Totality, Definability and Boolean Circuits

Antonio Bucciarelli and Ivano Salvo

Università di Roma "La Sapienza", Dipartimento di Scienze dell'Informazione,
via Salaria, 113 - 00198 Rome (Italy)
e-mail:{buccia,salvo}@dsi.uniroma1.it

Abstract. In the type frame originating from the flat domain of boolean values, we single out elements which are hereditarily total. We show that these elements can be defined, up to total equivalence, by sequential programs. The elements of an equivalence class of the totality equivalence relation (totality class) can be seen as different algorithms for computing a given set-theoretic boolean function. We show that the bottom element of a totality class, which is sequential, corresponds to the most eager algorithm, and the top to the laziest one. Finally we suggest a link between size of totality classes and a well known measure of complexity of boolean functions, namely their sensitivity.
Keywords: Logical Relations, Scott's Model, PCF, Boolean Circuits.

1 Introduction

Adding parallel constants to a programming language strictly increase the expressive power of the language, in general. For instance, extending Scott's PCF with parallel-or, one can define any finite continuous function [7]. However, it is an open problem whether parallelism adds expressive power, if we restrict our attention to total functions. Totality is a natural notion in domain theory: a ground object (such as an integer or a boolean), is total if it is defined (i.e. different from $\perp$), and a function is total if it gives total values on total arguments. Hence totality is a logical predicate [6]. An equivalent definition of totality may be given in terms of a logical (partial) equivalence relation: at ground types, $x \sim^T y$ if x and y are equal and different from $\perp$; at higher types, $f \sim^T g$ if, whenever $x \sim^T y$ at the appropriate type, then $f(x) \sim^T g(y)$. It turns out that $f \sim^T f$ if and only if f is total in the previously defined sense. Parallel-or is total, and it is $\sim^T$-equivalent to the strict-or function, which is sequential (PCF-definable). Our original motivation for this work was to explore the following conjecture, due to Berger [2]: *For any total, parallel function f there exists a sequential function g such that $f \sim^T g$*, where "parallel" means definable by PCF$^+$ (PCF extended by parallel-or), "sequential" means PCF-definable and the type frame we refer to is the Scott hierarchy of continuous functions over the flat domains of integer and boolean values[1].

[1] Berger's conjecture is slightly complicated by the fact that, for infinite types, one has to take into account also the $\exists$ functional.

If the conjecture holds, then parallelism is inessential for defining total functions, since $f \sim^T g$ means intuitively that f and g "compute" the same total function. Since f and g are *functions*, this last statement deserves some explanations: whether Scott's semantics is concerned with functions or with algorithms is a matter of the intended source language. If, as in the case of PCF, the language has built-in divergence (e.g. via fixpoints operators), then partially defined object are first class, and two programs which provide the same results on all total arguments can be operationally different. But if we restrict ourselves to total objects, then the behaviour of a given function on non-total argument is irrelevant, and we can say, for instance, that the parallel-or and the strict-or are two algorithms computing the (total) logical disjunction (namely, the laziest and the most eager algorithm, respectively).

In order to make this intuition precise we define a (binary) heterogeneous logical relation between the Scott's type frame[2] and the one of set-theoretic functions over the set {true, false}, which, at the ground type, is the identity restricted to total elements. By this relation we can define a bijection between set theoretic functions and lattices of continuous functions "implementing" them: these lattices are exactly the equivalence classes of $\sim^T$ (*totality classes*). We can summarize the situation by saying that the set theoretic type frame is the collapse of the Scott's type frame by the totality partial equivalence relation.

Then we turn our attention to first order, set theoretic functions, i.e. to functions taking tuples of booleans as argument and giving a boolean as result. These are particular kind of boolean circuits, known as *formulae* in Complexity Theory [3]. Since our construction provides, for any formula, a lattice of continuous functions implementing it, it is natural to ask if there is any relation between the structure of this lattice and the complexity of the formula. For the time being, we are able to characterize the most eager (resp. the laziest) algorithm for a given formula as the bottom (resp. the top) of the totality class of the formula. We also suggest a relation between the size of totality classes and the *sensitivity* of formulae [10].

We assume some familiarity with the language PCF, its parallel extension PCF^+ and their continuous model [7].

1.1 Overview of the Results

In Sect. 2, we show that Berger's conjecture holds for finite types. Our proof is based on the semantic characterization of bottom elements of totality classes, which turn out to be PCF-definable, at any type. This is in fact an alternative proof of a (more general) result due to Plotkin [8], showing that the conjecture does hold at any type where ι (the type of integers) does not occur negatively. In Sect. 2.1, we discuss this result and argue about the relevance of our approach.

In Sect. 3, we define the "heterogeneous" logical relation between the Scott's and set theoretic type frames. This relation is a surjective partial function, and it

[2] To be precise, we consider only finite types, i.e. the type frame of Scott continuous functions over the flat domains of boolean values.

induces a partial equivalence relation on the Scott's type frame; we show that it is exactly the totality logical relation. Hence we have a bijection between totality classes and boolean functionals, at any type.

In Sect. 4 we turn our attention to first order functions: we show that the bottom and top elements of a totality class implement respectively the most eager and the laziest algorithm for the total (set-theoretic) function corresponding to that class, via the heterogeneous relations. We also provide two (families of) terms of PCF, $\mathcal{B}_n$ and $\mathcal{T}_n$ such that for any n-ary function f in a given totality class, $[\![\mathcal{B}_n]\!] f$ is the bottom and $[\![\mathcal{T}_n]\!] f$ the top element of that class. We relate the size of totality classes to the sensitivity of the corresponding boolean function, and we discuss the relationship between lazy and parallel computations for implementing a given boolean function.

We are not aware of previous attempts of using denotational semantics in order to study the complexity of boolean circuits. For the simple and tight connection it establishes between circuits and continuous function, this work seems to provide a solid ground for such investigation.

1.2 Related Works

We have already discussed the connections with Plotkin's work on totality [8]. As for the heterogeneous relation described in Sect. 3, it is reminiscent of two recent works of T. Ehrhard [5] and N. Barreiro and T. Ehrhard [1]. In the former, it was proved that strongly stable functions are the extensional collapse of Berry-Curien's sequential algorithms, in the latter, that the set-theoretic coherent model of intuitionistic linear logic is the extensional collapse of the multiset-theoretic one.

2 The Definability Result

Definition 1. *The* simple finite types *(SFT) are defined by*

$$\sigma = o \mid \sigma \to \sigma$$

Definition 2. *The* simple finite type hierarchy $\{D_\sigma\}_{\sigma \in SFT}$ *is inductively defined by:*

D_o *is the flat domain of boolean values.*

$D_{\sigma \to \tau}$ *is the set of monotone functions from* D_σ *to* D_τ*, ordered pointwise.*

Definition 3. *The* totality logical relation $\{\sim^T_\sigma\}_{\sigma \in SFT}$, $\sim^T_\sigma \subseteq D_\sigma \times D_\sigma$ *is inductively defined by:*

$x \sim^T_o y$ *if* $x = y \neq \bot$

$f \sim^T_{\sigma \to \tau} g$ *if for all* $x \sim^T_\sigma y$, $f(x) \sim^T_\tau g(y)$

An element $x \in D_\sigma$ which is invariant with respect to the totality relation (i.e. such that $x \sim^T_\sigma x$) is called a *total* element. For $x, y \in D_\sigma$, the notation "$x \uparrow y$" stands for: "x and y have a common upper bound".

Proposition 1. *For all $\sigma \in SFT$:*

1. *$\sim^T_\sigma$ is a partial equivalence relation over D_σ*
2. *if $x \sim^T_\sigma y$ then $x \wedge y \sim^T_\sigma y$*
3. *if $x, y \in D_\sigma$ are such that $x \sim^T_\sigma x$ and $x \leq y$; then $x \sim^T_\sigma y$*
4. *for all $x \in D_\sigma$ there exists $y \in D_\sigma$ such that $x \leq y$ and y is total.*
5. *if $x \sim^T_\sigma y$ then $x \uparrow y$*

Proof. Statements 1,2 and 3 are easily proved by induction on SFT. For the second one remark that all D_σ's are finite, bounded complete cpo's, hence any set of elements does have a greatest lower bound. As for statement 4, recall that all elements of D_σ are definable by Plotkin's parallel extension of PCF [7]. Let M_x be a term defining x, and M_y be the term obtained by replacing all occurrences of Ω in M_x by, say, *true*. We have that $[\![M_y]\!] \geq x$ and, by the Basic Lemma of logical relations, that $[\![M_y]\!]$ is total[3]. Statement 5 is an easy consequence of 4. □

Fact 1 *Let $\sigma \in SFT$, and $[x] \subseteq D_\sigma$ be an equivalence class of $\sim^T_\sigma$ (hereafter, a "totality class"), then $\bigwedge\{y \in D_\sigma \mid y \in [x]\} \in [x]$.*

This is a trivial consequence of the second statement of Prop. 1 and of finiteness of D_σ. We call *canonical elements* the greatest lower bounds of totality classes. If x is total (i.e. $x \sim^T_\sigma x$), then $\underline{x}$ stands for the canonical element of $[x]$, and we note $\mathrm{CAN}(\sigma)$ the set of canonical elements of D_σ. Totality classes are clearly (finite) lattices by the previous fact and by Prop. 1.3–5.

Lemma 1. *Let $c, d \in \mathrm{CAN}(\sigma)$; then either $c = d$ or $c \not\uparrow d$.*

Proof. Let us suppose that there exists $e \in D_\sigma$ such that $c, d \leq e$. By Prop. 1, $c \sim^T_\sigma e$ and $d \sim^T_\sigma e$, hence $c \sim^T_\sigma d$ and, by canonicity, $c \leq d$ and $d \leq c$. □

In the rest of this section, we prove that Berger's conjecture holds for SFT, by showing that canonical elements are sequential, at any type. First, we provide a semantic characterization of canonical elements in term of their *traces*.

Definition 4. *Let $f \in D_{\sigma\to\tau}$, we define the* trace *of f, notation $tr(f)$, by:*

$$tr(f) = \{(c, d) \mid c \in D_\sigma,\ d \in D_\tau,\ d \neq \perp_\tau,\ f(c) = d,\ \forall c' < c,\ f(c') < d\}\ .$$

The idea behind the definition of traces is that $tr(f)$ is what remains of the graph of f once we retire from it all the information that can be inferred by the monotonicity of f. In particular, $f(x) = \bigvee\{d \mid \exists (c, d) \in tr(f),\ c \leq x\}$.

As traces are subsets of cartesian products, we note $\pi_1(t)$ (resp. $\pi_2(t)$) the first (resp. second) projection of the trace t.

[3] This argument is due to Plotkin [8]. It relies on the fact that finite functions can be defined without using fixpoint operators, and that fixpoints and Ω are the only constants of PCF whose standard interpretation is non-total.

Remark that, for $f, g \in D_{\sigma\to\tau}$, $f \leq g$ if and only if for all $(c,d) \in tr(f)$ there exists $(c',d') \in tr(g)$ such that $c \geq c'$ and $d \leq d'$. Moreover, any subset T of $D_\sigma \times D_\tau$ such that, if $(c,d),(c'd') \in T$ then $c \leq c' \Rightarrow d \leq d'$ and $c \uparrow c' \Rightarrow d \uparrow d'$, is the trace of a monotone function.

Next two lemmas provide a characterization of the traces of canonical elements.

Lemma 2. *If $f \in D_{\sigma\to\tau}$ is total, and $f' : D_\sigma \to D_\tau$ is defined by:*

$$f'(x) = \begin{cases} \underline{f(\underline{x})} & \text{if } x \sim^T_\sigma x \\ \bot & \text{otherwise} \end{cases}$$

then $f' \in D_{\sigma\to\tau}$, $f \sim^T_{\sigma\to\tau} f'$ and $f' \leq f$.

Proof. First, f' is a monotone function, since if $x \leq y \in D_\sigma$ is such that $f'(x) \neq \bot$, then $xR_\sigma x$, hence by Prop. 1, $x \sim^T_\sigma y$ and $y \sim^T_\sigma y$. Since $\underline{x} = \underline{y}$, we get $f'(x) = f'(y)$. Let us now check that $f \sim^T_{\sigma\to\tau} f'$: if $x \sim^T_\sigma x'$, then

$$f(x) \sim^T_\tau f(\underline{x}) = f(\underline{x'}) \sim^T_\tau \underline{f(\underline{x'})} = f'(x')$$

Last, $f' \leq f$ holds trivially. □

Lemma 3. *A function $f \in D_{\sigma\to\tau}$ is canonical if and only if $\pi_1(tr(f)) = \mathrm{CAN}(\sigma)$ and $\pi_2(tr(f)) \subseteq \mathrm{CAN}(\tau)$.*

Proof. The "only if" part follows from the previous lemma, since the function f' defined above is clearly such that $\pi_1(tr(f')) = \mathrm{CAN}(\sigma)$ and $\pi_2(tr(f')) \subseteq \mathrm{CAN}(\tau)$. As for the "if" part, if f is such that $\pi_1(tr(f)) = \mathrm{CAN}(\sigma)$ and $\pi_2(tr(f)) \subseteq \mathrm{CAN}(\tau)$, then f is total, since $x \sim^T_\sigma y \Rightarrow f(x) = f(y) = f(\underline{x}) \in \mathrm{CAN}(\tau)$, by Lemma 1, and if $f \sim^T_{\sigma\to\tau} f'$, then for all $x \in D_\sigma$, $f(x) \neq \bot \Rightarrow x \sim^T_\sigma x$, hence $f(x) \sim^T_\tau f'(x)$; moreover $f(x)$ is canonical, again by Lemma 1, hence $f(x) \leq f'(x)$. □

An element $x \in D_\sigma$ is *definable* if there exists a closed PCF-term $M_x : \sigma$ such that $[\![M_x]\!] = x$.

Definition 5. *If $A \subseteq D_\sigma$, then A is*

- definable *if for all $x \in A$, x is definable.*
- testable *if for all $x \in A$, there exists a closed PCF-term $N_x : \sigma \to o$ such that*

$$[\![N_x]\!](y) = \begin{cases} tt & \text{if } x \leq y \\ ff & \text{if } \exists z \in A,\ z \neq x \text{ and } z \leq y \\ \bot & \text{otherwise} \end{cases}$$

Remark that if A is testable, then the elements of A are pairwise unbounded.

The next lemma shows that all canonical elements are definable. We use the following abbreviations: $TEST(\sigma)$ (resp. $DEF(\sigma)$) stands for "$\mathrm{CAN}(\sigma)$ is testable" (resp. "$\mathrm{CAN}(\sigma)$ is definable"). Moreover, if $\mathrm{M}, \mathrm{N} : \sigma_1 \to \ldots \to \sigma_n \to o$, and $\mathrm{P} : o$, we write "**if** P **then** M **else** N" for "$\lambda x_1 : \sigma_1 \ldots x_n : \sigma_n$. **if** P **then** $\mathrm{M}x_1 \ldots x_n$ **else** $\mathrm{N}x_1 \ldots x_n$".

Lemma 4. *For all SFP-types σ and τ:*

1. $(\mathrm{TEST}(\sigma)$ *and* $\mathrm{DEF}(\tau)) \Rightarrow \mathrm{DEF}(\sigma \to \tau)$.
2. $(\mathrm{DEF}(\sigma)$ *and* $\mathrm{TEST}(\tau)) \Rightarrow \mathrm{TEST}(\sigma \to \tau)$.

Proof. 1) Let $f \in \mathrm{CAN}(\sigma \to \tau)$, $tr(f) = \{(c_1, d_1), \ldots, (c_k, d_k)\}$, and let TEST_1, $\ldots$, $\mathrm{TEST}_k : \sigma \to o$ be the test terms for $\mathrm{CAN}(\sigma)$ (TEST_i is a term testing c_i). Moreover, let $\mathrm{M}_1, \ldots, \mathrm{M}_k : \tau$ be terms defining $d_1, \ldots, d_k$, respectively. Define $\mathrm{M}_f : \sigma \to \tau$ by:

$$\begin{aligned} \mathrm{M}_f = \lambda x : \sigma.\ & \textbf{if } \mathrm{TEST}_1 x \textbf{ then } \mathrm{M}_1 \textbf{ else} \\ & \textbf{if } \mathrm{TEST}_2 x \textbf{ then } \mathrm{M}_2 \textbf{ else} \\ & \vdots \\ & \textbf{if } \mathrm{TEST}_k x \textbf{ then } \mathrm{M}_k \textbf{ else } \Omega \end{aligned}$$

In order to show that M_f defines f, remark that, by Lemma 1, $f(x) \neq \perp$ if and only if there exists a unique $c_i \in \mathrm{CAN}(\sigma)$ such that $c_i \leq x$, and in that case $f(x) = d_i$; hence $[\![\mathrm{M}_f]\!](x) = [\![\mathrm{M}_i]\!] = d_i$. It is easy to see that the converse does hold as well.

2) Let $\mathrm{CAN}(\sigma \to \tau) = \{f_1, \ldots, f_k\}$, $tr(f_i) = \{(c_1, d_1^i), \ldots, (c_l, d_l^i)\}$, where $l = |\ \mathrm{CAN}(()\sigma)\ |$, and $[\![M_1]\!] = c_1, \ldots, [\![M_l]\!] = c_l$. Moreover, for $1 \leq r \leq l$, $1 \leq s \leq k$, let TEST_r^s be a test term for d_r^s in $\mathrm{CAN}(\tau)$. Recall that this means:

$$[\![\mathrm{TEST}_r^s]\!](y) = \begin{cases} tt & \text{if } d_r^s \leq y \\ ff & \text{if } \exists x \in \mathrm{CAN}(\tau),\ d_r^s \neq x \text{ and } x \leq y \\ \perp & \text{otherwise} \end{cases}$$

A test term for f_i is then the following:

$$\mathrm{TEST}_i = \lambda f : \sigma \to \tau.\ \mathrm{AND}(\mathrm{TEST}_1^i(f(M_1)), \ldots, \mathrm{TEST}_l^i(f(M_l)))$$

where AND is a (sequential) l-ary conjunction.

First of all, remark that for all $g \in D_{\sigma \to \tau}$, $[\![\mathrm{TEST}_i]\!](g) \neq \perp$ if and only if for all $c_i \in CAN(\sigma)$ $g(c_i)$ is total, and this is the case if and only if g is total. Moreover, $[\![\mathrm{TEST}_i]\!](g) = tt$ if and only if $g \geq f_i$. □

The previous lemma, and the fact that $DEF(o), TEST(o)$ hold trivially, prove that all canonical elements are definable.

Corollary 1. *For all type σ and all $x \in \mathrm{CAN}(\sigma)$, x is definable.*

The main result of this section trivially follows, choosing canonical elements as sequential witnesses of totality classes:

Theorem 2. *Given $\sigma \in SFT$ and $x \in D_\sigma$ such that $x \sim_\sigma^T x$, there exists a definable $y \in D_\sigma$ such that $x \sim_\sigma^T y$.*

2.1 Beyond Finite Types

In this section, we give an overview of Plotkin's argument showing that, if a simple type σ does not have negative occurrences of ι, then Berger's conjecture holds at σ [8].

Definition 6. *Given two simple types σ and τ, $\sigma \lhd \tau$ if there exist two PCF-terms $M : \sigma \to \tau$ and $N : \tau \to \sigma$ such that:*

- $[\![M]\!]$ *and* $[\![N]\!]$ *are total.*
- $[\![\lambda x^{\sigma} N(M(x))]\!] \sim^{T}_{\sigma \to \sigma} [\![\lambda x^{\sigma} x]\!]$

It is easy to see that $\lhd$ is a preorder, and that if $\sigma \lhd \sigma'$ and $\tau \lhd \tau'$ then $(\sigma \to \tau) \lhd (\sigma' \to \tau')$.

Using the PCF-definability of the *fan functional*:$((\iota \to o) \to o) \to \iota$, computing the modulus of continuity of its argument, one can prove the following lemma:

Lemma 5. *If α is a type with no negative occurrences of ι, then $\alpha \lhd \iota$.*

The fact that Berger's conjecture holds at σ, whenever σ satisfies the hypothesis of the previous lemma, follows easily: let H be a total, PCF^{++}-definable[4] functional in D_{σ}, and let $M : \sigma \to \iota$, $N : \iota \to \sigma$ be the PCF-terms given by Def. 6. If P is a PCF^{++} term defining H, than $MP : \iota$ is PCF-definable, say by $n : \iota$, and $H' = [\![Nn : \sigma]\!]$ is a PCF-definable functional such that $H' \sim^{T} H$.

All this rests on the facts that, if σ does not contains negative occurrences of ι, then the total, PCF^{++}-definable elements of D_{σ} can be enumerated by PCF terms.

This result is stronger than the one we present in Sect. 2, but still we think that our proof provides new insights on totality for finite types: first, strictly speaking, Plotkin's argument cannot be formulated in the finite framework, since it is based on enumerations; second, and more important, we provide a semantic characterization of "sequential witnesses" in totality classes. In Sect. 4 we show that, at first order, we are able to characterize also maxima of totality classes.

The validity of Berger's conjecture is an open problem, in the general, infinite case.

3 Scott's Domain and Set Theoretic Boolean Functions

In this section we establish precise relationships between the Scott's type frame and boolean set-theoretic functions and we show that there is a one-to-one correspondence between the set of totality classes and the set of set-theoretic boolean functions. In order to do this first of all we define the hierarchy of set-theoretic functions and a heterogeneous logical relation between Scott's type frame and this hierarchy.

[4] PCF^{++} is a further extension of PCF^{+} with the second order $\exists : (\iota \to o) \to o$ functional, which tests whether its argument yields the value tt on some integer.

Definition 7. *The* set-theoretic boolean functions hierarchy $\{S_\sigma\}_{\sigma \in SFT}$ *is inductively defined by:*

S_o *is the set* $\{$*true, false*$\}$

$S_{\sigma\to\tau}$ *is the set of set-theoretic functions from* S_σ *to* S_τ*, usually written* $S_\tau^{S_\sigma}$.

Definition 8. *The* heterogeneous logical relation $\{\sim_\sigma^H\}_{\sigma\in SFT}$, $\sim_\sigma^H \subseteq D_\sigma \times S_\sigma$ *is inductively defined by:*

tt $\sim_o^H$ *true and ff* $\sim_o^H$ *false*

$f \sim_{\sigma\to\tau}^H \varphi$ if *for all* $x \sim_\sigma^H a$, $f(x) \sim_\tau^H \varphi(a)$

The heterogeneous relation $\sim^H$ induces in standard way [4] a partial equivalence relation $\sim_\sigma^S$ on each D_σ. The main result of this section is that this relation coincides with totality.

Definition 9. *The* extensional collapse *induced by* $\sim^H$ *is* $\{\sim_\sigma^S\}_{\sigma\in SFT}$, $\sim_\sigma^S \subseteq D_\sigma \times D_\sigma$, *defined by:*

$$f \sim_\sigma^S g \text{ if } \textit{there exists } \varphi \in S_\sigma \textit{ such that } f \sim_\sigma^H \varphi \textit{ and } g \sim_\sigma^H \varphi$$

In order to prove that $\sim^T = \sim^S$, we introduce the following notions:

Definition 10. *A relation* $\sim \subseteq \bigcup_{\sigma\in SFT} D_\sigma \times D_\sigma$ *is* logical at σ *if either* σ *is ground or* $\sigma \equiv \sigma_1 \to \sigma_2$ *and for all* $f, g \in D_\sigma$, $f \sim g$ *iff for all* $x, y \in D_{\sigma_1}$, $(x \sim y \Rightarrow f(x) \sim g(y))$.
Moreover $\sim$ *is* logical up to σ *if for all* τ *structurally smaller than* σ, $\sim$ *is logical at* τ.

The proof of the following theorem uses the fact that if two relations $R, S \subseteq \bigcup_{\sigma\in SFT} D_\sigma \times D_\sigma$ are such that R and S are equal at type o and are both logical up to σ, then they are equal up to σ.

Theorem 3. *For all* $\sigma \in SFT$ *the following statements hold:*

SPF_σ: $\sim_\sigma^H$ *is a surjective partial function from* D_σ *to* S_σ.
LUT_σ: $\sim^S$ *is logical up to* σ.

Proof. We prove SPF and LUT by simultaneous structural induction on SFT. As for SPF, which trivially holds for the ground type, let us show that it is preserved by arrow type constructor, i.e.:

1. $\forall \varphi \in S_{\sigma\to\tau} \,.\, \exists f \in D_{\sigma\to\tau} \,.\, f \sim_{\sigma\to\tau}^H \varphi$
2. $f \sim^H \varphi$ and $f \sim^H \psi \Rightarrow \varphi = \psi$

As for the first item, by LUT_τ we know that $\sim^S$ is logical up to τ, and hence $\sim^S = \sim^T$ up to τ. Consequently equivalence classes of $\sim_\tau^S$ are totality classes and by SPF_τ we conclude that the inverse image via $\sim_\tau^H$ of any given $b \in S_\tau$ is a lattice. Hence we are legitimate to define $k : S_\tau \mapsto D_\tau$ as follows:

$$k(b) = \bigvee \{y \in D_\tau \mid y \sim_\tau^H b\}$$

Let $\varphi \in S_{\sigma\to\tau}$ we define $f \in D_{\sigma\to\tau}$ as follows:

$$f(x) = \begin{cases} k(\varphi(a)) & \text{if } \exists a.\ x \sim^H_\sigma a \\ \bot_\tau & \text{otherwise} \end{cases}$$

Let us check that $f \in D_{\sigma\to\tau}$. Let $x \leq y$ be elements of D_σ. If $f(x) = \bot$ we are done. Otherwise $\exists a \in S_\sigma$ such that:

$$x \sim^H_\sigma a \Rightarrow x \sim^S_\sigma x \Rightarrow^{(1)} x \sim^S_\sigma y \Rightarrow^{(2)} y \sim^H_\sigma a \Rightarrow f(x) = f(y)$$

where (1) follows from LUT_σ and Proposition 1.4 and (2) follows from SPF_σ. Finally, $f \sim^H_{\sigma\to\tau} \varphi$ easily follows from the construction of f.

As for the second item let $f \in D_{\sigma\to\tau}$ and $\varphi, \psi \in S_{\sigma\to\tau}$, such that $f \sim^H \varphi$ and $f \sim^H \psi$. In order to show that $\varphi = \psi$, let $a \in S_\sigma$. By SPF_σ, there exists $x \in D_\sigma$ such that $x \sim^H a$. Hence $f(x) \sim^H \varphi(a)$ and $f(x) \sim^H \psi(a)$. By SPF_τ, $\varphi(a) = \psi(a)$.

As for $\mathrm{LUT}_{\sigma\to\tau}$, we have to show that:

$$f \sim^S_{\sigma\to\tau} g \quad \text{iff} \quad \forall x \sim^S_\sigma y.\ f(x) \sim^S_\tau g(y)$$

($\Rightarrow$) Let $f \sim^S_{\sigma\to\tau} g$. By definition of $\sim^S$ there exists $\varphi \in S_{\sigma\to\tau}$ such that $f \sim^H \varphi$ and $g \sim^H \varphi$. Let $x, y \in D_\sigma$ be such that $x \sim^S_\sigma y$. By definition of $\sim^S$ there exist $a \in S_\sigma$ such that $x \sim^H a$ and $y \sim^H a$. Hence $\varphi(a)$ is such that $f(x) \sim^H \varphi(a)$ and $g(y) \sim^H \varphi(a)$ (since $\sim^H$ is a logical relation) and hence we have $f(x) \sim^S_\tau g(y)$.

($\Leftarrow$) Let f and g such that $\forall x \sim^S_\sigma y\ .\ f(x) \sim^S_\tau g(y)$. Then, by definition of $\sim^S$, $x \sim^S_\sigma y$ implies that there exists $b \in S_\sigma$ such that $x \sim^H_\sigma b$ and $y \sim^H_\sigma b$. Similarly $f(x) \sim^S_\tau g(y)$ implies that there exists $c \in S_\tau$ such that $f(x) \sim^H_\tau c$ and $g(y) \sim^H_\tau c$. Since $\sim^H_\sigma$ and $\sim^H_\tau$ are partial surjective function we are done. In fact given f and g as above, we can choose $\varphi : S_\sigma \mapsto S_\tau$ as follows: for all $b \in S_\sigma$ there exists $x \in D_\sigma$ such that $x \sim^H_\sigma b$. Furthermore for all $y \sim^S_\sigma x$, we have that $y \sim^H_\sigma b$. By hypothesis $f(x) \sim^S_\tau g(y)$ and then there exists $c \in S_\tau$ such that $f(x) \sim^H_\tau c$ and $g(y) \sim^H_\tau c$. Since $\sim^H_\tau$ is a function, this element c is uniquely determined. Clearly for the map φ, such that $b \overset{\varphi}{\mapsto} c$, we have $f \sim^H_{\sigma\to\tau} \varphi$ and $g \sim^H_{\sigma\to\tau} \varphi$. □

Corollary 2. *For all $\sigma \in SFT$ we have $\sim^S_\sigma = \sim^T_\sigma$*

Proof. It suffices to observe that at ground type o we have $\sim^S_o = \sim^T_o$ and that, by above theorem, both relations are logical. □

Now we are able to prove the existence of a bijection between totality classes and set-theoretic boolean functions.

Corollary 3. *For all $\sigma \in SFT$ there exists a bijection $\mathcal{I}_\sigma : D_{\sigma/\sim^T} \to S_\sigma$*

Proof. It suffices to show that $D_{\sigma/\sim^T} \to D_{\tau/\sim^T} \simeq D_{\sigma\to\tau/\sim^T}$.
Define $\mathcal{I}_{\sigma\to\tau}([f]_{\sim^T}) = \varphi$, where $f \sim^H \varphi$ and check that $\mathcal{I}_{\sigma\to\tau}$ is a bijection using Theorem 3. □

Corollary 3 essentially states that the extensional collapse of $\{D_\sigma\}$ by the totality relation yields exactly $\{S_\sigma\}$. We remark that this result cannot be extended to infinite types: in fact, if we add the type ι of integers to simple types, interpreted by the flat domain D_ι and the set S_ι of natural numbers, respectively, the following cardinality argument can be applied[5]. Each D_σ is an $\omega-$algebraic domain, that is the set of its compact elements is countable. This implies, by algebraicity that the cardinality of each D_σ is at most $2^{\aleph_0}$, whereas the cardinality of $S_{(\iota\to\iota)\to\iota}$ (pure type 2) is already $2^{2^{\aleph_0}}$. By the way it could be interesting to investigate the class of continuous functionals defined at type σ as the inverse image of $\sim^H_\sigma$. These functionals are total in natural sense.

4 First Order Boolean Functions

In this section we turn to first order set-theoretic functions, i.e. functions of type $\underbrace{o\to o\to\ldots\to o}_{n}\to o$ for some n[6]. These functions are known as *formulae* in complexity theory. Our goal is to use the totality class T_φ associated by $\sim^H$ to any formula φ as tool to study its computational properties. In particular any elements of T_φ is a parallel algorithm to compute φ, since elements of T_φ are PCF$^+$ definable and some of them are sequential (PCF definable). In Sect. 2 we have shown that the bottom element of T_φ is always sequential. A natural question arise about the degree of parallelism [9] of the top element of T_φ: is it the maximum degree of parallelism in T_φ? Without going into details about degrees of parallelism we can answer negatively providing an example of totality class which contains parallel elements, but whose top is sequential.

Example 1. Consider the two argument constantly true function $\kappa\in S_{o\to o\to o}$ and the following PCF$^+$ terms:

$$S_{true}=\lambda x^o,y^o.\ \mathbf{true}$$
$$P_{true}=\lambda x^o,y^o.\ \mathbf{por}(\mathbf{por}(x,y),\mathbf{por}(\mathrm{NOT}(x),\mathrm{NOT}(y)))$$

The first term is a PCF term, and one can show that no PCF term can define $[\![P_{true}]\!]$. It is easy to see, furthermore, that $[\![S_{true}]\!],[\![P_{true}]\!]\in T_\kappa$, and that $[\![S_{true}]\!]$ is the top element in T_κ.

However top elements of totality classes do have computational relevance as laziest algorithms: this means that the top element $\overline{f}$ of a given totality class $[f]_{\sim^T}$ yields a total result using only "needed" information. More formally, $\overline{f}x_1\ldots x_n$ is defined whenever f is constant on maximal points of the principal ideal of $(x_1,\ldots,x_n)$. On the other hand, $\underline{f}$ is defined exactly on maximal point of D^n_o and this implies that $\underline{f}$ is not only sequential, but also definable in call-by-value PCF.

[5] The definitions of $\sim^H$ and $\sim^S$ go through this extension, definining $\sim^H_\iota=\{(n,n)\}$.

[6] Throughout this section we abbreviate this type by $o^n\to o$. We fell free of interchanging curried and uncurried versions of it.

We are able to define two families of PCF (resp. PCF$^+$) terms $\mathcal{B}_n$ (resp.$\mathcal{T}_n$) which transform a given total function f into $\underline{f}$ (resp. $\overline{f}$). We define them inductively on the arity of f:

$$\begin{array}{ll}
\mathcal{B}_0 = \lambda x^o.x & \mathcal{T}_0 = \lambda x^o.x \\
\mathcal{B}_{n+1} = \lambda f^{o^{n+1}\to o}. & \mathcal{T}_{n+1} = \lambda f^{o^{n+1}\to o}. \\
\quad \lambda x^o.\lambda y^{o^n}.\ \textbf{if}\ x & \quad \lambda x^o.\lambda y^{o^n}.\ \textbf{pif}\ x \\
\quad\ \textbf{then}\ \mathcal{B}_n(f\ \textbf{true})y & \quad\ \textbf{then}\ \mathcal{T}_n(f\ \textbf{true})y \\
\quad\ \textbf{else}\ \mathcal{B}_n(f\ \textbf{false})y & \quad\ \textbf{else}\ \mathcal{T}_n(f\ \textbf{false})y
\end{array}$$

where **pif** is the "parallel if" constant [7], as expressive as **por**, such that $[\![\ \textbf{pif\_then\_else\_}]\!](x,b,b) = b$ and $[\![\ \textbf{pif\_then\_else\_}]\!](b) = [\![\ \textbf{if\_then\_else\_}]\!](b)$ for $b \neq \bot$.

Since it is trivial to check that $[\![\ \mathcal{B}_n]\!](f) = \underline{f}$ (just remark that $[\![\ \mathcal{B}_n]\!](f)$ is defined exactly on total tuples), we restrict ourselves to prove the correctness of the definition of $\mathcal{T}_n$.

Proposition 2. *Let $f \in D_{o^n\to o}$ be a total function. Then the following statements hold:*

1. $\mathcal{T}_n f \sim^T f$
2. $\forall g.\ g \sim^T f.\ g \leq \mathcal{T}_n f$

Proof. Induction on n. (**0**) Obvious.
(**n + 1**) As for 1, it suffices to check the definition of $\mathcal{T}_{n+1}$. Let $g \sim^T f, g \in D_{o^{n+1}\to o}$ and $x \in D_{o^{n+1}}$. We show that $g(x) \leq (\mathcal{T}_{n+1}f)x$. We distinguish two cases:

- $x_1 \neq \bot$. Since $g(x_1) \sim^T f(x_1)$, the following holds:

$$g(x_1, x_2, \ldots, x_{n+1}) \overset{curry}{=} (gx_1)x_2 \ldots x_{n+1} \leq \overset{Ind.H}{\leq} \mathcal{T}_n(fx_1)x_2 \ldots x_{n+1} \overset{def}{=} (\mathcal{T}_{n+1}f)x_1x_2 \ldots x_{n+1}$$

- $x_1 = \bot$. Suppose that $g\bot x_2 \ldots x_{n+1} = b \neq \bot$ (otherwise the statements holds trivially). By monotonicity of g, $g(tt)x_2 \ldots x_{n+1} = g(ff)x_2 \ldots x_{n+1} = b \neq \bot$. By inductive hypothesis:

$$g(tt)x_2 \ldots x_{n+1} = \mathcal{T}_n(f(tt))x_2 \ldots x_{n+1}$$

and

$$g(ff)x_2 \ldots x_{n+1} = \mathcal{T}_n(f(ff)))x_2 \ldots x_{n+1}$$

and this implies that $\mathcal{T}_{n+1}f\,ff\,x_2 \ldots x_{n+1} = b$, by definition of **pif**.

□

In order to approach the issue of sensitivity, let us consider two classes of formulae: χ_n and κ_n computing respectively the n-ary parity function and the n-ary constant **true** function. We observe that T_{χ_n}, is a singleton for all n, whereas the size of T_{κ_n} grows exponentially in n. Intuitively the χ_n's are "difficult" to compute, whereas the κ_n's are "easy". This intuition is supported by the following definition of sensitivity [10].

Definition 11. *Let* $\varphi \in S_{o^n \to o}$ *and* $x \in S_o^n$. *Let* $x^{(i)} = (x_1, \ldots, \neg x_i, \ldots x_n)$. *The* sensitivity *of* φ *on* x *is*[7]*:*

$$s_x(\varphi) = \sum_{i=1}^{n} (\varphi(x) \ominus \varphi(x^{(i)}))$$

The sensitivity of φ *is:*

$$s(\varphi) = \sum_{x \in S_o^n} s_x(\varphi)$$

We remark that the sensitivity of χ_n is $2^n n$ and the sensitivity of κ_n is 0 for all n. Hence for these classes of formulae there is an inverse proportion between the size of totality classes and sensitivity. We believe that this phenomenon is general and we conjecture that the size of T_φ is functionally related to $s(\varphi)$. We checked this fact at type $o \to o \to o$, for which the following interesting relation holds:

$$2\lfloor \log_2 |T_\varphi| \rfloor + s(\varphi) = 2^2 2$$

Indeed the property we conjecture is not surprising since if φ has a low sensitivity w.r.t. some arguments, then there are many (inessentially) different ways to compute φ, taking decisions on evaluating them or not.

References

1. Barreiro, N., Ehrhard, T.: *Anatomy of an extensional collapse.* Submittend paper. (1997). Available from `http://hypatia.dcs.qmw.ac.uk/cgi-bin/sarah?q=ehrhard`.
2. Berger, U.: *Total Objects and Sets in domain Theory.* Annals of Pure and Applied Logic **60** (1993) 91–117
3. Boppana, R. B., Sipser, M.: *The Complexity of finite functions.* In: van Leeuwen, J. (ed.): Handbook of Theoretical Computer Science, vol. A. Elsevier (1990) 759–802
4. Bucciarelli, A.: *Logical Reconstruction of Bi-Domains.* Proc. of the 3rd Int. Conf. on Typed Lambda Calculi and Applications, LNCS 1210, Springer-Verlag (1997) 99–111
5. Ehrhard, T.: *A relative definability result for strongly stable functions, and some corollaries.* (1997) To appear in Information and Computation. Available from `http://hypatia.dcs.qmw.ac.uk/cgi-bin/sarah?q=ehrhard`.
6. Mitchell, J. C.: *Type Systems for Programming Languages.* In: van Leeuwen, J. (ed.): Handbook of Theoretical Computer Science, vol. B, Elsevier (1990) 365–458
7. Plotkin, G.: *LCF considered as a programming language.* Theoretical Computer Science **5** (1997) 223–256
8. Plotkin, G.: *Full Abstraction, Totality and PCF.* Available from `http://hypatia.dcs.qmw.ac.uk/authors/P/PlotkinGD/`
9. Sazonov, V. Y.: *Degrees of Parallelism in Computations.* Proc. Conference on Mathematical Foundations of Computer Science, LNCS 45, Springer-Verlag (1976)
10. Wegener, I.: *The Complexity of Boolean functions.* Wiley-Teubner Series in Comp. Sci., New York - Stutgart (1987)

[7] $\ominus : S_o^2 \to \{0, 1\}$ yields 0 if its arguments are equal and 1 otherwise.