A Topos-Based Approach to Building Language Ontologies

William Babonnaud

LORIA, Université de Lorraine, CNRS, INRIA Nancy Grand Est, Nancy, France
william.babonnaud@loria.fr

Abstract. A common tendency in lexical semantics is to assume the existence of a hierarchy of types for fine-grained analyses of semantic phenomena. This paper provides a formal account of the existence of such a structure. A type system based on the categorical notion of topos is introduced, and is shown to be possibly adaptable to several existing formal approaches where such hierarchies are used. A refinement of the type hierarchy based on Fred Sommers' ontological theory is also proposed.

Keywords: Formal semantics · Lexical semantics · Type theory · Type ontology · Category theory

1 Introduction

The work presented in this paper originates from a very general question: *what is a semantic type?* Although a definitive answer to this complex, almost philosophical question will not be provided here, it is still possible to look for a better understanding of the role of semantic types in formal and lexical semantics by browsing the history of their use in computational linguistics. If this field was to be described as a meeting of computer science and linguistics, unsurprisingly there would be various contributions to the notion of semantic type from each discipline; and indeed it will be argued that this notion seems to sum up contributions of different origins which happened to work well together.

The notion of type has been ubiquitous in logics and computer science since the early works of Russell, which paved the way for the simple type theory of Church [7]. This theory was then introduced in computational linguistics by the founding work of Montague [14], who merely used it as a formal tool to embed the Fregean view of predicates as functions into his grammar. Thus he provided a formal basis for the *principle of compositionality*, which states that the meaning of an expression can be derived from the meanings of its constituent parts and from the way these parts are syntactically combined. The type system used in this approach is a very minimalist one, directly inherited from Church's theory, with a type e for entities and a type t for propositions.[1]

[1] Which correspond respectively to ι and o in Church's notation. Although it is worth studying, the additional s type, denoting intension, will not be discussed in this

Following Montague's idea, those types can be given a practical interpretation: t denotes everything that can have a truth value, and e denotes individuals, which could be understood as the more basic expressions which can be used as arguments to another expression. This approach has been widely adopted as the foundation of formal semantics, and even if Montague's categories have been separated between syntactic and semantic ones, the use of a type for truth values and at least another one for predicate arguments, as well as the correspondence between syntactic categories and semantic types, are well-established.

However, a relatively recent tendency in computational semantics is to postulate a whole hierarchy of types of arguments, instead of a single one for entities, in order to account for specific semantic phenomena in a more fine-grained way. Such a hierarchy is intended to encode hypernymic relations between nouns, that is, inclusions between the "sets" (in a non-mathematical acceptance of the word) of entities satisfying the properties described by those nouns. Constructing hypernymic relations is exactly what we do when we make generalising statements, for instance:

(1) a. A dog is a mammal.
    b. A mammal is an animal.
    c. An animal is a living entity.

The type hierarchy thus reflects a part of common (yet *a priori* language-dependent) lexical knowledge; and far from being an abstract construction, such a structure can be acquired on the basis of an extensive empirical analysis as is done in the case of building hypernymic and hyponymic relations of synsets in the WordNet database [13].[2] This structure is at the core of lexical semantics, as it is generally assumed whenever type ontologies are needed. The minimal use of such a hierarchy enables one to formally encode the semantic presuppositions of a given entity, or the ones corresponding to the argument of another predicate.

Prior to the systematisation of this idea, philosophers, psychologists, and linguists came up with the notion of *semantic category*, which can arguably be seen as the conceptual ancestor of the current understanding of the notion of type. This notion arose notably in the early work of Husserl, with certain influence on the Polish school through Leśniewski and Ajdukiewicz. Yet to the best of our knowledge, one of the first proposals for a hierarchical structure of such categories was sketched by Chomsky [6]. His categories organised expressions in a tree structure which aimed at allowing for degrees in grammaticality judgements. Kiefer [10] elaborated on this idea while separating grammatical and semantic categories, and ordered the latter by means of so-called *esse-relations*, that is, categorial inclusions as in (1). It is also worth noticing that a similar notion

paper, for it is assumed to be a necessary feature for the set-theoretic models of the logic used by Montague. The conception of types as denotations of sets will be overlooked here as it is too specific.

[2] See the website `https://wordnet.princeton.edu/`. The online application provides the opportunity to explore the hierarchy by browsing a word, selecting its synset and accessing the list of hyponyms and hypernyms.

of semantic category is used in the field of psycholinguistics, starting with the early work of Wittgenstein [25] on the process of categorisation, which has led to relevant theories on language acquisition [4] and the concept of prototype [17,8], among others.

Another major contribution comes from the philosophical work of Fred Sommers. For a dozen years he has been interested in language ontologies and elaborated a complete theory of ontology through a series of articles (see [20,21,22] among others for a partial, yet wide-covering compendium). He examined Russell's notion of type as well as Ryle's idea of *category mistake* to propose a theory based on the question of predication. The core of this theory lies in the notion of *spanning* : a predicate is said to span an entity when it is predicable truly or falsely to this entity, but not absurdly. For instance, the predicate 'philosopher' spans Socrates and Julius Caesar, but not the Eiffel tower, because it is absurd to wonder whether the Eiffel tower is a philosopher or not, as it cannot be decided. Thus, Sommers rejected the Quinean approach of considering every absurd sentence as logically false. Instead, he proposed an analysis of the correctness of a sentence on different levels: on the first level is the question of "grammaticality", that is, syntactic well-formedness; on the second level is "category correctness", i.e. whether the predicates in the sentence are applied in a non-absurd way, without category mistakes in the sense of Ryle; and then, the third level is the question of "consistence", i.e. whether the sentence avoid contradiction or not. The fourth level consists eventually in determining the empirical truth of the sentence w.r.t. the context. Each level of analysis can be treated only under the condition that the sentence was correct for the previous levels; which means that we can give a truth value to a propositional sentence only if it has no category mistake.

Sommers' idea was then to use the predicates to recover the Russellian notion of type: considering a predicate P in the language, he defines $|P|$ to be the class of things that are spanned by P. Such classes are referred to as ontological classes, or *categories*. It appears then that different predicates can define the same categories, as for instance 'sad' and 'angry'; he also defined types of predicates, or *A-types*, so that two predicates are of the same A-type if they define the same category. Ontology being "the science of categories", Sommers' definitions create a language ontology comparable to a skeleton of the language (seen as the collection of its predicates), by quotienting it by the relation of "defining the same categories." But at this stage the onlotogy lacks structure. Sommers remedied this by formulating a structural principle:

> "If C_1 and C_2 are any two categories, then either C_1 and C_2 have no members in common or C_1 is included in C_2 or C_2 is included in C_1."[21, p. 355]

A lot of Sommers' works, notably in [20], aimed at establishing this law. It has many important consequences w.r.t. the ontology itself, as well as on some metaphysical questions. Inclusions of categories can be presented from the predicative point of view by defining the relation of *predicability* : P is predicable

of Q if anything that is spanned by Q is also spanned by P. Then the isomorphism between predicability and category inclusion is quite clear. Sommers also noticed that the vocabulary of any language is finite, and so is the number of categories we can define in its ontology. Thanks to finiteness and the structural principle, Sommers eventually showed that there is a category which includes all categories, and that there are categories which do not include any other. Thus, he gives to his ontology the shape of a tree, which is similar to the current type hierarchies. To make sure that this structure is correct, he also provided a way to handle equivocal predicates, such as 'hard' in (2), by splitting their different meanings in appropriate places on the tree.

(2) a. The chair is hard.
b. The question is hard.

Yet having a tree structure is a stronger assumption than just being a lattice, and Suzman [23] pointed out the idea that Sommers' ontology fails to account for entities that could be seen as being in the intersection of categories which are not included one in the other. Actually, Sommers discussed this point in [22] and admitted that imposing a tree structure on his ontology makes some entities lie outside of it. Amongst those entities are the absurd ones, such as 'red numbers', but there is also another sort which is worth dwelling upon: those entities which Sommers called *heterotypical composites*, and which are built upon two or more elementary categories from the ontology. The word 'Italy' in sentence (3a) is an example of such a composite. Heterotypical composites differ from equivocal predicates in the fact that contrary to the latter, there is no zeugmatic effect when combining simultaneously the former with two predicates of different A-types, as illustrated by comparing sentences in (3) below:

(3) a. Italy is sunny and democratic.
b. #The chair and the question are hard.

Thus Sommers proposed a way for catching any expression in its ontology. The influence of his work on the current understanding of type hierarchies remains unclear, but in any case, he proposed a powerful and elegant ontological theory which is worth shedding light on.

This sketch of history helps to draw an overview of some influences on the notion of type: on one hand, formal semantics needs a type of truth values and functional types in order to compose the lexical units properly, and on the other hand lexical semantics needs a structured ontology of types to get a more refined analysis of semantic phenomena. The combination of both approaches leads us to a rather standard approach in computational semantics: a type system *à la* Montague where the type e has been replaced by more precise types from an ontological hierarchy. Recent works using this kind of approach include Luo's theory of coercive subtyping (e.g. [12]), Pustejovsky and Asher's type composition logic (TCL) [2,1] and also Retoré's ΛTy_n framework [16].

The remainder of this article will sketch the basis of a type system where a hierarchy of entity types is properly integrated into the usual compositional

framework. This type system is intended to reflect the ontology of language, and rests upon a specific model from category theory, so that it should naturally lead to a λ-calculus. Although the complete calculus will not be defined here, a few lines of what properties can be expected from such a system will be given. Section 2 introduces necessary categorical tools for understanding topos theory; then the type system and its properties will be described in section 3. Finally, a short review of related works and concluding remarks will be given in section 4.

2 A Synopsis of Topos Theory

Category theory can be presented as a formalisation of mathematical structures, and is known to have strong connections with typed λ-calculi (see e.g. [3,19]). In this regard, a type system can be given a categorical[3] semantic model. How this kind of correspondence works will not be explored here. Rather, this section aims to give some basic tools to understand the notion of *topos*, which refers to categories with specific properties that will be used extensively in the rest of this paper. Only necessary notions will be introduced here for questions of space; more details can be found e.g. in Robert Goldblatt's book [9].

Definition 1. *A* category $\mathcal{C}$ *is a class of* objects $\mathcal{O}(\mathcal{C})$*, and for every pair of objects* $A, B \in \mathcal{O}(\mathcal{C})$ *a class of* morphisms *(or* arrows*)* $\mathcal{C}(A, B)$*, and a composition operator on morphisms* $\circ$ *such that:*

- *for all* $f \in \mathcal{C}(A, B)$ *and* $g \in \mathcal{C}(B, C)$*, there is a morphism* $g \circ f \in \mathcal{C}(A, C)$*;*
- *the composition is associative, i.e.* $h \circ (g \circ f) = (h \circ g) \circ f$*;*
- *for all object* A*, there is an identity morphism* $\mathrm{id}_A \in \mathcal{C}(A, A)$ *which is neutral for composition, i.e.* $g \circ \mathrm{id}_A = g$ *and* $\mathrm{id}_A \circ h = h$ *for all relevant* g *and* h*.*

A well-known example of a category is Set, whose objects are sets and whose morphisms are functions between sets, equipped with the usual notion of composition. The notation $f \in \mathcal{C}(A, B)$ will be replaced by $f : A \to B$ whenever $\mathcal{C}$ in understood. Elaborating from this definition, categories can be "enriched" by defining objects with specific properties (usually gathered under the term *universal property*). The prototypical objects with such properties are initial and terminal objects, as given in the following definition.

Definition 2. *A* terminal *(resp.* initial*) object in a category* $\mathcal{C}$ *is an object* 1 *(resp.* 0*) such that for all* $A \in \mathcal{O}(\mathcal{C})$*, there is exactly one morphism* $!_A : A \to 1$ *(resp.* $0_A : 0 \to A$*).*

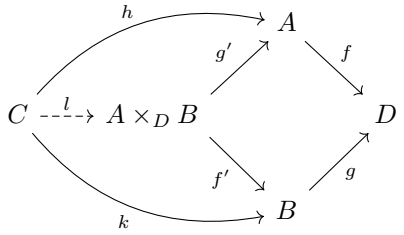
It is worth noticing that the objects introduced in this definition (and in subsequent ones as well) have the property to exist *up to isomorphism*. Two objects A and B are isomorphic (noted $A \cong B$) when there are morphisms

[3] As a convention, I will distinguish between the adjective *categorial* when talking about linguistics categories such as Chomsky's or Sommers', and the adjective *categorical* when talking about things from category theory.

$f : A \to B$ and $g : B \to A$ such that $g \circ f = \mathrm{id}_A$ and $f \circ g = \mathrm{id}_B$. Objects satisfying the universal property are generally not unique, but all of them can be proven to be isomorphic. Yet as this property is known it is common to refer to *the* terminal object instead of *a*, and similarly for other objects. In Set, any singleton is a terminal object, and the empty set is the initial object. Other relevant objects are products and pullbacks, which we define below. The second definition uses the notion of *diagram*, which can be understood as a representation of a small subpart of a category as a graph whose nodes are objects and edges are morphisms. A diagram is said to be *commutative* if all paths of morphism compositions between any two objects are equal.

Definition 3. *A* product *of two objects A and B is an object $A \times B$ equipped with two projections $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$, such that for all objects C and morphisms $f : C \to A$ and $g : C \to B$ there is a unique morphism $\langle f, g\rangle : C \to A \times B$ satisfying $\pi_1 \circ \langle f, g\rangle = f$ and $\pi_2 \circ \langle f, g\rangle = g$.*

Definition 4. *A* pullback *of two morphisms of the same codomain $f : A \to D$ and $g : B \to D$ in an object $A \times_D B$ equipped with two morphisms $f' : A \times_D B \to B$ and $g' : A \times_D B \to A$, such that for all objects C and morphisms $h : C \to A$ and $k : C \to B$ satisfying $f \circ h = g \circ k$, there is a unique morphism $l : C \to A \times_D B$ such that the following diagram commutes:*



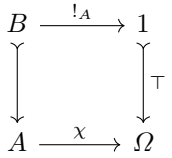
Another common kind of objects which appears in the definition of topoi are *exponentials*, but they will not be used in this paper, so mentioning their existence is enough. There is still a useful notion left, namely the *subobject classifier*. But first the notion of monomorphism should be defined:

Definition 5. *A morphism $f : A \to B$ is said to be a* monomorphism *(noted $f : A \rightarrowtail B$) if it is left-cancellable, i.e. if for any pair $g, g' : C \to A$ of arrows of codomain A, $f \circ g = f \circ g'$ implies $g = g'$.*

Monomorphism is actually a generalisation upon of notion of injective function, so that in Set monomorphisms are exactly those functions. Then, a *subobject* of an object A is an object B along with a monomorphism $B \rightarrowtail A$. Without loss of generality, it will be assumed that whenever B is a subobject of A, the associated monomorphism is unique. Then we can introduce properly the subobject classifier:

Definition 6. *A* subobject classifier *is an object Ω along with a monomorphism $\top : 1 \rightarrowtail \Omega$ such that, for all objects A and subobjects $B \rightarrowtail A$ of A, there is*

a unique morphism χ *(called the* character *of* B *in* A*) making the following diagram a pullback (i.e.* B *is the pullback object of* χ *and* $\top$*):*



The name *character* given to χ should help understand how the subobject classifier works: it represents an "object of truth values" and $\top$ corresponds to the value "true", so that any subobject can be associated with a morphism that distinguishes it by sending it on "true". In Set, any two-element set is a subobject classifier (again this notion is given up to isomorphism), and characters are exactly characteristic functions. Yet the previous definition imposes characters to be uniquely associated with subobjects, which is actually a very strong assumption. For any object A of a category $\mathcal{C}$ with subobject classifier, let call $\mathrm{Sub}(A)$ the class of subojects of A.[4] The definition above imposes then the following property, which is usually referred to as Ω*-axiom*:

Proposition 1 (Ω-axiom). *For all* $A \in \mathcal{O}(\mathcal{C})$*, we have* $\mathrm{Sub}(A) \cong \mathcal{C}(A, \Omega)$*.*

We have then all tools in hand for introducing the central categorical notion on which is based this paper:

Definition 7. *A* topos *is a category with initial and terminal objects, all products, all pullbacks, all exponentials, and a subobject classifier.*

The category Set is again the prototypical example of a topos, which satisfies two additional properties: it is *bivalent*, i.e. it has exactly two distinct morphisms $1 \to \Omega$, and it is *classical*, i.e. its subobject classifier is isomorphic to the coproduct $1 + 1$. The category Set^2 of pairs of sets is an example of classical but non-bivalent topos, while the category $\mathsf{Set}^{\to}$ whose objects are functions between sets is neither classical nor bivalent. Also, if M is a monoid which is not a group, then the category $\mathsf{M\text{-}Set}$ whose objects are sets together with an action of M on them is an example of non-classical but bivalent topos.

There are many other equivalent ways for defining a topos, but the one given above rests exclusively on the notions that will be used in the remainder of this paper. As a subobject classifier is an object of truth values, there are ways in a topos for introducing an internal logic: we can define morphisms $\bot : 1 \to \Omega$, $\neg : \Omega \to \Omega$ and $\wedge$, $\vee$, $\Rightarrow$ as morphisms $\Omega \times \Omega \to \Omega$. This internal logic can serve as semantic model for intuitionistic logic, see [9] for details. Using such morphisms with the Ω-axiom allows to define new kind of subobjects. If B and C are subobjects of A of character χ_B and χ_C, then $\bar{B}$, $B \cup C$, $B \cap C$ and $B \Rightarrow C$ are subobjects of A of respective characters $\neg \circ \chi_B$, $\vee \circ \langle \chi_B, \chi_C \rangle$, $\wedge \circ \langle \chi_B, \chi_C \rangle$ and $\Rightarrow \circ \langle \chi_B, \chi_C \rangle$. Those new operators have a particular behaviour in $\mathrm{Sub}(A)$:

[4] For linguistically motivated reasons, classes of subobjects of an object and classes of morphisms between two objects will be considered as sets in the rest of this paper.

Proposition 2. *In any topos,* $\langle \mathrm{Sub}(A), 0, A, \cap, \cup, \Rightarrow \rangle$ *is a Heyting algebra for the subobject ordering.*

Also, the exponential object of A and Ω will be noted $\mathscr{P}(A)$. It is referred to as *powerobject* of A, and corresponds to powersets in Set. It has also interesting properties; some of them will be introduced in due time in the next section.

3 Building a Topos-Based Type System

This work is mainly inspired by the proposition of categorical model for TCL sketched by Asher in [1]. As explained before, there is a strong relation between typed λ-calculi and categories: we can use objects of the category for representing types, and morphisms for λ-terms. This approach is also motivated by the potential usefulness of categorical models to ensure the consistency of a compositional framework. In [1], his high need of pullbacks and powerobjects leads Asher to propose topoi as type models. Prior to his approach, topoi had been used for linguistic and cognitive questions, for instance in [11].[5] The choice of topos as the categorical basis of this approach is actually motivated by theoretical and practical concerns: the necessity of truth values in language semantics and of subtyping in a language ontology naturally suggests the use of a subobject classifier, which also requires a terminal object; and in order to enable the definition of a typed λ-calculus from such a category, it has to be at least cartesian closed,[6] that is, to have products and exponentials. Finally, as the aim is ultimately to build logical formulæ for representing language semantics, we need an access to the usual quantifiers; and despite the subtleties that rule the distinction between the internal and external logics of the category, it is easier for now to add all pullbacks in it. The combination of these various requirements meets the definition 7, so that topos is the minimal categorical structure needed for our purposes.

Building up on this idea, the theory developed here takes a closer look at how the properties of topoi can lead to a new type system for natural language semantics. More precisely, a specific instance of a topos will be introduced, and some properties it has to satisfy in order to define a type ontology for natural language will be described. It will also be argued that the type system thus created shares many properties with Sommers' propositions, which is an originally unexpected but welcome result.

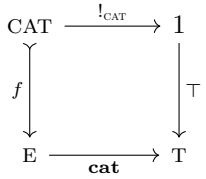
3.1 From Montague's *e* to a Hierarchy of Types

Let $\mathcal{T}$ be a topos. Following the general idea of objects as types, we would like $\mathcal{T}$ to have at least the two Montagovian types e and t. The key to this model is the way we handle the type of truth values: indeed, by definition $\mathcal{T}$ has an

[5] I am grateful to an anonymous reviewer for bringing this work to my attention.

[6] Actually, a monoidal closed category would suffice if we wanted a linear λ-calculus, but such a restriction is not justified here.

object which corresponds exactly to what this type is supposed to be, namely the subobject classifier. Let therefore T be the subobject classifier of $\mathcal{T}$, and let E be a distinguished object of the topos. Following Montague [14], the first-order monadic properties (including nouns) are to be considered as terms of type $\mathrm{E} \to \mathrm{T}$. Put in $\mathcal{T}$ those terms correspond to morphisms, so that each first-order monadic predicate in the language has a counterpart in $\mathcal{T}(\mathrm{E}, \mathrm{T})$, and conversely. Recalling the Ω-axiom, we get for each predicate a subobject of E. For instance, the predicate 'cat' defines the morphism $\mathbf{cat} : \mathrm{E} \to \mathrm{T}$, and enforces by axiom the existence of a subobject CAT of E, so that the following diagram is a pullback:



As we assimilate objects of the topos and semantic types, the type CAT thus defined is exactly what we would expect: the type of entities that are actually cats, that is, those entities that are true of the predicate 'cat'. In our topos, this can be established by the equality of morphism compositions in the diagram above. For any object A of $\mathcal{T}$, define $\mathrm{true}_A : A \to \mathrm{T}$ to be $\top \circ !_A$.[7] Then, the pullback above gives us the following equality:

(4) $\quad \mathbf{cat} \circ f = \mathrm{true}_{\mathrm{CAT}}$

We can even do better if we introduce the notion of *global elements*. A global element of an object A in any category is a morphism $1 \to A$. Thus, the morphisms $\top$ and $\bot$ in our topos are global elements of T. Note that contrary to the unique morphism of codomain 1 for any object, there is no reason that global elements exist in general. However, suppose that there is a global element $x : 1 \to \mathrm{CAT}$ in our previous example. The properties of the terminal object ensure that $!_{\mathrm{CAT}} \circ x = \mathrm{id}_1$. So, by composing x with each side of the equality in (4), we get the one in (5), which goes closer to the idea we could have of a predicate calculus based on this system, as it exactly states that applying the predicate **cat** to x of type CAT raises the value true.

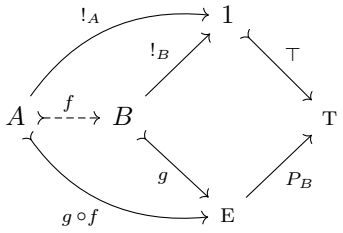
(5) $\quad \mathbf{cat} \circ f \circ x = \top$

It means that for any predicate in the language we obtain in $\mathcal{T}$ both a type (an object) and a predicate term (a morphism), which are related by the Ω-axiom. This duality of monadic predicates has been discussed by Retoré [16] and more recently by Chatzikyriakidis and Luo [5], who point out that such a property could make type checking undecidable, in general. It is difficult to say whether the system presented here could bring the basis of a solution for facilitating type checking, if any; this problem will not be addressed in this paper, but should be

[7] This morphism is actually the character of A as a subobject of itself.

explored in future works. Yet it is worth noticing that the two interpretations of a predicate are here related by virtue of the Ω-axiom.

At that point, we have only followed the isomorphism $\mathcal{T}(\text{E}, \text{T}) \cong \text{Sub}(\text{E})$ given by the axiom to introduce new types in our topos, but we have not specified which structure they have. By general properties of topoi, we know that Sub(E) is a Heyting algebra. Furthermore, a formal hierarchy can be reconstructed. Indeed, ontological inclusions can be encoded by saying that a type A is a *subtype* of B if and only if $A \cap B \cong A$, which means that any entity which satisifies the predicate P_A associated with A satisfies also the predicate P_B associated with B. In this case, the properties of intersection objects ensure that there is a monomorphism from A to B. Conversely, whenever two subobjects A and B of E are linked by a monomorphism $f : A \rightarrowtail B$, then P_B can be proved to be true on A. More specifically, if P_B is the character of B as suboject of E, we want to show that $P_B \circ g \circ f = \text{true}_A$, i.e. that the diagram below commutes:



In this diagram, $g : B \rightarrowtail \text{E}$ is the subobject induced by the predicate P_B, which is a monomorphism. Considering the hypothesis that $P_B \circ g = \text{true}_B$, we just need to compose f on the right of each side of the equality, and verify that $\text{true}_A = \text{true}_B \circ f$. This last equality comes from the universal property of the terminal object: $!_B \circ f$ is a morphism $A \to 1$; but such a morphism is actually unique, so $!_B \circ f = !_A$, and then we get the desired result by composing with $\top$ on the left. Also, the morphism h given by the pullback property is such that $g \circ f = g \circ h$, which then gives us that $f = h$ because g, as monomorphism, is left-cancellative. This result can be formalised in the property 1 below, which holds for our topos $\mathcal{T}$. It synthesises the structural condition which makes $\mathcal{T}$ a good categorical representation of a type ontology.

Property 1. A is a subtype of B in the type ontology of language if and only if there is a monomorphism from A and B in $\mathcal{T}$.

Hence we have a correspondence between subtypes as we would intend it in a type ontology, and monomorphisms in Sub(E). As a result, the predicates from the language were used to build a hierarchy of types solely upon the type E, corresponding to the Montagovian general type of entities. Now E is the greatest type in the hierarchy formed by the algebra Sub(E), and all of them enjoy the Ω-axiom thanks to the subobject classifier T. The types formed using the operations on the algebra receive the expected interpretation : $A \cup B$ is the union type of A and B, that is the type of entities satisfying $P_A \vee P_B$; $A \cap B$ is the type of entities satisfying $P_A \wedge P_B$, $A \Rightarrow B$ is the one of entities satisfying $P_A \Rightarrow P_B$,

and $\bar{A}$ the one of entities satisfying $\neg P_A$. However, it leads to a system which has arguably too many types, and we do not want to be bound to use them all. We will thus examine in the following how the types required for a semantic calculus could be restricted.

3.2 From Type Hierarchy to an Ontology of Types

We have seen that every first-order monadic predicate in the language defines its own type in the hierarchy Sub(E), so that the algebra has at least as many types as such predicates. Yet the vocabulary of a given language is finite, and so are the "base types" directly created from language predicates. However, building new types on those basic ones by the means of negation, union, intersection, and implication operators quickly leads to a combinatorial explosion of the number of types. One possible solution to avoid the troubles of a direct implementation of our system would be to select only a finite (and possibly small) substructure of Sub(E) to be used with all the terms of the future λ-calculus. This idea arises from the fact that the whole type hierarchy cannot be an ontology, as argued for instance in [24]. Indeed, ontological types can be understood as types which divide the class of entities at a general level: types arising from predicates like 'cat', 'unicorn' or 'fork' are too specific to divide the world in an important way, contrary to distinctions like physical or abstract, or animate or not.

This actually matches Sommers' view on categories: they are defined from predicates not by truth, but by span. When we said that this entity is a cat, neither the language nor the ontologist bother with whether this entity is actually a cat, but only with whether this entity has the required properties to wonder whether it is a cat or not. Thus ontological types would be the way to encode the presuppositions that are made whenever we apply a predicate to a given entity. Then, considering our current type system, is there a way to retrieve Sommers' ontological categories? It is actually quite easy when we compare what we have done so far to the construction proposed by Sommers. As presented in the introduction, Sommers built his categories from predicates, and then defined types of predicates which define the same categories; we have similarly built our types from predicates. The only difference between Sommers' categories and our types is that entities in categories are *spanned* by the defining predicate, whereas entities in our types are *true* for the defining predicate.

Moving from truth to span can be done directly by following the definition of what Sommers named a predicate: as detailled in [21], the class $|P|$ of entities spanned by a predicate P is exactly the class of entities which are P or un-P, that is, which satisfy either P or $\neg P$. In our topos, we can get such classes by considering only predicates of the form $P \vee \neg P$, and the types we obtain are of the form $C(A) \doteq A \cup \bar{A}$. Hence an important property that should be imposed to our topos if we want to get the same ontology as Sommers is to be non-classical. Indeed, if our topos is classical, then every subobject algebra, including Sub(E), becomes a Boolean algebra, which means that for every $A \in \text{Sub}(\text{E})$ we have $A \cup \bar{A} \cong \text{E}$. It is interesting to notice here that as a result, if we constrain $\mathcal{T}$ to be a classical topos and if we apply the procedure described here, all first-order

monadic predicates define the type E, so that a pure Montagovian system is retrieved. It is also important to point out that making our topos non-classical does not necessarily mean that we are adding one or several new truth values in our system, in the sense of morphisms $1 \to \mathrm{T}$: the topos $\mathcal{T}$ may still be bivalent, even if T does not behave like a two-element set. Rather, we are moving from classical logic to intuitionistic logic, where the law of excluded middle no longer applies: the application of a predicate to an entity may not raise a truth value, and if it happens we are in the situation of a category mistake in the sense of Ryle. Hence our type system based on a non-classical topos is able to capture the behaviour of Sommers' ontological categories. This leads to another property which holds for our topos $\mathcal{T}$:

Property 2. Sommers' ontological categories are definable in $\mathcal{T}$ if and only if $\mathcal{T}$ is non-classical.

As recalled above, Sommers made use of his categories to gather predicates which define the same categories under the same A-types. In our type system, this simply corresponds to refining the domain of the predicate morphisms to their span. For instance, consider the predicates **cat** and **dog** and suppose that their spans are the same, that is, $C(\mathrm{CAT}) \cong C(\mathrm{DOG})$. To fix the ideas, let the type of their span (up to isomorphism) be ANIMATE.[8] Then, we would have to replace **cat** and **dog** by restricted predicates $\mathbf{cat}'$ and $\mathbf{dog}'$ of type $\mathrm{ANIMATE} \to \mathrm{T}$. Such a transformation is enabled by composing with the corresponding monomorphism: whenever $f : A \rightarrowtail \mathrm{E}$ is a subobject of E, we can send any predicate $g : \mathrm{E} \to \mathrm{T}$ to $g \circ f : A \to \mathrm{T}$.[9] Therefore we can apply such a function to **cat** and **dog** because ANIMATE is a subobject of E, and the results are the desired $\mathbf{cat}'$ and $\mathbf{dog}'$. Henceforth we will only consider predicates with domains refined to their span, and ontological types — that is, span types in question — for typing entities. Thus the size of an implementation of a calculus based on this type system can be reduced since — following Sommers' arguments — our new ontological system is a skeleton of the complete hierarchy we had first, with far fewer types needed.

Yet at this point we are unsure about the structure of our ontology. In general in a topos, whenever two subobjects A and B are linked by a monomorphism then there is a monomorphism from $\bar{B}$ to $\bar{A}$,[10] and we can say nothing about the

[8] Whether or not the common span of 'cat' and 'dog' is really animate entities could be debated, in particular with some examples as in (i) where 'rock' seems also to belong to the span of 'dog':

(i) This is not a dog but just a rock.

However, the main idea to keep is that the spans of the two predicates are probably the same, as it does not seem absurd to say that anything that is not a dog *could* be a cat or not, and conversely.

[9] This transformation corresponds to a function $\mathcal{T}(\mathrm{E}, \mathrm{T}) \to \mathcal{T}(A, \mathrm{T})$ in Set, which is a specific case of application of the contravariant hom-functor $\mathcal{T}(-, \mathrm{T})$.

[10] This is actually a well-known property for Hilbert algebras, but it applies here as a Heyting algebra is a particuliar case of Hilbert algebra. The existence of a monomor-

relationship between $C(A)$ and $C(B)$. In our case however, Sommers brought a solution by formulating his structural principle. It was intended to reflect the "ontological behaviour" of the language, and claims that two overlapping categories are included one in the other. Consider for instance the types DOG, CANID and ANIMATE, with quite obvious subtyping relations between them. Even if DOG is a strict subtype of CANID, it seems reasonable to assume that their respective spans are actually the same, i.e. $C(\text{DOG}) \cong C(\text{CANID})$, because a term that would span one but not the other could hardly be found. However, as previously supposed, the ontological category $C(\text{DOG})$ is isomorphic to ANIMATE, which means that there is a monomorphism from $C(\text{DOG})$ to $C(\text{ANIMATE})$ which is not an isomorphism: in other words, the category $|\mathbf{dog}|$ is strictly included within the category $|\mathbf{animate}|$. If all predicates obey this principle we retrieve a hierarchical structure in our ontology, where the passage from general types to ontological ones has made some parts of the hierarchy "collapse" into the same span type, while other subtyping relations are preserved.

The construction presented here is admittedly rather abstract and gives no clue about how to build such an ontology in practice. In [16], Retoré provides an interesting discussion on what should be the base types of a semantic type system, and quickly reviews the different set of types that have been proposed so far. As he states himself, the choice of such a set depends notably "on one's philosophical convictions", and this problem will not be solved here. However, the type system presented in this paper can accomodate with any proposition, because it actually generates the greatest number of types possible by default, and as shown in this section this overgeneration does not preclude to refine the set of types used in practice as long as it keeps the structure of a sub-hierarchy. Thus the pure Montagovian system is retrieved when taking the span types in a classical topos, Chatzikyriakidis and Luo's proposition of common nouns as types [5] is captured quite easily from our original type construction, and intermediate sets as Sommers' and others' can also be used in such a framework.

3.3 A Short Account of Dot Objects

One of the main ideas of Asher [1] for his categorical model of TCL was to propose a type-theoretic account for the so-called *dot objects* [15,2]. Dot objects are lexical units which show the property of inherent polysemy: they can appear in contexts that are generally contradictory in terms of type requirement. A classical exemple of dot object is 'book', which can be treated as a physical object (6a), or as an informational content (6b), even though physical objects and informational content have distinct, non-overlapping spans.

(6) a. Mary picked up the book.
b. John didn't understand the book.

phism between two subobjects A and B corresponds to the natural order of those algebras: if we note $A \leq B$ when such a monomorphism exists, then $A \leq B$ iff $(E \cap A) \leq B$ iff $E \leq (A \Rightarrow B)$, which is equivalent to $E \cong A \Rightarrow B$ as expected for a natural order.

Hence dot objects are entities with several separate aspects, that is, several types. In Pustejovsky's and Asher's works, those objects are given a *dot type*, that is a complex type structure where every type aspect is represented. In the case of 'book', if we call P the type of physical objects and I the type of informational contents,[11] the entry for 'book' would receive the type $\text{P} \bullet \text{I}$.

The major concern about dot types is to understand where they should be placed in the hierarchy. It is commonly admitted that dot types cannot be intersection types, as the intersection of two uncompatible types is naturally supposed to be empty (cf. [1, chap. 5] for discussion). Another hypothesis is to consider such types to be pairs of types. As pointed out by Asher, this can be an interesting solution provided that the different aspects of a dot object are kept separate, instead of having the transformation only on the type side — that is, we do not want to consider a dot type to be a direct subtype of its components. To sum up, he introduces in his categorical model dot types as objects with "aspect projections" to some pullback objects.[12] Those pullback are more precisely defined from the relation between aspects of the dot object, lifted to power objects. For 'book' of type $\text{P} \bullet \text{I}$, let $ex : \text{I} \to \mathscr{P}(\text{P})$ and $in : \text{P} \to \mathscr{P}(\text{I})$ be those lifted relations: then, $\text{P} \bullet \text{I}$ has projections to the pullback objects of the diagrams $\text{I} \xrightarrow{ex} \mathscr{P}(\text{P}) \xleftarrow{id} \mathscr{P}(\text{P})$ and $\mathscr{P}(\text{I}) \xrightarrow{id} \mathscr{P}(\text{I}) \xleftarrow{in} \text{P}$.

As explained by Pustejovsky [15], the relation between aspects is part of the definition of a dot object, which means that for a given pair of types several different relations — and consequently several different dot objects — can be defined. The $\bullet$ operator only says that a relation exists, but does not provide it explicitly. Pustejovsky proposed to define several dot operators $\bullet_{R_1}, \ldots, \bullet_{R_n}$, one for each relation $R_1, \ldots, R_n$, for a proper account. In our topos, we can follow this idea by introducing the relations explicitely as subobjects of a product. Thus, the type of 'book' would be defined by an object BOOK equipped with a monomorphism $f : \text{BOOK} \rightarrowtail \text{P} \times \text{I}$. Such a definition is actually equivalent to Asher's proposition: indeed, the properties of power objects (which always exist in a topos) state that the the relation object BOOK implies the existence of the two lifted morphisms $book_{\text{P}} : \text{P} \to \mathscr{P}(\text{I})$ and $book_{\text{I}} : \text{I} \to \mathscr{P}(\text{P})$. Then the two projections proposed by Asher can be retrieved from the following compositions:

$$\begin{array}{ccccc} & & \text{BOOK} & & \\ & & \big\downarrow f & & \\ \text{P} \times \mathscr{P}(\text{I}) & \xleftarrow[\langle \pi_1, book_{\text{P}} \circ \pi_1 \rangle]{} & \text{P} \times \text{I} & \xrightarrow[\langle book_{\text{I}} \circ \pi_2, \pi_2 \rangle]{} & \mathscr{P}(\text{P}) \times \text{I} \end{array}$$

where $\pi_1 : \text{P} \times \text{I} \to \text{P}$ and $\pi_2 : \text{P} \times \text{I} \to \text{I}$ are the canonical projections of the product. Note that the morphisms above are not exactly the ones from Asher, as their codomains are products instead of pullbacks, that is in this case restricted products w.r.t. the satisfaction of the relation BOOK; but Asher's can be obtained

[11] In the remainder of this paper both types will be assumed to be ontological.

[12] There is actually more subtleties in his construction, but they will not be detailled here due to lack of space. The whole reasoning can be found in [1].

easily from those by *epi-monic factorisations*, which are always possible in a topos (see [9] for details).

We have therefore a way for representing dot types without using the dot notation, which thus permits placement of the relation between the different aspects at the centre of the definition of such a type. Being a dot type in $\mathcal{T}$ is equivalent to being a subobject of a product of types from Sub(E). When browsing Sommers' theory, it is difficult not to make the connection between dot objects and heterotypical entities, because of this common ability to show multiple types according to the context. Actually, some arguments pointing out that the former may be a particular case of the latter can be given. Besides this multiple typing property, it is interesting to notice that in general, relations such as BOOK do *not* belong to Sub(E), because the product of two subobjects is not itself a subobject. The projection morphisms from a product to its components has no reason to be a monomorphism, unless all components but one are terminal objects. Moreover, in general their compositions with subobjects of the product do not create monomorphisms either, which is also an expected behaviour: in the case of 'book', there are obviously many copies of the same book as well as copies compiling several books in one physical object, which means in a set-theoretic acceptance (for fixing ideas) several pairs with same image through the projections. Thus if our type ontology is included in Sub(E), then dot objects cannot be part of it — and this exactly is how heterotypicals behave. A last note we should make here is that relation types like BOOK, as a consequence of their definition, are not ontological; and following Sommers' philosophy the product type P • I is not ontological either. But that does not mean that we cannot use those types in practice. Rather, this should incite us to treat such types for what they really describe: relations between ontological types.

4 Related Works and Future Perspectives

Several works which have been great sources of inspiration for the topos type system presented here have been mentioned throughout this paper. It will thus not be surprising that connections to those works could be made. As shown above, the natural definition of types from language predicates share many common points with the proposition of common nouns as types advocated by Luo and Chatzikyriakidis [5]. Moreover, the organisation of types in a Heyting algebra of subobjects allows parallels with Luo's coercive subtyping [12], as categorical properties make compositions coercive rather than subsumptive: if we have in our topos $\mathcal{T}$ defined above a predicate $P : A \to \mathrm{T}$ to be applied to an entity x of type B (that is, a global element $x : 1 \to B$) with B subtype of A, then the only way to compose both morphisms in $\mathcal{T}$ a priori is to use the monomorphism f between B and A. This leads to the morphism $P \circ f \circ x$, where f is a subtyping coercion in the sense of Luo. It is also worth noticing that his complete theory partially originates from Martin-Löf's intuitionistic type theory: as natural basis for intuitionistic logic, topoi could be useful for a categorical-based account of this approach.

The connection between the initial proposition of Asher [1] for TCL and the present work is also unsurprising, as the latter is mainly a deeper look into the properties of topoi and their consequenses on type systems, heavily inspired by the former. However, nothing has been said about the contravariance problem for subtyping between monadic predicates in this paper. A categorical-based argument can be given to advocate for the existence of a covariant subtyping of first-order arrow types, and more developments on this idea could be given in the future. The work of Retoré and Mery on ΛTy_n (see [16]), a framework based on Girard's system F, has also been mentioned here, but further investigations seem to be necessary to determine whether the present work could be extended to a semantic model of their system.

As for the question of type ontologies, it has been argued above that the topos-based system described in these pages can be adapted to any set of base types, depending on one's philosophical convictions on that matter, including the traditional Montagovian system. It has also been shown that this type system shares interesting and welcome similarities with the theory of Fred Sommers. The ontological tree he proposed [20] can be reconstructed as a substructure of the algebra Sub(E) in a natural way, and the topos even seems to give a faithful account of the case of heterotypical composites as lying outside the tree structure. Although Sommers' theory has been questioned extensively in the 70s, it seems to have been somewhat forgotten since. However his view on ontologies could have useful applications in the fields of formal and lexical semantics. This has been also recently advocated by Saba [18], and the present work might serve as a logical basis for such approaches. It is also worth noticing that Sommers, as follower of the theory of meaning-in-use, proposed a concrete way of building his ontology from language. An actual hierarchy could thus be obtained by implementing and running his method on corpora.

This paper presented a sketch of what would be a type system based on topoi, introducing a specific instance of a topos which shows how to construct various kind of types ("classical", ontological, heterotypical), and how to organise them in a hierarchical structure able to produce the type systems usually assumed in formal and lexical semantics. Moreover, two main properties that have to be satisfied by the topos have been drawn: monomorphisms should represent subtyping relations, and the topos should be non-classical. As it has been pointed out several times in these pages, such a categorical type system should naturally lead to a typed λ-calculus, using objects of the topos as types and morphisms as terms. The formal construction and the properties of such a calculus are still to be explored, and should therefore constitute the general outline of future work on this subject. More particularly, the question of how this system can be improved in order to give new tools for a fine-grained account of type shifts, coercion and copredication phenomena will be investigated at some point.

References

1. Asher, N.: Lexical Meaning in Context: A Web of Words. Cambridge University Press, Cambridge (2011)

2. Asher, N., Pustejovsky, J.: A type composition logic for generative lexicon. Journal of Cognitive Science **7**(1), 1–38 (2006)
3. Berry, G.: Some syntactic and categorical constructions of lambda-calculus models. RR-0080, INRIA (1981)
4. Brown, R.: A First Language: The Early Stages. Harvard University Press, Cambridge (1973)
5. Chatzikyriakidis, S., Luo, Z.: On the interpretation of common nouns: Types versus predicates. In: Chatzikyriakidis, S., Luo, Z. (eds.) Modern Perspectives in Type-Theoretical Semantics, Studies in Linguistics and Philosophy, vol. 98, pp. 43–70. Springer (2017)
6. Chomsky, N.: Some methodological remarks on generative grammar. WORD **17**(2), 219–239 (1961)
7. Church, A.: A formulation of the simple theory of types. The Journal of Symbolic Logic **5**(2), 56–68 (1940)
8. Geeraerts, D.: Prototype theory. Linguistics **27**(4), 587–612 (1989)
9. Goldblatt, R.: Topoi: The Categorial Analysis of Logic, Studies in logic and the foundations of mathematics, vol. 98. North-Holland Publishings (1979)
10. Kiefer, F.: Some semantic relations in natural language. In: Josselson, H.H. (ed.) Proceedings of the Conference on Computer-related Semantic Analysis. pp. VII/1–23. Wayne State University, Detroit (1966)
11. La Palme Reyes, M., Macnamara, J., Reyes, G.E.: Reference, kinds and predicates. In: Macnamara, J., Reyes, G.E. (eds.) The Logical Foundations of Cognition, Vancouver Studies in Cognitive Science, vol. 4, pp. 91–143. Oxford University Press (1994)
12. Luo, Z.: Type-theoretical semantics with coercive subtyping. In: Li, N., Lutz, D. (eds.) Proceedings of SALT 20. pp. 38–56 (2010)
13. Miller, G.A.: Nouns in WordNet. In: Fellbaum, C. (ed.) WordNet: An Electronic Lexical Database, pp. 23–46. The MIT Press, Cambridge (1998)
14. Montague, R.: The proper treatment of quantification in ordinary english. In: Suppes, P., Moravcsik, J., Hintikka, J. (eds.) Approaches to Natural Language, pp. 221–242. Reidel, Dordrecht (1973)
15. Pustejovsky, J.: The semantics of lexical underspecification. Folia Linguistica **32**(3-4), 323–348 (1998)
16. Retoré, C.: The montagovian generative lexicon ΛTy_n: a type theoretical framework for natural language semantics. In: Matthes, R., Schubert, A. (eds.) Proceedings of the 19$^{\text{th}}$ International Conference on Types for Proofs and Programs. LIPICS, vol. 26, pp. 202–229 (2014)
17. Rosch, E.H.: Natural categories. Cognitive Psychology **4**, 328–350 (1973)
18. Saba, W.S.: Logical semantics and commonsense knowledge: Where did we go wrong, and how to go forward, again (2018), arXiv preprint
19. Seely, R.A.G.: Categorical semantics for higher order polymorphic lambda calculus. The Journal of Symbolic Logic **52**(4), 969–989 (1987)
20. Sommers, F.: The ordinary language tree. Mind **68**(2), 160–185 (1959)
21. Sommers, F.: Type and ontology. The Philosophical Review **72**(3), 327–363 (1963)
22. Sommers, F.: Structural ontology. Philosophia **1**(1-2), 21–42 (1971)
23. Suzman, J.: The ordinary language lattice. Mind **81**(3), 434–436 (1972)
24. Westerhoff, J.: The construction of ontological categories. Australasian Journal of Philosophy **82**(4), 595–620 (2004)
25. Wittgenstein, L.: Philosophical Investigations. Macmillan, New York (1953)