Constraint Grammar: From Universality to Regularity and Neural Parsing

Anssi Yli-Jyrä

Department of Modern Languages,
University of Helsinki, Finland
anssi.yli-jyra@helsinki.fi

Abstract

According to the standard view, Constraint Grammar (CG) is a DSL for writing reductionistic surface parsers for natural language. The current work presents an alternative, system theoretic view, according to which CG is a very powerful transition system that is equivalent to a Turing Machine but has interesting nondeterministic restrictions that are equivalent to a Linear Bounded Automaton, a Pushdown Automaton, and a Finite Automaton. This analysis is fruitful because it gives rise to several interesting connections and perspectives. A probabilistic CG is a system where each disambiguation and scanning step is probabilistic decision. Converting that kind of parser to a neural parsing architecture is possible. Most interestingly, CG can be seen as a very general two-way transition system whose attention moves back and forth in the input.

1. Introduction

There has been recently growing interest in the mini-complexity of two-way automata (Kapoutsis, 2012), and the complexity of bidirectional implementations (Hulden, 2011; Yli-Jyrä, 2011) of Constraint Grammar. A problem with a detailed view to CG implementation through automata is that it restricts the techniques too early and does not provide an overview on the full power of CG. The purpose of this article is to demonstrate that viewing CG as a transition system gives a natural and very flexible view to analyze the relationship between CG and automata-theoretic models of computation, Markov decision processes and neural parsing architectures.

2. CG as a Transition System

A transition system is a tuple (C, T, C_0, C_F) consisting of a set of configurations C, transitions $T \subseteq C \times C$ over them, and initial $C_0 \subseteq C$ and a final configurations $C_T \subseteq C$. In a deterministic transition system, T is a function from the previous configuration to the next configuration.

The classical Constraint Grammar (Karlsson, 1990) is a deterministic transition system operating on a sequence $w \in \Pi^*$ of elements of a powerset alphabet $\Pi = 2^\Sigma$ where each element in Σ is a possible label for a word. A word label can be an atomic part-of-speech tag or a complex label such as morphological analysis, morpho-syntactic category, supertag or a lexicalized syntactic category. The powerset alphabet of such label sets would be impractical to process directly and therefore various encodings have been proposed. An element of the powerset alphabet has been encoded either as as a string (Hulden, 2011) or an acyclic automaton (Yli-Jyrä, 2011), but whenever Π is finite these encodings are formally equivalent.

The initial configuration of the transition system is $\diamond w_{\text{lexical}}$ where the sequence $w_{\text{lexical}} \in \Pi^*$ is generated by the lexicon from the input string and the diamond $\diamond$ indicates the location of the write head. The main task of the transition system is to replace the powerset elements (aka cohorts in the CG literature) in w with their smallest nonempty subsets without losing the correct labeling of the whole string. Some transitions may also expand the powerset elements or interleave new powerset elements to the configuration.

Since the turns of the write head are not bounded and the size of the configuration can grow without bounds, it is not too difficult to show that CG can simulate a deterministic Turing machine, being thus computationally universal and equivalent to nondeterministic Turing machines. In the extended version of this paper, we give a formal proof on this by reduction of 2-stack automata to constraint grammars.

3. CG as a Linear-Space-Bounded System

A linear bounded automaton (LBA) is essentially a Turing machine whose space is bounded by a linear function of the input size. An inportant result by Kuroda (1964) states that a language is accepted by an LBA if it is a context sensitive language.

In linguistics, Generative Phonology is another Turing equivalent system for which the complexity of a practical instance is more important than the theoretically proven computational complexity. A similar argument for practical constraint grammars is possible. Although practical CG grammars may have some rules that make the configuration longer, they do this in a very controlled way. Most of the time the number of insertions to the sentence is bounded by a linear function.

The practical observations suggest that CGs, which still retains its iterated definition, are simulated by a linear bounded automaton. We prove this constructively by encoding both the input and the output configurations to the same linear-bounded tape. The converse reduction holds if every linear bounded automaton could be modelled by a linear-bounded Constraint Grammar. To this end, we need to introduce nondeterministic transitions to the CG.

4. CG as a Linear-Time-Bounded System

The third class of automata that could characterize the complexity of CGs under additional constraints is so-called Hennie machine, i.e. single-tape Turing machine whose visits to each tape cell is bounded by a constant. Related to this, Hennie proved that any Turing machine that

works in linear time recognizes a regular language (Hennie, 1965). More generally, any Turing machine running in $O(n \log n)$ time recognizes a regular language (Hartmanis, 1968). However, there are Turing machine instances that take more time but still recognize only a regular language. Therefore, it is undecidable wheter a given Turing machine is a Hennie machine (Průša, 2014), or more generally, recognizes a regular language. Practical CG instances run very efficiently, but the framework lacks formal constraints that would ensure a $O(n \log n)$ time complexity for all practical CG systems.

There is a way to constructively restrict a Turing machine, as well as a CG parser, to halt in linear time. For example, we can add a finite counter to each position in the configuration to tell the number of remaining visits in that location (Průša, 2014). Under such an additional processing constraint that is turned into a hard constraint of a system, we obtain the result that the classical CG (Karlsson, 1990) is regular and thus a very compactly represented finite-state grammar although its architecture involves context checks, two-way rewriting and iteration. However, since setting an arbitrary limit for the visit is formally awkward, we conjecture that a similar constraint can be stated probabilistically: when the size of the input grows, the number of visits per each position tend to a linear function whose constants depend on each grammar instance.

5. Extensions to Classical Constraint Grammar

Formal analysis of the CG framework using the transition systems and restrictions of a Turing machine opens interesting possibilities by providing means for analysing and presenting various variants of the framework in an understandable way.

5.1 Nondeterminism and Probabilities

New implementations of CG became possible when the system is understood as a transition system. Classical constraint grammar is a deterministic transition system. However, it is possible to introduce nondeterminism and probabilistic transitions to the system.

5.2 Neural CG

It is possible to use continuous space representations to turn a transition system into a neural transition models. The required neural architecture is in many respects nonconventional: (i) input and output layers must handle ambiguity classes by context embeddings, (ii) one location is changed at a time, (iii) the refinements of context embeddings must be feeded back to the network, (iv) there can be an unspecified number of iterations, (v) contexts must be evaluated after each change and (vi) beam search must be generalized to two-way search.

5.3 Full Parsing

If the label alphabet of the words consists of syntactic categories rather than morphological tags, CG generalizes to a transition-based syntactic parser. More generally, it is conjectured that any current transition-based parser architecture could be encoded as a Constraint Grammar whose complexity depends on the complexity of the encoded transition system. In particular, any linear transition-based parser is a special case of a two-way transition-based parser such as Constraint Grammar, although this is far from obvious if we view CG as a specific domain specific language that disguises the generality and elegancy of the underlying transition and automaton architecture.

References

Jik H. Chang, Oscar H. Ibarra, Michael A. Palis, and Bala Ravikumar. 1986. On pebble automata. *Theoretical Computer Science*, 44:111–121.

J. Hartmanis. 1968. Computational complexity of one-tape Turing machine computations. *J. ACM*, 15(2):325–339.

F.C. Hennie. 1965. One-tape, off-line Turing machine computations. *Information and Control*, 8(6):553–578.

Mans Hulden. 2011. Constraint grammar parsing with left and right sequential finite transducers. In *Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing*, pages 39–47.

Christos A. Kapoutsis. 2012. Minicomplexity. In Martin Kutrib, Nelma Moreira, and Rogério Reis, editors, *Descriptional Complexity of Formal Systems: 14th International Workshop, DCFS 2012, Braga, Portugal, July 23-25, 2012. Proceedings*, pages 20–42, Berlin, Heidelberg. Springer Berlin Heidelberg.

Fred Karlsson. 1990. Constraint grammar as a framework for parsing running text. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 3*, COLING '90, pages 168–173.

S.-Y. Kuroda. 1964. Classes of languages and linear-bounded automata. *Information and Control*, 7:207–223.

D. Průša. 2014. Weight-reducing Hennie machines and their descriptional complexity. In *LATA 2015*, number 8370 in LNCS, pages 553–564. Springer.

Anssi Yli-Jyrä. 2011. An efficient constraint grammar parser based on inward deterministic automata. In *Proceedings of the NODALIDA 2011 Workshop Constraint Grammar Applications*. NEALT Proceedings Series.