

## INDUCTIVE DEFINITION AND DOMAIN THEORETIC PROPERTIES OF FULLY ABSTRACT MODELS FOR PCF AND PCF<sup>+</sup>

VLADIMIR SAZONOV

Department of Computer Science, the University of Liverpool, Liverpool, L69 3BX, U.K.  
*e-mail address:* vyzsuk@yahoo.co.uk, v.sazonov@csc.liv.ac.uk

**ABSTRACT.** A construction of fully abstract typed models for PCF and PCF<sup>+</sup> (i.e., PCF + “parallel conditional function”), respectively, is presented. It is based on general notions of sequential computational strategies and wittingly consistent non-deterministic strategies introduced by the author in the seventies. Although these notions of strategies are old, the definition of the fully abstract models is new, in that it is given level-by-level in the finite type hierarchy. To prove full abstraction and non-dcpo domain theoretic properties of these models, a theory of computational strategies is developed. This is also an alternative and, in a sense, an analogue to the later game strategy semantics approaches of Abramsky, Jagadeesan, and Malacaria; Hyland and Ong; and Nickau. In both cases of PCF and PCF<sup>+</sup> there are definable universal (surjective) functionals from numerical functions to any given type, respectively, which also makes each of these models unique up to isomorphism. Although such models are non-omega-complete and therefore not continuous in the traditional terminology, they are also proved to be sequentially complete (a weakened form of omega-completeness), “naturally” continuous (with respect to existing directed “pointwise”, or “natural” lubs) and also “naturally” omega-algebraic and “naturally” bounded complete—appropriate generalisation of the ordinary notions of domain theory to the case of non-dcpo.

### 1. INTRODUCTION

**LCF**, a *Logic for Computable Functions*, was introduced in 1969 by Scott in a seminal paper [31] (published only in 1993). Its term language **PCF**—a typed version of the lambda calculus over integers and booleans with the least fixed point operator **Y**—was further considered in the middle of the seventies by Plotkin [25], Milner [21], and the author [27, 28, 29, 30]. In particular, the expressive power of **PCF** in the framework of a standard continuous model  $\{\mathbb{D}_\alpha\}$  for **PCF** was described in terms of (*sequential*) *computational strategies* as the Theorem: “*definable in PCF = sequentially computable*” [28]. Also, a *precise correspondence between operational and denotational semantics* in various formulations (and even an untyped version) was obtained in [29] (and independently, in somewhat different terms, by Hyland 1976, Plotkin 1977 and Wadsworth 1976). The full abstraction

*2000 ACM Subject Classification:* F.3.2.

*Key words and phrases:* LCF, PCF, full abstraction, sequentiality, computational strategies, game semantics, non-dcpo domain theory.

property of the standard continuous model for  $\mathbf{PCF}^+ = \text{“}\mathbf{PCF} + \text{parallel OR (or parallel if)”}$  (by definability of all finite continuous functionals) was stated, as well as the fact that  $\mathbf{PCF}^{++} = \text{“}\mathbf{PCF} + \text{parallel OR} + \exists\text{”}$  defines all computable continuous functionals (Plotkin [25] and, without publishing proofs, the author [27, 28]). Degrees of parallelism for continuous finite type functionals with various examples were introduced in [27] (see also *e.g.* [5, 34]).

A first, essentially syntactic, construction of a continuous *fully abstract model* for  $\mathbf{PCF}$  was given in 1977 by Milner [21]. The characteristic property of fully abstract models is as follows:

$$\forall \text{ ground type program contexts } \mathcal{C} \quad (\llbracket \mathcal{C}[M] \rrbracket \sqsubseteq \llbracket \mathcal{C}[N] \rrbracket) \implies \llbracket M \rrbracket \sqsubseteq \llbracket N \rrbracket$$

which says (for ‘=’ in place of ‘ $\sqsubseteq$ ’) that, if two program fragments behave equivalently in all computational contexts, then they should have the same denotational semantics. The main reason for focusing particular attention on this definition and on Milner’s model is that for the standard continuous model  $\{\mathbb{D}_\alpha\}$  and  $\mathbf{PCF}$  this natural property of the denotational semantics does *not* hold. As mentioned above,  $\mathbf{PCF}$  defines (exactly) all sequentially computable functionals, whereas the standard model contains some ‘extra’ elements, such as ‘parallel’ disjunction  $\text{OR} \in \mathbb{D}_{o,o \rightarrow o}$  and ‘parallel’ existential quantification  $\exists \in \mathbb{D}_{(\iota \rightarrow o) \rightarrow o}$ . This is essentially the reason for the violation of full abstraction. But although Milner’s fully abstract model satisfies desirable properties of continuity, it is not a satisfactory domain theoretic characterization of sequentiality due to the existence in it of non-sequential limit functionals (Normann [23]).

Also, having a syntactic nature, the definition of this model was considered as not very satisfactory in comparison with the standard model of *all* continuous functionals. Non-syntactic *game semantic* approaches to defining fully abstract models were developed by Abramsky, Jagadeesan, Malacaria [1]; Hyland, Ong [11] and Nickau [22]. Various approaches to sequentiality and full abstraction were considered also by Kahn and Plotkin [13], Berry and Curien [4], Bucciarelli and Ehrhard [6, 7]; Curien [9]; Jung and Stoughton [12]; O’Hearn and Riecke [24]; Marz, Rohr and Streicher [19, 20]; Sieber [32], Cartwright and Felleisen [8] and others. Unlike this paper some of them consider more general sequentiality concepts going outside  $\mathbf{PCF}$  and even outside the class of monotonic functionals such as sequentially realizable functionals (equivalent to some other approaches implicitly mentioned above); a unifying approach is presented by Longley [17].

Hyland and Ong [11] identified a very close analogy between the old approach to sequentiality of functionals via computational strategies in [28] and their game theoretic framework. One of the goals of this paper is to demonstrate *how computational sequential strategies could define a fully abstract model*  $\{\mathbb{Q}_\alpha\}$  for  $\mathbf{PCF}$  inductively, level-by-level in the finite type hierarchy in a direct computational way corresponding to the original definition and characterization of higher type sequentiality in [29, 28]. (The latter was applied only to the standard, non-fully-abstract continuous model  $\{\mathbb{D}_\alpha\}$  containing not only sequential functionals.) It is important to stress the straightforward, inductive character of the definition of  $\{\mathbb{Q}_\alpha\}$  which may be compared, at least partly, with the inductive definition of the continuous model  $\{\mathbb{D}_\alpha\}$ . Assuming, by induction, that we have the class of sequential functionals of types up to level  $l$ , we define what are sequential functionals of the level  $l + 1$  as those computable by sequential strategies. In this respect our approach differs from the game-semantic one based on a quotient construction for all types simultaneously. However,

proving the essential properties of the inductively defined model  $\{\mathbb{Q}_\alpha\}$  of hereditarily sequential functionals is not so direct and requires the quite involved theory of computational strategies and a quotient construction  $\tilde{\mathbb{Q}} \cong \mathbb{Q}$  giving an alternative, non-inductive definition of the same model. Reference in the inductive step of the definition of  $\mathbb{Q}$  to all sequential functionals of the previous levels also reflects the complicated character of this inductive definition. A finitary version referring only to the immediate subtypes of the given type should not be possible due to the undecidability result of Loader [16].

However, the definition of  $\mathbb{Q}$  is sufficiently straightforward, although involving some technical complications to make it mathematically correct and, additionally, to crucially simplify the correctness proof of the induction step.

As in [1], “we want to capture just those sequential computations in which the different parts or modules interact with each other in purely functional fashion” and, as in [11], “without recourse to the syntax or operational semantics of the language” (**PCF**). More precisely, we will use computability by sequential strategies to define (hereditarily) sequential functionals. Although **PCF** is a partial case of the general concept of a system of strategies, our definitions will not be reduced simply to doing things in **PCF**. We will work in the quite general terms of abstract computability in higher types in a “functional fashion”, by using “interpreted computations” (involving applicative terms) in the style of *denotational semantics*, to define a fully abstract model for **PCF**. Also note that the very term “sequential” primarily assumes “sequentially computable”. That is why involving some kind of computability approach at the level of denotational semantics is quite natural. In fact, we will also provide an alternative, generalized *operational semantics* of strategies—not only for ground types—and demonstrate that it is coherent with the denotational one (the approach originally presented in [29] but not in the “fully abstract framework” as in the present paper.) This distinction together with the interplay between operational ( $\langle\langle-\rangle\rangle$ ) and denotational ( $\llbracket-\rrbracket$ ) semantics ( $\llbracket\langle\langle A \rangle\rangle\rrbracket = \llbracket A \rrbracket$  for arbitrary finite type combinations of strategies) is one of the crucial points of this paper.

On the other hand, we read in [11] that: “we do not have a proper definition of higher-type sequentiality from first principles”. There could probably be various philosophical views concerning what are these “first principles”. However, for the simpler case of non-higher-type sequentiality, we see that its definition (say, for the conditional function **if-then-else**), reduces to the existence of a sequential strategy of computation of a function by asking of an Oracle the values of the arguments—here of a basic type. For higher types, we just extend this idea by allowing more general queries to the Oracle—applicative combinations (of a basic type) of the arguments and strategies. This approach recalls and generalizes that of Kleene [14, 15] for Turing computability of finite type functionals and is essentially an extensional one, despite its somewhat intensional-computational features, and can be also considered as a natural generalisation both of combinators and the conditional operation **if-then-else** having an evidently functional/extensional character. Moreover, this allows us to characterise, in abstract computational terms, the expressive power of **PCF** both in the standard model  $\{\mathbb{D}_\alpha\}$  of all continuous finite type functionals [28] and in the fully abstract model  $\{\mathbb{Q}_\alpha\}$  considered in this paper where all functionals prove to be definable in **PCF** + “all (one place numeric) functions of the type  $\iota \rightarrow \iota$ ”. By the way, the ordinary concept of continuous functions over dcpo domains, usually considered as non-intensional, is nothing more than a very abstract version of the idea of computability:  $fx = \bigsqcup_n fx_n$  for  $x = \bigsqcup_n x_n$  with  $fx$  of a basic type means that the value of  $fx$  can be “computed” by extracting “finite” information  $x_n$  from the argument  $x$ ; we abstract all

other details of a computation process. That is, it has some hidden intensional features. We should have just a natural balance, or interplay, between “intensional” and “extensional”. For computational strategies the former aspect corresponds to the operational semantics of strategies, and the latter is represented by the concept of interpreted computations leading to denotational semantics of strategies and to the extensional inductive definition of the fully abstract model of sequentially computable functionals.

Let us stress again, as this is an important point: denotational semantics of strategies, and thus the corresponding inductive definition of the fully abstract model, is based on *interpreted computations* in terms of “real” (applications of) finite type functionals. Therefore it has, despite computations involved, rather an extensional character, whereas the operational semantics of (combinations of) strategies is based on purely “syntactical”, *non-interpreted computations* in terms of strategies only (like in terms of the language **PCF** only) and without invoking “real” finite type functionals.

The main drawback of our approach, in comparison with game-theoretical ones, is the lack of a construction for a general category (not referring to finite types) like that of games with arrows representing suitable game strategies. However, this more concrete view allows us to construct, inductively, a monotonic order extensional fully abstract model for **PCF**, in a straightforward and natural way. Unfortunately, this inductive definition contrasts with the proof of the main domain-theoretic properties of  $\mathbb{Q}$  which involves a significant amount of machinery of computational strategies, including an isomorphic quotient construction  $\{\tilde{Q}_\alpha\} \cong \{Q_\alpha\}$  (reflected by the tilde symbol). In comparison, the game theoretic approach is based on a quotient construction in the very definition of the fully abstract model. In this respect, it looks more intensional.

It turns out that this model consists only of continuous functionals with respect to existing “pointwise”, or “natural” lubs. We need to consider this generalized and novel version of continuity, called *natural continuity*, because the poset of sequential functionals of a given type (starting with the level 3) is not  $\omega$ -complete, as was recently shown by Normann [23], and therefore this model is not isomorphic to the ‘limit-term’ model in [21]. Note that the model  $\{Q_\alpha\}$  satisfies the corresponding uniqueness property (the property formally different from, but similar to, that of the continuous fully abstract model of Milner) and is therefore isomorphic to the game models defined in [1, 11]. This leads to a generalized concept of *natural non-dcpo domains* most appropriate for describing the properties of the models of finite type functionals considered in this paper which will be shown to be sequentially complete (a weakened form of  $\omega$ -completeness), naturally continuous and also naturally  $\omega$ -algebraic and naturally bounded complete. This domain theoretic framework plays a crucial role in this paper and can serve as a kind of substitute for the categories of games mentioned above.

The more general concept of *wittingly consistent* non-deterministic computational strategies defined in [30] (Part II, §4) is also successfully used in the current paper to construct the fully abstract model  $\{W_\alpha\} \cong \{\tilde{W}_\alpha\}$  for **PCF**<sup>+</sup> satisfying definability properties such as the fully abstract model  $\{Q_\alpha\} \cong \{\tilde{Q}_\alpha\}$  for **PCF** discussed above. This gives a positive answer to the question stated in [18] (before Proposition 6):

*“It is worth remarking that there is no corresponding definability result for **PCF**<sup>+</sup>. It may well be that there can be none; it is not at all clear, however, how to even formulate a precise statement to that effect”.*

Although this question was seemingly related to the possibility of extending the game semantics results for **PCF** to **PCF**<sup>+</sup>, our approach via computational strategies is a natural and quite general alternative with some analogy to the game approach and might probably lead also to a corresponding extended game semantics solution. Note also that the fully abstract model  $\{\mathbb{W}_\alpha\}$  for **PCF**<sup>+</sup> is also *not*  $\omega$ -complete (even at the level 2)—this is clear from the known result that  $\exists$  is not definable in **PCF**<sup>+</sup>. But it is wittingly- $\omega$ -complete and satisfies all the above mentioned generalized, “natural” versions of (non-dcpo) domain theoretic properties.

**Organization.** We start with the generalized, “natural” version of non-dcpo (finite type) domain theory in Section 2. We define computational sequential strategies in Section 3 and their denotational semantics on the base of interpreted computations in Section 4. Then hereditarily sequential functionals are defined inductively (level-by-level) in Section 5. Sections 6 and 7 are devoted to demonstrating the full abstraction property of the resulting model  $\mathbb{Q} \cong \tilde{Q}$  for **PCF**. The definability of a universal functional  $U_\alpha : (\iota \rightarrow \iota) \rightarrow \alpha$  for each type  $\alpha$  is also stated, but not proved (see the details in [28]). Finitary ranked and other finite versions of strategies are introduced computing exactly all “naturally” finite sequential functionals to demonstrate the “natural” continuity of  $\tilde{Q}$  (implying other “natural” domain theoretic properties of  $\tilde{Q}$ ) which is actually used in the proof of the full abstraction property of this model. The class of finitary strategies is also shown to be effectively closed under application (on the base of a kind of normalizability property). Section 8 is devoted to a sketchy definition (by a very close analogy to the case of  $\mathbb{Q} \cong \tilde{Q}$  and **PCF**) of a fully abstract model  $\mathbb{W} \cong \tilde{W}$  for **PCF**<sup>+</sup> based on the concept of wittingly consistent non-deterministic strategies. Unlike the case of **PCF**, some details are given (but still with a reference to the old approach for **PCF** [28]) of a construction in **PCF**<sup>+</sup> of a universal (surjective) functional  $U_\alpha^+ : (\iota \rightarrow \iota) \rightarrow \alpha$  for each type  $\mathbb{W}_\alpha$ . It is also demonstrated in Section 8.2 that the model  $\mathbb{W}$  is not  $\omega$ -complete at level 2. Section 9 contains some concluding remarks and directions for further research. Finally, Appendix A presents an explicit construction of the typed version of a universal system of sequential strategies  $\langle Q, \mathcal{Q} \rangle$  from [30] which is used in previous sections for constructing  $\tilde{Q}$ .

## 2. DOMAINS AND TYPES—A GENERALISATION

**2.1. Basic Definitions.** Let us recall and generalize several well-known notions from domain theory (see, for example, [2, 26]), emphasizing some more subtle points related with their usage in this paper. Importantly, some of the known terms here have a meaning different from the traditional one. The goal is to find a version of domain theory most appropriate for the case of sequential (and other kinds of) functionals.

The term *poset* means a set  $D$  partially ordered by an *approximation* relation  $\sqsubseteq_D$ . Any poset  $D$  with the least (*bottom*, or *undefined*) element  $\perp$  will be called a *domain*. If  $A$  is any set, then  $A_\perp \doteq A \cup \{\perp\}$  is the corresponding *flat* domain where  $x \sqsubseteq y \Leftrightarrow (x = \perp) \vee (x = y)$ . A nonempty set  $X \subseteq D$  is called *directed* if, for any  $x, y \in X$ , we have  $x \sqsubseteq z$  and  $y \sqsubseteq z$  for some  $z \in X$ . The least upper bound (lub) of a set  $X$  is denoted by  $\bigsqcup X$ . If all directed sets have a lub in  $D$  then it is called a *directed complete* poset, or briefly, *dcpo*. However, the domains we will consider are typically not assumed to be dcpo. An element  $a$  of a domain (not necessarily a dcpo) is called *finite* (or *compact*) if  $a \sqsubseteq \bigsqcup X$  implies  $\exists x \in X. a \sqsubseteq x$  for

any directed set  $X$  for which  $\bigsqcup X$  exists. All elements of a flat domain are evidently finite. A domain  $D$  in which there are only countably many finite elements and each element  $x \in D$  is a directed lub of all its finite approximations is called  $\omega$ -algebraic. A monotonic mapping  $f$  between domains is called *continuous* if it preserves existing lubs  $\bigsqcup X$  of directed sets:  $f(\bigsqcup X) = \bigsqcup f(X)$  (that is, if  $\bigsqcup X$  exists then  $\bigsqcup f(X)$  is required to exist and satisfy this equality). Let  $(D \rightarrow E)$  or  $D \xrightarrow{\text{mon}} E$  denote the set of all monotonic mappings ordered pointwise:  $f \sqsubseteq g \iff \forall x \in D (fx \sqsubseteq gx)$ . For dcpos, let  $[D \rightarrow E]$  denote the set of continuous mappings also ordered pointwise. (We can suitably extend this denotation also for some special kinds of non-dcpo domains, called natural domains, by taking  $[D \rightarrow E]$  to be the set of all naturally continuous mappings; see Section 2.2.) If any two upper bounded elements  $c, d$  have least upper bound  $c \sqcup d$  in  $D$  then  $D$  is called *bounded complete*. A domain is called *finitely bounded complete* if, in the above, only finite  $c, d$ , and therefore  $c \sqcup d$ , are considered. If  $D$  is an algebraic dcpo then it is bounded complete if, and only if, it is finitely bounded complete. In fact, for dcpos bounded completeness is equivalent to existence of a lub for any bounded set, not necessarily finite. Algebraic and bounded complete dcpos are also known as *Scott domains* or as *complete  $f_0$ -spaces* of Ershov [10].

The above definitions are well-known and quite natural in the context of dcpos. We extended them to non-dcpo domains rather as a formal intermediate step before introducing in Section 2.2 so called “natural” versions of these notions. The general idea is that nonexistence of lubs of some directed sets is an indication that even existing lubs might be non-natural (existing “by a wrong reason”), and therefore the definitions of continuity, finite elements, etc. should be relativized to “natural” lubs only.

*Types* (or functional types) are defined as formal expressions built inductively from some *basic* types, in our case  $\iota$  and  $o$  (with the generic name Basic-type), by the arrow construct: if  $\alpha$  and  $\beta$  are types then  $(\alpha \rightarrow \beta)$  is a type. We usually write  $\alpha_1 \rightarrow \alpha_2 \rightarrow \dots \rightarrow \alpha_n \rightarrow \beta$  or  $\alpha_1, \alpha_2, \dots, \alpha_n \rightarrow \beta$  instead of  $(\alpha_1 \rightarrow (\alpha_2 \rightarrow (\dots \rightarrow (\alpha_n \rightarrow \beta) \dots)))$ . The *level* of any type  $\alpha = (\alpha_1, \dots, \alpha_n \rightarrow \text{Basic-type})$  is defined as

$$\text{level}(\alpha) = \max\{1 + \text{level}(\alpha_i) \mid 1 \leq i \leq n\}$$

and, in particular,  $\text{level}(\text{Basic-type}) = 0$ . The *arity* (or the number of arguments) of  $\alpha$  is the number  $n$  above.

For any type  $\alpha$  we define inductively, as usual, the corresponding (*standard*) domain  $\mathbb{D}_\alpha$  of all continuous functionals of type  $\alpha$  with  $\mathbb{D}_o = \mathbf{B}_\perp$ ,  $\mathbb{D}_\iota = \mathbf{N}_\perp$ , and  $\mathbb{D}_{\alpha \rightarrow \beta} = [\mathbb{D}_\alpha \rightarrow \mathbb{D}_\beta]$ , where  $\mathbf{B} = \{\mathbf{true}, \mathbf{false}\}$  and  $\mathbf{N} = \{0, 1, 2, \dots\}$ . All these  $\mathbb{D}_\alpha$  are  $\omega$ -algebraic, bounded complete dcpos. More general,

**Definition 2.1.** A (*typed monotonic order extensional applicative*) structure  $\{E_\alpha\}$  is a system of domains (with the least element  $\perp_\alpha$  in each) such that for any types  $\alpha$  and  $\beta$  there is a monotonic mapping  $\text{App}_{\alpha\beta} : E_{\alpha \rightarrow \beta} \times E_\alpha \rightarrow E_\beta$  (with  $\text{App}(f, x)$  abbreviated as  $fx$  and  $((\dots (fx_1)x_2) \dots x_n)$  abbreviated as  $fx_1 \dots x_n$ ) satisfying

- (i)  $\perp_{\alpha \rightarrow \beta} x = \perp_\beta$  for all  $x \in E_\alpha$ , and
- (ii) the extensionality condition: for all  $\alpha, \beta$  and  $f, f' \in E_{\alpha \rightarrow \beta}$ ,

$$f \sqsubseteq f' \iff \forall x \in E_\alpha (fx \sqsubseteq f'x).$$

Elements of  $E_\alpha$  are called *functionals* of type  $\alpha$ . An extensional structure  $\{E_\alpha\}$  is called a  $\lambda$ -*model* if it is sufficiently rich to contain all  $\lambda$ -definable functionals.

For the closure under  $\lambda$ -definability we can equivalently require that  $\{E_\alpha\}$  contains combinators  $\mathbf{S}_{\alpha\beta\gamma} : (\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$  and  $\mathbf{K}_{\alpha\beta} : \alpha \rightarrow (\beta \rightarrow \alpha)$  for all types  $\alpha, \beta, \gamma$  satisfying identities  $\mathbf{S}xyz = xz(yz)$  and  $\mathbf{K}uv = u$  for all  $x, y, z, u, v$  with  $\mathbf{S}, \mathbf{K}, x, y, z, u, v$  of appropriate types, omitted for brevity. We will also always assume that  $E_\iota = \mathbb{D}_\iota = \mathbf{N}_\perp$  and  $E_o = \mathbb{D}_o = \mathbf{B}_\perp$ . To simplify the exposition, let us take that  $\mathbf{B}_\perp \subseteq \mathbf{N}_\perp$  with  $\perp_o = \perp_\iota$ ,  $\mathbf{true} = 1$  and  $\mathbf{false} = 0$  and, hence, avoid using the Boolean type  $o$  at all in the “official” exposition. (However, we will use  $o$  in some examples for the convenience.) Then Basic-type will mean just  $\iota$ . Although in general the sets  $E_{\sigma \rightarrow \tau}$  and  $(E_\sigma \rightarrow E_\tau)$  may even not intersect, there is the natural embedding  $E_{\sigma \rightarrow \tau} \hookrightarrow (E_\sigma \rightarrow E_\tau)$  induced by the application operation. Moreover, without restricting generality we may also consider that the set

$$E_\alpha = E_{(\alpha_1, \dots, \alpha_n \rightarrow \iota)} \subseteq (E_{\alpha_1} \times \dots \times E_{\alpha_n} \rightarrow E_\iota) \quad (2.1)$$

consists of some monotonic mappings of the type shown, ordered pointwise,

$$f \sqsubseteq f' \iff \forall \bar{x} (f\bar{x} \sqsubseteq_\iota f'\bar{x}),$$

and for all  $f \in E_\alpha$  and  $x_1 \in E_{\alpha_1}$

$$f x_1 = \lambda x_2, \dots, x_n. f(x_1, x_2, \dots, x_n) \in E_{\alpha_2} \times \dots \times E_{\alpha_n} \rightarrow E_\iota \quad (2.2)$$

is the “residual” map. Indeed, any  $\{E_\alpha\}$  satisfying (2.1) and (2.2) and containing constant undefined functions  $\perp_{\bar{\alpha} \rightarrow \iota} = \lambda \bar{x}. \bar{x}. \perp_\iota$  is a monotonic, order extensional applicative structure. It is clear that such an  $\{E_\alpha\}$  is a restricted class of monotonic finite-type functionals.

**Definition 2.2.** A structure  $\{E_\alpha\}$  (with  $E_\alpha$  not necessarily a dcpo) is called *continuous* if for each type  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_k \rightarrow \iota)$  and variables  $f : \alpha$  and  $\bar{x} : \bar{\alpha}$ , the *full application map*  $\lambda f \bar{x}. f \bar{x} : E_\alpha \times E_{\alpha_1} \times E_{\alpha_2} \times \dots \times E_{\alpha_k} \rightarrow E_\iota$  is *continuous*. Equivalently, we can require the continuity of the application maps of two arguments  $\lambda f x_1. f x_1 : E_\alpha \times E_{\alpha_1} \rightarrow E_{\alpha_2, \dots, \alpha_k \rightarrow \iota}$ .

**2.2. Natural Non-dcpo Domains.** More generally,

**Definition 2.3.** In any monotonic, order extensional applicative structure a *pointwise lub*  $\biguplus_i f_i$  of an arbitrary (not necessarily directed) family of functionals (of the same type) is the ordinary lub  $\bigsqcup_i f_i$ , in the case of the basic type, and, for higher types, it is the ordinary lub which is also required to satisfy, inductively, the pointwise identity  $(\biguplus_i f_i)x = \biguplus_i (f_i x)$  (with  $\biguplus_i (f_i x)$  also pointwise) for all  $x$  of appropriate type.

Thus,  $f = \biguplus_i f_i$  implies  $f = \bigsqcup_i f_i$ , but, in general, not vice versa. That is,  $\biguplus$  is a restricted version of  $\bigsqcup$ . (See an example below.) Equivalently, we may require from  $\bigsqcup_i f_i$  the identity  $(\bigsqcup_i f_i)\bar{x} = \bigsqcup_i (f_i \bar{x})$  in the basic type. In fact,

$$f = \biguplus_i f_i \text{ iff } f\bar{x} = \bigsqcup_i (f_i \bar{x}) \text{ for all } \bar{x}, \quad (2.3)$$

assuming  $f\bar{x}$  is of the basic type. The concept of pointwise lub is quite natural and could also be called just *union*, or *natural lub*. This is even the ordinary set theoretic union if to identify monotonic functionals of the type  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_k \rightarrow \iota)$  with corresponding graph subsets of  $E_{\alpha_1} \times E_{\alpha_2} \times \dots \times E_{\alpha_k} \times \mathbf{N}$ . In this case  $\sqsubseteq$  also coincides with the set theoretic notion of inclusion  $\subseteq$ . Respectively, non-pointwise lubs are considered as non-natural in this sense. (However note that neither finite nor also “naturally” finite functionals considered below are necessarily represented as finite graph sets in the above sense.)

**Example 2.4.** To illustrate the above definition, consider a simple example in  $\{\mathbb{Q}_\alpha\}$  (the monotonic, order extensional  $\lambda$ -model of sequential functionals to be defined later) of a finite non-natural lub of two elements. Define two first order sequential functions  $O_i(x_1, x_2)$ ,  $i = 1, 2$ , as 0 if the corresponding  $x_i = 0$ , and  $\perp$  otherwise. Then  $O_1 \sqcup O_2 = \lambda x_1, x_2. 0$  is the constant zero function in  $\mathbb{Q}$ , and this is not a natural lub. The natural lub, if it would exist in  $\mathbb{Q}$ , should satisfy  $(O_1 \natural O_2)(x_1, x_2) = 0$  if  $x_1 = 0$  or  $x_2 = 0$ , and  $= \perp$  otherwise. But this is not a sequential function, that is, it lies outside of  $\mathbb{Q}$ .

**Definition 2.5.** A structure  $\{E_\alpha\}$  is called *naturally continuous* if for all types  $\alpha = \tau \rightarrow \sigma$  and  $f \in E_\alpha$  the map  $\lambda x. fx : E_\sigma \rightarrow E_\tau$  preserves directed natural lubs of the arguments whenever they exist:  $f(\natural_i x_i) = \natural_i fx_i$ . That is, if the directed natural lub to the left exists then the natural lub to the right exists too, and the equality holds.

We can require, equivalently, for each type  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_k \rightarrow \iota)$  and  $f \in E_\alpha$ , that the map  $\lambda \bar{x} : \bar{\alpha}. f \bar{x} : E_{\alpha_1} \times E_{\alpha_2} \times \dots \times E_{\alpha_k} \rightarrow E_\iota$  is naturally continuous (preserves natural lubs) in each argument. Evidently, natural continuity of  $fx$  or  $f\bar{x}$  in  $f$  is automatically satisfied by the definition of natural lub as the pointwise one. Also, *in a continuous structure* (that is, with continuous full application maps) *all existing directed lubs are natural (pointwise), and therefore any continuous structure is naturally continuous*. Further,

**Definition 2.6.** *Naturally finite* functionals are defined like the ordinary finite ones, but by using the natural lubs.

Each finite functional is also naturally finite (but probably not vice versa; see the discussion below).

**Definition 2.7.**

- (a) A structure  $\{E_\alpha\}$  is called *naturally  $\omega$ -algebraic* if each of its elements is a directed natural lub of naturally finite elements, and there are only countably many naturally finite elements in the structure.
- (b) It is called *naturally bounded complete* if any two upper bounded naturally finite elements have a lub (not necessarily a natural lub, but evidently also naturally finite element).

For any naturally algebraic and naturally bounded complete structure  $\{E_\alpha\}$  the sets of the form  $\check{a} \rightleftharpoons \{x \in E_\alpha \mid a \sqsubseteq x\}$ , for  $a$  naturally finite, constitute a base of a  $(T_0)$ -topology in each  $E_\alpha$  which makes  $E_\alpha$  satisfying this definition a (non-necessarily complete)  $f_0$ -space of Ershov [10]. Note that open sets in this topology are exactly those *naturally Scott open* (defined as usual, but with respect to the natural directed lubs).

By using Lemma 2.11 presented below, we will prove in Theorem 7.13 the natural continuity and the last two properties (a) and (b) defined above for the special case of the model of sequential functionals  $\{\mathbb{Q}_\alpha\}$ . That  $\mathbb{Q}$  is not a dcpo was actually shown by Normann [23].

**Hypotheses 2.8.** *It seems quite plausible that in  $\{\mathbb{Q}_\alpha\}$  there exist*

- (1) *a directed non-natural lub,*
- (2) *a naturally finite, but not a finite functional (being a proper directed lub),*
- (3) *a non-continuous (but naturally continuous) functional, and*
- (4) *a naturally finite (and naturally continuous), but not a continuous functional.*

*We could also expect that*



- (5) *a continuous (and therefore naturally continuous) lambda model exists whose higher type domains are not dcpos.*

**Note 2.9.**

- (a) We see that these hypotheses reveal a terminological problem (“naturally finite, but not finite”, etc.). Properly speaking, these are naturally finite functionals which are most naturally considered as full-fledged finite objects in the framework of  $\mathbb{Q}$ . Moreover, together with naturally continuous functionals, these concepts give rise to an appropriate non-dcpo generalisation of continuous,  $\omega$ -algebraic and bounded complete  $\lambda$ -models (originally considered over dcpos). This will be seen from the following considerations and Lemma 2.11. The more traditional definitions of continuous and finite functionals in terms of the ordinary directed lubs prove to be not very adequate in the framework of non-dcpo.
- (b) Another important point is that, being based on types, the natural lub  $\uplus$  as well as other related “natural” concepts are not purely order-theoretic ones. However, one can give an abstract definition of *natural (non-dcpo) domains* with a primitive partially defined operator  $\uplus : 2^D \dashrightarrow D$  in each domain which is a restricted version of  $\sqcup$  and has appropriate postulated properties. Then the special case of these natural domains satisfying the conditions (a) and (b) of Definition 2.7 corresponds exactly to the  $f_0$ -spaces of Ershov [10]. More detailed and general discussion on this generalized theory of non-dcpo domains and the mentioned correspondence requires a separate consideration to be presented elsewhere. It is also worth noticing that these domains appear in our presentation as natural non-dcpo domains rather than  $f_0$ -spaces. They prove to be  $f_0$ -spaces only a posteriori by using quite involved technical theory of computational strategies and applying Lemma 2.11 below.

### 2.3. Finitely Restricted Functionals.

**Conditions on  $\{E_\alpha\}$ .** For the rest of Section 2 let  $\{E_\alpha\}$  be any monotonic, order extensional  $\lambda$ -model which contains the first order equality predicate  $x = y$  (monotonic and strict in  $x$  and  $y$ ) and the ordinary (monotonic and sequential) conditional function **if  $x$  then  $y$  else  $z$**  for the basic type (and hence for all types by  $\lambda$ -definability).

Recall that a monotonic function  $\Psi : E \rightarrow E$  is called a *projection* if  $\Psi x \sqsubseteq x$  for all  $x \in E$ , and  $\Psi \circ \Psi = \Psi$ . We say also that  $\Psi$  is a projection from  $E$  onto its range  $\subseteq E$  which is also the set of all fixed points of  $\Psi$ . For any two projections,  $\Psi \sqsubseteq \Psi'$  iff  $\text{range}(\Psi) \subseteq \text{range}(\Psi')$ . Note that  $\Psi x$  is the largest  $\sqsubseteq$ -approximation to  $x$  from the range of  $\Psi$ .

Now, we will follow Milner [21], slightly simplifying and generalizing to the “natural” non-dcpo case.

**Definition 2.10.** Define projections  $\Psi_\alpha^{[k]} : E_\alpha \rightarrow E_\alpha$  for all types and any  $k \geq 0$  by letting

$$\begin{aligned} \Psi_l^{[k]} : E_l &\xrightarrow{\text{onto}} E_l^{[k]}, \quad E_l^{[k]} \equiv \{\perp, 0, 1, \dots, k\}, \\ \Psi_{\sigma \rightarrow \tau}^{[k]} f &\equiv \Psi_\tau^{[k]} \circ f \circ \Psi_\sigma^{[k]}, \\ E_\alpha^{[k]} &\equiv \text{Range}(\Psi_\alpha^{[k]}). \end{aligned}$$

Denote  $x^{[k]} \equiv \Psi_\alpha^{[k]} x$ . Elements  $x^{[k]}$  in  $E_\alpha^{[\omega]} \equiv \bigcup_k E_\alpha^{[k]} \subseteq E_\alpha$  are called *finitely restricted*.

These all are monotonic sequences on  $k$ . That  $\Psi_i^{[k]}$  and hence all other  $\Psi_{\sigma \rightarrow \tau}^{[k]}$  (representable by) elements of the  $\lambda$ -model (we write  $\Psi_\alpha^{[k]} \in E_{\alpha \rightarrow \alpha}$ ) follows from existence in it of both = and **if**. By induction on types, each  $E_\alpha^{[k]}$  is a finite set since  $x^{[k]}y = (xy^{[k]})^{[k]}$  at all types. Also, the application of  $k$ -restricted functionals to any argument is  $k$ -restricted. In particular, each finitely restricted functional  $\varphi$  has a tabular representation

$$\varphi x = \underbrace{\begin{bmatrix} b_0, \dots, b_{n-1} \\ a_0, \dots, a_{n-1} \end{bmatrix}}_{\varphi} x = \bigsqcup_{a_i \sqsubseteq x} b_i \text{ where } \varphi a_i = b_i \text{ and } a_i, b_i \text{ are finitely restricted.} \quad (2.4)$$

In each model  $\{E_\alpha\}$  (over  $E_\iota = \mathbf{N}_\perp$ ) satisfying the above conditions there are only countably many finitely restricted elements. This is another approach to the finiteness of higher type functionals. Without assuming any further conditions on  $\{E_\alpha\}$ , each  $\Psi_\alpha^{[k]}$  considered as a map  $\Psi_\alpha^{[k]} : E_\alpha \rightarrow E_\alpha$  is naturally continuous and, moreover, preserves all existing natural lubs (not necessarily directed). This follows by induction on the types:

$$\begin{aligned} (\Psi_\tau^{[k]} \circ (\bigsqcup_i f_i) \circ \Psi_\sigma^{[k]})x &= \Psi_\tau^{[k]}((\bigsqcup_i f_i)(\Psi_\sigma^{[k]}x)) = \Psi_\tau^{[k]}(\bigsqcup_i (f_i(\Psi_\sigma^{[k]}x))) = \\ & \bigsqcup_i \Psi_\tau^{[k]}(f_i(\Psi_\sigma^{[k]}x)) = \bigsqcup_i ((\Psi_\tau^{[k]} \circ f_i \circ \Psi_\sigma^{[k]})x) = (\bigsqcup_i (\Psi_\tau^{[k]} \circ f_i \circ \Psi_\sigma^{[k]}))x. \end{aligned}$$

It also follows that each finitely restricted element  $x^{[k]}$  is naturally finite:  $x^{[k]} \sqsubseteq \bigsqcup Z$  for a directed set  $Z$  implies  $x^{[k]} \sqsubseteq \bigsqcup \{z^{[k]} \mid z \in Z\} = z^{[k]} \sqsubseteq z$  for some  $z$  by natural continuity of  $\Psi^{[k]}$  and because  $E_\alpha^{[k]}$  is finite.

Moreover, if the model is naturally continuous then  $x = \bigsqcup_k x^{[k]}$  holds for all  $x$ . Indeed, assuming by induction on types that  $\bar{y} = \bigsqcup_k \bar{y}^{[k]}$ , we have  $x\bar{y} = \bigsqcup_k (x\bar{y}^{[k]}) = \bigsqcup_k (x\bar{y}^{[k]})^{[k]} = \bigsqcup_k (x^{[k]}\bar{y})$ . Thus  $\bar{x} = \bigsqcup_k \bar{x}^{[k]}$  by (2.3), as required.

Finally we note that, without any further assumptions on the model, any two upper bounded finitely restricted elements  $d, e$  have a (not necessarily natural) lub  $d \sqcup e$  which is also finitely restricted. Indeed, it can be obtained as the greatest lower bound  $\sqcap \{x^{[k]} \mid x \sqsupseteq d, e\}$  for any fixed  $k$  such that  $d, e \in E_\alpha^{[k]}$  because the glb of any finite nonempty set is definable from **if** and =.

The following Lemma is a generalisation of the Algebraicity Lemma of Milner in [21] to the case of non-dcpo's and to the ‘‘natural’’ case, but formulated for simplicity only for the models with the numerical basic values  $E_\iota = \mathbf{N}_\perp$ . It clearly demonstrates that the generalisations introduced are quite adequate and natural.

**Lemma 2.11.** *Let  $\{E_\alpha\}$  be any monotonic, order extensional  $\lambda$ -model, with  $E_\iota = \mathbf{N}_\perp$ , which contains first order equality and the conditional. Then*

- (a) *this model is naturally continuous if, and only if,*
  - (\*) *for any type  $\alpha = \alpha_1, \dots, \alpha_n \rightarrow \iota$  and elements  $f \in E_\alpha$  and  $\bar{x} \in E_{\bar{\alpha}}$ ,  $f\bar{x} = f\bar{d}$  holds for some finitely restricted  $\bar{d} \sqsubseteq \bar{x}$ ;*
- (b) *if the model is naturally continuous then (i) the naturally finite elements of each  $E_\alpha$  are exactly the finitely restricted ones, (ii)  $\{E_\alpha\}$  is naturally  $\omega$ -algebraic, and (iii) it is naturally bounded complete;*
- (c) *repeats (b), but with ‘‘naturally’’ omitted.*

*Proof.*

- (b) follows easily from the above considerations on projections  $\Psi^{[k]}$ .

- (c) It suffices to recall that continuous structures are also naturally continuous, and the concepts of directed lubs, and hence of finite functionals in these models, are equivalent to their “natural” versions. Note that we do not assume here that the  $E_\alpha$  are dcpos.
- (a) “If” follows from natural finiteness of all  $x^{[k]}$ . “Only if” follows from (b).  $\square$

The clause (a) of this Lemma (not considered in [21]) is used in Section 7.2.1 below to show that the model of sequential functionals  $\{\mathbb{Q}_\alpha\}$  is naturally continuous and satisfies the conditions (i)–(iii) from (b). In the application of this Lemma to  $\{\mathbb{Q}_\alpha\}$  the crucial point is that (\*) in (a) implies all the essential domain theoretic properties holding for this model.

Moreover, we will also show in Theorem 6.6 (b) that the model  $\{\mathbb{Q}_\alpha\}$  is also *sequentially complete* in the sense that it is closed under taking natural (pointwise) lubs of a special class of increasing sequences (determined by sequential strategies). For example, in  $\{\mathbb{Q}_\alpha\}$  we have the natural lub  $\biguplus_n f^n \perp$  giving the least fixed point of  $f : \alpha \rightarrow \alpha$  for all types  $\alpha$ .

**2.4. On Efficiency of Naturally Finite Functionals.** For the case of the standard continuous model  $\{\mathbb{D}_\alpha\}$ , the tabular representation (2.4) of naturally finite (finitely restricted) functionals proves to be quite effective and gives rise to an effective numbering of these functionals [10]. The main reason for that is that (by induction on types) any monotonic table as in (2.4) represents a finitely restricted functional in this model. This also holds for  $\{\mathbb{W}_\alpha\}$  (the non-dcpo fully abstract model for **PCF**<sup>+</sup>) where naturally finite functionals are the same as in  $\{\mathbb{D}_\alpha\}$ . The latter essentially follows from their definability in **PCF**<sup>+</sup> [25]. In fact, the predicates “ $\varphi \sqsubseteq \psi$ ”, “ $\varphi, \psi$  are upper bounded (consistent)” and the application operation “ $\varphi a$ ”, for naturally finite  $\varphi, \psi, a$ , are effectively computable in the cases of  $\{\mathbb{D}_\alpha\}$  and  $\{\mathbb{W}_\alpha\}$ .

Unfortunately, in the model of hereditarily-sequential functionals  $\{\mathbb{Q}_\alpha\}$  no such effective numbering is possible as can be shown by appropriate adaptation of the undecidability result of Loader [16]. In fact, we cannot generally, and effectively, decide which monotonic tables (2.4) represent sequential functionals in  $\mathbb{Q}_\alpha$ , let even for finitely many of  $k$ -restricted ones. But we can enumerate them by means of the *finitary strategies* introduced in Section 7.2.1 instead of using non-effective (in this case) tabular representation. In this sense the set of  $k$ -restricted functionals of a fixed type  $\alpha$  is finite and recursively enumerable but, in general, “undecidable”. However, it will be demonstrated in Theorem 7.16 that, under the above mentioned “finitary” representation of naturally finite functionals  $\varphi, \psi, a$ , the application “ $\varphi a$ ” is computable, and it easily follows that “ $\varphi \not\sqsubseteq \psi$ ”, unlike “ $\varphi \sqsubseteq \psi$ ”, is semidecidable in  $\{\mathbb{Q}_\alpha\}$  (and similarly for  $\{\mathbb{W}_\alpha\}$  in addition to the above tabular effective in this case and decidable representation).

This seemingly diminishes the role of naturally finite (= finitely restricted) functionals and their use (like in  $\{\mathbb{D}_\alpha\}$ ) to define effective functionals as those which are (natural) lubs of a recursively enumerable directed set of (naturally) finite approximations. Such a definition seems not very appropriate, not only for  $\{\mathbb{Q}_\alpha\}$ , but even for the case of  $\{\mathbb{W}_\alpha\}$ . At least, further research is required. For efficiency of functionals we should, in these cases, rather use the concept of an effective (sequential and, respectively, wittingly consistent) computational strategy *e.g.* as in Definition 4.5.

Finally, let us mention one more related question on  $\mathbb{Q}$  and  $\mathbb{W}$ : for naturally finite  $\varphi$  and any  $x$  the application  $\varphi x$  is evidently naturally finite, but *is its finitary representation computable from that of  $\varphi$  and a strategy representing  $x$  in general?* (However, for  $\varphi x : \iota$  it is computable.)

**2.5. Ideal Completion and Uniqueness of Fully Abstract Models.** Although our goal is the fully abstract non-dcpo (in fact, naturally continuous) models for **PCF** and **PCF**<sup>+</sup>, it make sense to relate them with the continuous dcpo model construction of Milner [21] via the ideal completion procedure.

Now, let  $\mathcal{E} = \{E_\alpha\}$  be any naturally continuous  $\lambda$ -model satisfying the assumption and the conclusions (i)–(iii) of Lemma 2.11. Consider its *ideal completion*  $\dot{\mathcal{E}} = \{\dot{E}_\alpha\}$  which is a continuous dcpo model defined as follows. A nonempty directed set  $\dot{x} \subseteq E_\alpha$  of naturally finite elements is called an *ideal* if  $a \sqsubseteq b \in \dot{x} \Rightarrow a \in \dot{x}$  for  $a, b$  naturally finite. Let  $\dot{E}_\alpha$  be the set of all ideals in  $E_\alpha$ . This is evidently a dcpo ordered by set inclusion  $\subseteq$  with  $\{\perp\}$  the least ideal and with directed lubs coinciding with set unions  $\bigcup_i \dot{x}_i$ . Let  $I(X) = \{a \mid \exists x \in X (a \sqsubseteq x \ \& \ a \text{ naturally finite})\}$  be the ideal generated by a directed set  $X$ , and  $I(x) = I(\{x\})$ . As  $I(x) \subseteq I(y) \iff x \sqsubseteq y$ , we have an order isomorphic embedding of posets  $I : E_\alpha \hookrightarrow \dot{E}_\alpha$  which is onto for the basic type. Note that always  $\dot{E}_{\iota \rightarrow \iota} \cong [\mathbf{N}_\perp \rightarrow \mathbf{N}_\perp]$ . If  $a$  is naturally finite in  $E_\alpha$  then  $I(a)$  is finite element in the dcpo  $\dot{E}_\alpha$ . For any  $\dot{x} \in \dot{E}_\alpha$ ,  $I(a) \subseteq \dot{x} \iff a \in \dot{x}$ . In fact,  $\dot{x}$  is a directed union of such  $I(a)$ , and  $\dot{E}_\alpha$  is an  $\omega$ -algebraic dcpo domain with finite elements  $I(a)$  for  $a$  naturally finite. It is also bounded complete because  $E_\alpha$  is naturally bounded complete. Further, we may define the application operation in  $\dot{\mathcal{E}}$  by  $f\dot{x} = I(\{\varphi a \mid \varphi \in \dot{f}, a \in \dot{x}\})$  for any  $\dot{f}$  and  $\dot{x}$  of appropriate types, which makes it a monotonic order extensional structure. For the latter use the fact that

$$\forall a \text{ naturally finite } \exists \psi \in \dot{g}(\varphi a \sqsubseteq \psi a) \implies \exists \psi \in \dot{g}(\varphi \sqsubseteq \psi).$$

It is easy to show that  $\dot{\mathcal{E}}$  is continuous, that is having the continuous application operation  $((\bigcup_i \dot{f}_i)(\bigcup_j \dot{x}_j) = \bigcup_{ij} \dot{f}_i \dot{x}_j$  holds for directed families). The application also agrees with the embedding  $I : \mathcal{E} \hookrightarrow \dot{\mathcal{E}}$ :

$$I(fx) = (I(f))(I(x)).$$

Moreover,  $\dot{\mathcal{E}}$  is a  $\lambda$ -model because for the combinators  $\mathbf{S}, \mathbf{K} \in \mathcal{E}$  we have

$$I(\mathbf{S})\dot{x}\dot{y}\dot{z} = \dot{x}\dot{z}(\dot{y}\dot{z}), \quad \text{and} \quad I(\mathbf{K})\dot{x}\dot{y} = \dot{x}$$

in  $\dot{\mathcal{E}}$  (where all directed lubs are natural/pointwise and “naturally finite” = “finite”). Assuming additionally the existence of a fixed point combinator in  $\mathcal{E}$  satisfying the **Y**-property

$$\mathbf{Y}f = f(\mathbf{Y}f) = \bigsqcup_n f^n(\perp) \tag{2.5}$$

for all  $f$  of appropriate type, its image in  $\dot{\mathcal{E}}$  behaves accordingly:

$$I(\mathbf{Y})\dot{f} = \bigsqcup_n \dot{f}^n(\{\perp\}).$$

The languages **PCF** and **PCF**<sup>+</sup> [25, 18] considered in this paper are based on  $\mathbf{S}, \mathbf{K}, \mathbf{Y}, 0$ , plus constants for some level one functions (successor, predecessor, first order equality and one of two versions of the conditional—sequential and parallel, respectively). For any  $\mathcal{E}$  satisfying the **Y**-property the meaning of all these constants is also not changed by the embedding  $I : \mathcal{E} \hookrightarrow \dot{\mathcal{E}}$ . Hence,

**Proposition 2.12.** *The meaning of **PCF**<sup>(+)</sup> terms in  $\dot{\mathcal{E}}$  agrees with that in  $\mathcal{E}$ .  $\square$*

**Definition 2.13.** Let  $\mathcal{C}[\ ]$  denote an arbitrary ground type program context in  $\mathbf{PCF}^{(+)}$ . A model  $\mathcal{E}$  satisfying the  $\mathbf{Y}$ -property (2.5) is called *fully abstract* relative to  $\mathbf{PCF}^{(+)}$  if

$$\forall \mathcal{C} (\llbracket \mathcal{C}[M] \rrbracket \sqsubseteq \llbracket \mathcal{C}[N] \rrbracket) \implies \llbracket M \rrbracket \sqsubseteq \llbracket N \rrbracket.$$

Evidently,  $\mathcal{E}$  is fully abstract iff  $\dot{\mathcal{E}}$  is such (relative to  $\mathbf{PCF}^{(+)}$  or, equivalently, relative to  $\mathbf{PCF}^{(+)}$  minus  $\mathbf{Y}$ ; use the Fixed-point Lemma in [21] for the dcpo case of  $\dot{\mathcal{E}}$ ).

**Proposition 2.14.** *Let  $\mathcal{E}$  be any fully abstract and naturally continuous  $\lambda$ -model of  $\mathbf{PCF}^{(+)}$  satisfying the  $\mathbf{Y}$ -property (2.5). Then the model  $\dot{\mathcal{E}}$  is also fully abstract and all finite elements in  $\dot{\mathcal{E}}$ , and therefore all naturally finite elements in  $\mathcal{E}$ , are definable in  $\mathbf{PCF}^{(+)}$  without using  $\mathbf{Y}$ . The same holds for  $\mathcal{E}$  fully abstract relative to the language  $\mathbf{PCF}^{(+)}$  minus  $\mathbf{Y}$  (although still satisfying  $\mathbf{Y}$ -property).*

*Proof.* The definability statement for the case of fully abstract continuous dcpo models (here  $\dot{\mathcal{E}}$ ) was actually shown in the proof of Theorem 3 in [21]. This implies the case of naturally continuous model  $\mathcal{E}$  by using Proposition 2.12.  $\square$

It follows as in [21], by taking  $\mathcal{C}[\ ] = [\ ]C_1 \cdots C_n : \iota$  with  $C_i$  defining finite elements, that on definable elements, and therefore on all finite elements such fully abstract  $\dot{\mathcal{E}}$ , if exists at all, is determined uniquely, up to isomorphism. A general construction of such a continuous dcpo model from some given level one functions is presented in [21].

Alternatively and extending to the case of non-dcpo, we will define two models  $\mathbb{Q}$  and  $\mathbb{W}$  for  $\mathbf{PCF}$  and  $\mathbf{PCF}^+$ , respectively, such that it will follow from Theorems 6.6 (b), 7.1, 7.2 and 7.13 below (on a generalization of the  $\mathbf{Y}$ -property, full abstraction property, universality and natural continuity of  $\mathbb{Q}$ , and corresponding versions for  $\mathbb{W}$ ) that

**Theorem 2.15.**

- (a)  $\dot{\mathbb{Q}}$  and  $\dot{\mathbb{W}}$ , are the only possible fully abstract continuous dcpo models for  $\mathbf{PCF}$  and  $\mathbf{PCF}^+$ , respectively (with  $\dot{\mathbb{Q}}$  also isomorphic to Milner's model in [21] and  $\dot{\mathbb{W}}$  isomorphic to  $\mathbb{D}$ ).
- (b) Therefore also  $\mathbb{Q}$  and  $\mathbb{W}$  are the only possible fully abstract naturally continuous<sup>1</sup> models for  $\mathbf{PCF}$  and  $\mathbf{PCF}^+$ , respectively, satisfying the  $\mathbf{Y}$ -property and in which all elements are definable from arbitrary type  $\iota \rightarrow \iota$  functions of the model where  $\mathbb{Q}_{\iota \rightarrow \iota} = \mathbb{W}_{\iota \rightarrow \iota} = \mathbb{D}_{\iota \rightarrow \iota} = [\mathbf{N}_{\perp} \rightarrow \mathbf{N}_{\perp}]$ —all monotonic functions.  $\square$

More general, in the latter uniqueness formulation we could consider for  $\mathbb{Q}_{\iota \rightarrow \iota}$  and  $\mathbb{W}_{\iota \rightarrow \iota}$  some other classes of type  $\iota \rightarrow \iota$  monotonic functions, say, all computable—as the minimal such a class. In the computable case only definability in pure  $\mathbf{PCF}^{(+)}$  may be used, without reference to type  $\iota \rightarrow \iota$  functions in (b).

---

<sup>1</sup>Note that the natural continuity requirement on  $\mathcal{E}$  here can be omitted and the proof of (b) can be done straightforwardly by showing first that (i) the denotational semantics of  $\mathbf{PCF}^{(+)}$  terms (possibly involving arbitrary type  $\iota \rightarrow \iota$  functions) of the type  $\iota$  corresponds exactly to the natural operational semantics, and (ii) Milner's Context Lemma [21] for the operational semantics holds. To this end, define a logical relation  $a R A$  between values  $a$  in  $\mathcal{E}$  and closed  $\mathbf{PCF}^{(+)}$  terms  $A$  by letting, for the type  $\iota$ ,  $a R_{\iota} A \iff a \sqsubseteq_{\iota}$  the value to which  $A$  operationally reduces, and show  $\llbracket A \rrbracket R_{\alpha} A$  for closed terms. (Thanks to Achim Jung who has drawn attention of the author to this proof of (i) and (ii).) The point is that only the  $\mathbf{Y}$ -property is used in the proof, and neither dcpo nor continuity properties of models considered are needed. We omit the details. Then the full abstraction property can be formulated in terms of operational semantics and thus leads to an operational characterisation of the relation  $\llbracket A \rrbracket \sqsubseteq \llbracket B \rrbracket$  for  $\mathbf{PCF}^{(+)}$  terms.

### 3. SEQUENTIAL STRATEGIES

#### 3.1. Definition, Informal Meaning and Examples.

3.1.1. *Preliminary Definitions and Conventions.* Let  $M$  be any set of abstract elements denoted as  $m, m', m_1, m_2$ , etc., each having a specified type (e.g.,  $m : \alpha$ ). That is, actually,  $M$  is a disjoint union of sets  $M_\alpha$  consisting of elements of the type  $\alpha$ . An additional structure on  $M$  considered below will allow us to call these elements (computational) *strategies* (over  $M$ ).

For each type  $\alpha$ , let us also fix an infinite list of *variables* of this type  $x_1^\alpha, x_2^\alpha, \dots$ . We will use,  $x, y, x'$ , etc. as meta-variables. However,  $x_i$  or  $\bar{x} = x_1, \dots, x_n$  will usually refer to the numbering in the above lists, assuming some typing. That is,  $x_i$  is  $i$ -th variable of a type which can be recovered from the context. Thus, given any types  $\alpha_1, \dots, \alpha_n$ , we have the corresponding *canonical* list of variables  $x_1^{\alpha_1}, x_2^{\alpha_2}, \dots, x_n^{\alpha_n}$  (first variable of the type  $\alpha_1$ , second variable of the type  $\alpha_2$ , etc.) or just  $x_1, \dots, x_n$  or  $\bar{x}$ , for brevity. Well-typed *applicative terms* over  $M$  constitute the least set containing *atomic terms* (i.e., variables  $x : \alpha$  and constants  $m : \alpha$ ), and closed under application: if  $A : \alpha \rightarrow \beta$  and  $B : \alpha$  then  $AB : \beta$ . Let  $\text{Basic-Terms}(M)$  be the set of all well-typed applicative terms of the Basic-type (actually,  $\iota$ ) built up from (typed) strategies of  $M$  and (typed) variables. If  $m : \alpha = (\alpha_1, \dots, \alpha_n \rightarrow \beta)$  then  $mx_1 \cdots x_n$  or  $m\bar{x}$  will denote the applicative term  $(\cdots((mx_1)x_2) \cdots x_n)$  of the type  $\beta$  with  $x_i : \alpha_i$  (the  $i$ -th variable of the type  $\alpha_i$ ). These notational agreements allow us to avoid type superscripts and related assumptions which, otherwise, would obscure the exposition. Strictly speaking, all variables, elements of  $M$  and terms are typed.

Additionally, let us agree that, depending on the context, we can identify any variable  $x : \alpha$  with some value in the corresponding set of values  $E_\alpha$ . This is in the same line as the tradition of using variables in ordinary mathematical texts. Again, this way we avoid extra complications in notation, relying on the context. Let us also assume that, by default,  $v, v', v_1, v_2, \dots$  range over  $\mathbf{N}$  whereas  $u, w$  range over  $\mathbf{N}^*$ . Say,  $v_1 v_2 \cdots v_k \in \mathbf{N}^*$  denotes the string of the length  $k$ , whereas  $uw \in \mathbf{N}^*$  is the concatenation of any two strings  $u, w \in \mathbf{N}^*$ , and  $uv \in \mathbf{N}^*$  is the concatenation of any string  $u$  with a one element string  $v$ , etc. We will use similar conventions for the case of  $S^*$  for any other set  $S$ .

#### 3.1.2. Main Definition.

**Definition 3.1.** A *system of sequential computational strategies*<sup>2</sup> is a pair  $\langle M, \mathcal{M} \rangle$  consisting of the set  $M$  of typed elements (strategies) and a *partial* function

$$\mathcal{M} : M \times \mathbf{N}^* \rightarrow \text{Basic-Terms}(M) \cup \mathbf{N},$$

satisfying the following condition:

**if**  $m : \alpha = (\alpha_1, \dots, \alpha_n \rightarrow \iota)$  is a strategy,  $x_1, \dots, x_n$  is the canonical list of variables of the types  $\alpha_1, \dots, \alpha_n$ , respectively, so that  $mx_1 \cdots x_n : \iota$ , and  $\mathcal{M}(m, w)$  is defined **then** either

---

<sup>2</sup>We will also consider, in Section 8.1, the more general concept of non-deterministic (non-sequential), wittingly consistent strategies. However, we will typically use the simple term “strategy” relying on the context.

- (1)  $\mathcal{M}(m, w) = A\{x_1, \dots, x_n\} \in \text{Basic-Terms}(M)$  (written also as the query  $\mathcal{M}(m, w) = "A\{\bar{x}\} = ?"$ ) with all variables in  $A$  contained in the list  $x_1, \dots, x_n$ , or
- (2)  $\mathcal{M}(m, w) \in \mathbf{N}$  is a (defined) basic value.

We also write  $\mathcal{M}(m, w) = \perp$  if  $\mathcal{M}(m, w)$  is undefined.

**Informal comments.** Any applicative term of the form

$$m\bar{x} = mx_1 \cdots x_n = (\cdots (mx_1) \cdots x_n)$$

is considered as the *query* or *task* “ $m\bar{x} = ?$ ” of finding its (basic) value by means of the strategy  $m$  with the help of an *Oracle* as follows:

- by asking, in the case 1 above, *queries* of the form “ $A\{\bar{x}\} = ?$ ” (concerning  $\bar{x}$ ) addressed to the Oracle, assuming that a finite sequence of answers  $w \in \mathbf{N}^*$  to previous queries (called also a *prompt* or *computation history* for the strategy  $m$ ) have been received from the Oracle, and
- by giving, in the case 2, a *resulting value (solution)* for the initial task “ $m\bar{x} = ?$ ”, based on the previous computation history  $w$ .

In particular, it is possible that  $\mathcal{M}(m, \Lambda) = v$  is a Basic-type value in  $\mathbf{N}$ , or  $\mathcal{M}(m, \Lambda)$  is undefined, where  $\Lambda$  denotes the *empty string* of the Oracle’s replies to the previous queries (i.e., when no queries to the Oracle have been asked yet—the empty history) and corresponds to the beginning state of the computation of strategy  $m$ . In the case of  $\mathcal{M}(m, \Lambda) = v \in \mathbf{N}$  we say that  $m$  defines (or is) a *constant strategy* giving rise to a final result  $v$  without asking the Oracle any questions. If  $\mathcal{M}(m, \Lambda)$  is undefined, then  $m$  is called an *undefined constant strategy*. In each of these cases we write, respectively,  $m = v_\alpha$  or  $m = \Omega_\alpha$  or even  $m = v$  or  $m = \Omega$ , especially when  $\alpha$  is itself a basic type. Intuitively, a constant strategy  $v_\alpha$  for  $\alpha = (\alpha, \dots, \alpha \rightarrow \iota)$  defines (computes) the constant functional  $\lambda x_1^{\alpha_1}, \dots, x_n^{\alpha_n}.v$  of the type  $\alpha$ . Analogously,  $\Omega_\alpha$  denotes  $\lambda x_1^{\alpha_1}, \dots, x_n^{\alpha_n}.\perp$ , the constant, undefined functional.

However, typically, the strategy  $m$  starts its computation by asking the Oracle sequentially some questions (concerning  $\bar{x}$ )

$$“A_1\{\bar{x}\} = ?”, “A_2\{\bar{x}\} = ?”, “A_3\{\bar{x}\} = ?”, \dots;$$

$$\mathcal{M}(m, \Lambda) = A_1\{\bar{x}\}, \mathcal{M}(m, v_1) = A_2\{\bar{x}\}, \mathcal{M}(m, v_1 v_2) = A_3\{\bar{x}\}, \dots,$$

assuming that the Oracle replied

$$“A_1\{\bar{x}\} = v_1”, “A_2\{\bar{x}\} = v_2”, \dots$$

We assume that the strategy  $m$  cannot continue computation until receiving the definite answer to the last asked query, if receiving any answer at all. This querying process can be either (i) finite with no result, if the Oracle does not answer a query, or (ii) infinite, or (iii) after some answers  $v_1, v_2, \dots, v_k$ ,  $m$  could “decide” that it has already received all the “required” answers from the Oracle and stop asking queries by returning a resulting value

$$\mathcal{M}(m, v_1 v_2 \cdots v_k) = v \in \mathbf{N},$$

instead of asking the next query  $A_{k+1}\{\bar{x}\}$ , if  $\mathcal{M}(m, v_1 v_2 \cdots v_k)$  is defined at all.

We say that  $m'$  is *descendant* to  $m$  if  $\mathcal{M}(m, w) = A\{x_1, \dots, x_n\}$  for some  $w$  and  $m'$  occurs in  $A\{x_1, \dots, x_n\}$  (that is,  $m$  asks about  $m'$ , or  $m'$  is a *child strategy* of  $m$ ) or, recursively,  $m'$  is descendant to a strategy occurring in  $A\{x_1, \dots, x_n\}$ . Intuitively, only descendant strategies matter for the meaning of the given strategy  $m$ .

3.1.3. *Additional Requirements on Systems of Strategies.* Without restricting generality we can impose the following natural requirements on systems of strategies.

- If  $\mathcal{M}(m, w) \in \mathbf{N}_\perp$  then  $\mathcal{M}(m, wu)$  is undefined for all non-empty  $u \in \mathbf{N}^*$ . (Contraposition: If  $\mathcal{M}(m, wu)$  is defined then  $\mathcal{M}(m, w)$  defines a query.)
- $\mathcal{M}(m, w)$  is defined only for *m-self-consistent* computational histories  $w = v_1 \cdots v_k$ , i.e. for such  $w$  which do not contain different answers to the same query by  $m$ : for all proper initial segments  $w^i = v_1 \cdots v_i$  and  $w^j = v_1 \cdots v_j$ ,

$$\mathcal{M}(m, w^i) = \mathcal{M}(m, w^j) \in \text{Basic-Terms}(M) \implies v_{i+1} = v_{j+1}.$$

Note that only computational histories satisfying these properties are realizable in the interpreted computations considered below in Section 4. The idea of consistency will be further generalized in Section 8.1 when considering nondeterministic wittingly consistent strategies.

Intuitively, each strategy  $m : \alpha$  computes some functional  $\llbracket m \rrbracket$  of the type  $\alpha$ . Let us first consider some simple examples.

3.1.4. *Examples of Strategies.* In these examples we assume that strategies compute functionals from the standard continuous model  $\{\mathbb{D}_\alpha\}$ . In the special case, when  $\mathcal{M}(m, \Lambda) = A\{\bar{x}\}$  and  $\mathcal{M}(m, v) = v$  for all basic values  $v \in \mathbf{N}$ , we represent (the behaviour of) such a strategy  $m$  by the formal equality

$$m\bar{x} = A\{\bar{x}\}.$$

This style of presentation allows us to avoid explicitly using  $\mathcal{M}$  when the behaviour of strategies is simple enough. It follows that the (typed) **PCF combinators** satisfying equalities<sup>3</sup>

$$\mathbf{I}x = x, \quad \mathbf{K}xy = x, \quad \mathbf{S}xyz = xz(yz), \quad \text{and} \quad \mathbf{Y}x = x(\mathbf{Y}x)$$

may be also considered as strategies. In fact, we can consider **PCF** [31, 25] as a system of strategies  $\langle \mathbf{PCF}, \mathcal{PCF} \rangle$  where  $\mathbf{PCF} = \{\mathbf{I}, \mathbf{K}, \mathbf{S}, \mathbf{Y}, \mathbf{if}, \text{and some evident basic arithmetical operations}\}$  with typing omitted for brevity<sup>4</sup>. Note that the *least fixed point operator*  $\mathbf{Y}$  is an example of a *recursive* strategy referring to itself. Another simple example of a strategy is the *conditional PCF* constant  $\mathbf{if} : \iota, \iota, \iota \rightarrow \iota$

$$\mathbf{if}xyz = \begin{cases} y, & \text{if } x = \mathbf{true} (= 1), \\ z, & \text{if } x = \mathbf{false} (= 0), \\ \perp, & \text{otherwise.} \end{cases}$$

This strategy asks at most two questions: first “ $x = ?$ ” and then, depending on the result **true** or **false**, it asks “ $y = ?$ ” or “ $z = ?$ ”, respectively. The answer received from the Oracle to the second question on  $y$  or  $z$  will be returned by  $\mathbf{if}$  as the final result of the computation. It is quite trivial to rewrite the above conditional equation for  $\mathbf{if}$  in terms of  $\mathcal{M}_{\mathbf{PCF}} = \mathcal{PCF}$ —in the style of Definition 3.1.

<sup>3</sup>Strictly speaking, we should use the canonical list of variables  $x_1, x_2, x_3, \dots$  instead of  $x, y, z$  and write, for example,  $\mathbf{S}x_1x_2x_3x_4 \cdots x_n = x_1x_3(x_2x_3)x_4 \cdots x_n$  for the base type terms.

<sup>4</sup>this is actually an infinite system.



Note that the following version of **if**, the *parallel conditional monotonic function*  $\mathbf{pif}_\iota : (o, \iota, \iota \rightarrow \iota)$  (and analogously for  $\mathbf{pif}_o : (o, o, o \rightarrow o)$ )<sup>5</sup> defined as

$$\mathbf{pif}_\iota p \text{ else } x \text{ then } y = \begin{cases} x, & \text{if } p = \mathbf{true}, \\ y, & \text{if } p = \mathbf{false}, \\ x, & \text{if } x = y, \\ \perp, & \text{otherwise,} \end{cases}$$

evidently has no computing it sequential strategy asking simple queries of the kind “ $p = ?$ ”, “ $x = ?$ ”, and “ $y = ?$ ” (and, in fact no sequential strategy at all, asking arbitrary queries). Say, if the first query asked by such a strategy is “ $p = ?$ ”, it may happen that the answer is undefined, leading to an undefined result of the whole computation, whereas it can be  $x = y \neq \perp$  which should give a defined result. Analogously, such a strategy could not start with “ $x = ?$ ” or “ $y = ?$ ”.

Unlike **pif**, every **PCF** constant can be considered as a sequential strategy. Say, the successor operation  $x + 1$  for  $x : \iota$  is defined by the evident strategy which asks the question “ $x = ?$ ” and, after getting a result  $v \in \mathbf{N}$  from the Oracle, returns the value  $v + 1$ .

As a less trivial example, consider the following strategy  $m$  computing the functional for the *weak sequential existential quantifier*  $\exists^{ws} : (\iota \rightarrow o) \rightarrow o$ :

$$\exists^{ws} P = \begin{cases} \mathbf{true}, & \text{if } Px = \mathbf{true} \text{ for some } x, \\ & \text{with } P(y) = \mathbf{false} \text{ for all } y < x, \\ \perp, & \text{otherwise} \end{cases}$$

To compute  $\exists^{ws} P$  (i.e.,  $mP$ ) this strategy starts by asking, sequentially, the queries “ $P0 = ?$ ”, “ $P1 = ?$ ”, ... to the Oracle. The strategy keeps asking these queries in this order while all the currently received answers are **false**. As soon as one of the answers obtained in this order is **true** or  $\perp$ , this value is the result of the computation. Alternatively,  $m$  could be defined as follows. Again,  $m$  starts with asking “ $P0 = ?$ ” ( $\mathcal{M}(m, \Lambda) = P0$ ). If the answer is **true**,  $m$  returns the result **true** ( $\mathcal{M}(m, \mathbf{true}) = \mathbf{true}$ ). Otherwise,  $m$  asks “ $m(\lambda x.P(x + 1)) = ?$ ”<sup>6</sup> and returns the answer of the Oracle to this query as the final result ( $\mathcal{M}(m, \mathbf{false}) = m(\lambda x.P(x + 1))$ ,  $\mathcal{M}(m, \mathbf{false} r) = r$ ). Here the lambda abstraction operator can be simulated, as usual, by combinatory strategies **S** and **K**. Then, to compute  $\exists^{ws}$ , the system of strategies should also contain strategies  $m$ , **S**, **K**, and  $+1$  (the successor). The functional  $\exists^{ws}$  can be also defined in **PCF** by the recursive equation

$$\exists^{ws} P = \mathbf{if} P0 \text{ then } \mathbf{true} \text{ else } \exists^{ws}(\lambda x.P(x + 1)),$$

or alternatively by using **Y**:

$$\exists^{ws} = \mathbf{Y} \lambda P. \mathbf{if} P0 \text{ then } \mathbf{true} \text{ else } \exists^{ws}(\lambda x.P(x + 1)).$$

Consider also the *finite sequential existential quantifiers*  $\exists_n^s : (\iota \rightarrow o) \rightarrow o$ ,  $n = 0, 1, \dots$  which can output both **true** and **false**:

$$\exists_n^s P = \begin{cases} \mathbf{true}, & \text{if } Px = \mathbf{true} \text{ for some } x \leq n, \\ & \text{with } P(y) = \mathbf{false} \text{ for all } y < x, \\ \mathbf{false}, & \text{if } P\perp = \mathbf{false}, \\ \perp, & \text{otherwise} \end{cases}$$

<sup>5</sup>Although we decided to avoid using the boolean type  $o$  in the general theory of strategies, the examples considered here are a little simpler and more natural when this type is used.

<sup>6</sup>This is a recursive query because  $m$  asks about itself.

The sequential strategy computing  $\exists_n^s P$  starts by asking  $n$  queries “ $P0 = ?$ ”, “ $P1 = ?$ ”,  $\dots$ , “ $Pn = ?$ ”. As soon as one of the answers obtained in this order will be **true** or  $\perp$  (undefined), this is the result of the computation. Otherwise, if all answers are **false**, the strategy asks “ $P\perp = ?$ ” and outputs the value of  $P\perp$ .

The sequence of functionals  $\exists_n^s$  is evidently increasing with a limit  $\exists^s \doteq \bigsqcup_n \exists_n^s$  which can be also defined as

$$\exists^s P = \begin{cases} \mathbf{true}, & \text{if } Px = \mathbf{true} \text{ for some } x, \\ & \text{with } P(y) = \mathbf{false} \text{ for all } y < x, \\ \mathbf{false}, & \text{if } P\perp = \mathbf{false}, \\ \perp, & \text{otherwise,} \end{cases}$$

or in terms of **PCF**:

$$\exists^s P = P(\mu x.Px), \quad \text{as well as,} \quad \exists_n^s P = P(\mu x \leq n.Px).$$

We omit the (well-known) definition in **PCF** of the  $\mu$ -operator. The sequential strategy computing  $\exists^s P$  reduces this task to the sub-task  $P(\mu x.Px)$ . The equation for  $\exists_n^s$  gives an analogous strategy. The main point here is that strategies may be quite arbitrarily complicated. As we will see in Theorem 7.2, all (effectively computable) strategies, however general, can be simulated in **PCF**, which characterises exactly its expressive power.

#### 4. INTERPRETED COMPUTATIONS AND THE DENOTATIONAL SEMANTICS OF STRATEGIES

**4.1. Preliminaries.** Let us fix a given system of strategies  $\langle M, \mathcal{M} \rangle$  and a monotonic, order extensional applicative structure  $\mathcal{E} = \{E_\alpha\}$  of finite type functionals, with  $E_\iota = \mathbf{N}_\perp$  and  $E_{\iota \rightarrow \iota} = [\mathbf{N}_\perp \rightarrow \mathbf{N}_\perp]$ . Our current goal is to define a *denotational semantics* of strategies

$$\llbracket - \rrbracket_\alpha = \llbracket - \rrbracket_\alpha^{\mathcal{M}} : M_\alpha \rightarrow E_\alpha, \quad \text{or briefly } \llbracket - \rrbracket : M \rightarrow \mathcal{E},$$

as the least fixed point of some operator  $\llbracket - \rrbracket \mapsto \llbracket - \rrbracket^+$ , that is, the least solution of the equation  $\llbracket - \rrbracket = \llbracket - \rrbracket^+$ . This equation is also understood as the requirement of *correctness* of the given semantics  $\llbracket - \rrbracket$ . In fact,  $\llbracket - \rrbracket^+ : M \rightarrow \mathcal{E}$  is defined via interpreted computations over  $\mathcal{E}$  performed by strategies of the system  $\langle M, \mathcal{M} \rangle$  relative to  $\llbracket - \rrbracket$ . The problem, however, concerns whether the operator  $\llbracket - \rrbracket \mapsto \llbracket - \rrbracket^+$  is well-defined and whether the required least fixed point  $\llbracket - \rrbracket$  exists. It does exist if  $\{E_\alpha\}$  is the standard continuous model  $\{\mathbb{D}_\alpha\}$ . It also exists for the monotonic model  $\mathbb{Q} = \{\mathbb{Q}_\alpha\}$  of hereditarily sequential functionals, which we will consider in Section 5. In both the definition of a system of strategies  $\langle M, \mathcal{M} \rangle$  and in earlier informal comments and examples it was *implicitly* assumed that both the Oracle and the strategy  $m$  always give *correct* (in a reasonable sense) answers/solutions to the queries/tasks they are “resolving”. This can be further clarified as follows.

**4.2. Formal Definitions.** Assume any semantic map  $\llbracket - \rrbracket : M \rightarrow \mathcal{E}$  is given. We can extend  $\llbracket - \rrbracket$  from  $M$  to terms  $\llbracket A\{\bar{x}\} \rrbracket$  with variables from the list  $\bar{x}$  as usual, by induction,  $\llbracket CD \rrbracket = \llbracket C \rrbracket \llbracket D \rrbracket$ , assuming that each variable  $x_i^{\alpha_i}$  has some associated value  $\llbracket x_i^{\alpha_i} \rrbracket \in E_{\alpha_i}$ . That is,  $\llbracket A\{\bar{x}\} \rrbracket$  depends on the values of  $\bar{x}$ . Then, for any computational strategy  $m : \alpha = (\alpha_1, \dots, \alpha_n \rightarrow \iota)$ , we define that the initial task “ $m\bar{x} = ?$ ” (to be “resolved” by  $m$ ) and all the queries “ $A\{\bar{x}\} = ?$ ” asked by  $m$  have corresponding *correct solutions* (with respect to  $\llbracket - \rrbracket$ )—just the unique basic values  $\llbracket m\bar{x} \rrbracket$  and  $\llbracket A\{\bar{x}\} \rrbracket$  of these Basic-Terms, respectively.

Let us now give the formal definition of *interpreted computation* of the basic value of  $m\bar{x}$  induced by a strategy  $m$  in a system of strategies  $\langle M, \mathcal{M} \rangle$  relative to some semantic map in  $\mathcal{E}$ ,  $\llbracket - \rrbracket : M \rightarrow \mathcal{E}$ , and some values of  $\bar{x}$  in  $\mathcal{E}$ . This is a maximal finite or infinite sequence of pairs

$$(A_1, v_1), (A_2, v_2), \dots \quad (4.1)$$

of queries and Oracle's answers, i.e. of terms  $A_i\{x_1^{\alpha_1}, \dots, x_n^{\alpha_n}\} \in \text{Basic-Terms}(M)$  and basic values  $v_i \in \mathbf{N}$ , which satisfy the following two conditions for each  $(A_i, v_i)$ :

$$\llbracket - \rrbracket_1: \mathcal{M}(m, v_1 \cdots v_{i-1}) = A_i,$$

$$\llbracket - \rrbracket_2: \llbracket A_i\{x_1^{\alpha_1}, \dots, x_n^{\alpha_n}\} \rrbracket = v_i \neq \perp \text{ (for the given values of } x_i^{\alpha_i} \text{ in } E_{\alpha_i}).$$

The latter means that *Oracle's answers  $v_i$  are correct* with respect to  $\llbracket - \rrbracket$  and the values of  $\bar{x}$ .

A finite (maximal) interpreted computation  $(A_1, v_1), \dots, (A_t, v_t)$ ,  $t \geq 0$ , is called *successful* with the *result*  $v \in \mathbf{N}$  if, additionally,

$$\llbracket - \rrbracket_3: \mathcal{M}(m, v_1 \cdots v_t) = v \in \mathbf{N}.$$

As sequential strategies are “deterministic”, the result  $v \in \mathbf{N}$  is determined uniquely, if it exists at all. If it does not exist, we also say that the result is *undefined* ( $\perp$ ). This is possible in the following cases:

- (i) the computation is infinite, or
- (ii) it is finite and consisting of  $t$  pairs, but unsuccessful, that is,  $\mathcal{M}(m, v_1, \dots, v_t)$  is either undefined, or = some  $A\{x_1^{\alpha_1}, \dots, x_n^{\alpha_n}\}$  with  $\llbracket A\{x_1^{\alpha_1}, \dots, x_n^{\alpha_n}\} \rrbracket = \perp$  (for the given values of  $x_j^{\alpha_j}$  in  $E_{\alpha_j}$ ).

Now let  $\llbracket m \rrbracket^{+\bar{x}}$  denote the result  $v$  in  $\mathbf{N}_\perp$  of the interpreted computation (according to  $\llbracket - \rrbracket_3$  above) of the value of  $m\bar{x}$  relative to  $\mathcal{E}$ ,  $\llbracket - \rrbracket$  and any values of  $\bar{x}$  in  $\mathcal{E}$ . Of course we would like to expect that  $\llbracket m \rrbracket^{+\bar{x}} = \llbracket m \rrbracket \bar{x}$  (i.e. that *the result of the computation is correct*) what, in general, is not true. For example, take  $\llbracket m \rrbracket = \perp$  for all  $m$  of a non-trivial system of strategies.

**Definition 4.1.**  $\llbracket - \rrbracket$  is called *computationally correct* if the equation  $\llbracket m \rrbracket^{+\bar{x}} = \llbracket m \rrbracket \bar{x}$  holds in  $\mathcal{E}$  wherever  $m\bar{x} : \iota$  or, briefly,  $\llbracket - \rrbracket = \llbracket - \rrbracket^+$ .

In general,  $\llbracket m \rrbracket^{+\bar{x}}$  is evidently monotonic on  $\bar{x}$ , as well as on  $\llbracket - \rrbracket$ , and defines a unique functional  $\llbracket m \rrbracket^+ : E_{\alpha_1} \times \cdots \times E_{\alpha_n} \xrightarrow{\text{mon}} E_\iota$ . But is this functional necessarily in  $E_\alpha \subseteq E_{\alpha_1} \times \cdots \times E_{\alpha_n} \xrightarrow{\text{mon}} E_\iota$ ? If true for all  $m$ , this defines a new semantic map  $\llbracket - \rrbracket^+ : M \rightarrow \mathcal{E}$  and a monotonic operator  $\llbracket - \rrbracket \mapsto \llbracket - \rrbracket^+$  (probably defined not for all  $\llbracket - \rrbracket$ ). In the case of the standard continuous model  $\{\mathbb{D}_\alpha\}$ , this operator, being computable in the above sense, is evidently well-defined and also continuous and, therefore, has the least fixed point which we also denote as  $\llbracket - \rrbracket$ . But in the general case of monotonic order extensional  $\{E_\alpha\}$  (and even of any continuous and directly complete  $\{E_\alpha\}$ , but containing possibly not all continuous functionals) the required value  $\llbracket m \rrbracket^+$  might not exist in the model and, even if it always exists, the monotonic operator  $\llbracket - \rrbracket \mapsto \llbracket - \rrbracket^+$  might be not continuous (in the case of arbitrary monotonic  $\{E_\alpha\}$ ) and may have no least fixed point.<sup>7</sup> But, when possible, we take  $\llbracket - \rrbracket$  to be the least solution of the equation  $\llbracket - \rrbracket = \llbracket - \rrbracket^+$ . Thus, we are interested in the least computationally correct denotational semantics of strategies.

<sup>7</sup>All of this seems quite plausible and desirable to confirm by example.

Moreover, for any model  $\mathcal{E} = \{E_\alpha\}$  and arbitrary system of sequential strategies  $\langle M, \mathcal{M} \rangle$ , let  $\llbracket m \rrbracket^0 \doteq \perp$  for all  $m \in M$  and  $\llbracket - \rrbracket^{n+1} \doteq (\llbracket - \rrbracket^n)^+$  assuming the latter is well-defined in  $\mathcal{E}$ . Evidently, those  $\llbracket - \rrbracket^n$  which exist are defined uniquely. It follows from the monotonicity of  $^+$  and monotonicity and order extensionality of  $\mathcal{E}$  by induction on  $n$  that  $\llbracket - \rrbracket^n \sqsubseteq \llbracket - \rrbracket^{n+1} \sqsubseteq \llbracket - \rrbracket$  assuming  $\llbracket - \rrbracket$  is an arbitrary computationally correct semantics.

**Definition 4.2.**  $\llbracket - \rrbracket$  is called *naturally defined* in  $\mathcal{E}$  if all  $\llbracket - \rrbracket^n$  exist and

$$\llbracket - \rrbracket = \biguplus_{n=0}^{\infty} \llbracket - \rrbracket^n,$$

that is,  $\llbracket m \rrbracket = \biguplus_{n=0}^{\infty} \llbracket m \rrbracket^n$  holds for each  $m \in M$  where  $\biguplus$  is the natural, or pointwise lub in  $\mathcal{E}$ , as defined in Section 2.2.

**Proposition 4.3.**

- (a) If  $\llbracket - \rrbracket : M \rightarrow \mathcal{E}$  is naturally defined in  $\mathcal{E}$  then it is  $\sqsubseteq$  any computationally correct semantics in  $\mathcal{E}$ . Thus, if  $\llbracket - \rrbracket$  is also computationally correct then it is the least one.
- (b) Moreover, if  $\mathcal{E}$  is naturally continuous and  $\llbracket - \rrbracket$  is naturally defined then it is, indeed, the least computationally correct semantics of  $\langle M, \mathcal{M} \rangle$ .

*Proof.*

- (a) The conclusion follows from the same statement on all  $\llbracket - \rrbracket^n$ .
- (b) Just the equality  $\llbracket - \rrbracket = \biguplus_{n=0}^{\infty} \llbracket - \rrbracket^n$  implies that  $\llbracket m \rrbracket \bar{x} \sqsubseteq \llbracket m \rrbracket^n \bar{x} \sqsubseteq \llbracket m \rrbracket^{n+\bar{x}} \sqsubseteq \llbracket m \rrbracket^{+\bar{x}}$  holds whenever  $m\bar{x} : \iota$  for some  $n$  depending on  $\bar{x}$ . The converse inequalities  $\llbracket m \rrbracket^{+\bar{x}} \sqsubseteq \llbracket m \rrbracket^{n+\bar{x}} \sqsubseteq \llbracket m \rrbracket \bar{x}$  hold for appropriate  $n$  depending on  $\bar{x}$  by using natural continuity of  $\mathcal{E}$ . It follows that  $\llbracket - \rrbracket = \llbracket - \rrbracket^+$ , as required.  $\square$

**Definition 4.4.** If the naturally defined semantics  $\llbracket - \rrbracket : M \rightarrow \mathcal{E}$  exists in  $\mathcal{E}$  and is (the least) computationally correct for all sequential systems of strategies then  $\mathcal{E}$  is called *sequentially complete*.

Besides the evident example of the standard directly complete continuous model  $\{\mathbb{D}_\alpha\}$ , the sequential completeness property holds also for the model  $\{\mathbb{Q}_\alpha\}$  of hereditarily sequential functionals considered below in Sections 5–7. An analogous result takes place for another model  $\{\mathbb{W}_\alpha\}$  and a more general concept of nondeterministic (wittingly consistent) strategies considered in Section 8.

**Definition 4.5.** Finite type functionals in  $\mathcal{E}$  of the form  $\llbracket m \rrbracket$  for any strategy  $m$  of any system  $\langle M, \mathcal{M} \rangle$  (for the least computationally correct semantics  $\llbracket - \rrbracket$ , if it does exist) are called *sequential*.<sup>8</sup> If  $m$  is a strategy from an (effectively) computable system of strategies  $\langle M, \mathcal{M} \rangle$  (i.e. with computable  $\mathcal{M}$ ), then  $\llbracket m \rrbracket$  is called an *effectively-sequential* functional.<sup>9</sup>

This is the way that sequential and effectively-sequential finite type functionals (in appropriate  $\{E_\alpha\}$ ) can be defined in quite general terms of computational strategies [28].

<sup>8</sup>We do not expect that this concept is really interesting for arbitrary  $\mathcal{E}$ . Although it is reasonable to restrict attention to naturally continuous and sequentially complete models, it may be unknown in advance that the given structure (such as  $\mathbb{Q}$  or  $\mathbb{W}$  considered below) satisfies these properties. Thus, for the sake of the argument, we need the general definition.

<sup>9</sup>A sequential functional can also be called *sequentially computable*, although the corresponding strategy could be not (effectively) computable at all. That is, the concept of sequential computability is, in fact, a relative one (see also [34, 27]).

The same approach works for the type-free version of sequentiality [29] in the Scott model  $\mathbb{D}_\infty \cong [\mathbb{D}_\infty \rightarrow \mathbb{D}_\infty]$ <sup>10</sup>. It could be also extended to more general type theories and models and also for more general kinds of basic values than the flat  $\mathbf{N}_\perp$ .

## 5. HEREDITARILY SEQUENTIAL FUNCTIONALS

### 5.1. Canonical Strategies.

**Definition 5.1.** A system of strategies is said to be in the *canonical form* if all queries “ $A\{\bar{x}\} = ?$ ” asked by these strategies  $m$  (with  $m\bar{x} = mx_1 \cdots x_n$  of the basic type) have the form

$$“x_i(m_1x_1 \cdots x_n) \cdots (m_{k_i}x_1 \cdots x_n) = ?” \quad (5.1)$$

where each  $m_k\bar{x} = m_kx_1 \cdots x_n$  has a type suitably depending on the type of the head variable  $x_i$ .

For example, a strategy  $m$  of the type  $(\iota \rightarrow \iota) \rightarrow \iota$  computing a functional  $mf : \iota$  with  $f : \iota \rightarrow \iota$  can ask queries of canonical form “ $f(m'f) = ?$ ” or, in particular, “ $fn = ?$ ” if  $m'f$  is a constant functional having the integer value  $n \in \mathbf{N}$ . Note that for sequential computability of such functionals it is insufficient to consider queries of the form “ $fn = ?$ ”. As we will see in Section 6, the canonical form of queries does not restrict the computational and denotational power of sequential strategies. Importantly, the descendant strategies  $m_k$  in (5.1) have evidently *the same, or lower, level* (of their types) than  $m$ . This will serve below as the base for the inductive definition of hereditarily sequential functionals in terms of canonical systems of strategies.

**5.2. The Main Inductive Definition.** By using the above property of levels of strategies in canonical systems we can give the following inductive (level-by-level) definition of a *monotonic order extensional structure*  $\{\mathbb{Q}_\alpha\}$  of *hereditarily sequential functionals* which will be shown later to be *fully abstract model for PCF*. The initial part of this model for types up to level  $l$  is denoted as  $\mathbb{Q}^{\leq l}$ .

**Definition 5.2.** For level 0, let  $\mathbb{Q}_\perp \equiv \mathbf{N}_\perp$  be the flat basic domain. Assume, by induction, that the initial part of the model  $\mathbb{Q}^{\leq l}$  satisfying (2.1) and (2.2) has been defined. For any  $\alpha = (\alpha_1, \dots, \alpha_n \rightarrow \iota)$  of level  $l+1$  take the minimal  $k \leq n$  such that  $(\alpha_{k+1}, \dots, \alpha_n \rightarrow \iota)$  is of the level  $\leq l$ , and let, up to uncurrying<sup>11</sup>,

$$\bar{\mathbb{Q}}_\alpha \cong (\mathbb{Q}_{\alpha_1} \times \cdots \times \mathbb{Q}_{\alpha_k} \xrightarrow{\text{mon}} \mathbb{Q}_{(\alpha_{k+1}, \dots, \alpha_n \rightarrow \iota)}).$$

More precisely, let

$$\bar{\mathbb{Q}}_\alpha \equiv \{f : \mathbb{Q}_{\alpha_1} \times \cdots \times \mathbb{Q}_{\alpha_n} \xrightarrow{\text{mon}} \mathbb{Q}_\perp \mid \forall \bar{x} \in \mathbb{Q}_{\alpha_1} \times \cdots \times \mathbb{Q}_{\alpha_k} (f\bar{x} \in \mathbb{Q}_{(\alpha_{k+1}, \dots, \alpha_n \rightarrow \iota)})\}.$$

<sup>10</sup> Actually, a closely related and “stronger” isomorphism  $\mathbb{D}_\infty \cong [\mathbb{D}_\infty \times \mathbb{D}_\infty \times \cdots \rightarrow \mathbb{D}_\infty]$  should be used. Note that this isomorphism evidently implies  $\mathbb{D}_\infty \cong [\mathbb{D}_\infty \rightarrow [\mathbb{D}_\infty \times \cdots \rightarrow \mathbb{D}_\infty]]$  and hence  $\mathbb{D}_\infty \cong [\mathbb{D}_\infty \rightarrow \mathbb{D}_\infty]$ . This allows us to consider strategies asking (infinite) applicative queries over  $\mathbb{D}_\infty$  of the basic type  $\iota$ , like in the typed approach.

<sup>11</sup>Note that the simpler definition  $\bar{\mathbb{Q}}_\alpha \equiv \{f : \mathbb{Q}_{\alpha_1} \times \cdots \times \mathbb{Q}_{\alpha_n} \xrightarrow{\text{mon}} \mathbb{Q}_\perp\}$  does not work because we need to have below  $\bar{\mathbb{Q}}^{\leq l+1}$  to be an appropriate structure closed under application.

Then

$$\bar{\mathbb{Q}}^{\leq l+1} \Rightarrow \mathbb{Q}^{\leq l} \cup \{\bar{\mathbb{Q}}_\alpha \mid \alpha \text{ is of level } l+1\} \quad (5.2)$$

can be considered as a monotonic, order extensional applicative structure up to level  $l+1$  with the application operator defined by taking the residual map, as in (2.2). Then, for any  $\alpha$  of level  $l+1$ , define  $\mathbb{Q}_\alpha \subseteq \bar{\mathbb{Q}}_\alpha$ :

$$\begin{aligned} \mathbb{Q}_\alpha \Rightarrow \{ \llbracket m \rrbracket \in \bar{\mathbb{Q}}_\alpha \mid m : \alpha \ \& \ m \in M \ \& \ [-] : M \rightarrow \bar{\mathbb{Q}}^{\leq l+1} \\ \text{for some canonical system of strategies } M \} \end{aligned} \quad (5.3)$$

as the set of all monotonic mappings in  $\bar{\mathbb{Q}}_\alpha$  which are computable/definable (as described in Section 4.2) by the strategies  $m$  of the type  $\alpha$  of any system of strategies in canonical form<sup>12</sup> for which the least correct semantics  $\llbracket - \rrbracket$  in the structure  $\bar{\mathbb{Q}}^{\leq l+1}$  exists. In fact, we can equivalently<sup>13</sup> require that  $\llbracket - \rrbracket$  is *naturally defined* (see Definition 4.4).

Alternatively, and equivalently (see the comments below), we can define for any  $\alpha$  of level  $l+1$

$$\begin{aligned} \mathbb{Q}_\alpha \Rightarrow \{ \llbracket m \rrbracket \llbracket m_1 \rrbracket \cdots \llbracket m_r \rrbracket \in \bar{\mathbb{Q}}_\alpha \mid m : (\gamma_1, \dots, \gamma_r \rightarrow \alpha) \ \& \ m_i : \gamma_i \\ \ \& \ \text{level}(\gamma_i) \leq l \ \& \ m, m_i \in M \\ \ \& \ [-] : M \rightarrow \bar{\mathbb{Q}}^{\leq l+1} \text{ correct and naturally defined} \\ \text{for some canonical system of strategies } M \}. \end{aligned} \quad (5.4)$$

(See also Proposition 4.3 (a).) Sets of functionals  $\mathbb{Q}_\alpha$  defined in this way for  $\alpha$  of level  $l+1$  are evidently nonempty and contain at least all the constant functionals. In particular, they contain the elements  $\perp_\alpha$  computable by the undefined strategies  $\Omega_\alpha$ . They are considered to be partially ordered pointwise by  $\sqsubseteq_\alpha$ . This defines the extension  $\mathbb{Q}^{\leq l+1}$  of  $\mathbb{Q}^{\leq l}$  which satisfies (2.1) and (2.2). (The latter property of the closure under application follows straightforwardly assuming (5.4). This is much more difficult to show, if the more intuitively plausible (5.3) is assumed instead; see the comments below.)

This makes the induction step mathematically correct because we assumed, and used, the fact that  $\mathbb{Q}^{\leq l}$  satisfies only (2.1) and (2.2). Thus, (5.4) defines a monotonic order extensional structure  $\mathbb{Q}$  by induction.

### Comments.

- (1) The induction step above defines *simultaneously* all  $\mathbb{Q}_\alpha$  of level  $l+1$ . The canonical form of strategies guarantees that *no*  $\mathbb{Q}_\tau$  of a higher level (not yet defined) will be needed in the induction step. By contrast, recall that, for example, Milner's definition of the fully abstract depo model, as well as later approaches to non-depo models, requires consideration of *all* types and levels at once.
- (2) Although (5.3) and (5.4) are, in fact, equivalent definitions of  $\mathbb{Q}_\alpha$  at the level  $l+1$ , unfortunately this is not so trivial and when taking the simpler equation (5.3) the proof

<sup>12</sup>Without restricting generality, these systems may be evidently considered as containing only strategies of types  $\tau$  up to level  $l+1$ .

<sup>13</sup>This will be clear later from isomorphic representation of  $\mathbb{Q}$  as  $\tilde{\mathbb{Q}}$  and Theorem 6.6 (b). See also Proposition 5.3.

of the *correctness* of the whole definition would be rather involved.<sup>14</sup> For the inductive step in Definition 5.2 to be legal in this case we must show that the resulting  $\mathbb{Q}^{\leq l+1}$  satisfies both (2.1) and (2.2). The condition (2.1) holds by definition, and (2.2) means that  $\mathbb{Q}^{\leq l+1}$  is closed under application (also for results of the level  $l+1$ ), that is under taking residual maps, like for  $\bar{\mathbb{Q}}^{\leq l+1}$ . This is quite straightforward in the case of (5.4), unlike the case of (5.3) although the latter looks more natural. This is the reason for our choice of (5.4) in the above definition<sup>15</sup>. The equivalence of (5.3) and (5.4) will be shown later, as well as that an arbitrary system of sequential strategies, not necessary in the canonical form, has the least correct and even naturally defined denotational semantics  $\llbracket - \rrbracket$  in  $\mathbb{Q}$  (that is  $\mathbb{Q}$  is sequentially complete), and that each element in  $\mathbb{Q}_\alpha$  should have the form  $\llbracket m \rrbracket$  for some (even canonical) strategy  $m : \alpha$ . The latter means that  $\mathbb{Q}$  consists of all, and only, sequentially computable functionals.

- (3) In general, we want to know that this structure is natural enough (although it is not a directly complete poset). That is it is a fully abstract model for **PCF**, sequentially complete, naturally continuous, naturally algebraic and naturally bounded complete; we establish this later. But now we can prove a conditional

**Proposition 5.3.** *If some sequentially complete model  $\mathbb{Q}'$  exists and each of its elements has the form  $\llbracket m' \rrbracket$  for a strategy in some system of strategies in canonical form for  $\llbracket - \rrbracket$  the (least) correct and naturally defined semantics in  $\mathbb{Q}'$  then  $\mathbb{Q}' \cong \mathbb{Q}$ . It follows that in this case all the mentioned variations of the Definition 5.2 give rise to the same  $\mathbb{Q}$ .*

*Proof.* Assuming that  $\mathbb{Q}'$  (as well as  $\mathbb{Q}$ ) satisfies (2.1) and (2.2) we can even show the identity  $\mathbb{Q}' = \mathbb{Q}$ . Thus, given by induction  $\mathbb{Q}'^{\leq l} = \mathbb{Q}^{\leq l}$  (as is definitely true for  $l = 0$ ) and therefore  $\bar{\mathbb{Q}}'_\alpha = \bar{\mathbb{Q}}_\alpha$ , for  $\alpha$  of level  $l+1$ , and  $\bar{\mathbb{Q}}'^{\leq l+1} = \bar{\mathbb{Q}}^{\leq l+1}$ , let us show that  $\mathbb{Q}'_\alpha = \mathbb{Q}_\alpha$ . But, according to (5.4) and our assumptions (in particular, the closure of  $\mathbb{Q}'$  under applications as taking residuals), we have

$$\begin{aligned} \mathbb{Q}'_\alpha &= \{ \llbracket m \rrbracket \llbracket m_1 \rrbracket \cdots \llbracket m_r \rrbracket \in \mathbb{Q}'_\alpha \mid \dots \mathbb{Q}'^{\leq l+1} \dots \} \\ &= \{ \llbracket m \rrbracket \llbracket m_1 \rrbracket \cdots \llbracket m_r \rrbracket \in \bar{\mathbb{Q}}'_\alpha \mid \dots \bar{\mathbb{Q}}'^{\leq l+1} \dots \} \\ &= \{ \llbracket m \rrbracket \llbracket m_1 \rrbracket \cdots \llbracket m_r \rrbracket \in \bar{\mathbb{Q}}_\alpha \mid \dots \bar{\mathbb{Q}}^{\leq l+1} \dots \} \\ &\equiv \mathbb{Q}_\alpha \end{aligned}$$

with the omitted parts “...” as in (5.4). In the second equality we use the routinely checked fact that the naturally defined and correct semantic map  $\llbracket - \rrbracket$  in  $\mathbb{Q}'^{\leq l+1}$  is also naturally defined and correct in the extension  $\bar{\mathbb{Q}}'^{\leq l+1} \supseteq \mathbb{Q}'^{\leq l+1}$  because  $\mathbb{Q}'^{\leq l+1}$  is closed under applications and all corresponding arguments and answers to all queries considered are evidently the same in both structures  $\mathbb{Q}'^{\leq l+1}$  and  $\bar{\mathbb{Q}}'^{\leq l+1}$ . (Proposition 4.3 (a) shows that  $\llbracket - \rrbracket$  is in fact the least correct semantics).  $\square$

<sup>14</sup>Note that even for the standard definition of (hereditarily) continuous functionals in  $\{\mathbb{D}_\alpha\}$  some correctness proof is necessary. Of course, the case of  $\{\mathbb{Q}_\alpha\}$  is more complicated. Instead of contrasting the continuous case with the sequential one we prefer to see some analogy here. Thus, both approaches are essentially extensional with some intensional component in each case, even if these intensional components have somewhat different flavour and complexity.

<sup>15</sup>Thanks to an anonymous referee for suggesting the formula (5.4) which crucially simplified (made it just straightforward) correctness proof of the induction step of the definition of  $\mathbb{Q}$ . Based originally on (5.3) it required the full theory of sequential strategies of the next sections. But, anyway, this theory is still needed to prove the main properties of  $\mathbb{Q}$ .

In particular, once the above shows  $\mathbb{Q} = \mathbb{Q}'$ , we have a simplified version of (5.3)

$\mathbb{Q}_\alpha = \{\llbracket m \rrbracket \in \mathbb{Q}_\alpha \mid \llbracket - \rrbracket \text{ is the least correct semantics of a canonical system in } \mathbb{Q}^{\leq l+1}\}$   
with the extensions  $\bar{\mathbb{Q}}_\alpha$  and  $\tilde{\mathbb{Q}}^{\leq l+1}$  no more necessary to mention.

**5.3. What Next?** For showing the required properties of  $\mathbb{Q}$  such as continuity and sequential completeness we will need more involved considerations and develop the corresponding general theory of sequential strategies [29, 28] in Sections 6 and 7.

In particular, to represent the application operation in  $\{\mathbb{Q}_\alpha\}$  we will need to define corresponding operation  $\langle\langle mm_1 \rangle\rangle$  for arbitrary strategies  $m : \alpha = (\alpha_1, \dots, \alpha_n \rightarrow \iota)$  and  $m_1 : \alpha_1$ , giving a “residual” strategy  $\langle\langle mm_1 \rangle\rangle$  of the type  $(\alpha_2, \dots, \alpha_n \rightarrow \iota)$ , such that  $\llbracket \langle\langle mm_1 \rangle\rangle \rrbracket = \llbracket m \rrbracket \llbracket m_1 \rrbracket$ ; cf. Theorem 6.6 (a). It is crucial here that  $\langle\langle - \rangle\rangle$  serves as the operational semantics of strategies of arbitrary, not necessarily the basic types.

In fact, we will *redefine* our model in a non-inductive, “quotient” form  $\{\tilde{\mathbb{Q}}_\alpha\} \cong \{\mathbb{Q}_\alpha\}$  where  $Q = \bigcup \{Q_\alpha\}$  is a unique *universal* system of sequential strategies (containing in a sense all other systems—the unique up to isomorphism terminal object of the category of all systems of strategies) and will work mainly in terms of  $Q$  and  $\tilde{Q}$ .

This general theory is based on the *operational semantics* of strategies and will culminate in Sections 6 in Theorem 6.6 and its Corollary 6.7 (using the above Proposition 5.3) that  $\tilde{Q} \cong Q$ . Moreover, we will also prove in Section 7 that  $\{\tilde{\mathbb{Q}}_\alpha\}$  is a fully abstract model of **PCF** and has further good domain theoretic properties discussed in Section 2.

## 6. SEQUENTIAL FUNCTIONALS AS QUOTIENT STRATEGIES

According to [29, 30], there exists a *universal* system of sequential strategies  $\langle Q, \mathcal{Q} \rangle$  (with  $Q$  of the cardinality of continuum) such that for any other system of strategies  $\langle M, \mathcal{M} \rangle$  there exist a *unique* homomorphism  $\varphi : \langle M, \mathcal{M} \rangle \rightarrow \langle Q, \mathcal{Q} \rangle$ . For the rest of this paper we will need only the existence of  $\langle Q, \mathcal{Q} \rangle$ , however its explicit construction is presented in Appendix A. In general, a *homomorphism*  $\varphi : \langle M, \mathcal{M} \rangle \rightarrow \langle M', \mathcal{M}' \rangle$  is a map  $\varphi : M \rightarrow M'$  preserving types such that

$$\begin{aligned} \mathcal{M}'(\varphi(m), w) &= (\mathcal{M}(m, w))^\varphi \quad \text{holds for all } m \in M, w \in \mathbf{N}, \text{ where} \\ (AB)^\varphi &= (A^\varphi)(B^\varphi), \quad m^\varphi = \varphi(m), \quad v^\varphi = v, \quad x^\varphi = x, \quad \text{and } \perp^\varphi = \perp \end{aligned}$$

for any applicative terms  $A, B$ , strategy  $m$ , basic value  $v$  and variable  $x$ . That is, a homomorphic image of a strategy has essentially “the same” behaviour.<sup>16</sup> The fact that  $\varphi$  can map different strategies in  $M$  to the same strategy in  $M'$  means that the latter is more “abstract” version of the former. Homomorphisms are evidently closed under compositions:  $\mathcal{M}''(\varphi \circ \psi(m), w) = (\mathcal{M}'(\psi(m), w))^\varphi = ((\mathcal{M}(m, w))^\psi)^\varphi$ .

Moreover, any strategy  $m$  and its homomorphic image  $\varphi(m)$  have the same denotational semantics in the following sense.

**Proposition 6.1.** *Let  $\varphi : \langle M, \mathcal{M} \rangle \rightarrow \langle M', \mathcal{M}' \rangle$  be a homomorphism.*

<sup>16</sup> In particular,  $\perp^\varphi = \perp$  means that both  $\mathcal{M}'(\varphi(m), w)$  and  $\mathcal{M}(m, w)$  are defined, or not. A more general concept of an *approximating homomorphism* is obtained by allowing the requirement  $\mathcal{M}'(\varphi(m), w) = (\mathcal{M}(m, w))^\varphi$  only in the case of  $\mathcal{M}(m, w) \neq \perp$ . That is,  $\varphi(m)$  has “the same or more definite” behaviour than  $m$ .



- (a) For any  $\llbracket - \rrbracket' : M' \rightarrow \mathcal{E}$  and its composition  $\llbracket - \rrbracket \Leftarrow \llbracket \varphi(-) \rrbracket' : M \rightarrow \mathcal{E}$  the corresponding results of the interpreted computations coincide:  $\llbracket m \rrbracket^+ \bar{x} = \llbracket \varphi(m) \rrbracket'^+ \bar{x}$  whenever  $m\bar{x} : \iota$ .
- (b)  $\llbracket - \rrbracket^n = \llbracket \varphi(-) \rrbracket'^n$  holds assuming  $\llbracket - \rrbracket'^n$  exists.<sup>17</sup>
- (c) If  $\llbracket - \rrbracket'$  is computationally correct (resp., naturally defined) then so is the composition  $\llbracket - \rrbracket \Leftarrow \llbracket \varphi(-) \rrbracket'$ .
- (d) For  $\mathcal{E}$  sequentially complete,  $\llbracket - \rrbracket = \llbracket \varphi(-) \rrbracket'$  holds for the (least) computationally correct and naturally defined semantics  $\llbracket - \rrbracket$  and  $\llbracket - \rrbracket'$  of these two systems, respectively.

*Proof.*

(a) follows from the similarity of the two interpreted computations via the homomorphism  $\varphi$ .

(a)  $\Rightarrow$  (b) (by induction):

$$\llbracket - \rrbracket^0 = \llbracket \varphi(-) \rrbracket'^0; \quad \llbracket - \rrbracket^n = \llbracket \varphi(-) \rrbracket'^n \implies \llbracket m \rrbracket^{n+} \bar{x} = \llbracket \varphi(m) \rrbracket'^{n+} \bar{x},$$

(a)  $\Rightarrow$  the first part of (c):

$$\llbracket m \rrbracket \bar{x} \Leftarrow \llbracket \varphi(m) \rrbracket' \bar{x} \stackrel{\text{corr.}}{\equiv} \llbracket \varphi(m) \rrbracket'^+ \bar{x} \stackrel{(a)}{\equiv} \llbracket m \rrbracket^+ \bar{x}.$$

(b)  $\Rightarrow$  the second part of (c):

$$\llbracket - \rrbracket' = \bigoplus_n \llbracket - \rrbracket'^n \implies \llbracket m \rrbracket \Leftarrow \llbracket \varphi(m) \rrbracket' = \bigoplus_n \llbracket \varphi(m) \rrbracket'^n \stackrel{(b)}{\equiv} \bigoplus_n \llbracket m \rrbracket^n.$$

(c)  $\Rightarrow$  (d). (See also Proposition 4.3 (a).) □

Therefore, it is natural to identify informally  $m$  with  $\varphi(m)$  and with their unique homomorphic image in  $\langle Q, \mathcal{Q} \rangle$ , and to consider the latter as a really universal system of strategies “containing” all possible strategies (up to homomorphism).

Various strategies in  $Q_\alpha \subseteq Q$  computing the same functional in  $\mathbb{Q}_\alpha$ ,  $\llbracket q \rrbracket = \llbracket q' \rrbracket$ , may be identified via an equivalence relation  $q \simeq_\alpha q'$  which will be also defined in Section 6.4 by using operational semantics of strategies over  $\langle Q, \mathcal{Q} \rangle$  so that we will actually have  $\mathbb{Q}_\alpha$  isomorphic to  $\tilde{Q}_\alpha \Leftarrow Q_\alpha / \simeq_\alpha$  and even could take the equality  $\mathbb{Q}_\alpha = \tilde{Q}_\alpha$  as (another) definition of  $\mathbb{Q}_\alpha$ . Moreover, we will define a preorder relation  $\preceq_\alpha$  on the strategies in  $Q_\alpha$  generating  $\simeq_\alpha$  as the corresponding equivalence relation and inducing the approximation relation  $\sqsubseteq_\alpha$  on  $\tilde{Q}_\alpha$  (that is,  $\sqsubseteq_\alpha = \preceq_\alpha / \simeq_\alpha$ ) which, in fact, exactly corresponds to the pointwise approximation relation on  $\mathbb{Q}_\alpha$  assumed in Section 5.2.

**6.1. Operational Semantics for Strategies (Informally).** Following [29], we will define an operation  $\langle\langle pq \rangle\rangle$  of the application of strategies (having appropriate types) of the universal system  $\langle Q, \mathcal{Q} \rangle$ . More generally, given any combination  $A$  of any type  $\alpha$  consisting only of strategies, a new strategy can be defined  $\langle\langle A \rangle\rangle \in Q$  of the same type  $\alpha$  (also denoted in the op. cit. as  $\hat{A}$ ). In particular,  $A$  and  $\langle\langle A \rangle\rangle$  should have the same denotational meaning in any “reasonable” model  $\mathcal{E}$ , that is,

$$\llbracket \langle\langle A \rangle\rangle \rrbracket = \llbracket A \rrbracket, \text{ or } \llbracket \langle\langle pq \rangle\rangle \rrbracket = \llbracket p \rrbracket \llbracket q \rrbracket.$$

This will be achieved in terms of a quite natural computation process induced by the strategies involved in  $A$ , without any reference to any model  $\{E_\alpha\}$ . That is why this may

<sup>17</sup>For approximating homomorphisms defined in Footnote 16 we rather have  $\llbracket m \rrbracket^n \sqsubseteq \llbracket \varphi(m) \rrbracket'^n$  for all  $n = 0, 1, \dots$

be considered as an *operational semantics*  $\langle\langle-\rangle\rangle$  for the terms  $A$ , unlike the denotational semantics  $\llbracket-\rrbracket$ .

Therefore, let us consider the formal expression  $\langle\langle A \rangle\rangle$  as a strategy (or we could take its unique homomorphic image in  $Q$ ). We need to define the action of  $\mathcal{Q}(\langle\langle A \rangle\rangle, u)$  for any string of the Oracle's answers  $u \in \mathbf{N}^*$ . It is both simpler and instructive to first consider the case when  $A$  and  $\langle\langle A \rangle\rangle$  have the basic type  $\iota$ . Such a strategy asks the Oracle no questions and “computes” some basic value  $\mathcal{Q}(\langle\langle A \rangle\rangle, \Lambda) = v \in \mathbf{N}$ , if defined at all, for  $u = \Lambda$  (the empty string of Oracle's answers). Thereby, the corresponding initial task “ $\langle\langle A \rangle\rangle = ?$ ” or task “ $A = ?$ ” of finding this basic value  $v$  will be resolved with the help of strategies participating in  $A$  by reducing this task (by induction) to some sub-sub- $\dots$ -tasks “ $C = ?$ ”. Here all  $C$  are terms of the basic type consisting only of strategies, and therefore having a numerical solution (if any) computed by induction in the same way until the original task “ $A = ?$ ” is resolved. In fact, each sub-sub- $\dots$ -task  $C$  has the form  $C = mD_1D_2 \cdots D_k$ , that is headed by a strategy  $m$  which asks further queries (reduces  $C$  to further immediate sub-tasks), and continues the computation of the value of  $C$  on the basis of the replies obtained. This generalizes the reduction process of lambda calculus or the natural (call-by-name) computation of the value of a closed **PCF** term of the basic type.

In the general case, when the strategy  $\langle\langle A \rangle\rangle$  or the term  $A$  has an arbitrary, non-basic type  $\alpha = (\alpha_1, \dots, \alpha_k \rightarrow \iota)$ , we need to consider the initial task “ $\langle\langle A \rangle\rangle \bar{y} = ?$ ” or “ $A \bar{y} = ?$ ” of the basic type  $\iota$ , with the variables  $y_j : \alpha_j$ . Then it will be reduced to various sub-sub- $\dots$ -tasks  $C : \iota$  which can now involve the variables  $\bar{y}$ . If  $C = mD_1D_2 \cdots D_k$  is headed by a strategy  $m$  then the further computation (reduction to further immediate sub-tasks of  $C$ ) proceeds as in the case above when all tasks considered had no variables. But it is also possible that  $C = y_j D_1 D_2 \cdots D_{n_j}$  is headed by a variable  $y_j$  in  $\bar{y}$ . Here we assume that the computation continues with the help of an arbitrary (now non-empty) prompt  $u$  by the Oracle because the head variable  $y_j$  itself does not have the “ability” to continue the computation of  $C$ .

For the initial task “ $A \bar{y} = ?$ ” we actually want to know/compute: under which prompts  $u$  from the Oracle, which sub-sub- $\dots$ -tasks  $C$  headed by a variable, or which resulting values in  $\mathbf{N}$  can be generated? (The tasks  $C$  headed by a strategy will continue the computation themselves.) This is essentially the way (with many details omitted) how  $\mathcal{Q}(\langle\langle A \rangle\rangle, u)$  can be defined (computed) by this process.

Formally, at each point we have a state of the computation like a “stack” (a finite string consisting of pending sub-sub- $\dots$ -tasks and basic values as the intermediate results) which may “pulsate” during time as we will see in the formal definition below.

## 6.2. Operational Semantics for Strategies—Formal Definitions. Consider

- a system of strategies  $\langle M, \mathcal{M} \rangle$ ,
- an applicative term  $C = mD_1 \cdots D_n \in \text{Basic-Terms}(M)$  (in the role of a currently considered task or sub-sub- $\dots$ -task of some initial task) with a head strategy  $m \in M$  and possibly involving variables.
- the canonical list of variables  $\bar{x} = x_1, \dots, x_n$  for  $m$  (such that  $m\bar{x} : \iota$ ), and
- a prompt  $w \in \mathbf{N}^*$ .

Three cases are possible:

(M1)  $\mathcal{M}(m, w) = v \in \mathbf{N}$ ,

(M2)  $\mathcal{M}(m, w)$  is undefined, or

(M3)  $\mathcal{M}(m, w) = B = B\{x_1, \dots, x_n\} \in \text{Basic-Terms}(M)$

in which we will, respectively, say that the task  $C = mD_1 \dots D_n$  (or “ $C = ?$ ”) is *w-reducible* (M1) to the *result*  $v$ , or (M2) to the *result*  $\perp$ , or (M3) to the *immediate sub-task*  $C' = B\{D_1, \dots, D_n\}$  — the result of substituting the terms  $D_1, \dots, D_n$  in  $B = B\{x_1, \dots, x_n\}$  for its free variables  $x_1, \dots, x_n$ .

Now, given  $\langle M, \mathcal{M} \rangle$ , consider the set  $\mathcal{H} = \mathcal{H}(M) \equiv \text{Basic-Terms}(M) \cup \mathbf{N}$ .<sup>18</sup> As usual,  $\mathcal{H}^*$  denotes the set of finite strings over the set  $\mathcal{H}$  considered now as consisting of atomic data. These strings can serve as *intermediate configurations* of a computation. Let the *initial configurations* have the form  $u(A\bar{y})$  where  $u \in \mathbf{N}^* \subseteq \mathcal{H}^*$  is a numerical string (the potential Oracle’s answers) and  $\bar{y}$  shown are the only occurrences of variables in  $A\bar{y} : \iota$ . We use parentheses around  $A\bar{y}$  to emphasize that this is a single element of  $\mathcal{H}$ .

Define a computational procedure consisting of a transformation of finite strings in  $\mathcal{H}^*$  by the following rules defining inductively a transformation relation  $\vdash \subseteq \mathcal{H}^* \times \mathcal{H}^*$ . For any  $C, C' \in \text{Basic-Terms}(M)$ ,  $h \in \mathcal{H}^*$ ,  $w \in \mathbf{N}^*$ , and  $v \in \mathbf{N}$  the following transformations (derivations) are allowed:

- (H1)  $hCw \vdash hv$ , if  $C$  is  $w$ -reducible to  $v$ ;
- (H2)  $hCw \vdash hCwC'$ , if  $C$  is  $w$ -reducible to the immediate sub-task  $C'$ ;
- (H3)  $vhC \vdash hv$ , if  $C$  has a head variable, i.e., has the form  $y_j D_1 D_2 \dots D_{n_j}$ <sup>19</sup>;
- (H4) Transitivity: if  $h \vdash h'$  and  $h' \vdash h''$  then  $h \vdash h''$ .

Note, that no two of the rules (H1–H3) are applicable simultaneously to a string in  $\mathcal{H}^*$ . It follows that  $\vdash$  determines a *deterministic* (sequential) computation process. The term  $C$  in the rules (H1), (H2) should be necessarily headed by a strategy, i.e., should have a form  $mD_1 D_2 \dots D_{n_m}$  with  $m \in M$ . A derivation terminating in a string of the form  $hCw$ , with  $C$   $w$ -reducible to  $\perp$ , is called *dead-ended*.

For any initial configuration  $u(A\bar{y}) \in \mathcal{H}^*$ , exactly one of three cases is possible:

- ( $\hat{1}$ )  $u(A\bar{y}) \vdash v$  (with  $u$  completely “exhausted” by using (H3)), where  $v \in \mathbf{N}$ ;
- ( $\hat{2}$ )  $u(A\bar{y}) \vdash (A\bar{y})hC$  (with  $u$  completely “exhausted” by using (H3)), where  $h \in \mathcal{H}^*$  and “sub-sub- $\dots$ -task”  $C \in \text{Basic-Terms}(M)$  is headed by a variable;
- ( $\hat{3}$ ) either there exists an infinite or dead-ended derivation starting with  $u(A\bar{y})$ , or  $u'(A\bar{y}) \vdash v$  holds for some initial segment  $u' \neq u$  of the string  $u$  (i.e. not all prompts from  $u$  are used).

Given any applicative term  $A$  of a type  $\alpha$  without variables consisting of strategies in  $M$ , consider a formal expression of the form  $\langle\langle A \rangle\rangle$  as a new strategy of the same type. Define a new system of strategies  $\langle\hat{M}, \hat{\mathcal{M}}\rangle$  where  $\hat{M}$  is the set of all such formal expressions  $\langle\langle A \rangle\rangle$  and  $\hat{\mathcal{M}}$  is a function making  $\hat{M}$  a system of strategies which is defined below with the help of a “splicing” function  $\delta : \text{Basic-Terms}(M) \rightarrow \text{Basic-Terms}(\hat{M})$ . We set  $\delta(C)$  to be the result of grouping in the term  $C$ , with the aid of  $\langle\langle - \rangle\rangle$ , all the maximal sub-terms not containing variables. For example,

$$\delta(m_1 m_2 y_1 (m_3 (m_4 y_2))) y_3 y_4 = \langle\langle m_1 m_2 \rangle\rangle y_1 (\langle\langle m_3 \rangle\rangle (\langle\langle m_4 \rangle\rangle y_2)) y_3 y_4.$$

<sup>18</sup>Recall that the union is considered here to be disjoint, and  $\text{Basic-Terms}(M)$  may also involve variables.

<sup>19</sup>Here  $v$  is considered as the Oracle’s prompt for the variable-headed task  $C = y_j D_1 D_2 \dots D_{n_j}$ . Thus, query  $C$  is replaced by the the prompt  $v$  which, actually, originates from an element in  $u$  of the initial configuration  $u(A\bar{y})$ . If  $u = u'vu''$  with  $u', u'' \in \mathbf{N}^*$  and  $v \in \mathbf{N}$  then, before applying this rule to the occurrence of  $v$ , the initial segment  $u'$  should have been used analogously as the Oracle’s answers on the previous steps of the computation.

Finally, we define  $\hat{\mathcal{M}}$  by setting, for any  $\langle\langle A \rangle\rangle \in \hat{M}$  and  $u \in \mathbf{N}^*$ ,

$$\hat{\mathcal{M}}(\langle\langle A \rangle\rangle, u) \Rightarrow \begin{cases} v \in \mathbf{N}, & \text{if } (\hat{1}), \\ \delta(C) \in \text{Basic-Terms}(\hat{M}), & \text{if } (\hat{2}), \\ \perp, & \text{if } (\hat{3}). \end{cases}$$

Thus, the system of strategies  $\langle\hat{M}, \hat{\mathcal{M}}\rangle$  is based on the computation process  $(\vdash)$  induced by the strategies of  $\langle M, \mathcal{M} \rangle$ . By (implicit) use of the unique homomorphism from  $\langle\hat{M}, \hat{\mathcal{M}}\rangle$  into the universal system of strategies  $\langle Q, \mathcal{Q} \rangle$ , this gives  $\langle\langle A \rangle\rangle \in Q$  for any applicative term  $A$  over  $Q$  without variables. In particular,  $\langle\langle pq \rangle\rangle \in Q$  for any two strategies  $p, q \in Q$ . For  $A$  of the basic type  $\iota$ , the strategy  $\langle\langle A \rangle\rangle$  computes a constant value  $v \in \mathbf{N}_\perp$  of this type (defined or not). This is also written as  $\langle\langle A \rangle\rangle = v$ .

**Note 6.2.** *For the case of arbitrary type, the resulting strategy  $\langle\langle A \rangle\rangle$  only asks queries headed by a variable (see  $(\hat{2})$  above) and may be slightly redefined in such a way that all these queries will be in the canonical form (5.1) (by the evident use of combinators  $\mathbf{S}$  and  $\mathbf{K}$  and the splicing function  $\delta$ ), even if the strategies participating in  $A$  were not canonical. Alternatively, we could trivially extend  $\langle\langle - \rangle\rangle$  to the case of  $\lambda$ -terms as  $\langle\langle \lambda \bar{x}. A \rangle\rangle$  for  $A$  involving no  $\lambda$  and use these  $\lambda$ -terms to get the canonical form.*

**6.3. Relating Denotational and Operational Semantics of Strategies for the Standard Continuous Model  $\{\mathbb{D}_\alpha\}$ .** The main result of [29] relates the *denotational* and *operational* semantics,  $\llbracket - \rrbracket$  and  $\langle\langle - \rangle\rangle$ , of strategies in the standard dcpo model  $\{\mathbb{D}_\alpha\}^{20}$  of all continuous finite type functionals over the given basic flat domain  $\mathbb{D}_\iota = \mathbf{N}_\perp$ . It consists in the following equality which holds for any typed applicative combination  $A$  of strategies containing no variables:

$$\llbracket \langle\langle A \rangle\rangle \rrbracket = \llbracket A \rrbracket \text{ or, in particular, } \llbracket \langle\langle pq \rangle\rangle \rrbracket = \llbracket p \rrbracket \llbracket q \rrbracket. \quad (6.1)$$

Here the right-hand side of the equality is the ordinary denotational semantics of an applicative term defined by the application operator in the model  $\{\mathbb{D}_\alpha\}$  and by  $\llbracket - \rrbracket$  eventually applied to the strategies comprising  $A$ . We will show in Theorem 6.6 (a) that the same equality holds in the model  $\{\tilde{Q}_\alpha\}$  (and therefore in its isomorphic version  $\{Q_\alpha\}$ ).

The equality (6.1) is essentially based on the *associativity law* for  $\langle\langle - \rangle\rangle$ :

$$\langle\langle \mathcal{A} \rangle\rangle = \langle\langle A \rangle\rangle \text{ or, in particular, } \langle\langle \langle\langle B \rangle\rangle \langle\langle C \rangle\rangle \rangle\rangle = \langle\langle BC \rangle\rangle \quad (6.2)$$

where  $A, B, C$  are any combinations of strategies in  $Q$ , and  $\mathcal{A}$  is obtained from  $A$  by grouping some sub-terms of  $A$  with the help of the operation  $\langle\langle - \rangle\rangle$ . The associativity law allows us to eliminate any nesting of  $\langle\langle - \rangle\rangle$  and can be proved by a thorough analysis of  $\vdash$ -computations defined by strategies  $\langle\langle \mathcal{A} \rangle\rangle$  and  $\langle\langle A \rangle\rangle$ ; cf. [29] for a detailed proof (for the untyped case and for more general non-deterministic strategies).

<sup>20</sup>more precisely, — in an untyped model  $\mathbb{D}_\infty \cong [\mathbb{D}_\infty \rightarrow \mathbb{D}_\infty]$ ; the case of typed model  $\{\mathbb{D}_\alpha\}$  is quite similar and a corresponding result like (6.1) is formulated without proof in [28] (see also Footnote 10)

6.4. **Definition of  $\preceq$ ,  $\simeq$  and  $\tilde{Q}$ .** Having the operational semantics  $\langle\langle - \rangle\rangle$ , we can define a relation  $\preceq_\alpha$  on strategies of the same type  $\alpha$  as follows.

$$q \preceq_\alpha q' \equiv \forall \bar{q}. (\langle\langle q\bar{q} \rangle\rangle \preceq_\iota \langle\langle q'\bar{q} \rangle\rangle) \quad (6.3)$$

where  $p \preceq_\iota p'$  relates the (constant) strategies of basic type  $\iota$  and means that the strategy  $p$  outputs the same basic value as the strategy  $p'$ , if the first value is defined at all. To simplify notation we will often omit the external  $\langle\langle - \rangle\rangle$  in inequalities  $\langle\langle A \rangle\rangle \preceq \langle\langle B \rangle\rangle$  for applicative terms  $A$  and  $B$  writing simply  $A \preceq B$ . Evidently,  $\preceq_\alpha$  is a preorder on the set of strategies  $Q_\alpha$  of the type  $\alpha$ . The corresponding equivalence relation is denoted as  $\simeq_\alpha$ , and the “undefined” strategy  $\Omega_\alpha$  is the  $\preceq$ -least element in each type. Due to the above associativity law, we have  $\langle\langle \langle\langle q \rangle\rangle \bar{q} \rangle\rangle \simeq \langle\langle q\bar{q} \rangle\rangle$  and, hence,  $\langle\langle q \rangle\rangle \simeq q$ . Therefore,

**Proposition 6.3.** *Any strategy  $q$  is  $\simeq$ -equivalent to a strategy in canonical form (see Note 6.2).  $\square$*

**Lemma 6.4.** *Operational semantics is monotonic in the sense that for any applicative term  $A\{q\}$  without variables which involves a strategy  $q$ ,*

$$q \preceq q' \Rightarrow \langle\langle A\{q\} \rangle\rangle \preceq \langle\langle A\{q'\} \rangle\rangle.$$

*Proof.* We can evidently consider that  $A$  has the basic type. Then the proof proceeds by induction on the length  $t$  of the computation  $\langle\langle A\{q\} \rangle\rangle = v \neq \perp$ . Let us write  $A$  for  $A\{q\}$  and  $A'$  for  $A\{q'\}$ , etc. Two cases are possible.

- (1)  $A = sA_1 \cdots A_n$  and  $A' = sA'_1 \cdots A'_n$  for the same head strategy  $s$ . The case if  $s$  is a constant strategy (with the value  $v$ ) is trivial. Otherwise,  $s$  reduces the computation of the value  $v$  of  $A$  to some length  $< t$  sub-computations of the (basic) values  $v_i$  of some sub-tasks  $B_i$ . By the induction hypothesis, corresponding  $B'_i$  evaluate to the same results  $v_i$ . It follows that  $A'$  also evaluates to  $v$  by the strategy  $s$ , as required.
- (2)  $A = qA_1 \cdots A_n$  and  $A' = q'A'_1 \cdots A'_n$  for the above  $q$  and  $q'$ . Then, as it was just proved,  $qA_1 \cdots A_n$  and  $q'A'_1 \cdots A'_n$  evaluate both to  $v$ , and it suffices to note that  $q \preceq q'$  and to use the definition of  $\preceq$  with  $\bar{q} = \langle\langle A' \rangle\rangle$  and associativity of  $\langle\langle - \rangle\rangle$ .  $\square$

The following Lemma (Theorem 6.4 in [29]) corresponds to the *context lemmas* in [21].

**Lemma 6.5.** *Given any types  $\alpha$  and  $\beta$ ,*

$$q \preceq_\alpha q' \iff \forall p : \alpha \rightarrow \beta. (\langle\langle pq \rangle\rangle \preceq_\beta \langle\langle pq' \rangle\rangle).$$

*In particular,*

$$q \preceq_\alpha q' \iff \forall p : \alpha \rightarrow \iota. (\langle\langle pq \rangle\rangle \preceq_\iota \langle\langle pq' \rangle\rangle).$$

*Proof.*

( $\Rightarrow$ ) follows from Lemma 6.4.

( $\Leftarrow$ ) Let us assume (for contraposition) that  $q\bar{q} \simeq v \not\preceq_\alpha q'\bar{q}$ . For any basic value  $c$ , define a strategy  $p$  by

$$px\bar{y} = \text{if } x\bar{q} = v \text{ then } c_\beta\bar{y} \text{ else } \Omega_\beta\bar{y}.$$

Then  $pq \simeq c_\beta \not\preceq_\beta \Omega_\beta \simeq pq'$ , as required.  $\square$

Now, our goal is to show that  $\mathbb{Q}_\alpha$  (cf. Definition 5.2) is isomorphic to the quotient  $\tilde{Q}_\alpha \equiv Q_\alpha / \simeq_\alpha$  where each  $q \in Q_\alpha$  generates the equivalence class  $[q] \in \tilde{Q}_\alpha$  and  $\sqsubseteq_\alpha$  is the partial order on  $\tilde{Q}_\alpha$  induced by  $\preceq_\alpha$ . The natural (typed) application operation in  $\tilde{Q}$  is defined by

$$[p][q] \equiv [\langle\langle pq \rangle\rangle] \quad (6.4)$$

which does not depend on representatives  $p$  and  $q$  of the equivalence classes. So defined structure  $\tilde{Q}$  is *monotonic and order extensional* by Lemma 6.4 and definition (6.3) of  $\preceq$ .

**6.5. Denotational Semantics of Strategies in  $\tilde{Q}$  and the isomorphism  $\tilde{Q} \cong \mathbb{Q}$ .** Let us consider  $[-]$  as the denotational semantics of  $Q$  in  $\tilde{Q}$ .

**Theorem 6.6.**

- (a) *Denotational semantics  $s \mapsto [s]$  of the universal system of strategies  $Q$  in  $\tilde{Q}$  is coherent with the operational one<sup>21</sup>:  $[\langle\langle A \rangle\rangle] = [A]$ .*
- (b)  *$\tilde{Q}$  is sequentially complete (in particular, satisfying the **Y**-property (2.5)) with  $[-]$  the least correct denotational semantics which is also naturally defined.*

*Proof.*

- (a) Apply associativity of  $\langle\langle - \rangle\rangle$  and the definition (6.4) of application in  $\tilde{Q}$ . For example,  $[\langle\langle p(qr) \rangle\rangle] = [\langle\langle p \langle\langle qr \rangle\rangle \rangle] = [p][\langle\langle qr \rangle\rangle] = [p]([q][r]) \equiv [p(qr)]$ .
- (b) First, show correctness of  $[-]$ . Consider the interpreted computation by a strategy  $q \in Q$  associated with the task “ $qx_1 \cdots x_n = ?$ ” of the basic type with some fixed values  $[q_i]$  in  $\tilde{Q}$  for the arguments  $x_i$  (and  $q_i \in Q$ ). We should assume that  $q$  receives correct replies to its queries “ $A\{x_1, \dots, x_n\} = ?$ ” where  $A$  is a combination of strategies  $s \in Q$  and the variables  $x_i$ . According to the assignment  $s \mapsto [s]$  and (a), the correct replies are obtained just by replacing all strategies  $s$  in  $A$  by  $[s]$  or, equivalently, by replacing  $A\{x_1, \dots, x_n\}$  by  $[A\{q_1, \dots, q_n\}] = [\langle\langle A\{q_1, \dots, q_n\} \rangle\rangle]$ . Then we must show that the resulting basic value  $v$  (possibly  $= \perp$ ) of the interpreted computation coincides with the value of the combination  $[q][q_1] \cdots [q_n] = [qq_1 \cdots q_n] = [\langle\langle qq_1 \cdots q_n \rangle\rangle]$ . However, the latter value is obtained by  $\vdash$ -computation, i.e. by essentially the same interpreted computation as above plus  $\vdash$ -sub-computations of the values  $[A\{q_1, \dots, q_n\}] = [\langle\langle A\{q_1, \dots, q_n\} \rangle\rangle]$  for all queries. The required correctness follows.

Let us show that  $\tilde{Q}$  is sequentially complete. First, we present a general consideration on the “approximating” semantics  $\llbracket - \rrbracket^k$  in any monotonic and order extensional structure  $\mathcal{E}$ . Given any system of strategies  $\langle M, \mathcal{M} \rangle$ , define its “approximating” version  $\langle M^A, \mathcal{M}^A \rangle$  by letting

$$M^A = \{m^k \mid m \in M \ \& \ k \in \mathbf{N}\},$$

$$\mathcal{M}^A(m^0, w) \equiv \perp,$$

$$\mathcal{M}^A(m^k, w) \equiv (\mathcal{M}(m, w))^{k-1},$$

where  $m^k$  is considered as a formal expression (a pair of  $m$  and  $k$ ),  $v^k = v$  for  $v \in \mathbf{N}_\perp$ ,  $(AB)^k = A^k B^k$  for applicative terms, and  $x^k = x$  for variables. For any structure  $\mathcal{E}$ , if a computationally correct  $\llbracket - \rrbracket^A : M^A \rightarrow \mathcal{E}$  exists then all  $\llbracket - \rrbracket^k : M \rightarrow \mathcal{E}$ ,  $k \in \mathbf{N}$ , exist too and  $\llbracket m^k \rrbracket^A = \llbracket m \rrbracket^k$  holds for all  $m \in M$ , and vice versa. In particular,  $\llbracket - \rrbracket^A$  is uniquely defined, if exists at all (iff all  $\llbracket - \rrbracket^k : M \rightarrow \mathcal{E}$ ,  $k \in \mathbf{N}$ , exist).

<sup>21</sup>Compare this with the equation (6.1) for the case of  $\{\mathbb{D}_\alpha\}$ .

Now, let  $\mathcal{E} = \tilde{Q}$ , and  $\varphi : M \rightarrow Q$  and  $\varphi^A : M^A \rightarrow Q$  be the unique homomorphisms. Then both  $\llbracket - \rrbracket \cong [\varphi(-)]$  and  $\llbracket - \rrbracket^A \cong [\varphi^A(-)]$  are computationally correct semantics of  $M$  and  $M^A$  in  $\tilde{Q}$  by the correctness of  $[-]$  and Proposition 6.1 (c). It follows from the latter that all  $\llbracket - \rrbracket^k : M \rightarrow \tilde{Q}$  exist, and, for sequential completeness of  $\tilde{Q}$ , it remains to show that  $\llbracket m \rrbracket = \bigoplus_k \llbracket m \rrbracket^k = \bigoplus_k \llbracket m^k \rrbracket^A$ , that is  $[\varphi(m)] = \bigoplus_k [\varphi^A(m^k)]$ , or equivalently, that for all strategies  $\bar{q}$  of appropriate types  $\varphi(m)\bar{q} \simeq_\iota \varphi^A(m^k)\bar{q}$  holds for some  $k$ . But the latter holds because, in each  $\vdash$ -computation giving a defined result in  $\mathbf{N}$ ,  $m^k$  behaves as  $m$  for sufficiently large  $k$  and gives the same result.

It follows that  $\llbracket - \rrbracket$  and therefore its special case  $[-]$  are naturally defined and computationally correct and hence (by Proposition 4.3 (a)) both are the least correct semantics of  $M$  and  $Q$ , respectively, in  $\tilde{Q}$ .  $\square$

**Corollary 6.7.**  $\{\mathbb{Q}_\alpha\} \cong \{\tilde{\mathbb{Q}}_\alpha\}$ .

*Proof.* Use Proposition 5.3.  $\square$

## 7. MAIN RESULTS ON FULL ABSTRACTION AND DOMAIN THEORETIC PROPERTIES OF $\mathbb{Q}$

### 7.1. Full Abstraction, Universality and PCF-Definability.

**Theorem 7.1.**  $\mathbb{Q} \cong \tilde{\mathbb{Q}}$  is fully abstract model of  $\mathbf{PCF}$ . The same holds for  $\mathbf{PCF}^-$  ( $\mathbf{PCF}$  with  $\mathbf{Y}$  omitted).

*Proof.* Assume  $q, q' \in Q_\alpha$  and  $Cq \preceq_\iota Cq'$  holds for all  $\mathbf{PCF}$  combinations  $C : \alpha \rightarrow \iota$ . Then, in particular,  $q\bar{c} \preceq_\iota q'\bar{c}$  for all  $\mathbf{PCF}^-$  definable terms  $\bar{c}$  of appropriate types. Let us infer  $q \preceq q'$ , or equivalently that  $q\bar{q} \preceq_\iota q'\bar{q}$  holds for all strategies  $\bar{q}$  of appropriate types. Indeed, according to Section 7.2.2 below, if  $q\bar{q} \vdash v$  for some  $v \in \mathbf{N}$  then  $q\bar{c} \vdash v$  holds also for some *finite* (and even *finitary ranked*) and therefore definable in  $\mathbf{PCF}^-$  strategies  $\bar{c} \preceq \bar{q}$  (see Lemma 7.12 (a) and Theorem 7.13 (b) below). It follows  $q'\bar{c} \vdash v$  and  $q'\bar{q} \vdash v$ , as required.  $\square$

As in [28] (the case of  $\{\mathbb{D}_\alpha\}$ ), [1, 11] and also [18] (the effective case), we have

**Theorem 7.2.** For any type  $\alpha$  there exists a  $\mathbf{PCF}$ -definable functional

$$U_\alpha \in \mathbb{Q}_{(\iota \rightarrow \iota) \rightarrow \alpha}$$

which is universal in the sense that its range is the whole set  $\mathbb{Q}_\alpha$  of sequential functionals. Moreover, there exists  $\mathbf{PCF}$ -definable  $U_\alpha^{\text{eff}} \in \mathbb{Q}_{\iota \rightarrow \alpha}$  which enumerates all elements of  $\mathbb{Q}$  definable by computable strategies (i.e. those in systems  $\langle M, \mathcal{M} \rangle$  with computable  $M$ ).

In particular,  $\mathbf{PCF}$  exactly grasps sequential computability over  $\mathbb{Q}$ , that is,  $\mathbf{PCF}$  definable = sequentially computable.

*Proof.* As in [28] for the case of  $\{\mathbb{D}_\alpha\}$ . It is omitted here, but see the proof in Section 8 of analogous result for  $\{\mathbb{W}_\alpha\}$  and  $\mathbf{PCF}^+$ .  $\square$

**Theorem 7.3** (Normann [23]). The (unique up to isomorphism) directly complete and continuous fully abstract model  $\{\hat{\mathbb{Q}}_\alpha\}$  for  $\mathbf{PCF}$  defined by Milner [21]<sup>22</sup> cannot be exhausted by sequentially computable functionals in  $\mathbb{Q}_\alpha \hookrightarrow \hat{\mathbb{Q}}_\alpha$ , i.e. by those definable in  $\mathbf{PCF} +$  all monotonic  $f : \mathbf{N}_\perp \rightarrow \mathbf{N}_\perp$ .  $\square$

<sup>22</sup>which is, more precisely, isomorphic to the limit (ideal) completion  $\hat{\mathbb{Q}}_\alpha$  of  $\mathbb{Q}_\alpha$ ; cf. Section 2.5.

More precisely, the proof in [23] shows that  $\mathbb{Q}_\alpha$  is *not* an  $\omega$ -complete domain for some  $\alpha$  of level 3. We know from Theorem 6.6 (b) that  $\mathbb{Q}$  is only sequentially complete.

**7.2. Deriving Domain Theoretic Properties of  $\mathbb{Q}$ .** We need to use Lemma 2.11, and this requires to work out appropriate versions of “finite” approximations of strategies.

7.2.1. *Finite, Finitely Restricted and Finitary Sequential Strategies.*

**Definition 7.4.** We say that a system of strategies  $\langle M', \mathcal{M}' \rangle$  is a *restriction*, or *subsystem* (or *approximation*<sup>23</sup>) of another system  $\langle M, \mathcal{M} \rangle$  if  $M' \subseteq M$ , and, as partial functions,  $\mathcal{M}' \subseteq \mathcal{M}$ .<sup>24</sup> A restriction  $\langle M', \mathcal{M}' \rangle$  is called *finite* if both the set  $M'$  and the function  $\mathcal{M}'$  are finite. Strategies (if any) from finite restrictions of  $\langle M, \mathcal{M} \rangle$  are called *finite*.

If  $\langle M', \mathcal{M}' \rangle$  is a restriction of  $\langle M, \mathcal{M} \rangle$  then the (unique) homomorphic image  $q'$  in  $Q$  of any  $m$  in  $\langle M', \mathcal{M}' \rangle$  is called a *restriction*, or *sub-strategy* (or *approximation*) of the homomorphic image  $q$  of the same  $m$  considered as a strategy of  $\langle M, \mathcal{M} \rangle$ . By abusing notation, we write  $q' \subseteq q$ .<sup>25</sup> Then, evidently,  $[q'] \sqsubseteq [q]$  (i.e.  $q' \preceq q$ ), holds in  $\tilde{Q}$ .

Let us introduce a more general concept than a finite strategy.

**Definition 7.5.** Given any system of strategies  $\langle M, \mathcal{M} \rangle$ , let  $\mathcal{M}^{[k]}(m, w)$  be defined and equal to  $\mathcal{M}(m, w)$  if, and only if, (i) the string  $w$  consists only of numbers  $\leq k$  and (ii)  $\mathcal{M}(m, w) \leq k$  in the case of  $\mathcal{M}(m, w) \in \mathbf{N}$ . The system  $\langle M, \mathcal{M}^{[k]} \rangle$  is called *k-restriction* of  $\langle M, \mathcal{M} \rangle$ . If, in fact,  $\mathcal{M}^{[k]} = \mathcal{M}$  then the original system is called *k-restricted*. Then *finitely restricted* means *k-restricted* for some  $k$ . A strategy  $q \in Q$  is called *k-restricted* if it is contained in the homomorphic image of some *k-restricted* system of strategies.

Evidently,  $\mathcal{M} = \bigcup_k \mathcal{M}^{[k]}$ , and also any finite  $\langle M, \mathcal{M} \rangle$  is finitely restricted (but not vice versa). *k-restricted* strategies “understand” only basic values  $\leq k$ , as if it was our basic domain  $\mathbf{N}_\perp$  so restricted to  $\{0, 1, \dots, k\}_\perp$ . Strategies from the original and restricted versions of a system of strategies, although formally having the same names, behave differently. Therefore, to emphasize that a restricted version is assumed, we will write  $m^{[k]}$  instead of  $m$  and  $\langle M^{[k]}, \mathcal{M}^{[k]} \rangle$  instead of  $\langle M, \mathcal{M}^{[k]} \rangle$ , whereas  $m$  will typically be considered as a strategy of the non-restricted system  $\langle M, \mathcal{M} \rangle$ .<sup>26</sup> In the following Lemma we identify strategies with their homomorphic images in  $Q$  and relate *k-restriction* with the projection maps  $\Psi^{[k]}$  defined in Section 2.2.

**Lemma 7.6.** *Functionals  $[m^{[k]}]$  in  $\tilde{Q}$  defined by finitely (*k*-)restricted strategies are also finitely (*k*-)restricted (as defined in Section 2.2).*

*Proof.* Consider projection functionals  $\Psi^{[k]}$ ,  $k = 0, 1, \dots$  and computing them sequential strategies  $\psi^{[k]}$ . Their behaviour can be described by the equality (in the basic type  $\iota$ , assuming  $f : \alpha$  and  $\psi^{[k]} : \alpha \rightarrow \alpha$ )

$$\psi^{[k]} f \bar{x} = f \bar{\psi}^{[k]}(\bar{x}), \text{ if the result is bounded by } k, \text{ and } = \perp \text{ otherwise.}$$

<sup>23</sup>but in a different sense than considered above system  $\langle M^A, \mathcal{M}^A \rangle$

<sup>24</sup>Note, that the embedding  $M' \hookrightarrow M$  is an approximating homomorphism; cf. Footnote 16.

<sup>25</sup>By considering the explicit construction of  $Q$  (cf. [30] or Appendix A), the relation  $\subseteq$  on  $Q$  may be treated, indeed, as set inclusion between strategies considered as graphs of partial functions of a special kind and is therefore a partial order. However, we will not need this fact.

<sup>26</sup>Note that, although there is a kind of analogy between the strategies  $m^{[k]}$  considered here, and  $m^k$  considered in the proof of Theorem 6.6, the behaviour of these strategies is different.



Here  $\bar{\psi}^{[k]}(\bar{x})$  means the application of  $\psi^{[k]}$  (of appropriate type) to each  $x_i$  in  $\bar{x}$ . Let us show that  $m^{[k]} \simeq \langle\langle \psi^{[k]}m \rangle\rangle$ . The task “ $\psi^{[k]}m\bar{x} = ?$ ” is reducible to “ $m\bar{\psi}^{[k]}(\bar{x}) = ?$ ”. By assuming that  $m$  asks queries in canonical form “ $x_i(m_1\bar{x}) \cdots (m_n\bar{x}) = ?$ ”, the task “ $m\bar{\psi}^{[k]}(\bar{x}) = ?$ ” is further reducible by  $m$  to the sub-task

$$“(\psi^{[k]}x_i)(m_1\bar{\psi}^{[k]}(\bar{x})) \cdots (m_n\bar{\psi}^{[k]}(\bar{x})) = ?”,$$

and then by  $\psi^{[k]}$  to

$$“x_i\psi^{[k]}(m_1\bar{\psi}^{[k]}(\bar{x})) \cdots \psi^{[k]}(m_n\bar{\psi}^{[k]}(\bar{x})) = ?^{[k]}”$$

with the head variable  $x_i$ , where  $?^{[k]}$  assumes that only answers  $\leq k$  will be taken into account. As  $m_1\bar{\psi}^{[k]}(\bar{x})$  is  $\vdash$ -computationally equivalent to  $(\psi^{[k]}m_1)\bar{x}$ <sup>27</sup>, the latter query is equivalent to

$$“x_i((\psi^{[k]}m_1)\bar{x}) \cdots ((\psi^{[k]}m_n)\bar{x}) = ?^{[k]}”.$$

All of this means that  $\psi^{[k]}m$  behaves computationally as  $m^{[k]}$  which asks similar queries “ $x_i(m_1^{[k]}\bar{x}) \cdots (m_n^{[k]}\bar{x}) = ?^{[k]}$ ” and reacts to the answers in the same way as  $m$  and  $\psi^{[k]}m$ , except considering the integer values bigger than  $k$  as if they were undefined. It follows that  $m^{[k]} \simeq \langle\langle \psi^{[k]}m \rangle\rangle$ , as required. Moreover,  $[m^{[k]}] = [\psi^{[k]}][m] = \Psi^{[k]}[m]$ . If the original system is  $k$ -restricted then  $m \simeq m^{[k]}$ , and therefore the functional  $[m] = \Psi^{[k]}[m]$  is  $k$ -restricted in  $\bar{Q}$ .  $\square$

**Definition 7.7.** A system of sequential strategies  $\langle M, \mathcal{M} \rangle$  is called *ranked* if (ignoring types)  $M$  is a disjoint union  $\bigcup_{i \in \mathbf{N}} M_i$  such that any strategy in  $M_i$  can ask queries only concerning the strategies in  $M_{i+1}$ .

We have actually considered a similarly ranked systems in the proof of Theorem 6.6 (b) but with the *inverse* ranking order. Our choice of the *ranking order* as in Definition 7.7 is based on the following Lemma. Independently of the choice of this order, ranked systems of strategies evidently remain ranked under restriction.

**Lemma 7.8.** *Any system of strategies  $\langle M, \mathcal{M} \rangle$  is homomorphic image of a ranked system.*

*Proof.* Indeed,  $\langle M, \mathcal{M} \rangle$  is homomorphic image of a ranked system  $\langle M \times \mathbf{N}, \mathcal{M}' \rangle$  with  $\mathcal{M}'$  defined for all  $m, w, n$  as

$$\mathcal{M}'(\langle m, n \rangle, w) \equiv \text{sub}_{n+1}(\mathcal{M}(m, w))$$

where  $\text{sub}_k A$  is obtained from  $A$ , for  $A$  any term, by replacing each occurrence of a strategy  $m'$  in  $A$  by  $\langle m', k \rangle$ , and  $\text{sub}_k v = v$  for any resulting basic output value  $v$ . The required homomorphism is  $\pi : \langle m, n \rangle \mapsto m$ .  $\square$

Moreover, if  $\varphi : \langle M_1, \mathcal{M}_1 \rangle \rightarrow \langle M_2, \mathcal{M}_2 \rangle$  is a homomorphism then  $\varphi^{\text{R}}(\langle m_1, n \rangle) \equiv \langle \varphi(m_1), n \rangle$  is also a homomorphism of corresponding ranked systems

$$\varphi^{\text{R}} : \langle M_1 \times \mathbf{N}, \mathcal{M}'_1 \rangle \rightarrow \langle M_2 \times \mathbf{N}, \mathcal{M}'_2 \rangle,$$

and the resulting square diagram commutes:  $\varphi \circ \pi = \pi \circ \varphi^{\text{R}}$ .

**Definition 7.9.** Strategies from ranked systems of strategies  $\langle M, \mathcal{M} \rangle$  with both  $M$  and  $\mathcal{M}$  finite are called *finitary*. (That is, essentially, finitary = finite & ranked, also = finite well-founded.) Equivalently, only  $\mathcal{M}$  may be required to be finite.

<sup>27</sup>if to replace the variables  $\bar{x}$  by arbitrary strategies  $\bar{q}$  of the same types

**Lemma 7.10.** *Finitary strategies are special case of finite strategies which, in turn, are special cases of finitely restricted ones and therefore define (in fact all; see Theorem 7.13) finitely restricted functionals in  $\tilde{Q} \cong \mathbb{Q}$ .*  $\square$

If  $M$  is finite and  $M = \bigcup_i M_i$  is the ranking then all  $M_i$  are empty for  $i$  large enough. In a reasonable sense finitary strategies are considered as *non-recursive*. Homomorphic images in  $Q$  of finitely restricted (resp., finitary) strategies can also be unofficially called *finitely restricted (resp., finitary)* ones. Any (finitary) strategy  $m_k \in M_k$  from a finite ranked system  $\langle M, \mathcal{M} \rangle$  with the ranking  $M = \bigcup_{i \in \mathbf{N}} M_i$  has a finite *rank* which is the length  $r$  of a maximal chain  $m_k, \dots, m_{k+r}$  of strategies (in  $M_k, \dots, M_{k+r}$ , respectively) starting with given  $m_k$  such that each  $m_{k+i}$ ,  $0 \leq i < r$ , asks a query on  $m_{k+i+1}$  (i.e.  $m_{i+1}$  is a child of  $m_i$ ). Now, König's Lemma entails more general

**Proposition 7.11.** *All strategies in  $\langle M, \mathcal{M} \rangle$  are finitary<sup>28</sup> iff for each  $m \in M$  there is only a finite number of computational histories  $w \in \mathbf{N}^*$  such that  $\mathcal{M}(m, w)$  is defined and there are no infinite chains  $m = m_0, m_1, m_2, \dots$  where  $m_i$  asks a query on  $m_{i+1}$  (i.e.  $\langle M, \mathcal{M} \rangle$  is well-founded).*

*Proof.* “Only if” case is trivial. For “if” case assume its condition, and let

$$M_{\geq r} \equiv \{m \in M \mid \exists m_0 = m, m_1, \dots, m_r \in M \forall i < r (m_{i+1} \text{ is a child of } m_i)\}.$$

Then  $M_r \equiv M_{\geq r} \setminus M_{\geq r+1}$  is an *inverse ranking* of  $\langle M, \mathcal{M} \rangle$  (in the evident sense dual to Definition 7.9). By König's Lemma, each  $m$  has only a finite set  $M^{[m]} \subseteq M$  of improper descendants (including  $m$  itself) which, if intersected with each  $M_r$ , gives a finite (inverse) ranked subsystem of  $\langle M, \mathcal{M} \rangle$ , as is essentially required.  $\square$

The finitary strategies of rank 0 are either constant strategies of any type (asking no queries to the Oracle) or strategies which can ask in each of finitely many possible ways of computation only (finitely many) queries which are applicative terms consisting of variables only. The finitary strategies of rank 1 are defined analogously, except that they can ask queries involving, besides variables, only strategies of rank 0. Etc., for finitary strategies of any rank.

But we need to be careful with such verbal descriptions. For example, the functional  $F(f) = \mathbf{if } f(0) = 0 \mathbf{ then } 0 \mathbf{ else } 1$  (and  $F(f) = \perp$  if  $f(0) = \perp$ ) computable by the evident rank 1 strategy is *not finitary* because, in its computation, the query  $f(0)$  can have *any* answer  $\neq 0$  leading to the definite result 1. In fact,  $\mathcal{M}$  describing the evident strategy computing functional  $F$  has an infinite domain.

7.2.2. *Observation on Computations and Finitary Strategies.* It follows from Lemma 7.8 that in computations only countable ranked systems of strategies  $\langle M, \mathcal{M} \rangle$  matter.

**Lemma 7.12.**

- (a) *For any combination of strategies  $A : \iota$  over  $\langle M, \mathcal{M} \rangle$ , if  $A \vdash_{\mathcal{M}} v$  then also  $A \vdash_{\mathcal{M}'} v$  over a finite restriction  $\langle M', \mathcal{M}' \rangle$  of  $\langle M, \mathcal{M} \rangle$ .*
- (b) *For any countable system of strategies,  $\langle M, \mathcal{M} \rangle = \bigcup_k \langle M^{(k)}, \mathcal{M}^{(k)} \rangle$  holds for some monotonic by set inclusion sequence of finite restrictions of  $\langle M, \mathcal{M} \rangle$ .*

<sup>28</sup>each in an appropriate finite ranked subsystem of  $\langle M, \mathcal{M} \rangle$

- (c) For any system represented as a monotonic union  $\langle M, \mathcal{M} \rangle = \bigcup_k \langle M^{(k)}, \mathcal{M}^{(k)} \rangle$  of some restrictions, any resulting computation  $A \vdash v$  over  $\langle M, \mathcal{M} \rangle$  is, in fact, a computation over some  $\langle M^{(k)}, \mathcal{M}^{(k)} \rangle$ , or equivalently over some  $\langle M, \mathcal{M}^{(k)} \rangle$ .
- (d) Let the strategy  $m^{(k)}$  be just  $m \in M$  considered as a strategy of  $\langle M, \mathcal{M}^{(k)} \rangle$  with  $\mathcal{M}^{(k)}$  as in (c).<sup>29</sup> By identifying these strategies with their homomorphic images in  $Q$ , this gives rise to the  $\sqsubseteq$ -increasing sequence  $[m^{(k)}]$  with the natural lub  $[m] = \bigoplus_k [m^{(k)}]$ .
- (e) In particular, any functional in  $\tilde{Q}$  is the natural lub of an increasing sequence of finitary presented functionals (and the same for any of the version of “finite” considered in Section 7.2.1).

*Proof.*

- (a) Let  $M'$  consist only of those finitely many strategies in  $M$  which participate in the original derivation  $A \vdash_{\mathcal{M}} v$  and (the finite)  $\mathcal{M}'(m, w)$  be defined if, and only if,  $m$  and the computational history  $w$  for  $m$  was really used in the derivation  $A \vdash_{\mathcal{M}} v$ .
- (b) Let  $M = \bigcup_k M_k$  with  $M_k$  any increasing sequence of finite subsets exhausting  $M$ . Let  $\mathbf{N}_k \doteq \{0, 1, \dots, k\}$  and  $\mathcal{M}^{(k)} \doteq \mathcal{M} \upharpoonright (M_k \times \mathbf{N}_k^{\leq k})$ , and define  $M^{(k)}$  to consist of all strategies participating in the domain and range of  $\mathcal{M}^{(k)}$ .
- (c) Like in (a), construct finite  $\langle M', \mathcal{M}' \rangle$  and embed it in appropriate  $\langle M^{(k)}, \mathcal{M}^{(k)} \rangle$ .
- (d) Use (c) with the equation (2.3) defining the natural lub as the ordinary pointwise defined lub  $\bigsqcup$  in the basic type by using an appropriate list of arguments.
- (e) Use Lemma 7.8 and (d) with  $\mathcal{M}^{(k)}$  as in (b). □

**Theorem 7.13.**

- (a) The model of sequential functionals  $\mathbb{Q} \cong \tilde{Q}$  is naturally continuous, naturally  $\omega$ -algebraic and naturally finitely bounded complete. Naturally finite elements of each  $\mathbb{Q}_\alpha$  are exactly finitely restricted ones (in the sense of Definition 2.10) or, equivalently, definable by finitary strategies or, equivalently, by finite strategies or, equivalently, by finitely restricted strategies.<sup>30</sup>
- (b) Naturally finite elements of  $\mathbb{Q} \cong \tilde{Q}$  are definable in **PCF** (even without using **Y**).

*Proof.*

- (a) follows from Lemma 2.11 whose condition (\*) is satisfied because of the above observations and Lemmas 7.6, 7.8, 7.10, and, most important, 7.12 (a). Also recall that the naturally finite natural lub of an increasing sequence in  $\mathbb{Q}$  must stabilize.
- (b) Use straightforward induction on the rank of finitary sequential strategies. Alternatively, apply the general Theorem 7.2 concerning definability in **PCF** (having much more involved proof). □

**Note 7.14.** It follows from the definition of naturally finite elements in  $\mathbb{Q} \cong \tilde{Q}$  that any finitely restricted or finite (possibly recursive) strategy is  $\simeq$  to some finitary (ranked, non-recursive) strategy, by representing the former as the natural (in fact, stabilizing up to  $\simeq$ ) lub of finitary strategies. But this proof is non-constructive, and by appropriate adaptation

<sup>29</sup> $m^{(k)}$  may be finite, or even finitary in the case of (b) and ranked  $\langle M, \mathcal{M} \rangle$ , or finitely restricted in the case  $\mathcal{M}^{(k)} = \mathcal{M}^{[k]}$  from Definition 7.5 with  $m^{(k)}$  denoted there as  $m^{[k]}$ .

<sup>30</sup>In [29], special non-deterministic (non-sequential) strategies  $\xi_a$  played the role analogous to that of sequential finitely restricted/finitary strategies considered here to define finite elements in  $\mathbb{D}_\infty$  (or in  $\{\mathbb{D}_\alpha\}$  in the typed case), and  $\mathbb{D}_\infty$  was also represented as a quotient of a universal system of (consistent) non-deterministic strategies.

of the technique of Loader [16] it should be possible to show that there is no corresponding “canonization” algorithm  $finite \mapsto finitary$  as there is no way to determine the moment of stabilization in the above lub. Also the related problem “ $p \simeq q$ ?” even for finitary (ranked) strategies should be undecidable.

Note also that Theorem 7.13 (b) and Lemma 7.12 (a) were actually used in the proof of Theorem 7.1 that the model  $\mathbb{Q} \cong \tilde{\mathbb{Q}}$  is fully abstract for **PCF** which was incomplete till this moment.

We conclude this section by proving that *the class of finitary strategies is effectively closed under taking applications*. This was actually used in Section 2.4 in representation of naturally finite functionals in  $\mathbb{Q}$  by finitary strategies (and, similarly, for  $\mathbb{W}$ ).

**Note 7.15.** *On the other hand, the closure of finitely ( $k$ -) restricted strategies under application is trivial. But, unlike the finitary strategies, they are not necessarily finite (and can be recursive). Also, arbitrary finite strategies are probably not closed under application (note that ranking is essentially used in the proof of the following theorem), however evidently giving rise to finitely restricted strategies.*

**Theorem 7.16.** *For any applicative term  $A$  consisting of finitary strategies, the strategy  $\langle\langle A \rangle\rangle$  is finitary, too, and (as a finite object understood in the evident sense) can be effectively computed from  $A$  and comprising its strategies.*

*Proof.* Let us slightly generalize the concept of the initial configuration  $uA\bar{y}$  from Section 6.2 (where  $u \in \mathbf{N}^*$  and  $\bar{y}$  is a list of variables making the term  $A\bar{y}$  be of the basic type  $\iota$ ) by allowing the term  $A$  to contain any variables. The statement which we will actually prove is a kind of *normalization (termination) property*: for each applicative term  $A$  involving only finitary strategies and any variables

(\*) *for any list of variables  $\bar{y}$  making  $A\bar{y}$  a term of the basic type there exists only a finite number of finite non-dead-ended computations<sup>31</sup> (sequences of derivation steps) starting from  $uA\bar{y} \vdash \dots$  for various  $u \in \mathbf{N}^*$  obtained by the rules ( $\mathcal{H}1$ – $\mathcal{H}3$ ) with  $u$  completely “exhausted”<sup>32</sup>.*

Then appropriate application of König’s Lemma will imply that  $\langle\langle A \rangle\rangle$  is indeed finitary and computable from  $A$ .

Following Tait [33] and the presentation by Barendregt [3] of the normalizability proof for typed calculi, (\*) can be shown for any  $A$  as follows.<sup>33</sup> Define classes of typed terms consisting of finitary strategies and variables:

$$\begin{aligned} \mathcal{C}_\iota &= \{A : \iota \mid A \text{ satisfies } (*)\}, \\ \mathcal{C}_{\alpha \rightarrow \beta} &= \{A : \alpha \rightarrow \beta \mid \forall B \in \mathcal{C}_\alpha (AB \in \mathcal{C}_\beta)\}, \\ \mathcal{C} &= \bigcup_{\sigma} \mathcal{C}_\sigma. \end{aligned}$$

<sup>31</sup>This requirement also means that for each numerical answer (either computed or taken from  $u$ ) to a strategy question during such a computation the strategy should be able to react in a definite way giving either a result in  $\mathbf{N}$ , as in the case of ( $\mathcal{H}1$ ), or a new query, as in ( $\mathcal{H}2$ ). If dead-ended computations would be allowed then we might have an infinite number of them for  $u \in \mathbf{N}^*$  with large values in  $\mathbf{N}$ . Indeed, only finitely many strategies—all being finitary descendants of those occurring in  $A$ —can participate in such computations, and they “do not understand” large numerical values.

<sup>32</sup>Exhaustion is necessary, otherwise infinitely many  $u$  of unbounded length would be admitted.

<sup>33</sup>We give the detailed proof to show the specifics of the concept of strategies.

Evidently,

$$A \in \mathcal{C} \iff \forall \bar{C} \in \mathcal{C}(A\bar{C} : \iota \Rightarrow A\bar{C} \text{ satisfies } (*)),$$

and  $\mathcal{C}$  is closed under taking applications of terms. Any variable satisfies (\*) and belongs to  $\mathcal{C}$ . Also any finitary strategy trivially satisfies (\*). It belongs to  $\mathcal{C}$  if its rank is 0, i.e. it is either a constant (defined or undefined) strategy or a strategy whose all possible (basic type) queries involve only variables. This is because  $\mathcal{C}$  is closed under applications, and therefore  $\mathcal{C}$ -substitution cases of such queries satisfy (\*). (That, in fact, all finitary strategies belong to  $\mathcal{C}$  can be concluded from the following considerations.)

Then we show by induction on the type of  $A$  that

$$A \in \mathcal{C} \Rightarrow A \text{ satisfies } (*). \tag{7.1}$$

Indeed, the base case  $A : \iota$  holds by definition. For  $A \in \mathcal{C}_{\alpha \rightarrow \beta}$  and any variable  $y_1 : \alpha$  we have  $y_1 \in \mathcal{C}_\alpha$ ,  $Ay_1 \in \mathcal{C}_\beta$ , and hence  $Ay_1$  satisfies (\*) by induction hypothesis. Then it follows straightforwardly that  $A$  itself satisfies (\*).

Finally, we show by induction on  $k$  that for any term  $A$  whose participating strategies have rank  $\leq k$

$$\text{any } \mathcal{C}\text{-substitution case of } A \text{ belongs to } \mathcal{C}. \tag{7.2}$$

The case  $k = 0$ : That (7.2) holds for atomic terms (variables and rank 0 strategies) was, in fact, shown above. The rest follows from the closure of  $\mathcal{C}$  and therefore of the class of  $A$  satisfying (7.2) under applications. For  $k > 0$  it again suffices to show (7.2) for atomic terms. The main case is finitary strategies  $m$  of rank  $k$  for which we should show that  $m\bar{C} \in \mathcal{C}$ . We need to show that  $m\bar{C} : \iota$  satisfies (\*) for any  $\bar{C} \in \mathcal{C}$  of appropriate types. But this follows from the fact that  $m\bar{y}$  asks a bounded number of queries  $B_i\{\bar{y}\} : \iota$ ,  $i < N$ , involving only variables  $\bar{y}$  and strategies of the rank  $< k$  and which therefore satisfy (7.2) by induction hypothesis, and hence  $B_i\{\bar{C}\} \in \mathcal{C}$  so that all such  $B_i\{\bar{C}\}$  satisfy (\*). Finally, this implies that  $m\bar{C}$  satisfies (\*). Indeed, from our requirements on the computations  $um\bar{C} \vdash \dots$  each value in  $u$  should be used either by  $m$  or by (its child strategies from) the subcomputations generated by  $B_i\{\bar{C}\}$ . Thus,  $u$  should have bounded both the length and participating numerical values. This concludes the proof.  $\square$

## 8. FULLY ABSTRACT MODEL FOR **PCF**<sup>+</sup>

For the case of **PCF**<sup>+</sup>, let us consider the more general concept of a *nondeterministic system of strategies* [29] extending the Definition 3.1 of sequential (deterministic) strategies by letting

$$\mathcal{M} : M \times \mathbf{N}^* \rightarrow \text{Basic-Terms}(M) \cup \mathbf{N} \cup \{\#\},$$

and adding the clause (third possibility for  $\mathcal{M}$ )

$$(3) \mathcal{M}(m, w) = \# \text{ (the nondeterministic state of computation).}$$

The nondeterministic state can be also considered as representing a specific query “ $\# = ?$ ”. The “correct” answer from the Oracle to this query is *any* numerical value  $r \in \mathbf{N}$ . However, such an extended concept of nondeterministic strategies is too general to grasp **PCF**<sup>+</sup> (unlike **PCF**<sup>++</sup> — the case which we will not consider in full detail). Thus, we need to appropriately restrict nondeterministic strategies to fit them with **PCF**<sup>+</sup>.

**8.1. Wittingly Consistent Strategies.** First, without restricting generality we can assume that the requirements from Section 3.1.3 hold also for non-deterministic systems of strategies. Further, a pair of prompts (computational histories)  $w = r_1 \cdots r_k$  and  $u = s_1 \cdots s_n \in \mathbf{N}^*$  for a strategy  $m$  is called *m-consistent* if they do not contain different answers to the same query by  $m$ , i.e. if for all proper initial segments  $w^i = r_1 \cdots r_i$  and  $u^j = s_1 \cdots s_j$ ,

$$\mathcal{M}(m, w^i) = \mathcal{M}(m, u^j) \in \text{Basic-Terms}(M) \implies r_{i+1} = s_{j+1}.$$

In this paper, we will additionally require for systems of nondeterministic strategies  $\langle M, \mathcal{M} \rangle$  that they should be *wittingly consistent* ([30], Chapter II, §4). This means that, for any  $m \in M$  and any  $m$ -consistent pair of prompts  $w$  and  $u$ , the strategy  $m$  cannot output two contradictory final results:

$$\mathcal{M}(m, w) \in \mathbf{N} \ \& \ \mathcal{M}(m, u) \in \mathbf{N} \implies \mathcal{M}(m, w) = \mathcal{M}(m, u).$$

Sequential (deterministic) systems of strategies are evidently wittingly consistent (assuming the first requirement of Section 3.1.3).

Consider one example of such a wittingly consistent strategy  $m_{\mathbf{pif}}$  computing a parallel conditional monotonic function  $\mathbf{pif}_\iota = \llbracket m_{\mathbf{pif}} \rrbracket : (o, \iota, \iota \rightarrow \iota)$  defined in Section 3.1.4:

$$\begin{aligned} \mathcal{M}(m_{\mathbf{pif}}, \Lambda) &= \#, \\ \mathcal{M}(m_{\mathbf{pif}}, 0) &= \text{“}p = \text{?”}, \\ \mathcal{M}(m_{\mathbf{pif}}, 0 \mathbf{true}) &= \text{“}x = \text{?”}, \ \mathcal{M}(m_{\mathbf{pif}}, 0 \mathbf{false}) = \text{“}y = \text{?”}, \\ \mathcal{M}(m_{\mathbf{pif}}, 0 \mathbf{true} v) &= v, \ \mathcal{M}(m_{\mathbf{pif}}, 0 \mathbf{false} v) = v, \\ \mathcal{M}(m_{\mathbf{pif}}, 1) &= \text{“}x = \text{?”}, \ \mathcal{M}(m_{\mathbf{pif}}, 1 v) = \text{“}y = \text{?”}, \ \mathcal{M}(m_{\mathbf{pif}}, 1 v v) = v. \end{aligned}$$

In all other cases  $\mathcal{M}(m_{\mathbf{pif}}, w)$  is undefined.

Consider also *parallel disjunction*  $\vee : (o, o \rightarrow o)$  (used in infix notation)

$$p \vee q \hat{=} \mathbf{pif} \ p \ \mathbf{then} \ \mathbf{true} \ \mathbf{else} \ q.$$

It is parallel (as well as **pif**) because it is **true** if *any one* of the arguments is **true** while the other may even be undefined ( $\perp$ ). Thus, there is no sequential way of evaluating the arguments, but an appropriate wittingly consistent strategy exists.

For wittingly consistent strategies, the *interpreted (nondeterministic) computation* is defined as before in Section 4.2. All the successful computations under *any* given interpretation of strategies  $\llbracket - \rrbracket$  should evidently lead to a unique value  $v \in \mathbf{N}$  independently of the non-deterministic steps. This gives rise, as before, to the concept of the (least correct and naturally defined) denotational semantics  $\llbracket - \rrbracket$  for any system of wittingly consistent strategies. As to operational semantics,  $\langle\langle - \rangle\rangle$ , we can easily show that the (appropriately defined as in Section 6.2) system of strategies  $\langle \hat{M}, \hat{\mathcal{M}} \rangle$  is wittingly consistent if  $\langle M, \mathcal{M} \rangle$  is.

In the most general case of nondeterministic strategies (the least) denotational semantics may give rise to  $\llbracket m \rrbracket = \top$ , the “over-defined” or “contradictory” value, for some “contradictory”  $m$  because for some values  $\bar{x}$  the interpreted computation of the value  $\llbracket m \rrbracket^+ \bar{x}$  gives different final results in  $\mathbf{N}$  for various paths of the computation. A weaker concept of consistency [29, 30] of a system of nondeterministic strategies (in a structure) means the mere possibility of giving (the least) denotational semantics with  $\llbracket m \rrbracket \bar{x} = \llbracket m \rrbracket^+ \bar{x} \neq \top$  for all strategies in  $M$  independently of the ways of computation. Witting consistency is a kind of guarantee, or sufficient condition, of the existence (say, in  $\{\mathbb{D}_\alpha\}$ ) of “non-contradictory” semantics. Otherwise this existence would be either somewhat accidental and unpredictable, or just fail, because of nondeterminism.

The theory for sequential strategies vs. **PCF** considered so far can be naturally and, in many cases, straightforwardly extended for the case of wittingly consistent nondeterministic strategies vs. **PCF**<sup>+</sup> (= **PCF** + **pif**), giving a fully abstract and naturally continuous order extensional model  $\mathbb{W} = \{\mathbb{W}_\alpha\}$  consisting exactly of all functionals definable in **PCF**<sup>+</sup> + all monotonic functions  $f : \mathbf{N}_\perp \rightarrow \mathbf{N}_\perp$ . (Corresponding results for  $\{\mathbb{D}_\alpha\}$ , instead of the case  $\{\mathbb{W}_\alpha\}$  considered here for the first time, were announced without proof in [30].) This model can be defined, like  $\mathbb{Q}$ , both inductively, level-by-level of types, and as a quotient  $\tilde{W}$  of the universal system  $\langle W, \mathcal{W} \rangle$  of wittingly consistent strategies. The universal functionals  $U_\alpha^+ \in \mathbb{W}_{(\iota \rightarrow \iota) \rightarrow \alpha}$  for each type can be constructed as for  $\mathbb{Q} \cong \tilde{Q}$  (and  $\mathbb{D}$ ) for sequential functionals. This gives a reasonable answer to a question of Longley and Plotkin in [18] concerning the mere possibility of a general approach to a fully abstract model for **PCF**<sup>+</sup> with definability properties like the above. (Cf. Introduction for a quotation.)

Everything for wittingly consistent strategies goes almost as smoothly as for sequential strategies, except we should make some additional technical considerations needed for the definability of universal functionals  $U_\alpha^+$  with the range being the whole  $\mathbb{W}_\alpha$ . (We mean additional considerations in comparison with the case of sequential functionals and **PCF** [28] — what is unchanged is presented below without proof.) Note, that universal functionals for a (countable) fully abstract term model for **PCF**<sup>+</sup> (of types  $\iota \rightarrow \alpha$ , rather than  $(\iota \rightarrow \iota) \rightarrow \alpha$ ) have also been defined in [18]. But we use our old technique for **PCF** and  $\{\mathbb{D}_\alpha\}$  (here — for the model  $\{\mathbb{W}_\alpha\}$ ) with appropriate additions.

Constructing  $U_\alpha^+$  is the primary goal of Section 8. However, for better understanding both of the nature of wittingly consistent strategies, and that witting consistency is an essential restriction, it makes sense to consider first some example demonstrating that  $\mathbb{W}$  is not  $\omega$ -complete and thus does not coincide with the standard continuous model  $\mathbb{D}$ . Otherwise, the reader can well skip the following subsection.

**8.2.  $\mathbb{W}_{(\iota \rightarrow o) \rightarrow o}$  is not  $\omega$ -Complete.** Although the undefinability result of this section is essentially well-known (in slightly different form) for the case of  $\{\mathbb{D}_\alpha\}$  (cf. [25, 27, 28]), it makes sense to present its proof in terms of wittingly consistent strategies which was not published yet, except in [30]. Applied to the case of  $\{\mathbb{W}_\alpha\}$ , this implies that  $\mathbb{W}_{(\iota \rightarrow o) \rightarrow o}$  is *not  $\omega$ -complete* and, therefore, it is a *proper subset* of  $\mathbb{D}_{(\iota \rightarrow o) \rightarrow o}$ .

Let us define functionals  $\exists$  and  $\exists_n \in \mathbb{D}_{(\iota \rightarrow o) \rightarrow o}$ ,  $n \geq 0$ , with  $P \in \mathbb{D}_{\iota \rightarrow o} = \mathbb{W}_{\iota \rightarrow o}$  any argument for them, by the following equation:

$$\exists_{(n)}P = \begin{cases} \mathbf{true} & \text{if } Px = \mathbf{true} \text{ for some } x (\leq n), \\ \mathbf{false} & \text{if } P\perp = \mathbf{false}, \\ \perp & \text{otherwise.} \end{cases}$$

Recall that **PCF**<sup>++</sup> = **PCF**<sup>+</sup> +  $\exists$  defines exactly all computable functionals (computable — in terms of recursive enumerability of finite approximations) in the standard continuous model  $\{\mathbb{D}_\alpha\}$ , and, by using arbitrary (actually, only strict) functions  $f \in \mathbb{D}_{\iota \rightarrow \iota}$ , this language defines all continuous functionals of this model [25, 27, 28]. On the other hand, each  $\exists_n$  is definable in **PCF**<sup>+</sup> by using the (wittingly consistent) parallel disjunction  $\vee$ :

$$\exists_n P = \mathbf{if} P0 \vee P1 \vee \dots \vee Pn \mathbf{ then true else } P\perp,$$

and therefore  $\exists_n \in \mathbb{W}_{(\iota \rightarrow o) \rightarrow o}$ . Moreover,  $\exists = \biguplus_{n \geq 0} \exists_n$  (pointwise), but  $\exists \notin \mathbb{W}_{(\iota \rightarrow o) \rightarrow o} \subseteq \mathbb{D}_{(\iota \rightarrow o) \rightarrow o}$  because of the following

**Proposition 8.1.**  $\exists$  is not a wittingly consistent functional and hence not definable in  $\text{PCF}^+$ . In particular,  $\mathbb{W}$  is not  $\omega$ -complete at the level 2.

*Proof.* Let us assume on the contrary that  $\llbracket m \rrbracket = \exists$  holds for some strategy  $m$  of the type  $(\iota \rightarrow o) \rightarrow o$  from a wittingly consistent system of strategies  $\langle M, \mathcal{M} \rangle$ . We may consider that all queries asked by the strategy  $m$  computing  $mP$  have the canonical form “ $P(m'P) = ?$ ” for some  $m' : (\iota \rightarrow o) \rightarrow \iota$  in  $M$ .

For each  $i \in \mathbf{N}$ , define  $P_i$  by  $P_i x \rightleftharpoons \mathbf{if } x = i \mathbf{ then true else } \perp$ . Let us show that for various  $i$ , the sets of sub-tasks in any successful interpreted computations for  $mP_i$  do not intersect. To this end, consider two successful interpreted computations of  $mP$  for  $P = P_i$  and  $P = P_j$ ,  $i \neq j$ , both giving a result (actually  $= \mathbf{true}$  by  $\llbracket m \rrbracket = \exists$ ), and assume on the contrary that the initial task “ $mP = ?$ ”, for  $P = P_i$  and  $P = P_j$ , is reduced to the same task “ $P(m'P) = ?$ ” (i.e., with the same  $m'$ ) in the course of these two computations. As both the computations should continue further to the result, we would have  $P_i(\llbracket m' \rrbracket P_i) \neq \perp$ ,  $P_j(\llbracket m' \rrbracket P_j) \neq \perp$ , both actually  $= \mathbf{true}$  by the definition of  $P_i$  and  $P_j$ , and hence  $i = \llbracket m' \rrbracket P_i = \llbracket m' \rrbracket (P_i \sqcup P_j) = \llbracket m' \rrbracket P_j = j$ , contrary to  $i \neq j$ .

Now, let us consider an arbitrary  $m$ -prompt  $w$  giving a defined boolean result  $\mathcal{M}(m, w) = r$ , and show that the only possibility is  $r = \mathbf{true}$ . Indeed, the corresponding  $m$ -computation along  $w$  involves only finite number of queries ( $\mathcal{M}(m, w') \in \text{Basic-Terms}(M)$  for  $w'$  initial segments of  $w$ ) which, by the above consideration, may also participate in successful interpreted computations of  $\llbracket m \rrbracket P_i$  only for a finite number of  $i$ . Therefore, for  $i$  outside this finite set,  $m$ -prompt  $w$  is  $m$ -consistent with the  $m$ -prompt  $w^{(i)}$  arising in some interpreted computation of the value  $\llbracket m \rrbracket P_i$  giving a defined result, which should be  $\mathbf{true}$  by the assumption  $\llbracket m \rrbracket = \exists$ . From the definition of witting consistency, it follows that  $r = \mathbf{true}$ , as required.

Thus, the values of  $\llbracket m \rrbracket P$  for any predicate  $P : \iota \rightarrow o$  may only be  $\perp$  or  $\mathbf{true}$ , and  $\llbracket m \rrbracket$  cannot be  $\exists$  (for which  $\exists P = \mathbf{false}$  is possible), contrary to the main assumption.  $\square$

As to sequential functionals, the increasing sequence  $\exists_n^s \in \mathbb{Q}_{(\iota \rightarrow o) \rightarrow o}$ ,  $n = 0, 1, \dots$  analogous to  $\exists_n \in \mathbb{W}_{(\iota \rightarrow o) \rightarrow o}$  cannot demonstrate that  $\mathbb{Q}$  is not  $\omega$ -complete because this sequence has the limit  $\exists^s$  existing also in  $\mathbb{Q}_{(\iota \rightarrow o) \rightarrow o}$ , as we have shown in Section 3.1. Thus, demonstrating the incompleteness of  $\mathbb{Q}$  requires the more subtle considerations of [23] at the level 3.

It is useful to note that *strictly sequential functionals* of the type  $(\iota \rightarrow o) \rightarrow o$ , i.e. those computable by the sequential strategies asking only simple queries of the form “ $Pi = ?$ ” with  $i \in \mathbf{N}$ , are closed under  $\omega$ -limits. (Hint: first note, that if  $F$  is strictly sequential then so is any  $F' \sqsubseteq F$ , and consider limits of finite, in the sense of  $\{\mathbb{D}_\alpha\}$ , strictly sequential functionals.) Further, for a functional of the type  $(\iota \rightarrow \iota) \rightarrow \iota$  or  $(\iota \rightarrow o) \rightarrow o$ , to be strict (see below) and sequential is equivalent to be strictly sequential. Moreover, looking for limits of sequences of more complicated, non necessarily strict sequential functionals of this type (based on the general queries of the form “ $P(m'P) = ?$ ”) will also fail. In fact, the minimal level of  $\mathbb{Q}_\alpha$  where non- $\omega$ -completeness holds is 3 [23].

**8.3. Definability in  $\text{PCF}^+$  of Strict Continuous Functionals  $F : (\iota \rightarrow \iota) \rightarrow \iota$ .** Here we will consider strict level 2 functionals. We will also rely on some definability concepts and ideas due to Plotkin [25]. A similar definability technique was assumed also in the corresponding results announced in [27, 28], but without presenting details and proofs.



A function  $f \in \mathbb{D}_{\iota \rightarrow \iota}$  is called *strict* if  $f \perp = \perp$ . Given any  $a_i, b_i \in \mathbf{N}$ ,  $i < n$ ,  $n \geq 0$ , with all  $a_i$  different, let  $\begin{bmatrix} b_0, \dots, b_{n-1} \\ a_0, \dots, a_{n-1} \end{bmatrix}$  denote a *strict (naturally) finite function* in  $\mathbb{D}_{\iota \rightarrow \iota}$  such that

$$\begin{bmatrix} b_0, \dots, b_{n-1} \\ a_0, \dots, a_{n-1} \end{bmatrix} x = \begin{cases} b_i, & \text{if } x = a_i \text{ for some } i < n, \\ \perp, & \text{otherwise} \end{cases}$$

or, equivalently,

$$\begin{bmatrix} b_0, \dots, b_{n-1} \\ a_0, \dots, a_{n-1} \end{bmatrix} x = \bigsqcup_{a_i \sqsubseteq x} b_i. \quad (8.1)$$

Recall that more general *finite* (not necessarily strict) functions in  $\mathbb{D}_{\iota \rightarrow \iota}$  are defined by such tables with  $a_i, b_i$  arbitrary elements of  $\mathbb{D}_\iota$ , possibly  $= \perp$ , satisfying a natural consistency requirement, and defined by equation (8.1), and analogously (by induction) for finite elements of arbitrary  $\mathbb{D}_{\alpha \rightarrow \beta}$  with  $a_i$  and  $b_i$  being finite elements, respectively, of  $\mathbb{D}_\alpha$  and  $\mathbb{D}_\beta$ . Note, that any (constant) function in  $\mathbb{D}_{\iota \rightarrow \iota}$  such that  $f \perp \neq \perp$  is also finite ( $f = \begin{bmatrix} c \\ \perp \end{bmatrix}$  for some  $c$ ), but not strict. Let  $\varphi_a$ ,  $a \in \mathbf{N}$ , be an *effective numbering* of all strict finite functions in  $\mathbb{D}_{\iota \rightarrow \iota}$  such that, given  $a$ , the numbers  $n, a_i, b_i$  (all  $\neq \perp$ ) can be recovered.

We can also consider *strict finite functionals* of the form  $\begin{bmatrix} b \\ \varphi \end{bmatrix} \in \mathbb{D}_{(\iota \rightarrow \iota) \rightarrow \iota}$  with  $\varphi$  strict finite and  $b \neq \perp$ :

$$\begin{bmatrix} b \\ \varphi \end{bmatrix} f = \begin{cases} b, & \text{if } \varphi \sqsubseteq f, \\ \perp, & \text{otherwise.} \end{cases}$$

In general, any continuous functional  $F \in \mathbb{D}_{(\iota \rightarrow \iota) \rightarrow \iota}$  is called *strict* if, for all  $f, f' : \iota \rightarrow \iota$ , the coincidence of  $f$  and  $f'$  on all type  $\iota$  arguments  $\neq \perp$  implies  $Ff = Ff'$ . Equivalently,  $F$  is strict if for each  $f$  there exists a strict (and therefore exists a strict finite)  $\varphi \sqsubseteq f$  such that  $Ff = F\varphi$ .

**Lemma 8.2.**

- (a) All strict functionals  $F \in \mathbb{D}_{(\iota \rightarrow \iota) \rightarrow \iota}$  are (uniformly) definable in **PCF**<sup>+</sup> from strict functions of type  $\iota \rightarrow \iota$  and are, in fact, wittingly consistent.
- (b) The same holds for the functionals  $G : \iota, (\iota \rightarrow \iota) \rightarrow \iota$  which are strict in the first type  $\iota$  argument and either constant or strict in the second type  $\iota \rightarrow \iota$  argument (and can be identified with arbitrary sequences  $G_m : (\iota \rightarrow \iota) \rightarrow \iota$ ,  $m = 0, 1, \dots$ , of constant or strict functionals).

*Proof.*

- (a) First, note that parallel disjunction can be generalized to *bounded quantification*. This can be defined in **PCF**<sup>+</sup> recursively (for  $P : \iota \rightarrow o$ ):

$$(\exists i < n. Pi) = \mathbf{if} \ n = 0 \ \mathbf{then} \ \mathbf{false} \ \mathbf{else} \ (\exists i < n - 1. Pi) \vee P(n - 1).$$

In particular,  $(\exists i < \perp. Pi) = \perp$ . This allows us to define in **PCF**<sup>+</sup> a functional

$$\begin{aligned} \# & : (\iota, (\iota \rightarrow \iota) \rightarrow o), \\ \#cf & = \exists i < n (f(a_i) \neq b_i), \end{aligned}$$

assuming that  $\varphi_c = \begin{bmatrix} b_0, \dots, b_{n-1} \\ a_0, \dots, a_{n-1} \end{bmatrix}$  and  $\neq$  is understood as a strict predicate. Here we rely on the simple fact that the number  $n$  and functions  $a_i$  and  $b_i$  of  $i < n$  are computable and **PCF**-definable from  $c \in \mathbf{N}$ . The value of  $\#cf$  is **true** if the strict finite function

$\varphi_c = \left[ \begin{smallmatrix} b_0, \dots, b_{n-1} \\ a_0, \dots, a_{n-1} \end{smallmatrix} \right]$  is inconsistent with  $f$ ;  $\#cf = \mathbf{false}$  if  $\left[ \begin{smallmatrix} b_0, \dots, b_{n-1} \\ a_0, \dots, a_{n-1} \end{smallmatrix} \right] \sqsubseteq f$ ; otherwise,  $\#cf = \perp$ . Also,  $\#\perp f = \perp$ .

Now, any strict  $F$  can be evidently represented as  $F = \bigsqcup_{\alpha(k) \neq \perp} \left[ \begin{smallmatrix} \beta(k) \\ \varphi_{\alpha(k)} \end{smallmatrix} \right]$ , or as

$$F = F_0 \text{ where } F_r = \bigsqcup_{k \geq r, \alpha(k) \neq \perp} \left[ \begin{smallmatrix} \beta(k) \\ \varphi_{\alpha(k)} \end{smallmatrix} \right] = \left[ \begin{smallmatrix} \beta(r) \\ \varphi_{\alpha(r)} \end{smallmatrix} \right] \sqcup F_{r+1}, \quad r \geq 0,$$

with appropriate strict one place numeric functions  $\alpha, \beta : \iota \rightarrow \iota$  such that  $\beta(k) = F\varphi_{\alpha(k)}$ . Although we can take  $\alpha(k) = k$ , we will need the general case. Note that for arbitrary  $\alpha$  and  $\beta$  this lub may not exist if  $\varphi_{\alpha(k)}$  and  $\varphi_{\alpha(k')}$  are consistent, but  $\beta(k) \neq \beta(k')$  for some  $k$ . We can evidently assume that  $\alpha$  and  $\beta$  are defined ( $\neq \perp$ ) on the *same* initial segment of  $\mathbf{N}$ , finite or the whole  $\mathbf{N}$ . (In fact, only two cases suffice here: the whole  $\mathbf{N}$ , or the empty segment, if  $F = \perp$ . But the case of an arbitrary segment will be needed later.) Then for arbitrary  $\alpha$  and  $\beta$  for which the lub  $F_r$  exists we have  $F_r\varphi_{\alpha(k)} = \beta(k)$  if  $\alpha(k) \neq \perp$ , and also  $F_r = \perp$  if  $\alpha(r) = \perp$ . Then  $F_r$  is also definable in  $\mathbf{PCF}^+$  recursively on  $r$  and thus by using the least fixed point operator  $\mathbf{Y}$  as well as the parallel conditional function **pif**:

$$F'_r f = \mathbf{pif} \# \alpha(r) f \text{ then } F'_{r+1} f \text{ else } \beta(r).$$

Let us show that the two definitions are equivalent ( $F_r = F'_r$ ). First note that  $F_r$  satisfying the first definition should also satisfy this formula with  $=$  replaced by  $\sqsubseteq$ , thus giving  $F'_r \sqsubseteq F_r$ . Indeed, the value of the right-hand side, when defined, is equal either to  $\beta(r)$ , if  $\varphi_{\alpha(r)} \sqsubseteq f$ , or to  $F_{r+1}f$ . In both cases the left-hand side,  $F_r f$ , has evidently the same value. For the converse,  $F_r \sqsubseteq F'_r$ , it suffices to show that for the second definition we have  $F'_r\varphi_{\alpha(k)} = \beta(k)$ , for all  $k \geq r$  with defined  $\alpha(k)$ , assuming that the above union does exist, and  $\alpha$  and  $\beta$  are defined on the same initial segment of  $\mathbf{N}$ . This can be shown by induction on  $k - r$ : if  $\varphi_{\alpha(r)}$  contradicts  $\varphi_{\alpha(k)}$  then  $F'_r\varphi_{\alpha(k)} = F'_{r+1}\varphi_{\alpha(k)} = \beta(k)$ ; otherwise,  $F'_{r+1}\varphi_{\alpha(k)} = \beta(k) = \beta(r)$ , and hence again  $F'_r\varphi_{\alpha(k)} = \beta(k)$ .

We can define, in  $\mathbf{PCF}$ , the *correction operator*  $\alpha, \beta \mapsto \alpha', \beta'$  with  $\alpha' \sqsubseteq \alpha$  and  $\beta' \sqsubseteq \beta$  by restricting  $\alpha' = \alpha \upharpoonright \{k \in \mathbf{N} \mid k \leq n\}$ , and the same for  $\beta$ , for the maximal  $n$  (possibly  $= \infty$ ) such that the union  $\bigsqcup_{k \geq 0, \alpha'(k) \neq \perp}^k \left[ \begin{smallmatrix} \beta'(k) \\ \varphi_{\alpha'(k)} \end{smallmatrix} \right]$  exists. Evidently, if the unrestricted union exists for the original  $\alpha$  and  $\beta$  then  $\alpha' = \alpha$  and  $\beta' = \beta$ . This, together with the definition of  $F_r$ , constructs, in  $\mathbf{PCF}^+$ , a *universal functional*  $\tilde{U}\alpha\beta : ((\iota \rightarrow \iota) \rightarrow \iota)$  for all strict continuous functionals of the type  $((\iota \rightarrow \iota) \rightarrow \iota)$ .

Finally, for  $F = \tilde{U}\alpha\beta$ , the functional  $Ff$  can be computed by the strategy  $s$  whose behaviour is definable from the functions  $\alpha'(k)$  and  $\beta'(k)$  as follows:

$$\begin{aligned} \mathcal{M}(s, \Lambda) &\quad \rightleftharpoons \quad \# \\ \mathcal{M}(s, k) &\quad \rightleftharpoons \quad \text{“}f(a_0) = \text{?”}, \\ \mathcal{M}(s, kb_0) &\quad \rightleftharpoons \quad \text{“}f(a_1) = \text{?”}, \\ \mathcal{M}(s, kb_0b_1) &\quad \rightleftharpoons \quad \text{“}f(a_2) = \text{?”}, \\ &\quad \dots \\ \mathcal{M}(s, kb_0b_1 \cdots b_{n-2}) &\quad \rightleftharpoons \quad \text{“}f(a_{n-1}) = \text{?”}, \\ \mathcal{M}(s, kb_0b_1 \cdots b_{n-2}b_{n-1}) &\quad \rightleftharpoons \quad \beta'(k), \end{aligned}$$

where  $\varphi_{\alpha'(k)} = \left[ \begin{smallmatrix} b_0, \dots, b_{n-1} \\ a_0, \dots, a_{n-1} \end{smallmatrix} \right]$ . It is easy to see that  $s$  is wittingly consistent.

(b) Define, essentially,

$$G_m f \equiv \mathbf{if} \ G_m \perp \text{ is constant } c_m \ \mathbf{then} \ c_m \\ \mathbf{else} \ \text{as in the proof of (a), by using some } \alpha_m, \beta_m : \iota \rightarrow \iota.$$

This leads to an universal functional for the required class of type  $(\iota, (\iota \rightarrow \iota) \rightarrow \iota)$  functionals.  $\square$

These definability considerations were devoted mainly to strict type  $(\iota \rightarrow \iota) \rightarrow \iota$  functionals of the standard continuous model  $\{\mathbb{D}_\alpha\}$ . For the monotonic non-dcpo model  $\{\mathbb{W}_\alpha\}$  we have isomorphisms  $\mathbb{W}_\iota \cong \mathbb{D}_\iota$ ,  $\mathbb{W}_{\iota \rightarrow \iota} \cong \mathbb{D}_{\iota \rightarrow \iota}$  (and also for all level 1  $\mathbb{W}_\alpha$ ), but  $\mathbb{W}_{(\iota \rightarrow \iota) \rightarrow \iota} \not\cong \mathbb{D}_{(\iota \rightarrow \iota) \rightarrow \iota}$ , (by Section 8.2). (The same holds for  $\mathbb{Q}_\iota$  and  $\mathbb{Q}_{\iota \rightarrow \iota}$ , whereas  $\mathbb{Q}_{(\iota \rightarrow \iota) \rightarrow \iota}$  is strictly embeddable in  $\mathbb{W}_{(\iota \rightarrow \iota) \rightarrow \iota}$  which is also strictly embeddable in  $\mathbb{D}_{(\iota \rightarrow \iota) \rightarrow \iota}$  and consisting, thereby, of continuous functionals only.) Moreover,  $\mathbb{W}_{(\iota \rightarrow \iota) \rightarrow \iota}$  contains all (but not only) strict continuous functionals. The latter holds because the above Lemma 8.2 on the (relative) definability of strict continuous functionals holds in the **PCF**<sup>+</sup>-model  $\mathbb{W}$ , as well as in  $\mathbb{D}$ .

**8.4. On Denotational Semantics of Wittingly Consistent Strategies.** Let us look again at denotational semantics of any wittingly consistent system of strategies  $\langle M, \mathcal{M} \rangle$ .

For any strategy  $m \in M$  of the type  $\alpha = (\alpha_1, \dots, \alpha_n \rightarrow \iota)$  define a 1-1 computable enumeration of the basic terms  $A_{ma}\{\bar{x}\}$ ,  $a \in \mathbf{N}$ , over  $M$  with variables from the canonical list  $\bar{x} = x_1 : \alpha_1, \dots, x_n : \alpha_n$  only which contains all queries to the Oracle potentially “asked” by the strategy  $m$ .

For any such system  $\langle M, \mathcal{M} \rangle$ , let us construct a system of continuous functionals  $G_m^{\mathcal{M}} : (\iota \rightarrow \iota) \rightarrow \iota$ ,  $m \in M$ , such that the denotational semantics  $\llbracket - \rrbracket$  of the system  $\langle M, \mathcal{M} \rangle$  in the model  $\{\mathbb{D}_\alpha\}$  (respectively, in  $\{\mathbb{W}_\alpha\}$ ) may be equivalently defined (instead of explicitly using the interpreted computations) as the least solution of the system of equations<sup>34</sup>

$$\llbracket m \rrbracket \bar{x} = G_m^{\mathcal{M}}(\lambda a. \llbracket A_{ma}\{\bar{x}\} \rrbracket), \quad m \in M. \quad (8.2)$$

Here  $x_i$  are ranging over  $\mathbb{D}_{\alpha_i}$  (or, alternatively, over  $\mathbb{W}_{\alpha_i}$ ) and, for all  $m \in M$ ,  $\lambda a. \llbracket A_{ma}\{\bar{x}\} \rrbracket$  are considered as strict functions in  $\mathbb{D}_{\iota \rightarrow \iota} = \mathbb{W}_{\iota \rightarrow \iota}$ .

The required functionals  $G_m^{\mathcal{M}}$  can be defined as  $G_m^{\mathcal{M}}(f) = v \in \mathbf{N}$  if, and only if, for some  $w = r_1 \cdots r_k \in \mathbf{N}^*$  the following two conditions hold:

- (1)  $\mathcal{M}(m, w) = v$  (with  $\mathcal{M}(m, w') \notin \mathbf{N}$  for all initial segments  $w'$  of  $w$ ), and
- (2) for all  $i < k$ , if  $\mathcal{M}(m, r_1 \cdots r_i) = A_{ma}$  ( $\neq \#$ ) then  $r_{i+1} = f(a)$ .

This definition is correct ( $v$  does not depend on the choice of  $w$ ) because the system of strategies  $\langle M, \mathcal{M} \rangle$  is wittingly consistent. Indeed, let  $u = s_1 \cdots s_n$  satisfy the analogous condition as  $w$  with  $\mathcal{M}(m, u) = v' \neq v$ . It follows that the pair  $u, w$  is not  $m$ -consistent and for some proper initial segments  $w' = r_1 \cdots r_i$  and  $u' = s_1 \cdots s_j$ ,  $\mathcal{M}(m, w') = \mathcal{M}(m, u') = A_{ma} \in \text{Basic-Terms}(M)$  and  $f(a) = r_{i+1} \neq s_{j+1} = f(a)$  — the contradiction.

The functional  $G_m^{\mathcal{M}}(f)$  is also computable by a strategy  $m^\star : (\iota \rightarrow \iota) \rightarrow \iota$  induced by  $m$ :  $\llbracket m^\star \rrbracket = G_m^{\mathcal{M}}$ . It behaves in the same way as  $m$ , except that instead of the queries “ $A_{ma} = ?$ ” it asks “ $f(a) = ?$ ” for  $a \in \mathbf{N}$ . The resulting system of strategies is denoted as  $\langle M^\star, \mathcal{M}^\star \rangle$ . Evidently,  $\langle M^\star, \mathcal{M}^\star \rangle$  is sequential/wittingly consistent if  $\langle M, \mathcal{M} \rangle$  is.

<sup>34</sup>This means that the fixed point equation  $\llbracket - \rrbracket = \llbracket - \rrbracket^+$  considered formerly can be represented in this form for appropriate  $G_m^{\mathcal{M}}$ .

The equation (8.2) and its versions (8.3) and (8.4) below considerably simplify the corresponding equation in [28]<sup>35</sup> for the sequential case. They will be needed for the construction in  $\mathbf{PCF}^+$  of a universal functional  $U_\alpha^+$  in Section 8.7.

**8.5. Definability of  $G_m^{\mathcal{M}}(f)$ .** Without restricting generality we can consider that the given wittingly consistent system of strategies  $\langle M, \mathcal{M} \rangle$  is countable. Elements of  $M$  may be numbered, or even identified with the natural numbers:  $M = \mathbf{N}$ . Our current goal is to define the functional  $G_m^{\mathcal{M}}(f)$  in  $\mathbf{PCF}^+$  from some type  $\iota \rightarrow \iota$  numerical functions which can be computed from  $\mathcal{M}^\star$  (so that if  $\mathcal{M}$  is effectively computable, such are these numerical functions, too).

According to the strategy  $m$  in  $\langle M, \mathcal{M} \rangle$  or  $m^\star$  in  $\langle M^\star, \mathcal{M}^\star \rangle$ , the functional  $G_m^{\mathcal{M}}(f)$  is evidently either constant  $c_m$  or strict. Therefore  $\lambda m f. G_m^{\mathcal{M}}(f)$  is definable in  $\mathbf{PCF}^+$  from some strict type  $\iota \rightarrow \iota$  functions by Lemma 8.2 (b). Note, that the constants  $c_m$ , the (partial) predicate “ $G_m^{\mathcal{M}}$  is a constant functional  $\neq \perp$ ” and the corresponding numerical functions  $\alpha_m, \beta_m : \iota \rightarrow \iota$  for the strict  $G_m^{\mathcal{M}}(f)$  used in the Lemma are effectively computable from  $\mathcal{M}^\star$  and  $m^\star$ .

**8.6. A Universal Functional for Special Wittingly Consistent Systems of Strategies.** Let us fix an arbitrary Basic-term  $A\{j, \bar{y}, \bar{x}\} : \iota$  constructed from

- symbols of the language  $\mathbf{PCF}$ ,
- a variable  $j : \iota$  and a fixed list of variables  $\bar{y} = y_1, \dots, y_s$  of the same type  $\gamma = (\gamma_1, \dots, \gamma_n \rightarrow \iota)$ , and
- a fixed list of variables  $\bar{x} = x_1 : \gamma_1, \dots, x_n : \gamma_n$ .

Let us also fix a set  $M = \mathbf{PCF} \cup \{\mu_0, \mu_1, \mu_2, \dots\}$  of strategies (the constant symbols) with all  $\mu_p$  of the same type  $\gamma$ . Consider the class  $\mathcal{K}^A$  of all wittingly consistent systems of strategies  $\langle M, \mathcal{M} \rangle$ , with  $M$  fixed as above and  $\mathcal{M}$  varying, but with the ordinary reductions for the constants of  $\mathbf{PCF}$  and such that the terms  $\mu_p x_1, \dots, x_n$  can only be  $\mathcal{M}$ -reduced to terms of the form

$$A\{\underline{j}, \mu_{p_1}, \mu_{p_2}, \dots, \mu_{p_s}, x_1, \dots, x_n\}, \text{ or shortly } A\{\underline{j}, \bar{\mu}_{\bar{p}}, \bar{x}\}$$

with the same fixed  $A$ , where  $\underline{j}$  is a numeral ( $0 + 1 + \dots + 1$ ) and  $p_1, \dots, p_s$  are arbitrary natural numbers. The class of effective systems in  $\mathcal{K}^A$  is called  $\mathcal{K}_{\text{eff}}^A$ .

**Lemma 8.3.** *Both for  $\{\mathbb{D}_\alpha\}$  and  $\{\mathbb{W}_\alpha\}$ , a universal functional  $U_\gamma^A : (\iota \rightarrow \iota) \rightarrow \gamma$  for some superset of  $\mathcal{K}^A$ -computable functionals is definable in  $\mathbf{PCF}^+$ . Specifically,  $U_\gamma^A f$  ranges over some superset of  $\mathcal{K}^A$ -computable ( $\mathcal{K}_{\text{eff}}^A$ -computable) type  $\gamma$  functionals, if  $f$  ranges over all (respectively, all effective) strict monotonic functions of the type  $\iota \rightarrow \iota$ . In particular, each  $\mathcal{K}^A$ -computable ( $\mathcal{K}_{\text{eff}}^A$ -computable) type  $\gamma$  functional is definable in  $\mathbf{PCF}^+$  from some (effective, in the case of  $\mathcal{K}_{\text{eff}}^A$ )  $f : \iota \rightarrow \iota$ .*

*Proof.* The above recursive equation (8.2) becomes now

$$\llbracket \mu_p \rrbracket \bar{x} = G^{\mathcal{M}}(p, \lambda j \bar{p}. \llbracket A\{j, \bar{\mu}_{\bar{p}}, \bar{x}\} \rrbracket) \quad (8.3)$$

with  $x_i$  ranging over  $\mathbb{D}_{\alpha_i}$  (respectively, over  $\mathbb{W}_{\alpha_i}$ ). It is inessential that  $G^{\mathcal{M}}$  here has a slightly different type than in (8.2). So, it is still definable in  $\mathbf{PCF}^+$  from some type  $\iota \rightarrow \iota$  strict functions computable from  $\mathcal{M}$ .

<sup>35</sup>for a functional denoted there as  $H$

Now, consider a variable  $u : \iota \rightarrow \gamma$  and the following version of the above recursive equation

$$up\bar{x} = G^{\mathcal{M}}(p, \lambda j\bar{p}.A\{j, up_1, \dots, up_s, \bar{x}\}) \quad (8.4)$$

By using combinators **S** and **K** to simulate lambda abstraction, and the least fixed point combinator **Y** of an appropriate type, this gives rise to a **PCF**<sup>+</sup>-term  $\hat{U}_\gamma^A\{\bar{f}\} : \iota \rightarrow \gamma$  (corresponding to the above variable  $u : \iota \rightarrow \gamma$ ) depending on some, actually strict, functions  $\bar{f} : \iota \rightarrow \iota$  which were involved in the **PCF**<sup>+</sup>-definition of  $G^{\mathcal{M}}$ . By some trivial encoding this gives rise to the required **PCF**<sup>+</sup>-term  $U_\gamma^A : (\iota \rightarrow \iota) \rightarrow \gamma$  involving no variables  $\bar{f}$  at all.  $\square$

**Note 8.4.** *Lemma 8.3 may be easily generalized to the case of any finite number of terms  $\bar{A}\{j, \bar{y}, \bar{x}\} : \iota$  with the same variables, giving rise to the universal functional  $U_\gamma^{\bar{A}}$  for  $\mathcal{K}^{\bar{A}}$ -computable functionals of the type  $\gamma$ .*

**8.7. A Universal Functional for all Wittingly Consistent Functionals of a Given Type.** The general universal **PCF**<sup>+</sup>-definable functional  $U_\alpha^+ \in \mathbb{W}_{(\iota \rightarrow \iota) \rightarrow \alpha}$ , or its version  $\in \mathbb{D}_{(\iota \rightarrow \iota) \rightarrow \alpha}$ , for all wittingly consistent functionals of any given type  $\alpha$  can be obtained from  $U_\gamma^{\bar{A}}$  for suitable  $\gamma$  and  $\bar{A}$  by using only **PCF**. Here we also employ the fact that, without restricting generality, we can consider only systems of strategies  $m$  asking queries in the canonical form (5.1). Given any such  $m$ , this allows us to “concentrate”, by some encoding most of the strategies descendant to  $m$  (having levels  $\leq$  the level of  $m$ ) in a finite number of types, and, even in only one type  $\gamma$ , (and, analogously, to further restrict the form of queries). That is, the general wittingly consistent systems of strategies can be reduced to the special systems of some class  $\mathcal{K}^{\bar{A}}$  considered above. We omit the details which are presented in [28].

## 9. CONCLUSION

A generalized non-dcpo domain theoretic framework for finite type functionals which are not necessarily closed under directed limits was presented in this paper in terms of pointwise (natural) least upper bounds, and corresponding natural continuity, natural algebraicity and natural bounded completeness properties.

An inductive definition of a monotonic fully abstract model  $\mathbb{Q}$  for **PCF** satisfying the above properties and based on a quite general concept of sequential strategies was also given. This model consists hereditarily of all finite type functionals computable by the sequential strategies which also prove to be uniformly definable in **PCF** from (strict) functions of the type  $\iota \rightarrow \iota$ . This is the universality property also characterising precisely the expressive power of **PCF**. Thereby we have demonstrated that the old concept of sequential strategies [29, 28] can be used quite naturally for defining the fully abstract model along with the more recent game approach [1, 11, 22]. The uniqueness of  $\mathbb{Q}$  was also shown. The essential feature of our definition is its straightforward, inductive and computational character. For each level we just hereditarily restrict the class of monotonic functionals to those that are sequentially computable. However, either the correctness proof of the induction step of this definition, if based on (5.3), or (in the case of alternative definition based on (5.4) with a simpler correctness) proving the main properties of  $\mathbb{Q}$  is more complicated and requires developing a general and quite involved theory of all computational strategies with their generalized

operational semantics coherent with the denotational one. In this way the above “natural” non-dcpo domain theoretic continuity and other properties of  $\mathbb{Q} \cong \tilde{\mathbb{Q}}$  are also shown.

Quite analogous inductive definition of a fully abstract model  $\mathbb{W} \cong \tilde{\mathbb{W}}$  for  $\mathbf{PCF}^+ = \mathbf{PCF} +$  “parallel OR” satisfying the above non-dcpo domain theoretic properties + the universality property relative to  $\mathbf{PCF}^+$  was also briefly outlined in terms of wittingly consistent nondeterministic strategies. The model  $\mathbb{W}$  proves to be not  $\omega$ -complete, as well as the model of sequential functionals  $\mathbb{Q}$  for which this was shown in [23].

As the future perspective, it would be interesting to develop a game semantics version of wittingly consistent strategies. Recall also several domain theoretic hypotheses from Section 2.2 on the model  $\mathbb{Q}$  (equally applicable to  $\mathbb{W}$ ) related with the fact that it is not  $\omega$ -complete, as well as the hypotheses concerning effectiveness of representation of naturally finite functionals in Section 2.4 and the related Notes 7.14 and 7.15 on finite and finitary strategies.

#### ACKNOWLEDGEMENT

The author is grateful to Gordon Plotkin for fruitful discussions on the subject, to Achim Jung for his comments on the domain theoretic part, and to Michael Fisher for his kind help in polishing the English. Thanks to the referees for numerous useful comments helping to considerably improve the exposition and in particular for the amending Definition 5.2 which made its correctness proof just straightforward.

#### REFERENCES

- [1] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000.
- [2] S. Abramsky and A. Jung. Domain theory. In *Handbook of Logic in Computer Science*, volume III, pages 1–168. Clarendon Press, 1994.
- [3] H. P. Barendregt. *The Lambda Calculus, its Syntax and Semantics*. Mir, Moscow, 1985. Russian Translation.
- [4] G. Berry. and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20(3):265–321, 1982.
- [5] A. Bucciarelli. Degrees of parallelism in the continuous type hierarchy. *Theoretical Computer Science*, 177(1):59–71, 1997.
- [6] A. Bucciarelli and T. Ehrhard. Sequentiality and strong stability. In *Proc. 6th Ann. Symp. on Logic in Computer Science*, pages 138–145, New York, 1991. IEEE.
- [7] A. Bucciarelli and T. Ehrhard. A theory of sequentiality. *Theoretical Computer Science*, 113:273–292, 1993.
- [8] R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. In *Proc. 19th POPL*, pages 328–342. ACM Press, 1992.
- [9] P.-L. Curien. Sequentiality and full abstraction. In P.T. Johnstone *et al.*, editor, *Applications of Categories in Computer Science*, pages 66–94. Cambridge Univ. Press, Cambridge, UK, 1992.
- [10] Yu.L. Ershov. Computable functionals of finite types. *Algebra and Logic*, 11(4):367–437, 1972. The journal is translated in English; available via <http://www.springerlink.com> (doi: 10.1007/BF02219096).
- [11] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163:285–408, 2000.
- [12] A. Jung and A. Stoughton. Studying the fully abstract model of PCF within its continuous function model. In *LNCS*, volume 664, pages 230–244, 1993.
- [13] G. Kahn and G.D. Plotkin. Concrete domains. *Theoretical Computer Science*, 121:187–277, 1993. First appeared in French, as INRIA-LABORIA technical report, 1978.

- [14] S.C. Kleene. Turing-machine computable functionals of finite types I. In P. Suppes, editor, *Proc. of the 1960 Congress for Logic, Methodology, and the Philosophy of Science*, page 3845, 1960.
- [15] S.C. Kleene. Turing-machine computable functionals of finite types II. *Proc. London Math. Soc.*, 12:245–258, 1962.
- [16] R. Loader. Finitary PCF is not decidable. *Theoretical Computer Science*, 266:341–364, 2001.
- [17] J. Longley. The sequentially realizable functionals. *Annals of Pure and Applied Logic*, 117:1–93, 2002.
- [18] J. Longley and G. Plotkin. Logical full abstraction and PCF. In J. Ginzburg, Z. Khasidashvili, C. Vogel, J.-J. Levy, and E. Vallduvi, editors, *Tbilisi Symposium on Logic, Language and Computation*, pages 333–352. SiLLI/CSLI, 1998.
- [19] M. Marz. *A Fully Abstract Model for Sequential Computation*. PhD thesis, Darmstadt, 1999.
- [20] M. Marz, A. Rohr, and T. Streicher. Full abstraction and universality via realisability. In *14th Symposium on Logic in Computer Science July 02 - 05, 1999 Trento, Italy*, pages 174–182, 1999.
- [21] R. Milner. Fully abstract models of typed  $\lambda$ -calculi. *Theoretical Computer Science*, 4:1–22, 1977.
- [22] H. Nickau. *Hereditarily-Sequential Functionals: A Game-Theoretic Approach to Sequentiality*. PhD thesis, Siegen, 1996.
- [23] D. Normann. On sequential functionals of type 3. *Mathematical Structures in Computer Science*, 16(2):279–289, 2006.
- [24] P.W. O’Hearn and J.G. Riecke. Kripke logical relations and PCF. *Information and Computation*, 120:107–116, 1995.
- [25] G. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–256, 1977.
- [26] G. Plotkin. *Post-graduate lecture notes in advanced domain theory (incorporating the “Pisa Notes”)*. Dept. of Computer Science, Univ. of Edinburgh, 1981. Available from <http://www.dcs.ed.ac.uk/home/gdp/publications/>.
- [27] V. Yu. Sazonov. Degrees of parallelism in computations. In *MFCS’76, Lecture Notes in Computer Science*, volume 45, pages 517–523, 1976. Available via <http://www.springerlink.com> (doi: 10.1007/3-540-07854-1\_223).
- [28] V. Yu. Sazonov. Expressibility of functionals in D.Scott’s LCF language. *Algebra and Logic*, 15(3):308–330, 1976. The journal is translated in English; available via <http://www.springerlink.com> (doi: 10.1007/BF01876321).
- [29] V. Yu. Sazonov. Functionals computable in series and in parallel. *Sibirskii Matematicheskii Zhurnal*, 17(3):648–672, 1976. The journal is translated in English; available via <http://www.springerlink.com> (doi: 10.1007/BF00967869).
- [30] V. Yu. Sazonov. *On Semantics of the Applicative Algorithmic Languages*. PhD thesis, Novosibirsk, Institute of Mathematics, 1976. (In Russian.) Available, e.g., from the Russian State Library in Moscow.
- [31] D. S. Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121(1&2):411–440, 1993. Böhm Festschrift. Article has been widely circulated as an unpublished manuscript since 1969.
- [32] K. Sieber. Reasoning about sequential functions via logical relations. In M. P. Fourman *et al.*, editor, *Applications of Categories in Computer Science*, pages 66–94. Cambridge University Press, Cambridge, UK, 1992.
- [33] W. Tait. Intensional interpretation of functionals of finite types I. *J. Symbolic Logic*, 32:198–212, 1967.
- [34] M. B. Trakhtenbrot. Relationships between classes of monotonic functions. *Theoretical Computer Science*, 2:225–247, 1976.

## APPENDIX A. UNIVERSAL SYSTEM OF SEQUENTIAL STRATEGIES $\langle Q, \mathcal{Q} \rangle$

Here we give a construction of the typed version of the universal system of strategies  $\langle Q, \mathcal{Q} \rangle$  [29] (with the details which are a bit more complicated than in the untyped case presented formerly only in [30]).

Let  $\bar{\mathbf{N}}$  consist of duplicates of natural numbers  $\bar{0}, \bar{1}, \bar{2}, \dots$  so that  $\bar{\mathbf{N}}$  is disjoint with  $\mathbf{N}$  (and with any other set considered below), and  $\square_\alpha$  be the “empty” constant (placeholder for a strategy from  $Q_\alpha$ ) of a type  $\alpha$  for each  $\alpha$ . Then, according to Section 3.1.1,  $\text{BT}_\square \equiv$

Basic-Terms( $\{\square_\alpha \mid \alpha \in \text{Types}\}$ ) is the set of basic terms (possibly with variables) over the set of constants  $\square_\alpha$ .

Define  $Q$  (recursively) as the set of all functions

$$q : (\mathbf{N} \cup \bar{\mathbf{N}})^* \cup \{\mathbf{type}\} \rightarrow \mathbf{N}_\perp \cup \text{BT}_\square \cup \text{Types}$$

(considered as partial due to  $\perp \in \mathbf{N}_\perp$ ) satisfying the following conditions for all  $u, w \in (\mathbf{N} \cup \bar{\mathbf{N}})^*$  and  $\bar{j} \in \bar{\mathbf{N}}$ :

- (1)  $q((\mathbf{N} \cup \bar{\mathbf{N}})^*) \subseteq \mathbf{N}_\perp \cup \text{BT}_\square$ .
- (2)  $q(\mathbf{type}) \in \text{Types}$ .

We write  $q : \alpha$  if  $q(\mathbf{type}) = \alpha$  and take  $Q_\alpha \equiv \{q \in Q \mid q : \alpha\}$ . If we have a map  $q' : (\mathbf{N} \cup \bar{\mathbf{N}})^* \rightarrow \mathbf{N}_\perp \cup \text{BT}_\square$  (i.e.  $q'$  is undefined on  $\mathbf{type}$ ) then writing  $q' : \alpha$  also means “assignment” of the type  $\alpha$  to  $q'$ , i.e. adding  $\{\mathbf{type} \mapsto \alpha\}$  to the graph of  $q'$  so that  $q = (q' : \alpha)$  is a map with  $q(\mathbf{type}) = \alpha$ .

- (3)  $q(u) \in \mathbf{N}_\perp \Rightarrow q(uw) = \perp$  for non-empty  $w$ .
- (4) **If**  $q(u) = B \in \text{BT}_\square$  and  $u \in \mathbf{N}^*$  **then** all variables in  $B$  are from the canonical list  $x_1, \dots, x_n$  for the type of  $q$  (the condition similar to that in Definition 3.1(1)).
- (5) **If**  $q(u) = B \in \text{BT}_\square$  (with  $u \in (\mathbf{N} \cup \bar{\mathbf{N}})^*$ ) **and**  $B$  contains  $< j$  occurrences of the symbol  $\square$  **then**  $q(u\bar{j}w) = \perp$ , **otherwise**, if the  $j$ -th occurrence of  $\square$  in  $B$  has the type  $\beta_j$  then  $q_j = ((\lambda w. q(u\bar{j}w)) : \beta_j) \in Q$  (in fact,  $\in Q_{\beta_j}$ ).

More precisely, we take the set  $Q$  to be the *largest* one whose elements  $q$  satisfy the above conditions, i.e. the largest set satisfying

$$Q \subseteq \{q : (\mathbf{N} \cup \bar{\mathbf{N}})^* \cup \{\mathbf{type}\} \rightarrow \mathbf{N}_\perp \cup \text{BT}_\square \cup \text{Types} \mid \Phi(q, Q)\}$$

where  $\Phi(q, Q)$  is the (universally quantified by  $u, w$  and  $\bar{j}$ ) conjunction of the above conditions (1)–(5), which is monotonic on  $Q$ . (Note that the least such set is just empty. Thus, the definition of  $Q$  is, in fact, *co-recursive*.)

Define a function  $\mathcal{Q} : Q \times \mathbf{N}^* \rightarrow \mathbf{N}_\perp \cup \text{Basic-Terms}(Q)$ , making the pair  $\langle Q, \mathcal{Q} \rangle$  a system of strategies, by taking for all  $q \in Q$  and  $u \in \mathbf{N}^*$

$$\mathcal{Q}(q, u) \equiv \begin{cases} r, & \text{if } q(u) = r \in \mathbf{N}_\perp, \\ A[q_1, q_2, \dots] \in \text{Basic-Terms}(Q), & \text{if } q(u) = A \in \text{BT}_\square \text{ and} \\ & q_j = (\lambda w. q(u\bar{j}w)) : \beta_j, \\ & j \geq 1. \end{cases} \quad (\text{A.1})$$

Here  $A[q_1, q_2, \dots]$  is the term obtained as the result of the substitution in  $A$  of the strategies  $q_1, q_2, \dots$ , respectively, in place of the first, second, etc. occurrences of  $\square$  in  $A$ , and  $\beta_1, \beta_2, \dots$  are the types of these occurrences.

Our goal is to show the universality of the defined system of strategies  $\langle Q, \mathcal{Q} \rangle$ . First, define  $\mu_j(B)$ , for any  $B \in \text{Basic-Terms}(M)$ , as the  $j$ -th occurrence of an element from  $M$  in term  $B$ . If  $B$  has  $< j$  occurrences of elements from  $M$  then  $\mu_j(B)$  is undefined. (If  $B \in \text{BT}_\square$  then  $\mu_j(B)$  is the  $j$ -th occurrence of a  $\square$ -symbol in  $B$ .) Denote by  $\square_M(B)$  the result of “erasing” in  $B$  of all occurrences of elements from  $M$ , i.e. the result of replacement of all such occurrences by the symbol  $\square$  (of appropriate type). Evidently,

$$B = \square_M(B)[\mu_1(B), \mu_1(B), \dots]$$

(and dually for  $B \in \text{BT}_\square$ ). It will also be convenient to define  $\square_M(r) = r$  for  $r \in \mathbf{N}_\perp$  and  $\square_M(\alpha) = \alpha$  for  $\alpha \in \text{Types}$ .



Given any system of strategies  $\langle M, \mathcal{M} \rangle$  and  $m \in M$ , define two functions

$$\bar{\mathcal{M}}_m : (\mathbf{N} \cup \bar{\mathbf{N}})^* \cup \{\mathbf{type}\} \rightarrow \mathbf{N}_\perp \cup \text{Basic-Terms}(M) \cup \text{Types},$$

$$\mathcal{M}_m : (\mathbf{N} \cup \bar{\mathbf{N}})^* \cup \{\mathbf{type}\} \rightarrow \mathbf{N}_\perp \cup \text{BT}_\square \cup \text{Types},$$

by letting  $\bar{\mathcal{M}}_m(\mathbf{type}) = \mathcal{M}_m(\mathbf{type}) =$  the type of  $m$ , and, iteratively, for any  $u \in \mathbf{N}^*$ ,  $w \in (\mathbf{N} \cup \bar{\mathbf{N}})^*$  and  $\bar{j} \in \bar{\mathbf{N}}$ ,

$$\begin{aligned} \bar{\mathcal{M}}_m(u) &= \mathcal{M}(m, u), \\ \bar{\mathcal{M}}_m(u\bar{j}w) &= \begin{cases} \bar{\mathcal{M}}_{m_j}(w), & \text{if } \mathcal{M}(m, u) = B \in \text{Basic-Terms}(M) \\ & \text{and } m_j = \mu_j(B), \\ \perp, & \text{if } \mathcal{M}(m, u) \notin \text{Basic-Terms}(M) \\ & \text{or } \mu_j(B) \text{ is undefined,} \end{cases} \end{aligned} \quad (\text{A.2})$$

$$\mathcal{M}_m(w) = \square_M(\bar{\mathcal{M}}_m(w)).$$

For any system of strategies  $\langle M, \mathcal{M} \rangle$  the elements  $q$  of the set  $Q_{\mathcal{M}} = \{\mathcal{M}_m \mid m \in M\}$  satisfy the conditions (1)–(5) above. Therefore  $Q_{\mathcal{M}} \subseteq Q$  (preserving types).

**Lemma A.1.** *If  $\varphi : \langle M, \mathcal{M} \rangle \rightarrow \langle M', \mathcal{M}' \rangle$  is a homomorphism then  $\mathcal{M}'_{\varphi(m)} = \mathcal{M}_m$  for all  $m \in M$ .*

*Proof.* First note that for all  $m \in M$  and  $w \in (\mathbf{N} \cup \bar{\mathbf{N}})^*$  the equality  $\bar{\mathcal{M}}'_{\varphi(m)}(w) = (\bar{\mathcal{M}}_m(w))^\varphi$  evidently holds. It follows that  $\mathcal{M}'_{\varphi(m)}(w) = \square_{M'}(\bar{\mathcal{M}}'_{\varphi(m)}(w)) = \square_{M'}((\bar{\mathcal{M}}_m(w))^\varphi) = \square_M(\bar{\mathcal{M}}_m(w)) = \mathcal{M}_m(w)$ .  $\square$

**Lemma A.2.**  $Q_q = q$  for all  $q \in Q$ . Therefore  $Q = \bigcup_{\langle M, \mathcal{M} \rangle} Q_{\mathcal{M}}$ .

*Proof.* As  $Q_q = \square \circ \bar{Q}_q$  and  $Q_q(\mathbf{type}) = \bar{Q}_q(\mathbf{type}) = q(\mathbf{type})$ , it evidently suffices to show that, for  $u \in (\mathbf{N} \cup \bar{\mathbf{N}})^*$  and  $q \in Q$ ,

$$\bar{Q}_q(u) = \begin{cases} r, & \text{if } q(u) = r \in \mathbf{N}_\perp, \\ A[q_1, q_2, \dots] \in \text{Basic-Terms}(Q), & \text{if } q(u) = A \in \text{BT}_\square \text{ and} \\ & q_j = (\lambda w. q(u\bar{j}w)) : \beta_j, \\ & \beta_j = \text{type of } \mu_j(A), j \geq 1. \end{cases} \quad (\text{A.3})$$

This equality is proved by induction on the number of occurrences of symbols from  $\bar{\mathbf{N}}$  in the string  $u$ . If  $u \in \mathbf{N}^*$  then the equality evidently follows from (A.1) and (A.2). Let  $u = y\bar{i}z$  where  $y \in \mathbf{N}^*$ ,  $\bar{i} \in \bar{\mathbf{N}}$  and  $z \in (\mathbf{N} \cup \bar{\mathbf{N}})^*$ . Two cases are possible.

(1)  $q(y) = B \in \text{BT}_\square$ . Then, using (A.1),  $Q(q, y) = B[p_1, p_2, \dots]$  for  $p_i = \lambda t. q(y\bar{i}t) : \gamma_i$  and an appropriate type  $\gamma_i$ , and hence  $\bar{Q}_q(y\bar{i}z) = \bar{Q}_{p_i}(z)$  according to (A.2). The number of occurrences from  $\bar{\mathbf{N}}$  in string  $z$  is less than in  $u$ . Therefore the formula (A.3) is applicable by induction:

$$\bar{Q}_{p_i}(z) = \begin{cases} r, & \text{if } p_i(z) = q(y\bar{i}z) = q(u) = r \in \mathbf{N}_\perp, \\ A[q_1, q_2, \dots], & \text{if } p_i(z) = q(u) = A \in \text{BT}_\square, \text{ and} \\ & q_j = \lambda w. p_i(z\bar{j}w) : \beta_j \\ & = \lambda w. q(y\bar{i}z\bar{j}w) : \beta_j \\ & = \lambda w. q(u\bar{j}w) : \beta_j, \\ & \beta_j = \text{type of } \mu_j(A), j \geq 1. \end{cases}$$

Since  $\bar{Q}_q(u) = \bar{Q}_q(y\bar{i}z) = \bar{Q}_{p_i}(z)$ , this gives exactly the equality (A.3).

- (2)  $q(y) \in \mathbf{N}_\perp$ . Then  $\mathcal{Q}(q, y) \notin \text{Basic-Terms}(\mathcal{Q})$  and therefore  $\bar{\mathcal{Q}}_q(u) = \bar{\mathcal{Q}}_q(y\bar{i}z) = \perp$  by (A.2). Then  $q(y) \in \mathbf{N}_\perp$  entails  $q(u) = q(y\bar{i}z) = \perp$  by the definition of  $\mathcal{Q}$ . Therefore (A.3) holds in this case as well.  $\square$

**Theorem A.3.**  *$\langle \mathcal{Q}, \mathcal{Q} \rangle$  is the unique up to isomorphism universal system of strategies. For any system  $\langle M, \mathcal{M} \rangle$ , the map  $m \mapsto \mathcal{M}_m$  is the unique homomorphism  $\langle M, \mathcal{M} \rangle \rightarrow \langle \mathcal{Q}, \mathcal{Q} \rangle$ .*

*Proof.* The uniqueness of the homomorphism follows from Lemmas A.1 and A.2. For, if  $\varphi : \langle M, \mathcal{M} \rangle \rightarrow \langle \mathcal{Q}, \mathcal{Q} \rangle$  is a homomorphism then  $\varphi(m) = \mathcal{Q}_{\varphi(m)} = \mathcal{M}_m$  for any  $m \in M$ .

To establish that  $m \mapsto \mathcal{Q}_m$  is a homomorphism we need to show that for all  $m \in M$  and  $u \in \mathbf{N}^*$  that

- (i)  $\mathcal{M}(m, u) = r \in \mathbf{N}_\perp \implies \mathcal{Q}(\mathcal{M}_m, u) = r$ , and
- (ii)  $\mathcal{M}(m, u) = A[m_1, m_2, \dots] \in \text{Basic-Terms}(M) \implies \mathcal{Q}(\mathcal{M}_m, u) = A[\mathcal{M}_{m_1}, \mathcal{M}_{m_2}, \dots]$ .

Here we assume that  $A \in \text{BT}_\square$  and  $m_j : \beta_j$ ,  $j \geq 1$ . The first implication is easy. In the second, we need to show by the definition of  $\mathcal{Q}$  that  $\mathcal{M}_m(u) = A$ , and  $\mathcal{M}_{m_j} = \lambda w. \mathcal{M}_m(u\bar{j}w) : \beta_j$ ,  $j \geq 1$ . Both equalities follow from (A.2). The first is easy. For the second, we get  $\mathcal{M}_m(u\bar{j}w) = \bar{\mathcal{M}}_{m_j}(w)$  for all  $w \in (\mathbf{N} \cup \bar{\mathbf{N}})^*$  and  $j \geq 1$ , and apply the erasing operator  $\square_M$ .  $\square$