Logique & Analyse **234** (2016), 133-155

PREDICATION AND COMPUTABLE CONCEPTS

Max A. Freund

Abstract

Our interest in this paper is in the concept of predication itself and the extent to which such a concept can consistently be assumed to be fully or semi-computable. We address this problem relative to the logical context provided by a certain formal system that we label *LC* and several well founded results from standard computability theory. System *LC* is a conceptualist second order logical formal system for reasoning with computable concepts. The system also contains an axiom assigning correlates to semi-computable concepts. Concept-correlates are entities whose content include a general specification of the conditions under which the correlated concepts can be truly predicated or cannot be truly predicated of objects or an encoding of such a specification. The axiom in question is justified on the basis of Church's thesis and a discussion concerning the relationship between computable concepts and Turing machines. We show, by means of a semantic model, that predication can be consistently assumed to be semi-computable, when restricted in its application to fully computable or semi-computable concepts. Within the context of *LC*, we also prove that a contradiction ensues if the concept of predication is held to be fully computable. If the concept of predication is assumed to be a semi-computable concept applicable to any concept (computable or otherwise), we show that a consequence follows within *LC* that would contradict a well-established result from computability theory.

Introduction

Conceptualism, as a philosophical theory of predication, looks at concepts as the semantic grounds for the attribution of predicates. That is to say, concepts are to be viewed as the meaning of predicates and, as such, as the basis for predication. A contemporary version of conceptualism has extended such a view so as to cover all linguistic expression, with the exception of sentences. Thus, not only predicates will represent concepts, but also many other sorts of linguistic items such as universal quantifiers and definite descriptions. Different kinds of concepts are postulated corresponding to the diversity of linguistic expressions.

In addition to its general semantic approach based on concepts, the above contemporary version of conceptualism understands concepts as cognitive

doi: 10.2143/LEA.234.0.3159738.

capacities or cognitive structures otherwise based upon such capacities[1]. Exercise of such capacities or structures makes possible the use of the associated linguistic expressions. On the basis of this idea that concepts can be exercised, an important distinction among concepts can be introduced into conceptualism. This is the distinction between computable and non-computable concepts. When computable concepts are exercised, they will differ from the other sort of concepts by their effectiveness. Among computable concepts, one can differentiate those that are *semi-computable* from those that are *fully computable*[2].

In this paper, we shall assume as philosophical background the above contemporary version of conceptualism and focus on the concept of predication itself. This concept is a cognitive capacity or cognitive structure that allows us to identify predications as well as to determine the formal conditions for their truth. We shall be here interested in deciding the extent to which such a concept can consistently be assumed to be (fully or semi-) computable. By means of the concept of predication one determines whether or not a certain mental act constitutes a predication and, if it does, whether the necessary formal conditions for falling under a concept have been fulfilled. We shall intend to determine whether the capacity or structure in question can consistently be assumed to be fully or semi-effective.

The above problem derives its interest from the importance of a more general problem in *metacognition*, that is, in the cognition of cognition or thinking about thinking. Metacognition involves second-order cognitive abilities and capacities, planning, monitoring and evaluation. How effective such abilities, capacities, planning, etc., might be is a central problem in that field[3]. A concept of predication as a cognitive capacity or structure is clearly metacognitive, since its objects are themselves cognitive acts. Thus, an answer to the question regarding its effectiveness would be a step forward towards a complete solution of the aforementioned general problem.

Clearly, the interpretation of a concept as a cognitive capacity or structure allows us to link directly the particular problem of this paper with the above general problem in metacognition. Moreover, such a feature (together with others of the variant of conceptualism here presupposed) provides a clear philosophical ground for connecting metacognitive problems to logical and philosophical problems. Alternative philosophical theories of predication (viz., realism and nominalism) do not offer such a straightforward natural connection. In other words, the possibility of establishing more direct ties to metacognitive problems commends the variant of conceptualism in

[1] For an example of this sort of contemporary form of Conceptualism, see Cocchiarella (2009).

[2] We have introduced the distinctions in question in Freund (1994).

[3] See, for example, Cox (2005).

question as an appropriate theory of predication for inquiring into those sorts of problems.

Now, we shall here not address the particular problem on the concept of predication in the full generality expressed above. Rather, we shall consider it relatively to the logical context provided by a certain formal system that we shall label *LC* and several well founded results in standard computability theory. More clearly, given the logical syntax of *LC*, we shall state several principles that would constitute different interpretations of the question we are here concerned with. Then, we shall inquire whether any of those principles are acceptable within the context of *LC*, given certain outcomes of computability theory. The results proved in this paper might throw light on the relationship between predication and computable concepts from the standpoint of conceptualism. This is because the axiomatic basis of *LC* is philosophically justified on the basis of the form of conceptualism here assumed. But also, they would be relevant for the study of metacognition. The axiomatic basis of *LC* involves principles that might hold for that field.

As an outcome of the present inquiry, it will be proved that a contradiction ensues within the context of *LC*, when the concept of predication is assumed to be a fully computable concept. It will also be established that a consequence would follow that contradicts a demonstrated result from computability theory if the concept of predication is supposed to be a semi-computable concept applicable to any concept (computable or otherwise). Finally, it will be shown that predication can be consistently assumed (relative to *LC*) to be semi-computable, when restricted in its application to fully or semi-computable concepts.

1. Predication, conceptualism and computable concepts

In general, conceptualism as a theory of predication presupposes that general terms stand for concepts, that is, concepts constitute the semantic basis of general terms. Thus, in accordance to the conceptualist view, common nouns and adjectives, for example, are the sorts of linguist expressions that would have concepts as their meaning. This approach to general terms contrasts with those of nominalism and realism. These philosophical theories assume that general terms stand rather for properties and relations, in the case of realism, or for sets, in the case of modern versions of nominalism.

Another important feature of conceptualism concerns the nexus of predication. The fundamental sort of predication for conceptualism is the predication of concepts, which is to be understood as equivalent to the notion of something falling under a concept. So, for example, the assertion that *John is a person* is to be interpreted as equivalent to the judgment that John falls under the concept of being a person.

It is important to note that conceptualism, like nominalism and realism, also accepts the idea that there is predication in language, that is, that we can attribute (monadic or relational) predicates of individuals. Nominalism, in its original version, assumes that predication in language is the only manner in which we can justifiably talk of predication. That is, according to nominalism the only sort of predication is the attribution of a predicate (or rather of a predicate token). Modern versions of nominalism interpret predication in terms of membership in a class.

In the case of realism, in addition to predication in language, there is predication of properties and relations. As such, predication should be understood as the instantiation of a property or a relation. It is this sort of predication that constitutes the basis and accounts for predication in language.

Contrariwise to realism and nominalism, conceptualism looks at predication in language as grounded on the predication of concepts. This latter sort of predication is tantamount to the notion of an object falling under a concept, as remarked above.

In this paper, we shall go beyond the above general tenets of conceptualism. And, as already indicated in the introduction, we shall here commit to a contemporary variant of such a theory. This variant extends the above view regarding general terms to all meaningful linguistic expressions, with the exception of sentences[4]. So, for example, all sort of referential expressions like quantifier phrases, definite descriptions, etc., would also stand for concepts.

In addition to the above projection to a diversity of linguistic expressions, the contemporary version of conceptualism in question proposes an interpretation of concepts as cognitive capacities or cognitive structures based on such capacities. For instance, the sortal concept of being a house is a cognitive capacity whose exercise would allow us to identify, classify and count houses. The complex predicate “to be round and red” stands for a cognitive structure based on the concepts of being round and being red. This structure is a construction from those two concepts by means of the logical operation of conjunction. When exercised, the structure in question would allow us to classify red and round objects. A referential expression like “every animal” also stands for a cognitive structure. This structure is formed from the concept of being an animal and the logical operation of relative quantification. Exercise of this cognitive structure would allow us to refer to the objects identified and classified by the concept of being an animal.

Different kinds of concepts are postulated by the above modern variant of conceptualism. They account for the roles that different sorts of linguistic

[4] Sentences stand for propositions viewed as mental acts yielded by the joint exercise of a predicable concept and a referential concept, in the case of simple staments, and logical operations, in the case of complex sentences.

expressions are meant to fulfill. We shall here focus, however, only on those concepts that would allow us to classify, relate or identify objects. We shall refer to them as *predicable concepts.* These concepts are represented by predicates.

Among predicable concepts, we shall distinguish those that are computable from those that are not. In general, computable concepts differ from other sorts of predicable concepts by their effectiveness when they are exercised.

In the case of computable concepts, we can distinguish those that are semi-computable from those that are fully computable. A fully computable concept is a concept whose exercise would allow the agent to determine in a finite amount of time, without resorting to random devices or to his ingenuity *whether or not* given objects should be identified, characterized or related in the ways the concept does it. For example, the concept of being the sum of two numbers is fully computable. When exercised, such a concept will permit us to determine *effectively* (that is, in a finite amount of time, without resorting to random devices or to our ingenuity) whether or not a certain number is the sum of two other given numbers. In other words, exercise of a fully computable concept will make possible to the agent to determine effectively that given objects fall under the concept, if they actually do. But also, exercise of such a concept will determine effectively that given objects do not fall under the concept, if they actually do not fall under the concept.

On the other hand, exercise of a semi-computable concept will only allow the agent to determine effectively *whether* given entities should be identified, characterized or related in the ways the concept does it. An example of this kind of concept is the concept of being a theorem of pure first-order logic: if a given formula is a theorem of pure first-order logic, then exercise of that concept will permit the agent to determine effectively that such a formula is a theorem of pure first-order logic. However, if the formula is not a theorem of pure first-order logic, then exercise of the concept is question will not allow the agent to determine effectively that this is so. For this reason, the concept of being a theorem of pure first-order logic is not fully computable. Thus, a semi-computable concept will always allow the agent to determine effectively that given objects fall under the concept, if they actually do it. But, exercise of certain semi-computable concepts will not necessarily determine effectively that given objects do not fall under the concept, if they, as a matter of fact, do not fall under such a concept. Clearly, fully computable concepts constitute a subclass of the class of semi-computable concepts.

Owing to the fact that concepts are the semantic grounds for the application of predicates, any computable concept a predicate stands for will provide an effective rule for legitimate applications and/or non-applications of the linguistic expression itself.

Now, our above characterization of computable concept involves the notion of *exercising a concept.* Different uses of linguistic expressions are among the most perspicuous cases that can illustrate such a notion. Speech acts such as assertions or inferences involving the predicate “red”, for instance, would show different ways to exercise the concept of being red. Uses of the mathematical sign of addition in connection with different numbers manifest the exercise of the concept of addition. But not only in linguistic behavior is it where we find situations that indicate possession and exercise of concepts. Cases where it is clear that there is categorization or identification of individuals, without previous acquisition of language, as in infants, reveals exercise and possession of concepts. This also applies to actions clearly derived from the identification of individuals without previous (internal or external) linguistic acts, as when John runs after identifying a leopard without internally or externally proffering a word.

The above examples, and others that might be provided, suggest understanding the act of exercising a concept as a mental act that is constrained by conditions internal and, sometimes, external to the agent carrying out such an act. For example, exercising the concept of being red can be viewed as a mental act that requires both the agent to be able to see (an internal condition) and certain amount of light to be present in the environment (an external condition). A case where one of the conditions is not met might be the sum of certain extremely big numbers requiring a memory capacity surpassing that of a human agent. In this case, the internal condition cannot be fulfilled by such an agent and so the mental act in question cannot be carried out.

2. Computable concepts, concept-correlates and Turing machines

Another important feature of the variant of conceptualism assumed in this paper concerns nominalized predicates. According to such a variant, the possible denotations of nominalized predicates (such as “redness” and “humanity”) are objects[5]. These sorts of objects are called *concept-correlates* and possess an internal link to the possible applications of the concepts. More specifically, concept-correlates contain the conditions under which the concept can be or cannot be truly predicated of objects. That is, they include a general specification of those truth-conditions or an encoding of such a specification.

[5] Objects are entities that do not a have predicable nature by themselves. This would justify considering predicate tokens to be objects: by themselves, they cannot be predicated. They are derivately predicable. That is, they can only be predicated when associated to concepts. A predicate token of “dog” is an object and can be predicated of dogs as long as it is associated to the concept of being a dog.

We shall here interpret correlates of computable concepts as (numerical codes of) Turing machines. We shall also commit to the thesis that every computable concept has a correlate[6]. In what follows, we shall offer two sources of justification for this thesis. These sources will also justify our interpretation of the correlates of computable concepts.

The first source of justification is the existing link between algorithms and computable concepts: there is a need for computable concepts in the apprehension of an algorithm as a procedure for computing a problem or set of problems with respect to a certain entity. Computable concepts provide the cognitive basis for such an apprehension. But also, an algorithm can be formulated for every computable concept.

More clearly, the content of a computable concept provides the intuitive idea of certain procedure whose implementation would link, in a necessary way, its input objects with its output objects. That is, at any possible world, time, space, etc., at which the procedure were to be implemented, no variation would have to be expected in the input-output objects as they are connected by the procedure. Such content is apprehended in the different algorithms associated to the same problem. In other words, each algorithm is cognized as a version of the procedure that would compute the solutions to the problems related to a certain entity. For this reason, the content of the computable concept would provide a unity to the different algorithms for the same function: they can be understood as different versions of the procedure for computing the problems regarding the function. But even in the case where we just know just one algorithm, this algorithm would be cognized as one of the possible ways to achieve the computational procedure for solving those problems.

As an example, take into account the case of computable numerical functions. The values of the same numerical function can be computed by different algorithms, but we associate all of these possible algorithms to the same procedure that constitutes the function. This is because they are cognitively apprehended as implementing in different ways a procedure for computing its values. Were it not for such an apprehension, the different algorithms would be to an agent just sets of instructions, which given the same input-data, yield the same output-data. That is, no connection would be established between the algorithms and the function. The apprehension is possible only if a computable concept constituting the numerical function has been constructed. For instance, if no concept of a number being the sum

[6] In Tichý (1969), a connection is postulated between concepts and sets of Turing machines, viz.: concepts are identified with equivalence classes of Turing-machines. This approach clearly differs from ours, since we are not identifying concepts with classes. Concepts are capacities. Also, concept-correlates in general and concept-correlates of computable concepts, in particular, are not identified with the classes in question.

of two other numbers had been formed, it would not be possible to apprehend an algorithm as computing the values of addition.

The above explains the cognitive relation of algorithms to concepts. But there is also a cognitive relation of concepts to algorithms: there is always the possibility, in principle, of formulating an algorithm corresponding to a computable concept, once the content of such a concept is grasped. The algorithm will express in its set of instructions the content of the concept. Implementation of such an algorithm will classify, identify or relate objects in the same way as it would be done by the exercise of the concept associated to the algorithm. For example, once the content of the concept of a number being the product of two other numbers is grasped, it is possible, in principle, to formulate an algorithm yielding the multiplication of two numbers. Different algorithms associated to the same concept will constitute different ways of expressing the same content of the concept. Addition, for example, can be expressed by an infinite number of algorithms.

In sum, every algorithm can in principle be associated to a computable concept and vice-versa. On the one hand, algorithms are cognized as cases of a certain procedure whose implementation would link, in a necessary way, the input objects with the output objects of such algorithms. The idea of such a procedure is intuitively included in the content of the computable concept. On the other hand, for every computable concept an algorithm can in principle be formulated that would express a version of the procedure.

Now, we have claimed above that we have two sources of justification for the thesis that every computable concept has a Turing machine as correlate. The first source is the above connection between computable concepts and algorithms. The second source of justification is to be found in the so called *Church's thesis* and the arithmetization of *T*-machines.

As is well known, Church's thesis postulates the equivalence of effective computability (i.e., computability by an algorithm) to Turing-computability. From this, it clearly follows that computable concepts are necessarily connected to *T*-machines: computable concepts are necessarily associated to algorithms, as it was justified above, and algorithms are necessarily linked (via Church's thesis) to *T*-machines, and, consequently, computable concepts are necessarily tied to *T*-machines. Implementation of these *T*-machines will classify, identify or relate objects in the same way as it would be done by exercising the concept.

The above link between computable concepts and *T*-machines provides a ground for assigning correlates to computable concepts on the basis of an arithmetization of the *T*-machines. For this purpose, assume the arithmetization of *T*-machines by Davis (1985). Thus, for every computable concept there would be a set of numerical codes associated with it, namely: the gödel numbers of *T*-machines computing characteristic functions whose domains would be the extensions of the computable concept. Take as correlate for a computable concept the smallest number of such a set. These numerical codes will satisfy the necessary conditions for being correlates of computable concepts. The reader should recall that an object can be a correlate of a concept if and only if it contains a general specification of the truth-conditions under which the concept can be or cannot be truly predicated of objects or an encoding of such a specification.

By above, we have justified the thesis that every computable concept has a Turing machine (or rather its numerical code) as a correlate. The nominalization of a predicate standing for a computable concepts will denote one of such numerical codes[7].

3. Logical Syntax

On the basis of the above philosophical assumptions, we shall formulate a logical system that we shall label *LC*. Within the context of *LC*, we shall inquire whether predication could be assumed to be a fully or semi-computable concept. In the present section, we characterize the logic syntax of that system.

We shall here take a language L to be a countable set of individual and predicate constants. We shall assume the availability of denumerable many individual variables as well as denumerable many n-place predicate variables (for each natural number n). We shall also use x, y, z and w, with or without numerical subscripts, to refer in the metalanguage to individual variables and F^n, G^n and R^n to refer to n-place predicate variables. We shall usually drop the superscript when the context makes clear the degree of a predicate variable or when otherwise does not matter what degree it is. For convenience, we shall also use u in order to refer to variables in general.

As primitive logical constants, we shall take &, =, $\sim$, λ, (,), $\forall$ and $\forall^s$. We shall intuitively interpret these constants, respectively, as classical conjunction, identity, classical negation, the lambda abstract operator, left and right parentheses, the universal quantifier and the universal quantifier for semi-computable concepts.

[7] Our interpretation of concept correlates as (numerical codes) of Turing machines assumes Church's Thesis. We think this is not a problematic assumption, given its wide acceptance. For a recent attempt to prove Church's Thesis, see Derschowitz and Gurevich (2008). For discussions on the nature of Church's thesis see Oszwelski-Wolenski-Januz (2007), Coopeland (2002), Mendelson (1990), Shapiro (1981). Also, see Freund (2005) for a discussion of the connection between conceptualism, classical computability theory and Church's thesis. Now, the thesis has been questioned in Kálmar (1959), Péter (1959) and Bowie (1973). But, to our opinion, later critiques have shown those positions not to be correct. For a critique of Kálmar and Péter, see Mendelson (1963), and for that of Bowie see Ross (1974).

Given a language L, we recursively define the set of meaningful expressions of type n of L, (in symbols, $M_n(L)$) as follows:

(1) Every individual variable or constant is in $M_0(L)$; every n-place predicate variable or constant is in both $M_{n+1}(L)$ and $M_0(L)$

(2) If a, $b \in M_0(L)$, then $(a = b) \in M_1(L)$

(3) If $\pi \in M_{n+1}(L)$ and $a_1, \ldots, a_n \in M_0(L)$, then $\pi(a_1, \ldots, a_n) \in M_1(L)$

(4) If $\delta \in M_1(L)$ and $x_1, \ldots, x_n$ are pairwise distinct individual variables, then $[\lambda x_1, \ldots, x_n\ \delta] \in M_{n+1}(L)$, for $n > 0$

(5) If $\delta \in M_1(L)$, then $\sim \delta \in M_1(L)$.

(6) If δ, $\sigma \in M_1(L)$, then $(\delta \ \& \ \sigma) \in M_1(L)$.

(7) If $\delta \in M_1(L)$ and F is a predicate variable, then $\forall F\delta \in M_1(L)$.

(8) If $\delta \in M_1(L)$ and F is a predicate variable, then $\forall^s F\delta \in M_1(L)$.

(9) If $\delta \in M_1(L)$, x is an individual variable and F is a predicate variable, then $\forall x\delta \in M_1(L)$.

(10) If $\delta \in M_1(L)$, then $[\lambda\delta] \in M_0(L)$

(11) If $n > 0$, then $M_{n+1}(L) \subseteq M_0(L)$.
We set $M(L) = \bigcup_{n\in\omega} M_n(L)$, that is, the set of meaningful expressions of L.

We shall use δ, σ and α to refer to meaningful expressions of L.

We understand the well-formed formulas (wffs) of L to be the members of $M_1(L)$. Whenever $t \in M_0(L)$, we shall say that t is a term of L. We shall use a, t and b, with or without numerical subscripts, to refer to terms in general. On the other hand, for $n \geq 1$, we take the n-place predicate of L to be the members of $M_{n+1}(L)$. By clause 11, note that any n-place predicate can also function as a term, for $n \geq 1$. When the latter is the case, the term in question formally represents the nominalization of the predicate. Finally, regarding clause 10, we should note that "$[\lambda\delta]$" should be read as "that δ", that is, as the nominalization of the well-formed formula δ. In other words, clause 10 captures our capacity to linguistically refer to the proposition expressed by a given statement.

The concepts of a bound and free occurrence of a (predicate or individual) variable are understood as usual. An occurrence of a term b in a wff or term σ is said to be a *bound occurrence of* b in σ if some occurrence of a variable in b is a free occurrence of that variable in b but a bound occurrence of that variable in σ. If a and b are terms, then by φ^a/b, where φ is *wff* (or a term), we shall mean the *wff* (or term) that results by replacing each free occurrence of b in φ by a free occurrence of a, if such a *wff* or term exists, and otherwise we take φ^a/b to be just φ itself. We shall say that a is free for b in φ, if φ^a/b is not φ unless a is b.

The quantifier $\forall$ when *applied to predicate variables* should be intuitively interpreted as the universal quantifier whose range is the class of all concepts (computable or otherwise), that is, "$\forall F$" should be read as "for every concept F". When the quantifier is applied to an individual variable x, it should be read as "for every individual x". We shall include abstract objects such as numbers and sets among the individuals falling within the range of a quantifier over individuals. Finally, "$\forall^s F$" should be read as "for every semi-computable concept F".

The existential quantifiers are defined in terms of the universal quantifiers as usual:

- $\exists F\varphi \;=_{df}\; \sim\forall F\sim\varphi$
- $\exists x\varphi \;=_{df}\; \sim\forall x\sim\varphi$
- $\exists^s F\varphi \;=_{df}\; \sim\forall^s F\sim\varphi$

The truth-functional sentential connectives "$\rightarrow$" (material implication), "$\leftrightarrow$" (material equivalence) and "$\vee$" (disjunction) can be defined in the customary way, in terms of the sentential connectives assumed here as primitive.

The universal and existential quantifiers whose range is the class of fully computable concepts will be here defined as follows:

- $\forall^c F\varphi =_{df} \forall F((\exists^s G(G = [\lambda x_1 \ldots x_n Fx_1 \ldots x_n]) \;\&$ $\exists^s R(R = [\lambda x_1 \ldots x_n \sim Fx_1 \ldots x_n])) \rightarrow \varphi)$
- $\exists^c F\varphi =_{df} \sim\forall^c F\sim\varphi$

That is, F is a fully computable concept if and only if there are semi-computable concepts G and R such that the exercise of G will determine that given objects fall under F whenever they do fall under F, and the exercise of R will determine that given objects do not fall under F whenever they do not fall under F.

4. System LC

Given the above logical syntax, we are now able to express in the following formula our assumption (from section 2) that every semi-computable concept has a correlate:

$$(R)\quad \forall^s G\exists x(F = x))$$

We are also able to express the view, of the conceptualist framework presupposed in this paper, that a stage of concept formation can be reached at which every predicate stands for a concept. This presupposition can be conveyed by the following schema:

$$(CP)\quad \exists F(F = [\lambda x_1 \ldots x_n \varphi]),$$

provided F is a variable which does not occur free in φ. If we add this schema and the above formula R to the following axioms and rules, we obtain a logical system for reasoning with computable concepts:
where φ, ψ, σ, γ are *wffs* and u is a predicate or individual variable,

the axioms are

A0. all tautologies
A1. $(a = a)$
A2. $\forall u(\varphi \to \psi) \to (\forall u\varphi \to \forall u\psi)$
A3. $\forall^s F(\varphi \to \psi) \to (\forall^s F\varphi \to \forall^s F\psi)$
A4. $\varphi \to \forall u\varphi$, *provided u does not occur free in* φ.
A5. $\forall x \exists y(x = y)$
A6. $\forall F\varphi \to \forall^s F\varphi$
A7. $([\lambda x_1 \ldots \lambda x_n\, Rx_1 \ldots x_n] = R)$, where R is either a predicate variable or constant.

(LL). $(a = b) \to (\varphi \leftrightarrow \psi)$
(*where* ψ *comes from* φ *by replacing one or more free occurrences of* a *by free occurrences of* b)

(λ-CONV) $[\lambda x_1 \ldots \lambda x_n \varphi](a_1 \ldots a_n) \leftrightarrow \exists x_1 \ldots \exists x_n(a_1 = x_1\ \&\ \ldots\ a_n = x_n\ \&\ \varphi)$ (*where no* x_j *is free in any* a_k, *for* $1 \leq k, j \leq n$)

(Rw) $[\lambda z_1 \ldots z_n \sigma] = [\lambda y_1 \ldots \lambda y_n \sigma(y_1/z_1 \ldots y_n/z_n)]$ where no y_i occurs in σ.

and *the rules* are:

- *MP*: infer φ from $\gamma \to \varphi$ and γ
- *UG*: infer $\forall F\varphi$ from φ

UG/o: infer $\forall x\varphi$ from φ

The above rules and axioms together with R and CP conforms the logical system to which we have referred, in the previous sections, as *LC*.

As usual, if there is a finite sequence of well-formed formulas such that every member of the sequence is either an axiom of *LC* or follows from previous members of the sequence by one of the rules of *LC*, then we shall say that the last formula φ of the sequence is a *theorem of LC*, (in symbols $\vdash_{LC} \varphi$). From now on, every proof of a theorem or derived rule, requiring reasoning in accordance with principles and rules of classical propositional logic, will be indicated by the expression PL[8].

[8] The notion of a derivation in *LC* can be defined as follows: $\Gamma \vdash \varphi$ if and only if there are well formed formulas $\psi_1 \ldots \psi_n \in \Gamma$ such that $\vdash (\psi_1 \& \ldots \& \psi_n) \to \varphi$. For this reason, we are not stipulating any restriction on the UG and UG/o rules.

In what follows, we shall state several theorems and two derived rules that we shall make use of later on.

Derived rules

$UG(s)$: if $\vdash_{LC} \varphi$, then if $\vdash_{LC} \forall^s F\varphi$

(UG^c): if $\vdash_{LC} \varphi$, then $\vdash_{LC} \forall^c F\varphi$.

Theorems

T1 $\vdash_{LC} \forall^s G\exists F\,(G = F)$

T2 $\vdash_{LC} \varphi \to \forall^s F\varphi$, provided F does not occur free in φ.

T3 $\vdash_{LC} (\forall F^n\varphi \to \varphi[\lambda x_1 \ldots x_n\, \sigma]/F)$
(provided F does not occur free in σ, and $[\lambda x_1 \ldots x_n \sigma]$ is free for F in φ)

T4 $\vdash_{LC} \exists x(x = a) \to (\forall x\varphi \to \varphi^a/x)$
(provided x does not occur free in a, and α is free for x in φ).

T5 $\vdash_{LC} \exists^s F^n(F = [\lambda x_1 \ldots x_n \sigma]) \to (\forall^s F^n\varphi \to \varphi[\lambda x_1 \ldots x_n \sigma]/F)$,
(provided F does not ocur free in σ, $[\lambda x_1 \ldots x_n \sigma]$ is free for F in φ)

T6 $\vdash_{LC} \varphi \to \forall^c F\varphi$, provided F does not occur free in φ,

T7 $\vdash_{LC} \forall^c F(\varphi \to \sigma) \to (\forall^c F\varphi \to \forall^c F\sigma)$.

T8 $\vdash_{LC} \forall^c F\exists G(F = G)$.

T9 $\vdash_{LC} \exists^c F^n(F = [\lambda x_1 \ldots x_n \sigma]) \to (\forall^c F^n\varphi \to \varphi[\lambda x_1 \ldots x_n \sigma]/F)$
(provided F does not occur free in σ, $[\lambda x_1 \ldots x_n \sigma]$ is free for F in φ)

T10 $\vdash_{LC} \forall^c F\exists^c G(F = G)$

T11 $\vdash \forall F^s\varphi \to \forall^c F\varphi$

5. Predication and computable concepts

Apart from being a logical system for reasoning with computable concepts, *LC* is also a system that has conceptualism as its justificational ground. Thus, *LC* is an appropiate framework for considering the main problem of this paper, that is, the problem of whether the concept of predication can be consistently assumed to be computable within the context of conceptualism.

In terms of the logical syntax of *LC*, the above problem can be stated as the question of whether any of the following theses are consistent with *LC*:

I. $(\exists^c G)(\forall^c F)(\forall x_1)\ldots(\forall x_n)(G(F, x_1, \ldots x_n) \leftrightarrow F(x_1, \ldots x_n))$

II. $(\exists^c G)(\forall^s F)(\forall x_1)\ldots(\forall x_n)(G(F, x_1, \ldots x_n) \leftrightarrow F(x_1, \ldots x_n))$

III. $(\exists^s G)(\forall^s F)(\forall x_1)\ldots(\forall x_n)(G(F, x_1, \ldots x_n) \leftrightarrow F(x_1, \ldots x_n))$

IV. $(\exists^s G)(\forall^c F)(\forall x_1)\ldots(\forall x_n)(G(F, x_1, \ldots x_n) \leftrightarrow F(x_1, \ldots x_n))$

V. $(\exists^s G)(\forall F)(\forall x_1)\ldots(\forall x_n)(G(F, x_1, \ldots x_n) \leftrightarrow F(x_1, \ldots x_n))$

VI. $(\exists^c G)(\forall F)(\forall x_1)\ldots(\forall x_n)(G(F, x_1, \ldots x_n) \leftrightarrow F(x_1, \ldots x_n))$

Thesis I states that predication is a fully computable concept when restricted to computable concepts and thesis II that it is a fully computable when restricted to semi-computables concepts. Thesis III asserts that predication is a semi-computable concept when confined to semi-computable concepts and thesis IV a semi-computable concept when restricted to fully-computable concepts. Finally, thesis V estates that predication is a semi-computable and thesis VI that is a fully computable concept, independently of the concepts involved.

By T11 and A6, it can easily be shown that within LC either theses II and VI implies I. But, as the following proof shows, thesis I relative to *LC* leads directly into a contradiction:

Let $LC(+)$ be the result of adding thesis I to LC. Then,

1. $\vdash_{LC(+)} \forall^c F^2\, \exists^c G(G = [\lambda x \sim Fxx])$
   (by definitions, A6, T3, UG, A2, A4, PL)
2. $\vdash_{LC(+)} \exists^c G \forall^c F \forall x(G(F, x) \leftrightarrow Fx)$
   (by Thesis I)
3. $\vdash_{LC(+)} \exists^c G \forall x(G([\lambda x \sim Gxx], x) \leftrightarrow [\lambda x \sim Gxx]x)$
   (by 1, 2, Definitions, T7, UG^c, T9, PL)
4. $\vdash_{LC(+)} \exists^c G \forall x(G([\lambda x \sim Gxx], x) \leftrightarrow \sim Gxx)$
   (by 3, A2, LL, PL, UG/o, A4, λ-Conv, A3, UG^c)
5. $\vdash_{LC(+)} \exists^c G(G([\lambda x \sim Gxx], [\lambda x \sim Gxx]) \leftrightarrow \sim G([\lambda x \sim Gxx], [\lambda x \sim Gxx]))$
   (by 1, 4, *R*, T11, T9, T4, UG^c, T7, T11, PL)

   But, by PL and UG^s
   $\vdash_{LC(+)} \forall^c G \sim (G([\lambda x \sim Gxx], [\lambda x \sim Gxx]) \leftrightarrow \sim G([\lambda x \sim Gxx], [\lambda x \sim Gxx]))$.

Therefore, none of the theses I, II and VI are *LC*-consistent and so predication cannot be assumed to be a computable concept. We are left then with theses III, IV and V.

According to a well-known result from computability theory, there is no algorithm for deciding whether an alleged algorithm for computing the values of a total numerical function is indeed such an algorithm (see, for example, Davis (1985), p. 78, Theorem 61). On the basis of this result, we can show that thesis V would be unacceptable: assuming thesis V within the context of *LC* would allow us to easily prove (by Thesis V, T3, UG(s), A3 and PL) the following:

$$(\exists^s G)(\forall x)(G([\lambda x(\exists^c F)(F = x)], x)) \leftrightarrow [\lambda x(\exists^c F)(F = x)]x)).$$

But this formula expresses the proposition that a semi-computable concept G exists that would allow us to effectively decide whether an object is the correlate of a fully computable concept *F* and so a *T*-machine, given our above assumption regarding concept-correlates of semi-computable concepts. On the other hand by this same assumption, a *T*-machine would exist as a correlate for the concept *G*. This means that we would have an algorithm for deciding whether an object is the algorithm of a fully computable concept, in particular, of those concepts corresponding to total numerical functions. But this will contradict what we know from computability theory that there is not such an algorithm. Thus, thesis *V* cannot be accepted.

Finally, let us consider theses IV and III. By T11 and PL, thesis IV clearly follows from III. But since thesis III is consistent relatively to *LC*, as we show in the next section, then our original problem is solved with respect to theses III and IV. Thus, relatively to LC, predication can be assumed to be a semi-computable concept when restricted in its application to computable or semi-computable concepts.

6. *L*-interpretations and the consistency of LC + III

By constructing a formal semantic interpretation for a second order language with nominalized predicates and lambda operators, we show the consistency of thesis III with respect to *LC*. In this semantic proof, we also appeal to certain results from computability theory.

The formal semantic interpretation showing the consistency of *LC* together with thesis III will be based on what we shall call a *frame for nominalized predicates* (*N*-frame, for short). We start by characterizing this latter notion.

By a *N-frame* we shall understand a structure $<D, S_n, Y_n, f>_{n\in\omega}$, where ω is the set of natural numbers and such that (1) D is a non-empty set; (2) for all $n \in \omega$, $S_n \subseteq Y_n \subseteq \wp(D^n)$, where "$\wp(D^n)$" stands for the power set of D^n (for $n = 0$, we set $D^0 = \{\emptyset\} = 1$); (3) there is a set D^* such that $D \subset D^*$ (*i.e.* D is a proper subset of D^*) and (4) f is a function from $D^* \cup \bigcup_{n\in\omega} \wp(D^n)$ into D^* such that

(i) for all $d \in D^*$, $f(d) = d$,

(ii) for every $z \in \bigcup_{n\in\omega} S_n$ there is a $d \in D$ such that $f(z) = d$, and

(iii) for every $n \in \omega$, f restricted to $\wp(D^n)$ is one-to-one.

D is the range of values of the bound individual variables and D^* the range of values of the free individual variables. Sets S_n and Y_n are, respectively, the range of values of the n-place variables bound by the quantifier "$\forall^s$" and the range of values of the n-place variables bound by the universal quantifier "$\forall$". The function f when restricted to $\bigcup_{n\in\omega} S_n$ set-theorically represents the correlation of semi-computable concepts with objects.

Where $\mathfrak{A}$ is a N-frame, we shall say that A is an assignment (of values to variables in $\mathfrak{A}$) if A is a function with the set of variables as domain and such that (1) for all $n \in \omega$, all n-place predicate variables F^n, $A(F^n) \in \mathbf{P}(D^n)$ and (2) for each individual variable x, $A(x) \in D^*$. Also, we set $A(d/u) = (A - \{<u, A(u)>\} \cup \{<u, d>\}$, *i.e.*, $A(d/u)$ is that referential assignment which is exactly like A except (at most) for its assigning d to u.

Where L is a language and $\mathfrak{A}$ a N-frame, we shall say that $I = <h, \mathfrak{A}>$ *is* a model for L (a L-model, for short), if h is a function with L as domain such that for all $n \in \omega$ and all n-place predicate constants $P \in L$, $h(P) \in \wp(D^n)$ and for each individual constant $c \in L$, $h(c) \in D^*$.

Let $I = <g, \mathfrak{A}>$ be an L-model. We shall say that I is an *interpretation of the meaningful expressions of* L (an interpretation of L, for short) if and only for every assignment A in $\mathfrak{A}$ there is function $inter_{I,A}$ from $M(L)$ into $D^* \cup \bigcup_{n\in\omega} (\wp(D^n))$ such that:

1. If a is a variable, then $inter_{I,A}(a) = A(a)$. If $c \in L$ (*i.e.* c is a predicate or individual constant), then $inter_{I,A}(c) = g(c)$.

If $\sigma \in M_{n+1}(L)$, then:

2. If σ is $a = b$, where $a, b \in M_0(L)$, then for all $i \in W$, $inter_{I,A}(\sigma) = 1$ iff $f(inter_{I,A}(a)) = f(inter_{I,A}(b))$;
3. If σ is $\pi(a_1 \ldots a_n)$, where $\pi \in M_{n+1}(L)$ and $a_1 \ldots a_n \in M_0$, then $inter_{I,A}(\sigma) = 1$ *iff* $< f(inter_{I,A}(a_1)) \ldots f(inter_{I,A}(a_n)) > \in inter_{I,A}(\pi)$;
4. If σ is $[\lambda x_1 \ldots x_n \varphi]$, where $\varphi \in M_1(L)$, then $inter_{I,A}(\sigma) = \{<d_1, \ldots, d_n> \in D^n$: $inter_{I,A(d_1/x_1 \cdot d_n/x_n)}(\varphi) = 1\}$;
5. If σ *is* $\sim \varphi$, where $\varphi \in M_1(L)$, then $inter_{I,A}(\sigma) = 1$ iff $inter_{I,A}(\varphi) = 0$
6. If σ is $(\varphi \ \& \ \gamma)$, where $\varphi, \gamma \in M_1(L)$, then $inter_{I,A}(\sigma) = 1$ iff both $inter_{I,A}(\varphi) = 1$ and $inter_{I,A}(\gamma) = 1$
7. If σ is $\forall F^n \gamma$, where $\gamma \in M_1(L)$ then $inter_{I,A}(\sigma)(i) = 1$ iff for every $d \in Y^n$, $inter_{I,A(d/F)}(\gamma) = 1$.
8. If σ is $\forall^s F^n \gamma$, where $\gamma \in M_1(L)$, then $inter_{I,A}(\sigma) = 1$ iff for every $d \in S_n$, $inter_{I,A(d/F)}(\gamma) = 1$
9. If σ is $\forall x \gamma$, where $\gamma \in M_1(L)$, then $inter_{I,A}(\sigma) = 1$ iff for every $d \in D$, $inter_{I,A(d/x)}(\gamma) = 1$
10. If σ is $[\lambda\varphi]$, where $\varphi \in M_1(L)$, then for all $i \in W$, $inter_{I,A}(\sigma) = inter_{I,A}(\varphi)$.

Let $I = <g, \mathfrak{A}>$ be an interpretation of L and A an assignment in $\mathfrak{A}$. We define satisfaction and truth of a *wff* φ of L as follows:

- A satisfies φ in I iff $inter_{I,A}(\varphi) = 1$;
- φ is true in I iff every assignment in A satisfies φ in I.

We now proceed to construct an interpretation for a formal language L in which every theorem of LC+III would be true. We show in this way the consistency of *LC* with thesis *III.* Construction of the interpretation will require some results stemming from computability theory, which we shall introduce before the construction. Those results have been formulated and proved, for example, in Davis (1985).

Let us consider the set of first order well-formed formulas of the language of arithmetic. By a *numerical predicate* we shall mean any of such well-formed formulas containing free variables. Clearly, numerical predicates qualify as *predicates* in the sense of (Davis (1985), p. xxii) and so by (Davis (1985), p. 66), an n-place numerical predicate $P(x_1 \dots x_n)$ is *semi-computable* if and only if there is a partially computable function whose domain is the set $\{<n_1 \dots n_n> \in \omega^n\text{: } P(n_1 \dots n_n)\}$. [Briefly, by a *partially computable function* it is understood a partial numerical function computable by a *T*-machine (cf. Davis (1985), p. 10)].

By a *semicomputable set* S^n we shall mean a set of n-tuples of natural numbers for which there is an n-place semi-computable numerical predicate $P(x_1 \dots x_n)$ such that

$$S^n = \{<n_1 \dots n_n> \in \omega^n : P(n_1 \dots n_n)\}.$$

Now, in accordance with so called *Klenee's enumeration theorem* [*cf.* (Davis, (1985), chapter 5, *theorem* 1.4),], for every semicomputable numerical predicate $P(x_1 \dots x_n)$ there is a natural number z such that

$$P(x_1 \dots x_n) \leftrightarrow (\exists y)\, T_n(z, x_1 \dots x_n, y)$$

where the predicate $T_n(z, x_1 \dots x_n, y)$ is defined as "z is the Gödel number of a Turing machine Z, y is the Gödel number of a computation with respect to Z having only the (Turing representation) of $x_1 \dots x_n$ on the tape in its initial state". (We should note that this predicate is primitive recursive). Consequently, by the definition of a semi-computable set and Kleene's enumeration theorem, for every semicomputable set S^n, the set $G_{S_n} = \{z \in \omega :$ for every $x_1 \dots x_n \in \omega$, $<x_1 \dots x_n> \in S^n \leftrightarrow (\exists y)\, T_n(z, x_1 \dots x_n, y)\}$ is not empty. Let L_{S_n} be the least element of G_{S_n}.

On the basis of the abovementioned results, we proceed to construct a model for the language of arithmetic. As we will show, this model is an interpretation in which all theorems of *LC+III* are true.

Let ω be the set of natural numbers,
$D^* = \omega \cup \bigcup_{n\in\omega} \{A \subseteq \omega^n : \textit{A is not semicomputable}\}$, and $\mathfrak{B}$ be the structure
$< \omega, S^n, \wp(\omega^n), g >_{n\in\omega}$, where

1) $S^n = \{A \in \wp(\omega^n) : A$ is semicomputable$\}$,
2) g is the function from $D^* \cup \bigcup_{n\in\omega} \wp(\omega^n)$ into D^* such that:
   i) for all $d \in D^*$, $g(d) = d$,
   ii) for every $n \in \omega$, if $z = B \in S^n$(*i.e.* B is a semicomputable subset of ω^n), then $g(z) = L_B$ (*i.e.*, the least element of G_B).

Clearly, $\mathfrak{B}$ is a *N*-frame.

Now, let L_{Ar} be the language containing just the set of non-logical symbols X, +, $'$ and 0. Also, let $H = < h, \mathfrak{B} >$, where h is the function with L_{Ar} as domain assigning, respectively, the multiplication, addition, successor relations and the number zero to the constants X, +, $'$ and 0. Clearly, H is an L_{Ar}-model. We now prove that H is an interpretation of L_{Ar}.

Theorem: *H is an interpretation of* L_{Ar}.

Proof: It is clear that for every assignment A in $\mathfrak{B}$ there is a function Int_A from M (L_{Ar}) into $D^* \cup \bigcup_{n\in\omega} \wp(D)$ such that:

0. If a is a variable, then $inter_{I,A}(a) = A(a)$. If $c \in L_{Ar}$, then $Int_A(c) = h(c)$.
1. If σ is $a = b$, where $a, b \in M_0(L)$, then $Int_A(\sigma) = 1$ iff $g(Int_A(a)) = g(Int_A(b))$;
2. If σ is $\pi(a_1 \ldots a_n)$, where $\pi \in M_{n+1}$ and $a_1 \ldots a_n \in M_0$, then $Int_A(\sigma) = 1$ *iff* $< g(Int_A(a_1)) \ldots g(Int_A(a_n)) > \in Int_A(\pi)$;
3. If σ is $[\lambda x_1 \ldots x_n \varphi]$, where $\varphi \in M_1(L)$, then $Int_A(\sigma) = \{< d_1, \ldots, d_n > \in \omega_n : Int_{A(d_1/x_1..d_n/x_n)}(\varphi) = 1\}$;
4. If σ is $\sim \varphi$, where $\varphi \in M_1(L)$, then $Int_A(\sigma) = 1$ iff $Int_A(\varphi) = 0$.
5. If σ is $(\varphi \;\&\; \gamma)$, where $\varphi, \gamma \in M_1(L)$, then $Int_A(\sigma) = 1$ iff both $Int_A(\varphi) = 1$ and $Int_A(\gamma) = 1$
6. If σ is $\forall F^n\gamma$, where $\gamma \in M_1(L)$ then $Int_A(\sigma) = 1$ iff for every $d \in \wp(\omega^n)$, $Int_{A(d/F)}(\gamma) = 1$.
7. If σ is $\forall^s F^n\gamma$, where $\gamma \in M_1(L)$, then $Int_A(\sigma) = 1$ iff for every $d \in S^n$, $Int_{A(d/F)}(\gamma) = 1$.
8. If σ is $\forall x\gamma$, where $\gamma \in M_1(L)$, then $Int_A(\sigma) = 1$ iff for every $d \in \omega$, $Int_{A(d/x)}(\gamma) = 1$
9. If σ is $[\lambda\varphi]$, where $\varphi \in M_1(L)$, then $Int_A(\sigma) = Int_A(\varphi)$.

For every assignment A in $\mathfrak{B}$, the corresponding function Int_A clearly satisfies all clauses on page 10 and so H is an L_{Ar}-interpretation.

We now show that the theorems of *LC*+III are true in *H*.

Theorem: *If φ is a theorem of LC+III, then φ is true in H*

Proof: Axioms *A*0 $-$ *A7*, λ-Conv, Rw, and *R* are clearly true in H. By an inductive argument, it can be shown that LL is true in H. By clauses 3 and 7 of Int_A (for every assignment A in $\mathfrak{B}$) and the fact that a quantifier over $n - ary$ concepts ranges over all the power set of ω^n, *CP* can be easily seen to be true in H. On the other hand, MP, UG, and *UG/o* are truth-preserving rules. Then, it remains to be shown that thesis III is true in *H:*

Let $P^{n+1} = \{< z,\ x \dots x_n > \in \omega^{n+1} | (\exists y)\, T_n(z,\ x_1 \dots x_n,\ y)\}$. Clearly, since $(\exists y) T_n(z, x_1 \dots x_n, y)$ is a semi-computable numerical predicate, P^{n+1} is a semi-computable set and, consequently, $P^{n+1} \in S^{n+1}$. Let A be any assignment in $\mathfrak{B}$ and assume C is a $n - ary$ semi-computable set such that $A(F^n) = C$, where F^n is a n-place predicate variable. Since C is semi-computable, then (by the definition of the correlation function g) there is natural number k such that $g(C) = k$ and for every $< m_1 \dots m_n > \in C$, $(k, m_1 \dots m_n)) \in P^{n+1}$. Clearly then $< m_1 \dots m_n > \in A(F^n)$ if and only if $(g(A(F^n)), m_1 \dots m_n)) \in P^{n+1}$. So, there is a $K \in S^{n+1}$ (*viz.* P^{n+1}) such that for every $C \in S^{n+1}$, and every $m_1 \dots m_n \in \omega$, $Int_{A(K/G,\ C/F,\ m_1/x_1 \dots m_n/x_n)}$ $(G(F, x_1 \dots x_n) \leftrightarrow F(x_1 \dots x_n)) = 1$, where G is any arbitrarily selected m-place predicate variable. Hence,

$$Int_A(\exists^s G)(\forall^s F)(\forall x_1) \dots (\forall x_n)(G(F, x_1, \dots x_n)) \leftrightarrow Fx_1, \dots x_n)) = 1.$$

Therefore, thesis III is true in *H.*

Corollary: *LC+III is consistent.*

7. A comprehension principle for semi-computable concepts

We have formulated a comprehension principle for concepts in general, but not one for semi-computable concepts. Now, the version of conceptualism assumed in this paper does not provide by itself the theoretical grounds for its characterization. It is the standard theory of computability together with the connection established between such a theory and conceptualism (via concept correlates) that will allow us to express a comprehension principle for semi-computable concepts.

Thus, if we take into account the theorem that the intersection of recursively enumerable sets is recursively enumerable [*see, for example*, Rogers (1987), *sections* 5.4 and 5.5], one could justify the following minimal comprehension principle for semi-computable concepts:

$$(SCP)\ \forall^s F_1 \dots \forall^s F_n \exists^s\ G(G = [\lambda x_1 \dots x^n\, \varphi])$$

provided (1) neither the negation nor the identity sign occur in φ, (2) quantifiers over concepts (computable or otherwise) or over semi-computable concepts occur in φ only in lambda abstracts occurring in subject position, *i.e.*, as abstract singular terms (3) no constant predicate occurs in φ, (4) $F_1 \ldots F_n$ are all the predicate variables occurring in φ (5) G does not occur in φ.

The restriction regarding negation in clause 1 is based on the well known result that there are negations of recursively enumerable sets which are not recursively enumerable (see Davis 1987, p. 68, Theorem 1.6). As for identity, clearly the set of true empirical identities is not recursive enumerable. As for clause 2, as far as we know there are not general results regarding functionals on partially recursive functions.

By induction on the complexity of formulas, *SCP* can be shown to be true in our above model *H*, taking into account Theorems 3.1 in Davis (1985), according to which the conjunction of semi-computable predicates is a semi-computable predicate. So, SCP is consistent with LC. Whether SCP can be extended to a more comprehensive schema will be left here as an open problem.

8. Conclusion

In this paper, we have focused on the concept of predication itself, that is, on our cognitive capacity or cognitive structure to identifying acts of predications and to determine the formal conditions for the truth of such predications. Our particular interest was in the question of whether such a concept can consistently be assumed to be computable. We pointed out that this problem derives its interest from the importance of a more general problem in *metacognition*, namely: the question of how effective might be, in the computational sense, our second-order cognitive abilities and capacities, planning, monitoring and evaluation, that are involved in metacognition. The concept of predication would clearly be one of such metacognitive capacities or abilities. Thus, an answer to the question regarding its effectiveness would be a step forward towards a complete solution of the aforementioned general problem.

The solution that we have offered to the particular problem of this paper is relative to the formal system LC and certain results stemming from computability theory. Given the principles and rules of LC and outcomes from computability theory, we have showed that (1) predication can be consistently assumed to be semi-computable, when restricted in its application to fully or semi-computable concepts; (2) when such an application is open to any concept (computable or otherwise), a consequence would follow within LC that would contradict a demonstrated result from computability theory;

and, finally, (3) if it is assumed that the concept of predication is computable, then a contradiction would follow within the context of *LC*.

The concept of predication would be a metacognitive instrument for the assessment of predications. It would allow us to determine whether the exercise of a concept corresponds to an act of predication; and, if it does correspond, the concept of predication will also allow us to decide whether *the formal conditions* for falling under a concept have been fulfilled. The latter is a necessary step towards an evaluation regarding the truth-value of a certain predication. According to our results, partial computational effectiveness is to be expected in the evaluation of a predication, when the exercise of fully or semi-computable concepts are in consideration. That is, if the subjects of a predication actually fall under the attributed concept, the concept of predication will effectively indicate that it is so, if the predicated concept is computable. If such subjects do not fall under the attributed concept, computational effectiveness is not to be generally expected on the part of the concept of predication. Fully computational effectiveness cannot be generally expected for the assessment of the exercise of concepts (computable or otherwise) in acts of predication. So, the concept of predication would be a necessary metacognitive element but not a sufficient one to effectively evaluate such acts (in the computational sense of effectiveness).

Now, even though the results of this paper depend on LC, they might be relevant for the study of metacognition. The axiomatic basis of LC involves principles and rules that might hold for that context, in the sense that human agents might implicitly make use of them in certain metacognitive tasks[9]. The contents of axioms A0-7, LL, λ-Conv and Rw, and the primitive rules of LC are intuitively enough from the cognitive point of view. All of them are either general logical principles regarding the use of concepts or principles and rules of classical first-order logic. On the other hand, the comprehension principle CP expresses our cognitive capacity to construct complex concepts from simpler concepts and logical operations, even if this involves impredicativity. Finally, axiom R is based on our cognitive capacity to nominalize predicates and assign denotations to such nominalizations. Clearly, such a axiom would state a possible stance regarding those assignments. We have here shown some of its consequences.

[9] We should point out that there is a controversy in cognitive psychology regarding the use of deductive logical rules by human agents in their reasoning. Some authors have upheld the thesis that the cognitive basis of our inferences are not implicit deductive rules (see, for example, Johnson-Laird (2005) and (1983)) or even deductive standards of rationality (see, for example, Kahneman, Slovic & Tversky (1982)). However, as indicated in Evans (2002), the theoretical debate between rule based theories and non-rule based theories remains unresolved. Moreover, it has been shown that for certain contexts logical rules are essential (see, Stanovich (1999). Metacognitive evaluations might be contexts of this sort.

Acknowledgment

We are grateful to the referees for their helpful comments and suggestions

References

[1] Cocchiarella, Nino (2009). *Formal Ontology and Conceptual Realism*, Springer Verlag.
[2] Cox, Michael (2005). "Metacognition in Computation: A selected research review", *Artificial Intelligence*, Vol. 169, 104–141.
[3] Bowie, G.L. (1973). "An Argument Against Church's Thesis", *The Journal of Philosophy*, Vol. LXX, 8, 66–76.
[4] Copeland, J. (2002). "Church-Turing Thesis". In *Stanford Encyclopedia of Philosophy*, Ed Zalta (ed.). URL = <http://plato.stanford.edu/entries/churchturing/>
[5] Derschowitz, N., and Gurevich, Y. (2008). "A natural axiomatization of computability and proof of Church's Thesis", *Bulletin of Symbolic Logic*, Vol. 14, 3, 299–350.
[6] Davis, M. (1985). *Computability and Unsolvability*, Dover Publications, New-York.
[7] Evans, J. (2002). "Logic and Human Reasoning: An Assessment of the Deduction Paradigm", *Psychological Bulletin*, Vol. 128, 6, 978–996.
[8] Freund, M. (2005). "Conceptualismo realista y computabilidad", *Critica*, Vol. 37, 3–38.
[9] Freund, M. (1994). "A Minimal Logical System for Computable Concepts and Effective Knowability", *Logique et Analyse*, Vol. 147–148, 339–366.
[10] Heyting A. (Editor) (1959). *Constructivity in Mathematics*, North Holland, Groningen.
[11] Holoyak, K. and Sternberg R. J. (Eds.) (2005). *The Cambridge Handbook of Thinking and Reasoning*, Cambridge University Press, Cambridge, England.
[12] Johnson-Laird, P.N., (1983). *Mental Models*, Cambridge University Press, Cambridge, England.
[13] Johnson-Laird, P.N., (2005). "Mental models in thought". In K. Holoyak and R. J. Sternberg (Eds.), *The Cambridge Handbook of Thinking and Reasoning*, 179–212, Cambridge University Press, Cambridge, England.
[14] Kálmar, L. (1959). "An argument against the plausibility of Church's Thesis". In A. Heyting (ed.), *Constructivity in Mathematics*, 72–80, North Holland, Groningen.
[15] Kahneman, D., Slovic, P., and Tversky, A. (1982). *Judgement under uncertainty: Heuristics and biases.* Cambridge University Press, Cambridge, England.
[16] Kleene, S.C. (1952). *Introduction to Metamathematics*, North Holland Publishing Co., Amsterdam.
[17] Mendelson, E., (1990). "Second Thoughts About Church's Thesis and Mathematics", *Notre Dame of Formal Logic*, Vol. 87, 5, 225–233.
[18] Mendelson, E., (1963). "On some recent criticism of Church's Thesis", *Notre Dame Journal of Formal Logic*, Vol. 4, 3, 581–614
[19] Oszwelski, A., Wolenski, J. and Januz, R., (editors) (2007). *Church's Thesis after 70 Years*, Ontos Verlag, Frankfurt.

[20] Rogers, H. (1987). *Theory of Recursive Functions and Effective Computability*, MIT Press, Cambridge, Mass.
[21] Ross, D. (1974). "Church's Thesis: What its Difficulties Are and Are Not", *Journal of Philosophy*, Vol. 10, 3, 515–25.
[22] Shapiro, S. (1981). "Understanding Church's Thesis", *Journal of Philosophical Logic*, Vol. 10, 3, 353–65.
[23] Stanovich, K.E. (1999). *Who is rational? Studies of individual differences in reasoning.* Erlbaum, Mahwah, New Jersey.
[24] Tichý, P. (1969). "Intension in terms of Turing Machines", *Studia Logica*, Vol. 24, 1, 7–25.

Max A. Freund
Posgrado en Filosofía y Posgrado en
Ciencias Cognitivas Universidad de Costa Rica
San Pedro Montes de Oca San José Costa Rica
Email: mfreundcr@gmail.com