Tree-Rewriting Models of Multi-Word Expressions

William Schuler
Department of Linguistics
The Ohio State University
schuler@ling.osu.edu

Aravind K. Joshi
Dept. Computer and Information Science
University of Pennsylvania
joshi@linc.cis.upenn.edu

Abstract

Multi-word expressions (MWEs) account for a large portion of the language used in day-to-day interactions. A formal system that is flexible enough to model these large and often syntactically-rich non-compositional chunks as single units in naturally occurring text could considerably simplify large-scale semantic annotation projects, in which it would be undesirable to have to develop internal compositional analyses of common technical expressions that have specific idiosyncratic meanings. This paper will first define a notion of functor-argument decomposition on phrase structure trees analogous to graph coloring, in which the tree is cast as a graph, and the elementary structures of a grammar formalism are colors. The paper then presents a formal argument that tree-rewriting systems, a class of grammar formalism that includes Tree Adjoining Grammars, are able to produce a proper superset of the functor-argument decompositions that string-rewriting systems can produce.

1 Introduction

Multi-word expressions (MWEs), whose structure and meaning cannot be derived from their component words as they occur independently, account for a large portion of the language used in day-to-day interactions. Indeed, the relatively low frequency of comparable single-word paraphrases for elementary spatial relations like 'in front of' (compare to 'before') or 'next to' (compare to 'beside') suggest a fundamentality of expressions, as opposed to words, as a basic unit of meaning in language (Becker, 1975; Fillmore, 2003). Other examples of MWEs are idioms such as 'kick the bucket' or 'spill the beans', which have figurative meanings as expressions that sometimes even allow modification ('spill some of the beans') and variation in sentence forms ('which beans were spilled?'), but are not available when the component words of the MWE occur independently. A formal system that is flexible enough to model these large and often syntactically-rich non-compositional chunks as single units in naturally occurring text could considerably simplify large-scale semantic annotation projects, in which it would be undesirable to have to develop internal compositional analyses of common technical expressions that have specific idiosyncratic meanings.

Models have been proposed for MWEs based on string-rewriting systems such as HPSG (Sag et al., 2002), which model compositionality as string adjacency of a functor and an argument substring. This string-rewriting model of compositionality essentially treats each projection of a head word as a functor, each capable of combining with an argument to yield a higher-level projection or functor. The set of projections from a lexical head can therefore be thought of as a single elementary structure: an n-ary functor, subsuming the arguments of the individual functors at each projection. This kind of approach is intuitive for fully-compositional analyses (e.g. in which a transitive verb like 'hold' is a functor and a NP complement like 'the basket' is an argument), but is less natural when applied to substrings of MWEs (e.g. treating *pick* as a functor and *up* as an argument in the verb-particle MWE *pick ... up*), since some of these arguments do not have any semantic significance (in the *pick ... up* example , there is no coherent meaning for *Up* such that $[\![\text{pick } X \text{ up}]\!] = \mathit{Pick}([\![X]\!], \mathit{Up})$).

This paper will argue that tree-rewriting systems, a class of grammar formalisms that includes Tree Adjoining Grammars (Joshi, 1985; Joshi and Schabes, 1997), are a more natural candidate for modeling MWEs since they can model entire fragments of phrase structure trees as elementary (locally non-compositional) semantic building blocks, in addition to the set of head-projections used in string-rewriting

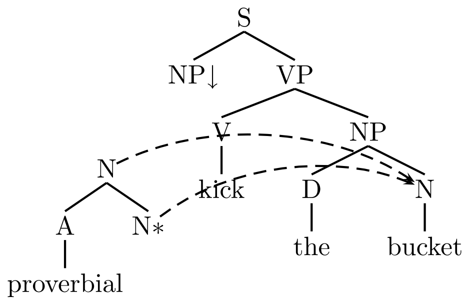

Figure 1: Composition of elementary trees for idiom MWE 'kick the bucket' and adjective 'proverbial,' with the same semantics as an adverb 'proverbially' adjoining at the VP.

systems. This allows more flexibility in defining the functor-argument decomposition of a given phrase structure tree.

This will be demonstrated by reducing the functor-argument decompositions (compositional accounts of semantics assigned to portions of phrase structure trees) of string-rewriting systems to a special case of functor-argument decompositions of tree-rewriting systems. Discussion in this paper will focus on string-rewriting systems augmented with unification (such as HPSG) because in this framework the issue of multi-word expressions has been discussed (Sag et al., 2002). The arguments in this paper also apply to other string rewriting systems such as categorial grammars (Ajdukiewicz, 1935; Bar-Hillel, 1953; Steedman, 2000), but in these formalisms the issues concerning MWEs have not been extensively developed. Essentially, this paper formalizes the intuition (Abeillé, 1993) that the extended domain of locality of tree-rewriting systems allows them to provide a compositional account of the semantics assigned to multi-word or idiomatic portions of phrase structure trees using elementary units that, after composition, may end up partially discontinuous in these trees. For example, a portion of a phrase structure tree for 'kick the bucket' with a single interpretation equivalent to 'die' can be modified through adjunction of the adjective 'proverbial' at the noun constituent 'bucket' without postulating separate semantics for 'kick' (see Figure 1).

2 Definitions

String rewriting systems are sets of rules for replacing symbols with other symbols in strings. A rewriting of some start symbol into a set of lexical symbols is called a derivation. Rewrite rules in a string rewriting system can be defined to have designated functor and argument symbols. Any derivation τ can therefore yield a functor-argument decomposition $\mathcal{D}(\tau)$, essentially defining a set of semantic functor-argument dependencies among structured elementary categories.

For simplicity, a functor-argument decomposition will be defined as a mapping from the constituent nodes in a phrase structure tree to the nodes in the elementary structures used to derive that tree. This can be thought of as a coloring of phrase structure nodes, in which colors correspond to elementary structures in the rewriting system. The elementary structures used in such a decomposition may then be considered n-ary functors, which may take several arguments, each of a different color.

In string-rewriting systems such as HPSG, these n-ary functors consist of a head word and its projections, and the arguments of the functor are the non-projecting child of each such projection. Figure 2 shows feature-based and categorial analyses for the MWE '*... to the ... power*' (as in '*raise Y to the X power*') which is taken here to have unambiguous meaning (in a technical context) as Y^X or $Pow(Y, X)$, and is analyzed here to wrap around an ordinal number argument X and then adjoin onto a verb phrase '*raise Y*' as a modifier.[1] Because their elementary structures are projected up from individual head words, these systems prohibit an analysis of this MWE as a single wrapping functor. Instead, MWEs like this must be decomposed into individual functor words (e.g. *power*) and argument words (e.g. *the*, and *to*).

Tree-rewriting systems, on the other hand, allow elementary structures to contain nodes which are neither projections nor argument sites. This permits an analysis of '*to the ... power*' as a single functor wrapped around its argument (see Figure 3), without having to specify functor-argument relations between *power*, *to*, and *the*.

More generally, string-rewriting systems use elementary structures (n-ary functors) that originate at the lexical item and exhibit a bottom-up branching structure, branching to an argument site and a higher level projection at each step. In contrast, tree-rewriting systems use elementary structures that originate at a phrasal or clausal node and exhibit

[1]We are using the MWE '...to the ...power' as a simple example with an unambiguous meaning in the domain of mathematics to illustrate our main points in the context of both adjunction and substitution operations. Alternative analyses are possible (e.g. with '*the*' or additional modifiers adjoining in, to allow variations like '*to every even power under six*'), but in any case the words '*to*' and '*power*' on either side of the X argument are taken to be idiosyncratic to this expression of Y^X. Since it is analyzed as a modifier, this example can be used to demonstrate coindexation of structure in a tree-rewriting system.

$$\begin{bmatrix} \text{label}:\text{power} \\ \text{left} : \begin{bmatrix}\text{label}:\text{ORD}\end{bmatrix} \\ \text{proj} : \begin{bmatrix} \text{label}:\text{N1} \\ \text{left}:\begin{bmatrix}\text{label}:\text{the}\end{bmatrix} \\ \text{proj}: \begin{bmatrix} \text{label}:\text{NP}\\ \text{left}:\begin{bmatrix}\text{label}:\text{to}\end{bmatrix} \\ \text{proj}: \begin{bmatrix}\text{label}:\text{PP}\\ \text{left}: \begin{bmatrix}\text{label}:\text{VP}\\ \boxed{1}\end{bmatrix} \\ \text{proj}:\begin{bmatrix}\text{label}:\text{VP}\\ \boxed{1}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix} =$$

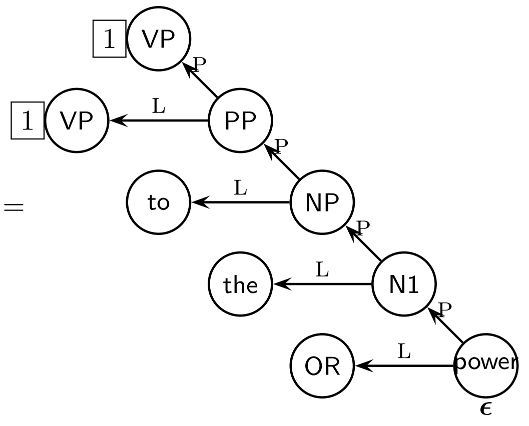

Figure 2: Elementary structures for a verb-phrase-modifying preposition in a functor-argument analysis derived from a feature structure grammar. Here, ϵ indicates the origin node and boxed numbers indicate coindexations.

a top-down branching structure that mirrors that of a phrase structure tree. As one might expect, there are tree-rewriting systems (namely those whose elementary structures contain multiple lexical items) that can produce functor-argument decompositions ('colorings') of a phrase structure tree which cannot be produced by a string-rewriting system. More surprisingly however, this paper will show that the converse is not true: in other words, for any string-rewriting system there always exists a tree-rewriting system that can produce the same functor-argument decomposition of a phrase structure tree. Thus, the set of functor-argument decompositions that can be produced by tree-rewriting systems is a proper superset of those that can be produced by string-rewriting systems.

This is surprising because, taken as a class, there is no inherent difference in recognition complexity between string-rewriting systems and tree-rewriting systems (as may be the case between specific members of these classes, say between CGs and TAGs), since both are worst-case exponential if unconstrained coindexation of structure is allowed (as in unification grammars). This is also surprising because, since they branch upward, the elementary structures of string-rewriting systems can specify complex functors as arguments, which the downward-branching elementary structures of tree-rewriting systems cannot. However, this paper will show that this ability to specify complex functors as arguments does not confer any additional flexibility in calculating functor-argument decompositions of phrase structure trees, and can be factored out with no loss in expressivity.

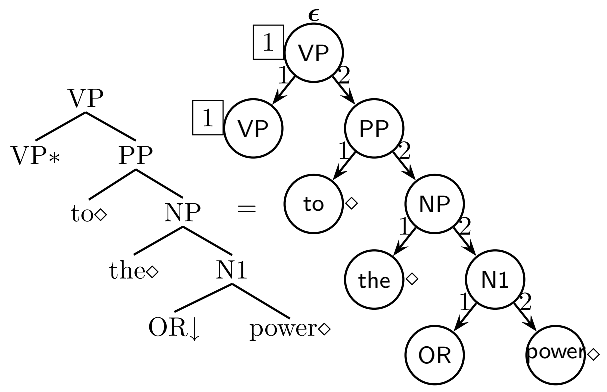

Figure 3: Elementary structure for a verb-phrase-modifying prepositional phrase '*to the ... power*' in a tree-rewriting system, derived from a tree-adjoining grammar. Here, ϵ indicates the origin node, $\diamond$ indicates a non-argument node (or lexical 'anchor'), and boxed numbers indicate coindexations.

3 Reduction of string-rewriting systems to tree-rewriting systems

The first step will be to define an n-ary functor in a string-rewriting system as a kind of elementary structure α (a tree in fact), whose nodes α_μ branch 'upward' into sub-structure nodes (connected by departing arcs labeled L, R, or P,) specifying a left or right argument category ($\alpha_{\mu\cdot\text{L}}$ or $\alpha_{\mu\cdot\text{R}}$) and a projected category ($\alpha_{\mu\cdot\text{P}}$), rather than branching 'downward' into left and right child constituents as in an ordinary phrase structure tree.[2] In order to extend this reduction to feature-based systems, these elementary structures will also be augmented with coindexation sets I of elementary structure nodes that must be identical (in terms of labels and departing arcs) in any functor-argument decomposition of a phrase structure tree.

[2]Here, a node α_μ is defined by the path of concatenated arcs μ that lead to it from the origin or root α_ϵ.

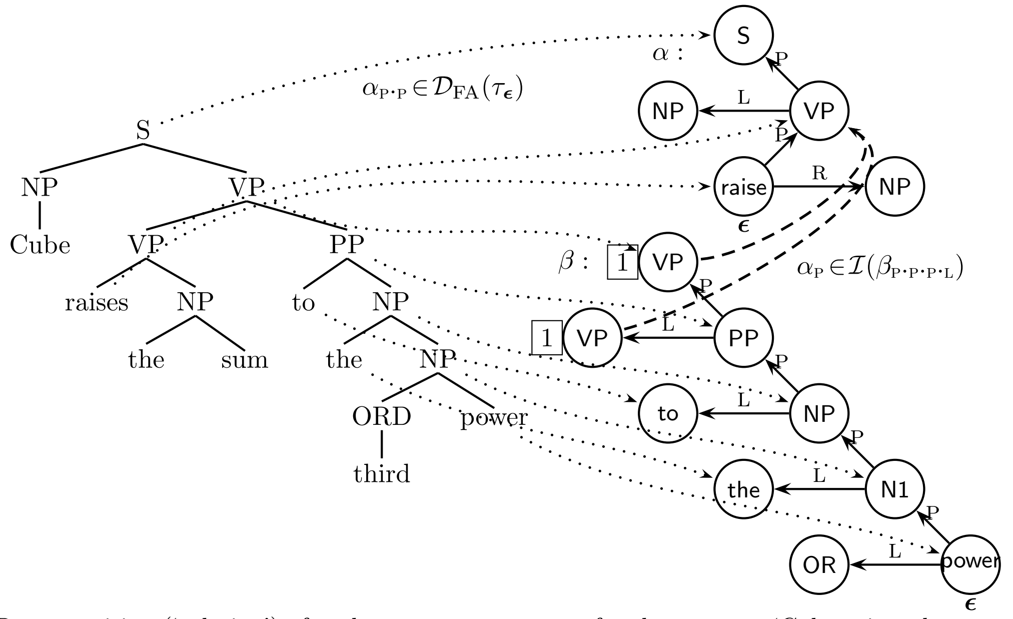
S
NP
VP
Cube
VP
PP
raises
NP
to
NP
the
sum
the
NP
ORD
power
third
$\alpha_{\text{P}\cdot\text{P}} \in \mathcal{D}_{\text{FA}}(\tau_{\epsilon})$
α :
S
P
NP
L
VP
P
raise
R
NP
ϵ
β :
1
VP
$\alpha_{\text{P}} \in \mathcal{I}(\beta_{\text{P}\cdot\text{P}\cdot\text{P}\cdot\text{L}})$
P
1
VP
L
PP
P
to
L
NP
P
the
L
N1
P
OR
L
power
ϵ

Figure 4: Decomposition ('coloring') of a phrase structure tree τ for the sentence 'Cube raises the sum to the third power', using elementary structures α and β shown at right. Dotted lines from phrase structure tree nodes τ_η to elementary structure nodes α_μ indicate that α_μ generates τ_η in the functor-argument decomposition: $\alpha_\mu \in \mathcal{D}_{\text{FA}}(\tau_\eta)$. Dashed lines from elementary structure nodes β_ν to other elementary structure nodes α_μ indicate that α_μ is among the nodes identified with β_ν as arguments of β in the decomposition. Boxed identifiers indicated coindices between nodes β_ν and $\beta_{\nu'}$ in β such that $\exists I \in \beta \ . \ \beta_\nu, \beta_{\nu'} \in I$.

Figure 4 shows a functor-argument decomposition (or 'coloring') of a phrase structure tree using these upward-branching elements.

The upward-branching elementary structures used in any such decomposition can then be converted into a normal form in which all argument nodes are *atomic* (have no departing arcs), using the following transformations of elementary structures to equivalent structures that fit together generate the same functor-argument decomposition. This is done by simultaneously excising 'matched' material from both the argument branch of an elementary structure and the top of the elementary structure that is its argument in the given decomposition.

The use of coindexation sets complicates this transformation somewhat. Initial configurations of coindexation sets in upward-branching elementary structures can be exhaustively partitioned into three classes, defined with respect to the 'trunk' of the elementary structure, which is the set of nodes connected to the origin by paths containing only P arcs. These classes are:

1. coindexations with more than one coindexed node on the trunk,
2. coindexations with fewer than one coindexed node on the trunk, and
3. coindexations with exactly one coindexed node on the trunk.

Elementary structures in the first class, with more than one coindexed node on the trunk, are equivalent to graphs with directed cycles, and are ordinarily excluded from feature-based analyses, so they will be ignored here.

Elementary structures in the second class, with fewer than one coindexed node on the trunk, can be converted to equivalent structures with no coindices (which trivially satisfies the above argument-atomicity requirement), using the simultaneous excision of 'matched' structure in functor and argument structures described above, by simply extending this to cover the portion of the argument elementary structure that extends all the way to the top of the trunk.

Elementary structures in the third class, with exactly one coindexed node on the trunk, can be converted to equivalent structures that satisfy argument-atomicity using a three-step process. First, the upward-branching sub-structures above these coindexed nodes (if any) are unified, so the arcs departing from each coindexed node will be recursively identical (this must be possible in any feature-based grammar, or the coindexation would be ill-formed, and should therefore be excluded). The coindexation is then recursively slid up along the P arc departing from each such node, until the coindexation set contains nothing but atomic categories (with no departing arcs). Finally, the argument nodes are made to be atomic using the simultaneous excision of 'matched' structure in functor and argument structures described above, leaving an (atomic) coindexation at each (atomic) argument position in each affected branch.

Elementary structures with multiple class 3 coindexation sets I and I' (which cannot be deleted as described above for class 2 sets) can be transformed into structures with a single coindexation set I by copying the portion of the trunk between the (unique) on-trunk members of each initial set I and I' onto every other node in the set I' that contains the lower trunk node (this copy should include the coindex belonging to I). The coindexation set I' containing the lower on-trunk node is then simply deleted.

The normal-form upward-branching structures resulting from this transformation can now be converted into downward-branching elementary trees in a tree-rewriting system (with coindexed nodes corresponding to 'root' and 'foot' nodes as defined for tree-adjoining grammars) by simply replacing each pair of argument and conclusion arcs with a pair of left-child and right-child arcs departing the conclusion node. Since the normal form for upward-branching elementary structures allows only atomic arguments, this re-drawing of arcs must result in well-formed downward-branching elementary trees in every case.[3]

In particular, this conversion results in a subset of tree-rewriting systems in which each (binary) branch of every elementary tree must have exactly one argument position and one non-argument position among its two children. This is a special case of a more general class of tree-rewriting systems, which may have two argument positions or no argument positions among the children at each binary branch. Such trees are not equivalent to trees with a single argument position per branch, because they will result in different functor-argument decompositions ('colorings') of a target phrase structure tree. Moreover, it is precisely these non-string-rewriting-equivalent elementary trees that are needed to model the local non-compositionality of larger multi-word expressions like 'threw X to the lions' (see Figure 5), because only downward branches with multiple non-

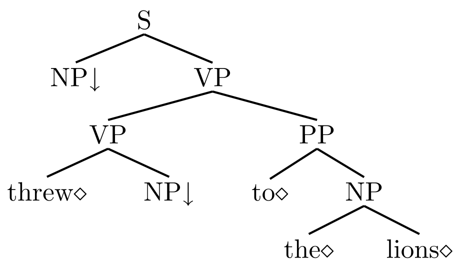

Figure 5: Elementary structure for MWE idiom '*threw ... to the lions*,' allowing modification to both VP, PP and NP sub-constituents (e.g. '*threw your friends today right to the proverbial lions*).

argument children can produce the multi-level subtrees containing the word 'threw' and the word 'lions' in the same elementary unit.

4 Conclusion

This paper has shown that tree-rewriting systems are able to produce a superset of the functor-argument decompositions that can be produced by string-rewriting systems such as categorial grammars and feature-structure grammars such as HPSG. This superset additionally allows elementary units to contain multiple (lexical) leaves, which a string-rewriting system cannot. This makes tree-rewriting systems ideally suited to the analysis of natural language texts that contain many multi-word expressions with idiosyncratic (non-compositional) meanings. Although neither the tree-rewriting nor the string-rewriting analyses defined above can be generated in guaranteed polynomial time (since they may require the construction of unbounded stacks of unrecognized structure during bottom-up recognition), they can both be made polynomial (indeed, cubic) by the introduction of 'regular form' constraints (Rogers, 1994), which limit this stack in the same way in both cases.

In contrast with representations like that of (Villavicencio et al., 2004), in which concepts are distributed over several lexical entries, a tree-rewriting representation such as the one described in this paper allows only a single lexical entry to be listed for each concept. For example:

... throw ... to the lions:
(s(np0!)(vp(v)(np1!)(pp(p)(np(d)(n)))))
... to the ... power:
(vp(vp0*)(pp(p)(np(d)(n(a1!)(n)))))

(using the notation '!' and '*' for substitution sites and foot nodes, respectively). It is anticipated that this will simplify the organization of lexical resources for multi-word expressions.

[3]Recognition and parsing of feature-based grammars, and of tree-rewriting systems whose elementary trees contain multiple foot nodes, are both exponential in the worst case. However, both types of grammars are amenable to regular-from restrictions which prohibit recursive adjunction at internal (non-root, non-foot) tree nodes, and thereby constrain recognition and parsing complexity to cubic time for most kinds of natural language grammars (Rogers, 1994).

References

Abeillé, Anne. 1993. The flexibility of french idioms: a representation with lexicalized tree adjoining grammar. In A. Schenk and E. van der Linden, editors, *Idioms*. Erlbaum.

Ajdukiewicz, Kazimierz. 1935. Die syntaktische konnexitat. In S. McCall, editor, *Polish Logic 1920-1939*. Oxford University Press, pages 207–231. Translated from Studia Philosophica 1: 1–27.

Bar-Hillel, Yehoshua. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Becker, Joseph D. 1975. The phrasal lexicon. In *Proceedings of the Workshop on Theoretical Issues in Natural Language Processing, Workshop in Computational Linguisitcs, Psychology, and AI*, Cambridge, MA.

Fillmore, Charles J. 2003. Multiword expressions, November. Invited talk at the Institute for Research in Cognitive Science (IRCS), University of Pennsylvania. http://www.cis.upenn.edu/∼ircs/colloq/2003/fall/fillmore.html.

Joshi, Aravind and Yves Schabes. 1997. Tree-adjoning grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*. Springer-Verlag, Berlin, pages 69–123.

Joshi, Aravind K. 1985. How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars. In L. Karttunen D. Dowty and A. Zwicky, editors, *Natural language parsing: Psychological, computational and theoretical perspectives*. Cambridge University Press, Cambridge, U.K., pages 206–250.

Rogers, James. 1994. Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94)*.

Sag, Ivan, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: a pain in the neck for nlp. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLING'02)*, pages 1–15, Mexico City, Mexico.

Steedman, Mark. 2000. *The syntactic process*. MIT Press/Bradford Books, Cambridge, MA.

Villavicencio, Aline, Ann Copestake, Benjamin Waldron, and Fabre Lambeau. 2004. Lexical encoding of mwes. In Takaaki Tanaka, Aline Villavicencio, Francis Bond, and Anna Korhonen, editors, *Second ACL Workshop on Multiword Expressions: Integrating Processing*, pages 80–87, Barcelona, Spain, July. Association for Computational Linguistics.