information

Article

Agile Logical Semantics for Natural Languages

Vincenzo Manca

Dipartimento di Informatica, University of Verona, 37134 Verona, Italy; vincenzo.manca@univr.it

Abstract: This paper presents an agile method of logical semantics based on high-order Predicate Logic. An operator of *predicate abstraction* is introduced that provides a simple mechanism for logical aggregation of predicates and for logical typing. Monadic high-order logic is the natural environment in which predicate abstraction expresses the semantics of typical linguistic structures. Many examples of logical representations of natural language sentences are provided. Future extensions and possible applications in the interaction with chatbots are briefly discussed as well.

Keywords: logical semantics; predicate logic; natural language processing; large language models

check for updates

Citation: Manca, V. Agile Logical Semantics for Natural Languages. *Information* **2024**, *15*, 64. https://doi.org/10.3390/info15010064

Academic Editor: Peter Revesz

Received: 6 December 2023
Revised: 17 January 2024
Accepted: 19 January 2024
Published: 21 January 2024

1. Introduction

Epochal changes and new possibilities in the interaction between humans and artificial systems capable of processing information have been brought about by the recent advances in Natural Language Processing (NLP), which are based on Machine Learning and Artificial Neural Networks [1–4]. Large Language Models (LLMs) [5,6], in particular, represent the start of a new line of development that will have enormous implications for the entire field of artificial intelligence and numerous applications involving our societies globally. LLMs are the foundation of recent systems that are widely available to the public.

The kind of "understanding" that these systems are capable of achieving in conversation with humans is among their most contentious features. There are a wide range of opinions in the current debate between the extremes that they (i) converse without really understanding the other person and (ii) converse while gaining knowledge that could eventually approach that of humans and animals. In any event, these systems do display intelligent characteristics, making consideration of broad approaches to natural language text interpretation a critical theme for the development of LLM systems in the future.

Semantics is a very old topic; Leibniz is credited with the earliest modern mathematical formulation of it in his **Characteristica Universalis** [7].

After millennia of development, the logical representation of natural language texts is today a well developed field with a vast body of books and articles. Specifically, in the 1970s, Richard Montague, a student of Alfred Tarski (who founded both set-theoretic model theory and logical semantics [8]), developed a valuable theory proving that higher-order predicate logic generates coherent and comprehensive representations of texts [9–11]. Richard Montague's famous article "English as a Formal Language" was followed by similar works. Montague's theory s formally complex, using *intensional logic* and Alonzo Church's lambda abstraction [12].

In short, from Montague's point of view every linguistic element has a semantic that is provided by a high-order function that is represented in an appropriate space by a lambda term. Our formalism, as we will demonstrate, enables us to formalize sentences in natural languages by decomposing them into their component parts—all predicates associated with words—and joining these parts with constants and logical operations (connected concepts are provided in [13]). A logical operator of "predicate abstraction", which is present neither in Montague's work nor in analogous subsequent logical approaches [14,15], provides an advancement of Montague's grammars in terms of simplification of the logical apparatus and adherence to the linguistic structures. Moreover, monadic high-order predicates allow us to eliminate variables.

Apart from Montague's approach, formalisms of logical representation constitute a large field of investigation in artificial intelligence [16,17]. However, the spirit and the finalities of these systems are very different from those of the present work. In fact, they are essentially oriented towards knowledge representation (KR), very often focusing on specific knowledge domains (e.g., programming, datasets, query languages, semantic webs, belief revision, medicine). In these contexts, natural language is considered an instrument on which representations are based rather than an object of investigation in itself. Moreover, they use variables, first-order or second-order logic, and modal or temporal operators, and the rules of composition are very complex, making KR languages comparable to programming languages. In certain cases they differ radically from classical predicate logic, and can follow very different principles and presuppositions [17].

Although the basis of our formalism is logically sophisticated (high-order predicate logic, logical types, lambda abstraction), we can explain the method in a very intuitive way because monadic predicates naturally resemble the conceptual organization of words and completely avoid variables. The ability to produce correct logical representations lies essentially in the choice of the involved constants, the right aggregation of parts by means of parentheses, and the right logical types of the constituents, which is managed using the operator of predicate abstraction. The simplicity of the formalism is proven by the conversation with ChatGPT 3.5 reported in the final section, where, after one page of conversation, the chatbot is able to show a basic familiarity with the presented formalism.

The main ingredients of our logical representations are words, constants, parentheses, and predicate abstraction. This means that semantics reduces to a relational system of words from which morphology and syntax are removed and the logical essence of the relationship between words is extracted. The relevance of this for LLM models could be considerable, and surely needs further analyses and experiments that can be developed using the most recent chatbots. In addition, as addressed in our conclusions, this fact raises a number of problems around the proprietary nature of these systems, as training strategies and finalities are under the control of the companies producing and maintaining them.

2. Materials and Methods

In this section, we first define the main aspects of logical semantics, then outline predicate high-order logic by providing the first examples of the logical representation of sentences.

2.1. Logical Semantics

Let us begin by observing the intrinsic principle of **duality** in semantics. Meanings denote both objects and relations between them; therefore, when associating meanings with symbolic expressions of a certain type, it is necessary to presuppose both objects and relations.

What is essential is the **application** of a predicate to complementary entities of the application, called **arguments**. The **proposition** obtained as result of this application expresses the occurrence of relationships between the two types of entities. We can write

$$P(a, b)$$

to express the validity of a relation associated with P on the **arguments** a, b (in the order they appear). P is called a **predicate**, and denotes a relation; thus, $P(a, b)$ is called a **predication** or **atomic proposition**. The **individual constants** a, b designate the arguments of the predicate P.

However, because a predicate can be an argument for a predicate of a higher type, predicates are arranged along a hierarchy of levels, or **logical types**; according to Russell's theory of logical types, this situation can occur indefinitely [18,19].

In addition, it is possible to conceive of relations that are simultaneously both objects and relations. However, as these possibilities often lead to logical inconsistencies, they should only be considered in specific and well-controlled contexts with appropriate precautions. We exclude them from the following discussion, assuming individuals at level zero and predicates of levels 1, 2, 3, . . . (the semantics of natural languages rarely require predicates of order higher than 3).

The **negation** of $P(a,b)$,

$$\neg \mathbf{P}(\mathbf{a},\mathbf{b})$$

indicates the opposite of $P(a,b)$, i.e., its non-validity. The **disjunction** is a proposition,

$$\mathbf{P}(\mathbf{a},\mathbf{b}) \vee \mathbf{P}(\mathbf{b},\mathbf{a})$$

indicating that at least one of the two propositions connected by $\vee$ is true, while the **conjunction**

$$\mathbf{P}(\mathbf{a},\mathbf{b}) \wedge \mathbf{P}(\mathbf{b},\mathbf{a})$$

indicates that both propositions connected by $\wedge$ are true. The arrow $\rightarrow$ denotes **implication**; thus, the proposition

$$\mathbf{P}(\mathbf{a},\mathbf{b}) \rightarrow \mathbf{P}(\mathbf{b},\mathbf{a})$$

read as "if P(a, b), then P(b, a)" is equivalent to

$$\neg \mathbf{P}(\mathbf{a},\mathbf{b}) \vee \mathbf{P}(\mathbf{b},\mathbf{a})$$

and finally

$$\mathbf{P}(\mathbf{a},\mathbf{b}) \leftrightarrow \mathbf{P}(\mathbf{b},\mathbf{a})$$

is the logical **equivalence** equal to $(\mathbf{P}(\mathbf{a},\mathbf{b}) \rightarrow \mathbf{P}(\mathbf{b},\mathbf{a})) \wedge (\mathbf{P}(\mathbf{b},\mathbf{a}) \rightarrow \mathbf{P}(\mathbf{a},\mathbf{b}))$.

The symbols $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ are called **connectives** (negation, disjunction, conjunction, implication, equivalence), while the symbols $\forall$ and $\exists$ are called **quantifiers** (universal, existential).

If we consider a variable x, then

$$\forall x P(x,b)$$

asserts that, for every value a taken by x, $P(a,b)$ holds, while

$$\exists x P(x,b)$$

asserts that there exists a value a of x for which $P(a,b)$ holds. Connectives between propositions can be extended to predicates. In particular, if P and Q are predicates with only one argument, then $(P \rightarrow Q)$ denotes the predicate such that $(P \rightarrow Q)(a)$ holds when proposition $(P(a) \rightarrow Q(a))$ holds.

2.2. Formalizing Natural Language Sentences

Predicate logic [10,12,20] is a formal system used to represent the logical structure of propositions. Chapter 6 of [20] develops, in more than 100 pages, the first modern attempt at logical analysis of natural language in terms of predicate logic. It provides a way to express relationships between objects and describe actions, properties, and concepts. In this text, we explore how high-order predicate logic can be used to represent the meaning of sentences and concepts in natural language in a systematic and agile way. The method is independent from any specific language, and is adequate for teaching logical analysis to artificial systems.

In predicate logic, we have three main components.

Predicates, Objects, and Logical Symbols

Predicates: these are symbols that represent relationships or properties. They describe how objects are related to each other or specify attributes of objects; for example, "love", "eat", and "happy" are predicates.

Objects: these are the entities to which predicates are applied. They can be individuals, things, or concepts. In natural language, objects can include people and animals as well as relations and properties. In this sense, there is a need for both predicates that can be applied to objects and for objects that are predicates to which other predicates can be applied.

Logical Symbols: **connectives** and **quantifiers** are used to express the logical operations $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$. Parentheses and commas are additional symbols that are needed for writing logical formulas.

Objects and predicates are denoted by: (i) **constants** (letters or strings) denoting objects and relations (at every level) and (ii) **variables** (letters or strings, different from those used for constants) ranging over objects and relations (at every level). We adopt a convention that we call of **implicit typification**, in which: (i) lowercase letters denote individuals (objects at zero level); (ii) strings of letters with one uppercase letter denote first-order predicates (over individuals); (iii) strings with two uppercase letters denote second-order predicates (over first-order predicates); and (iv) analogously for the third order and higher orders. Strings of letters including x, y, z, X, Y, Z (possibly with subscripts or superscripts) are variables, while strings including other letters (lowercase or uppercase, possibly with subscripts or superscripts) are constants. In this way, the form of a string assigns to it the role of a constant or a variable and determines its logical type.

Predicates are associated with words in a given language. In this case, the logical types of such predicates can be deduced from the types of their arguments.

A **predicative theory** is provided by a list of propositions (as indicated below, on subsequent lines).

Below, we affirm a principle whose validity has been proven by the applications of Mathematical Logic from the late 19th century to the present day.

The semantics of every symbolic expression can always be reduced to an appropriate predicative theory.

In practice, predicative theories use additional symbols to make them easier to read and write. For example, the equality symbol = is used to affirm that two symbolic expressions have the same meaning. In formal terms, equality is defined by the following proposition:

$$a = b \leftrightarrow \forall X(X(a) \leftrightarrow X(b)).$$

However, all symbols extending the kernel of Predicate Logic can be formally defined in the basic setting provided above, and can be reduced to Frege's logical basis of negation, universal quantifier, and implication ($\neg, \forall, \rightarrow$).

Predicates associated with words include: (i) **lexemes** from a dictionary (in a predefined language), including proper names; and (ii) grammatical elements, called **grammemes**.

Obviously, the choice of dictionary determines the lexemes, while grammatical predicates depend on the terminological choices of the reference grammar. For example, we could use "ComplOgg" to indicate the role of an object complement or consider transitive verbs as predicates with two arguments (subject and object). For example, "a loves b" becomes $Love(a) \wedge ComplOgg(Love, b)$, or: $Love(a, b)$.

Furthermore, we can consider the predicate "I" or a predicate such as "1st-Pers-sing" (first-person, singular), and analogously for pronouns, prepositions, and conjunctions. Even a proper noun is a predicate; thus, Julia(a) indicates that "a" is named "Julia".

Thus, in predicative theory, the simple sentence "I love Helen" is expressed as

I(a)
Helen(b)
Love(a, b).

I(a): this indicates that the individual constant a is the "I" of the sentence.

Helen(b): this indicates that the individual constant b denotes an individual named "Helen".

Love(a, b): this asserts that "a loves b".

Of course, grammatical terminology is entirely arbitrary, and any equivalent terminology essentially express the same logical relationships between objects and predicates.

Let us consider the sentence "Yesterday, I was walking without shoes". Its predicative representation is as follows, where $\widehat{\text{Walk}}$ denotes the predicate abstraction of "Walk", which we explain in the next section:

I(a)
$\widehat{\text{Walk}}$(P)
$Without\text{–}shoe(x) = \forall y(Wear(x, y) \to \neg Shoe(y))$
$Without\text{–}shoe(a)$
PastProgressive(P)
YesterDay(P)
P(a).

Intuitively, the formalization of the sentence can be paraphrased as: (1) P is a walking motion; (2) a is without shoes (any object that a is wearing is not a shoe); (3) P is yesterday and is in the past (imperfect); (4) the constant a is the "I" of the sentence; (5) a satisfies predicate P.

3. Results

In this section, the logical operator of predicate abstraction in introduced, which is related to Church's lambda abstraction. Logical representations of a Chinese sentence are provided and High-order Monadic Logic (HML) is introduced, which is a special kind of high-order Predicate Logic. Finally, many examples of logical representations in HML are provided.

3.1. Predicate Abstraction

Predicate abstraction is a powerful logical operation in the context of natural languages. It allows us to elevate the logical order of a predicate. When we say $Love(a)$, we mean that individual a loves someone; however, when $\widehat{\text{Love}}$(P) holds, this means that P possesses the property of loving. Thus, P is a predicate including all the typical characteristics of loving, because $\widehat{\text{Love}}$ denotes a predicate over first-order predicates, which is a second order predicate.

We present the following informal definition of the Predicate Abstraction operator:

Given a first-order predicate $Pred$**, the second-order predicate** $\widehat{Pred}$ **is a predicate expressing the property of all predicates that imply the predicate** $Pred$**.**

In general, when applied to a predicate of order i, the predicate abstraction operator provides a new predicate of order $i + 1$. The sentence "Every man is mortal" has the following very simple representation showing the expressive power of predicate abstraction:

$$\widehat{Mortal}(Man)$$

namely, the predicate Man has the property of all predicates that imply mortality.

By using predicate abstraction, the sentence "I love Helen" becomes:

I(a)
Helen(b)
$\widehat{\text{Love}}$(P)
P(a, b).

Apparently, this seems a way of making difficult a very simple proposition: $Love(a, b) \wedge I(a) \wedge Helen(n)$. However, in the representation above it is possible to add other propositions having a P as argument, which can enrich P with other particular aspects.

For example, the sentence "I love Helen very much" is obtained by adding a further second-order predication to P:

I(a)
Helen(b)
$\widehat{\text{Love}}$(P)
VeryMuch(P)
P(a, b).

A formal definition of Predicate Abstraction can be provided by means of "lambda abstraction", introduced by Alonzo Church around 1920. Today, we prefer to express it in Python notation. Let us consider a Python expression $\mathcal{E}(a, B, i)$ built with some operations applied to data and variables, such as $(2 * a + B[i])$, where a is an integer, B is a list of integers, and i is an index (integer):

def funct(a, B, i)
    result = $\mathcal{E}(a, B, i)$
    return result.

This is essentially a way of expressing the function corresponding to the expression $\mathcal{E}$, independently from the choice of variables occurring in $\mathcal{E}$ as well as from the particular values assumed by the variables, that is, the result produced by the function is the evaluation of $\mathcal{E}$ when the variables occurring in it are instantiated with the arguments of funct. This mechanism is essential in programming languages, as it distinguishes the definition of a function from its application (the calling of function) in many possible contexts. It is a basic logical mechanism on which high-order predicate logic can be founded, together with application and implication.

The following is the formalization of the prior sentence regarding "walking without shoes" using predicate abstraction:

$\widehat{\text{Walk}}$(P)
$\widehat{\text{Without-shoes}}$(P)
I(a)
PassImperf(P)
Yesterday(P)
P(a).

This second representation of the sentence is more correct than the previous one; because $\widehat{\text{Without-shoes}}$ has P as argument, it is not expressing a property of the individual a (who sometimes may wear shoes), and instead characterizes P as a property of the walking of a (together with the predicates $Yesterday$ and $PassImperf$).

It can be verified that any discourse can be rigorously represented within the logical structure outlined here.

The number of basic words in a natural language is only a few thousand, while grammatical predicates are a few hundred and logical symbols a few dozen. By adding letters for constants and variables, it is possible to express the meanings of natural language texts with predicates of logical types that generally do not exceed the third level. However, with respect to Montague's approach, predicate abstraction permits a very simple way of constructing meanings incrementally by taking a basic predication $P(a)$ and adding other propositions with high-order predicates that provide further characterizations to P. As we show, this modularity avoids many complications of Montague's semantics by providing logical representations that are very close to the linguistic form of sentences.

3.2. Representing Meaning across Languages: The Chinese Example

Let us consider a sentence in Simplified Chinese:
昨天我去海散步
(yesterday, I went for a walk by the seaside).
The following words constitute the predicates used to build the predicative representation:

昨天 YesterDay
我 I
去 Going
海 Sea
边 Side
散 Scattered
步 Step
地方 Place

A predicative theory representing this sentence is as follows (QQ and RR are second order predicate constants):

我 (a)
$\forall X(QQ(X) \leftrightarrow \forall xy(X(x,y) \rightarrow$ 去 (x, y) ∧ 地方 (y)))
QQ(P)
昨天 (P)
海 (c)
边 (b, c)
$\forall X(RR(X) \leftrightarrow$ (步 (X) ∧ 散 (X))
RR(P)
P(a, b).

The predicate RR expresses that the action of P (already characterized as walking) is carried out "in steps" and in a "scattered" manner, i.e., distributed in space (in English, a walking). Let us use *abs* to denote predicate abstraction. For a predication such as $(Place(b))(P)$, expressing that predicate P is located at place b, the previous representation becomes

我 (a)
(abs(去)) (P)
昨天 (P)
海 (c)
边 (b, c)
(地方) (b)
(abs(步 ∧ 散)) (P)
(地方(b)) (P)
P(a).

This example demonstrates that our method is entirely independent of the language being considered; when the words have been associated with predicates, formulas can represent the sentences by indicating how predicates apply at the different logical levels.

In the last example, no variable occurs. This situation can be generalized using a particular type of high-order logic, which we present in the next section.

3.3. High-Order Monadic Logic

High-order predicate logic with only monadic (unary) predicates (HML) is a powerful environment for developing logical representations of natural language sentences. This short section provides a rigorous basis for the analysis of the logical types of high-order predicate logic. As it is more technical, readers who are not interested in the foundations of our formalizations can skip it without compromising their understanding of the following discourse.

High-order Monadic Logic (HML) can be expressed using three logical symbols:
(1) λ for (functional) **abstraction**;
(2) $\rightarrow$ for **implication**;
(3) **parentheses** ( ).

In HML, there are two categories of expressions, namely, **objects** and **types**. An object is associated with one and only one type.

There are two kinds of basic objects, **individuals** and **truth values**, with respective types *ind* and *tt*.

If σ, τ denote generic types, then $(\sigma \rightarrow \tau)$ is a type denoting functions transforming objects of type σ into objects of type τ.

For any type τ, there is an infinite list of constants and variables of that type (with τ as a subscript indicating the type).

The constants F (false) and T (true) denote the two possible truth values (of type *tt*).

In HML, there are three rules for obtaining expressions denoting objects starting from logical symbols, constants, and variables:

Abstraction rule: if ξ_σ is a σ-variable and φ denotes a τ-object, then $\lambda\xi_\sigma(\varphi)$ denotes an object of type $(\sigma \rightarrow \tau)$.

Application rule: if Θ denotes an object of type $(\sigma \rightarrow \tau)$ and ζ denotes an object of type σ, then $\Theta(\zeta)$ denotes an object of type τ. Moreover, if $\Theta[\xi]$ is an expression of type τ including a variable ξ_σ on which no λ-abstraction is applied and η is an expression of type σ, then

$$(\lambda\xi_\sigma(\Theta[\xi]))(\eta) = \Theta[\eta],$$

where $\Theta[\eta]$ denotes the expression $\Theta[\xi]$ after replacing all the occurrences of ξ_σ with η.

Implication rule: if φ and ψ denote truth values, then $(\varphi \rightarrow \psi)$ denotes a truth value. In general, if ϕ and Ψ are predicates of type $(\sigma \rightarrow tt)$, then $(\phi \rightarrow \Psi)$ is a predicate of the same type, such that for any expression η of type σ it is the case that

$$(\phi \rightarrow \Psi)(\eta) = \phi(\eta) \rightarrow \Psi(\eta).$$

It can be shown that all the logical operators of high-order predicate logic can be expressed in HML. In particular, using the symbols $\lambda, \rightarrow, T, F$, negation $\neg\varphi$ is expressed by $(\varphi \rightarrow F)$ and quantification $\forall x(\varphi)$ is expressed by $(\lambda x(\varphi(x)) = (\lambda x(T))$.

Expressions denoting truth values are called **propositions** (a predication can be considered as an "atomic proposition"), while those denoting objects of type $(\sigma \rightarrow tt)$, which we indicate with $pred_\sigma$, denote (unary) predicates of type σ. Objects of type $(ind \rightarrow tt)$ correspond to first-order predicates, and are simply indicated by *pred*, while objects of type $((ind \rightarrow tt) \rightarrow tt)$ are second-order predicates.

3.4. Predicate Abstraction in HML

Let us consider a binary predicate P over two individuals and the predication $P(a,b)$ of P over the arguments a, b. We can express this proposition by means of two unary applications: $(P'(a))(b)$, where $(P'(a))$ is the monadic predicate $(P(a,-))$ obtained by P when its first argument is put equal to a, which holds on b when $P(a,b)$ holds:

$$(P((a,-))(b) = P(a,b).$$

Therefore, P' is a function taking an individual as argument and providing the unary predicate $(P'(a))$.

Let $SecondArgument(b)$ be a second-order predicate satisfied by the monadic predicates $X(a,-)$ holding on b. Consequently,

$$P(a,b) = (P(a,-))(b) = ((SecondArgument(b))(P(a,-)).$$

However, $(P(a,-))(b)$ means that

$$(P\widehat{(a,-))}(b)(P)$$

therefore,

$$((SecondArgument(b))(P(a,-)) = (P\widehat{(a,-))}(b)(P).$$

In other words, the monadic reduction of a first-order binary predicate is definable in term of predicate abstraction. In conclusion, $P(a,b)$ is completely represented by

$$\begin{array}{l} \exists y P(a,y) \\ ((SecondArgument(b))(P(a,-)), \end{array}$$

and we can simply write

$$\begin{array}{l} P(a) \\ (SecondArgument(b))(P). \end{array}$$

Of course, the mechanism described for binary predicates can be naturally extended to predicates of any number of arguments.

Place, time, manner, instrument, possession, and other natural language complements logically relate to an (implicit) application of predicate abstraction. Specifically, we employ an implicit predicate abstraction (after accepting an individual as an argument) to express verb complements by supplying a predicate's property. As an illustration, the predication $(with(c))(P)$ states that P has the property $with(c)$ (a property of properties), the logical type of $with$ is $(ind \to (pred \to tt))$ (in fact. $with(c)$ has type $(pred \to tt)$), and finally, $(with(c))(P)$ is a proposition (of type tt).

The monadic nature of HML enables a very synthetic way of expressing the predicative structure of sentences: enumerating all constants and listing for each of them the predicates taking a given constant as argument. For example, the previously considered sentence "Yesterday I was walking without shoes" becomes:

$$a : P$$

$$P : Yesterday, Past, Progressive, \widehat{Walk}, Whithout(Shoe)$$

The above formalization corresponds to a Python dictionary structure of the following type (where "abs" stands for predicative abstraction):

'a': ['I', 'P'], 'P': ['Yesterday', 'Past', 'Progressive', 'abs(Walk)', '(Without(Shoe))(Wear)']

It is apparent that, by avoiding the explicit use of variables, the monadic setting of HML forces the formalization to fit closely with the linguistic form. Specifically, unary predicates naturally impose high-order logical types, with consequent elimination of variables. Moreover, a constant may occur as an argument and as a predicate at the same time $(P(a), Yesterday(P))$.

Certain aspects are crucial in the determination of the above Python dictionary: (1) the introduction of the right constants to which the predicates refer; (2) possible "hidden predicates" that do not occur as words in the sentence, which generally are of grammatical nature but in the case above include the lexical term "Wear"; and (3) the logical type of predicates and the pattern according to which they apply. For example,

$$(Without(Shoe))(Wear)$$

implicitly provides the following type assignments, where $pred$ abbreviates the type $(ind \to tt)$:

Whear : $pred$
Shoe : $pred$
Without : $(pred \to (pred \to pred))$.

In natural languages, complements, modifiers (adjectival and adverbial forms), and pronouns realize the reference mechanism, usually based on grammatical marks and morphological concordance (gender, number, tense . . .). A pronoun refers to a component having the same marks. Moreover, the aggregation of components is realized on the basis of concordance, which corresponds to the use of parentheses in mathematical expressions.

In HML, reference is realized by means of constants and aggregation is realized by parentheses, with the different levels of application expressing the logical levels of predicates in a rigorous way.

The sentence "Mary goes home with the bike" provides the following HML translation:

$Mary(a)$
$\widehat{Go}(P)$
$(Place(b))(P)$
$(With(c))(P)$
$Home(b)$
$Bike(c)$
$The(c)$
$P(a)$.

Synthetically,

$$a : Mary, P$$

$$b : Home$$

$$c : Bike$$

$$P : \widehat{Go}, With(c), Place(b).$$

A more complex example involving a relative clause is the sentence "I am searching for a bike with a leather saddle".

We can consider the logical definition of Any as a function of type $((ind \to tt) \to ind)$ satisfying the condition

$$\forall X(\exists x(X(x)) \to X(Any(X)))$$

$I(a)$
$Progressive\text{-}present(P)$
$(Search\text{–}For(Any(Q)))(P)$
$\widehat{Leather}(Q)$
$\widehat{Saddle}(Q)$
$(Of(Bike))(Part)(Q)$
$P(a)$.

Synthetically,

$$a : P, I,$$

$$P : (Search\text{–}For(Any(Q))), Progressive\text{–}present,$$

$$Q : \widehat{Leather}, \widehat{Saddle}, (Of(Bike))(Part).$$

We collected a number of translation exercises involving different kinds of linguistic constructions, several of which were long and complex, to confirm the power and adequacy of HML to represent natural language logic. The expressive mechanisms of lambda abstraction, high-order types, application, and implication, together with parentheses and constants, apply to any kind of natural language.

4. Teaching Logic to Chatbots

Here, we report a conversation with ChatGPT 3.5 that seems to be very informative about the potential of the discourse developed in the previous sections. Logical analysis could introduce a new kind of interaction with chatbots, opening up an interesting perspective on the AI discussion.

In many aspects, the reaction of ChtGPT is really surprising and impressive. While we do not report the whole conversation, a particular point deserves attention. In many discussions, people claim that chatbots do not really understand what is being said. Certainly, at this point in time they understand in a way that is very different from the way in which humans understand; indeed, we are not even sure how we comprehend. However, in general terms, abstraction is a necessary feature of any kind of comprehension. In fact, in order to understand meanings, information has to be organized according to general rules abstracted from single cases. In a previous conversation (not reported here) focused on the comprehension of sentences, ChatGPT, when stimulated by specific questions, explained that any word corresponds to a vector of real numbers (features), and that in the process of sentence elaboration these vectors generate a vector of thousands of values. Therefore, the process of sentence synthesis/analysis corresponds to suitable operations in many-dimensional vector spaces. This means that mathematical structures and related operations are involved in the dialogue activity. In other words, this activity relies on the capability of dominating abstract structures. On this basis, it could be said that these systems exhibit a particular form of comprehension.

Conversation

> I want to ask you to give logical representations of sentences, according to a simple and basic logical mechanism, that is, "predication". According to it a predicate is applied to an argument, also called the subject of predication. In this way any sentence is reduced to a list of subjects, and each of them has a list of predicates applied to it Subjects are denoted by constants represented by strings, while predicates are usually words. This basic schema becomes not easy if we consider that in the sentences of natural languages the predicates associated to the words are arranged along a natural hierarchy of logical types. In fact, there are initial predications where subjects are considered as having type zero, and their predicates type 1, but some of these predicates can be subjects of other predications where predicates are of type 2, 3, and so forth. (**Author**)
>
> It seems like you're describing a basic approach to logical representation using predication, where sentences can be broken down into subjects and predicates, and there's a consideration of logical types. This approach aligns with predicate logic and the concept of logical types as introduced by Bertrand Russell. In this framework, you would assign types to individuals (constants/subjects) and predicates to ensure meaningful and well-formed sentences. For instance, if a constant (individual) has type 0, the predicates associated with it would be of type 1. Then, if a predicate or constant has type 1, it can be applied to subjects (constants) of type 0, and so on. Let me know if you have a specific sentence you would like to represent in this logical framework, and I can help you break it down into subjects and predicates based on your described schema. (**ChatGPT**)
>
> Yes, exactly! Let us start with simple examples. At the beginning I provide you the representation, and then I will ask you to do the same. The first sentence is "I love Helen" In this case, Constants: are: a, b, P. Predicates are: I, Love, Helen, P, Object. Predications are: I(a), Helen(b), (abs(Love))(P), (Object(b))(P), P(a). Moreover, "abs" is an operator that indicates that abs(love) holds for predicates that imply the property of loving, while (Object(b))(P) indicated that predicate P has b as object. Let me know if this is clear, and provide me the representation of "Helen goes home with her bike". (**Author**)
>
> Certainly, I understand the structure you've outlined. Let's apply this to the sentence "Helen goes home with her bike". Using the constants and predicates you provided:
>
> Constants: a,b,c
>
> Predicates: Helen, Goes, Home, With, Her, Bike, P, Object
>
> Predications:
> Helen(a)

Goes(b)
Home(c)
(abs(With))(P)
(abs(Her))(P)
(abs(Bike))(P)
(Object(c))(P)

So, the logical representation of "Helen goes home with her bike" would include these predications involving the constants and predicates you've defined. If you have further sentences or questions, feel free to share them! (**ChatGPT**)

Let me show an improvement of your representation: (**Author**)

Certainly, feel free to provide an improved representation or any adjustments you'd like to make. I'm here to assist and discuss the logical representation further. (**ChatGPT**)

Helen(a), (With(b))(P), (abs(Go)(P), P(a), Home(c), Bike(b), (Place(c))(P), (Of(a))(b). (**Author**)

Great! Let's incorporate the improvements into the logical representation for "Helen goes home with her bike".

Constants: a,b,c

Predicates: Helen, Goes, Home, With, Her, Bike, P, Object, Place, Of

Predications:
Helen(a)
(With(b))(P)
(abs(Go))(P)
P(a)
Home(c)
Bike(b)
(Place(c))(P)
(Of(a))(b)

This representation reflects the refined structure you provided. If you have more sentences or if there's anything else you'd like to discuss or modify, feel free to let me know! (**ChatGPT**)

Ok, now please, could you explain me the logical motivation of the last predication above? (**Author**)

Certainly! Predication (Of(a))(b) represents the relationship between the constant a (Helen) and b (her bike) using the predicate Of. Let's break down the logical motivation:

- Predicate Of is used to express a relationship of possession or association. In this context, it signifies the association between Helen and her bike.
- The argument a represents Helen, and b represents her bike.
- Therefore, (Of(a))(b) is read as "Helen possesses/has her bike" or "The bike of Helen".

So, in the logical structure, this predication captures the idea that the constant a (Helen) is associated with the constant b (her bike) through the predicate Of, indicating ownership or possession. This is a common way to represent relationships and associations between entities in logical frameworks. (**ChatGPT**)

Let me ask another little improvement. In the list of predicates is present Goes, but is not present Go, while in the logical representation Go is present, but Goes is absent. How do you can fix this discrepancy? (**Author**)

Certainly, we can address this discrepancy by adding both "Go" and "Goes" to the list of predicates in the representation. Here's the corrected version: (**ChatGPT**)

But now Goes is in the list of predicates, but does not occur in the representation, hence which is the correct situation? (**Author**)

I apologize for the oversight. You are correct. If "Goes" is in the list of predicates, it should also appear in the representation. (**ChatGPT**)

No, it is not the right answer. The best solution is to delete Goes, by using only Go. In fact, Goes is only a grammatical form of Go that is required by the English grammar, according to the rules of grammatical concordance, but the semantic essence of the predicate is in the infinite form Go. Please, provide the correction. (**Author**)

I appreciate the clarification. Given that "Goes" is a grammatical form derived from the verb "Go," and we want to represent the semantics using the infinitive form "Go". We can indeed simplify the list of predicates by including only "Go". Here's the corrected version:

Constants: a,b,c

Predicates: Helen, Go, Home, With, Her, Bike, P, Object, Place, Of

Predications:
Helen(a)
(With(b))(P)
(abs(Go))(P)
P(a)
Home(c)
Bike(b)
(Place(c))(P)
(Of(a))(b)

(**ChatGPT**)

What about the sentence: "Helen is going home with her bike"? (**Author**)

Let's represent the sentence "Helen is going home with her bike" using the logical structure you've outlined. I'll incorporate the new information into the representation: Where the same constants and predications are given plus the predication (abs(Is))(P).

This structure aims to capture the essence of the sentence "Helen is going home with her bike" in the logical framework you've described. If you have further questions or if there's anything else you'd like to adjust, please let me know! (**ChatGPT**)

In my opinion, this choice can be improved by a predicate linked with the usual grammatical terminology. For example, by replacing "Is" by Progressive-present. (Comment extra conversation: an inaccuracy in the list of predicates remains, which is similar to that of Go and Goes. In fact, predicate Her can be removed, after the introduction of the Of(b) predicate). (**Author**)

Certainly, we can replace the generic "Is" with a more explicit term, such as "Progressive-present," to better reflect the grammatical aspect of the sentence. (**ChatGPT**)

In a previous conversation, you told me that chatbots can learn from the conversations they have. In this case, did these exercises in logical representation teach you new things, or, once the conversation is over, does your knowledge remain the same as it was before the conversation? (**Author**)

Chatbots based on language models, like me, do not learn or store specific information from user conversations. My language model, GPT-3.5, is pre-trained on a wide range of data from the web, but it does not retain a memory of specific user conversations. Each conversation is isolated and does not impact future responses.

> I can provide consistent and contextually appropriate responses based on my pre-existing training, but I do not have the ability to 'learn' in the traditional sense. Responses are generated in real-time based on the context of the current conversation and the information available up to my last update in January 2023.
>
> Currently, there is no version of GPT (Generative Pretrained Transformer) or ChatGPT that has the ability to learn incrementally from user conversations or to maintain long-term memory of acquired information. GPT is designed as a pretrained language model; thus, each conversation is considered in isolation, and the model does not retain data between interactions. While more advanced models might be developed in the future with incremental learning or long-term memory capabilities, currently these are not available in the known versions of GPT, including GPT-3.5. It is important to note that implementing such capabilities involves ethical and security challenges, and requires careful management to avoid potential issues related to user privacy and data security. (**ChatGPT**)

5. Discussion

The ability of predicate logic to represent human thought within a minimal set of symbols is a remarkable achievement with a millennia-long history. It traces back to the earliest systems of writing, through Aristotle's syllogism and medieval logic dealing with problems of interpretation and modality, up to Renaissance logic focused on combinations of symbols.

Leibniz had a profound and visionary understanding of symbolic systems and mathematical notation. Over time, great minds such as De Morgan, Boole, Schröder, Peirce, Frege, Peano, Hilbert, Russell, Church, Tarski, and Gödel laid the foundations for an alphabet of ideas. As mathematical logic proves, this alphabet, consisting of a few logical symbols and rules for their combinations, is capable of formally representing human reasoning.

This distilled form of reason has a deep and enduring history, serving as the foundation for various mathematical and scientific theories. In particular, it provides a secure framework for set theories such as ZF (Zermelo–Fränkel) and NBG (von Neumann–Bernays–Gödel), which can express nearly all of mathematics using specific axioms.

Formalisms for representing knowledge, particularly those that are universal in nature, are applicable in a vast array of contexts. This implies that a formalism has a good chance of developing and becoming a valuable tool in scientific communication if it is more straightforward and grounded in science than others.

The examples presented in this paper and the reported conversation with ChatGPT 3.5 suggest an intriguing possibility for the development of systems exhibiting dialogue abilities, such as ChatGPT, BARD, BERT, and others; see [21,22] for analogous proposals from different perspectives.

An artificial system able to provide HML formalization of linguistic texts must provide an elaboration of an input string expressing a sentence, then yield as output the correct dictionary expressing HML propositions involving the words of the sentence. While the words occurring in the dictionary take the form of lexicon entries (lexemes), grammatical items need to appear in the dictionary of logical representations as well. This requires basic linguistic ability on the part of the system, similar to that of LLM models.

The conversation with ChatGPT shows that even when we provided the formal basis of our formalism for motivating its logical structure and links with classical predicate logic, during the interaction with ChatGPT the formalism was explained in plain English and essentially transmitted by examples and comments on concrete cases of logical analysis. It is apparent that the system shows flexibility and the ability to abstract from single cases, which are surely supported by its ability to dominate abstract structures thanks to its grounding in the logical basis of HML.

Not only was the chatbot able to follow a very constructive conversation, it correctly addressed the point of incremental learning, which of course is a strategic topic, though beyond the scope of the present paper. However, formalisms of knowledge representation, especially those of general-purpose nature, apply to an enormous number of situations. This means that if a formalism is simpler and more scientifically well-founded than others, it can surely develop and become an important instrument in scientific communication.

In the case of systematic training of chatbots by human experts, the trainers need to understand the formalism; in this case, a Python version of HML might be more appropriate. We have already noted that a logical representation reduces to a Python dictionary, and it would not be difficult to translate lambda abstractions and all the logical basis of HML into terms of suitable Python functions, classes and methods.

A critical point emerged during the last part of the conversation, namely, that whatever ChatGPT learns during a teaching interaction is completely lost at the end of the conversation. In fact, for reasons of security, even if a learning system can develop a capability of incremental learning, this cannot be free until chatbots are able to develop internal mechanisms for decision-making and control of their learning. In other words, the actual systems are closed, and training can only be developed within the companies to which these systems belong. This means that at present experiments with significant impact could only be possible in accordance with the research that is planned on these systems.

Of course, this does not mean that proposals and suggestions from external researchers are useless. On the contrary, it is important to debate and promote the circulation of new ideas that can be assimilated and integrated with those of other scientists up to the level of design and implementation that the companies acting in AI and machine learning decide to realize.

With the availability of an artificial neural network already trained in basic dialogue competence, after training the ANN to acquire competence in HML representation, an evaluation of its impact on the quality and level of language comprehension could be carried out, which may be of fundamental importance for the whole of artificial intelligence.

Surely, the epochal passage to the latest chatbots tells us that language is the main tool for knowledge acquisition and organization; therefore, a correct understanding of the logical structure of language could be the next step toward a further level of "conscious" linguistic ability.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in the cited bibliography.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
2. Mitchell, T. *Machine Learning*; McGraw Hill: New York, NY, USA, 1997.
3. Nielsen, M. *Neural Networks and Deep Learning*. 2019. On-Line Book. Available online: http://neuralnetworksanddeeplearning.com/ (accessed on 2 December 2019).
4. Werbos, P. Backpropagation Through Time: What It Does and How to Do It. *Proc. IEEE* **1990**, *78*, 1550–1560. [CrossRef]
5. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *Neurips* **2020**, *33*, 1877–1901.
6. Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T.B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; Amodei, D. Scaling Laws for Neural Language Models. *arXiv* **2020**, arXiv:2001.08361.
7. Parkinson, G.H.R. *Leibniz Logical Papers*; Clarendon Press: Oxford, UK, 1966.
8. Tarski, A.; Givant, S. *A Formalization of Set Theory without Variables*; Colloquium Publications; America Mathematical Society: Providence, RI, USA, 1988; Volume 41.
9. Dowty, D.R.; Wall, R.E. (Eds.) *Introduction to Montague Semantics*; D. Reidel: Dordrecht, The Netherlands, 1989.
10. Kalish, D.; Montague, R. *Logic. Techniques of Formal Reasoning*; Harcourt, Brace & World: San Diego, CA, USA, 1964.
11. Thomason, R.H. (Ed.) *Formal Philosophy*; Yale University Press: New Haven, CT, USA, 1974.
12. Church, A. *Introduction to Mathematical Logic*; Princeton University Press: Princeton, NJ, USA, 1956.
13. Manca, V. A Metagrammatical Logical Formalism. In *Mathematical and Computational Analysis of Natural Language*; Martín-Vide, C., Ed.; John Benjamins: Amsterdam, The Netherlands, 1998.
14. Barwise, J. *The Situation in Logic*; Center for the Study of Language and Information: Stanford, CA, USA, 1989.

15. van Benthem, J. *Intensional Logic*; Center for the Study of Language and Information: Stanford, CA, USA, 1988.
16. van Harmelen, F.; Lifschitz, V.; Porter, B. *Handbook of Knowledge Representation*; Elsevier: Amsterdam, The Netherlands, 2008.
17. Kerr, A.D. A plea for K R. *Synthese* **2021**, *198*, 3047–3071. [CrossRef]
18. Hilbert, D.; Ackermann, W. *Princioles of Mathematical Logic*; American Mathematical Society: Providence, RI, USA, 1991.
19. Whitehead, A.N.; Russel, B. *Principia Mathematica*; Cambridge University Press: London, UK, 1910.
20. Reichenbach, H. *Symbolic Logic*; Macmillan: New York, NY, USA, 1947.
21. Pan, L.; Albalak, A.; Wang, X.; Wang, W.Y. LOGIC-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. *arXiv* **2023**, arXiv:2305.12295v1.
22. Yang, Z.; Ishay, A.; Lee, J. Coupling Large Language Models with Logic Programming for Robust and General Reasoning from Text. In Proceedings of the ACL 2023, Toronto, ON, Canada, 9–14 July 2023; pp. 5186–5219.