# Conjunctive Types and Useless-code Elimination (Extended Abstract)

F. DAMIANI
*Università di Torino, Italy*

**Abstract**

We investigate the use of *conjunctive non-standard type inference* for the elimination of useless-code in higher-order typed functional programs. In particular, we present a non-standard type assignment system for detecting useless-code and a mapping that simplifies a program by removing all the useless-code that can be detected by using the system.

**Keywords**

non-standard type inference, program analysis, useless-code elimination.

## 1 Introduction

*Useless-code elimination* is a program transformation technique that eliminates subexpressions that do not contribute to the final result of a program. Useless-code elimination focused on the elimination of useless function's parameters is often called *useless-variable elimination*. Various techniques for removing useless-code in higher-order functional programs have been proposed (see, for instance, [10, 1, 2, 5, 9, 11, 7, 6]).

The basic idea for detecting useless-code via a type system is to use type inference rules to decorate terms with *non-standard types*. The decoration points out that some subterms are useless-code while others may be useful to compute the value of the term. Take, for instance, the language PCF [8] with the primitive type of natural numbers, $\mathsf{nat}$. We will consider two non-standard types for PCF terms of type $\mathsf{nat}$: $\delta^{\mathsf{nat}}$ (saying that the term *may be used*) and $\omega^{\mathsf{nat}}$ (saying that the term *is not used)*. For instance, given the term $P = (\lambda x^{\mathsf{nat}}.3)y^{\mathsf{nat}}$ it is easy to see that the decoration $P' = ((\lambda x.3)y^{\omega^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$, stating that the term $P$ may be used while its subterm $y^{\mathsf{nat}}$ is not used, is correct[1]. Another decoration of $P$ is $P'' = ((\lambda x.3)y^{\delta^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$ which, although correct, is "less informative" than $P'$, since it states that every subterm of $P$ may be used. Given a non-standard type assignment system for useless-code analysis $\vdash$ and a term $M$, we call *optimal* $\vdash$*-typing of* $M$

[1] Under a *call-by-name* evaluation strategy.

any decoration of $M$ by $\vdash$ that shows all the useless-code of $M$ that can be detected by using $\vdash$. Note that the decoration $P'$ shows all the useless-code of $P$, so $P'$ is optimal w.r.t. all the systems where it can be derived.

Take a PCF program $M = (\lambda f^{\rho}.U)V$, and suppose that the variable $f^{\rho}$ in $U$ has two different uses and two different corresponding non-standard types $\phi_1$ and $\phi_2$. What is the non-standard type we should assign to $V$? We can consider three different solutions. *(i)* Considering only those decorations in which $\phi_1$ and $\phi_2$ are equal. This solution (which is essentially the one adopted in [1, 7, 6]) results in a useless-variable elimination and may prevent us to discover some useless-code. *(ii)* Considering only those decorations in which both $\phi_1$ and $\phi_2$ are entailed by a more informative non-standard type, say $\phi$, that can be assigned to $V$. Also this solution (which is essentially the one adopted in [2, 4]) may prevent us to discover some useless-code. *(iii)* Assign both $\phi_1$ and $\phi_2$ to $V$, i.e., assign to $V$ the *conjunction*, $\phi_1 \wedge \phi_2$, of the two non-standard types. In this way there is no restriction on the decoration that may prevent us to discover the useless-code exploited by the two different non-standard types of $V$.

In Section 2 of this paper we introduce the programming language we are dealing with and its operational semantics. In Section 3 we briefly show how program properties can be represented by partial equivalence relations on a term model of the programming language. In Section 4 we describe the set of non-standard types for useless-code analysis and its semantics. In Section 5 we present the non-standard type assignment system and a mapping that simplifies a program by removing all the useless-code that can be detected by using the system.

A preliminary version of the material presented in this paper appeared in Chapter 8 of [3], and a restricted version of the system, where conjunction is limited to rank 2 and not used to deal with recursive programs, has been presented in [5] (see also Chapter 9 of [3]). The main result given in the present paper, which is the existence of optimal simplifications, is new.

## 2 The Language PCFP

In this section we introduce the functional programming language PCFP ("PCF with Pairs") and its operational semantics.

**Definition 1 (PCFP types)** *The only* ground *type considered is the set of natural numbers,* $\mathsf{nat}$. *The set of* types *(*$\mathbf{T}$*), ranged over by* $\rho$, $\sigma$, $\tau$, *is defined by the following grammar:* $\rho ::= \mathsf{nat} \mid \rho \to \rho \mid \rho \times \rho$ .

PCFP terms are defined from a set $\mathcal{V}$ of typed *term variables* (ranged over by $x^{\rho}, y^{\sigma}, \ldots$) and a set $\mathcal{K} = \{0^{\mathsf{nat}}, 1^{\mathsf{nat}}, \ldots +^{\mathsf{nat}\times\mathsf{nat}\to\mathsf{nat}}, -^{\mathsf{nat}\times\mathsf{nat}\to\mathsf{nat}}\}$ of typed *term constants* (ranged over by $\mathsf{k}$). The type of a constant $\mathsf{k}$ is denoted by $\mathbf{T}(\mathsf{k})$.

**Definition 2 (PCFP terms)** *The set of* PCFP terms $\Lambda_{\mathbf{T}}$ *(ranged over by M, N, ...) is defined by:* $\Lambda_{\mathbf{T}} = \cup_{\rho\in\mathbf{T}} \Lambda_{\rho}$, *where* $\Lambda_{\rho}$ *is the set of the* PCFP terms of type

(Var) $x^\rho \in \Lambda_\rho$ (Con) $\mathsf{k}^\rho \in \Lambda_\rho$

$(\to \mathrm{I})\ \dfrac{M \in \Lambda_\sigma}{\lambda x^\rho.M \in \Lambda_{\rho\to\sigma}}$ $(\to \mathrm{E})\ \dfrac{M \in \Lambda_{\rho\to\sigma} \quad N \in \Lambda_\rho}{MN \in \Lambda_\sigma}$

$(\times \mathrm{I})\ \dfrac{M_1 \in \Lambda_{\rho_1} \quad M_2 \in \Lambda_{\rho_2}}{\langle M_1, M_2\rangle \in \Lambda_{\rho_1\times\rho_2}}$ $(\times \mathrm{E}_i)\ \dfrac{M \in \Lambda_{\rho_1\times\rho_2}}{\mathsf{prj}_i M \in \Lambda_{\rho_i}}\ i \in \{1,2\}$

$(\mathrm{Fix})\ \dfrac{M \in \Lambda_\rho}{\mathsf{fix}\, x^\rho.M \in \Lambda_\rho}$ $(\mathrm{Ifz})\ \dfrac{N \in \Lambda_{\mathsf{nat}} \quad M_1 \in \Lambda_\rho \quad M_2 \in \Lambda_\rho}{\mathsf{ifz}\, N\, \mathsf{then}\, M_1\, \mathsf{else}\, M_2 \in \Lambda_\rho}$

Figure 1: Rules for PCFP term formation

$\rho$*, defined by the rules in Fig. 1.*

In a PCFP term $M$, the types of variables and constants are explicitly mentioned. In the following we will often omit to write types which are clear from the context. The finite set of the *free variables* of a term $M$ is denoted by $\mathrm{FV}(M)$. We will identify terms modulo renaming of the bound variables. A substitution is a finite function mapping term variables to terms, denoted by $[x_1 := N_1, \ldots, x_n := N_n]$ ($[\vec{x} := \vec{N}]$ for short), which respects the types, i.e., each $x_i^{\rho_i}$ is substituted by a term $N_i$ of the same type. Substitution acts on free variables, the renaming of the bound variables is implicitly assumed.

Let $\Lambda^{\mathbf{c}}_{\mathbf{T}}$ be the set of the *closed terms*, i.e., $\Lambda^{\mathbf{c}}_{\mathbf{T}} = \{M \in \Lambda_{\mathbf{T}} \mid \mathrm{FV}(M) = \emptyset\}$. The operational semantics of PCFP is given by an inductively defined *evaluation relation*, $M \Downarrow K$, where $M$ is a closed term and $K$ is a closed term in *weak head normal form* (*w.h.n.f.*), i.e., an element of the set of *values*: $\mathbf{V}_{\mathbf{T}} = \mathcal{K} \cup \{\lambda x^\rho.N \mid \lambda x^\rho.N \in \Lambda^{\mathbf{c}}_{\mathbf{T}}\} \cup \{\langle M_1, M_2\rangle \mid \langle M_1, M_2\rangle \in \Lambda^{\mathbf{c}}_{\mathbf{T}}\}$. Any PCFP constant has either type $\mathsf{nat}$ or $\mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$. The meaning of a functional constant $\mathsf{k}$ can be given by a set $\mathbf{mean}(\mathsf{k})$ of pairs, i.e., if $(K, \mathsf{k}_1) \in \mathbf{mean}(\mathsf{k})$ then $\mathsf{k}K$ evaluates to $\mathsf{k}_1$. For example $(\langle 1, 3\rangle, 4) \in \mathbf{mean}(+)$.

**Definition 3 (Value of a term)** *Let $M \in \Lambda^{\mathbf{c}}_{\mathbf{T}}$. We write $M \Downarrow K$, and say that $M$ evaluates to $K$, if this statement is derivable by using the rules in Fig. 2.*

$M \Downarrow$ ("$M$ is convergent") means that, for some $K$, $M \Downarrow K$. $M \Uparrow$ ("$M$ is divergent") means that there exists no $K$ for which $M \Downarrow K$.

Let $(\mathcal{C}[\ ]^\rho)^\sigma$ denote a typed context of type $\sigma$ with a hole of type $\rho$ in it.

**Definition 4 (Ground contextual equivalence)** *Let $M$ and $N$ be terms of type $\rho$. Define $M \preceq_{\mathrm{obs}} N$ whenever, for all closed contexts $(\mathcal{C}[\ ]^\rho)^{\mathsf{nat}}$, if $\mathcal{C}[M]$ and $\mathcal{C}[N]$ are closed terms, then $\mathcal{C}[M] \Downarrow$ implies $\mathcal{C}[N] \Downarrow$. The relation $\preceq_{\mathrm{obs}}$ is the* ground contextual preorder *and the equivalence induced by $\preceq_{\mathrm{obs}}$, denoted by $\simeq_{\mathrm{obs}}$, is the* ground observational equivalence.

Let $\mathbf{I}(\rho)$ denote the set of the equivalence classes of the relation $\simeq_{\mathrm{obs}}$ on the closed PCFP terms of type $\rho$. Let $[M]$ denote the equivalence class of the closed

$$(\text{CAN})\ \frac{K \in \mathbf{V_T}}{K \Downarrow K} \qquad (\text{APP})\ \frac{M \Downarrow \lambda x.P \quad P[x := N] \Downarrow K}{MN \Downarrow K}$$

$$(\mathcal{K}\text{APP})\ \frac{M \Downarrow \mathsf{k} \quad N \Downarrow \langle N_1, N_2\rangle \quad N_1 \Downarrow \mathsf{k}_1 \quad N_2 \Downarrow \mathsf{k}_2}{MN \Downarrow \mathsf{k}_3}\ (\langle \mathsf{k}_1, \mathsf{k}_2\rangle, \mathsf{k}_3) \in \mathbf{mean}(\mathsf{k})$$

$$(\text{PRJ}_i)\ \frac{P \Downarrow \langle M_1, M_2\rangle \quad M_i \Downarrow K}{\mathsf{prj}_i P \Downarrow K}\ i \in \{1,2\} \quad (\text{FIX})\ \frac{M[x := \mathsf{fix}\, x.M] \Downarrow K}{\mathsf{fix}\, x.M \Downarrow K}$$

$$(\text{IFZ}_1)\ \frac{N \Downarrow \mathsf{0} \quad M_1 \Downarrow K}{\mathsf{ifz}\, N\, \mathsf{then}\, M_1\, \mathsf{else}\, M_2 \Downarrow K} \qquad (\text{IFZ}_2)\ \frac{N \Downarrow \mathsf{n} \quad M_2 \Downarrow K}{\mathsf{ifz}\, N\, \mathsf{then}\, M_1\, \mathsf{else}\, M_2 \Downarrow K}\ \mathsf{n} \neq \mathsf{0}$$

Figure 2: “Natural semantics” evaluation rules

term $M$, and let the extension of the the relation $\preceq_{\text{obs}}$ to equivalence classes be defined by: $[M] \preceq_{\text{obs}} [N]$ if and and only if $M \preceq_{\text{obs}} N$. The *closed term model* $\mathcal{M}$ of PCFP is defined by interpreting each type $\rho$ as the poset $\mathbf{I}(\rho)$ ordered by $\preceq_{\text{obs}}$. The equivalence class of the divergent computation of type $\rho$, $[\mathsf{fix}\, x^{\rho}.x]$, is the least element of $\mathbf{I}(\rho)$. An *environment* is a mapping $e : \mathcal{V} \to \cup_{\rho \in \mathbf{T}} \mathbf{I}(\rho)$ such that, for all $x^{\rho}$, $e(x^{\rho}) \in \mathbf{I}(\rho)$. The interpretation of a term $M$ in an environment $e$ is defined by: $[\![M]\!]_e = [M[x_1 := N_1, \ldots, x_n := N_n]]$, where $\{x_1, \ldots, x_n\} = \text{FV}(M)$ and $[N_l] = e(x_l)$ $(1 \leq l \leq n)$. Note that $\mathcal{M}$ is an extensional model.

# 3 P.e.r.s over $\mathcal{M}$ as Program Properties

A *partial equivalence relation* (*p.e.r.* for short) over a set $A$ is a symmetric and transitive binary relation over $A$. Let $\mathcal{R}$ be a p.e.r. over a set $A$, we call *domain of* $\mathcal{R}$ the subset of $A$ defined as: $\text{Dom}(\mathcal{R}) = \{E \mid (E,E) \in \mathcal{R}\}$. Note that $\mathcal{R}$ is reflexive on its domain.

In the following, let “p.e.r. over $\rho$” mean “p.e.r. over $\mathbf{I}(\rho)$” ($\mathbf{I}(\rho)$ is the interpretation of the type $\rho$ in the closed term model $\mathcal{M}$). In this section we explain how some p.e.r.’s over $\rho$ can be read as properties of PCFP programs of type $\rho$.

The following definition formally explains what is meant by “a program $P$ of type $\rho$ satisfies the property (p.e.r.) $\mathcal{R}$ over $\rho$”.

**Definition 5** (*P* **satisfies** $\mathcal{R}$) *Let* $\mathcal{R}$, $\mathcal{R}_1, \ldots,$ $\mathcal{R}_n$ $(n \geq 0)$ *be p.e.r. over types* $\rho$, $\rho_1, \ldots, \rho_n$, *respectively. A program $P$ of type* $\rho$ *with free variables* $x_1^{\rho_1} \ldots, x_n^{\rho_n}$ satisfies *the property* $\mathcal{R}$ *under the assumptions* $\mathcal{R}_i$ *for* $x_i^{\rho_i}$ $(1 \leq i \leq n)$ *if, for all the environments $e$ and $e'$ such that* $(e(x_i^{\rho_i}), e'(x_i^{\rho_i})) \in \mathcal{R}_i$, *we have that* $([\![P]\!]_e, [\![P]\!]_{e'}) \in \mathcal{R}$.

For every type $\rho \in \mathbf{T}$, the diagonal p.e.r. over $\rho$: $\Delta^{\rho} = \{([M],[M]) \mid [M] \in \mathbf{I}(\rho)\}$, which equates each element of $\mathbf{I}(\rho)$ with itself only, is the property stating that the value of a term *does matter* (i.e., *it may be used*).

For every type $\rho \in \mathbf{T}$, the trivial p.e.r. over $\rho$: $\Omega^{\rho} = \{([M],[N]) \mid [M],[N] \in \mathbf{I}(\rho)\}$, which equates all the elements of $\mathbf{I}(\rho)$, is the “true” property, stating that the value

of a term *does not matter* (i.e., *it is not used*).
Given a p.e.r. $\mathcal{R}_1$ over $\rho_1$ and a p.e.r. $\mathcal{R}_2$ over $\rho_2$, let $\mathcal{R}_1 \twoheadrightarrow \mathcal{R}_2$ be the p.e.r. over $\rho_1 \to \rho_2$ defines as: $\mathcal{R}_1 \twoheadrightarrow \mathcal{R}_2 = \{([F],[G]) \mid \forall([M],[N]) \in \mathcal{R}_1.([FM],[GN]) \in \mathcal{R}_2\}$. For instance $\Omega^{\mathsf{nat}} \to \Delta^{\mathsf{nat}}$ is the property (satisfied by all the, necessary constant, functions of type $\mathtt{nat} \to \mathtt{nat}$ which do not use their argument) which says that the value of the application of the function can be used without using the argument. Note that, for every $\rho_1, \rho_2 \in \mathbf{T}$, $\Delta^{\rho_1} \twoheadrightarrow \Delta^{\rho_2} = \Delta^{\rho_1 \to \rho_2}$ and $\Omega^{\rho_1} \twoheadrightarrow \Omega^{\rho_2} = \Omega^{\rho_1 \to \rho_2} = \mathcal{R}_1 \twoheadrightarrow \Omega^{\rho_2}$ (for any p.e.r. $\mathcal{R}_1$ over $\rho_1$).
Given a p.e.r. $\mathcal{R}_1$ over $\rho_1$ and a p.e.r. $\mathcal{R}_2$ over $\rho_2$, let $\mathcal{R}_1 \times\!\!\!\times \mathcal{R}_2$ be the p.e.r. over $\rho_1 \times \rho_2$ defined as: $\mathcal{R}_1 \times\!\!\!\times \mathcal{R}_2 = \{([M],[N]) \mid \forall i \in \{1,2\}.([\mathsf{prj}_i M],[\mathsf{prj}_i N]) \in \mathcal{R}_i\}$. For every $\rho_1, \rho_2 \in \mathbf{T}$, $\Delta^{\rho_1} \times\!\!\!\times \Delta^{\rho_2} = \Delta^{\rho_1 \times \rho_2}$ and $\Omega^{\rho_1} \times\!\!\!\times \Omega^{\rho_2} = \Omega^{\rho_1 \times \rho_2}$.

**Example 1** *The function $G = (\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat} \times \mathsf{nat}}.\mathsf{prj}_1 y)z$ has, under the assumption $\Omega^{\mathsf{nat}}$ for the free variable z, the property $(\Delta^{\mathsf{nat}} \times\!\!\!\times \Omega^{\mathsf{nat}}) \twoheadrightarrow \Delta^{\mathsf{nat}}$. In fact, z is not used and the second component of any pair of numbers supplied to G is not used.*

The set-theoretic intersection between p.e.r.'s over a type $\rho$ is still a p.e.r. over $\rho$ and it represents the logical conjunction between properties, i.e., if a program $P$ has both the properties $\mathcal{R}_1$ and $\mathcal{R}_2$, then $P$ has also the property $\mathcal{R}_1 \cap \mathcal{R}_2$ (and vice-versa). Moreover, the set-theoretic inclusion between p.e.r.'s over a type $\rho$ represents a logical implication between properties, i.e., if $\mathcal{R}_1 \subseteq \mathcal{R}_2$ and a program $P$ has the property $\mathcal{R}_1$, then $P$ has also the property $\mathcal{R}_2$. For instance, for every p.e.r. $\mathcal{R}$ over $\rho$, $\mathcal{R} \subseteq \Omega^{\rho}$, according to the fact that $\Omega^{\rho}$ represents the "true" property. We also have $\Omega^{\rho_1} \twoheadrightarrow \Delta^{\rho_2} \subseteq \Delta^{\rho_1} \twoheadrightarrow \Delta^{\rho_2}$, since a function that ignores its argument can be used in any context where a function that may use its argument is expected.

# 4 Evaluation Types

In this section we introduce the set of conjunctive non-standard types for useless-code analysis, that we call *evaluation types* (*e-types* for short).

The syntax of the e-types does not allow conjunctions to the right of an arrow type constructor. This restriction makes the type manipulation easier without affecting the power of the logic.

**Definition 6 (Evaluation types)** *The set of* e-types $\mathbf{L}$*, ranged over by* $\phi$ *and* $\psi$*, is defined by:* $\mathbf{L} = \cup_{\rho \in \mathbf{T}} \mathbf{L}(\rho)$*, where* $\mathbf{L}(\rho)$ *is the sets of the* e-types with underlying type $\rho$*, defined by the rules in Fig. 3.* Conjunctions *of elements in* $\mathbf{L}(\rho)$*, i.e., expressions of the shape* $\phi_1 \wedge \cdots \wedge \phi_n$*, for some* $n \geq 1$ *and* $\phi_1, \ldots, \phi_n \in \mathbf{L}(\rho)$*, are ranged over by* $\xi$ *and* $\nu$*.*

In the following we will often write, since no confusion may arise, $\delta$ and $\omega$ as shorts for $\delta^{\mathsf{nat}}$ and $\omega^{\mathsf{nat}}$, respectively. Let $\phi^{\rho}$ ($\xi^{\rho}$) range over e-types (conjunctions) with underlying type $\rho$.

The meaning of an e-type $\phi^{\rho}$ is given by interpreting it as a p.e.r., $[\![\phi^{\rho}]\!]$, over the type $\rho$ (see Section 3).

$$(\delta)\ \delta^{\mathsf{nat}} \in \mathbf{L}(\mathsf{nat}) \qquad (\omega)\ \omega^{\rho} \in \mathbf{L}(\rho) \qquad (\to)\ \frac{\phi_1,\ldots,\phi_n \in \mathbf{L}(\rho) \quad \psi \in \mathbf{L}(\sigma)}{\phi_1 \wedge \cdots \wedge \phi_n \to \psi \in \mathbf{L}(\rho \to \sigma)} \begin{cases} n \geq 1 \\ \text{and } \psi \neq \omega^{\sigma} \end{cases}$$

$$(\times)\ \frac{\psi_1 \in \mathbf{L}(\rho_1) \quad \psi_2 \in \mathbf{L}(\rho_2)}{\psi_1 \times \psi_2 \in \mathbf{L}(\rho_1 \times \rho_2)}\ \exists i \in \{1,2\}.\psi_i \neq \omega^{\rho_i}$$

Figure 3: Evaluation types

**Definition 7 (Semantics of e-types)** *The interpretation $[\![\phi]\!]$ of an e-type $\phi$ is defined by:*

$$\begin{array}{rcl@{\qquad}rcl} [\![\delta^{\mathsf{nat}}]\!] & = & \Delta^{\mathsf{nat}} & [\![\omega^{\rho}]\!] & = & \Omega^{\rho} \\ [\![\xi \to \psi]\!] & = & [\![\xi]\!] \twoheadrightarrow [\![\psi]\!] & [\![\psi_1 \times \psi_2]\!] & = & [\![\psi_1]\!] \times\!\!\!\times [\![\psi_2]\!] \\ [\![\phi_1 \wedge \cdots \wedge \phi_n]\!] & = & \cap_{1 \leq i \leq n} [\![\phi_i]\!] & & & \end{array}$$

The e-type syntax (Definition 6) has been designed in such a way that syntactically different *conjunction-free* e-types denote (Definition 7) different p.e.r.'s. This observation justifies the choice of not having e-types of the form $\delta^{\rho}$, $\xi \to \omega^{\sigma}$ and $\omega^{\sigma_1} \times \omega^{\sigma_2}$, where $\rho \neq \mathtt{nat}$, $\sigma, \sigma_1, \sigma_2$ are any types, and $\xi$ is any conjunction.

**Definition 8 ($\omega$-e-types and $\delta$-e-types)** *We call $\omega$-e-types the e-types in the set $\mathbf{L}_{\omega} = \{\omega^{\rho} \mid \rho \in \mathbf{T}\}$.*
*The set of the $\delta$-e-types ($\mathbf{L}_{\delta}$) is the subset of the e-types which does not contain subexpressions of the form $\omega^{\rho}$ (for some $\rho$). For every type $\rho \in \mathbf{T}$, let $\delta(\rho)$ be the corresponding unique conjunction-free $\delta$-e-type.*

The following proposition shows that $\omega$-e-types and $\delta$-e-types provide a syntactical characterization for the set of the e-types $\phi^{\rho}$ such that $[\![\phi^{\rho}]\!] = \Omega^{\rho}$, and for the set of the e-types $\psi^{\sigma}$ such that $[\![\psi^{\sigma}]\!] = \Delta^{\sigma}$, respectively.

**Proposition 1** *1. $\phi^{\rho} \in \mathbf{L}_{\omega}$ (i.e. $\phi = \omega^{\rho}$) if and only if $[\![\phi^{\rho}]\!] = \Omega^{\rho}$, and*

*2. $\phi^{\rho} \in \mathbf{L}_{\delta}$ implies $[\![\phi^{\rho}]\!] = \Delta^{\rho}$.*

The entailment relation between e-types, $\leq$, models the set-theoretic inclusion between the interpretation of the e-types.

**Definition 9 (Entailment relation $\leq$)** *We write $\xi_1 \leq \xi_2$ to mean that this judgement is derivable by the rules in Fig. 4. By $\cong$ we denote the equivalence relation induced by $\leq$.*

Note that $\leq$ is reflexive and transitive. The most interesting rules are $(\wedge)$ and $(\times)$. Rule $(\wedge)$ subsumes the standard rules of conjunction introduction and elimination. Rule $(\times)$ allows to compare conjunctions of $n \geq 1$ products. Note that restricting the rule to e-types (i.e., requiring $n = 1$) reduces the power of the system. For instance, we have both $\delta \wedge \omega \leq \delta \wedge \delta$ and $\omega \wedge \delta \leq \delta \wedge \delta$, so, by rule $(\times)$, we get $\delta \times \omega \wedge \omega \times \delta \leq \delta \times \delta \wedge \delta \times \delta$ (and then, by rule $(\wedge)$, $\delta \times \omega \wedge \omega \times \delta \leq \delta \times \delta$), which is valid (since $[\![(\phi_1 \times \psi_1) \wedge \cdots\cdots \wedge (\phi_n \times \psi_n)]\!] = [\![\phi_1 \wedge \cdots \wedge \phi_n]\!] \times\!\!\!\times [\![\psi_1 \wedge \cdots \wedge \psi_n]\!]$) but cannot be derived when rule $(\times)$ is restricted to e-types.

$$(\delta)\ \delta^{\mathsf{nat}} \leq \delta^{\mathsf{nat}} \qquad (\omega)\ \phi^{\rho} \leq \omega^{\rho} \qquad (\rightarrow)\ \frac{\xi_2 \leq \xi_1 \quad \psi_1 \leq \psi_2}{\xi_1 \rightarrow \psi_1 \leq \xi_2 \rightarrow \psi_2}\ \psi_2 \notin \mathbf{L}_\omega$$

$$(\times)\ \frac{\phi_1 \wedge \cdots \wedge \phi_n \leq \psi_1 \wedge \cdots \wedge \psi_n \quad \phi'_1 \wedge \cdots \wedge \phi'_n \leq \psi'_1 \wedge \cdots \wedge \psi'_n}{(\phi_1 \times \phi'_1) \wedge \cdots \wedge (\phi_n \times \phi'_n) \leq (\psi_1 \times \psi'_1) \wedge \cdots \wedge (\psi_n \times \psi'_n)} \left\{ \begin{array}{l} \forall i \in \{1, \ldots, n\}. \\ \{\phi_i, \phi'_i\} \not\subseteq \mathbf{L}_\omega \\ \text{and } \{\psi_i, \psi'_i\} \not\subseteq \mathbf{L}_\omega \end{array} \right.$$

$$(\wedge)\ \frac{{\phi_{j_1}}^{\rho} \leq {\psi_1}^{\rho} \cdots {\phi_{j_n}}^{\rho} \leq {\psi_m}^{\rho}}{{\phi_1}^{\rho} \wedge \cdots \wedge {\phi_n}^{\rho} \leq {\psi_1}^{\rho} \wedge \cdots \wedge {\psi_m}^{\rho}}\ j_1, \ldots, j_m \in \{1, \ldots, n\}$$

Figure 4: Entailment rules for e-types and conjunctions (system $\leq$)

**Theorem 1 (Soundness of $\leq$)** $\phi \leq \psi$ *implies* $[\![\phi]\!] \subseteq [\![\psi]\!]$.

The entailment rules are not complete. Take, for instance, $\phi_1 = \omega \rightarrow \delta \rightarrow \delta$, $\phi_2 = \delta \rightarrow \omega \rightarrow \delta$, and $\psi = \omega \rightarrow \omega \rightarrow \delta$. We have $[\![\phi_1 \wedge \phi_2]\!] = [\![\psi]\!]$, but $\phi_1 \wedge \phi_2 \not\leq \psi$.

# 5 Using E-types for Useless-code Elimination

The basic idea for using e-types to detect the useless-code in a given PCFP term $M$ is as follows: if we can prove that the term has a $\delta$-e-type while some of its subterms have $\omega$-e-types, then the value of $M$ can be computed without using these subterms, so such subterms are useless-code.

## 5.1 An E-type Assignment System for Detecting Useless-code

An assumption for a term variable $x$ of type $\rho$ is an expression of the form $x : \xi$, where $\xi$ is a conjunction of underlying type $\rho$. A basis is a set $\Sigma$ of assumptions $x : \xi$ for term variables such that each term variable occurs at most once in $\Sigma$. E-types are assigned to terms $M \in \Lambda_{\mathbf{T}}$ by a set of inference rules for judgements of the form $\Sigma \vdash M'$ where: *(i)* $M'$ is a *decoration* of the term $M$, i.e., it has written in it (some of) the e-types assigned to its subterms; and *(ii)* $\Sigma$ is a basis containing an assumption for each free variable of the term $M$.

For any e-type $\phi$ (conjunction $\xi$) let $\varepsilon(\phi)$ ($\varepsilon(\xi)$) denote the underlying type of $\phi$ ($\xi$), and for any basis $\Sigma$ let $\mathrm{Dom}(\Sigma)$ denote the set of terms variables occurring in $\Sigma$, i.e., $\mathrm{Dom}(\Sigma) = \{x \mid x : \xi \in \Sigma, \text{ for some } \xi\}$. Let $\Sigma, x : \xi$ denote the basis $\Sigma \cup \{x : \xi\}$ where it is assumed that $x$ does not appear in $\Sigma$.

**Definition 10 (E-type assignment system $\vdash$)** $\Sigma \vdash M^{\phi}$ *means that this judgement can be derived by the rules in Fig. 5.*

A sequence $\{N^{\phi_1}; \cdots; N^{\phi_n}\}$ is just a way of keeping track of $n \geq 1$ decorations of a subterm. These decorations correspond to different uses of the subterm. When $n = 1$ we write $N^{\phi_1}$ instead of $\{N^{\phi_1}\}$. Let $\Lambda^{\vdash}_{\mathbf{T}}$ be the set of $\vdash$-decorations, i.e.,

$$(\omega)\ \frac{\mathrm{Dom}(\Sigma) \supseteq \mathrm{FV}(M) \quad M \in \Lambda_\rho}{\Sigma \vdash M^{\omega^\rho}} \qquad (\mathrm{Con})\ \Sigma \vdash \mathsf{k}^{\delta(\mathbf{T}(\mathsf{k}))}$$

$$(\mathrm{Var})\ \frac{\phi_i \leq \phi}{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash x^\phi}\ i \in \{1, \ldots, n\} \text{ and } \phi \notin \mathbf{L}_\omega$$

$$(\to \mathrm{I})\ \frac{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M^\psi}{\Sigma \vdash (\lambda x.M)^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi}}\ \psi \notin \mathbf{L}_\omega \qquad (\to \mathrm{E})\ \frac{\Sigma \vdash M^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi} \quad \Sigma \vdash N^{\phi_1} \ \ldots\ \Sigma \vdash N^{\phi_n}}{\Sigma \vdash (M\{N^{\phi_1}; \cdots; N^{\phi_n}\})^\psi}$$

$$(\times \mathrm{I})\ \frac{\Sigma \vdash {M_1}^{\psi_1} \quad \Sigma \vdash {M_2}^{\psi_2}}{\Sigma \vdash \langle M_1, M_2 \rangle^{\psi_1 \times \psi_2}}\ \exists i \in \{1,2\}.\psi_i \notin \mathbf{L}_\omega \quad (\times \mathrm{E}_i)\ \frac{\Sigma \vdash M^{\psi_1 \times \psi_2}}{\Sigma \vdash (\mathsf{prj}_i M^{\psi_1 \times \psi_2})^{\psi_i}}\ i \in \{1,2\} \text{ and } \psi_i \notin \mathbf{L}_\omega$$

$$(\mathrm{Fix})\ \frac{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M^{\phi_1} \ \cdots\ \Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M^{\phi_n}}{\Sigma \vdash (\mathsf{fix}\, x.\{M^{\phi_1}; \cdots; M^{\phi_n}\})^{\phi_1}}\ \forall i \in \{1, \ldots, n\}.\phi_i \notin \mathbf{L}_\omega$$

$$(\mathrm{If})\ \frac{\Sigma \vdash N^{\delta^{\mathsf{nat}}} \quad \Sigma \vdash {M_1}^{\phi_1} \quad \Sigma \vdash {M_2}^{\phi_2} \quad \phi_1 \leq \psi \quad \phi_2 \leq \psi}{\Sigma \vdash (\mathsf{ifz}\, N\, \mathsf{then}\, {M_1}^{\phi_1}\, \mathsf{else}\, {M_2}^{\phi_2})^\psi}\ \psi \notin \mathbf{L}_\omega$$

Figure 5: Rules for e-type assignment (system $\vdash$)

$\Lambda_{\mathbf{T}}^{\vdash} = \{M' \mid \Sigma \vdash M'$ for some basis $\Sigma\}$. For any decoration $M' \in \Lambda_{\mathbf{T}}^{\vdash}$, let $\varepsilon(M')$ denote the underlying PCFP term and, for every term $M \in \Lambda_{\mathbf{T}}$, let $\Lambda_{\mathbf{T}}^{\vdash}(M) = \{M' \in \Lambda_{\mathbf{T}}^{\vdash} \mid \varepsilon(M') = M\}$.

For any finite set $\Gamma$ of term variables let $\delta(\Gamma)$ and $\omega(\Gamma)$ denote the bases $\{x^\sigma : \delta(\sigma) \mid x^\sigma \in \Gamma\}$ and $\{x^\sigma : \omega^\sigma \mid x^\sigma \in \Gamma\}$, respectively. For any term $M \in \Lambda_{\mathbf{T}}$, we define $\delta(M)$ as the decoration of $M$ in which each subterm is annotated by the corresponding conjunction-free $\delta$-e-type. It is immediate to verify that $M \in \Lambda_\rho$ implies both $\delta(\mathrm{FV}(M)) \vdash \delta(M)$ and, for all basis $\Sigma$ such that $\mathrm{Dom}(\Sigma) \supseteq \mathrm{FV}(M)$, $\Sigma \vdash M^{\omega^\rho}$.

To state the soundness of the e-type assignment system we introduce the following definition.

**Definition 11** *Two environments $e_1$, $e_2$ are $\Sigma$-related if and only if, for all $x : \xi \in \Sigma$, $\langle e_1(x), e_2(x) \rangle \in [\![\xi]\!]$.*
*Let $\Sigma \vdash M^\psi$ and $\Sigma \vdash N^\psi$. We write $\varepsilon(M^\psi) \sim_\psi^\Sigma \varepsilon(N^\psi)$ to mean that, for all $e_1$, $e_2$, if $e_1$ and $e_2$ are $\Sigma$-related, then $\langle [\![\varepsilon(M^\psi)]\!]_{e_1}, [\![\varepsilon(N^\psi)]\!]_{e_2} \rangle \in [\![\psi]\!]$.*

Note that, for any program $P$ of type $\rho$ with free variables $x_1^{\rho_1} \ldots, x_n^{\rho_n}$, if $P \sim_\chi^{\{x_1^{\rho_1}:\xi_1 \ldots, x_n^{\rho_n}:\xi_n\}} P$, then $P$ satisfies (in the sense of Definition 5) the property $\mathcal{R} = [\![\chi]\!]$ under the assumptions $\mathcal{R}_i = [\![\xi_i]\!]$ for $x_i^{\rho_i}$ $(1 \leq i \leq n)$.

**Theorem 2 (Soundness of $\vdash$)** *Let $\Sigma \vdash P^\chi$. Then $\varepsilon(P^\chi) \sim_\chi^\Sigma \varepsilon(P^\chi)$.*

We give a first example of useless-code that can be detected by system $\vdash$, showing how conjunction is used, in rules $(\to \mathrm{I})$ and $(\to \mathrm{E})$, to assign different e-types to the argument of a function in order to exploit its different uses in the function's body.

**Example 2 (Use of rules $(\rightarrow \mathrm{I})$ and $(\rightarrow \mathrm{E})$)** *Take the term of type* nat,

$$M \;=\; (\lambda f^{(\mathsf{nat}\rightarrow\mathsf{nat})\rightarrow\mathsf{nat}\rightarrow\mathsf{nat}}.+\langle f(\lambda x^{\mathsf{nat}}.3)P, f(\lambda y^{\mathsf{nat}}.y)Q\rangle)\;\;(\lambda h^{\mathsf{nat}\rightarrow\mathsf{nat}}.\lambda z^{\mathsf{nat}}.hz)\;\;,$$

*where* $\mathrm{FV}(M)=\mathrm{FV}(P)=\{z^{\mathsf{nat}}\}$ *and* $\mathrm{FV}(Q)=\emptyset$.

*Let* $\phi_1=(\omega\rightarrow\delta)\rightarrow\omega\rightarrow\delta$ *and* $\phi_2=(\delta\rightarrow\delta)\rightarrow\delta\rightarrow\delta$*. The* $\vdash$*-typing* $\Sigma\vdash M'$*, where* $\Sigma=\{z:\omega\}$ *and* $M'$ *is the decoration*

$$((\lambda f.+\langle f(\lambda x.3)^{\omega\rightarrow\delta}P^{\omega}, f(\lambda y.y)^{\delta\rightarrow\delta}Q^{\delta}\rangle^{\delta\times\delta})\;\;\{(\lambda h.\lambda z.hz^{\omega})^{\phi_1};(\lambda h.\lambda z.hz^{\delta})^{\phi_2}\})^{\delta}\;\;,$$

*shows that the value of M can be computed without using the subterm P. In fact, we can replace P (which has e-type* $\omega$*) by any other term of type* nat *without changing the meaning of M (which has e-type* $\delta$*).*

*Note that, since it does not exist a* conjunction-free *e-type* $\phi$ *which entails both* $\phi_1$ *and* $\phi_2$*, the useless-code P cannot be detected without using conjunction.*

## 5.2 A Mapping for Removing Useless-code

In this section we define a useless-code elimination mapping, **O**, that, given a $\vdash$-typing $\Sigma\vdash M^{\Psi}$, simplifies $\varepsilon(M^{\Psi})$ by replacing the useless-code showed by the decorations with "dummy variables".

### 5.2.1 Dummy Variables and Pruning Relation

For every type $\rho$, we consider a countable set $\{\ \mathsf{d}^{\rho}, \mathsf{d}_1^{\rho}, \mathsf{d}_2^{\rho}, \ldots, \}$ of *dummy variables* of type $\rho$. We remark that dummy variables are not present in the original programs: they are introduced by the useless-code elimination mapping **O** as place-holders for the useless-code removed. So, it is safe to assume that every dummy variable is useless-code and that all the occurrences of dummy variables in a program are free and distinct. For every term $M$, let $\mathrm{DV}(M)$ be the set of the dummy variables in $M$, and let $\mathrm{FNDV}(M)=\mathrm{FV}(M)-\mathrm{DV}(M)$.

The below defined pruning relation on terms allows to compare simplifications of a given term $M$.

**Definition 12 (Pruning relation)** *Let* $M\in\Lambda_{\rho}$*. We say that a term N is a* pruning *of M, and write* $M\preceq_{\mathrm{prn}} N$*, if N can be obtained from M by replacing some subterms by fresh dummy variables of the corresponding type.*

It is immediate to verify that, for every term $M$, if $M\in\Lambda_{\rho}$ and $M\preceq_{\mathrm{prn}} N$ then $N\in\Lambda_{\rho}$ and $\mathrm{FNDV}(N)\subseteq\mathrm{FNDV}(M)$. Let $\Lambda_{\mathbf{T}}^{\preceq_{\mathrm{prn}}}(M)$ be the set of the prunings of $M$, i.e., $\Lambda_{\mathbf{T}}^{\preceq_{\mathrm{prn}}}(M)=\{N\mid M\preceq_{\mathrm{prn}} N\}$. In the following we will identify the prunings of a term $M$ modulo renaming of the dummy variables. So, modulo this identification, we have that for every term $M$ the set $\Lambda_{\mathbf{T}}^{\preceq_{\mathrm{prn}}}(M)$ is finite.

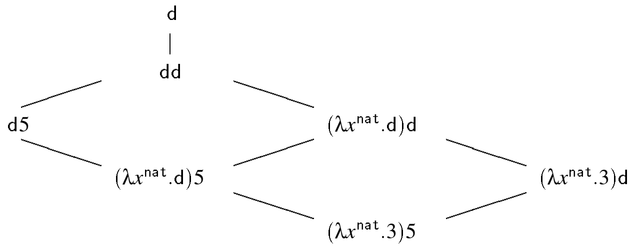


Figure 6: The lattice $\Lambda_{\mathbf{T}}^{\preceq_{\mathrm{prn}}}(P)$

.

**Example 3** *Take the term $P = (\lambda x^{\mathsf{nat}}.3)5$. The set $\Lambda_{\mathbf{T}}^{\preceq_{\mathrm{prn}}}(P)$ has (modulo renaming of the dummy variables) 7 elements and forms a complete lattice (see Fig. 6).*

**Definition 13 (Operation inf)** *For every term $M$ of type $\rho$ the binary operation* **inf** *on $\Lambda_{\mathbf{T}}^{\preceq_{\mathrm{prn}}}(M)$ is inductively defined by the clause*

$$\mathbf{inf}(M, \mathsf{d}^{\rho}) = \mathbf{inf}(\mathsf{d}^{\rho}, M) = M, \text{ for every term } M \text{ of type } \rho$$

*together with the clauses specifying that* **inf** *commutes with the term-formation operators:* $\mathbf{inf}(\mathsf{k}^{\rho}, \mathsf{k}^{\rho}) = \mathsf{k}^{\rho}$, $\mathbf{inf}(x^{\rho}, x^{\rho}) = x^{\rho}$, $\mathbf{inf}(\lambda x^{\rho}.M, \lambda x^{\rho}.N) = \lambda x^{\rho}.\mathbf{inf}(M,N)$, $\mathbf{inf}(M_1M_2, N_1N_2) = \mathbf{inf}(M_1,N_1)\mathbf{inf}(M_2,N_2)$*, etc.*

It is immediate to verify that the operation **inf** is idempotent, commutative, and associative. Moreover the following property holds.

**Theorem 3** *For every term $M$ of type $\rho$, the set $\Lambda_{\mathbf{T}}^{\preceq_{\mathrm{prn}}}(M)$ is a finite complete lattice with order relation $\preceq_{\mathrm{prn}}$, meet operation* **inf***, bottom element $M$, and top element $\mathsf{d}^{\rho}$.*

#### 5.2.2 The Simplification Mapping

**Definition 14 (Simplification mapping O)** *The function $\mathbf{O} : \Lambda_{\mathbf{T}}^{\vdash} \to \Lambda_{\mathbf{T}}$ is defined by the clauses in Fig. 7, where the occurrence of "$\mathsf{d}$" (first row) denotes a fresh dummy variable of the proper type.*

The simplification mapping produces well-typed terms and, for every $\vdash$-typing of $M$, $\Sigma \vdash M'$, we have that $M \preceq_{\mathrm{prn}} \mathbf{O}(M')$. Moreover the simplification mapping is correct w.r.t. the e-type semantics, as the following property shows.

**Lemma 1 (Correctness of O)** *For every term $M$, if $\Sigma \vdash M'^{\Psi}$ is a $\vdash$-typing of $M$ and $N$ is term such that $M \preceq_{\mathrm{prn}} N \preceq_{\mathrm{prn}} \mathbf{O}(M'^{\Psi})$, then*

$$\begin{array}{l}
\mathbf{O}(M^{\psi}) = \mathsf{d}, \quad \text{if } \psi \in \mathbf{L}_{\omega}, \quad \text{otherwise:} \\[1ex]
\mathbf{O}(\mathsf{k}^{\psi}) = \mathsf{k} \\
\mathbf{O}(x^{\psi}) = x \\
\mathbf{O}(\langle M_1, M_2\rangle^{\psi_1\times\psi_2}) = \langle \mathbf{O}(M_1{}^{\psi_1}), \mathbf{O}(M_2{}^{\psi_2})\rangle \\
\mathbf{O}((\mathsf{prj}_i M^{\psi_1\times\psi_2})^{\psi_i}) = \mathsf{prj}_i \mathbf{O}(M^{\psi_1\times\psi_2}), \quad \text{where } i \in \{1,2\} \\
\mathbf{O}((M\{N^{\phi_1};\cdots;N^{\phi_n}\})^{\psi}) = \mathbf{O}(M^{\phi_1\wedge\cdots\wedge\phi_n\to\psi})\mathbf{inf}(\mathbf{O}(N^{\phi_1}),\ldots,\mathbf{O}(N^{\phi_n})) \\
\mathbf{O}((\lambda x.M)^{\phi_1\wedge\cdots\wedge\phi_n\to\psi}) = \lambda x^{\varepsilon(\phi_1)}.\mathbf{O}(M^{\psi}) \\
\mathbf{O}((\mathsf{fix} x.\{M^{\phi_1};\cdots;M^{\phi_n}\})^{\phi_1}) = \mathsf{fix} x.\mathbf{inf}(\mathbf{O}(M^{\phi_1}),\ldots,\mathbf{O}(M^{\phi_n})) \\
\mathbf{O}((\mathsf{ifz}\, N\, \mathsf{then}\, M_1^{\phi_1}\, \mathsf{else}\, M_2^{\phi_2})^{\psi}) = \mathsf{ifz}\, \mathbf{O}(N^{\delta^{\mathsf{nat}}})\, \mathsf{then}\, \mathbf{O}(M_1{}^{\phi_1})\, \mathsf{else}\, \mathbf{O}(M_2{}^{\phi_2})
\end{array}$$

Figure 7: The simplification mapping **O**

1. $\Sigma' \vdash N'^{\psi}$, *where* $\Sigma' = \{x : \xi \in \Sigma \mid x \in \mathrm{FNDV}(N)\} \cup \{\mathsf{d}^{\sigma} : \omega^{\sigma} \mid \mathsf{d}^{\sigma} \in \mathrm{DV}(N)\}$ *and* $N'^{\psi}$ *is the decoration of* $N$ *obtained from* $M'^{\psi}$ *by replacing by* $\mathsf{d}$ *the subterms which have been replaced by* $\mathsf{d}$ *in* $N$.
2. $M \sim_{\psi}^{\Sigma\cup\Sigma'} N$.

In order to use the mapping **O** to remove useless-code, we identify a subset of $\vdash$-typings (that we call *faithful*) for which the $\sim_{\phi}^{\Sigma}$ relations implies the $\simeq_{\mathrm{obs}}$ relation. A faithful $\vdash$-typing says that the term can be used (it has a $\delta$-e-type) and that any its free variable is either used (it has a $\delta$-e-type) or not used at all (it has an $\omega$-e-type).

**Definition 15 (Faithful $\vdash$-typing)** *A* $\vdash$*-typing of a term* $M$ *of type* $\rho$, $\Sigma \vdash M'^{\phi}$, *is a* faithful $\vdash$*-typing and* $M'^{\phi}$ *is a faithful* $\vdash$*-decoration, if* $\mathrm{Dom}(\Sigma) = \mathrm{FV}(M)$, $\phi = \delta(\rho)$, *and, for all assumptions* $x^{\sigma} : \nu \in \Sigma$, *either* $\nu = \delta(\sigma)$ *or* $\nu = \omega^{\sigma}$.

The fundamental property of faithful $\vdash$-typings is the following.

**Lemma 2** *Let* $\Sigma_1 \vdash M'^{\delta(\rho)}$ *and* $\Sigma_2 \vdash N'^{\delta(\rho)}$ *be faithful* $\vdash$*-typings (of terms* $M$ *and* $N$*, respectively) such that* $\Sigma_1$ *and* $\Sigma_2$ *contains the same assumptions for the variables in* $\mathrm{FV}(M) \cap \mathrm{FV}(N)$. *Then* $M \sim_{\delta(\rho)}^{\Sigma_1\cup\Sigma_2} N$ *implies* $M \simeq_{\mathrm{obs}} N$.

Lemma 1 and Lemma 2 imply the following theorem, which shows that the dummy variables introduced by applying the mapping **O** on a faithful $\vdash$-typing of $M$ are useless-code.

**Theorem 4 (O on faithful $\vdash$-typings preserves $\simeq_{\mathrm{obs}}$)** *Let* $\Sigma \vdash M^{\delta(\rho)}$ *be a faithful* $\vdash$*-typing. Then* $\varepsilon(M^{\delta(\rho)}) \simeq_{\mathrm{obs}} \mathbf{O}(M^{\delta(\rho)})$.

**Example 4** *The* $\vdash$*-typing* $\Sigma \vdash M'$ *of Example 2 is faithful. By applying the simplification mapping* **O** *to* $M'$ *we get the term*

$$N \;=\; (\lambda f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}\to\mathsf{nat}}.+\langle f(\lambda x^{\mathsf{nat}}.3)\mathsf{d}, f(\lambda y^{\mathsf{nat}}.y)Q\rangle)\;\;(\lambda h^{\mathsf{nat}\to\mathsf{nat}}.\lambda z^{\mathsf{nat}}.hz)\;\;,$$

*which is observationally equivalent to* $M$.

The next example illustrates the other essential use of conjunction, that is in rule (Fix), to analyze recursive terms .

**Example 5 (Use of rule** (Fix)**)** *Take the type* $\rho = (\mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ *and the recursive term* $\mathsf{fix}\, f^{\rho}.M$*, where*

$$M = \lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\mathsf{ifz}\ z\ \mathsf{then}\ gx\ \mathsf{else}\ f(\lambda u^{\mathsf{nat}}.3)yx(-\langle z,1\rangle) \quad .$$

*Let* $\phi_1 = (\delta \to \delta) \to \delta \to \omega \to \delta \to \delta$ *and* $\phi_2 = (\omega \to \delta) \to \omega \to \omega \to \delta \to \delta$. *We have that* $\emptyset \vdash (\mathsf{fix}\, f.\{M^{\phi_1}; M^{\phi_2}\})^{\phi_1}$*, where*

$$\begin{array}{rcl} M^{\phi_1} & = & (\lambda g.\lambda x.\lambda y.\lambda z.\mathsf{ifz}\ z^{\delta}\ \mathsf{then}\ gx^{\delta}\ \mathsf{else}\ f(\lambda u.3)^{\omega\to\delta}y^{\omega}x^{\omega}(-\langle z,1\rangle^{\delta\times\delta})^{\delta})^{\phi_1} \\ M^{\phi_2} & = & (\lambda g.\lambda x.\lambda y.\lambda z.\mathsf{ifz}\ z^{\delta}\ \mathsf{then}\ gx^{\omega}\ \mathsf{else}\ f(\lambda u.3)^{\omega\to\delta}y^{\omega}x^{\omega}(-\langle z,1\rangle^{\delta\times\delta})^{\delta})^{\phi_2} \quad . \end{array}$$

*So, when we consider the application of* $\mathsf{fix}\, f^{\rho}.M$ *to any closed terms* $Q_1$, $Q_2$, $Q_3$ *of type* $\mathsf{nat}$*:* $R = (\mathsf{fix}\, f^{\rho}.M)(\lambda v^{\mathsf{nat}}.v)Q_1Q_2Q_3$*, we have the faithful* $\vdash$*-typing:* $\emptyset \vdash (\mathsf{fix}\, f.\{M^{\phi_1}; M^{\phi_2}\}(\lambda v.v)^{\delta\to\delta}Q_1^{\delta}Q_2^{\omega}Q_3^{\delta})^{\delta}$*, and, by applying the simplification mapping* **O***, we get the observationally equivalent term:* $S = (\mathsf{fix}\, f^{\rho}.N)(\lambda v^{\mathsf{nat}}.v)Q_1\mathsf{d}_3^{\mathsf{nat}}Q_3$*, where*

$$N = \lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\mathsf{ifz}\ z\ \mathsf{then}\ gx\ \mathsf{else}\ f(\lambda u^{\mathsf{nat}}.3)\mathsf{d}_1^{\mathsf{nat}}\mathsf{d}_2^{\mathsf{nat}}(-\langle z,1\rangle) \quad .$$

*Note that there is a (conjunction-free) derivation of* $\emptyset \vdash (\mathsf{fix}\, f.M^{\phi_2})^{\phi_2}$*, but there is* no *derivation for* $\emptyset \vdash (\mathsf{fix}\, f.M^{\phi_1})^{\phi_1}$*. Indeed, the above simplification of R cannot be done without using conjunction.*

## 5.3 Optimal Faithful $\vdash$-typings and Optimal Simplifications

In this section we prove that for every term $M$ there is an *optimal* faithful $\vdash$-typing, i.e., a typing which allows the mapping **O** to remove from $M$ all the useless-code that can be proved by the system $\vdash$.

### 5.3.1 Optimal Faithful $\vdash$-typings

Given two faithful $\vdash$-typings of $M$, $\Sigma_1 \vdash M_1^{\delta(\rho)}$ and $\Sigma_2 \vdash M_2^{\delta(\rho)}$, the one which proves more useless-code is the one that assigns $\omega$-e-types to more subterms. This fact induces a preorder relation (that we will denote by $\sqsubseteq$) on the set of the $\vdash$-typings of $M$. The preorder $\sqsubseteq$ (for e-types, basis, $\vdash$-decorations, and $\vdash$-typings) is defined as follows.

**Definition 16 (The $\sqsubseteq$ preorder)** *For every type* $\rho \in \mathbf{T}$*, the preorder* $\sqsubseteq$ *on the set of e-types* $\mathbf{L}(\rho)$ *is defined by the rules in Fig. 8* [2].

[2] The preorder $\sqsubseteq$, which is covariant in both the arguments of the constructor $\to$, should not be confused with the e-type entailment relation $\leq$, which is contravariant in the left argument of $\to$.

$$(\omega)\ \phi^{\rho} \sqsubseteq \omega^{\rho} \qquad (\delta)\ \delta^{\mathsf{nat}} \sqsubseteq \delta^{\mathsf{nat}} \qquad (\rightarrow)\ \frac{\xi_1 \sqsubseteq \xi_2 \quad \psi_1 \sqsubseteq \psi_2}{\xi_1 \rightarrow \psi_1 \sqsubseteq \xi_2 \rightarrow \psi_2}\ \psi_2 \notin \mathbf{L}_{\omega}$$

$$(\times)\ \frac{\phi_1 \sqsubseteq \psi_1 \quad \phi_2 \sqsubseteq \psi_2}{\phi_1 \times \phi_2 \sqsubseteq \psi_1 \times \psi_2}\ \exists i \in \{1,2\}.\psi_i \notin \mathbf{L}_{\omega}$$

$$(\wedge)\ \frac{\phi_{j_1} \sqsubseteq \psi_1 \ \cdots\ \phi_{j_m} \sqsubseteq \psi_m}{\phi_1 \wedge \cdots \wedge \phi_n \sqsubseteq \psi_1 \wedge \cdots \wedge \psi_m}\ j_1, \ldots, j_m \in \{1, \ldots, n\}$$

Figure 8: Relation $\sqsubseteq$ on $\mathbf{L}(\rho)$

*For every set of term variables $\Gamma$, the preorder $\sqsubseteq$ on the set of basis $\{\Sigma \mid \mathrm{Dom}(\Sigma) = \Gamma\}$ is defined by: $\Sigma_1 \sqsubseteq \Sigma_2$ if, for every $x^{\rho} : \nu_2 \in \Sigma_2$, there exists $x^{\rho} : \nu_1 \in \Sigma_1$ such that $\nu_1 \sqsubseteq \nu_2$.*
*For every term $M$, the preorder $\sqsubseteq$ on the set of $\vdash$-decorations $\Lambda^{\vdash}_{\mathbf{T}}(M)$ is defined by the rules in Fig. 9.*
*For every term $M$ and for every set of term variables $\Gamma$, such that $\Gamma \supseteq \mathrm{FV}(M)$, the preorder $\sqsubseteq$ on the set of the $\vdash$-typings of $M$ in which the basis contains exactly the variables in $\Gamma$ is defined by: $\Sigma_1 \vdash M_1^{\phi_1} \sqsubseteq \Sigma_2 \vdash M_2^{\phi_2}$ if $\Sigma_1 \sqsubseteq \Sigma_2$ and $M_1^{\phi_1} \sqsubseteq M_2^{\phi_2}$.*
*Let $\equiv$ be the equivalence relation induced by $\sqsubseteq$.*

The following lemma says that bigger (in the sense of $\sqsubseteq$) faithful $\vdash$-typings prove more useless-code.

**Lemma 3** *For every term $P$, if $\Sigma_1 \vdash {P_1}^{\chi_1}$ and $\Sigma_2 \vdash {P_2}^{\chi_2}$ two $\vdash$-typings of $P$ such that $\Sigma_1 \vdash {P_1}^{\chi_1} \sqsubseteq \Sigma_2 \vdash {P_2}^{\chi_2}$, then $\mathbf{O}({P_1}^{\chi_1}) \preceq_{\mathrm{prn}} \mathbf{O}({P_2}^{\chi_2})$.*

For every $M \in \Lambda_{\rho}$, consider the set of equivalence classes of the relation $\equiv$:
$\mathbf{faithful}^{\vdash}(M)/\equiv \ = \ \{[\Sigma \vdash M'^{\delta(\rho)}]^{\equiv} \mid \Sigma \vdash M'^{\delta(\rho)}$ is a faithful $\vdash$-typing of $M\}$ .
Let $\sqsubseteq^{\equiv}$ be the relation induced by $\sqsubseteq$ on the equivalence classes. The following property guarantees the existence of optimal faithful $\vdash$-typings, i.e., faithful $\vdash$-typings proving all the useless-code that can be proved by the system $\vdash$.

**Theorem 5** *For every term $M$, the set $\mathbf{faithful}^{\vdash}(M)/\equiv$ is a finite complete lattice with partial order relation $\sqsubseteq^{\equiv}$ and bottom element $[\delta(\mathrm{FV}(M)) \vdash \delta(M)]^{\equiv}$.*

Optimal faithful $\vdash$-typing can now be defined as follows.

**Definition 17 (Optimal faithful $\vdash$-typing)** *For every PCFP term $M$ we call* optimal faithful $\vdash$-typing of *$M$ every faithful $\vdash$-typing of $M$ whose equivalence class is the top of the lattice* $\mathbf{faithful}^{\vdash}(M)/\equiv$.

For instance, if $\emptyset \vdash Q^{\delta}$, $\emptyset \vdash Q_1^{\delta}$, and $\emptyset \vdash Q_3^{\delta}$ are optimal faithful $\vdash$-typings, then both the faithful $\vdash$-typing $\Sigma \vdash M'$ of Example 2 and the faithful $\vdash$-typing $\emptyset \vdash (\mathsf{fix}\, f.\{M^{\phi_1}; M^{\phi_2}\}(\lambda v.v)^{\delta\to\delta} Q_1^{\delta} Q_2^{\omega} Q_3^{\delta})^{\delta}$ of Example 5 are optimal.

$$(\omega)\ M^{\phi} \sqsubseteq \varepsilon(M)^{\omega^{\varepsilon(\phi)}} \qquad (\mathrm{Var})\ \frac{\phi \sqsubseteq \psi}{x^{\phi} \sqsubseteq x^{\psi}}\ \psi \notin \mathbf{L}_{\omega} \qquad (\mathrm{Con})\ \mathsf{k}^{\delta(\mathbf{T}(\mathsf{k}))} \sqsubseteq \mathsf{k}^{\delta(\mathbf{T}(\mathsf{k}))}$$

$$(\to \mathrm{I})\ \frac{\xi \sqsubseteq \nu \quad M^{\phi} \sqsubseteq N^{\psi}}{(\lambda x.M)^{\xi\to\phi} \sqsubseteq (\lambda x.N)^{\nu\to\psi}}\ \psi \notin \mathbf{L}_{\omega}$$

$$(\to \mathrm{E})\ \frac{F^{\phi_1\wedge\cdots\wedge\phi_n\to\phi} \sqsubseteq G^{\psi_1\wedge\cdots\wedge\psi_n\to\psi} \quad \{M^{\phi_1};\cdots;M^{\phi_m}\} \sqsubseteq \{N^{\psi_1};\cdots;N^{\psi_n}\}}{(F\{M^{\phi_1};\cdots;M^{\phi_m}\})^{\phi} \sqsubseteq (G\{N^{\psi_1};\cdots;N^{\psi_n}\})^{\psi}}$$

$$(\times \mathrm{I})\ \frac{M_1^{\phi_1} \sqsubseteq N_1^{\psi_1} \quad M_2^{\phi_2} \sqsubseteq N_2^{\psi_2}}{\langle M_1, M_2\rangle^{\phi_1\times\phi_2} \sqsubseteq \langle N_1, N_2\rangle^{\psi_1\times\psi_2}}\ \exists i \in \{1,2\}.\psi_i \notin \mathbf{L}_{\omega}$$

$$(\times \mathrm{E}_i)\ \frac{M^{\phi_1\times\phi_2} \sqsubseteq N^{\psi_1\times\psi_2}}{(\mathsf{prj}_i M^{\phi_1\times\psi_2})^{\phi_i} \sqsubseteq (\mathsf{prj}_i N^{\psi_1\times\psi_2})^{\psi_i}}\ i \in \{1,2\} \text{ and } \psi_i \notin \mathbf{L}_{\omega}$$

$$(\mathrm{Fix})\ \frac{\{M^{\phi_1};\cdots;M^{\phi_m}\} \sqsubseteq \{N^{\psi_1};\cdots;N^{\psi_n}\} \quad \phi_1 \sqsubseteq \psi_1}{(\mathsf{fix}\,x.\{M^{\phi_1};\cdots;M^{\phi_m}\})^{\phi_1} \sqsubseteq (\mathsf{fix}\,x.\{N^{\psi_1};\cdots;N^{\psi_n}\})^{\psi_1}}\ \begin{cases} \forall i \in \{1,\ldots,m\}.\phi_i \notin \mathbf{L}_{\omega} \\ \text{and } \forall j \in \{1,\ldots,n\}.\psi_j \notin \mathbf{L}_{\omega} \end{cases}$$

$$(\mathrm{If})\ \frac{B^{\delta^{\mathsf{nat}}} \sqsubseteq C^{\delta^{\mathsf{nat}}} \quad M_1^{\phi_1} \sqsubseteq N_1^{\psi_1} \quad M_2^{\phi_2} \sqsubseteq N_2^{\psi_2} \quad \phi \sqsubseteq \psi}{(\mathsf{ifz}\,B\,\mathsf{then}\,M_1^{\phi_1}\ \mathsf{else}\,M_2^{\phi_2})^{\phi} \sqsubseteq (\mathsf{ifz}\,C\,\mathsf{then}\,N_1^{\psi_1}\ \mathsf{else}\,N_2^{\psi_2})^{\psi}}\ \psi \notin \mathbf{L}_{\omega}$$

$$(\wedge)\ \frac{M^{\phi_{j_1}} \sqsubseteq N^{\psi_1}\ \ \ldots\ M^{\phi_{j_m}} \sqsubseteq N^{\psi_m}}{\{M^{\phi_1};\cdots;M^{\phi_n}\} \sqsubseteq \{N^{\psi_1};\cdots;N^{\psi_m}\}}\ j_1,\ldots,j_m \in \{1,\ldots,n\}$$

Figure 9: Relation $\sqsubseteq$ on $\Lambda^{\vdash}_{\mathbf{T}}(M)$

#### 5.3.2 Optimal Simplifications

We can now define a function $\mathbf{O}^{\vdash} : \Lambda_{\mathbf{T}} \to \Lambda_{\mathbf{T}}$ that eliminates from a term $M$ all the useless-code that can be proved by $\vdash$. Given $M$, the function $\mathbf{O}^{\vdash}$ returns $\mathbf{O}(M')$, where, for some basis $\Sigma$, $\Sigma \vdash M'$ is an optimal faithful $\vdash$-typing of $M$. This mapping is idempotent, i.e., the following property holds.

**Theorem 6** *For every PCFP term M,* $\mathbf{O}^{\vdash}(M) = \mathbf{O}^{\vdash}(\mathbf{O}^{\vdash}(M))$.

The output of $\mathbf{O}^{\vdash}$ can be further simplified by applying a clean-up procedure, like the one described in [4], which eliminates some of the place-holders $\mathsf{d}$. For instance, the simplified term $S$ of Example 5 can be replaced by the observationally equivalent term $T = (\mathsf{fix}\, f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.U)(\lambda v^{\mathsf{nat}}.v)Q_1Q_3$, where $U = \lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\mathsf{ifz}\ z\ \mathsf{then}\ gx\ \mathsf{else}\ f(\lambda u^{\mathsf{nat}}.3)\mathsf{d}_1^{\mathsf{nat}}(-\langle z, 1\rangle)$, where the useless-parameter $y$ and the place-holders $\mathsf{d}_2$ and $\mathsf{d}_3$ have been removed.

#### Acknowledgments

I thank Stefano Berardi, Mario Coppo, Viviana Bono and the anonymous referees for comments and suggestions.

## References

[1] S. Berardi. Pruning Simply Typed Lambda Terms. *Journal of Logic and Computation*, 6(5):663–681, 1996.

[2] S. Berardi and L. Boerio. Using Subtyping in Program Optimization. In *TLCA'95*, LNCS 902. Springer, 1995.

[3] F. Damiani. *Non-standard type inference for functional programs*. PhD thesis, Dipartimento di Informatica, Università di Torino, February 1998.

[4] F. Damiani and P. Giannini. Automatic useless-code elimination for HOT functional programs. *Journal of Functional programming*. To appear.

[5] F. Damiani and F. Prost. Detecting and Removing Dead Code using Rank 2 Intersection. In *TYPES'96*, LNCS 1512, pages 66–87. Springer, 1998.

[6] A. Fischbach and J. Hannan. Type Systems and Algoritms for Useless-Variable Elimination, 1999. Submitted.

[7] N. Kobayashi. Type-Based Useless Variable Elimination. In *PEPM'00*. ACM, 2000. To appear.

[8] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223–255, 1977.

[9] O. Shivers. *Control Flow Analysis of Higher-Order Languages*. PhD thesis, Carnegie-Mellon University, 1991.

[10] Y. Takayama. Extraction of Redundancy-free Programs from Constructive Natural Deduction Proofs. *Journal of Symbolic Computation*, 12:29–69, 1991.

[11] M. Wand and I. Siveroni. Constraints Systems for Useless Variable Elimination. In *POPL'99*, pages 291–302. ACM, 1999.

**Ferruccio Damiani** is with the Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149 Torino, Italy. E-mail: damiani@di.unito.it.