Automatic dead-code detection and elimination for typed functional programs: an introduction

Ferruccio Damiani and Paola Giannini

Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino (Italy),
Fax. (+39)-011-75.16.03, Tel. (+39)-011-670.67.11,
E-mail: {damiani,giannini}@di.unito.it

Abstract

In this paper we present a type inference system for the detection and elimination of dead-code in higher-order typed functional programs. Type inference can be performed in an efficient and complete way, by reducing it to the solution of a system of inequalities. The main application of the technique is the optimization of programs extracted from proofs in logical frameworks but it could be used as well in the elimination of dead-code determined by program transformations.

1 Introduction

Dead-code analysis for typed functional programming languages has been mainly studied in the context of *logical frameworks*, like Coq [BBC+98] (see [Pfe96] for a short survey on logical frameworks), to remove redundant code from functional programs extracted from formal proofs (see [Pau89b, Pau89a] for an introduction to the subject). In fact, programs extracted from proofs usually contain large parts that are useless for the computation of the final result (i.e. dead-code) and some sort of simplification is mandatory. To this aim various simplification techniques have been developed in the last ten years, see for instance [Tak91, Ber96, Boe95, BB95, BB97]. More in general dead-code, even though is unlikely to occur in human produced code, could be introduced by program transformations or could be present in programs in which general-purpose functions (taken, for instance, from standard libraries) are used in particular contexts.

In this paper we present a non-standard type assignment system and a simplification mapping for detecting and removing dead-code in simply typed higher-order functional programs.

In the non-standard type inference approach to program analysis, properties are represented by particular types (called *non-standard/annotated/refinement* types), and program analyses are presented via a system of logical inference rules. These systems are usually built by two sets of rules. One set of rules (the "subtyping" rules) determines an entailment relation between the properties under investigation, and another set of rules (the "type assignment" rules) defines the way in which properties are assigned to the terms of the language. Non-standard type systems tailored to a specific analysis, such as strictness, totality, binding-time etc. have been introduced (e.g. [KM89, Jen92, Ben92, Wri92, HL94, Sol95, DHM95, Mos97, BS95, Dam98]).

The basic idea (inspired by the technique of [Ber93, Ber96]) of the dead-code detection and elimination techniques presented in this paper is to take a typed functional program and to decorate it by non-standard types (that we call *evaluation types*) that are used to prove that the program can be evaluated without evaluating some of its subexpressions (which are therefore dead-code). For the programs of type nat (the type of natural numbers) there are two evaluation types:

- δ^{nat}, which is the property of the terms of type nat that can be evaluated, and so they can only be replaced by terms with the same behaviour, and
- ω^{nat}, which is the property of the terms of type nat that are not evaluated, and so they can be replaced by any term of the same type.

Evaluation types for programs of higher types can be constructed from these basic evaluation types using the standard type constructors. For instance the evaluation type $\omega^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$ is the property of all the functions of type $\mathsf{nat} \to \mathsf{nat}$ that, when applied to some argument, can be evaluated without using the argument (like $\lambda x^{\mathsf{nat}}.Q$ where x does not occur in Q). In other words, $\omega^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$ characterizes all the (necessarily constant) functions of type $\mathsf{nat} \to \mathsf{nat}$ that do not use their argument. The evaluation type $\delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$, instead, does not give any information about dead-code, since it says that the application of a function to an argument can be evaluated if the argument can be evaluated. To detect the dead-code in a given term M we will assume for it an evaluation type containing only δ's (i.e. we assume that M can be be evaluated) and we will try to assign evaluation types containing only ω's to as many subterms of M as possible, i.e., to detect as many useless subterms as possible. Let us look at a couple of simple examples of dead-code detection and elimination.

Example 1.1 • *Let $M = (\lambda x^{\mathsf{nat}}.3)P$ where P is a term of type nat. Since x is never used in the body of the λ-abstraction $F = \lambda x^{\mathsf{nat}}.3$, we have that F has the property $\omega^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$. This implies that P is not used in the computation of the value of M and could be replaced by any term of the same type. In fact, in a call-by-name language, M behaves like the term $M' = (\lambda x^{\mathsf{nat}}.3)\mathsf{d}$, where d is a place-holder for the dead-code removed.*

Note that, in this case, it is possible to remove the dead-code in a more substantial way, i.e. replacing M with the body of the λ-abstraction (the constant 3).

- *Take*

$$N = (\lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, Q_2\rangle)P$$

for some term Q_1, Q_2, and P of type nat such that Q_1 does not contain any occurrence of x. Since the projection proj_1 returns the first component of the pair, the term Q_2 is dead-code. Moreover, since Q_1 does not contain any occurrence of x, the λ-abstraction $G = \lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, Q_2\rangle$ has the property $\omega^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$. This implies that also P is dead-code. So, in a call-by-name language, N behaves like the term

$$N' = (\lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, \mathsf{d}\rangle)\mathsf{d}.$$

Also in this case it is possible to perform a clever simplification, i.e. N' can be replaced by the simpler term Q_1.

The soundness of the evaluation type assignment system and of the induced program transformations is proved via a partial equivalence relation semantics of the evaluation types, showing that the simplified programs are observationally equivalent to the original ones.

In Section 2 of this paper we introduce the programming language we are dealing with and its operational semantics. Section 3 shows how program properties can be represented by partial equivalence relations on a term model of the programming language.

In Section 4 we describe the language of evaluation types and its semantics, we also give a complete axiomatization of the logical implication between evaluation types. Section 5 presents an evaluation type assignment system and a program simplification based on the information provided by the evaluation types assigned to a program and to its subexpressions. It is also proved that a program and its simplified version are observationally equivalent.

$$
\begin{array}{ll}
(\mathrm{Var}) \vdash x^{\rho} : \rho & (\mathrm{Con}) \vdash \mathsf{k}^{\rho} : \rho \\[2ex]
(\to \mathrm{I})\ \dfrac{\vdash M : \sigma}{\vdash \lambda x^{\rho}.M : \rho \to \sigma} & (\to \mathrm{E})\ \dfrac{\vdash M : \rho \to \sigma \quad \vdash N : \rho}{\vdash MN : \sigma} \\[2ex]
(\times \mathrm{I})\ \dfrac{\vdash M_1 : \rho_1 \quad \vdash M_2 : \rho_2}{\vdash \langle M_1, M_2 \rangle : \rho_1 \times \rho_2} & (\times \mathrm{E}_{1 \leq i \leq 2})\ \dfrac{\vdash M : \rho_1 \times \rho_2}{\vdash \mathsf{proj}_i M : \rho_i}\ i \in \{1, 2\} \\[2ex]
(\mathrm{Fix})\ \dfrac{\vdash M : \rho}{\vdash \mathsf{fix}\, x^{\rho}.M : \rho} & (\mathrm{Ifz})\ \dfrac{\vdash N : \mathsf{nat} \quad \vdash M_1 : \rho \quad \vdash M_2 : \rho}{\vdash \mathsf{ifz}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \rho}
\end{array}
$$

Figure 1: Rules for PCFP term formation (system $\vdash_{\mathbf{T}}$)

2 The language PCFP

In this section we introduce a simple functional programming language and its operational semantics. We use the acronym PCFP (which stands for "Programming Computable Functions with Pairs") for this language since it is the dialect of the language PCF [Plo77] obtained by using natural numbers instead of integers and adding a type constructor for pairs.

2.1 PCFP syntax

The set of PCFP types is defined assuming as *ground* type the set of natural numbers, nat.

Definition 2.1 (PCFP types) *The language of* types *($\mathbf{T}$) is defined by the following grammar:* $\rho ::= \mathsf{nat} \mid \rho \to \rho \mid \rho \times \rho$ *.*

The metavariables ρ, σ, and τ range over types. PCFP terms are defined from a set of typed *term constants*

$$\mathcal{K} = \{0^{\mathsf{nat}},\ 1^{\mathsf{nat}},\ \ldots,\ \mathsf{succ}^{\mathsf{nat}\to\mathsf{nat}},\ \mathsf{pred}^{\mathsf{nat}\to\mathsf{nat}},\ +^{\mathsf{nat}\times\mathsf{nat}\to\mathsf{nat}},\ -^{\mathsf{nat}\times\mathsf{nat}\to\mathsf{nat}}\}\ ,$$

(ranged over by k), and a set $\mathcal{V}$ of typed *term variables* (ranged over by $x^{\rho}, y^{\sigma}, \ldots$). The type of a constant k is denoted by $\mathbf{T}(\mathsf{k})$. PCFP terms, ranged over by M, N, ..., are defined as follows.

Definition 2.2 (PCFP terms) *We write $\vdash_{\mathbf{T}} M : \rho$, and say that M is a term of type ρ, if $\vdash M : \rho$ is derivable by the rules in Fig. 1.*

According to Definition 2.2, in a PCFP term M the types of variables and constants are explicitly mentioned. In the following we often omit to write types which are clear from the context. The finite set of the *free variables* of a term M, denoted by $\mathrm{FV}(M)$, is defined in the standard way. In the following we take PCFP terms to be α-equivalence classes of syntax tree, i.e. we will identify terms modulo renaming of the bound variables.

As usual 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 supposed.

Let $\Lambda_{\mathbf{T}}$ be the set of PCFP terms, i.e. $\Lambda_{\mathbf{T}} = \{M \mid \vdash_{\mathbf{T}} M : \rho$ for some type $\rho\}$, and $\Lambda_{\mathbf{T}}^{\mathsf{c}}$ be the set of the *closed* terms, i.e. $\Lambda_{\mathbf{T}}^{\mathsf{c}} = \{M \mid M \in \Lambda_{\mathbf{T}}$ and $\mathrm{FV}(M) = \emptyset\}$.

2.2 PCFP evaluation rules

The process of evaluating a program is specified in a standard way by giving a *structural operational semantics* (see [Plo81, Kah88]) in the form of an inductively defined *evaluation relation*, $M \Downarrow K$,

$$(\mathrm{CAN})\ \frac{K \in \mathbf{V_T}}{K \Downarrow K}$$

$$(\mathrm{APP})\ \frac{M \Downarrow \lambda x.P \quad P[x := N] \Downarrow K}{MN \Downarrow K}$$

$$(\mathcal{K}\mathrm{APP}_1)\ \frac{M \Downarrow \mathsf{k} \quad N \Downarrow \mathsf{k}_1}{MN \Downarrow \mathsf{k}_2}\ (\mathsf{k}_1, \mathsf{k}_2) \in \mathbf{mean}(\mathsf{k})$$

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

$$(\mathrm{PROJ}_{1 \leq i \leq 2})\ \frac{P \Downarrow \langle M_1, M_2 \rangle \quad M_i \Downarrow K}{\mathsf{proj}_i P \Downarrow K}\ i \in \{1, 2\}$$

$$(\mathrm{FIX})\ \frac{M[x := \mathsf{fix}\, x.M] \Downarrow K}{\mathsf{fix}\, x.M \Downarrow K}$$

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

Figure 2: "Natural semantics" evaluation rules

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*

$$\mathbf{V_T} = \mathcal{K} \cup \{\lambda x^\rho.N \mid \lambda x^\rho.N \in \Lambda^{\mathbf{c}}_{\mathbf{T}}\} \cup \{\langle M_1, M_2 \rangle \mid \langle M_1, M_2 \rangle \in \Lambda^{\mathbf{c}}_{\mathbf{T}}\}.$$

Any PCFP constant has either type nat or $\mathsf{nat} \to \mathsf{nat}$ or $\mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$. The meaning of a functional constant k is given by a set $\mathbf{mean}(\mathsf{k})$ of pairs, such that if $(K, \mathsf{k}_1) \in \mathbf{mean}(\mathsf{k})$ then $\mathsf{k}K$ evaluates to k_1. For example $(5, 6) \in \mathbf{mean}(\mathsf{succ})$ and $(\langle 1, 3 \rangle, 4) \in \mathbf{mean}(+)$.

Definition 2.3 (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.*

Let $M \Downarrow$, to be read "M is convergent", mean that, for some K, $M \Downarrow K$, and let $M \Uparrow$, to be read "M is divergent", mean that, for no K, $M \Downarrow K$.

2.3 Ground contextual equivalence and bisimilarity

Two expressions M and N of a programming language are equivalent if any occurrence of M in a complete program can be replaced by N, and viceversa, without changing the results of program execution. To formalize this for a particular programming language is necessary to specify which are the observable results of program execution.

Following [Pit97] we introduce the *ground contextual equivalence* on PCFP terms, which is the congruence on terms induced by the contextual preorder that compares the termination behaviour of programs just at the ground type nat. This amounts to assume that *complete PCFP programs* are closed terms of type nat, and that the only observable behaviour of a complete program P is its divergence or converge to some natural number.

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

Definition 2.4 (Ground contextual equivalence) *Let M and N be terms of type ρ. Define $M \preceq^{\mathrm{gnd}}_{\mathrm{obs}} N$ whenever, for all contexts $(\mathcal{C}[\ ]^\rho)^{\mathsf{nat}}$, if $\mathcal{C}[M]$ and $\mathcal{C}[N]$ are closed terms, then $\mathcal{C}[M] \Downarrow$*

(bis 1a)	$(M \; \mathcal{B}_{\mathsf{nat}} \; N \text{ and } M \Downarrow \mathsf{k})$ implies $N \Downarrow \mathsf{k}$
(bis 1b)	$(M \; \mathcal{B}_{\mathsf{nat}} \; N \text{ and } N \Downarrow \mathsf{k})$ implies $M \Downarrow \mathsf{k}$
(bis 2)	$M \; \mathcal{B}_{\rho\to\sigma} \; N$ implies, for all P such that $\vdash_{\mathbf{T}} P : \rho$, $MP \; \mathcal{B}_\sigma \; NP$
(bis 3)	$M \; \mathcal{B}_{\rho\times\sigma} \; N$ implies $(\mathsf{proj}_1(M) \; \mathcal{B}_\rho \; \mathsf{proj}_1(N)$ and $\mathsf{proj}_2(M) \; \mathcal{B}_\sigma \; \mathsf{proj}_2(N))$

Figure 3: Bisimulation conditions for PCFP

implies $\mathcal{C}[N] \Downarrow$. *The relation* $\preceq^{\mathrm{gnd}}_{\mathrm{obs}}$ *is the* ground contextual preorder *and the equivalence induced by* $\preceq^{\mathrm{gnd}}_{\mathrm{obs}}$, *denoted by* $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$, *is the* ground observational equivalence.

In [Pit97] a co-inductive characterization of ground contextual equivalence in terms of bisimulation is given. This characterization allow to prove more easily the equivalence results on terms needed to show that the optimizations presented in this paper are correct. We give here the definition of bisimulation for PCFP, and state the two main results presented in [Pit97]. Such results will be used in this paper.

Definition 2.5 (PCFP bisimulations and bisimilarity) *A* PCFP bisimulation $\mathcal{B}$ *is a type indexed family of relations on closed terms,*

$$\mathcal{B}_\rho \subseteq \{\langle M, N\rangle \mid \vdash_{\mathbf{T}} M : \rho, \vdash_{\mathbf{T}} N : \rho \text{ and } \mathrm{FV}(M) \cup \mathrm{FV}(N) = \emptyset\}$$

($\rho \in \mathbf{T}$), satisfying the conditions in Fig. 3. PCFP bisimilarity *is the largest bisimulation and will be denoted by* $\simeq$.

A PCFP bisimulation $\mathcal{B}$ can be extended to a relation on (possibly) open terms $\mathcal{B}^{\mathbf{o}}$ (called the *open extension* of $\mathcal{B}$). For any PCFP terms P and Q such that $\mathrm{FV}(P) \cup \mathrm{FV}(Q) \subseteq \{x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\}$, define: $M \; \mathcal{B}^{\mathbf{o}}_\rho \; N$ if and only if, for every substitution $[x_1^{\sigma_1} := N_1, \ldots, x_n^{\sigma_n} := N_n]$ such that $\bigcup_{1\le i\le n} \mathrm{FV}(N_i) = \emptyset$,

$$P[\vec{x} := \vec{N}] \; \mathcal{B}_\rho \; Q[\vec{x} := \vec{N}].$$

Let $\simeq^{\mathbf{o}}$ denote the *PCFP open bisimilarity.* The following results hold.

Theorem 2.6 (Operational extensionality for PCFP) *Ground contextual equivalence coincides with bisimilarity, that is:*

$$M \simeq^{\mathrm{gnd}}_{\mathrm{obs}} N \;\; \textit{if and only if} \;\; M \simeq^{\mathbf{o}} N.$$

Theorem 2.7 (A "compactness" property of evaluation) *For each natural number n, define* $\mathsf{fix}^{(n)}\, x.M$ *as follows:*

$$\begin{array}{lcl} \mathsf{fix}^{(0)}\, x.M & = & \mathsf{fix}\, z.z \\ \mathsf{fix}^{(n+1)}\, x.M & = & M[x := \mathsf{fix}^{(n)}\, x.M] \ . \end{array}$$

For every PCFP context of type nat, $(\mathcal{C}[\,]^\rho)^{\mathsf{nat}}$, *if* $(\mathcal{C}[\mathsf{fix}\, x.M]^\rho)^{\mathsf{nat}} \Downarrow \mathsf{k}$ *then exists* $m \ge 0$ *such that* $(\mathcal{C}[\mathsf{fix}^{(m)}\, x.M]^\rho)^{\mathsf{nat}} \Downarrow \mathsf{k}$.

2.4 A closed term model of PCFP

Let $\mathbf{I}(\rho)$ denote the set of the equivalence classes of the relation $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$ on the closed terms of type ρ in $\Lambda^{\mathbf{c}}_{\mathbf{T}}$, and let $[M]$ denote the equivalence class of the closed term M w.r.t. the relation $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$. The preorder $\preceq^{\mathrm{gnd}}_{\mathrm{obs}}$ induces a partial order on $\mathbf{I}(\rho)$: define $[M] \preceq^{\mathrm{gnd}}_{\mathrm{obs}} [N]$ if and only if $M \preceq^{\mathrm{gnd}}_{\mathrm{obs}} N$.

The *closed term model* $\mathcal{M}^{\text{gnd}}$ of PCFP is defined by interpreting each type ρ as the poset $\mathbf{I}(\rho)$ ordered by $\preceq^{\text{gnd}}_{\text{obs}}$. For every type ρ, $[\mathsf{fix}\, x^\rho.x]$ (the equivalence class of the canonical divergent element, which represents the notion of endless computation), is the least element, w.r.t. $\preceq^{\text{gnd}}_{\text{obs}}$, of $\mathbf{I}(\rho)$. An *environment* is a mapping $e : \mathcal{V} \to \bigcup_{\rho \in \mathbf{T}} \mathbf{I}(\rho)$ which respects types, i.e. a mapping such that, for all x^ρ, $e(x^\rho) \in \mathbf{I}(\rho)$. The interpretation of a term M in an environment e is defined in a standard way 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, using Theorem 2.6, we can prove that $\mathcal{M}^{\text{gnd}}$ is an extensional model.

3 Partial equivalence relations and program properties

In this section we show how some properties of PCFP programs P of type ρ can be represented as partial equivalence relations over the interpretation, $\mathbf{I}(\rho)$, of the type ρ in the closed term model $\mathcal{M}^{\text{gnd}}$.

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:

$$\text{Dom}(\mathcal{R}) = \{E \mid (E, E') \in \mathcal{R} \text{ or } (E', E) \in \mathcal{R} \text{ for some } E'\} \; .$$

Note that $\mathcal{R}$ is reflexive on its domain. In the following let "p.e.r. over a type ρ" mean "p.e.r. over the set $\mathbf{I}(\rho)$". The language of program properties $\mathbf{L}$ (introduced in Section 4 as a language of non-standard types over $\mathbf{T}$) which is at the basis of the program analysis and transformation techniques proposed in this paper, will be interpreted as a subset of the properties represented by p.e.r.s over types $\rho \in \mathbf{T}$. The property expressed by $\mathcal{R}$ is satisfied by the elements of $\text{Dom}(\mathcal{R})$. Moreover elements in the same equivalence class are indistinguishable w.r.t. the property expressed by $\mathcal{R}$.

- Consider the diagonal p.e.r. over $\mathbf{I}(\rho)$,

  $$\Delta^\rho = \{([M], [M]) \mid [M] \in \mathbf{I}(\rho)\}.$$

  If we look at Δ^ρ, as a relation between terms, it can be seen as the property satisfied by any term M of type ρ such that its value *matters* (can be evaluated), since is satisfied by all the terms of type ρ, and the terms that are indistinguishable are the terms ground observationally equivalent.

- For every type $\rho \in \mathbf{T}$, the trivial p.e.r. over $\mathbf{I}(\rho)$

  $$\Omega^\rho = \{([M], [N]) \mid [M], [N] \in \mathbf{I}(\rho)\},$$

  relates all the elements of $\mathbf{I}(\rho)$. Ω^ρ can be seen as the "true" property (satisfied by every term of type ρ) giving *no information* about terms except their type.

- Given a p.e.r. $\mathcal{R}_1$ over ρ_1 and a p.e.r. $\mathcal{R}_2$ over ρ_2, let $\mathcal{R}_1 \times \mathcal{R}_2$ be the p.e.r. over $\rho_1 \times \rho_2$ defined as follows:

  $$\mathcal{R}_1 \times \mathcal{R}_2 \quad = \quad \{([M], [N]) \mid \forall i \in \{1, 2\}.([\mathsf{proj}_i M], [\mathsf{proj}_i N]) \in \mathcal{R}_i\}.$$

  $\mathcal{R}_1 \times \mathcal{R}_2$ is the property of the programs P such that, for $i \in \{1, 2\}$, the program $\mathsf{proj}_i P$ has the property $\mathcal{R}_i$. For instance $\Delta^{\mathsf{nat}} \times \Delta^{\mathsf{nat}}$ is the the property (satisfied by all the closed terms of type $\mathsf{nat} \times \mathsf{nat}$) of all the terms P such that both the values of $\mathsf{proj}_1 P$ and $\mathsf{proj}_2 P$ can be evaluated. Note that, for any program P, having a property of the form $\mathcal{R}_1 \times \mathcal{R}_2$ does not imply that $P \Downarrow \langle P_1, P_2 \rangle$, for some P_1 and P_2. (In particular this is not the case when $\mathcal{R}_1 = \Omega^{\rho_1}$ and $\mathcal{R}_2 = \Omega^{\rho_2}$).

  It is easy to show that

– $\Delta^{\rho_1} \times\!\!\times \Delta^{\rho_2} = \Delta^{\rho_1 \times \rho_2}$, and
– $\Omega^{\rho_1} \times\!\!\times \Omega^{\rho_2} = \Omega^{\rho_1 \times \rho_2}$.

- Given a p.e.r. $\mathcal{R}_1$ over ρ_1 and a p.e.r. $\mathcal{R}_2$ over ρ_2, let $\mathcal{R}_1 \twoheadrightarrow \mathcal{R}_2$ be the p.e.r. over $\rho_1 \to \rho_2$ defined as follows:

$$\mathcal{R}_1 \twoheadrightarrow \mathcal{R}_2 \quad = \quad \{([F],[G]) \mid \forall([M],[N]) \in \mathcal{R}_1.([FM],[GN]) \in \mathcal{R}_2\}.$$

$\mathcal{R}_1 \twoheadrightarrow \mathcal{R}_2$ is the property of the programs F such that, for every program M having the property $\mathcal{R}_1$, the program FM has the property $\mathcal{R}_2$. For instance $\Omega^{\mathsf{nat}} \to \Delta^{\mathsf{nat}}$ is the property satisfied by the closed terms of type $\mathsf{nat} \to \mathsf{nat}$ which represent constant functions, since the result of the function matters but it is independent of its input.

According to the $\mathcal{M}^{\mathrm{gnd}}$ semantics we have 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 all the p.e.r.s $\mathcal{R}_1$ over ρ_1).

Example 3.1 *The term $G = \lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\langle x, x\rangle$ has the property*

$$\Delta^{\mathsf{nat}} \twoheadrightarrow \Omega^{\mathsf{nat}} \twoheadrightarrow (\Delta^{\mathsf{nat}} \times\!\!\times \Delta^{\mathsf{nat}}) \ ,$$

according to the fact that the application of the function to two arguments does not involve the use of the second argument.

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

Definition 3.2 (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. We say that a program P of type ρ 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 $([\![M]\!]_e, [\![M]\!]_{e'}) \in \mathcal{R}$.*

The set-theoretic inclusion between p.e.r.s over a type ρ 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 ρ, $\mathcal{R} \subseteq \Omega^{\rho}$, according to the fact that Ω^{ρ} represents the "true" property.

4 Evaluation types

In this section we introduce a language of non-standard types over $\mathbf{T}$, the language of the *evaluation types* (*e-types* for short), which is at the heart of the program analyses and transformations technique presented in this paper. We first introduce the e-types syntax and semantics and then a complete axiomatization of the logical implication between e-types.

4.1 E-types syntax and semantics

Let ϕ range over e-types and ϕ^{ρ} range over e-types with *underlying type* ρ. In the following we will often omit the superscript ρ when it is either not relevant or clear from the context.

Definition 4.1 (Evaluation types) *The language of the e-types, $\mathbf{L}$, is defined by: $\mathbf{L} = \cup_{\rho \in \mathbf{T}} \mathbf{L}(\rho)$, where the sets $\mathbf{L}(\rho)$ are defined by the rules in Fig. 4. The symbols δ and ω are called* basic properties.

$$(\delta)\ \delta^{\mathsf{nat}} \in \mathbf{L}(\mathsf{nat}) \qquad (\omega)\ \omega^{\rho} \in \mathbf{L}(\rho)$$

$$(\to)\ \frac{\phi \in \mathbf{L}(\rho) \quad \psi \in \mathbf{L}(\sigma)}{\phi \to \psi \in \mathbf{L}(\rho \to \sigma)}\ \psi \neq \omega^{\sigma}$$

$$(\times)\ \frac{\phi_1 \in \mathbf{L}(\rho_1) \quad \phi_2 \in \mathbf{L}(\rho_2)}{\phi_1 \times \phi_2 \in \mathbf{L}(\rho_1 \times \rho_2)}\ \exists i \in \{1,2\}.\phi_i \neq \omega^{\rho_i}$$

Figure 4: Evaluation types

The meaning of e-types is given by interpreting each e-type, ϕ^{ρ}, as a p.e.r., $[\![\phi^{\rho}]\!]$ over the interpretation of the type ρ in the model $\mathcal{M}^{\mathrm{gnd}}$ (see Section 3).

Definition 4.2 (Semantics of e-types) *The interpretation $[\![\phi]\!]$ of an e-type ϕ is defined by:*

$$\begin{aligned}
[\![\delta^{\mathsf{nat}}]\!] &= \Delta^{\mathsf{nat}} \\
[\![\omega^{\rho}]\!] &= \Omega^{\rho} \\
[\![\phi \to \psi]\!] &= [\![\phi]\!] \twoheadrightarrow [\![\psi]\!] \\
[\![\phi_1 \times \phi_2]\!] &= [\![\phi_1]\!] \mathbin{\times\!\!\!\!\times} [\![\phi_2]\!] \ .
\end{aligned}$$

The previous definition justifies the choice of not having e-types of the form

- $\phi^{\rho_1} \to \omega^{\rho_2}$ and $\omega^{\rho_1} \times \omega^{\rho_2}$ (where ϕ is any e-type, and ρ_1 and ρ_2 are any types), since the first would denote the equivalence relation $\Omega^{\rho_1 \to \rho_2}$, and the second would denote $\Omega^{\rho_1 \times \rho_2}$, and
- δ^{ρ} where $\rho \neq \mathsf{nat}$, since $\Delta^{\rho_1} \twoheadrightarrow \Delta^{\rho_2} = \Delta^{\rho_1 \to \rho_2}$ and $\Delta^{\rho_1} \mathbin{\times\!\!\!\!\times} \Delta^{\rho_2} = \Delta^{\rho_1 \times \rho_2}$ for all ρ_1 and ρ_2.

Indeed the e-type syntax has been defined in such a way that (syntactically) different e-types denote different p.e.r.s[1].

According to Definition 4.2, an e-type ω^{ρ} denotes the trivial p.e.r. Ω^{ρ}. Other "special" e-types are the δ-e-types defined as follows.

Definition 4.3 (ω-e-types and δ-e-types) *1. We call ω-e-types the e-types in the set $\mathbf{L}_{\omega} = \{\omega^{\rho} \mid \rho \in \mathbf{T}\}$.*

2. The set of the δ-e-types ($\mathbf{L}_{\delta}$) is the subset of the e-types which do not contain subexpressions of the form ω^{ρ} (for some ρ). For every type $\rho \in \mathbf{T}$, let $\delta(\rho)$ be the unique δ-e-type of type ρ.

The following proposition shows that ω-e-types provide a a syntactical characterization for the set of the e-types ϕ^{ρ} such that $[\![\phi^{\rho}]\!] = \Omega^{\rho}$ and δ-e-types for the set of the e-types ϕ^{ρ} such that $[\![\phi^{\rho}]\!] = \Delta^{\rho}$.

Proposition 4.4 *1. $\phi^{\rho} \in \mathbf{L}_{\omega}$ (i.e. $\phi = \omega^{\rho}$) if and only if $[\![\phi^{\rho}]\!] = \Omega^{\rho}$, and*

2. $\phi^{\rho} \in \mathbf{L}_{\delta}$ if and only if $[\![\phi^{\rho}]\!] = \Delta^{\rho}$.

Proof.

[1]As it will be clear after Definition 4.6 and Theorem 4.9.

1. ($\Longrightarrow$). Immediate from Definition 4.2.
($\Longleftarrow$). Let $[\![\phi^\rho]\!] = \Omega^\rho$. By structural induction on $\phi \in \mathbf{L}$. The cases $\phi = \delta^{\mathsf{nat}}$ and $\phi = \omega^\rho$, are immediate by Definition 4.2.
Let $\phi^\rho = {\phi_1}^{\rho_1} \to {\phi_2}^{\rho_2}$, by definition $\phi_2 \notin \mathbf{L}_\omega$. Let $[\![\phi]\!] = \Omega^\rho$, by induction hypothesis $[\![\phi_2]\!] \neq \Omega^{\rho_2}$ So there are $[P], [Q] \in \mathbf{I}(\rho_2)$ such that $([P], [Q]) \notin [\![\phi_2]\!]$. But this would imply that $([\lambda z^{\rho_1}.P], [\lambda z^{\rho_1}.Q]) \notin [\![\phi_1 \to \phi_2]\!]$ which is absurd.
For $\phi^\rho = {\phi_1}^{\rho_1} \times {\phi_2}^{\rho_2}$, by definition either $[\![\phi_1]\!] \notin \mathbf{L}_\omega$ or $[\![\phi_2]\!] \notin \mathbf{L}_\omega$. However $[\![\phi^\rho]\!] = \Omega^\rho$ implies that both $[\![\phi_1]\!] = \Omega^{\rho_1}$ and $[\![\phi_2]\!] = \Omega^{\rho_2}$, which is absurd.

2. ($\Longrightarrow$). We prove that, for all $\rho \in \mathbf{T}$, $[\![\delta(\rho)]\!] = \Delta^\rho$. By induction on $\rho \in \mathbf{T}$. The case $\rho = \mathsf{nat}$ is immediate.

   Let $\rho = \rho_1 \to \rho_2$. By induction hypothesis both $[\![\delta(\rho_1)]\!] = \Delta^{\rho_1}$ and $[\![\delta(\rho_2)]\!] = \Delta^{\rho_2}$. By definition of $[\![\delta(\rho_1 \to \rho_2)]\!]$ we have that

   (∗) $([P], [Q] \in [\![\delta(\rho_1 \to \rho_2)]\!]$ if and only if, for all M and N such that $([M], [N]) \in [\![\delta(\rho_1)]\!]$, $([P\,M], [Q\,N]) \in [\![\delta(\rho_2)]\!]$.

   We first show that

   (a) $([P], [Q]) \in [\![\delta(\rho_1 \to \rho_2)]\!]$ implies $P \simeq Q$, and
   (b) $P \simeq Q$ implies $([P], [Q]) \in [\![\delta(\rho_1 \to \rho_2)]\!]$.

   The first implication is proved by absurd. If $P \not\simeq Q$, then, by definition of bisimilarity there would be an M such that $P\,M \not\simeq Q\,M$. So (by induction hypothesis) $([P\,M], [Q\,M]) \notin [\![\delta(\rho_2)]\!]$, and therefore (from (∗)) $([P], [Q]) \notin [\![\delta(\rho_1 \to \rho_2)]\!]$. For the second one, let $P \simeq Q$. This implies that, for all R and S such that $R \simeq S$, $P\,R \simeq P\,S \simeq Q\,S \simeq Q\,R$. So (by induction hypothesis and (∗)) $([P], [Q]) \in [\![\delta(\rho_1 \to \rho_2)]\!]$.
   Therefore from the equivalence between $\simeq$ and contextual equivalence (Theorem 2.6) we get the result.

   The case $\rho = \rho_1 \times \rho_2$ is similar.
   ($\Longleftarrow$). Assume that $[\![\phi^\rho]\!] = \Delta^\rho$. From ($\Longrightarrow$) $\Delta^\rho = [\![\delta(\rho)]\!]$. So Theorem 4.9[2] implies $\phi^\rho \cong \delta(\rho)$, and Definition 4.6 implies $\phi^\rho = \delta(\rho)$. □

The previous proposition shows that the family of relations $\{(M, N) \mid ([M], [N]) \in [\![\delta(\rho)]\!]\}_{\rho \in \mathbf{T}}$ is the PCFP bisimilarity. Another useful property of the e-types semantics is the following.

Proposition 4.5 *If, for all $n \geq 0$, $([(\mathcal{C}_1[\mathsf{fix}^{(n)}\, x.M]^\rho)^\sigma], [(\mathcal{C}_2[\mathsf{fix}^{(n)}\, x.M]^\rho)^\sigma]) \in [\![\phi^\sigma]\!]$ then $([(\mathcal{C}_1[\mathsf{fix}\, x.M]^\rho)^\sigma], [(\mathcal{C}_2[\mathsf{fix}\, x.M]^\rho)^\sigma]) \in [\![\phi^\sigma]\!]$*

Proof. By structural induction on ϕ^σ. The case $\phi^\sigma = \omega^\sigma$ is immediate. For the case $\phi^\sigma = \delta^{\mathsf{nat}}$, by contraposition assume that $([(\mathcal{C}_1[\mathsf{fix}\, x.M]^\rho)^{\mathsf{nat}}], [(\mathcal{C}_2[\mathsf{fix}\, x.M]^\rho)^{\mathsf{nat}}]) \notin [\![\delta^{\mathsf{nat}}]\!]$. Then, by Theorem 2.7,

1. either $(\mathcal{C}_1[\mathsf{fix}\, x.M]^\rho)^{\mathsf{nat}} \Downarrow \mathsf{k}_1$ and $(\mathcal{C}_2[\mathsf{fix}\, x.M]^\rho)^{\mathsf{nat}} \Downarrow \mathsf{k}_2$, with $[\mathsf{k}_1] \neq [\mathsf{k}_2]$,
2. or $(\mathcal{C}_1[\mathsf{fix}\, x.M]^\rho)^{\mathsf{nat}} \Downarrow \mathsf{k}_1$ and $(\mathcal{C}_2[\mathsf{fix}\, x.M]^\rho)^{\mathsf{nat}} \Uparrow$
3. or $(\mathcal{C}_1[\mathsf{fix}\, x.M]^\rho)^{\mathsf{nat}} \Uparrow$ and $(\mathcal{C}_2[\mathsf{fix}\, x.M]^\rho)^{\mathsf{nat}} \Downarrow \mathsf{k}_2$.

In each case we have that there exists $m \geq 0$ such that $([(\mathcal{C}_1[\mathsf{fix}^{(m)}\, x.M]^\rho)^{\mathsf{nat}}], [(\mathcal{C}_2[\mathsf{fix}^{(m)}\, x.M]^\rho)^{\mathsf{nat}}]) \notin [\![\phi^\sigma]\!]$.

Let $\phi^\sigma = \phi_1^{\sigma_1} \to \phi_2^{\sigma_2}$. Again by contraposition assume that $([(\mathcal{C}_1[\mathsf{fix}\, x.M]^\rho)^\sigma], [(\mathcal{C}_2[\mathsf{fix}\, x.M]^\rho)^\sigma]) \notin [\![\phi^\sigma]\!]$. Then, for some $([P], [Q]) \in [\![\phi_1^{\sigma_1}]\!]$, we have

$$([(\mathcal{C}_1[\mathsf{fix}\, x.M]^\rho)^\sigma P], [(\mathcal{C}_2[\mathsf{fix}\, x.M]^\rho)^\sigma Q]) \notin [\![\phi_2^{\sigma_2}]\!].$$

[2]The proof of Theorem 4.9 does not use Proposition 4.4.2.

$$(\mathrm{Ref})\ \phi \leq \phi \qquad\qquad (\omega)\ \phi^{\rho} \leq \omega^{\rho}$$

$$(\rightarrow)\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_2 \rightarrow \psi_1 \leq \phi_1 \rightarrow \psi_2}\ \psi_2 \notin \mathbf{L}_{\omega}$$

$$(\times)\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_1 \times \psi_1 \leq \phi_2 \times \psi_2}\ \text{either } \phi_2 \notin \mathbf{L}_{\omega} \text{or } \psi_2 \notin \mathbf{L}_{\omega}$$

Figure 5: Entailment rules for e-types (system $\leq$)

By induction hypothesis there exists $m \geq 0$ such that $([(\mathcal{C}_1[\mathsf{fix}^{(m)}\, x.M]^{\rho})^{\mathsf{nat}}P], [(\mathcal{C}_2[\mathsf{fix}^{(m)}\, x.M]^{\rho})^{\mathsf{nat}}Q]) \notin [\![\phi_2^{\sigma_2}]\!]$. Therefore $([(\mathcal{C}_1[\mathsf{fix}^{(m)}\, x.M]^{\rho})^{\mathsf{nat}}], [(\mathcal{C}_2[\mathsf{fix}^{(m)}\, x.M]^{\rho})^{\mathsf{nat}}]) \notin [\![\phi^{\sigma}]\!]$.

The proof for $\phi^{\sigma} = \phi_1^{\sigma_1} \times \phi_2^{\sigma_2}$ is similar to the previous case. □

4.2 A complete entailment relation for e-types

In this section we introduce an entailment relation between e-types, $\leq$. This relation models the set-theoretic inclusion between the interpretation of the e-types, and so it represents the logical implication between properties.

Definition 4.6 (Entailment relation $\leq$) *Let $\phi, \psi \in \mathbf{L}$. We write $\phi \leq \psi$ to mean that $\phi \leq \psi$ is derivable by the rules in Fig. 5. By $\cong$ we denote the equivalence relation induced by $\leq$.*

It is immediate to show that $\leq$ is reflexive and transitive. Moreover $\phi \cong \psi$ implies $\phi = \psi$.

The $\leq$ entailment relation between e-types is sound and complete w.r.t. the p.e.r. interpretation of Definition 4.2.

Theorem 4.7 (Soundness of $\leq$) *$\phi \leq \psi$ implies $[\![\phi]\!] \subseteq [\![\psi]\!]$.*

Proof. By induction on the derivation of the judgement $\phi \leq \psi$. □

The proof of completeness relies on the definition, for each e-type ϕ, of a set of pairs of terms, $\mathbf{Chr}^{\phi}$, which characterize the pairs of terms in $[\![\phi]\!]$. In particular the characteristic sets $[\![\phi]\!]$ have the properties expressed by the following lemma.

Lemma 4.8 (Characteristic set $\mathbf{Chr}^{\phi}$) *1. $\{([P],[Q]) \mid (P,Q) \in \mathbf{Chr}^{\phi}\} \subseteq [\![\phi]\!]$.*

2. $\{([P],[Q]) \mid (P,Q) \in \mathbf{Chr}^{\phi}\} \subseteq [\![\psi]\!]$ implies $\phi \leq \psi$.

The construction of the set $\mathbf{Chr}^{\phi}$, which is rather technical, and the proof of Lemma 4.8 are omitted (they can be found in [DG]).

Theorem 4.9 (Completeness of $\leq$) *$[\![\phi]\!] \subseteq [\![\psi]\!]$ implies $\phi \leq \psi$.*

Proof. By contraposition. Assume $\phi \not\leq \psi$. Then by Lemma 4.8.2 $\{([P],[Q]) \mid (P,Q) \in \mathbf{Chr}^{\phi}\} \not\subseteq [\![\psi]\!]$ and by Lemma 4.8.1 $\{([P],[Q]) \mid (P,Q) \in \mathbf{Chr}^{\phi}\} \subseteq [\![\phi]\!]$. Therefore $[\![\phi]\!] \not\subseteq [\![\psi]\!]$. □

5 Using e-types for detecting and removing dead-code

In this section we first introduce an e-type assignment system for detecting dead-code in PCFP programs (Section 5.1) . Then we give a mapping for removing from a program the dead-code proved by using the e-type assignment system (Section 5.2) and consider the problem of finding and removing *all* the dead-code that can be proved by the e-type assignment system (Section 5.3).

The basic idea for using e-types to detect the dead-code in a given PCFP term M of type ρ is the following: if we can prove that (under suitable assumptions on its free variables) the term has a δ-e-type while some of its subexpressions have ω-e-types, then (according to the e-type semantics) the value of M can be computed without using these subexpressions, so they are dead-code that can be removed without changing the semantics of the program.

5.1 An e-type assignment system for detecting dead-code

If x^ρ is a term variable of type ρ an assumption for x^ρ is an expression of the shape $x^\rho : \phi$, or $x : \phi$ for short, where $\phi \in \mathbf{L}(\rho)$. A basis is a set Σ of e-types assumptions for term variables.

E-types are assigned to PCFP terms by a set of inference rules for judgements of the form $\Sigma \vdash_1 M^\phi$ where M^ϕ is a *decorated* term, i.e. it has written in it (some of) the e-types assigned to its subterms. Such a decorated term can then be processed by a transformation procedure (like the one described in Section 5.2) that simplifies it according to the information supplied by the e-types.

Notation 5.1 *For any e-type $\phi \in \mathbf{L}(\rho)$ let $\epsilon(\phi)$ denote the underlying type, ρ, of ϕ. For any basis Σ let $\epsilon(\Sigma)$ denote the set of term variables in Σ, and for any decorated term M^ϕ define $\epsilon(M^\phi)$ be the term obtained from M^ϕ by erasing all the e-type decorations.*

Definition 5.2 (E-type assignment system $\vdash_1$) *A $\vdash_1$-typing statement is an expression $\Sigma \vdash_1 M^\phi$ where Σ is a basis containing an assumption for each free variable of M, and M^ϕ is a decorated version of M. $\Sigma, x : \psi$ denotes the basis $\Sigma \cup \{x : \psi\}$ where it is assumed that x does not appear in Σ. We write $\Sigma \vdash_1 M^\phi$ to mean that $\Sigma \vdash M^\phi$ can be derived by the rules in Fig. 6.*

$$(\omega)\ \frac{\epsilon(\Sigma) \supseteq \mathrm{FV}(M) \qquad \vdash_{\mathbf{T}} M : \rho}{\Sigma \vdash M^{\omega^\rho}}$$

$$(\mathrm{Var})\ \frac{\phi \leq \psi}{\Sigma, x : \phi \vdash x^\psi}\ \psi \notin \mathbf{L}_\omega \qquad (\mathrm{Con})\ \Sigma \vdash \mathsf{k}^{\delta(\mathbf{T}(\mathsf{k}))}$$

$$(\to \mathrm{I})\ \frac{\Sigma, x : \phi \vdash M^\psi}{\Sigma \vdash (\lambda x.M)^{\phi \to \psi}}\ \psi \notin \mathbf{L}_\omega \qquad (\to \mathrm{E})\ \frac{\Sigma \vdash M^{\phi \to \psi} \qquad \Sigma \vdash N^\phi}{\Sigma \vdash (MN^\phi)^\psi}$$

$$(\times \mathrm{I})\ \frac{\Sigma \vdash {M_1}^{\phi_1} \qquad \Sigma \vdash {M_2}^{\phi_2}}{\Sigma \vdash \langle M_1, M_2\rangle^{\phi_1 \times \phi_2}}\ \exists i \in \{1,2\}.\phi_i \notin \mathbf{L}_\omega$$

$$(\times \mathrm{E}_{1 \leq i \leq 2})\ \frac{\Sigma \vdash M^{\phi_1 \times \phi_2}}{\Sigma \vdash (\mathsf{proj}_i M^{\phi_{3-i}})^{\phi_i}}\ \phi_i \notin \mathbf{L}_\omega$$

$$(\mathrm{Fix})\ \frac{\Sigma, x : \phi \vdash M^\phi}{\Sigma \vdash (\mathsf{fix}\, x.M)^\phi}\ \phi \notin \mathbf{L}_\omega$$

$$(\mathrm{If})\ \frac{\Sigma \vdash N^{\delta^{\mathsf{nat}}} \qquad \Sigma \vdash {M_1}^\phi \qquad \Sigma \vdash {M_2}^\phi}{\Sigma \vdash (\mathsf{ifz}\, N \,\mathsf{then}\, M_1 \,\mathsf{else}\, M_2)^\phi}\ \phi \notin \mathbf{L}_\omega$$

Figure 6: Rules for e-type assignment (system $\vdash_1$)

Note that, being $\vdash_1$ an inference system, the same terms can have different decorations.

Example 5.3 *Take the terms M and N of Example 1.1:*

- $M = (\lambda x^{\mathsf{nat}}.3)P$, *and*
- $N = (\lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, Q_2\rangle)P$, *where* Q_1 *does not contain any occurrence of* x.

Assume for simplicity that P, Q_1 *and* Q_2 *are closed. It it easy to check that the following* $\vdash_1$*-typings hold:*

- $\emptyset \vdash_1 ((\lambda x.3)P^{\omega^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$, *and*
- $\emptyset \vdash_1 ((\lambda x.\mathsf{proj}_1\langle Q_1, Q_2\rangle^{\omega^{\mathsf{nat}}})P^{\omega^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$.

To state the soundness of the e-type assignment system w.r.t. the semantics we introduce the following definition.

Definition 5.4 *1. Two environments* e_1, e_2 *are* Σ*-related if and only if, for all* $x : \psi \in \Sigma$, $(e_1(x), e_2(x)) \in [\![\psi]\!]$.

2. Let $\Sigma \vdash_1 M^\phi$ *and* $\Sigma \vdash_1 N^\phi$. *We write* $\epsilon(M^\phi) \sim^\Sigma_\phi \epsilon(N^\phi)$ *to mean that for all* e_1, e_2, *if* e_1 *and* e_2 *are* Σ*-related, then* $([\![\epsilon(M^\phi)]\!]_{e_1}, [\![\epsilon(N^\phi)]\!]_{e_2}) \in [\![\phi]\!]$.

Now we can state the soundness of the system $\vdash_1$ w.r.t. Definitions 4.2 and 3.2.

Theorem 5.5 (Soundness of $\vdash_1$) *Let* $\Sigma \vdash_1 P^\chi$. *Then* $\epsilon(P^\chi) \sim^\Sigma_\chi \epsilon(P^\chi)$.

Proof. By induction on the derivation of $\Sigma \vdash_1 P^\chi$. We only present three cases.

- If the last rule applied is (Var), let e and e' be such that $(e(x), e'(x)) \in [\![\phi]\!]$. By Theorem 4.7 (soundness of $\leq$), $[\![\phi]\!] \subseteq [\![\psi]\!]$, so $(e(x), e'(x)) \in [\![\psi]\!]$ which proves the result.
- Let the derivation end with the rule ($\rightarrow$ I). So $P = \lambda x.M$ and $\chi = \phi \rightarrow \psi$.

  We have to prove that for every e and e' which are Σ-related $([\![\lambda x.M]\!]_e, [\![\lambda x.M]\!]_{e'}) \in [\![\phi \rightarrow \psi]\!]$, i.e. $([(\lambda x.M)[\bar{y} := \bar{Q}]], [(\lambda x.M)[\bar{y} := \bar{Q'}]]) \in [\![\phi \rightarrow \psi]\!]$, where $\bar{y} = y_1 \cdots y_k$ $(k \geq 0)$ are the variables in Σ, $e(y_i) = [Q_i]$, $e'(y_i) = [Q'_i]$, ϕ_i is the e-type of y_i in Σ, and $(e(y_i), e'(y_i)) \in [\![\phi_i]\!]$. By definition this amounts to show that, for all $([N], [N']) \in [\![\phi]\!]$, $([(\lambda x.M)[\bar{y} := \bar{Q}]N], [(\lambda x.M)[\bar{y} := \bar{Q'}]N']) \in [\![\psi]\!]$, i.e., since $\mathrm{FV}(N) = \mathrm{FV}(N') = \emptyset$, $([M[x := N, \bar{y} := \bar{Q}]], [M[x := N', \bar{y} := \bar{Q'}]]) \in [\![\psi]\!]$, that is $([\![M]\!]_{e[x:=[N]]}, [\![M]\!]_{e[x:=[N]]}) \in [\![\psi]\!]$, which follows from the induction hypothesis applied to the derivation $\Sigma, x : \phi \vdash_1 M : \psi$.
- Let the derivation end with the rule (Fix). So $P = \mathsf{fix}\, x.M$.

  We have to prove that $([\![\mathsf{fix}\, x.M]\!]_e, [\![\mathsf{fix}\, x.M]\!]_e) \in [\![\phi]\!]$, i.e. $([(\mathsf{fix}\, x.M)[\bar{y} := \bar{Q}]], [(\mathsf{fix}\, x.M)[\bar{y} := \bar{Q'}]]) \in [\![\phi]\!]$, where $\bar{y} = y_1 \cdots y_k$ $(k \geq 0)$ are the variables in Σ, $e(y_i) = [Q_i]$, $e'(y_i) = [Q'_i]$, ϕ_i is the e-type of y_i in Σ, and $(e(y_i), e'(y_i)) \in [\![\phi_i]\!]$.

  Since $\mathsf{fix}^{(0)}\, x.P = \mathsf{fix}\, z.z$ for any P, we have $([\mathsf{fix}^{(0)}\, x.M], [\mathsf{fix}^{(0)}\, x.M]) \in [\![\phi]\!]$. Assuming that $([\![\mathsf{fix}^{(n)}\, x.M]\!]_e, [\![\mathsf{fix}^{(n)}\, x.M]\!]_{e'}) \in [\![\phi]\!]$ and applying the induction hypothesis to the derivation $\Sigma, x : \phi \vdash_1 M : \phi$ we get $([\![M]\!]_{e[x:=[\![\mathsf{fix}^{(n)}\, x.M]\!]_e]}, [\![M]\!]_{e'[x:=[\![\mathsf{fix}^{(n)}\, x.M]\!]_{e'}]}) \in [\![\phi]\!]$, i.e. $([M[x := (\mathsf{fix}^{(n)}\, x.M)[\bar{y} := \bar{Q}],\ \bar{y} := \bar{Q}]], [M[x := (\mathsf{fix}^{(n)}\, x.M)[\bar{y} := \bar{Q}],\ \bar{y} := \bar{Q}]]) \in [\![\phi]\!]$, i.e. $([(\mathsf{fix}^{(n+1)}\, x.M)[\bar{y} := \bar{Q}]], [(\mathsf{fix}^{(n+1)}\, x.M)[\bar{y} := \bar{Q}]]) \in [\![\phi]\!]$, i.e. $([\![\mathsf{fix}^{(n+1)}\, x.M]\!]_e, [\![\mathsf{fix}^{(n+1)}\, x.M]\!]_{e'}) \in [\![\phi]\!]$. Then, by induction on n, we have that for all n, $([\![\mathsf{fix}^{(n)}\, x.M]\!]_e, [\![\mathsf{fix}^{(n)}\, x.M]\!]_{e'}) \in [\![\phi]\!]$. So we can conclude by Proposition 4.5 that $([\![\mathsf{fix}\, x.M]\!]_e, [\![\mathsf{fix}\, x.M]\!]_{e'}) \in [\![\phi]\!]$. □

5.2 A dead-code elimination

We now introduce a dead-code elimination mapping $\mathbf{O}_1$ that takes a $\vdash_1$-decorated term M^ϕ and returns a simplified version of it. The simplified version of the term has dummy variables in place of subterms that are not needed for the evaluation of the term.

5.2.1 Dummy variables

For each type ρ, define d^ρ, d_1^ρ, d_2^ρ, ..., be *dummy variables* of type ρ. We remark that dummy variables are not present in the original programs: they are introduced by the dead-code elimination mapping $\mathbf{O}_1$ as placeholders for the dead-code removed[3]. So in the following we assume that all the occurrences of dummy variables in a program are free (i.e. there are no bound dummy variables) and distinct (i.e. each dummy variable occurs at most once in a program).

Notation 5.6 *For every term M, let $\mathrm{DV}(M)$ be the set of the dummy variables in M.*

5.2.2 The simplification mapping

Let $\Lambda_{\mathbf{T}}^{\vdash_1}$ be the set of $\vdash_1$-decorated PCFP terms, i.e.,

$$\Lambda_{\mathbf{T}}^{\vdash_1} = \{M^\phi \mid \Sigma \vdash_1 M^\phi \text{ for some e-type } \phi \text{ and basis } \Sigma\}.$$

Definition 5.7 (Simplification mapping $\mathbf{O}_1$) *1. The function*

$$\mathbf{O}_1 : \Lambda_{\mathbf{T}}^{\vdash_1} \to \Lambda_{\mathbf{T}}^{\vdash_1}$$

is defined by the clauses in Fig. 7 , where the occurrence of "d" (second row) denotes a fresh dummy variable of the proper type.

2. If Σ is a basis then

$$\mathbf{O}_1(\Sigma) = \{x : \chi \mid x : \chi \in \Sigma \textit{ and } \chi \notin \mathbf{L}_\omega\}.$$

We have immediately that if $\Sigma \vdash_1 M^\phi$ then $\mathbf{O}_1(\Sigma) \subseteq \Sigma$ and $\Sigma' \vdash_1 \mathbf{O}_1(M^\phi)$, where $\Sigma' = \mathbf{O}_1(\Sigma) \cup \{\mathsf{d}^\sigma : \omega^\sigma \mid \mathsf{d}^\sigma \in \mathrm{DV}(\epsilon(\mathbf{O}_1(M^\phi))\}$. Moreover we have that the simplification mapping is correct w.r.t. the e-type semantics.

Proposition 5.8 (Correctness of $\mathbf{O}_1$ w.r.t. the e-types semantics) *If $\Sigma \vdash_1 M^\phi$ then $\epsilon(M^\phi) \sim_\phi^{\Sigma'} \epsilon(\mathbf{O}_1(M^\phi))$, where $\Sigma' = \Sigma \cup \{\mathsf{d}^\sigma : \omega^\sigma \mid \mathsf{d}^\sigma \in \mathrm{DV}(\epsilon(\mathbf{O}_1(M^\phi)))\}$.*

Proof. First notice that (by definition of $\mathbf{O}_1$) every dummy variable in $\mathbf{O}_1(M^\phi)$ corresponds to a subexpression of M with an ω-e-type (and viceversa). Moreover $\Sigma \vdash_1 M^\phi$ implies both $\Sigma' \vdash_1 M^\phi$ and $\Sigma' \vdash_1 \mathbf{O}_1(M^\phi)$. The proof is by induction on the derivation of $\Sigma' \vdash_1 M^\phi$. □

In order to use the simplification mapping $\mathbf{O}_1$ to simplify terms while preserving their meaning (w.r.t. $\mathcal{M}^{\mathrm{gnd}}$) we identify a subset of $\vdash_1$-typings for which the $\sim_\phi^\Sigma$ relations implies the $\simeq_{\mathrm{obs}}^{\mathrm{gnd}}$ relation.

Definition 5.9 (Faithful $\vdash_1$-typing) $\Sigma \vdash_1 M^\phi$ *is a* faithful *$\vdash_1$-type assignment statement if $\phi \in \mathbf{L}_\delta$, and for all $x : \psi \in \Sigma$, $\psi \in \mathbf{L}_\omega \cup \mathbf{L}_\delta$.*

Example 5.10 *Both the two $\vdash_1$-typings in Example 5.3 are faithful. By applying the simplification mapping $\mathbf{O}_1$ we get the following simplifications of the decorated version of M and N:*

- $M'^\delta = ((\lambda x.3)\mathsf{d}^{\omega^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$, *and*
- $N'^\delta = ((\lambda x.\mathsf{proj}_1\langle Q_1, \mathsf{d}_1\rangle^{\omega^{\mathsf{nat}}})\mathsf{d}_2^{\omega^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$.

Then, by erasing all the e-type decorations, we get:

- $M' = (\lambda x^{\mathsf{nat}}.3)\mathsf{d}^{\mathsf{nat}}$, *and*

[3] Dummy variables are simply a tool for proving the soundness of the simplification mapping $\mathbf{O}_1$. Indeed, since in a simplified program every dummy variable is dead-code, in a real implementation we can replace them with some polymorphic "dummy constant" d.

$\mathbf{O}_1(M^\psi) \;=\; (\mathbf{o}_1(M,\psi))^\psi, \quad$ where

$$
\begin{array}{rcl}
\mathbf{o}_1(M,\psi) & = & \mathsf{d}, \quad \text{if } \psi \in \mathbf{L}_\omega\ . \quad \text{Otherwise:} \\
\mathbf{o}_1(\mathsf{k},\psi) & = & \mathsf{k} \\
\mathbf{o}_1(x,\psi) & = & x \\
\mathbf{o}_1(\langle M_1, M_2\rangle, \psi_1 \times \psi_2) & = & \langle \mathbf{o}_1(M_1,\psi_1), \mathbf{o}_1(M_2,\psi_2)\rangle \\
\mathbf{o}_1(\mathsf{proj}_i M^{\psi_{3-i}}, \psi_i) & = & \mathsf{proj}_i(\mathbf{o}_1(M, \psi_1 \times \psi_2))^{\psi_{3-i}} \quad \text{where } i \in \{1,2\} \\
\mathbf{o}_1(MN^\phi, \psi) & = & \mathbf{o}_1(M, \phi \to \psi)(\mathbf{o}_1(N,\phi))^\phi \\
\mathbf{o}_1(\lambda x.M, \psi_1 \to \psi_2) & = & \lambda x.\mathbf{o}_1(M,\psi_2) \\
\mathbf{o}_1(\mathsf{fix}\, x.M, \phi) & = & \mathsf{fix}\, x.\mathbf{o}_1(M,\phi) \\
\mathbf{o}_1(\mathsf{ifz}\, N \,\mathsf{then}\, M_1 \,\mathsf{else}\, M_2, \psi) & = & \mathsf{ifz}\, \mathbf{o}_1(N, \delta^{\mathsf{nat}}) \,\mathsf{then}\, \mathbf{o}_1(M_1,\psi) \,\mathsf{else}\, \mathbf{o}_1(M_2,\psi)
\end{array}
$$

Figure 7: Mapping $\mathbf{O}_1$ on terms

- $N' = (\lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, \mathsf{d}_1^{\mathsf{nat}}\rangle)\mathsf{d}_2^{\mathsf{nat}}$.

The proof that the simplification performed by the mapping $\mathbf{O}_1$ on faithful decorated terms preserve $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$ relys on the following lemma.

Lemma 5.11 *Let $\Sigma \vdash_1 M^\phi$ and $\Sigma \vdash_1 N^\phi$ be faithful $\vdash_1$-typings. Then $\epsilon(M^\phi) \sim^\Sigma_\phi \epsilon(N^\phi)$ implies $\epsilon(M^\phi) \simeq^{\mathrm{gnd}}_{\mathrm{obs}} \epsilon(N^\phi)$.*

Proof. Let Σ' be the restriction of Σ to the variables $\mathrm{FV}(M) \cup \mathrm{FV}(N)$. Since $\Sigma \vdash_1 M^\phi$ is faithful, then for all $x : \psi \in \Sigma'$, $\psi \in \mathbf{L}_\omega \cup \mathbf{L}_\delta$. So for all e, e is Σ' related with e. Let now $\epsilon(M^\phi) \sim^\Sigma_\phi \epsilon(N^\phi)$, then (by Definition 5.4.2) $(\llbracket \epsilon(M^\phi) \rrbracket_e, \llbracket \epsilon(N^\phi) \rrbracket_e) \in \llbracket \phi \rrbracket$. Therefore, from Proposition 4.4, since $\phi \in \mathbf{L}_\delta$, we get for all e, $\llbracket \epsilon(M^\phi) \rrbracket_e \simeq^{\mathrm{gnd}}_{\mathrm{obs}} \llbracket \epsilon(N^\phi) \rrbracket_e$. Using the definition of $\simeq^{\mathbf{o}}$ and Theorem 2.6 we conclude that $\epsilon(M^\phi) \simeq^{\mathrm{gnd}}_{\mathrm{obs}} \epsilon(N^\phi)$. $\square$

We can now show how to use the system $\vdash_1$ and the simplification mapping $\mathbf{O}_1$ to simplify terms producing observationally equivalent terms. In particular we show that all the dummy variables introduced by applying the mapping $\mathbf{O}_1$ on a faithful $\vdash_1$-typing are dead-code.

Theorem 5.12 ($\mathbf{O}_1$ on faithful $\vdash_1$-typings preserves $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$) *Let $\Sigma \vdash_1 M^\phi$ be a faithful $\vdash_1$-typing. Then $\epsilon(M^\phi) \simeq^{\mathrm{gnd}}_{\mathrm{obs}} \epsilon(\mathbf{O}_1(M^\phi))$.*

Proof. Let $\Sigma' = \Sigma \cup \{\mathsf{d}^\sigma : \omega^\sigma \mid \mathsf{d}^\sigma \in \mathrm{DV}(\epsilon(M^\phi))\}$. Since $\Sigma \vdash_1 M^\phi$ is a faithful $\vdash_1$-typing then also $\Sigma' \vdash_1 M^\phi$ and $\Sigma' \vdash_1 \mathbf{O}_1(M^\phi)$ are faithful $\vdash_1$-typings. So the result follows immediately from Proposition 5.8 and Lemma 5.11. $\square$

Example 5.13 *Let $\vdash_{\mathbf{T}} M : \mathsf{nat}$ where $\mathrm{FV}(M) = \{f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}}, u^{\mathsf{nat}}, v^{\mathsf{nat}}\}$ and*

$$M = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}. + \langle +\langle fg, gx\rangle, (\lambda y^{\mathsf{nat}}.1)P\rangle)(\lambda z^{\mathsf{nat}}.3)Q,$$

with P, Q terms of type nat *such that* $\mathrm{FV}(P) = \{u^{\mathsf{nat}}\}$ *and* $\mathrm{FV}(Q) = \{v^{\mathsf{nat}}\}$.

Take the faithful $\vdash_1$*-typing* $\Sigma' \vdash_1 M'^{\delta^{\mathsf{nat}}}$, *where (writing, for short,* δ *and* ω *instead of* δ^{nat} *and* ω^{nat}*):* $\Sigma' = \{f : (\delta \to \delta) \to \delta, u : \omega, v : \omega\}$ *and*

$$M'^{\delta} = ((\lambda g.\lambda x. + \langle +\langle f g^{\omega\to\delta}, g x^{\omega}\rangle^{\delta\times\delta}, (\lambda y.1) P^{\omega}\rangle^{\delta\times\delta})(\lambda z.3)^{\omega\to\delta} Q^{\omega})^{\delta} \ .$$

Applying the $\mathbf{O}_1$ *simplification mapping we get* $\mathbf{O}_1(\Sigma') \cup \{\mathsf{d}_1 : \omega, \mathsf{d}_2 : \omega, \mathsf{d}_3 : \omega\} \vdash_1 \mathbf{O}_1(M'^{\delta})$, *where* $\mathbf{O}_1(\Sigma') = \{f : (\delta \to \delta) \to \delta\}$ *and*

$$\mathbf{O}_1(M'^{\delta}) = ((\lambda g.\lambda x. + \langle +\langle f g^{\omega\to\delta}, g \mathsf{d}_1^{\omega}\rangle^{\delta\times\delta}, (\lambda y.1) \mathsf{d}_1^{\omega}\rangle^{\delta\times\delta})(\lambda z.3)^{\omega\to\delta} \mathsf{d}_1^{\omega})^{\delta} \ .$$

Let N_1 *be the term obtained from* $\mathbf{O}_1(M'^{\delta})$ *by erasing the e-type decorations, i.e.,* $N_1 = \epsilon(\mathbf{O}_1(M'^{\delta}))$. *We have* $\vdash_{\mathbf{T}} N_1 : \mathsf{nat}$, $\mathrm{FV}(N_1) = \{f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}}, {\mathsf{d}_1}^{\mathsf{nat}}, {\mathsf{d}_2}^{\mathsf{nat}}, {\mathsf{d}_3}^{\mathsf{nat}}\}$ *and*

$$N_1 = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}. + \langle +\langle f g, g \mathsf{d}_1^{\mathsf{nat}}\rangle, (\lambda y^{\mathsf{nat}}.1)\mathsf{d}_2^{\mathsf{nat}}\rangle)(\lambda z^{\mathsf{nat}}.3)\mathsf{d}_3^{\mathsf{nat}} \ .$$

5.3 Optimal faithful $\vdash_1$-typings

In order to be able to detect all the dead-code that can be proved by system $\vdash_1$ in a given PCFP term, say M, is useful to see if there is a faithful $\vdash_1$-typing of M that shows all the dead-code that is showed by some faithful $\vdash_1$-typing of M. If such an "optimal" faithful $\vdash_1$-typing exists then the problem of detecting all the dead-code that can be proved by system $\vdash_1$ would be reduced to the problem of computing the "optimal" faithful $\vdash_1$-typing.

Given two faithful $\vdash_1$-typings of M, $\Sigma' \vdash_1 M'^{\phi}$ and $\Sigma'' \vdash_1 M''^{\phi}$, the one which allows the mapping $\mathbf{O}_1$ to remove more dead-code that the other is the one that assigns ω-e-types to more subterms. This fact induces a partial order relation, $\sqsubseteq$, on the set of the $\vdash_1$-typings of M. The partial order $\sqsubseteq$ (for basic properties, e-types, basis, $\vdash_1$-decorated terms, and $\vdash_1$-typings) is defined as follows.

Definition 5.14 (The $\sqsubseteq$ partial order) *1. For basic properties the order* $\sqsubseteq$ *is defined by:* $\delta \sqsubseteq \delta$, $\delta \sqsubseteq \omega$ *and* $\omega \sqsubseteq \omega$.

2. *For e-types the partial order* $\sqsubseteq$ *is defined by the rules in Fig. 8*[4].

3. *For basis the partial order* $\sqsubseteq$ *is defined by:* $\Sigma \sqsubseteq \Sigma'$ *if for every* $x^{\rho} : \psi \in \Sigma'$ *there exists* $x^{\rho} : \phi \in \Sigma$ *such that* $\phi \sqsubseteq \psi$.

4. *For* $\vdash_1$*-decorated terms the partial order* $\sqsubseteq$ *is defined by the rules in Fig. 9.*

5. *For* $\vdash_1$*-typings the partial order* $\sqsubseteq$ *is defined by:* $\Sigma \vdash_1 M^{\phi} \sqsubseteq \Sigma' \vdash_1 M'^{\psi}$ *if* $\Sigma \sqsubseteq \Sigma'$ *and* $M^{\phi} \sqsubseteq M'^{\psi}$.

It is straightforward to check that the relation $\sqsubseteq$ is a partial order.

Definition 5.15 (Optimal faithful $\vdash_1$-typing) *For every a PCFP term M the* optimal faithful $\vdash_1$-typing of M *is the maximum w.r.t.* $\sqsubseteq$ *of the set*

$$\{\Sigma \vdash_1 M'^{\phi} \mid \Sigma \vdash_1 M'^{\phi} \ \text{ is a faithful } \vdash_1\text{-typing of } M\} \ .$$

In [DG] it is proved that for every PCFP term M there exists the optimal faithful $\vdash_1$-typing of M, and it is given an algorithm which computes it.

[4] Do not confuse the partial order $\sqsubseteq$ for e-types, which is is covariant in both the arguments of the "$\to$" e-type constructor, and the e-type entailment relation $\leq$, which is contravariant in the left argument of $\to$.

$$\phi^\rho \sqsubseteq \omega^\rho$$

$$\frac{\phi_1 \sqsubseteq \psi_1 \quad \phi_2 \sqsubseteq \psi_2}{\phi_1 \rightarrow \phi_2 \sqsubseteq \psi_1 \rightarrow \psi_2}\ \phi_2, \psi_2 \notin \mathbf{L}_\omega$$

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

Figure 8: Relation $\sqsubseteq$ for e-types

Acknowledgments

We thank Stefano Berardi and Mario Coppo for valuable remarks and suggestions. The first author would also to thank Jean-Christophe Filliâtre for stimulating discussions on the topic of program extraction.

References

[BB95] S. Berardi and L. Boerio. Using Subtyping in Program Optimization. In *TLCA'95*, LNCS 902. Springer–Verlag, 1995.

[BB97] S. Berardi and L. Boerio. Minimum Information Code in a Pure Functional Language with Data Types. In *TLCA'97*, LNCS 1210. Springer–Verlag, 1997.

[BBC+98] B. Barras, S. Boutin, C. Cornes, J. Courant, Y. Coscoy, D. Delahaye, D. de Rauglaudre, J.-C. Filliâtre, E. Giménez, H. Herbelin, G. Huet, H. Laulhère, C. Muñoz, C. Murthy, C. Parent-Vigouroux, P. Loiseleur, C. Paulin-Mohring, A. Saïbi, and B. Werner. *The Coq Proof Assistant Reference Manual Version 6.2.* INRIA-Rocquencourt-CNRS-ENS Lyon, May 1998.

[Ben92] P. N. Benton. *Strictness Analysis of Lazy Functional Programs.* PhD thesis, University of Cambridge, Pembroke College, 1992.

[Ber93] S. Berardi. Pruning Simply Typed Lambda Terms, 1993. Dipartimento di Informatica, Università di Torino, Internal report.

[Ber96] S. Berardi. Pruning Simply Typed Lambda Terms. *Journal of Logic and Computation*, 6(5):663–681, 1996.

[Boe95] L. Boerio. *Optimizing Programs Extracted from Proofs.* PhD thesis, Università di Torino, 1995.

[BS95] E. Barendsen and S. Smetsers. A Derivation System for Uniqueness Typing . In *SEGRAGRA'95.* Elsevier, Electronic Notes in Theoretical Computer Science, 1995.

[Dam98] F. Damiani. *Non-standard type inference for functional programs.* PhD thesis, Università di Torino, February 1998. Available from `htpp://www.di.unito.it/~damiani`.

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

[DHM95] D. Dussart, F. Henglein, and C. Mossin. Polymorphic Recursion and Subtype Qualifications: Polymorphic Binding-Time Analysis in Polynomial Time. In *SAS'95*, LNCS 983, pages 118–135. Springer–Verlag, 1995.

$$M^{\phi} \sqsubseteq \epsilon(M)^{\omega^{\epsilon(\phi)}}$$

$$\frac{\phi \sqsubseteq \psi}{x^{\phi} \sqsubseteq x^{\psi}}\ \psi \notin \mathbf{L}_\omega$$

$$\mathsf{k}^{\delta(\mathbf{T}(\mathsf{k}))} \sqsubseteq \mathsf{k}^{\delta(\mathbf{T}(\mathsf{k}))}$$

$$\frac{\phi_1 \sqsubseteq \psi_1 \qquad M^{\phi_2} \sqsubseteq N^{\psi_2}}{(\lambda x.M)^{\phi_1 \to \phi_2} \sqsubseteq (\lambda x.N)^{\psi_1 \to \psi_2}}\ \phi_2, \psi_2 \notin \mathbf{L}_\omega$$

$$\frac{F^{\phi_1 \to \phi_2} \sqsubseteq G^{\psi_1 \to \psi_2} \qquad M^{\phi_2} \sqsubseteq N^{\psi_2}}{(FM^{\phi_1})^{\phi_2} \sqsubseteq (GN^{\psi_1})^{\psi_2}}$$

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

$$\frac{M^{\phi_1 \times \phi_2} \sqsubseteq N^{\psi_1 \times \psi_2}}{(\mathsf{proj}_i M^{\phi_{3-i}})^{\phi_i} \sqsubseteq (\mathsf{proj}_i N^{\psi_{3-i}})^{\psi_i}}\ i \in \{1,2\} \text{ and } \phi_i, \psi_i \notin \mathbf{L}_\omega$$

$$\frac{M^{\phi} \sqsubseteq N^{\psi}}{(\mathsf{fix}\, x.M)^{\phi} \sqsubseteq (\mathsf{fix}\, x.N)^{\psi}}\ \phi, \psi \notin \mathbf{L}_\omega$$

$$\frac{B^{\delta^{\mathsf{nat}}} \sqsubseteq C^{\delta^{\mathsf{nat}}} \qquad M_1^{\phi} \sqsubseteq N_1^{\psi} \qquad M_2^{\phi} \sqsubseteq N_2^{\psi}}{(\mathsf{ifz}\, B\, \mathsf{then}\, M_1\, \mathsf{else} M_2)^{\phi} \sqsubseteq (\mathsf{ifz}\, C\, \mathsf{then}\, N_1\, \mathsf{else} N_2)^{\psi}}\ \phi, \psi \notin \mathbf{L}_\omega$$

Figure 9: Relation $\sqsubseteq$ for decorated terms

[HL94] C. Hankin and D. Le Métayer. Deriving algorithms for type inference systems: Applications to strictness analysis. In *POPL'94*, pages 202–212. ACM, 1994.

[Jen92] T. P. Jensen. *Abstract Interpretation in Logical Form.* PhD thesis, University of London, Imperial College, 1992.

[Kah88] G. Kahn. Natural semantics. In K. Fuchi and M. Nivat, editors, *Programming Of Future Generation Computer.* Elsevier Sciences B.V. (North-Holland), 1988.

[KM89] T. M. Kuo and P. Mishra. Strictness analysis: a new perspective based on type inference. In *Functional Programming Languages and Computer Architecture*, pages 260–272. ACM, 1989.

[Mos97] C. Mossin. *Flow analysis of Typed Higher-Order Programs.* PhD thesis, DIKU, University of Copenhagen, 1997. Revised version.

[Pau89a] C. Paulin-Mohring. Extracting F_ω's Programs from Proofs in the Calculus of Constructions. In *POPL'89.* ACM, 1989.

[Pau89b] C. Paulin-Mohring. *Extraction de Programme dans le Calcul des Constructions.* PhD thesis, Université Paris VII, 1989.

[Pfe96] Frank Pfenning. The practice of logical frameworks. In Hélène Kirchner, editor, *Proceedings of the Colloquium on Trees in Algebra and Programming*, pages 119–134, Linköping, Sweden, April 1996. Springer-Verlag LNCS 1059. Invited talk.

[Pit97] A. M. Pitts. Operationally-based theories of program equivalence. In A. M. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*, pages 241–298. Cambridge University Press, 1997.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223–255, 1977.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, 1981.

[Sol95] K. L. Solberg. *Annotated Type Systems for Program Analysis.* PhD thesis, Aarhus University, Denmark, 1995. Revised version.

[Tak91] Y. Takayama. Extraction of Redundancy-free Programs from Constructive Natural Deduction Proofs. *Journal of Symbolic Computation*, 12:29–69, 1991.

[Wri92] D. A. Wright. *Reduction Types and Intensionality in the Lambda-Calculus.* PhD thesis, University of Tasmania, 1992.