From Logic to Grammars via Types*

Daniela Dudau-Sofronie, Isabelle Tellier, Marc Tommasi

LIFL-Grappa Team and Université Charles de Gaulle-lille3
59 653 Villeneuve d'Ascq Cedex, FRANCE
dudau@lifl.fr, tellier,tommasi@univ-lille3.fr
http://www.grappa.univ-lille3.fr

Abstract. This paper investigates the inference of Categorial Grammars from a new perspective. To learn such grammars, Kanazawa's approach consists in providing, as input, information about the structure of derivation trees in the target grammar. But this information is hardly arguable as relevant data from a psycholinguistic point of view. We propose instead to provide information about the *semantic type* associated with the words used. These types are considered as general semantic knowledge and their availability is argued. A new learning algorithm from types is given and discussed.

1 Introduction

Learning a foreign language from texts is like trying to decipher hieroglyphs without the Rosetta stone: it is an unreachable challenge. Without the indications about their meaning found in the stone, hieroglyphs would probably still remain mysterious. This paper can be interpreted as a tentative explanation of how semantics can help syntax learning.

Categorial Grammars are well known for their formalized connection with semantics ([Mon74], [DWP81]). They provide a good compromise between formalism and linguistic expressivity ([OBW88]). Previous works have studied the learnability of such grammars ([Adr92], [Kan98], [MO98]) but neither of them uses the syntax/semantics interface to help the syntactic learning process.

Links between Kanazawa's learning strategy and semantic information have been shown in [Tel99]. This first approach is still not satisfactory as it does not avoid combinatorial explosion. This paper is a new way of considering learning Categorial Grammars from semantic knowledge. The original contribution consists in the use of *semantic types* associated with words.

Types are very general information, allowing to distinguish facts from entities and from properties satisfied by entities. Most knowledge representation languages use this notion, and it ususally seems possible to deduce types from lexical semantics. A new learning algorithm taking as input data both syntactically correct sentences and the corresponding sequences of types is explained.

Sections 2 and 3 are preliminaries introducing the class of Grammars we want to learn and the nature of the information admitted as input to the learning

* this work was supported by the ARC INRIA project GRACQ

process. Section 4 exposes the heart of our proposition, illustrated with a detailed example, and criticizes its limitations.

The conclusion argues about the cognitive plausibility of the data provided to this algorithm, in comparision with the data usually used in other learning algorithms. Perspective issues are also evoked.

2 Grammatical Inference

The Problem of *Grammatical Inference* from positive examples (or: from texts) consists in the design of algorithms able to identify a formal grammar from a sample of sentences it generates. This problem is a rough formal approximation of how children manage to learn the grammar of their mother language ([WC80]). From a theoretical point of view, identifying a formal grammar from texts is very difficult since regular (and therefore context-free) grammars are not learnable from texts in usual learning models ([Gol67,Val84]).

2.1 Categorial Grammars

In the following, we will use formal grammars belonging to the class of *AB-Categorial Grammars* ([HGS60]). In this formalism, every member of the vocabulary (every word) is associated with a finite set of categories, expressing its combinatorial power. The set C' of every possible category is built from a finite set C of basic categories in the following way: C' is the smallest set satisfying $C \subset C'$ and for every $X \in C'$ and $Y \in C'$ then $(X/Y) \in C'$ and $(Y\backslash X) \in C'$[1].

Syntactic rules are reduced to the following rewriting schemas (where "." denotes the concatenation operation): for every category X and Y in C'

- **FA** (Forward Application) : $(X/Y).Y \rightarrow X$;
- **BA** (Backward Application) : $Y.(Y\backslash X) \rightarrow X$.

These schemas justify the fractional notations of the operators / and \. Categories of the form X/Y (resp. $Y\backslash X$) can be considered as *oriented functors* expecting an argument of category Y on their right (resp. on their left) and providing a result of category X. Given a vocabulary Σ, a Categorial Grammar G is defined by an axiomatic category S and an association between words in Σ and categories in C'. The language recognized by G is the set of finite concatenations of elements of Σ for which there exists an assignment of categories that can be *rewritten* with the schemas into S. The class of (string) languages that are recognizable by Categorial Grammars is the class of context-free languages.

Example 1 (A basic Categorial Grammar). Let us define a Categorial Grammar for the analysis of a small subset of natural language whose vocabulary is {a, man, woman, John, runs, walks, fast}. The set of basic categories is $C = \{S, T, CN\}$ where S is the axiomatic category of sentences, T stands for

[1] For a sake of clarity, parentheses are omitted in non ambiguous expressions

term and CN means *common noun*. The assignment of categories to words is:
John: T; man, woman: CN; runs, walks: $(T\backslash S)$;
a: $((S/(T\backslash S))/CN)$; fast: $((T\backslash S)\backslash(T\backslash S))$.

This grammar allows to recognize sentences like : *"a man runs"* and *"John runs fast"* because:

$$\begin{array}{cccc} a & man & runs & \text{(twice with the } \mathbf{FA} \text{ rule)} \\ ((S/(T\backslash S))/CN) & . \ \text{CN} & . \ (T\backslash S) \rightarrow (S/(T\backslash S)) & . \ (T\backslash S) \rightarrow S \end{array}$$

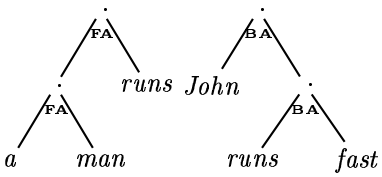
$$\begin{array}{ccccc} John & runs & fast & \text{(twice with the } \mathbf{BA} \text{ rule)} \\ T & . \ (T\backslash S) & . \ ((T\backslash S)\backslash(T\backslash S)) & \rightarrow T \ . \ (T\backslash S) \rightarrow S \end{array}$$

2.2 Grammatical Inference of Categorial Grammars

Learning a Categorial Grammar consists in identifying the categories assigned to each word of its vocabulary. The rewriting schemas are supposed to be known.

The formal learnability of Categorial Grammars has been studied in a variant of the PAC model ([Adr92]) and in Gold's model ([Kan96,Kan98]). The most powerful result obtained uses the notion of *Structural Example*. A Structural Example is derived from a syntactic analysis structure by deleting intermediate categories while preserving the terminal symbols and the reduction schemas used.

Example 2. The Structural Examples dervived from analyses of Example 1 are:



Kanazawa has proved that the class $\mathcal{G}_k$ of Categorial Grammars assigning at most k different categories with any given word is learnable in Gold's model from positive Structural Examples ([Kan96,Kan98]). Basically, the learning strategy, extending an algorithm earlier proposed by [BP90], relies on the unification of variable categories assigned to each nodes of the Structural Examples. When $k = 1$, the grammars are called "rigid" and they can be efficiently learned. When $k > 1$, the learnability is NP-hard ([Flo00]).

The main problem in this approach is that Structural Examples are hardly arguable as relevant data from a psycholinguistic point of view. The learnability result from Structural Examples can be extended to a learnability result from texts, but in this case the algorithm first needs to enumerate every possible tree structure corresponding with every string of words provided as example, and thus becomes exponential. Efficiency and cognitive plausibility don't seem to go together well.

Our purpose is also to learn Categorial Grammars, but from a new kind of input data, more informative than texts and more relevant than Structural Examples, i.e. *sequences of types*. Types will be associated with words and can be interpreted as semantic information.

3 Semantic information

3.1 A typing system

A well known interest of Categorial Grammars is their connection with semantics. This connection, first formalized by Montague ([Mon74,DWP81]) is inspired by the Principle of Compositionality ([Jan97]). One of its consequences is a strong correspondence between syntactic categories and semantic types. These types are what we propose to use in the learning process.

We will not define here a full semantic language, as types can be associated with very different kinds of semantic representation (the next subsection will provide clues to extract types from a usual semantic language). For us, the only semantic information needed will be a typing system making a basic distinction between *entities* and *facts* and allowing to express the types of *functors*, among which are the *predicates*. This typing system is a un-intensional version of Montague's intensional logic. The set Θ of possible types is defined by:

- elementary types : $e \in \Theta$ (type of entities) and $t \in \Theta$ (type of truth values) are the elementary types of Θ;
- Θ is the smallest set including every elementary type and satisfying : if $u \in \Theta$ and $v \in \Theta$ then $\langle u, v \rangle \in \Theta$ (the composed type $\langle u, v \rangle$ is the type of functors taking an argument of type u and providing a result of type v).

These types combine following rewriting rules similar with the ones of 2.1.

- **TF** (Type Forward) : $\langle u, v \rangle \ . \ u \rightarrow v$;
- **TB** (Type Backward) : $u \ . \ \langle u, v \rangle \rightarrow v$.

Example 3. For the grammar of Example 1, the types of the words are:
John: e; man, woman, runs, walks: $\langle e, t \rangle$; a: $\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$; fast: $\langle \langle e, t \rangle, \langle e, t \rangle \rangle$.
These types express that "John" denotes an entity; "man", "woman", "runs" and "walks" characterize one-place predicates and "fast" is a one-place predicate modifier. The type of "a" comes from its semantic translation in Montague's system (see 3.2).

The main difference between categories in C' and types in Θ is that the *direction* of a functor-argument application in categories is indicated by the operator used ($/$ or $\backslash$), whereas this direction is lost in types (as both are replaced by a ","). The functor-argument application of types is commutative, whereas it is not for categories. Note that for a given type containing n elementary types, there are 2^{n-1} possible ways to replace each of its $n-1$ "," by either / or \.

Another difference is that in our typing system, the set of elementary types is reduced to $\{e, t\}$, whereas it just needs to be finite in Categorial Grammars.

3.2 From logic to types

Before entering the learning process, let us slightly deepen the links between semantics and types. As a matter of fact it seems possible, under conditions, to

deduce the type corresponding with a word from its meaning representation in a Montague-like system.

The typing system we use is perfectly well adapted to the language of first order predicate logic extented with typed-lambda calculus (corresponding with un-intensional Montague's intensional logic).

Example 4. The semantic translation of the words of our little grammar in this logical language are the following:

- John : $John'$ (the prime symbol distinguishes logical constants from words)
- man : $\lambda x.man'_1(x)$; woman : $\lambda x.woman'_1(x)$
- runs : $\lambda x.run'_1(x)$; walks : $\lambda x.walk'_1(x)$
- a : $\lambda P_1.\lambda Q_1.\exists x[P_1(x) \wedge Q_1(x)]$
- fast : $\lambda P_1.\lambda x.[P_1(x)]$

The types associated with these words are easily deductible from their logical translation. An algorithm working in this case has been implemented.

Nevertheless, the reader should note that we need to impose some syntactic rules (not restrictive in the language of the logic) in order to derive types from logical formulas. For the language evoked, they include the following:

- atomic symbols noted x, y, etc. and isolated logical constants are of type e;
- symbols denoting predicates or functions must be used *in extension*, i.e. displaying their arity and all their arguments: for example, a two-place predicate P will appear as $\lambda x \lambda y.P_2(x)(y)$;
- formulas associated with words must be close, to avoid free variables

In fact, typed languages are usually defined with rules taking into account conditions on types. Types are thus deductible from formulas of these languages if the formulas themselves can be analyzed without ambiguity. The precise conditions allowing type deduction in the general case are being explored.

4 A new learning algorithm from types

We adopt here a *semantic-based theory of syntax learning*, i.e. we consider that the capacity of acquiring a grammar is conditioned by the ability to build a representation of the situation described. In previous semantic-based methods of learning ([HW75,And77,Lan82,Hil83,Fel98]), word meanings are supposed to be already known when the grammatical inference mechanism starts. We make here the smoother hypothesis that the crucial information to be extracted from the environment is the *semantic type of words*. Recognizing that a word represents an entity or a property satisfied by a (or several) entity(ies) is the prerequisite of our learning system. Section 3.2 suggests that types can be deduced from semantic representations, and are thus less informative.

4.1 Layout of the algorithm

The input of our learning algorithm is a sample of couples composed of a syntactically correct sentence and a corresponding sequence of types. The purpose is to build a rigid Categorial Grammar able to generate this sample. The missing information is the *direction* of the functor/argument applications, as each type of the form $\langle u, v\rangle$ can combine with a type u placed either on its left (with a **TB** rule), or on its right (with a **TF** rule). In the first case, the type $\langle u, v\rangle$ syntactically behaves like $u\backslash v$ with the rule **BA**, in the second one like v/u with the rule **FA** in the Categorial Grammar formalism. The identification of the operators / and \ will be processed in two steps: a parsing step and a step to get categories from types.

Parsing types We assume that every sentence given as input is syntactically correct and thus that the associated sequence of types can be reduced using the **TB** and **TF** rules to the type of truth values t. The first step of the algorithm is to find such a reduction.

Example 5.

a	*man*	*runs*
$\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$	$\langle e,t\rangle$	$\langle e,t\rangle$

Only **TF** can apply as a first reduction and combines the first two types into the type $\langle\langle e,t\rangle,t\rangle$. Again using **TF**, the final reduction leads to t. See Figure 1.

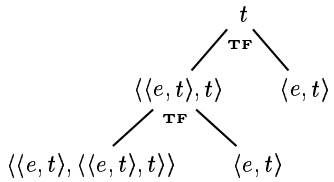


Fig. 1. A parse tree for types.

This reduction can be viewed as a parse in a context free grammar. In this setting, **TF** : $\langle u, v\rangle.u \to v$ and **TB** : $u.\langle u, v\rangle \to v$ are interpreted as schemes of rules where u and v are instantiated by types in the input sequence. As every instantiated rule fitting one of these schemes is in Chomsky Normal Form, it is possible to adapt a standard parsing algorithm to find the parse on types. In this paper, we modify the Cocke-Younger-Kasami (CYK) algorithm.

The whole process is the search for replacing the "," in type expressions to get categories. Possible values for "," are /, \ or "left unchanged" (a "," is left unchanged when the corresponding type is never in a functor place). To this aim, we identify every "," with a distinct variable. A reduction via **TF** or **TB** implies some equalities between categories and subcategories that must be propagated.

For instance, when one applies a **TF** : $\langle u, v\rangle.u \rightarrow v$ rule on *a man* associated with types $\langle\langle e\ \underline{x_1}\ t\rangle\ x_2\ \langle\langle e\ x_3\ t\rangle\ x_4\ t\rangle\rangle$ and $\langle e\ \underline{x_5}\ t\rangle$, the constraint $x_1 = x_5$ arises.

Hence, type reductions translate into variable equalities and a given parse leads to a system of equalities. We depict such equalities in Figure 2 by the underlined variables.

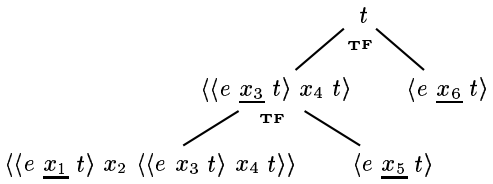


Fig. 2. Parsing types and introducing variable equalities. We underline variables that generate constraints to be fulfilled at each level of the parse tree.

Note that there could be more than one parse for a given input. The outcome of the parser is thus a set of parses each of which being described by a set of variable equalities. We will discuss complexity issues at the end of this section.

Getting categories Now given such contraints on types, we easily deduce categories and hence a Categorial Grammar that accepts the sequence of words as input. To this aim, we consider that **TB** and **TF** have **BA** and **FA** as a counterpart in the categorial grammar formalism. The association between types and categories is done in the following way:

- t is associated with S.
- any distinct type that never occurs at a functor place in the parse is associated with a new variable category.
- If **TF** : $\langle u\ x_i\ v\rangle.u \rightarrow v$ and u (resp. v) is associated with the category A (resp. B) then $x_i = /$ and $\langle u\ x_i\ v\rangle$ is associated with the category B/A.
- If **TB** : $u.\langle u\ x_i\ v\rangle \rightarrow v$ and u (resp. v) is associated with the category A (resp. B) then $x_i = \backslash$ and $\langle u\ x_i\ v\rangle$ is associated with the category $A\backslash B$.

Thus, while getting categories, we add equality constraints of the form $x_i = /$ or $x_i = \backslash$ for some i. The process is always guaranteed to give a set of categories because of the properties of the sequences in input.

Learning process An on-line and incremental learning algorithm can be designed using the strategy describe above. It consists in inferring equality constraints between variables and equality constraints fixing the value of variables from each input couple (in the example below, both are inferred at the same time). We also take into account that the target Categorial Grammar is rigid, that is every word is associated with at most one category.

Example 6. A first input is provided ...

a	man	$runs$
$\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$	$\langle e,t\rangle$	$\langle e,t\rangle$
$\langle\langle e\ x_1\ t\rangle\ x_2\ \langle\langle e\ x_3\ t\rangle\ x_4\ t\rangle\rangle$	$\langle e\ x_5\ t\rangle$	$\langle e\ x_6\ t\rangle$

leads to the system $\begin{array}{l} x_1 = x_5 \\ x_2 = / \\ x_3 = x_6 \\ x_4 = / \end{array}$.

At this point, the Categorial Grammar inferred consists in the associations: $a : (S/B)/A$; $man : A$; $runs : B$.

Consider a second input. Since every word has at most one category we consider the same variables in the type of words already encountered in a previous sentence. The set of variable equalities is enriched.

a	$woman$	$walks$
$\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$	$\langle e,t\rangle$	$\langle e,t\rangle$
$\langle\langle e\ x_1\ t\rangle\ x_2\ \langle\langle e\ x_3\ t\rangle\ x_4\ t\rangle\rangle$	$\langle e\ x_7\ t\rangle$	$\langle e\ x_8\ t\rangle$

$\begin{array}{l} x_1 = x_7 = x_5 \\ x_2 = / \\ x_3 = x_8 = x_6 \\ x_4 = / \end{array}$

Thus, *man* and *woman* (resp. *walks* and *runs*) are of the same category. Consider a third input. One can find that the set of variable equalities becomes:

a	$woman$	$walks$	$fast$
$\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$	$\langle e,t\rangle$	$\langle e,t\rangle$	$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$
$\langle\langle e\ x_1\ t\rangle\ x_2\ \langle\langle e\ x_3\ t\rangle\ x_4\ t\rangle\rangle$	$\langle e\ x_7\ t\rangle$	$\langle e\ x_8\ t\rangle$	$\langle\langle e\ x_9\ t\rangle\ x_{10}\ \langle e\ x_{11}\ t\rangle\rangle$

$$\begin{array}{l} x_1 = x_7 = x_5 \\ x_2 = / \\ x_3 = x_8 = x_6 = x_{11} = x_9 \\ x_4 = / \\ x_{10} = \backslash \end{array}$$

Finally, if the last sentence is *"John walks"*, equality constraints propagate $x_8 = \backslash$ to several categories.

John	*walks*
e	$\langle e,t\rangle$
e	$\langle e\ x_8\ t\rangle$

$\begin{array}{l} x_1 = x_7 = x_5 \\ x_2 = / \\ x_3 = x_8 = x_6 = x_{11} = x_9 = \backslash \\ x_4 = / \\ x_{10} = \backslash \end{array}$.

To summerize, after renaming e in T and $\langle e\ x_1\ t\rangle$ in CN, the Categorial Grammar obtained consists in *John*: T; *man, woman*: CN; *runs, walks*: $T\backslash S$; a: $(S/(T\backslash S))/CN$; *fast*: $(T\backslash S)\backslash(S\backslash T)$. So, with this sample, we exactly obtain the grammar of Example 1. This grammar is able to recognize sentences like "John runs fast", not given as example. Some generalization has thus occurred.

In more complicated cases where more than one parse is possible on a given input, the algorithm must maintain a set of sets of constraints representing a set of candidate grammars.

Criticism A parse in a context free grammar in Chomsky Normal Form can be computed in polynomial time in the size of the input. This complexity result is stated for a given context free grammar. In our setting, we do not have a

context free grammar but a "set of possible ones" defined by schemes of rules. This changes a lot the complexity issues as illustrated by the following example:

Example 7. Let us consider the following input:

a	a	$\ldots$	a	b	a	$\ldots$	a
e	e	$\ldots$	e	$\langle e, \langle \ldots, \langle e, t \rangle \rangle \rangle$	e	$\ldots$	e

with n letters a on both sides of a b. It can be proved that there are $\binom{2n}{n}$ different Categorial Grammars recognizing this input. As a matter of fact, the type associated with b expects $2n$ arguments among which n are on its left and n are on its right. So, **TF** and **TB** must both be applied n times, no matter in which order. Thus, a parser that builds every possible parse runs in exponential time *relatively to the total number of "," in the input sequence of types.*

As an interpretation of this example, two interesting facts can be said. First, we wanted to provide more information as input than simple sentences in order to find linguistically relevant and computationally acceptable learning procedures. The process of finding structural data from sole string input data being too expensive we propose to add semantic information represented by types. It can be observed that even with the richer data of types provided as input, the problem is far from trivial in terms of time and space complexity.

Second, a naive approach consists in trying every possible assignment of a value in $\{"\backslash", "/", "-"\}$ with every "," in the input sequence of types. For n "," in this sequence, there are 3^n different assignments. The worst case complexity of our approach is comparable with the complexity of this naive algorithm.

Nonetheless, the algorithm is not exponential in the number of words in the sentence given as input. The problem arises from the intrinsic number of possible solution grammars. But we are not interested in finding all acceptable Categorial Grammars (w.r.t. the input): only one is enough. Since the parser can work in polynomial time in the case where there is only one acceptable Categorial Grammar, there is still some hope to find relevant adaptations of our method. We will pursue these research directions in the near future.

4.2 Implementation details

We now describe a simplified scheme of our algorithm. The global structure is similar to CYK. We also follow as much as possible the notations in [HU79]. The algorithm builds a table. Rows and columns range from 1 to the length of the input. In the CYK algorithm, a cell (i, j) contains the non-terminals types that the subsequence of length j starting at position i reduces to. Here we add two modifications. First, there might be more than one parse. Therefore cells in the table are indexed by the starting position in the sequence, the length of the subsequence, the number of the parse. Second, we add some information in cells in order to propagate the equality constraints over variables and the partial parse of types called the *partial structure* (that is the parse of a subsequence of types). Hence, cells in the analysis table contains a (non-terminal) type, equality constraints over variables and a partial structure.

The input consists in sequences of words and types. Types are built with variables in place of ",".
The procedure `apply-tf(table[i][k][p],table[i+k][j-k][q],table[i][j])` tries to apply a **TF** rule to a subsequence starting at position i and of length j. The first cell corresponds to the *pth* parse of the subsequence of length k starting at position i and the second cell corresponds to the *qth* parse of the subsequence of length $j-k$ starting at position $i+k$. Basically, `apply-tf` checks that `table[i][k][p].Type` is of the form $\langle u_1\ x_i\ v\rangle$, `table[i+k][j-k][q].Type` is of the form u_2. Then, `apply-tf` searches for a substitution (a variable replacement) σ such that $\sigma(u_1) = \sigma(u_2)$. If it is possible, `apply-tf` adds a new cell `table[i][j][x]` in the list `table[i][j]` in the following way:

- `table[i][j][x].Type` is $\sigma(v)$;
- `table[i][j][x].Var` combines the substitution σ and the substitutions in `table[i+k][j-k][q].Var` and in `table[i][k][p].Var`;
- `table[i][j][x].Stru` is:
  TF(`table[i][k][p].Stru`, `table[i+k][j-k][q].Stru`).

The procedure `apply-tb` is of course similar to `apply-tf`. The main algorithm is given in algorithm 1.

5 Conclusion

Learning a Categorial Grammar means associating categories with words. Categories are built from basic categories and operators. Learning categories from strings of words seems impossible in reasonable time. So, richer input data need to be provided.

The approach developed so far by Buskowsky & Penn and Kanazawa consisted in providing Structural Examples, i.e. giving the nature of the operators / and \ and letting the basic categories to be learnt. Our approach is the exact dual, as it consists in providing the nature of the basic categories (under the form of types), but letting the operators to be learnt.

From a cognitive point of view, our choice seems more relevant, because the data we provide can be interpreted as coming from semantic information. If we admit the existence of a universal symbolic mental language ([Fod75]), types are related with this language and are thus language-independent. On the contrary, the operators used in categories are connected with *the word order* of a specific natural language and this linguistic parameter is not arguably innate.

The ability to identify types, i.e. for example to distinguish an entity from a predicate describing a property satisfied by entities can be compared with what psychologists call *categorization*. This very general ability does not anticipate on the way this semantic distinction will be grammaticallized in a particular natural language. For example, common nouns and intransitive verbs receive the

Algorithm 1 Parse algorithm

Input: $(w_1, \tau_1) \ldots (w_n, \tau_n)$ the sequence of words with the corresponding associated types built with variables $x_1, \ldots, x_p$.

```
 1: for i = 1 to n do
 2:   table[i][1][0].Type= τ_i; // the first partial type in cell ij
 3:   table[i][1][0].Var= ∅; // the first set of equality constraints in cell ij
 4:   table[i][1][0].Stru= w_i; // the first partial structure in cell ij
 5:   table[i][1].size-cell=1; // the number of elements in the cell ij
 6: end for
 7: for j = 2 to n do
 8:   for i = 1 to n − j + 1 do
 9:     table[i][j].size-cell=0;
10:     for k = 1 to j − 1 do
11:       for p = 1 to table[i][k].size-cell do
12:         for q = 1 to table[i+k][j-k].size-cell do
13:           apply-tf(table[i][k][p], table[i+k][j-k][q],table[i][j])
14:           apply-tb(table[i][k][p], table[i+k][j-k][q],table[i][j])
15:         end for
16:       end for
17:     end for
18:   end for
19: end for
```

Output: The global derivation table calculated.

same semantic type corresponding with a one-place predicate. But, in English, intransitive verbs can combine with individual terms (proper names) to provide a sentence whereas common nouns never appear at a functor place. So, both kinds of words are syntactically distinguished by their different combinatorial properties and the initially identical semantic types finally correspond with different syntactic categories.

The algorithm we propose here is able to identify any rigid Categorial Grammar. It still needs to be extended to k-valued Categorial Grammars. In fact, two sources of polymorphism should be distinguished: the case where a word must be associated with different semantic types (for example words like "and"), and the case where a word must be associated with different syntactic categories corresponding with the same semantic type (which must be the case for words like "a"). The first case should be handled as a natural extension of our learning strategy, where each semantically distinct instance of the word is treated as a new word, but the second case should be harder to deal with. Another extension to be explored is the case when only some of the types associated with words are known (for example, those associated with lexical words), whereas some others (those associated with grammatical words) remain unknown.

Finally, our implemantation still needs to be applied on real corpuses to compare its performance with other implemented methods.

References

[Adr92] P. W. Adriaans. *Language Learning from a Categorial Perspective.* PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1992.

[And77] J. R. Anderson. Induction of augmented transition networks. *Cognitive Science*, 1:125–157, 1977.

[BP90] W. Buszkowski and G. Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454, 1990.

[DWP81] D. R. Dowty, R. E. Wall, and S. Peters. *Introduction to Montague Semantics.* Linguistics and Philosophy. Reidel, 1981.

[Fel98] J. A. Feldman. Real language learning. In *ICGI'98, 4th International Colloquium in Grammatical Inference*, pages 114–125, 1998.

[Flo00] Costa Florncio. On the complexity of consistent identification of some classes of structure languages. In *ICGI'2000, 5th International Colloquium on Grammatical Inference*, volume 1891 of *Lecture Notes in Artificial Intelligence*, pages 89–102. Springer Verlag, 2000.

[Fod75] J. Fodor. *The Language of Thought.* Harvester Press, 1975.

[Gol67] E.M. Gold. Language identification in the limit. *Inform. Control*, 10:447–474, 1967.

[HGS60] Y. Bar Hillel, C. Gaifman, and E. Shamir. On categorial and phrase structure grammars. *Bulletin of the Research Council of Israel*, 9F, 1960.

[Hil83] J. C. Hill. A computational model of language acquisition in the two-year-old. *Cognition and Brain Theory*, 3(6):287–317, 1983.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[HW75] H. Hamburger and K. Wexler. A mathematical theory of learning transformational grammar. *Journal of Mathematical Psychology*, 12:137–177, 1975.

[Jan97] T. M. V. Janssen. Compositionality. In J. V. Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 417–473. MIT Press, 1997.

[Kan96] Makoto Kanazawa. Identification in the limit of categorial grammars. *Journal of Logic, Language, and Information*, 5(2):115–155, 1996.

[Kan98] M. Kanazawa. *Learnable Classes of Categorial Grammars.* The European Association for Logic, Language and Information. CLSI Publications, 1998.

[Lan82] P. Langley. Language acquisition through error discovery. *Cognition and Brain Theory*, 5:211–255, 1982.

[MO98] T. Briscoe M. Osborne. Learning stochastic categorial grammars. In *CoNLL97: Computational Natural Language Learning*, pages 80–87, 1998.

[Mon74] R. Montague. *Formal Philosophy; Selected papers of Richard Montague.* Yale University Press, 1974.

[OBW88] Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors. *Categorial Grammars and Natural Language Structures.* D. Reidel Publishing Company, Dordrecht, 1988.

[Tel99] I. Tellier. Towards a semantic-based theory of language learning. In *12th Amsterdam Colloquium*, pages 217–222, 1999.

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[WC80] K. Wexler and P. Culicover. *Formal Principles of Language Acquisition.* MIT Press, 1980.