Complexity of Grammar Induction for Quantum Types

Antonin Delpeuch
École Normale Supérieure
45 rue d'Ulm
75005 Paris, France
antonin.delpeuch@ens.fr

Most categorical models of meaning use a functor from the syntactic category to the semantic category. When semantic information is available, the problem of grammar induction can therefore be defined as finding preimages of the semantic types under this forgetful functor, lifting the information flow from the semantic level to a valid reduction at the syntactic level. We study the complexity of grammar induction, and show that for a variety of type systems, including pivotal and compact closed categories, the grammar induction problem is NP-complete. Our approach could be extended to linguistic type systems such as autonomous or bi-closed categories.

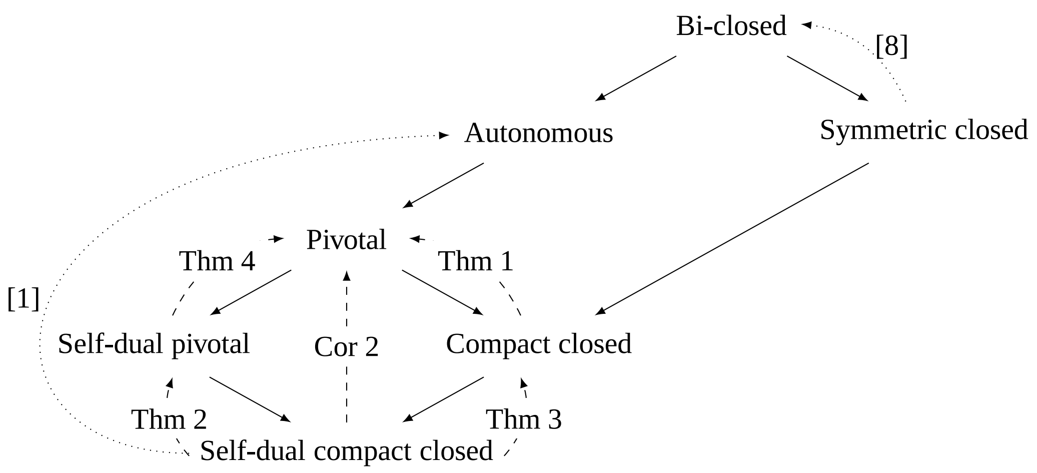
1 Introduction

1.1 Overview

Category theoretic approaches to linguistics are flourishing. They provide a convenient abstract framework for both syntax and semantics [4], and these insights enable some progress on natural language processing tasks [11]. This framework is flexible, because it allows for different types of grammars, such as the Syntactic Calculus of Lambek [16] or Compact Bilinear Logic [19], also known as pregroups [15]. It also allows for different kinds of compositional semantics, which can be distributional [4], Montagovian and extensional [18], Montagovian and intensional [7], or even hybrid models [20]. But whatever the syntax or the semantics are, these approaches rely on a functor from the syntactic category to the semantic category to give meaning to a sentence.

$$\begin{tikzcd} \mathscr{S} \arrow[r, "F"] & \mathscr{C} \\ \text{syntax} & \text{semantics} \end{tikzcd}$$

We propose to study the complexity of lifting the information flow at the semantic level to a valid expression at the syntactic level. In a quantum setting, this could correspond to representing a family of quantum circuits as (planar) string diagrams, for instance. In a linguistic framework, this is the task of grammar induction. Given a set of example sentences belonging to a language, the problem is to infer a grammar of this language. Originally motivated by the study of language acquisition by children [17], this task has been widely investigated in the field of formal languages [6]. If the example sentences are just raw strings, the problem is known to be intractable for most expressive classes of grammars [10]. Hence variations have been introduced, one of them consisting in adding some semantic information about the words in the example sentences. In a categorical framework, words are given syntactic types, which are objects in a monoidal category. The semantic type of a word is the image of this syntactic type under a monoidal functor to the semantic category. The categories we will use are defined in Section 2.2 and are summarised in figure 1. Our results focus on the lower part of our hierarchy of categories, which consists in quantum structures, whereas the linguistic type systems are higher up in the hierarchy.



Plain lines are functors, dashed lines are complexity results and dotted lines are existing algorithms.

Figure 1: A hierarchy of type systems

Since the grammatical correctness of a sentence is witnessed by an arrow from the product of its syntactic types to S (the type of a sentence), the problem of grammar induction can be seen as *lifting* an arrow from the semantic category to the syntactic category, as we will see in Section 4.1.

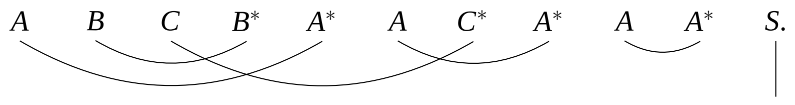
It turns out that many instances of this problem are *hard*, in the sense of computational complexity theory. This is mainly because we require that the syntactic type assigned to each word remains consistent among all the example sentences. This creates global constraints which restrict the solutions of the inference problem. In Section 4, we use this fact to reduce NP-complete problems to our grammar learning problem.

1.2 An example

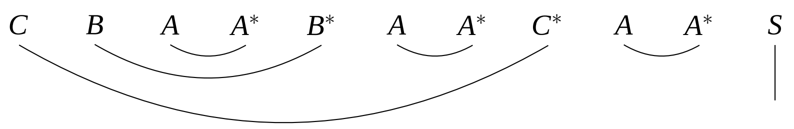
Suppose we use a compact closed category for the semantics and a pivotal category for the syntax. We have to infer the possible syntactic types t_i based on their images $F(t_i)$, where F is the canonical monoidal functor from the free pivotal category to the free compact closed category on a given set of generators. In the following expressions, the tensor product $\otimes$ is implicit.

Syntax	t_1	t_2	t_3	t_4	$\to S$	(1)
Semantics	ABC	B^*A^*A	C^*A^*A	A^*S	$\to S$	(2)

There are many different arrows of the required domain and codomain at the semantic level. One of them is



As the only difference between a free compact closed category and a free pivotal category is the symmetry, the problem bends down to finding a permutation of the basic types of each t_i such that the type reduction holds at the syntactic level. In other words, we have to find a diagrammatic reduction without crossing, such as this one:



In this particular example, one can see that it is necessary that C occurs before B in t_1. We can add a second sentence:

Syntax	t_1	t_5	t_6	t_7	$\to S$	(3)
Semantics	ABC	B^*C^*C	A^*C^*C	C^*S	$\to S$	(4)

This examples forces A to occur before B in t_1. Hence every solution of the learning problem made of these two sentences will be such that C and A occur before B in t_1. In Section 4, this technique enables us to reduce the problem of betweenness [12] to our grammar lifting problem. This problem is known to be NP-complete.

2 A grammar hierarchy

2.1 Monoidal categories as type systems

We define how monoidal categories can be used as type systems. Both the syntactic and the semantic categories will be seen as type systems in our induction problem.

Definition 1. *A **type system** $(\mathscr{C}, S)$ is a strict monoidal category $\mathscr{C}$ with a distinguished object S in $\mathscr{C}$.*

When the object S is clear from the context, the type system is simply noted $\mathscr{C}$. The objects of this category will be used to denote types. We require the category to be monoidal, so that we can define the sentence type as the product of the types of its words. The distinguished object will play the role of the type for a grammatical sentence. The arrows in the category play the role of reductions: A reduces to B when $\mathscr{C}(A,B)$ is not empty.

The type systems we will consider are monoidal categories with some additional structure (which will be detailed in section 2.2), and freely generated by a basic category, whose objects are called **basic types** and morphisms are understood as subtyping relations: there is a morphism between two basic types A and B when A is a subtype of B.

Definition 2. *A **lexicon** l over a set of words W and a type system $(\mathscr{C}, S)$ is a function $l : W \to \mathscr{C}$.*

Although it is interesting to consider the case where multiple types can be assigned to a single word, the previous definition restricts our lexicons to one type per word. We restruct ourselves to rigid grammars, according to the terminology of [1].

Definition 3. *A sequence of words $w_1, \dots, w_n \in W$ is **grammatical** for a lexicon l when $\mathscr{C}(l(w_1) \otimes \cdots \otimes l(w_n), S)$ is not empty.*

In this definition, S is the distinguished type of the underlying type system.

Definition 4. *A **functor** of type systems from $(\mathscr{C}_1, S_1)$ to $(\mathscr{C}_2, S_2)$ is a functor of monoidal categories $F : \mathscr{C}_1 \to \mathscr{C}_2$ such that $F(S_1) = S_2$.*

From this definition, the following property follows immediately:

Proposition 1. *Let $F : \mathscr{T}_1 \to \mathscr{T}_2$ be a functor of type systems. If a sentence $w_1, \dots, w_n$ is grammatical for the lexicon $\mathscr{L}_1$ over $\mathscr{T}_1$, then it is grammatical for the lexicon $F \circ \mathscr{L}_1$ over $\mathscr{T}_2$.*

This property expresses that if a sentence is correct at the syntactic level, then there is a valid reduction at the semantic level.

2.2 Various structures in monoidal categories

We now move on to the definition of the categories involved in the hierarchy of figure 1.

Definition 5. *A **bi-closed category** is a monoidal category in which for all object B, the functor $\_\otimes B$ has a right adjoint $\_/B$ and the functor $B\otimes\_$ has a right adjoint $B\backslash\_$.*

In other words, this means that for every pair of objects A, B, we have morphisms $\text{eval}^l_{A,B} : B\otimes(B\backslash A)\to A$ and $\text{eval}^r_{A,B} : (A/B)\otimes B\to A$ satisfying some coherence equations, and similarly some morphisms $A\to(A\otimes B)/B$ and $A\to B\backslash(B\otimes A)$. Type systems built on bi-closed categories correspond to grammars defined in the Syntactic Calculus of Lambek.

Definition 6. *An **autonomous category**[1] is a monoidal category where for each object A, there are two objects, the left (A^l) and right (A^r) adjoints, equipped with four morphisms $\varepsilon^l_A : A^l\otimes A\to 1$, $\varepsilon^r_A : A\otimes A^r\to 1$, $\eta^l_A : 1\to A\otimes A^l$ and $\eta^r_A : 1\to A^r\otimes A$ satisfying the following equalities :*

$$(\varepsilon^r_A\otimes 1_A)\circ(1_A\otimes\eta^r_A) = 1_A \qquad (\varepsilon^l_A\otimes 1_{A^l})\circ(1_{A^l}\otimes\eta^l_A) = 1_{A^l}$$
$$(1_A\otimes\varepsilon^l_A)\circ(\eta^l_A\otimes 1_A) = 1_A \qquad (1_{A^r}\otimes\varepsilon^r_A)\circ(\eta^r_A\otimes 1_{A^r}) = 1_{A^r}$$

Type systems built on a free autonomous category define pregroup grammars. For instance, let n be the type of a noun phrase and s be the distinguished type of a sentence. If we give the type n to the words *Mary* and *John*, and the type $n^r\otimes s\otimes n^l$ to *loves*, the sentence *Mary loves John* has the type $n\otimes n^r\otimes s\otimes n^l\otimes n$. This type reduces to s through the morphism

$$(\varepsilon^r_n\otimes 1_s\otimes\varepsilon^l_n) : n\otimes n^r\otimes s\otimes n^l\otimes n\to s$$

See [15] for a linguistic presentation of pregroup grammars and [19] for the links with category theory.

The distinction between n^l and n^r is important at a syntactical level to reject ill-formed sentences. For instance, we can give the type $s^r\otimes s$ to adverbs placed at the end of a sentence. If $s^l = s^r$, then the type $s^r\otimes s = s^l\otimes s$ reduces to 1 through ε^l_s, hence the adverb can be written at any place in the sentence, which does not reflect the usual rules of grammar. As one can show that for any object n, $n^{rl}\simeq n\simeq n^{lr}$, the iterated adjoints of a type n are of the form

$$\dots,n^{lll},n^{ll},n^l,n,n^r,n^{rr},n^{rrr}\dots$$

so we can write $n^{ll} = n^{-2}, n^l = n^{-1}, n = n^0, n^r = n^1, n^{rr} = n^2$, and so on.

However, it makes sense to drop the distinction between left and right adjoints at the semantic level: in terms of flow of information, an adjoint is just something that can consume a resource, no matter whether it comes from the left or the right side.

Definition 7. *A **pivotal category** is an autonomous category with a monoidal natural isomorphism between A^r and A^l. We set $A^* = A^l$.*

Pivotal categories correspond to groups, in the sense that in a free pivotal category, two objects have an arrow between them if and only if they are equal in the corresponding free group (where * plays the role of the inverse, hence * will be sometimes noted $^{-1}$).

The canonical morphism between the free pregroup and the free group is defined by

$$h : t_1^{e_1}\otimes\cdots\otimes t_n^{e_n}\mapsto t_1^{(-1)^{e_1}}\otimes\cdots\otimes t_n^{(-1)^{e_n}}$$

where $t_1^{e_1}\otimes\cdots\otimes t_n^{e_n}$ is the canonical form of a pregroup element.

[1]Some authors use the name **compact closed category** instead, but this term has been used for both symmetric and planar categories. As we want to insist on the fact that these categories are not symmetric (contrary to some other ones in this article), we follow the terminology of [22].

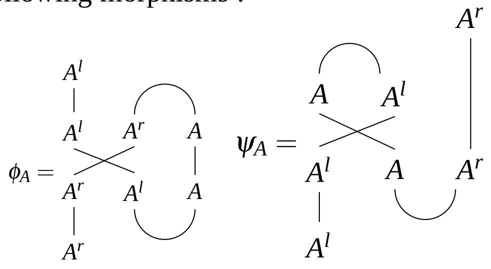
Definition 8. *A* ***compact closed category*** *is an autonomous category which is symmetric, i.e. for each objects A and B there is a monoidal natural isomorphism* $s_{A,B} : A \otimes B \to B \otimes A$ *such that* $s_{A,B}^{-1} = s_{B,A}$.

For instance, the category of finite-dimensional vector spaces is compact closed. One can wonder why we introduced the isomorphism $A^l \simeq A^r$ before adding the symmetries $s_{A,B}$. The following fact explains our choice.

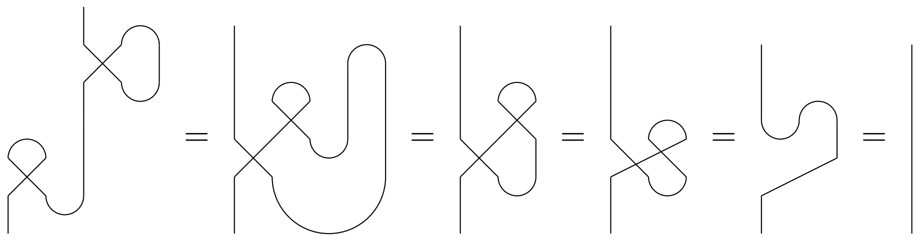
Proposition 2. *Compact closed categories are pivotal.*

This property is well known (it is stated in [4], and implicitly in [22]) but I have never seen a proof of it.

Proof. Let ϕ_A and ψ_A be the following morphisms :



We have $\psi_A \circ \phi_A = 1_{A^l}$ and $\phi_A \circ \psi_A = 1_{A^r}$. By symmetry, let us show the first equality only.



Moreover, one can check with similar techniques that this isomorphism is monoidal and natural. □

Definition 9. *A* ***self-dual compact closed category*** *is a compact closed category with a family of isomorphisms* $h_A : A \to A^*$.

Self-dual compact closed categories have been studied in detail by Selinger in [21]. The definition we adopt here corresponds to his first option, namely self-duality without coherence. As a finite-dimensional vector space is isomorphic to its dual, the category of finite-dimensional vector spaces is self-dual. This category has been widely used as the underlying semantic category for models of meaning, such as in [4], [20] or [7]. The objects in this category have also been used in [1] as semantic types in a learning task. However, they did not introduce a whole typing system at the semantic level, as they had no notion of reduction on semantic types.

We have introduced the commutativity first and then the isomorphism between A and A^*. It is possible to swap these properties, although it requires to be more careful:

Definition 10. *A free* ***self-dual pivotal category*** *is the free pivotal category generated by a category* $\mathscr{C}$ *where for each object* $A \in \mathscr{C}$, $A \simeq A^*$.

A self-dual pivotal category models a rewriting system where any two identical adjacent letters cancel.

It is important to notice that we require that $A \simeq A^*$ only for basic objects. If this were true for all objects, then as noted by Selinger [22], we would get the following isomorphism

$$A \otimes B \simeq (A \otimes B)^* \simeq B^* \otimes A^* \simeq B \otimes A$$

. This isomorphism is not a symmetry in general but would have the same effects on our type system.

A widespread category for semantic types in the linguistic literature is the free symmetric monoidal closed category. It has been used, among others, in [2] and [9].

Definition 11. *A **symmetric closed category** is a symmetric bi-closed category. For all objects A and B, $B\backslash A \simeq A/B$, so we note $A|B = A/B$.*

The objects of this category can be thought of simple types for the simply-typed λ-calculus with pairs. The object $A|B$ plays the role of the type $B \to A$ and we have a morphism $\mathrm{eval}_{A,B} : (A|B) \otimes B \to A$ satisfying the required coherence conditions.

3 Functional types

3.1 Restricting the set of possible types

Not all types are likely to be used in a type-logical grammar. We expect types to be functional, i.e. to be built using only abstractions, the operations $\backslash$ and $/$.

For instance, the type $n \otimes s \otimes n$ belongs to the free pregroup generated by n and s, but cannot be constructed by iterated abstractions. The type $n^r \otimes s \otimes n^l$ however can be constructed as $n\backslash(s/n)$ or $(n\backslash s)/n$.

Definition 12. *Let $\mathscr{L}$ be the free bi-closed monoidal category. The set $P \subset Ob(L)$ is the closure by $/$ and $\backslash$ of the set of basic types. Given a type system $(\mathscr{C}, S)$ and a bi-closed functor $F : \mathscr{L} \to \mathscr{C}$ the set of **functional types** in $\mathscr{C}$ is $F(P)$.*

Restricting our search of types to this form of type reduces our search space. This restriction makes sense because these types are more likely to be relevant from a linguistic point of view. For instance, [15] builds a fairly advanced grammar of English and he uses only functional types in his grammar, while not mentioning this constraint at all.

3.2 Properties of functional types

The generative power of pregroup grammars is not reduced when we require functional types: the proof given in [3] that every ε-free context free grammar is weakly equivalent to a pregroup grammar uses only functional types.

For group grammars (i.e. type systems built on pivotal categories), restricting the assignments to functional types does not harm the expressiveness either, as it is enough to multiply by $a^{-1}a$ the types that are not functional to get an equivalent grammar with functional types only. This remark will be made clear by the following proposition, which characterises functional types in pivotal categories.

Proposition 3. *In a pivotal category, functional types are exactly those which are either*

- *basic types (generators of the free autonomous category), or*
- *products of basic types with exponents $t_1^{e_1} \otimes \cdots \otimes t_n^{e_n}$, where at least one e_i is -1 and at least one e_i is $+1$.*

Proof. By induction on a functional Lambek type t, let us show that $F(t)$ satisfies the characterization above. If $t = a$, a basic type, then $F(t) = a$, falling into the first option. If $t = u/v$, then $F(t) = F(u)F(v)^{-1}$. By induction, there is a basic type occurring with a $+1$ exponent in $F(u)$, so it occurs again with the same exponent in $F(t)$. Similarly, there is a basic type occurring with a $+1$ exponent in $F(v)$, so it occurs with a -1 exponent in $F(t)$.

Conversely, let us show by induction on the length of a group type $t = t_1^{e_1} \otimes \cdots \otimes t_n^{e_n}$ satisfying the characterization that it is the image of a functional Lambek type. If $n = 1$, then $t = a$ where a is a basic type, so $F(a) = t$. If $n > 1$, there are several cases:

- $e_n = -1$ and $e_1, \dots, e_{n-1}$ satisfies the characterization. Then by induction we can find a functional Lambek type u such that $F(u) = t_1^{e_1} \otimes \cdots \otimes t_{n-1}^{e_{n-1}}$ and hence $F(u/t_n) = F(u)F(t_n)^{-1} = t$.
- $e_n = -1$ and $e_1, \dots, e_{n-1} = +1$: then $(-e_2), \dots, (-e_n)$ satisfies the characterization and hence there is a functional u such that $F(u)^{-1} = t_2^{e_2} \otimes \cdots \otimes t_n^{e_n}$, hence $F(t_1/u) = t$.
- if $e_n = +1$ and $(-e_1), \dots, (-e_{n-1})$ satisfies the characterization. Then by induction we can find a functional Lambek type u such that $F(u)^{-1} = t_1^{e_1} \otimes \cdots \otimes t_{n-1}^{e_{n-1}}$ and hence $F(u \backslash t_n) = F(u)^{-1} F(t_n) = t$
- if $e_n = +1$ and $e_1, \dots, e_{n-1} = -1$: then $e_2, \dots, e_n$ satisfies the characterization and hence there is a functional u such that $F(u) = t_2^{e_2} \otimes \dots t_n^{e_n}$, hence $F(t_1 \backslash u) = F(t_1)^{-1} F(u) = t$.

This completes the proof. □

Corollary 1. *In a compact closed category, the characterization of functional types is the same. In a self-dual compact closed category, every type but* 1 *is functional.*

4 Complexity of the grammar induction problem

4.1 Definition of the problem

We study the complexity of learning syntactic types based on positive samples (i.e. a set of grammatical sentences) with semantic types. Each word occurrence in the samples comes with a semantic type. The nature of the syntactic and semantic types depends on the problem.

Definition 13. *A* ***training sample*** *for a type system $(\mathscr{C}, S)$ and a finite set of variables V is a finite set of sentences, where each sentence is a finite sequence of the form $(v_1, t_1), \dots, (v_n, t_n)$, where $v_i \in V$ and $t_i \in \mathscr{C}$ is functional, and such that all the sentences are grammatical for their respective type assignment.*

Note that we do not require that a variable is always paired with a single type. The type of the word can depend on the context in which it appears.

In the following sections, we study the complexity of inducing a grammar, given a finite training sample. First we give a definition of the problem.

Definition 14. *Let $(\mathscr{C}, S)$ be the syntactic type system, $(\mathscr{C}', S')$ be the semantic type system and F be a morphism from $(\mathscr{C}, S)$ to $(\mathscr{C}', S')$. We call* ***grammar induction*** *the problem of, given a training sample T for the type system $(\mathscr{C}', S')$, find a lexicon $h : V \times Ob(\mathscr{C}') \to \mathscr{C}$ such that*

$$\text{for all pair } (v_i, t_i) \in T, F(h(v_i, t_i)) \simeq t_i \text{ and } h(v_i, t_i) \text{ is functional}$$

$$\text{and for all sentence } (v_1, t_1), \dots, (v_n, t_n) \in T, \mathscr{C}(h(v_1, t_1) \otimes \cdots \otimes h(v_n, t_n), S) \text{ is not empty}$$

In other words, the problem is to find functional syntactic types that are compatible with the semantic types and all the sentences are grammatical at the syntactic level. Note that we require that each pair (variable, semantic type) is associated to an unique syntactic type, following [1]. Without this restriction, the problem is trivial as the syntactic types can be chosen independently for each sentence.

4.2 Learning pivotal categories from compact closed categories

Theorem 1. *Type inference from a compact closed category to a pivotal category is NP-complete.*

Proof. We give a reduction of the betweenness problem [12] to our grammar induction problem. The betweenness problem is as follows. Given a finite set A and a set of triples $C \subset A^3$, the problem is to find a total ordering of A such that for each $(a,b,c) \in C$, either $a < b < c$ or $c < b < a$. This problem is NP-complete [12].

The compact closed category we will consider contains the objects a for each $a \in A$ and $d_{a,b,c}$ for each $(a,b,c) \in C$, with the following reduction between basic types: $a \to d_{a,b,c}$ and $c \to d_{a,b,c}$. We set $y = \prod_{x \in A} x$. The preimage of this type will define the total order satisfying the constraints induced by the sentences. For each triple $(a,b,c) \in C$, we define the following compact closed types:

$$w = \prod_{x \in A \setminus \{a,b,c\}} x \qquad c_1 = d_{a,b,c}^{-1} w^{-1} w \qquad c_2 = b^{-1} w^{-1} w$$

and add the following sentence to the training sample:

$$(Y,y)(W_{a,b,c,1}, c_1)(W_{a,b,c,2}, c_2)(W_{a,b,c,3}, c_1)(W_{a,b,c,4}, w^{-1})$$

where the W are words chosen to be different from any word previously seen.

This reduction is polynomial. Let us show that this grammar induction problem has a solution if and only if the corresponding betweenness problem has a solution. If there is a total ordering $<$ of A satisfying the constraints, let $A = \{x_1, \dots, x_n\}$ where $x_1 < \dots < x_n$. One can check that with the following preimages, the sample is grammatical in the pivotal category:

- The type of Y becomes $y' = \prod_{i=1}^{n} x_i$.
- For each $(a,b,c) \in C$, let p,q,r and s be such that

  $$y' = p \cdot a \cdot q \cdot b \cdot r \cdot c \cdot s \qquad \text{or} \qquad y' = p \cdot c \cdot q \cdot b \cdot r \cdot a \cdot s$$

  (where p, q, r and s are possibly equal to 1). y' reduces to $p \cdot d_{a,b,c} \cdot q \cdot b \cdot r \cdot d_{a,b,c} \cdot s$.

  P possible type assignment for the W is:

  $$W_{a,b,c,1} : s^{-1} d_{a,b,c}^{-1} spqr(pqr)^{-1} \quad W_{a,b,c,2} : (rs)^{-1} b^{-1} rs(pq)(pq)^{-1} \quad W_{a,b,c,3} : (qrs)^{-1} d_{a,b,c}^{-1} (qrs) pp^{-1}$$

  One can check that the image of this assignment is equal to the assignment from the training sample and that it makes the sentences grammatical in the pivotal category.

Conversely, if there exists a pivotal type assignment, then as the type b does not occur in the types assigned to $W_{a,b,c,1}$ and $W_{a,b,c,3}$, there is an a or a c on the right side of the occurrence of b, and similarly on the left side. But as there cannot be two occurrences of the same basic type in y', we have either $a < b < c$ or $c < b < a$.

Hence the problem is NP-hard. As one can check a solution in polynomial time, the problem is NP-complete. □

4.3 Learning self-dual pivotal categories from self-dual compact closed categories

Similarly, the previous proof can also be carried when $a \simeq a^{-1}$, giving the following theorem:

Theorem 2. *Type inference from a self-dual compact closed category to a self-dual pivotal category is NP-complete.*

4.4 Learning compact closed categories from self-dual compact closed categories

The problem of grammar induction from a self-dual compact closed category to a compact closed category bends down to assigning exponents to the types. It can be reduced to an integer linear programming problem where we are interested in nonnegative solutions only. This problem is NP-complete and we will show that the grammar induction problem itself is actually NP-complete.

Theorem 3. *Type inference from a self-dual compact closed category to a compact closed category is NP-complete.*

We give a polynomial reduction from 3-SAT to the problem of learning symmetric pivotal types from self-adjoint (symmetric pivotal) types. As 3-SAT is NP-complete [14], and the learning problem is in NP, this will complete the proof. Let $\phi = c_1 \wedge \cdots \wedge c_n$ be a conjunction of 3-clauses. We adopt an approach similar to the strategy of [5]:

(i) We replace each positive occurrence of a variable x_i by a new variable $x_{i,1}$ and each negative occurrence by a different variable $x_{i,0}$. We add to the training sample one sentence, ensuring that all the clauses are satisfied (lemma 1).

(ii) We add one sentence for each pair of variables $x_{i,0}, x_{i,1}$, ensuring that they are assigned opposite truth values (lemma 2).

The full training example will hence encode satisfiability, completing the proof.

Let $c = x_{i_1,b_1} \vee x_{i_2,b_2} \vee x_{i_3,b_3}$ be a 3-clause, where $x_{i,1}$ stands for x_i and $x_{i,0}$ for $\neg x_i$. For each i and b we define a self-adjoint type $v_{i,b} = z_{i,b} y_{i,b} y_{i,b}$. Let $t(c) = d z_{i_1,b_1} z_{i_2,b_2} z_{i_3,b_3}$ be a self-adjoint type. Our idea is that we will force d to have a -1 exponent in the corresponding group type, and hence this group type will be functional if and only if one of the $z_{a,b}$ occurs with a $+1$ exponent. As a clause is true when at least one of the literals it contains is true, this will encode satisfiability.

Lemma 1. *For each literal $x_i^b, i \in \{1, \dots p\}, b \in \{0,1\}$, let $n_{i,b}$ be the number of occurrences of $x_{i,b}$ in ϕ. The assignments making the following sentence grammatical*

$$(S,s)\prod_{i=1}^{n}((C_i,t(c_i))(D_i,d))\prod_{i=1}^{p}\prod_{b=0}^{1}(X_i^b,v_{i,b})^{n_{i,b}}$$

are exactly those for which $x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, \dots, x_{p,0}, x_{p,1}$ are assigned values making all the clauses true.

Proof. Let us show first that if ϕ is satisfiable, then the grammar induction problem has a solution. Let $x_1 = a_1, \dots, x_p = a_p$ be a satisfying boolean assignment. We give X_i^b the group type $z_{i,b}^e y_{i,b}^{-1} y_{i,b}$, where $e = -1$ if $b = a_i$ and $e = 1$ otherwise. This type is functional.

We give C_i the type $c^{-1} z_{i_1,b_{i,1}}^{e_1} z_{i_2,b_{i,2}}^{e_2} z_{i_3,b_{i,3}}^{e_3}$ where $e_k = 1$ if $b_{i,k} = a_{i_k}$ and $e_k = -1$ otherwise. As the clause c_i is satisfied, there is at least one $k \in \{1,2,3\}$ such that $b_{i,k} = a_{i_k}$, hence the type is functional.

Let us show that the sentence is grammatical. As the exponent of $z_{i,b}$ in the type assigned to a clause only depends on a_i, there are $n_{i,b}$ occurrences of $z_{i,b}$, with the same exponent, in $\prod_{j=1}^{n}(c_j, t_j)$. By construction, the exponent of $z_{i,b}$ is inversed in the type assigned to x_i^b, and there are $n_{i,b}$ such occurrences in the sentence. Hence all the $z_{i,b}$ cancel. The type d assigned to D_i cancels with d^{-1} in the type assigned to $t(c_i)$, and the $y_{i,b}$ cancel as well. Hence only s remains: the sentence is grammatical.

Conversely, suppose there are functional group types r_j preimage of $t(c_j)$ and $w_{i,b}$ preimage of $v_{i,b}$ such that the sentences are grammatical at the syntactic level. Note that the types of the words S and D_i are basic types, so the only functional syntactic types compatible with the learning problem are these basic types. As the pregroup type d occurs with exponent $+1$ n times in the product due to the words D_j, all the occurences of d in the r_j have the exponent -1, otherwise they would not cancel. For each r_j, it is functional so one of the $z_{i,b}$ has exponent $+1$. For each x_i^b, $z_{i,b}$ occurs $n_{i,b}$ times with the same exponent, thanks to $w_{i,b}$, and $n_{i,b}$ other times in the clauses, so the exponent assigned to $z_{i,b}$ is the same in every r_j. □

Lemma 2. *The assignments making the following sentence grammatical*

$$(S,s)(X_i^0, v_{i,0})(Z_i, z_{i,0}z_{i,1})(X_i^1, v_{i,1})$$

are exactly those for which the $z_{i,0}$ and $z_{i,1}$ in $v_{i,0}$ and $v_{i,1}$ get opposite exponents.

Proof. One can check that the following assignment is valid, for $e = \pm 1$:

$$s \cdot z_{i,0}^{e} y_{i,0}^{-1} y_{i,0} \cdot z_{i,0}^{-e} z_{i,1}^{e} \cdot z_{i,1}^{-e} y_{i,1}^{-1} y_{i,1} \to s$$

Conversely, as the type assigned to Z_i has to be functional, $z_{i,0}$ and $z_{i,1}$ get opposite exponents in any solution of the grammar inference problem. □

4.5 Learning pivotal categories from self-dual pivotal categories

The construction of Theorem 3 can be adapted to work in non-commutative structures, hence the following theorem:

Theorem 4. *Type inference from self-dual pivotal categories to pivotal categories is NP-complete.*

4.6 Composing complexity results

Suppose we know the complexity of the grammar induction problem between $\mathscr{C}_1$ and $\mathscr{C}_2$, and between $\mathscr{C}_2$ and $\mathscr{C}_3$. What can be said about grammar induction between $\mathscr{C}_1$ and $\mathscr{C}_3$?

$$\begin{tikzcd}
\mathscr{C}_1 \arrow[r] \arrow[rr, bend right, dotted, "?"'] & \mathscr{C}_2 \arrow[r] & \mathscr{C}_3
\end{tikzcd}$$

Given a syntactic category $\mathscr{C}$ and a semantic category $\mathscr{C}'$, we introduce the notion of exact samples.

Definition 15. *A training sample is said* ***exact*** *for some syntactic type t when it contains a word-type pair $(w, F(t))$ such that for all solutions h of this training sample, $h(w, F(t)) = t$.*

We say that a grammar induction problem ***has exact samples*** *when there exists exact samples for each syntactic type t.*

In other words, a grammar induction problem has exact samples when we can build sentences forcing the preimage of a particular type.

Lemma 3. *If the grammar induction problem has exact samples, then for all finite set of syntactic types $T = \{t_1, \dots, t_n\}$ there exists a training sample which is exact for $t_1, \dots, t_n$.*

Proof. Take an exact training samples for each element of T. Make these training samples disjoint by ensuring that they use different words. The concatenation of these training samples satisfies the property claimed. □

Lemma 4. *If grammar induction from $\mathscr{C}_1$ to $\mathscr{C}_2$ and from $\mathscr{C}_2$ to $\mathscr{C}_3$ has exact samples, then so does grammar induction from $\mathscr{C}_1$ to $\mathscr{C}_3$.*

Proof. Take an exact sample S for t from $\mathscr{C}_2$ to $\mathscr{C}_3$. For each type t' occurring in S, take exact samples for t' from $\mathscr{C}_1$ to $\mathscr{C}_2$. Concatenate these samples with the image of S under the functor from $\mathscr{C}_2$ to $\mathscr{C}_1$. ☐

Proposition 4. *If grammar induction from $\mathscr{C}_1$ to $\mathscr{C}_2$ has (polynomial) exact samples and grammar induction from $\mathscr{C}_2$ to $\mathscr{C}_3$ is NP-complete, then grammar induction from $\mathscr{C}_1$ to $\mathscr{C}_3$ is NP-complete.*

Proof. Take an instance of SAT. It can be represented as an equivalent training sample from $\mathscr{C}_2$ to $\mathscr{C}_3$. Take the image of this training sample by the functor from $\mathscr{C}_2$ to $\mathscr{C}_1$ and force this new sample to have the original preimages in $\mathscr{C}_2$ by adding an exact sample. This problem has a solution if and only if the instance of SAT is satisfiable. ☐

$$\begin{tikzcd} \mathscr{C}_1 \arrow[r, "\text{exact samples}"] \arrow[rr, dotted, bend right, "\text{exact samples}"'] & \mathscr{C}_2 \arrow[r, "\text{exact samples}"] & \mathscr{C}_3 \end{tikzcd}$$

$$\begin{tikzcd} \mathscr{C}_1 \arrow[r, "\text{exact samples}"] \arrow[rr, dotted, bend right, "\text{NP-complete}"'] & \mathscr{C}_2 \arrow[r, "\text{NP-complete}"] & \mathscr{C}_3 \end{tikzcd}$$

Lemma 5. *The following grammar induction problems have exact samples:*

- *from self-dual compact closed to self-dual pivotal*
- *from self-dual compact closed to compact closed*
- *from self-dual pivotal to pivotal*
- *from compact closed to pivotal*

Proof. Use the same techniques as the ones we developped for our reductions. ☐

Corollary 2. *Grammar induction from a self-dual compact closed category to a pivotal category is NP-complete.*

Proof. Combine Lemma 5 and Theorem 1 with Proposition 4. ☐

5 Future work

A number of questions remain open. Being able to classify the complexity of the inference problem in the higher half of the hierarchy would enable us to give complexity results on the problems studied in [1] and [8].

Another issue is the expressivity of the classes of grammars defined by the categories in the lower half of the hierarchy. These grammars generate sub-classes of the context-free grammars, but it would be interesting to relate these sub-classes to known classes from the field of formal languages.

One could also use this framework to study inference problems in which the structure of a parse is known, but the types are unknown. This notion of learning with structural examples has been studied for the syntactic calculus [13].

Acknowledgements

This work has been supported by the Quantum Group in the Department of Computer Science of the University of Oxford. I am grateful to Thomas Bourgeat, Amar Hadzihasanovic, Anne Preller and Isabelle Tellier for their help, and the reviewers for their useful comments. Special thanks go to Jamie Vicary who helped me a lot by supervising and reviewing my work.

References

[1] Denis Béchet, Annie Foret & Isabelle Tellier (2007): *Learnability of Pregroup Grammars*. *Studia Logica* 87(2-3), pp. 225–252, doi:10.1007/s11225-007-9085-2.

[2] Wojciech Buszkowski (1984): *A note on the Lambek-van Benthem calculus*. *Bulletin of the Section of Logic* 13(1), pp. 31–35.

[3] Wojciech Buszkowski & Katarzyna Moroz (2008): *Pregroup grammars and context-free grammars*. *Computational Algebraic Approaches to Natural Language, Polimetrica*, pp. 1–21.

[4] Bob Coecke, Edward Grefenstette & Mehrnoosh Sadrzadeh (2013): *Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus*. *Ann. Pure Appl. Logic* 164(11), pp. 1079–1100, doi:10.1016/j.apal.2013.05.009.

[5] Shay B. Cohen & Noah A. Smith (2010): *Viterbi training for PCFGs: Hardness results and competitiveness of uniform initialization*. In: *In Proc. of ACL*.

[6] Colin De La Higuera (2005): *A bibliographical study of grammatical inference*. *Pattern recognition* 38(9), pp. 1332–1348, doi:10.1016/j.patcog.2005.01.003.

[7] Antonin Delpeuch & Anne Preller (2014): *From Natural Language to RDF Graphs with Pregroups*. In: *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, Association for Computational Linguistics, Gothenburg, Sweden, pp. 55–62. Available at `http://www.aclweb.org/anthology/W14-1407`.

[8] Daniela Dudau-Sofronie, Isabelle Tellier & Marc Tommasi (2001): *From Logic to Grammars via Types*. In: *Proceedings of the 3rd Workshop on Learning Language in Logic*, pp. 35–46.

[9] Daniela Dudau-Sofronie, Isabelle Tellier & Marc Tommasi (2003): *A learnable class of Classical Categorial Grammars from typed examples*. In: *8th Conference on Formal Grammar*, pp. 77–88.

[10] E Mark Gold (1967): *Language identification in the limit*. *Information and control* 10(5), pp. 447–474, doi:10.1016/S0019-9958(67)91165-5.

[11] Edward Grefenstette & Mehrnoosh Sadrzadeh (2011): *Experimental Support for a Categorical Compositional Distributional Model of Meaning*. In: *EMNLP*, pp. 1394–1404. Available at `http://www.aclweb.org/anthology/D11-1129`.

[12] Walter Guttmann & Markus Maucher (2006): *Variations on an ordering theme with constraints*. In: *Fourth IFIP International Conference on Theoretical Computer Science-TCS 2006*, Springer, pp. 77–90, doi:10.1007/978-0-387-34735-6_10.

[13] Makoto Kanazawa (1996): *Identification in the limit of categorial grammars*. *Journal of Logic, Language and Information* 5(2), pp. 115–155, doi:10.1007/BF00173697.

[14] Richard M Karp (1972): *Reducibility among combinatorial problems*. Springer, doi:10.1007/978-1-4684-2001-2_9.

[15] J Lambek (2008): *Pregroup Grammars and Chomsky's Earliest Examples*. *Journal of Logic, Language and Information* 17(2), pp. 141–160, doi:10.1007/s10849-007-9053-2.

[16] Joachim Lambek (1968): *Deductive systems and categories*. *Mathematical Systems Theory* 2(4), pp. 287–318, doi:10.1007/bf01703261.

[17] Steven Pinker (2009): *Language Learnability and Language Development, With New Commentary by the Author*. Harvard University Press.

[18] Anne Preller (2005): *Category theoretical semantics for pregroup grammars*. In: *Logical aspects of computational linguistics*, Springer, pp. 238–254, doi:10.1007/11422532_16.

[19] Anne Preller & Joachim Lambek (2007): *Free Compact 2-categories*. *Mathematical. Structures in Comp. Sci.* 17(2), pp. 309–340, doi:10.1017/S0960129506005901.

[20] Anne Preller & Mehrnoosh Sadrzadeh (2011): *Semantic Vector Models and Functional Models for Pregroup Grammars*. *Journal of Logic, Language and Information* 20(4), pp. 419–443, doi:10.1007/s10849-011-9132-2.

[21] Peter Selinger (2010): *Autonomous categories in which $A \simeq A^*$*. In: *Proceedings of the 7th International Workshop on Quantum Physics and Logic (QPL 2010), Oxford*, pp. 151–160.

[22] Peter Selinger (2011): *A survey of graphical languages for monoidal categories*. In: *New Structures for Physics*, *Lecture Notes in Physics* 813, Springer, pp. 289–233, doi:10.1007/978-3-642-12821-9_4.