Constraint Grammar as a SAT problem

1 Introduction

We represent Constraint Grammar (CG) (Karlsson et al.1995) as a Boolean satisfiability (SAT) problem. This is attractive from several reasons: formal logic is well-studied, and serves as an abstract language to reason about the properties of CG. Constraint rules encoded in logic capture richer dependencies between the tags than standard CG.

Applying logic to reductionist grammars has been explored earlier by (Lager1998; Lager and Nivre2001), but it was never adopted for use. Since those works, SAT solving techniques have improved significantly, and they are used in domains such as microprocessor design and computational biology—these problems easily match or exceed CG in complexity.

Due to these advances, we were able to revisit the idea and develop it further. Missing from the previous logic-based works, we solve eventual rule conflicts by finding a solution that discards the least number of rules. We test our implementation by parsing texts in the order of 10,000s–100,000s words, using grammars with hundreds of rules.

2 CG as a SAT problem

Let us demonstrate our approach with the following example in Spanish.

```
"<la>"
        "el" det def f sg
        "lo" prn p3 f sg
"<casa>"
        "casa" n f sg
        "casar" v pri p3 sg
        "casar" v imp p2 sg
```

The ambiguous passage can be either a noun phrase, *la*<det> *casa*<n> 'the house' or a verb phrase *la*<prn> *casa*<v><pri><p3> '(he/she) marries her'. We add the following rules:

```
REMOVE prn IF (1 n) ;
REMOVE det IF (1 v) ;
```

Standard CG will apply one[1] of the rules to the word *la*; either the one that comes first, or by some other heuristic. The second rule will not fire, because it would remove the last reading. All readings of *casa* are left untouched by these rules.

The SAT solver performs a search, and starts building possible models that satisfy both constraints. In addition to the given constraints, we have default rules to emulate the CG principles: an analysis is true if no rule affects it, and at least one analysis for each word is true—the notion of "last" is not applicable.

With these constraints, we get two solutions. The interaction of the rules regarding *la* disambiguates the POS of *casa* for free, and ordering of the rules doesn't matter.

```
1) "<la>"
            "el" det def f sg
   "<casa>"
            "casa" n f sg

2) "<la>"
            "lo" prn p3 f sg
   "<casa>"
            "casar" v pri p3 sg
            "casar" v imp p2 sg
```

The most important differences between the traditional and the SAT-based approach are summarised below:

Condition of being unambiguously tagged is irrelevant. Rather than waiting for a word to get disambiguated, the SAT solver starts by making assumptions (e.g. "*casa* is a noun") and working under them, discarding the assumption if it doesn't lead to a model that satisfies all constraints.

[1] To be more cautious, we could require the word at position +1 be disambiguated fully (1C instead of 1), but in that case, neither of the rules would be applied.

Rules are unordered. We have experimented with different strategies:

- Maximise number of rules used—We discarded this strategy, because it was too strong: if a rule leads to a conflict in one case, it was not be applied anywhere.

- Emulate sequential order—Enter clauses produced by each rule one by one, and assume the solver state reached so far is correct; if a conflict is introduced by new clauses, discard them and move on to next.

- ✓ Maximise number of rule applications—If there is a conflict, find the smallest number of rule applications to discard so that the conflict is solved.

The chosen heuristic produced the best results with examples in the order of tens or hundreds of rules. Larger rule sets are yet to be tested.

Rules disambiguate more. Considering our example phrase and rules, standard CG implementation can only remove readings for the word which is tagged as `prn` or `det`. The SAT-based implementation interprets the rules as "determiner and verb together are illegal", and is free to take action that concerns also the word in the condition (`1 n` or `1 v`). This means that the rules are logically flipped: `REMOVE prn IF (1 n)` translates into the same logical formula as `REMOVE n IF (-1 prn)`. A rule with more conditions can be written as many rules, each condition taking its turn to be in position to be removed or selected.

These three features influence the way that rules are written. We predict that less rules are needed; whether this holds in the order of thousands of rules remains to be tested. Getting rid of ordering could ease the task of the grammar writer, since it removes the need to estimate the best sequence of rule applications.

However, the results using our approach should not be too different from the existing ones—our preliminary evaluation against VISL CG-3 shows promising results in this regard.

# words	# rules	SATCG	VISL CG-3
384,155	19	13.7s	4.2s
124,493	99	55.7s	4.8s
384,155	130	1m14.1s	6.1s
384,155	261	2m53.7s	10.7s

Table 1: Execution times.

3 Results and evaluation

Time The worst-case complexity of SAT is exponential, whereas the standard implementations of CG are polynomial, but with advances in SAT solving techniques, the performance in the average case in practice is much more feasible than in the previous works done in 90s–00s. We used the open-source SAT solver MiniSat (Eén and Sörensson2004).

Table 1 shows the execution time compared to VISL CG-3. The number of rules in the grammar is more significant than the word count; this holds also for the performance of VISL CG-3. From the SAT solving side, maximisation is the most costly operation. Parsing Don Quijote (384,155 words) with the rule set of 261 rules, the function was called 147,253 times, and with 19 rules, 132,255 times. The difference in the execution times suggests that there are other reasons for the worse performance—this is to be expected, SATCG is currently just a naive proof-of-concept implementation with no optimisations.

Performance against VISL CG-3 We took a tagged corpus[2] of 21865 words for Spanish, and a small constraint grammar[3], produced independently of the authors. We only took select and remove rules (no substitute/iff), after which the constraint grammar had 261 rules. With this setup, we took the text of the tagged corpus with all ambiguities, and ran both SATCG and VISL CG-3 with the same grammar. Treating the corpus as the gold standard, the disambiguation by SATCG achieves 75.09 % correct result and VISL CG-3 75.80 %. Similar patterns were observed with small ($<$20) rule sets written by the authors; depending on the subset, SATCG and VISL CG-3 had a difference of at most $\pm$ 1.5 %. Introducing rules one by one up to 19, the performance improved in a very similar rate, with less than 0.5 % difference between

[2] https://svn.code.sf.net/p/apertium/svn/branches/apertium-swpost/apertium-en-es/es-tagger-data/es.tagged

[3] https://svn.code.sf.net/p/apertium/svn/languages/apertium-spa/apertium-spa.spa.rlx

the systems at each new rule.

4 Conclusions

SAT-solvers are nowadays powerful enough to be used for dealing with Constraint Grammar. A logic-based approach to CG has possible advantages over more traditional approaches; a SAT-solver may disambiguate more words, and may do so more precisely. Also, the SAT-solver requires less rules, and these rules are simpler. The ordering of rules is something we found was incompatible with a logic-based approach. We compensate this by maximising rule applications. Whether this is an advantage or disadvantage remains to be seen. Our initial experimental results are promising.

References

Niklas Eén and Niklas Sörensson. 2004. An Extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila. 1995. *Constraint Grammar: a language-independent system for parsing unrestricted text*, volume 4. Walter de Gruyter.

Torbjörn Lager and Joakim Nivre. 2001. Part of speech tagging from a logical point of view. In *Logical Aspects of Computational Linguistics, 4th International Conference, LACL 2001, Le Croisic, France, June 27-29, 2001, Proceedings*.

Torbjörn Lager. 1998. Logic for part of speech tagging and shallow parsing. In *Proceedings of the 11th Nordic Conference on Computational Linguistics*.