Extensional Collapse Situations I: non-termination and unrecoverable errors

Antonio Bucciarelli

Extensional collapse situations I: non-termination and unrecoverable errors

Antonio Bucciarelli

PPS, UMR 7126 CNRS and Université Paris Diderot, France

Abstract. We consider a simple model of higher order, functional computations over the booleans. Then, we enrich the model in order to encompass non-termination and unrecoverable errors, taken separately or jointly. We show that the models so defined form a lattice when ordered by the *extensional collapse situation* relation, introduced in order to compare models with respect to the amount of "intensional information" that they provide on computation. The proofs are carried out by exhibiting suitable applied λ-calculi, and by exploiting the fundamental lemma of logical relations.

1 Introduction

Properties of programs are often considered as split in two categories: the *extensional* properties and the *intensional* ones. Typical intensional properties are those concerning complexity issues, while typical extensional ones are those concerning input-output, observable, behaviours. This dichotomy originated from recursion theory: a predicate $\mathcal{P}(M)$ on Turing Machines is extensional if, whenever it holds for a T.M. M, it holds for all the T.M. computing the same function as M. Otherwise, $\mathcal{P}$ is intensional. So, for instance, whether the computation on 0 takes more or less than 100 steps is an intensional property, while undefinedness on 0 is an extensional one.

It is the intended meaning of a T.M., namely the underlying partial recursive function, which determines the frontier between what is (to be considered as) extensional and what is (to be considered as) intensional.

Therefore, a tentative definition of the notion of extensional property could be the following: a property of programs is extensional if the fact whether or not a program P enjoys it does affect the denotation of P; it is intensional otherwise.

Stated this way, being extensional or intensional is a matter of the intended meaning of programs, and, as such, it is a relative notion[1].

The following examples, written in a call by name simply typed λ-calculus with constants, illustrate this claim:

[1] The very same happens for the dichotomy "function vs algorithm", which could be misleadingly considered as an absolute one. For instance, the well-known parallel-or "function" is an algorithm relatively to the model $\mathcal{S}$ of *total objects*, and a function relatively to the model $\mathcal{C}$ of *partial objects*, defined below.

Example 1. The terms:

(1) λ`x:bool. true`

(2) λ`x:bool. if x then true else true`

get the same interpretation in a model where the type of booleans is interpreted by the set $\{tt, ff\}$, and higher types by the corresponding function space in **Set**; they get different interpretations in a model that accommodates non-terminating computations, denoted by a suitable element $\perp$.

Otherwise stated, strictness is an intensional property relatively to the very simple, set-theoretical model, an extensional one relatively to a more sophisticated one, based on partial orders.

The linguistic feature that forces strictness to be taken into account extensionally is anything allowing for non-termination (e.g. fixpoint operators).

Example 2. Similarly, the terms:

(3) λ`x:bool.` Ω

(4) λ`x:bool. if x then` Ω `else` Ω

where Ω is any diverging term, get the same interpretation in the model of non-termination introduced above; they get different interpretations in a model that accommodates also errors, denoted by suitable element $\top$.

More examples could be provided, concerning for instance the property of linearity, or the evaluation-order dependency, but they go beyond the semantic analysis that we propose in the present work.

Hence, the frontier between extensional and intensional properties is determined essentially by the model we refer to. Nevertheless, certain features of programs have to be taken into account extensionally by all models, namely those features that affect the operational behaviour of terms (for instance, strictness in a language allowing for non-termination).

By the way, a fully abstract model is one that keeps implicit (i.e. intensional) all the properties that can be kept implicit.

It appears that the models of simply typed λ-calculi may be classified with respect to the amount of information on programs that they provide explicitly, i.e. extensionally.

In this paper, we propose a way of performing this classification, and show how it works on some very basic models of higher order computation over the booleans.

The basic tool used for the classification is the notion of *extensional collapse situation*. An extensional collapse situation is given by two models[2], and a binary pre-logical relation between them which is a partial surjective function, at all types. In this case, the small model, i.e. the target of the surjective function, can be considered as what is left of the big one when some kind of computational behaviour (in this paper, we focus on non-termination and errors) is forbidden.

Elements of the big model may be mapped onto the same element of the small one by the surjection: this is the case, for instance, of the interpretations

[2] The models considered in this paper are families of sets indexed by simple types, usually called *type frames*.

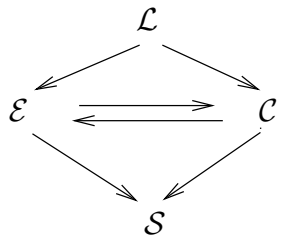

Fig. 1. Extensional collapse situations. We write $\mathcal{M} \to^{\mathrm{E}} \mathcal{N}$ the fact that $\mathcal{M}$, $\mathcal{N}$ and the logical relation E determine an extensional collapse situation. When E at ground types is a canonical surjection, clear from the context, we write $\mathcal{M} \to \mathcal{N}$.

of the terms (1) and (2) above, when passing from the partially ordered model to the simple set-theoretic one. They may also "vanish", since the surjection is partial: this is the case of the interpretation of (3) in the aprtially ordered model, the *non-total* constantly undefined function.

In order to prove that a given pair of models $\mathcal{M}, \mathcal{N}$, determines an extensional collapse situation, we follow the following pattern:

- define a simply typed λ-calculus $\Lambda^{\mathcal{N}}$, such that $\mathcal{N}$ is fully complete w.r.t. $\Lambda^{\mathcal{N}}$ (i.e. such that all elements of $\mathcal{N}$, at all types, are definable by closed $\Lambda^{\mathcal{N}}$-terms),
- show that $\mathcal{M}$ is a model of $\Lambda^{\mathcal{N}}$,
- use the fundamental lemma of logical relations (Lemma 6) to exhibit a suitable pre-logical relation, induced by $\Lambda^{\mathcal{N}}$, and conclude.

An interesting by-product, which is an immediate corollary of the fundamental lemma of logical relations, is that the $\Lambda^{\mathcal{N}}$-theory of $\mathcal{M}$ is included in that of $\mathcal{N}$.

We consider the following models:

- $\mathcal{S}$: the full type hierarchy over the set $\{tt, ff\}$. Its elements are "higher order boolean circuits".
- $\mathcal{C}$: the standard model of monotonic[3] functions over the partial order $\bot < tt, ff$.
- $\mathcal{E}$: the dual of $\mathcal{C}$, i.e. the model of monotonic function over the partial order $tt, ff < \top$, modelling unrecoverable errors in the absence of non-termination.
- $\mathcal{L}$: the model of monotonic functions over the lattice $\top < tt, ff < \top$, modelling non-termination and errors.

We focus on the extensional collapse situations represented in Figure 1. The λ-

[3] In the general case, the morphisms of the standard model are the Scott continuous functions. Focusing on finite domains, the monotonic functions and the Scott continuous ones coincide.

calculi used for proving that those are actually extensional collapse situations are $\Lambda^{\mathcal{S}}$, whose constants are $\texttt{true}, \texttt{false}$ and $\texttt{if}$, and $\Lambda^{\mathcal{C}}$, the parallel extension of finitary PCF, due to Plotkin [13]. It is not necessary to introduce a language $\Lambda^{\mathcal{E}}$, in order to deal with the model $\mathcal{E}$[4]: we prove instead that $\mathcal{E}$ and $\mathcal{C}$ are *logically isomorphic* [5], and conclude since extensional collapse situations do compose.

The existence of an extensional collapse situation between two models $\mathcal{M}$ and $\mathcal{N}$ witnesses the fact that the target of the collapse is obtained by forgetting one (or more) computational aspect(s) that are explicitly taken into account in the source.

Let us consider, for instance, the collapse of $\mathcal{C}$ over $\mathcal{S}$, described in [5]: unsurprisingly, at the ground type bool the surjection is the partial identity, undefined on $\bot$.

At first-order types, a monotonic funcion $f \in \mathcal{C}_\sigma$ is surjected onto the boolean circuit $c \in \mathcal{S}_\sigma$ exactly when it provides the same value as c on total tuples (i.e. on tuples not containing $\bot$). Hence, f can be considered as an algorithm implementing c.

We have already seen in Example 1 that there exist two implementations of the constantly tt function from $\mathcal{S}_{\mathrm{bool}}$ to $\mathcal{S}_{\mathrm{bool}}$, namely the strict and the non-strict constantly tt functions from $\mathcal{C}_{\mathrm{bool}}$ to $\mathcal{C}_{\mathrm{bool}}$.

To go a little further, let us consider the n-ary disjunction or_n, of type $\underbrace{\mathrm{bool} \to \ldots \to \mathrm{bool}}_{n \text{ times}} \to \mathrm{bool}$, yielding the result tt whenever at least one of the arguments is tt and ff if all the arguments are ff.

They can be implemented in several ways, ranging from the most lazy (and parallel) algorithm yielding the result tt whenever at least one of the arguments is tt, to the most eager one, yielding the result tt whenever all the argument are different from $\bot$, and at least one of them is tt.

In the case $n = 2$, this gives four different implementations of the disjunction, which are usually named, from the laziest to the most eager, *parallel-or*, *left-or*,*right-or* and *strict-or*.

In the general case, it is not difficult to realise that the algorithms implementing or_n form a lattice whose size grows exponentially in n. The laziest algorithm, is the bottom, and the most eager is the top of the lattice.

Summing up, and generalising, we have that for all types σ, and all boolean functionals $f \in \mathcal{S}_\sigma$, the set of implementations of f is a sub cpo of $\mathcal{C}_\sigma$, called the *totality class* of f, which is a lattice.

Hence, the model $\mathcal{S}$ is obtained by collapsing $\mathcal{C}$ via the totality relation; in the same way $\mathcal{C}$ is obtained by collapsing $\mathcal{L}$ via the "error-freedom" relation, which, at ground type, is the partial identity function, undefined on $\top$.

---

[4] $\Lambda^{\mathcal{E}}$ could be defined as the dual of $\Lambda^{\mathcal{C}}$ with respect to the inversion of $\bot$ and $\top$.

[5] i.e. that they form an extensional collapse situation in either way.

1.1 Related works

In the literature, a model of the simply typed λ-calculus is called extensional if all its elements, at all types, are invariant with respect to the logical relation defined as the identity at ground types[6].

When a model is *not* extensional, its extensional collapse is performed by eliminating the non-invariant elements. The result is an extensional model.

This pattern has been followed , for instance, for game models [1,10], or models obtained by *sequentiality relations* [14,11].

Sometimes, the resulting extensional model happens to have been defined and studied independently: those are instances of what we call here extensional collapse situation.

Examples[7] of this kind are:

- sequential algorithms collapsing on strongly stable functions [6].
- the relational model collapsing on Scott-continuous functions (between complete lattices) [7].

Nevertheless, extensional collapse situations as defined in the present paper cover a broader landscape than these extensional collapses. The essential difference is that the extensional collapse of a model is, by definition, unique, whereas different extensional collapse situations may concern a given model (as it is the case of the lattice model $\mathcal{L}$, which collapses over $\mathcal{C}$, $\mathcal{E}$, and $\mathcal{S}$). Even choosing two particular models $\mathcal{M}$ and $\mathcal{N}$, there can be more than one extensional collapse situation between them. For instance, an extensional collapse situation between $\mathcal{S}$ and $\mathcal{C}$ has already been showed in [5], using the *totality* logical relation. The use of the language $\Lambda^{\mathcal{S}}$ and of the associated (canonical) pre-logical relation provides a different partial surjection between those two models, and, by the way, makes the proof presented here easier.

Extensional collapse situations, defined in a slightly different way, have been used in [3] for constructing, given two models $\mathcal{M}, \mathcal{N}$ of a given applied λ-calculus, a quotient model $\mathcal{M}/\mathcal{N}$ whose theory is a super-set of both $\mathrm{Th}(\mathcal{M})$ and $\mathrm{Th}(\mathcal{N})$.

2 Logical versus pre-logical relations

Logical relations have been introduced by Plotkin in [12], in order to characterize λ-definability in the full type hierarchy. The main features of logical relations are:

- they contain all the tuples of the form $(\llbracket M \rrbracket, \ldots, \llbracket M \rrbracket)$, where M is a closed, simply typed λ-term. This is known as "the fundamental property"of logical relations.

[6] Otherwise stated: $\forall \sigma, \tau, f, g : \sigma \to \tau$ if $\forall x \in \sigma\ f(x) = g(x)$ then $f = g$.

[7] It has to be noticed that in these examples the collapse is not a logical relation. It is defined as a categorical notion, independently from the hierarchy of simple types and obviously instantiating to an extensional collapse situation in the sense defined in this paper.

– they are completely determined by their behaviour at ground types. Hence, defining a logical relation boils down to specify it at ground types.

The main drawback of logical relations, from our point of view at least, is that they are not closed under (relational) composition; actually, the property stated in the second item above is incompatible with the closure under composition. In [9], Honsell and Sannella define *pre-logical relations*, as a weakening of the notion of logical relation which enjoy the fundamental property and is closed under (relational) composition. The price to pay is that pre-logical relation are not determined by their behaviour at ground types, and that they are *language-dependent*: at higher-types, pre-logical relations have to obey a closure condition which depends from the specific simply typed λ-calculus Λ one is interested in [8].

Nevertheless, pre-logical relations are very well adapted to our framework: in general, we will prove the existence of an extensional collapse situation $\mathcal{M} \rightarrow^{\mathrm{E}} \mathcal{N}$ by exhibiting a λ-calculus $\Lambda^{\mathcal{N}}$, with respect to which $\mathcal{N}$ is fully complete. In such a situation, if $\mathcal{M}$ is a model of $\Lambda^{\mathcal{N}}$, the relation $\{([\![P]\!]^{\mathcal{M}}, [\![P]\!]^{\mathcal{N}}) \mid P \in (\Lambda^{\mathcal{N}}_{\sigma})^0\}_{\sigma \in Types}$ is a partial surjective function at all types, and it is a $\Lambda^{\mathcal{N}}$-pre-logical relation (which is not logical in general).

We want to be able to compose such pre-logical relations. Of course, the composition of relations which are partial surjective functions is a partial surjective function. Moreover, we know that, given a simply typed λ-calculus Λ, the composition of two Λ-pre-logical relations is Λ-pre logical [9].

The point here is that we will compose relations which are pre-logical relatively to different λ-calculi: the set of constants of $\Lambda^{\mathcal{N}}$ will of course depend on $\mathcal{N}$. Nevertheless, when composing a $\Lambda^{\mathcal{N}}$-pre-logical relation and a $\Lambda^{\mathcal{N}'}$-pre-logical relation, one gets a Λ-pre-logical relation for any language Λ which is a sub-language of both $\Lambda^{\mathcal{N}}$ and $\Lambda^{\mathcal{N}'}$[9]. Hence, we consider the poorest possible language, namely the simply typed λ-calculus *without* constants, over a given set of ground types (for our purposes, a unique ground type bool is enough), call it Λ, and define extensional collapse situations as Λ-pre-logical relations which are partial surjctive functions at all types.

It has to be noticed that logical relations are, *a fortiori*, Λ-pre-logical. Hence the composition of two logical relations, which is not logical in general, is Λ-pre-logical.

So, in general, we can exhibit different extensional collapse situations between two given models. Let us consider again, for instance, the case of $\mathcal{C}$ and $\mathcal{S}$. We will see in Section 3 that the simply typed λ calculus equipped with the ground boolean constants `true` and `false` and the first-order constant `if`, noted $\Lambda^{\mathcal{S}}$, determines, following the lines described above, an extensional collapse situation $\mathcal{C} \rightarrow^{E} \mathcal{S}$, which is different from the totality collapse described in Section 1. For instance, the parallel-or function is related to the binary disjunction in the

[8] Moreover, just like logical relations, they have to be closed with respect to the interpretations of the constants of Λ.

[9] This is an immediate consequence of the definition of pre-logical relations and of proposition 1.

totality collapse, but not in E, since no term of $\Lambda^{\mathcal{S}}$ defines the parallel-or (by the way, the three "sequential" disjunctions strict-or, left-or and right-or are $\Lambda^{\mathcal{S}}$-definable).

3 Type frames, logical and pre-logical relations

Since all the models considered in this paper are extensional, we provide here a very simple definition of type frame, where higher types are interpreted by sets of *functions*[10].

The set of *simple types* over a set K of *ground type constants* is the smallest set containing K and closed w.r.t. the operation $\sigma, \tau \mapsto \sigma \to \tau$.

Definition 1. *A* type frame $\mathcal{M}$ *over a set K of ground types is a family of sets indexed by simple types over K, such that*
$\mathcal{M}_{\sigma\to\tau} \subseteq \{f \mid f$ *is a function from* $\mathcal{M}_\sigma$ *to* $\mathcal{M}_\tau\}$.
A type frame is finite *if all the sets of the family are finite sets.*

When $\mathcal{M}_{\sigma\to\tau} = \mathcal{M}_\tau^{\mathcal{M}_\sigma}$, the type frame is *full*.

Definition 2. *Given two type frames $\mathcal{M}$ and $\mathcal{N}$ over K, a* binary logical relation R *between $\mathcal{M}$ and $\mathcal{N}$ is a family of binary relations $\mathrm{R}_\sigma \subseteq \mathcal{M}_\sigma \times \mathcal{N}_\sigma$, indexed by simple types over K, such that, for all $\sigma, \tau, f \in \mathcal{M}_{\sigma\to\tau}, g \in \mathcal{N}_{\sigma\to\tau}$:*
$(f,g) \in \mathrm{R}_{\sigma\to\tau} \Leftrightarrow [\forall x \in \mathcal{M}_\sigma, y \in \mathcal{N}_\sigma\ [(x,y) \in \mathrm{R}_\sigma \Rightarrow (f(x), g(y)) \in \mathrm{R}_\tau]]$

For defining pre-logical relation, we use the notion of environment-based interpretation of the typed λ-calculus in type frames, which will be introduced in Section 4. In the following definition, let Λ be the simply typed λ-calculus, without constants, over a set K of ground types.

Definition 3. *A* binary pre-logical relation R *between $\mathcal{M}$ and $\mathcal{N}$ is a family of binary relations $\mathrm{R}_\sigma \subseteq \mathcal{M}_\sigma \times \mathcal{N}_\sigma$, indexed by simple types over K, such that:*

- *for all $\sigma, \tau, f \in \mathcal{M}_{\sigma\to\tau}, g \in \mathcal{N}_{\sigma\to\tau}$: $(f,g) \in \mathrm{R}_{\sigma\to\tau} \Rightarrow [\forall x \in \mathcal{M}_\sigma, y \in \mathcal{N}_\sigma\ [(x,y) \in \mathrm{R}_\sigma \Rightarrow (f(x), g(y)) \in \mathrm{R}_\tau]]$*
- *Given:*
  - *$P \in \Lambda_\tau$, for any type τ,*
  - *a $\mathcal{M}$-environment ρ and a $\mathcal{N}$-environment ρ' such that, for all types δ and variable $x : \delta$, $(\rho(x), \rho'(x)) \in \mathrm{R}_\delta$,*
  - *a variable $z : \sigma$, for any type σ,*

  the following holds:
  $[\forall (a,b) \in \mathrm{R}_\sigma\ (\llbracket P \rrbracket^{\mathcal{M}}_{\rho[z:=a]}, \llbracket P \rrbracket^{\mathcal{N}}_{\rho'[z:=b]}) \in \mathrm{R}_\tau] \Rightarrow (\llbracket \lambda z.M \rrbracket^{\mathcal{M}}_\rho, \llbracket \lambda z.M \rrbracket^{\mathcal{N}}_{\rho'}) \in \mathrm{R}_{\sigma\to\tau}$

It is easy to see that any logical relation is pre-logical. We are interested in a particular kind of pre-logical relations:

[10] In section 6 we provide the more general definition, encompassing non extensional models.

Definition 4. *A binary pre-logical relation* R *between two type frames* $\mathcal{M}$ *and* $\mathcal{N}$ *over* K *is a* pre-logical surjection *if:*

- *at all types,* R *is surjective:* $\forall\sigma\ \forall y \in \mathcal{N}_\sigma\ \exists x \in \mathcal{M}_\sigma\ (x,y) \in \mathrm{R}_\sigma$
- *at all types,* R *is a partial function:* $\forall\sigma\ \forall x \in \mathcal{M}_\sigma\ \forall y,y' \in \mathcal{N}_\sigma\ (x,y),(x,y') \in \mathrm{R}_\sigma \Rightarrow y = y'$

Lemma 1. *A binary pre-logical relation between* $\mathcal{M}$ *and* $\mathcal{N}$ *which is a partial function at ground types and surjective at all types is a pre-logical surjection.*

Proof. Straightforward, by induction on types. The extensionality of $\mathcal{N}$ is required here.

Definition 5. *An* extensional collapse situation *is a triple* $\mathcal{M},\mathcal{N},\mathrm{E}$*, where* $\mathcal{M}$ *and* $\mathcal{N}$ *are type frames over a given set* K *of ground types and* E *is a pre-logical surjection from* $\mathcal{M}$ *to* $\mathcal{N}$*.*

We note $\mathcal{M} \to^{\mathrm{E}} \mathcal{N}$ the fact that $\mathcal{M},\mathcal{N},\mathrm{E}$ is an extensional collapse situation.

Extensional collapse situations are closed under composition, in the following sense:

Proposition 1 ([9], Prop. 5.5). *If* $\mathcal{M} \to^{\mathrm{E}} \mathcal{N}$ *and* $\mathcal{N} \to^{\mathrm{F}} \mathcal{P}$ *then* $\mathcal{M} \to^{\mathrm{F}\circ\mathrm{E}} \mathcal{P}$*, where* $(\mathrm{F}\circ\mathrm{E})_\sigma = \mathrm{F}_\sigma \circ \mathrm{E}_\sigma$*.*

Hence, the set of type frames over a given set K of ground types is pre-ordered by the relation $\mathcal{N} \leq \mathcal{M} \Leftrightarrow \mathcal{M} \to^{\mathrm{E}} \mathcal{N}$, for some E. Let $\equiv$ denote the equivalence relation associated to this pre-order.

We call *pre-logical isomorphism* a pre-logical relation E such that both E and E^{-1} are pre-logical surjections.

3.1 Type frames over bool

In order to define the type frames we are interested in, let us introduce the following notation: if A and B are sets (resp. partially ordered sets), $A \Rightarrow B$ (resp. $A \Rightarrow_m B$) denotes the set of all the functions from A to B (resp. the partially ordered set of monotonic functions from A to B, ordered pointwise).

We consider four type frames over a single ground type bool: $\mathcal{S}$, $\mathcal{C}$, $\mathcal{E}$ and $\mathcal{L}$, defined as follows:

- $\mathcal{S}^{\mathrm{bool}} = \{tt, ff\}$,
  $\mathcal{S}^{\sigma\to\tau} = \mathcal{S}^\sigma \Rightarrow \mathcal{S}^\tau$.
- $\mathcal{C}^{\mathrm{bool}} = \{\bot, tt, ff\}$, partially ordered by $\bot < tt, ff$,
  $\mathcal{C}^{\sigma\to\tau} = \mathcal{C}^\sigma \Rightarrow_m \mathcal{C}^\tau$.
- $\mathcal{E}^{\mathrm{bool}} = \{tt, ff, \top\}$, partially ordered by $tt, ff < \top$,
  $\mathcal{E}^{\sigma\to\tau} = \mathcal{E}^\sigma \Rightarrow_m \mathcal{E}^\tau$.
- $\mathcal{L}^{\mathrm{bool}} = \{\bot, tt, ff, \top\}$, partially ordered by $\bot < tt, ff < \top$,
  $\mathcal{L}^{\sigma\to\tau} = \mathcal{L}^\sigma \Rightarrow_m \mathcal{L}^\tau$.

The type frames $\mathcal{C}$ and $\mathcal{E}$ are dual. We show that $\mathcal{C} \equiv \mathcal{E}$ by exhibiting a pre-logical isomorphism.

Proposition 2. *The logical relation* E *between* $\mathcal{C}$ *and* $\mathcal{E}$ *defined by* $\mathrm{E}_{\mathrm{bool}} = \{(tt, tt), (ff, ff), (\bot, \top)\}$ *is a pre-logical isomorphism.*

Sketch of the proof. First of all, the relation defined above is logical, hence, *a fortiori*, pre-logical. In order to show that it is a pre-logical isomorphism, one can prove by simultaneous induction on types, the following statements:

(i_σ) E_σ is a bijection.
(ii_σ) $\forall (x, y), (x', y') \in E_\sigma \ (x \leq x' \Leftrightarrow y' \leq y)$.

4 Applied λ-calculi

The type frames introduced in the previous section are models of simply typed λ-calculi endowed with constants, called applied λ-calculi:

Definition 6. *Given a set* K *of ground types, an* applied λ-calculus Λ *over* K *is given by a family of typed constants* $C(\Lambda)_\sigma$*, indexed by simple types over* K*.*

The terms of the calculus are simply typed λ*-terms built by application and* λ*-abstraction starting from the typed constants and variables.*

The operational semantics of an applied λ*-calculus is specified by a set of* δ*-rules, stipulating the behaviour of the constants, and by the* β*-rule.*

As a matter of notation, we will write Λ_σ (resp. Λ^0_σ) for the set of terms (resp. closed terms) of type σ.

The usual, environment based, interpretation of an applied λ-calculus in a type frame $\mathcal{M}$, given a suitable map C from the constants of the calculus to elements of the appropriate type, is defined as follows:

- $[\![x]\!]^{\mathcal{M}}_\rho = \rho(x)$, $[\![c]\!]^{\mathcal{M}}_\rho = C(c)$
- $[\![PQ]\!]^{\mathcal{M}}_\rho = [\![P]\!]^{\mathcal{M}}_\rho([\![Q]\!]^{\mathcal{M}}_\rho)$, $[\![\lambda x : \sigma.P]\!]^{\mathcal{M}}_\rho = d \in \mathcal{M}_\sigma \mapsto [\![P]\!]^{\mathcal{M}}_{\rho[x \leftarrow d]}$

The interpretation of a closed term P will be noted simply $[\![P]\!]^{\mathcal{M}}$, or $[\![P]\!]$, when non ambiguous.

Two conditions have to be satisfied for a type frame $\mathcal{M}$ to be a model of a given applied λ-calculus:

1. In the fourth item of the definition above, one has that the function $d \in \mathcal{M}_\sigma \mapsto [\![P]\!]^{\mathcal{M}}_{\rho[x \leftarrow d]}$ is an element of $\mathcal{M}_{\sigma \to \tau}$, for the appropriate type τ.
2. The map C is sound: for any δ-rule $P \to^\delta Q$, one has $[\![P]\!]^{\mathcal{M}} = [\![Q]\!]^{\mathcal{M}}$.

When these conditions are satisfied, the model is such that $P \to^* Q \Rightarrow [\![P]\!]^{\mathcal{M}} = [\![Q]\!]^{\mathcal{M}}$, where the rewriting relation $\to$ is the contextual closure of $\to^\beta \cup \to^\delta$.

Note that, for the type frames $\mathcal{S}$,$\mathcal{C}$,$\mathcal{E}$ and $\mathcal{L}$, the first condition is always satisfied, since for all of them $[\![\sigma \to \tau]\!]$ is the exponential object $[\![\tau]\!]^{[\![\sigma]\!]}$ in some Cartesian closed category (actually there exists a ccc which is an ambient category for all those type frames, namely the category of finite partial orders and

monotone functions). Hence, in order to show that a given type frame among $\mathcal{S}$,$\mathcal{C}$,$\mathcal{E}$ and $\mathcal{L}$ is a model of a given applied λ-calculus, it is sufficient to provide a sound interpretation of the constants of the language.

Among the models of a given applied λ-calculus, the fully complete ones are those whose elements are definable (in general, one asks for the definability of *finite* elements. We skip this condition here since we focus on finite type frames).

Definition 7. *A model $\mathcal{M}$ of an applied λ-calculus Λ is* fully complete *if for all σ and for all $d \in \mathcal{M}_\sigma$ there exists a closed Λ-term D of type σ such that $[\![D]\!]^{\mathcal{M}} = d$.*

We define now the applied λ-calculi that will be used in the sequel to prove some extensional collapse situations.

We call these calculi $\Lambda^{\mathcal{S}}$ and $\Lambda^{\mathcal{C}}$ respectively, to emphasise the full completeness of the corresponding type frames.

4.1 The basic boolean calculus

Here is the definition of the constants of $\Lambda^{\mathcal{S}}$ and of their operational semantics:

Definition 8. – $C(\Lambda^{\mathcal{S}})_{\mathrm{bool}} = \{\mathtt{true}, \mathtt{false}\}$
– $C(\Lambda^{\mathcal{S}})_{\mathrm{bool}\to\mathrm{bool}\to\mathrm{bool}\to\mathrm{bool}} = \{\mathtt{if}\}$
– $\mathtt{if\ true}\ M\ N \to^{\delta} M$, $\mathtt{if\ false}\ M\ N \to^{\delta} N$

Lemma 2. *$\mathcal{S}$ is a fully complete model of $\Lambda^{\mathcal{S}}$.*

Sketch of the proof. First of all, the interpretation mapping the ground constants on the corresponding boolean values and such that $[\![\mathtt{if}]\!]^{\mathcal{S}} d\ e\ f = \begin{cases} e & \text{if } d = tt \\ f & \text{if } d = ff \end{cases}$ is clearly sound, hence $\mathcal{S}$ is a model of $\Lambda^{\mathcal{S}}$.

Concerning full completeness : consider an element $f = \{(a_1, b_1), \ldots, (a_n, b_n)\} \in \mathcal{S}_{\sigma\to\tau}$. If the a_i, the b_j and the equality predicate on $\mathcal{S}_\sigma$ were $\Lambda^{\mathcal{S}}$-definable, then it would be easy to write down a $\Lambda^{\mathcal{S}}$-term defining f, as a sequence of nested `if`.

On the other hand, one can $\Lambda^{\mathcal{S}}$-define the equality predicate on $\mathcal{S}_{\sigma\to\tau}$ if one is able to define the elements of $\mathcal{S}_\sigma$ and the equality predicate on $\mathcal{S}_\tau$.

Hence, the proof consists in a straightforward simultaneous induction on types, for the two following properties:

DEF(σ): all the elements of $\mathcal{S}_\sigma$ are $\Lambda^{\mathcal{S}}$-definable.

EQ(σ): the equality predicate on $\mathcal{S}_\sigma$ is $\Lambda^{\mathcal{S}}$-definable.

By the way, EQ(bool) is provided by the term

`λx:bool y:bool. if x (if y true false) (if y false true).`

Let $P(\sigma) = \forall e \in \mathcal{S}_\sigma \exists P \in \Lambda^{\mathcal{S}0}_{\sigma}\ [\![P]\!]\mathcal{S} = e$,

Lemma 3. *$\mathcal{C}$ and $\mathcal{E}$ are models of $\Lambda^{\mathcal{S}}$.*

Proof. The ground constants `true` and `false` are interpreted by the corresponding booleans, in both models. Here is the interpretation of the constant `if` in the two models:

$$[\![\texttt{if}]\!]^{\mathcal{C}} d\ e\ f = \begin{cases} e & \text{if } d = tt \\ f & \text{if } d = ff \\ \bot & \text{otherwise} \end{cases} \qquad [\![\texttt{if}]\!]^{\mathcal{E}} d\ e\ f = \begin{cases} e & \text{if } d = tt \\ f & \text{if } d = ff \\ \top & \text{otherwise} \end{cases}$$

these interpretations clearly validate the δ-rules concerning `if`.

4.2 Adding non-termination

Non-termination can be added to the basic calculus by means of fixpoint combinators, for instance. Nevertheless, a single new constant Ω : bool, whose intended interpretation is the undefined boolean value $\bot$ is enough. In fact, fixpoint combinators do not add expressive power in the case of finite ground types, apart from the possibility of divergence. In order to achieve the full completeness of $\mathcal{C}$, we add a parellel-or constant to the language, following Plotkin [13].

$\Lambda^{\mathcal{C}}$ is the extension of $\Lambda^{\mathcal{S}}$ defined as follows:

Definition 9. *$\forall\sigma\ C(\Lambda^{\mathcal{S}})_\sigma \subseteq C(\Lambda^{\mathcal{C}})_\sigma$ and:*

- $\Omega \in C(\Lambda^{\mathcal{C}})_{\text{bool}}$
- $C(\Lambda^{\mathcal{C}})_{\text{bool}\to\text{bool}\to\text{bool}} = \{\texttt{por}\}$
- $\texttt{por}\ \texttt{true}\ M\ \to^{\delta} \texttt{true}$, $\texttt{por}\ M\ \texttt{true} \to^{\delta} \texttt{true}$
- $\texttt{por}\ \texttt{false}\ \texttt{false} \to^{\delta} \texttt{false}$

Lemma 4. *$\mathcal{C}$ is a fully complete model of $\Lambda^{\mathcal{C}}$.*

Proof. This is a corollary of Plotkin's proof of full abstraction of the Scott model of parallel PCF [13].

Lemma 5. *$\mathcal{L}$ is a model of $\Lambda^{\mathcal{C}}$.*

Proof. The ground constants `true` and `false` and Ω are interpreted by tt, ff and $\bot$ respectively. Here there are the interpretations of `if` and `por` in $\mathcal{L}$:

$$[\![\texttt{if}]\!]^{\mathcal{L}} d\ e\ f = \begin{cases} \bot & \text{if } d = \bot \\ e & \text{if } d = tt \\ f & \text{if } d = ff \\ \top & \text{if } d = \top \end{cases}$$

$[\![\texttt{por}]\!]^{\mathcal{L}} d\ e = \{tt \mid d \geq tt \text{ or } e \geq tt\} \vee \{ff \mid d \geq ff \text{ and } e \geq ff\}\}$

Otherwise stated, the interpretation of `por` in $\mathcal{L}$ is given by the following truth table:

por	$\bot$	ff	tt	$\top$
$\bot$	$\bot$	$\bot$	tt	tt
ff	$\bot$	ff	tt	$\top$
tt	tt	tt	tt	tt
$\top$	tt	$\top$	tt	$\top$

This interpretation validates the δ-rules concerning `if` and `por`.

Notice that the interpretation of the parallel-or constant in $\mathcal{L}$ given above is the only one which is sound with respect to the δ-rules of `por`. Actually:

$\sharp\sharp = [\![\texttt{true}]\!]^{\mathcal{L}} = [\![\texttt{por true } x]\!]^{\mathcal{L}}_{[x:=\top]} = [\![\texttt{por}]\!]^{\mathcal{L}}\sharp\sharp\top$

5 Extensional Collapse Situations

In this section, we prove the six extensional collapse situations exhibited in Figure 1. An effective way of providing an extensional collapse situation $\mathcal{M} \to^{\mathrm{E}} \mathcal{N}$, is to define an applied λ-calculus $\Lambda^{\mathcal{N}}$ such that $\mathcal{N}$ is a fully complete model of $\Lambda^{\mathcal{N}}$, and $\mathcal{M}$ is a model of $\Lambda^{\mathcal{N}}$; then full completeness may be used to exhibit a suitable pre-logical surjection. The key fact is expressed by the fundamental lemma of (pre-)logical relations below:

Lemma 6 ([9], Lemma 4.1). *If $\mathcal{M}$, $\mathcal{N}$ are models of an applied λ-calculus Λ, and* E *is a pre-logical relation between $\mathcal{M}$ and $\mathcal{N}$, such that for all constants $c : \sigma$ of Λ, $([\![c]\!]^{\mathcal{M}}, [\![c]\!]^{\mathcal{N}}) \in \mathrm{E}_\sigma$, then, for all types σ and for all $P \in \Lambda^0_\sigma$, $([\![P]\!]^{\mathcal{M}}, [\![P]\!]^{\mathcal{N}}) \in E_\sigma$.*

Corollary 1. *Let $\mathcal{M}$, $\mathcal{N}$ be type frames over K, and suppose that there exists an applied λ-calculus Λ such that $\mathcal{N}$ is fully complete for Λ.*

Then there exist a pre-logical surjection E such that $\mathcal{M} \to^{\mathrm{E}} \mathcal{N}$.

Proof. Define $E_\sigma = \{([\![P]\!]^{\mathcal{M}}, [\![P]\!]^{\mathcal{N}}) \mid P \in \Lambda^0_\sigma\}$.

As shown in [9], Exemple 3.7, this is a pre logical relation.

We have to prove that E_σ is a partial surjective function, for all σ. For ground types, the statement holds by full completeness of $\mathcal{N}$, under the hypothesis that different ground constants of Λ are not identified in $\mathcal{M}$, that we take for granted.

By using Lemma 1 we are left with proving that E is surjective at higher types, and this is guaranteed by the full completeness of $\mathcal{N}$.

Given an applied λ-calculus Λ and one of its models $\mathcal{M}$, let $Th^{\Lambda}(\mathcal{M}) = \{P = Q \mid P, Q \in \Lambda^0 \text{ and } [\![P]\!]^{\mathcal{M}} = [\![Q]\!]^{\mathcal{M}}\}$.

Corollary 2. *Let $\mathcal{M}$ and $\mathcal{N}$ be models of an applied λ-calculus Λ, and $\mathcal{M} \to^{E} \mathcal{N}$ be an extensional collapse situation. If, for all constants $c : \sigma$ of Λ, $([\![c]\!]^{\mathcal{M}}, [\![c]\!]^{\mathcal{N}}) \in E_\sigma$, then $Th^{\Lambda}(\mathcal{M}) \subseteq Th^{\Lambda}(\mathcal{N})$.*

Proof. Let us suppose that $[\![P]\!]^{\mathcal{M}} = [\![Q]\!]^{\mathcal{M}}$, for two given closed Λ-terms P and Q. By lemma 6 we have that $([\![P]\!]^{\mathcal{M}}, [\![P]\!]^{\mathcal{N}}), ([\![Q]\!]^{\mathcal{M}}, [\![Q]\!]^{\mathcal{N}}) \in E_\sigma$. Since E_σ is a function, we conclude that $[\![P]\!]^{\mathcal{N}} = [\![Q]\!]^{\mathcal{N}}$.

Here is our main result:

Proposition 3. *The following extensional collapse situations do hold:*

$\mathcal{C} \to \mathcal{S}$, $\mathcal{E} \to \mathcal{S}$, $\mathcal{C} \to \mathcal{E}$, $\mathcal{E} \to \mathcal{C}$, $\mathcal{L} \to \mathcal{C}$, $\mathcal{L} \to \mathcal{E}$.

Proof. – $\mathcal{C} \to \mathcal{S}$ and $\mathcal{E} \to \mathcal{S}$ follow from Corollary 1, where the applied λ-calculus is $\Lambda^{\mathcal{S}}$, and Lemmas 2, 3.
– $\mathcal{C} \to \mathcal{E}$ and $\mathcal{E} \to \mathcal{C}$ follow from Proposition 2
– $\mathcal{L} \to \mathcal{C}$ follows from Corollary 1, where the the applied λ-calculus is $\Lambda^{\mathcal{C}}$, and Lemmas 4, 5.
– $\mathcal{L} \to \mathcal{E}$ follows from the two items above and Proposition 1.

Note that, by composition of extensional collapse situations, we obtain $\mathcal{L} \to \mathcal{S}$.

By Corollary 2, the proposition above yields in particular:

Corollary 3. – $\mathrm{Th}^{\Lambda^S}(\mathcal{C}) \subseteq \mathrm{Th}^{\Lambda^S}(\mathcal{S})$
– $\mathrm{Th}^{\Lambda^C}(\mathcal{L}) \subseteq \mathrm{Th}^{\Lambda^C}(\mathcal{C})$

The examples presented in Section 1 show that the inclusions above are strict.

6 Conclusion and further work

We have provided a general definition of "inclusion" of models of higher order, functional computations via the notion of extensional collapse situation, and shown this notion at work on some simple models over bool.

As suggested by the title, this work should be considered as a first step settling the ground for the study of extensional collapse situations on more complicated models. As a matter of fact, the definitions of section 3 have to be generalised a bit to encompass non extensional models. In particular, if $\mathcal{C}$ is a Cartesian closed category without enough points, the corresponding type frame $\mathcal{C}_\sigma$ is defined by:

– $\mathcal{C}_{\mathrm{bool}} = \mathcal{C}(1, C_{\mathrm{bool}})$ for a suitable object C_{bool}
– $\mathcal{C}_{\sigma\to\tau} = \mathcal{C}(1, \mathcal{C}_\sigma \Rightarrow \mathcal{C}_\tau)$

and, given $f \in \mathcal{C}_{\sigma\to\tau}$ and $x \in \mathcal{C}_\sigma$, one defines $f(x) = ev \circ <f, x>$.

Then, the fundamental lemma 6 holds and the construction of extensional collapse situations via applied λ-calculi goes through. This generalisation of logical relations for categories without enough points is described for instance in [2], section 4.5.

Let us introduce a direction for further work via an example, referring to the ones showed in the introduction. The terms:

`λx:bool. x`

`λx:bool. if x then x else x`

get the same interpretation in all the models considered in the present paper. They get different interpretation, for instance[11], in the model $\mathcal{R}$ of set and relations [8], endowed with the finite multiset exponential (see for instance [4] for a direct description of that model as a Cartesian closed category).

[11] Actually, these terms get different interpretations in all models for which *linearity* is an extensional property. This character of the model is often termed "resource sensitivity".

Using Ehrhard's result on the extension collapse of $\mathcal{R}$ over $\mathcal{L}$ [7], we should obtain the following chain of extensional collapse situations:

$\mathcal{S} < \mathcal{C} \equiv \mathcal{E} < \mathcal{L} < \mathcal{R}$

An interesting sub-problem is the definition of an extension of (finitary) PCF w.r.t. which $\mathcal{L}$ is fully complete.

What comes after? More intensional models, like game models, could collapse over $\mathcal{R}$. Also, where is the place of models based on stable orderings, as the stable or strongly stable ones, in this hierarchy?

We already know some partial answer (for instance, the fact that Berry and Curien's sequential algorithms collapse over the model of strongly stable functions [6]).

Quite a number of different models of higher order, functional computation exist in literature.

Having a better overview of the poset of extensional collapse situations relating these models may contribute to a better understanding of the whole picture.

Acknowledgments. Thanks to an anonymous referee of a previous version of this work, for suggesting that pre-logical relations are the proper framework to deal with extensional collapse situations.

References

1. Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Inf. Comput.*, 163(2):409–470, 2000.
2. Roberto Amadio and Pierre-Louis Curien. *Domains and lambda-calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998.
3. Antonio Bucciarelli. Bi-models: Relational versus domain-theoretic approaches. *Fundam. Inform.*, 32(3-4):251–266, 1997.
4. Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Not enough points is enough. In Jacques Duparc and Thomas A. Henzinger, editors, *CSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 298–312. Springer, 2007.
5. Antonio Bucciarelli and Ivano Salvo. Totality, definability and boolean ciruits. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 808–819. Springer, 1998.
6. Thomas Ehrhard. Projecting sequential algorithms on strongly stable functions. *Annals of Pure and Applied Logic*, 77:201–244, 1996.
7. Thomas Ehrhard. The Scott model of linear logic is the extensional collapse of its relational model. Technical report, 2009. HAL hal-00369831, to appear in Theoretical Computer Science.
8. Jean-Yves Girard. Normal functors, power series and lambda-calculus. *Annals of Pure and Applied Logic*, 37(2):129–177, 1988.
9. Furio Honsell and Donald Sannella. Pre-logical relations. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *CSL*, volume 1683 of *Lecture Notes in Computer Science*, pages 546–561. Springer, 1999.

10. J. M. E. Hyland and C.-H. Luke Ong. On Full Abstraction for PCF: I, II, and III. *Inf. Comput.*, 163(2):285–408, 2000.
11. Peter W. O'Hearn and Jon G. Riecke. Kripke logical relations and PCF. *Information and Computation*, 120(1):107–116, 1995.
12. Gordon Plotkin. Lambda definability in the full type hierarchy. In J. P. Seldin and J. R. Hindley, editors, *To H.B.Curry : essays on combinatory logic, lambda calculus, and formalism*, pages 363–373. Academic Press, 1980.
13. Gordon D. Plotkin. LCF considered as a programming language. *Theor. Comput. Sci.*, 5(3):225–255, 1977.
14. Kurt Sieber. Reasoning about sequential functions via logical relations. In M. Fourman, P. Johnstone, and A.Pitts, editors, *Proceedings of the LMS Symposium on Applications of Categories in Computer Science*, volume 177 of *LMS Lecture Notes Series*. Cambridge University Press, 1992.