

# Combinatory Categorical Grammars and Finite Elasticity

Christophe Costa Florêncio

UiL OTS, Utrecht University,  
Trans 10, 3512 JK Utrecht, the Netherlands  
Tel.: +31.30.253.6178, Fax: +31.30.253.6000  
*costa@let.uu.nl*

## 1 Introduction

In (Kanazawa, 1998) the learnability (in the technical sense of identification in the limit, see (Gold, 1967)) of some classes of *combinatory categorical grammars* (see for example (Steedman, 1987), (Szabolcsi, 1987) and most recently (Steedman, 2000)) was investigated. Using the notion of *general combinatory grammar* it was shown that rigid grammars (to be defined in Section 3) are learnable, regardless of the collection of combinators used. However, the relation between collections of combinators and *finite elasticity*, an important notion within Golds paradigm, was left an open problem. This work addresses these issues and gives some sufficient conditions for finite elasticity in the form of restrictions over combinatory rules.

## 2 Finite elasticity

In the literature on computational learning theory various characterizations for identifiability in the limit from positive data have been proposed. One of them is based on the notion of *finite elasticity*:

**Definition 1** *Finite and Infinite Elasticity* (Motoki et al., 1991)

A class  $\mathcal{L}$  of languages is said to have infinite elasticity if there exists an infinite sequence  $\langle s_n \rangle_{n \in \mathbb{N}}$  of sentences and an infinite sequence  $\langle L_n \rangle_{n \in \mathbb{N}}$  of languages in  $\mathcal{L}$  such that for all  $n \in \mathbb{N}$ ,  $s_n \notin L_n$ , and  $\{s_0, \dots, s_n\} \subseteq L_{n+1}$ . A class  $\mathcal{L}$  of languages is said to have finite elasticity if it does not have infinite elasticity.

Finite elasticity is a pleasant property from an ‘engineering’ standpoint, it is closed under union of classes and is a sufficient condition for learnability. See (Moriyama and Sato, 1993) for details.

It is a sufficient condition for *effective* learnability under two conditions, characterized in (Angluin, 1980):<sup>1</sup>

### Theorem 2 (Angluin)

Let  $\langle \Omega, \mathbf{S}, L \rangle$  be a grammar system for which universal membership is decidable, and let  $\mathcal{G}$  be an r.e. subset of  $\Omega$ . Then  $\mathcal{G}$  is learnable if and only if there is a computable partial function  $\psi: \Omega \times \mathbf{N} \mapsto \mathbf{S}$  with the following properties:

1. For all  $n \in \mathbf{N}$ ,  $\psi(G, n)$  is defined if and only if  $G \in \mathcal{G}$  and  $L(G) \neq \emptyset$ .
2. For all  $G \in \mathcal{G}$ ,  $T_G = \{\psi(G, n) \mid n \in \mathbf{N}\}$  is a finite subset of  $L(G)$ , called a tell-tale subset.
3. For all  $G, G' \in \mathcal{G}$ , if  $T_G \subseteq L(G')$ , then  $L(G') \not\subseteq L(G)$ .

Wright (Wright, 1989) used Angluin’s theorem to prove the following:

**Theorem 3** Let  $\langle \Omega, \mathbf{S}, L \rangle$  and  $\mathcal{G}$  be as defined in Theorem 2. If  $L(\mathcal{G})$  has finite elasticity, then  $\mathcal{G}$  is learnable.

In (Kapur, 1991) a blueprint for an algorithm is given that learns classes under these conditions with *consistent* and *conservative* behavior. That is, at all times will the proposed hypothesis be consistent with all the data presented so far, and hypotheses will only be changed when the algorithm is forced to do so, by data it cannot account for with its previous conjecture.

The following theorem is from (Kanazawa, 1998).<sup>2</sup> Let  $\Sigma$  and  $\Upsilon$  be two alphabets, a relation  $R \subseteq \Sigma^* \times \Upsilon^*$  is said to be *finite-valued* just

<sup>1</sup>In fact, in (Angluin, 1980) only the case were  $\Omega = \mathcal{G}$  is considered, but the result easily generalizes to the present setting.

<sup>2</sup>This is in fact a generalization of a theorem from (Wright, 1989), which states that finite elasticity is closed under union.

if for every  $s \in \Sigma^*$ , there are at most finitely many  $u \in \Upsilon^*$  such that  $Rsu$ . If  $M$  is a language over  $\Upsilon$ , define a language  $R^{-1}[M]$  over  $\Sigma$  by  $R^{-1}[M] = \{s \mid \exists u(Rsu \wedge u \in M)\}$ .

**Theorem 4** *Let  $\mathcal{M}$  be a class of languages over  $\Upsilon$  that has finite elasticity, and let  $R \subseteq \Sigma^* \times \Upsilon^*$  be a finite-valued relation. Then  $\mathcal{L} = \{R^{-1}[M] \mid M \in \mathcal{M}\}$  also has finite elasticity.*

### 3 Classical Categorical Grammar

The classes defined in (Kanazawa, 1998) are based on a formalism for ( $\epsilon$ -free) context-free languages called classical categorical grammar (CG)<sup>3</sup>. In this section the relevant concepts of CG will be defined, using notation from (Kanazawa, 1998).

In CG each symbol in the alphabet  $\Sigma$  gets assigned a finite number of *types*. Types are constructed from *primitive types* by the operators  $\backslash$  and  $/$ . We let  $\text{Pr}$  denote the (countably infinite) set of primitive types, the set of types  $\text{Tp}$  is defined as follows:

**Definition 5** *The set of types  $\text{Tp}$  is the smallest set satisfying the following conditions:*

1.  $\text{Pr} \subseteq \text{Tp}$ ,
2. if  $A \in \text{Tp}$  and  $B \in \text{Tp}$ , then  $A \backslash B \in \text{Tp}$ ,
3. if  $A \in \text{Tp}$  and  $B \in \text{Tp}$ , then  $B/A \in \text{Tp}$ .

The intuition behind having two directional slashes is that it allows one to code the syntactic order of (for example) arguments of a verb in a lexicalized grammar: a transitive verb, which takes an NP to the left and an NP to the right and yields a sentence, could be written as  $NP \backslash (S/NP)$ .

One member  $t$  of  $\text{Pr}$  is called the *distinguished type*. In CG there are only two modes of type combination, *backward application*,  $A, A \backslash B \Rightarrow B$ , and *forward application*,  $B/A, A \Rightarrow B$ . In both cases, type  $A$  is an *argument*, the complex type is a *functor*. *Grammars* consist of type assignments to symbols, i.e.  $\text{symbol} \mapsto T$ , where  $\text{symbol} \in \Sigma$ , and  $T \in \text{Tp}$ .

A grammar is said to be  *$K$ -valued* just if it has at most  $k$  types assigned to any symbol in  $\Sigma$ . In the special case  $k = 1$  this property is also known as *rigid*.

<sup>3</sup>This form of categorical grammar is also known as *AB grammar*.

**Definition 6** *A derivation of  $B$  from  $A_1, \dots, A_n$  is a binary branching tree that encodes a proof of  $A_1, \dots, A_n \Rightarrow B$ .*

Through the notion of derivation the association between grammar and language is defined. All structures contained in some given structure language correspond to a derivation of type  $t$  based solely on the type assignments contained in a given grammar. The *string language* associated with  $G$  consists of the strings corresponding to all the structures in its structure language, where the string corresponding to some derivation consists just of the leaves of that derivation.

The class of all categorical grammars is denoted  $\text{CatG}$ , the grammar system under discussion is  $(\text{CatG}, \Sigma^F, \text{FL})$ . The symbol  $\text{FL}$  is an abbreviation of functor-argument language, which is a structure language for CG. Structures are of the form  $\text{symbol}$ ,  $\text{fa}(s1, s2)$  or  $\text{ba}(s1, s2)$ , where  $\text{symbol} \in \text{Pr}$ ,  $\text{fa}$  stands for forward application,  $\text{ba}$  for backward application and  $s1$  and  $s2$  are also structures.

### 4 Combinatory Categorical Grammar

The formalism Combinatory Categorical Grammar (CCG), employs additional combinatory rules in addition to the application rules of CG. Our main motivation for studying CCG is that this formalism is a (linguistically motivated) extension of the (weakly) context-free formalism CG. Its expressive power is strictly greater than that of CG: it was shown in (Vijay-Shanker and Weir, 1990; Vijay-Shanker and Weir, 1994) to be weakly equivalent to the mildly context-sensitive formalism known as linear indexed grammar (LIG). Indexed grammar was introduced in (Aho, 1968), see (Gazdar, 1988; Michaelis and Wartena, 1997; Michaelis and Wartena, 1998) for a discussion of variations on the IG formalism and its relevance for the study of natural language.

In (Kanazawa, 1998), learnability and elasticity of subclasses of  $\text{CatG}$  was studied, it was shown for example that the class  $\mathcal{FL}_{\text{rigid}}$  has finite elasticity. Some subclasses of variations on CG were shown to be learnable, amongst others the rigid subclasses of the so-called  $\mathcal{S}$ -grammars, a notational variant of CCG. However, the question of finite elasticity was left open, we will now address this issue.

#### 4.1 Sufficient conditions over combinatory rules for finite elasticity

The following rules are commonly used in CCG:

Forward Composition:  
 $C/B, B/A \Rightarrow C/A$   
 Backward Composition:  
 $A \setminus B, B \setminus C \Rightarrow A \setminus C$   
 Forward Substitution:  
 $(C/B)/A, B/A \Rightarrow C/A$   
 Backward Substitution:  
 $A \setminus B, A \setminus (B \setminus C) \Rightarrow A \setminus C$

A given grammar  $G$  interpreted under these rules can be compiled into a grammar  $G'$  interpreted under just (a generalized version of) the application rules without affecting the functor language. The naming function for this interpretation will be written as  $FL_{\cdot}$ .

Let the regular slashes be labeled with  $fa$  and  $ba$  (depending on their direction), and let the introduced slashes be indexed with labels for the rules they are compiled for. Combinatory rules are assumed to take the (binary) form of either **Functor, Argument**  $\Rightarrow$  **Result** or **Argument, Functor**  $\Rightarrow$  **Result**<sup>4</sup>, where **Argument** is not a variable type, their corresponding rewrite rules are

$(\mathbf{Argument} | \dots) \mapsto (n | \dots), (\mathbf{Functor} | \dots) \mapsto (\mathbf{Result} /_{label} n | \dots)$ <sup>5</sup> or  
 $(\mathbf{Argument} | \dots) \mapsto (n | \dots), (\mathbf{Functor} | \dots) \mapsto (n \setminus_{label} \mathbf{Result}) | \dots$ , respectively.

The combinatory rules mentioned above result in the following rewrite rules which can be (recursively) applied to the types in a rigid grammar  $G$ , yielding *copies* of these types. Application terminates when none of these rules can be applied anymore. Assigning these copies to the same symbols as their originals were assigned to, combined with the assignments in  $G$ , yields  $G'$ , which can be interpreted with just application.<sup>6</sup>

<sup>4</sup>Conform The Principle of Consistency, see (Steedman, 2000, page 54).

<sup>5</sup>Here  $A/B$  denotes either  $A/B$  or  $B \setminus A$ , and  $\dots$  denotes the rest of the categorial type, which is assumed to be identical on both sides of  $\mapsto$ . Let  $n$  denote some freshly introduced (unique) primitive type.

<sup>6</sup>The operators  $\setminus$  and  $/$  are now shorthand for  $\setminus_{ba}$  and  $/_{fa}$ , respectively.

Forward Composition:  $\langle$   
 $(B/A) | \dots \mapsto n | \dots,$   
 $(C/B) | \dots \mapsto ((C/A) /_{fc} n) | \dots$   
 Backward Composition:  $\langle$   
 $(A \setminus B) | \dots \mapsto n | \dots,$   
 $(B \setminus C) | \dots \mapsto (n \setminus_{bc} (A \setminus C)) | \dots$   
 Forward Substitution:  $\langle$   
 $(B/A) | \dots \mapsto n | \dots,$   
 $((C/B)/A) | \dots \mapsto (C/A) /_{fs} n | \dots$   
 Backward Substitution:  $\langle$   
 $(A \setminus B) | \dots \mapsto n | \dots,$   
 $(A \setminus (B \setminus C)) | \dots \mapsto (n \setminus_{bs} (A \setminus C)) | \dots$

**Definition 7** *The degree of type  $A$  is:*

$\text{degree}(A) = 0$ , if  $A \in Pr$ ,  
 $\text{degree}(A/B) = 1 + \text{degree}(A) + \text{degree}(B)$ ,  
 $\text{degree}(B \setminus A) = 1 + \text{degree}(A) + \text{degree}(B)$ .

Let *application degree* be defined as above, restricted to slashes with index  $fa$  or  $ba$  and other slashes not counted.

**Definition 8** *A conservative combinatory rule (CCR) is a combinatory rule in which the resulting type has a degree that is lower than the degree of the functor type and does not introduce new variable types.*

A conservative tuple of rewrite rules (CTRR) is a tuple of rewrite rules corresponding to a conservative combinatory rule.

Note that our restrictions disallow the two versions of composition.

The first rule in a CTRR rewrites the argument type, its result type is always of lower degree than the argument type. The result type of the second rule in a CTRR has an application degree that is lower than the application degree of the functor type: an extra slash in the result type is never labeled with  $fa$  or  $ba$ , thus this slash does not add to the application degree of this type. We will slightly abuse notation and let  $ctrr[G]$  denote the grammar obtained by applying some CTRRs to grammar  $G$ . Then the following should be obvious:

**Proposition 9** *For any finite grammar  $G$ ,  $G' = ctrr[G]$  is finite.*

**Proof (sketch):** Let  $R$  be the set of CTRRs corresponding to a given set of CCRs. It is clear that all rewrite rules in  $R$  can only be applied a finite number of times to any grammar  $G$ , since all types in  $G$  have a finite degree and these rules reduce the degree of the types they rewrite. Thus only a finite number of new types can be assigned in  $G'$ , the rewritten version of  $G$  interpreted under generalized application.  $\square$

The following defines the relation  $\text{dcom}$ , and, implicitly, structure language for CCG. Note that the application labels are included in the alphabet:

**Definition 10** Let  $\text{dcom}: \Upsilon^F \rightarrow \Sigma^F$  be the homomorphism that maps each indexed application label to a non-indexed application label:

$$\begin{aligned} \text{dcom}(c) &= c, \\ \text{for all } c &\in \Sigma, \\ \text{dcom}(\text{ba}(\text{label}, S_1, S_2)) &= \\ \text{ba}(\text{dcom}(S_1), \text{dcom}(S_2)), & \\ \text{for all } \text{label}, S_1, S_2 \in \Upsilon^F, & \\ \text{dcom}(\text{fa}(\text{label}, S_2, S_1)) &= \\ \text{fa}(\text{dcom}(S_2), \text{dcom}(S_1)), & \\ \text{for all } \text{label}, S_2, S_1 \in \Upsilon^F. & \end{aligned}$$

The following proposition states that, under the proper interpretation,  $\text{ctrr}[G]$  is strongly equivalent to  $G$ :

**Proposition 11** Let  $G$  be a grammar under the set of CCRs  $R$ . Then  $\mathcal{FL}_{\mathcal{R}}(G) = \text{FL}_{\text{}}(\text{ctrr}[G])$ , where  $\text{ctrr}$  denotes application of the CTRRs corresponding to  $R$ .

**Lemma 12** Let  $G$  be a rigid grammar under the set of CCRs  $R$ . Then there is a  $k$ -valued grammar  $G''$  such that  $\text{FL}(G'') = \text{dcom}(\text{FL}_{\text{}}(\text{ctrr}[G]))$ , where  $\text{ctrr}$  denotes application of the CTRRs corresponding to  $R$ .

The value of  $k$  depends on both  $R$  and (the degree of) the types in  $G$  (which is finite but not bounded). Clearly  $\text{dcom}$  defines a finite-valued relation.

**Lemma 13** (Kanazawa, 1998)  $\mathcal{FL}_{k\text{-valued}}$  has finite elasticity.

**Theorem 14** The class  $\mathcal{G}$  of rigid CCGs interpreted under any set of CCRs has finite elasticity.

**Proof (sketch):** Let  $R$  be the set of CTRRs corresponding to a given set of CCRs. By Lemma 12, these rewrite rules constitute a finite-valued relation between  $\mathcal{FL}_{\text{rigid}}$  and  $\mathcal{FL}_{k\text{-valued}}$ , for some constant  $k$ . Using Theorem 4 and Lemma 13, it can be shown that under any set of conservative combinatory rules the class of rigid grammars has finite elasticity.  $\square$

It follows directly that  $\mathcal{FL}_{\mathcal{R}k\text{-valued}}$ ,  $\mathcal{F}_{\mathcal{R}\text{rigid}}$  and  $\mathcal{F}_{\mathcal{R}k\text{-valued}}$ , under CCRs, also have finite elasticity.

Restricting grammar systems to CCRs seems to severely limit their expressive power, it precludes Type-raising, for example. However, Type-raising is used in CCG under the restriction that the resulting range type is a parametrically licensed category for the language. This is assumed to restrict it to a finite set of categories (see (Steedman, 2000), page 44). Thus the rules

$$\begin{aligned} \text{Type-raising: } A &\Rightarrow T/(A \setminus T) \\ A &\Rightarrow (T/A) \setminus T \end{aligned}$$

are actually shorthand for (finite) lists of rules in which  $A$  and  $T$  are replaced by types that only contain primitive types. Adding such lists to a collection of CCRs obviously does not affect finite elasticity.

What is *not* allowed by our restrictions are rules of the form  $C/B, B/A \Rightarrow A/C$ , since type  $A$  is moved from a domain to a range position, so that the degree of the resulting type may be higher than that of the argument or functor type. This type of rule does not seem to be linguistically plausible, so this does not really pose a problem.

## 5 Concluding Remarks

In this paper work from (Kanazawa, 1998), which applies Golds notion of identification in the limit to subclasses of categorial grammars, has been extended to show that the rich formalism of Combinatory Categorial Grammar has natural subclasses that have the pleasant property of finite elasticity. This implies that there exist algorithms that learn these classes while conforming to consistency and conservativity constraints on their behavior prior to convergence. It also implies that union with other

classes with finite elasticity yields classes with the same property.

## References

- Alfred V. Aho. 1968. Indexed grammars. *Journal of the Association for Computing Machinery*, 15:647–671.
- Dana Angluin. 1980. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135.
- Gerald Gazdar. 1988. Applicability of indexed grammars to natural languages. In Uwe Reyle and Christian Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 69–94. Dordrecht: Reidel.
- E. M. Gold. 1967. Language identification in the limit. *Information and Control*, 10:447–474.
- Makoto Kanazawa. 1998. *Learnable Classes of Categorical Grammars*. CSLI Publications, Stanford University.
- Shyam Kapur. 1991. *Computational Learning of Languages*. Available as technical report 91-1234, Department of Computer Science, Cornell University.
- Jens Michaelis and Christian Wartena. 1997. How linguistic constraints on movement conspire to yield languages analyzable with a restricted form of LIGs. In G.J.M. Kruijff, G.V. Morrill, and R.T. Oehrle, editors, *Formal Grammar 1997. Linguistic Aspects of Logical and Computational Perspectives on Language. Proceedings of the Conference*, pages 158–168, Aix-en-Provence, August 9–10.
- Jens Michaelis and Christian Wartena. 1998. Unidirectional inheritance of indices. A weakly context free facet of LIGs. In G. Bouma, G.J.M. Kruijff, and R.T. Oehrle, editors, *Proceedings of the FHCG'98, Joint Conference on Formal Grammar, Head-Driven Phrase Structure Grammar and Categorical Grammar*, pages 258–267, Saarbrücken, August 14–16.
- Takashi Moriyama and Masako Sato. 1993. Properties of language classes with finite elasticity. In Klaus P. Jantke, Shigenobu Kobayashi, Etsuji Tomita, and Takashi Yokomori, editors, *Algorithmic Learning Theory, 4th International Workshop, ALT '93*, volume 744 of *Lecture Notes in Computer Science*, Tokyo, Japan, November 8–10. Springer.
- T. Motoki, T. Shinohara, and K. Wright. 1991. The correct definition of finite elasticity: Corrigendum to identification of unions. In *The Fourth Workshop on Computational Learning Theory*. San Mateo, Calif.: Morgan Kaufmann.
- Mark Steedman. 1987. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5:403,439.
- Mark Steedman. 2000. *The Syntactic Process*. MIT Press/Bradford Books.
- Anna Szabolcsi. 1987. Bound variables in syntax. In *Proceedings of the Sixth Amsterdam Colloquium*, pages 331–351, Institute for Language, Logic and Information. Universiteit van Amsterdam.
- Krishnamurti Vijay-Shanker and D. J. Weir. 1990. Polynomial time parsing of combinatory categorical grammars. In *Proc. of the 28<sup>th</sup> ACL*, pages 1–8, Pittsburgh.
- Krishnamurti Vijay-Shanker and D. J. Weir. 1994. The equivalence of four extensions of context-free grammar. *Mathematical Systems Theory*, 27:511–546.
- Keith Wright. 1989. Identification of unions of languages drawn from an identifiable class. In *The 1989 Workshop on Computational Learning Theory*, pages 328–333. San Mateo, Calif.: Morgan Kaufmann.