A computational grammar for Portuguese

Bruno Cuconato[1] and Alexandre Rademaker[1,2]

[1] FGV/EMAp
[2] IBM Research

Abstract. This work presents an ongoing effort towards a Portuguese grammar under the Grammatical Framework (GF) formalism. GF and the new grammar are briefly introduced, and then we employ the grammar to parse HPSG's Matrix MRS test suite. We will demonstrate the use of the grammar in the parsing of text and in natural language applications.

Keywords: grammatical framework · computational grammar · type theory · functional programming

1 Introduction

Grammatical Framework (GF) is a programming language for grammar writing. It is a functional programming language, with syntax inspired by the Haskell programming language [2]; it draws from intuitionistic type theory for its type system [3].

GF's *forte* lies at multilingual processing. It applies to natural languages the distinction made for programming languages: that of abstract and concrete syntaxes. Separating them allows GF to specify a single abstract grammar for several concrete languages. Translation between two natural languages therefore becomes parsing of concrete syntax to its abstract representation, and then further linearization to the target language.

Writing a grammar for even a fragment of a natural language is a complex task. GF boasts a module system, so GF grammars can import other grammars for code reusing. GF grammars can thus be divided in resource and application grammars: while the former intend to provide useful linguistic constructs for downstream grammars in a suitable and stable application programming interface (API) (like software libraries do to programs [4]), the latter aim to apply these and other definitions to domain-specific applications.

The GF Resource Grammar Library (RGL) declares a common abstract syntax for resource grammars, with a number of grammatical categories, construction functions, and a small test lexicon. Each resource grammar then defines this structure in parallel, and is also free to add language-specific extensions.

Listing 1.1. RGL API, resource grammar, and application grammar output examples

```
> import -retain present/TryEng.gfo
```

```
> cc -one mkS (mkCl (mkNP this_Det (mkN "candy")) (mkA "good"))
this candy is good

> import present/LangEng.gfo
> p -lang=Eng "these fish are rotten"
PhrUtt NoPConj (UttS (UseCl (TTAnt TPres ASimul) PPos
  (PredVP (DetCN (DetQuant this_Quant NumPl) (UseN fish_N))
    (UseComp (CompAP (PositA rotten_A))))) NoVoc

> import FoodsEng.gf FoodsPor.gf
> p -lang=Eng -tr "that pizza is delicious" | l -lang=Por
Pred (That Pizza) Delicious
essa pizza é deliciosa
```

In listing 1.1, we can see (in order) the user importing and using the English resource grammar API to build a simple sentence; the user importing the English resource grammar interface and parsing a sentence with it (notice the detailed output of the syntactic structure); finally, the user imports a domain-specific application grammar, parses a sentence with it, and linearizes the obtained tree in Portuguese. Because the application grammar is specialized to a domain, it can produce smaller and more semantic trees. [3]

2 The Portuguese resource grammar

The current GF RGL supports more than thirty languages, with varying degrees of completeness. This work presents current work on the addition of a Portuguese resource grammar (henceforth PRG) to the RGL.

As an example of the utility of the PRG, a programmer wanting to create a multilingual application grammar involving a Portuguese lexicon would have to hard code the lexicon's inflection tables in the application. With the PRG, she can import the resource grammar, which includes a concrete syntax and a complete set of paradigms for building words. She can then use an overloaded constructor `mkC` (for any given class `C`) which accepts a variable number of arguments dependent on the word's irregularity. For most words, simply providing their uninflected form is sufficient to obtain the correct inflection table [1].

3 Experiments and Discussion

In order to test the PRG, we used HPSG's Matrix MRS test suite of 107 sentences in English. [4] Our experiment was as follows: we parsed the English sentences into trees, removing spuriously ambiguous ones, and linearized the resulting trees into Portuguese. The Portuguese linearizations were then compared to their corresponding sentences in the test suite, and analyzed with respect to

[3] Generally, application grammars also produce less trees than resource grammars.
[4] http://moin.delph-in.net/MatrixMrsTestSuiteEn

grammatical correctness. We do not test the translated sentences for equivalence because translation equivalence is not a goal of the RGL [4].

Even when parsing the simple sentences of the test suite, the issue of ambiguity arises. Consider the sentence [*Some bark*]. Considering the context of the other sentences it is clear that 'bark' here is meant as a verb. But our grammar can not know such a thing, and then outputs three possible trees, one for bark as a noun, and two for bark as a verb.

Another example of ambiguity is in [*the dog could bark*]. The RGL distinguishes between 'can' in the sense of 'know' and in the sense of 'being capable'. These have the same linearization in English, but the Portuguese grammar can then offer two possible translations [*o cachorro sabia ladrar*] and [*o cachorro podia ladrar*].

The test suite allowed us to find several mistakes in our implementation. For instance, the handling of compound nouns is wrong, translating [*the tobacco garden dog barked*] to *[*o tabaco o jardim o cachorro ladrava*].

Besides correcting the mistakes found in the Portuguese linearizations, there are missing constructors that prevented the linearization of some trees, and a few phenomena that are still to be implemented, such as the contraction in *[*havia gatos em o jardim*].

4 Conclusion

When complete, the Portuguese resource grammar will be one of few freely-available computational grammars for Portuguese. In addition to also being open source, GF offers a whole ecosystem of tools for the use of GF grammars in NLP applications: compilation of grammars to several formats (such as a portable binary format and formats for speech recognition grammars), the possibility of embedding grammars in Haskell, Java, Python, and C# programs, and of course, the use of the RGL for multilingual applications.

In the demo, we will give examples of morphology paradigms of GF and their use in Portuguese, as well as offer examples of application grammars using the PRG, such as a logic to natural language translator following [5].

References

1. Détrez, G., Ranta, A.: Smart paradigms and the predictability and complexity of inflectional morphology. In: Proceedings of the 13th Conference of the EACL. pp. 645–653. ACL, Stroudsburg, PA, USA (2012)
2. Marlow, S., et al.: Haskell 2010 language report (2010)
3. Ranta, A.: Grammatical Framework: a type-theoretical grammar formalism. Journal of Functional Programming **14**(2), 145189 (2004)
4. Ranta, A.: The GF resource grammar library. Linguistic Issues in Language Technology **2**(2), 1–63 (2009)
5. Ranta, A.: Translating between language and logic: What is easy and what is difficult. In: CADE. pp. 5–25. Springer (2011)