Constraint Grammar as a SAT problem

Constraint Grammar as a SAT problem

Inari Listenmaa Koen Claessen
Chalmers University of Technology, Gothenburg, Sweden
{inari,koen}@chalmers.se

Abstract

We represent Constraint Grammar (CG) as a Boolean satisfiability (SAT) problem. Encoding CG in logic brings some new features to the grammars. The rules are interpreted in a more declarative way, which makes it possible to abstract away from details such as cautious context and ordering. A rule is allowed to affect its context words, which makes the number of the rules in a grammar potentially smaller. Ordering can be preserved or discarded; in the latter case, we solve eventual rule conflicts by finding a solution that discards the least number of rule applications. We test our implementation by parsing texts in the order of 10,000s–100,000s words, using grammars with hundreds of rules.

1 Introduction and previous research

Constraint Grammar (CG) (Karlsson et al.1995) is a relatively young formalism, born out of practical need for a robust and language-independent method for part-of-speech tagging. In this work, we present CG as a Boolean satisfiability (SAT) problem, and describe an implementation using a SAT solver. This is attractive for 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 (Marques-Silva2010), and they are used in domains such as microprocessor design and computational biology—these problems easily match or exceed CG in complexity. Thanks to these advances, we were able to revisit the idea and develop it further.

Our work is primarily inspired by (Lager1998), which presents constraint rules as a disjunctive logic program, and (Lager and Nivre2001), which reconstructs four different formalisms in first-order logic. Other works combining logic to CG include (Eineborg and Lindberg1998) and (Sfrent2014), both using Inductive Logic Programming to learn CG rules from a tagged corpus.

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 of the rules to the word *la*; either the one that comes first, or by some other heuristic. The other rule will not fire, because it would remove the last reading. If we use the cautious mode (`1C n` or `1C v`), which requires the word in the context to be fully disambiguated, neither of the rules will be applied. In any case, 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 part of speech of *casa* for free, and the order of the rules does not 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 described in the following sections.

2.1 Rules disambiguate more

Considering our example phrase and rules, the standard CG implementation can only remove readings from the target word (`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 (`n` or `v`).

This behaviour is explained by simple properties of logical formulae. When the rules are applied to the text, they are translated into implications: `REMOVE prn IF (1 n)` becomes *casa*<n> $\Rightarrow$ $\neg$*la*<prn>, which reads "if the `n` reading for *casa* is true, then discard the `prn` reading for *la*". Any implication $a \Rightarrow b$ can be represented as a disjunction $\neg a \vee b$; intuitively, either the antecedent is false and the consequent can be anything, or the consequent is true and the antecedent can be anything. Due to this property, our rule translates into the disjunction $\neg$*casa*<n> $\vee$ $\neg$*la*<prn>, which is also equivalent to another implication, *la*<prn> $\Rightarrow$ $\neg$*casa*<n>. 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 corresponds to many rules, each condition taking its turn to be the target.

2.2 Cautious context is irrelevant

Traditional CG applies the rule set iteratively: some rules fire during the first iteration, either because their conditions do not require cautious context, or because some words are unambiguous to start with. This makes some more words unambiguous, and new rules can fire during the second iteration.

In SAT-CG, the notion of cautious context is irrelevant. Instead of removing readings immediately, each rule generates a number of implications, and the SAT solver tries to find a model that will satisfy them.

Let us continue with the earlier example. We can add a word to the input:

la casa grande 'the big house'

and a rule that removes verb reading, if the word is followed by an adjective:

```
REMOVE v IF (1 adj) ;
```

The new rule adds the implication *grande*<adj> $\Rightarrow$ $\neg$*casa*<v>, which will disambiguate *casa* to a noun[1]. As the status of *casa* is resolved, the SAT solver can now discard the model where *casa* is a verb and *la* is a pronoun and we get a unique solution with `det n adj`.

Contrast this with the behaviour of the standard CG. With the new rule, standard CG will also remove the verb reading from *casa*, but it is in no way connected to the choice for *la*. It all depends of the order of the two rules; if the `det` reading of *la* is removed first, then we are stuck with that choice. If we made the first rules cautious, that is, keeping the determiner open until *casa* is disambiguated, then we get the same result as with the SAT solver. Ideally, both ways of grammar writing should yield similar results; traditional CG rules are more imperative, and SAT-CG rules are more declarative.

2.3 Rules can be unordered

As hinted by the previous property, the SAT solver does not need a fixed order of the rules. Applying a rule to a sentence produces a number of clauses, and those clauses are fed into the SAT solver. However, in the unordered scheme, some

[1] Assuming that `adj` is the only reading for *grande*, it must be true, because of the restriction that at least one analysis for each word is true. Then the implication has a true antecedent (*grande*<adj>), thus its consequent ($\neg$*casa*<v>) will hold.

information is lost: the following rule sets would be treated identically, whereas in the traditional CG, only the first would be considered as a bad order.

```
1) SELECT v ;
   REMOVE v IF (-1 det) ;

2) REMOVE v IF (-1 det) ;
   SELECT v ;
```

Without order, both of these rule sets will conflict, if applied to an input that has sequence `det v`. The SAT solver is given clauses that tell to select a verb and remove a verb, and it cannot build a model that satisfies all of those clauses. To solve this problem, we create a variable for every instance of rule application, and request a solution where maximally many of these variables are true. If there is no conflict, then the maximal solution is one where all of these variables are true; that is, all rules take action.

In case of a conflict, the SAT solver makes it possible to discard only minimal amount of rule applications. Continuing with the example, it is not clear which instances would be discarded, but if the rules were part of a larger rule set, and in the context the REMOVE rule was the right one to choose, it is likely that the interaction between the desired rules would make a large set of clauses that fit together, and the SELECT rule would not fit in, hence it would be discarded.

This corresponds loosely to the common design pattern in CGs, where there is a number of rules with the same target, ordered such that more secure rules come first, with a catch-all rule with no condition as the last resort, to be applied if none of the previous has fired. The order-based heuristic in the traditional CG is replaced by a more holistic behaviour: if the rules conflict, discard the one that seems like an outlier.

We can also emulate order with SAT-CG. To do that, we enter clauses produced by each rule one by one, and assume the solver state reached so far is correct. If a new clause introduces a conflict with previous clauses, we discard it and move on to the next rule. By testing against gold standard, we see that this scheme works better with ready-made CGs, which are written with ordering in mind. It also runs slightly faster than the unordered version.

These three features influence the way rules are written. We predict that less rules are needed; whether this holds in the order of thousands of rules remains to be tested. On the one hand, getting rid of ordering and cautious context could ease the task of the grammar writer, since it removes the burden of estimating the best sequence of rules and whether to make them cautious. On the other hand, lack of order can make the rules less transparent, and might not scale up for larger grammars.

3 Evaluation

For evaluation, we measure the performance against the state-of-the-art CG parser VISL CG-3. SAT-CG fares slightly worse for accuracy, and significantly worse for execution time. The results are presented in more detail in the following sections.

3.1 Performance against VISL CG-3

We took a manually tagged corpus[2] containing approximately 22,000 words of Spanish news text, and a small constraint grammar[3], produced independently of the authors. We kept only SELECT and REMOVE rules, which left us 261 rules. With this setup, we produced an ambiguous version of the tagged corpus, and ran both SAT-CG and VISL CG-3 on it. Treating the original corpus as the gold standard, the disambiguation by VISL CG-3 achieves F-score of 82.6 %, ordered SAT-CG 81.5 % and unordered SAT-CG 79.2 %. We did not test with other languages or text genres due to the lack of available gold standard.

We also tested whether SAT-CG outperforms traditional CG with a small rule set. With our best performing and most concise grammar[4] of only 19 rules, both SAT-CG and VISL CG-3 achieve a F-score of around 85 %. This experiment is very small and might be explained by overfitting or mere chance, but it seems to indicate that rules that work well with SAT-CG are also good for traditional CG.

[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

[4] https://github.com/inariksit/cgsat/blob/master/data/spa_smallset.rlx

# rules	SAT-CG$_u$	SAT-CG$_o$	VISL CG-3
19	39.7s	22.1s	4.2s
99	1m34.1s	1m14.9s	6.1s
261	2m54.1s	2m31.6s	10.7s

Table 1: Execution times for 384,155 words.

3.2 Execution 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 more feasible than in the previous works done in 90s–00s. We used the open-source SAT solver MiniSat (Eén and Sörensson2004).

We tested the performance by parsing Don Quijote (384,155 words) with the same Spanish grammars as in the previous experiment. Table 1 shows execution times compared to VISL CG-3; SAT-CG$_u$ is the unordered scheme and SAT-CG$_o$ is the ordered. From the SAT solving side, maximisation is the most costly operation. Emulating order is slightly faster, likely because the maximisation problems are smaller. In any case, SAT does not seem to be the bottleneck: with 261 rules, the maximisation function was called 147,253 times, and with 19 rules, 132,255 times, but the differences in the execution times are much larger, which suggests that there are other reasons for the worse performance. This is to be expected, as SAT-CG is currently just a naive proof-of-concept implementation with no optimisations.

4 Applications and future work

Instead of trying to compete with the state of the art, we plan to use SAT-CG for grammar analysis[5]. There has been work on automatic tuning of hand-written CGs (Bick2013), but to our knowledge no tools to search for inconsistencies or suboptimal design.

The sequential application of traditional CG rules is good for performance and transparency. When a rule takes action, the analyses are removed from the sentence, and the next rules get the modified sentence as input. As a downside, there is no way to know which part comes directly from the raw input and which part from applying previous rules.

A conflict in an ordered scheme can be defined as a set of two or more rules, such that applying the first makes the next rules impossible to apply, regardless of the input. We can reuse the example from Section 2.3:

```
SELECT v ;
REMOVE v IF (-1 det) ;
```

The first rule selects the verb reading everywhere and removes all other readings, leaving no chance for the second rule to take action. If the rules are introduced in a different order, there is no conflict: the REMOVE rule would not remove verb readings from all possible verb analyses, so there is a possibility for the SELECT rule to fire.

Ordered SAT-CG can be used to detect these conflicts without any modifications, as a side effect of its design. After applying each rule, it stores the clauses produced by the rule and commits to them. In case of a conflict, the program detects the particular rule that violates the previous clauses, with the sentence where it is applied. Thus we get feedback which rule fails, and on which particular word(s).

Unordered SAT-CG with maximisation-based conflict solving is not suitable for this task: the whole definition of conflict depends on ordering, and the unordered scheme deliberately loses this information. On a more speculative note, an unordered formalism such as Finite-State Intersection Grammar (Koskenniemi1990) might benefit from the maximisation-based technique in conflict handling.

Finally, we intend to test for conflicts without using a corpus. Let us illustrate the idea with the same two rules, `SELECT v` and `REMOVE v IF (-1 det)` in both orders. Assume we have the tag set {`det, n, v`}, and we want to find if there exists an input such that both rules, applied in the given order, remove something from the input. There are no inputs that satisfy the requirement with the first order, but several that work with the second, such as the following:

```
"<w1>"
        det
        v
"<w2>"
        n
        v
```

Thus we can say that the first rule order is conflicting, but the second one is not. Implementing and testing this on a larger scale is left for future work.

[5]We thank Eckhard Bick for the idea.

5 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, capturing more dependencies between tags. We experimented with both ordered and unordered rules, and found the ordered scheme to work better with previously written grammars. For future direction, we intend to concentrate on grammar analysis, especially finding conflicts in constraint grammars.

Acknowledgments

We thank Eckhard Bick, Tino Didriksen, Francis Tyers and Anssi Yli-Jyrä for comments and suggestions, as well as the anonymous reviewers and everyone who participated in the discussion at the CG workshop.

References

Eckhard Bick. 2013. ML-Tuned Constraint Grammars. In *Proceedings of the 27th Pacific Asia Conference on Language, Information and Computation*.

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.

Martin Eineborg and Nikolaj Lindberg. 1998. Induction of constraint grammar-rules using progol. In David Page, editor, *Inductive Logic Programming*, volume 1446 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.

Kimmo Koskenniemi. 1990. Finite-state parsing and disambiguation. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90. Association for Computational Linguistics.

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*.

João Marques-Silva. 2010. Boolean Satisfiability Solving: Past, Present & Future. Presentation given at the Microsoft Research International Workshop on Tractability, Cambridge, UK, July 5–6.

Andrei Sfrent. 2014. Machine Learning of Rules for Part of Speech Tagging. Master's thesis, Imperial College London, United Kingdom.