Extending Abstract Categorial Grammars with Feature Structures: Theory and Practice

Philippe de Groote, Maxime Guillaume, Agathe Helman, Sylvain Pogodalla, Raphaël Salmon

Extending Abstract Categorial Grammars with Feature Structures: Theory and Practice

Philippe de Groote[1] Maxime Guillaume[1,2] Agathe Helman[2]
Sylvain Pogodalla[1] Raphaël Salmon[2]

[1]Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
[2]Yseop

1 Introduction

Abstract Categorial Grammars (ACGs) are a categorial formalism, developed with the intent of being a kernel for a grammatical framework. They can represent several grammar models [3, 4] such as context-free grammars or tree-adjoining grammars [5].

An ACG is distinguished by its capacity to generate two distinct languages: the abstract language, specifying the underlying derivation structure, and the object language, representing the surface forms of linguistic expressions. They are defined using a small common mathematical foundation: typed linear λ-calculus and high-order signatures. Various paradigms of grammar composition are available in the ACGs, which allows for the construction of complex architectures for natural language modeling. A further key feature is their inherent reversibility, which enables their utilization for both linguistic analysis and generation tasks.

However, modeling morphosyntactic phenomena, such as agreement, results in a substantial enlargement of grammars. Consequently, creating and maintaining wide-coverage grammars become time-consuming. In addition, syntactic analysis becomes less efficient due to the number of grammar rules to consider. To illustrate this point, suppose that we want to define the derivation structures that allow for the construction of the following sentence: *The leverage increases*. Such abstract language may be defined as presented in Figure 1a. However, if we aim to introduce the plural version of this sentence and reject the agrammatical ones, we will need a notion of morphological number in the types, which will double the grammar size as shown in Figure 1b.

Although the size of this toy example is small, it demonstrates the combinatorial explosion that occurs when introducing morphosyntactic constraints. ACGs lack a commonly utilized mechanism in grammatical engineering: feature structures. These structures are advantageous for succinctly describing the morphosyntactic rules of languages, such as agreement. The main goals of this paper are both theoretical and practical. We aim to: (i) introduce a conservative extension to the core

$$\mathrm{S}, \mathrm{N}, \mathrm{NP} :: \mathsf{TYPE}$$
$$\mathit{the} : \mathrm{N} \multimap \mathrm{NP}$$
$$\mathit{leverage} : \mathrm{N}$$
$$\mathit{increases} : \mathrm{NP} \multimap \mathrm{S}$$

(a) Abstract language definition

$$\mathrm{S}, \mathrm{N}_{sg}, \mathrm{N}_{pl}, \mathrm{NP}_{sg}, \mathrm{NP}_{pl} :: \mathsf{TYPE}$$
$$\mathit{the} : \mathrm{N}_{sg} \multimap \mathrm{NP}_{sg}; \mathit{the} : \mathrm{N}_{pl} \multimap \mathrm{NP}_{pl}$$
$$\mathit{leverage} : \mathrm{N}_{sg}; \mathit{leverages} : \mathrm{N}_{pl}$$
$$\mathit{increases} : \mathrm{NP}_{sg} \multimap \mathrm{S}; \mathit{increase} : \mathrm{NP}_{pl} \multimap \mathrm{S}$$

(b) Abstract language definition with morphological variants

Figure 1: Abstract language examples

ACGs that incorporates feature structures; (ii) experimentally demonstrate the benefits of the extension through a rewrite of a wide-coverage grammar; (iii) empirically confirm that the size reduction impacts positively syntactic analysis.

2 Proposed Extension

The questions that arise from the introductory example are how to represent feature structures and how to link them with syntactic categories. In this regard, our proposition involves the representation of feature structures by incorporating well-established constructs, namely records. To annotate syntactic categories with feature structures, we advocate for the adoption of dependent types. Extending ACG with dependent types has already been explored [2, 8]. However, it has been demonstrated that parsing becomes undecidable with fully fledged dependent types. Our primary objective is to propose an extension that does not augment the inherent expressive power of the framework. Particularly for parsing in second-order ACG which can be reduced to the polynomial evaluation of a Datalog program [6, 7].

Our extension, ACG with feature structures (f-ACG), utilizes typed feature structures, where the types of feature structures, denoted as $\mathcal{F}$, can only be either an enumeration or a record type. On the side of type dependency on terms, only feature structure terms F are allowed. This constraint is formulated using an upper level above types, referred to as *kind*, symbolized by $\mathcal{K}$. Furthermore, we extend ACG types written $\mathcal{T}$ with two new constructors: type abstraction and type application. Given our objective to express concisely morphosyntactic combinations using feature variables, it is necessary to add dependent products. However, great care needs to be taken in order to prevent undecidability. These dependent products are restricted to always appear in prenex form through the introduction of a new type level, referred to as *generalized types* ($\mathcal{D}$). The inclusion of dependent products in the kernel leads at the term level to the introduction of feature abstractions, feature applications, and case analysis. Figure 2 succinctly summarizes these principles through the definition of a fragment of f-ACG abstract syntax. Due to space constraints, we are unable to present the full details. However, we present in Figure 3, a formulation of the introductory example written in this extension.

The notion of a f-ACG signature Σ, the basis of the languages, is redefined to take into account the kinding relation and type constants with generalized types. In

$$
\begin{aligned}
\mathcal{F} &::= \{e_1 \mid \cdots \mid e_n\} \mid \left[l_1 : \mathcal{F}, \ldots, l_n : \mathcal{F}\right] \\
F &::= e \mid x \mid \left[l_1 = F, \ldots, l_n = F\right] \mid F.l \\
\mathcal{K} &::= \mathsf{TYPE} \mid \mathcal{F} \to \mathcal{K} \\
\mathcal{D} &::= \mathcal{T} \mid \Pi x : \mathcal{F}.\mathcal{D} \\
\mathcal{T} &::= a \mid \mathcal{T}F \mid \mathcal{T} \multimap \mathcal{T} \mid \Lambda x : \mathcal{F}.\mathcal{T} \\
t &::= c \mid x \mid t\,t \mid t \bullet F \mid \lambda^{\circ} x : \mathcal{T}.t \mid \lambda x : \mathcal{F}.\,t \mid \mathsf{case}\ f\ \{e_1 \to t \mid \cdots \mid e_n \to t\} \\
\Sigma &::= () \mid \Sigma; a :: \mathcal{K} \mid \Sigma; c : \mathcal{D}
\end{aligned}
$$

Figure 2: Abstract syntax of f-ACG components

$$
\begin{aligned}
\mathrm{S} &:: \mathsf{TYPE} \\
\mathrm{N}, \mathrm{NP} &:: \left[number : \{sg, pl\}\right] \to \mathsf{TYPE} \\
\textsc{the} &: \mathrm{N}\left[number = \ \ x\right] \multimap \mathrm{NP}\left[number = \ \ x\right] \\
\textsc{increase} &: \mathrm{NP}\left[number = x\right] \multimap \mathrm{S} \\
\textsc{leverage} &: \mathrm{N}\left[number = \ \ x\right]
\end{aligned}
$$

Figure 3: Abstract language definition with f-ACG

core ACGs, the lexicon $\mathcal{L}$ interprets the abstract language into the object language through homomorphisms. A noteworthy aspect of the f-ACG lexicon lies in the sharing of feature structures between the two languages, which are consequently interpreted as they are. The kinds of atomic types within the abstract signature and their corresponding interpretations in the object signature are equivalent. In addition, the interpretation of dependent products always preserves the feature type parameter. The overall definition of a grammar remains unaltered, except for the distinguished type denoted as S. To allow types with a feature dependency as distinguished types, it is imperative that S represents a proper type in its normal form, devoid of any instances of linear implications.

3 Theoretical Results

This extension does not augment the expressive power of ACG, as it can be established that any f-ACG is transformable into a core ACG. The transformation relies on the finite nature of our definition of feature structures. It involves partial generation and a dependency-less translation of proper types. Formally, the following proposition holds.

Proposition. *Let $\mathcal{G}$ be a f-ACG grammar, there exists a transformation that enables the construction of a core ACG grammar, $\mathcal{G}'$, wherein the abstract and object languages generated by $\mathcal{G}'$ are isomorphic to those generated by $\mathcal{G}$.*

Nonetheless, this transformation exhibits inefficiency when applied to practical real-world applications since factorizations are completely eliminated. One potential alternative approach involves utilizing the underlying context-free barebone and subsequently filtering out inappropriate solutions. However, encoding feature structures directly into the Datalog reduction for second-order ACG is more natural and likely more efficient. To extend the Datalog reduction, we rely on the following principles to integrate feature structures: (i) atomic feature structure terms are encoded as Datalog constants; (ii) records are flattened and eliminated; (iii) feature types are encoded as predicates.

4 Experiments

To measure the performance and factorization gains of the overall extension and reduction, we have developed a f-ACG compiler in Java and conducted a complete rewrite of a proprietary French grammar [9]. This grammar is based on the encoding of a wide-coverage French tree-adjoining grammar [1]. It is constructed from the composition of two ACGs where the level of derivation trees is the pivot. The first ACG defines the correspondence between the derivation and the semantic level, while the other establishes the interpretation of the derivation tree into a derived tree. This grammar is used for text generation from data.

As Figure 4 shows, the grammar of the French language without feature structures consists of 51,246 unanchored trees, whereas the translation in f-ACG comprises 2,792 trees. It corresponds to a reduction of the grammar by eighteen. The parts of the grammar that are the most impacted by the addition of feature structures are the verbal, prepositional and nominal phrases. The category 'other' is also significantly affected. It includes trees for various purposes, such as punctuation and linking words between sentences.

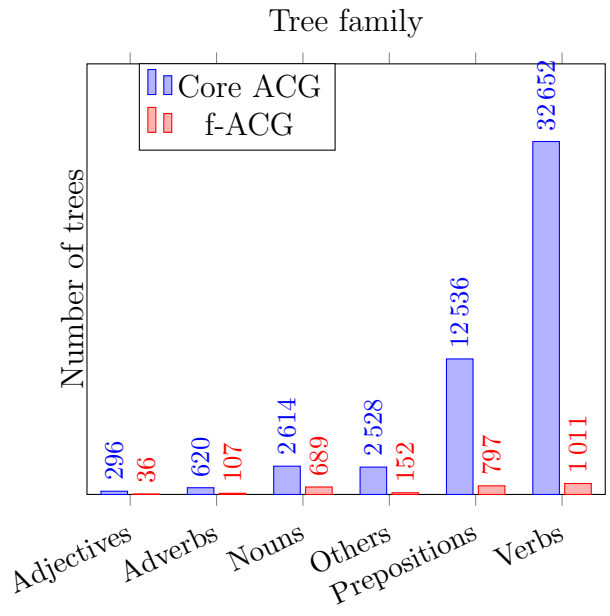

Figure 4: Comparison of the numbers of trees in the two grammars

In terms of performances, the latency of text generation using this grammar heavily depends on the size of the input semantic graph and the number of used lexicalizations. However, we have found that using the compiler mentioned above, when comparing the time spent on generation with a core ACG and its factorized version in f-ACG, the latency is reduced by four, which shows that in addition to an interesting factorization power, the proposed extension also has a positive effect on performances.

In conclusion, the proposed extension offers substantial benefits in terms of grammar size reduction and improved performances while retaining the formal properties of core ACG. However, our definition of feature structures may seem unconventional since certain aspects of feature structures, such as reentrancy and unification, remain unaddressed in this work. These elements offer promising avenues for future development, as they could further increase the factorization power.

References

[1] Anne Abeillé. *Une grammaire électronique du français*. Sciences du langage. CNRS Éditions, 2002.

[2] Ph. de Groote and S. Maarek. Type-theoretic Extensions of Abstract Categorial Grammars. In *New Directions in Type-theoretic Grammars (NDTTG 2007), ESSLLI 2007 workshop*, Dublin, 2007.

[3] Philippe de Groote. Towards Abstract Categorial Grammars. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 148–155, 2001. ACL anthology: P01-1033.

[4] Philippe de Groote and Sylvain Pogodalla. On the expressive power of Abstract Categorial Grammars: Representing context-free formalisms. *Journal of Logic, Language and Information*, 13(4):421–438, 2004.

[5] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto K. Salomaa, editors, *Handbook of formal languages*, volume 3, chapter 2. Springer, 1997.

[6] Makoto Kanazawa. Parsing and generation as datalog queries. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 176–183, Prague, Czech Republic, 6 2007. Association for Computational Linguistics. ACL anthology: P07-1023.

[7] Makoto Kanazawa. Parsing and Generation as Datalog Query Evaluation. *IfCoLog Journal of Logics and their Applications*, 4(4):1103–1211, 2017. Special Issue Dedicated to the Memory of Grigori Mints.

[8] Florent Pompigne. *Modélisation logique de la langue et Grammaires Catégorielles Abstraites*. PhD thesis, Université de Lorraine, 12 2013.

[9] Raphael Salmon. *Natural Language Generation Using Abstract Categorial Grammars*. PhD thesis, Sorbonne Paris Cité, 2017.