteq\rangle \ \underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\ \langle \mathbb{T}^{\mathrm{C}},\ \supseteq\rangle$$

such that:

$$\alpha(T) \triangleq \{\langle H,\ m\rangle \mid \langle \Lambda \mathtt{y} \in \mathbb{X} \cdot \{H(\mathtt{y})\},\ m\rangle \in T\}$$
$$\gamma(T') \triangleq \{\langle \Lambda \mathtt{y} \in \mathbb{X} \cdot \{H(\mathtt{y})\},\ m\rangle \mid \langle H,\ m\rangle \in T'\}$$

The design of $\mathbf{T}^{\mathrm{C}}[\![e]\!]$ is by algebraic simplification of the expression $\alpha(\mathbf{T}^{\mathrm{PC}}[\![e]\!])$ using an $\supseteq$-upper approximation when necessary. One proceeds by induction on the syntax of e. For $e = \mathtt{x}$, one has:

$$\begin{aligned}
&\alpha(\mathbf{T}^{\mathrm{PC}}[\![\mathtt{x}]\!]) \\
&= \{\langle H,\ m\rangle \mid \langle \Lambda \mathtt{y} \in \mathbb{X} \cdot \{H(\mathtt{y})\},\ m\rangle \in \mathbf{T}^{\mathrm{PC}}[\![\mathtt{x}]\!]\} && \text{by def. } \alpha \\
&= \{\langle H,\ m\rangle \mid m \in \{H(\mathtt{x})\}\} && \text{by def. } \mathbf{T}^{\mathrm{PC}}[\![\mathtt{x}]\!] \\
&= \{\langle H,\ H(\mathtt{x})\rangle \mid H \in \mathbb{I}^{\mathrm{C}}\} && \text{by def. } \in \\
&\triangleq \mathbf{T}^{\mathrm{C}}[\![\mathtt{x}]\!]
\end{aligned}$$

[3] Formally $\exists e \in \mathbb{E} : \mathbf{T}^{\mathrm{PC}}[\![e]\!] \supset \ddot{\gamma}^{\mathrm{MM}}_{\mathrm{p}} \circ \gamma^{\mathrm{DM}}_{\mathrm{p}}(\mathbf{T}^{\mathrm{DM}^2}[\![e]\!])$.

[4] A *Galois insertion* $\langle L,\ \leq\rangle \ \underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\ \langle M,\ \preceq\rangle$ is a Galois connection $\langle L,\ \leq\rangle \ \underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\ \langle M,\ \preceq\rangle$ such that α is onto (which is equivalent to γ is one-to-one and is equivalent to $\alpha \circ \gamma = \mathbf{Id}$ where $\mathbf{Id}$ is the identity on M).

For $e = \mathtt{let}\,\mathtt{x} = e_1\ \mathtt{in}\,e_2$, one has ($\mathtt{y} \in \mathbb{X}$ is a program variable):

$$\begin{aligned}
&\alpha(\mathbf{T}^{\mathrm{PC}}[\![\mathtt{let}\,\mathtt{x} = e_1\ \mathtt{in}\,e_2]\!]) \\
&= \{\langle H,\ m\rangle \mid \langle \Lambda \mathtt{y} \cdot \{H(\mathtt{y})\},\ m\rangle \in \mathbf{T}^{\mathrm{PC}}[\![\mathtt{let}\,\mathtt{x} = e_1\ \mathtt{in}\,e_2]\!]\} \\
& && \text{by definition of } \alpha \\
&= \{\langle H,\ m_2\rangle \mid \exists p_1 \neq \emptyset : \forall m_1 \in p_1 : \langle \Lambda \mathtt{y} \cdot \{H(\mathtt{y})\},\ m_1\rangle \in \\
&\qquad \mathbf{T}^{\mathrm{PC}}[\![e_1]\!] \wedge \langle \Lambda \mathtt{y} \cdot \{H(\mathtt{y})\}[\mathtt{x} \leftarrow p_1],\ m_2\rangle \in \mathbf{T}^{\mathrm{PC}}[\![e_2]\!]\} \\
& && \text{by definition of } \mathbf{T}^{\mathrm{PC}}[\![\mathtt{let}\,\mathtt{x} = e_1\ \mathtt{in}\,e_2]\!] \\
&\supseteq \{\langle H,\ m_2\rangle \mid \langle \Lambda \mathtt{y} \cdot \{H(\mathtt{y})\},\ m_1\rangle \in \mathbf{T}^{\mathrm{PC}}[\![e_1]\!] \wedge \\
&\qquad \langle \Lambda \mathtt{y} \cdot \{H(\mathtt{y})\}[\mathtt{x} \leftarrow \{m_1\}],\ m_2\rangle \in \mathbf{T}^{\mathrm{PC}}[\![e_2]\!]\} \\
& && \text{by restricting } p_1 \text{ to } \{m_1\}
\end{aligned}$$

The restriction of the polytype p_1 to the monotype $\{m_1\}$ is a loss of information. The best solution to assign a monotype to $\mathtt{let}\,\mathtt{x} = e_1\ \mathtt{in}\,e_2$ would be to locally use a polytype for $\mathtt{x}$ in e_2. However this approximation allows for the same monotype system to be used in all subexpressions.

$$\begin{aligned}
&= \{\langle H,\ m_2\rangle \mid \langle \Lambda \mathtt{y} \cdot \{H(\mathtt{y})\},\ m_1\rangle \in \mathbf{T}^{\mathrm{PC}}[\![e_1]\!] \wedge \\
&\qquad \langle \Lambda \mathtt{y} \cdot \{H[\mathtt{x} \leftarrow m_1](\mathtt{y})\},\ m_2\rangle \in \mathbf{T}^{\mathrm{PC}}[\![e_2]\!]\} \\
& && \text{by definition of assignment } H[\mathtt{x} \leftarrow t] \\
&= \{\langle H,\ m_2\rangle \mid \langle H,\ m_1\rangle \in \alpha(\mathbf{T}^{\mathrm{PC}}[\![e_1]\!]) \wedge \langle H[\mathtt{x} \leftarrow m_1],\ m_2\rangle \in \\
&\qquad \alpha(\mathbf{T}^{\mathrm{PC}}[\![e_2]\!])\} && \text{by definition of } \alpha \\
&\supseteq \{\langle H, m_2\rangle \mid \langle H, m_1\rangle \in \mathbf{T}^{\mathrm{C}}[\![e_1]\!] \wedge \langle H[\mathtt{x} \leftarrow m_1], m_2\rangle \in \mathbf{T}^{\mathrm{C}}[\![e_2]\!]\} \\
& && \text{by induction hypothesis} \\
&\triangleq \mathbf{T}^{\mathrm{C}}[\![\mathtt{let}\,\mathtt{x} = e_1\ \mathtt{in}\,e_2]\!]
\end{aligned}$$

The other cases are equally simple.

8 Abstract Semantics and Interpreters

An *abstract semantics*:

$$\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathbf{T}^\sharp[\![\bullet]\!]\rangle$$

is given by a poset of abstract properties/types $\langle \mathbb{T}^\sharp,\ \leq^\sharp\rangle$, where $\leq^\sharp$ is an abstract version of logical implication, and an abstract semantic function:

$$\mathbf{T}^\sharp[\![\bullet]\!] \in \mathbb{E} \mapsto \mathbb{T}^\sharp.$$

A *compositional abstract semantics*

$$\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathcal{P}^\sharp\rangle$$

is parameterized by *semantic primitives*

$$\mathcal{P}^\sharp \ = \ \{\Psi^\sharp_e \mid e \in \mathbb{E}\}$$

where each primitive $\Psi^\sharp_e$ for program expression

$$e[\mathtt{x}_1, \ldots, \mathtt{x}_m](e_1, \ldots, e_n)$$

with locally bound variables $\mathtt{x}_1, \ldots, \mathtt{x}_m$ (as $\mathtt{x}$ in $\boldsymbol{\lambda}\mathtt{x} \cdot e'$) and subexpressions $e_1, \ldots, e_n$ (as e' in $\boldsymbol{\lambda}\mathtt{x} \cdot e'$) has signature $\mathbb{X}^m \mapsto (\mathbb{T}^{\sharp n} \mapsto \mathbb{T}^\sharp)$. The corresponding semantics $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathbf{T}^\sharp[\![\bullet]\!]\rangle$ is defined compositionally ($\Psi^\sharp_e(\mathsf{u}_1, \ldots, \mathsf{u}_n)$ is a shorthand for $\Psi^\sharp_e[\mathtt{x}_1, \ldots, \mathtt{x}_m](\mathsf{u}_1, \ldots, \mathsf{u}_n)$):

$$\mathbf{T}^\sharp[\![e]\!] \triangleq \Psi^\sharp_e(\mathbf{T}^\sharp[\![e_1]\!], \ldots, \mathbf{T}^\sharp[\![e_n]\!])$$

by induction on the syntax of program expressions. A compositional abstract semantics $\langle \mathbb{T}^\sharp, \leq^\sharp, \mathcal{P}^\sharp\rangle$ is *monotone* when all primitives are monotone, that is:

$$\langle T_1,\ \ldots,\ T_n\rangle \ \leq^\sharp\ \langle T'_1,\ \ldots,\ T'_n\rangle$$
$$\Longrightarrow\ \Psi^\sharp_e(T_1, \ldots, T_n) \ \leq^\sharp\ \Psi^\sharp_e(T'_1, \ldots, T'_n)\ .$$

When $\langle \mathbb{T}^\sharp,\ \leq^\sharp\rangle$ is computer-representable and the primitives $\{\Psi^\sharp_e \mid e \in \mathbb{E}\}$ are computable, this is the specification of a generic *abstract interpreter* defined by induction on the structure of programs and parameterized by modules for implementing the primitives with above signature.

9 Abstraction and Soundness

The abstract semantics $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathbf{T}^\sharp[\![\bullet]\!]\rangle$ is said to be the *abstraction* of the concrete semantics $\langle \mathbb{T}^\flat,\ \leq^\flat,\ \mathbf{T}^\flat[\![\bullet]\!]\rangle$ whenever there is a concretization map $\gamma \in \mathbb{T}^\sharp \mapsto \mathbb{T}^\flat$ which is monotone and such that:

$$\forall e \in \mathbb{E} : \mathbf{T}^\flat[\![e]\!] \leq^\flat \gamma(\mathbf{T}^\sharp[\![e]\!]).$$

If the concrete and abstract semantics are defined compositionally then, by induction on the structure of e, the following *general compositional abstraction condition*:

$$\begin{aligned} &\forall i = 1, \ldots, n : \mathbf{T}^\flat[\![e_i]\!] \ \leq^\flat\ \gamma(\mathbf{T}^\sharp[\![e_i]\!]) \qquad (3)\\ \Longrightarrow\ &\Psi^\flat_e(\mathbf{T}^\flat[\![e_1]\!], \ldots, \mathbf{T}^\flat[\![e_n]\!]) \ \leq^\flat\ \gamma(\Psi^\sharp_e(\mathbf{T}^\sharp[\![e_1]\!], \ldots, \mathbf{T}^\sharp[\![e_n]\!]))\end{aligned}$$

is sufficient. $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathcal{P}^\sharp\rangle$ is said to be an *upper approximation* of $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathcal{P}^\natural\rangle$ when it is an abstraction through the identity map, in which case the general compositional abstraction condition degenerates to the following *upper approximation condition*:

$$\begin{aligned} &\forall i = 1, \ldots, n : \mathbf{T}^\natural[\![e_i]\!] \ \leq^\sharp\ \mathbf{T}^\sharp[\![e_i]\!]\\ \Longrightarrow\ &\Psi^\natural_e(\mathbf{T}^\natural[\![e_1]\!], \ldots, \mathbf{T}^\natural[\![e_n]\!]) \ \leq^\sharp\ \Psi^\sharp_e(\mathbf{T}^\sharp[\![e_1]\!], \ldots, \mathbf{T}^\sharp[\![e_n]\!]).\end{aligned}$$

A *sound* abstract semantics is an abstraction of the collecting semantics $\langle \mathbb{P},\ \subseteq,\ \mathbf{C}[\![\bullet]\!]\rangle$. Observe that an upper approximation of a sound abstraction is a sound abstraction.

For monotone compositional abstract semantics $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathcal{P}^\sharp\rangle$, the following *monotone compositional concretization condition*:

$$\Psi^\flat_e(\gamma(\mathbf{T}^\sharp[\![e_1]\!]), \ldots, \gamma(\mathbf{T}^\sharp[\![e_n]\!])) \leq^\flat \gamma(\Psi^\sharp_e(\mathbf{T}^\sharp[\![e_1]\!], \ldots, \mathbf{T}^\sharp[\![e_n]\!]))$$

implies the general compositional abstraction condition (3) hence that the corresponding abstract semantics $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathbf{T}^\sharp[\![\bullet]\!]\rangle$ is the abstraction of $\langle \mathbb{T}^\flat,\ \leq^\flat,\ \mathbf{T}^\flat[\![\bullet]\!]\rangle$.

Observe that the abstraction of a sound abstraction is a sound abstraction so that abstract semantics and interpreters can be designed by successive abstractions. This remains true when the correspondence is a Galois connection since they compose:

$$\begin{aligned} &\langle \mathbb{T}^\flat,\ \leq^\flat\rangle \ \underset{\alpha^\natural}{\overset{\gamma^\natural}{\leftrightarrows}}\ \langle \mathbb{T}^\natural,\ \leq^\natural\rangle \wedge \langle \mathbb{T}^\natural,\ \leq^\natural\rangle \ \underset{\alpha^\sharp}{\overset{\gamma^\sharp}{\leftrightarrows}}\ \langle \mathbb{T}^\sharp,\ \leq^\sharp\rangle\\ \Longrightarrow\ &\langle \mathbb{T}^\flat,\ \leq^\flat\rangle \ \underset{\alpha^\sharp \circ \alpha^\natural}{\overset{\gamma^\natural \circ \gamma^\sharp}{\leftrightarrows}}\ \langle \mathbb{T}^\sharp,\ \leq^\sharp\rangle\ .\end{aligned}$$

When the correspondence between concrete and abstract compositional semantics is through a Galois connection:

$$\langle \mathbb{T}^\flat,\ \leq^\flat\rangle \ \underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\ \langle \mathbb{T}^\sharp,\ \leq^\sharp\rangle \qquad (4)$$

The *best abstraction* of $\langle \mathbb{T}^\flat,\ \leq^\flat,\ \mathcal{P}^\flat\rangle$ is defined as $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathcal{P}^\natural\rangle$ with primitives:

$$\Psi^\natural_e(T_1, \ldots, T_n) \triangleq \alpha(\Psi^\flat_e(\gamma(T_1), \ldots, \gamma(T_n)))\ .$$

This terminology is justified by the fact that if $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathcal{P}^\sharp\rangle$ is an abstraction of the monotone concrete semantics $\langle \mathbb{T}^\flat,\ \leq^\flat,\ \mathcal{P}^\flat\rangle$ then the corresponding semantics is an upper approximation of the best abstract semantics, since:

$$\forall e \in \mathbb{E} : \mathbf{T}^\natural[\![e]\!] \ \leq^\sharp\ \mathbf{T}^\sharp[\![e]\!]\ .$$

Reciprocally, observe that any upper approximation of the best abstraction of a monotone compositional concrete semantics is an abstraction of that concrete semantics.

It follows that the exhibition of a Galois connection is a precious guide for designing a type system, since one can use the best abstraction:

$$\Lambda\langle T_1,\ \ldots,\ T_n\rangle \centerdot \alpha(\Psi^\flat_e(\gamma(T_1), \ldots, \gamma(T_n)))$$

as a starting point and derive an upper approximation $\Psi^\sharp_e$ by elimination of α and γ in the previous formula. However, when (4) holds and $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathcal{P}^\sharp\rangle$ is monotone, the following *monotone compositional abstraction condition*:

$$\alpha(\Psi^\flat_e(\mathbf{T}^\flat[\![e_1]\!], \ldots, \mathbf{T}^\flat[\![e_n]\!])) \leq^\sharp \Psi^\sharp_e(\alpha(\mathbf{T}^\flat[\![e_1]\!]), \ldots, \alpha(\mathbf{T}^\flat[\![e_n]\!]))$$

implies that the corresponding abstract semantics $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathbf{T}^\sharp[\![\bullet]\!]\rangle$ is the abstraction of $\langle \mathbb{T}^\flat,\ \leq^\flat,\ \mathbf{T}^\flat[\![\bullet]\!]\rangle$. As already illustrated by the design of $\mathbf{T}^{\mathrm{C}}[\![e]\!]$ from $\mathbf{T}^{\mathrm{PC}}[\![e]\!]$, the methodology consists of starting from $\alpha(\Psi^\flat_e(\mathbf{T}^\flat[\![e_1]\!],\ \ldots,\ \mathbf{T}^\flat[\![e_n]\!]))$ so as to rewrite it in the form $\Psi^\sharp_e(\alpha(\mathbf{T}^\flat[\![e_1]\!]),\ \ldots,\ \alpha(\mathbf{T}^\flat[\![e_n]\!]))$ hence providing the definition of $\Psi^\sharp_e$. It remains to check that $\Psi^\sharp_e$ is monotone.

10 Typable Programs Cannot Go Wrong

A (prescriptive) *sound type system* $\mathcal{T}^\sharp$ is a tuple $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathbf{T}^\sharp[\![\bullet]\!],\ \gamma^\sharp,\ \mathcal{E}^\sharp\rangle$ where $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathbf{T}^\sharp[\![\bullet]\!]\rangle$ is a sound abstract semantics for the monotone concretization function

$$\gamma^\sharp \ \in\ \mathbb{T}^\sharp \mapsto \mathbb{P}$$

(recall that $\mathbb{P} \triangleq \wp(\mathbb{S})$) and the map

$$\mathcal{E}^\sharp \ \in\ \mathbb{T}^\sharp \mapsto \wp(\mathbb{R})$$

such that

$$\forall T \in \mathbb{T}^\sharp : \forall \mathsf{R} \in \mathcal{E}^\sharp(T) : \forall \phi \in \gamma^\sharp(T) : \Omega \notin \phi(\mathsf{R})$$

defines *admissible environments*. For example, $\langle \mathbb{T}^{\mathrm{PC}},\ \leq^{\mathrm{PC}},\ \mathbf{T}^{\mathrm{PC}}[\![\bullet]\!],\ \gamma^{\mathrm{PC}},\ \mathcal{E}^{\mathrm{PC}}\rangle$ is a sound type system where:

$$\mathcal{E}^{\mathrm{PC}}(T) \ \triangleq\ \{\mathsf{R} \mid \exists\langle H,\ m\rangle \in T \wedge \mathsf{R} \in \gamma_3^{\mathrm{PC}}(H)\}$$

A program e is *typable in the environment* $\mathsf{R} \in \mathbb{R}$ for the sound type system $\mathcal{T}^\sharp$, if and only if:

$$\mathsf{R} \in \mathcal{E}^\sharp(\mathbf{T}^\sharp[\![e]\!])$$

Proposition 2 *A typable program e in the environment $\mathsf{R} \in \mathbb{R}$ for any sound type system $\mathcal{T}^\sharp$ cannot go wrong in that $\mathbf{S}[\![e]\!]\mathsf{R} \neq \Omega$.*

It is interesting to note that instead of *"formalizing the type system by a set of type rules, and verifying that program execution of well-typed programs cannot produce type errors."* [2], the abstract interpretation design methodology ensures that type systems will be sound by construction, this soundness requirement being used as a guideline for designing the type system.

11 Herbrand Abstraction

The Herbrand abstraction (with associative and commutative variants) is the most extensively and almost exclusively used abstraction for type inference. It is also of common use in abstract interpretation of logic programs [18].

Let T be a denumerable set of terms. Let $V \subseteq T$ be a denumerable infinite subset of variables. Variables in V are called *generic* since they are instanciable whereas variables not in V, called *parametric* are not instanciable (hence can be considered as constants). $W \hookrightarrow T$ stands for the set of idempotent substitutions σ assigning a term $\sigma(v) \in T$ to variables $v \in W$. Let $\mathbf{ground}_V(T) \subseteq T$ be the subset of ground terms, with no variable in V. By [27], one has the Galois insertions ($\alpha \in V$ is a generic variable):

$$\langle \wp(\mathbf{ground}_V(T)),\ \subseteq,\ \emptyset,\ \mathbf{ground}_V(T),\ \cup,\ \cap\rangle \underset{\mathbf{lcg}_V}{\overset{\mathbf{ground}_V}{\leftrightarrows}} \langle T^{\emptyset}/_{\equiv_V},\ \leq_V,\ \emptyset,\ [\alpha]_{\equiv_V},\ \mathbf{lcg}_V,\ \mathbf{gci}_V\rangle$$

$$\langle \wp(T),\ \subseteq,\ \emptyset,\ T,\ \cup,\ \cap\rangle \underset{\mathbf{lcg}_V}{\overset{\mathbf{inst}_V}{\leftrightarrows}} \langle T^{\emptyset}/_{\equiv_V},\ \leq_V,\ \emptyset,\ [\alpha]_{\equiv_V},\ \mathbf{lcg}_V,\ \mathbf{gci}_V\rangle$$

where $\mathbf{inst}_V(t)$ is the set of instances in T of term $t \in T$ for generic variables in V while $\mathbf{ground}_V(t)$ is the set of ground ones with no variable in V. $\mathbf{lcg}_V$ is the least common generalization (also called most specific generalization or least common anti-instance) for generic variables in V. $\leq_V$ is the instantiation preorder ($t_1 \leq_V t_2 \iff \mathbf{inst}_V(t_1) \subseteq \mathbf{inst}_V(t_2)$). $\equiv_V$ is the corresponding equivalence relation. $T^{\emptyset}/_{\equiv_V}$ is the set of equivalence classes $[t]_{\equiv_V}$ augmented with a new infimum $\emptyset$. $\mathbf{gci}_V(\theta)$ is the greatest common instance of a set $\theta \subseteq T$ of terms that is $\mathbf{mgu}_V(T')t$ where $t \in T'$, $\mathbf{mgu}_V(T')$ being the most general unifier of T' and T' being a renaming of T such that no two terms in T' share a common generic variable in V. If $\emptyset \in T$ or no such unifier exists (in which case $\mathbf{mgu}_V(T') = \emptyset$) then $\mathbf{gci}_V(\theta) = \emptyset$. The complete lattice $\langle T^{\emptyset}/_{\equiv_V},\ \leq_V,\ \emptyset,\ [\alpha]_{\equiv_V},\ \mathbf{lcg}_V,\ \mathbf{gci}_V\rangle$ satisfies the ascending chain condition but has infinite strictly decreasing chains with limit $\emptyset$. When all variables are generic, or V is understood from the context, index V is omitted.

12 Exactness and Principality

A principal typing is a typing that subsumes all others [17, 31]. More generally, this means that there is an exact representation for all possible typings. This can be understood as a relation between abstract semantics, as follows:

Definition 3 *If* $\mathbf{T}^\natural\llbracket\bullet\rrbracket \in \mathbb{E} \mapsto \mathbb{T}^\natural$ *and* $\langle \mathbb{T}^\natural,\ \leq^\natural\rangle \underset{\alpha^\sharp}{\overset{\gamma^\sharp}{\leftrightarrows}} \langle \mathbb{T}^\sharp,\ \leq^\sharp\rangle$ *then the abstract semantics* $\mathbf{T}^\sharp\llbracket\bullet\rrbracket \in \mathbb{E} \mapsto \mathbb{T}^\sharp$ *such that* $\forall e \in \mathbb{E} : \alpha^\sharp(\mathbf{T}^\natural\llbracket e\rrbracket) \leq^\sharp \mathbf{T}^\sharp\llbracket e\rrbracket$ *is* exact *if and only if* $\forall e \in \mathbb{E} : \gamma^\sharp(\mathbf{T}^\sharp\llbracket e\rrbracket) = \mathbf{T}^\natural\llbracket e\rrbracket$. *It is* principal *if and only if it is exact and moreover* $\mathbb{T}^\sharp = \mathbb{T}^\natural$.

This means that the abstraction loses no information for the semantics $\mathbf{T}^\natural\llbracket\bullet\rrbracket$ and $\mathbf{T}^\sharp\llbracket\bullet\rrbracket$ which are isomorphic although other elements x of $\mathbb{T}^\natural$ not in $\{\gamma^\sharp(\mathbf{T}^\sharp\llbracket e\rrbracket) \mid e \in \mathbb{E}\}$ may have a strict approximation $x <^\natural \gamma^\sharp(\alpha^\sharp(x))$.

Proposition 4 *If* $\mathbf{T}\llbracket\bullet\rrbracket \in \mathbb{E} \mapsto \mathbb{T}^\natural$ *and* $\langle \mathbb{T}^\sharp,\ \leq^\sharp\rangle \underset{\alpha^\sharp}{\overset{\gamma^\sharp}{\leftrightarrows}} \langle \mathbb{T}^\sharp,\ \leq^\natural\rangle$ *then the only possible exact semantics is* $\mathbf{T}^\sharp\llbracket e\rrbracket \triangleq \alpha^\sharp(\mathbf{T}\llbracket e\rrbracket)$ *which is the case if and only if* $\gamma^\sharp(\mathbf{T}^\sharp\llbracket e\rrbracket) = \mathbf{T}\llbracket e\rrbracket$.

A remarkable property of exactness is that it can be used both as an upper and a lower approximation:

Proposition 5 *If* $\langle \mathbb{T}^\natural,\ \leq^\natural,\ \mathbf{T}^\natural\llbracket\bullet\rrbracket\rangle$, *resp.* $\langle \mathbb{T}^\natural,\ \geq^\natural,\ \mathbf{T}^\natural\llbracket\bullet\rrbracket\rangle$, *with* exact *abstraction* $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathbf{T}^\sharp\llbracket\bullet\rrbracket\rangle$ *is an abstraction of the concrete semantics* $\langle \mathbb{T}^\flat,\ \leq^\flat,\ \mathbf{T}^\flat\llbracket\bullet\rrbracket\rangle$ *then* $\langle \mathbb{T}^\sharp,\ \leq^\sharp,\ \mathbf{T}^\sharp\llbracket\bullet\rrbracket\rangle$ *(resp.* $\langle \mathbb{T}^\sharp,\ \geq^\sharp,\ \mathbf{T}^\sharp\llbracket\bullet\rrbracket\rangle$*) is also an abstraction of* $\langle \mathbb{T}^\flat,\ \leq^\flat,\ \mathbf{T}^\flat\llbracket\bullet\rrbracket\rangle$.

Proof Assume that $\langle \mathbb{T}^\natural,\ \geq^\natural,\ \mathbf{T}^\natural\llbracket\bullet\rrbracket\rangle$ is the abstraction of $\langle \mathbb{T}^\flat,\ \leq^\flat,\ \mathbf{T}^\flat\llbracket\bullet\rrbracket\rangle$ by $\gamma^\natural$, $\langle \mathbb{T}^\natural,\ \leq^\natural\rangle \underset{\alpha^\sharp}{\overset{\gamma^\sharp}{\leftrightarrows}} \langle \mathbb{T}^\sharp,\ \leq^\sharp\rangle$ and $\mathbf{T}^\sharp\llbracket e\rrbracket \triangleq \alpha^\sharp(\mathbf{T}^\natural\llbracket e\rrbracket)$[5]. By exactness $\gamma^\sharp(\mathbf{T}^\sharp\llbracket e\rrbracket) = \mathbf{T}^\natural\llbracket e\rrbracket$ so $\mathbf{T}^\flat\llbracket e\rrbracket \leq^\flat \gamma^\natural(\mathbf{T}^\natural\llbracket e\rrbracket) = \gamma^\natural \circ \gamma^\sharp(\mathbf{T}^\sharp\llbracket e\rrbracket)$ as required. Any $\geq^\sharp$-approximation $\mathbf{T}^{\times}\llbracket e\rrbracket$ of $\mathbf{T}^\sharp\llbracket e\rrbracket$ is also an abstraction since $\gamma^\sharp(\mathbf{T}^\sharp\llbracket e\rrbracket) \geq^\natural \gamma^\sharp(\mathbf{T}^{\times}\llbracket e\rrbracket)$ whence $\mathbf{T}^\flat\llbracket e\rrbracket \leq^\flat \gamma^\natural \circ \gamma^\sharp(\mathbf{T}^\sharp\llbracket e\rrbracket) \leq^\flat \gamma^\natural \circ \gamma^\sharp(\mathbf{T}^\sharp\llbracket e\rrbracket)$ by monotony. $\square$

13 Hindley Monotype Semantics

Hindley type abstract interpreter [21] is a computer-implementable exact abstraction of Church/Curry monotype abstract semantics $\langle \mathbb{T}^{\mathrm{C}},\ \supseteq,\ \mathbf{T}^{\mathrm{C}}\llbracket\bullet\rrbracket\rangle$. Hindley types are as follows:

$\text{'a} \in \mathbb{V}$	type variables
$\tau \in \mathbb{M}^{\mathrm{H}}_{\mathrm{v}}$	monotype with variables
$\tau ::= \mathtt{int} \mid \text{'a} \mid \tau_1 \mathbin{\texttt{->}} \tau_2$	
$H \in \mathbb{H}^{\mathrm{H}} \triangleq \mathbb{X} \mapsto \mathbb{M}^{\mathrm{H}}_{\mathrm{v}}$	type environment
$T \in \mathbb{T}^{\mathrm{H}} \triangleq \mathbb{H}^{\mathrm{H}} \times \mathbb{M}^{\mathrm{H}}_{\mathrm{v}}$	program typing

The Hindley type semantics:

$$\mathbf{T}^{\mathrm{H}}\llbracket\bullet\rrbracket \in \mathbb{E} \mapsto \mathbb{T}^{\mathrm{H}\,\emptyset}/_{\equiv}$$

(where $\mathbb{T}^{\mathrm{H}\,\emptyset}/_{\equiv}$ is $\mathbb{T}^{\mathrm{H}}$ up to variable renaming augmented with an infimum $\emptyset$) is defined as follows (the most general environment $\mathcal{H} \in \mathbb{H}^{\mathrm{H}}$ is such that $\forall \mathtt{x} \in \mathbb{X} : \mathcal{H}(\mathtt{x}) \in \mathbb{V}$ and $\forall \mathtt{x} \neq \mathtt{y} \in \mathbb{X} : \mathcal{H}(\mathtt{x}) \neq \mathcal{H}(\mathtt{y})$. 'a, 'b and 'c are fresh type variables):

$$\begin{aligned}
\mathbf{T}^{\mathrm{H}}\llbracket\mathtt{x}\rrbracket &\triangleq \langle \mathcal{H},\ \mathcal{H}(\mathtt{x})\rangle\\
\mathbf{T}^{\mathrm{H}}\llbracket\boldsymbol{\lambda}\mathtt{x}\cdot e\rrbracket &\triangleq (\mathbf{T}^{\mathrm{H}}\llbracket e\rrbracket = \langle H,\ \tau\rangle\ ?\\
&\qquad\qquad \langle H[\mathtt{x} \leftarrow \text{'a}],\ H(\mathtt{x}) \mathbin{\texttt{->}} \tau\rangle \mid \emptyset)\\
\mathbf{T}^{\mathrm{H}}\llbracket e_1(e_2)\rrbracket &\triangleq (\mathbf{T}^{\mathrm{H}}\llbracket e_2\rrbracket = \langle H_2,\ \tau_2\rangle \wedge \mathbf{gci}\{\mathbf{T}^{\mathrm{H}}\llbracket e_1\rrbracket, \langle H_2,\\
&\qquad \tau_2 \mathbin{\texttt{->}} \text{'a}\rangle\} = \langle H,\ \tau_2 \mathbin{\texttt{->}} \tau\rangle\ ?\ \langle H,\ \tau\rangle \mid \emptyset)\\
\mathbf{T}^{\mathrm{H}}\llbracket\boldsymbol{\mu}\mathtt{f}\cdot\boldsymbol{\lambda}\mathtt{x}\cdot e\rrbracket &\triangleq (\mathbf{T}^{\mathrm{H}}\llbracket\boldsymbol{\lambda}\mathtt{x}\cdot e\rrbracket = \langle H,\ \tau\rangle\ \wedge\\
&\qquad \sigma = \mathbf{mgu}\{\text{'a} \mathbin{\texttt{->}} \text{'b}, H(\mathtt{f}), \tau\} \neq \emptyset\ ?\\
&\qquad\qquad \langle \sigma(H)[\mathtt{f} \leftarrow \text{'c}],\ \sigma(\tau)\rangle \mid \emptyset)\\
\mathbf{T}^{\mathrm{H}}\llbracket\mathbf{1}\rrbracket &\triangleq \langle \mathcal{H},\ \mathtt{int}\rangle\\
\mathbf{T}^{\mathrm{H}}\llbracket e_1 - e_2\rrbracket &\triangleq \mathbf{gci}\{\langle \mathcal{H},\ \mathtt{int}\rangle, \mathbf{T}^{\mathrm{H}}\llbracket e_1\rrbracket, \mathbf{T}^{\mathrm{H}}\llbracket e_2\rrbracket\}\\
\mathbf{T}^{\mathrm{H}}\llbracket(e_1\ \mathbf{?}\ e_2 : e_3)\rrbracket &\triangleq (\mathbf{T}^{\mathrm{H}}\llbracket e_1\rrbracket = \langle H_1,\ \mathtt{int}\rangle\ ?\\
&\qquad \mathbf{gci}\{\langle H_1,\ \text{'a}\rangle, \mathbf{T}^{\mathrm{H}}\llbracket e_2\rrbracket, \mathbf{T}^{\mathrm{H}}\llbracket e_3\rrbracket\} \mid \emptyset)
\end{aligned}$$

The soundness of Hindley type semantics is by construction:

Proposition 6 $\langle \mathbb{T}^{\mathrm{H}\,\emptyset}/_{\equiv},\ \geq,\ \mathbf{T}^{\mathrm{H}}\llbracket\bullet\rrbracket\rangle$ *is an exact abstraction of Church/Curry monotype abstract semantics* $\langle \mathbb{T}^{\mathrm{C}},\ \supseteq,\ \mathbf{T}^{\mathrm{C}}\llbracket\bullet\rrbracket\rangle$ *by the Galois insertion* $\langle \mathbb{T}^{\mathrm{C}},\ \subseteq\rangle \underset{\mathbf{lcg}}{\overset{\mathbf{ground}}{\leftrightarrows}} \langle \mathbb{T}^{\mathrm{H}\,\emptyset}/_{\equiv},\ \leq\rangle$ *since for all* $e \in \mathbb{E}$:

$$\mathbf{T}^{\mathrm{H}}\llbracket e\rrbracket \triangleq \mathbf{lcg}(\mathbf{T}^{\mathrm{C}}\llbracket e\rrbracket), \qquad \mathbf{T}^{\mathrm{C}}\llbracket e\rrbracket = \mathbf{ground}(\mathbf{T}^{\mathrm{H}}\llbracket e\rrbracket).$$

Proof By structural induction on e. For short, typical cases are considered.

[5] In general $\mathbf{T}^\sharp\llbracket e\rrbracket$ is <u>not</u> an abstraction of $\mathbf{T}^\natural\llbracket e\rrbracket$ (hence of $\mathbf{T}^\flat\llbracket e\rrbracket$ through $\gamma^\natural$) since $\mathbf{T}^\natural\llbracket e\rrbracket \leq^\natural \gamma^\sharp(\alpha^\sharp(\mathbf{T}^\natural\llbracket e\rrbracket))$ by the Galois connection whence $\mathbf{T}^\natural\llbracket e\rrbracket \leq^\natural \gamma^\sharp(\mathbf{T}^\sharp\llbracket e\rrbracket)$ whereas a $\geq^\natural$-upper-approximation is required.

- $\mathbf{T}^{\mathrm{H}}[\![\mathtt{x}]\!]$

$\triangleq \mathbf{lcg}(\mathbf{T}^{\mathrm{C}}[\![\mathtt{x}]\!])$ by def. $\mathbf{T}^{\mathrm{H}}[\![\mathtt{x}]\!]$

$= \mathbf{lcg}(\{\langle H,\ H(\mathtt{x})\rangle \mid H \in \mathbb{H}^{\mathrm{C}}\})$ by def. $\mathbf{T}^{\mathrm{C}}[\![\mathtt{x}]\!]$

$= \langle \mathcal{H},\ \mathcal{H}(\mathtt{x})\rangle$ by def. $\mathbf{lcg}$ and $\mathcal{H}$

$\mathbf{ground}(\mathbf{T}^{\mathrm{H}}[\![\mathtt{x}]\!])$

$= \{\langle \sigma(\mathcal{H}),\ \sigma(\mathcal{H}(\mathtt{x}))\rangle \mid \sigma \in \mathbb{V} \hookrightarrow \mathbb{M}^{\mathrm{C}}\}$ by def. $\mathbf{ground}$

$= \{\langle H,\ H(\mathtt{x})\rangle \mid H \in \mathbb{H}^{\mathrm{C}}\}$ by def. $\mathcal{H}$ and substitution

$= \mathbf{T}^{\mathrm{C}}[\![\mathtt{x}]\!]$ by def. $\mathbf{T}^{\mathrm{C}}[\![\mathtt{x}]\!]$

- $\mathbf{T}^{\mathrm{H}}[\![\mu\mathtt{f}\cdot\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]$

$\triangleq \mathbf{lcg}\big[\mathbf{T}^{\mathrm{C}}[\![\mu\mathtt{f}\cdot\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\big]$ by def. $\mathbf{T}^{\mathrm{H}}[\![\mu\mathtt{f}\cdot\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]$

$= \mathbf{lcg}\big[\{\langle H,\ m\rangle \mid \langle H[\mathtt{f}\leftarrow m],\ m\rangle \in \mathbf{T}^{\mathrm{C}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\}\big]$ by def. $\mathbf{T}^{\mathrm{C}}[\![\mu\mathtt{f}\cdot\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]$

$= \mathbf{lcg}\big[\{\langle H,\ m_1 \to m_2\rangle \mid \langle H[\mathtt{f}\leftarrow m_1 \to m_2],\ m_1 \to m_2\rangle \in \mathbf{T}^{\mathrm{C}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\}\big]$ by def. $\mathbf{T}^{\mathrm{C}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]$

$= \mathbf{lcg}\big[\{\langle H,\ m_1 \to m_2\rangle \mid \langle H[\mathtt{f}\leftarrow m_1 \to m_2],\ m_1 \to m_2\rangle \in \mathbf{ground}(\mathbf{T}^{\mathrm{H}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!])\}\big]$ by induction hypothesis

$= \mathbf{lcg}\big[\{\langle H, m_1 \to m_2\rangle \mid \langle H[\mathtt{f}\leftarrow m_1 \to m_2], m_1 \to m_2\rangle \in \{\langle\sigma(A), \sigma(\tau)\rangle \mid \sigma \in \mathbb{V} \hookrightarrow \mathbb{M}^{\mathrm{C}} \wedge \langle A,\ \tau\rangle = \mathbf{T}^{\mathrm{H}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\}\}\big]$ by def. $\mathbf{ground}$

$= \mathbf{lcg}\big[\{\langle H,\ m_1 \to m_2\rangle \mid H[\mathtt{f}\leftarrow m_1 \to m_2] = \sigma(A) \wedge m_1 \to m_2 = \sigma(\tau) \wedge \sigma \in \mathbb{V} \hookrightarrow \mathbb{M}^{\mathrm{C}} \wedge \langle A,\ \tau\rangle = \mathbf{T}^{\mathrm{H}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\}\big]$ by def. $\in$ and $\langle\bullet,\ \bullet\rangle$

$= \mathbf{lcg}\big[\{\langle H, m_1 \to m_2\rangle \mid \forall \mathtt{y} \neq \mathtt{f} : H(\mathtt{y}) = \sigma(A)(\mathtt{y}) \wedge \sigma(A)(\mathtt{f}) = m_1 \to m_2 = \sigma(\tau) \wedge \sigma \in \mathbb{V} \hookrightarrow \mathbb{M}^{\mathrm{C}} \wedge \langle A,\ \tau\rangle = \mathbf{T}^{\mathrm{H}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\}\big]$ by def. $H[\bullet\leftarrow\bullet]$

$= \mathbf{lcg}\big[\{\langle H,\ m_1 \to m_2\rangle \mid H = \sigma(A[\mathtt{f}\leftarrow \mathtt{'c}]) \wedge m_1 \to m_2 = \sigma(A(\mathtt{f})) = \sigma(\tau) \wedge \sigma \in \mathbb{V} \hookrightarrow \mathbb{M}^{\mathrm{C}} \wedge \langle A,\ \tau\rangle = \mathbf{T}^{\mathrm{H}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\}\big]$ by def. $A[\bullet\leftarrow\bullet]$, $A(\bullet)$ and substitution where 'c is a fresh type variable

$= \mathbf{lcg}\big[\{\langle\sigma(A[\mathtt{f}\leftarrow \mathtt{'c}]),\ \sigma(\mathtt{'a} \to \mathtt{'b})\rangle \mid \sigma(\mathtt{'a} \to \mathtt{'b}) = \sigma(A(\mathtt{f})) = \sigma(\tau) \wedge \sigma \in \mathbb{V} \hookrightarrow \mathbb{M}^{\mathrm{C}} \wedge \langle A,\ \tau\rangle = \mathbf{T}^{\mathrm{H}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\}\big]$ by def. $=$ and substitution where 'a and 'b are fresh type variables

$= (\mathbf{T}^{\mathrm{H}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!] = \langle A,\ \tau\rangle \wedge \varsigma = \mathbf{mgu}\{\mathtt{'a} \to \mathtt{'b}, A(\mathtt{f}), \tau\} \neq \emptyset\ ?\ \mathbf{lcg}\big[\{\langle\sigma' \circ \varsigma(A[\mathtt{f}\leftarrow \mathtt{'c}]),\ \sigma' \circ \varsigma(\tau)\rangle \mid \sigma' \in \mathbb{V} \hookrightarrow \mathbb{M}^{\mathrm{C}}\}\big] \mid \emptyset)$ by def. of $\mathbf{mgu}$

$= (\mathbf{T}^{\mathrm{H}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!] = \langle A,\ \tau\rangle \wedge \varsigma = \mathbf{mgu}\{\mathtt{'a} \to \mathtt{'b}, A(\mathtt{f}), \tau\} \neq \emptyset\ ?\ \mathbf{lcg}\big[\mathbf{ground}(\langle\varsigma(A[\mathtt{f}\leftarrow \mathtt{'c}]),\ \varsigma(\tau)\rangle)\big] \mid \emptyset)$ by def. of $\mathbf{ground}$

$= (\mathbf{T}^{\mathrm{H}}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!] = \langle A,\ \tau\rangle \wedge \varsigma = \mathbf{mgu}\{\mathtt{'a} \to \mathtt{'b}, A(\mathtt{f}), \tau\} \neq \emptyset\ ?\ \langle\varsigma(A[\mathtt{f}\leftarrow \mathtt{'c}]),\ \varsigma(\tau)\rangle \mid \emptyset)$ since $\mathbf{lcg} \circ \mathbf{ground} = \mathbf{Id}$

The proof that $\mathbf{ground}(\mathbf{T}^{\mathrm{H}}[\![e]\!]) = \mathbf{T}^{\mathrm{C}}[\![e]\!]$ is obtained by the terms between square brackets $\big[\ldots\big]$ when reading the above proof backwards. $\square$

14 A Type Collecting Semantics

The question "What is a type system" has hardly received a formal answer. Type systems can be viewed as abstract semantics in the lattice of abstract interpretations [9] which are more abstract than a *type collecting semantics* which is the most general type system in that it is more precise than the reduced product [9] of all existing type systems. With respect to the general collecting semantics $\mathbf{C}[\![\bullet]\!]$, this type collecting semantics factors out all approximations introduced by type systems such as "run-time values, except ω, are not taken into account"[6] or the even more debatable "both branches of a conditional should have the same type".

The approximation is mainly introduced through the definition of typings $\mathbb{P}^{\mathrm{Co}}$ and their correspondence $\langle\alpha^{\mathrm{Co}},\ \gamma^{\mathrm{Co}}\rangle$ with program properties. Further approximations are introduced by the requirement that the type collecting semantics semantics $\mathbf{T}^{\mathrm{Co}}[\![\bullet]\!]$ should be a *compositional* upper-approximation of the best typing $\alpha^{\mathrm{Co}}(\mathbf{C}[\![\bullet]\!])$[7]. Finally fixpoint approximations later introduce further approximations[8].

In what follows $\perp^{\mathrm{Co}}$ is the type of never terminating computations. `int` is the type of computations returning an integer if and when terminating. $\langle t_1,\ t_2\rangle$, traditionally written $t_1 \to t_2$, is the type of functions which given an argument of type t_1 deliver, if ever, a result with type t_2. An object of polytype $T \subseteq \mathbb{P}^{\mathrm{Co}}$ has all types $t \in T$. The collecting polytypes $\mathbb{P}^{\mathrm{Co}}$ are defined as follows:

$$\begin{array}{rcll}
\mathbb{P}^{\mathrm{Co}\,0} & \triangleq & \{\perp^{\mathrm{Co}}, \mathtt{int}\} & \text{basic types}\\
\mathbb{P}^{\mathrm{Co}\,\delta+1} & \triangleq & \mathbb{P}^{\mathrm{Co}\,\delta} \cup \wp(\mathbb{P}^{\mathrm{Co}\,\delta} \times \mathbb{P}^{\mathrm{Co}\,\delta}) & \delta+1 \text{ successor ordinal}\\
\mathbb{P}^{\mathrm{Co}\,\lambda} & \triangleq & \bigcup_{\delta<\lambda} \mathbb{P}^{\mathrm{Co}\,\delta} & \lambda \text{ limit ordinal}\\
t \in T \subseteq \mathbb{P}^{\mathrm{Co}} & \triangleq & \bigcup_{\delta\in\mathbb{O}} \mathbb{P}^{\mathrm{Co}\,\delta} & \mathbb{O} \text{ is the class of ordinals}
\end{array}$$

The meaning of collecting polytypes is defined by the *concretization function*:

$$\gamma_1^{\mathrm{Co}} \in \mathbb{P}^{\mathrm{Co}} \mapsto \wp(\mathbb{U})$$

The definition of γ_1^{Co} makes use of $\mathbb{O}$-termed sequences of functions $\vec{\gamma}_1^{\mathrm{Co}\,\delta} \in \mathbb{P}^{\mathrm{Co}\,\delta} \mapsto \wp(\mathbb{U})$, $\delta \in \mathbb{O}$ and $\vec{\gamma}_1^{\mathrm{Co}\,\delta} \in (\mathbb{P}^{\mathrm{Co}\,\delta} \times \mathbb{P}^{\mathrm{Co}\,\delta}) \mapsto \wp(\mathbb{U})$, $\delta \in \mathbb{O}$ defined by transfinite recursion as follows:

$$\gamma_1^{\mathrm{Co}\,0}(\perp^{\mathrm{Co}}) \triangleq \{\perp\} \qquad \gamma_1^{\mathrm{Co}\,0}(\mathtt{int}) \triangleq \{\uparrow(\mathsf{z}) :: \mathbb{Z}_\perp \mid \mathsf{z} \in \mathbb{Z}\} \cup \{\perp\}$$

$$\begin{array}{rcll}
\vec{\gamma}_1^{\mathrm{Co}\,\delta}(t_1 \to t_2) & \triangleq & \multicolumn{2}{l}{\{\uparrow(\varphi) :: [\mathbb{U} \mapsto \mathbb{U}]_\perp \mid \varphi \in [\mathbb{U} \mapsto \mathbb{U}] \wedge \forall \mathsf{u} \in \gamma_1^{\mathrm{Co}\,\delta}(t_1) : \varphi(\mathsf{u}) \in \gamma_1^{\mathrm{Co}\,\delta}(t_2)\} \cup \{\perp\}, \quad t_1 \to t_2 \in \mathbb{P}^{\mathrm{Co}\,\delta} \times \mathbb{P}^{\mathrm{Co}\,\delta}}\\
\gamma_1^{\mathrm{Co}\,\delta+1}(t) & \triangleq & \gamma_1^{\mathrm{Co}\,\delta}(t) & \text{if } t \in \mathbb{P}^{\mathrm{Co}\,\delta}\\
\gamma_1^{\mathrm{Co}\,\delta+1}(T) & \triangleq & \bigcap_{t_1 \to t_2 \in T} \vec{\gamma}_1^{\mathrm{Co}\,\delta}(t_1 \to t_2) & \text{if } T \subseteq \mathbb{P}^{\mathrm{Co}\,\delta} \times \mathbb{P}^{\mathrm{Co}\,\delta}\\
\gamma_1^{\mathrm{Co}}(\emptyset^{\mathrm{Co}}) & \triangleq & \mathbb{U} & \text{when } T = \emptyset^{\mathrm{Co}} \triangleq \emptyset\\
\gamma_1^{\mathrm{Co}\,\lambda}(t) & \triangleq & \gamma_1^{\mathrm{Co}\,\delta}(t) & \text{if } \lambda \text{ limit ord., } \delta<\lambda \text{ and } t \in \mathbb{P}^{\mathrm{Co}\,\delta}\\
\gamma_1^{\mathrm{Co}}(t) & \triangleq & \gamma_1^{\mathrm{Co}\,\delta}(t) & \text{where } t \in \mathbb{P}^{\mathrm{Co}\,\delta}
\end{array}$$

Collecting polytypes t, more precisely the set $\gamma_1^{\mathrm{Co}}(t)$ of values that they describe, are ideals [28, 41] in that $\gamma_1^{\mathrm{Co}}(t)$ is not empty since $\perp \in \gamma_1^{\mathrm{Co}}(t)$, $\gamma_1^{\mathrm{Co}}(t)$ is downwards closed (if $\mathsf{u}, \mathsf{v} \in \mathbb{U}$, $\mathsf{u} \sqsubseteq \mathsf{v}$ and $\mathsf{v} \in \gamma_1^{\mathrm{Co}}(t)$ then $\mathsf{u} \in \gamma_1^{\mathrm{Co}}(t)$) and $\gamma_1^{\mathrm{Co}}(t)$ is closed under lubs of increasing chains (if $\mathsf{u}_\delta, \delta < \epsilon$ is a $\sqsubseteq$-increasing chain of elements of $\gamma_1^{\mathrm{Co}}(t)$ then $\sqcup_{\delta<\epsilon}\mathsf{u}_\delta \in \gamma_1^{\mathrm{Co}}(t)$).

[6] This is certainly not the case for intervals in PASCAL or dependent types.

[7] In absence of best abstractions, approximations can be introduced dynamically by widenings [11] which are omitted here for short.

[8] Widenings [6, 7] also play a central rôle for precise fixpoint approximation. See [33] for type widenings.

The *subtyping* preorder $\leq^{\mathrm{Co}}$ is defined on collecting polytypes $\mathbb{P}^{\mathrm{Co}}$ as:

$$t_1 \leq^{\mathrm{Co}} t_2 \triangleq \gamma_1^{\mathrm{Co}}(t_1) \subseteq \gamma_1^{\mathrm{Co}}(t_2)$$

which is an abstract version of logical implication. By considering the quotient set $\mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}}$ with equivalence

$$t_1 \equiv^{\mathrm{Co}} t_2 \triangleq (t_1 \leq^{\mathrm{Co}} t_2 \wedge t_2 \leq^{\mathrm{Co}} t_1)$$

$\langle \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}}, \leq^{\mathrm{Co}}\rangle$ is a partial order. The subtyping order is left antitone and right monotone for functional types (as, e.g., in [28]):

Lemma 7 *$t_1 \to t_2 \leq^{\mathrm{Co}} \bot^{\mathrm{Co}} \to t_2$. If $t_1' \neq \bot^{\mathrm{Co}}$ then $t_1 \to t_2 \leq^{\mathrm{Co}} t_1' \to t_2'$ if and only if $t_1' \leq^{\mathrm{Co}} t_1 \wedge t_2 \leq^{\mathrm{Co}} t_2'$.*

Let us define $\bigwedge^{\mathrm{Co}} \in \wp(\mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}}) \mapsto \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}}$ as:

$$\begin{aligned}
\textstyle\bigwedge^{\mathrm{Co}} T &\triangleq \bot^{\mathrm{Co}} \quad \text{if } \bot^{\mathrm{Co}} \in T \text{ or } T \text{ contains both } \mathtt{int} \text{ and } t_1 \to t_2\\
&\triangleq \mathtt{int} \quad \text{if } T = \{\mathtt{int}\}\\
&\triangleq \emptyset^{\mathrm{Co}} \quad \text{if } T = \emptyset\\
&\triangleq \textstyle\bigcup T \text{ if } \emptyset \neq T \subseteq \wp(\mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}} \times \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}})
\end{aligned}$$

γ_1^{Co} preserves existing greatest lower bounds (glbs):

Lemma 8 *$\gamma_1^{\mathrm{Co}}(\bigwedge^{\mathrm{Co}} T) = \bigcap_{t \in T} \gamma_1^{\mathrm{Co}}(t)$ and $\bigwedge^{\mathrm{Co}} T$ is the $\leq^{\mathrm{Co}}$-greatest lower bound of T.*

γ_1^{Co} preserves glbs and is injective so that:

Lemma 9 $\langle \wp(\mathbb{U}), \subseteq\rangle \underset{\alpha_1^{\mathrm{Co}}}{\overset{\gamma_1^{\mathrm{Co}}}{\leftrightarrows}} \langle \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}}, \leq^{\mathrm{Co}}\rangle$

is a Galois insertion where, as it is the case for all Galois connections [9], α_1^{Co} is uniquely determined as:

$$\alpha_1^{\mathrm{Co}}(U) \triangleq \textstyle\bigwedge^{\mathrm{Co}}\{t \in \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}} \mid U \subseteq \gamma_1^{\mathrm{Co}}(t)\}$$

and, α_1^{Co} being surjective, $\alpha_1^{\mathrm{Co}}(\wp(\mathbb{U})) \triangleq \{\alpha_1^{\mathrm{Co}}(U) \mid U \subseteq \mathbb{U}\} = \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}}$ is a complete lattice:

Lemma 10 *$\langle \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}}, \leq^{\mathrm{Co}}, \bot^{\mathrm{Co}}, \emptyset^{\mathrm{Co}}, \bigvee^{\mathrm{Co}}, \bigwedge^{\mathrm{Co}}\rangle$ is a complete lattice where $\bigvee^{\mathrm{Co}} T \triangleq \alpha_1^{\mathrm{Co}}(\bigcup_{t\in T} \gamma_1^{\mathrm{Co}}(t))$.*

Next, the abstraction is extended to environment properties approximated by type environments:

$$H \in \mathbb{H}^{\mathrm{Co}} \triangleq \mathbb{X} \mapsto \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}} \qquad \text{type environment}$$

with pointwise ordering:

$$\begin{aligned}
\alpha_2^{\mathrm{Co}}(\mathcal{R}) &\triangleq \Lambda \mathtt{x} \in \mathbb{X} \cdot \alpha_1^{\mathrm{Co}}(\{\mathsf{R}(\mathtt{x}) \mid \mathsf{R} \in \mathcal{R}\})\\
\gamma_2^{\mathrm{Co}}(H) &\triangleq \{\mathsf{R} \in \mathbb{R} \mid \forall \mathtt{x} \in \mathbb{X} : \mathsf{R}(\mathtt{x}) \in \gamma_1^{\mathrm{Co}}(H(\mathtt{x}))\}\\
H_1 \mathrel{\dot{\leq}^{\mathrm{Co}}} H_2 &\triangleq \forall \mathtt{x} \in \mathbb{X} : H_1(\mathtt{x}) \leq^{\mathrm{Co}} H_2(\mathtt{x})
\end{aligned}$$

so that:

Lemma 11

$$\langle \wp(\mathbb{R}), \subseteq, \emptyset, \mathbb{R}, \cap, \cup\rangle \underset{\alpha_2^{\mathrm{Co}}}{\overset{\gamma_2^{\mathrm{Co}}}{\leftrightarrows}} \langle \mathbb{H}^{\mathrm{Co}}, \dot{\leq}^{\mathrm{Co}}, \dot{\bot}^{\mathrm{Co}}, \dot{\emptyset}^{\mathrm{Co}}, \dot{\bigvee}^{\mathrm{Co}}, \dot{\bigwedge}^{\mathrm{Co}}\rangle$$

is a Galois insertion between complete lattices. Finally the abstraction is extended to program properties $P \in \mathbb{P}$ approximated by typings. A typing maps type environments to program types where a type environment assigns types to global/free variables:

$$\theta \in \mathbb{T}^{\mathrm{Co}} \triangleq \mathbb{H}^{\mathrm{Co}} \mapsto \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}} \qquad \text{typing}$$

with pointwise ordering:

$$\begin{aligned}
\alpha^{\mathrm{Co}}(P) &\triangleq \Lambda H \in \mathbb{H}^{\mathrm{Co}} \cdot \alpha_1^{\mathrm{Co}}(\{\phi(\mathsf{R}) \mid \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H) \wedge \phi \in P\})\\
\gamma^{\mathrm{Co}}(\theta) &\triangleq \{\phi \mid \forall H \in \mathbb{H}^{\mathrm{Co}} : \forall \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H) : \phi(\mathsf{R}) \in \gamma_1^{\mathrm{Co}}(\theta(H))\}\\
\theta_1 \mathrel{\ddot{\leq}^{\mathrm{Co}}} \theta_2 &\triangleq \forall H \in \mathbb{H}^{\mathrm{Co}} : \theta_1(H) \leq^{\mathrm{Co}} \theta_2(H)
\end{aligned}$$

so that one gets Galois connection between complete lattices:

Proposition 12

$$\langle \mathbb{P}, \subseteq, \emptyset, \mathbb{S}, \cap, \cup\rangle \underset{\alpha^{\mathrm{Co}}}{\overset{\gamma^{\mathrm{Co}}}{\leftrightarrows}} \langle \mathbb{T}^{\mathrm{Co}}, \ddot{\leq}^{\mathrm{Co}}, \ddot{\bot}^{\mathrm{Co}}, \ddot{\emptyset}^{\mathrm{Co}}, \ddot{\bigvee}^{\mathrm{Co}}, \ddot{\bigwedge}^{\mathrm{Co}}\rangle$$

The type collecting semantics has been designed using the soundness requirement (see **Prop. 15** below) as follows:

$$\begin{aligned}
\mathbf{T}^{\mathrm{Co}}[\![\mathtt{x}]\!] &\triangleq \Lambda H \cdot H(\mathtt{x})\\
\mathbf{T}^{\mathrm{Co}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \Lambda H \cdot \textstyle\bigwedge^{\mathrm{Co}}\big\{t \to \mathbf{T}^{\mathrm{Co}}[\![e]\!]H[\mathtt{x} \leftarrow t] \bigm| t \in \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}} - \{\bot^{\mathrm{Co}}\}\big\}\\
\mathbf{T}^{\mathrm{Co}}[\![e_1(e_2)]\!] &\triangleq \Lambda H \cdot \textstyle\bigwedge^{\mathrm{Co}}\big\{t \in \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}} \bigm| \mathbf{T}^{\mathrm{Co}}[\![e_1]\!]H \leq^{\mathrm{Co}} \{\mathbf{T}^{\mathrm{Co}}[\![e_2]\!] \to t\}\big\}\\
\mathbf{T}^{\mathrm{Co}}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \Lambda H \cdot \mathbf{lfp}^{\leq^{\mathrm{Co}}}_{\{\emptyset^{\mathrm{Co}} \to \bot^{\mathrm{Co}}\}} \Lambda t \in \mathbb{P}^{\mathrm{Co}}/_{\equiv^{\mathrm{Co}}} \cdot \mathbf{T}^{\mathrm{Co}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]H[\mathtt{f} \leftarrow t]\\
\mathbf{T}^{\mathrm{Co}}[\![\mathbf{1}]\!] &\triangleq \Lambda H \cdot \mathtt{int}\\
\mathbf{T}^{\mathrm{Co}}[\![e_1 - e_2]\!] &\triangleq \Lambda H \cdot \mathtt{int} \wedge^{\mathrm{Co}} \mathbf{T}^{\mathrm{Co}}[\![e_1]\!] \wedge^{\mathrm{Co}} \mathbf{T}^{\mathrm{Co}}[\![e_2]\!]\\
\mathbf{T}^{\mathrm{Co}}[\![(e_1 \mathbin{?} e_2 \mathbin{:} e_3)]\!] &\triangleq \Lambda H \cdot (\mathbf{T}^{\mathrm{Co}}[\![e_1]\!]H = \bot^{\mathrm{Co}} \mathbin{?} \bot^{\mathrm{Co}} \mid \mathbf{T}^{\mathrm{Co}}[\![e_1]\!]H = \mathtt{int} \mathbin{?} \mathbf{T}^{\mathrm{Co}}[\![e_2]\!]H \vee^{\mathrm{Co}} \mathbf{T}^{\mathrm{Co}}[\![e_3]\!]H \mid \emptyset^{\mathrm{Co}})
\end{aligned}$$

Example 13 We could have avoided explicit fixpoint definitions $\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e$ using the eager call by value $\mathbf{Y}$ combinator $\mathbf{Y} = \boldsymbol{\lambda}\mathtt{f} \cdot \mathbf{W}(\mathbf{W})$ where $\mathbf{W} = \boldsymbol{\lambda}\mathtt{x} \cdot \boldsymbol{\lambda}\mathtt{y} \cdot \mathtt{f}(\mathtt{x}(\mathtt{x}))(\mathtt{y})$. The collecting type of $\mathbf{Y}$ is such that:

$$\mathbf{T}^{\mathrm{Co}}[\![\mathbf{Y}]\!] \mathrel{\ddot{\leq}^{\mathrm{Co}}} \Lambda H \cdot \textstyle\bigwedge^{\mathrm{Co}}\big\{\{\{t_1 \to t_2\} \to \{t_1 \to t_2\}\} \to \{t_1 \to t_2\} \bigm| t_1 \in \mathbb{P}^{\mathrm{Co}} - \{\bot^{\mathrm{Co}}\} \wedge t_2 \in \mathbb{P}^{\mathrm{Co}}\big\} \qquad \square$$

Any $\ddot{\leq}^{\mathrm{Co}}$-upper-approximation of the best abstraction of the standard collecting semantics is sound:

Lemma 14 *$\forall e \in \mathbb{E}$: $\forall H \in \mathbb{H}^{\mathrm{Co}}$: $\forall t \in \mathbb{P}^{\mathrm{Co}}$:*
$[\alpha^{\mathrm{Co}}(\mathbf{C}[\![e]\!])H \leq^{\mathrm{Co}} t] \iff [\forall \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H) : \mathbf{S}[\![e]\!]\mathsf{R} \in \gamma_1^{\mathrm{Co}}(t)]$

Proof

$$\begin{aligned}
& \forall \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H) : \mathbf{S}[\![e]\!]\mathsf{R} \in \gamma_1^{\mathrm{Co}}(t)\\
\iff & \forall \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H) : \{\mathbf{S}[\![e]\!]\mathsf{R}\} \subseteq \gamma_1^{\mathrm{Co}}(t) && \text{def. } \cup\\
\iff & \forall \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H) : \alpha_1^{\mathrm{Co}}(\{\mathbf{S}[\![e]\!]\mathsf{R}\}) \leq^{\mathrm{Co}} t && \text{def. Galois con.}\\
\iff & \textstyle\bigvee^{\mathrm{Co}}\{\alpha_1^{\mathrm{Co}}(\{\mathbf{S}[\![e]\!]\mathsf{R}\}) \mid \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H)\} \leq^{\mathrm{Co}} t && \text{def. lubs}\\
\iff & \alpha_1^{\mathrm{Co}}(\cup\{\{\mathbf{S}[\![e]\!]\mathsf{R}\} \mid \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H)\}) \leq^{\mathrm{Co}} t && \alpha_1^{\mathrm{Co}} \text{ preserves existing lubs}\\
\iff & \alpha_1^{\mathrm{Co}}(\{\mathbf{S}[\![e]\!]\mathsf{R} \mid \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H)\}) \leq^{\mathrm{Co}} t && \text{def. } \cup\\
\iff & \alpha_1^{\mathrm{Co}}(\{\phi(\mathsf{R}) \mid \phi \in \{\mathbf{S}[\![e]\!]\} \wedge \mathsf{R} \in \gamma_2^{\mathrm{Co}}(H)\}) \leq^{\mathrm{Co}} t && \text{def. } \in\\
\iff & \alpha^{\mathrm{Co}}(\{\mathbf{S}[\![e]\!]\}) \leq^{\mathrm{Co}} t && \text{def. } \alpha^{\mathrm{Co}}\\
\iff & \alpha^{\mathrm{Co}}(\mathbf{C}[\![e]\!]) \leq^{\mathrm{Co}} t && \text{def. } \mathbf{C}[\![\bullet]\!]. \qquad \square
\end{aligned}$$

This soundness requirement provides a guideline for deriving the definition of $\mathbf{T}^{Co}[\![\bullet]\!]$ by upper-approximation of $\alpha^{Co}(\mathbf{C}[\![e]\!])$:

Proposition 15 *If the type collecting semantics $\mathbf{T}^{Co}[\![\bullet]\!]$ is monotone*[9] *then it is well-defined and sound since for all $e \in \mathbb{E}$:*

$$\alpha^{Co}(\mathbf{C}[\![e]\!]) \ \ddot{\leq}^{Co} \ \mathbf{T}^{Co}[\![e]\!]$$

Proof

$$\begin{aligned}
&\alpha^{Co}(\mathbf{C}[\![e]\!])\\
=\ &\alpha^{Co}(\{\mathbf{S}[\![e]\!]\}) && \text{def. } \mathbf{C}[\![\bullet]\!]\\
=\ &\alpha_1^{Co}(\{\phi(\mathsf{R}) \mid \phi \in \{\mathbf{S}[\![e]\!]\} \wedge \mathsf{R} \in \gamma_2^{Co}(H)\}) && \text{def. } \alpha^{Co}\\
=\ &\alpha_1^{Co}(\{\mathbf{S}[\![e]\!]\mathsf{R} \mid \mathsf{R} \in \gamma_2^{Co}(H)\}) && \text{def. } \in\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \alpha_1^{Co}(\{\mathbf{S}[\![e]\!]\mathsf{R} \mid \mathsf{R} \in \gamma_2^{Co}(H)\}) \leq^{Co} t\} && \text{def. glbs}\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \{\mathbf{S}[\![e]\!]\mathsf{R} \mid \mathsf{R} \in \gamma_2^{Co}(H)\}) \subseteq \gamma_1^{Co}(t)\} && \text{Galois connection}\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \forall \mathsf{R} \in \gamma_2^{Co}(H) : \mathbf{S}[\![e]\!]\mathsf{R} \in \gamma_1^{Co}(t)\} && \text{def. } \subseteq
\end{aligned}$$

The proof goes on by structural induction on e. For short, only few typical cases are considered.

- $\alpha^{Co}(\mathbf{C}[\![\mathtt{x}]\!])$

$$\begin{aligned}
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \forall \mathsf{R} \in \gamma_2^{Co}(H) : \mathbf{S}[\![\mathtt{x}]\!]\mathsf{R} \in \gamma_1^{Co}(t)\}\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \forall \mathsf{R} \in \mathbb{R} : \forall \mathtt{y} \in \mathbb{X} : \mathsf{R}(\mathtt{y}) \in \gamma_1^{Co}(H(\mathtt{y})) \Longrightarrow \mathsf{R}(x) \in \gamma_1^{Co}(t)\} && \text{by def. } \gamma_2^{Co}(H) \text{ and } \mathbf{S}[\![\mathtt{x}]\!]\mathsf{R}\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \forall \mathsf{R} \in \mathbb{R} : \mathsf{R}(\mathtt{x}) \in \gamma_1^{Co}(H(\mathtt{x})) \Longrightarrow \mathsf{R}(\mathtt{x}) \in \gamma_1^{Co}(t)\} && \text{since } \mathsf{R}(\mathtt{x}) \text{ is independent of } \mathsf{R}(\mathtt{y}),\ \mathtt{y} \neq \mathtt{x}\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \forall \mathsf{u} \in \mathbb{U} : \mathsf{u} \in \gamma_1^{Co}(H(\mathtt{x})) \Longrightarrow \mathsf{u} \in \gamma_1^{Co}(t)\} && \text{by def. } \mathbb{R} \text{ and } \mathsf{R}(\mathtt{x})\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \gamma_1^{Co}(H(\mathtt{x})) \subseteq \gamma_1^{Co}(t)\} && \text{by def. } \subseteq\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid H(\mathtt{x}) \leq^{Co} t\} && \text{by def. } \leq^{Co}\\
=\ &H(\mathtt{x}) && \text{by def. glbs}\\
\triangleq\ &\mathbf{T}^{Co}[\![\mathtt{x}]\!] && \text{since } \alpha^{Co} \text{ and } \gamma^{Co} \text{ have been eliminated.}
\end{aligned}$$

- $\alpha^{Co}(\mathbf{C}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!])$

$$\begin{aligned}
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \forall \mathsf{R} \in \gamma_2^{Co}(H) : \mathbf{S}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\mathsf{R} \in \gamma_1^{Co}(t)\}\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \forall \mathsf{R} \in \gamma_2^{Co}(H) : \\
&\qquad \mathbf{lfp}^{\sqsubseteq}_{\uparrow(\Lambda\mathsf{u}\cdot\,\bot)::[\mathbb{U}\mapsto\mathbb{U}]_\bot}\ \Lambda\mathsf{f}\cdot\, \mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\mathsf{R}[\mathtt{f} \leftarrow \mathsf{f}] \in \gamma_1^{Co}(t)\}\\
=\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \forall \mathsf{R} \in \gamma_2^{Co}(H) : \mathsf{f}^0 = \uparrow(\Lambda\mathsf{u}\cdot\,\bot) :: [\mathbb{U} \mapsto \mathbb{U}]_\bot\\
&\qquad \wedge \forall n \in \mathbb{N} : \mathsf{f}^{n+1} = \mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\mathsf{R}[\mathtt{f} \leftarrow \mathsf{f}^n] \wedge \bigsqcup_{n\in\mathbb{N}} \mathsf{f}^n \in \gamma_1^{Co}(t)\}\\
&&& \text{by continuity and Kleene fixpoint theorem}
\end{aligned}$$

Requiring all iterates of the fixpoint, hence all intermediate steps of the computation to be typed, one gets, by definition of glbs, an upper approximation:

$$\begin{aligned}
\leq^{Co}\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \exists t_n, n \in \mathbb{N} : \forall \mathsf{R} \in \gamma_2^{Co}(H) : \mathsf{f}^0 = \uparrow(\Lambda\mathsf{u}\cdot\,\bot) :: [\mathbb{U} \mapsto \mathbb{U}]_\bot \wedge t_0 = \{\emptyset^{Co} \to \bot^{Co}\} \wedge \forall n \in \mathbb{N} :\\
&\mathsf{f}^{n+1} = \mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\mathsf{R}[\mathtt{f} \leftarrow \mathsf{f}^n] \in \gamma_1^{Co}(t_{n+1}) \wedge \bigsqcup_{n\in\mathbb{N}} \mathsf{f}^n \in \gamma_1^{Co}(t)\}
\end{aligned}$$

Observe that $\forall n \in \mathbb{N}$: $\mathsf{f}^n \in \gamma_1^{Co}(t_n)$ so $\bigsqcup_{n\in\mathbb{N}} \mathsf{f}^n \in \gamma_1^{Co}(\bigvee^{Co}_{n\in\mathbb{N}} t_n)$ since types are ideals. So $\bigsqcup_{n\in\mathbb{N}} \mathsf{f}^n \in \gamma_1^{Co}(t)$ can be ensured by the stronger requirement $\bigvee^{Co}_{n\in\mathbb{N}} t_n \leq^{Co} t$. In this way, one gets an upper approximation:

$$\begin{aligned}
\leq^{Co}\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \exists t_n, n \in \mathbb{N} : \forall \mathsf{R} \in \gamma_2^{Co}(H) : \mathsf{f}^0 = \uparrow(\Lambda\mathsf{u}\cdot\,\bot) :: [\mathbb{U} \mapsto \mathbb{U}]_\bot \wedge t_0 = \{\emptyset^{Co} \to \bot^{Co}\} \wedge \forall n \in \mathbb{N} :\\
&\mathsf{f}^{n+1} = \mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\mathsf{R}[\mathtt{f} \leftarrow \mathsf{f}^n] \wedge \mathsf{f}^n \in \gamma_1^{Co}(t_n) \wedge \bigvee^{Co}_{n\in\mathbb{N}} t_n \leq^{Co} t\}
\end{aligned}$$

since, moreover, $\mathsf{f}^0 \in \gamma_1^{Co}(t_0)$. The iterative definition of the f^n, $n \in \mathbb{N}$ leads to a similar idea for the t_n, $n \in \mathbb{N}$. To do this, one knows by induction hypothesis that for all $H' \in \mathbb{H}^{Co}$:

$$\alpha^{Co}(\{\mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\})H' \leq^{Co} \mathbf{T}^{Co}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]H'$$

whence by **Lem. 14**:

$$\forall \mathsf{R} \in \gamma_2^{Co}(H') : \mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\mathsf{R} \in \gamma_1^{Co}(\mathbf{T}^{Co}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]H')$$

Besides for all $n \in \mathbb{N}$, $\mathsf{f}^n \in \gamma_1^{Co}(t_n)$ whence by definition of γ_2^{Co} and $H[\bullet \leftarrow \bullet]$ one has $\mathsf{R}[\mathtt{f} \leftarrow \mathsf{f}^n] \in \gamma_2^{Co}(H[\mathtt{f} \leftarrow t_n])$ for all $\mathsf{R} \in \gamma_2^{Co}(H)$. It follows that for all $\mathsf{R} \in \gamma_2^{Co}(H)$ one has $\mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\mathsf{R}[\mathtt{f} \leftarrow \mathsf{f}^n] \in \gamma_1^{Co}(\mathbf{T}^{Co}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]H[\mathtt{f} \leftarrow t_n])$. Hence by choosing $t_{n+1} = \mathbf{T}^{Co}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]H[\mathtt{f} \leftarrow t_n]$ one can guarantee that $\mathsf{f}^n \in \gamma_1^{Co}(t_n)$. An upper approximation as been obtained:

$$\begin{aligned}
\leq^{Co}\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \exists t_n, n \in \mathbb{N} : t_0 = \{\emptyset^{Co} \to \bot^{Co}\} \wedge\\
&\forall n \in \mathbb{N} : t_{n+1} \in \mathbf{T}^{Co}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]H[\mathtt{f} \leftarrow t_n] \wedge \bigvee^{Co}_{n\in\mathbb{N}} t_n \leq^{Co} t\}
\end{aligned}$$

By defining $\Phi_H \triangleq \Lambda t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \cdot \mathbf{T}^{Co}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]H[\mathtt{f} \leftarrow t]$, one recognizes the first iterates of $\mathbf{lfp}^{\leq^{Co}}_{\{\emptyset^{Co} \to \bot^{Co}\}}\ \Phi_H$ in t_n, $n \in \mathbb{N}$ provided one can prove Φ_H to be monotone. While designing $\mathbf{T}^{Co}[\![\bullet]\!]$ this was impossible since $\mathbf{T}^{Co}[\![\bullet]\!]$ was not yet completely known. So the proof was postponed and the monotonicity hypothesis was added. So assuming $\mathbf{T}^{Co}[\![\bullet]\!]$ to be monotone, Φ_H is also monotone and $\mathbf{lfp}^{\leq^{Co}}_{\{\emptyset^{Co} \to \bot^{Co}\}}\ \Phi_H$ is well-defined. Moreover $\bigvee^{Co}_{n\in\mathbb{N}} t_n \leq^{Co} \mathbf{lfp}^{\leq^{Co}}_{\{\emptyset^{Co} \to \bot^{Co}\}}\ \Phi_H$ by the constructive version of Tarski's fixpoint theorem [8] so that $\mathbf{lfp}^{\leq^{Co}}_{\{\emptyset^{Co} \to \bot^{Co}\}}\ \Phi_H \leq^{Co} t$ implies $\bigvee^{Co}_{n\in\mathbb{N}} t_n \leq^{Co} t$. Therefore one obtains a further upper-approximation:

$$\begin{aligned}
\leq^{Co}\ &\bigwedge^{Co}\{t \in \mathbb{P}^{Co}/_{\equiv^{Co}} \mid \mathbf{lfp}^{\leq^{Co}}_{\{\emptyset^{Co} \to \bot^{Co}\}}\ \Phi_H \leq^{Co} t\}\\
=\ &\mathbf{lfp}^{\leq^{Co}}_{\{\emptyset^{Co} \to \bot^{Co}\}}\ \Phi_H && \text{by def. of glbs}\\
\triangleq\ &\mathbf{T}^{Co}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!]H
\end{aligned}$$

since this expression is defined in terms of elements of the type collecting semantics only and can hardly be simplified. □

Checking for monotonicity was postponed:

Proposition 16 *For all $e \in \mathbb{E}$, $\mathbf{T}^{Co}[\![e]\!]$ is monotone: $\forall H_1, H_2 \in \mathbb{H}^{Co}$: $H_1 \dot{\leq}^{Co} H_2 \Longrightarrow \mathbf{T}^{Co}[\![e]\!]H_1 \leq^{Co} \mathbf{T}^{Co}[\![e]\!]H_2$.*

This **Prop. 16** shows, in retrospect, that typings could have restricted to monotone ones ($A \vdash\!\mathrm{m}\!\rightarrow B$ is the set of monotone maps of A into B):

$$\theta \in \mathbb{T}^{Co} \triangleq \mathbb{H}^{Co} \vdash\!\mathrm{m}\!\rightarrow \mathbb{P}^{Co}/_{\equiv^{Co}} \qquad \text{typing}$$

Finally it remains to turn the type collecting semantics $\mathbf{T}^{Co}[\![\bullet]\!]$ into a sound prescriptive type system:

[9] Monotonicity of $\mathbf{T}^{Co}[\![\bullet]\!]$ is proved in later **Prop. 16**.

Corollary 17 *Let* $\mathcal{E}^{\text{Co}} \triangleq \Lambda\theta \in \mathbb{T}^{\text{Co}} \cdot \bigcup\{\gamma_2^{\text{Co}}(H) \mid H \in \mathbb{H}^{\text{Co}} \wedge \theta(H) \neq \emptyset^{\text{Co}}\}$*. Then* $\langle \mathbb{T}^{\text{Co}},\ \ddot{\leq}^{\text{Co}},\ \mathbf{T}^{\text{Co}}\llbracket\bullet\rrbracket,\ \gamma^{\text{Co}},\ \mathcal{E}^{\text{Co}}\rangle$ *is a sound type system.*

A different type collecting semantics is given by the "general types" of [32, 33, 35, 36]. This collecting semantics is based on a big-step operational semantics so that it is more abstract with respect to nontermination (which has to be handled rather indirectly). In other aspects it is much more refined since the operational semantics uses explicit closures (which have to be abstracted to (recursive) functions). Moreover non-deterministic type choices $t_2 \oplus t_3$ to handle conditionals $(e_1\ \mathbf{?}\ e_2 : e_3)$ are more precise than $t_2 \vee^{\text{Co}} t_3$ since e.g. $\texttt{int} \oplus \texttt{int->int} \neq \emptyset^{\text{Co}}$. However the type collecting semantics $\mathbf{T}^{\text{Co}}\llbracket\bullet\rrbracket$ is simpler and refined enough to be instanciable into recursive types [3], conjunctive types [5] and Milner's polymorphic type schemes [17, 29, 37].

15 Fixpoint Approximation

Let us recall the following fixpoint approximation theorem from [9, 10]:

Proposition 18 *If* $\langle L,\ \sqsubseteq,\ \bot,\ \top,\ \sqcup,\ \sqcap\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L^\sharp,\ \sqsubseteq^\sharp,\ \bot^\sharp,\ \top^\sharp, \sqcup^\sharp, \sqcap^\sharp\rangle$*,* $F \in L \overset{\text{m}}{\longmapsto} L$*,* $a \sqsubseteq F(a)$*,* $F^\sharp \in L^\sharp \overset{\text{m}}{\longmapsto} L^\sharp$ *and* $\alpha \circ F \sqsubseteq^\sharp F^\sharp \circ \alpha$ *then* $\alpha\left(\mathbf{lfp}_a^{\sqsubseteq} F\right) \sqsubseteq^\sharp \mathbf{lfp}_{\alpha(a)}^{\sqsubseteq^\sharp} F^\sharp$*. If moreover* $\alpha(a) \sqsubseteq^\sharp a^\sharp \sqsubseteq^\sharp F^\sharp(a^\sharp)$ *then* $\alpha\left(\mathbf{lfp}_a^{\sqsubseteq} F\right) \sqsubseteq^\sharp \mathbf{lfp}_{a^\sharp}^{\sqsubseteq^\sharp} F^\sharp$*.*

This proposition provides a method for upper-approximation of concrete fixpoints by abstract fixpoints. In particular $F^\sharp$ can be chosen as $\alpha \circ F \circ \gamma$ since:

Proposition 19 *If* $\langle L,\ \sqsubseteq,\ \bot,\ \top,\ \sqcup,\ \sqcap\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L^\sharp,\ \sqsubseteq^\sharp,\ \bot^\sharp,\ \top^\sharp,\ \sqcup^\sharp,\ \sqcap^\sharp\rangle$*,* $F \in L \overset{\text{m}}{\longmapsto} L$*,* $F^\sharp \in L^\sharp \overset{\text{m}}{\longmapsto} L^\sharp$ *then* $\alpha \circ F \sqsubseteq^\sharp F^\sharp \circ \alpha$ *if and only if* $\alpha \circ F \circ \gamma \sqsubseteq^\sharp F^\sharp$*.*

16 Design of the Church/Curry Polytype Semantics by Abstraction of the Type Collecting Semantics

In order to simplify the presentation, the soundness proof of the Church/Curry polytype semantics has been left pending. To do so, $\langle \mathbb{T}^{\text{PC}},\ \supseteq,\ \mathbf{T}^{\text{PC}}\llbracket\bullet\rrbracket\rangle$ must be proved to be an abstraction of the type collecting semantics $\langle \mathbb{T}^{\text{Co}},\ \leq^{\text{Co}},\ \mathbf{T}^{\text{Co}}\llbracket\bullet\rrbracket\rangle$.

$$
\begin{aligned}
\gamma_1^{\text{Po}} &\in \mathbb{M}^{\text{PC}} \mapsto \mathbb{P}^{\text{Co}}/_{\equiv^{\text{Co}}} \\
\gamma_1^{\text{Po}}(\texttt{int}) &\triangleq \texttt{int} \\
\gamma_1^{\text{Po}}(m_1 \texttt{->} m_2) &\triangleq \{\gamma_1^{\text{Po}}(m_1) \texttt{->} \gamma_1^{\text{Po}}(m_2)\} \\
\gamma_2^{\text{Po}} &\in \mathbb{P}^{\text{PC}} \mapsto \mathbb{P}^{\text{Co}}/_{\equiv^{\text{Co}}} \\
\gamma_2^{\text{Po}}(p) &\triangleq \bigwedge_{m \in p}^{\text{Co}} \gamma_1^{\text{Po}}(m) \\
\gamma_3^{\text{Po}} &\in \mathbb{H}^{\text{PC}} \mapsto \mathbb{H}^{\text{Co}} \\
\gamma_3^{\text{Po}}(H) &\triangleq \Lambda \mathbf{x} \in \mathbb{X} \cdot \gamma_2^{\text{Po}}(H(\mathbf{x})) \\
\gamma_4^{\text{Po}} &\in \mathbb{I}^{\text{PC}} \mapsto \mathbb{P} \\
\gamma_4^{\text{Po}}(\langle H,\ m\rangle) &\triangleq \Lambda A \in \mathbb{H}^{\text{Co}} \cdot \bigwedge^{\text{Co}}\{\gamma_1^{\text{Po}}(m) \mid A \leq^{\text{Co}} \gamma_3^{\text{Po}}(H)\} \\
\gamma^{\text{Po}} &\in \mathbb{T}^{\text{PC}} \mapsto \mathbb{P} \\
\gamma^{\text{Po}}(T) &\triangleq \bigwedge^{\text{Co}}\{\gamma_4^{\text{Po}}(\langle H,\ m\rangle) \mid \langle H,\ m\rangle \in T\}
\end{aligned}
$$

There is no subtyping for monotypes:

Lemma 20 $\forall m_1, m_2 \in \mathbb{M}^{\text{PC}}$*:* $\gamma_1^{\text{Po}}(m_1) \leq^{\text{Co}} \gamma_1^{\text{Po}}(m_2) \iff m_1 = m_2$*.*

For polytypes, the Galois connection:

$$
\langle \mathbb{P}^{\text{Co}}/_{\equiv^{\text{Co}}},\ \leq^{\text{Co}},\ \bot^{\text{Co}},\ \emptyset^{\text{Co}},\ \bigvee^{\text{Co}},\ \bigwedge^{\text{Co}}\rangle \underset{\alpha_2^{\text{Po}}}{\overset{\gamma_2^{\text{Po}}}{\leftrightarrows}} \langle \mathbb{P}^{\text{PC}},\ \supseteq,\ \mathbb{M}^{\text{PC}},\ \emptyset,\ \cap,\ \cup\rangle
$$

can be extended pointwise to environments and typings, as was the case for the type collecting semantics. This leads to the abstract typing domain $(\mathbb{X} \mapsto \mathbb{P}^{\text{PC}}) \overset{\text{m}}{\longmapsto} \mathbb{P}^{\text{PC}}$. However to follow the tradition established by [29] of restricting polytypes to environments[10], one can chose to define $\mathbb{T}^{\text{PC}} \triangleq \wp((\mathbb{X} \mapsto \mathbb{P}^{\text{PC}}) \times \mathbb{M}^{\text{PC}})$, see (1). The corresponding Galois connection is:

$$
\langle \mathbb{T}^{\text{Co}},\ \ddot{\leq}^{\text{Co}},\ \ddot{\bot}^{\text{Co}},\ \ddot{\emptyset}^{\text{Co}},\ \ddot{\bigvee}^{\text{Co}},\ \ddot{\bigwedge}^{\text{Co}}\rangle \underset{\alpha^{\text{Po}}}{\overset{\gamma^{\text{Po}}}{\leftrightarrows}} \langle \mathbb{T}^{\text{PC}},\ \supseteq,\ \mathbb{H}^{\text{PC}} \times \mathbb{M}^{\text{PC}},\ \emptyset,\ \cap,\ \cup\rangle
$$

where:

Lemma 21
$$
\begin{aligned}
\alpha^{\text{Po}}(\theta) &= \{\langle H,\ m\rangle \mid \theta(\gamma_3^{\text{Po}}(H)) \leq^{\text{Co}} \gamma_2^{\text{Po}}(\{m\})\} \\
&= \{\langle H,\ m\rangle \mid m \in \alpha_2^{\text{Po}} \circ \theta \circ \gamma_3^{\text{Po}}(H)\}
\end{aligned}
$$

$\mathbf{T}^{\text{PC}}\llbracket e\rrbracket$ is monotone in the sense that ($H \mathrel{\dot{\subseteq}} H'$ if and only if $\forall \mathbf{x} \in \mathbb{X} : H(\mathbf{x}) \subseteq H'(\mathbf{x})$):

Lemma 22 *If* $\langle H,\ m\rangle \in \mathbf{T}^{\text{PC}}\llbracket e\rrbracket$ *and* $H \mathrel{\dot{\subseteq}} H'$ *then* $\langle H',\ m\rangle \in \mathbf{T}^{\text{PC}}\llbracket e\rrbracket$*.*

Corollary 23 *If* $p_1 \subseteq p_2 \in \mathbb{P}^{\text{PC}}$ *and* $\langle H[\mathbf{x} \leftarrow p_1], m\rangle \in \mathbf{T}^{\text{PC}}\llbracket e\rrbracket$ *then* $\langle H[\mathbf{x} \leftarrow p_2],\ m\rangle \in \mathbf{T}^{\text{PC}}\llbracket e\rrbracket$*.*

This corollary corresponds to lemma 1 of [17]. Soundness of Church/Curry polytype semantics follows from:

Proposition 24 $\forall e \in \mathbb{E}$*:* $\mathbf{T}^{\text{Co}}\llbracket e\rrbracket \ \ddot{\leq}^{\text{Co}}\ \gamma^{\text{Po}}(\mathbf{T}^{\text{PC}}\llbracket e\rrbracket)$

Proof The proof is by structural induction on e. Only a few typical cases are considered.

- $\alpha^{\text{Po}}(\mathbf{T}^{\text{Co}}\llbracket\mathbf{x}\rrbracket)$

$$
\begin{aligned}
&= \alpha^{\text{Po}}(\Lambda H \cdot H(\mathbf{x})) && \text{by def. } \mathbf{T}^{\text{Co}}\llbracket\mathbf{x}\rrbracket \\
&= \{\langle H,\ m\rangle \mid \gamma_3^{\text{Po}}(H)(\mathbf{x}) \leq^{\text{Co}} \gamma_1^{\text{Po}}(m)\} && \text{by } \textbf{Lem. 21} \\
&= \{\langle H,\ m\rangle \mid \gamma_2^{\text{Po}}(H(\mathbf{x})) \leq^{\text{Co}} \gamma_1^{\text{Po}}(m)\} && \text{by def. } \gamma_3^{\text{Po}} \\
&= \{\langle H,\ m\rangle \mid \Big(\bigwedge_{m' \in H(\mathbf{x})}^{\text{Co}} \gamma_1^{\text{Po}}(m')\Big) \leq^{\text{Co}} \gamma_1^{\text{Po}}(m)\} && \text{by def. } \gamma_2^{\text{Po}} \\
&\supseteq \{\langle H,\ m\rangle \mid \exists m' \in H(\mathbf{x}) : \gamma_1^{\text{Po}}(m') \leq^{\text{Co}} \gamma_1^{\text{Po}}(m)\} && \text{by def. glbs} \\
&= \{\langle H,\ m\rangle \mid \exists m' \in H(\mathbf{x}) : m' = m\} && \text{by } \textbf{Lem. 20} \\
&= \{\langle H,\ m\rangle \mid m \in H(\mathbf{x})\} && \text{by def. } = \\
&\triangleq \mathbf{T}^{\text{PC}}\llbracket\mathbf{x}\rrbracket
\end{aligned}
$$

- $\alpha^{\text{Po}}(\mathbf{T}^{\text{Co}}\llbracket\mu\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e\rrbracket)$

$$
\begin{aligned}
&= \alpha^{\text{Po}}(\Lambda H \cdot \mathbf{lfp}_{\{\emptyset^{\text{Co}} \texttt{->} \bot^{\text{Co}}\}}^{\leq^{\text{Co}}}\ \Lambda t \in \mathbb{P}^{\text{Co}}/_{\equiv^{\text{Co}}} \cdot \mathbf{T}^{\text{Co}}\llbracket\boldsymbol{\lambda}\mathbf{x} \cdot e\rrbracket H[\mathbf{f} \leftarrow t]) \\
& \qquad\qquad \text{by def. } \mathbf{T}^{\text{Co}}\llbracket\mu\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e\rrbracket \\
&= \{\langle H,\ m\rangle \mid \Big(\mathbf{lfp}_{\{\emptyset^{\text{Co}} \texttt{->} \bot^{\text{Co}}\}}^{\leq^{\text{Co}}}\ \Phi_H^{\text{Co}}\Big) \leq^{\text{Co}} \gamma_2^{\text{Po}}(\{m\})\} \text{ where } \Phi_H^{\text{Co}} \\
&\quad \triangleq \Lambda t \in \mathbb{P}^{\text{Co}}/_{\equiv^{\text{Co}}} \cdot \mathbf{T}^{\text{Co}}\llbracket\boldsymbol{\lambda}\mathbf{x} \cdot e\rrbracket \gamma_3^{\text{Po}}(H)[\mathbf{f} \leftarrow t] && \text{by } \textbf{Lem. 21}
\end{aligned}
$$

[10] For example Milner's algorithm W returns a monotype, which can be "manually" transformed into a polytype by adding a quantifier.

$= \{\langle H,\ m\rangle \mid \alpha_2^{\text{Po}}((\mathbf{lfp}^{\leq^{\text{Co}}}_{\{\emptyset^{\text{Co}} \to \perp^{\text{Co}}\}}\ \Phi_H^{\text{Co}}) \supseteq \{m\}\}$
since $\langle\alpha_2^{\text{Po}},\ \gamma_2^{\text{Po}}\rangle$ is a Galois connection

$\supseteq \{\langle H,\ m\rangle \mid m \in \mathbf{lfp}^{\supseteq}_{\mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}}\ \Phi_H^{\text{PC}}\}$ by **Prop. 18**

since $\alpha_2^{\text{Po}}(\{\emptyset^{\text{Co}} \to \perp^{\text{Co}}\}) \supseteq \{m_1 \to m_2 \mid m_1, m_2 \in \mathbb{M}^{\text{PC}}\} \triangleq \mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}$ and by defining Φ_H^{PC} such that $\alpha_2^{\text{Po}} \circ \Phi_H^{\text{Co}} \supseteq \Phi_H^{\text{PC}} \circ \alpha_2^{\text{Po}}$ and $\Phi_H^{\text{PC}}(\mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}) \subseteq \mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}$

$= \{\langle H,\ m\rangle \mid m \in \mathbf{gfp}^{\subseteq}_{\mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}}\ \Phi_H^{\text{PC}}\}$ by duality

$\triangleq \mathbf{T}^{\text{PC}}[\![\boldsymbol{\mu}\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e]\!]$

It remains to design Φ_H^{PC} such that:

$\alpha_2^{\text{Po}} \circ \Phi_H^{\text{Co}}(t)$

$= \alpha_2^{\text{Po}}(\mathbf{T}^{\text{Co}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]\gamma_3^{\text{Po}}(H)[\mathtt{f} \leftarrow t])$ by def. of Φ_H^{Co}

$\supseteq \alpha_2^{\text{Po}} \circ \mathbf{T}^{\text{Co}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!] \circ \gamma_3^{\text{Po}}(H[\mathtt{f} \leftarrow \alpha_2^{\text{Po}}(t)])$
by monotony of $\alpha_2^{\text{Po}}(t)$ and $\mathbf{T}^{\text{Co}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]$ since $\gamma_2^{\text{Po}} \circ \alpha_2^{\text{Po}} \geq^{\text{Co}} t$ so that $\gamma_3^{\text{Po}}(H)[\mathtt{f} \leftarrow t] \mathrel{\dot{\leq}^{\text{Co}}} \gamma_3^{\text{Po}}(H)[\mathtt{f} \leftarrow \gamma_2^{\text{Po}} \circ \alpha_2^{\text{Po}}(t)] = \gamma_3^{\text{Po}}(H[\mathtt{f} \leftarrow \alpha_2^{\text{Po}}(t)])$

$= \{m \mid \langle H[\mathtt{f} \leftarrow \alpha_2^{\text{Po}}(t)],\ m\rangle \in \{\langle H',\ m\rangle \mid m \in \alpha_2^{\text{Po}} \circ \mathbf{T}^{\text{Co}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!] \circ \gamma_3^{\text{Po}}(H')\}\}$ by def. $\in$

$= \{m \mid \langle H[\mathtt{f} \leftarrow \alpha_2^{\text{Po}}(t)],\ m\rangle \in \alpha^{\text{Po}}(\mathbf{T}^{\text{Co}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!])\}$ by **Lem. 21**

$= \{m \mid \langle H[\mathtt{f} \leftarrow \alpha_2^{\text{Po}}(t)],\ m\rangle \in \mathbf{T}^{\text{PC}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]\}$ by ind. hyp.

$= \Phi_H^{\text{PC}} \circ \alpha_2^{\text{Po}}(t)$

by defining $\Phi_H^{\text{PC}} \triangleq \Lambda p \cdot \{m \mid \langle H[\mathtt{f} \leftarrow p],\ m\rangle \in \mathbf{T}^{\text{PC}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]\}$

Obviously $\Phi_H^{\text{PC}}(\mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}) \subseteq \mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}$ by definition of $\mathbf{T}^{\text{PC}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]$.

- Finally, let us observe that by Tarski's fixpoint theorem $\mathbf{gfp}^{\subseteq}_{\mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}}\ \Phi_H^{\text{PC}} = \bigcup\{p \subseteq \mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}} \mid p \subseteq \Phi_H^{\text{PC}}(p)\}$. It follows that $\{m\} \subseteq \mathbf{gfp}^{\subseteq}_{\mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}}\ \Phi_H^{\text{PC}}$ if and only if $\exists p \subseteq \mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}$: $\{m\} \subseteq p \wedge p \subseteq \Phi_H^{\text{PC}}(p)$. Therefore, one has:

$\mathbf{T}^{\text{PC}}[\![\boldsymbol{\mu}\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e]\!]$

$= \{\langle H,\ m\rangle \mid m \in \mathbf{gfp}^{\subseteq}_{\mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}}\ \Phi_H^{\text{PC}}\}$

$= \{\langle H,\ m\rangle \mid \exists p \subseteq \mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}} : \{m\} \subseteq p \wedge p \subseteq \{m' \mid \langle H[\mathtt{f} \leftarrow p],\ m'\rangle \in \mathbf{T}^{\text{PC}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]\}\}$

$= \{\langle H, m\rangle \mid \exists p \subseteq \mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}} : m \in p \wedge \forall m' \in p : \langle H[\mathtt{f} \leftarrow p], m'\rangle \in \mathbf{T}^{\text{PC}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]\}$ □

The type semantics $\mathbf{T}^{\text{PC}}[\![\bullet]\!]$ leads to a polymorphic typing rule (2) for recursive definitions $\boldsymbol{\mu}\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e$, as is the case of [37] for Milner's [29] polymorphic type schemes. A $\supseteq$-upper approximation $\mathbf{T}^{\text{MC}}[\![\bullet]\!]$ à la Milner [29] with $\mathbf{T}^{\text{MC}}[\![e]\!]$ can be defined as $\mathbf{T}^{\text{PC}}[\![e]\!]$ but for the monomorphic typing of recursive definitions $\boldsymbol{\mu}\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e$:

$$\mathbf{T}^{\text{MC}}[\![\boldsymbol{\mu}\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e]\!] \triangleq \{\langle H,\ m_1 \to m_2\rangle \mid \langle H[\mathtt{f} \leftarrow \{m_1 \to m_2\}],\ m_1 \to m_2\rangle \in \mathbf{T}^{\text{MC}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]\}$$

This leads to the typing rule:

$$\frac{H[\mathtt{f} \leftarrow \{m_1 \to m_2\}] \stackrel{\text{MC}}{\vdash} \boldsymbol{\lambda}\mathbf{x} \cdot e \Rightarrow m_1 \to m_2}{H \stackrel{\text{MC}}{\vdash} \boldsymbol{\mu}\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e \Rightarrow m_1 \to m_2}$$

Soundness is obvious:

Proposition 25 $\forall e \in \mathbb{E}$: $\mathbf{T}^{\text{Co}}[\![e]\!] \mathrel{\ddot{\leq}^{\text{Co}}} \gamma^{\text{Po}}(\mathbf{T}^{\text{MC}}[\![e]\!])$

Proof

- $\alpha^{\text{Po}}(\mathbf{T}^{\text{Co}}[\![\boldsymbol{\mu}\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e]\!])$

$\supseteq \{\langle H, m\rangle \mid \exists p \subseteq \mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}} : m \in p \wedge \forall m' \in p : \langle H[\mathtt{f} \leftarrow p], m'\rangle \in \mathbf{T}^{\text{MC}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]\}$ as proved for $\mathbf{T}^{\text{PC}}[\![\boldsymbol{\mu}\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e]\!]$

$\supseteq \{\langle H,\ m\rangle \mid m \in \{m_1 \to m_2\} \wedge \forall m' \in \{m_1 \to m_2\} : \langle H[\mathtt{f} \leftarrow \{m_1 \to m_2\}],\ m'\rangle \in \mathbf{T}^{\text{MC}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]\}$ by restricting the choice of p to $\{m_1 \to m_2\} \subseteq \mathbb{M}^{\text{PC}} \to \mathbb{M}^{\text{PC}}$

$\supseteq \{\langle H,\ m_1 \to m_2\rangle \mid \langle H[\mathtt{f} \leftarrow \{m_1 \to m_2\}],\ m_1 \to m_2\rangle \in \mathbf{T}^{\text{MC}}[\![\boldsymbol{\lambda}\mathbf{x} \cdot e]\!]\}$ by def. $\in$

$\triangleq \mathbf{T}^{\text{MC}}[\![\boldsymbol{\mu}\mathbf{f} \cdot \boldsymbol{\lambda}\mathbf{x} \cdot e]\!]$ □

17 À la Damas-Milner-Mycroft Semantics

If Hindley semantics $\mathbf{T}^{\text{H}}[\![\bullet]\!]$ is an exact abstraction of Curry/-Church monotype semantics $\mathbf{T}^{\text{C}}[\![\bullet]\!]$ by Herbrand abstraction, it is clear that this same abstraction is <u>not</u> exact for Curry/Church polytype semantics $\mathbf{T}^{\text{PC}}[\![\bullet]\!]$. The argument is similar to that of [23] showing that Milner's polymorphic type schemes do not have the principal typing property. One can imagine objects `x` of polytype $\{\mathtt{int}, \mathtt{int} \to \mathtt{int}\}$. For example `x` might be an integer which is $\Lambda \mathtt{u} \cdot \mathtt{x}$ when used as a function or a function of type $\mathtt{int} \to \mathtt{int}$ which is $\mathtt{x}(0)$ (or better $\mathtt{x}(\perp)$ with call-by-name) when used as an integer. The meaning of $[\mathtt{x} : \{\mathtt{int}, \mathtt{int} \to \mathtt{int}\}] \stackrel{\text{PC}}{\vdash} \mathtt{x}(\mathtt{x}) \Rightarrow \mathtt{int}$ cannot be the same as its Herbrand abstraction $[\mathtt{x} : [\alpha]_{\equiv}] \stackrel{\text{H}'}{\vdash} \mathtt{x}(\mathtt{x}) \Rightarrow \mathtt{int}$. This problem could be solved by considering a refinement $\mathbf{T}^{\text{H}'}[\![\bullet]\!]$ of $\mathbf{T}^{\text{H}}[\![\bullet]\!]$ involving a disjunctive completion [13] allowing for judgments of the form $[\mathtt{x} : \{\mathtt{int}, \mathtt{int} \to \mathtt{int}\}] \stackrel{\text{H}'}{\vdash} \mathtt{x}(\mathtt{x}) \Rightarrow \mathtt{int}$ or $[\mathtt{x} : \{\mathtt{int}, \forall \mathtt{'a}. \mathtt{'a} \to \mathtt{'a}\}] \stackrel{\text{H}'}{\vdash} \mathtt{x}(\mathtt{x}) \Rightarrow \mathtt{int}$. The classical solution given by [29], can be understood as consisting in considering a further abstraction $\mathbf{T}^{\text{M}}[\![\bullet]\!]$ of $\mathbf{T}^{\text{PC}}[\![\bullet]\!]$ ruling out such undesirable environments. The abstract domains is now defined together with the corresponding abstractions.

Closed monotypes are similar to Church simple types:

$\mu \in \mathbb{M}_{\text{c}}^{\text{DM}}$		closed monotype
$\mu ::= \mathtt{int} \mid \mu_1 \to \mu_2$		

The set $\mathbb{M}_{\text{v}}^{\text{DM}}$ of *monotypes with variables* à la Hindley is preordered by the instance relation $\leq_{\text{v}}^{\text{DM}} \triangleq \leq_{\mathbb{V}}$ with corresponding equivalence $\equiv_{\text{v}}^{\text{DM}}$:

$\mathtt{'a} \in \mathbb{V}$	type variable[11]
$\tau \in \mathbb{M}_{\text{v}}^{\text{DM}}$	monotype with variables
$\tau ::= \mathtt{int} \mid \mathtt{'a} \mid \tau_1 \to \tau_2$	

Let $\mathbf{ftv}(t)$ be the set of free type variables of t. To provide a context-free syntax of type schemes, parametric/free variables (like `'a` in $\forall \mathtt{'b}. \mathtt{'b} \to \mathtt{'a}$) are differentiated from generic/bound type variables (like `'b` in $\forall \mathtt{'b}. \mathtt{'b} \to \mathtt{'a}$):

$\mathtt{'a} \in \mathbb{V}_{\text{p}}$	parametric/free type variables
$\mathtt{'b} \in \mathbb{V}_{\text{g}}$	generic/bound type variables
$\mathtt{'a}, \mathtt{'b} \in \mathbb{V} \triangleq \mathbb{V}_{\text{g}} \cup \mathbb{V}_{\text{p}}$	type variable ($\mathbb{V}_{\text{g}} \cap \mathbb{V}_{\text{p}} = \emptyset$)

[11] Type variables `'a`, `'c` are used instead of the traditional α, γ to avoid confusion with the abstraction/concretization functions.

Generic polytypes are type schemes without free variable (like `∀'b.'b->int`):

$$\begin{aligned} &\varsigma \;\in\; \mathbb{P}^{\text{DM}}_{\text{g}} && \text{closed/generic polytype} \\ &\varsigma \;::=\; \mu \mid \forall\,\text{'b}_1 \ldots \text{'b}_\ell.\tau \;\text{ where } \{\text{'b}_1, \ldots, \text{'b}_\ell\} = \mathbf{ftv}(\tau) \subseteq \mathbb{V}_{\text{g}} \end{aligned}$$

$\mathbb{P}^{\text{DM}}_{\text{g}}$ is preordered by the instance relation $\le^{\text{DM}}_{\text{g}}$ with corresponding renaming equivalence $\equiv^{\text{DM}}_{\text{g}}$ (e.g. `∀'b.'b->int` $\equiv^{\text{DM}}_{\text{g}}$ `∀'c.'c->int`):

$$\forall\,\text{'b}_1 \ldots \text{'b}_\ell.\tau \le^{\text{DM}}_{\text{g}} \forall\,\text{'b}'_1 \ldots \text{'b}'_m.\tau' \;\triangleq\; \tau \le^{\text{DM}}_{\text{v}} \tau'$$

$\mathbb{P}^{\text{DM}^{\emptyset}}_{\text{g}} /_{\equiv^{\text{DM}}_{\text{g}}}$ is $\mathbb{P}^{\text{DM}}_{\text{g}}$ up to renaming of generic/bound variables $\text{'b} \in \mathbb{V}_{\text{g}}$ and introduction of an empty generic polytype $\emptyset^{\text{DM}}_{\text{g}}$. There is an isomorphism $\mathbb{M}^{\text{DM}}_{\text{v}} \underset{\mathbf{gen}}{\overset{\mathbf{elim}}{\leftrightarrows}} \mathbb{P}^{\text{DM}}_{\text{g}}$ between Hindley's monotypes with variables and generic polytypes given by:

$$\begin{aligned} &\mathbf{elim}(\mu) \;\triangleq\; \mu \\ &\mathbf{elim}(\forall\,\text{'b}_1 \ldots \text{'b}_\ell.\tau) \;\triangleq\; \tau \\ &\mathbf{gen}(\tau) \;\triangleq\; \forall\,\text{'b}_1 \ldots \text{'b}_\ell.\tau \quad \text{where } \{\text{'b}_1, \ldots, \text{'b}_\ell\} = \mathbf{ftv}(\tau) \end{aligned}$$

It follows, by composition of Galois insertions, that: $\langle \mathbb{P}^{\text{PC}}, \subseteq\rangle \underset{\alpha^{\text{DM}}_{\text{g}}}{\overset{\gamma^{\text{DM}}_{\text{g}}}{\leftrightarrows}} \langle \mathbb{P}^{\text{DM}^{\emptyset}}_{\text{g}} /_{\equiv^{\text{DM}}_{\text{g}}}, \le^{\text{DM}}_{\text{g}}\rangle$ where:

$$\alpha^{\text{DM}}_{\text{g}} \;\triangleq\; \mathbf{gen} \circ \mathbf{lcg} \qquad \gamma^{\text{DM}}_{\text{g}} \;\triangleq\; \mathbf{ground} \circ \mathbf{elim}$$

is a Galois insertion between à la Church/Curry polytypes and generic polytypes.

Parametric polytypes are type schemes with parametric/free type variables (like `∀'b.'b->'a`):

$$\begin{aligned} &\pi \;\in\; \mathbb{P}^{\text{DM}}_{\text{p}} && \text{parametric polytype} \\ &\pi \;::=\; \tau \mid \forall\,\text{'b}_1 \ldots \text{'b}_\ell.\tau \;\text{ where } \{\text{'b}_1, \ldots, \text{'b}_\ell\} = \mathbf{ftv}(\tau) \cap \mathbb{V}_{\text{g}} \end{aligned}$$

$\mathbb{P}^{\text{DM}}_{\text{p}}$ is preordered by the instance relation $\le^{\text{DM}}_{\text{p}}$ with corresponding equivalence $\equiv^{\text{DM}}_{\text{p}}$ (e.g. `∀'b.'b->'a` $\equiv^{\text{DM}}_{\text{p}}$ `∀'c.'c->'a` but `∀'b.'b->'a` $\not\equiv^{\text{DM}}_{\text{p}}$ `∀'c.'c->'d`):

$$\pi \le^{\text{DM}}_{\text{p}} \pi' \;\triangleq\; \forall \sigma \in \mathbb{V}_{\text{p}} \hookrightarrow \mathbb{M}^{\text{DM}}_{\text{c}} : \sigma(\pi) \le^{\text{DM}}_{\text{g}} \sigma(\pi')$$

$\mathbb{P}^{\text{DM}^{\emptyset}}_{\text{p}} /_{\equiv^{\text{DM}}_{\text{p}}}$ is $\mathbb{P}^{\text{DM}}_{\text{p}}$ up to renaming of generic/bound variables $\text{'b} \in \mathbb{V}_{\text{g}}$ and introduction of an empty polytype $\emptyset^{\text{DM}}_{\text{p}}$.

Generic type environments map program variables to generic polytypes:

$$H \;\in\; \mathbb{H}^{\text{DM}}_{\text{g}} \;\triangleq\; \mathbb{X} \mapsto \mathbb{P}^{\text{DM}^{\emptyset}}_{\text{g}} /_{\equiv^{\text{DM}}_{\text{g}}} \quad \text{generic type environment}$$

The preordering is pointwise $H \mathrel{\dot{\le}}^{\text{DM}}_{\text{g}} H \triangleq \forall \mathtt{x} \in \mathbb{X} : H(\mathtt{x}) \le^{\text{DM}}_{\text{g}} H(\mathtt{x})$ with corresponding renaming equivalence $\dot{\equiv}^{\text{DM}}_{\text{g}}$. The correspondence between à la Church/Curry polytypes and generic polytypes is extended pointwise to type environments $\langle \mathbb{H}^{\text{PC}}, \dot{\subseteq}\rangle \underset{\dot{\alpha}^{\text{DM}}_{\text{g}}}{\overset{\dot{\gamma}^{\text{DM}}_{\text{g}}}{\leftrightarrows}} \langle \mathbb{H}^{\text{DM}}_{\text{g}}, \dot{\le}^{\text{DM}}_{\text{g}}\rangle$ so that $\dot{\gamma}^{\text{DM}}_{\text{g}}(H) \triangleq \Lambda \mathtt{x} \in \mathbb{X} \cdot \gamma^{\text{DM}}_{\text{g}}(H(\mathtt{x}))$.

Parametric type environments map program variables to parametric polytypes:

$$H \;\in\; \mathbb{H}^{\text{DM}}_{\text{p}} \;\triangleq\; \mathbb{X} \mapsto \mathbb{P}^{\text{DM}^{\emptyset}}_{\text{p}} /_{\equiv^{\text{DM}}_{\text{p}}} \quad \text{parametric type environment}$$

The preordering is pointwise $H \mathrel{\dot{\le}}^{\text{DM}}_{\text{p}} H \triangleq \forall \mathtt{x} \in \mathbb{X} : H(\mathtt{x}) \le^{\text{DM}}_{\text{p}} H(\mathtt{x}) \iff \forall \sigma \in \mathbb{V}_{\text{p}} \hookrightarrow \mathbb{M}^{\text{DM}}_{\text{c}} : \sigma(H) \mathrel{\dot{\le}}^{\text{DM}}_{\text{g}} \sigma(H)$ with corresponding renaming equivalence $\dot{\equiv}^{\text{DM}}_{\text{p}}$. $\mathbb{H}^{\text{DM}}_{\text{p}} /_{\dot{\equiv}^{\text{DM}}_{\text{p}}}$ is $\mathbb{H}^{\text{DM}}_{\text{p}}$ up to renaming of parametric/free variables $\text{'a} \in \mathbb{V}_{\text{p}}$. For example `[x : ∀'b.'b->'a; y : 'a]` $\dot{\equiv}^{\text{DM}}_{\text{p}}$ `[x : ∀'c.'c->'d; y : 'd]` since parametric type variables like 'a and 'd are externally or globally quantified, which is formalized by the injective glb-preserving concretization function $\dot{\gamma}^{\text{DM}}_{\text{p}} \in \mathbb{H}^{\text{DM}}_{\text{p}} \mapsto \wp(\mathbb{H}^{\text{PC}})$ defined by $\dot{\gamma}^{\text{DM}}_{\text{p}}(H) \triangleq \{\dot{\gamma}^{\text{DM}}_{\text{g}}(\sigma(H)) \mid \sigma \in \mathbb{V}_{\text{p}} \hookrightarrow \mathbb{M}^{\text{DM}}\}$ so that $\langle \wp(\mathbb{H}^{\text{PC}}), \subseteq\rangle \underset{\dot{\alpha}^{\text{DM}}_{\text{p}}}{\overset{\dot{\gamma}^{\text{DM}}_{\text{p}}}{\leftrightarrows}} \langle \mathbb{H}^{\text{DM}}_{\text{p}}, \dot{\le}^{\text{DM}}_{\text{p}}\rangle$ is a Galois insertion.

Since $\mathbf{T}^{\text{PC}}[\![\bullet]\!]$ has no principal typing for the Herbrand abstraction, one cannot look for an exact Herbrand abstraction of $\mathbf{T}^{\text{PC}}[\![\bullet]\!]$ (as was the case of $\mathbf{T}^{\text{H}}[\![\bullet]\!]$ for $\mathbf{T}^{\text{C}}[\![\bullet]\!]$) but instead for an Herbrand abstraction of a $\supseteq$-upper approximation $\mathbf{T}^{\text{M}}[\![\bullet]\!]$ of $\mathbf{T}^{\text{PC}}[\![\bullet]\!]$ defined by:

$$\begin{aligned} \mathbf{T}^{\text{M}}[\![e]\!] \;&\triangleq\; \alpha^{\text{M}}(\mathbf{T}^{\text{PC}}[\![e]\!]) \\ \alpha^{\text{M}}(T) \;&\triangleq\; \{\langle H,\, m\rangle \in T \mid H \in \dot{\gamma}^{\text{DM}}_{\text{g}}(\mathbb{H}^{\text{DM}}_{\text{g}})\} \end{aligned}$$

such that $\langle \mathbb{T}^{\text{PC}}, \supseteq\rangle \underset{\alpha^{\text{M}}}{\overset{\mathbf{Id}}{\leftrightarrows}} \langle \mathbb{T}^{\text{PC}}, \supseteq\rangle$. $\mathbf{T}^{\text{M}}[\![\bullet]\!]$ is obviously sound since it is an $\supseteq$-upper approximation of the sound abstract semantics $\mathbf{T}^{\text{PC}}[\![\bullet]\!]$.

Generic typings correspond to an attribute-independent abstraction:

$$T \;\in\; \mathbb{T}^{\text{MM}}_{\text{g}} \;\triangleq\; \mathbb{H}^{\text{DM}}_{\text{g}} \vdash\!\!\text{m}\!\!\rightarrow \mathbb{P}^{\text{DM}^{\emptyset}}_{\text{g}} /_{\equiv^{\text{DM}}_{\text{g}}} \quad \text{generic program typing}$$

This attribute-independent abstraction is the one considered by [38]. It is often used in extension of Damas-Milner polymorphic type schemes to logic programs [39]. Formally, this would be a restriction to [17, 29] since infinitely many generic typings are necessary to express the property specified by a single parametric typing.

Parametric typings correspond to a relational abstraction:

$$T \;\in\; \mathbb{T}^{\text{MM}}_{\text{p}} \;\triangleq\; \left(\mathbb{H}^{\text{DM}}_{\text{p}} \vdash\!\!\text{DM}\!\!\rightarrow \mathbb{P}^{\text{DM}^{\emptyset}}_{\text{p}} /_{\equiv^{\text{DM}}_{\text{p}}}\right) /_{\ddot{\equiv}^{\text{MM}}_{\text{p}}} \quad \text{parametric program typing}$$

$\mathbb{T}^{\text{MM}}_{\text{p}}$ is preordered by the pointwise instance relation $\ddot{\le}^{\text{MM}}_{\text{p}}$:

$$\begin{aligned} T_1 \mathrel{\ddot{\le}}^{\text{MM}}_{\text{p}} T_2 \quad &\triangleq\quad \forall H \in \mathbb{H}^{\text{DM}}_{\text{p}} : T_1(H) \le^{\text{DM}}_{\text{p}} T_2(H) \\ &\iff\quad \forall H \in \mathbb{H}^{\text{DM}}_{\text{g}} : T_1(H) \le^{\text{DM}}_{\text{g}} T_2(H) \end{aligned}$$

with corresponding equivalence $\ddot{\equiv}^{\text{MM}}_{\text{p}}$ which is used to define $\mathbb{T}^{\text{MM}}_{\text{p}}$ up to renaming of parametric/free variables $\text{'a} \in \mathbb{V}_{\text{p}}$. The notation $\vdash\!\!\text{DM}\!\!\rightarrow$ states that $T \in \mathbb{T}^{\text{MM}}_{\text{p}}$ satisfies:

$$\forall H_1, H_2 \in \mathbb{H}^{\text{DM}}_{\text{p}} : H_1 \mathrel{\dot{\le}}^{\text{DM}}_{\text{p}} H_2 \Longrightarrow T(H_1) \le^{\text{DM}}_{\text{p}} T(H_2) \quad (5)$$

$$\forall H \in \mathbb{H}^{\text{DM}}_{\text{p}} : \mathbf{ftv}(T(H)) \subseteq \mathbf{ftv}(H) \quad (6)$$

$$\forall H \in \mathbb{H}^{\text{DM}}_{\text{p}} : \forall \sigma \in \mathbb{V}_{\text{p}} \hookrightarrow \mathbb{M}^{\text{DM}}_{\text{c}} : T(\sigma(H)) = \sigma(T(H)) \quad (7)$$

The monotony condition (5) corresponds e.g. to **Prop. 16**. Condition (6) states that a free type variable not free in the type environment should be quantified[12]. Condition (7) states that free variables in H are globally quantified. By (6), it is equivalent to:

[12] This has to be done "manually" in Milner's algorithm W which returns a monotype, which must be quantified to be transformed into a polytype.

$$\forall H \in \mathbb{H}_{\mathrm{p}}^{\mathrm{DM}} : \forall \sigma \in \mathbb{V}_{\mathrm{p}} \hookrightarrow \mathbb{M}_{\mathrm{c}}^{\mathrm{DM}} : \sigma(T(\sigma(H))) = \sigma(T(H)) \quad (8)$$

The correspondence $\ddot{\alpha}_{\mathrm{p}}^{\mathrm{MM}} \in \mathbb{T}^{\mathrm{PC}} \mapsto \mathbb{T}_{\mathrm{p}}^{\mathrm{MM}}$ and $\ddot{\gamma}_{\mathrm{p}}^{\mathrm{MM}} \in \mathbb{T}_{\mathrm{p}}^{\mathrm{MM}} \mapsto \mathbb{T}^{\mathrm{PC}}$ is defined by ($\mathbf{gen}_H\, \tau \triangleq \forall\, '\mathtt{b}_1 \ldots '\mathtt{b}_\ell . \tau$ where $\{'\mathtt{b}_1, \ldots, '\mathtt{b}_\ell\} = \mathbf{ftv}(\tau) - \mathbf{ftv}(H)$[13]):

$$\ddot{\alpha}_{\mathrm{p}}^{\mathrm{MM}}(T) \triangleq \Lambda H \in \mathbb{H}_{\mathrm{p}}^{\mathrm{DM}} \cdot \mathbf{gen}_H \circ \mathbf{lcg}\{\tau \in \mathbb{M}_{\mathrm{v}}^{\mathrm{DM}} \mid \forall \sigma \in \mathbb{V} \hookrightarrow \mathbb{M}_{\mathrm{c}}^{\mathrm{DM}} : \langle \dot{\gamma}_{\mathrm{g}}^{\mathrm{DM}}(\sigma(H)),\, \sigma(\tau)\rangle \in T\}$$

$$\begin{aligned}\ddot{\gamma}_{\mathrm{p}}^{\mathrm{MM}}(T) &\triangleq \{\langle \dot{\gamma}_{\mathrm{g}}^{\mathrm{DM}}(H),\, \mu\rangle \mid H \in \mathbb{H}_{\mathrm{g}}^{\mathrm{DM}} \wedge \mu \in \gamma_{\mathrm{g}}^{\mathrm{DM}}(T(H))\} \\ &= \{\langle \dot{\gamma}_{\mathrm{g}}^{\mathrm{DM}}(\sigma(H)),\, \mu\rangle \mid H \in \mathbb{H}_{\mathrm{p}}^{\mathrm{DM}} \wedge \sigma \in \mathbb{V}_{\mathrm{p}} \hookrightarrow \mathbb{M}_{\mathrm{c}}^{\mathrm{DM}} \wedge \mu \in \gamma_{\mathrm{g}}^{\mathrm{DM}}(\sigma(T(H)))\}\end{aligned}$$

so that:

$$\langle \mathbb{T}^{\mathrm{PC}}, \subseteq, \emptyset, \mathbb{I}^{\mathrm{PC}}, \cup, \cap\rangle \underset{\ddot{\alpha}_{\mathrm{p}}^{\mathrm{MM}}}{\overset{\ddot{\gamma}_{\mathrm{p}}^{\mathrm{MM}}}{\leftrightarrows}} \langle \mathbb{T}_{\mathrm{p}}^{\mathrm{MM}}, \ddot{\preceq}_{\mathrm{p}}^{\mathrm{MM}}\, \Lambda H \cdot \emptyset_{\mathrm{p}}^{\mathrm{DM}},\, \Lambda H \cdot \forall\, '\mathtt{b}.\, '\mathtt{b},\, \ddot{\bigvee}_{\mathrm{p}}^{\mathrm{MM}},\, \ddot{\bigwedge}_{\mathrm{p}}^{\mathrm{MM}}\rangle$$

where:

$$\ddot{\bigvee_{i\in\Delta}}{}_{\mathrm{p}}^{\mathrm{MM}}\, T_i \triangleq \Lambda H \cdot \mathbf{gen}_H \circ \mathbf{lcg}\{\mathbf{elim}(T_i(H)) \mid i \in \Delta\}$$

$$\ddot{\bigwedge_{i\in\Delta}}{}_{\mathrm{p}}^{\mathrm{MM}}\, T_i \triangleq \Lambda H \cdot \mathbf{gen}_H \circ \mathbf{gci}\{\mathbf{elim}(T_i(H)) \mid i \in \Delta\}$$

is a Galois insertion.

The Milner-Mycroft type semantics can now be designed ($'\mathtt{a}$ is a fresh type variable):

$$\begin{aligned}
\mathbf{T}^{\mathrm{MM}}[\![\mathtt{x}]\!] &\triangleq \Lambda H \cdot H(\mathtt{x}) \\
\mathbf{T}^{\mathrm{MM}}[\![\mathtt{let}\ \mathtt{x} = e_1\ \mathtt{in}\ e_2]\!] &\triangleq \Lambda H \cdot \mathbf{T}^{\mathrm{MM}}[\![e_2]\!]H[\mathtt{x} \leftarrow \mathbf{T}^{\mathrm{MM}}[\![e_1]\!]H] \\
\mathbf{T}^{\mathrm{MM}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \Lambda H \cdot \mathbf{gen}_H \circ \mathbf{lcg}\{\tau_1 \to \mathbf{elim}(\pi_2) \mid \\
&\qquad \tau_1 \in \mathbb{M}_{\mathrm{v}}^{\mathrm{DM}} \wedge \\
&\qquad \pi_2 = \mathbf{T}^{\mathrm{MM}}[\![e]\!]H[\mathtt{x} \leftarrow \tau_1] \neq \emptyset_{\mathrm{p}}^{\mathrm{DM}}\} \\
\mathbf{T}^{\mathrm{MM}}[\![e_1(e_2)]\!] &\triangleq \Lambda H \cdot (\tau_2 = \mathbf{elim}(\mathbf{T}^{\mathrm{MM}}[\![e_2]\!]H) \wedge \\
&\qquad \tau_2 \to \tau = \mathbf{gci}\{\mathbf{elim}(\mathbf{T}^{\mathrm{MM}}[\![e_1]\!]H), \tau_2 \to {}'\mathtt{a}\}\ ? \\
&\qquad \mathbf{gen}_H(\tau) \mid \emptyset_{\mathrm{p}}^{\mathrm{DM}}) \\
\mathbf{T}^{\mathrm{MM}}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \Lambda H \cdot \mathbf{gfp}^{\preceq_{\mathrm{p}}^{\mathrm{DM}}}_{\forall '\mathtt{b}_1, '\mathtt{b}_2 . '\mathtt{b}_1 \to '\mathtt{b}_2}\ \Psi_H
\end{aligned}$$

where $\Psi_H \triangleq \Lambda \pi \in \mathbb{P}_{\mathrm{p}}^{\mathrm{DM}} \cdot \mathbf{T}^{\mathrm{MM}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]H[\mathtt{f} \leftarrow \mathbf{gen}_{H[\mathtt{f} \leftarrow \emptyset_{\mathrm{p}}^{\mathrm{DM}}]}(\pi)]$

$$\begin{aligned}
\mathbf{T}^{\mathrm{MM}}[\![\mathbf{1}]\!] &\triangleq \Lambda H \cdot \mathtt{int} \\
\mathbf{T}^{\mathrm{MM}}[\![e_1 - e_2]\!] &\triangleq \Lambda H \cdot \mathbf{gci}\{\mathtt{int}, \mathbf{elim}(\mathbf{T}^{\mathrm{MM}}[\![e_1]\!]H), \\
&\qquad \mathbf{elim}(\mathbf{T}^{\mathrm{MM}}[\![e_2]\!]H)\} \\
\mathbf{T}^{\mathrm{MM}}[\![(e_1\ \mathbf{?}\ e_2 \mathbf{:} e_3)]\!] &\triangleq \Lambda H \cdot (\mathtt{int} \leq \mathbf{elim}(\mathbf{T}^{\mathrm{MM}}[\![e_1]\!]H)\ ? \\
&\qquad \mathbf{gen}_H \circ \mathbf{gci}\{\mathbf{elim}(\mathbf{T}^{\mathrm{MM}}[\![e_2]\!]H), \\
&\qquad \mathbf{elim}(\mathbf{T}^{\mathrm{MM}}[\![e_3]\!]H)\} \mid \emptyset_{\mathrm{p}}^{\mathrm{DM}})
\end{aligned}$$

$\mathbf{T}^{\mathrm{MM}}[\![\bullet]\!]$ is sound by construction:

Proposition 26 *$\langle \mathbb{T}_{\mathrm{p}}^{\mathrm{MM}}, \ddot{\preceq}_{\mathrm{p}}^{\mathrm{MM}}, \mathbf{T}^{\mathrm{MM}}[\![\bullet]\!]\rangle$ is an abstraction of Milner's approximation $\langle \mathbb{T}^{\mathrm{PC}}, \supseteq, \mathbf{T}^{\mathrm{M}}[\![\bullet]\!]\rangle$ of Church/Curry polytype abstract semantics $\langle \mathbb{T}^{\mathrm{PC}}, \supseteq, \mathbf{T}^{\mathrm{PC}}[\![\bullet]\!]\rangle$ by the Galois insertion $\langle \mathbb{T}^{\mathrm{PC}}, \subseteq\rangle \underset{\ddot{\alpha}_{\mathrm{p}}^{\mathrm{MM}}}{\overset{\ddot{\gamma}_{\mathrm{p}}^{\mathrm{MM}}}{\leftrightarrows}} \langle \mathbb{T}_{\mathrm{p}}^{\mathrm{MM}}, \ddot{\preceq}_{\mathrm{p}}^{\mathrm{MM}}\rangle$ since for all $e \in \mathbb{E}$:*

$$\begin{aligned}
\mathbf{T}^{\mathrm{MM}}[\![e]\!] &\ddot{\succeq}_{\mathrm{p}}^{\mathrm{MM}} \ddot{\alpha}_{\mathrm{p}}^{\mathrm{MM}}(\mathbf{T}^{\mathrm{M}}[\![e]\!]) = \ddot{\alpha}_{\mathrm{p}}^{\mathrm{MM}}(\mathbf{T}^{\mathrm{PC}}[\![e]\!]) \\
\mathbf{T}^{\mathrm{M}}[\![e]\!] &\supseteq \ddot{\gamma}_{\mathrm{p}}^{\mathrm{MM}}(\mathbf{T}^{\mathrm{MM}}[\![e]\!])
\end{aligned}$$

As shown by **Ex. 1**, the abstraction is not exact since e.g. although $\mathbf{T}^{\mathrm{PC}}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!]$ may be expressible by a type of $\mathbb{T}_{\mathrm{p}}^{\mathrm{DM}}$, this may not be the case for the polytype of the recursive calls of $\mathtt{f}$ within e.

The Damas-Milner-Mycroft isomorphic semantics $\langle \mathbb{T}^{\mathrm{DM}^2}, \supseteq, \mathbf{T}^{\mathrm{DM}^2}[\![\bullet]\!]\rangle$ is useful for deriving typing rules:

$$\begin{aligned}
\mathbb{T}^{\mathrm{DM}^2} &\triangleq \wp\big((\mathbb{H}_{\mathrm{p}}^{\mathrm{DM}} \times \mathbb{M}_{\mathrm{v}}^{\mathrm{DM}})/_{\equiv_{\mathrm{p}}^{\mathrm{DM}}}\big) \\
\mathbf{T}^{\mathrm{DM}^2}[\![e]\!] &\triangleq \alpha_{\mathrm{p}}^{\mathrm{DM}}(\mathbf{T}^{\mathrm{MM}}[\![e]\!]) \\
\alpha_{\mathrm{p}}^{\mathrm{DM}}(T) &\triangleq \{\langle H,\, \tau\rangle \mid \tau \in \mathbf{inst} \circ \mathbf{elim}(T(H))\}
\end{aligned}$$

where:

$$\gamma_{\mathrm{p}}^{\mathrm{DM}}(\mathbf{T}^{\mathrm{DM}^2}[\![e]\!]) = \mathbf{T}^{\mathrm{MM}}[\![e]\!]$$

and:

$$\gamma_{\mathrm{p}}^{\mathrm{DM}}(T) \triangleq \Lambda H \in \mathbb{H}_{\mathrm{p}}^{\mathrm{DM}} \cdot \mathbf{gen}_H \circ \mathbf{lcg}(\{\tau \mid \langle H,\, \tau\rangle \in T\})$$

The Damas-Milner-Mycroft semantics $\mathbf{T}^{\mathrm{DM}^2}[\![\bullet]\!]$ is obtained as follows:

$$\begin{aligned}
\mathbf{T}^{\mathrm{DM}^2}[\![\mathtt{x}]\!] &\triangleq \{\langle H,\, \tau\rangle \mid \tau \preceq_{\mathrm{v}}^{\mathrm{DM}} \mathbf{elim}(H(\mathtt{x}))\} \\
\mathbf{T}^{\mathrm{DM}^2}[\![\mathtt{let}\ \mathtt{x} = e_1\ \mathtt{in}\ e_2]\!] &\triangleq \{\langle H,\, \tau_2\rangle \mid \langle H,\, \tau_1\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![e_1]\!] \wedge \\
&\qquad \langle H[\mathtt{x} \leftarrow \mathbf{gen}_H(\tau_1)],\, \tau_2\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![e_2]\!]\} \\
\mathbf{T}^{\mathrm{DM}^2}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \{\langle H,\, \tau_1 \to \tau_2\rangle \mid \tau_1 \in \mathbb{M}_{\mathrm{v}}^{\mathrm{DM}} \wedge \\
&\qquad \langle H[\mathtt{x} \leftarrow \tau_1],\, \tau_2\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![e]\!]\} \\
\mathbf{T}^{\mathrm{DM}^2}[\![e_1(e_2)]\!] &\triangleq \{\langle H,\, \tau\rangle \mid \langle H,\, \tau_2\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![e_2]\!] \wedge \\
&\qquad \langle H,\, \tau_2 \to \tau\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![e_1]\!]\} \\
\mathbf{T}^{\mathrm{DM}^2}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!] &\triangleq \{\langle H,\, \tau\rangle \mid \tau \preceq_{\mathrm{v}}^{\mathrm{DM}} \mathbf{gfp}^{\preceq_{\mathrm{v}}^{\mathrm{DM}}}_{['\mathtt{a} \to '\mathtt{b}]_{\equiv_{\mathrm{v}}^{\mathrm{DM}}}}\ \Psi_H
\end{aligned}$$

where $\Psi_H \triangleq \Lambda \tau \in \mathbb{M}_{\mathrm{v}}^{\mathrm{DM}} \cdot$
$\mathbf{lcg}\{\tau' \mid \langle H[\mathtt{f} \leftarrow \mathbf{gen}_{H[\mathtt{f} \leftarrow \emptyset_{\mathrm{p}}^{\mathrm{DM}}]}(\tau)],\, \tau'\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\}$

$$\begin{aligned}
\mathbf{T}^{\mathrm{DM}^2}[\![\mathbf{1}]\!] &\triangleq \{\langle H,\, \mathtt{int}\rangle \mid H \in \mathbb{H}_{\mathrm{p}}^{\mathrm{DM}}\} \\
\mathbf{T}^{\mathrm{DM}^2}[\![e_1 - e_2]\!] &\triangleq \{\langle H,\, \mathtt{int}\rangle \mid \langle H,\, \mathtt{int}\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![e_1]\!] \cap \\
&\qquad \mathbf{T}^{\mathrm{DM}^2}[\![e_2]\!]\} \\
\mathbf{T}^{\mathrm{DM}^2}[\![(e_1\ \mathbf{?}\ e_2 \mathbf{:} e_3)]\!] &\triangleq \{\langle H,\, \tau\rangle \mid \langle H,\, \mathtt{int}\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![e_1]\!] \wedge \\
&\qquad \langle H,\, \tau\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![e_2]\!] \cap \mathbf{T}^{\mathrm{DM}^2}[\![e_3]\!]\}
\end{aligned}$$

One can consider a $\supseteq$-upper approximation $\mathbf{T}^{\mathrm{DM}}[\![\bullet]\!]$ with $\mathbf{T}^{\mathrm{DM}}[\![e]\!]$ which is defined à la Milner [29] as $\mathbf{T}^{\mathrm{DM}^2}[\![e]\!]$ but for the monomorphic typing of recursive definitions $\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e$:

$$\mathbf{T}^{\mathrm{DM}}[\![\boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e]\!] \triangleq \{\langle H,\ \tau_1 \to \tau_2\rangle \mid \langle H[\mathtt{f} \leftarrow \{\tau_1 \to \tau_2\}],\, \tau_1 \to \tau_2\rangle \in \mathbf{T}^{\mathrm{DM}}[\![\boldsymbol{\lambda}\mathtt{x} \cdot e]\!]\}$$

By defining judgments as:

$$H \overset{\mathrm{DM}}{\vdash} e \Rightarrow \tau \triangleq \langle H,\, \tau\rangle \in \mathbf{T}^{\mathrm{DM}^2}[\![e]\!]$$

The Damas-Milner semantics can be presented in the equivalent rule based form [17]:

[13] A type variable renaming may be necessary to have bound generic type variables $'\mathtt{b}_1, \ldots, '\mathtt{b}_\ell$ in $\mathbb{V}_{\mathrm{g}}$ and free parametric type variables $\mathbf{ftv}(\tau) \cap \mathbf{ftv}(H)$ in $\mathbb{V}_{\mathrm{p}}$.

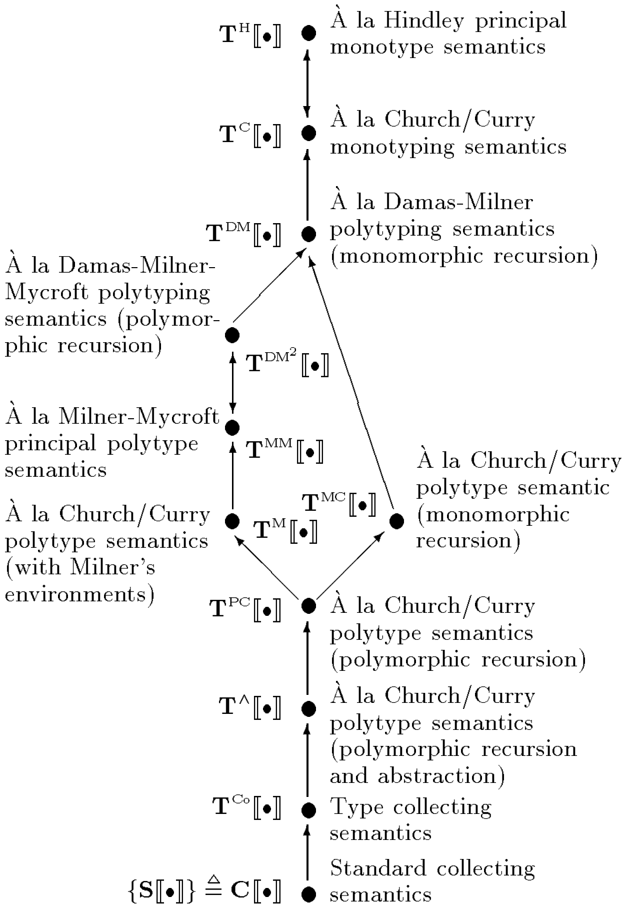

Figure 1: Lattice of type abstract interpretations

$$\frac{\tau \leq_{\mathrm{v}}^{\mathrm{DM}} \mathbf{elim}(H(\mathtt{x}))}{H \overset{\mathrm{DM}}{\vdash} \mathtt{x} \Rightarrow \tau} \qquad \frac{H[\mathtt{x} \leftarrow \tau_1] \overset{\mathrm{DM}}{\vdash} e \Rightarrow \tau_2}{H \overset{\mathrm{DM}}{\vdash} \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow \tau_1 \rightarrow \tau_2}$$

$$\frac{H \overset{\mathrm{DM}}{\vdash} e_1 \Rightarrow \tau_1 \rightarrow \tau_2, \ H \overset{\mathrm{DM}}{\vdash} e_2 \Rightarrow \tau_1}{H \overset{\mathrm{DM}}{\vdash} e_1(e_2) \Rightarrow \tau_2}$$

$$\frac{H[\mathtt{f} \leftarrow \tau_1 \rightarrow \tau_2] \overset{\mathrm{DM}}{\vdash} \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow \tau_1 \rightarrow \tau_2}{H \overset{\mathrm{DM}}{\vdash} \boldsymbol{\mu}\mathtt{f} \cdot \boldsymbol{\lambda}\mathtt{x} \cdot e \Rightarrow \tau_1 \rightarrow \tau_2}$$

$$H \overset{\mathrm{DM}}{\vdash} \mathbf{1} \Rightarrow \mathtt{int} \qquad \frac{H \overset{\mathrm{DM}}{\vdash} e_1 \Rightarrow \mathtt{int}, \ H \overset{\mathrm{DM}}{\vdash} e_2 \Rightarrow \mathtt{int}}{H \overset{\mathrm{DM}}{\vdash} e_1 - e_2 \Rightarrow \mathtt{int}}$$

$$\frac{H \overset{\mathrm{DM}}{\vdash} e_1 \Rightarrow \mathtt{int}, \ H \overset{\mathrm{DM}}{\vdash} e_2 \Rightarrow \tau, \ H \overset{\mathrm{DM}}{\vdash} e_3 \Rightarrow \tau}{H \overset{\mathrm{DM}}{\vdash} (e_1 \ \mathbf{?} \ e_2 \ \mathbf{:} \ e_3) \Rightarrow \tau}$$

18 Lattice of Type Abstract Interpretations

The considered type semantics can be organized within the *lattice of abstract interpretations* as shown in **Fig. 1** ($\rightarrow$ denotes abstraction and $\leftrightarrow$ equivalence).

19 Conclusion

It has been shown that à la Curry type systems can be designed constructively by abstract interpretation of a standard denotational semantics. This leads to a hierarchy of type systems (including recursive types, intersection types, etc. not considered for short) abstracting the type collecting semantics. The view of *types as abstract interpretations* should be useful to type system designers. The formalization corresponds to a shift from syntax (where defining languages by typing rules is understood as a context-sensitive syntax [31]) to semantics (where types are understood as approximate semantic properties). Clearly operational semantics which was used in the original definition of abstract interpretation [6, 7, 9] could also have been used. The approach should be applicable to Church typing, where the type system is part of the language definition, by designing a semantics incorporating runtime types and type checking. Moreover, this abstraction idea goes beyond the few type systems considered in this paper as shown by [32, 33, 36], including system F [35].

This application of abstract interpretation to the design of type systems shows once again after [11, 13] that the use of a denotational versus an operational semantics is no problem for the Galois connection-based framework introduced in [6, 7, 9]. In particular their is no need to abandon the idea of best abstraction in favor of weaker safety proofs by logical relations [24, 38] when such a best abstraction does exist. If it doesn't, a concretization function will do equally well [11]. The key and simple idea is to define the standard collecting semantics properly.

The comparison of type systems and program analysis goes beyond equivalence results such as [40] because both type system and program analysis are not only compared but formalized within the same abstract interpretation framework. For example the comparison of abstraction functions only is sufficient to perform expressiveness comparisons.

As far as type system implementors and compiler writers are concerned, this work should be useful for the rapprochement of type and abstract interpretation theory, maybe to go beyond current type-based or effect-based program analyses, which heavily, if not exclusively, rely on the representation of program properties by terms [4, 25, 26] and/or have no realistic type/property inference algorithm [22]. With the term representation of program properties, it is sometimes very heavy and difficult to obtain as powerful analyzes as considered in abstract interpretation (such as e.g. [7, 14, 16]). However by choosing to combine these abstract domains with those considered for type systems, instead of trying to use a uniform but inexpressive term-encoding of abstract properties, one should get interesting new perspectives in program analysis.

These new abstract domains combining typing properties with other abstract properties might even be useful to improve the present type inference algorithms, without necessarily changing the corresponding type systems. A sound type inference algorithm which would be more precise than required by the typing rules would be harmless for the programmer and certainly useful to an optimizing compiler. This is because formal type inference rules are often not used/usable in practice (and sometimes not even provided with the language definition). No one is interested in these rules for programs which are well-typed by the compiler. For programs which are rejected by the compiler, most programmers seem not think in term of type inference rules (that is $\mathbf{T}[\![\bullet]\!]$) but in terms of some abstraction of program execution (that is $\alpha(\mathbf{C}[\![\bullet]\!]) \neq \mathbf{T}[\![\bullet]\!]$). If this point of view is accepted, more powerful program analysis-based, hence undecidable type systems, might be considered. This would be a step towards the ideal unattainable situation where *untypable programs should go wrong* while *programs that cannot go wrong should be typable!*

Acknowledgments

This work was partly supported by Esprit BRA 8130 LOMAPS. I thank R. Cousot, R. Cridlig, É. Goubault, J. Goubault-Larrecq, L. Mauborgne, B. Monsuez, F. Védrine and A. Venet for their comments on the draft of this paper.

References

[1] H. Barendregt. Lambda calculi with types. Vol. 2 of *Handbook of Logic in Computer Science*, pp. 117–309. Clarendon Press, 1992.

[2] L. Cardelli. Type systems. *ACM Comput. Surv.*, 28(1):262–267, 1996.

[3] M. Coppo. A completeness theorem for recursively defined types. LNCS 194:120–129. Springer, 1985.

[4] M. Coppo, F. Damiani, & P. Giannini. Refinement types for program analysis. LNCS 1145:143–158. Springer, 1996.

[5] M. Coppo & M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Formal Logic*, 21(4):685–693, 1980.

[6] P. Cousot. Semantic foundations of program analysis. Ch. 10 of *Program Flow Analysis: Theory and Applications*, pp. 303–342. Prentice-Hall, 1981.

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

[8] ———. Constructive versions of Tarski's fixed point theorems. *Pacific J. Math.*, 82(1):43–57, 1979.

[9] ———. Systematic design of program analysis frameworks. *6^{th} ACM POPL*:269–282, 1979.

[10] ———. Abstract interpretation and application to logic programs. *J. Logic Prog.*, 13(2–3):103–179, 1992.

[11] ———. Abstract interpretation frameworks. *J. Logic and Comp.*, 2(4):511–547, 1992.

[12] ———. Inductive definitions, semantics and abstract interpretation. *19^{th} ACM POPL*:83–94, 1992.

[13] ———. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages). *ICCL'94*:95–112. IEEE Comp. Soc. Press, 1994.

[14] ———. Formal language, grammar and set-constraint-based program analysis by abstract interpretation. In 7^{th} ACM FPCA:170–181, 1995.

[15] ———. Compositional and inductive semantic definitions in fixpoint, equational, constraint, closure-condition, rule-based and game-theoretic form. LNCS 939:293–308. Springer, 1995.

[16] P. Cousot & N. Halbwachs. Automatic discovery of linear restraints among variables of a program. *5^{th} ACM POPL*:84–97, 1978.

[17] L. Damas & R. Milner. Principal type-schemes for functional programs. *9^{th} ACM POPL*:207–212, 1982.

[18] S. Debray. Formal bases for dataflow analysis of logic programs. Ch. 3 of *Advances in Logic Programming Theory*, pp. 115–182. Clarendon Press, 1994.

[19] C. Gunter. The semantics of types in programming languages. Vol. 3 of *Handbook of Logic in Computer Science*, pp. 395–475. Clarendon Press, 1994.

[20] C. Gunter & D. Scott. Semantic domains. Vol. B of *Handbook of Theoretical Computer Science*, pp. 633–674. Elsevier, 1990.

[21] R. Hindley. The principal type-scheme of an object in combinatory logic. *Trans. Amer. Math. Soc.*, 146:29–60, 1969.

[22] T. Jensen. Disjunctive strictness analysis. *7^{th} LICS*:174–185. IEEE Comp. Soc. Press, 1992.

[23] T. Jim. What are principal typings and what are they good for? *23^{rd} ACM POPL*:42–53, 1996.

[24] N. Jones & F. Nielson. Abstract interpretation: a semantics-based tool for program analysis. Vol. 4 of *Handbook of Logic in Computer Science*, pp. 527–636. Clarendon Press, 1995.

[25] P. Jouvelot & D. Gifford. Algebraic reconstruction of types and effects. *18^{th} ACM POPL*:303–310, 1991.

[26] T. Kuo & P. Mishra. Strictness analysis: A new perspective based on type inference. *3^{rd} ACM FPCA*:260–272, 1989.

[27] J.-L. Lassez, M. Maher, & K. Marriott. Unification revisited. *Foundations of Deductive Databases and Logic Programming*, pp. 587–625. Morgan Kaufmann, 1988.

[28] D. MacQueen, G. Plotkin, & R. Sethi. An ideal model for recursive polymorphic types. *Inf. & Comp.*, 71:95–130, 1986.

[29] R. Milner. A theory of polymorphism in programming. *J. Comput. Sys. Sci.*, 17(3):348–375, 1978.

[30] R. Milner & M. Tofte. Co-induction in relational semantics. *TCS*, 87:209–220, 1991.

[31] J. Mitchell. Type systems for programming languages. Vol. B of *Handbook of Theoretical Computer Science*, pp. 365–458. Elsevier, 1990.

[32] B. Monsuez. Polymorphic typing by abstract interpretation. LNCS 652:127–138. Springer, 1992.

[33] ———. Polymorphic types and widening operators. LNCS 724:267–281. Springer, 1993.

[34] ———. Polymorphic typing for call-by-name semantics. LNCS 735:156–169. Springer, 1993.

[35] ———. System F and abstract interpretation. LNCS 983:279–295. Springer, 1995.

[36] ———. Using abstract interpretation to define a strictness type inference system. *ACM PEPM'95*:122–133, 1995.

[37] A. Mycroft. Polymorphic type schemes and recursive definitions. LNCS 167:217–228. Springer, 1984.

[38] A. Mycroft & N. Jones. A relational framework for abstract interpretation. LNCS 215:156–171. Springer, 1986.

[39] A. Mycroft & R. O'Keefe. A polymorphic type system for Prolog. *IA*, 23:289–298, 1984.

[40] J. Palsberg & P. O'Keefe. A type system equivalent to flow analysis. *TOPLAS*, 17(4):576–599, 1995.

[41] A. Shamir & W. Wadge. Data types as objects. LNCS 52:465–479. Springer, 1977.