Bounded Graph Rewriting for Natural Language Processing

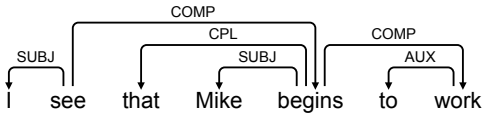
Bruno Guillaume
LORIA – Inria Nancy Grand-Est

So far, a large amount of works in Natural Language Processing (NLP) rely on trees as the core mathematical structure to represent linguistic information (e.g. in Chomsky's work). However, some linguistic phenomena do not cope properly with trees. In some former papers, we showed the benefit of encoding linguistic structures by graphs and of using graph rewriting rules to compute on those structures. Justified by linguistic considerations, the needed graph rewriting setting is characterized by two features: first, there is no node creation along computations and second, there are non-local edge modifications. In this paper, we explain linguistic motivations, describe briefly the Graph Rewriting framework considered and some of its formal properties.

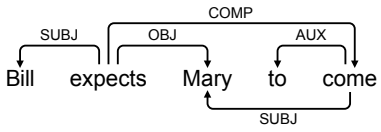
1 Introduction

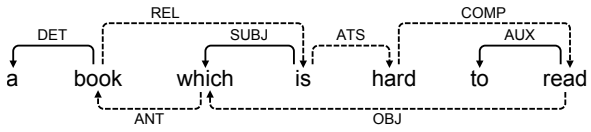
Linguists introduce different levels to describe a sentence in natural language. Starting from a sentence given as a sequence of sounds or as a sequence of words; among the linguistic levels, two are deeply considered in literature: the syntactic level (a grammatical analysis of the sentence) and the semantic level (a representation of the meaning of the sentence). These two representations involve mathematical structures such as logical formulae, λ-terms, trees or graphs.

One of the usual ways to describe syntax is to use the notion of dependency [15]. A dependency structure is an ordered sequence of words, together with some relations between these words. For instance, the sentence *"I see that Mike begins to work"* can be represented by the structure below.



There is a large debate in the literature about the mathematical nature of the structures needed for natural language syntax: do we have to consider trees or graphs? Trees are often considered for their simplicity; however, it is clearly insufficient. Let us illustrate limitations of tree-representations through two linguistic examples: *"Bill expects Mary to come"* and *"a book which is hard to read"*. In most of the annotated corpora, dependency strucures are trees. Examples of such trees are given below if we consider only the relations drawn above the words.



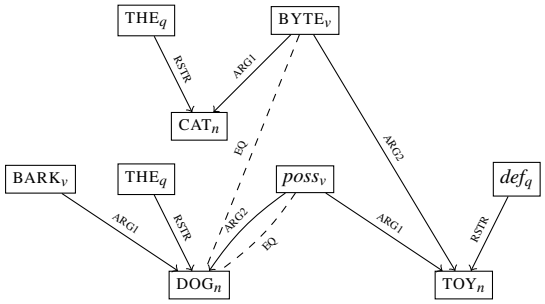


But, if we want to interpret the sentences, we have to know on the first example that *"Mary"* is the logical subject of *"come"* and, on the second, that the logical object of the verb *"read"* is *"book"* (this is

To appear in EPTCS.

expressed by two relations because of the relative construction). These new dependency relations (drawn below the words in figures) turn our structures into a DAG in the first case and even a graph with cycle in the second case (edges in the cycle are drawn with dashed line).

For the semantic representation of natural language, first order logic formulae are widely used. To deal with natural language ambiguity, a more compact representation of a set of logic formulae (called underspecified semantic representation) may be used. Dependency Minimal Recursion Semantics (DMRS) [5] is an example of this compact representation. For instance, the DMRS structuregt of the sentence *"The dog whose toy the cat bit barked"* is given below.



To describe transformations between syntactic and semantics structures, there are solutions based on many computational models (finite state automata, λ-calculus). It is somewhat surprising that Graph Rewrite (GR) have been hardly considered so far ([10, 1, 6, 11]). To explain that, GR implementations are usually considered to be too inefficient to justify their extra-generality. For instance, pattern matching does not take linear time where this is usually seen as an upper limit for fast treatment.

However, if one drops for a while the issue of efficiency, the use of GR is promising. Indeed, linguistic considerations can be most of the time expressed by some relations between a few words. Thus, they can be easily translated into rules. To illustrate this point, we proposed several applications of Graph Rewriting to different kinds of NLP tasks: building semantics out of syntax [2]; building deep syntactic structures [4] or even parsing [9].

In these studies, we tried to delineate what are the key features of Graph Rewriting in the context of NLP. Roughly speaking, node creations are strictly restricted, edges may be shifted from one node to another and there is a need for negative patterns. Based on this analysis, we define here a suitable framework for NLP (see Section 3).

Compared to term rewriting, the semantics of graph rewriting is problematic: different choices can be made in the way the context is glued to the rule application [14]. As far as we see, our notion does not fit properly the Double-Pushout approach due to unguarded node deletion nor the Sngle-Pushout approach due to the shift command, as we shall see. Thus we will provide a complete description of our notion. We have chosen to present it in an operational way.

In practice, we need to verify termination properties: in our NLP applications, any computation should terminate. If it is not the case, it means that the rules where not correctly drawn.

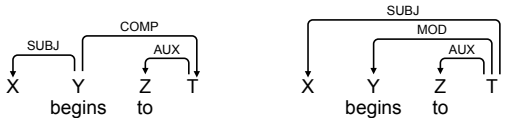
2 Linguistic motivations

Without any linguistic exhaustiveness, we highlight in this section some crucial points of the kind of linguistic transformations we are interested in and the relative features of rewriting we have to consider.

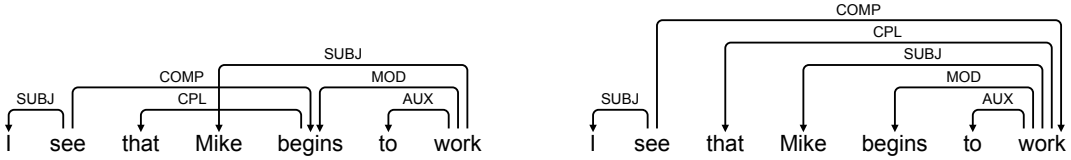
Node preservation property. As linguistic examples above suggest, the goal of linguistic analysis considered is mainly to describe different kinds of relations between elements that are present in the input structure. As a consequence, the set of nodes in the output structure is directly predictable from the

input. In practice, these node creations can be anticipated in some enriched input structure on which the whole transformation can be described as a non-size increasing process.

Edge shifting. In the first example of the introduction (for the sentence *"I see that Mike begins to work"*), the verb *"begins"* is called a raising verb and we know that *"Mike"* is the *deep subject* of the verb *"work"*; *"begins"* being considered as a modifier of the verb. To recover this deep subject, one may imagine a local transformation of the graph which turns the first pattern graph (below on the left) into the second one (below on the right).

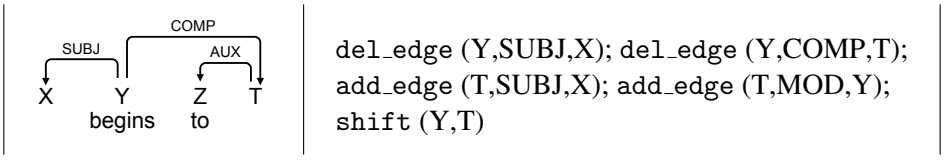


However, in our example above, a direct application of such a transformation leads to the structure below on the left which is not the wanted structure. Indeed, the transformation should shift what the linguists call the head of the phrase *"Mike begins to work"* from the word *"begins"* to the word *"work"* with all relative edges. In that case, the transformation should produce the structure below on the right:

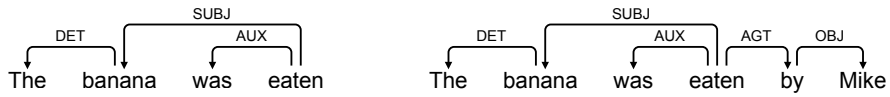


In other words, our transformation may have to specify the fact that all incident edges of some node (here Y) must be transported to some other node (here T). We call this operation `shift`.

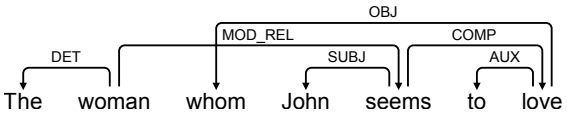
To describe our graph rewriting rules, we introduce a system of commands (like in [7]) which expresses step by step the modifications applied on the input graph. The transformation becomes:



Negative conditions. In some situations, rules must be aware of the context of the pattern to avoid unwanted ambiguities. For instance, when computing semantics out of syntax, one has to deal with passive sentences; the two sentences below show that the agent *"by Mike"* is optional.

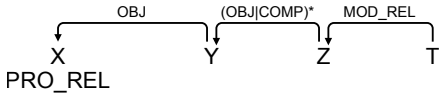


In order to switch to the corresponding active form, two different linguistic transformations have to be defined for these two sentences; but, clearly, the first graph is a subgraph of the second one. The transformation defined for the short passive should not be applied to the long passive on the right. Thus, we need to express a negative condition like "there is no out edge labelled by AGT out of the main verb" to prevent the unwanted transformation to occur.

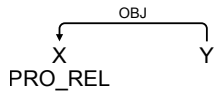


Long distance dependencies. Most of the linguistic transformations can be expressed with successive local transformations like the one above. Nevertheless, there are some cases where more global rewriting is required; consider the sentence *"The women whom John seems to love"*, for which we consider the syntactic structure on the right. One of the steps in the semantic construction of this sentence requires to compute the antecedent of the relative pronoun *"whom"* (*"woman"* in our example).

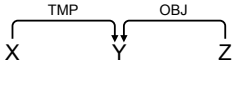
The subgraph we have to search in our graph (which is depicted as a non-local pattern) and the graph modification to perform are given on the right. The number of OBJ or COMP relations to consider (in the relation depicted as (OBJ|COMP)* in the figure) is unbounded (in linguistics, this phenomenon is called long distance dependencies); it is possible to construct grammatical sentences with an arbitrary large number of relations. As we want to stay in the well-known framework of local rewriting, we use several local transformations to implement such a non-local rule.



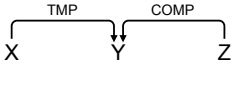
`add_edge` (X,ANT,T)



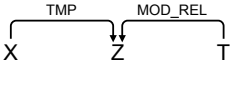
`add_edge` (X,TMP,Y)



`del_edge` (X,TMP,Y)
`add_edge` (X,TMP,Z)



`del_edge` (X,TMP,Y)
`add_edge` (X,TMP,Z)



`del_edge` (X,TMP,Z)
`add_edge` (X,ANT,T)

The second and the third rules above preserve the set of nodes and the number of edges of each kind. Hence, this kind of rule will require special treatment with respect to termination issues.

3 Graph Rewriting for NLP

In this section, we recall the main results presented in [3] and we let the reader refer to this publication for technical details.

Graphs The graphs we consider are directed graphs with labels on nodes and labels on edges (both taken from finite sets). We restrict the edge set: given some edge label e, there is at most one edge labelled e between two given nodes α and β. This restriction reflects the fact that, in NLP applications, our edges are used to encode linguistic information which are relations. We make no other explicit hypothesis on graphs: in particular, graphs may be disconnected, or have loops.

Graph Morphism Given two graphs P and G, a function μ mapping nodes of P into nodes of G is called a *graph morphism* if it respects the labelling and if each edge in the input graph P exists in the image $\mu(P)$ in G. Moreover, we will consider only injective graph matching where two different nodes of P cannot be mapped by μ to the same node. We write such an injective graph morphism $\mu : P \hookrightarrow G$.

As said above, we need to express negative conditions which prevent rule application. Then a rule left-hand-side is described by a graph P and k graphs $N_1, \ldots, N_k$ for negative conditions such that for all $1 \leq i \leq k$, there is an injective morphism $\eta_i : P \hookrightarrow N_i$. Now, the rule apply to a graph G if P is found in the graph but this matching cannot be extended, for any i, to a mapping of N_i into G. Formally, there is some $\mu : P \hookrightarrow G$ and for all $1 \leq i \leq k$, there is no morphism η'_i such that $\eta_i \circ \eta'_i = \mu$.

Commands The right hand side of a rule is defined by a list of atomic commands which describe how the matched part of the graph must be modified by the rule application. Atomic commands are:

`label`(change node label), `add_edge`, `del_edge`, `del_node` and `shift`. The `shift` is parametrized by two pattern nodes X and Y. Its effect on the graph is to modify each edge incident to X and make it incident to Y.

Graph Rewrite System A Graph Rewrite System (GRS) is composed of a set of rules. A graph G is called a normal form if no rules of the GRS can be applied to it. When a GRS is applied to a graph, we consider all finite sequences of rules applications ending with normal forms.

Termination We recall that a GRS is said to be (strongly) terminating whenever there is no infinite rewriting sequence. Actually, for non-size increasing GRS as presented above, we have immediately the decidability of non-uniform termination. That is, given some GRS and some graph, one may decide whether there is an infinite sequence. Indeed, once node labels and edge labels are fixed, there is a finite number of graphs of a given size. Then if we explore all rewriting paths starting from G, either they are all finite (and we can conclude for termination) or at some point, we obtain a graph that is equal to a graph already seen (with a cardinality argument) in the same path (and we can conclude for non-termination).

However, uniform termination —given a GRS, is it terminating?— of non-size increasing GRS remains an open problem. Uniform termination was proved undecidable when we drop the property of non-size increasingness (cf. Plump [13]). As a consequence, there is a need to define some termination method pertaining to non-size increasing GRS. Compared to standard work in termination [12, 8], there are two difficulties: first, our graphs may be cyclic, thus forbidding methods developed for DAGs such as term-graphs. Second, using term rewriting terminology, our method should operate for some non-simplifying GRS, that is GRS for which the output may be "bigger" than the input. Indeed, the NLP programmer sometimes wants to compute some *new* relations, so that the input graph is a strict sub-graph of the resulting graph.

In [3], we propose a method based on weight analysis. As is it possible to add edges, we consider negative weight for such edges. As a consequence the weight of a graph can be negative. But thanks to cardinality consideration again, we can give a lower bound for these negative weights and prove termination for weighted GRS. The major difficulty of the proof is linked to the `shift` commands which modified an unbounded number of edges in one step; some abstraction of the set of possible environments is needed to prove termination. Moreover, we show that weighted GRS terminates in quadratic time and that it is decidable to know if a given GRS is weighted or not. Systems like the one described above for implementation of long distance dependencies cannot be weighted. We propose lexicographic extension of the weight analysis method with which it is possible to prove termination of this system.

4 Conclusion

We have presented a Graph Rewriting formalism, dedicated to NLP. In our applications, we consider set of small GRS (called modules), each of these modules deals with a linguistic aspect of the global transformation. Most of the modules written in practice are confluent and so we can compute efficiently the unique normal form with these modules. We conjecture that this property is decidable, but we do not have nowadays automatic procedure to ensure the confluence of a module.

Up to now, finite sequences of modules are considered but we ran into cases where some couple of modules must be ordered in one way for one linguistic example and in the other way on another example. We can handle these examples with sequences where the same module is used several times. A better solution would be to add a more powerful language in which we can express stragies like the iteration of a sub-sequence of modules until normal forms are reached.

In recent years, most of the works in NLP focus on statistical and machine learning based approaches which are well-adapted to these applications because natural language is full of ambiguities and often external knowledge may be needed to resolve these ambiguities. This knowledge may be captured by learning methods using a large set of linguistic examples. At a first sight, our approach which is rule-based and does not use statistical information may not be able to deal with theses problems. Based on several applications to real size NLP problem we have conducted, we believe that our approach can be combined with machine learning based methods. Indeed, with non-confluent modules we can easily defined by rules a set of linguistically correct interpretations of a given input and use a statistical approach to rank these interpretations.

References

[1] B. Bohnet & L. Wanner (2001): *On using a parallel graph rewriting formalism in generation*. In: *EWNLG '01: Proceedings of the 8th European workshop on Natural Language Generation*, Association for Computational Linguistics, pp. 1–11, doi:10.3115/1117840.1117847.

[2] G. Bonfante, B. Guillaume, M. Morey & G. Perrier (2011): *Modular Graph Rewriting to Compute Semantics*. In: *IWCS 2011*, Oxford, UK, pp. 65–74.

[3] Guillaume Bonfante & Bruno Guillaume (2013): *Non-simplifying Graph Rewriting Termination*. In Rachid Echahed & Detlef Plump, editors: *TERMGRAPH*, 7th International Workshop on Computing with Terms and Graphs, Rome, Italy, pp. 4–16. Available at `https://hal.inria.fr/hal-00921053`.

[4] Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah & Éric Villemonte De La Clergerie (2014): *Deep Syntax Annotation of the Sequoia French Treebank*. In: *LREC*, Reykjavik, Iceland. Available at `https://hal.inria.fr/hal-00969191`.

[5] A. Copestake (2009): Invited Talk: *Slacker Semantics: Why Superficiality, Dependency and Avoidance of Commitment can be the Right Way to Go*. In: *Proceedings of EACL 2009*, Athens, Greece, pp. 1–9.

[6] D. Crouch (2005): *Packed Rewriting for Mapping Semantics to KR*. In: *Proceedings of IWCS*.

[7] R. Echahed (2008): *Inductively Sequential Term-Graph Rewrite Systems*. In: *Proceedings of the 4th international conference on Graph Transformations*, ICGT '08, Springer-Verlag, Berlin, Heidelberg, pp. 84–98.

[8] E. Godard, Y. Métivier, M. Mosbah & A. Sellami (2002): *Termination Detection of Distributed Algorithms by Graph Relabelling Systems*. In A. Corradini, H. Ehrig, H.-J. Kreowski & G. Rozenberg, editors: *ICGT*, *Lecture Notes in Computer Science* 2505, Springer, pp. 106–119, doi:10.1007/3-540-45832-8_10.

[9] Bruno Guillaume & Guy Perrier (2015): *Dependency Parsing with Graph Rewriting*. In: *IWPT 2015, 14th International Conference on Parsing Technologies*, Bilbao, Spain, pp. 30–39. Available at `https://hal.inria.fr/hal-01188694`.

[10] E. Hyvönen (1984): *Semantic Parsing as Graph Language Transformation - a Multidimensional Approach to Parsing Highly Inflectional Languages*. In: *COLING*, pp. 517–520, doi:10.3115/980491.980601.

[11] V. Jijkoun & M. de Rijke (2007): *Learning to Transform Linguistic Graphs*. In: *Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing, Rochester, NY, USA*.

[12] D. Plump (1995): *On Termination of Graph Rewriting*. In: *Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science*, WG '95, Springer-Verlag, London, UK, pp. 88–100, doi:10.1007/3-540-60618-1_68.

[13] D. Plump (1998): *Termination of Graph Rewriting is Undecidable*. *Fundamenta Informaticae* 33(2), pp. 201–209, doi:10.3233/FI-1998-33204.

[14] G. Rozenberg, editor (1997): *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific.

[15] L. Tesnière (1959): *Eléments de syntaxe structurale*. Librairie C. Klincksieck, Paris.