LAMBDA THEORY : INTRODUCTION OF A PSEUDO-CONSTANT FOR THE VOID INTO A 1^{ST} ORDER LANGUAGE AND INTO SET THEORY.

LAURENT DUBOIS

Ph.D. Student, laurent.dubois@ulb.ac.be

Abstract

The purpose of this article is to elaborate an axiomatic set theory expressed in a first order language in which the pseudo-constant Lambda, Λ, denoting the void has been introduced. The introduction of Λ into the first order language of set theory allows to build the empty set from the void by means of the axiom of the parts. On a conceptual level, taking into account Λ makes it possible to redefine the concept of set and to give the empty set a positive definition and a status of emblematic set. On the other hand, the introduction of Λ into the language of a set theory with an empty universe makes it possible to highlight a zero-order logic different and probably more legitimate than the propositional logic, which classicaly deserves the title of zero-order logic because of the absence of quantification on propositional variables. The presence of Λ within a formal language for a theory without elements of type 1 or higher highlights a zero-order logic where quantification is possible on type-0 element. The semantics of such a language will appear empty, but will be the opportunity to introduce the pre-truth value *Empty*.
In a first order theory with a non-empty universe, the interpretation of Lambda as a pseudo-constant and of *Empty* as a pre-truth value, i.e. as different from the "neither... nor" would let the possibility to work with a standard logic rather than with a paracomplete one.

First Lambda will be defined conceptually. After that will be described a zero-order logic based on Lambda for a theory with an empty universe. Then the highlighting of the pre-truth value "Empty" will follow. The second section will be about the description of first-order language and theory taking into account the pseudo-constant Λ, about the axiom of the void, and about the construction of the empty set and that of a hierarchical universe from Λ. Finally, we will pay attention to the conceptual importance and consequences of Lambda.

Keywords: cut, void, potential, pre-element, pseudo-constant, zero-order logic, Empty, pre-truth value, empty set, relative nothing.

1 Lambda

1.1 Short history of the genesis of the idea of Lambda

The initial questioning was: does a hierarchy of empty sets exist, in the same way that a hierarchy of infinite sets was highlighted by Cantor thanks to the definition of equipotence? The answer seems to be yes, thanks to the introduction of a predicate of potentiality that allows to build a relation of potential membership. This will be the object of another article.
Anyway, potential membership implies potential and it appeared that this potential could be identified with a void, a relative void different from the empty

set and symbolized by the Greek letter Lambda in capital form, Λ. For our purpose, it appeared that Λ could be identified with a cut and could constitue a condition of possibility of sets. In a more fundamental way, cuts could also constitute a condition of possibility of the formal language in which the set theory is developed.
The next section is an attempt at defining this astonishingly multi-faceted notion.

1.2 Definition and Specificities of Lambda

1.2.1 Cuts as conditions of possibility of the language itself

Even if the interpretation of Lambda as condition of possibility of the formal language in which the set theory is developed is not the object of this article, it can help to grasp its very true nature. The definition of a formal language, and more generally of the most part of natural languages, consist in a vocabulary, a syntax and a semantics. The basic constituent of a language is the alphabet. An alphabet consist in letters, sounds or symbols. But there is maybe something more fundamental than letters, sounds or symbols. We cannot have different symbols without *cuts*. Cuts are abstract separators. With respect to the language, the alphabet is constituted not only of letters, but also of (abstract) cuts which make it possible to have at least two different letters, and even one letter; indeed, the letters a and b, for example, are different from each other thanks to the cut that allows to create two different letters; and each of these letters are different from a cut! From another perspective, cuts can be seen as what makes it possible for a letter to be expressed in a finite time.

1.2.2 Void as condition of possibility of sets

Now, and this is the real object of this article, cuts play a fundamental role in contexts larger than a language's building. Indeed, cuts prove essential in set theory as separators between elements of the theory, i.e. sets, and as constituents of sets. Even if abstract, cuts have a more concrete aspect in set theory than in language. Indeed, in set theory, cuts can be assimilated to a *void*.
The void,[a] in the sense of an abstract vacant space, is a pre-element and constitutes the only non-element[b] of the theories defined in this article. In a set

[a] In French, the same word *vide* is used for the various English terms "emptiness", "vacuum" and "void". It is extremely difficult to give privilege to one of these different acceptations. Intuitively, "emptiness" would seem to be the more appropriate term for a set theory concept, but it is too empty, too abstract : one cannot say that one fills in an emptiness; but this concept will reveal useful for qualifying a pre-truth value; "vacuum" contains very interesting properties like potentiality and the idea of a vacant space that one can fill in, it also highlights the fact that we are dealing with a kind of abstract physics, but it is too concrete, too specifically physically connoted; "void" is closer to what we mean by "vide" : vacant space that one can fill in, it contains the idea of potentiality with the best compromise between abstract and concrete (in a mereological sense) aspects of Lambda. Let's be notice that we avoid right away exclusive use of the concept of "nothing" to which Russell assimilates the empty set, in the chapter "Classes" of "The Indefinables of Mathematics", where he makes no use of any of the three words we have analysed.

[b] In axiomatic set theory like ZFC that will be a reference for this work, every element is a set : $\forall x(x \text{ is a set})$. The property " being a set " is defined as follows : x is a set iff x is a class and $\exists y(x \in y)$. Every set is a class or a collection; the reverse is not true. The notions of class or collection don't belong to the Lambda theory, they are used for convenience. The

theory with a non-empty universe, the void, denoting a vacant space, a potential, and as a cut, represents a condition of possibility of the elements of the theory.
In theories where the universe is empty, the void is just a potential.
We will start to formalize Lambda for this particular case.

2 Zero-order logic with Lambda or Lambda-0 logic

Zero-order logic will be here reconsidered and the use of Lambda formalized. The language described in this chapter is intended for a set theory with no element of type 1 or higher, with no... set, i.e. with an empty universe of discourse. Lambda only, which is a (pre-) element of type 0, is taken into account. The use of Lambda in a theory with an empty universe is a very special case. Indeed, here, Lambda does not play the role of cut, it is not a condition of possibility, since there is no element. So, Lambda, as a void, is just a potential and has only a passive role. It is classicaly admitted that Propositional Calculus deserves the title of zero-order logic because of the absence of quantification on propositional variables. If we suppose a theory where the only (pre) element is the potential, we can elaborate a logic that will deserve the title of zero-order logic because quantification will be on type zero (pre-)element. The introduction of the pseudo-constant Lambda of type 0 among logical symbols leads to the highlighting of a zero-order logic where quantification is possible on variables that will be instantiated by Lambda denoting the pseudo-object "void", the only (pre-)element of type zero.[c]

2.1 Syntax of zero-order logic and theory with Lambda

2.1.1 Vocabulary

Logical symbols

- Only one pseudo-variable-constant of type 0: Λ. This is a pseudo-constant for "nothing", for a blank, for the void. As Λ constitutes the only possible instantiation of a variable in the zero-order logic, it can play both the role of a variable and of a pseudo-constant.
In addition, it makes it possible to make the distinction between 0-order variable (Λ) and first-order variables $x, y, z....$
- A set of functions, each of some valence (or arity) ≥ 0 which are expressed by lowercase letters f, g, h.
- Symbols denoting logical operators: $\vee, \wedge, \Rightarrow, \Leftrightarrow, \neg$.
- symbols denoting quantifiers: $\forall, \exists$.
- Left and right parenthesis: (,).
- Identity or equality symbol: $=$. Syntactically it behaves like a binary predicate.[d]

property " x is a class " is defined as follows : x is a class iff $(x = \emptyset \vee \exists z(z \in x))$. Lambda isn't the empty set nor a non-empty set, thus it is not a class and consequently it is not a set. But we will see that its treatment in the theory is legitimate as a pre-class.

[c] We have an "objectualist" position.

[d] Actually, the equality symbol in not necessary

- The pre-Truth constant E for "empty" (this is not the "neither true nor false").

So we have two kinds of logical constants: the propositional connectives/operators and the pseudo-constant Lambda. On the other hand, we don't have the Truth constants T for "true" and F for "false", operators of valence 0.

Non-logical symbols

- As we work in the frame of an axiomatic set theory, we have only one relation symbol of arity (valence) 2: $\in$.

- A set of function symbols, each of some valence ≥ 0, which are denoted by lowercase letters f, g, h,... . Function symbols of valence 0 are called constant symbols, and are often denoted by lowercase letters at the beginning of the alphabet a, b, c,... .
These non-logical symbols constitute the signature Σ of our language.

2.1.2 Language : Formation Rules

Terms
- The pseudo-variable-constant Lambda is a term.
- Any expression $f(t_1, ..., t_n)$ of $n \geq 1$ arguments (where each argument t_i is a zero-order term and f is a function symbol of valence n) is a term.
- Closure clause: nothing else is a term.

Well-formed formulas

- Simple and complex predicates: if P is a relation of valence ≥ 1 and the a_i are terms, then $P(a_1, ..., a_n)$ is well-formed. If equality is considered as part of the logic, then $(a_1 = a_2)$ is well-formed. All such formulas are said to be *atomic*. In this respect, the only atomic formula are $\Lambda \in \Lambda$ and $\Lambda = \Lambda$.
- Inductive clause 1: if φ is a wff, then $\neg\phi$ is a wff.
- Inductive clause 2: if φ and ψ are $wffs$, then are $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$, $\varphi \Leftrightarrow \psi$.
- Inductive clause 3: if φ is a wff and Λ is a variable, then $\forall\Lambda\varphi$ and $\exists\Lambda\varphi$ are wffs.[e]

2.2 Semantics of Lambda-0 language

It is clear that neither deductive nor modeling approach is possible in a Lambda-0 theory. Indeed, the atomic formulas $\Lambda \in \Lambda$ and $\Lambda = \Lambda$ being empty formulas, the complex formulas built from them are empty: $\varphi \wedge \psi$, $\varphi \vee \psi$, $\neg\varphi$, $\varphi \Rightarrow \psi$, $\varphi \Leftrightarrow \psi$, $\forall\Lambda\varphi$, $\exists\Lambda\varphi$ are empty expressions. This can be seen in at least two ways: one can consider that the zero-order logic language contain a variable that can be instantiated by Lambda only; in this case, Lambda plays the role both of a variable and of a pseudo-constant. Or one can consider that Lambda denotes the absence of variable: as Lambda is the only (pseudo-)constant of

[e] These are not vacuous quantifier as in, for example, $\forall x(P(y) \Rightarrow Q(y))$.

the zero-order logic, there is no need for variables. Empty formulas lead to the highlighting of the notion of "Empty", which is not a value but a pre-value. It is important to note that it is not equivalent to the "neither true nor false" of the paracomplete logic. *Empty* will reveal essential in semantics of Lambda-1 theory.
We will see that a first order formula considered from the perspective of its zero-order structure reveals the skeleton of the formula, i.e. when all variables are instantiated by the pseudo-object, the pre-element Lambda only: we have a concatenation of logical constants and non logical operators ($\in$ and $=$). So the semantics of the zero-order logic is *empty*!
The domain D of M is empty, which does not prevent to give assignment to Lambda.
Only one possible assignment α: $\alpha[\Lambda \mapsto \Lambda](\Lambda) = \Lambda$.

2.3 Lambda-0 calculus

We have an empty calculus consisting in a simple concatenation of logical operators. Logical constants applied to formulas that contain only the Lambda pseudo-constant probably provide the true zero-order logic, which therefore is not the traditional propositional calculus. Our zero-order logic is a trivial logic where formulas are semantically empty.
Syntactically, we have $T \vdash \Lambda$; semantically, $T \models \Lambda$

No truth values but the pre-truth value Empty is associated with our 0-order calculus.
The crucial question now is: what is the interest of Lambda if neither proof theory nor model theory make sense for our zero-order theory? First, it makes it possible to highlight the true zero-order logic, which is not propositional calculus. Secondly, it allows the introduction of the pre-value "Empty", E. It is essential to understand that "Empty" is not a third value, but a "pre-value"; it is not equivalent to the "neither true nor false" since we are in a situation where we have no reference values! Also, Empty is not to be confused with "vacuous truths" either. Thirdly, Lambda becomes useful when considered in the frame of a first-order theory. Indeed, the possibility to quantify on the 0-order (pre-)element Lambda, allows us (as we will see in section 4) to construct the empty set in a positive way.

2.4 Propositional calculus

Propositional calculus is traditionally assimilated to the zero-order logic because of the absence of quantification on propositional variables. Actually, it is the nature of the object of the logic that must determine its degree. Lambda is much simpler than the individual object of the 1st order logic; this is not the case for propositions. Quantification is possible in zero-order logic, but it is empty because the variables can be instantiated by Lambda denoting the pseudo-object, the pre-element "void", only, which is a *relative nothing*! In the frame of languages for theories that include Lambda, propositional calculus gives the basic structure of the logics, including the zero-order logic. Propositional calculus becomes the fundamental *structure* of the logic.

3 First-order logic and theory with Lambda or Lambda-1 logic and theory

The following chapters will describe the syntax and the semantics of a 1^{st} order language for a theory where the Lambda pseudo-constant denoting the void plays an active role. Let us note this: in the zero-order theory described above, the universe was empty but here the language described is intended for a set theory with a non-empty universe of discourse.
Now Lambda is not used to say that there is nothing in the universe, but to denote the *nothing*. If Lambda was just used to say that the universe is empty, it could not be used in a theory where the universe of discourse is not emtpy!

3.1 Syntax of First-order logic with Lambda

3.1.1 Vocabulary

Alphabet (set of all symbols of the language)

Logical symbols
- An infinite set of variables, symbolized by lowercase letters at the end of the alphabet $x, y, z....$
- The only pseudo-constant of type 0: Λ. This is a pseudo-constant for "nothing", for a blank, for the void. The *void* is a pseudo-object. Lambda denotes this pre-element. In 1^st-order language, Lambda plays the role of a pseudo-constant only; its role of 0-order variable is absorbed by 1^{st}-order variables $x, y, z....$
- A set of functions, each of some valence (or arity) ≥ 0 expressed by lowercase letters f, g, h.
- Symbols denoting logical operators: $\vee, \wedge, \Rightarrow, \Leftrightarrow, \neg$.
- symbols denoting quantifiers: $\forall, \exists$.
- Left and right parenthesis: (,).
- Identity or equality symbol: $=$. Syntactically it behaves like a binary predicate.
- The truth constants T for "true" and F for "false", operators of valence 0.
- The pre-truth constant E for "empty".
- The logical constants $\top$ (tautology), and $\bot$ (antilogy).[f]

Non-logical symbols
- A set of constants, type-1 elements, symbolized by lowercase letters at the beginning of the alphabet: $a, b, c....$
- As we work in the frame of an axiomatic set theory, we have only one relation (predicate) symbol of arity (valence) 2: $\in$.

- A set of function symbols, each of some valence ≥ 0, which are represented by lowercase letters f, g, h,... . Function symbols of valence 0 are called constant symbols, and are often represented by lowercase letters at the beginning of the alphabet a, b, c,... .

[f] In first-order theories, tautologies constitue a proper subset of logical validities.

These non-logical symbols constitute the signature Σ of our logic. Our first order logic with its signature constitutes the langage ℓ of our theory Γ.

3.1.2 Language : formation Rules

Terms
The set of terms is recursively defined by the following rules:
- The pseudo-constant Lambda is a term.
- Any constant is a term.
- Any variable is a term.
- Any expression $f(t_1, ..., t_n)$ of $n \geq 1$ arguments (where each argument t_i is a zero-order term and f is a function symbol of valence n) is a term.
- Closure clause: nothing else is a term.

The variable Lambda is absorbed by the type-1 variables $x, y, z...$

Well-formed formulas
Let us recall that the Lambda-1 language here developed is devoted to axiomatic set theory, where the binary operator of membership $\in$ is the only relation-predicate.[g] Concerning atomic formulas, the variables at the right side of the operator must range over a set of (pre-)elements of type equal or greater than that of the (pre-)elements of the set over which the variables at the left side of the operators range. In other words, we cannot instantiate the right part of the relation with Lambda if the left member represents a set. In all other cases, 1^{st}-order variables $x, y, z...$ can be instantiated by a set only. On the other hand, by default, the variable Lambda is absorbed by the type-1 variables $x, y, z...$
The expression $x \in \Lambda$ is not a wff. The expression $\Lambda \in \Lambda$ is legitimate but empty. We will see that the expression $\Lambda \in x$ is always true.
Concerning the equality symbol, the left and the right members of the relation must be of the same type or degree.
The set of well-formed formulas (wffs or just formulas) is recursively defined by the following rules:
- The following expressions are atomic formulas: $x \in y$, $x = y$.
- Simple and complex predicates: if P is a relation of valence ≥ 0 and the a_i are terms, then $P(a_1, ..., a_n)$ is well-formed. If equality is considered as a part of the logic, then $(a_1 = a_2)$ is well-formed. All such formulas are said to be *atomic*.
- Inductive clause 1: if φ is a wff, then $\neg\phi$ is a wff.
- Inductive clause 2: if φ and ψ are wff, then $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$, $\varphi \Leftrightarrow \psi$ are $wffs$.
- Inductive clause 3: if φ is a wff and x is a variable, then $\forall x\varphi$ and $\exists x\varphi$ are $wffs$.
- Closure clause: nothing else is a wff.

A sentence is a well-formed formula with no free variable of any sort.

Free and bound variables
- Atomic formulas: if φ is an Atomic formula, then x is free in φ if and only if x occurs in φ.
- Inductive clause I: x is free in $\neg\varphi$ if and only if x is free in φ.

[g] s = t is an abbreviation for $\forall x(s \in x \Leftrightarrow t \in x) \wedge \forall x(x \in s \Leftrightarrow x \in t)$.

- Inductive clause II: x is free in $(\varphi \Rightarrow \psi)$ if and only if x is free in either φ or ψ.
- Inductive clause III: x is free in $\forall y\ \varphi$ if and only if x is free in φ and x is different than y.
- Closure clause: x is bound in f if and only if x occurs in f and x is not free in f.

Substitution
If t is a term and f is a formula possibly containing the variable x, then f$[t/x]$ is the result of replacing all free instances of x by t in f.
This replacement results in a formula that logically follows the original one provided that no free variable of t becomes bound in this process. If some free variable of t becomes bound, then to substitute t for x it is first necessary to change the names of bound variables of f to something other than the free variables of t.

3.1.3 *Lambda language*

Let's take the language of the set theory $\ell(\in, =)$. The class of the well-formed formulas (wff) of the theory is the class obtained starting from the atomic formulas $x \in y$ and $x = y$, by means of the following rules: if φ and ψ are wff, then $\varphi \wedge \psi$, $\varphi \vee \psi$, $\neg\phi$, $\varphi \Rightarrow \psi$, $\varphi \Leftrightarrow \psi$, $\forall x\varphi$ and $\exists x\varphi$ are wffs too. Here one uses a 1^{st} order logic.[h] The language of the Lambda theory is the language of set theory $(\in, =)$ admitting blanks, empty/vacant spaces, (identified by our Λ) under certain conditions, in the atomic formulas.
The set theory for which our first-order language is defined obeys the following principle: the Lambda pseudo-constant denotes a pre-element and this only pre-element of the theory (the void) is of a type lower than that of a set and it constitutes the only possible syntactic case outside the type-1 elements, i.e. sets, without having to add something to the language and to the theory (contrary to what happens when one adds ur-elements to the theory). Being of a degree lower than that of a set, the pre-element or pseudo-object "void" cannot contain an object of order 1 or higher,[i] in other words, it cannot be put at the right side of the membership relation if the left side is instantiated by a set, but it can belong to any set. We will note the Lambda language $\ell(\Lambda, \in, =)$.

3.2 Proof Theory

3.2.1 Inference rules

An inference rule is a function from sets of (well-formed) formulas, called premises, to sets of formulas called conclusions. In most well-known deductive systems, inference rules operate on a set of formulas and give a single conclusion.

[h] Concerning the language, we always use a first order logic, at the price of some minor specifications, among which are some restrictions on the use of Lambda in the atomic formulas, and with the option of not introducing a general variable which can be instantiated by a set constant or Lambda. We have highlighted the zero-order logic which underlies the 1^{st} order logic.

[i] The logic of the potential will establish that Lambda can contain higher order elements but only potentially, at a *potential* level. We are working here at an *effective* level.

For instance, the classical inference rule, modus ponens, states that if φ and $\varphi \Rightarrow \psi$ are both theorems, then ψ is a theorem. This can be written as following; if $T \vdash \varphi$ and $T \vdash \varphi \Rightarrow \psi$, then $T \vdash \psi$.

3.2.2 Axioms

Here follows a description of the axioms of first-order logic. As explained above, a given first-order theory has further, non-logical axioms. The following logical axioms characterize our first-order logic of this article.

Quantifiers axioms
Quantifier axioms change according to how the vocabulary is defined, how the substitution procedure works, what the formation rules are and which inference rules are used. Here follows a specific example of these axioms:

PRED-1: $(\forall x Z(x)) \Rightarrow Z(t)$
PRED-2: $Z(t) \Rightarrow (\exists x(Z(x))$
PRED-3: $(\forall x(W \Rightarrow Z(x))) \Rightarrow (W \Rightarrow \forall(x)Z(x))$
PRED-4: $(\forall(x)(Z(x) \Rightarrow W)) \Rightarrow (\exists x Z(x) \Rightarrow W)$

Equality and its axioms

There are several different conventions for using equality (or identity) in first-order logic. The various conventions all give essentially the same results even if they differ in terminology.
The most common convention is to consider the equality symbol as primitive and to add the axioms for equality to the axioms for first-order logic. This is the option we chose. The equality axioms are
- $x = x$ (reflexivity)
- $x = y \Rightarrow f(..., x, ...) = f(..., y, ...)$ for any function f
- $x = y \Rightarrow (P(..., x, ...) \Rightarrow P(..., y, ...))$ for any relation P (Leibniz's law)

where P is a metavariable ranging over *wffs* of the object language. Leibniz's law is sometimes called "the principle of substitutivity", "the indiscernibility of identicals", or "the replacement property". The above forms are axiom schemata: they specify an infinite set of axioms of the above forms, called their instances. Notice that the second schema involving the function symbol f is (equivalent to) a special case of the last schema, namely $x = y \rightarrow (f(..., x, ...) = z \rightarrow f(..., y, ...) = z)$. From the above axioms both symmetry and transitivity for equality follow. Moreover, by the symmetry of equality, the right hand side of the last schema (Leibniz's law) could be strengthened to a biconditional.
In theories with no function symbols and a finite number of relations, it is possible to define equality in terms of these relations, by defining the two terms s and t to be equal if any relation is unchanged by changing s to t in any argument. This is the case of the Lambda-1 theory, which is a set theory with one relation $\in$; we may define s = t to be an abbreviation for $\forall x(s \in x \Leftrightarrow t \in x) \wedge \forall x(x \in s \Leftrightarrow x \in t)$. This definition of equality then automatically satisfies the axioms for equality. In this case, one should replace the usual axiom of extensionality, $\forall x \forall y(\forall z(z \in x \Leftrightarrow z \in y) \Rightarrow x = y)$, by $\forall x \forall y(\forall z(z \in x \Leftrightarrow z \in$

$y) \Rightarrow \forall z(x \in z \Leftrightarrow y \in z))$, i.e. if x and y have the same elements, then they belong to the same sets

3.3 Semantics of Lambda-1 Theory

3.3.1 Interpretation

The (logico-mathematical) interpretation of our formal language will assign a denotation to each non-logical constant or pseudo-constant occurring in language (L) or formulas (F). To individual constants it assigns individuals (from some universe of discourse); to the pseudo-constant Lambda, it assigns the *void*; to predicates of degree 1 it assigns properties (more precisely sets) ; to predicates of degree 2 it assigns binary relations of individuals; to predicates of degree 3 it assigns ternary relations of individuals, and so on; and to sentential letters it assigns truth-values.
More precisely, an interpretation of our formal language L or of a sentence F of L, consists of a non-empty domain D (i.e. a non-empty set) as the universe of discourse together with an assignment that associates with each n-ary operation or function symbol of L or of F an n-ary operation with respect to D (i.e. a function from D_n into D); with each n-ary predicate of L or of F an n-ary relation among elements of D and (optionally) with some binary predicate I of L, the identity relation among elements of D.
In this way an interpretation provides meaning or semantic values to the terms or formulas of the language. The study of the interpretations of formal languages is called formal semantics. In mathematical logic an interpretation is a mathematical object that contains the necessary information for an interpretation in the former sense.
The symbols used in our formal language include variables, logical constants, quantifiers and punctuation symbols as well as the non-logical constants and the pseudo-constant Lambda. The interpretation of a sentence or of a language therefore depends on which non-logical constants it contains. Languages of the sentential (or propositional) calculus allow sentential symbols as non-logical constants. Languages of the first order predicate calculus allow, in addition, predicate symbols and operation or function symbols. We will see how to take into account Lambda in the interpretation of our Lambda-1 language.

3.3.2 First-order structures or models

The model is a pair $\langle D, I\rangle$, where D is a set of elements called the domain of discourse while I is an interpretation of the elements, the non-logical terms of the signature (constants, pseudo-constant, functions and predicates).
The domain D is a set of elements;
The interpretation I is a function that assigns something to constants, pseudo-constant, functions and predicates:
- each constant symbol c is assigned a value of the domain $I(c)$
- the pseudo-constant Lambda is assigned the pre-value "void" or "nothing"

$I(\Lambda)$ of the domain
- each function symbol f of arity n is assigned a function $I(f)$ from Dn to D
- each predicate symbol P of arity n is assigned a relation $I(P)$ over Dn or, equivalently, a function from Dn to True,False. Thus each predicate symbol is interpreted by a Boolean-valued function on D.

Lambda rules

Some remarks about Lambda are required. The different cases of interpretation of an atomic formula where "$\in$" occurs are recursively defined by the following rules (with $x, y \neq \Lambda$):

- Clause 1: $\Lambda \in \Lambda$ is empty according to the semantics of zero-order logic (see supra);
- Clause 2: $x \in \Lambda$ is forbidden; it is not a wff;
- Clause 3: $\Lambda \in x$ is always true by the Axiom of Ante-element (see infra);
- Clause 4: $x \in y$ is let to evaluation $\langle M, \alpha \rangle$;
- Closure clause: there is no other possible type of combination.

The model also includes an interpretation of the signature. Since the elements of the signature are function symbols and predicate symbols, the interpretation gives the "value" of functions and predicates.

Empty and non-empty formulas

We have seen that the (empty) semantics of the zero-order logic based on Lambda included the only pre-value Empty. The semantics of first-order language based on Lambda cannot be satisfied with this sole pre-value. Classical truth values True and False are indispensable. But what about Empty? Is it to be included into the set of classical values in order to constitute a trivalent logic? No, it is not necessary! Actually, we have three possibilities:
- a paracomplete logic with True, False and Empty (in this case, Empty is equivalent to the "neither true nor false")
- Empty is absorbed by False
- Empty replace False

In this article, we chose the second option; empty formulas will be assimilated to false formulas.
With these restrictions, a standard logic will be sufficient as it is the case in set theories including ur-elements. Let us recall that Lambda is a kind of ur-ur-element.

3.3.3 Evaluation

The evaluation of a formula consists in a model, with its domain and assignment, and an interpretation of the value of the variables.

A formula evaluates to True, False or Empty given the model and an interpretation of the value of the variables. Such an interpretation α associates every variable to a value of the domain.
The evaluation of a formula under the model $M = \langle D, I \rangle$ and an interpretation α of the variables is defined from the evaluation of a term under the same pair. Note that the model itself contains an interpretation (which evaluates constants, pseudo-constant, functions, and predicates); we additionally have, separated from the model, an interpretation. So, with the interpretation of the model and the variable assignment, we have:
- every constant is assigned its value according to the interpretation of the model, that is, the value I(c) of c;
- the pseudo-constant Lambda is assigned its value I(Λ) of Λ, i.e. the void, the relative nothing;
- every variable is associated its value according to α;
- a term $f(t_1, ..., t_n)$ is associated the value given by the interpretation of the function and the interpretation of the terms: if $(v_1, ..., v_n)$ are the values associated to $(t_1, ..., t_n)$, the term is associated the value $I(f)(v_1, ..., v_n)$; recall that $I(f)$ is the interpretation of f, and so is a function from $D^n toD$.

Next, each formula is assigned a truth value. The inductive definition used to make this assignment is called the T-schema.

1. **Atomic formulas (1).** A formula $P(t_1, ..., t_n)$ is associated the value true or false depending on whether $\langle v_1, ..., v_n \rangle \in I(P)$, where $v_1, ..., v_n$ are the evaluation of the terms $t_1, ..., t_n$ and $I(P)$ is the interpretation of P, which by assumption is a subset of Dn.

2. **Atomic formulas (2).** A formula $t1 = t2$ is assigned true if $t1$ and $t2$ evaluate to the same object of the domain of discourse.

3. **Logical connectives.** A formula in the form $\neg\phi, \phi \rightarrow \psi$, etc. is evaluated according to the truth table for the connective in question, as in propositional logic.

4. **Existential quantifiers.** A formula $\exists x \phi(x)$ is true according to M and α if there exists an evaluation α' of the variables that only differs from α regarding the evaluation of x and such that φ is true according to the interpretation M and the variable assignment α'. This formal definition captures the idea that $\exists x \phi(x)$ is true if and only if there is a way to choose a value for x such that $\varphi(x)$ is satisfied.

5. **Universal quantifiers.** A formula $\forall x \phi(x)$ is true according to M and α if $\varphi(x)$ is true for every pair composed by the interpretation M and some variable assignment α' that differs from α only on the value of x. This captures the idea that $\forall x \phi(x)$ is true if every possible choice of a value for x causes $\varphi(x)$ to be true.

If a formula does not contain free variables, and so is a sentence, then the initial variable assignment does not affect its truth value. In other words, a sentence is true according to M and α if and only if it is true according to M and any other variable assignment α'.

Evaluation process

More formally, given an alphabet of predicates and functions (constants will be assimilated to 0-arity functions, Λ being in this case a pseudo-function), each with its respective arity, $P_1, ..., P_i, ..., f_1, ..., f_i, ...$, an interpretation of our first order language is

$$I = (\Delta^I, P_1^I, ..., P_i^I, ..., f_1^I, ..., f_i^I, ...) \tag{1}$$

where:
- Δ^I is the domain (set of objects)
- if P_i is a k-arity predicate, then $P_i^I \subseteq \Delta^I \times ... \times \Delta^I$ (k times)
- if f_i is a k-arity function, then $f_i^I : \Delta^I \times ... \times \Delta^I \longrightarrow \Delta^I$ (k times)
- if f_i is a constant or a pseudo-constant(i.e., 0-arity function), then $f_i^I : () \longrightarrow \Delta^I$ (i.e., denotes exactly one object or the pseudo-object "void", "relative nothing" of the domain)

Let $Vars$ be a set of individual variables, then given an interpretation I, an assignment is a function
$\alpha : Vars \longrightarrow \Delta^I$
that assigns to each variable $x \in Vars$ an object $\alpha(x) \in \Delta^I$.
We can extend the notion of assignment to terms by defining a function $\overline{\alpha} : Terms \longrightarrow \Delta^I$ inductively :
- $\overline{\alpha}(x) = \alpha(x)$, if $x \in Vars$
- $\overline{\alpha}(f(t_1, ..., t_k)) = f^I(\overline{\alpha}(t_1), ..., \overline{\alpha}(t_k))$

In any case, the instanciation of a variable by Lambda must be taken into account.

For constants $\overline{\alpha}(c) = c^I$.
We say that a formula ϕ is true in an interpretation I and an assignment α, writen $I, \alpha \models \phi$:

- $I, \alpha \models P(t_1, ..., t_k)$ if $(\overline{\alpha}(t_1), ..., \overline{\alpha}(t_k)) \in P^I$;
- $I, \alpha \models \neg\phi$ if $I, \alpha \not\models \phi$;
- $I, \alpha \models \phi \wedge \psi$ if $I, \alpha \models \phi$ and $I, \alpha \models \psi$;
- $I, \alpha \models \phi \vee \psi$ if $I, \alpha \models \phi$ or $I, \alpha \models \psi$;
- $I, \alpha \models \phi \Rightarrow \psi$ if $I, \alpha \models \phi$ implies $I, \alpha \models \psi$;
- $I, \alpha \models \exists x\phi$ if for some $a \in \Delta^I$, we have $I, \alpha[x \mapsto a] \models \phi$;
- $I, \alpha \models \forall x\phi$ if for every $a \in \Delta^I$, we have $I, \alpha[x \mapsto a] \models \phi$.

A formula ϕ that is not true is either false or empty. In this article, empty is equivalent to false.

Note that $\alpha[x \mapsto a]$ stands for the new assignment obtained from α as follows :

$\alpha[x \mapsto a](x) = a$
$\alpha[y \mapsto a](y) = \alpha(y)$ with $(y \neq x)$

for constants, $\overline{\alpha}(c) = c^I$.

Closed formulas For closed formulas, or sentences, we can straightforwardly define what it means for ϕ to be true in an interpretation, written $I \models \phi$, without mentioning the assignment, since the assignment α does not play any role in verifying $I, \alpha \models \phi$.

Open formulas Open formulas are strongly related to queries. Let ϕ be a FOL query with free variables $(x_1, ..., x_k)$, sometimes written $\phi(x_1, ..., x_k)$; given an interpretation I, the needed assignments are those that map the variables $x_1, ..., x_k$, and only those. One will sometimes write such assignment explicitly: i.e., $\alpha(x_i) = a_i (i = 1, ..., k)$, is written simply as $\langle a_1, ..., a_k \rangle$. Now we define the answer to a query $\phi(x_1, ..., x_k)$ as follows : $\phi(x_1, ..., x_k)^I = \{(a_1, ..., a_k) | I, \langle a_1, ..., a_k \rangle \models \phi(x_1, ..., x_k)\}$.
We will also use the notation: ϕ^I, keeping the free variables implicit, and $\phi(I)$ making apparent that ϕ becomes a function from interpretation to set of tuples.

FOL boolean query A FOL boolean query is a FOL query without free variables. Hence the answer to a boolean query $\phi()$ is as follows

$\phi()^I = \{() | I, \langle\rangle \models \phi()\}$.

Such an answer is $\langle\rangle$ if $I \models \phi$ and $\emptyset$ if $I\neg \models \phi$. As an obvious convention we read $\langle\rangle$ as true and $\emptyset$ as false.

3.3.4 Validity, satisfiability, and logical consequence

If a sentence ϕ evaluates to True under a given interpretation M, one says that M satisfies ϕ; this is denoted $M \models \phi$. A sentence is satisfiable if there is some interpretation under which it is true.
Satisfiability of formulas with free variables is more complicated, because an interpretation on its own does not determine the truth value of such a formula. The most common convention is that a formula with free variables is said to be satisfied by an interpretation if the formula remains true regardless which individuals from the domain of discourse are assigned to its free variables. This has the same effect as saying that a formula is satisfied if and only if its universal closure is satisfied.
A formula is logically valid (or simply valid) if it is true in every interpretation. These formulas play a role similar to tautologies in propositional logic.
A formula ϕ is a logical consequence of a formula ψ if every interpretation that makes ψ true also makes ϕ true. In this case one says that ϕ is logically implied by ψ. More formally:

- Satisfiability : ϕ is satisfiable iff there is an I and α such that $I, \alpha \models \phi$; unsatisfiable otherwise; and if ϕ is satisfiable, then $\phi \neq \emptyset$, i.e. the query returns some tuples.
- Validity : ϕ is valid iff for all I and α, we have $I, \alpha \models \phi$; and if ϕ is valid, then $\phi^I = \Delta^I X ... X \Delta^I$, i.e., the query returns all the tuples of I.
- Logical implication : ϕ logically implies ψ, written $\phi \models \psi$ iff for all I and α, if $I, \alpha \models \phi$, then $I, \alpha \models \psi$; and if ϕ logically implies ψ, then $\phi^I \subseteq \psi^I$ for all I, written $\phi \subseteq \psi$, i.e., the answer to ϕ is contained in that of ψ in every interpretation; this is called query containment.
- Logical equivalence : ϕ is logically equivalent to ψ, iff for all I and α, $I, \alpha \models \phi$ iff $I, \alpha \models \psi$, i.e. $\phi \models \psi$ and $\psi \models \phi$; and if ϕ is logically equivalent to ψ, then $\phi^I = \psi^I$ for all I, written $\phi = \psi$, i.e., the answer to the two queries is the same in every interpretation. This is called query equivalence and corresponds to query containment in both directions.

There are analogous tasks if we have axioms, i.e., constraints on the admissible interpretations.

3.3.5 First-order theories, models and elementary classes

A first-order theory consists in a set of axioms in a particular first-order signature. The set of axioms is often finite or recursively enumerable, in which case the theory is called **effective**. Some authors require theories to also include all logical consequences of the axioms.
A first-order structure that satisfies all sentences in a given theory is said to be a **model** of the theory. An **elementary class** is the set of all structures satisfying a particular theory. These classes are a main subject of study in model theory.
Many theories have an intended interpretation, a certain model that is kept in mind when studying the theory. For example, the intended interpretation of Peano arithmetic consists of the usual natural numbers with their usual operations. However, the LöwenheimSkolem theorem shows that most first-order theories will also have other, nonstandard models.
A theory is **consistent** if it is not possible to prove a contradiction from the axioms of the theory. A theory is **complete** if, for every formula in its signature, either that formula or its negation is a logical consequence of the axioms of the theory.
Gödel's incompleteness theorem shows that effective first-order theories that include a sufficient portion of the theory of the natural numbers can never be both consistent and complete.

3.4 Lambda-1 calculus

Classicaly, first-order predicate calculi properly extend propositional calculus[j]. Within the Lambda theory, this is not really the case. First-order calculus here extends Lambda-0 calculus, which itself extends propositional calculus. But as we have seen, Lambda-0 calculus is empty. In general, propositional calculus is the calculus of reference for any language. If a propositional calculus is defined

[j]For simplicity, by a predicate calculus we always mean one that is sound and complete with respect to classical model theory.

with a suitable set of axioms (or axiom schemata) and modus ponens as single rule of inference (this can be done in many ways), then a predicate calculus can be defined from it by adding the "universal generalization" inference rule. It will be the case for our Lambda-1 calculus as it was the case for our Lambda-0 calculus, which also is a predicate calculus but of order 0! As axioms and rules for the predicate calculus with equality we take:
- laws of thought: identity, non-contradiction, excluded middle
- all tautologies of the propositional calculus;
- the quantifier axioms, given above;
- the above axioms for equality;
- modus ponens;
- universal generalization.
- It is necessary to mention that an expression stating that a set belongs to Λ is not legitimate: $\forall x \forall y (x \in y \Rightarrow (x \neq \Lambda \wedge y \neq \Lambda) \vee (x = \Lambda \wedge y \neq \Lambda))$. An expression such that $x \in \Lambda$ with $x \neq \Lambda$ is not a wff. The expression $\Lambda \in \Lambda$ is empty!
As "PC" is generally reserved for Propositional Calculus, 'quantificational calculus' would be a more appropriate label for our Lambda-1 calculus. A sentence is defined to be provable (demonstrable) in the calculus if it can be derived from the axioms of the predicate calculus by repeatedly applying its inference rules. In other words: all axioms of the calculus are provable (in the calculus); if the premises of an inference rule are provable, then so is the conclusion. If T is a set of formulas and f a single formula, we define a derivation of f from T (in the calculus), in symbols $T \vdash_Q C\varphi$ (we often omit the subscript), as a list of formulas $\varphi_1, ..., \varphi_n$ such that $\varphi_n = \varphi$ and each φ_i either

(i) is an axiom; or
(ii) follows from previous φ_j, φ_k (possibly $j = k$) by a rule of inference.
If $T \vdash \varphi$ then for some finite $T' \subseteq T$ we have $T' \vdash \varphi$. The fact that a sentence is always provable from a finite set of sentences, - if it is provable from any set at all -, is a consequence of the fact that every derivation in the system is a finite list of formulas. Notice that provability is a special case of derivability from the empty set of premises. In this sense, each calculus K gives rise to a derivability relation $\vdash_K$. Since we are taking 'predicate calculus' to mean one that is sound and complete with respect to classical model theory, each calculus gives rise to the same derivability relation (taken extensionally).

3.4.1 Provable identities

$\neg \forall x P(x) \Leftrightarrow \exists x \neg P(x)$
$\neg \exists x P(x) \Leftrightarrow \forall x \neg P(x)$
$\forall x \forall y P(x, y) \Leftrightarrow \forall y \forall x P(x, y)$
$\exists x \exists y P(x, y) \Leftrightarrow \exists y \exists x P(x, y)$
$\forall x P(x) \wedge \forall x Q(x) \Leftrightarrow \forall x (P(x) \wedge Q(x))$
$\exists x P(x) \vee \exists x Q(x) \Leftrightarrow \exists x (P(x) \vee Q(x))$
$P \wedge \exists x Q(x) \Leftrightarrow \exists x (P \wedge Q(x))$ (where x must not occur free in P)
$P \vee \forall x Q(x) \Leftrightarrow \forall x (P \vee Q(x))$ (where x must not occur free in P)

3.4.2 Additional inference rules

With respect to deduction, we will use the following inference rules:
- modus ponens
- universal instantiation
- existential instantiation
- universal generalization
- existential generalization

3.4.3 Axioms of the theory

In set theory defined in this article, the cuts take the form of the void.
As we have said in the introduction, the void, the potential, in the sense of an abstract vacant space, is a pre-element and constitutes the only non-element of our theory. Denoting a vacant space, the void, assimilated to a cut, represents a condition of possibility of the elements of the theory.

Internal and external cut
With respect to set theory, the void-potential, as an internal and an external cut,[k] is a condition of possibility of sets. As an internal cut, it makes it possible for a set to contain elements; as an external cut, it makes it possible to distinguish two different sets.[l] Without the internal cut that the void represents, a set would be an atom, an ur-element. Now, the void is a condition of possibility of an ur-element too, but as an external cut only. The void belongs to the empty set and to any nonempty set ($\forall x(x \neq \Lambda \Rightarrow \Lambda \in x)$).
This is the fundamental axiom of Λ-1 theory. This void, denoted by Lambda, is thus not a set but it is not the absolute nothing either.[m] Its potential aspect distinguishes it from the absolute void. As a potential, the void of the Lambda theory is a relative void; the absolute void is that towards which the relative void tends.[n]

Axiom of the void
The fundamental axiom of Lambda-1 theory is the axiom of the void or of the potential or of the condition or of the pre-element:
Axiom of pre-element: $\forall x(x \neq \Lambda \Rightarrow \Lambda \in x)$.
In particular, $\Lambda \in \emptyset \subseteq x$.
This expresses the idea that Lambda is not a set and will allow to give a positive definition of the empty set, as we will see in the next section.
Lambda is the only thing for which it is not possible not to belong to any set precisely because it is not a thing; it is a *sub-thing*, a pre-element, a potential, contrary to ur-elements, for example, which are real things, elements, even if of

[k] Similarities and differences with Dedekind cuts will be studied in another paper.

[l] Every element can play the role of a cut compared to the other elements, from which it is different, but then it is a 1^{st} order cut. Lambda constitutes a zero-order cut because it denotes a potential, the condition of possibility of the elements.

[m] While the empty set is not nothing, neither absolute nor relative, Lambda does not denote the absolute nothing, but rather a relative nothing.

[n] The assimilation of the void to a potential naturally leads to the elaboration of a logic of the potential. This is the subject of a further article.

a type/nature different from that of sets.

Additional axioms
axiom of Extensionality: $\forall x \forall y(\forall z(z \in x \Leftrightarrow z \in y) \Rightarrow x = y)$,
axiom of Pairing: $\forall x \forall y(\exists z(x \in z \wedge y \in z)$,
axiom Schema of Separation: $\forall X \forall p \exists Y \forall u(u \in Y \Leftrightarrow u \in X \wedge \varphi(u, p))$,
axiom of Union: $\forall X \exists Y \forall u(u \in Y \Leftrightarrow \exists z(z \in X \wedge u \in z))$,
axiom of Power Set: $\forall x \exists y(\forall z(\forall t(t \in z \Rightarrow t \in x) \Rightarrow z \in y))$,
axiom of Infinity: $\exists S(\emptyset \in S \wedge \forall x(x \in S \Rightarrow x \cup x \in S))$,
axiom Schema of Replacement, $\forall x \forall y \forall z(\varphi(x, y, p) \wedge \varphi(x, z, p) \Rightarrow y = z) \Rightarrow \forall X \exists Y \forall y(y \in Y \Leftrightarrow (\exists x \in X)\varphi(x, y, p))$,
axiom of Regularity: $\forall x(\exists y(y \in x \wedge \neg \exists z(z \in y \wedge z \in x)))$,
axiom of Choice: $\forall X(X \neq \emptyset \wedge \emptyset \notin X) \Rightarrow (\exists f : X \rightarrow \cup_{Y \in X} Y, \forall x \in X, f(x) \in x)$.

4 The Empty Set as Power set of Lambda

The most remarkable result of the use of the void and the description of the zero-order logic in a set theory context is the construction of the empty set by means of the axiom of the parts applied to Lambda.

4.1 *Empty Set Theorem*

***Theorem* 1.** $\emptyset = \wp(\Lambda)$

Proof. By the axiom of the ante-element, Lambda belongs to every set: $\forall x(x$ is a set $\Rightarrow \Lambda \in x)$, in particular $\Lambda \in \emptyset$. Lambda is thus not a set: $\forall x(x$ is a set $\Rightarrow \Lambda \neq x)$. So no set is included in the void, since the void is not a set: $\neg \exists x(x \subseteq \Lambda)$, which involves that : $\neg(\emptyset \subseteq \Lambda)$; consequently, Lambda does not have parts and the set of its parts is the empty set : $\emptyset = \wp(\Lambda)$ □

Lambda has no property of set and most Λ-formulas applying to sets will be false, but the axiom of the parts, which is, with the axiom of infinity, the only constructive axiom, and which describes an overset, can be applied to Lambda as it can be applied to second order elements in the frame of a second-order theory, and of course to first-order elements. And this is made possible by the zero-order logic that allows to consider Λ as a legitimate instanciation of a variable x in a quantified formula; Lambda denotes the pseudo-object "void", the sub-thing "relative nothing" or "potential".
Let's elaborate this. The axiom of the parts says: $\forall x \exists y(\forall z((z \subseteq x) \Rightarrow z \in y))$. More rigorously, $\forall x \exists y(\forall z(\forall t(t \in z \Rightarrow t \in x) \Rightarrow z \in y))$. If we instantiate Λ for x, there would be a priori only two possibilities of instantiation for z: Lambda itself and the empty set. If we consider $z = \varnothing$, we have for $\Lambda, \exists y($for $\varnothing(\Lambda \in \varnothing \Rightarrow \Lambda \in \Lambda) \Rightarrow \varnothing \in y)$. But it is not possible because Lambda is not a set; so nothing is included in Lambda. In any case, $\Lambda \in \Lambda$ is empty and makes the implication empty. So, for $\Lambda, \exists y(\Lambda)$, and $\wp(\Lambda) = \emptyset$; if $z = \Lambda$, we have $(\Lambda \in \Lambda \Rightarrow \Lambda \in \Lambda) \Rightarrow \Lambda \in y$.

The axiom of the empty set, such as found, for example, in ZF, is no longer necessary. It can be reformulated in a positive way: $\exists x(x$ is a set $\wedge \forall y(y \in x \Rightarrow$

$y = \Lambda)$) or $\exists x(x \neq \Lambda \wedge \forall y(y \in x \Rightarrow y = \Lambda))$. In set notation: $\emptyset = \{y : y = \Lambda\}$. No use of a contradictory property. The empty set is the only set that contains potential only. Other sets contain potential and elements.

Lambda is not included in Lambda, but by the axiom of the ante-element, it belongs to any set.
Once again, thanks to the zero-order logic, the sub-object "void", "potential", denoted by Lambda, is taken into account in the interpretation of the formulas. The objection according to which the set of the parts of, for example, an ur-element, is empty, doesn't hold because an ur-element, contrary to Lambda, doesn't belong in a natural way to the set theory and to our theory. Thus, even if the zero-order logic is a trivial logic, it leads to a remarkable result since it makes it possible to build in a positive way the empty set.[o]
From there, by means of the axioms of the ZFC theory, the Lambda theory allows the construction of the structure $\langle \Lambda, X, \in \rangle$.

4.2 *Kernel-Structure*

The cumulative hierarchy is a collection of sets V_α indexed by the class of ordinal numbers, in particular, V_α is the set of all sets having ranks less than α. Thus there is one set V_α for each ordinal number α; V_α may be defined by transfinite recursion as follows:

- Let V_0 be $\wp(\Lambda) = \emptyset$.
- For any ordinal number β, let $V_{\beta+1}$ be the power set of V_β.
- For any limit ordinal λ, let V_λ be the union of all the V-stages so far: $V_\lambda := \bigcup_{\beta<\lambda} V_\beta$.

The class V is defined to be the union of all the V-stages: $V := \bigcup_\alpha V_\alpha$.

5 Conclusion

5.1 Synthesis

"*Nothing*" is added to set theory. The play on words means this: the introduction of the pseudo-constant Lambda denoting the "nothing", the void, the potential, in the language of set theory does not imply that some thing is added to the theory. The nothing, the void is subjacent to the standard set theory as a sub-thing, a pre-element; this is a sub-, a pseudo-object! According to the context, the void will have a passive or an active role. In a set theory with an empty universe, the void has a passive role; it is just a potential. Let us note that the domain is empty with regard to the classical theory. In the frame of the Lambda-0 theory, the domain is *relatively* empty since it contains the *potential* as sub-/pre-object. In a set theory with a non-empty universe, the void has an active role; it is not only a potential, but a cut, a condition of possibility of the elements of the theory. The void as cut has two functions: as an internal cut, it makes it possible for a set to contain elements, it is the fundamental constituent

[o] If the empty set, which is the simplest set and the only set included in every set, is not included in Lambda, Lambda cannot be a set.

of any set, this is what is expressed by the axiom of the ante-element; as an external cut, it makes it possible to have different sets; this is what can be more specifically studied by formal ontology. So the content's distinction between sets is made possible by the more fundamental cuts. Cuts are so fundamental that they constitute a condition of possibility of the language itself, enabling to distinguish between letters of the alphabet, and between more complex elements of the language; but this is a matter for linguistics. To take into account the void in the language of a set theory with an empty universe made it possible to highlight a zero-order logic different from the propositional logic. Contrary to what is traditionnally alleged, quantification is possible in zero-order logic. We quantify on Lambda, the 0-order symbol denoting the pre-element "void". The semantics of the language based on this zero-order logic is empty. Deprived of the classical truth values true and false, it stays on the pre-value Empty. As for the Lambda-1 theory, it is satisfied with a standard logic if restrictions on instanciations of variables are put; but no need of a trivalent or a paracomplete logic! The prevalue Empty can be assimilated to False. Empty formulas will be interpreted as false formulas. Now, one could consider a bivalent logic where the values would be *Empty* and *True* (non-Empty). In this case, false formula would be assimilated to empty formula. To take into account the void in the language of a set theory with a non-empty universe made it possible to build the empty set by means of the axiom of the parts applied to Lambda: $\emptyset = \wp(\Lambda)$. As a pre-element, the case of Lambda must be considered without requiring the introduction of a general variable into the language of the theory; it is what the zero-order logic allows. According to us, the notion of "set" acquires a real ontological dimension. The Lambda theory legitimates and gives an emblematic status to the empty set, now defined in a positive way, as the only set that contains only potential, and makes it possible to redefine the concept of set: a set is the expression of a potential. The Lambda theory also makes useless the axiom of the empty set or the construction of the empty set by means of a contradictory property. "Something", or rather a *pseudo-thing* belongs to the empty set: the void, and the empty set is the only set to which belongs Lambda only. The Lambda theory allows us to distinguish the void from the empty set, solving what can be called the puzzle of Lambda as found in Russell. It also allows us to distinguish the empty set from ur-elements, which are generally considered as kinds of empty sets. No thing belongs to an ur-element, even not Lambda, the *relative nothing*. This is why a ur-element is a kind of atom.

5.2 Prospects

So one can see that not only does the empty set exist, but that the void itself is not exactly *nothing*: it is a relative nothing that constitutes a potential, and as such a *transcendental* (pre-)element. Thus the Lambda theory makes it possible to modify the opinion of the greek philosopher Parmenides who could only bring himself to believe that there is but one way of not being; there are at least two : the absolute void and the relative void, the absolute void being that towards which the relative void tends. Finally, we will see in another paper that the anomaly of the intersection of an empty collection is solved thanks to Lambda; it is no longer necessary to restrict the intersection to non empty

collections.[p]
For the theory, the balance in terms of investment and profits is clearly positive: the investment is *quasi-nul*, the gains numerous.

References

[1] A.N. Whitehead, B. Russell. *The Principia Mathematica.* Éd. W. W. Norton and Company, England, (1996).

[2] P. Halmos. *Naive set theory.* Éd. Princeton, NJ: D. Van Nostrand Company, 1960. Reprinted by Springer-Verlag, New York, (1974).

[3] T. Jech. *Set Theory, The Third Millennium Edition, revised and expanded.* Éd. Springer-Verlag, Berlin Heildelberg New-York, (2006).

[4] J. Lowe. *Ontological dependence.* http://plato.stanford.edu/entries/dependence-ontological/ (2005).

[5] B. Russell. *The Principles of Mathematics.* Éd. W. W. Norton and Company, England, (1996).

[6] G. Sher and R. Tieszen. *Between logic and intuition: essays in honor of charles parsons.* Éd. Sher-Tieszen, (2000).

[7] D. Vernant. *Sur les fondements de la mathématique de Stanislaw Lesniewski, Étude critique.* http://web.upmf-grenoble.fr/SH/PersoPhilo/DenisVernant/Lesniewski.pdf (2007).

[p]The paradox of the empty set as universe disappears.