# Constructive Semantics for Extensional PTQ

Rogelio Dávila Pérez
Computer Science Department
University of Texas at El Paso
El Paso, Texas, USA, 79968-0518

**Paper ID:** P0416

**Keywords:** formal semantics, computational semantics, natural language processing.

**Contact Author:** rdav9@hotmail.com

**Under consideration for other conferences (specify)?** none

**Abstract**

In this paper, the syntax and semantics for a fragment of English will be provided within the framework of Martin-Löf's Type Theory. The first to explore the potential benefits of this application was Aarne Ranta (Ranta, 1991). The motivation for the present work is to develop a more traditional approach (Montague-style semantics) to semantics based upon the constructive framework. The work is focused on the parsing problem, which consists of providing the interpretation of a given English sentence as an expression in the Type Theory. In this paper it is shown that the parsing and semantic interpretation of a sentence can be accomplished in a compositional fashion by defining semantic rules that work in a one-to-one correspondence with the syntactic ones.

# Constructive Semantics for Extensional PTQ

Paper ID: P0416

**Abstract**

In this paper, the syntax and semantics for a fragment of English will be provided within the framework of Martin-Löf's Type Theory. The first to explore the potential benefits of this application was Aarne Ranta (Ranta, 1991). The motivation for the present work is to develop a more traditional approach (Montague-style semantics) to semantics based upon the constructive framework. The work is focused on the parsing problem, which consists of providing the interpretation of a given English sentence as an expression in the Type Theory. In this paper it is shown that the parsing and semantic interpretation of a sentence can be accomplished in a compositional fashion by defining semantic rules that work in a one-to-one correspondence with the syntactic ones.

## 1 Introduction

In his Intuitionistic Categorial Grammar (ICG) works (Ranta, 1991, 1994), Ranta provided the basis for a categorization of English in terms of Constructive Type Theory (CTT). In his work, Ranta concentrated on the problem of generation (what he refers to as *sugaring*), which consists of generating plausible English translations of expressions in the CTT. From his work it is not clear how we can accomplish the converse process. The aim of the present work is to show that it is possible to provide a constructive semantic interpretation of a fragment of English in a compositional way, by defining semantic rules that work in a one-to-one correspondance with the syntactic ones.

The present work is based on Martin-Löf's Type Theory (MLTT)(Martin-Löf, 1984); it consists of an Intuitionistic Type Theory that originally was developed to serve as a framework for constructive mathematics. The theory is a theory of judgments instead of a theory of propositions, where a judgment is the assertion of the truth of a proposition. MLTT has been used as a framework for undertaking constructive mathematics and intuitionistic logic[1].

The following sections present an intuitionistic categorial grammar for a fragment of English. We offer a categorization of some English constructions in the Type Theory and then by defining a set of grammar rules in a way similar to the one Montague uses to introduce his categorial grammar in PTQ (Montague, 1974b). These rules define combinators that map parse-trees into expressions in the Type Theory. At this stage we are not particularly concerned with intensionality and will make reference to this concept only when it is required to understand the topic under discussion.

## 2 Constructive vs. Classical Approaches

Before engaging in building an antirealistic characterization of natural language semantics, it is worthwhile to point out some of the important differences between this approach and the classical (or realistic) ones. The main difference comes from the constructive nature of the theory; that is, the semantics will be based on a proof-theoretic approach rather than the traditional truth-theoretic one. This means that even when some of the categorizations of the natural language structures presented in this paper look very similar to those introduced by Montague in his PTQ paper (Montague, 1974b), they are intrinsically different, because they are associated with proof-objects rather

[1] A good introduction to the Type Theory in relation to natural language is found in (Ranta, 1994)

than truth-conditions. As shown in the following sections, all natural language interpretations are provided at the level of types. That is, every sentence will define a structured type that we call its semantic interpretation. The structure of the type associated with each sentence will define the form that a proof object that makes that sentence true should have. Using the notion of *context* in MLTT, it will be possible to tie together the meaning of various sentences into complex types. These complex types will define structured proof objects that are combinations of the proof objects corresponding to the different sentences in the fragment under analysis. This way of structuring the meanings of various sentences into a more complex one makes it possible to recover any of the particular meanings of the sentences involved from the meaning of the whole fragment. This fact will have important implications in the analysis of discourse.

Another important difference comes from the fact that classical approaches to semantics depend on what is called the *Domain*, that is, a set to which it is assumed all individuals belong to. In the Type Theory, being a constructive set theory, every individual belongs to a particular set (e.g. the set of men, women, tables, houses, etc.), but since in order to define a set we need to provide all its cannonical elements, it is not possible to build such a *Domain*. This has important consequences when a categorization is provided for some of the constituents of language (e.g., nouns).

In the following sections we introduce a categorial grammar for a fragment of English based on these ideas.

## 3 The Type Structure of the Language

Using a categorial grammar in the formalization of a language (or a fragment of it) requires one to differenciate among the constituents in the language; traditionally this is achieved by developing a hierarchy of types. In this work,we use Martin-Löf's Monomorphic Type Theory[2] as a logical framework. This provides a rich range of types on which to structure our fragment.

In order to build our type hierarchy, we introduce a basic type: the type of sets (propositions). We use the judgmental form

$$set\ type$$

to state that *set* ( *prop* ) is a type. Then we extend our basic types, by introducing the general rule

$$\frac{X :\ set}{X :\ type}$$

which states that any element of type $set\tilde{(}\ \ prop\tilde{)}$ is a type. That is, any set defined within the theory will be treated as a type. This provides our framework with a rich range of basic types on which to build our hierarchy.[3]

Now, we need to provide a way of building function types. To this end, we recall the rules of inference for functional types introduced in MLTT:

Formation

$$\frac{A\ type \qquad B\ type\ [x : A]}{(x : A)B\ type}$$

Abstraction

$$\frac{A\ type \quad b : B\ [x : A]}{(x)b : (x : A)B}$$

Application

$$\frac{a : A \qquad c : (x : A)B}{c(a) : B(x/a)}$$

$\beta$-Conversion

$$\frac{a : A \qquad b : B\ [x : A]}{((x)b)(a) = b(x/a) : B(x/a)}$$

Using these rules we can define complex types from our basic ones. For example, suppose that we have the following judgments:

[2]The Monomorphic Type Theory is is presented in (Nordstrom et al., 1990)

[3]This is true, at least when compared with Russell's simple type theory, where the basic types are the type of individuals $e$ and the type of true values $t$. Montague's Intensional Logic (Montague, 1974a) adds to these basic ones the type of intensional objects $s$.

$john : Man$
$john\ whistles :\ prop$

The first judgment asserts that *john* is an element of the set (type) $Man$, and the second that *john whistles* is a proposition. If then we apply the abstraction rule to the second judgment, we obtain a structured object of the form

$(x)\ x\ whistles : (x : Man)\ prop$

which indicates that the result is a propositional function on the set $Man$. This means that if we have a judgment of the form,

$peter : Man$

we may apply *whistles* to *peter* using the application rule to obtain

$peter\ whistles :\ prop$

giving a new object of type *prop*. We may proceed to build more complex objects in this fashion. For example from the sentence

$a\ man\ whistles :\ prop$

we may abstract on the object *whistles* and get a new object of the form,

$(Y)\ a\ man\ Y : (Y : (Man)\ prop)\ prop$

where $(Y)\ a\ man\ Y$ is a function such that given an argument of type $(Man)\ prop$, renders an object of type *prop*. Then we may abstract on *man* obtaining the complex object

$(X)(Y)\ \mathrm{a}\ X\ Y :$
$(X :\ set)(Y : (X)\ prop)\ prop$

which, when applied to a set $X$ and an intransitive verb $Y$ of type $(X)\ prop$, renders an object of type *prop*. This is precisely the type of the set constructors $\Pi$ and $\Sigma$.

## 4 Basic Categories of English

The Monomorphic Type Theory (Nordstrom et al., 1990), provides a logical framework with a rich hierarchy of type structures in which to formalize our fragment of English. We proceed in this section to categorize our fragment. This means that we must assign suitable types to the different words in the fragment in order to capture the actual contribution of each word to the meaning of the sentences in which it takes part.

Traditionally, words are grouped into syntactic categories. These categories are important because they reveal the internal structure of the sentence. Intuitively we would expect words grouped in the same syntactic category to have a similar contribution to the semantics of sentences. As we know, this is not always the case (especially when talking about determiners), but even so the syntactic categories provide important guide-lines for the process of categorization.

We now assign constructive types to some of the basic syntactic categories in English.

### 4.1 Sentences and Common Nouns

Classically, sentences have been identified with propositions, objects about which it makes sense to say that they are true or false. We follow the tradition and assign to sentences the type of propositions. Hence, we may introduce judgments with the form

$the\ man\ works\ in\ a\ factory :\ prop$
$a\ kid\ broke\ the\ window :\ prop$

We must remember that we are talking about propositons in the constructive sense, that is, as proof-theoretic objects. This means that in order to show that a sentence is true, we must provide a proof of that sentence. The form of that proof (or proof object) will depend on the actual structure of the sentence, as will be shown in the following sections.

In the Type Theory, we identify propositions with sets (according to the Curry-Howard's *propositions as types (sets)* correspondence (Howard, 1980)); then we also interpret sentences as sets. This fact brings some interesting results to the analysis of language, associated with discourse and cross-sentencial anaphora.

In our common use of language we normally identify nouns like *man, donkey, dog, ...* with sets of individuals:

$man = \{john,\ peter,\ paul,\ \ldots\}$

$donkey = \{chiquita, \ldots\}$
$dog = \{fido, lassie, \ldots\}$

In the classical tradition we generally categorize nouns as propositional functions. This categorization has its roots on the Model-Theoretic semantics. In model theory, we rely on the existence of a set called the *Domain*, from which we can take elements which may or may not satisfy the property of being in the set which is the referent of the noun (men, donkeys, dogs, etc.). In the Type Theory, syntax and semantics go together as a whole, and we do not have such a thing as the *Domain*; instead, every individual belongs to a particular set. Hence, a natural way of categorizing nouns in the Type Theory is as sets. That is, nouns introduce particular domains for quantification. We realize this idea introducing judgments of the form

$man : set$
$donkey : set$
$dog : set$

This idea also applies to modified nouns of the form

$man\ who\ loves\ mary: set$
$dog\ that\ has\ a\ collar : set$

In MLTT propositions and sets are considered to define the same type. Therefore, nouns and sentences belong to the same type within the system.

### 4.2 Determiners

The natural language determiners, such as *every, some and a*, have a complex structure, as shown in the example in Section 3. There we obtained for the determiner *a* the functional type

$$a : (X : set)(Y : (X)\ prop)\ prop$$

This is the type of the set-constructors operators $\Pi$ and $\Sigma$ in MLTT. This fact is not strange, as they are precisely the constructive counterparts of the classical quantifiers *forall* and *exists*.

In categorizing the English quantifiers, we assume that we have a countable (possibly infinite) sequence of variables of the form $x_n (n = 1, 2, 3, ...)$ to stand as arbitrary objects of the set over which the quantifier acts. The categorization is

$every = (X, Y)\Pi(X, (x_n)Y(x_n)) :$
$\quad (X : set)(Y : (X)\ prop)\ prop$
$a,\ some = (X, Y)\Sigma(X, (x_n)Y(x_n)) :$
$\quad (X : set)(Y : (X)\ prop)\ prop$

These categorizations work well when applying the quantifiers to atomic sets (common nouns) as follows:
$every\ man =$
$\quad (Y)\Pi(man, (x)Y(x)) : ((man)\ prop)\ prop$
a car $=$
$\quad (Y)\Sigma(car, (x)Y(x)) : ((car)\ prop)\ prop$
where the noun phrase *every man* is a propositional function which takes as an argument a function of the type $(man)\ prop$, that is, a property of men (e.g. *drive, fight and talk*). Now, if we introduce a modified noun, such as *man who drives*, and we apply the quantifier *every* to it, we get something like

$every\ man\ who\ drives =$
$\quad (Y)\Pi(man\ who\ drives, (x)Y(x)) :$
$\quad\quad ((man\ who\ drives)\ prop)\ prop$

As will be shown in Section 4.6, the modified noun now has the structured type:

$man\ who\ drives =$
$\quad \Sigma(man, (z)drives(z)) : set$

Hence the noun phrase *every man who drives* will be analyzed as

$every\ man\ who\ drives =$
$(Y)\Pi(\Sigma(man, (z)drives(z)), (x)Y(x)) :$
$\quad ((\Sigma(man, (z)drives(z)))\ prop)\ prop$

which again is a function expecting a function as an argument. Now the argument is of type:

$$(\Sigma(man, (z)drives(z)))\ prop$$

If we try a possible continuation to the noun phrase, such as the verb *whistles*, attempting to complete the sentence

$$every\ man\ who\ drives\ whistles$$

we realize that the type of *whistles*, which we may consider to be something like

$whistles : (man)\ prop$

does not corresponds to the type of the argument of the noun phrase *every man who drives*, $(\Sigma(man, (z)drives(z)))\ prop$. Therefore, we have a type-mismatch.

In order to give a solution to this problem, observe that every modified noun is interpreted using the $\Sigma$–constructor[4]. The noun-modifiers (in this case relative clauses) restrict the set defined by the noun by adding constraints. Now, as we know, a $\Sigma$-constructed set allows us to take elements from inside by means of the $\mathsf{p}()$ and $\mathsf{q}()$ operators defined by the $\Sigma$–elimination rules. Thus, it is possible to obtain from a structured object of type $\Sigma$ the part which corresponds to the original noun from the modified expression. For the example shown above, it would be

$\mathsf{p}(\mathrm{w}) : \mathrm{man}$
*where:* $\mathrm{w}{:}\Sigma(man, (z)drives(z))$

We propose to introduce a type-matching function app, defined as

$\mathsf{app}(\mathrm{X},\mathrm{Y},\mathrm{V}) =$

$$\begin{cases} \mathsf{app}(A, Y, \mathsf{p}(V)) & \text{if } X \text{ has the form } \Sigma(A, B) \\ Y(V) : prop & \text{otherwise} \end{cases}$$

where $X : set, Y : (X)\ prop$ and $V : X$.

This function will convert a propositional function defined on an atomic set, such as those corresponding to verb phrases and other linguistic constructions, into a function defined on a structured type. With the definition of the type-matching function app we do not have to modify our categorization of the English determiners, as this function should operate at the level of the grammar rules. We will see this when introducing the *Determiner-Noun Rule* in Section 5.4, where we take care to provide the function app with the right arguments.

### 4.3 Proper Names

(Ranta, 1991) takes proper names to be individuals of a particular set, for example,

$John : man$
$Mary : woman$

This treatment seems very natural but it makes it hard to distinguish between the *Sense* (meaning) of a proper name like *John* and its *Reference*, the name holder, that is, the actual individual whose name is *John*. This is a difficult point, involving much philosophical controversy. We follow Montague (Montague, 1974b), in taking the *Sense* of a name as the set of properties that the name holder has. This approach can be formalized as

$$John = (P)P(john) : (P : (man)\ prop)\ prop$$
$$Mary = (P)P(mary) : (P : (woman)\ prop)\ prop$$

That is, *john* is the individual of type *man* whose name is *John*, and *mary* is the actual woman called *Mary*.

### 4.4 Intransitive and Transitive Verbs

Intransitive verbs like *walk and talk*, behave in general as propositional functions, that is, objects which, when given an individual as an argument, form a proposition. To categorize verbs as propositional functions in a general way, we must include as part of the type, the name of the particular set that individual belongs to. This can be formalized in two different ways; the first makes use of the *context* and is shown below:

$$walk = walks : (X)\ prop\ [X : set]$$
$$talk = talks : (X)\ prop\ [X : set]$$

The transitive verbs such as *love, hate and beat,* require two arguments to form propositions. We may reflect this requirement on their types as

$$love = loves : (X, Y)\ prop\ [X : set, Y : set]$$
$$hate = hates : (X, Y)\ prop\ [X : set, Y : set]$$

### 4.5 The verb Be

The verb *be*, despite being a transitive verb, deserves a rather different treatment, as it introduces equality among individuals. Let us look at an example:

[4]The disjoin union operator $\Sigma$ is the constructor we used to define the constructive counterpart of classical conjunction (Martin-Löf, 1984)).

*Sally is a woman,*

In strict terms this sentence should be interpreted as a judgment of the form

$$sally : woman$$

where *sally* is the referent of the name *Sally* (see Section 4.3) and *woman* is a set. But we can not introduce judgments in this way, as we would be asserting that *sally* is an object of the set *woman* without having a proof of it. What we really want is to interpret *Sally is a woman* as an object of type proposition, something like

$$Sally\ is\ a\ woman : prop$$

which is a proposition that requires a proof object to become true. For building the representation of such a proposition we must interpret the verb *be* as an equality at the level of propositions, not at the level of judgments. We can do that by tying the meaning of *be* to the equality proposition, using the following judgment

$$be = (x, y)\mathsf{I}(A, x, y) : \\ (x : A, y : A)prop[A : set]$$

which captures the intuitive meaning of the verb *be*.

### 4.6 Relative Pronouns

The relative pronouns (*such that, who, which* and *that*), allow the generation of modified noun phrases. Their function in language is to restrict the scope of the noun by adding constrains to it. They add information to the noun by introducing relative clauses. There are two different categories of relative pronouns, according to their arguments. The first one is represented by the pronoun *such that*. This relative pronoun allows the introduction of a full sentence as a modifier, as these examples show:

*woman such that she loves Harry,*
*donkey such that it carries the corn sacks,*

This pronoun is categorized as

$$such\ that = (X)(Y)\Sigma(X, (x_n)Y) : \\ (X : set)(Y : prop)\ set$$

The second category, which includes the relative pronouns *who, which and that*, allows the use of a verb phrase as a modifier. Some examples are

*man who owns a donkey,*
*dog which bites the cat,*

The following categorization is for these relative pronouns:

$$who,\ that,\ which = \\ (X)(Y)\Sigma(X, (x_n)Y(x_n)) : \\ (X : set)((X)\ prop)\ set$$

The same type-mismatching problem that we had with determiners and definite noun phrases occurs when we have a modified noun with an additional relative clause, so the rule for combining nouns with relative clauses must include the function app for their interpretation[5].

In the next section we introduce the rules for the grammar and present some examples of how they are used in the analysis of language.

## 5 Grammar Rules

### 5.1 Determiner-Noun Rule

This rule corresponds to Montague's (S2) rule in PTQ.

S2. If $\alpha$ is in D and $\beta$ is in N then $\mathcal{F}_2(\alpha\ ,\beta\ )$ is in NP (noun phrase) , where $\mathcal{F}_2(\alpha\ ,\beta\ ) = \alpha\ \beta$.

The corresponding translation rule is

T2.

$$\frac{\alpha : (X : set)((X)\ prop)\ prop \qquad \beta : set}{(Z)\alpha'(\beta', (z)\mathsf{app}(\beta', Z, z)) : ((\beta)\ prop)\ prop}$$

where $\alpha$ is in D, $\beta$ is in N, $\alpha, \beta$ translate into $\alpha', \beta'$ respectively, and app is the type-matching function defined on Section 4.2. For example,

$$every\ man = (Z)\Pi(man, (x_0)Z(x_0))$$

### 5.2 Subject-Predicate Rule

This rule corresponds to Montague's (S4) rule in PTQ:

S4. If $\zeta$ is in NP and $\delta$ is in IV then $\mathcal{F}_2(\zeta\ ,\delta\ )$ is in S.

[5] See Section 5.4

The associated translation rule is:

T4.

$$\frac{\zeta : ((X : set)prop)prop \quad \delta : (Y)prop\,[Y : set]}{\zeta'(\delta') : \ prop\,[Y = X]}$$

where $\zeta$ is in NP, $\delta$ is in IV and $\zeta$ , $\delta$ translate into $\zeta', \delta'$ respectively. For example:

*every man walks* =
$\Pi(man, (x_0)walks(x_0))$

### 5.3 Transitive Verbs Rule

This rule corresponds to Montague's (S5) rule in PTQ.

S5. If $\alpha$ is in $TV$ and $\zeta$ is in $NP$ then $\mathcal{F}_2(\alpha, \zeta)$ is in $IV$.

The corresponding translation rule is the following:

T5.

$\alpha : (X)(Y)\ prop\ [X, Y : \ set]$
$\zeta : ((Z : \ set)\ prop)\ prop$

---

$(\mathrm{w})\zeta'(\alpha'(w)) : (X)\ prop\ [X : \ set, Y = Z]$

where $\alpha$ is in $TV$, $\zeta$ is in $NP$ and $\alpha, \zeta$ translate into $\alpha', \zeta'$ respectively. For example:

*every man loves a woman*=
$\Pi(man, (x_0)\Sigma(woman, (x_1)loves(x_0, x_1))))$

### 5.4 Relative Clauses Rules

This rule corresponds to Montague's (S3) rule in PTQ:

S3. If $\delta$ is in N, $\xi$ is in RP, and $\phi$ is in S then $\mathcal{F}_3(\delta, \xi, \phi)$ is in N, where $\mathcal{F}_3(\delta\ , \xi\ , \phi\ ) = \delta\ \xi\ \phi$.

The rule S3(a) is not part of PTQ; it introduces another variant of relative clauses:

S3. (a) $If \delta$ is in N, $\xi$ is in RP, and $\sigma$ is in IV, then $\mathcal{F}_3(\delta\ , \xi\ , \sigma\ )$ is in N.

The corresponding translation rules are

T3. (a)

$$\frac{\delta : \ set \quad \xi : (X : \ set)(Y : \ prop)\ set \quad \phi : \ prop}{\xi'(\delta')(\phi') : \ set}$$

where $\delta$ is in N, $\xi$ is in RP, $\phi$ is in S, and $\delta, \xi, \phi$ translate into $\delta', \xi', \phi'$ respectively.

T3. (b)

$\delta : \ set$
$\xi : (X : \ set)((X)\ prop)\ set$
$\sigma : (Z)\ prop[Z : \ set]$

---

$\xi'(\delta', (z)\mathsf{app}(\delta', \sigma', z) : \ set$

where $\delta$ is in N, $\xi$ is in RP, $\sigma$ is in VP, $\delta, \xi, \sigma$ translate into $\delta', \xi', \sigma'$ respectively, and app is the type-matching function defined in Section 4.2. For example

*farmer who owns a donkey*
$= \Sigma(farmer, (x_1)\Sigma(donkey, (x_2)owns(x_1, x_2)))$

### 5.5 Proper Names Rule

This rule corresponds only partially to Montague's (S1) rule in PTQ.

S1.b If $\xi$ is in PN, then $\mathcal{F}_1(\xi)$ is in NP.

The associated translation rule is

T1.(b)

$$\frac{\xi : ((X : \ set)\ prop)\ prop}{\xi' : ((X : \ set)\ prop)\ prop}$$

where $\xi$ is in PN, and $\xi$ translates into $\xi'$, and $\xi' : X$. For example:

*John loves Mary*
$= loves(john, mary)$

### 5.6 Conjunction and Disjunction Rules

These rules correspond to Montague's (S11, S12 and S13) rules in PTQ.

S11. If $\phi$ is in S and $\psi$ is in S then $\mathcal{F}_{11}(\phi, \psi)$ is in S, where
$\mathcal{F}_{11}(\phi, \psi) = \phi\ and\ \psi$, $\mathcal{F}_{11}(\phi, \psi) = \phi\ or\ \psi$.

S12. If $\beta$ is in IV and $\eta$ is in IV then $\mathcal{F}_{11}(\beta, \eta)$ is in IV.

S13. If $\delta$ is in NP and $\rho$ is in NP then $\mathcal{F}_{11}(\delta, \rho)$ is in NP.

The corresponding translation rules are[6]

T11.

$$\frac{\phi : \mathit{prop} \quad \psi : \mathit{prop}}{\begin{array}{l}\phi' \wedge \psi' : \mathit{prop} \\ \phi' \vee \psi' : \mathit{prop}\end{array}}$$

*where* $\phi$ is in $S$, $\psi$ is in $S$ and $\phi$, $\psi$ translate into $\phi', \psi'$ respectively.

T12.

$$\frac{\begin{array}{l}\beta : (X)\ \mathit{prop}\,[X : \mathit{set}] \\ \eta : (Y)\ \mathit{prop}\,[Y : \mathit{set}]\end{array}}{\begin{array}{l}(\mathrm{z})\ (\ \beta'(z) \wedge \eta'(z)) : \\ \quad (X)\ \mathit{prop}\,[X : \mathit{set}, Y = X] \\ (\mathrm{z})(\beta'(z) \vee \eta'(z)) : \\ \quad (X)\ \mathit{prop}\,[X : \mathit{set}, Y = X]\end{array}}$$

*where* $\beta$ is in IV, $\eta$ is in IV and $\beta$, $\eta$ translate into $\beta', \eta'$ respectively.

T13.

$$\frac{\begin{array}{l}\delta : ((X : \mathit{set})\ \mathit{prop})\ \mathit{prop} \\ \rho : ((Y : \mathit{set})\ \mathit{prop})\ \mathit{prop}\end{array}}{\begin{array}{l}(\mathrm{Z})\ (\delta'(Z) \wedge \rho'(Z)) : \\ \quad ((X : \mathit{set})\ \mathit{prop})\ \mathit{prop}\,[Y = X] \\ (\mathrm{Z})\ (\delta'(Z) \vee \rho'(Z)) : \\ \quad ((X : \mathit{set})\ \mathit{prop})\ \mathit{prop}\,[Y = X]\end{array}}$$

*where* $\delta$ is in NP, $\rho$ is in NP and $\delta$ , $\rho$ translate into $\delta', \rho'$ respectively. For example:

*a man walks and talks*
$= \Sigma(man, (x_0)walks(x_0) \wedge talks(x_0))$

## 6 Final Remarks

In this work we have succeeded in building the constructive interpretation of our fragment of English. A categorization of English words has been provided in terms of the Monomorphic Type Theory. These categorizations attempt to capture the contribution of the meaning of each word to the meaning of the sentences in which they occur.

A grammar for a fragment of English has been proposed which provides for a compositional analysis of the fragment. This grammar consists of a set of syntactic and semantic rules in a one-to-one correspondence. Every semantic rule defines a combinator which produces a semantic interpretation of its corresponding syntactic rule as an expression of the type theory.

[6] In these definitions, the constructive counterparts of the conjunction and disjunction operators are used.

## References


W. A. Howard. 1980. The Formulae-as-Types Notion of Construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, London.

P. Martin-Löf. 1984. Intuitionistic Type Theory. Napoli. Studies in Proof Theory (Lecture Notes).

R. Montague. 1974a. Pragmatics and Intensional Logic. In R. H. Thomason, editor, *Formal Philosophy. Selected Papers of Richard Montague.* Yale University Press, New Haven.

R. Montague. 1974b. The Proper Treatment of Quantification in Ordinary English. In R. H. Thomason, editor, *Formal Philosophy. Selected Papers of Richard Montague.* Yale University Press, New Haven.

B. Nordstrom, K. Petersson, and J. M. Smith. 1990. *Programming in Martin-Löf's Type Theory: An Introduction.* Clarendon Press, Oxford.

A. Ranta. 1991. Intuitionistic Categorial Grammar. *Linguistics and Philosophy*, 14:203–239.

A. Ranta. 1994. *Type-Theoretical Grammar.* Oxford University Press, Oxford.