Can you make a constraint grammar with only weights?

Flammie A Pirinen
UiT Norgga árktalaš universitehta
Institute of Languages and Cultures / Divvun
Tromsø, Norway
flammie.pirinen@iki.fi

Abstract

Traditionally, constraint grammar is originated as a system to deal with ambiguity by means of disallowing and removing readings that are unacceptable. However, to judge a plausible grammatical parse of a word-form as definitely unacceptable given only one context condition is quite risky. What I explore in this experiment is an idea where constraint grammar's judgements are not strict but only give relative ordering or plausibility. My hypotheses are that originally the strict deletion system of constraint grammar was at least partially motivated by limitations of the hardware decades ago, that we no longer have to follow. Furthermore, the construction of parsing system that loops indefinitely on its own input until the output does not change can be avoided when the weighted constraint grammar rules are merely given a single reweighting vote type feedback. In this experiment, I have only implemented a subset of constraint grammar, also forcing the linguistic description to stick with rules that are either directly related to a linguistic evidence—giving a likelihood for a morphological reading because we simultaneously connect it to a syntactic reading—, or a pure disambiguation preference—when forced to select between *a* and *b*, a is more likely unless specifically supported by *c*. I present an experimental usage of the resulting parsing system as a tool for parsing corpus into dependency treebank that is post-edited and verified by human.

1 Introduction

One of the most traditional use cases of constraint grammar is based on having all plausible morphological analyses of word-forms in a sentence as a starting point, and reducing the ones that are grammatically unsuitable or unlikely (Karlsson, 1990). However, writing grammatical rules that express absolute certainties are quite hard to write, and it turns out, that in many mature constraint grammar descriptions average rules span several maybe even dozen of lines, with more and more contexts and exceptions added along the years. This makes the rules sometimes hard to read and maintain. I've been working with such rulesets for several years and my impression is that majority of the extra exceptions that are being added seem like they are often exceptional interactions that get copied to several or even most of the rules in the ruleset, this is one reason why I began experimenting with the idea that maybe encoding such exceptions as new reweighting rules would make sense.

I believe one of the reasons of the idea for constraint grammar to operate on absolute deletion of readings while disambiguating is based on optimisations which were necessary at the time, a computational requirement that is no longer as critical as it was. On the other hand, when experimenting with the weighting constraint grammar logic implementation, I found out I could manage with single pass over the sentence, since readings are not removed and rules do not feed on the weights previously used, it can also be quite fast.

Instead of using the existing constraint grammar system I have performed this experiment with a re-implementation of the rules and grammars in python. I have limited my re-implementation to a few rule-types that are most common and also the ones that I find make sense linguistically; one of my design goals in this experimetn is to write a system that encodes linguistic knowledge into grammars and their parses and clean up some hackier components of our grammars altogether. The main functionality in my grammar system is based on `SELECT`, `IFF`, and `REMOVE` logics, but instead of operating on removal of readings, we operate on

giving the not-removed reading *weight* (penalties[1] The rules I have implemented can have zero or one contexts, if they have zero contexts (or only context is the same cohort), they are local disambiguation preferences; I implemented this first because they are one of the most common types of rules in mature cg rulesets, and most useful in disambiguation tasks; they are linguistically kind of motivated, since it is natural to say things like: "if you have to decide between instructive and genitive, prefer genitive" or "if you have to decide between 'a dog' and 'a clothes moth' prefer dog"; I find also linguistically more pleasant to say than: "instructives (that clash with genitives) **do not exist!**", or "you may **never** speak about 'a clothes moth' (where it clashes with forms of dog)". The other form of rule where we have one context is a kind of linguistic evidence rule; my impression is that most well-formulated rules are based on linguistic evidence, for this reason I have made them work on reweighting *and* dependency drawing principle. This means if you have a rule of disambiguating or selecting, say accusative reading because there's a verb reading in the sentence on the left, it also must mean that, e.g. the accusative must have an `obj` dependency to that verb (or from, depending on your dependency grammar preferences).

Some of the potential use cases for this type of grammars are: machine translation, treebanking, grammar-checking and correction, with human in the loop, etc. We have performed some initial experiments with dependency treebanking: the sentences with all possible readings shown to the post-editor and annotator, they can easily select the highest ranking tree, or remove some readings, or edit the tree by hand with a text editor in CONLL-u format. In future experiments I could foresee using the future revisions of the system in machine translation such that we do not have the problem of early disambiguation hiding the readings that would be useful or just flat out creating a system that can produce n-best translations instead of one. For grammar-checking and correction similarly we currently have specialised grammars that work very carefully to avoid removing information that will be useful further down in the pipeline to provide corrections.

[1] I use the word *weight* throughout the article to mean penalty weight, as is standard in e.g. weighted finite-state algorithms, I know that in other sub-fields of NLP some extra weight can be a positive thing, s.t. heavier is better, but in my systems more weight always means worse.

References

Fred Karlsson. 1990. Constraint grammar as a framework for parsing running text. In *COLING 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics*.